

① Classification → classificare i dati in 2 o +类群。

↳ Regression → Associazione fra due valori risultante del modello in grado di rappresentare percorso spazio!

② Predictive clustering: determinare appartenenza ad un cluster basandosi sulla vicinanza..

③ Sub ...: ricerca di un salviogruppo del cluster che ha una peculiarità particolare

④ Descripire Clusteri: ricerca dei parametri cluster con quali si possono individuare insieme di elementi

↳ Associazioni Rule Discovery: trovare regole di associazione di diversi insiemi di oggetti nel dataset

$$\hat{c}: X \rightarrow C \quad (x, c(x)) \in X \times C$$

con:  $x \in X$  e  $c(x) \in C$  ✓  
corretta oppurt.

Scopo: approssimare  $G$

BINARIA:  $G = \{C_1, C_2\}$

Predicci <sup>+</sup>	Predicci <sup>-</sup>
TP	
	FN

+	' '	' ''
-	FP	TN

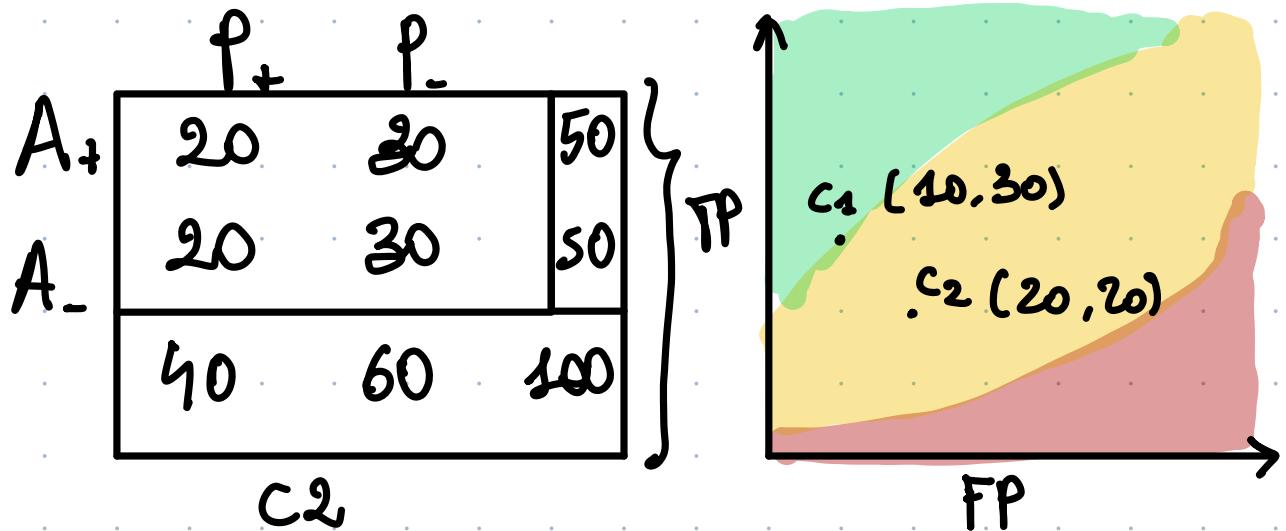
accuracy:  $\frac{1}{|\mathcal{T}|} \sum_{x \in \mathcal{T}} I[\hat{C}(x) = C(x)]$

tasso di errore:  $1 - \text{acc}$

$$\text{TPR}: \frac{\sum_{x \in \mathcal{T}} I[\hat{C}(x) = C(x) = +]}{\sum_{x \in \mathcal{T}} I[C(x) = +]}$$

$$\text{Precision}: \frac{\sum_{x \in \mathcal{T}} I[\hat{C}(x) = C(x) = +]}{\sum_{x \in \mathcal{T}} I[\hat{C}(x) = +]}$$

$$= \frac{TP}{TP + FP} = P(C(x) = + \mid \hat{C}(x) = +)$$



- Avg. recall =  $\frac{TPr + TMr}{2}$

- ROC plot: grafico su TPr e FPr

① Tracci i classificatori paralleli  
alla retta  $c_1 r = \frac{\text{Neg}}{\text{Pos}}$  hanno stessa accuracy

② Tracci i c. paralleli alla retta  
con pendente 1 hanno eguale avg.recall

- Scoring Classifier:  $\hat{s}: X \rightarrow \mathbb{R}^k$

dove  $\hat{s}$  è un vettore di k score  $\hat{s}(x) = (\hat{s}_1(x), \hat{s}_2(x), \dots, \hat{s}_k(x))$

(Se bilancia la classificazione  $|\hat{S}(x)| = 1$   
o vero basta sapere che sia positiva o  
negativa)

- Margine di errore: Positivo quando la classif.  
è corretta, negativo altrimenti.

$$z(x) = c(x)\hat{S}(x) = \begin{cases} + |\hat{S}(x)| & \text{se } \hat{S} \text{ CORRETTO} \\ - |\hat{S}(x)| & \text{altrimenti} \end{cases}$$

- Funzione loss: funzione che regola la  
penalità in base al margine di errore  
del classificatore...

$$L: \mathbb{R} \rightarrow [0, \infty)$$

$$L(z) \geq 1 \quad \text{se } z < 0$$

$$L(z) = 1 \quad \text{se } z = 0$$

$$0 \leq L(z) < 1 \quad \text{se } z > 0$$

} INPUT: Valore reale  
(margine steso)

OUTPUT: penalità  
 $L(z(x))$  che va da  
0 a  $\infty$  e vale 1  
se margine nullo

Guida un algoritmo di apprendimento al  
suo miglioramento

① 0-1 LOSS:  $L_{0,1}(z) = \begin{cases} 1 & \text{se } z \leq 0 \\ 0 & \text{se } z > 0 \end{cases}$

② Hinge loss:  $L_h(z) = \begin{cases} 1-z & \text{se } z \leq 1 \\ 0 & \text{se } z > 1 \end{cases}$

↳ Crescita lineare delle penalizzazioni per errori sempre maggiori

③ Logistics loss:  $L_{\log}(z) = \log_2(1 + \exp(-z))$

↳ approssima  $L_h(z)$  ma è continuo e mai differentiabile.

④ Exponential loss:  $L_{\exp}(z) = \exp(-z)$

↳ Penalizza fortemente errori molto altri

⑤ Squared loss:  $L_{sq}(z) = (1-z)^2$

↳ Penalizza maggiormente errori negativi ma anche molto positivi!

- Ranking: ordinamento degli esempi basato sui punteggi

↳ Ranking error rate:  $\sum_{x \in T^+} I[\hat{s}(x) < \hat{s}(x')] +$

$$\frac{1}{2} I[\hat{s}(x) = \hat{s}(x')] / (\text{Pos} \cdot \text{Neg})$$

Avero 1 punto per errore di ordinamento e  
ancora punto per l'uso corretto negativa.

↳ Esercizio: Vark-err(-,-,-,+,+,+,-,+)=

$$\frac{5+5+5}{5 \cdot 3} = \frac{15}{15} = 1 \text{ (errore massimo, caso err.)}$$

- Class probability estimation: utilizza una funzione di stima  $\hat{P}: X \rightarrow [0,1]^K$  che calcola le prob di appartenenza alle classi. k-estima associata ad un valore di output  $\hat{P}(x) = (\hat{P}_1(x), \hat{P}_2(x), \dots, \hat{P}_k(x))$ , con il vincolo

$$\sum_{i=1}^K \hat{P}_i(x) = 1$$

↳ Da questo possiamo ricavare un probability estimation tree

- Squared error:  $SE(x) = \frac{1}{2} \| \hat{P}(x) - I_{C(x)} \|_2^2$

Scegli  $I_{C(x)}$  è un vettore che contiene 1 per la classe corretta e 0 per le altre!

- Mean Squared Error:  $\frac{\sum_{x \in T} se(x)}{|T|}$  / 

errore medio
comune su tutti gli esempi

- Eupiastic Probability:  $p(S) = \left( \frac{m_1}{|S|}, \dots, \frac{m_k}{|S|} \right)$   
metodo per stimare delle probabilità facendo da un insieme di dati.  
Problema: trochi dati = poco precisare  
Problema 2: non funziona con valori nulli;

- Laplace Correction: una sorta di bavaglina che elimina valori nulli aggiungendo in ciascuna "fake" a ciascuna classe:

$$p_i(S) = \frac{m_i + 1}{|S| + k}$$

(se  $m_i$  vale 0, il ris. non è nullo)  
(se aggiungiamo  $k$  "fakes" al num. dobbiamo poi dividere per  $k$  sotto)



$$p_i(S) = \frac{m_i + m \cdot \pi_i}{|S| + m}$$

} correttezze NON uniformi  
nel caso in cui abbiamo + info riguardo alle prob. delle classi:

quando  $m = k$  &  
 $\pi_i = \frac{1}{k}$  per tutte le classi allora si

- $\pi_i$ : prob. a priori
- $m$ : numero di istanze fake da aggiungere

otteniamo la prima formula (UNIFORME)

## CLASSIFICAZIONE MULTICLASSE

- Caratteri classificatori binari, metodi:

### ① One-vs-rest:

- non ordinati:

initial dataset
1 3 2 1 3

$$\hat{C}_1: 1 \text{ vs } \{2, 3\}: y = 1, -1, -1, 1, -1 \mid x = (\dots)$$

$$\hat{C}_2: 2 \text{ vs } \{1, 3\}: y = -1, -1, 1, -1, -1 \mid x = (\dots)$$

$$\hat{C}_3: 3 \text{ vs } \{1, 2\}: y = -1, 1, -1, -1, 1 \mid x = (\dots)$$

riportati in una output code matrix:

$$\begin{matrix} \{1\} & \{2\} & \{3\} \\ \downarrow & \downarrow & \downarrow \\ C_1 & \begin{pmatrix} +1 & -1 & -1 \\ -1 & +1 & -1 \\ -1 & -1 & +1 \end{pmatrix} & \left\{ \begin{array}{l} 1 \text{ vs } 2, 3 \\ 2 \text{ vs } 1, 3 \\ 3 \text{ vs } 1, 2 \end{array} \right. \\ C_2 & & \\ C_3 & & \end{matrix}$$

- Ordinato: si fa su confronto delle rotte, il maniera ordinata, se il primo classificatore riconosce l'esempio come appartenente alla classe  $C_1$  allora abbiamo finito di già, dunque escludiamo la prima e mettiamo il dubbio sulle altre due:

$$C_1 \begin{pmatrix} 1 & 1 & 1 \\ +1 & 0 \\ -1 & +1 \\ -1 & -1 \end{pmatrix} \quad C_2 \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{pmatrix} \quad C_3 \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{pmatrix}$$

f 1 vs [2,3] f  
f 2 vs [3] f → non {1} dor  
che è stato escluso!

② One-vs-one: classi contracciate a coppie 1 vs 1.

- Simmetrico: non consideriamo l'ordine delle classi, il cui combinatorio richiedeva:  
= bbe sdare tutte le combinazioni di segno:

$$C_1 \begin{pmatrix} 1 & 1 & 1 \\ +1 & +1 & 0 \\ -1 & 0 & +1 \\ 0 & -1 & -1 \end{pmatrix}$$

- Asimmetrico: il suo contrario, quando non è possibile il caso simmetrico:

$$\begin{pmatrix} 1 & 2 & 3 & 1 & 3 & 2 \\ 1 & 2 & 1 & 3 & 1 & 3 \\ 2 & 1 & 1 & 3 & 3 & 2 \end{pmatrix}$$

$$C_1 \begin{pmatrix} +1 & -1 & +1 & -1 & 0 & 0 \\ -1 & +1 & 0 & 0 & +1 & -1 \\ 0 & 0 & -1 & +1 & -1 & +1 \end{pmatrix}$$

- Calcolo della distanza (utilizzando una matrice o.c.)

$$\delta(w, c) = \sum_i \frac{(1 - c_i w_i)}{2}$$

C<sub>i</sub>: classe corretta  
w<sub>i</sub>: vettore risultante di classificazione

• Problematica: problemi di bilanciamento degli ove-vs-rest → divide il dataset in maniera sbilanciata: con 10 classi e 10 istanze X classi su un dataset da 100 istanze il dataset è lo vs go, cioè anai sbilanciato.

Un ove-vs-ove questo non succede in questo caso introduce l'etichetta 0 per le istanze che sono estranee!

3.2 REGRESSIONE: problemi con output non-finito, nei redi. Utilizzo della funzione regressore  $\hat{f}: \mathcal{X} \rightarrow \mathbb{R}$  (o funzione di stima) per iuparole dei singoli esempi  $x_i$ . Il vero scopo del regressore è approssimare al meglio l'andamento delle funzioni  $f$  custodite l'overfitting.

3.2.1. BIAS-VARIANCE TRADEOFF: non basta purificare il modello + semplice per ridurre l'overfitting ed avere un buon risultato:

- modelli molto complessi hanno alta varianza ma non hanno bisogno del bias
- modelli molto semplici non soffrono di varianza ma hanno bisogno del bias in

larga pane!

↳ form.  $E[(f(x) - \hat{f}(x))^2]$  = (errore quadratico)

$$\underbrace{(f(x) - E[\hat{f}(x)])^2}_{\text{Bias}^2(\hat{f}(x))} + \underbrace{E[(\hat{f}(x) - E[\hat{f}(x)])^2]}_{\text{Var}(\hat{f}(x))}$$

- Bias: vale 0 se il regressore è mediamente corretto
- Var: errore causato dalle variazioni dei dati rispetto alle loro media

3.3 MODELLI DISCRETI: disponiamo di dati, siamo già a conoscenza delle classificazioni, scopo è descriverli!

3.3.1 CLUSTERING: scoprire all'interno dei dati gruppi omogenei con caratteri comuni.

- PREDITTO: apprendere una funzione in grado di etichettare delle istanze dato un addestramento su gruppi priori di etichette.
- DESCRITTO: apprendere le etichette di un certo dataset D.

↳ concetto chiave: DISTANZA.

3.3.2 Subgroup Discovery: tipo di apprendimento supervisivo con due dati su dataset etichettato  $\{(x, \ell(x))\}$  trova una funzione

$\hat{g}: D \rightarrow \{\text{True}, \text{False}\}$  tale che

$G = \{x \in D \mid \hat{g}(x) = \text{True}\}$  ha una distribuzione nelle classi fondamentali differenti delle popolazioni originali di  $D$ ,  $G \subseteq E$  ed è detto estensione.

• Solitamente...

- vengono preferiti gruppi grandi sottogruppi
- le misure di valutazione sono solitamente simmetriche  $\rightarrow \text{val}(\text{sottogruppo}) = \text{valore}(\text{sottogruppo})$

3.3.3 Association Group Discovery:

Dati  $b$  e  $h$  due insiemi di attributi o due colori;  
Dato su dataset  $D$  senza etichette lo scopo  
è trovare regole delle forme  $\{b \rightarrow h\}$  tali  
che  $b \cup h$  sia frequente e che  $h$  corrisponda  
strettamente a  $b$ .

• Per decidere quale regola sia migliore tra tutte

facciamo uso di una confidence measure che faccia uso delle support threshold:

$\frac{\text{supp}(b \cup h)}{\text{supp}(b)}$  } · Produce file troppe regole  
· computazionale costo  
(si deve fare un calcolo per ogni attributo!)

↳ solitamente allora si basa sulle frequenze.

