

Estrazione dati

- Altre attività:
- Data cleaning utili, come ~~preprocessing~~ (rimozione dati non necessarie)
 - Visualizzazione dati

Pattern di classe:

- Valori: veri per tutti i dati
 - Utile
 - Inutile
 - Congruibili
-
- Devono avere qualche caratteristica

Un diagramma di clustering è

- Descrittivo: identificare pattern comprensibili e descrivibili dei dati
- Preddittivo: creare un modello per predire i dati futuri.

I dati con cui si ha a che fare

- Grotti
- Eleganti dimensioni (molti attributi)
- Confini

Si usano strumenti avanzati per le tracce automatiche dei dati

Ottenerli è la principale analisi.

Dati \neq Componenti

Dopo l'analisi dei dati, veniamo a patti con i dati

Il Dato Numpy non è solo estremo. Tanti
dati e analizzati tutti, bisogna estremare
i dati BENE, quelli che servono.

Non riusciamo estremare dati inutili per
analizzarli,abbiamo messo a punto
che servono essere utili

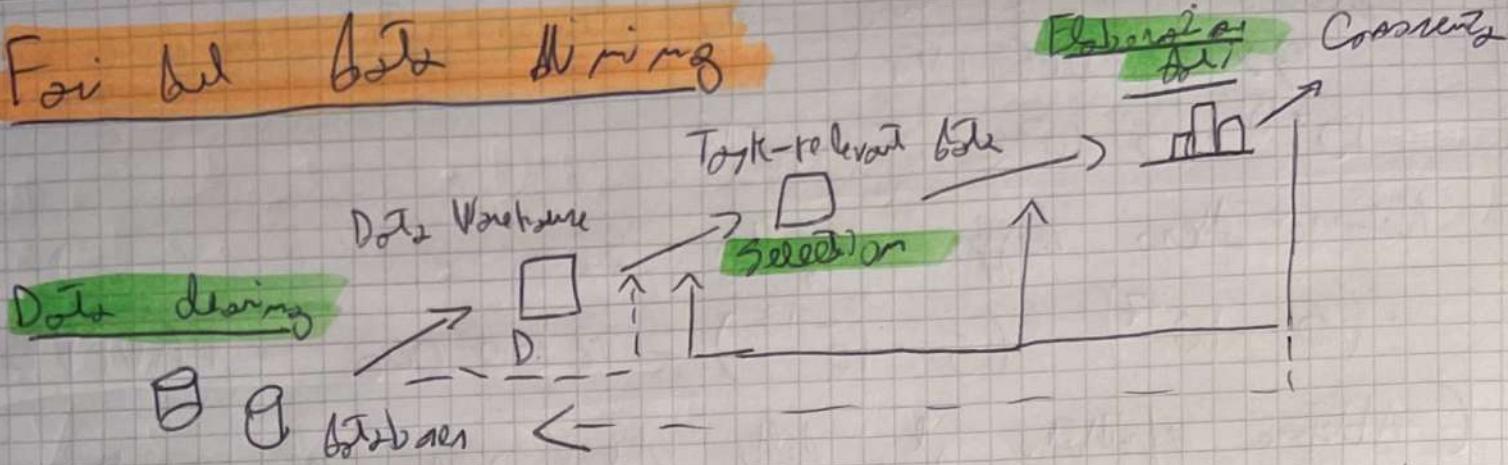
Avrò troppi dati se i confronti

Piaggio & Brambilla: se il numero di occorrenze
altro, nell'ipotesi di banchette, è rappresentabile
già alto del numero di filese fatti che si
riesce di trovare altrettante informazioni altramente
possibile essere considerate spazzatura

Dunque condiziona una proprietà sui dati abbiano
essere vicini che queste proprietà non vengano
oversata troppo volte nel caso in cui i dati
sono le robuste

Potremo dire che c'è un N_2 un lower
e Upper bound del numero di dati da
analizzare

Fai del best mining



Data troppo grandi \leftrightarrow Eseguo di applicazione di Boltzman

- 10^9 persone trascorre
- Ogni $\approx 1\%$ del tempo in hotel
- 10^7 persone in hotel al giorno
- ovvero 10^{10} persone in hotel in 1000 giorni
- 10^5 hotel, ognuno con 100 persone
- Quante coppie al giorno in un intero hotel in due giorni diversi?
- Prob. una persona vada in un hotel $\frac{1}{100}$
- P e Q vanno nello stesso hotel stessa giorno

$$\frac{1}{100} \cdot \frac{1}{100} \cdot \frac{1}{10^5} = \frac{1}{10^9} = 10^{-9} \rightarrow \text{probabilità 1000 persone in 2 hotel}$$

$$- \text{probabilità 1000 persone in 2 giorni diversi} \quad 10^{-9} \cdot 10^{-9} = 10^{-18}$$

$$\text{Coppie giornaliari} \quad \binom{1000}{2} = \frac{1000!}{2! \cdot (998)!} = 5 \cdot 10^{15}$$

$$\text{come persone distinte} \quad \frac{10^9 \cdot (10^9 - 1)}{2} \cong 5 \cdot 10^{17}$$

Calcolo delle probabilità

Evento: probabilità che avvenga o non avvenga per es.

Esperimento: Esce testa \rightarrow Evento: Esce testa

Variable aleatoria: quantità numerica che descrive il possibile esito di un esperimento

L'insieme delle ~~possibili~~ possibilità dei valori possibili della variable X è detto range $R(X)$

Range continuo \rightarrow Variable aleatoria continua
// Discreto \rightarrow // discreta

Allora ogni variable A è un valore a cui corrisponde una probabilità $P(A=a)$ \rightarrow indica quanta è la probabilità che $A=a$

A volte si dice $P(a)$

Distribuzione di probabilità $P(X)$: insieme delle probabilità che X assume bei certi valori (uno b. en.)

Distribuzione cumulativa: trovare le probabilità che X assume valori minori o uguali a x

$$F(x=x) = \sum_{x' \leq x} P(X=x')$$

Per una variabile continua non vi può parlare di distribuzione (i valori sono infiniti).

Si parla di densità di probabilità (L e R potrebbero essere $+\infty$)

Dato X e L, R estremi inferiori e superiori della range

n intere la funzione di densità $f(x)$.

La densità di probabilità in (a, b) con

$$L \leq a \leq b \leq R$$

$$P(a < X < b) = \int_a^b f(x) dx$$

Funzione cumulativa di probabilità: probabilità che X sia $\leq x$

$$F(x \leq x) = \int_L^x f(x) dx$$

Proprietà Probabilità

Nel discreto

$$P(X=x) \geq 0, \forall x \in R(X)$$

$$\sum_{x \in R(X)} P(X=x) = 1$$

Nel continuo

$$f(x) \geq 0 \quad \forall x \in R(X)$$

$$\int_L^R f(x) dx = 1$$

Normalizzazione

Le probabilità assumono valori tra 0 e 1 escluso.

0 → valore non possibile

1 → certo

Di solito si cerca lavoro più variati.
Si cercano modelli più variati alle stesse condizioni contemporaneamente

Probabilità di interazione

$P(X=x, Y=y)$ oppure $P(X=x \text{ and } Y=y)$
Ovvero probabilità che accadano entrambi così

Unione

$P(X=x \text{ or } Y=y)$ o uno dei due eventi
Come per la legge degli insiem

$$P(X=x \text{ or } Y=y) = P(X=x) + P(Y=y) - P(X=x, Y=y)$$

Distribuzione di prob. Condiz.

$P(X, Y)$: insieme probabilità che X e Y assumano certi valori contemporaneamente (in $R(X)$ e $R(Y)$ rispettivamente)

$P(x, y)$ prob. $\frac{1}{2}$

$$\cdot P(x, y) \geq 0$$

$$\cdot \sum_{x \in R(X)} \sum_{y \in R(Y)} P(x, y) = 1$$

Nel caso continuo \rightarrow densità di probabilità continua

$f(x, y)$

$$\cdot f(x, y) \geq 0$$

$$\cdot \int_x \int_y f(x, y) dx dy = 1$$

Distribuzione marginale

La distribuzione marginale fornisce informazioni anche sulle componenti di una singola variabile.

Distribuzione marginale \rightarrow distribuzione di una delle sue variabili soltanto e pertanto dalla distribuzione congiunta

Questo process è detto marginalizzazione

Nel discutere

$$P(X=x) = \sum_{Y \in R(Y)} P(X=x, Y=y)$$

$$P(Y=y) = \sum_{X \in R(X)} P(X=x, Y=y)$$

Nel continuo \rightarrow integrazione al posto di somma

Dato:

$$P(X=z) = \sum_{Y \in \mathbb{R} \dots \mathbb{R}} P(X=z, Y=y) \quad \text{ENNE}$$

La marginalizzazione si estende a 3 o più variabili

$$P(X_1=x_1, \dots, X_n=x_n) = \sum_{x_i \in R(X_i)} P(X_1=x_1, \dots, X_n=x_n)$$

Le variabili da marginalizzare possono anche 2 o più

post.

Probabilità Condizionale

Dati $X = x$ e $Y = y$ con y da t_2 in alto valore

La prob. dc $X=x$ b2x $Y=y$ è probabilità condizionale
e denotata com $P(X=x | Y=y)$

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

$$\cdot P(X=x | Y=y) \neq P(Y=y | X=x)$$

Si può generalizzare a n variabili

$$P(X_{k+1}=x_{k+1}, \dots, X_n=x_n | X_1=x_1, \dots, X_k=x_k) = \frac{P(X_1=x_1, \dots, X_n=x_n)}{P(X_1=x_1, \dots, X_k=x_k)}$$

Terzo di Bayes

- $P(X=x, Y=y) = P(Y=y) \times P(X=x | Y=y)$
- $P(Y=y, X=x) = P(X=x) \times P(Y=y | X=x)$
- $P(X=x, Y=y) = P(Y=y, X=x) \downarrow$
- $- P(Y=y | X=x) = \frac{P(Y=y) \times P(X=x | Y=y)}{P(X=x)}$

X e Y sono indipendenti se

$$P(X=x, Y=y) = P(X=x) \times P(Y=y)$$

Da cui $P(X=x | Y=y) = P(X=x)$

In genere X_1, \dots, X_n indipendenti se

$$P(X_1=x_1, \dots, X_n=x_n) = P(X_1=x_1) \times \dots \times P(X_n=x_n)$$

Combinazione variabili

Unendo operatori su variabili aleatorie si ottiene X nuova variabile aleatoria e il suo $R(X)$ che è dato dai valori ottenuti combinando in ogni modo possibile i valori delle n variabili secondo l'operatore.

Variabile somma $X = X_1 + X_2 + \dots + X_n$

Valore atteso o media

$$\mathbb{E}[X] \text{ è } \sum_{x \in R(x)} x \cdot p(X=x)$$

$\stackrel{n \text{ usi}}{\text{anche}}$
 N_x

Varianza

$$G_X^2 = \mathbb{E}(X - \mathbb{E}[X])^2 = \sum_{x \in R(x)} (x - \mathbb{E}[X])^2 p(X=x)$$

Varianza: media dei quadrati delle differenze tra i valori che può avere X e il valore atteso

- risulta la σ^2 dell'esperienza della variabile aleatoria X in base al valore medio
- valore probato di $G_X^2 \rightarrow G_X$ detta deviazione standard

Se X continua ha definizioni di media

e varianza analoghe

$$\mathbb{E}[X] = \int_{-\infty}^{\infty} x \cdot p(x) dx$$

$$G_X^2 = \int_{-\infty}^{\infty} (x - \mathbb{E}[X])^2 p(x) dx$$

Mefo e Varianzza di variabili

$$X = X_1 + \dots + X_n$$

$$\cdot E[X] = E[X_1] + \dots + E[X_n]$$

$$\cdot \sigma^2_X = \sigma^2_{X_1} + \dots + \sigma^2_{X_n}$$

Ma cosa è combinazione lineare $X = \alpha_1 X_1 + \dots + \alpha_n X_n$

$$\cdot E[X] = \alpha_1 E[X_1] + \dots + \alpha_n E[X_n]$$

$$\cdot \sigma^2_X = \cancel{\alpha_1^2 \sigma^2_{X_1} + \dots + \alpha_n^2 \sigma^2_{X_n}}$$

Entropia (tirofisi)

Così X è distribuzione di prob. $P(X)$, l'entropia è $H(P(X))$

$$H(P(X)) = - \sum_{x \in \text{dom}(X)} P(X=x) \log(P(X=x))$$

E' un valore positivo e il log è in base 2

L'entropia indica quanto è difficile prevedere il valore della variabile avendo la distribuzione $P(X)$

Entropia: minima (0) se tutta X è dist. di prob. $P(X)$ passa tutte le possibili con certezza il valore delle variabili

: Massima se tutti i valori della distribuzione sono equiprobabili

Se p rappresenta questa prob (uguale per tutti)

$$H(P(X)) = \log \frac{1}{p}$$

Entropia e Varianza misurano entrambi l'incertezza
del valore di una variabile

- Perciò:
- Entropia definita solo in base alle probabilità dei possibili valori e non in base ai valori stessi
 - Varianza invece dipende anche dai valori stessi

Entropia relativa

Date $P(X_0)$ e $Q(X_1)$ due distribuzioni di prob. associate a X_0 e X_1 , che hanno stesso range.

L'entropia di P rispetto a Q è

$$H(P||Q) = \sum_{x \in R(X)} P(X_0=x) \log \frac{P(X_0=x)}{Q(X_1=x)}$$

L'entropia relativa minima la somma delle entropie

$$\cdot H(P||Q) \neq H(Q||P)$$

$$\cdot J(P||Q) = H(P||Q) + H(Q||P) \text{ è detta entropia relativa tra le due distribuzioni}$$

Covarianza e matrice di covarianza

Nel caso di due var. aleatorie X e Y si definisce
la covarianza

$$\text{Cov}_{X,Y} = \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])] =$$

$$= \sum_{x \in R(X), y \in R(Y)} (x - \mathbb{E}[X])(y - \mathbb{E}[Y]) P(x,y)$$

nel caso continuo \rightarrow integrale

Date n variabili X_1, X_n costituiscono una
matrice simmetrica detta di covarianza.

Gli elementi sono le covarianze di tutte le possibili
coppie

$$\Sigma = \begin{pmatrix} \sigma_{11} & \sigma_{12} & \dots & \sigma_{1n} \\ \sigma_{21} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ \sigma_{n1} & \dots & \sigma_{nn} \end{pmatrix}$$

E' simmetrica perché

$$\sigma_{21} = \sigma_{12} \text{ ecc ecc.}$$

Usando covarianza e deviazioni standard delle

X e Y posso calcolare la correlazione di X e Y (di Pearson)

$$\rho_{X,Y} = \frac{\text{Cov}_{X,Y}}{\sigma_X \sigma_Y} \quad i valori vanno da -1 a 1$$

1) Ottengo 1 se $X = Y$

-1) Ottengo -1 se

Se $\rho_{X,Y} > 0$ sono correlate positivamente

Se $\rho_{X,Y} < 0$ sono correlate negativamente

Se $\rho_{X,Y} = 0$ sono indipendenti

Con cosa si correlazione?

Minimizza il grado di dipendenza lineare tra le 2 variabili

Quindi se ci è dipendenza lineare ovvero

Y approssima $aX + b$ le variabili sono correlate

$a > 0 \rightarrow P_{X,Y} > 0$ (posit. correlate)

$a < 0 \rightarrow P_{X,Y} < 0$ (negat. correlate)

Se le due var. X e Y sono indipendenti $\rightarrow P_{X,Y} = 0$

Ma non è tutto il viceversa, (potrebbe avere una correlazione a tipo quadratico o altro tipo)

Condizione di Spearman

• Confrontiamo i rank

Sono facendo la correlazione di Pearson tra i rank delle due variabili

Minimizza la correlazione tra rank dei valori delle due variabili

Rank: posizione nell'insieme dei valori ordinato in maniera crescente

Calcolare il rank

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
3	4	5	6	5	1	4	3	2	5

• Ordiniamo

$\frac{x_6}{1}$	$\frac{x_9}{2}$	$\frac{x_1}{3}$	$\frac{x_8}{3}$	$\frac{x_2}{5}$	$\frac{x_3}{5}$	$\frac{x_5}{5}$	$\frac{x_7}{5}$	$\frac{x_3}{5}$	$\frac{x_{10}}{5}$
-----------------	-----------------	-----------------	-----------------	-----------------	-----------------	-----------------	-----------------	-----------------	--------------------

Se gli x condividono la "posizione" di i allora le loro posizioni sono uguali

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
3,5	6,5	8,5	6,5	6,5	1	6,5	3,5	2	9,5

• x_1 e x_8 hanno prima 3 e 5 $\rightarrow (3+5)/2 = 3,5$

Dunque $\bar{x} \rightarrow$ rank R_x ottiene uno ciak di ogni valore il suo rank

La distribuzione di prob. di R_x è uguale a quella di X (escluso per la densità nel continuo)

Combinazione di Spearman:

$$P_{X,Y} = \frac{6 R_{X,Y}}{6 R_X R_Y}$$

Spearman misura la concordanza fra rank e non le variabili

Inoltre ne esiste una relazione monotonica tra le variabili ovvero ne esiste una f monotonica (non per forza lineare) che descrive Y in funzione di X

Spearman è in generale più applicabile a Pearson

Se la funzione che lega Y e X non è monotonica Spearman non riesce a ottenere questa relazione

o scindere in classi

Nelle prob.: fino a ora Pearson è più quello di Spearman (se quello di Pearson non ci ha dato info)

Matrice di confusione

come con le variabili date X_1, \dots, X_m

Continuiamo:

$$P = \begin{pmatrix} 1 & p_{12} & \dots & p_{1m} \\ \vdots & & & \vdots \\ p_{m1} & \dots & \dots & 1 \end{pmatrix}$$

elementi: combinazioni
tra tutte le
possibili copie

Distribuzione di Bernoulli

(2 esiti possibili)

Y può avere solo 2 valori: 0 e 1

Sia $P(Y=1)$ con prob p , allora $P(Y=0) = 1-p$

La distribuzione di Bernoulli di Y è

$$P(Y=y) = p^y (1-p)^{1-y} \quad \text{per } y = 0, 1$$

$$\mathbb{E}[Y] = p \quad \text{probabilità media}$$

$$\text{Var}[Y] = p(1-p)$$

Distribuzione binomiale

E' possibile vedere come una sequenza di esperimenti

Bernoulli indipendenti tra loro.

(il loro totale numero è indipendente dal probabile)

• M : esperimenti di Bernoulli (indipendenti)

• ogni esperimento ha prob p

Allora la distribuzione binomiale per Y sarà il numero

$$P(Y=y) = \binom{M}{y} p^y (1-p)^{M-y} \quad \text{per } y = 0, 1, \dots, M$$

$$\text{NB} \quad \binom{M}{y} = \frac{M!}{y!(M-y)!}$$

combinazioni
di y

$$\begin{aligned} P(C \cap T) &= P(C) \cdot P(T) \cdot P(C) = \\ &= (1-p) \cdot p \cdot (1-p) = p^1 (1-p)^{M-y} \end{aligned}$$

Molti e somme di variabili binarie

Si parla di somma come moltiplicazione di variabili binarie. Somma di molti e variazioni delle singole variabili è Bernoulli.

$Y_1, Y_2, Y_3, \dots, Y_n$ variabili di Bernoulli.

$$Y = Y_1 + \dots + Y_n$$

$$\mathbb{E}[Y] = \mathbb{E}[Y_1] + \dots + \mathbb{E}[Y_n] = np$$

$$\text{Var}^2 = \text{Var}^2_{Y_1} + \dots + \text{Var}^2_{Y_n} = np(1-p)$$

Distribuzione ipergeometrica

Uma e N palline $\rightarrow M$ rosse, $N-M$ bianche

Percorso in palline così (verso l'intera flotta)

Y : palline rosse estratte dall'urna

$$P(Y=y) = \frac{\binom{M}{y} \binom{N-M}{m-y}}{\binom{N}{m}}$$

con $y = A, A+1, \dots, B$

$$A = \max(0, m + N - n)$$

$$B = \min(m, n)$$

A e B determinano

i "confini" dei valori possibili di Y

Distribuzione Uniforme

Y ha la distribuzione uniforme i valori di Y sono

$$a, a+1, \dots, a+b-1 \quad \text{con } b > 1 \quad \text{e la}$$

probabilità che Y annui un dei b valori è

$$\frac{1}{b}$$

$$P(Y=y) = \frac{1}{b} \text{ per } y=a, \dots, a+b-1$$

media $E[Y] = a + \frac{b-1}{2}$

$$\text{Var}_Y = \frac{b^2 - 1}{12}$$

Distribuzione geometrica

E' simile alla binomiale

Considerare reg. di Bernoulli: trials indipendenti con prob di successo p

n NON è finito

Y è il numero di successi prima di un fallimento

Se $Y=y$: ho ottenuto y successi già ora e
un fallimento

La variabile Y ha una distribuzione geometrica:

$$P(Y=y) = (1-p)p^y \quad \text{con } y=0, 1, \dots$$

Distribuzione di Poisson

Ho un esperimento di Bernoulli ripetuto infinite volte in un certo tempo T e il numero medio di successi è $\lambda (> 0)$

Y è il numero di successi in T

E' simile alla binomiale ma invece di avere n ho il numero medio di successi nell'intervallo T

$$P(Y=y) = \frac{e^{-\lambda} \lambda^y}{y!} \quad \text{per } y=0, 1, \dots$$

media e varianza $\Rightarrow \lambda$

C'è una relazione tra la media e la varianza

La Poisson ci può definire un'approssimazione della binomiale se $n \rightarrow \infty$, $p \rightarrow 0$ e $np = \lambda$ costante.

$$n \rightarrow \infty, p \rightarrow 0, np = \lambda$$

- Queste condizioni su n e p sono molto frequenti.
- Ponendo queste, con buoni risultati, la Poisson è simile alla binomiale.
- La Poisson è già facile da usare dato che ha solo un parametro (λ) mentre la binomiale 2 (n e p).

