

Bisogna definire la funzione distanza

Ci si deve poi effettuare il clustering

La funzione D è definita in una coppia di punti in cui spazio metrisco, soddisfa le proprietà:

- 1) $D(X, Y) \geq 0 \quad \forall X, Y \in S \quad \text{e} \quad D(X, Y) = 0 \Leftrightarrow X = Y$
- 2) $D(X, Y) = D(Y, X) \quad \forall X, Y \in S \quad (\text{prop. simmetria})$
- 3) $D(X, Y) + D(Y, Z) \geq D(X, Z) \quad \forall X, Y, Z \in S$

Proprietà Triangolare

Spazio metrico \rightarrow Spazio Euclideo \mathbb{R}^m con m dimensioni dello spazio

Nella pratica le coordinate (ovvero le componenti dei vettori) rappresentano gli attributi o feature degli oggetti nello spazio metrico

Distanza euclidea in \mathbb{R}^m

$$X = (x_1, x_2, \dots, x_m) \quad Y = (y_1, y_2, \dots, y_n)$$

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

Distanza di Manhattan

$$D(X, Y) = \sum_{i=1}^m |x_i - y_i|$$

Norma L_r

$$D(X, Y) = \left(\sum_{i=1}^m |x_i - y_i|^r \right)^{1/r}$$

Norma L_{infinity}

$$D(X, Y) = \max_{1 \leq i \leq m} |x_i - y_i|$$

Distanza del coseno

$$D(X, Y) = \arccos \frac{\sum_{i=1}^m x_i y_i}{\sqrt{\sum_{i=1}^m x_i^2} \sqrt{\sum_{i=1}^m y_i^2}}$$

La media di un insieme di punti è un punto detto **centroide** (centro geometrico)

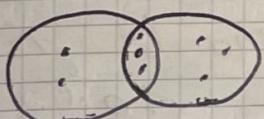
Negli spazi non Euclidei il centroide non è definito ma si definisce il meboide (1-mediana) ovvero un punto che minimizza la distanza media delle distanze dagli altri punti dell'insieme.

Per alcuni spazi non euclidei si potrebbe usare comunque le distanze euclidiene (esempio \mathbb{H}^2)

Distanza di Jaccard

Distanza di 2 insiem S e T

$$D(S, T) = 1 - \frac{|S \cap T|}{|S \cup T|} \quad \text{Similante la Jaccard}$$



$$D(S, T) = 1 - \frac{3}{8} = 5/8$$

Vario sempre tra 0 e 1

Se gli insiem sono coincidenti $\rightarrow D=0$
Se gli insiem sono disgiunti $\rightarrow D=1$

Distanza di edit

Date 2 stringhe X e Y questo è il numero minimo di modifiche da stringa X per ottenere Y oppure a cui occorre

Operazioni: Cancellazione | Inserimento | Sostituzione

$$A = abcde$$

1) cancella b

$$B = acdefg$$

2) metti z dopo c
3) metti f dopo e

$$D(A, B) = 3$$

Distanza di Hamming

$$A(1, 0, 1, 0, 1)$$

$$D(A, B) = 3$$

$$B(1, 1, 1, 1, 0)$$

Numero di bit per cui differiscono

Classificazione dati & clustering

- 1) Metodi gerarchici e agglomerativi: all'inizio ogni punto è un cluster. I cluster vengono poi progressivamente mesi insieme seguendo una normale di ricchezza. Continuano fino a un opportuno criterio di terminazione.
Esempio: k -means
- 2) Metodi di partizionamento: All'inizio i punti sono partiti in k cluster (ogni punto è in un solo cluster). Si individua l'errore (dist) cluster iniziali e poi i punti vengono via via assegnati al cluster obiettivo. Outlier = punto in nessun cluster.
- 3) Metodi sulla densità: I cluster probabili sono estesi finché la densità (numero di punti) in un dato intorno raggiunge una soglia. Questi metodi trovano outliers e cluster di qualsiasi forma. Esempio: DBSCAN, OPTICS
- 4) sulla griglia: Ho delle celle che formano una struttura o griglia. L'algoritmo è applicato su questa griglia (è uno spazio quantizzato). La probabilità di appartenenza di ogni cella è calcolata nel numero di celle in ogni dimensione. Esempio: STING
- 5) sul modello: Si ipotizza un modello per ogni cluster e si trova la miglior classificazione dei dati. Esempio: EM, COBWEB e SOM

Algoritmi per distinguere oggetto & cluttering

- tipo di spazio metrico (euclideo / non euclideo)
- Utilizzo del bias

Problema dimensionale

Ogni volta le coppe in un
insieme fanno più punti ab
dimensione sono eguibidenti

Alta dimensione \rightarrow molti attributi per ogni oggetto

Troppo features \rightarrow Causano liquidazione triste,
punti.

E' importante tenere la dimensione per
calcare delle distanze più o meno distinte

Esempio: 50 dimensioni \rightarrow 5 dimensioni

Tutti eguibidenti

Due distanze ben
distinte

E' importante selezione

poché feature discriminanti

Esempio

Sono n voci. Cosa è tra 0 e 1

E' possibile dimostrare che la D media è $\frac{1}{3}$

Poniamo a α dimensioni $D(x,y) = \sqrt{\sum_{i=1}^{\alpha} (x_i - y_i)^2}$

α grande \rightarrow probabile che c'è una v. i per cui

$$|x_i - y_i| \leq 1 \quad \text{Seque}$$

1) $D(x,y)$ è almeno 1. Si crez un limite
inferiore superiore a 1 per D

2) $D(x,y)$ è pari al minimo a $\sqrt{\alpha}$

Tutte le coppe hanno un limite superiore inferiore

a $\sqrt{\alpha}$. Il convergenza tra i due limiti è la
media.

Ogni tutti i punti hanno D vicino a $\sqrt{\alpha}/3$

Allora la distanza media cresce al incremento di α
(recendo la scacca quadrata)

Clustering gerarchico

Punto base

- a) Amegna un punto ad un cluster rapporto
- b) Unisci i 2 cluster già vicini in uno unico cluster
- c) Ripeti b fino a raggiungere qualche criterio

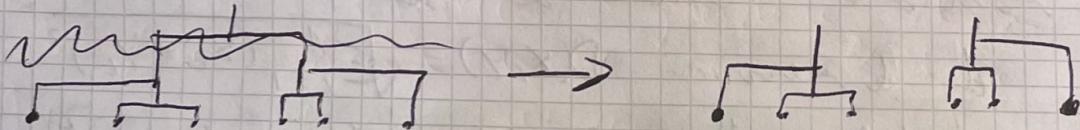
Non si può decidere quando i criteri

Dunque l'algoritmo come notazione di ricorsa
Considera le distanze dei centroidi

Dunque prendi i 2 cluster con centroidi già vicini. Si crea un unico cluster e bisogna calcolare il centroide.

Al clustering gerarchico è associato un altro fatto: sempre che le due sono state costituiti i cluster

Togliendo il dendogramma (di cui tanti) si notano che restano solo i cluster probati fino a un certo livello.



Quindi ogni cluster contiene tutto il dendrogramma già "tagliato" in base ai nostri criteri.