---

## CONCEPT LEARNING e VERSION SPACE

→ generico

· Gool: apprendere concetti, ovvero ipotesi: fare a fuoco da elementi descritti trovare feature, vogliano attribuire una classe corretta all'esempio

→ Es:

- Concept: "giorni nei quali il mio amico Aldo ha fatto sport acquatici"
- Task: Predire il valore di EnjoySotk (Es) per un giorno ambito basandosi sugli altri valori
- Valori: Sky (Sunny, Rainy), Temp (Warm, Cold), Humidity (Normal, high), Wind (Strong, ?), Water (Warm, Cool), Forecast (Same, Change), Es (Yes, No).

• Tabella:

SUNNY	WARM	NORM	STRONG	WALK	SAME	YES
SUNNY	WARM	HIGH	"	"	"	YES
RAINY	COLD	HIGH	"	"	CHANGE	NO
SUNNY	WARM	HIGH	"	COOL	"	YES

- Ipotesi: ma ipotesi  $h$  è una congiunzione di attributi e valori. Ogni valore può essere su d., ?, o Ø.

↳  $\langle \text{Sunny}, ?, ?, \text{Strong}, ?, \text{Same} \rangle$  è esempio

- Target Function:  $f: \text{EnjoySports} \times \{0, 1\}$

- Training Examples: eserci: posizioni/meglio/worst della Target function in forma  $\langle x_1, f(x_1) \rangle, \dots, \langle x_n, f(x_n) \rangle$

↳ Il numero di ipotesi cresce esponenzialmente rispetto al numero di features!

- Vapnik-Chervonenkis (VC) dimensione: misura della capacità di un modello da un algoritmo di classificazione binaria: Definito come la cardinalità del più grande set di punti che tale algoritmo può SPACCARE (ovvero, dividendo i punti nelle due regioni definite x cluse)

## SPAZIO DELLE IPOTESI

• Ordine delle ipotesi:

$\rightarrow h_j \geq h_k : \text{sse } \forall x \in X: [(\underbrace{h_k(x) = 1}_{= 1})] \xrightarrow{\text{relazione di ordine PARZIALE su } H}$

•  $k, \text{TRAD.}$  "  $h_k$  classifica  $x$  come positiva"

•  $k_2, \text{TRAD.}$  "  $h_{k_2}$  è più generale (o uguale) a..."

• Find-S Algorithm:

1.  $h = \text{Most\_specific\_hypothesis}(H)$

2. for  $x$  in Only\_positive\_training\_instance:

    for  $a_i$  in  $h$ : // attribuire constraint  $a_i$ :

        if is\_satisfied( $a_i$ ,  $b_i$ :  $h$ ):

            do nothing()

    else:

        replace ( $a_i$ , Next\_more\_general())

return  $h$

↳ Spiegazione: find-S parte delle ipotesi + specifiche ed ogni volta controlla che i vincoli dagli attributi coincidano con le ipotesi, altrimenti va verso il più generale, nella speranza che vada meglio.

## ↳ Proprietà:

- ① restituirà in aifut la più specifica  $h \in H$  t.c. essa è ancora consistente con gli esempi di Training
- ②  $h$  sarà anche consistente con gli esempi negativi (se il target concept  $\subseteq h$ )

## ↳ Problemi:

- ① Non sappiamo se il learner converge o meno con il Target concept (trova una ipotesi, ma tutte!)
- ② Non sappiamo dire se gli esempi di Training sono incostistenti, dato che ignorano i negativi
- ③ Prendiamo sempre l'ipotesi più specifica, anche quando facciamo avere una più generale...

- Version Space: un set di ipotesi valide e consistenti con i dati:  $VS_{H,D} = \{h \in H \mid Cons(h, D)\}$  dove  $Cons(h, D) := \forall \langle x, c(x) \rangle \in D \quad h(x) = c(x)$
- List-Then-Eliminate Algorithm: algoritmo che, dati tutti i version space, rimuove da essi ogni ipotesi inconsistente con gli esempi di Training.

- General boundary:  $\underbrace{\text{set di membri di VS più generali.}}$   
 $(G)$
- Specific boundary:  $" " " " " "$  specifici.  
 $(S)$

Ogni membro di VS giace tra questi due boundary!

### Candidate Elimination Algorithm:

$G, S = \text{set di ipotesi più gen. e specifiche}$   
for  $d$  in examples:

if is-positive ( $d$ ):

$G.\text{pop}(\text{inconsistenti (con: } d, \text{da: } G))$

for  $s$  in inconsistenti (con:  $d$ , da:  $S$ ):

$S.\text{pop}(s)$

$S.\text{add}(\text{minimal\_generalization}(h, of: } S))$

// Tale che qualche  $g \in G$  ( $g \geq h$ )

$S.\text{pop}(\text{more\_general\_than\_another}(h))$

else:

$S.\text{pop}(\text{inconsistenti (con: } d, \text{da: } S))$

for  $g$  in inconsistenti (con:  $d$ , da:  $G$ ):

$G.\text{pop}(g)$

$G.\text{add}(\text{minimal\_specialization}(h, of: } G))$

// Tale che qualche  $s \in S$  ( $s \leq h$ )

$G.\text{pop}(\text{less_general_than_another}(h))$

// Converge alla stessa risoluzione del VS mantenendo l'ordine

• Unbiased learner: sceglio un set di ipotesi H.T.C. esclusa ogni possibile concetto INSEGNABILE, avendo che  $H = \text{Power-set}(X) \rightarrow$  totalmente insensato, dato che mai è razionale avere un learner che non faccia omissioni sull'ideazione del target concept: non riuscirebbe mai a costruire ipotesi mai viste prima!

• Judicious Bias: trasforma un sistema induttivo in un sistema deduttivo che trae le sue concl.

• Pari:

$L$  = algoritmo di concetti learning

$X$  = insieme

$C$  = target concept

$L(x_i, D_c)$  = la classificazione dell'esempio  $x_i$  fatta da  $L$  dopo il training su  $D_c$

• Allora:

l'inductive bias di  $L$  è il set minima di afferzioni B.T.C.  $\forall c$  su  $D_c$ :

$$\underbrace{(\forall x_i \in X) [B \wedge D_c \wedge x_i]}_{\text{Premessa}} \vdash L(x_i, D_c) \quad \underbrace{\downarrow}_{\text{Conclusione}}$$

"segue logicamente che..."

- learners e biases:

- ① Rule Learner: conserva gli esempi, che riconosce se già li ha già visti e li associa con la sua memoria.  
↳ No indicative bias, Non generalizza
  - ② VSC algorithm: qui il bias è rappresentato dal fatto che  $H$  contiene il target concept  $c$ : classifico tutte i membri del VS con d'accordo.
  - ③ Find-S: il bias è rappres. del fatto che  $H$  contiene  $c$  e le  $h$  negative, a meno che l'opposto è entailed dalla sua conoscenza o non valida entailed dalle ipotesi imparate.
- 

## TREE MODELS

- Vantaggi:

- Vantaggi:
  - Mativamente auto-descrittivi
  - Facilmente generalizzabili ( $b \rightarrow K$  classi)
  - Feature Tree  $\rightarrow$  Decision Tree

- Decision Tree: modello ad albero che si ottiene da un feature tree potuto (corta overfitting / buone misure computazionali), modelli più

espressive, migliori nelle predizioni.

→ Qui D.T. corrisponde ad una espressione logica  
Simplificata in DNF → bisogna limitare perciò gli  
esempi per far sì che l'albero generalizzi bene:

[ + esempi == + over fitting ug

→ Per questo introduciamo:

- BiAS, oppure
- depositanti allo il linguaggio (-espressività):  
induttive BiAS.

- Feature Tree: albero cui nodi sono etichettati con  
feature e archi con intervalli, ciascuna foglia è  
un'espressione logica.
- Split: insieme dei intervalli dell'arco di un nodo
- Seguenza dello spazio delle istanze: l'insieme delle  
istanze coperte nel percorso radice-foglia (path.)
- Omogeneità: su dataset  $D$  è omogeneo a  
secondo del Task e del dominio: nel caso multi-  
-classe  $D$  è omogeneo se tutti gli esempi appartengono  
- alla stessa classe.
- Algoritmo di costruzione del Feature Tree:

- InPut: dataset  $D$ , feature set  $F$ .

• Output: feature tree  $T$  con foglie etichettate.

\* if  $\text{DenseTree}(\Delta)$ :

    return Label( $\Delta$ )

// se sono dense si possono etichettare i dati

$S = \text{Best-Split}(\Delta, F)$  // migliore suddivisione per  
// il dataset in quel dato istante.

$\Delta_i = \text{split\_dataset}(\Delta)$  // divide il dataset  $\Delta$  in

// sottogruppi in accordo con lo split  $S$ !

for  $i$  in range( $\text{len}(\Delta_i)$ ): // per ogni partizione  $i$  di  $\Delta_i$

    if  $\text{len}(\Delta_i) \neq 0$ : // controllano che non sia vuota

$T_i = \text{new\_branch}(\Delta_i, F)$  // NON VUOTA: trovato

        // nuovo albero su cui applicare ricorsivamente

        // questo algoritmo

    else:

$T_i = \text{new\_leaf}(\text{Label}(\Delta))$  // VUOTA: creiamo una

        // nuova foglia etichettata

\* Return  $T$  // un albero con alla radice  $S$  e  $T_i$  come  
// figli della radice

↳ Algoritmo greedy: non ritrova un albero ottimale

↳ Righe con \* garantiscono la terminazione

## DECISION TREE

- Purezza di uno split:

① Empirical Probability:  $p = \frac{n^+}{n^+ + n^-}$  ← esempi positivi  
← esempi totali

② Classe Majoritaria:  $1 - \max(p)$

③ Gini Index: nelle Bernoulliane è la  $\text{Var}(p)$   
 ↳ caso binario:  $2p(1-p)$  nelle roulette è l'errore atteso  
 ↳ " generico:  $\sum p_i(1-p_i)$   
 ↳ greeco δ usa la sua RADICE + accurato!

④ Entropia:

↳ caso binario:  $-p \log_2(p) - (1-p) \log_2(1-p)$

↳ caso generico:  $-\sum p_i \log_2 p_i$

Rappresenta la quantità, in bit, di informazione che possi trasmettere dato un evento:

• lucio di una moneta:  $-\frac{1}{2} \log_2 \frac{1}{2} = 1 \text{ bit}$

Dall'entropia ricaviamo l'information gain:

entropia

$$I(E) = -\log_2 p$$

$$\overbrace{H(E)}^{\text{averagine inf. gain}} = -\sum_{i=1}^n p_i \log_2 (p_i)$$

averagine inf. gain

Tecnicamente è la somma di quantità di impurità per una partizione del dataset.

- esempio di calcolo  $E$  per una feature:

...

$$\text{length} = [3, 4, 5] \rightarrow [2+, 0-] [1+, 3-] [2+, 2-]$$

$$-\left(\frac{1}{3}\right) \log_2 \left(\frac{1}{3}\right) - \left(\frac{3}{5}\right) \log_2 \left(\frac{3}{5}\right) = 0.81 \quad 1$$

$$E_{\text{tot}} = \frac{2}{10} \cdot 0 + \frac{4}{10} \cdot 0.81 + \frac{4}{10} \cdot 1 = 0.72 \text{ (split bruciato...)}$$

- Algoritmo BestSplitClon ( $\Delta, F$ ):

$I_{\min} = 1$  // impurità minima, fissa da 1

for  $f$  in  $F$ : // per ogni feature e feature set...

$\Delta_i = \text{SPLIT\_dataset} (\Delta, \text{given} : \text{This. } f. v_f)$

if  $\text{impurity} (\Delta_i) < I_{\min}$ :

$I_{\min} = \text{impurity} (\Delta_i)$  // teniamo l'impurità min

$f\_best = \text{This. } f$  // e la sua feature!

return  $f\_best$  // feature su cui fare lo split

Tipi di albero:

## RANKING TREE

- Albero su cui si ordinano le probabilità euristiche sulle foglie

## STIMATORE DI PROBABILITÀ

- Prende la p. di una foglia per rendere la stima più robusta anche x le foglie con pochi esempi; con la Laplace Correction o la correzione delle stime-n.

## CUIA CLASSIFICAZIONE

Trovare il punto operativo attuale della ROC curve utilizzando  $\text{cLR} = \frac{\text{Pos}}{\text{Neg}}$  e cost ratio  $c = \frac{C_{FN}}{C_{FP}}$ , lo fa trovando il punto di intersezione delle curve isometriche con la retta  $\frac{1}{c} \cdot \text{cLR}$ .

- Algoritmo di ricavare feature tree: PruneTree( $T, D$ )

for  $N$  in  $T$ : // per ogni nodo interno  $N$  del  $T$ , ricavando  
 $T_N = \text{subtree}(T, \text{root}=N)$  della fine dei francesi...  
 $D_N = K\text{-covered}(N)$  //  $\{x \in D \mid x \text{ generato da } N\}$   
if accuracy( $T_N, \text{over}=D_N$ ) < majority-class( $D_N$ ):  
     $T.\text{replace}(T_N, \text{new\_leaf}(\text{label}=\text{majority-class}(D_N)))$   
return  $T$  // ritorna la variabile pruned di  $T$

- Scilla della generalizzazione degli errori:

→ utilizzata per poi non dover fare pruning dopo ma per controllare l'albero e controllare se over.

Sarà o meno. Abbiamo diversi modi di applicare:

① Penalità κ. A foglia che non permette di creare un'altra foglia se l'errore del modo padre non diminuisce di almeno  $k+1$ . Modo COMPLESSO.

② Utilizzo di una formula di fine l'errore sul test set (a partire dal training set).

$$\rightarrow \text{Errore sul training set} \quad \bar{E}_{Tr} = \sum_{i=1}^N e_i^* \quad \left[ \begin{array}{l} \text{errori sulla} \\ \text{foglia } i\text{-esima} \end{array} \right]$$
$$\rightarrow \text{Errore sul test set} \quad \bar{E}_{Te} = \sum_{i=1}^N e_i^* \quad \left[ \begin{array}{l} \text{m° di foglie} \\ \downarrow \end{array} \right]$$

Approssimato, dunque:  $\bar{E}'_{Te} = \bar{E}_{Tr} + N * 0.5$

penalità per foglia

Esempio: con 30 foglie, 10 errori, 1000 istanze totali del training set  $\rightarrow \bar{E}_{Tr} = (10/1000) = 1\%$   
e  $\bar{E}'_{Te} = (10 + 30 * 0.5) / 1000 = 2.5\%$ .

↑ anche detto ERRORE DI GENERALIZZAZIONE

Nota: si può "gaufiare" manualmente su training set per valutare meglio suo spl:it in condizioni di nuove (gaufare significa però aumentare i tempi di addestramento di tanti poco).

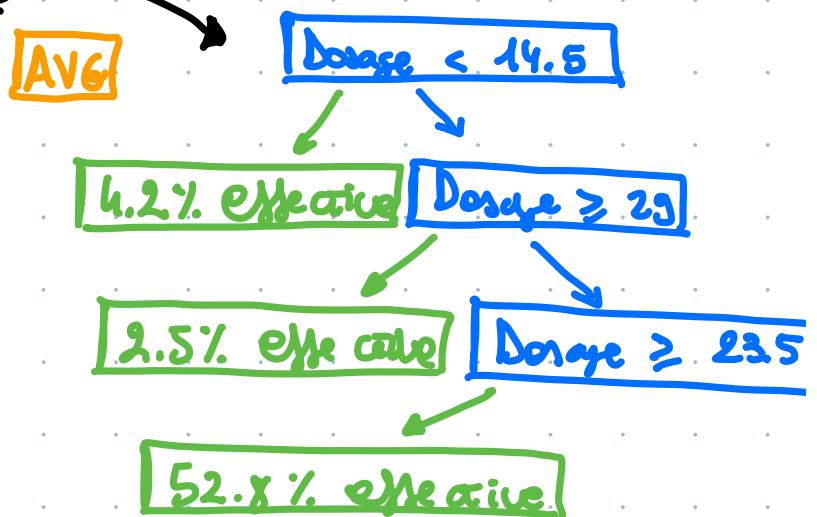
- IMPURITÀ RELATIVA: rapporto tra l'impurità del nodo padre e quella del figlio.

$$\rightarrow \frac{m_1^+ m_2^- / m_2}{m^+ m^- / m} \quad | \quad \sqrt{\frac{m_1^+}{m^+} \frac{m_1^-}{m^-}} \quad / \quad \begin{array}{l} \text{IMPURITÀ} \\ \text{RELATIVA} \\ \text{GINI INDEX} \end{array}$$

Questo valore non cambia se moltiplichiamo tutti i valori della classe positiva per un fattore c: esso darà ancora i nodi che coprono + esauri con la radice del gini index.