Distribuzione multinomiale

È una generalizzazione della binomiale.

Abbiamo m Bernoulli Trials, con $M \geq 3$ OUT

biensi

La prob di avere i ($i=1..m$) è la stessa per tutti i trials ed è p_i .

Ho Y_1, Y_2, \dots, Y_m variabili dove Y_i indica il risultato quando i quanti volte è stato osservato i negli m trials.

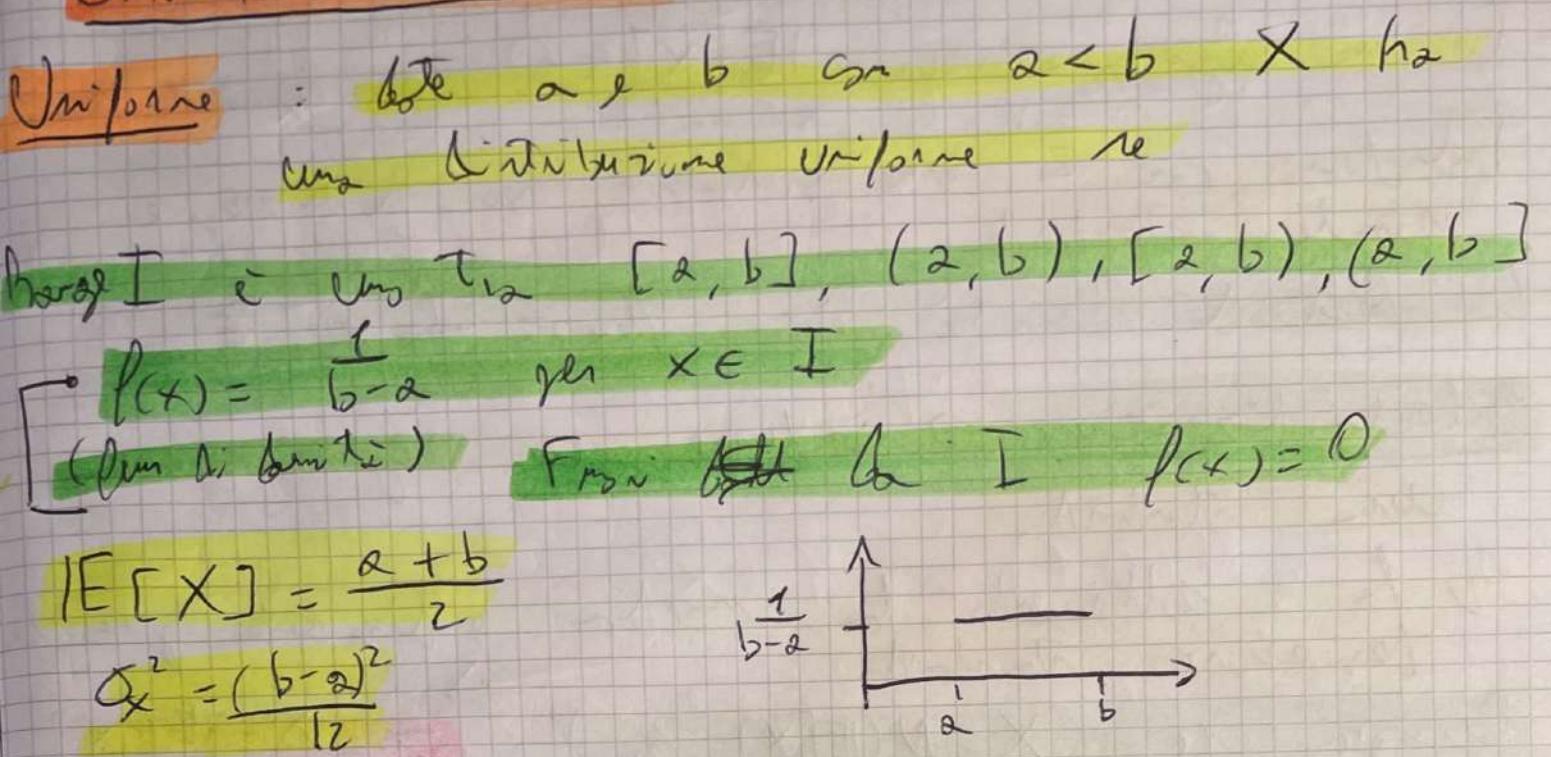
Dove Y_i è una variabile binomiale con

$$\text{media} = M p_i \quad \text{varianza} = M p_i (1 - p_i)$$

La prob che $Y_i = y_i$ ($i=1..M$) è

$$P(Y_1=y_1, \dots, Y_m=y_m) = \frac{m!}{\prod_{i=1}^m y_i!} \prod_{i=1}^m p_i^{y_i}$$

Distribuzione continua

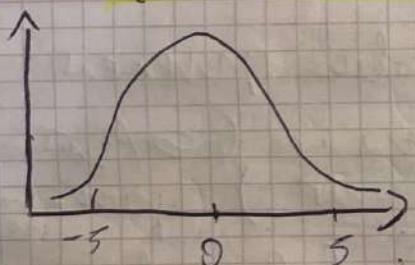


Normale (Gaussiana)

X ha una dist. Gaussiana se ha campo $(-\infty, +\infty)$ e $f(x)$ funzione densità è

$f(x) = \frac{1}{\sqrt{2\pi}\beta} e^{-\frac{(x-\alpha)^2}{2\beta^2}}$ $\text{Var}[X] = \beta^2$ se $\alpha = 0$

con α e β parametri della dist.
 $(-\infty < \alpha < +\infty, \beta > 0)$



Forma più comune $\rightarrow f(x) = \frac{1}{\sqrt{2\pi}\sigma_x} e^{-\frac{(x-\mu)^2}{2\sigma_x^2}}$

Una variabile aleatoria che ha la dist. normale è indicata con $N(E[X], \sigma_x^2)$

Scarsità Standard

E' una Gaussian con $\text{IE}[X] = 0$ e $\sigma_x^2 = 1$

Indichiamo con Z una variabile con distribuzione Normale Standard.

La sua densità è $p(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$

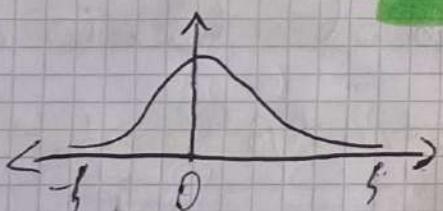
Standardizzazione \rightarrow Combinare una Gaussiana a una standard

E' fa: sottrarre la media e dividere per la deviazione standard. \rightarrow

Dunque se $X = N(\text{IE}[X], \sigma_x^2)$ $Z = \frac{X - \text{IE}[X]}{\sigma_x}$

Se x è valore orario di X

allora $z = \frac{x - \text{IE}[x]}{\sigma_x}$ è detto z-score



Applicazioni della normale / Gaussian Standard

1)

Se vogliamo confrontare due valori provenienti da due diverse distribuzioni Gaussiane (corrispondenti a risponditori su valori nella dist.) possiamo usare gli z-score invece di calcolare le probabilità di ottenere valori tali nell'unisono.

2)

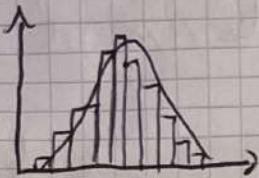
E' possibile usare per calcolare le probabilità di ottenere valori in una qualsiasi distribuzione

Trovare entro 1-2-3 deviazioni standard dalla med. (regola delle 1-2-3 dev standard)

Con questa regola si può approssimare probabilità di variabili con distribuzione simile alla normale

Esiste un rapporto tra normale e binomiale

Inoltre la binomiale si comporta come approssimazione della gauss.



Distribuzione esponenziale

Applicazione in biologia sopravvivenza

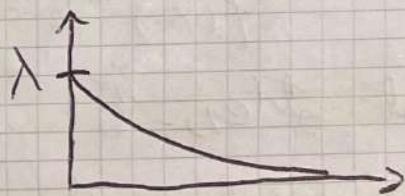
X con dist. esponenziale ha spettrum di valori

$$f(x) = \lambda e^{-\lambda x} \text{ con } x > 0$$

funzione cumulativa $\rightarrow F(x) = 1 - e^{-\lambda x}$

media (stessa) $E[X] = \frac{1}{\lambda}$

Varianza $\sigma^2 = \frac{1}{\lambda^2}$



Distribuzione Chi-quadro

Usata per testare ipotesi statistiche

$H_0: X_1, X_2, \dots, X_n$ n variabili stazio. con una dist. normale standard $N(0,1)$

Creiamo $X = X_1^2 + X_2^2 + \dots + X_n^2$ detta X^2 con n gradi di libertà

La funzione di densità è

$$f(x) = \begin{cases} 0 & \text{se } x < 0 \\ \frac{1}{\Gamma(\frac{m}{2})} \left(\frac{1}{2}\right)^{\frac{m}{2}} x^{\frac{m}{2}-1} e^{-\frac{1}{2}x} & \text{se } x \geq 0 \end{cases}$$

Dove $\Gamma(v) = \int_0^{+\infty} e^{-t} t^{v-1} dt$ (Funzione Gamma)

valore $\mu_{X_m} = M$

$$\text{Varianza} = 2M$$

grafico \rightarrow simmetrico attorno alla linea
di y anche da m.

Dati T-student

H₀ $Z = N(0,1)$ e $Y = X^2$ con m gradi di libertà

\rightarrow Z variabile aleatoria indipendente

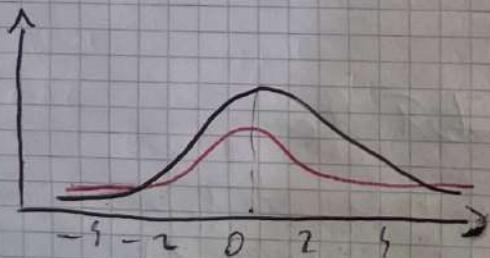
H₀ $X = \frac{Z}{\sqrt{m}}$ di distribuzione secondo la T-student con m gradi di libertà

La funzione di densità è

$$f(x) = \frac{\Gamma(\frac{m+1}{2})}{\Gamma(\frac{m}{2})} \frac{1}{\sqrt{m\pi}} \left(1 + \frac{x^2}{m}\right)^{-\frac{1}{2}(m+1)}$$

valore $\mu_{X_m} = 0$ se $m > 1$

$$\text{Varianza} = \frac{m}{m-1} \text{ se } m > 2$$



StD - Normal

grado $m = 1$

già m è alto già i
sopravvive alla messa

Dati multi variabili multivariante

E' un esempio di distribuzione di probabilità congiunta continua.

Si considera come una generalizzazione della distribuzione multivariata del caso di più variabili.

Ho:

- $\vec{X} = (X_1, \dots, X_m)$ vettore di m variabili
- $\vec{N} = (N_1, N_2, \dots, N_m)$ vettore delle m variabili
- Σ = matrice di covarianza
- $\vec{x} = (x_1, \dots, x_m)$ vettore di m valori con $x_1 \in R(X_1), \dots, x_m \in R(X_m)$

X_1, \dots, X_m hanno una

multivariata se la loro distribuzione di probabilità congiunta è

$$f(\vec{X} = \vec{x}) = \frac{1}{(2\pi)^{m/2} |\Sigma|^{1/2}} e^{-\frac{1}{2} (\vec{x} - \vec{\mu})^T \Sigma^{-1} (\vec{x} - \vec{\mu})}$$

$\nabla |\Sigma|$ determinante della matrice Σ di covarianza

MARKET BASKET Model

Sezione relativa alle trig di elementi:

- item (oggetti)
- basket (o transazione)

Ogni basket è un itemset (insieme di item)

N-basket >> M oggetti -> in un basket

Comunemente il basket potrebbe essere un DB

Il market è chiamato così perché un'elaborazione

regole trig si sostituisce dai singoli item (i singoli item nel basket) al mercato (market)

Trovare itemset frequenti ovvero insiemi di item che sono osservati spesso in molti basket

Frequenze: definite in 0 fatto solo a rapporto.

Il rapporto di un itemset è il numero di itemset basket in cui è presente

Se il rapporto ≥ 0 è detto frequente

Applicazioni

Cercare concetti condivisi \rightarrow item = parola basket = documenti
Trovare coppie di gruppi di parole che appaiono insieme in molti documenti e che quindi potrebbero legarsi tra loro

Programma: documenti che hanno molte parole in comune \rightarrow parole chiave

(Item = documenti
basket = frase)

Scelta del problema

D'obbligo la ricerca di itemset frequenti è fatta su DB esistenti.

Avere algoritmi efficienti è fondamentale, bisogna ottimizzare tempo e spazio.

Bisogna sapere bene anche la logica di supporto.

- alto \rightarrow pochi soluzioni interessanti

- basso \rightarrow Trope soluzioni

D'obbligo la soglia è l'10% del numero di basket

Regole di associazione

La ricerca di itemset frequenti è finalizzata alla costruzione di regole if-then delle regole di associazione.

La regola è un'implicazione $I \rightarrow j$ dove I è un itemset e j un item.

$I \rightarrow j$ indica che nei canali dove si osserva l'itemset I è probabile osservare j .

Confidenza di una regola

La probabilità dell'implicazione è quantificata dalla confidenza della regola.

$Supp(X) = \text{support}(X)$

$$\text{Conf}(I \rightarrow j) = \frac{Supp(I \cup \{j\})}{Supp(I)}$$

La confidenza rappresenta la percentuale dei canali in cui si osserva I e j insieme.

$Supp(I \cup \{j\})$ supporto della regola.

$Supp(I)$ è detta correlazione della regola, cioè quanto meno si può applicare la regola.

La confidenza è utile se il rapporto dell'influenza a $I \times (I \rightarrow j)$ è abbastanza alto.

Infatti se abbiamo $\text{Conf} = 100\%$ in un $I \times j$, la confidenza è poco interessante. Ma in realtà anche rapporto e Conf. altri non danno informazioni molto interessanti.

Ese: $\text{Pasta} \rightarrow \text{Salz}$ E' quasi ovvio, non ha molte info. (anche rapporto e Conf sono alti)

Invece $\text{Pomodoro} \rightarrow \text{Bitter}$ potrebbe essere interessante con rapporto e Conf alti.

Interesse di una regola

$\text{Int}(I \rightarrow j)$ definisce l'influenza di un insieme di oggetti su un oggetto.

$$\text{Int}(I \rightarrow j) = \text{Conf}(I \rightarrow j) - \frac{\text{Supp}(j)}{N}$$

N : numero basket

I non ha influenza su $j \rightarrow$ per centuale di basket va con I che j è circa uguale a per centuale di basket con j .

L'interesse può essere positivo o negativo. Ci indica quanto i loro i basket ~~sono~~ un'associazione.

Fatto posso \rightarrow pasta è banale, $\text{Int} = 0$

Pomodoro \rightarrow pasta non è banale

$\text{Lift} < 1$: più giù basso è lift e maggiore è influenza media di I su j ,

$\text{Lift} > 1$: più alta è lift e maggiore è influenza media di I su j

Meno attenzivo all'influenza di I su j

$$\text{lift}(I \rightarrow j) = N \times \frac{\text{Sup}(I \cup j)}{\text{Sup}(I) \times \text{Sup}(j)}$$

N : non bastet

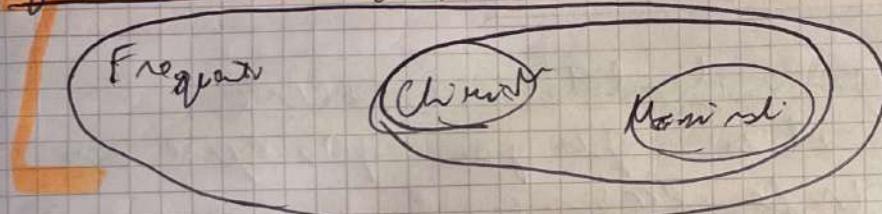
lift corrisponde al rapporto tra il supporto composto dalla regola e il supporto stesso. supponendo non ci sia alcuna dipendenza tra I e j

$\text{lift} < 1$: I influenza negativamente j
 $\text{lift} > 1$: positiva

Inicien Frequenti Monostri e chiavi

- Insieme Frequenti e chiavi se non c'è un ruolo nega - insieme con lo stesso supporto
- Insieme Freq. è una singola se non esiste un ruolo nega insieme Frequenti

Gli inizi Frequenti trovati sono anche Chiavi



Un algoritmo che riesce a trovare insieme Frequenti e Chiavi a trovare anche quelli chiavi e monostri sono

Algoritmi

Mitolo Nire: genera tutti gli itemset e calcola per ciascuno supporto e vedi se supera la soglia

Se m numero item, numero copie è $\binom{m}{2}$, numero copie è $\binom{m}{3}$.

Mitolo Nire non è particolarmente efficiente

Dato I , su ogni $S \subseteq I$, $\text{Sup}(I) \leq \text{Sup}(S)$

Dunque il supporto di un I non supera mai quello dei suoi sottoinsiemi (il supporto di un I è ~~mai~~ al massimo di un sottoinsieme di I)

E' detta anti-monotonicità del supporto

- Se un itemset I è frequentante allora ogni sottoinsieme di I è frequentante oppure
 - Se un itemset I non è frequentante, allora ~~esiste~~ nessun itemset che contiene I è frequentante
- Dato insieme A-Priori (non vuol il viceversa)

L'algoritmo A-Priori → bottom-up

- Setta set tascabili per cardinalità crescente
portando da singoli item per poi frequentare con le coppie poi con le triple ecc...
- Se itemset con cardinalità K non è frequentante allora non alcuna estensione è contenuta in itemset con $K+1$ item (perché quest'ultimo sicuramente non sarà frequentante)

Ciò ci permette di evitare l'esplorazione di molti candidati

Teniamo conto anche del fatto che itemset frequenti con cardinalità elevata (≥ 3) sono rari.

Algoritmo

- 1) Creare L_1 degli ~~item~~ frequenti
 - 2) Per valori k te creare (partendo da $K=1$)
 - a) Creare C_{k+1} degli item contatti b' cardinali $k+1$ partendo da L_k
 - b) Rimuovi da C_{k+1} gli item che hanno almeno un item con k item che non sono frequenti
 - c) Collega nippo a ogni item in C_{k+1}
 - d) Continua L_{k+1} formato dagli item di C_{k+1} frequenti
- Algoritmo termina quando i contatti finiscono

Generazione card dati

Le i tabella con $k+1$ colonne ($+1$ è il rapporto)

L'insieme C_{k+1} si ottiene mediante l'applicazione del join b' L_k

Per esempio le righe A e B di L_k imponiamo

- 1) I primi $k-1$ item di A e B sono uguali
- 2) k -esimo item di A \leftarrow Nessun item di B

Se i due righe non ci sono contatti con $k=2$

itemset	rapporto	L_k
b, c	1	
b, j	2	
b, m	1	
c, j	3	
c, m	2	
j, m	2	
m, p	2	

itemset	rapporto	L_k
//	//	
//	//	
/	/	
/	/	

C_{k+1}
b, c, j
b, c, m
b, j, m
c, j, m

Eseguo confronto su slide (risposte frequenti: 3E-55)

Utilizzo delle memoie

Dobbiamo fare i conti con la gestione delle memoie
Programma attivo: calcoli del rapporto
Dati in basket e item in RAM
Obiettivo: avere solo in RAM le tabelline con i
supporti degli item (singoli)

Con una Hash map associamo ad ogni item un intero
identificativo

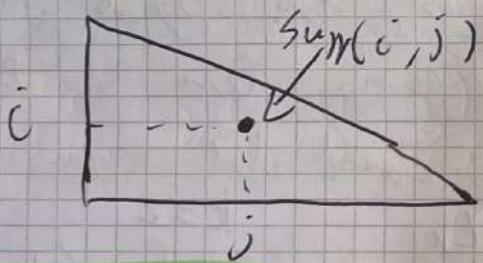
L'hash è più rappresentazione come l'indirizzo dell'item nel supporto

Per calcolare il rapporto delle copie ci sono due tecniche

- metodi matrice triangolare
- metodi delle triple

Matrice triangolare

Per ogni coppia di item con $i < j$ con $i < j$
memorizziamo con il rapporto della copia (i, j)



Nella pratica viene usata un array (invece di una
matrice $M \times M$ dove metti e imutilizzate)

$Sup(i, j)$ sta nella posizione k .

$$k = (c-1)M + k - (c-1)(M - \frac{1}{2}) + j - 1$$

Possiamo avere le coppie $(1,2)$ $(1,3) \dots (1,n)$ e poi
 $(2,3)$ $(2,4) \dots (2,n)$ e così via

Metodo delle triple

Il reporto \rightarrow di (i,j) è memorizzato come triple
 (i,j,m)

Le triple sono memorizzate in un basket così
che la coppia (i,j) è la chiave e m il
corrispondente valore.

La scelta del metodo dipende dal numero di copie
che appaiono in qualche basket

- Se almeno $\frac{1}{3}$ delle $\binom{M}{2}$ copie di interi sono presenti
nel basket scriverà la matrice triangolare nel
caso il metodo delle triple

Per iniziare con 3 o più elementi n . Usare le triple
e basket (è più conveniente)

Triple vs matrice Triangolare

Triple: memorizza solo copie che appaiono in qualche
basket

Matrice triangolare: richiede un solo basket da memorizzare
ognes il reporto.

Il metodo delle triple memorizza: i, j e m

Algorithmi di Park CHEN e YN PCY

• Prima fase: contano supporto item e leggono copie parziali
item in ogni basket

Le copie delle aree vengonoate in bucket U_{ijkl}
funzione hash

Se supporto bucket $< n_{min} \rightarrow$ rimuovi copia di item
più frequente (bitwise)

Per determinare se un bucket è frequente usi un bit $(1 \ 0 \ 0) \rightarrow$ bitmap

Il primo delle copie c_{ijkl} sarà formato da copie c_{ij}

• i, j item frequenti

• coppia (i, j) cade in un bucket frequente

$H(i, j) = x \rightarrow B_1 (i^1, j^1, 1)$

B_2

$B_3 (i, j, 2) (i^{11}, j^{11}, 1) \ S=3$

Quindi se B

questo algoritmo: generare meno copie codificate

Algorithmi Multistage

Sulle copie (i, j) che cadono in un bucket frequente effettua un ulteriore raggruppamento in bucket con una 2a fnc hash

E si può applicare il process più volte

Algorithmi multihash

Si usano 2 o più fnc hash in parallelo nello stesso step.