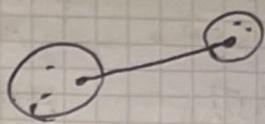
Criteri di terminazione

- a) Se raggiungi già quanti cluster mi serve: (ai quali non c'è più)
- b) Celi tutto l'embogramma (utile in biologia)
- c) Termina l'algoritmo nel momento in cui una fusione probabile un cluster indegno (es: la pizza ha centroidi di punti del cluster calice troppo)

Altre misure di distanza tra cluster

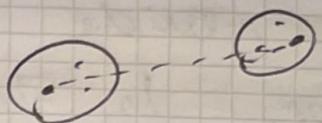
• Single link

$$D(X, Y) = \min_{x \in X, y \in Y} d(x, y)$$



• Complete link

$$D(X, Y) = \max_{x \in X, y \in Y} d(x, y)$$



• Average Link

Distanza media tra tutte le distanze di punti $x \in X$, $y \in Y$

$$D(X, Y) = \frac{1}{|X| \times |Y|} \sum_{x \in X} \sum_{y \in Y} d(x, y)$$

• Medoids di distanze

Distanza tra i medoids (mediane)

Il medoid è il punto del cluster tale che la somma degli altri punti del cluster sia zero e minima. Il medoid è sempre un punto del cluster.

Potremmo scegliere altri criteri di fusione

raggio del cluster : dist. max tra centroide e
un punto cluster

Diametro del cluster : dist. max tra 2 punti
del cluster.

In base a quale scelgo... generalmente convergono in modo tale che il cluster finestrante abbia
il minimo.

Clustering Agglomerativo vs Divisivo

1) Agglomerativo è un processo bottom-up

All'inizio tutti i punti sono cluster e n'è grande

Diviso con le fusioni

2) Tutti i punti sono nello stesso cluster.

Si procebe poi con le meridiane fino a un certo punto.

E' detto approccio "Top-down"

Le metriche che si usano per gli algoritmi di meridiane sono le stesse che si usano per gli algoritmi agglomerativi.

Complessità

Passo iniziale: $O(n^2)$ \leftarrow devo calcolare distanze tra ogni coppia di cluster &

La complessità è $O(n^3)$ in giov, oggi sono $O(n^2)$

Ottimizzazioni

$O(n^3) \rightarrow O(n^2 \log n)$ Usando le cache di punti.

La cosa jettate di prendere il minimo in $O(1)$ e fare inserimento e cancellazione in $O(\log n)$

Ad ogni iterazione

a) Toglie dalla coda dei punti le distanze tra i簇 di 2 cluster da fondere e i restanti cluster ($\leq j \leq 2n$) $\rightarrow O(n \log n)$

b) Metti nella coda le distanze tra il nuovo cluster e di altri cluster ($\leq j \leq n$)
 $O(n \log n)$

Anche con queste ottimizzazioni il clustering generalmente è poco efficiente

Nel spazio non-ndim. il centroide non è definito
Allora si "elige" un punto del cluster e
"centroide" che sarebbe rappresentare il punto
centrale del cluster

Le minime di distanza degli spz. end. rimangono
válide. Basta sostituire il centroide col
centroide

K-means

Lavoriamo su dati non abbiano regole
e non il numero di cluster (k)
È normale trovare per dedurre le im. molti buoni

Psi

- 1) Scegli le punti che abbiano alta probabilità
di far parte del cluster diversi.
- 2) Considera k cluster che chiammo come centroidi
i te punti scelti in 1
- 3) Assegna ogni punto al cluster più vicino
- 4) Dopo aver assegnato tutti i punti
ai cluster dei centroidi
- 5) Riconosci tutti i punti che appartengono
più vicino (ci sono punti che appartengono a cluster
per sbaglio in altri)
- 6) Ripeti 1 e 5 fino a quando non ci sono
più spostamenti oppure fino a quando una
funzione obiettivo non diventa stabile.

Scelta dei K centroidi (Greedy)

- 1) 1° punto random e i metti in S
- 2) Metti in S P che minimizza la d. minima di P
dai punti in S $P = \arg \max_{P \in S} \min_{x \in X} D(P, x)$
- 3) Ripeti 2 finché $|S| < k$

Quando termina l'algoritmo?

Se non ci sono più sostanziali variazioni nella funzione obiettivo non soltanto dei valori.

La funzione ha raggiunto alla media dei punti di clustri dei risultati dei risultati centroidi.

$$E = \sum_{c=1}^k \sum_{x \in P_k} \|x - c_k\|^2$$

P_1, \dots, P_k sono i clustri
 c_k i centroidi.

L'algoritmo termina se la differenza tra i valori delle funzioni in 2 iterazioni successive è sotto a una soglia.

Se non convergono K abbiamo eseguito l'algoritmo più volte e rilanciare le quali dei clustri.

Volture obiettivo: facendo riferimento alla distanza media tra punti e centroidi.

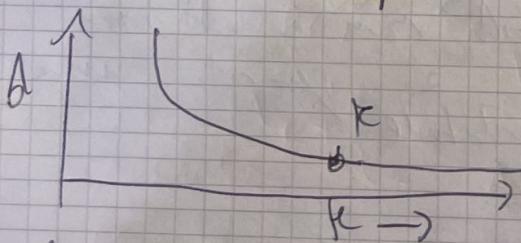
Se il clustering è di alta qualità = di basso costo.

Scelta valore di K

Ad aumentare di K la media dei punti dei centroidi diminuisce.

La funzione prima è brutta e poi si ottengono valori ideali.

Valore ideale: K quello in cui la media inizia a variare poco.



La riduzione sembra essere esaurita nel effettuare le mean più volte per ottenere valori di E .

Soluzione più efficiente: ricerca binaria

Siamo x e y nella stessa classe con differenza di k . Abbiamo grande.

1) Calcoliamo $z = (x+y)/2$ e facciamo il clustering per $k=2$.

2) Se la δ per $k=2$ è vicina a δ per $k=1$ poniamo $y=z$, se la δ per $k=2$ è vicina a δ per $k=3$ poniamo $x=z$.

3) Si rigetate 1) e 2) finché l'intervallo di ricerca non è vuoto.

Complessità

iterazioni t , cluster K , m (per i punti dobbiamo calcolare la distanza)

E' più efficiente del gerarchico perché

• Sposto comunque a una soluzione ottimale

• Non riesce a trovare cluster con forme non convinte o di dimensioni diverse (rotte)

• Senibile al rumore e outliers (anche un numero piccolo di essi influenza le posizioni dei centroidi)

Dobbiamo o dare K in input o designare il suo miglior valore

K-means su Big Data

BFR e CURE: ottimizzazione

↓
Usa una rappresentazione
comune di
cluster
(matriciale)

→ Estensione del k-means per trovare cluster per ogni punto.
Ogni cluster è descritto da dei punti ~~distinti~~ rappresentativi che sono i più vicini per l'affinità di appartenenza.

DB Scan: Algoritmo di mining basato su densità

Cluster:oggiuno con punti; connetti con densità abbastanza alta.

"Oggiuno denso": Punti con almeno ϵ punti in un intorno di raggio fisico.

Nel DBScan definiscono 2 parametri:

- ϵ : raggio massimale cluster
- MinPts: numero minimo punti che deve avere un cluster.