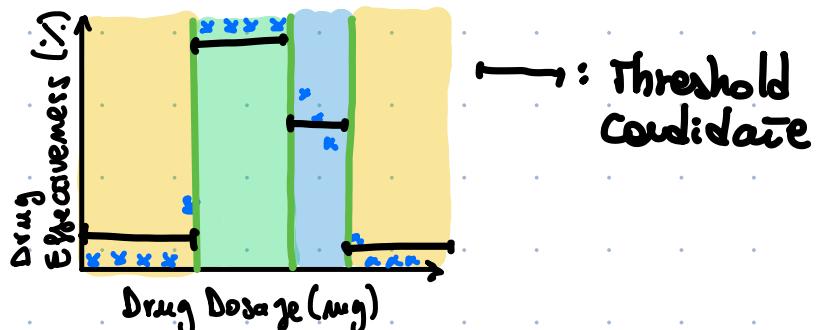
- Ricetta di Peter Flach per il learning di un dec. tree:
  - 1) Generare un feature tree in grado di fare branching sugli esempi;
  - 2) Usare la radice del gini index come misura di impurità;
  - 3) Per fare stime probabilistiche, occorre fare correzioni come Laplace Correction o m-estimate per robustificare i dati con pochi esempi!
  - 4) Individuare il miglior punto x il threshold o per cui assegnare le label.
  - 5) Fare pruning sui rami delle foglie con le stesse label

- Alberi di regressione: tipo di decision tree con feature numeriche sulle foglie, utilizzati quando non solo binari non basta per filtrare i dataset! Esempio:



- Per calcolare **AVG** per ogni gruppo poniamo utilizzare una misura di impurità (relativa) come squared Gini index / entropy.

- Questi threshold che dividono lo spazio di regressione sono euristici: misurano la somma dei residuals (distanza tra Threshold e punto) al quadратo.



• Apprendimento di alberi per riduzione di varianza:

→ Prendiamo ad esempio una variabile Bernoulliana:

$$\text{Var}(y) = p(1-p) \quad [\text{mezza del valore binario!}]$$

Mettiamo in problemi di regressione:

$$\text{Var}(y) = \frac{1}{|y|} \sum_y (y - \bar{y})^2$$

e la varianza media pesata se  $y$  viene suddiviso:

$$\text{Var}(y_1, \dots, y_r) = \sum_{j=1}^r \frac{|y_j|}{|y|} \text{Var}(y_j) = \dots = \sum_y y^2 - \underbrace{\sum_{j=1}^r \frac{|y_j|}{|y|} \bar{y}^2}_{\substack{\text{costante} \\ \text{Termine da min}}}$$

→ La misura di impurità dovrà utilizzare dunque  $\text{Var}(y)$  per minimizzare la distanza quadratica media attorno al centroide  $\bar{y}$  (vogliamo particolari BEN DISTANCE)

→ Label(y) ritorna il valore medio presente identifico delle foglie

→ NaiveBayes(y) ritorna vero se  $\text{Var}(y)$  appartiene ad un certo threshold.

→  $\text{Dis} : X \times X \rightarrow \mathbb{R}$  misura la dissimilità tra due elementi  $x_1$  e  $x_2 \in X$ . Per esempio per misurare tra cluster:

$$\text{Dis}(D) = \frac{1}{|D|^2} \sum_{x_1} \sum_{x_2} \text{Dis}(x_1, x_2)$$

e, generalmente (ma clusters)

$$\text{Dis}(D_1, \dots, D_i) = \sum_{j=1}^i \frac{|D_j|}{|D|} \cdot \text{Dis}(D_j) \left\{ \text{da max} \right.$$

### • Alberi di clustering:

utilizziamo la Dis. come funzione obiettivo; ovvero,  
la dis. media tra tutti i suoi clusters.

↳ Bassa dissimilità + cluster piccoli: **ABRATTIBI**  
Venne applicato ciò del PRUNING e rimozione  
degli outliers!

↳ come scegliere le label? Scelta del medioide,  
ovvero l'esempio con dis. totale minore.

→ Usate la distanza euclidea per calcolare la dis:

$$\text{Dis}(x_1, x_2) = \sum_{i=1}^d (x_{2i} - x_{1i})^2$$

L'iscrizione poi la Var per un set di N oggetti:

$$\text{Var}_i(\chi) = \frac{1}{N} \sum_{j=1}^N [\vec{x}_{ji} - \underbrace{\vec{x}_i}_{\text{centroide } \vec{x}_i}]^2$$

Dunque la somma delle var. come la dist. euclidea quadratica media tra i vettori dello spazio d-dim.

$$\sum_{i=1}^d \text{Var}_i(\chi) = \frac{1}{N} \sum_{i=1}^d \sum_{j=1}^N [\vec{x}_{ji} - \vec{x}_i]^2 = \frac{1}{N} \sum_{j=1}^N (\vec{x}_j - \vec{x})^2$$

Moltre la Var equivale alla media della distanza quadratica media tra ciascuna coppia di oggetti:

$$\text{Dis}(\chi) = \frac{1}{N^2} \sum_{\vec{x}'} \sum_{\vec{x}''} (\vec{x}' - \vec{x}'')^2 = 2 \text{Var}(\chi)$$

e  
 $\text{Var}(\chi) = \sum_{i=1}^d \text{Var}_i \chi$

Conclusioni: possiamo calcolare la dis. greve del vettore delle medie delle istanze  $\vec{x}$  e confrontando le singole varianze  $\text{Var}_i(\chi)$ .

## MODELLO A REGOLE (ordinate)

- Scopo: apprendere una regola alla volta che identifica un segnale nello spazio delle istanze e la cui conseguenza sia la label della classe predetta dal modello. Si dovrà rifare il learning ogni volta eliminando gli esempi individuati dalla regola corrente per generare spazi diversi ogni volta → generata poi una lista di regole ordinate
- Ci interessa la purezza di un solo figlio (e non di tutti come nei modelli ad albero)
- Algoritmo di generazione di regole ordinate:

$R = [] // [] = \emptyset$

while  $D \neq [] //$  fino a quando gli esempi non finiscono

$r = \text{LearnRule}(D) //$  impara la regola

$R.\text{apprend}(r) //$  appresa alle liste di  $R$

$D = \text{new\_d}() // \{x \in D | x \text{ is covered by } r\}$

↑ divide e si impara

return  $R$

- Algorithm Learntule (D):

$b = \text{true}$  // il body iniziale della regola

$L = \text{available\_literals}()$

while not homogeneous ( $D$ ): // seguendo la purezza

$l = \text{best\_literal} (D, L)$  // quello con purezza maggiore

$b = b \text{ and } l$  // CNF

$D = \text{new\_d}()$  //  $\{x \in D \mid x \text{ is covered by } b\}$

$L = L - \text{same\_feature\_literals}(L)$  // andiamo ad

// eliminare i letterali che usano le stesse feature

$C = \text{Label}(D)$

$r = f'' \text{ if } \neg b \text{ Then } \text{class} = \{C\}$  // scriviamo la regola

return  $r$  //  $r = \text{Antecedente + Consequente}$

- Per calcolare l'errore totale di ranking usiamo la "solita" formula:

$$\text{rank\_err} = \sum_{x \in T, x' \in T} \left( \frac{I(s(x) < s(x')) + 0.5 I(s(x) = s(x'))}{\text{Pos} + \text{Neg}} \right)$$

Che voluta 1 una misclassificazione è  $\frac{1}{2}$  un tie!

- Modelli a regole NON ORDINATE:

- 1) Si seleziona una cluse
- 2) Si seleziona il body che copre il maggior numero di cluse selezionate da ① tale da avere la maggior precisione ( $TP / (TP + FP)$ ) possibile.

Non ci interessa l'ordinamento doco che ogni regola diventa estremamente precisa sulla sua cluse considerata

- Algorithm Learn Rule Set ( $D$ ): (Mordere le rule fin  $R$ ) as output:

$$R = []$$

for  $c$  in  $C$ : // A cluse per una introduzione del bias,  
// che stabilirebbe il dataset

$D_i = D$  // faccio una copia del dataset

while  $D_i$ . contains (c.examples):

$r = \text{Learn Rule For Clue } (D_i, c)$  // lo vediamo ora...

$R = R + \{r\}$

$D_i = D_i - \{x \in c \mid x \text{ is covered by } r\}$

// rimuovo dal dataset tutti gli esempi positivi

// appena coperti dalla regola  $r$ !

return  $R$

- Algorithm Learn Rule For Clue ( $D, C_i$ ):

// imparare una sola regola per una cluse data

uguale all'algoritmo leonardo ma con:

$l = \text{BestLiteral}(D, L, C_i)$  invece di

$l = \text{BestLiteral}(D, L)$ , ovvero il focus è sulla classe!

- Valutazione probabilistica: oltre alla precisione ci interessa considerare leggi valide per una generalizzazione del problema  $\rightarrow$  introduciamo

- 1) la combinazione di Laplace  $\rightarrow$  preferenze regole più generali.
- 2) Ricerca Bayes invece che greedy

- Modello a regole non ordinate vs ordinate:

- Il modello M.O. raggruppa sottoinsiemi dello spazio delle istanze che gli a. non raggruppano
- Il modello N.O. permette di spartire lo spazio delle istanze in 2<sup>n</sup> segmenti, invece che N come in O.

- Dauardo: cosa predirebbe un modello se 2 regole diverse fossero soddisfatte contemporaneamente ma predicessero cose totalmente opposte?

- ↳ andremo a scegliere quella con la maggiore likelihood (prob. massima)
- Subgroup Discovery: produzione di ipotesi verificate in sottosinsiemi dello spazio delle istanze per utilizzarle in audizioni relazionali del tipo sottogruppo / popolarezione intera
  - ↳ funzione di mappatura
  - ↳ Sottogruppo:  $\hat{g}: X \rightarrow \{\text{true}, \text{false}\}$
  - appresi da ↳
  - Esempio di esempi:  $(x_i, l(x_i))$  dove  $l = \text{label}$
- I migliori sottogruppi sono quelli: dove le dist. delle classi è statisticamente differente dalla pop. di Panacea (originale)
- ↳ Per valutare i migliori sottogruppi utilizziamo l'averagge recall  $((TP + TN) / 2)$ , secco questo valore non coincide con le precisioni.
  - ↳  $TP / (TP + FP)$
- Un modo per apprendere + regole è quello di non eliminare gli esempi già corretti ma di decrementare il peso e loro associato + utilizzare un'unistica di ricerca valutata sulla somma

dei pesi, vediamo ora l'algoritmo che lo fa:

- Algorithm WeightedCovering ( $\Delta$ )  $\rightarrow$  Luke list  $R$

$R = [] // R = \emptyset$

while some examples in  $\Delta$  have weight 1:

// tutti gli esempi dovrebbero essere coperti da almeno  
// una regola... è una condizione sufficiente!

$R.append(r)$

examples.decrease\_weight(covered: r)

return  $r$   $\uparrow //$  riduce il peso degli esempi coperti da  $r$

- Item set: Regole che associano insiemi di oggetti diversi a feature booleane per identificare la presenza.

↳ Un item set  $I$  è un insieme di oggetti  $i_1, \dots, i_n$  t.c. ogni  $i_x \in I$  appartiene alla stessa trasformazione  $T$ .

↳ Un frequent item set  $I$  è un item set ... t.c ... con  $|I| \geq f_0$  (supporto minimo stabilito)

↳ Un closed item set è un item set tale che non esiste un altro item set  $I_2 \subseteq I$  con lo stesso supporto (avendo,  $f(I_2) = f(I)$ )

↳ Un item set maximale I è chiuso e frequente e t.c. non esiste un  $I_2$  t.c.  $I \subset I_2$  con supporto sufficiente. Ovvero, sono venuti fuori al fondo inferiore del grafo, scrivendo item frequenti da quelli meno frequenti.

↳ Proprietà item sets:

- ① Scendendo di livello il supp. decresce, dato che è massimo.
- ② L'insieme degli item set freq. è convesso.
- ③ La proprietà ① permette di cercare tutti gli item set frequenti con un algoritmo di ricerca in avanti/indietro nel grafo.

→ Ci interessa cercare l'item set maximale dove c'è un'ottima generalizzazione degli item sets.

• Algoritmo Frequent Itemset (D, f<sub>0</sub>):

// trovare tutti gli I massimali che eccedono una certa soglia threshold f<sub>0</sub> ↳ M

$$M = [] // M = \emptyset$$

Q = Priority Queue ([I]) // contiene gli item set

// quando non è vuoto!

while  $\text{iteu}(Q) > 0$ : // fino a quando qualcosa è vuoto.

$I = \max_{i \in \text{iteu-set}}(Q)$  // L'IS cancellato dal sop di Q

max = True // max ci indica se I è massimale

for  $I'$  in powerset(I):

if  $\text{supp}(I') \geq f_0$ :

max = False // Trovato item set frequente

// dunque I non è massimale

$Q.\text{append}(I')$  //  $I'$  alla fine di Q

if max:

$M = M + \{I\}$  // item set massimale aggiunto a M

ritorna M

### Differenze tra item sets:

1) Gli item sets chiusi rappresentano classi di equivalenza degli item set con lo stesso supporto.

2) L'estrazione degli item set frequenti non produce perdita di informazione degli item set con suff. sufficiente ma generalmente risultati.

3) L'estrazione dei massimali riduce la mole di molto ma al prezzo della perdita di info sulle freq. dei frequenti ma non massimali.

4) Considerare gli i.s. chiusi: può essere utile confrontarli per trovare info. su frequenti ma non chiusi a causa del loro supp. equivalente.

- Association Rules: regole della forma "if  $B$  then  $H$ " dove  $\{B, H\}$  sono item set freq. nelle stesse transazioni.
  - ↳ viene valutata la confidence di una regola: mira di quanto si può essere fiduciosi circa la presenza del conseguente in transazioni con l'antecedente:
    - regole con conf = 1 sono le regole che uniscono i due insiemi chiusi (perché il secondo ha lo stesso supporto del superset, quindi il rapporto è = 1)

↳ Generazione A.R.: si cercano gli I.S. freq., si seleziona il corpo e la testa delle regole di ciascun insieme freq. su, si escludono quelle sotto il threshold  $f_0$ .

