Classificazione e Predizione

Gianluca Amato

Corso di Laurea in Economia Informatica
Università “G. D'Annunzio” di Chieti-Pescara

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:
  - Machine learning
  - Statistica
  - Neurobiologia
- Con lo sviluppo del concetto di data mining si è posto particolare interesse al problema della scalabilità di questi algoritmi.

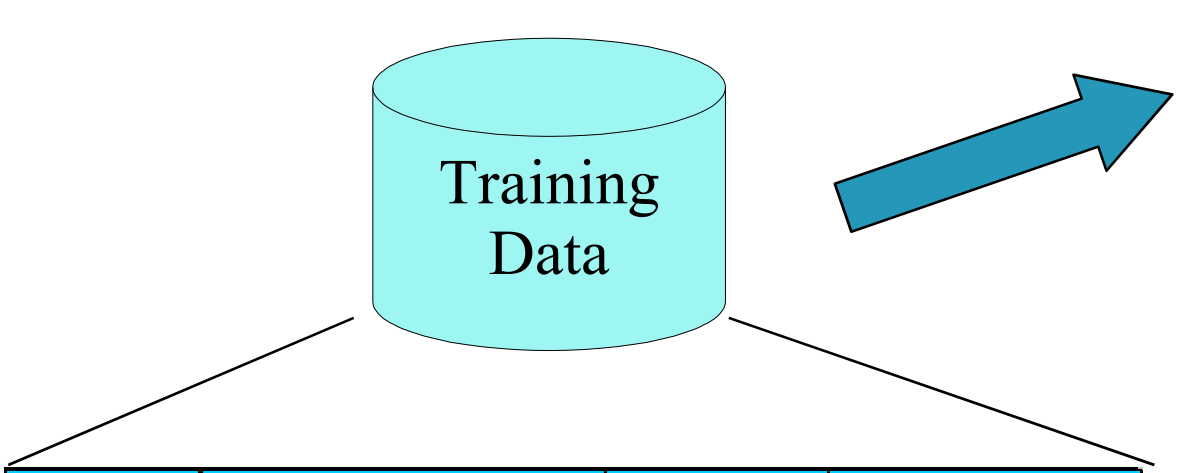
Classificazione (1)

- 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 consulto un specifico attributo, chiamato "class label attribute" (attributo classificatore).
  - Proprio perchè 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..

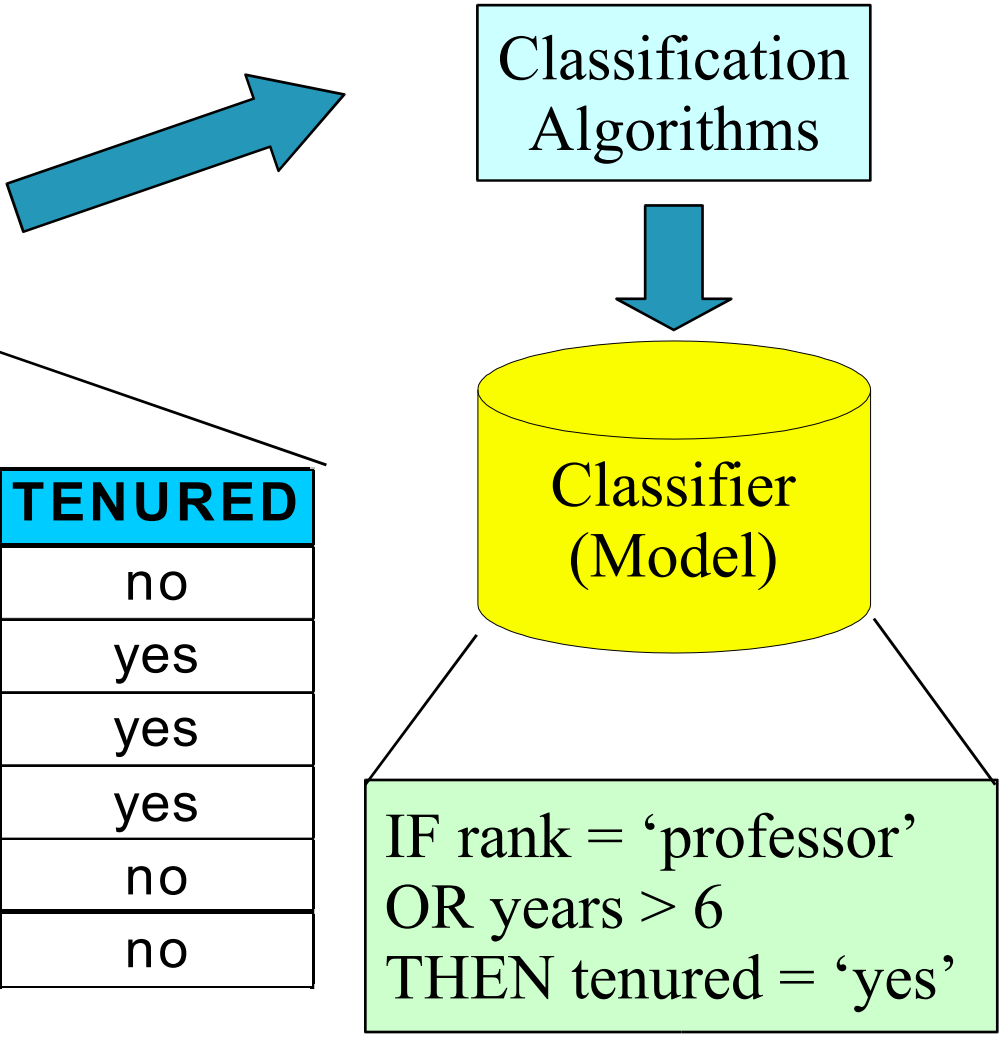
Classificazione (2)