Algoritmo:

• ϵ -intorno di Q : insieme $N_\epsilon(Q)$ dei punti con $d \leq \epsilon$ da Q

• Punto direttamente raggiungibile per densità

P è direttamente raggiungibile per densità da Q se ϵ e MinPts sono

$P \in N_\epsilon(Q)$

Q è un core-point se $|N_\epsilon(Q)| \geq \text{MinPts}$

• Punti raggiungibili per densità

P è raggiungibile per densità da Q se ϵ è minPts e esiste una catena di punti A_1, \dots, A_n tali che $A_1 = Q$, $A_n = P$ tale che A_i è un core-point e A_{i+1} è un punto raggiungibile per densità da A_i .

• Punto comune per densità

Se ϵ e MinPts

P è comune per densità a Q se esiste O tale che $P \in O$ e $Q \in O$ e O è uno cluster raggiungibile per densità da O .

Un cluster in DB non

Il cluster si definisce come un insieme massimale di punti comuni per densità.

Se D è l'insieme dei punti da clusterizzare
il cluster C rispetto a E e M_{pts} , è un
sottinsieme non vuoto di D tale che

a) $\forall P, Q \in D$ se $P \in C$ e Q è raggiungibile da
densità da P (rispetto a E e M_{pts})
allora che $Q \in C$

b) $\forall P, Q \in C$ P è comune da densità a Q

Proprietà a): Massività

Proprietà b): Connessione

Algoritmo

1) Scegli P non ancora visitato (random)

2) Calcola E -intorno S di P .

Se $|S| \geq M_{pts}$ calca cluster C e vai a 3)
se no manda P come outlier e vai a 1)

3) Metti P e S nel cluster C

4) Aggiungi al E -intorno dei punti di C in C
finché i punti non sono tutti.

NB: Data P e S se prendo $Q \in S$ e me
calcolo l' E -intorno non sarà forse da
punti raggiungibili da densità da P .

5) Ritorna a 1) finché tutti i punti non
sono stati visitati.

Alla fine si formano cluster e outliers

Punti inizialmente marcati come outliers potranno
essere meno necessariamente dentro altri cluster

Come scegliere i parametri?

Sì pren $\text{Min Pts} \geq D+1$ dove D è dim spazio.

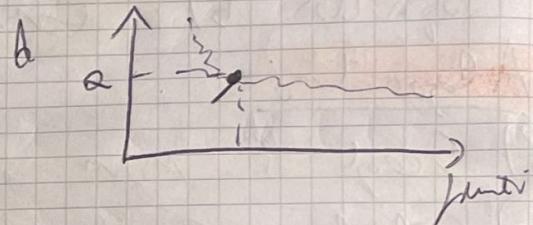
Per avere cluster significativi Min Pts dovrebbe essere più alto di $D+1$ perché più grande è il dataset maggiormente il rumore.

Fatto il pa Min Pts si cerca un valore ottimale per ϵ .

Ordiniamo i punti del dataset secondo distanza dal k-imo punto più vicino.

E' ordinata così la maggioranza dei punti.

E' ottenere un grafico e il valore ottimale di ϵ è l'ordinata del punto in cui la curva scende più drasticamente.



E alto: cluster troppo grandi

E basso: troppi outliers

Complessità

Poco più grande: globale - intenso

Con struttura dati elaborare \rightarrow (logn)

L'E è globale una sola volta per punto

(al logn)

Vantaggi:

- Non serve conoscere a priori nessun cluster
- I cluster possono avere forma arbitraria
- Ci sono gli outliers
- L'andare non cura minimi i punti influisce poco

Svantaggi:

- Scelta parametri molto legante al tipo di dati
- Se i cluster hanno densità molto differente

DB non sono le individua tratti

Condizioni sugli algoritmi di clustering

Non c'è un algoritmo migliore degli altri in
quale generale. Hanno tutti criticità che dicono
che provo trovare più o meno vantaggiose in
una direzione.

Coefficiente di Silhouette

Qualifica la qualità dei cluster ottenuti

D_i : la media tra un'osservazione i e tutti i
junti dello stesso cluster

Per ogni i è per ogni X cluster i calcola
 D_X media tra i e i junti di X

Consideriamo il cluster C con D_X minima

$$C_i = \delta \text{ da } C \text{ e } i$$

$$S_i = \frac{C_i - D_i}{\max(C_i, D_i)}$$

S_i più vicina 1 è
il meglio è

$S_i > 0$ è ben
clusterizzato

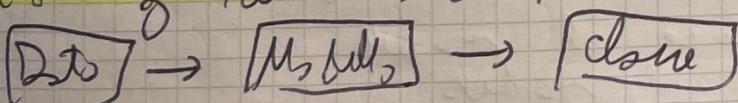
$S_i < 0$ è in un
cluster sbagliato

$S_i = 0$ è a metà
tra due cluster.

Classificazione

Preface l'etichetta delle classi a cui appartiene
un oggetto.

Classifica con modello un dato bombardare sul
training set e lo classificare sui punti delle labels



Preditore

Funzione nel continuo e ad ogni istante emette
un valore. Ese: Colore \rightarrow Valore

~~X X~~

~~% %~~

Classification

Distinguendo il modello

Ogni osservazione ha un class , quindi insieme di type è detto training set.

Il modello è rappresentato con regole di classificazione, alberi decisioni ecc...

Usare il modello : classificazioni nuovi oggetti senza sapere l'etichetta.

Tuttavia la accuratezza del modello.

Usare un secondo dataset con test che deve essere indipendente dal training set.

Stimiamo l'accuracy riflettendo su quale classificazione va bene.

Se l'accuracy passa un test di validazione il modello è pronto.

La classificazione è un supervised learning

- EugenioWeb : i testi sono accompagnati da etichette e prima di usare il classificatore
- Omegavision : etichette non note, si vuole suddividerle in gruppi omogenei

Un classificatore deve avere questi attributi / requisiti

- Accuratezza | classificatore : prende etichette classificate | preditore : prende corretto valore
- Velocità | tempo per costruire il modello | tempo per usare il modello
- Robustezza : capacità di manipolare dati con tranne e mancanze
- Scalabilità : efficienza sui dati nel disco
- Dove fare risultati componibili

Albero decisionale

Ogni nodo contiene un test su un attributo e utilizza
una qualche sotto-albero per fare la predizione

Ogni foglia ha un'etichetta
Quindi la domanda è una tappa è una foglia in cui finisce

Ad ogni nodo vi associano 5x i valori delle
tutte le possibili condizioni testate portando tutti
i valori fino ad arrivare al nodo X

L'albero non può finire come una serie di testi

IF-THEN

Dobbiamo creare queste regole per costruire l'albero

Costruzione albero decisionale

E' costruita con tecniche top-down divide et impera

1) Se tutte le tuple hanno valore uguale su un attributo dominante non faccio moltiplicazione
e basta una foglia

2) Se ho numerose su un attributo A sono ancora
tutte e ri-ordino le tuple per valore S degli
elementi sulla base dei valori di A (splitting)

3) Creo tantissime foglie del nodo e quindi 380 i
valori di A

4) Per ogni foglio X e C

1) Se X è gesso (tutte le tuple associate hanno lo stesso valore sull'attributo dominatore) fermati

2) Se X è ingesso e ci sono attributi lo puoi
ridurre appena tecnicamente i valori precedenti
in X

3) Se X è ingesso ma non ci sono attributi lo
puoi ridurre fermati e aggiungi a X
l'etichetta con già tutte l'osservazioni.