Le copie codificate sono quelle che cadono in bucket frequenti sulla base di ogni funzione

(le copie non cadono in Bitmap 1 e 2 in bucket frequenti)

Algoritmi Randomizati

Applicare l'Algoritmo sugli itemset in modo tale che la ricerca di itemset frequenti sia applicata su un sottogruppo random di basket invece di tutto il dataset.

Algoritmo SON

Divide il dataset in chunk

$S = \text{numero minimo}$ $p = \text{percentuale di basket in ogni chunk}$

- 1) Su ogni chunk applica Algor. (o una sua ottimizzazione). Il numero minimo è ridotto a $p \times S$
- 2) Considera l'unione di tutti gli ~~elementi~~ gli itemset che sono risultati frequenti in uno o più chunk
- 3) Per ogni itemset combattuto calcola il rapporto nel dataset iniziale

SON è molto buono in architettura distribuita

TOIVONEN

frontiera negativa: inviare gli itemset che non sono frequenti nel sottogruppo S del dataset D)

ma i cui sottoinsiemi immediati (ottenuti togliendo un solo item) sono frequenti in S

Ese. $\{A, B, C\}$ non è in S ma $\{A, B\}$, $\{A, C\}$ e $\{B, C\}$ lo sono perché $\{A, B, C\}$ è parte della frontiera negativa di S .

Poss Toivonen

- 1) Considera S e D forniti da p bank.
Se rapporto minore di D
la lista S sarà $\delta \times p \times S$ dove δ è
tra 0 e 1
- 2) Calcola i tempi frequenti in S e quelli in frontiera
negativa
- 3) Calcola rapporto degli itemset trovati in 2)
 - a) Se nessun itemset della frontiera negativa è
frequente in D bisceguo tutti gli item che
sono risultati frequenti in S
 - b) Se c'è o gli itemset della frontiera negativa
sono frequenti in D allora si ripete l'algoritmo
in un modo congiunto S_2

Perché ripetere Toivonen? (nel punto 3b)

I punti presentati
sono corretti; perche potrebbe esserci super-inver-
sione (I (I è nella frontiera negativa di S) che
non sono frequenti in S ma in D).

Dunque la rigetta perché dall'algoritmo sarebbe
stata riconosciuta con un altro congiunto

5: determina il rapporto minimo richiesto nel S .

Determina anche la percentuale di successo

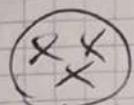
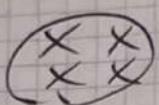
6 buono \rightarrow memoria usata maggiormente ma l'algoritmo
finisce in meno passi.

6 cattivo \rightarrow memoria più grande ma usata meno

Clustering

Processo che dato un insieme di oggetti li riunisce in gruppi detti cluster segnando una posizione di "distanza" tra gli oggetti.

Distanza corta = oggetti poco simili



$\times \times \leftarrow$ Oggetti con caratteristiche simili

Il clustering è applicato in diversi campi

Il clustering è un processo di unsupervised learning infatti non serve per dividere un insieme di dati in classi senza alcuna conoscenza di quale è quella classe le classi (e etichette)

Invece la classificazione è un processo di supervised learning

Si parte da dei dati etichettati, come la classe di appartenenza della training set

Si ottiene un classificatore partendo dal training set e imparando le regole di associazione tra gli attributi dei dati e la loro corrispondente classe (esempio: poiché nella base di addestramento l'etichetta è già data) *

Ottimizzato il classificatore può poter stabilire la classe di un nuovo dato senza classe vera e propria (ma esplicitamente) ***

* Si mette

• In pratica diamo un oggetto con la sua etichetta e il classificatore rialza le caratteristiche dell'oggetto in modo da capire già o meno le sue caratteristiche che lo sono di un oggetto con cui era confrontato

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)

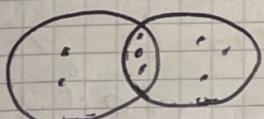
Negli spazi non Euclidei il centroide non è definito ma si definisce il meboide (1-mediana) ovvero un punto che minimizza la distanza media delle distanze dagli altri punti dell'insieme.

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

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



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

Variabile 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

Operazioni: Cancellazione | Inserimento | Sostituzione

$$A = abcde$$

1) cancella b

$$B = acdefg$$

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

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

Distanza di Hamming

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

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

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

Numeri 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) dei cluster iniziali e poi i punti vengono via via assegnati al cluster corretto. 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

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

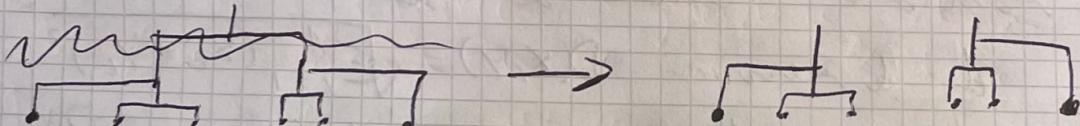
Non si può evitare di considerare i centroidi

Dunque l'algoritmo come notazione di visualizzazione
considera le distanze dei centroidi

Dunque prende 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
che consente di sapere quando sono stati costituiti i cluster

Tagliando il dendogramma (al suo tono) i nodi obiettivi che fanno parte di uno stesso cluster vengono tagliati.



Quindi potrete controllare tutto il dendrogramma per poi "tagliare" in base ai vostri criteri.

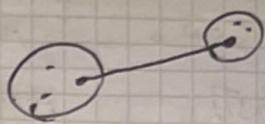
Criteri di terminazione

- Se sappiamo già quanti cluster ha l'oggetto:
fa i primi k sono i primi k punti
- Celi tutto l'embogramma (utile in biologia)
- Trovare l'algoritmo nel momento in cui una funzione globale sui cluster è adeguata
(es: la distanza fra centroidi dei punti del cluster coincide troppo)

Altre misure di distanza tra cluster

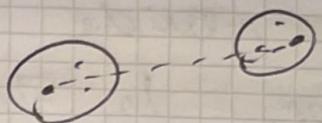
• Single link

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



• 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) Controlla i 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 aggiorna le
posizioni dei centroidi
- 5) Ripetere tutti i punti al cluster col centroide
più vicino (ci sono punti che abbondano i cluster
per subire in altri)
- 6) Ripeti 1 e 5 fino a quando non ci sono
più spostamenti oppure fino a quando una
funzione obiettivo si mantenga 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 criteri.

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

P_1, \dots, P_k sono i dati
 c_k i criteri.

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 qualsiasi dei criteri.

Valutare l'ideale: facendo riferimento alla distanza media fra i punti e i criteri.

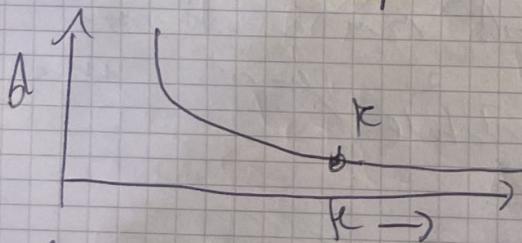
Se il clustering è di alta qualità = di basso costo.

Scelta valore di K

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

La funzione prima è brutta e poi si ottengono.

Valore ideale: K quello in cui la media iniziale varia poco.



La riduzione superficiale eseguita nel effettuare le mean più volte per trovare valori di K .

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 direttamente 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 solo 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.

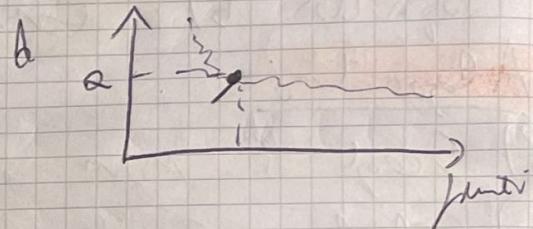
Fatto il pa Min Pts si cerca un valore ottimale
per ϵ .

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

E' ordinata così la maggior parte minore.

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 fissa 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 differenti

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 situazione.

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

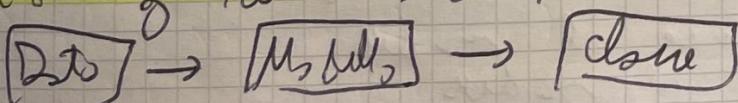
$S_i < 0$ è in un
cluster sbagliato

$S_i = 0$ 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 un attributo A non ancora
scelto e si crea una partizione S degli
elementi sulla base dei valori di A (splitting)

3) Gli attributi figli del nodo tengono bene i
valori di A

4) Per ogni figlio 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 potrei
scendere appena tecnicamente i suoi predecessori
in X

3) Se X è ingesso ma non ci sono attributi da
poter scendere 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. Qui c'è insomma senso della logica una distribuzione di probabilità più invecchiata più 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 voluta nella base di P

Splitting

a) Categorie att.

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

b) attributi continuo

- Definisci δ soglia e partition $S_1 \cup X$ in S_1 e S_2
- a secondi che $A \geq \delta \Rightarrow A < \delta$
- S_2 soglia è quella in moto che ogni partizione
- Aggiungi $\delta \times 2$ nodi finali per S_1 e S_2

c) Attributi binari

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

Costruzione albero decisivo

La costruzione si fa dall'ordine con cui N ricalcano gli attributi.

Obiettivo:

• Scoprire l'albero più semplice e capace possibile.

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 più facile a delle foglie

Per finire a 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 tante foglie di t quanti sono i valori di A
- Applicare ricorsivamente i passi precedenti sui nodi inner considerando i contenuti 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 continuo negli attributi you fare lo splitting basato negli un valore di soglia.

Se l'attributo è continuo i diversi colture già valori di goodness per l'attributo: uno per ogni possibile valore della soglia.

E 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 (x) - Entropia c'è

S_x è un sottoinsieme abbinato a classi $P \in N$

S_x di P contiene p elementi, b. N_m .

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

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

Entropia media di S_x rispetto ad A

Misura presetta delle entropie dei singoli S_i (parte della partizione di S_x)

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

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

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

$$\text{come } \text{GainRatio}(A) = \frac{\text{Gain}(A)}{\text{Spltinfo}(A)}$$

Lo splitratio è un "moltiplicatore" del gain

NB: Spltinfo è portativo

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 tiglie 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 delle frequenze delle classi in Sx

Gini index misura la prob che i sia \neq j

Dobbiamo che sia gini basso possibile

a) Possibilità di raffigurare una osservazione come classe i in Sx

b) Prob. regolare $j \neq i$. Vede la distribuzione nel modo

a) P_i

b) $1 - P_i$

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
atteso prima di fare lo splitting

2) Calcolo $E_p'(T)$ dopo lo splitting

3) Se $E_p'(T) > E_p(T)$

2) Sostituisci T con una foglia

Noti che le tickette sono quelle 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

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

$\# \text{ tuple in } S_x$

c'_i : dichetto in maggioranza in S_x

Cost - Complexity Pruning CART

Conservare $T_0 \dots T_m$ come $T_0 \dots l$ altri, decidendo
che T_m quello con la minima cost.

T_i è ottenuto da T_{i-1} rimuovendo un sottoalbero
da T_{i-1} e sostituendolo con una foglia così
che dichetto e quella di maggioranza che c'è
nel sottoalbero

Se $E(T_i)$ è l'cost 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
di una foglia

Le regole sono mutuamente esclusive ed esaurienti
e $t_1 \wedge t_2 \dots \wedge t_n$ lungo un cammino non
coincidente in AND

IF test 1 AND Test 2 THEN

Stabilire le quanti di una regola per una classe C.

Coverage: numero relativo degli esempi della regola.

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

Una buona regola deve avere un coverage non a caso.

Eredità positivi: tregole classe C coperte dalla regola.

Eredità negativi: tregole NON C coperte dalla regola.

E' possibile avere ribattere nelle regole.

Dobbiamo trovare il giusto equilibrio tra le regole nella classe C che forniscono gli eredità positivi.

Algoritmi di Covering: FOIL, AQ, CN2, RIPPER

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

Si mettono tutte le tregole del TR.

Regole separate una per volta.
Greedy basato sulla qualità. Usando un criterio della valutazione.

Ottendo una regola e ripetuta ogni volta.

Processo ripetuto finché non riesce tutto le tregole.

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

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

IF TRUE THEN C

Iniziamo ad aggiungere condizioni che aumentano C e accrescendo il numero di classi il più possibile.

IF true THEN $C = x$

Copri 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 altri gli esempi positivi.

Algoritmi Fois e Rizet

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

L₂ misura la qualità di Fois gain

• pos : esempi positivi • neg : negativi

• pos' : dopo sostituzione 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: N₂ puntini che negativi sono coperti dalla regola. I negativi definiti come gli esempi falsi della regola.

Classificatori generativi

Probabilità è etichetta y in mobilità (zona blu su un modello probabilistico sono basati i teoremi di Bayes)

Theoremi 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 alta

Prob 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 dataset.

$P(X | H_c)$ 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 dal 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 $P(X|C_i)$ usare $\log P(X|C_i)$

Nella moltitudine 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 .

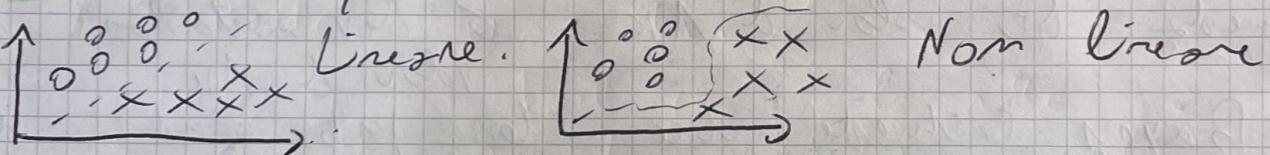
Sono tre classificazioni più comuni, sono molto robuste.
Il calcolo del valore della funzione decisione è facile.

Riassunto una costruzione 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.

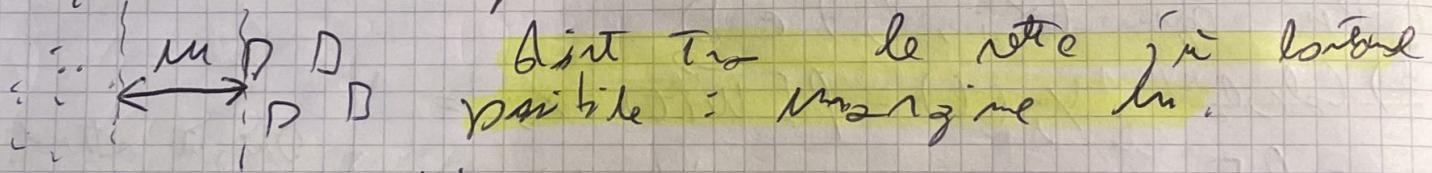
Dati linearmente separabili → trova iperplane migliore

Dati non linearmente separabili:

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

- In questo modo spazio l'SVM trova il piano separatore
 - Un piano a due dimensioni occupa un volume, i dati
 - i dati sono in dimensione
 - Esiste sempre uno spazio in cui trovare il separatore
- L'iperplane separatrice ottimale è trovata usando vettori di supporto (tuple di frontiera) e margini.

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



L'iperplane ottimale è quello che massimizza m , quindi faccio il gioco possibile l'uno.

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

Ottieni nel training set n punti $\vec{x}_1 \dots \vec{x}_n$

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

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

Equazione iperplane separatrice

$$b = bias$$

w vettore ric.

hors i dati normalizzati sono 0 e 1

Eq del 2 iperbole che definiscono i margini

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

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 ai margini sono negative

Se $y_i = 1 \rightarrow \vec{w} \cdot \vec{x}_i - b \geq 1$ ← H_1 o sopra H_2

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

Quindi dobbiamo risolvere

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

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

Questa 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 $\epsilon_1 \dots \epsilon_m$ dette variabili stoc

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

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 finito che soddisfa $y_i(\vec{w} \cdot \vec{x}_i - b) > 1 - \epsilon_i$

Obiettivo: avere ϵ_i piccoli

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

Il problema diventa

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

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'ipersfera separatrice

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 (lora) 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

$$y = \vec{w} \cdot \varphi(\vec{x}) - b \quad (\text{e' ora re separabile})$$

i dati non sono lineariamente
separabili)

Nuovo problema

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

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 dispendioso in quanto
ogni k il mapping.
Pertanto 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: $e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$

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 abbastanza lineari.

Infatti la complessità dipende solo dai numeri di training 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 dati.

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 abbattendo le dimensioni molto rapidamente.

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

KNN

Finito K , prendi le tighe già vicine alle tighe x da classificare.

Analogo a x la classe più vicinanza tra le K tighe.

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 di risposta.

Metodi di regressione: lineare, non lineare, generalizzati, Poisson, log-lineare, others di regressione

Regressione lineare semplice

y di 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 fit

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 ~~tra~~ le stime (valori 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

$$\begin{aligned} \text{Eg: } Y &= X_0 + W_1 X + W_2 X^2 + W_3 X^3 \quad \text{è convet} \\ \text{in } Y &= W_0 + X + W_1 X_2 + W_3 X_3 \quad \text{con } X_2 = X^2 \\ &\quad \quad \quad X_3 = X^3 \end{aligned}$$

In generale è facile trasformare in lineari funzioni polinomiali e funzioni potenza. Alcune funzioni, tuttavia, inerentemente non lineari, 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 (caso questo random con ripetizione)

2) Addestra M_i su T_i

3) Combinare i risultati dei modelli

• Regressione: sia la media • classificazione: la media dei maggioranza

Random forest

Combina i risultati di diversi alberi decisionali
verbale & tecnica bootstraping

Il Random forest adatta ogni albero su un
bootstrap sample (albero di un attributo)

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

Gli riduiscono la
correlazione tra alberi

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

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 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.

Accuraccia
Percentuale corretti
classificati bene.

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

$$19 = 5 + 3 + 11$$

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

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

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.

A questo uso la regresione per calcolare uno
score a una mail e se supera la θ
la classifica SPAM o se non lo supera la
classifica a 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.

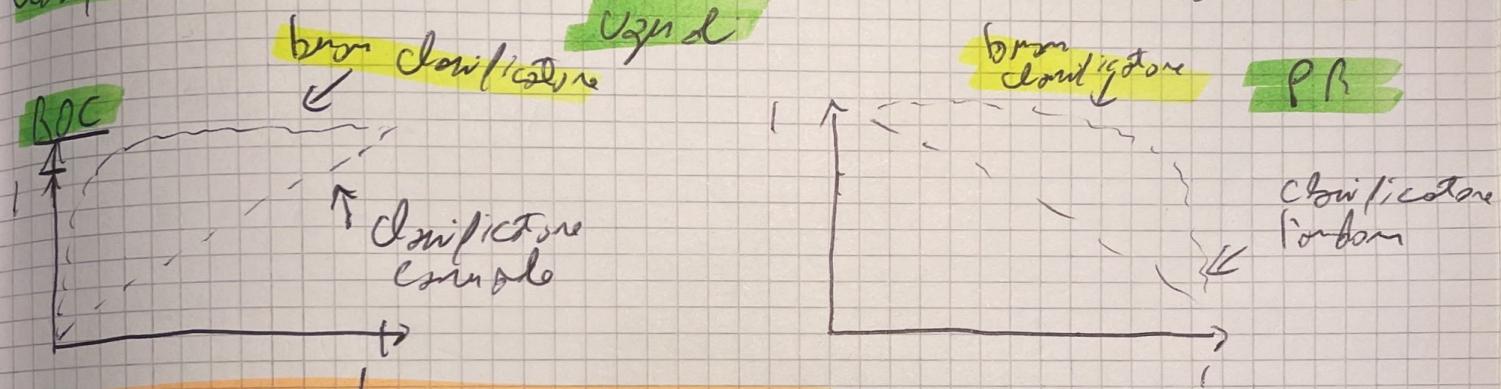
Curva ROC

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 valori tra 0 e 1.

$AUC=1 \Rightarrow$ classificatore perfetto.

Velocità classificatore

Metrica half-cut

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

~~è~~ si applica sul TR, si applica sul 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.

Metodo k-fold cross-validation

- Finito k si divide il dataset in $D_1, D_2 \dots D_k$ di items di m .
- Alla i -esima iterazione con i tra 1 e k si usa D_i come test e il resto come T_R
- Leave-one-out: Vengono fatti k i passi al numero di tuple. Anche questo metodo potrebbe essere implementato giusto per calcolo delle medie.

Sistemi di Raccomandazione

Utenti che cercano di prendere preferenze di un utente sulla base delle preferenze espresse in precedenza. Esempio: suggerire articoli in base agli articoli letti in precedenza.

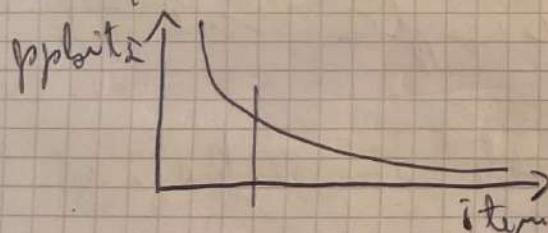
Esistono 2 tipi principali di raccomandazione:

- Sistemi content-based: effettuano raggiamenti basati sugli items
- Sistemi collaborativi filtering: suggeriscono sullo base delle similità tra utenti e items. Suggeriscono ad un utente \times items preferiti da altri utenti simili a \times

In sostanza i content-based sono basati sugli items mentre i secondi sono basati su similità tra utenti.

Fenomeno Long Tail

Oltre 100 milioni di item dal più popolare a quello di item in funzione della popolarità



I item che non vengono raccomandati non ottengono la lunga sba

Sviluppo:

- Per funzionare bene un utente & raccomandazione deve avere dati.
 - Più dati abbiamo più è scorsa la prefezione.
 - I dati di raccomandazione possono essere utili anche per propagarla
 - Il utente di raccomandazione rientra in un insieme di preferenze conosciute espresse da Utenti per degli items.
 - Essere rappresentato dalla matrice di similitudine M .
- Righe: Utenti
Colonne: items

$M[i, j]$ potrebbe essere booleano se all'utente i piace l'item j oppure un valore numerico (0.1-0.9) che indica il grado di preferenza di i per j.

