Metodi di Classificazione

Regole di Decisione

Alberi di Decisione

Esempio più Vicino

Macchine a Supporto Vettoriale

Classificazione Bayesiana, Regressioni, Reti Neuronali

Classificazione

Mettere in ordine / distribuire un insieme d´oggetti descritti da dati altodimensionali in classi o sottostrutture comprensibili ed assegnare nuovi oggetti ad una di queste classi

Significativi:

sistema di classificazione

apprendimento da esempi

clustering

Classificazione come Sistema di Classificazione

Un sistema per ordinare oggetti

libri in una biblioteca
brevetti in ufficio (Europea) brevetti
elementi chimici (sistema periodico)
prodotti, malattie, ... (standardizzazione internazionale)

per facilitare trattamento, ricupero, communicazione, ... d´oggetti assistito da strumenti di teoria di concetti, linguistica, ...

Classificazione come Apprendimento da Esempi

Discriminazione
Assegnazione di classi
Apprendimento supervisionato
Pattern recognition

riconoscere/distinguere/identificare
- **malattie dai sintomi di un paziente**
- **cellule di sangue normali e anormali**
- **clienti leali e rischiosi**
- **banconote vere e false**
- **lettere e numeri da indirizzi scritti a mano**
- **parlatori, immagini**

Apprendimento da Esempi

Ogni oggetto proviene da una di *m* classi $C_1,...,C_m$
è descritto da un vettore di *k* attributi $x=(x_1,...,x_k)$
la classe sconosciuta deve essere identificata da x

possibile se:
le classi sono distintamente caratterizzate da distribuzioni differenti delle variabili

ma in pratica:
- **intersezioni fra classi**
- **variazioni casuali in classi**

una separazione perfetta non può essere ottenuta e misclassificazioni accadono

Classificatore

Un classificatore è una funzione f specificando per ogni oggetto x:
- **l´ indice della classe a cui appartiene x: $f(x)\in\{1,...,m\}$**
- **oppure *m* probabilità $f_1(x),...,f_m(x)$**

Per generare il classificatore, le proprietà delle classi sono descritte da

- ***m* distribuzioni probabilistiche o**
- **un insieme d´addestramento A**
  - **per ogni oggetto di A sono conosciuti**
    - **l´ indice della classe**
    - **e il vettore x di features**

Passi di Classificazione

1 **Selezionare le variabili che possono distinguere le classi**
Qualche metodo è intollerante agli attributi irrilevanti o non può trattare moltissimi variabili

2 **Acquirare informazioni su charatteristiche statistiche**
qui: insieme d´addestramento

3 **Costruire il classificatore**

4 **Valutare la bontà del classificatore (errore)**

5 **Applicare il classificatore ad oggetti nuovi**

Classificatione come Clustering

Costruzione di classi
Agglomerazione
Tassonomia
Apprendimento non supervisionato

Un insieme di *N* oggetti è dato
Ogni oggetto descritto da un vettore $x=(x_1,...,x_k)$
Problema:
Trova sottoinsiemi $C_1,...,C_m$ di oggetti
che sono omogenei

- **oggetti della stessa classe sono più simili**
- **classi sono separate**

Le classi sono sconosciute

Esempi di Clustering

Agglomerare consumatori a tipi di stile di vita

Costruire segmenti di mercati

Dividere clienti in gruppi rischiosi

Trovare tipi di malattie

Ragruppare dipartimenti secondo strutture organizzative

Regola

Se S allora T (oppure $S \rightarrow T$) S, T sottogruppi

Criteri elementari di valutazione:

Precisione (confidenza): $|S \cap T| / |S|$
= probabilità condizionale $P(T|S)$

Supporto: $|S \cap T| / N$ (oppure $|S| / N$)

N

$P_0 = |T| / N$

	T	Non T
S	VP Veri positivi	FP Falsi positivi
Non S	FN Falsi negativi	VN Veri negativi

Regole di Classificazione

Regole di classificazione: if S_i then T

T è fisso, scoperta di un sistema di regole $\{S_i \rightarrow T\}$ per predire T

spesso è dato una variabile target (t), T : $t=t_0$, T_i: $t=t_i$ i=1,…,m

p.e. un insieme di regole che predicono se un cliente è fraudolente
ogni regola descrive una condizione che implica
(con alta probabilità) che un cliente sia fraudolente