Varietà

Nel caso un moto logico non crede per ragioni di attributi da voler fare solo è ingenuo scommettere sulla logica una distribuzione di probabilità più incauta già probabile.

Dato C , $P(C)$ è lo prob. di avere con dato di classe C nel S delle tuple di moto logico.

$$P(C) = \# \text{On. } C \text{ in } S / \# \text{On. in } S$$

L'etichetta associa alla volta volata nella base di P

Splitting

a) Categorie att.

- Crea l'impresario, una per ogni valore dell'attributo
- Partitiona S in K nei bambini sui valori di A
- Crea moto per ogni S_i

b) attributi continuo

- Definisci δ soglia e partitiona $S \& X$ in S_1 e S_2
- a secondi che $A \geq \delta \& A < \delta$
- S_2 soglia è scelta in modo che ogni partizione
- δ sia un minimo di distanza.
- Aggiungi $\delta \times 2$ nodi finali per S_1 e S_2

c) Attributi binari

- Partitiona S in S_1 e S_2 se $A = T \& A = F$
- Aggiungi $\delta \times 2$ nodi finali in S_1 e S_2

Costruzione albero decisivo

La costruzione di grande ordine come con N rami per gli attributi.

Oggetto:

• Scoprire l'albero più semplice e compatibile.

Trovare l'albero, simile a NP-hard

Viene dunque approssimato

Strategy greedy

Seleziona ad ogni passo l'attributo che divide i dati in modi che sono "relativamente "genuini"
In pratica cerca di ottenere il grafo possibile
a delle foglie

Per finire ID3

Algoritmo greedy ricorsivo

1) Punto base

- Se un nodo è già completamente in foglia
- Se il nodo non è pieno ma non ci sono più entro la sua voluttà di creare foglie, l'etichettare con quella di maggioranza

$$t = \text{Node full if there}$$

2) Punto ricorsivo

- Seleziona attr. A che minimizza una misura di goodness
- Fare partizione dell'universo S in base agli elementi associati a t sulla base di A
- Creare tutti i nodi di t quanti sono i valori di A
- Applicare ricorsivamente i punti precedenti sui nodi minori considerando i corrispondenti attributi

Algoritmo in Prendo Tabella

Build-DT (S, Attributi)

IF tutti gli esempi hanno stessa etichetta (valore dell'attr. classificazione)

THEN RETURN (nodo foglia con etichetta)

ELSE

IF Attributi = \emptyset THEN RETURN (foglia con etichetta di maggioranza)

ELSE

Seleziona A con goodness max come tabella

FOR EACH valore V di A

Creare dimensione della tabella con condizione $A = V$

Build-DT ($\{x \in S : x.A = V\}$, Attributi - {A})

Nel caso i valori non sono continui né ordinati, per fare lo splitting bisogna scegliere un valore di soglia.

Se l'attributo è continuo si devono calcolare gli valori di goodness per l'attributo: uno per ogni possibile valore della soglia. Si ricorda come valore della soglia quello con goodness più alta.

Misura di goodness

Gli algoritmi sono di solito identici e meno della misura di goodness.

Algoritmo ID3: usa come misura Information gain

Qui si sceglie l'attributo per abbassare progressivamente l'entropia

E' giusto dire infatti che + Purezza (α) - Entropia (β)

S_x è un insieme e abbiamo 2 classi $P \in N$
 S_x di P contiene p elementi, di N n .

Entropia di S_x (NB $H(X) = \frac{2}{T_X} \log\left(\frac{2}{T_X}\right)$)

$$H(S_x) = -\frac{P}{P+n} \log \frac{P}{P+n} - \frac{n}{P+n} \log \frac{n}{P+n}$$

Entropia media di S_x rispetto ad A

Media presa delle entropie dei singoli S_i (parte della partizione di S_x)

$$H_A(S_x) = \sum_{i=1}^k \frac{P_i + M_i}{P+n} H(S_i)$$

$$k = |\text{Part}(S_x)|$$

Nel caso generale

S_x ha m classi $C_1 \dots C_m$

$$H(S_x) = - \sum_{i=1}^m \frac{\text{Prog}(C_i, S_x)}{|S_x|} \log \frac{\text{Prog}(C_i, S_x)}{|S_x|}$$

A può avere come valori $\{x_1, \dots, x_K\}$

Partizioniamo S_x nei S_j in base ai valori di $A(k)$.

$$\bar{H}_A(S_x) = \sum_{j=1}^K \frac{|S_j|}{|S_x|} \cdot H(S_j)$$

L'informazione gain si definisce come la riduzione di entropia ottenuta partizionando S_x sulla base del valore di A .

$$Gain(A) = H(S_x) - \bar{H}_A(S_x)$$

L'algoritmo decide come ~~valore di A~~ studiare quella

con gain massimo.

Considerando che $H(S_x)$ è fisso in un modo l'obiettivo allora è cercare $\bar{H}_A(S_x)$ minimo.

Questo approccio permette di trovare la classe in un'altra superficie.

Limiti

Questo minima l'entropia e forse di troppo con molti citi.

Potrebbe favorire tent con molti citi simili ma poco significativi per la predizione come ad esempio negli ID univoci in un DB.

Ogni partizione basata su ID ha una rotta tra le branche ~~isolate~~ verso creare un solo ID che ha informazione gain massimo ma non ce ne sono alcun vantaggio dividere un DB per gli ID.

Gain Ratio C5,5

Il gain ratio è uguale a C5,5 che riduce il bias.

$$\text{Spltinfo}(A) = - \sum_{i=1}^K \frac{|S_i|}{|S_x|} \log \frac{|S_i|}{|S_x|}$$

Ottimale questo valore se gain ratio è definito

Come $\text{Gain Ratio}(A) = \frac{\text{Gain}(A)}{\text{Spltinfo}(A)}$

Lo split ratio è "congugato" al gain

NB: Spltinfo è partitivo

Quindi A deve avere gain alto e Splt basso

Spltinfo basso \leftrightarrow poche ramificazioni
(e ben distinte) \leftarrow Poco il Ratio va alto

Gini Index CART*

Misura l'importanza delle tighe b in modo Sx

Abbiamo una classe i e una tiglia T di classe i
resta a calcolare.

Supponiamo di assegnare a T una classe j. Probabilmente non è la classe i. Allora si calcola la probabilità che la classe i sia $\neq j$.

Gini index misura la prob che i sia $\neq j$.
 vogliamo che sia gini basso possibile

- Possibilità di raffigurare una osservazione come classe i in Sx
- Prob reale essere $j \neq i$ quindi la distribuzione nel modo
- P_i
- $1 - P_i$

Il raggruppamento vede per ogni c

$$gini(S_x) = \sum_{i=1}^m \left(\sum_{j=1, j \neq i}^m p_j \right) = \\ = 1 - \sum_{i=1}^m p_i^2$$

Dato un nodo con S_x tuple con n dati id
gini index è

$$gini(S_x) = 1 - \sum_{i=1}^n p_i^2$$

p_i : frequenza della classe i in S_x

Gini index di uno split

S_x è partitionato in $S_1 \dots S_k$

$$gini_{split}(S_x) = \sum_{i=1}^k \frac{|S_i|}{|S_x|} gini(S_i)$$

E' una media pesata di $gini(S_i)$

Negli alberi che minimizzano $gini_{split}(S_x)$

Ogni ramiha minima e più grande possibile e
meno l'elaborazione possibile.