La matrice potrebbe avere entry nulle che indicano che non si conosce la preferenza di i per j.
L'obiettivo è prevedere i valori non noti di M.

Costruire la matrice di utilità

- Approccio esplicito: chiedere valutazione / percezione all'utente
- Approccio implicito: apprendere la valutazione dal comportamento dell'utente

Sistemi Content-based

Idea: suggerire all'utente U oggetti simili a quelli già acquistati da U.
Il sistema è basato sulle proprietà dell'oggetto.

A ogni ~~oggetto~~ item i è associato un vettore di elementi. Ogni elemento è una proprietà.

Oggetti simili hanno vettori simili.

Bisogna costruire il profilo anche di U per l'item I.

A U per l'item I si associa un vettore di n elementi.

Questi elementi rappresentano il grado medio di preferenza espresso dall'utente per gli item contenenti quella proprietà.

La similarità tra il profilo di U e quello di I rappresenta la preferenza predetta da U su I .

Schemi generali

- 1) tra le proprietà conosciute per gli oggetti che U ha già valutato si escludono p_1, p_2, \dots, p_k proprietà.
- 2) Per ogni I valutato da U costruire il suo profilo ovvero un vettore di k elementi dove $c_j = 1$ se l'item ha la proprietà p_j e 0 se no.
- 3) Costruire profilo di U cioè vettore di k elementi dove v_j è il grado medio di preferenza di U per gli item valutati da U contenenti la proprietà p_j .
- 4) Costruire profilo di item I non valutato da U .
- 5) Calcola similarità tra profili di U e di I , per predire il grado di preferenza di U per I .

Profili di un item descrive quali proprietà contiene un item.

Le proprietà rappresentate

category, tag o parole chiave da associare a item

Nei documenti si fa text mining

$$TF_{ij} = \frac{f_{ij}}{\max_k p_k} \quad IDF_i = \log \frac{N}{n_i} \quad TF \cdot IDF_{ij} = TF_{ij} \times IDF_i$$

N : documenti, n_i : doc i che hanno parola i.
 f_{ij} : frequenza parola i nel doc j.

Profili Utente

Punto 6 fatto valutazioni già effettuate su altri item
è perciò "unire" le valutazioni che riguardano item
con lo stesso profilo

Funzione aggregazione media: media valutazioni

Esempio: ha valutato Batman 2, Star Wars 10 e
CAPTAIN AMERICA 8. Il punto su cui è valutato Comeby, gli altri
2 N.

Profili utente per Comeby $\Rightarrow \frac{10+8}{2} = 9$
Facendo così per ogni genere dei film
e altresì per utente (profili utente)

Similitudine tra profili

Viamo a misurare similitudine

Dati 2 vettori di k elementi: il profilo dell'utente
e il profilo dell'item.

$$\text{cosim}(\vec{U}, \vec{V}) = \frac{\vec{U} \cdot \vec{V}}{\|\vec{U}\| \cdot \|\vec{V}\|} = \frac{\sum_{i=1}^k u_i v_i}{\sqrt{\sum_{i=1}^k u_i^2} \sqrt{\sum_{i=1}^k v_i^2}}$$

Tiene conto della direzione

dei vettori e non delle distanze euclideanhe

I valori vanno da -1 (antiparalleli) a 1 (paralleli)

Ottieni i valori da -1 a 1 (nel nostro caso da
0 a 1 nei film con voti da 0 a 10) di modo
che il range è aggiunto come -0.

Esempio: da 2 a 10: ottieni 0,6 per la
similitudine dell'utente con l'item

Punto 6: bellezza = \circ cerca nuovo utente con
profili sovraccarico (ha meno riga
nella matrice di utilità e vota)

• Se ci sono profili
nuovi?

• Cosa significa item?

Caso: facile, non richiede
interazione con altri
stati,

Sistema Collaborativo Filtering

Idee: gli oggetti da suggerire a U sono quelli voluti meglio da utenti simili a U

Il filtro suggerisce a U item già visti a utenti simili a U.

E' più incentrato sul comportamento utenti

Differenze rispetto al content-based

- Profil item → colonne matrice utilità

- Profil utenti → righe matrice utilità

Schemi generali (base in similitudine utenti)

1) individua N utenti simili a U che hanno voluto I

2) calcola media per rating dati dagli N utenti su I. I valori sono negli stessi ambienti tra cui siamo degli N utenti su I

3) Il valore è il rating predetto per U su I

Similitudine tra utenti

Il profilo dell'utente è rappresentato dalla sua riga nella matrice di utilità.

N.B.: valore mancante \Rightarrow 0

Uscita la distanza del coseno per le distanze tra utenti.

Dobbiamo però fare attenzione alla differenza tra 0 per mancanza dati e 0 per valutazione bassa

Dobbiamo centrare i valori rispetto a 0.

Rating bassi \Rightarrow valori negativi

Rating alti \Rightarrow valori positivi

E' possibile sottrarre a ogni valore di rating
conosciuto la media dei rating associati dall'utente
a quei item. (dove ci sono rating \Rightarrow ~~utente~~ in media)

Abbiamo così
possiamo prendere il valore per un item da
un utente.

- Prendo N elementi/profilo più vicini (distanza coseno)
- Calcolo la media pesata dei rating null'item
dagli N profili (NB: il peso dipende dalla distanza)
- Il valore calcolato è il valore preetto.

Schemi alternativi (basato su similarità item)

- 1) Individua N item più vicini all'item I su cui l'utente ha effettuato il rating
 - 2) Calcola media pesata dei rating dati dall'utente sugli N item.
I geri sono dati da rese di similarità tra gli N item ed I
 - 3) Il valore ottenuto è il rating preetto per I
- Si le due schemi generano una parte
della similarità tra item.

Profilo item = colonne matrice utility

La similarità si potrebbe calcolare ancora con
la dist del coseno.

Confronto

Schemi basati su similarità item:

- più informativo e affidabile (ma più lento)

Schemi basati su similarità utenti

- più efficiente se vogliamo prendere i rating dell'utente U

Pregi del collaborazione filtering

- lavora con tutti gli item anche se non appartengono al progetto
- Non serve una relazione di progetto/pertinente degli item

Difetti:

- impossibile creare predizione per nuovi utenti o nuovi schema basato su variabilità tra item
- impossibile creare predizione riguardante nuovi item con lo schema che non hanno degli utenti

Nella pratica

I sistemi di raccomandazione sono degli ibridi tra content-based e collaborazione filtering.

Singular Value Decomposition SVD

I sistemi di classificazione potrebbero avere un'attuale di utilità.

Dobbiamo far fronte all'efficienza e al problema della dimensionalità

Dobbiamo abituare tecniche di dimensionality reduction che permette di lavorare con vettori e matrici più piccole

Una tecnica utile per ridurre la dimensionalità è il clustering

Applicando gli item id clustering possiamo trovare le colonne degli item con solo colonne che hanno i valori medi (questo si chiama cluster)

Praticamente utile ma 2 volte inefficiente se
ci sono tanti dati

Questo giudizio richiede di fare clustering su
tanti dati. Esempio su slide

Decomposizione di matrici

Potremmo trovare un insieme "piccoli" di feature
di item o utenti biammati.

Queste feature sono rappresentative di intere
categorie o gruppi di feature

Dato l'insieme di feature più importanti di tutte
ci viene un'idea: usare tecniche di
decomposizione della matrice che portano a 2 matrici
esprimendo come prodotto di 2 matrici
 $m \times n \Rightarrow m \times r + r \times n$

In altri algoritmi $m \times n \Rightarrow m \times r + r \times n$

Fra le tecniche di decomposizione c'è la SVD

SVD

Dato $M \in \mathbb{R}^{m \times n}$ si esprimiamo nel prodotto
di 3 matrici

$$M = U \Sigma V^T$$

• U : matrice semi-unitaria
 $m \times r$

$$U^T U = I$$

• Σ : matrice diagonale $r \times r$ dove non ci sono
numeri negativi

$$V^T V = I$$

• V : matrice unitaria
 $r \times n$ ($V^T = r \times m$)

La decomposizione si chiama decomposizione

Gli elementi di Σ sono detti valori singolari di
 M . Il numero di valori non nulli di Σ è
il rango di M

colonne di U e V sono dette vettori singolari
rispettivamente per M

- vettori singolari di sinistra di M : autovettori di MM^T
 - vettori singolari destri di M : autovettori di M^TM
- I valori singolari di M sono le lunghezze degli autovettori di MM^T e M^TM quadrate.

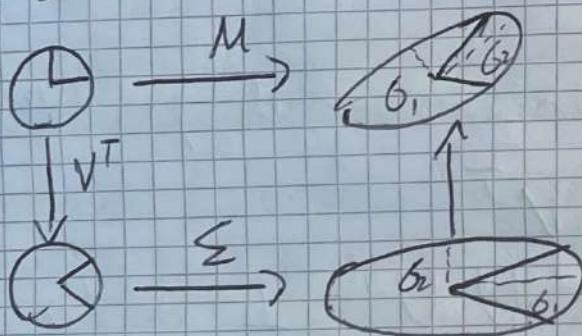
Per trovare la decomposizione SVD abbiamo
bisogno di trovare autovettori e autovetori a MM^T e M^TM .

Interpretazione geometrica di SVD

$$M: 2 \times 2$$

La SVD ruota e trasforma il cerchio in un ellisse dove i semiasse hanno lunghezze pari ai valori singolari nulli di M .

Cerchio: taggio unitario in \mathbb{R}^2 dove i "taggi"
sono i due vettori unitari canonicici.



$$M = U \cdot \Sigma \cdot V^T$$

Si può applicare SVD sulla matrice di
UTELI per avere 3 matrici più piccole
dove i conti da fare saranno più facili
e veloci.

In pratica avendo m utenti e n item si può
mappare i dati in un nuovo spazio con
 U , Σ e V dove i user sono le categorie.

$$r = \text{rang}(M) = |\text{categorie}|$$

Dati M con m utenti e n item la
corrispondono in $U \in VT$ dove:

- U è $m \times r$ (m utenti, r t categorie)
- Σ $r \times r$
- V è $n \times r$ (n item e r categorie)

Con SVD primo calcolare le predizioni di un
utente per tutti gli item con prodotto tra u e v .

- 1) Un utente è X vettore di m ratings per gli item.
(Rating non scritto $\equiv 0$)
- 2) Si moltiplica X per V (risparmia spazio i rating
sui spazi delle categorie)
 $Y = X V \leftarrow Y$ è un vettore con i rating per le
categorie.
- 3) Si moltiplica Y per U^T (risparmia spazio i
valori sulli spazio delle categorie sullo spazio originale)
- 4) $R = Y U^T$ è un vettore che contiene le
predizioni dei rating di U negli item

Quante sono le categorie? (r) R è un vettore
 $|Categorie| = r = \text{range di } M$ che ha i
rating per gli item

Esempio su riferito

Valutazioni riassumibili

Approccio binario = un utente di raccomandazione si
può vedere come un classificatore o
predittore.

Possiamo valutare l'utente U applicando le valutazioni sui valori
di rating.

Supponiamo la matrice di utilità sia binaria
ovvero like / not like (oppure preferisco
non direne nulla) mettendo il rating in alto/basso.

Possiamo a questo punto usare le misure della matrice di confusione vista per la classificazione.

True positive: L'utile che il rating ha predetti.

True negative: Non utile che il rating ha predetto.

Possiamo valutare l'affidabilità del rating con l'elenco ROC ad esempio (o comunque altre curve con il medesimo scopo).

Approssimazione (non binaria)

Una misura che ci fornisce come la grande libeza in media il ~~rating~~ rating di raccomandazione.

RMSE = Root Mean Square Error

$$RMSE(\hat{R}, R) = \sqrt{\frac{\sum_{i=1}^{|R|} (\hat{R}_i - R_i)^2}{|R|}}$$

\hat{R} = vettore rating predetti.

R = vettore rating veri \Rightarrow stessa entità.

Questo rating di valutazione è buono per valutare rating di raccomandazione non binari così non basta solo utile/utile.

Un esempio è un rating dove ogni item ha un rating (da parte di un utente) da 0 a 5.

Rete e modelli torbidi

Sistema complesso: reti in cui varie entità interagiscono tra loro.

Esempio: infrastruttura di comunicazione

"Complesso": fatto a numero finite entità e tipo di interazioni.

Reti: modelli matematici che rappresentano il funzionamento di un sistema complesso.

Graph = (V, E) $V \equiv$ insieme vertici (entità)

$E \equiv$ insieme archi (interazioni entità)

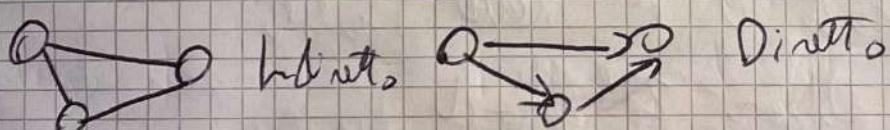
Network ricerca: scienza che studia i grafici
E' una disciplina molto giovane che abbraccia
varie materie come matematica, fisica, informatica ...

Grafi diretti e indiretti

$(a, b) \in E \rightarrow b$ è adiacente a a

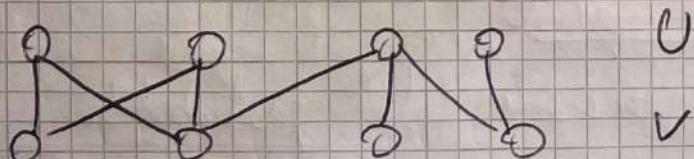
Grafo diretto: se $(a, b) \in E$ ricavamente $(b, a) \in E$

Grafo indiretto: non è diretto

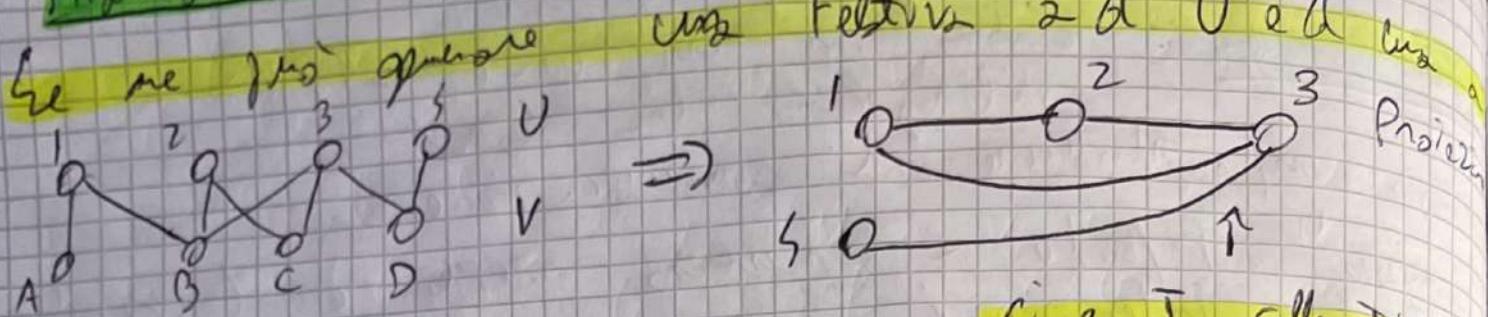


Si parla di rete gestita se ogni arco ha un numero associato che ne rappresenta il peso
Si possono associare valori anche ai nodi, potrebbero essere delle etichette.

Grafo bipartito: nodi divisi in U e V disgiunti
ogni suo collega un nodo a un nodo dell'altra parte.



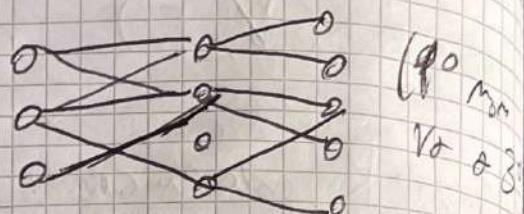
Proiezione del grafo bipartito



Si può considerare anche la proiezione V

Grafo multipartito

Estensione del bipartito



Grafo completo

Grafo in cui tutte le coppie di punti sono collegate da un arco

Grafo regolare

Tutti i nodi hanno lo stesso grado
(grado di un nodo è il numero di archi collegati)

Se un nodo ha grado 0 è detto isolato

Nei grafici binetti abbiano 2 gradi

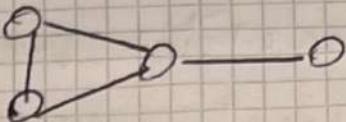
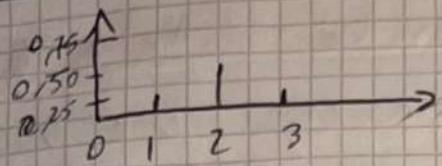
- Grafo uscente di m : nodi adiacenti a m
- Grafo entrante di m : nodi a cui m è adiacente
- Grafo totale di m : gradi uscenti + gradi entranti

Grado medio del grafo $\langle k \rangle = \frac{|E|}{|V|}$

Distribuzione dei gradi

È una distribuzione di probabilità dove

P_k è il numero relativo di nodi con grado k



$$P_K = N_K / N \quad \text{dove} \quad N_K = \text{num. dei gradi di } K \\ N = \text{num.}$$

Cammino tra 2 nodi e Distanza

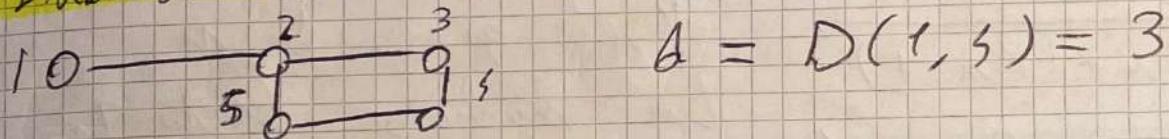
Cammino tra v e w è una seq. ordinata di m archi $(v_0, i_1) - (v_1, i_2) - \dots - (v_{n-1}, i_n) - (v_n, w)$ dove $v_0 = v$ e $v_n = w$

m = lunghezza cammino

Il cammino minimo è quello con lung. minima
Distanza D tra 2 nodi = lung del cammino minimo

Diametro di un grafo

Distanza minima tra 2 nodi in un grafo



Ciclo

Particolare cammino che torna al nodo di partenza
Un particolare ciclo è quello di lung. 1
che non contiene un vertice.

Un grafo senza cicli è detto semplice o albero.

Connessione

Se 5 sono connesi se c'è un cammino tra
ogni 2 di essi.

Un grafo è连通的 se ogni coppia è connesse
se non è disconnesso.

Un grafo disconnesso è l'unione di tutti i componenti.
I componenti sono gli sottografi connessi.

Nel grafo orientato vi è connettività come e tale
 G è fortemente connesso se per ogni $U \in V$
 esiste il cammino da $U \rightarrow V$

G è debolmente connesso se per ogni $U \in V$
 esiste un cammino da $U \rightarrow V$ se esiste
 un cammino in G' (grafo come G in cui non
 c'è direzione)

Componente \Rightarrow può essere debolmente o fortemente
 connessa

Coefficiente di clustering

$$C_m \leftarrow m$$

Misura di quanto gli individui di un
 loro connesso tra loro.

$$C_m = \frac{2L_m}{K_m \times (K_m - 1)}$$

K_m : grado di m

$$C_m \in [0 \dots 1]$$

L_m : numero stich tra
 i K_m adiacenti di m

Il coefficiente di clustering misura la densità locali
 delle reti in un nodo m .

Più densamente interconnesso è il vicinato di m
 più alto è il coefficiente di clustering

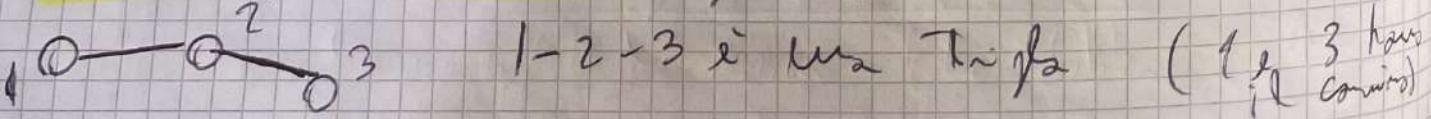
Coefficiente di clustering medo

Media dei coefficienti di clustering di ogni nodo

$$\langle C \rangle = \frac{1}{N} \sum_{i=1}^N C_i$$

Coefficiente di clustering globale

C_S rapporto tra numero triangoli nella rete
 e il numero di triple di nodi connessi tra loro



1-2-3 è una tripla

(1 è 3 hanno connnesso)

Centralità di un nodo

Maius l'importanza di un nodo nella rete

Esistono diverse misure di centralità

Degree centrality: è la già servita, il fatto
di essere collegato ad altri nodi
Più è alto il grado più è importante
E' una misura troppo semplice.

Betweenness centrality

Dato V e c, j i nodi c, j sono la
frizione di cammini minimi tra c e j che
passano per v

La Betweenness di v è ottenuta sommando b_{ij}
per ogni c, j

Dunque un nodo è centrale / importante se
sta in mezzo a molti nodi

Closeness centrality

È basata sulla vicinanza media di un nodo
rispetto agli altri nodi

Dato V i closely L_v lunghezza dei cammini
minimi da v agli altri nodi.

La closeness centrality di v è $1/L_v$

Dunque misura la velocità media con cui
un'informazione partendo da v arriva gli altri nodi

Page rank centrality

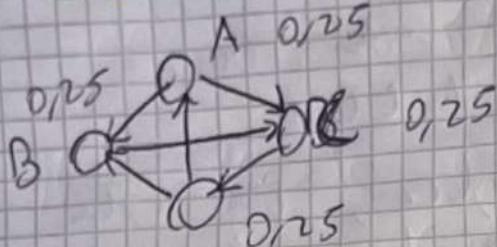
Assumiamo che nodi con elevato grado hanno
comunicazioni con maggiore rispetto alle
comunicazioni a nodi di grado minore

$$PR(u) = \sum_{v \in B_u} \frac{PR(v)}{K_v}$$

K_v : grado di v

B_u : nodi con v
adiacenti a v

Simulazione Page Rank



Ogni nodo inizialmente ha peso zero.