Una regola include la probabilità per la classe
le probabilità per le *m* classi

Rappresentazione espressiva e facile da comprendere

Predizione è trasparente

Anche applicabile per esplorare / conoscere un dominio

Tipi d´Insiemi di Regole

Tre forme di insiemi di regole:

un solo insieme associato ad una classe distinta
p.e. clienti rischiosi
se un cliente corrisponde ad una delle regole, viene classificato rischioso

m insiemi
ogni set associato ad una delle *m* classi e costruito indipendentemente

un singolo insieme collettivamente trattando le *m* classi
le regole del set sono ordinate, la prima regola verificata viene applicata

Risolvere Ambiguità

Dato *m* insiemi di regole (per le *m* classi)
Un oggetto può osservare diverse regole,
possibilmente da diversi insiemi

p.e. risolvere ambiguità,
se ogni regola include un vettore di probabilità
$\mathbf{p_1,...,p_m}$ $\mathbf{p_i}$ = probabilità per la classe i dato la condizione della regola

1 Sommare i vettori di probabilità di queste regole
2 Scelgiere la classe con la somma più alta

Componenti

Linguaggio per sottogruppi (termine sinistra della regola, condizione)

Strategia di ricerca (trovare regole buone)

Misura di qualità (per ordinare le regole)

Raffinamento (postprocessing dei risultati, dopo la ricerca)
Restrizioni (*constraints* per regole da trovare)
Test statistica (per filtrare regole significativi)

Linguaggi per Sottogruppi

Linguaggio per termine sinistra della regola, parte condizionale

Monorelazionale:

una singola tabella con attributi $A_1,...,A_n,A_{n+1}$ D_i dominio di A_i
A_{n+1} attributo di classe (categorico)

if $A_1 \in V_1 \wedge A_2 \in V_2 \wedge ... \wedge A_n \in V_n$ then $A_{n+1} = a$ $V_i \subseteq D_i$
omettere $A_i \in V_i$, nel caso che $V_i = D_i$

if sesso=maschio $\wedge$ età=18-21 then rischioso

A_i categorico:

selettori semplici:	$A_i=a$	sesso = maschio
selettori negativi:	$A_i \neq a$	stato $\neq$ sposato
disgiunzioni interni:	$A_i=a \vee b \vee ... \vee g$	stato = divorziato $\vee$ vedovo

A_i continuo, ordinato: V_i = intervallo

Strategie di Ricerca

Approccio *sequential covering (consecutiva coperta)*

iterare:
1 ricercare la regola migliore
2 eliminare tutti gli oggetti coperti dalla regola (o parte sinistra)

modi:
la migliora regola per una classe fissa (->qualità)
la migliora regola per tutte le classe (->qualità)

ricercare la regola migliora:
strategia euristica, da generale a specifico, di generare e validare
(general to specific, generate and test heuristic)

Euristico, generale allo specifico, genera ed esamina strategia

Heuristic: non tutte le regole, p.e. *beam search* ricerca di raggio

General to specific: operatore di espansione aggiunge una congiunzione

livello 0 regola vuota

livello 1 1 selettore sesso=maschio,sesso=femina,età=giovane,età=anziano,...

livello 2 2 selettori sesso=maschio∧età=giovane,sesso=maschio∧ età=anziano,...

...

Beam search, larghezza w: le w migliori regole d´un livello vengono espanse

disgiunzioni interni: operatore toglie un disgiunzione

variabili ordinati v: selettori v<a, v≥a per ogni a∈Dominio(v)

Generate and test:

tutti i successori vengono generati dai operatori di espansione

i successori generati vengono esaminati e alcuni scelti da espandere

Funzioni di Qualità

Entropia: $-\sum p_i \log(p_i)$ $p_i = |S \cap T_i| / |S|$ i= 1,...,m (m valori del target)
CN2, ricercando regole per tutte le classi

Precisione: $p_i = |S \cap T_i| / |S|$ ricercando regole per la classe i

molte altre, p.e. (3) $\sqrt{(|S|/(N-|S|))} \quad (|S \cap T_i| / |S| - |T_i| / N)|$

Supplementi

Test per la significatività: solo regole con significative probabilità
CN2: likelihood ratio statistic, cioè significativa differenza?