- Fase 2 (stima dell'accuratezza): si stima l'accuratezza del modello usando un insieme di test.
  - Abbiamo già visto varianti di questo metodo come cross-validation e bootstrap.
- 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

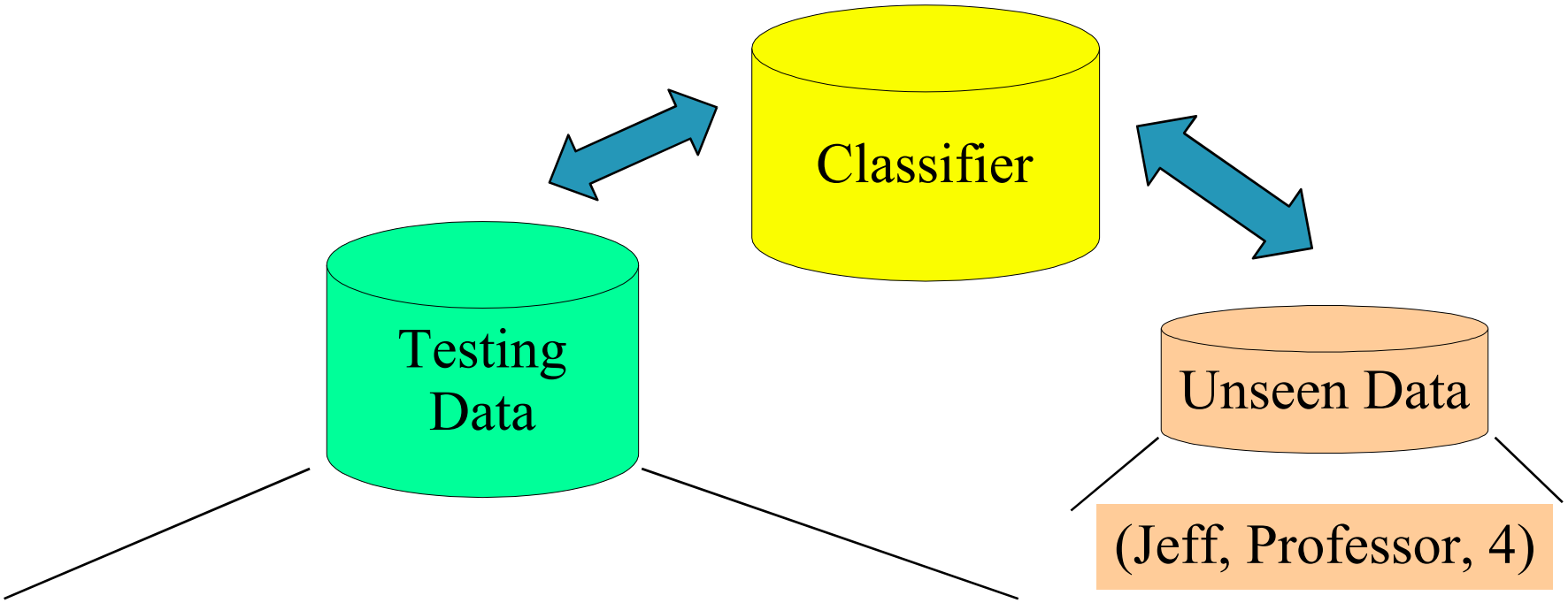
Costruzione del modello



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



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 polizia 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 effetturare 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