Pruning

Consiste nel rimuovere rami che non contribuiscono
ad una corretta classificazione.

Il Pruning induce ad un albero di essere più
semplice il modello

Il pruning si fa se l'albero è troppo complesso
(potrebbe essere complesso perché è sbagliato
troppo al training set)

• Prepruning = fatto in fase di costruzione dell'albero

• Post Pruning = fatto dopo aver creato tutto l'albero

Post-pruning: già N nodi non più da rec.

CART usa "primitve pruning"

CART usa "cost complexity pruning"

Pruning Primitivo

Ho un albero T con $\text{tobac} \times$
Se definisco error rate di T le tighe Sx devono
essere entro i limiti.

Una tiglia è stata il Sx albero bontà ϵ
nulla variazione di error rate.

- Variazione = puntura \rightarrow quindi l'errore rate è già
stato se c'è il sotto-albero in sostituzione
con una foglia

ϵ è la ~~puntura~~ di questa variazione che è primitiva.
(pruning) \rightarrow borsa in ϵ

Albero \Rightarrow Pruning P. Poni

- 1) Calcolo $E_p(T)$ misura primitiva dell'errore
sotto forma di base o splitting
- 2) Calcolo $E_p'(T)$ dopo lo splitting
- 3) Se $E_p'(T) > E_p(T)$

2) Sostituisci T con una foglia
N tickette sarà quella di maggioranza delle
le voci delle tickette delle foglie di T

Stima primitiva

$$E_p(T) = \frac{\# \text{ tighe dove } c' \neq c \text{ in } Sx + \epsilon}{\# \text{ tighe in } Sx}$$

Perciò fare lo

splitting:

c : tickette di classe in
maggioranza

introduzione
pruning

Dopo splitting

$$E' p(T) = \frac{\sum_{i=1}^k \# \text{ tuple dove } c'_i \neq c_i \text{ in } S_x}{\# \text{ tuple in } S_x} + k \varepsilon$$

c'_i : dichetto in maggioranza in S'_x

Cost - Complexity Pruning CART

Conservare $T_0 \dots T_m$ dove $T_0 \dots T_{m-1}$ l'albero decisionale
e T_m quello con la più bassa $E(T)$.

T_i è ottenuto da T_{i-1} rimuovendo un sottoalbero
da T_{i-1} e sostituendolo con una foglia così
che dichetto in maggioranza del sottoalbero che c'era
può sopravvivere.

Se $E(T_i)$ è l'errore totale ottenuto per T_i
il sottoalbero t da rimuovere è quello che
minimizza

$$\frac{E(\text{prune}(T_i, t)) - E(T_i)}{| \text{leaves}(T_i) | - | \text{leaves}(\text{prune}(T_i, t)) |}$$

$\text{prune}(T_i, t)$: albero ottenuto togliendo da T_i il
 $\text{leaves}(T_i)$: foglie di T_i

Estrazione Regole da un albero decisionale

Dall'albero si provi estrazione le IF... THEN...

Abbiamo una regola per ogni cammino dalla radice
a una foglia

Le regole sono mutuamente esclusive ed esaurienti
2 test (o più) lungo un cammino sono
coincidenti in AND

IF test 1 AND Test 2 THEN

Stabilire le quanti di una regola per una classe C.

Coverage: numero relativo degli esempi delle regole.

Accordanza: numero relativo delle regole di classe C coperte dalla regola.

Una buona regola deve avere un coverage non a caso.

Esigenze positivi: tuple classe C coperte dalla regola.

Esigenze negativi: tuple NON C coperte dalla regola.

E' possibile avere di bombardare nelle regole.

Dobbiamo trovare il giusto equilibrio tra le regole sulle classi C che soddisfano gli esigenze positivi.

Algoritmi di Covering: FOIL, AQ, CN2, RIPPER

L'obiettivo è individuare una regola per una classe C che copre molte tuple di classe C e nessuna o poche tuple altre classi.

Si mettono tutte le tuple del TR.

Regole apparse una per volta.
greedy basato sulla qualità. Usando un criterio della valutazione.

Ottendo una regola e ripetuta per ogni pos.

Processo ripetuto finché non riesce tutto le tuple.

Foil e Ripper usano la strategia greedy per generare la regola da aggiungere sono pos.

C close : partiamo dalla regola più grande possibile per ottenere C

IF TRUE THEN C

Iniziamo ad aggiungere condizioni che aumentano C e accrescerà l'abbassare la coverage il più possibile.

IF true THEN C=x

Copre tutte le tuple del training set.

Aggiungendo condizioni la regola diventa più specifica e il numero di esempi positivi scende. Sempre anche il numero di esempi negativi.

La regola va resa più specifica possibile riducendo gli esempi negativi il più possibile mantenendo alti gli esempi positivi.

Algoritmi Fois e Rizzet

Le condizioni sono aggiunte fino quando la regola manterrà un livello di qualità massimale ad una regola.

L'idea di qualità è Fois gain

• pos: esempi positivi • neg: negativi

• pos': dopo costruzione regola • neg': dopo contr. reg.

$$\text{Fois gain } f(R) = \text{pos}' \cdot \left(\log \frac{\text{pos}'}{\text{pos}' + \text{neg}'} - \log \frac{\text{pos}}{\text{pos} + \text{neg}} \right)$$

Il Fois gain favorisce regole con accuratezza alta e coprono molti esempi positivi.

NB: Nelle regole che negativi sono coperti dalla regola. I negativi definiti come gli esempi falsi della regola.

Dominio generativo

Probabilità è etichetta per probabile (probabilità su un modello probabilistico sono basati i teoremi di Bayes)

Teoremi di Bayes

H_c : ipotesi X è classe C

$P(H_c | X)$ è quelli che vogliono calcolare ed è detta prob a posteriori.

- $P(X)$ prob di osservare X , è detta evidenza
- $P(H_c)$ prob che H_c sia vero a priori
- $P(X | H_c)$ prob di osservare X dato H_c . È detta likelihood

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

La classe C che minimizza $P(H_c | X)$ è quella che ha la più bassa

Probabilità X è costante $P(X)$.

C'è bisogno minimizzare $P(X | H_c) \cdot P(H_c)$

Dobbiamo calcolare $P(X | H_c)$ e $P(H_c)$.

$P(H_c)$ è facile da calcolare, è una stima da fare dato il dato ut.

$P(X | H_c)$ è una funzione di probabilità condizionata a variabile dove ogni variabile è un attributo di X

Naïve Bayes

Assume le variabili sono indipendenti condizionalmente
(indipendenti tra loro fatta una variabile)

$$\cdot P(X|C_i) = \prod_{k=1}^m P(X_k|C_i) = P(X_1|C_i) \cdot \dots \cdot P(X_m|C_i)$$

Il calcolo della likelihood è facile a questo
punto da ~~calcolare~~, bisogna considerare le
frequenze.