↳ Algoritmo Association Rules (D, f<sub>0</sub>, C):

↓

$R = []$

$H = \text{FrequentItem}(D, f_0)$

for  $m$  in  $H$ :

for  $([H \text{ in } m] + [B \text{ in } m]) \text{ t.c. } H \cap B = \emptyset$ :

if  $\text{supp}(B \cap H) - \text{supp}(B) \geq c_0$ :

$R = R + \text{"if } B \text{ Then } H"$

→ Migliorare l'algoritmo:

- 1) Invece di considerare  $H \subseteq m$  e  $B \subseteq m$  t.c.  $H \cap B = \emptyset$  potremo considerare un ordine sulla cardinalità di  $H \rightarrow$  prius i set minimi con card. minore.
- 2) Inizializzare a valori regole con confidenza maggiore e quando questa inizia a scendere si può ammettere le nuove.
- 3) per valutare la buona delle regole produrre potremo utilizzare il lift:

$$\text{Lift}(\text{if } B \text{ Then } H) = \frac{n \cdot \text{supp}(B \cup H)}{\text{supp}(B) \cdot \text{supp}(H)} \quad \begin{cases} \text{se } \text{den} > \text{num}: \\ \text{e' piu' robusto} \\ \text{che altrimenti} \end{cases}$$

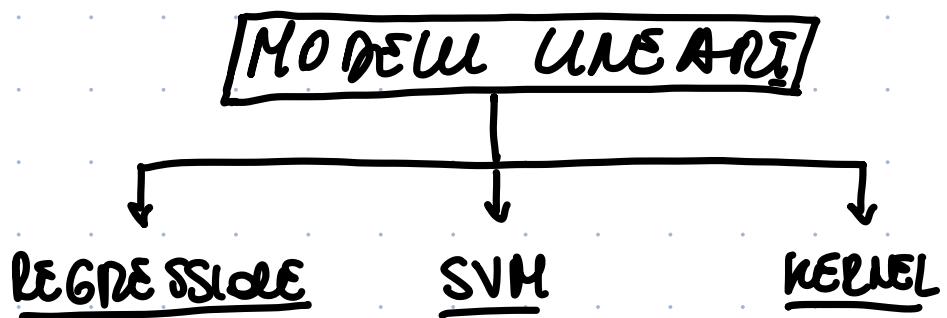
Dove  $n = \text{numero di Transazioni}$ .

Notiamo che:

→ il lift è transitivo:  $\text{lift}(\text{if } B \text{ Then } H) = \text{lift}(\text{if } H \text{ Then } B)$

→  $\text{Lift}(\text{if } B \text{ Then } H) = 1$ , se non che  $\text{supp}(B \vee H)$  è interamente determinato dalle frequenti marginali  $\text{supp}(B)$  e  $\text{supp}(H)$ : solitamente ci interessano  $\text{Lift}(\dots) > 1$ .

→ Si scopre che se usiamo i reg. set chiusi nel corso e nella testa delle regole si ottengono regole + significative.

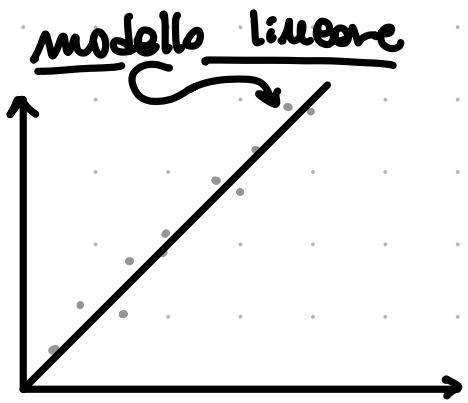


- Regessione: modello definito geometricamente come una retta su un piano che approssima un insieme di coordinate.

↳ Capitolo: Trovare i valori  $C, D$  per definire:  
 $(Cx + D = y)$

Dove  $\Delta$  rappresenta il bias e permette un vettore (1) davanti: cioè a sinistra di farizzare la retta a passare per l'origine. Il tutto viene messo a sistema del tipo:

$$\begin{cases} Cx_1 + \Delta = y_1 \\ Cx_2 + \Delta = y_2 \\ \vdots \\ Cx_m + \Delta = y_m \end{cases}$$



Che per necessità computazionali trasformiamo in una matrice della forma

$$Xw = y \quad \left\{ \begin{array}{l} X \text{ contiene i coeff di } C, \Delta \\ w \text{ rappresenta } C, \Delta \\ y \text{ è il loro prodotto} \end{array} \right.$$

$$X = \begin{pmatrix} 1 & 1 \\ x_1 & x_2 \end{pmatrix} \begin{pmatrix} C \\ \Delta \end{pmatrix} = Cx_1 + \Delta x_2 \quad \left\{ \begin{array}{l} \text{Variando il vettore } w \\ \text{vado a cercare tutti} \\ \text{i possibili valori nello} \\ \text{spazio vettoriale.} \end{array} \right.$$

- Per trovare il valore di  $w$ :  $w = X^{-1} y$   $\leftarrow$  inversione  
Ma non posso invertire una matrice non quadrata!

(Nella fig. sugli appunti si vede meglio dato che

$p = \chi \hat{w}$  appartiene al piano  $C(\chi)$  [combinazione lineare dei vettori  $x_1, x_2$ ] mentre  $y$  si allontana da esso)

- La distanza tra  $y$  e  $p$  è  $e = y - p$ , un vettore SEMPRE perpendicolare al piano. Quindi con  $\chi^T e = 0$  abbiamo la sua perpendicolarità.
- Metodo dei minimi quadrati: trovare il vettore  $p$  sul piano tale per cui  $e$  è minima  $J$

$$\|e\|_2 = \|y - p\|_2 = \sqrt{\sum_{i=1}^n (y_i - p_i)^2}$$

e a questo punto trovare il modello lineare si tratterà solo + di minimizzare la somma delle diverse dei quadrati delle componenti di  $y$  e  $p$ :

$$\text{minimize}_{\hat{w}} \|\underbrace{\chi \hat{w} - y}_p\|_2^2$$

Ma visto che il vettore  $e$  deve essere perpendicolare al piano  $C(\chi)$  siamo costretti ad aggiungere il vettore seguente:

$\chi^T e = 0 \leftarrow$  il prodotto scalare dei due vettori è

$$\text{se } \chi = \begin{pmatrix} 1 & 2 & 3 \\ 0 & -1 & 2 \end{pmatrix}$$

allora  
 $\chi^T = \begin{pmatrix} 1 & 0 \\ 2 & -1 \\ 3 & 2 \end{pmatrix}$

$$\chi^T(y - \chi\hat{w}) = 0$$

$$\chi^T y - \chi^T \chi \hat{w} = 0$$

$$\chi^T \chi \hat{w} = \chi^T y$$

non possiamo  
moltiplicare a sinistra  
dato che la mtrice  
matriciale non  
è commutativa!

$$\hat{w} = \underbrace{(\chi^T \chi)^{-1}}_{\chi^T \chi \text{ è quadrata!}} \chi^T y$$

$\chi^T \chi$  è quadrata!

- Il problema del metodo dei minimi quadrati è che anche un singolo outlier può influenzare negativamente la nostra regressione: se si va a vedere la sua loss è chiaro notare che l'errore cresce in modo quadratico in prop. alla distanza dal modello

↳ Soluzione: regressione regolarizzata

- Regressione ridge:

$$w^* = \underset{w}{\operatorname{argmin}} (y - \chi w)^T (y - \chi w) + \lambda \|w\|_2^2$$

↳ Ricordo che:

1)  $(y - \hat{y}_w)^T(y - \hat{y}_w) = \|\hat{y}_w - y\|_2^2$ , è solo una iscrizione!

2) Ciò vuol dire che aggiungiamo solamente il termine penalizzante  $\lambda \|w\|_2^2$  che penalizza la soluzione al crescere di  $w$ .  
↳  $\lambda$ : fattore multipli continuo compreso tra 0 e  $+\infty$

{ se 0: regressione classica

~

se  $\infty$ : ignoriamo tutto!

↳ Soluzione in forma chiosa:  $\hat{w} = (X^T X + \lambda I)^{-1} X^T y$   
(uguale a quelle precedenti se non per l'aggiunta del termine  $\lambda I$  che regole le regolarizzazioni dove  $I$  = matrice identità)

↳ "aggiunta di  $\lambda$  alla diagonale di  $X^T X$ "

• Regressione lasso: (least absolute shrinkage and selection operator) → si usa la norma 1 invece del quadrato della norma 2.

$\|w\|_1 = \sum_i |w_i|$  è basicamente la somma dei valori abs delle componenti

Riduce a 0 numerosi pesi, favorendo soluzioni SPARSE

- Regolarizzazione con la minimizzazione della norma del vettore  $w$ :

se partiamo dall'assunzione che  $X$  sia affetta da un errore specificato in  $D$ , che cosa gli outliers abbiano quindi

$$(X + D)w = Xw + \underbrace{Dw}_{\downarrow}$$

Quindi maggiore è la norma di  $w$ , maggiore è l'incertezza del fattore  $Dw$  che descrive l'errore puro sul risultato finale: da qui capiamo facilmente come minimizzare la norma del vettore  $w$  sia una buona cosa.

→ Principio del rosario di Occam dice che ha + senso scegliere vettori di per sé + piccoli, e questo viene espressamente cercato con la regressione ridge e lasso.

→ Minimizzare la norma di  $w$  è come introdurre un BIAS per l'algoritmo di apprendimento, dunque chi più autore di questo bias, la varianza dei dati si riduce!

$$\Gamma E = b^2 + V + \dots$$

↓

Nel corso delle f. ridge e lasso come visto prima la min. della norma del vettore  $w$  si traduce in un aumento del quadrato del bias  $b^2$  che riduce la varianza  $V$ .

• Confronto tra f. di regressione: ovvero, confronto

- 1) Regressione con metodo dei minimi quadrati
- 2) Regressione ridge, ovvero regressione ① regolarizzata con il quadrato della norma 2 di  $w$
- 3) Regressione lasso, ovvero la ① regolarizzazione con la norma 1 di  $w$ .

★ Sezione da guardare su appunti (principi di base)

↳ L'algoritmo di apprendimento mantiene + conveniente ridurre le features nell'intervallo  $[-1, 1]$  Per minimizzare la penalizzazione delle f. quadratiche, all'interno di questo intervallo non è più molto conveniente costruire a 0 i valori compresi fra -1 e 1 come facendo la lasso.

• Classificazione con metodo dei min. quadrati:

$$\hat{C}(x) = \begin{cases} 1 & \text{se } x^T \hat{w} - \tau > 0 \\ 0 & \text{se } x^T \hat{w} - \tau = 0 \leftarrow \text{"margine" } \\ -1 & \text{se } x^T \hat{w} - \tau < 0 \end{cases}$$


---

- Support Vector Machines: "macchine" con lo scopo di trovare l'iperpiano, ovvero la linea che separa in due le classi.

→ Margine: distanza fra l'esempio e l'iperpiano

↳ Funzionale: atti a ridurre errori

↳ Geometrico: atti a separare classi

⇒ Margine Funzionale:  $y_i (w x_i - \tau) > 0$

entro questo  $\rightarrow$  soltanze della classif.,  
deve avere la stessa classe (1 · 1)  
per essere  $> 0$ !

↳ Allora entri in valore E T.C. valga le prof. :

$y_i(\omega x_i - \tau) \geq \varepsilon$  DIMOSTRAZIONE:

$$C \cdot y_i(\omega \cdot x - \tau) \geq \varepsilon \cdot C \xrightarrow{(1)} \frac{C}{\varepsilon} y_i(\omega x - \tau) \geq \frac{\varepsilon}{\varepsilon} C \xrightarrow{(2)} y_i\left(\frac{C}{\varepsilon} \omega x - \frac{C}{\varepsilon} \tau\right) \geq C$$

↑  
mult. da entrambi lati  
per una cost. C

Si noti inoltre che l'esempio + vicino all'iperv piano soddisfa l'egualanza:  $H y_i(\omega x_i - \tau) = H \varepsilon$

Dove  $H$  è il fattore moltiplicativo positivo che può essere scelto come una costante arbitraria.



Il decision boundary sarà così l'intersezione degli infiniti iperpiani che classificano gli esempi con il piano a quale essi appartengono: l'intersezione rimane invariata ma cambia il COEFFICIENTE ANG. fra gli iperpiani delle sol. e il piano degli esempi che varia secondo  $H$ .

→ A questo punto fissiamo una distanza dall'  
= iper piano Vesuvio:

$y_i(\omega x_i - \tau) \geq 1 \leftarrow$  "non ci possono essere esempi

con una distanza dall'iperpiano inferiore ad uno".

Di conseguenza i valori + vicini all'iperpiano saranno del tipo:  $y_i(wx_i - \tau) = 1$  che è un SUPPORT VECTOR!

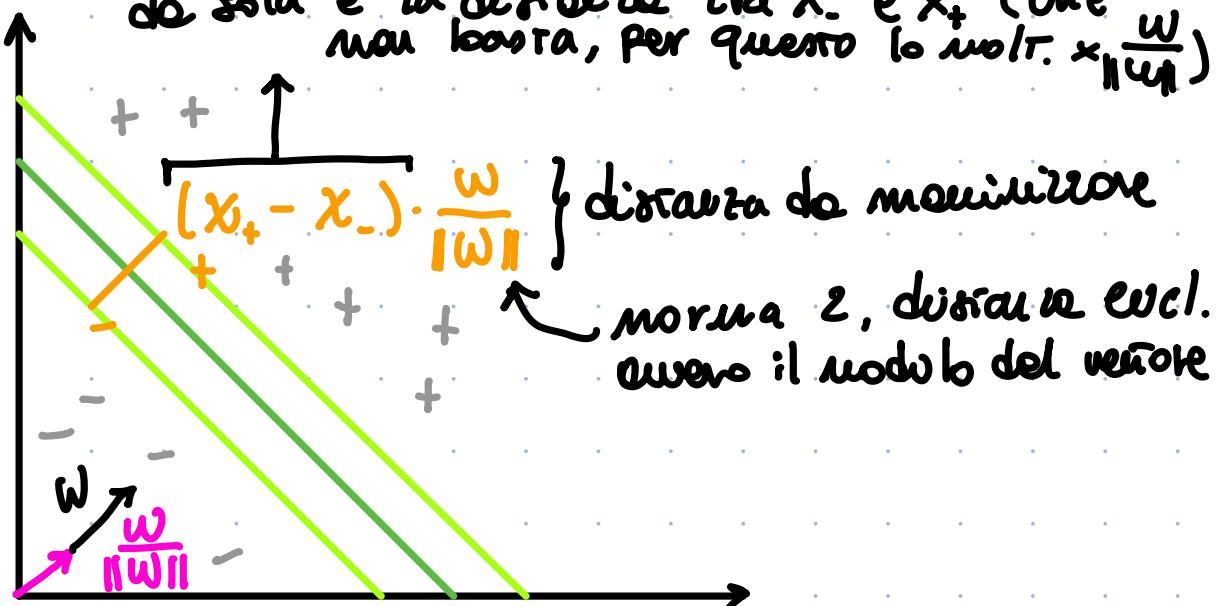
↳ "esempi positivi e neg. sul margine"

Se consideriamo la retta  $w\tau - \tau = 0$ ,  $w$  sarà ortogonale alla nostra retta, ovvero:

$$w\chi - \tau = 0 \quad \text{con } \chi_1, \chi_2 \in \text{retta}$$

$$\begin{aligned} w\chi_1 - \tau &= 0 & w\chi_2 - \tau &= 0 \\ \downarrow && \downarrow & \\ (\chi_1 - \chi_2)w &= \frac{w\chi_1}{\tau} - \frac{w\chi_2}{\tau} & & \end{aligned}$$

da sola è la distanza tra  $\chi_-$  e  $\chi_+$  (che non basta, per questo lo mult.  $\times \frac{w}{\|w\|}$ )



Il margine  $\mu$  sarà quindi:

$$\mu = (x_+ - x_-) \cdot \frac{w}{\|w\|} = \frac{x_+ \cdot w}{\|w\|} - \frac{x_- \cdot w}{\|w\|}$$

cioè vuol dire che:  $\begin{cases} x_+ \cdot w - \tau = 1 \rightarrow x_+ \cdot w = 1 + \tau \\ x_- \cdot w - \tau = -1 \rightarrow x_- \cdot w = \tau - 1 \end{cases}$

Sostituendo nella formula precedente, ottieniamo il mio margine geometrico!

$$\mu = \frac{1+\tau}{\|w\|} - \frac{\tau-1}{\|w\|} = \frac{2}{\|w\|}$$

• Formulare il problema di ottimizzazione:

minimizza  $w, \tau$   $\frac{1}{2} \|w\|^2$  //  $\frac{1}{2}$  viene tenuto x convenienza

subject to  $y_i(w \cdot x_i - \tau) \geq 1; 0 \leq i \leq m$

• Formulazione del problema duale:

→ Perché: vogliamo formulare un problema equivalente al primale ma utilizzabile nei software, ovvero più semplice e facile da risolvere.