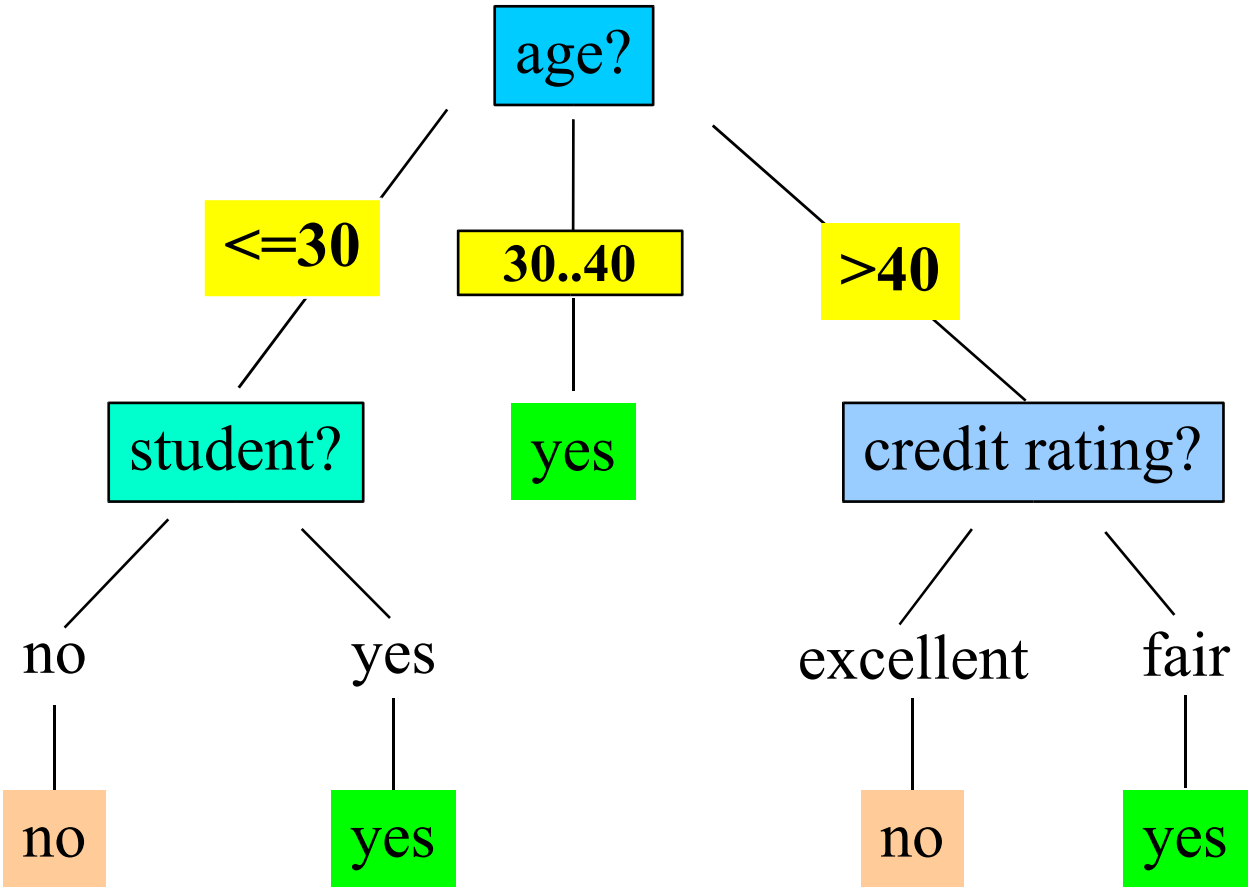
- Albero di decisione: un albero in cui
  - I nodi interni rappresentano un test su un attributo.
  - I rami rappresentano i risultati del test al nodo padre.
  - Le foglie rappresentano una classe o una distribuzione di probabilità per le classi.
- L'apprendimento è un processo a due fasi
  - Costruzione 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

Esempio introdotto da Quinlan per l'algoritmo ID3

age	income	student	credit_rating
<=30	high	no	fair
<=30	high	no	excellent
31…40	high	no	fair
>40	medium	no	fair
>40	low	yes	fair
>40	low	yes	excellent
31…40	low	yes	excellent
<=30	medium	no	fair
<=30	low	yes	fair
>40	medium	yes	fair
<=30	medium	yes	excellent
31…40	medium	no	excellent
31…40	high	yes	fair
>40	medium	no	excellent

Albero di decisione



L'algoritmo ID3

- L'abbiamo già visto in una lezione precedente....
- Qui ricordiamo solo che l'algoritmo ID3 costruisce un albero dividendo ricorsivamente l'insieme di tutte le istanze in base ai risultati dei vari test
  - scegliendo ogni volta l'attributo migliore sulla base del quale dividere (quello che ha il maggiore guadagno di informazione)
- La versione vista a lezione si ferma solo quando tutte le partizioni hanno la stessa classe o non esistono altri attributi da utilizzare.