Alcun' esempio: $P(X_k|C_i)$ è il rapporto tra
numeri tuple di classi C_i con valore X_k per A_k
e numeri di tuple di classi C_i .

Alcun' esempio: $P(X_k|C_i)$ calcolato come valore
assunto per X_k nella distribuzione gaussiana.

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$x = X_k$$

μ media dei valori di A_k

σ dev. standard de
valori di A_k

Naïve Bayes richiede che le prob. condizionali
siano diverse da 0

Se le prob. sono piccole il loro prodotto potrebbe
portare a problemi di underflow

Per ovviare al problema si usa il log-likelihood
invece di usare $P(X|C_i)$ visto $\log P(X|C_i)$

Nella probabilità si viene a creare una somma
di tanti logaritmi.

Vantaggi: facile e veloce e si solito di buon
risultato.

Svantaggi: ci sono assunzioni di indipendenza che
potrebbero essere troppo forti.

Le dipendenze tra variabili non sono gestite da Naïve Bayes

Classificazione Binaria

Cercasi di predir la classe di appartenenza o non
di dati sconosciuti.

- Eseguendo la funzione F portando da dati sconosciuti come per ogni attributo.
- Se X è un nuovo dato $\Rightarrow F(X)$ è la classe di X .

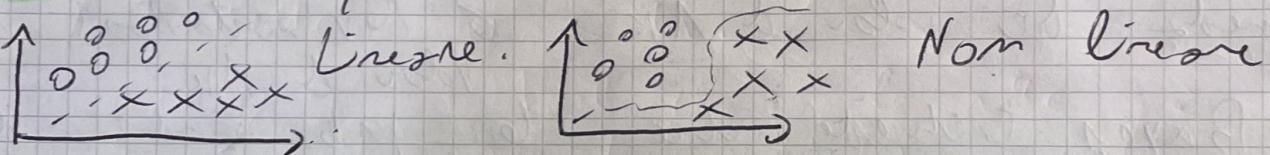
Sono tre classificazioni più comuni, sono molto robuste.
Il calcolo del valore della funzione decisione è facile.

Riassunto: una classificazione del modello molto laboriosa.

Classificazione Lineare vs. non lineare

$y = f(\sum_{i=1}^m w_i x_i)$ La classificazione è basata sul valore di una funzione lineare degli attributi x .

Nella non-lineare la classificazione è fatta usando una funzione non lineare.



Quello che accade nella classificazione lineare
non è un iperipiano regolare.

Perceptron

Algoritmo di classificazione lineare binaria.

Dato X tali $f(\vec{X}) = \begin{cases} 1 & \text{se } \vec{w} \cdot \vec{X} + b > 0 \\ 0 & \text{se no} \end{cases}$

Gli elementi del training set sono processati una
alla volta e i dati aggiornati ogni volta da
una funzione.

w e X hanno stessa cardinalità (sono vettori)

D = $\{(x_1, b_1), \dots, (x_n, b_n)\}$ x_j è la terna e
 b_j è la sua classe

τ : learning rate ($0 < \tau < 1$)

Un altro valore va considerato nello passare W

$y_j = f(x_j)$ dove f è output per la j -esima terna

$x_{j,i}$ è i -esimo attributo della terna x_j

$x_{j,0} = 1$

$w_i(t)$: peso i -esimo relativo al tempo t

Dato che $x_{j,0} = 1 \rightarrow w_0(t)$ equivale a b

Perceptron

1) Inizializza $w_i(0) = 0$ o valore piccolo

2) Per ogni $(x_j, b_j) \in D$

2) Calcola $y_j(t) = p[\vec{w}(t) \cdot \vec{x}_j] =$
 $= p[w_0(t)x_{j,0} + \dots + w_m(t)x_{j,m}]$ Output attuale

b) Aggiorna i pesi

$w_i(t+1) = w_i(t) + \tau(b_j - y_j(t))x_{j,i}$

3) In caso di learning offline fatti 2) finché

$\frac{1}{M} \sum_{j=1}^M |b_j - y_j(t)|$ è inferiore a un soglia o
finché non viene fatto un numero di iterazioni
sufficie per la classificazione.

Se: b_j sono linearmente separabili l'algoritmo
perceptron converge.

SVM Support Vector Machines

Algoritmo di classificazione binaria non lineare da non lineare.

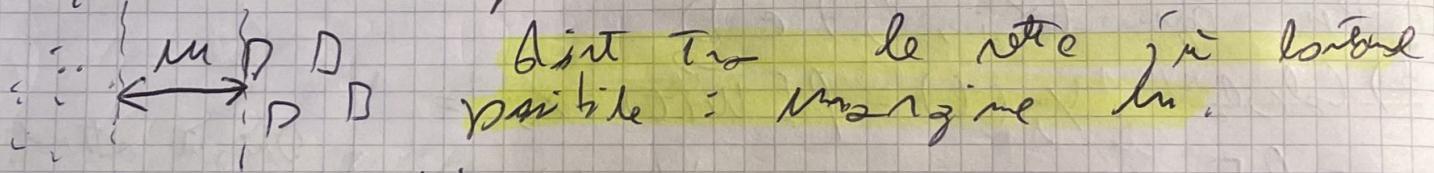
Dati linearmente separabili \rightarrow trova ierzione migliore

Dati non linearmente separabili:

effettua mapping dei dati in uno spazio a dimensione maggiore in cui è possibile trovare una ierzione separazione.

- In questo modo spazio l'SVM trova il suo separatore
 - Un separatore è qui dimensione decisa dai valori, i dati
 - i dati sono in dimensione
 - Esiste sempre uno spazio in cui trovare il separatore
- L'ierzione separazione ottimale è trovata usando vettori di supporto (tuple di frontiera) e margini.

Se i dati sono linearmente separabili ci sono infinite rette separate.



L'ierzione ottimale è quella che massimizza la distanza quindi facendo il più possibile l'errore.

I due ierzioni sono trovati con i vettori di supporto (tuple di frontiera)

Ottieni nel training set m punti $\vec{x}_1 \dots \vec{x}_m$

e classi $y_1, y_2 \dots y_m$ classi (valori 1 o -1)

$$\text{Sic } y = \vec{w} \cdot \vec{x} - b = 0$$

Equazione ierzione separazione

$$b = bias$$

w vettore ric

hors i dati normalizzati sono 0 e 1

Eq del 2 iperbole che definiscono i margini

$$H_1 = \vec{w} \cdot \vec{x} - b = 1$$
$$H_2 = \vec{w} \cdot \vec{x} - b = -1$$

le tangenti ai margini sono le vette di appoggio
distanza geometrica

Il margine è $\frac{2}{\|\vec{w}\|}$

Dobbiamo minimizzare $\|\vec{w}\|$ ora tale problema

del prodotto scalare di \vec{w} con se stesso.

Variabile

Se $y_i = -1 \rightarrow \vec{w} \cdot \vec{x}_i - b \leq -1$ ↪ le tangenti sono a sinistra di H_1

Se $y_i = 1 \rightarrow \vec{w} \cdot \vec{x}_i - b \geq 1$ ↪ le tangenti sono a destra di H_2

Ovvvero $y_i(\vec{w} \cdot \vec{x}_i - b) \geq 1$

Quindi dobbiamo risolvere

$$\begin{cases} \min \|\vec{w}\|^2 \\ y_i(\vec{w} \cdot \vec{x}_i - b) \geq 1 \end{cases} \leftarrow \begin{array}{l} \text{S'usa il gradiente per} \\ \text{comodità.} \end{array}$$

Dobbiamo trovare il min di $\|\vec{w}\|^2$ tale che vale il rimeso.