→ le condizioni per la dualità forte di Lagrange:  
condizioni di Karush-Kuhn-Tucker

infimum: cancella  $x$  come il minimo

$$g(\alpha, v) = \inf_x \Lambda(x, \alpha, v)$$

$$= \inf_x \left( f_0(x) + \sum_{i=1}^m \alpha_i g_i(x) + \sum_{i=1}^p v_i g_i(x) \right)$$

$\leq 0 \quad \textcircled{2}$        $= 0 \quad \textcircled{3}$

moltiplicazioni di  
Lagrange

↳ Analisi:  $\Lambda(x, \alpha, v)$

•  $f_0(x)$ : funzione obiettivo originale

- $\textcircled{2}$  : la sommatoria esterna del primo vincolo moltiplicato per il coeff.  $\alpha_i$ ,
- $\textcircled{3}$  : una analoga sommatoria per il secondo vincolo moltiplicato per il coeff.  $v_i$ .

→ Per ogni punto ammissibile con valori negativi di  $g_i(x)$  [che quindi rispettano il vincolo del problema principale] e  $\alpha_i$  positivo, si ha un risultato negativo. Il secondo termine ha un discorso simile ma testa nullo → Il risultato è che  $f_0(x)$  sarà sempre maggiore che vogliamo.

• Dato che vogliamo ottenere questa formula da sopra, prima calcoliamo  $\Lambda(w, \tau, \alpha_i)$ , poi mi troveremo l'infimum! sotto del vincolo del prob. min.

$$\textcircled{1} \quad \Lambda(w, \tau, \alpha_i) = \underbrace{\frac{1}{2} \|w\|^2}_{\text{funzione obiettivo del prob.}} - \sum_{i=1}^m \alpha_i (y_i (w \cdot x_i - \tau) - 1)$$

*Funzione obiettivo del prob.  
più facile di minimizzare*

*Non dipendono dalle  
somme, li estrai*

$$= \frac{1}{2} \|w\|^2 - \sum_{i=1}^m \alpha_i y_i (w \cdot x_i) + \alpha_i y_i \tau + \sum_{i=1}^m \alpha_i$$

$$\frac{1}{2} w \cdot w - w \cdot \left( \sum_{i=1}^m \alpha_i y_i x_i \right) + \tau \left( \sum_{i=1}^m \alpha_i y_i \right) + \sum_{i=1}^m \alpha_i$$

*Ponendo fuori*

② Calcoliamo l'inf. facciamo la derivata parziale rispetto ai termini  $w$  e  $\tau$ .

- Prima cosa cancello i termini della formula preced. per cui la derivata parziale sarebbe 0.

$$\cancel{\frac{1}{2} w \cdot w - w \cdot \left( \sum_{i=1}^m \alpha_i y_i x_i \right) + \tau \cdot \left( \sum_{i=1}^m \alpha_i y_i \right) + \sum_{i=1}^m \alpha_i}$$

questo può derivare  $\tau$

E rimango con:  $\sum \alpha_i y_i$ , dunque ora deriviamo:

$$\frac{\partial L}{\partial \tau} = \sum_i \alpha_i y_i$$

$$\frac{\partial L}{\partial w} = w - \sum_i \alpha_i y_i x_i$$

③ Supponiamo che alle due derivate che diano 0:

$$\sum_i \alpha_i y_i = 0$$

$$w = \sum_i \alpha_i y_i x_i$$

Ovvero la combinazione lineare degli esempi per il vettore

④ Ricapitoliamo il tutto nella formulazione duale del P:

$$\text{Maximize}_x -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^m \alpha_i \alpha_j y_i y_j x_i x_j + \sum_{i=1}^n \alpha_i$$

subject to  $\alpha_i \geq 0, i = 1, \dots, n$

$$\sum_{i=1}^n \alpha_i y_i = 0$$

- Considerazioni sul problema duale:

- 1) I moltiplicatori lagrangiani positivi sono quelli con i support vector  $\rightarrow$  tutti gli esempi che non sono support vector sono associati a ledi di Lagrange NULLI.
- 2) La (1) è una conseguenza della proprietà KKT
 
$$\alpha_i(y_i(w\chi_i - \tau) - 1) = 0$$

$\curvearrowright$  coeff. lagrangiano

E quindi distinguiamo per  $\alpha_i$  i seguenti casi:

  - a)  $\alpha_i = 0$ , per tutti gli esempi che non sono support vector, e allora il prodotto sarà 0
  - b)  $\alpha_i > 0$ , allora effettivamente abbiamo un support vector ( $\chi$ ), ovvero il secondo fattore è necessariamente = 0!
- 3) Gli esempi con  $\alpha_i > 0$  sono un sottoinsieme dei support vector, tutti gli altri esempi NON SONO INFLUENTI e il calcolo di  $w$  come le classificazioni dell'iperpiano, sono ottenibili come prodotto scalare dei support vector.

4) Le formulazioni duali del problema permette di suddividere i task di classificazione non lineari grazie al metodo Kernel: il prodotto scalare viene sostituito al prodotto lineare in spazio arbitrario per ottenere funzioni + complete di queste!

- Margini di errore: fino ad ora abbiamo mantenuto il forte vincolo del margine fra esempio e piano pari ad 1, nel momento ora introducendo il termine  $\xi_i$ , che tollera due tipologie di errori:

$$y_i(\mathbf{w}x_i - \tau) \geq 1 - \xi_i$$

↪ In questo modo:  $\uparrow$  "SLACK VARIABLE"

- Permettono eserci + vicini alla soluzione
- Se  $\xi_i$  è abbastanza alto, tolleriamo errori di classificazione

- Le formulazioni del problema duale diventa:

$$\underset{\mathbf{w}, \tau, \xi}{\text{minimizza}} \quad \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i \quad // \begin{array}{l} \text{cerchiamo di rendere } \xi_i \text{ a 0} \\ \text{quanto più possibile} \end{array}$$

subject to  $y_i(\mathbf{w}x_i - \tau) \geq 1 - \xi_i; \quad 1 \leq i \leq n$

$\xi_i \geq 0; \quad 1 \leq i \leq n$

↳ La costante  $C$  è detta parameetro di complessità e regola l'importanza di  $\xi_i$ , ovvero la complessità della SVM:

- Se  $C$  grande: aumenta la penalizzazione per l'errore concavo
- Se  $C$  piccolo: siamo + flessibili su  $\xi_i$
- Se  $C = 0$ : il classificatore è libero di classificare come vuole anche esempi sbagliati e  $w$  sarà un vettore nullo (tutto a 0)
- Se  $C$  eccessivo: gli errori non sono tollerati, e si tenderebbe a scegliere valori di  $\xi_i$  nulli, facendo scomparire il secondo termine nella funzione di misura nel vettore  $y_i(w \cdot x_i - b) \geq 1 - \xi_i$  (risulta neanche solo 1)

↳ Ricostruiamo ora la formulazione doppia logaristica:

$$\Lambda(w, \tau, \xi_i, \alpha_i, \beta_i) = \rightarrow$$

$$\underbrace{\frac{1}{2} \|w\|^2 + C \sum_{i=1}^m \xi_i}_{f. obiettivo da minimizzare} - \underbrace{\sum_{i=1}^m \alpha_i (y_i (w \cdot x_i - \tau) - (1 - \xi_i))}_{\text{primo vincolo problema primale}} - \underbrace{\sum_{i=1}^m \beta_i \xi_i}_{\text{secondo v.}} =$$

f. obiettivo da minimizzare      primo vincolo problema primale      secondo v.

$$= \frac{1}{2} \|w\|^2 + C \sum_{i=1}^m \xi_i - \sum_{i=1}^m \alpha_i (y_i (w \cdot x_i - \tau) - 1) - \sum_{i=1}^m \alpha_i \xi_i - \sum_{i=1}^m \beta_i \xi_i$$

Il primo e terzo addendo sono esattamente  $\Lambda(w, \tau, \kappa_i)$   
ovvero la f. lagrangiana precedente!

$$\Lambda(w, \tau, \xi_i, \kappa_i, \beta_i) = \Lambda(w, \tau, \alpha_i) + \sum_{i=1}^m C \xi_i - \sum_{i=1}^m \alpha_i \xi_i - \sum_{i=1}^m \beta_i \xi_i = \\ \text{cioè} = \Lambda(w, \tau, \kappa_i) + \sum_{i=1}^m \xi_i (C - \alpha_i - \beta_i)$$

- A questo punto per trovare l'infimum facciamo le derivate parziali rispetto a  $w, \tau$  e  $\xi_i$  ( $w$  e  $\tau$  le avevamo già calcolate):

$\rightarrow //$  ovvero  $\alpha_i = C - \beta_i$   
 $C - \alpha_i - \beta_i = 0$ , ovvero non rimarrà altro che

$\Lambda(w, \tau, \xi_i, \kappa_i, \beta_i) = \Lambda(w, \tau, \alpha_i)$ , dunque le formule finali del problema dovranno svolgere il precedente con il vincolo di  $\xi_i^{(c)}$  in più!

Maximizza  $-\frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \kappa_i \alpha_j y_i y_j x_i x_j + \sum_{i=1}^m \alpha_i$

subject to  $0 \leq \alpha_i \leq C, i = 1, \dots, m$   
 $\sum_{i=1}^m \alpha_i y_i = 0$

(il vincolo  $\beta_i \geq 0$  è stato omesso,  $\beta_i$  non compare nella f.o. da massimizzare)

- L'ultima condizione KKT da considerare è:

$\beta_i \gamma_i = 0$ , ovvero se un esempio viene classificato con un certo margine di errore vuol dire che  $\beta_i = 0$ . E dunque sappiamo che  $\gamma_i = C$ !

- Come vengono classificati gli esempi (nuovi):

- $\alpha_i = 0$ , esempi classificati correttamente ma non sono support vector!
- $0 < \alpha_i < C$ , esempi che sono support vector.
- $\alpha_i = C$ , errore di margine (esemp: distanza oltre la tolleranza dell'errore)
- Kernel: o kernel trick, serve per trasformare algoritmi per problemi lineari per risolvere problemi non lineari

→ IDEA ALLA BASE: riconvertire lo spazio delle ipotesi, ovvero trasformare un dominio non-l.s. in un dominio l.s., con un adeguato mapping!

→ IDEA CONCRETA: mappa i miei elementi del piano in un paraboloide, interseco il p. con un piano

e poi n<sup>o</sup> trasformo il p. nel piano, ora separato!

Formalmente, con un vettore di 2 elementi  $x$ , un mapping  $\phi$  è una funzione di classif. lineare  $\hat{C}(x)$  di SVM:

$$x = (x_1, x_2) \quad \phi(x) = (x_1^2, \sqrt{2}x_1x_2, x_2^2, c)$$

$$\hat{C}(x) = \text{sign}(w \cdot x - b)$$

La funzione kernel  $K(x, y)$  è il prodotto interno di  $\phi(x)$  e  $\phi(y)$  mappati in uno spazio di Hilbert arbitrario.

$$K(x, y) = \langle \phi(x), \phi(y) \rangle$$

esempio: se vogliamo fare il prodotto interno di  $w \cdot x$

$$\begin{aligned}\hat{C}(x) &= \text{sign}(K(w, x) - b) = \text{sign}(\phi(w) \cdot \phi(x) - b) \\ &= \text{sign}((w_1^2, \sqrt{2}w_1w_2, w_2^2, c) \cdot (x_1^2, \sqrt{2}x_1x_2, x_2^2, c) - b) \\ &= \text{sign}(w_1^2x_1^2 + 2w_1w_2x_1x_2 + w_2^2x_2^2 + c^2 - b)\end{aligned}$$

che non è più una funzione lineare bensì quadratica!

Prodotto interno: generalizzazione del prodotto scalare in spazi vettoriali arbitrari, che ha delle

condizioni per considerare il mapping su prodotto intero.

1) Simmetria:  $\langle x, y \rangle = \langle y, x \rangle$

2) linearità:  $\langle ax + by, z \rangle = a\langle x, z \rangle + b\langle y, z \rangle$

3) Definito positivo:  $\langle x, x \rangle \geq 0$ ;  $\langle x, x = 0 \rangle \iff x = 0$

Il Kernel è allora una funzione  $K: V \times V \rightarrow \mathbb{R}$ , dove:

- $V$ : spazio vettoriale complesso o reale per le quali esiste un mapping  $\Phi: V \rightarrow \mathbb{F}$
- $\mathbb{F}$ : spazio di Hilbert, ovvero uno spazio vettoriale coniugato su cui è definito un prodotto interno.

→ è uno spazio ideale per le operazioni dato che dispone di molte proprietà tipo calcolo delle distanze, angoli, ortogonalità e completezza.

→ Kernel come funzioni di similità: se due vettori sono simili (ovvero "puntano" nella stessa direzione) allora il loro prodotto scalare sarà molto alto, basso per vettori dissensi e nullo per vettori ortogonali.

- Kernel polinomiale: ha due versioni:

$$K(x,y) = (x \cdot y)^d$$

$$K(x,y) = (x \cdot y + 1)^d$$

→ d è il grado determinante del polinomo:

- 1)  $d=1$ : Kernel lineare
- 2)  $d=2$ : Kernel quadratico
- 3)  $d$  : polinomo generico di grado d

↳ Se prima si parla di fare il mapping e poi il prodotto ora si fa il contrario!

- Kernel Gaussiano:  $K(x,y) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)$

Utilizzato per il calcolo del RBF e lo spazio di Hilbert in cui viene mappato ha infinte dimensioni.  
Il parametro σ serve per regolare l'ampiezza delle funzioni radiali.

- Combinazione di kernel: (caso prodotto e somma)

$$K(x,y) = K_1(x,y)K_2(x,y) \quad // \text{AND equivalent}$$

$$K(x, y) = k_1(x, y) + k_2(x, y) \text{ // OR EQUIVALENT}$$

- Si può moltiplicare su  $K$  per uno scalare  $\alpha > 0$  ed ottenere un altro kernel
- $K(x, y) = f(x)f(y)$ : un kernel può anche essere il prodotto di due funzioni reali applicate a  $x$  e  $y$ .
- Condizioni per le funzioni kernel:

Perché un kernel sia valido deve dare rispettare le condizioni di Mercer: una funz.  $K$  è un kernel valido se: Viene fatto di punti  $\{x_1, \dots, x_n\}$ . La matrice  $K_{ij} = K(x_i, x_j)$  con  $i, j = 1, \dots, n$ , dove  $x_i$  e  $x_j$  sono esempi presi dall'insieme di punti appena introdotto, è simmetrico e semidefinita positiva.