$(|S \cap T_1| / |S| \ ,..., \ |S \cap T_m| / |S| \ ) \ - \ (|T_1| / N| \ ,..., \ |T_m| / N|)$

qualità (3) include test di significatività

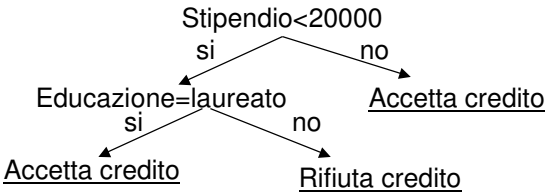
Restrizioni: minimo supporto, precisione, positivi esempi, ...

Potare regole: regola più semplice si adatta meglio su test dati

Problema: regole per classi con pochissimi esempi tipicamento non vengono trovate (o applicate risolvendo ambiguità)

Regole contra Alberi

Regole d´un set (per una classe *i*) o
regole di diverse set (per due classi *i,j*) possono sovrapporsi
R1: if Maschio then rischioso
R2: if Cittadino then rischioso
R3: if Anziano then non-rischioso
p.e. Maschio $\wedge$ Anziano $\neq \varnothing$



Foglie d´un albero sono disgiunte
Stipendio<20000 $\wedge$ Educazione=laureato
Stipendio<20000 $\wedge$ Educazione $\neq$ laureato, Stipendio$\geq$20000

Perché Alberi?

Sono comprensibili (ma non sempre)

Classificazione trasparente

Precisione non molto peggiore di tecniche più sofistiche

Facile da trasformarli in interrogazioni di base di dati

Alberi di Classificazione: Componenti

Linguaggio per sottogruppi (nodi) $\rightarrow$ regole
CART: $A_i \in V_i$ C4.5: $A_i=a$

Strategia di ricerca (trovare una buona spaccatura / split d´un nodo)

Misura di qualità (per ordinare gli split)

Raffinamento (postprocessing: potare un albero)
Restrizioni (della complessità d´un albero)
Valori mancanti

Costruire un Albero

Algoritmo ricorsivo: split d´un nodo (fino ai nodi *puri*)

Ogni nodo interno ha un test con un solo attributo

Successori di un nodo corrispondono ai outcome del test
split binario (CART) o più di due successori (ID3, C4.5)

Ogni foglia ha una classe associata

Algoritmo greedy: non tutti gli alberi possibili vengono costruiti (esponenziale col numero degli attributi)

Per classificare un oggetto, un sentiero viene proseguito secondo i valori dell´oggetto, dalla radice ad una foglia, applicando i test

Split d´un Nodo

Che attributo ha localmente il miglioramento ottimo della precisione per la classificazione?
Altri valutazioni d´un split sono migliori

Set di test: insieme di test che possono essere costruiti dagli attributi e loro valori (per un nodo col set E d´oggetti)
p.e. test binari: $A_i \in V_i$, $A_i < v$

Sia T un test di questo set di test, T fa una partizione d´un insieme d´oggetti E in sottoinsiemi $E_1,...,E_k$ (*k* outcome di T)

Sia I una funzione d´impurità: $I(E)$, $I(E_1)$, ..., $I(E_k)$

Qualità(E,T)=$I(E) - \sum (|E_i|/|E| \; I(E_i))$ qualità del split/test
Selezzionare l´ottimo split

Funzioni di Qualità

Entropia: $I(E) = -\sum p_i \log(p_i)$ $p_i = |E \cap C_i| / |E|$ i=1,...,m; m classi

Information gain: $G := I(E) - \sum (|E_i|/|E| \; I(E_i))$
i=1,...,k; k outcome del test (C4.5, CART)

Gini index: $1 - \sum p_i^2$ (CART)

Information gain favoreggia test con molti outcome (k), poi:

$P := -\sum |E_i|/|E| \log(|E_i|/|E|)$ i=1,...,k
Gain ratio: G/P

Potare un Albero

Fase del costruire: ricorsivamente partizionare gli esempi

Fase del potare:
gli alberi perfetti (con foglie pure) si adattono agli irregolarità del insieme d´addestramento (overfit)