Quello è l'Hard margin.

Nel "soft" non ammette la possibilità che dei punti siano nella regione tra i vette di appoggio

Così chiamano un Hard margin con margine stretto

Dobbiamo trovare un numero margini da riducere al minimo.

Hard margin \rightarrow overlapping

Soft troppo stretto \rightarrow compromette l'accuratezza

Introduzione $\varepsilon_1 \dots \varepsilon_m$ dette variabili stoc

$$\varepsilon_i = \max(0, 1 - y_i(\vec{w} \cdot \vec{x}_i - b))$$

ε_i vale 0 se $y_i(\vec{w} \cdot \vec{x}_i - b) \geq 1$ ovvero il vettore è classificato

Se \vec{x}_i non soddisfa il vincolo ε_i è proporzionale alla distanza del punto dal margine della classe corrispondente

ε_i è il numero non negativo finché che soddisfa $y_i(\vec{w} \cdot \vec{x}_i - b) \leq 1 - \varepsilon_i$

Obiettivo: avere ε_i piccoli

Introduciamo λ che controlla il trade-off tra lunghezza margine e numero violazioni tollerate

Il problema diventa

$$\begin{cases} \min \left(\frac{1}{m} \sum_{i=1}^m \varepsilon_i + \lambda \|\vec{w}\|^2 \right) \\ y_i(\vec{w} \cdot \vec{x}_i - b) \leq 1 - \varepsilon_i \\ \varepsilon_i \geq 0 \end{cases}$$

Più grande è λ più traenzabile è $\|\vec{w}\|^2$

Ovvero diventa meno importante la dimensione del margine.

Vorremo λ minimizzare il perimetro che ha la dimensione del margine nel calcolo dell'ipersuperficie

Il problema si può proiettare sulla linearizzazione del perimetro

$$\begin{cases} \max \left(\sum_{i=1}^m c_i - \frac{1}{2} \sum_{i,j=1}^m y_i c_i (\vec{x}_i \cdot \vec{x}_j) y_j c_j \right) \\ \sum_{i=1}^m c_i y_i = 0 \\ 0 \leq c_i \leq 1/m \lambda \end{cases}$$

$c_1, c_2 \dots c_m$: coefficienti di Lagrange

Calcolati i c_i il vettore be' per \vec{c} :

$$\vec{w} = \sum_{i=1}^m c_i y_i \vec{x}_i$$

b (lato) si puo' calcolare prendendo un vettore a
rispetto \vec{x}_i e risolvendo questa eq.

$$y_i (\vec{w} \cdot \vec{x}_i - b) = 1 \Rightarrow b = \vec{w} \cdot \vec{x}_i - y_i^{-1}$$

SVM con dati non lineariamente separabili

Spesso i dati non sono separabili in uno spazio con
dimensione più alta.

Si fa così la fine mapping φ

$$\vec{x} = (x_1, x_2, x_3)$$

$$\vec{\varphi}(\vec{x}) = (x_1, x_2, x_3, x_1^2, x_1 x_2, x_1 x_3)$$

L'eq dell'iperpiano separatore è

$$y = \vec{w} \cdot \varphi(\vec{x}) - b \quad (\text{e' che non sono lineariamente separabili})$$

Nuovo problema

$$\left\{ \begin{array}{l} \max \left(\sum_{i=1}^m c_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m y_i y_j c_i (\varphi(\vec{x}_i) \cdot \varphi(\vec{x}_j)) c_j \right) \\ \sum_{i=1}^m c_i y_i = 0 \quad \forall i \in \{1 \dots n\} \\ 0 \leq c_i \leq \frac{1}{2n} \lambda \end{array} \right.$$

I vari problemi restano pressoché identici in quanto potremo avere tante dimensioni.

Si introduce allora la fine Kernel

$$k(\vec{x}_i, \vec{x}_j) = \varphi(\vec{x}_i) \cdot \varphi(\vec{x}_j) \quad (\text{calcola lo stesso})$$

Vedrete che è molto meno dimensionale in quanto anziché il mapping
puntualmente il mapping non viene fatto.

Permette di sostituire al probotto soluzioni che non rispettano il voto interno.

Lavoriamo sempre nello spazio originale.

Tighe function Kernel

Kernel polinomiale grado k : $(\vec{x}_i \cdot \vec{x}_j + 1)^k$

Kernel gaussiano: $e^{-\frac{\|\vec{x}_i - \vec{x}_j\|^2}{2\sigma^2}}$

Kernel sigmoidale: $\tanh(K \vec{x}_i \cdot \vec{x}_j - \delta)$

Diverse implementazioni di SVM classificano e

già classificate.

E' giusta bene per i dati ad alta dimensione.
Infatti la complessità dipende solo dai numeri di tuple e non dagli altri dati.

SVM ha meno overfitting rispetto ad altri classificatori.

Classification Lazy

Numeri 222 Un training set è calcolata la funzione decisione per ogni punto in un certo insieme di classificazione.

Un algoritmo lazy è il K-nearest neighbors.
Il tempo di preddizione è otto ma quello di training è praticamente nullo.

Sono classificatori molto lenti quando abbotti le dimensioni.

Sono sempre abbotti a dataset grandi a bassa dimensione e che si aggiornano continuamente.

KNN

Finito K , prendi le tuple già vicine allo tuple x da classificare.

Analogo a X la classe più vicinanza tra le K tuple.

Dopo aver aggiunto un punto in base alla distanza
CNN punto

- Augura un per mille base della distanza del punto da dove viene
- I punti più vicini hanno il più maggiore KNN e può usare per prevedere valori continuo.

Previsione

Diversa dalla classificazione in quanto produce un valore continuo e non un'etichetta.

La tecnica più importante è la regressione.
Nella relazione tra variabile predittiva e una variabile detta risposta.

Metodi di regressione: lineare, non lineare, generalizzati, di Poisson, log-lineare, altro di regressione.

Regressione lineare semplice

y risposta prediction: \hat{X}

Vogliamo calcolare $y = W_0 + W_1 X$

I nostri parametri da trovare sono W_0 (intercetta) e W_1 , il coefficiente angolare.

Ottenerne W_0 e $W_1 \Leftrightarrow$ Metodo dei minimi quadrati

X e \hat{Y} : retta di valori fatti

Trova $y = W_0 + W_1 X$ che minimizza l'errore
osservato fra i dati attuali e le stime ottenute
con la retta

$$W_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2} \quad W_0 = \bar{y} - W_1 \bar{x}$$

Controlla la retta calcolando l'errore totale
sottraendo (vallo retta) e i valori rei.
Perciò la retta che ha errore minore.

Regessione Lineare multipla

\hat{y} ha già variabili.

$$Y = W_0 + W_1 X_1 + W_2 X_2$$

Il problema è risolvere con un'estensione del metodo dei minimi quadrati.