DIM:

1) SIMMETRIA:  $K$  è simmetrica poiché il prod. interno è uno stesso simmetrico

2) SEMIDEFINITA POSITIVA: (si annulla nel punto e intorno ha valori strettamente positivi)

$$z^T K z = \sum_i \sum_j z_i z_j K_{ij}$$

$$\begin{aligned}
 &= \sum_i \sum_j z_i z_j \underbrace{K(x_i, x_j)}_{\text{funzione kernel}} \\
 &= \sum_i \sum_j z_i z_j \underbrace{\langle \phi(x_i), \phi(x_j) \rangle}_{\text{prodotto interno}} \\
 &= \sum_i \sum_j \underbrace{\langle z_i \phi(x_i), z_j \phi(x_j) \rangle}_{\text{Dato che è lineare in entrambi gli argomenti}} \\
 &= \left\langle \sum_i z_i \phi(x_i), \sum_j z_j \phi(x_j) \right\rangle \quad \text{dove i due sommatori sono lineari} \\
 &= \left\langle \sum_i z_i \phi(x_i), \sum_i z_i \phi(x_i) \right\rangle \geq 0 \quad \text{scegli indici, risultato } \geq 0, \text{ Dato}
 \end{aligned}$$

Facciamo la stessa cosa alle sommatorie per lo stesso motivo di sopra!

## MODELLO BAYSIAN SU DISTANZA

Distanza di Minkowski: sia  $\mathcal{X} = \mathbb{R}^d$ , la Minkowski distanza di ordine  $p > 0$  è definita come:

$$Dis_p(x, y) = \left( \sum_{j=1}^d |x_j - y_j|^p \right)^{\frac{1}{p}} = \|x - y\|_p$$

Ovvero non c'è altro che la norma  $p$  del vettore  $(x-y)$ . Ricordiamo che:

- Norma 2: distanza euclidea
- Norma 1: distanza di Manhattan

→ Norma  $\infty$ : distanza di chebyshew, ovvero:

$$\text{Dis}_{\infty}(x, y) = \max_j |x_j - y_j|$$

→ Norma 0: conta il numero di elementi non nulli di un vettore, usata per dire quando due vettori sono uguali o meno, ma è una distanza di Minkowski; ma è definita come:

$$(\text{Dis}_0(x, y) = \sum_{j=1}^d (x_j - y_j)^0 = \sum_{j=1}^d I[x_j \neq y_j])$$

Se  $x$  e  $y$  sono stringhe binarie la Norma 0 è detta distanza di Hamming (indica il vettore di bit che occorre sommare ad  $x$  per ottenere  $y$ ). Per stringhe user bisogna si generalizzino nella distanza di Levenshtein (edit distane) o nella distanza di Jaccard.

• Distanza metrico: devo uno spazio delle istanze  $X$ , una distanza metrica è una  $f: X \times X \rightarrow \mathbb{R}$  t.c.

✓  $x, y, z \in X$ :

1. La distanza tra su punto e se stesso è 0
2.  $\text{Dis}(x, y) > 0$  se  $x \neq y$
3. Le distanze sono simmetriche:  $\text{Dis}(x, y) = \text{Dis}(y, x)$

4. Una strada con punti intermedi non puo' essere  
più corta della distanza diretta (in aria) tra due  
punti:  $\text{Dis}(x,z) \leq \text{Dis}(x,y) + \text{Dis}(y,z)$   
 $\hookrightarrow (\text{dis.} \text{Triang.})$

- Se vi lasciamo il ② (avendo ponendo  $\text{Dis}(x,y) = 0$  anche se  $x \neq y$ ) allora la distanza è PSEUDO-METRICA.
- Nella scelta del parametro critico  $P$ , se  $P < 1$  allora la disug. triang. non è verificata.

• Distanza di Mahalanobis: come facciamo a scoprire se delle features sono correlate fra loro?

La matrice ellittica  $H$  è spesso stimata dai dati a partire dalla matrice di covarianza inversa  $\Sigma^{-1}$ , in particolare la distanza di  $H$ :

$$\text{Dis}_H(x,y|\Sigma) = \sqrt{(x-y)^T \Sigma^{-1} (x-y)} \quad \text{dove la } \Sigma^{-1},$$

matrice di covariata ha l'effetto di decomporre e normalizzare le features per un iperellisseide.

~ La distanza euclidea è un caso particolare della J.M. dove la matrice di covariata è la matrice di identità, ovvero:  $\text{Dis}_E(x,y) = \text{Dis}_H(x,y|I)$

» Spiegazione intuitiva: si consideri il problema della stima di probabilità che un punto nel suo spazio euclideo appartenga ad un insieme, da cui sono dati degli altri punti che sicuramente già appartengono a tale insieme. Naturalmente, più il punto sarà vicino al centro delle masse, più è verosimile che appartenga a tale insieme. Bisogna inoltre considerare se l'insieme è distribuito su piccole o grandi distanze!

L'approccio più semplice è calcolare la deviazione standard dei campioni dal centro di massa: se la distanza tra i punti in esame e il centro di massa è minore della d.s. allora sicuramente appartiene all'insieme! Overo, maggiore distanza, minor verosimiglianza.

Questo approccio intuitivamente può essere formalizzato in:  $\frac{x-y}{\sigma}$ , ma l'auutuzione è che i punti siano distribuiti secondo  $\sigma$  ad un'ipotesi alternativa al centro di massa. Se però le distribuzioni non c'è sfinge (ma: perelli = ellisoidale, ad esempio) le sue opposte non puo' dipendere solamente dalla distanza, ma anche dalla sua direzione. L'iperellioidale che meglio rappresenta l'insieme di probabilità puo' essere ottenuto attraverso la matrice di covarianza dei campioni: la dist. di  $M$  sarà dunque la dist. del punto in esame dal centro delle masse normalizzata all'ampiezza dell'ellisoidale nella sua diretta.

→ Costruire la matrice di covarienza:  $\Sigma$  ha  $n$   $\sigma_{ij}$  I

$$\sigma_{ij} = \frac{1}{n} \sum_{k=1}^n (x_{ki} - \hat{x}_i)(x_{kj} - \hat{x}_j), \text{ dove:}$$

- $\sigma_{ij}$  = singolo elemento della matrice di cov  $\Sigma$  che è la covariante tra l'oggetto i-esimo e j-esimo.

- ↳ Se  $i, j$  non concordi: prodotto negativo
- ↳ Se  $i, j$  indipendenti: prodotto vicino allo 0
- ↳ Se  $i=j$ : prodotto uguale alla varianza di  $i$ .  
cole la matrice simmetrica!

→ (esempio) distanza di Mah.:

data la matrice di cov.  $\Sigma = \begin{bmatrix} 0.3 & 0.2 \\ 0.2 & 0.3 \end{bmatrix}$

ed i seguenti punti:  $A = (0.5, 0.5)$ ,  $B = (0, 1)$  e  $C = (1.5, 1.5)$  le loro M. dist. :

$$\text{Mahal}(A, B) = 5 \quad | \quad \text{Mahal}(A, C) = 4$$

Si puo' notare come tramite la moltiplicazione per  $\Sigma^{-1}$  la distanza di  $(A, C) < (A, B)$ , che è il contrario della distanza euclidea!

## • Media e Distanza:

"la media aritmetica  $\mu$  (cardoide) di misure di dati  $D$  in uno spazio euclideo è il punto unico che minimizza la somma delle distanze quadratiche dai punti"

Dimostrazione: arguiamo  $\sum_{x \in D} \|x - y\|^2 = \mu$  (prova a minimizzare la somma delle distanze quadratiche dei punti, se coincide con  $\mu$ , avremo ragione)

Per cercare il minimo basta fare il gradiente della somma e verificare dove si annulla:

$$\nabla_y \sum_{x \in D} \|x - y\|^2 = -2 \sum_{x \in D} (x - y) = -2 \sum_{x \in D} x + 2|D|y = 0$$

e la sua derivazione sarebbe  $y = \frac{1}{|D|} \sum_{x \in D} x = \mu$  CVD

↳ Se è un punto: media geometrica

↳ Caso univariato: Valori medi

↳ esempio nel duezioni: medaide (per trovarlo infatti basta valutare l'elemento la distanza totale da tutti gli altri punti e scegliere quello che ne minimizza la somma).

• Classificazione lineare di base: se  $x$  è più vicino all'asse plate della classe positiva  $\mu^+$  allora  $x$  positivo, negativo altrimenti (se si usa la distanza euclidea è un classificatore lineare di base, se si utilizza quella di Manhattan si ottiene una regione non convessa, cioè dati 2 punti la loro interpolazione non è all'interno della regione stessa)

• Classificazione KNN (nearest-neighbour):

→ IDEA: scegliere la classe dell'esempio basandosi sulla classificazione di  $k$  vicini (di training). Preferibilmente evitando un  $k$  dispari!

→ COMPLICATISSIMA TEMPOREALE:  $O(n)$ , usa tutti dati del training set e  $O(1)$  per la singola predizione.

→ SCELTA DEL k: problema complesso, solitamente si ripete la costruzione del dataset con diversi  $k$  per vedere quale funziona meglio.



(1NN): separa perfettamente le classi → BASSO BIAS, ALTA VARIANZA, bontà poco ruvide x fare casuale  
↳ aumentando  $k$  → Si aumenta bias, si riduce varianza.

→ classificazione KNN: raggruppamento (voronei: rettangoli ellittici) + pesi (distanze pesate) → pesi decrescono esponenzialmente con la distanza: il conseguente aumento di  $k$  è molto + piccolo.

↳ Modelli locali vs globali: locali (sono una f. estimativa su una sottosamplificazione locale del training set) e globali (sono valide globalmente nel training set).

↳ Per valori piccoli di  $k$ : KNN si configura come una combinazione di modelli locali.

↳ Per valori grandi di  $k$ : KNN si configura come un unico (o qualche) modello globale.

↳ KNN come regressione: aggrega le  $k$  predizioni dei  $k$  vicini per effettuare una media (aggregatore k-means, da non confondere con l'algoritmo k-means)

• DBSCAN: algoritmo basato sulla densità (numero di punti entro un certo raggio,  $\text{Eps}$ ): esso itera su tutto il dataset e valuta la densità di un punto, se punto è selezionato come centrale se ha su conto n. di punti ( $\text{MinPts}$ ) entro  $\text{Eps}$ .

→ Punto di bordo: ha meno  $\text{MinPts}$  ma è vicino al

Punto di centro

→ Punto di ruote: ciò che non è di centro e di bordo.



• Algorithm Learn Rule list (D): // con DBSCAN)

1. Labelliano ogni punto come di "ruote", "centro" o "bordo".
2. Eliminano i punti di ruote
3. Pongono su arco tra quelli di centro e/o bordo
4. Pone ogni gruppo di centro connessi in un cluster  
**SEPARATO**
5. Assegna ai punti di bordo ai rispettivi cluster  
dei punti di centro.

• Clustering (non-sup.) basato sulla distanza:

Per valutare i cluster utilizziamo l'**SSE**, ovvero  
**SSE = BSS + WSS**. (sum squares error)

→ Cluster Cohesion: misura quanto direttamente sono collegati gli elementi di un cluster:

$$WSS = \sum_i \sum_x (x - m_i)^2 \rightarrow \text{cluster molto denso} = \text{bassa cohesione!} \quad (\text{within clusters sum of squares})$$

→ Cluster separator: misura quanto ben separato è un cluster dall'altro.  $BSS = \sum |C_i| (\mu - \mu_i)^2$  dove  $|C_i|$  è la cardinalità del cluster. (Between clusters sum of squares).

→ Matrice di Scatter: in caso di esempi multivariati l'SSE si può trovare con la matrice di scatter. Data una matrice  $X$  la sua matrice di scatter  $S$  è:

$$S = (X - \mu)^T (X - \mu) = \sum_{i=1}^m (x_i - \mu)^T (x_i - \mu)$$

Dove  $\mu$  è il vettore che contiene i centroidi delle colonne di  $X$ .

Definiamo allora lo scatter di  $X$  con  $Scat(X)$  cioè:

$Scat(X) = \sum_{i=1}^m \|x_i - \mu\|^2 \rightarrow$  l'equivalente di sommare tutti gli elementi sulla diagonale della matrice di scatter.

→ Se si immagina di suddividere il dataset in  $K$  partitioni  $D_1, \dots, D_K = D$ , con  $\mu_j$  centoide per  $D_j$  e  $S_j$  mat. di scatter per  $D_j$  allora:

$$1. S = \sum_{j=1}^K S_j + B$$

2.  $B$  è la matrice di scatter che si ottiene riunendo  
osservi puro in  $D$  col corrispondente centroide  $\mu_j$ .  
 $B$  descrive lo spread dei centroidi.

3. Ciascuna  $S_j$  è una matrice di scatter entro il cluster  
e ne descrive la compactezza.

$$4. \text{Scat}(D) = \sum_{j=1}^k \text{Scat}(D_j) + \sum_{j=1}^k \|D_j\| \cdot \|\mu_j - \mu\|^2$$

- $\mu$  problema K-means è quello di trovare una  
partizione che mi minimizzi il primo termine  
(o massimizzi il secondo)

↳ NP-COMPLETE

↳ L'algoritmo K-means deriva da iterazioni  
il seguendo meccanismo basato sul centroide +  
vicino, per poi calcolare il centroide per ogni partizione  
così avanti.

- Algoritmo K-means ( $D, k$ ):

- 1. Genero casualmente i centroidi, poi
- 2. Calcolo il centroide di ogni gruppo
- 3. Correggo le nuove partizioni  
aggiungendo il punto di ingresso ( $x_i$ )  
al centroide + vicino
- 4. Ricalo i centroidi per i nuovi  
gruppi fino alla convergenza

1. Inizializzo random  $\{\mu_1, \dots, \mu_k\}$  (vettori)
2. while  $\{\mu_1, \dots, \mu_k\}$  sono cambiati da prima:
3. Assegno ogni  $x \in D$  ad argmin <sub>$j$</sub>   $\text{Dis}_2(x, \mu_j)$
4. for  $j$  in range(1,  $k$ ):

5:  $D_j = \{x \in D \mid x \text{ assegno al cluster } j\}$

6:  $M_j = \frac{1}{|D_j|} \sum_{x \in D_j} x$

7: ritorna  $\{M_1, \dots, M_K\}$

→ Il parameetro  $K$  deve essere arbitrariamente preso a caso  
(e non avrebbe molto senso scegliere  $K=n$ , purtroppo)

• Algoritmo K-Medoids ( $D, K, Dis$ ): invece di utilizzare i  
centri di massima: medoidi:

Cambia solo la riga 6 →

6:  $M_j = \arg \min_x \sum_{x' \in D_j} Dis(x, x')$

• Svantaggi di K-means:

→ Tende a creare clusters di forme globulare

→ Nel caso di risciacquo rispetto ad un'ona tende  
a generare i cluster in metà

• Silhouette: misura di valutazione molto popolare  
dato che tende a ridurre le somme dei pesi tra nodi  
e massimizzare la separazione.