Ogni nodo deve calcolare il suo tesserello in funzione dell'angolo 200 centri.

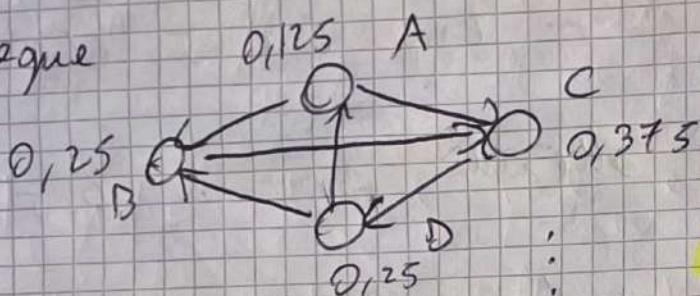
A: calcola $0,125 \times B + C$

C: calcola $0,125 \times D$

D: $0,125 \times A + B$

B: $0,125 \times C$

Segue



Il ranking poi varia molto e arriva a una situazione di equilibrio in cui i pesi non variano più e quindi ci fermiamo.

Modello Random

Permette di generare dei grafici ER con determinate proprietà.

Erdos - Renyi

Finora p è numero di nodi: N

1) Si creano N nodi isolati

2) Per ogni coppia di nodi genera un arco con probabilità p

Si ottiene alla fine un grafo Erdos o reti di Erdos - Renyi

Variante G(N, L): qui non c'è p ma L che è il numero di archi che abbiamo fatto

Si generano dunque N nodi isolati e n gerarchie
di nodi come le riportate qui sotto.

Risulta quindi non solo L archi

Proprietà del grado random

Con N nodi, P_k è prob che un nodo abbia grado k

P_k è il prodotto di 3 probabilità

- prob che sia generato a k nodi binari $\equiv p^k$
- prob che non sia generato si restano $N-1-k$ nodi
ovvero $(1-p)^{N-1-k}$

- numero nodi in cui si possono riferire le nodi
che collegano a i , da scegliere tra $N-1$
 $\binom{N-1}{k}$

La distribuzione dei probabilità sui gradi random è
una binomiale $P_k = \binom{N-1}{k} p^k (1-p)^{N-1-k}$

Il grado medio della rete è $\langle k \rangle = p(N-1)$

Per $N \gg \langle k \rangle$ si può approssimare a una
distribuzione di Poisson. $P_k = e^{-\langle k \rangle} \frac{\langle k \rangle^k}{k!}$

Dato che la distribuzione è
binomiale tutti i nodi hanno circa lo stesso grado

In reti grandi molti nodi hanno grado zero e $\langle k \rangle$
mentre pochi sono con grado diverso.

Distanza media fra nodi

$$\langle d \rangle \approx \frac{\ln N}{\ln \langle k \rangle}$$

$\langle d \rangle / \ln \langle k \rangle$ indica

che più è bassa la rete più sono piccole le
distanze.

$\ln N \ll N$ dunque le
distanze fra nodi nelle
reti random sono piccole
in media.

Se K_m è il grado di un nodo allora si ha
 $\langle L_n \rangle = p \frac{K_m(K_m - 1)}{2}$

Il coefficiente di clustering di un nodo è

$$C_m = \frac{\langle L_n \rangle}{K_m(K_m - 1)/2} = p = \frac{\langle k \rangle}{N-1}$$

Più è grande la rete più è basso il coefficiente di clustering. La stessa considerazione è fatta per il coefficiente di clustering della rete.

NB: il coefficiente di clustering è indipendentemente dal grado del singolo nodo

I grafici di reti reali e random sono significativamente diversi.

Nelle reti reali: molti nodi con grado basso e pochi nodi con grado alto.

I grafici presentano poi differenze sia per la distribuzione dei gradi sia per il coefficiente di clustering (nodo e rete).

In particolare il coeff. di clustering per una rete reale è più alto di quelli di una rete random.

SMALL WORLD

Unica proprietà che i grafici random possiedono è quella dell'abbondanza delle connessioni tra nodi.

Il fenomeno ci indica che in una rete grande 2 nodi sono collegati tramite pochi nodi.

Da qui nasce la teoria dei 6 gradi di separazione.

Poiché viene l'esperimento di Milgram che conferma che effettivamente tra 2 individui ci sono molti nodi, segue quindi 6 nodi di separazione.

Definizione di Small World

Una rete sociali ha la proprietà se $\langle d \rangle \approx \frac{\ln N}{\ln(K)}$

Le reti reali sovrisono le le proprietà
ad esempio internet ha una $\langle d \rangle = 6,58$

Modello di Watts - Strogatz

Erdos - Renyi proponeva la proprietà "Small World"
ma non il coefficiente di clustering.

Il modello di Watts - Strogatz è un'estensione del
di Modello per creare un'interazione tra
2 graf.

• Un grafo regolare = alto coll. & clustering tra
non ha proprietà small world

• Un grafo casual

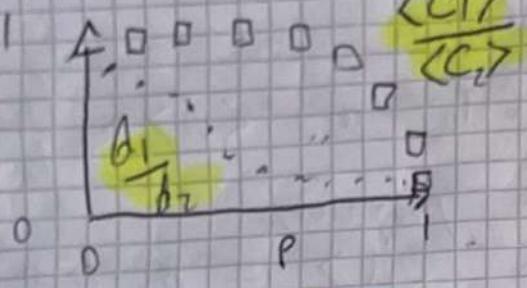
La distribuzione dei gradi è simile una
dist. di Poisson.

Il tipo di grafo dipende da un parametro p
detto parametro di rewiring. (p è una probabilità)

Algoritmo

Dati N (num nodi), il parametro p e un
 $d \geq 1$ intero

- 1) Siano $1, 2, \dots, N$ gli id dei nodi. Collega ogni
nodo i a $i+1, i+2, \dots, i+d$. Ottieniamo un grafo regolare
 - 2) Per ogni arco, con probabilità p , cambia
il suo destinazione dall'arco con un altro
nodo casual
 - Il grafo ottenuto ha caratteristiche interne che
il regolare e il casual
- $p=0$ \Rightarrow rete regolare $p=1$ \Rightarrow rete casual



$\langle C_1 \rangle$: cell b clustering con variante p che cresce \rightarrow

$\langle C_2 \rangle$: cell b clustering in una rete regolare.

α_1 : la media con variante p che cresce \rightarrow

α_2 : la media in una rete regolare.

NB: Se una rete è regolare vuol dire che la variante $p = 0$

Analizzando il grafico si può cogliere per quale p si può ottenere un coefficiente di clustering alto e una la media tra i nodi bassa contemporaneamente.

Distribuzione power-law

Nel modello di Watts - Strogatz la distribuzione di gradi è simile a una Poisson.

Nelle reti reali la distribuzione è diversa ed è detta power-law

$$P_k \propto k^{-\gamma}$$

Oltre a γ c'è un altro esponente β di grado

Nelle reti reali in genere

$$2 < \gamma < 3$$

Quindi chiamiamo questo modello approssimazione della distribuzione dei gradi.

Reti Scale-free

Una rete è scale-free se la distribuzione di gradi segue una power-law.

Nelle scale-free vi è una variazione piccola di nodi con gradi elevati e una grande variazione con gradi bassi.

Non con gradi alto: Hub

Una rete scale-free è caratterizzata dalla presenza di certi hub, cioè nodi che hanno molti connettori e altri hub sono small-world.

Il comportamento ~~scale-free~~ scale-free delle reti reali è simile nella Regla di Barabási-Albert (80/20).

Nelle reti scale-free comunque vi sono molti già "connessi". "Scale-free" si riferisce al fatto non c'è una scala (con valore di riferimento) che permette di stabilire il grado di un nodo.

In una rete random la media si trova facilmente ma in una rete scale-free non vi è alcuna informazione per stimare un valore, infatti la varianza è troppo alta.

In una rete scale-free però un nodo è difficile impossibile stimarne il grado.

Robustezza

Gli hub sembrano la rete più vulnerabile per gli attacchi esterni e nel caso un nodo venga "tagliato"

In una rete random la "cavità" di un nodo può compiere mettere tutta la rete.

In una rete scale-free il problema è grave solo se colpisce gli hub.

Per questo in una rete i proteggono meglio gli hub.

Proprietà Ultra small-world

La proprietà ~~metà~~ degli hub riduce la dicitura tra i nodi.

Per $2 < \gamma < 3$ (valori tipici nella rete reale) si ottiene la proprietà ultra small world ovvero la dicitura è ancora più bassa rispetto allo small-world.

Per $g > 3$ la rete è small-world se $\langle k \rangle < 6$
molto rade ad una rete random

Per $g < 2$ si tratta di un grafo
la media è la valenza di un nodo
Perciò non possono esistere nodi gradi con $g <$

Perciò le reti reali sono scale-free?

Molti dei dati sanno il comportamento
Caratteristiche reti reali

- Crescita: nelle reti il numero di nodi cresce continuamente nel tempo
- Preferential attachment: nelle reti un nodo nuovo tende a legarsi agli hub già esistenti che si sono perfezionati (cioè con più connessioni)

Modelli di Barabási - Albert

Il preferential attachment è ciò che favorisce la formazione degli hub

Un nodo con alto grado ha più probabilità di stabilire nuove connessioni e diventare più importante

- "Rich gets richer"

Col modelli di Barabási - Albert si prova a creare reti scale-free

Crescita e preferential attachment sono cioè creare / aggiornare reti scale-free

Algoritmo

- 1) Crea tete fatto iniziale con M_0 nudi come ogni nodo ha grado almeno 1
- 2) Aggiungi nuovo nodo e collegalo a $m \leq M_0$ nodi della tete.
Prob. che i collegi a un nodo ci sia prob K_i :
$$P(K_i) = \frac{K_i}{\sum_j K_j}$$
- 3) Ripeti 2 finché non si arriva al numero M di nodi desiderato.
- 4) per 2 garantisce le proprietà del preferenziale altrimenti

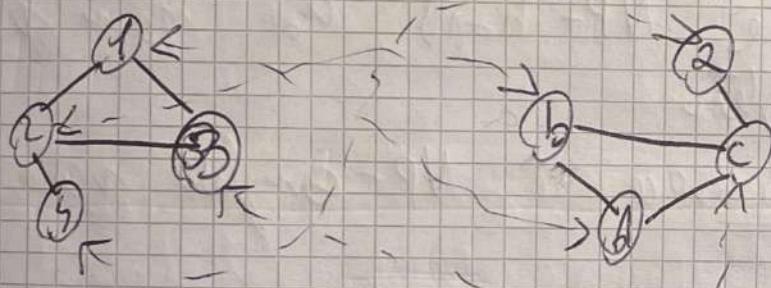
Graph Matching

Grande nel cercare isomorfismo fra grafi.

Dobbiamo verificare se 2 grafi hanno lo stesso struttura

G_1 e G_2 sono isomorfi se esiste $\rho: V_1 \rightarrow V_2$ biunivoco tale che $(u, v) \in E_1$ e solo se $(\rho(u), \rho(v)) \in E_2$

In altri termini i possiamo stabilire i nodi di G_1 , coi nomi di G_2 mantenendo la corrispondenza tra gli stessi



Mapping

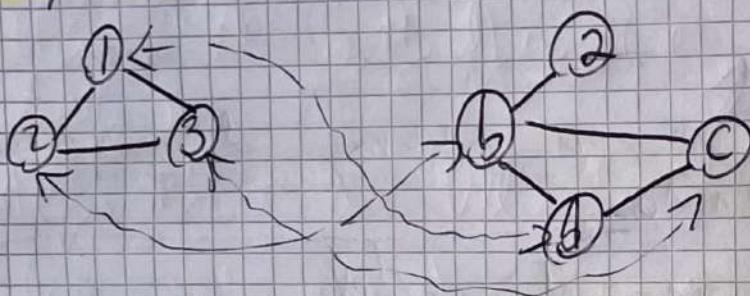
1	\rightarrow	a
2	\rightarrow	b
3	\rightarrow	c
4	\rightarrow	d
5	\rightarrow	2

Cioè tramite la funzione mapping applicata a G_1 otengo G_2 e viceversa.

La funzione è biunivoca

Sub-graph matching

G_1 è sottografo isomorfo di G_2 se esiste una pre-iniezione $f: V_1 \rightarrow V_2$ detta mapping tale che $(u, v) \in E_1$, e $(f(u), f(v)) \in E_2$. La funzione è totale iniettiva, non è necessario che tutti i nodi di G_2 (il gatto grande) vengano mappati. Imposta che vengano mappati tutti i nodi di G_1 .

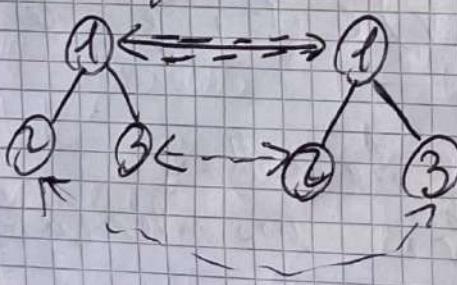


Mapping
 $1 \mapsto a$
 $2 \mapsto b$
 $3 \mapsto c$

Potrebbero esserci più soluzioni in quanto potrei mappare 1 in b e 2 in a. Più soluzioni si potrebbero avere anche nel graph matching classico.

Automa spino

homomorfismo tra un grafo G e un altro.



Il graph matching è NP-hard ma momentaneamente è NP-completo.

Sub-graph matching invece è NP-completo rispetto.

Algoritmi

Si tratta di trovare la forma canonica del grafo per
 una data classe ma l'appresentazione
 L'appresentazione è univoca ed è una struttura.
 Questi algoritmi sono efficienti perché trovare la
 forma canonica è semplice.

Un granli tool Nauty è il migliore algoritmo
 ma in generale non c'è un algoritmo
 migliore di altri. Nauty è detto anche Trivial.
 Molti di questi tool / algoritmi hanno utility
 interessanti (Ese: generare tutti i possibili grafi
 connessi su K_n nodi)

Soluzione Brute-force ← → Subgraph Matching

E' ottenere un albero con root (non se ne)
 e non binario.

- 1) 1° nodo lo mappa con ogni possibile nodo
 del 2° grafo.
- 2) 2° nodo lo mappa con ogni possibile nodo
 rimasto (nel sottoalbero)
- ⋮

La soluzione brute-force cerca tutte le
 possibili combinazioni per poi verificare solo
 quelle i buone.

In realtà non c'è bisogno di costruire un
 sottoalbero se non sono utilizzate le associazioni.
 Il metodo brute-force ha complessità polinomiale.
 Dobbiamo evitare di esplorare tutto l'albero
 tramite qualche strategia.

- Look-ahead: Prendere due il matching puzzle non periti e una soluzione finale.
 - Backtracking: Eliminare soluzioni parziali se ci sono soluzioni che non portano a soluzioni.
- Alcune tecniche per evitare di esplorare zone (insiemi) altri i problemi base nel grafo.

Se in un nodo mappa 2 in b, se hanno grafo diversi, non ha senso continuare per ottenere il matching (è vicino che non trovi nulla)

Algorithmi di Ullman

Usa il grafo del modo per filtrare i sotto-alberi.

Se il grafo del modo d'aperto è minore o uguale al grafo del nodo di target estremo allora non us.

Algorithmi di VF lib

E' basato sul concetto di State Space Representation

Il processo è una successione di stati.

Ad ogni stato s è associato $M(s)$, un insieme di coppie di nodi già mappati.

Aaggiungere una ~~nuova~~ coppia nuova di nodi da mappare in $M(s)$ per trasmettere da uno stato all'altro.

Ogni coppia deve ~~essere~~ avere un insieme di regole delle regole di fattibilità.

$M(s) \equiv$ Matching puzzle

Prembo calice VF lib

Match(1)

$S \equiv \epsilon$ uno stato

IF $M(1)$ corrisponde tutti i modi del grafico γ c'è ~~Then~~
Then scrive $M(1)$

Else Crea la $P(1)$ ormai i nuove possibili copie
combinate di modi da aggiungere a $M(1)$

For Each $P \in P(1)$

IF P soddisfa le regole di fattibilità

THEN Aggiungi P a $M(1)$

Cassa stato γ ottenuto dopo aver

aggiunto P a $M(1)$

CALL Match(γ')

Regole di Fattibilità

Per $G_1 \Rightarrow M_1, T_1, V_1'$

$G_1 \equiv \text{Query}$

Per $G_2 \Rightarrow M_2, T_2, V_2'$

$G_2 \equiv \text{Target}$

In M_1 e M_2 vi sono modi di rapporto

In T_1 e T_2 vi sono modi accostati a quelli
di rapporto (ma non ancora mappati)

In V_1' e V_2' vi sono di altri modi

$P(1)$ è sostituito dalle copie (M, m) dove

$M \in T_1(1)$ e $m \in T_2(1)$

$(M, m) \in P(1)$ viene aggiunto a $M(1)$ se:

- 1) $\forall M' \in M_1(1)$ corrisponde a m , il corrispondente $M' \in M_2(1)$
 - 2) è corrisposto a m . Regole di corrispondenza dello stato
- il numero di modi corrispondenti a m in $T_1(1)$ è minore o uguale al numero di modi in $T_2(1)$ corrispondenti a m (Regole look ahead a 1 livello)

3) Il numero di modi in $N_1'(1)$ è minore o uguale al numero di modi in $N_2'(1)$.
 Il numero di modi in $N_1'(1)$ è minore o uguale al numero di modi in $N_2'(1)$.
 Comincia a m . (Regole look ahead a 2 volte)

Nei quali tutti i modi fanno regole

(m, m) in $P(1)$ è raggiunto in $M(1)$ se:

- 1) Per ogni $m' \in M_1(1)$ predecedente a m , il corrispondente $m' \in M_2(1)$ è predecedente a m .
- 2) Per ogni $m' \in M_1(1)$ successore di m , il corrispondente $m' \in M_2(1)$ è ~~successore~~ a m .

3) Il numero di modi in $T_1^{in}(1)$ comincia a m è minore o uguale al numero di modi in $T_2^{in}(1)$ comincia a m .

4) Il numero di modi in $T_1^{out}(1)$ comincia a m è minore o uguale al numero di modi in $T_2^{out}(1)$ comincia a m .

5) Il numero di modi in $V_1'(1)$ comincia a m è minore o uguale al numero di modi in $V_2'(1)$ comincia a m .

Per coniugare l'algoritmo e trovare la parola più corta con lunghezza $\leq d =$ mille regole.

Combinazioni con N modi

Ogni modo ha $\Theta(N)$ stati, allora il costo di esplorazione di un singolo modo è $\Theta(N)$.

Caso migliore: ogni passo un solo modo combina soltanto le regole. Ci sono N modi.

$$\Theta(N^2), \text{ Ullman: } \Theta(N^3)$$

Caso peggiore: dev'essere esplorazione dell'intero albero

$$\Theta(N!N), \Theta(N!N^3) \text{ per Ullman}$$

Ci sono $N!$ stati

Complexità spazio: $\Theta(N^2)$, Utente $\Theta(N^3)$

Mentre ho le informazioni relative ad uno stato per ogni nodo ho una quantità di info ormai mapping e appartamento a cui non interessa.

Al massimo N stati (con le informazioni associate) stanno in memoria, in parallelamente a un certo istante dell'esecuzione

VF2 ottimizza lo spazio visto e ha complessità $\Theta(N)$ spazio

Viene fatta una tabella globale contiene trai.
Per tutti gli i si evita la ~~ogni~~ memorizzazione
di diverse opere delle info sui nodi dello stato

• Core_1, Core_2 contengono il mapping corrente
 $Core_1[M] = m$ se m è in uno stato
ovvero

• in_1, out_1, in_2 e out_2 descrivono
l'appartenenza dei nodi agli indici terminali
Il valore memorizzato corrisponde alla probabilità
nell'albero di finire nello stato in cui il
nodo è entrato nell'apposito insieme

Così teniamo traccia sia dell'appartenenza del nodo
agli insiami terminali e dello stato della
computazione corrispondente.

Se si fa backtracking: rigettare valori di nodi

B1

L'algoritmo RI si basa sull'ordine σ_m cui sono
ordinati i nodi della query nell'altro di riga.
Un ordinamento efficiente velocizza il matching.
L'ordine σ è indipendente dal grafico Target in
dice static ordering.

Altimenti è fatto dynamic ordering.

RI si basa su static ordering dei nodi della query.

Algoritmo RI

Pre Processing

1) Ordina i nodi della query in modo che la
probabilità di trovarne un comune nell'altro
sia possibile nel minimo.

2) Eseguire tutto questo di ricerca:

Si segue l'ordine trovato nel pre processing.

Si mappano i nodi della query a nodi del target
usando la regola del grafo (la stessa di VLSM).
Quando si trova un match completo \Rightarrow Menghera
Occhio

3) Raggiungi finché non vi è esplorato
l'intero spazio

Ordinamento *

Succede solo se innanzitutto c'è quello che ha

1) Max valore di $|V_m, v_{is}|$

2) Se c'è partita in 1: max valore di $|V_m, n_{ij}|$

3) Partita in 2: max valore di $|V_m, v_{mi}|$

4) Partita in 3: scelta arbitraria

Quale sono le corrispondenze? (quindi entrambi gli nomi e calcoli si troverebbero nel caso ma oppure non sarà accettata)

$(u_i, M(u_i))$ è accettata se e solo se:

- 1) Né u_i e $M(u_i)$ sono già stati mappati nella tabella principale
- 2) I due nomi hanno la stessa chiave primaria (se c'è)
- 3) Il numero di nomi comuni a $M(u_i)$ è maggiore o uguale al numero di nomi comuni a u_i

~~•~~ U inoltre è il nome

com
poco
ma

* Data $O^{m-1} = (u_1, \dots, u_{m-1})$, che è la sequenza di robot, il conteggio di un robot

costituito da N definisce sui 3 insiem

- 1) $V_{M,N}$: nomi in O^{m-1} accostati a V
- 2) ~~•~~ $V_{M,match}$: nomi in O^{m-1} ~~accostati a~~ V e altrimenti un nome M_m in O^{m-1}
- 3) $V_{m,MV}$: nomi comuni a V che sono in O^{m-1} e M_m sono comuni a nomi in O^{m-1}

Graph Matching in DB & profi

Con un database di profili e un query Q
Voglio trovare tutti i profili che appartengono a come
profilo (o come query)

Soluzione banale: graph matching di Q su ogni target, è poco efficiente

Per migliorare la query abbiamo indicizzare i profili del database

Dunque con l'informazione ottenuta la ~~query~~
l'apprendimento del profilo

Individuazione basata su feature

Rappresenta il grafo con un insieme F di
feature.