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 determinto 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

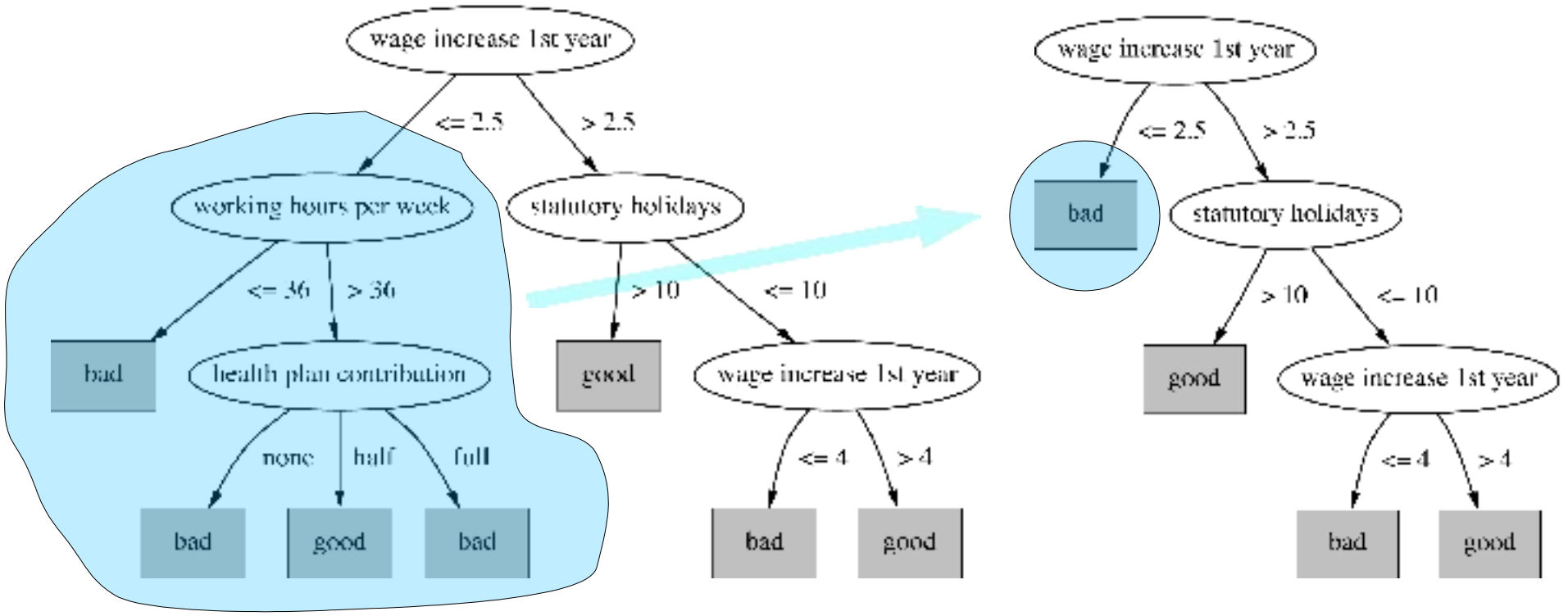
- Una interruzione prematura della costruzione dell'albero.
- Esempio classico, funzione XOR
  - Nessun attributo da solo è correlato in maniera significativa con il risultato dello XOR
  - Ma la coppia di input della funzione ha una correlazione fortissima.
  - Il prepruning rischierebbe di non accorgersi 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