Le foglie vengono iterativamente potate nodi interni diventano foglie
Un albero potato contiene foglie impure

Potare con un separato insieme di validazione -> Breiman, CART
Potare col criterio di MDL (minimal description length) -> Fayad, GID3
Potare coll´estimo dell´errore vero di classificazione-> Quinlan, ID3, C4.5
´Potare´ durante la costruzione quando il supporto d´un nodo diventa basso
o il split migliore non sia più significativo (criterio di stopping)

Supplementi

Valori mancanti metodi per costruire ed applicare l´albero

Alberi di regressione variabile continuo di target: media in foglia

Alberi di modello variabile continuo di target: funziona in foglia

Estrarre regole da albero folgie sono regole, potare queste regole

Scalibilità parallelità, statistiche sufficienti

bagging, boosting multipli alberi aumentano la precisione

costi (non solo 0-1)

Che fare se gli alberi sono troppo complessi?

Alberi sono troppo grandi (anche dopo potare): p.e migliai di foglie
data mining: moltissimi oggetti e variabili

Se alberi comprensibili sono più importante di alberi precisi:

osservazione: solo piccola riduzione di precisione con un albero molto meno complesso

restrizioni su numero di foglie o precisione
trovare l´albero ottimo osservando questi restrizioni
approcci di dynamic programming durante la fase della costruzione

Esempio più Vicino

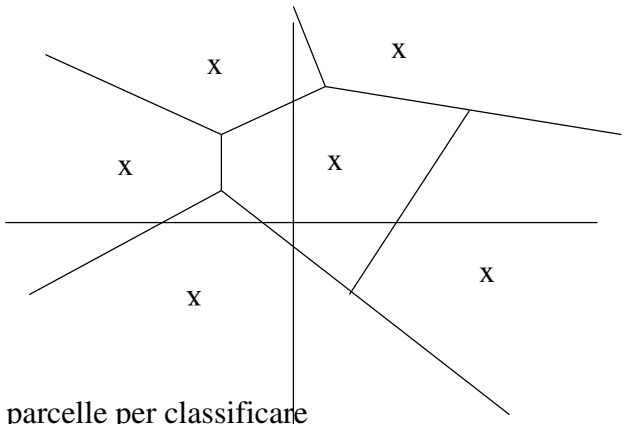
Nearest neighbor (NN)
Memory based reasoning
Apprendimento basato su istanze
Apprendimento pigro

Idea molto semplice:
Predire il valore o la classe y_{nuovo} d´un oggetto nuovo x_{nuovo} trovando l´oggetto più simile d´un insieme E d´esempi e prendendone il valore/la classe: $y_{nuovo}=\text{argmax}(y)_{(x,y)\in E}\text{sim}(x,x_{nuovo})$

Apprendimento: memorizzare gli esempi (pigro)
Classificare: trovare l´esempio più simile (costa tempo)

Classificazione trasparente

Voronoi Diagramma



Molti piccole parcelle per classificare
Molti ipotesi locali
Precisione alta, ma rischio di overfitting

k-esempi più vicini (k-NN)

Mitigare l´effetto d´un esempio atipico/outlier:
prendere *k* prossimi esempi, questi hanno diversi target valori
Pesare i vicini con la similarità all´oggetto da classificare

Variabile target *y* è **discreto**:
decisione della maggioranza fra y_j := target valori dei prossimi esempi

$y_{nuovo}=\text{argmax}(y_j) \sum \text{sim}(x,x_{nuovo})^a \ (x,y_j)\in N_j$ N_j:= prossimi esempi col valore y_j

Variabile target *y* è **continuo**:
media pesata dei target valori *y* dei prossimi esempi

$y_{nuovo} = \sum (y\ \text{sim}(x,x_{nuovo})^a) / \sum \text{sim}(x,x_{nuovo})^a$ $(x,y)\in N$:= prossimi esempi

p.e. a=1 a=2

Similarità e Distanza

Attributi numerici (continui):
distanza = distanza Euklidiana, distanza Mahalanobis, ...