↳ DATI UNI:

1.  $x_i$ : singolo esempio nel dataset  $D_i$
2.  $d(x_i, D_i)$ : distanza media di  $x_i$  dai punti del  $D_i$
3.  $J(i)$ : indice del cluster a cui  $x_i$  appartiene
4.  $a(x_i) = d(x_i, D_{J(i)})$ : distanza media di  $x_i$  dal punto del suo cluster  $D_{J(i)}$  (cohesion media)
5.  $b(x_i) = \min_{k \neq J(i)} d(x_i, D_k)$ : distanza media dei punti del \_\_\_\_\_ suo cluster + numero  $k$ . (separation media)

! Ci aspettiamo che  $a(x_i) > b(x_i)$  ma non avviene sempre... La silhouette di  $x_i$  allora sarà:

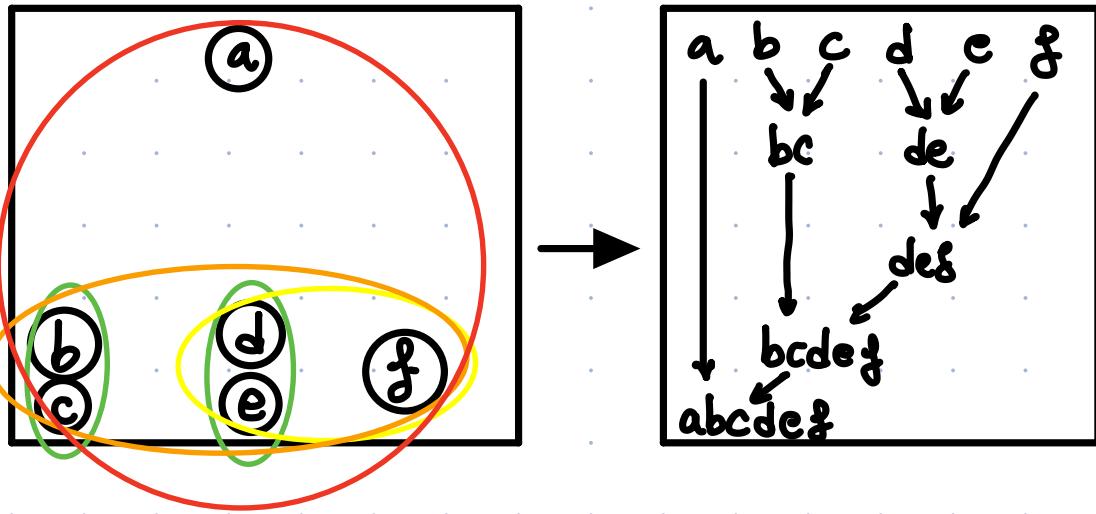
$$\delta(x_i) = \frac{b(x_i) - a(x_i)}{\max(a(x_i), b(x_i))} = \frac{\text{separation - cohesion}}{\max(\text{sep}, \text{coh})}$$

- ! Nel caso in cui  $a(x_i) > b(x_i)$  si decide in media che i membri del cluster cui sono + vicini a  $x_i$  appartengono ai membri dello stesso cluster di  $x_i$ .
- ! Più è alta la silhouette migliore è la classificazione per  $x_i$ : 1 è il valore massimo.
- Clustering gerarchico: clustering mirato a creare gerarchie  $\rightarrow$  diagrammi di dendrogramma / albero.  
 ↳ Non richiede di fissare il valore dei cluster a priori ma tende a formare un albero e ad indi-

- = ridurre poi in una foglie simili.
- Dopo i raggruppamenti per ogni delle foglie si possono ancora formare i clusters con questi raggruppamenti.
- Si utilizza un dendogramma, spazio di rappresentazione dimensionale fra i vari punti del dataset.  
Più formalmente esso è un albero binario con gli elementi di D sulle foglie e i somminimi di D nei suoi nodi interi.  
Il livello del nodo rappresenta la distanza tra i due cluster rappresentati dai figli dei nodi.
- Funzioni di Linkage:  $L: 2^X \times 2^X \rightarrow \mathbb{R}$   
calcola la distanza tra due soluzioni dello spazio delle istanze:

1. Single Linkage: usare omologhe, la distanza tra due cluster è la minima esistente tra due dei loro elementi.
2. Complete linkage: usare paraloghe, ...
3. Average linkage: la distanza tra i due cluster è la media minima tra i collegamenti esistenti tra i loro elementi.
4. Centroid linkage: la distanza tra i due cluster è la distanza tra i due centroidi.

## Esempio di Agglomerative clustering:



Metodo di raggruppamento: raggruppo prima i più vicini, poi vado sempre ad allargare il loro possibile range!

**■**: primo step, gruppi: {bc} & {def}

**■**: secondo step, gruppo {defg}

**■**: terzo step, gruppo {bcd} & {fg}

**■**: ultimo e quarto step, gruppo {abcdefg}

- NB: si è usata la distanza euclidea come metrica di distanza in questo esempio... "ad occhio".

## • Algoritmo HAC<sup>(D,L)</sup> (Hierarchical Agglomerative Clustering):

1. Inizializza i clusters come le foglie dei fuoi di D.
2. While Tutti i fuoi del dataset sono in cluster:
  3. Trova  $(X, Y)$  coppia di cluster t.c. abbia il più basso linkaggio l, merge X e Y.
  4. Crea il nodo figlio di  $(X, Y)$  al livello l.
  5. Ritorna l'albero creatosi. (BOTTOM-UP)

→ Si parla costruzione dendrogramma, clusters per scoprire criteri da applicare ed evitare clustering spurio.

## • Dai Kernel Alle Distanze: (Tipi di Features)

### • Nominali:

→ Operazioni:  $=, \neq$

→ esempi: ID, colore occhi, zip codes

### • Ordinali:

→ Operazioni:  $<, >$

→ esempi: ranking, voti, indirizzi

### • Quantitativi:

→ su intervalli  $(+, -)$ , sul continuo  $(+, -, *, /)$

→ esempi: intervalli: (date), scale (temperatura, tempo, costi, lunghezze)

• Trasformare feature di distanza in similitudine e viceversa:

Feature	Dissimilitudine $d(x,y)$	Similitudine $s(x,y)$
Nomivale	$d = \begin{cases} 0 & \text{se } x=y \\ 1 & \text{altrimenti} \end{cases}$	$s = \begin{cases} 1 & \text{se } x=y \\ 0 & \text{altrimenti} \end{cases}$
Ordinalle	$d = \frac{ x-y }{n-1}$	$s = 1 - d$
Quantitativi	$d =  x-y $	$s = -d$ $s = (1/(1+d))$

↳ Poi combiniamo il valore della prossimità delle sig. features con la formula di Minkowski:

$$dis_p(x,y) = \left[ \sum_{k=1}^d \alpha_k \cdot d(x_k, y_k)^p \right]^{\frac{1}{p}}$$

↳ Dato che il kernel è una funzione di similitudine tra punti, potremo riprodurre il kernel tra i modelli basati sulle distanze  $\rightarrow$  sostituiamo  $x$  la distanza euclidea con una sua versione con i prodotti:

$$Dis_2(x,y) = \|x-y\|_2 = \sqrt{(x-y)(x-y)} = \sqrt{xx - 2xy + yy}$$

Rendono il termine translation-

- invariant, visto che il dot product non lo è!

• Dot Product: prodotto tra due vettori definito come

$$a \cdot b = \sum_{i=1}^n a_i b_i = a_1 b_1 + a_2 b_2 + \dots + a_n b_n = a^T b$$

• In geometria euclidea:  $a \cdot b = \|a\| \|b\| \cos(\theta)$

• Dove  $\|a\|$  e  $\|b\|$  sono le lunghezze dei vettori e  $\theta$  è l'angolo tra loro.

↳ Rimpiazzando il dot prodotto nello  $f$  precedente otteniamo la stessa distanza ma kernelizzata:

$$\text{Dis}_k(x, y) = \sqrt{k(x, x) - 2k(x, y) + k(y, y)}$$

Definisce una pseudo-metrica quando  $k$  è un kernel semi-definito positivo.

$$\hookrightarrow k(x, y) \geq 0, \forall (x, y)$$

• Algoritmo Kernel-Kmeans ( $D, K$ ):

inizializza random  $K$  clusteri  $D_1, \dots, D_K$

while (nuovi cambiamenti in  $D_1, \dots, D_K$ ):

    assegni ogni  $x \in D$  ad  $\arg \min_j \frac{1}{\|D_j\|} \sum_y \text{Dis}_k(x, y)$   
     for  $i$  in range( $1, K$ ):

$$D_j = \{x \in D \mid x \text{ assegnato al cluster } j\}$$

return  $D_1, \dots, D_K$

→ Cosine Similarity: via alternativa di trasformare in dati produtti nelle distanze →

$$\text{sim} = \cos \theta = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \cdot \|\mathbf{y}\|} = \frac{\mathbf{x} \cdot \mathbf{y}}{\sqrt{(\mathbf{x} \cdot \mathbf{x})(\mathbf{y} \cdot \mathbf{y})}}$$

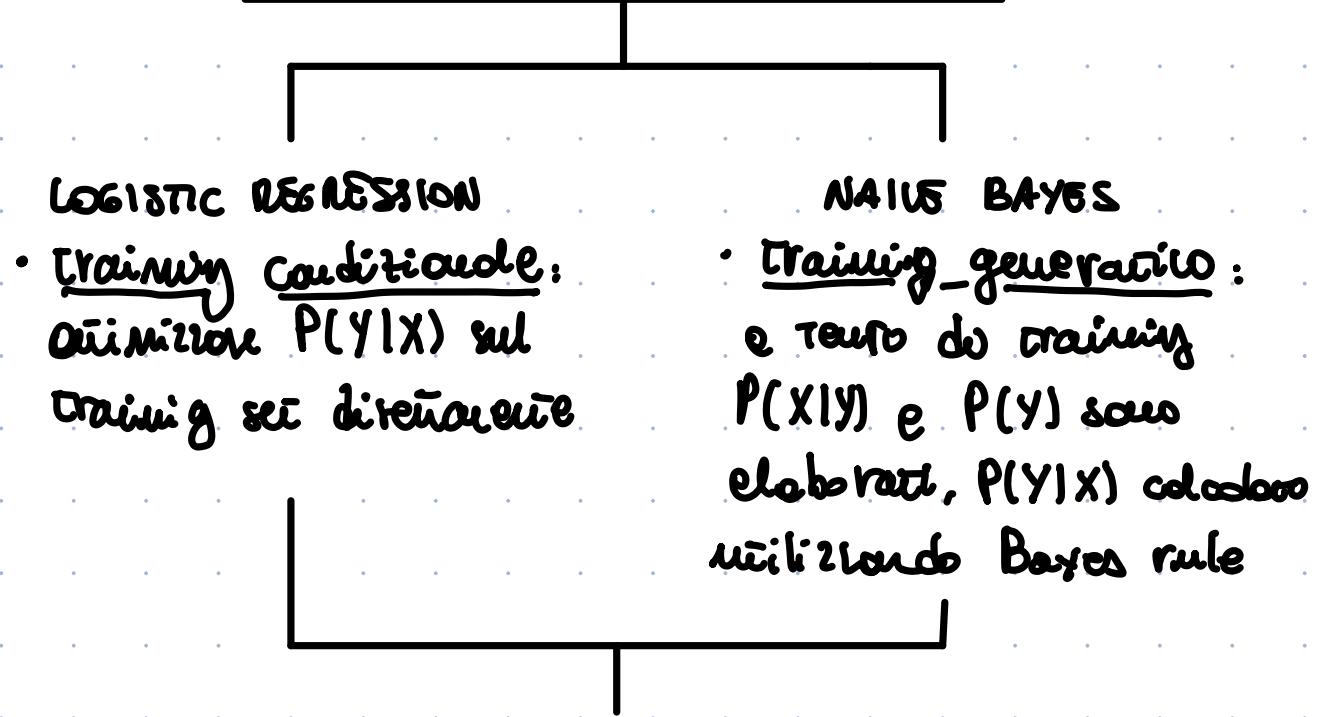
1. Non dipende dalla lunghezza dei vettori (come invece la distanza Euclidea)
2. Non è Translation-independen
3. Non è una metrica di distanza per-sé, e neanche non si prende  $1 - \cos \theta$
4. Kernelizzata in  $\cos(\theta) = \frac{k(x,y)}{\sqrt{k(x,x)k(y,y)}}$
5.  $\begin{cases} (+1) = 2 \text{ vettori uguali} \\ (-1) = Opposti uguali \end{cases}$

→ 5-fold-validation: dati sei divisi in 5 fold, 4 usati per training, 1 per test, poi nell'iterazione succ. vengono permutati.

## MODELLO PROBABILISTICO (cap 9)

- Due tipologie: modelli discriminativi (o descrittivi), che modellano la fortezza della frontiera delle classi in cui c'è la separazione delle classi (il bordo) e modelli generativi, che tendono a modellare le leggi che governano la classe (o che "genera" esempi).

## Modelli Naturalmente Probabilistici



Obiettivi sono: (AUT-AUT)

1) Maximizzare la P. a Posteriori: BAYES-OPTIMALITÀ

$\hat{y} = \operatorname{argmax}_y P(Y=y | X)$ : la classe predetta  
è anche quella più probabile

2) Maximizzare la P. a priori: MAXIMUM LIKELIHOOD

$\hat{y} = \operatorname{argmax}_y P(X=x | Y=y) \rightarrow$  Finito se solo una classe  
equi-probabile

3) Stimare il max della P. a posteriori: MAP

$\hat{y} = \operatorname{argmax}_y P(X=y | Y=y) P(Y=y)$

• Modelli descrittivi:

- Modellova la prob. a posteriori di eventi
- Focalizzati sulla definizione delle differente classi apprendendo una funzione discriminativa.
- le classi sono V.A. indipendenti restituite da una funzione d-dimensionale (dove d = n° feature rappresentate nel vettore  $V = v_1, \dots, v_d$ )
- Funzione discriminante:  $g_k(V) = g_{k_0}(V, \dots, V)$   
mappa ogni esempio d-dimensionale in un val. reale.
- la f. descr. determina  $\frac{L(L-1)}{2}$  confronti di classi
- Confusione tra due classi ( $i$  e  $C_k$ ):  $g_i(V) - g_k(V) = 0$

• Modelli generativi:

- Mentre i modelli d. modellano la p. a post.  $P(Y|X)$   
i modelli ge. modellano  $P(Y, X)$  del target  $Y$  e feature  $X$ .
- Ricordiamo che:

$$P(X) = \sum_y P(y, X) \quad \text{e} \quad P(Y|X) = \frac{P(Y, X)}{P(X)}$$

- Possiamo descrivere i modelli ge. con le likelihood:  $P(X|Y)$  che permette di ottenere facilmente:

$$P(Y, X) = P(X|Y) P(Y) \quad \text{e la distr. a priori!}$$

→ Vantaggi / svantaggi: necessità di memorizzare le prob. condizionali = molto spazio richiesto (es. pol.) oppure si semplifica il modello, perdendo accuratezza.

### • Apprendimento di un modello generativo:

#### il processo di riduzione dell'incertezza

- 1) Si assume una distr. a priori iniz. sulle classi
- 2) Dopo aver classificato qualche esempio, si passa a quella a posteriori  $P(Y|X)$ : se è meno inf. rispetto a quella a priori accettiamo la classe + probabile.
- 3) Si usa la p. post. come da base una nuova p. priori (Processo di extrazione Naïve Bayes) e si va avanti fino a quando non si è estratti + molte info.

- p-distribuzione: distribuzione incertaità su una cosa  $\theta$  (probabilità di un evento) che viene aggiornata quando  $\theta$  cambia: in p si lascia libero il valore di  $\theta$  (al contrario di MAP dove invece c'è una costante finita) e lo si fa variare sull'asse  $x$ .
- Ottimalità di Bayes: proprietà dei classificatori che conseguono sempre  $\text{argmax}_y P^*(Y=y | X=x)$  ad  $x$ , dove

$P^*$  è la distribuzione REALE a posteriori.

- Selezione di modello: l'idea è quella di selezionare il modello sul principio dell'ottimalità di Bayes.