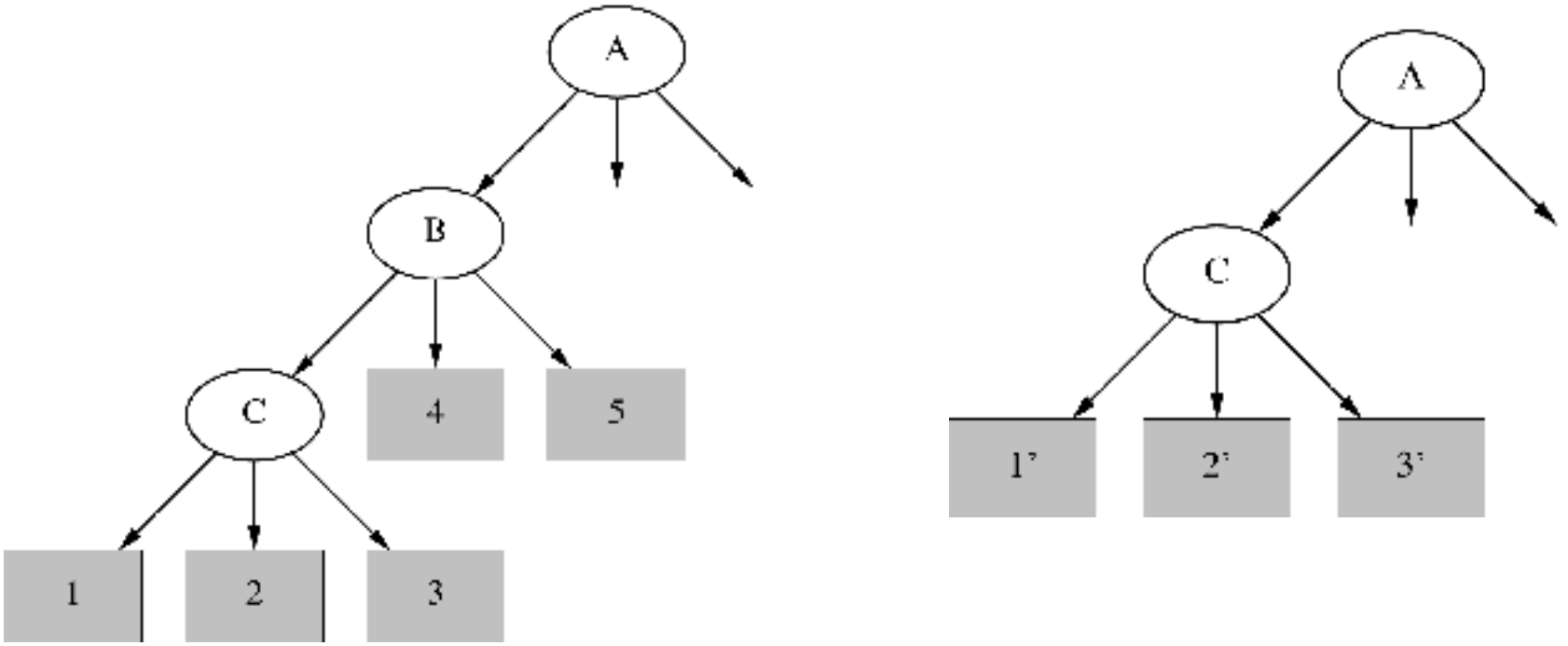
Subtree-replacement

- 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. Fa ovviamente delle stime pessimistiche 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 già 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 perchè 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.

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 classficazione.
  - ... 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 *age* = "<=30" AND *student* = "*no*" THEN *buys_computer* = "*no*"
IF *age* = "<=30" AND *student* = "*yes*" THEN *buys_computer* = "*yes*"
IF *age* = "31…40" THEN *buys_computer* = "*yes*"
IF *age* = ">40" AND *credit_rating* = "*excellent*" THEN *buys_computer* = "*yes*"
IF *age* = ">40" AND *credit_rating* = "*fair*" THEN *buys_computer* = "*no*"