Distanza è una metrica::
dist(x,x) =0
dist(x,y)=dist(y,x)
$\text{dist}(x,y)\leq \text{dist}(x,z)+\text{dist}(z,y)$

dist(x,y) := 1–sim(x,y) $0 \leq \text{dist}(x,y) \leq 1$

Similarità

Siano x,y oggetti; $A_1,...,A_n$ attributi; $w_1,...,w_n$ pesi degli attributi

$sim(x,y) := \sum w_i \, sim_i(x[i],y[i]) \; / \; w \qquad w = \sum w_i \quad i=1,...n \quad x[i] := A_i(x) \quad y[i] := A_i(y)$

A_i numerico: $sim_i(v,w) := 1 - |v-w| \; / \; (max(A_i) - min(A_i)) \quad v,w \in Dominio(A_i)$

A_i discreto: $sim_i(v,w) := 1$ se $v=w$, altrimento $:=0$

la distanza dist = 1 – sim è una metrica

Pesi

I pesi $w_1,...,w_n$ degli attributi $A_1,...,A_n$ sono importante

Tutti gli attributi vengono inclusi nel calcolo della similarità
Attributi irrilevanti disturbano, perché influenzano il valore di similarità
Attributi correlati aumentano l´effetto della dimensione data da questi attributi

Questo problema può essere mitigato dai pesi adatti

Insomma per il metodo di esempi prossimi il problema d´una selezione giusta degli attributi resta da risolvere. Anche una **definizione più elaborata della similarità** secondo le esigenze dell´applicazione è determinante per il successo del metodo.

Parametri per Cross Validazione

Parametri (*k* o i pesi) possono essere determinati ricercando parametri col risultato migliore (p.e. precisione della classificazione). La precisione viene stimato per cross validazione.

m-fold:
$E = E_1 \cup ... \cup E_m \quad E_i \cap E_j = \varnothing \quad |E_i| = |E_j|$
h_i funzione appresa su $E \setminus E_i$

$Errore_{CV} = \sum errore_{E_i}(h_i) \ / \ m$
Stima per errore vero. Per metodi con una fase d´apprendimento molto cara, si prende un valore basso per m.

qui: $|E_i| = 1$ „omettere uno“, la fase d´apprendimento non costa niente: predire y d´un solo oggetto x con *k* prossimi esempi da $E \setminus \{x\}$ per ogni x per tutti k=1,...,(15) e scelgiere l´ottimo *k*

Supplementi

Ridurre calcolo d´identificare i *k* prossimi esempi ordinarli

Selezionare esempi
Un sottoset degli esempi viene determinato nella fase d´apprendimento escludendo esempi “inutili”. Così la classificazione costa meno.

Prototipi e Rettangoli
Costruire nuovi oggetti (prototipi) representando un gruppo d´esempi e usarli in luogo. Costruire rettangoli representando un gruppo.

Regressione localmente pesata (nella vicinanza di x_q)
Approssimazione esplicita del target $y = w_0 + w_1 a_1(x) + ... + w_n a_n(x)$ minimizzando l´errore quadrato su *k* prossimi esempi di x_q pesati con la similarità a x_q

Oggetti complessi
p.e. oggetti multirelazionali. Si occorre (solo) una funzione di similarità.

Macchine a Supporto Vettoriale

Support Vector Machine SVM Vapnik: Statistical Learning Theory

Insieme di l esempi $E=\{(x^1,y_1),...,(x^l,y_l)\}$ x^i vettore di *n* numeri reali, $y_i \in \{1,-1\}$

Classificazione: y=1 esempio positivo, y=–1 esempio negativo

Un gran numero *n* di variabili $x_1,...,x_n$

p.e. pixel d´un foto, variabile: luminosità d´un pixel
parola in un testo, variabile: frequenza d´una parola

identificare la faccia d´una persona, un testo su un tema ...