Dunque se un grafo non ha le stesse feature
della query rispetto a prior che puoi non
puoi fare l'algoritmo

Individuazione non basata su feature

Grafi memorizzati in uno hash tramite B-tree.
o B+tree.

Questo metodo è indicato per database dinamico

Possible features

- Ricerci grafi
- Alberi
- Cammini (SING) \leftarrow più facilmente catenare

Esempio Cammini lunghezza 2 slide

○ □ △ etichette mobi

Algoritmo base

- 1) Preprocessing: da ogni grafo si estraggono le features
Per ogni feature puoi usare questo grafo o qualsiasi altro
(e quante volte)
- 2) Filtering: dalla query si estraggono le features e si trovano i grafî del database
compatibili (ovvero quelli che hanno le features
della query)
- 3) Matching: per ogni grafo compatibile si applica
un algoritmo di graph matching per
il confronto della query

Iniziazione inversa del Preprocessing

$$p_1 \rightarrow g_1, g_2, g_3$$

La p_1 fa in g_1, g_2 e g_3

$$p_2 \rightarrow g_1, g_3$$

Nella cella g_1 ogni volta oltre

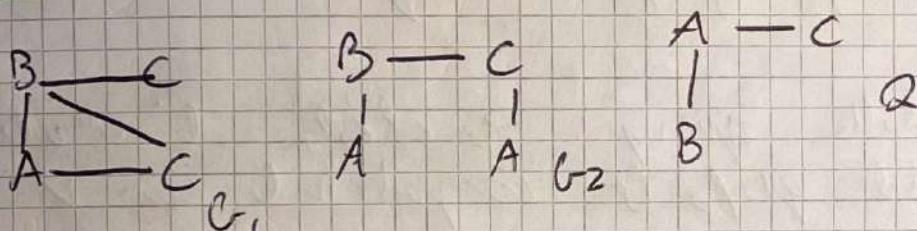
$$p_3 \rightarrow g_3$$

a un grafo c'è l'occasione

Trovare le p della query è tramite operazioni di intersezione vediamo quali grafici contengono tutte le feature di Q (tenendo conto anche dell'occorrenza)

Algoritmo SING

Agli ogni feature spesso associare non solo l'occorrenza ma anche il nodo "iniziale"



AB e AC stanno sia in C_1 che C_2 ma solo C_1 contiene Q

Dunque succede perché AB e AC partono dalla stessa nodo in G_1 ~~ma~~ e da nodi diversi in G_2

Avremo 2 indici:

- indice inverso globale (qualsiasi query)
- indice inverso locale; ne abbiamo 1 per grafo, riportata come una bitmap

$$p_1 \rightarrow \begin{matrix} v_1 \\ v_3 \end{matrix}$$

Indice da quali v parte la feature

$$p_2 \rightarrow \begin{matrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{matrix}$$

$$\nwarrow v_1 \swarrow v_6$$

Procedural query in SING

- 1) Filtering 1: Per ogni b e d gradi i grafici in cui b compare un numero di volte maggiore o uguale al numero di occorrenze in Q . Di questi inserire facendo l'intersezione e lo chiamiamo R .
- 2) Per ogni C in R Unione l'indice locale. Scartiamo i grafici per cui almeno un nodo di C non ha molti compatibili successori.
- 3) Matching: Si può filtrare tra entrambi i pochi graph mining

Filtering 2

Dato un V della query: N classi. L'insieme dei nodi del grafo G compatibili con V \times b e c compatibile con V se tutte le festività che partono da v in Q partono da v in G .

Sia S l'insieme delle festività che partono da V .

E' classificato nell'indice locale b & un AND degli tra i nodi l'insieme associati alle festività in S

Mining di Grafi

Dai mobili trasformare posizioni di mobili a grafi

Transazione \rightarrow Grafo

Item \rightarrow Nod

Relazioni \rightarrow archi

Trasare itemset frequenti \rightarrow trovare sotto grafi frequenti

Frequent Subgraph Mining FSM

Consiste nel trovare i sottografi che c' sono frequentemente in un database di graph.

Supporto a un sottografo: numero graphi in cui c'è.

Se il supporto raggiunge una soglia θ , il sottografo è detto frequente.

Output di FSM \Rightarrow insieme sottografi frequenti (grado supporto)

La frequenza è espressa o in valori assoluti o in percentuale. Questi si potrebbero esprimere come un percentuale.

Supporto \Leftrightarrow Vlue ass. %

Frequenza \Leftrightarrow supp / N

Rappr. Azioni:

Il supporto di un sottografo S in un database non può essere maggiore di quello di S

Se S non è frequente neanche il numero graphi con S come sottografo sarà frequente.

Schemi FSM

1) Cerca nodi e archi frequenti e metti in O

2) Per $K \geq 3$ fasi

a) Genera candidati: partendo dai sottografi frequenti coi $K-1$ nodi comuni insieme C dei sottografi candidati con K elementi/nod.

b) Pruning: elimina sottografi rimbombanti e i restano i candidati con sottografi da $K-1$ nodi non frequenti.

c) Counting: calcola supporto di ogni candidato e inserisce nei sottografi frequenti con le nodi aggiungili.

3) Torna a 0

Approccio BFS e DFS

Il primo approccio berentro è BFS (in arancione)

Una altra strategia è DFS che richiede meno memoria ma è meno efficace

BFS: prima di generare i comb. con $k+1$ nodi
si calcola su i sottografi frequenti con k nodi

DFS: si calcola rispetto ai vari sottografi combinati
di K nodi. Se il frequente è intermedio.

Come generare i comb.?

- a) Join-based
- b) Extend-based

Join-based

Un comb. con $k+1$ nodi si ottiene dalla
unione di ~~R mod~~ graphi con k nodi
(frequenti)

L'unione è possibile solo se i 2 sottografi
hanno un simile 1 sottografo con $k-1$ nodi
in comune. Questo sottografo è detto
core graph

Lo stemo ragionamento si può fare con gli
archi

Con otterremo un grafo con un core in più
e optionalmente un nodo tra gli archi

Nella Join-based un join tra 2 graf. può
essere già costituito perché

- a) I due sottografi hanno già un core in comune
- b) Un core già avere già le corrispondenze

Da join bisogna però generare lo stesso sottografo

Nella ext-heap

Un sottografo candidato con $K+1$ nodi è generato estendendo un sottografo preesistente con K nodi aggiungendo un nodo.

Per evitare riconoscere a partire da un grafo l'estensione di passo nel cammino right-most

Il nodo è aggiunto solo a partire dai nodi nel cammino già eletti nell'albero binario operato da una DFS sul grafo

Anche qui i passi, invece di aggiungere un nodo, aggiungono un arco

La generazione può portare riconoscenze ovvero grafici indesirabili.

Dobbiamo essere una rappresentazione di grafi che aiuti agli algoritmi a generare grafici isomorfi / riconoscibili

Non possiamo usare metriche o liste di adiacenze in quanto 2 grafi isomorfi potrebbero avere metriche diverse

Stringa di adiacenza

Stringa ottenuta considerando le righe della matrice, se faccio mi grafi indetti solo alla fine non ho migliore.

Se com M nodi $\Rightarrow M!$ possibili stringhe di adiacenza

Cerchiamo la forma canonica del grafo ovvero la stringa di adiacenza lessicograficamente più piccola.

La forma canonica di un grafo è univoca

Due sottografi sono allora riconoscibili se con la stessa forma canonica

Dopo aver ottenuto i rottagli frequenti abbiamo
che si è significativa statisticamente.

Facciamo un post - processamento sui risultati per rimuovere
i background

La significatività si calcola fissando un database
di background contenuto in precedenza

Post - processamento

- 1) Partendo dal database D di N grafici, continuo il database di background B formato da N' grafici
random ottenuti da D .
- 2) Ricerca $1/K$ volte, in ottengono K database
di background
- 3) Calcola rapporto X del rottaglio S in ogni database.
Ottieni una distribuzione di probabilità di valori X .
- 4) Calcola probabilità di osservare un $X \geq$
 σX_0 ovvero il rapporto nel database D .
Questa prob. si dice p -value
- 5) Se p -value $< \alpha$ allora S è un rottaglio
frequentemente significativo.

Di solito $\alpha = 0,05$

Algoritmo eff - swag

Ci serve per generare il database B .
Permette di ottenere un grafico $A \neq G$
con la stessa distribuzione di grafi di G .

Colice (G)

- 1) Prendi 2 ordi card di G, (a, b) e (c, d)
1. Gl ordi sono disjunti ($\cap = \emptyset$ nli in comune)
- Im G non ci sono (a, d) e (b, c)
- 2) Scambia le vert. di due ordi
- 3) Itera 1 e 2 un conto numeri di volte
(d'alto 100)

La scelta degli ordi in 1 garantisce che alla fine G e R abbiano la stessa distribuzione di gradi anche se sono grafici diversi.

Algoritmo FSG

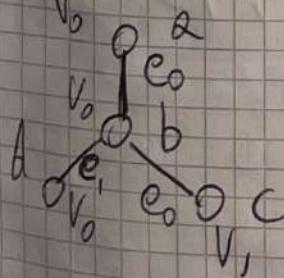
Usa una struttura BFS per trovare i sotto-graf frequenti
compartiti comuni \Rightarrow Join board (nli)
I sotto-graf sono rappresentati dalla forma canonica

Premesso a, b, c in ordine

Ottiene la forma canonica usando la radice di
sotocenzo.

1) Partizione per grado

2) Partizionare ogni partizione per etichetta



	a	b	c	d	e
a	0	0	0	0	e_0
b	0	0	0	0	e_1
c	0	0	0	0	e_2
d	0	0	0	0	0
e	c_0	c_1	c_0	c_1	0

a e b nello stesso parzionale perché stessa gradi e stessa etichetta

Per ogni combinazione partizione genera le combinazioni

In questo caso

$a \neq b \neq c \neq d \neq e$

\neq questa è la combinazione

**Trova tutti i sottografi
frequenti con 1 e 2
nodi**

**Generazione dei
candidati:**

**Genera sottografi
candidati con k nodi,
mediante join di
sottografi frequenti
con k-1 nodi**

**Conteggio della
frequenza**

**Seleziona candidati la cui
frequenza è al di sopra di una
soglia minima di supporto**

Step 3:

- Generazione core
- Join
- Cancellazione degli Attri
Non frequenti

Ottimizzazione: profile:

- Per ogni sottografo k numero di come compare
nei profili $k-1$ - sottografi frequenti
- Una indicizzazione inversa
- Usare ~~cache~~ cache di automorfismi di core frequenti

Ottimizzazione: Inverted List

Quando vogliamo contare le frequenze di un k -sottografo, esistono due casi: se la query è una query sulle transazioni e addizioni in pratica si risolve un $M \times M_K$ problema di subgraph matching (M : query nel DB)

Con le inverted list riduciamo il costo

- 1) Ad ogni S (sottografo frequente) associa la transazione identificata da (TID)
E' la lista delle transazioni contenenti S
- 2) Scegliere Freq di un $k+1$ -sottografo comune
intersecare le TID dei suoi sottografi frequenti
- 3) Se la m intersezione < mfp : i conta il sottografo
Se no i conta la sua
frequenza solo nelle transazioni
della sua intersezione

Algoritmo g-SPAN

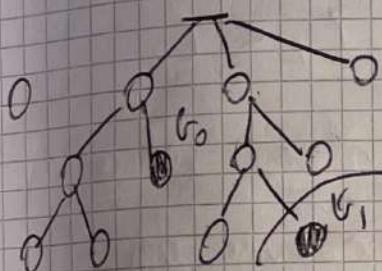
Usa una struttura DFS per ricercare i rotoli frequenti.

Rappresentazione rotoli = DFS sbe

Spatio di ricerca = albero costruito con DFS sbe
di tutti i grafi

I rotoli frequenti sono inseriti aggiungendo caselli
sia alla volta con extend-based

Lo spazio di ricerca



Spazio di ricerca via un ordinamento lexicografico per i vertici.
Vertice al livello m rappresenta un rotolo con m caselli

I vertici con DFS sbe già giunti sono resenti finiti. Se abbiamo visitato b_0 con DFS sbe X non serve più visitare b_1 con DFS sbe X perché così entriamo in "pisti inutili".

Il DFS ~~sceglie~~ sarà la lista dei nodi non visitati in un grafo (ognuno un simbolo).

Saranno un albero in pratica.

Se eseguiamo DFS: sceglierà tratta ricerca in profondità.

Viamo a scrivere per ottenere una sequenza ordinata di simboli fatto colDFS DFS

Definito l'ordine tra i due DFS succesi sul grafo il colDFS più piccolo (un grafo può avere più alberi DFS)

- Per:
- vertice right-most: ultimo nodo visitato
 - cammino right-most: cammino da vertice - nodo right-most
- $E_{l,T} = \{e \mid \forall i, j \quad i < j \quad e = (v_i, v_j)\}$ znd forward
- $E_{b,T} = \{e \mid \forall i, j \quad i > j \quad e = (v_i, v_j)\}$ znd backward

La sequenza ordinata N delle file edice DFS
ha queste proprietà:

- 1) Data $v \in N$ si ordi backward $V_{1:n}$ dietro
oppure prima di forward
- 2) Tra forward ha precedenza l'uno con
festinazione un modo vincolo prima
 $(i, V_{J_1}) < (i, V_{J_2}) \Rightarrow J_1 < J_2$
- 3) Tra backward ha precedenza l'uno con
sorgente un modo vincolo prima.
 $(i, V_{J_1}) < (i, V_{J_2}) \Rightarrow i < J_1 < i_2$
- 4) Se due backward partono dalla stessa radice
ha precedenza l'uno con festinazione un
modo vincolo prima
 $(i, V_{J_1}) < (i, V_{J_2}) \Rightarrow J_1 < J_2$

Ordine lexicografico tra codici DFS

- E' grande il che si trovi la sequenza minima
- Se i tempi sono uguali: ci si basa sul
ordine lexicografico dei tentativi compiuti
(Occhiaia 1° vertice, Cicchetta arco e 2° vertice)

Estensione retrograda

Dato G è l'altro DFS T su slice minimo
Ci sono 2 possibili estensioni

- Estensione backward: aggiungi arc to il vertice right-most e uno su vertice del cammino rightmost
- Estensione forward: aggiungi nodi v e s negati
o un qualiasi nodo del cammino rightmost
di SPAN opera verso condizioni di FSA

Con queste tecniche di espansione il codice DFS
degli figli aperto sarà un'estensione del slice
del padre

Mining in un singolo gfo

Dal G Thorson tutti i sottografi frequenti
la frequenza è definita in base al numero di
occorrenze in G

Sottografo frequente in G = "motivo" di G

Problema: Overlap di ocorrenze ovvero ci sono
occorrenze con nodi in comune

In base alla situazione posso decidere se
contare tra le ocorrenze o no anche queste
overlap.

Caso in cui non voglio sovrapposizioni: Vole la regola
A prima

Caso contrario: Non vole l'Aggiornamento
(potrebbe non volere)

Di solito un sottografo ricorrente / frequente è
detto motivo se è "significativo".

Calcolare i significativi di S

Crea variante tabular di G e contare frequenze di S
nelle varianti

Il χ^2 -value ottenuto dalla distribuzione delle
frequenze di S nelle varianti determina la
significatività di S

Difficile non molti sviluppatori vorrebbero calcolare
il p-value in maniera molto più efficiente
perciò dovete generare le varianti tabular.

Strategy di ricerca

Occhio curva i sottografi (escluso)

Network - centric: ricercare tutti i possibili sottografi con K nodi nella rete e poi riunire insomma ovvero creare gruppi di grafi (in ogni gruppo i grafi sono isomorfi)

Node - centric: generare tutti i possibili sottografi con K nodi e cercare poi in C la verifica di un sottografo è costata, ma genera tutti i grafi possibili con K nodi con le nodi con le reti e' molto costoso.

Set - centric: Un mixto dei primi 2.

Prima cercano alcuni sottografi e poi trovano i sottografi isomorfi nella rete e poi raggruppano quei sottografi della rete isomorfi

Ricerca cattura: Trova TUTTI i sottografi con K nodi

Sampling: Campionamento random di un numero minimo di sottografi con K nodi e successivo contagio frequenze nel campione

- 1) Seleziona un set o modo random real
- 2) Estratti il set fino ad avere K nodi
- 3) Il sottografo campionato ha i K nodi e sia real

S campionato
bene

Una variante del Sampling prevede di fare una ricerca detta A di campione S in C

Pregi:

- Non si spende molto dalla fin. della rete
- Con molti modi campioni ottiene risultati efficaci e veloci

Difetti: può perdere di motivi

Rete neurale

Punto sul comportamento dei neuroni.
 Ogni neurone riceve un input regolato da altri neuroni, li elabora e fa un output.
 Una rete neurale ha un insieme di nodi che initi i neuroni.

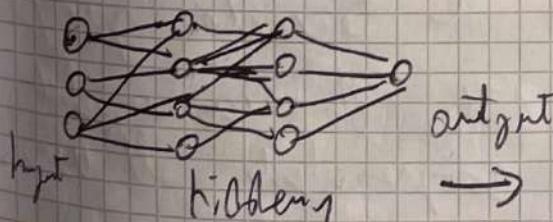
Un nodo riceve un input, ognuno con un peso più o meno bias.

Fa la somma e produce un output. La funzione è detta di attivazione.

La rete neurale fa rettangolo come un insieme di layer dove:

- **layer input**: vettore di n valori $\vec{x} = x_1, \dots, x_n$
- **hidden layer**: costituiti da 1 o più nodi.
 Ogni nodo riceve 1 o più input dal layer precedente e fa 1 o più output al layer successivo
- **output layer**: formato da 1 o più nodi che costituiscono il risultato.

E' il **deep neural network** se abbiano una rete con molti hidden layer



Tensione

Analisi multi-variale

Nel modello delle reti neurali:

• input e output di un nodo sono le tensioni

• un nodo in un layer produce diverse tensioni secondo

Comunicazioni

Rete densa: molte comunicazioni

Si dicono layer totalmente connessi se c'è ogni possibile collegamento tra i 2

Random: finito in ogni nodo riceve output solo da un solo random precedente

Pooled: partizioniamo i nodi del layer in K cluster. Il layer successivo, detto pooled, sarà formato da K nodi. Il nodo associato a un cluster riceve tutti e solo gli output del layer precedente che fanno parte di quel cluster.

Convolutional: vettore i nodi di un layer disposti in una griglia.

Il nodo di coordinate (i, j) riceve tutti e solo gli output dei nodi vicini allo stesso i, j nel layer precedente.

La rete neurale è un modello black-box, ovvero un operatore vede il risultato di output verso presentarsi di sé in stato precedente.

Explainable AI: Una serie di tecniche per fare modelli che un risultato probabile da un modello come una rete neurale sia comprensibile all'uomo.

Progettare una rete neurale

Fondamentale per costruire classificatori e generativi.

Primeri step:

- Quanti layer?
- Come connettere molti layer?
- Quanti nodi in ogni layer?
- Che funzione usare per ogni nodo?
- (attivazione)

Definito la struttura abbiamo obiettivo (numero di training set) da trovare i valori ottimali dei pesi relativi a ogni input

Definisco la funzione loss F , abbiamo trovare i pesi che minimizzano F

Problema dei pesi / struttura

E' uno studio empirico, va molto a tentativi.

Ci sono comunque delle proprietà che ci aiutano a capire se la rete risolva il problema.

• pesi by eye \Rightarrow tutti gli elementi e le uscite sono simili

• pesi by eye \Rightarrow formazione di risolvene problemi decisionali più complessi

• Troppi layer \Rightarrow Overfitting

• 3 layer \Rightarrow di solito abbastanza buona

• troppi nodi in un hidden \Rightarrow permette di memorare tutto senza sprecare memoria

Prevenire overfitting: partire da un numero basso di nodi nel hidden e aumentarli finché non c'è overfitting

Funzioni di attivazione

Dati \vec{x} , \vec{w} e b ovvero: vettore di valori di input, vettore di pesi e costante di bias

Fatta una funzione di attivazione f un modo è la funzione che prende in input 2 cose:

$$z = \vec{w} \cdot \vec{x} + b$$

In un layer tutti i nodi hanno lo stesso F

Come scrivere la funzione obiettivo
per minimizzare i pesi (minimizzare la loss function)
Il metodo più visto è la discesa del gradiente
Per far funzionare al meglio l'algoritmo di discesa
del gradiente la funzione deve avere certe
proprietà:

- 1) Continua e differenziabile in ogni punto (o quasi)
- 2) La derivata non deve tendere a 0.
Derivate消极的 tollerano la ricerca dei pesi ottimali
- 3) La derivata non deve tendere a infinito poiché creerebbe instabilità nella ricerca dei pesi

Funzione step

$$step(z) = \begin{cases} 1 & z \geq 0 \\ 0 & z < 0 \end{cases}$$

La funzione ha due valori binari, per questo motivo il solo è detto percezione.

La funzione non soddisfa ne la 2) né la 3)
dunque non ha unico minimo allo stesso del gradiente.