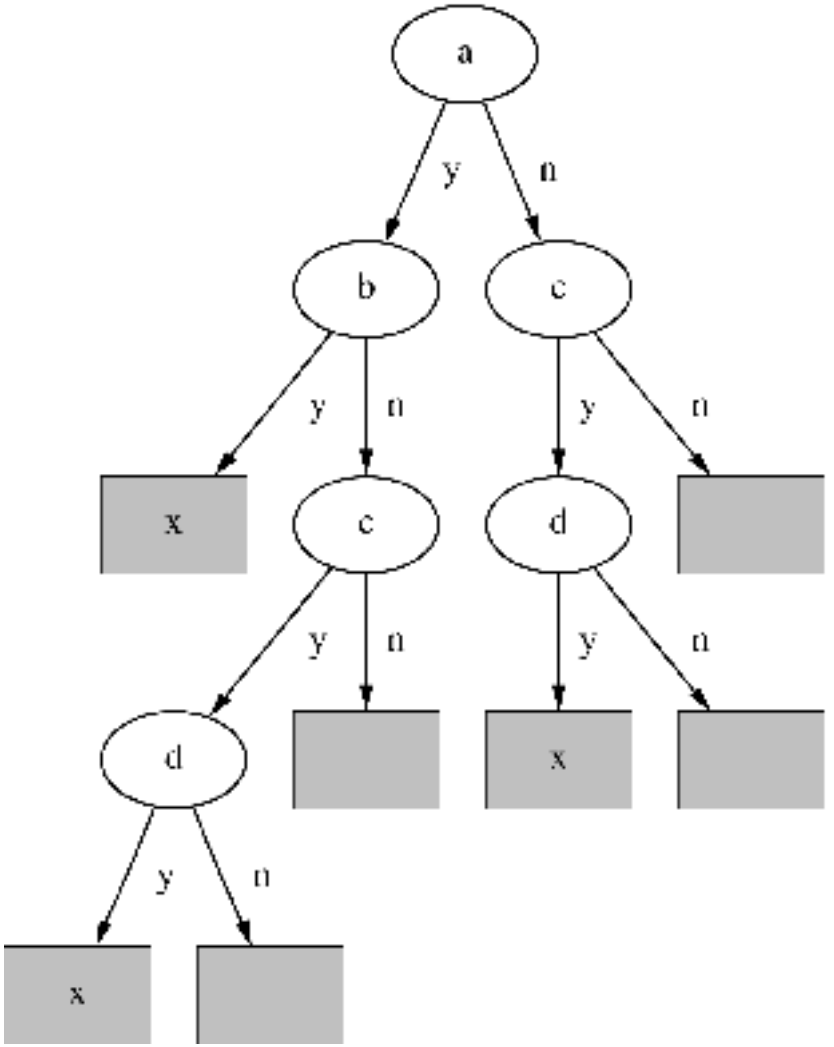
Alberi e regole di classificazione (2)

- 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

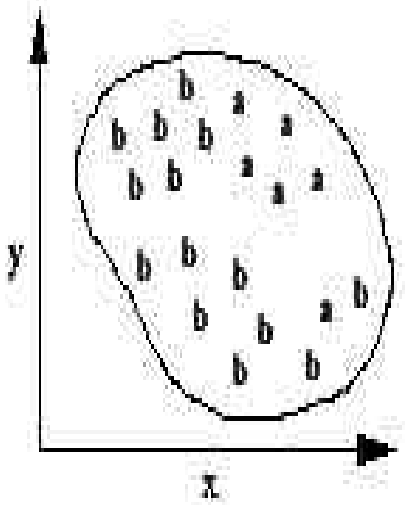
  si ottiene l'albero e destra.

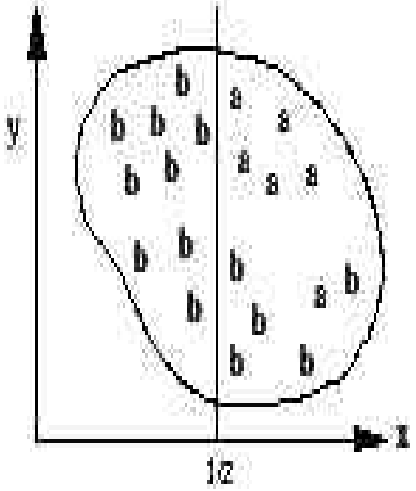


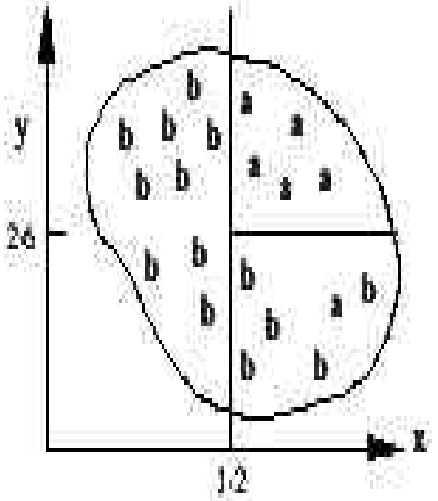
Algoritmi di copertura

- 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 perchè 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 definscono allora i concetti di
  - Copertura: corrisponde al valore t
  - Accurarezza: 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 attribute 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)

- 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 (2)

- 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 (3)

- 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 (4)

- Regola migliore aggiunta
  - If astigmatism=yes and tear prod. rate=normal then reccomendation=hard
- Istance 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 (5)

- 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 perchè ha copertura maggiore.