Iperpiani: regole lineari

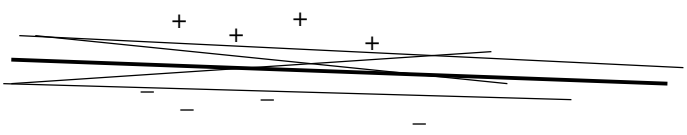
$h(x) = \text{sign}\ (b + w\ x) = \text{sign}\ (b+w_1\ x_1 + ...+ w_n\ x_n)$

w vettore di pesi dei variabili

iperpiano $b + w\ x=0$ è ortogonale à w p.e. $b+x_1+x_2=0$ ortogonale a (1,1)

$b + w\ x>0$ $h(x)=1$ $b + w\ x<0$ $h(x)= -1$

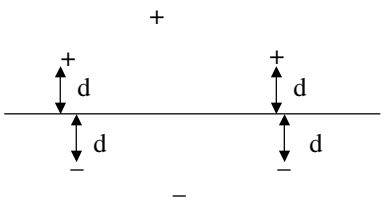
Trova b, w così che $y_i\ (w\ x^i + b) > 0$ i=1,...,l



Iperpiano Ottimale

Trova b, w così che y_i (w x^i + b) > 0 i=1,...,l

La distanza fra gli esempi e l´iperpiano sia massimale



||w|| := lunghezza euclidiana del vettore w

(w x + b) / ||w|| = distanza di x dall´ iperpiano

Trova b, w così che y_i / ||w|| (w x^i + b) ≥d i=1,...,l e d massimale

I vettori x^i con la distanza = d =:vettori di supporto

formano il sottoinsieme minimale degli esempi col stesso iperpiano

Programma Quadratico da Ottimare

Problema equivalente:

Trova $z_1,...,z_l$ $z_i \in \Re$ $z_i \geq 0$ così che:

$-\sum z_i + 1/2 \sum y_i y_j \langle x_i,x_j \rangle z_i z_j$ minimale

$\sum y_i z_i = 0$ i,j = 1,...,l

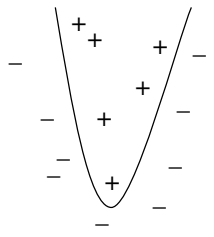
Poi: $w = \sum z_i y_i x_i$ $z_i > 0 \Leftrightarrow x_i$ vettore di supporto

componente $w^k = 0$ -> variabile x^k irrilevante $x=(x^1,...,x^n)$

Algoritmi speciali per questo problema (programma quadratico) di ottimazione che sono efficienti per grandi *l*

SVM nonlineari

Non tutti insiemi d´esempi sono separabili da iperpiani lineari
Poi: Trasformare esempi in uno spazio di dimensione più alta



Funzioni di nucleo

polinomiali: $K(x,z)=(<x,z>+1)^d$

Radial Basis: $K(x,z)=\exp(c\ (x-z)^2)$

d=2, n=2, $x=(x^1,x^2)$, $z=(z^1,z^2)$: $K(x,z)=$
$(x^1 z^1+ x^2 z^2 +1)^2 = <T(x), T(z)>$ con
$T(a,b)=(a^2,b^2,\sqrt{2}\ a\ b, \sqrt{2}\ a, \sqrt{2}\ b,1)$
$T:\Re^2 \dashrightarrow \Re^6$
separabile con $w=(-1,0,0,0, \sqrt{2},0)$ dopo
trasformazione T in uno spazio altodimensionale
Si usa solo $K(x,z) =<T(x), T(z)>$ (e non T)
(sostituire $<x_i,x_j>$ per $<T(x_i),T(x_j)>$ nelle formule SVM

Supplementi

SVM con separazione molle

Ammettere errori d´addestramento: esempi non separati

$y_i\ (w\ x^i + b) \geq 1-e_i \quad i=1,...,l \quad e \quad ||w||^2 + C\sum e_i$ minimale

Approcci dalla Statistica

Metodi di regressione per la classificazione (discriminazione)
discriminazione lineare (Fisher, 1936), due classi C_1 e C_2
distribuzione multivariata normale di x, matrice di covarianze Σ uguale per C_1 e C_2
discriminante lineare $L=b+w\ x$ predice le due classi: $L(x)>1/2(L(\mu_1)+ L(\mu_2))$
μ_i media di x in C_i
b, w così che la probabilità di misclassificazioni sia minimale $w= \Sigma^{-1} (\mu_2-\mu_1)$

discriminazione quadratica
regressione logistica

approcci robusti ACE, Smart, MARS