La funzione step è usata per classificare binario multiclasse. Esempio: Perception (output a 1 solo solo)

Se vogliamo una classificazione multiclasse
l'output sarà più solo (1 per classe)

L'output sarà un vettore
e gli altri 0
essere un solo

Funzione logistica

$$\sigma(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{1+e^x}$$

valore $\frac{1}{2}$ se $x=0$

per valori piccoli da
 $x \rightarrow 0$

per valori grandi $\rightarrow 1$

Se abbiamo un vettore

come inputabbiamo semplicemente applicare la funzione a ogni componente

$$\sigma(\mathbf{x}) = (\sigma(x_1), \sigma(x_2), \dots, \sigma(x_n))$$

Possiamo $y = \sigma(x)$

$$\frac{dy}{dx} = y(1-y)$$

Al momento da $x=0$ in una delle due divisioni lo si vede, tende a 0.

Non rigetta la 2)

Dopo che la logistica tira valori tra 0 e 1 (non binari) al contrario della ~~perceptron~~ classificazione precedente poniamo ancora a ogni nodo di output (quindi una classe) una probabilità.

Tangente isterologica

$$\tanh = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Faz parte delle funzioni sigmoidi

Ha una relazione con la logistica

$$\tanh(x) = 2\sigma(2x) - 1$$

E' in più una versione scalata e trasposta della logistica

Ha valore tra -1 e 1 ed è simmetrica rispetto all'asse Y

Ha le stesse proprietà della logistica

Funzione soft max

Ogni \downarrow nell'intero vettore (non compreende le componenti zero)

$$\vec{x}, \nu(\vec{x}) = (\nu(x_1), \nu(x_2) \dots \nu(x_n))$$

$$\nu(x_i) = \frac{e^{x_i}}{\sum_{j=1}^m e^{x_j}}$$

Torna un valore tra 0 e 1

Se sommiamo gli m risultati il risultato è 1.

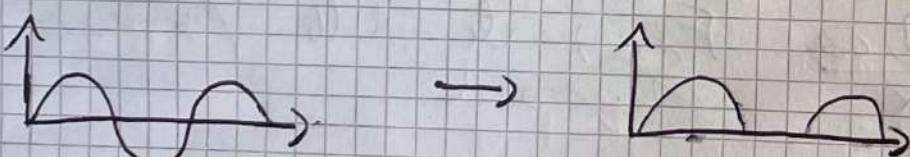
Questa funzione è ottima per ottenere una distribuzione di probabilità delle classi molto distinte.

Funzione ReLU Rectified Linear Unit

Prende spunto dai tabellizzatori a singola memoria usati in elettronica.

Il circuito trasforma un segnale alternativo in un segnale unidirezionale.

In pratica sfida un'onda con valori positivi e negativi ottenendo un'onda con solo valori positivi o nulli.



$$f(x) = \max(0, x) = \begin{cases} x & \text{se } x \geq 0 \\ 0 & \text{se } x < 0 \end{cases}$$

La funzione rigetta la 2* quindi è molto efficiente nelle sigmoidi nella fase di training. * Per x > 0

Il osleso di p e della derivata è molto semplice e veloce

Sopra si istanzia (non rigetta nulla 2) se i valori di x sono negativi

Funzione Exponential Linear Unit ELU

E' una variante della ReLU che evita il problema della saturazione con x negativo.

$$f(x) = \begin{cases} x & \text{se } x \geq 0 \\ \alpha(e^x - 1) & \text{se } x < 0 \end{cases}$$

$$\alpha \geq 0$$

ignorando
vigne tempi

limi brontate
il learning

Il learning è provato già
volte usando un α diverso
per trovarne un valore ottimo

La funzione Loss

Quella usata nella fine di learning.
 Per la quantifica la differenza tra le predizioni
 e un modello e gli output reali.
 Quindi quindi l'errore medio di generalizzazione.

I valori ottimali sono quelli che minimizzano F .

Ci sono 2 tipi di funzioni loss in base al problema

• **Regession Loss**
 $\text{output} \in \mathbb{R}$ già reali

• **Classification Loss**
 $\text{output} \in \{0, 1\}$ binarie reali

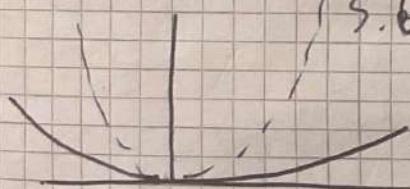
Regession Loss

Squadrato erresto loss: $L(y, \hat{y}) = (y - \hat{y})^2$

Huber loss: $L(y, \hat{y}) = \begin{cases} (y - \hat{y})^2 & \text{se } |y - \hat{y}| \leq \delta \\ 2\delta(|y - \hat{y}| - \frac{1}{2}\delta) & \text{altrimenti} \end{cases}$

$\delta \equiv$ costante

S.E.L.



Huber $\delta = 1$

$$\text{MSE} : \text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

Un'altra misura è l'RMSE ovvero la radice del MSE.

Nella pratica si usa l'MSE perché il calcolo della sua derivata è già semplice ed efficiente per l'algoritmo di ottimizzazione del gradiente.

A causa del problema di ottimizzazione e per ridurre di molto il valore di MSE

Per evitare questo problema è comune usare la costante d'errore medio variabile Huber da

Nel caso l'output abbia già valori quindi si usa un valore il calcolo finisce lo stesso ma con $\|y - \hat{y}\|$ al posto di $|y - \hat{y}|$

Classification Loss

Abbiamo m classi.

Il training set è formato da coppie (\vec{x}, \vec{p}) dove \vec{x} è l'input e \vec{p} è la m. A prob.

$p_i = \text{prob. } x \in \text{classe } c_i$

E se la classe è progettata per avere in output una funzione di prob. \vec{q} deve essere in softmax tra \vec{p}^0 e \vec{p}^1 che quest'ultimo ha distanza

Entropia

Ci sia un alfabeto di n simboli
Si vuole trasmettere un messaggio con d in simboli
In ogni punto del mix la prob di ricevere
l'i-esimo simbolo è P_i
Shannon: il numero di bit necessari per codificare
il messaggio è $H(\vec{P}) = -\sum_{i=1}^n P_i \log P_i$
 $-\log P_i$ è il numero di bit necessari per codificare il simbolo i .

Uno schema ottimale via $H(\vec{P})$ bit.

Supponiamo l'output della reti sia \vec{q}
Per ogni simbolo i verremo $-\log q_i$ bit

Sono in uno schema di codifica sub-ottimale
e serve questo numero di bit (medio)

$$C(\vec{P} \parallel \vec{q}) = -\sum_{i=1}^n P_i \log q_i$$

Detto opposto è detta Entropia incocciata e
ci fa capire a quanto \vec{q} (output) è \vec{p} (reti TB)
sono vicini o lontani.

Quanto minore ci dice quanto meno simili
 $C(\vec{P} \parallel \vec{q}) \geq H(\vec{P})$

Vogliamo che C sia più vicina possibile a H .

Vogliamo minimizzare la divergenza Kullback-Leibler
che è la differenza tra C e H .

Minimizza il numero medio di bit in cui che
serve per ogni bit

$$KL(\vec{P} \parallel \vec{q}) = C(\vec{P} \parallel \vec{q}) - H(\vec{P}) = \sum_{i=1}^n P_i \log \frac{P_i}{q_i}$$

Minimizzare $KL \Leftrightarrow$ Minimizzare C . \rightarrow Nella pratica
Finito è C_{min} KL

Training di uno ret. neurale

Caricato nel lavoro i dati che minimizzano il costo medio in un training set.

L'obiettivo è garantire un costo medio basso.
Data la gran numero di parametri il rischio di overfitting è alto.

Gradiente

$$x = x_1 \dots x_m$$

Il gradiente di y rispetto a x è il vettore le cui componenti sono le derivate parziali prime di y rispetto ad ogni x_i .

$$\nabla_x y = (\frac{\partial y}{\partial x_1} \dots \frac{\partial y}{\partial x_m})$$

La matrice Jacobiana detta in gergo rete di output delle funzioni non è y ma \tilde{y} .

La matrice Jacobiana di \tilde{y} rispetto a \tilde{x} è la matrice formata dalle derivate parziali prime di ogni componente di \tilde{y} rispetto a ogni componente di \tilde{x} .

$$\tilde{J}_x(\tilde{y}) = \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \dots & \frac{\partial y_m}{\partial x_1} \\ \vdots & & \vdots \\ \frac{\partial y_1}{\partial x_m} & \dots & \frac{\partial y_m}{\partial x_m} \end{pmatrix}$$

Metodo di ricerca del gradiente

È una tecnica per trovare il minimo di una funzione anche se già variabile.

La funzione da minimizzare è la loss.

Permetto da una posizione iniziale il metodo iterativo cerca la direzione di massima discesa del valore della funzione e aggiorna i valori procedendo in quella direzione.

La direzione di massima discesa è opposta al gradiente o della Jacobiana se l'output è un vettore.

Algoritmo

1) $\mathbb{R}^n \rightarrow \mathbb{R}$ funzione da minimizzare

2) \vec{x}_t vettore di n valori calcolato all'iterazione t

3) $t = 0$ Si inizia \vec{x}_0 l'ambra

4) Calcola $y_t = p(\vec{x}_t)$ e gradiente $\nabla_{\vec{x}_t} y_t$

5) $\vec{x}_{t+1} = \vec{x}_t - \eta \nabla_{\vec{x}_t} y_t$

6) $t++$, ripeti da 2

Il metodo è fatto fino a convergenza.

E' analogo per la Jacobiana.

I mini local (ma non assoluti) possono dare problemi

Learning rate η

Determina la velocità con cui si avanza che il metodo converge

Valor bassi \Rightarrow troppe iterazioni

Valor alti \Rightarrow causano oscillazioni troppo alte entendo di arrivare a convergenza

Un metodo per trovare un buon valore di η consiste nel partire da un valore alto e ad ogni passo moltiplicare η per β ($0 < \beta < 1$) fino a quando non trovi un valore idoneo

Inizializzazione dei pesi delle reti

Occhio partire da dei valori iniziali dei pesi.
Invece di fare completamente casuali facendo i pesi
che si ricomincino ogni volta diversi.

Ci sono 2 approcci:

- Scegli tra -1 e 1 un solo vettore per tutto. Uniforme
- Scegli in maniera random secondo una Normale

Stochastic gradient descent

Variante dove ogni iterazione lavora su un piccolo
campione di dati del training nel senso che solo

più veloce del metodo classico e sbatto a Th grad
il gradiente deve essere calcolato in qualche modo.

Un altro modo è la back propagation che usando
il grafo computazionale calcola il gradiente in
maniera veloce.

Il grafo computazionale è un grafo oids di rete

DAG che contiene il flusso di dati di una rete neurale

A ogni nodo è associato un operando e
oggi solitamente un operatore

Operando può essere: un valore, un vettore, una matrice

L'operatore può essere un operatore algebrico, una
funzione d'attivazione o una Loss.

Un arco collega A e B se l'output prodotto da A
è versato al nodo B

Se ad un nodo è associato un operatore l'output richiede

- Assegnare l'operatore ai valori input ricevuti
- Assegnare il risultato dell'operatore del nodo

Uno node del grafo organizzando per ottenere la creare la sequenza di operazioni da eseguire in un modo normale.

Back propagation

Allora se il grafo in senso inverso, ovvero dall'ultimo alla prima operazione.

Si parte dal gradiente della funzione Loss L rispetto a \vec{y} (output).

A titolo esempio calcoliamo i gradienti di L rispetto agli altri nodi del grafo.

Faciamo così perché non è ~~ottimale~~ ovvero il gradiente di L rispetto alle matrice dei pesi W

Per questi calcoli si usa la formula della catena.

Regole della catena

Se $y = g(x)$ $z = p(y) = p(g(x))$ segue

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x}$$

La regola si estende a funzioni di più variabili.

$$\frac{\partial z}{\partial x} = \sum_{i=1}^m \frac{\partial z}{\partial y_i} \frac{\partial y_i}{\partial x}$$

La regola si estende a funzioni di vettori usando gradienti e Jacobiane.

Se $\vec{y} = g(\vec{x})$ $z = p(\vec{y}) = p(g(\vec{x}))$ segue

$$\nabla_{\vec{x}} z = J_{\vec{x}}(\vec{y}) \quad \nabla_{\vec{y}} z$$

Se $\vec{y} = (\vec{y}_1, \dots, \vec{y}_m)$ $\vec{y}_i = g_i(\vec{x})$... $\vec{y}_m = g_m(\vec{x})$ segue

$$\nabla_{\vec{x}} z = \sum_{i=1}^m J_{\vec{x}}(\vec{y}_i) \quad \nabla_{\vec{y}_i} z$$

Ad ogni passo il gradiente di L rispetto ad un operando O di un nodo N è calcolato prendendo in considerazione i gradienti di L rispetto agli operatori nei nodi vicini.

Il gradiente di L rispetto ad O è calcolato usando le formulazioni della regola della catena.

La formulazione visto di sopra dal numero di vicino N è del tipo di operando (numero o vettore).

L'ultimo passo dell'algoritmo consiste nel calcolo di ∇W_L . W è una matrice, non sono uscite la regole della catena.

Vediamo W come un vettore di un vettori $W_1 \dots W_m$
 $W_i \equiv$ i-esima colonna.

Allora vediamo calcolare $\nabla W_C(L)$ e si conseguono anche ∇w_L .

Continuiamo con algoritmo

Rete neurale R , training set T

- 1) Amezza W matrice per i primi 10 nodi
- 2) Allena R con W in T
- 3) Calcola vettore di output predetti e gli output reali L con L media.
- 4) Fai back propagation e calcola ∇w_L
- 5) Usando il metodo di discesa del gradiente aggiorna la matrice $W = W - \eta \nabla w_L$
- 6) Ripeti da 2 a 5 fino a che la loss L non decresce più in modo significativo oppure itera un numero limitato di volte.
(n si dice "epoch")

Con i dati ottenuti ad ogni iterazione si calcola la Loss.

Obiettivo: far diminuire ad ogni epoca il valore della Loss sia sul Training set sia sul Test set.

Lo scopo finale è minimizzare la Loss media.

Un problema comune è quello di costruire un modello molto buono sul training set ma che non generalizza e non funziona su nuovi input. Siamo in Overfitting.

Risolve overfitting \rightarrow Metodi di regolarizzazione

E' ovvio che i dati hanno vere associazioni e costi produttivi modelli in generale sono generalizzabili.

E' forse il metodo del gradiente ascendente aggiungendo un termine di perdita alla Loss.

In questo modo i pesi saranno per i costi.

Regolarizzazione L1 e L2

$$(1) L = L_0 + \alpha \sum_{i=1}^k |W_k|$$

LASSO

$$(2) L = L_0 + \alpha \sum_{i=1}^k W_k^2$$

RIDGE

α è un iper-parametro tra 0 e 1 \equiv parametro di regolarizzazione

L2 penalizza i pesi vicini a 0 mentre con L1 almeno un peso sarà diverso dall'uno e zero.

L1 è preferibile se si vuole fare feature selection ponendo a 0 le feature meno importanti.

Visualizzazione

In base alla regolarizzazione che vengono minimizzati i termini alla fine ci sono

Dropout

Scop: Evitare i bias in diverse sub

Tecnica simile al batch averaging.

Prefabile se la test sample è grande.

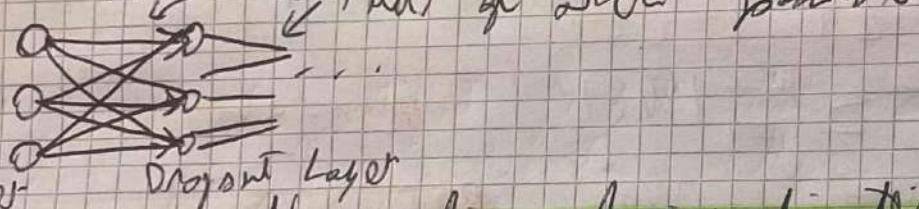
Ad ogni epoca si seleziona un numero ~~casuale~~
di nob. feature e si cancellano.
Gella rete ridotta si effettua l'iterazione.

La frequenza di nob. è detta ad ogni passo del v
batch dropout rate

Alla fine del training set i valori vengono filtrati
cioè moltiplicati per il dropout rate

Perché? Training più veloce e si abbassa la ret
a lavorare su dati ancora più diversi

Al layer si applica il dropout layer prima un
layer seguente totalmente connesso al layer.

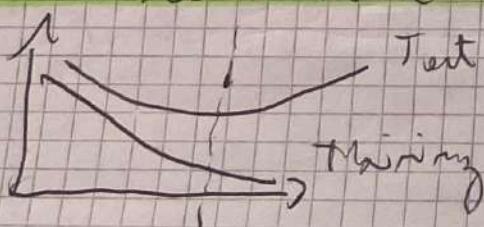


Non applica funzione di attivazione o altro
Filtra solo mente il layer corrispondente
(ovvero il precedente).

Il filtro si fa distinguendo alcuni nob. del
dropout layer

Early Stopping

Nella realtà si osserva che non sempre che si
ottimizza la funz. L, ma decresce regre sul
training set e potrebbe a un certo punto iniziare
a crescere sul test set



La tecnica per evitare
ogni problema consiste
in stoppare appena la loss nel
test inizia a crescere

Con l'early stopping si può evitare in entrambi i casi di overfitting sul training set.

Poiché invece ci sono due set da valutare, si effettua l'early stopping. Viene training e validation set.

Si stoppa l'ottimizzazione del training appena il loss rispetto al validation set inizia a crescere.

Aumento del training set

L'aumento delle tute neural aumenta se ci sono più dati nel training set.

Training set piccolo \Rightarrow Aggiungere dati artificiali.
I dati originali applicando trasformazioni ai dati del training set o aggiungendo tumore.

Aggiungendo (per esempio) tumore molto avanzato, più difficile e addirittura la tuta.

Esempio: tute usate per classificare immagini di tumori. Aggiungere immagini tumorali le immagini bidimensionali aggiungendo un po' di tumore...

Tipologie di tute neurali

Feed - Forward Networks FFNs

Le informazioni si muovono in una sola direzione. Ciò non avviene nei cicli.

Convolutional Neural Networks CNNs

Contiene 1 o più layer convolutionali.
I dati degli input sono gli stessi per tutti i nodi del layer convolutionale stesso.

In genere si alternano layer convolutionali a layer pooled con un numero di nodi che minimizza in modo progressivo.

Le CNN sono viste solitamente nelle classificazioni delle immagini. Minimo i fotocettori dell'occhio.
I nodi di un convolutional layer si pensa alle immagini come di filtri.

Il filtro ~~è~~ deve lavorare su un quadrato $p \times p$.
A volte nelle righe nel layer precedente.
 p : dim filtro. Nelle immagini: $p \times p$ pixels.

Considerato il pixel x_{ij} il filtro è applicato sul quadrato dove x_{ij} è in alto a sin.

Output del nodo $x_{ij} \Rightarrow z_{ij} = \sum_{k=0}^p \sum_{l=0}^p w_{kl} x_{i+k, j+l}$
E' proprio vero anche altri schemi: x_{ij} pixel centrale
Ottimale Obiettivo convolutional layer: output stessa dim input
Obiettivo Pooling Layer: output stessa dim minore dell'input

Stride

Per calcolare gli output prodotti dai nodi del convolutional layer basta osservare:

- Scorrere il filtro
- Applicare ad ogni posizione

Stride: indica di quanto posizioni il filtro si sposta per arrivare da una posizione alla successiva.
Solitamente $\text{Stride} = 1 \Rightarrow$ Output stesso dim input
Se $\text{Stride} > 1 \Rightarrow$ output più piccolo

La matrice di output potrebbe rimettere problematica da calcolare se ricorda sulla dim dell'input che potrebbe non permettere una completa convolution.

Soluzione: Zero - Padding ovvero aggiungere righe e colonne 0 alla matrice

Pooling layer

Prende come input l'output di un convolutional layer, produce un output più piccolo.

Ribassone dimensione: fornita a una funzione di pooling come la funzione \max . Essa aggiunge una regola a valori a un solo valore.
La funzione di pooling agisce su un quadrato di $k \times k$ di valori input

Pur ottenere l'output completo della convolution del filtro col parametro $k \times k$

1 step → output più piccolo

2 step → output più piccolo

Valori eccessivamente alti di $f \Rightarrow$ perdita informazione

CNN si può applicare anche nelle immagini a colori: 3 canali R G B

Il filtro non sarà più un quadrato $f \times f$ ma una struttura $f \times f \times 3$

Un solo di un layer convolutionale avrà $f \times f \times 3$ per

Esempio netto semplice

Convolutional immagini: VGG16

Recurrent Neural Networks (RNN)

Utili nel text mining

Pensare come un circuito

L'output di un layer può diventare l'input per
un altro layer (o lo stesso layer)

E' equivalente a una rete in cui tutti i passi
si fondono in un solo

I layer riguardano il tempo come memoria delle stesse
per ricordare valori avvistati in precedenza

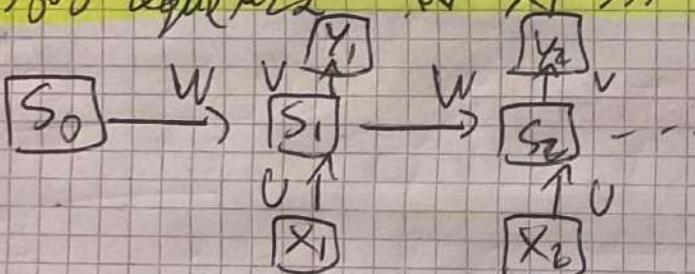
Sono abili quindi a rappresentare temporali di valori.
Eg: testo. Si può stabilire un contesto binario nel tempo.

Questo perché le RNN sono abili nella predizione di
valori successivi a partire da una sequenza di
eventi avvistati.

Input: $(\vec{x}_1 \dots \vec{x}_n)$ Output: $(\vec{y}_1 \dots \vec{y}_n)$

I passi sono rappresentati da 3 matrici U , V e W

\vec{s}_t stato nascosto funzione di memoria
di tutte info avviste prima a tempo t nella
sequenza $\vec{x}_1 \dots \vec{x}_t$



Per calcolare s_t normale
 W e U per y
rimuovere V

\vec{s}_0 è un vettore di 0.