Esempio: lenti a contatto (6)

- 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'algortimo PRISM

- Quello che abbiamo descritto è l'algoritmo PRISM.
  - 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
```

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 per riconoscere lo spam dalle mail "buone".
- Abbiamo già analizzato il classificatore bayesiano naive.

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
  - $P(X \mid C_i)$ può essere difficile da calcolare.

Classificatori naive

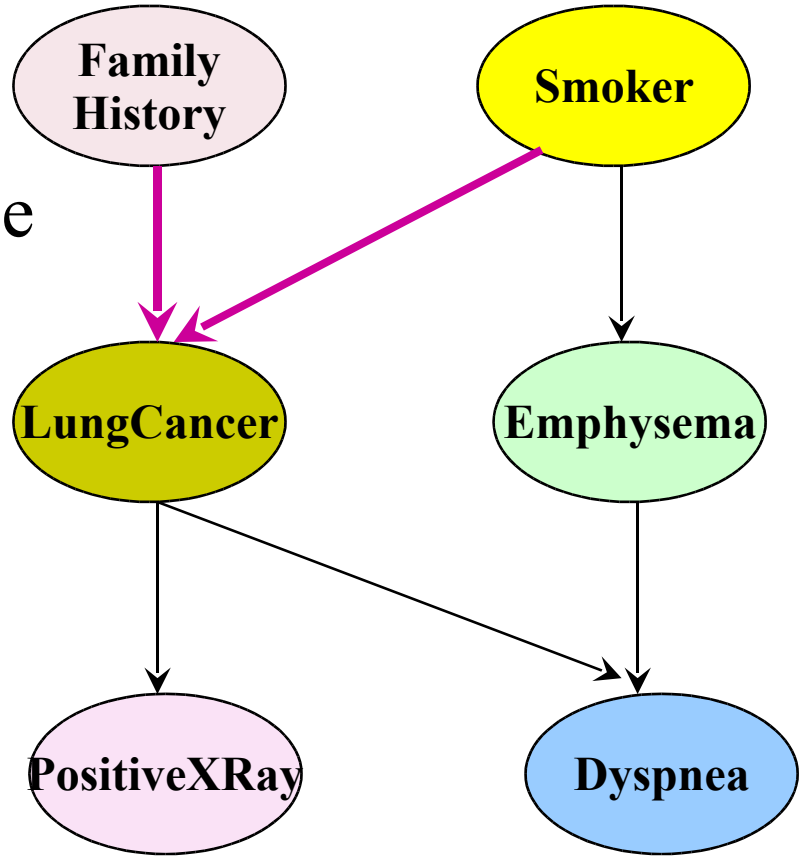
- Assunzione dei classificatori naive: indipendenza degli attributi.
- Se X è composta dagli attributi $A_1 ... A_m$, otteniamo

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

- 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.

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 aleatori 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

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

Reti Bayesiane (3)

- Alcune variabili casuali corrispondono ad attributi del nostro insieme dei dati.
  - Una di queste corrisponde all'attributo classe.
- Altre variabili casuali non corrispondono a nessun attributo. Si parla di "variabili nascoste".
- Varie possibilità per l'addestramento di reti bayesiane:
  - La struttura della rete è nota e non ci sono variabili nascoste;
    - Addestramento simile a quello dei classificatori naive
  - La struttura della rete è nota e ci sono variabili nascoste;
    - Si usano metodi di "discesa del gradiente", in maniera simile a quanto si fa per le reti neurali.
  - La struttura della rete non è nota.

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!
- Associative classification (Liu et al. 98)
  - 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

- Un pattern emergente è un itemset il cui supporto cresce di molto quando ci si sposta 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.
- Il metodo CAEP (classif. by aggregating emerging patterns):
  - determina tutti i pattern emergenti nell'insieme di addestramento
  - 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.

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 dà luogo a metodi incrementali
  - Hanno accuratezza maggiore perchè 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:

$$a_i=\frac{v_i-min\,v_i}{max\,v_i-min\,v_i}$$

  - 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 computazione 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 cusare instabilità.
    - Necessità di una fase inziale che elimini quelli irrilevanti

k-nearest neighbor e predizione

- Il metodo *k*-nearest neighbor può essere utilizzato anche per problemi di predizione.
- Si ottiene come risultato dell'applicazione la media delle classi delle *k* istanze più vicine, invece della moda.

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).

Predizione

Predizione e regressione

- Il metodo più comune per affrontare problemi di predizione è il metodo della regressione, noto dalla statistica.
  - non è l'unico, ad esempio abbiamo visto il k-nearest neighbor.
- Molti problemi possono essere risolti col metodo della regressione lineare.
  - Oppure, con dei cambiamenti di variabile, si può trasformare un problema non lineare in uno lineare.
- Esistono molti software statistici dedicati alla risoluzione di problemi di regressione
  - SAS (http://www.sas.com/)
  - SPSS (http://www.spss.com/)
  - S-Plus (http://www.mathsoft.com/)

Regressione lineare bivariata

- È il più semplice metodo di regressione.
- 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 risolti 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.

Altri tipi di regressione

- Regressione lineare multipla
  - $Y=a + b_1X_1 + b_2X_2$
- 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 attributi nominali