Regessione Non Lineare

In alcuni casi c'è già ricordare in una funzione lineare

Esempio: $Y = X_0 + W_1 X + W_2 X^2 + W_3 X^3$ E' corretto
in $Y = W_0 + X + W_1 X_2 + W_3 X_3$ con $X_2 = X^2$
 $X_3 = X^3$

In generale è facile trasformare in lineari funzioni polinomiali e funzioni potenza.
Alcune funzioni, tuttavia, in modo non lineare, non sono convertibili.

Classificatori Ensemble

Fondi i risultati di un classificatore

Ricorda molto già costruzione quindi è una con modelli (tra cui solo uno tipo) vedi come gli altri decisionali.

Il problema consiste nel combinare i classificatori.

Bagging / Bootstrapping

Sono M, \dots, M_k classificatori

ensemble learning

1) Si suddivide in k sottoinsiemi T_1, \dots, T_k il training set (congiunto formato con ripetizione)

2) Addestra M_i su T_i

3) Combinare i risultati dei modelli

• Regressione: sia la media • classificazione: la maggioranza

Random forest

Combina i risultati di diversi alberi decisionali
Molti è tecnica bootstrap

Il Random forest adatta ogni albero su un
bootstrap sample e in attributi

Se $P = \text{num. attributi} \Rightarrow m = \sqrt{P}$ classificazione

Gli riduiscono la
correlazione tra alberi

$$m = \sqrt{P}/3 \text{ per regressive}$$

Numeri alberi

Senza bagging negli attributi gli alberi sarebbero
molto correlati

Valutazione di un classificatore

Matrice di confusione: rappresenta l'acconciatura
della classificazione

Righe: valori reali

Colonne: valori predetti

Elemento i, j : numero volte che ha classificato
la classe vera i come j

Alta accuratezza: lungo diagonale $\neq 0$ e gli altri
forni uguali o vicini a 0

Probabilità

real	gatto	cane	cav. gatto	Somma
gatto	5	2	0	7
cane	3	3	2	8
cav. gatto	0	1	11	12
Somma	18	6	13	27

7 Predetti; 8 gatti.

Accuratezza
Percentuale corrette
classificate bene.

$$\text{Acc}(M) = 19/27$$

$$19 = 5 + 3 + 11$$

$$27 = \text{totale animali}$$

$$\begin{aligned} \text{err-M}(M) &= \checkmark \\ &= 1 - \text{acc}(M) = \\ &= 1 - 19/27 = 8/27 \end{aligned}$$

Suppongo \exists due classi P e N
pos: tipo classe P neg: tipo classe N

Ottimismo \hookrightarrow ottimismo

True Positive: tipo classe P classificata come P

True Negative: // // N // // N

False Positive: // // P // // N

False negative: // // N // // P

Misure di accuratezza

$$\text{Recall} = \frac{\text{TP}}{|\text{Pos}|} (= TPR)$$

$$\text{Specificità} = \frac{\text{TN}}{|\text{Neg}|}$$

$$\text{False Positive Rate} = \frac{\text{FP}}{|\text{Neg}|} (FPR)$$

$$\text{False Discovery Rate} = \frac{\text{FP}}{(\text{TP} + \text{FP})}$$

$$\text{Precisione} = \frac{\text{TP}}{(\text{TP} + \text{FP})}$$

$$\text{Accuracy} = \frac{(\text{TP} + \text{TN})}{(|\text{Pos}| + |\text{Neg}|)}$$

$$F_1 \text{ score (tra } 0 \text{ e } 1) = 2 \cdot \frac{\text{Precisione} \cdot \text{Recall}}{\text{Precisione} + \text{Recall}}$$

(Vicino a 1 vuol dire
accuracy alta)

In un classificatore binario la distinzione potrebbe
essere fatta sulla base di una soglia θ

θ : potrebbe essere il risultato di una regresione.

Ad esempio usi la regressione per calcolare uno
score di una mail e se supera la θ
la classifica SPAM o se non lo supera la classifica
non SPAM

A variazione di θ continua la curva ROC che
rappresenta il TPR in funzione del FPR e
risponde a θ

Un'altra curva è la Precision-Recall.
Rappresenta la ~~Precision~~ Precision in funzione della
Varianza di θ .

ROC: Vista per dataset binari
PR: Vista per dataset multipli.

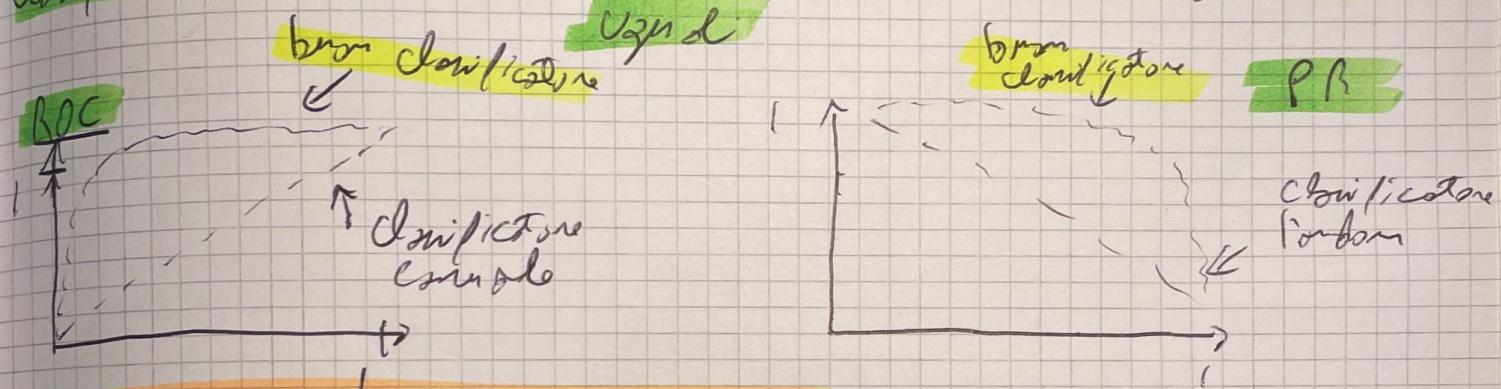
Curva ROC

Caratteristiche: TPR raggiunge 1 e FPR
non raggiunge 0.

Caso limite superiore: tutte classificate come
negative. $FPR=0$ $TPR=0$

Caso limite inferiore: tutte classificate come
positive. $FPR=1$ $TPR=1$

Classificatore random: TPR e FPR assumono valori



Area Under the Curve AUC

Minimizziamo l'area al di sotto della curva che
ha valore tra 0 e 1.

$AUC=1 \Rightarrow$ classificatore perfetto.

Verifica classificatore

Metrica hold-out

Primo percentile $X \Rightarrow 100-X$ per TEST e X per TR

~~Si~~ apprendiamo su TR, si applica su TE
e si misura l'accuracy.

Variante random sampling: Si ripete le volte, n
calcolo la media

Metodo k-fold cross-validation

- Fissato k si divide il dataset in $D_1, D_2 \dots D_k$ N items di m .
 - Alla i -esima iterazione con i tra 1 e k si usi D_i come Test e il resto come TR
 - Leave-one-out: Vengono fatte k ripetizioni di k fold con numeri di triple.
- Anche questo metodo potrebbe essere fruttuoso più volte per calcolare delle medie.