Se si ottiene applicando una funzione di attivazione
non lineare σ allo input al tempo t e \vec{s}_{t-1}

$$\vec{s}_t = \sigma(U\vec{x}_t + W\vec{s}_{t-1} + b) \quad b = \text{bias}$$

L'output al tempo t è ottenuto applicando una
sombra lineare di attivazione considerando lo stato
attuale.

$$Y_t = g(V\vec{s}_t + \vec{c}) \quad \vec{c} = \text{vettore bias}$$

In certe applicazioni è necessario un solo output
allo stadio del process.

In questo caso si pensa di ottenere prodotti ab
ogni step su tutto il layout totalmente connesi
per generare l'output finale.

Gestire la lunghezza variabile (n : le parole)

zero-padding: si pone una lunghezza max.
Se una parola è meno di n si "completa"
con zeri artificiali. Se è più di n si tronca.

Bucketizing: si raggruppa le parole nella base
della loro lunghezza e si entra in una
RNN diversa per ogni parola volta a lung.

Risultato si vede un approccio ibrido.

Si usano diversi bucket che gestiscono ragazzi con
certa lunghezza.

Ogni parola si associa ad un bucket in modo da
gestire separate o in quella lung. O leggermente
più lunghe.

Se necessario ad un bucket scelto si fa padding.

Linti

La memoria non è "a lungo termine".

Determinare l'elenco tra parole ricche, non basta

Un verbo potrebbe essere parodiato entro dal soggetto
in una frase

Per il gradiente questo è tipico in problemi
di imballo di valori (riduzione o espansione)

Long-Short-Term Memory LSTM

Modificano la RNN per ricordare relazioni a lungo distanza.

Proprietà

- Forget: eliminare dalla memoria info non più necessarie.
- Save: cogliere le nuove parole/info nella memoria a lungo termine.
- Focus: capacità di focalizzarsi su argomenti della memoria che sono rilevanti.
Questo in base alla situazione cognitiva in corso focalizzarsi sulla memoria.

Per avere questo tipo di proprietà non serve una memoria a lungo termine e una memoria cortante.

- Long-term: ha info già acquisite
- Cortante: info si immobilizza e si leva

Struttura

Gli stati rappresentano i 2 tipi di memoria.

- Stato nascosto: s_t in t indica stato corrente
- Cell state: c_t in t indica memoria a lungo termine

h_t : spazio temporale a s_t

Salvo

vettori usati per far passare in modo selettivo alcune info di uno stato, il resto è zero.

- Input gate: \tilde{I}_t determina cosa di F_t viene messa in memoria a lungo termine
- Forget gate: \tilde{F}_t indica quali aspetti del long term memory
- Output gate: O_t indica cosa uscire dalla long term alla somma

Ajornamento del cell state (memoria a lungo termine)

- 1) Calcola update terprano dell' stato & $\vec{s}_t = \vec{h}_t$
- 2) Calcola input gate e forget gate
- 3) Ajornare memoria lungo termine

$$\vec{c}_t = \vec{c}_{t-1} \circ \vec{p}_t + \vec{h}_t \circ \vec{i}_t$$
 - \circ = Probabilità di Haverward, molto grande tra 0 e 1

Ajornamento stato nascosto (memoria cortante)

- 1) Calcola output gate (ha a che fare con input del stato precedente \vec{s}_{t-1})
- 2) Ajornare memoria cortante

$$\vec{s}_t = \text{tanh}(\vec{c}_t \circ \vec{o}_t)$$

Calcolo dell'output

Come una qualcosa RNN

$$\vec{y}_t = g(V\vec{s}_t + \vec{b})$$

$$V \equiv \text{wei}$$

$$\vec{b} \equiv \text{bias}$$

Catene di Motivo

Definizione ($Q, I = \{p(\pi_1=s)\}_{s \in S}, A\}$)

- $Q = \{1, 2, \dots, k\}$ insieme finito di stati
 - π_1, π_2 denotano gli stati omogenei o ogni istante
 - I : insieme prob. iniziali
 - A : insieme prob. di transizione debole da un per ogni $U, V \in Q$
- $$a_{UV} = P(\pi_t=V | \pi_{t-1}=U)$$

In questo si parla di catene del 1° ordine, ovvero catene dove la prob di uscire in π_t dipende solo da π_{t-1}

Si dice che la catena non ha memoria

s_1, \dots, s_t : sequenza stati omogenei

$$P(\pi_t=s_t | \pi_1=s_1, \dots, \pi_{t-1}=s_{t-1}) =$$

$$= P(\pi_t=s_t | \pi_{t-1}=s_{t-1}) = a_{s_{t-1}, s_t}$$

Sia s, r, r, r, sn, sn

$$P(s, r, r, r, sn, sn) =$$

$$= P(s) \times P(r|s) \times P(r|r) \times P(r,r) \times P(sn|r) \times P(sn)$$

$$P(\pi_1, \dots, \pi_m) = P(\pi_1) \cdot \prod_{i=1}^{m-1} P(\pi_{i+1} | \pi_i)$$

Motiva Stocistica

	s_1	\dots	s_k
s_1			
s_2			
\vdots			
s_k			

$$\sum_{j=1}^k P(\pi_t=s_j | \pi_{t-1}=s_i) = 1$$

Ovvero a_{s_i, s_j} è prob di uscire in s_k ($\forall k$) al momento t e somma dei numeri 1

Proprietà Stocistica

Catene di Markov Istruttive

Dopo ogni stato c'è una transizione uniforme alle altre.

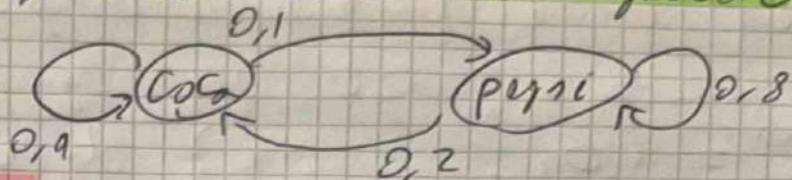
Aperiodico

Periodo T_{per} è il minimo numero di passi in cui $P^{T_{\text{per}}}$ è diverso da 0.

Se $\text{per}(\gamma) > 1$, γ è periodico.

Un caten Markov Aperiodico ha 0 stato periodico.

$$P = \begin{bmatrix} 0,9 & 0,1 \\ 0,2 & 0,8 \end{bmatrix}$$



Prob. eventi successivi

Per una persona che compra geypi a t quel 'c' è
prob che a $t+2$ compri Coca Cola

$$\begin{aligned} P(\pi_{t+2} = \text{Coca} \mid \pi_t = \text{Pepsi}) &= P(\pi_{t+2} = \text{Coca}, \pi_{t+1} = \text{Coca} \mid \pi_t = \text{Pepsi}) \\ &+ P(\pi_{t+2} = \text{Coca}, \pi_{t+1} = \text{Pepsi} \mid \pi_t = \text{Pepsi}) = \\ &= 0,2 \cdot 0,9 + 0,8 \cdot 0,2 = \underline{\underline{0,35}} \end{aligned}$$

$$P^2 = \begin{bmatrix} 0,9 & 0,1 \\ 0,2 & 0,8 \end{bmatrix} \begin{bmatrix} 0,9 & 0,1 \\ 0,2 & 0,8 \end{bmatrix} = \begin{bmatrix} 0,83 & 0,17 \\ 0,35 & 0,66 \end{bmatrix}$$

Basta moltiplicare P per n tempi per avere le prob
di non stare (tempo 2 a t)

Per destino $t+3$? P^3

Mettiamo tutte queste prob iniziali

0,6 Coca 0,4 Pepsi

Quante persone che comprano Coca negli ultimi 3 anni?

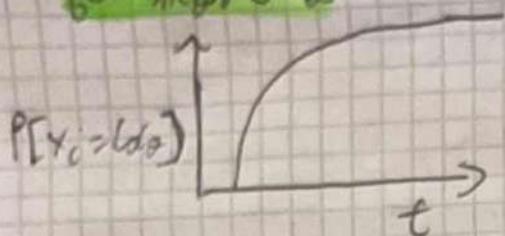
$$D_3 = D_0 * P^3$$

D_i = dist al tempo i

$$D_0 = (0,6, 0,4)$$

dist iniziale

Anzalduo t all'inizio da D dove era istante
di inizio



Si dimostra che: $\lim_{n \rightarrow +\infty} P(\Pi_n = j | \Pi_1 = c) = \lim_{n \rightarrow +\infty} A_{cj}^n = G_j$
dove G_j è il prob., converge al limite

Per le cui i valori per ogni j ottengono
 $\vec{G} = (G_1, \dots, G_k)$ che è la dist. stazionaria

Se \vec{G} ha valori non nulli $\rightarrow G = dist. stazionaria$

Se \vec{G} ha tutti valori non nulli $\rightarrow G = chia dist. staz.$

Hidden Markov Model HMM

Ci sono "mostri"

Ogni stato emette un certo simbolo con una prob.

L'osservazione vale solo la sequenza di simboli

~~Stati mostri~~

$$H_1 \rightarrow H_2 \dots \rightarrow H_L$$

$$\downarrow \\ X_1 \rightarrow X_2 \dots \rightarrow X_L$$

~~dati osservabili~~

HMM è una quintupla $(Q, I = \{p(\Pi_i = s)\}, A, \Sigma, E)$

$Q = \{1, 2, \dots, K\}$ insieme finito di stati

Π_1, Π_2 : stati osservati a ogni istante

T : prob iniziali

A : prob A_{UV} $\forall U, V$ in Q

$\Sigma = \{S_1, \dots, S_m\}$ alfabeto di simboli

X_1, X_2, \dots : simboli emessi a ogni istante

E : matrice emissioni $B_{m \times m} = E_{UX}$

E contiene più dati che oggi vanno bene per prob che è anche b.

$$C_{ab} = P(X_t = b | \Pi_t = 1)$$

Problemi principali in HMM

Evaluation: Data HMM M e seq. X calcolare $P(X)$

Decoding: Data HMM M e seq X , trovare seq Π di stati che maximizza $P(\Pi | X)$

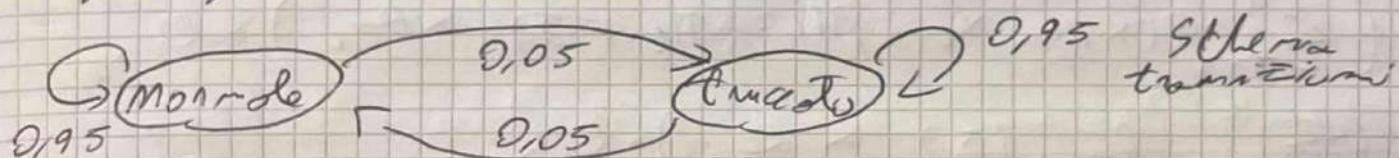
Learning: Data HMM M con prob b di transizioni non specificate e seq X trovare parametri $Z = (e_{ij}, \alpha_i)$ del modello che maximizzano $P(X | Z)$

Entro il Chorger dimostrato

Dato mondo: $P(i) = 1/6$

Dato Chorger $P(1) = P(2) = \dots = P(5) = 1/10 \quad P(6) = 1/2$

Chorger può contenere solo



Data una seq. di luce, quale è la prob che va stato generata dal modello?

Quando il chorger ha fatto il suo mondo è vero?

Stimare matrice di transizione e matrice di emissioni

1) Data M e X $P(X) = ?$ Evaluation

$$P(X_1, \dots, X_n, \Pi_1, \dots, \Pi_m) = \prod_{i=1}^m P(\Pi_i, \dots, \Pi_m) \times P(X_1, \dots, X_n | \Pi_1, \dots, \Pi_m) = \prod_{i=1}^m P(\Pi_i) \times \prod_{i=1}^{m-1} P(\Pi_{i+1} | \Pi_i) \times P(X_1 | \Pi_1, \dots, \Pi_m) =$$

$$= P(\Pi_1) \times \prod_{i=1}^{m-1} P(\Pi_{i+1} | \Pi_i) \times \prod_{i=1}^{m-1} P(X_i | \Pi_i, \dots, \Pi_m) =$$

$$= P(\Pi_1) \times \prod_{i=1}^{m-1} P(\Pi_{i+1} | \Pi_i) P(X_i | \Pi_i) = P(\Pi_1) \prod_{i=1}^{m-1} \delta_{\Pi_i, \Pi_{i+1}} C_{\Pi_i, X_i}$$

Per avere $P(X)$ si devono sommare tutte le probabilità associate a tutti die simboli presenti in X

$$P(X) = \sum_i P(X, \Pi) \quad \Pi \text{ regge } (\text{tutte le})$$

Definisco $p_t(i) = P(X_1 \dots X_t, \Pi_1 \dots \Pi_t = i)$ Probabilità condizionata

E' la probabilità assoluta di reggere il simbolo i al qualsiasi t alla fine di tutti i trascorsi del stato i .

Si può calcolare in maniera induttiva

$$p_t(i) = P(\Pi_t = i) e_{ix_i}$$

Passo induttivo

$$\begin{aligned} p_t(i) &= P(X_1 \dots X_t, \Pi_1 \dots \Pi_t = i) = \\ &= \left(\sum_{j=1}^K P(X_1 \dots X_{t-1}, \Pi_1 \dots \Pi_{t-1} = j) \times \alpha_{ji} \right) \times e_{ix_i} = \\ &= \left(\sum_{j=1}^K p_{t-1}(j) \times \alpha_{ji} \right) \times e_{ix_i} \end{aligned}$$

Il passo induttivo deriva dal fatto che se i è in t stato j allora da un qualsiasi J (in $t-1$) dopo aver emesso i simbolo $X_1 \dots X_{t-1}$

Per ragione di i a J si fa una transizione, metto dunque x_t

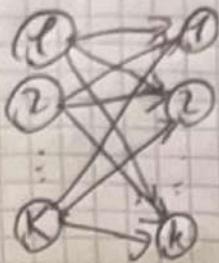
Grazie a queste proprietà posso ottenere $P(X)$

$$\begin{aligned} P(X) &= \sum_i P(X, \Pi) = \sum_{i=1}^K P(X_1 \dots X_m, \Pi_1 \dots \Pi_m = i) \\ &= \sum_{i=1}^K p_m(i) \end{aligned}$$

Grazie a questo prob si ottiene un algoritmo (ogni finito) che calcola $P(X)$ in $\Theta(K^2 m)$

$K^2 \equiv$ stati
 $m \equiv lung(X)$

• Anteprima l'algoritmo forward equivale a percorre tutti i simboli osservati in grafo detto Lattice



Osservazioni (simboli)

$x_1 \ x_2 \ \dots \ x_m$

Decoding

Dato HMM M e seq X trovare $\Pi = (\pi_1^*, \dots, \pi_m^*)$ che minimizza la prob congiunta di osservazione di m simboli emessi dagli n stati

$$P(x_1, \dots, x_m, \pi_1^*, \dots, \pi_m^*) = \max_{\pi_1, \dots, \pi_m} P(x_1, \dots, x_m, \pi_1, \dots, \pi_m)$$

Scrivere dove

$$P(x_1, \dots, x_m, \pi_1, \dots, \pi_m) = P(\pi_1) \prod_{i=1}^{m-1} \sum_{\pi_i, \pi_{i+1}} e_{\pi_i} x_i$$

Usciamo l'algoritmo di Viterbi

Invece di tenere i contributi provenienti da ogni stato al volto in volta rendiamo il contributo migliore (vol max)

Nel lattice andiamo a trovare un percorso lungo m

$$V_{tE}(c) = \max_{\pi_1, \dots, \pi_{tE-1}} P(x_1, \dots, x_t, \pi_1, \dots, \pi_{tE-1}, \pi_{tE}=c)$$

Caso base: $V_t(c) = P(\pi_1=c) \times e_c x_t$

$$\begin{aligned} \text{Passo induttivo: } V_t(c) &= \max_{\pi_1, \dots, \pi_{tE-1}} P(x_1, \dots, x_t, \pi_1, \dots, \pi_{tE-1}, \pi_{tE}=c) = \\ &= \left[\max_{j=1 \dots K} \max_{\pi_1, \dots, \pi_{tE-2}} P(x_1, \dots, x_{t-1}, \pi_1, \dots, \pi_{tE-1}=j) \times e_j c \right] \times e_c x_t = \end{aligned}$$

$$= \left[\max_{j=1 \dots K} V_{t-1}(j) \times e_j c \right] e_c x_t$$

Prob finale

$$P(x_1 \dots x_n, \pi_1^*, \dots \pi_n^*) = \max_{\pi_1 \dots \pi_n} P(x_1 \dots x_n, \pi_1 \dots \pi_n)$$

$$= \max_{c=1 \dots k} \max_{\pi_1 \dots \pi_{n-1}} P(x_1 \dots x_n, \pi_1 \dots \pi_{n-1}, \pi_n = c) = \max_{c=1 \dots k} \pi_n(c)$$

Possiamo costruire un algoritmo che calcola la prob finale in $\Theta(k^2 n)$

Nell'algoritmo è utile una struttura dati detta tabella ϕ_t che tiene traccia ad ogni t degli stati finora avvenuti per raggiungere ogni c .

$\phi_{t-1}(c) \equiv$ stato migliore da cui partire al $t-1$ per arrivare in c in t

Tanto questa struttura possa contenere il percorso ottimo (quello che minimizza) la prob di arrivare in c reg. di simboli.

Posterior decoding + Caso particolare di decoding

Data X e visto solo avere a posteriori lo stato c più probabile in t

$$P(\pi_t = c | X) = \arg \max_{c=1 \dots k} P(\pi_t = c | X)$$

$$P(\pi_t = c | X) = P(\pi_t = c, X) / P(X)$$

~~$$P(\pi_t = c, X) = P(x_1 \dots x_t, \pi_1 \dots \pi_{t-1}, \pi_t = c, \pi_{t+1} \dots \pi_n)$$~~

$$= P(x_1 \dots x_t, \pi_1 \dots \pi_{t-1}, \pi_t = c, \pi_{t+1} \dots \pi_n, x_{t+1} \dots x_n) =$$

$$= P(x_1 \dots x_t, \pi_1 \dots \pi_{t-1}, \pi_t = c) P(\pi_{t+1} \dots \pi_n, x_{t+1} \dots x_n | \pi_t = c)$$

forward $f_t(i)$

backward $b_t(i)$

$$P(\pi_t = i \mid X) = b_t(i) \cdot b_t(i) / P(X)$$

Prob Backward

Sigui ricavare anche una iniziale

$$\begin{aligned} b_t(i) &= P(\pi_{t+1} \dots \pi_n, x_{t+1} \dots x_n \mid \pi_t = i) = \\ &= \sum_{j=1} b_{t+1}(j) \times c_{jx_{t+1}} \times \alpha_{ij} \end{aligned}$$

Anche l'algoritmo per calcolare la backward ha complessità $\Theta(K^2 n)$

Learning

2 situazioni

Grazie a questo ci siamo

\swarrow Non so piano

• Trovare i λ che minimizzano $P(X \mid Z)$

$$Z = (c_{ij}, \alpha_{ij})$$



X, π reg stati nota

$m =$ simboli distinti $k =$ simboli diversi in HMM

$A_{st} =$ # volte s passa da t in π

$E_{sb} =$ # volte che s in π emette il simbolo b nella stessa posizione nella seq. πX

I valori sono quelli delle matrici tale che

$$\alpha_{st} = \frac{A_{st}}{\sum_{i=1}^k A_{si}}$$

$$c_{sb} = \frac{E_{sb}}{\sum_{j=1}^m E_{sj}}$$

Se abbiamo solo dati i valori dei parametri trovati non saranno ottimali

- Q** X, Π 1a stati non nate
 Iniziamo le valori fatti in Ase e Esb
 Aggiorniamo via via i valori in base alla combinazione
 appena ab ogni passo i risultati adattamente
 Algoritmo: Expectation - Maximization EM

- 1) Stima di Ase e Esb usando i valori correnti
 Ase e Esb fatti mettendo la transizione ab attivata
- 2) Aggiornano i parametri in base a Ase e Esb
- 3) Ripeti 1 e 2 fino a convergenza

Definiamo $S_e(i, j) =$

$$= P(\Pi_t = i, \Pi_{t+1} = j | X) =$$

$$= \frac{f(i) \times \alpha_{ij} \times e_j x_{t+1} \times b_{t+1}(j)}{\sum_{c=1}^K \sum_{j=1}^K p_c(i) \times \alpha_{cj} \times e_j x_{t+1} \times b_{t+1}(j)}$$

$\delta_t(i)$ Prob a posteriori di trovarsi nel tempo t
 nello stato i $\delta_t(i) = P(\Pi_t = i | X) = \sum_{j=1}^K S_e(i, j)$

α_{ij} = Prob di passare da i a j nello volte
 Volte in cui do i passano a qualche stato
 $\alpha_{ij} = \frac{\sum_{t=1}^{M-1} S_e(i, j)}{\sum_{t=1}^{M-1} \delta_t(i)}$

e_{ib} : Prob di entrare b nello stato i nello
 Volte in cui sono in i (entro)

$$e_{ib} = \frac{\sum_{t=1, x_t = b}^M \delta_t(i)}{\sum_{t=1}^{M-1} \delta_t(i)}$$

Possiamo continuare l'algoritmo a Baum-Welch

• iniziazione:

• inizializza matrici di transizione ed emissione fortemente (o in base a un'esperienza magica)

• iterazione

• Calcola le prob forward α

• Calcola le prob backward β

• Aggiorna i parametri Z calcolando le stime A_{ij}, E_{ib}

• Calcola $P(X|Z)$ ovvero la prob di ottenere la seq X usando i parametri Z

• Ripeti fino a convergenza ($P(X|Z)$ non cambia più il valore in maniera significativa)

• Complesità: $O(k^2 n)$

• Non trova i migliori parametri (è stocastico)

• Potrebbe convergere a un minimo locale