  - addestramento: si calcola una curva di regressione per ogni valore della classe, settando la var. dipendente a 1 per le istanze che appartengono alla classe, 0 altrimenti.
    - si chiama variabile caratteristica
  - utilizzo: quando abbiamo una nuova istanza, si calcolano le variabili caratteristiche dei valori delle varie classi, e si sceglie quella più alta.

Combinazione di classificatori

Boosting e Bagging

- Sono due metodi standard per combinare dei classificatori C_1, ... C_T per produrre un classificatore C* più accurato.
- Analogia con i medici
  - Supponiamo di voler diagnosticare una malattia. Possiamo rivolgerci a vari medici invece che ad uno solo.
  - Bagging: prendo le risposte di tutti i medici e considero come diagnosi valida quella prodotta in maggioranza.
  - Boosting: peso la diagnosi di ogni medico in base agli errori fatti in precedenza.

Bagging

- Sia S un insieme di *s* istanze.
- Per ogni t in {1, ... , T} si ottiene l'insieme S_t ottenuto campionando *s* valori da S con rimpiazzo
- Ottengo un classificatore C_t
- Per una nuova istanza, provo tutti i classificatori e scelgo la risposta che occorre con più frequenza.

Boosting

- Sia S un insieme di *s* istanze.
- Ogni istanza ha un peso (inizialmente uguale per tutte)
- Dopo aver addestrato il classificatore C_t, aggiorno i pesi in modo da far contare di più le istanze classificate incorrettamente da C_t.
- Per una nuova istanza, uso tutti i classificatori e peso i risultati in base alla loro accuratezza.

Conclusioni

Conclusioni

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

Riferimenti (1)

- C. Apte and S. Weiss. Data mining with decision trees and decision rules. Future Generation Computer Systems, 13, 1997.
- L. Breiman, J. Friedman, R. Olshen, and C. Stone. Classification and Regression Trees. Wadsworth International Group, 1984.
- P. K. Chan and S. J. Stolfo. Learning arbiter and combiner trees from partitioned data for scaling machine learning. In Proc. 1st Int. Conf. Knowledge Discovery and Data Mining (KDD'95), pages 39-44, Montreal, Canada, August 1995.
- U. M. Fayyad. Branching on attribute values in decision tree generation. In Proc. 1994 AAAI Conf., pages 601-606, AAAI Press, 1994.
- J. Gehrke, R. Ramakrishnan, and V. Ganti. Rainforest: A framework for fast decision tree construction of large datasets. In Proc. 1998 Int. Conf. Very Large Data Bases, pages 416-427, New York, NY, August 1998.
- M. Kamber, L. Winstone, W. Gong, S. Cheng, and J. Han. Generalization and decision tree induction: Efficient classification in data mining. In Proc. 1997 Int. Workshop Research Issues on Data Engineering (RIDE'97), pages 111-120, Birmingham, England, April 1997.

Riferimenti (2)

- J. Magidson. The Chaid approach to segmentation modeling: Chi-squared automatic interaction detection. In R. P. Bagozzi, editor, Advanced Methods of Marketing Research, pages 118-159. Blackwell Business, Cambridge Massechusetts, 1994.
- M. Mehta, R. Agrawal, and J. Rissanen. SLIQ : A fast scalable classifier for data mining. In Proc. 1996 Int. Conf. Extending Database Technology (EDBT'96), Avignon, France, March 1996.
- S. K. Murthy, Automatic Construction of Decision Trees from Data: A Multi-Diciplinary Survey, Data Mining and Knowledge Discovery 2(4): 345-389, 1998
- J. R. Quinlan. Bagging, boosting, and c4.5. In Proc. 13th Natl. Conf. on Artificial Intelligence (AAAI'96), 725-730, Portland, OR, Aug. 1996.
- R. Rastogi and K. Shim. Public: A decision tree classifer that integrates building and pruning. In Proc. 1998 Int. Conf. Very Large Data Bases, 404-415, New York, NY, August 1998.
- J. Shafer, R. Agrawal, and M. Mehta. SPRINT : A scalable parallel classifier for data mining. In Proc. 1996 Int. Conf. Very Large Data Bases, 544-555, Bombay, India, Sept. 1996.
- S. M. Weiss and C. A. Kulikowski. Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems. Morgan Kaufman, 1991.