Sia  $M$  l'insieme di tutti i possibili modelli applicabili in per fare stime, allora:

$$P(y | \chi = x) = \sum_{m \in M} P(y, M=m | \chi = x)$$

Si utilizza poi la chain rule per combinare il condizionamento con la priorità

$$P(y | \chi = x) = \sum_{m \in M} P(y | M=m, \chi = x) P(M=m | \chi = x)$$

Ma siccome modello e esempio sono v.a. indipendenti:

$$P(y | \chi = x) = \sum_{m \in M} P(y | M=m, \chi = x) P(M=m)$$

Scopriamo che anche scegliendo un certo modello  $m^*$  che ci piace non c'è dubbio che il giudizio critico sia rispettato → le 2 formule non sono uguali!

- Mixture Model: modello probabilistico che rappresenta una sovrapposizione dentro una pop. totale

## Distribuzione normale multivariata:

Panico del caso univariato, dove le sue dist. è:

$$P(x|\mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}} = \frac{1}{E} e^{-\frac{x^2}{2}}, \text{ dove } E = \sqrt{2\pi}\sigma$$

Nel caso multivariato dobbiamo trasformare delle var:

- deviazioni standard  $\rightarrow$  matrice di cabriauza
- Scontro quadrato ad esponente  $\rightarrow$  distanza di Mahalanobis.

$$P(x|\mu, \Sigma) = \frac{1}{E_d} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}, \text{ dove}$$

$$E_d = (2\pi)^{\frac{d}{2}} \cdot \sqrt{|\Sigma|}, \quad d = \text{m° di eutie del vettore esempio}$$

$\mu = \text{vettore target}$ ,  $\Sigma = \text{inverso m.c.}$

$|\Sigma| = \text{determinante m.c.}$

↳ Questo è il cosiddetto <sup>Gaussian</sup> mixture model, un modello fatto de 2 gaussiane, ma per classificare classe.

↳ Guardare figure su slides / appunti.

• Likelihood Ratio:  $LR(x) = \frac{P(x|+)}{P(x|-)} =$

$$= \frac{\sigma^-}{\sigma^+} \exp \left( -\frac{1}{2} \left[ \left( \frac{x - \mu^+}{\sigma^+} \right)^2 - \left( \frac{x - \mu^-}{\sigma^-} \right)^2 \right] \right)$$

- Se  $LR(x) > 1$ :  $P(x|+) > P(x|-)$  → + scelta
- Se  $LR(x) < 1$ :  $P(x|-) > P(x|+)$  → - scelta
- Se  $LR(x) == 1$ : Siamo esauriti nel decision boundary, come scegliere? Abbiamo così:

1)  $\sigma^+ = \sigma^- = \sigma$ : due classi di cui  $\sigma$  è uguale ma distribuite in modo diverso secondo i loro (diversi) centroidi  $\mu^+$  e  $\mu^-$ : il luogo di massima likelihood è  $\mu$  e quindi abbiamo già conosciuto la retta verticale (coincide con il classificatore lineare di base). Se ne ricava la form. precedente:

$$= \frac{\mu^+ - \mu^-}{\sigma^2} \cdot \left[ x - \frac{\mu^+ + \mu^-}{2} \right]$$

2)  $\sigma^+ \neq \sigma^-$  Non vale più: per esempio in un'università  $\sigma^+ = 1$  e  $\sigma^- = 2$ , quindi  $\sigma^- = 2\sigma^+$  e dunque  $LR(x) = 2 \exp(-[(x-1)^2 - (x-2)^2/4]/2) = 2 \exp(3x^2/8)$ . Dunque  $LR(x) = 1 \iff x = \pm(8/3)\ln 2 = \pm 1.85$ : questo (impraticabile) calcolo ci suggerisce (boh) che il decision boundary sarà dato da due aree separate (la regione negativa è separata a metà da un'area centrale due è l'area positiva)

3) Caso bivariato: assumendo di avere  $\mu_1^+ = \mu_2^+ = 1$

$$\text{e } \bar{\mu}_1 = \bar{\mu}_2 = -1 : P(x_1, x_2 | \mu_1, \mu_2, \sigma_1, \sigma_2, p) = \\ = \frac{1}{E_2} \exp\left(-\frac{1}{2(1-p^2)}(z_1^2 + z_2^2 - 2pz_1 z_2)\right) \text{ con}$$

$E_2 = 2\pi\sigma_1\sigma_2\sqrt{1-p^2}$  con  $z$  (dev. std. tra  $x$  e media) =  
 $z = \frac{x_i - \mu_i}{\sigma_i}$ , dove  $p$  è una normalizzazione che rende  
 le dev. std. delle due feature in appena. Vediamo  
 ora i due sottocasi 1,2 ma nel caso bivariato!

3.1)  $\sigma_1^+ = \sigma_1^- = 1$ : assumendo  $p=0$  (nessuna correl.  
 tra le sue feature, le due feature si pongono (a loro  
 ad eguale distanza uguale alla std. dev.).

Il decision boundary è una retta ortogonale al  
 seguente dei collega: due centroidi (coincide con  
 il chiaffriccio lineare di box)

3.2) Fidiamoci l'ipotesi che le due deviazioni std.  
 siano uguali, ma poniamo la correlazione tra  
 feature:  $\sigma_1^+ = \sigma_1^- = 1, \sigma_2^+ = \sigma_2^- = \sqrt{2}$  e  $p^+ = p^- = \sqrt{2}/2$ :  
 il decision boundary è sempre lineare ma non coincide  
 con il chiaffriccio lineare di box (la retta non è  
 più ortogonale alla retta che contiene i due centroidi)

3.3)  $p$  diverso per le due classi: decision boundary non

è l'unico luogo di punti: bensì si è ora decisa in favore  
dell'area dell'altra classe

3.4) Generalizzazione della Gaussiana nel caso d-variano:

$$P(x|\mu, \Sigma) = \frac{1}{E_d} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right)$$

$$\text{con } E_d = (2\pi)^d / 2^{\frac{d}{2}} |\Sigma| \quad \text{e} \quad \mu = (\mu_1, \dots, \mu_d)^T$$

Riconosciamo la formula delle distanze di Mahalanobis.  
In questa versione si usa la covarianza per normalizzare  
le distanze tra i punti ed il centroide: il clou.  
di base coincide con il bayes-ottimo con la matrice  
di cov.  $\Sigma$  quadrata

$$LR(x) = \frac{\sqrt{|\Sigma^+|}}{\sqrt{|\Sigma^-|}} \exp\left(-\frac{1}{2} [(x-\mu^+)^T (\Sigma^+)^{-1} (x-\mu^+) - (x-\mu^-)^T (\Sigma^-)^{-1} (x-\mu^-)]\right)$$

Assumendo però che le  $\Sigma$  siano uguali ed invertibili  
(cioè che le feature delle due classi NON abbiano correl.)  
avremo ovviamente che:

$$LR(x) = \exp\left(-\frac{1}{2} [||x-\mu^+||^2 - ||x-\mu^-||^2]\right)$$

Ma visto che siano insieme stati ad ottenere  $L(x) = 1$  cioè il suo esponente  $= 0$ . Assumendo  $\Sigma = \mathbb{I}$  siano nel caso appena descritto  $\forall x$  equidistanti da  $\mu^+$  e  $\mu^-$ . Siamo perciò al primo caso: il classificatore è bayes-ott. quando le due classi hanno stessa std. dev. e non hanno alcuna correlazione.

↪ caso generale:

- se le due  $\Sigma$  sono uguali: d.b. ancora lineare e interseca nel vettore  $(\mu^+ - \mu^-)$  ma non gli sarà ortogonale.
- se le due  $\Sigma$  non sono uguali: d.b. iperbolico.

• legame tra interpretazione geometrica e probabilistica:

Abbiamo visto come la distanza di Mahalanobis sia ad esaurire della formula della gaussiana.

Siamo in statistica usiamo la likelihood come arg. di un logaritmo (negativo) uguagliato a zero, dato che ci si concentra sul Terreno a fronte: Ci semplifica la vita quando studiamo a vedere come calcolare la likelihood dell'intero dataset.

• Stima delle massime likelihood:

→ Se vogliamo stimare i parametri ( $\mu$  e  $\Sigma$ ) di una Gaus. multivariata la max likelihood ci permette di partire dai dati e stimare quale potrebbe essere il valore dei param. che maximizza la likelihood.

→ Stima di  $\mu$ : avendo  $X$  indipendenti, da cui conosce già  $\Sigma$ , la likelihood corrisponde più ovviamente alla scrittura del prodotto su ogni singolo  $x \in X$  e la stima della Massima Likelihood sarà:

$$\hat{\mu} = \arg \max_{\mu} \prod_{x \in X} P(x | \mu, \Sigma), \text{ e visto che ci vogliono}$$

molte, applichiamo i seguenti passi algebrici per generalizzare il teorema sugli algoritmi di clustering:

$$= \arg \max_{\mu} \prod_n \frac{1}{E_d} \exp \left( -\frac{1}{2} (\text{Dis}_n(x, \mu | \Sigma))^2 \right) =$$

$$= \arg \max_{\mu} \sum_{x \in X} \left[ \ln E_d + \frac{1}{2} (\text{Dis}_n(x, \mu | \Sigma))^2 \right] =$$

$$= \arg \max_{\mu} \sum_{x \in X} (\text{Dis}_n(x, \mu | \Sigma))^2$$

E troviamo che le  $\hat{\mu}$  è il punto che minimizza la distanza di Mah. quadratica totale in realtà i punti di  $X$ .

• Modelli probabilistico per dati su numerici:

Prendiamo l'esempio della cosparsa di parole in testi.  
Poniamo tattiche quante feature categoriche in 2 modi:

- 1) Indicando con un bool se una parola è presente o no
- 2) o con una var. continua la sua freq. di cosparsa.

1) Utilizziamo una Bernoulliana (mutuamente) per ogni parola che compare nel documento.

$$\begin{cases} \theta = P(X=1) \\ 1-\theta = P(X=0) \end{cases} \rightarrow \begin{cases} E(X) = \theta \\ E[(X-E(X))^2] = \theta(1-\theta) \end{cases}$$

Generalizzando su  $n$  esperimenti abbiamo una binomiale che conta i  $k$  successi:

$$P(S=k) = \binom{n}{k} \theta^k (1-\theta)^{n-k} \quad \begin{cases} E(X) = n\theta \\ E[(X-E(X))^2] = n\theta(1-\theta) \end{cases}$$

2) Utilizziamo una dist. multinomiale per misurare le frequenze, dando in input in esame i category:

$$P(X=(x_1, \dots, x_d)) = n! \frac{\theta_1^{x_1}}{x_1} \cdots \frac{\theta_d^{x_d}}{x_d} \quad \text{con } n = \sum_{i=1}^d x_i$$

dove la  $K$ : a denominatore a tenere conto delle tutte le disposizioni delle parole.

- Naive Bayes: bayes Theorem + ipotesi che le feature siano indipendenti  $\rightarrow$  riconoscimento più che verosimile visto che l'esame sono le parole nelle pagine: ci permette di avere probabilità condizionate molto = plausibile tra di loro.

Formalmente:  $P(y|x_1, \dots, x_n) = \frac{P(x_1, \dots, x_n | y) P(y)}{P(x_1, \dots, x_n)} =$

$= P(x_1, \dots, x_n | y) P(y)$  [togliere il denominatore una volta scelta la classe]  $\leftarrow$  valore likelihood da massimizzazione.

- MAXIMUM LIKELIHOOD:  $\operatorname{argmax}_y P(X|Y)$
- MAP: utile quando non possiamo assumere egiprobabilità delle classi  $\rightarrow \operatorname{argmax}_y P(X|Y) P(Y)$
- RECALIBRATION LIKELIHOOD: si contano anche i pesi delle singole probabilità  $\rightarrow \operatorname{argmax}_y w_y P(X|Y)$

↳ esempi di calcoli: da guardare su appunti

• Logistic Regression: modello discriminativo che mira a trovare il decision boundary tra le due classi nell'immagine.

→ Combinare feature descrittive degli esempli  $x_i$  con dei pesi  $\beta_i$ .

→ Modello applicabile solo a valori continuo

→ Vogliamo predire uno score di appartenenza: per farlo modelliamo la probabilità di trovarci in una classe con una regressione lineare.

$y_i$	$P(y_i   x_i)$
1	$P(y_i = 1   x_i) = p$
0	$1 - P(y_i = 1   x_i) = 1 - p$

↳ Utilizziamo una Bernoulli con  $k=1$  di successi, rappresentiamo la distribuzione:

$$f(y_i | x_i) = p^{y_i} [1 - p(y_i | x_i)]^{1-y_i}$$

$$\left\{ \begin{array}{l} \text{Dove vogliamo calcolare } E[y_i | x_i] = p(1 | x_i) \\ \text{con } \text{Var}[y_i | x_i] = p(y_i | x_i) [1 - p(y_i | x_i)] \end{array} \right\}$$

Quindi l'idea è di prevedere le prob. della classe positiva una volta inserito l'esempio  $i$ -esimo con una regressione lineare:

$$P(Y_i = 1 | \mathbf{x}_i) = \underbrace{\beta_0 + \beta_1 x_{i1} + \dots + \beta_d x_{id}}_{\text{comb. lineare}}$$

Applicando poi una f. simile che limita il risultato da  $[-\infty, +\infty]$  a  $[0, 1]$  così che diventa una probabilità

$$\hookrightarrow P(Y_i = 1 | \mathbf{x}_i) = \frac{\exp(\text{comb. lineare})}{1 + \exp(\text{comb. lineare})}$$

$\hookrightarrow 0$ : la classe predetta è quella negativa

$\approx 1$ : " " " " " " positiva

Per semplificare i calcoli si introduce l'odds-ratio = rapporto fra le 2 prob. opposte:

$$\exp(\text{comb. lineare}) = \text{Odds}(Y_i | \mathbf{x}_i) \quad \text{e}$$

$$\ln(\text{Odds}(Y_i | \mathbf{x}_i)) = \underbrace{\text{comb. lineare}}$$

Scorre delle regressioni lineari,

ovvero ricordiamo:  $\beta_0 + \beta_1 x_{i1} + \dots + \beta_d x_{id}$

- Modelli lineari generalizzati: la regressione logistica utilizza le regressioni per predire  $\ln(\text{odds ratio}) \rightarrow$  in caso particolare dei modelli lineari generalizzati, ovvero modelli che dopo aver calcolato la f. di regressione che produce un target  $\hat{y}_i$ , ne applicano a questo una funzione  $f$ :  $f(\hat{y}_i) = g(x_i) = w_0 + w_1 x_{i1} + \dots + w_d x_{id} = \mathbf{w}^T \mathbf{x}_i$

Dove  $\delta(Y_i)$  può essere:

- $\delta(Y_i) = Y_i$  : caso + semplice
- $\delta(Y_i) = \frac{1}{Y_i}$
- $\delta(Y_i) = \log(Y_i)$
- $\delta(Y_i) = \log(Y_i / 1 - Y_i) \rightarrow$  caso della regressione log. che abbiamo appena visto!

• Rappresentare logistica per feature categoniche: (casuali)

→ δ: sostituisce con gli Odds nello f. di regressione:

- esempio:

X/Y	1	0	Total
A	$M_{A1}$	$M_{A0}$	$M_A$
B	$M_{B1}$	$M_{B0}$	$M_B$
Total	$M_1$	$M_0$	$M$

$$\Rightarrow \begin{cases} P(Y=1 | X=A) = \frac{M_{A1}}{M_A} \\ P(Y=0 | X=A) = \frac{M_{A0}}{M_A} \end{cases}$$

e con gli Odds (A) =  $\frac{P(Y=1 | X=A)}{P(Y=0 | X=A)} = \frac{M_{A1}}{M_{A0}}$

e con gli Odds (B) =  $\frac{P(Y=1 | X=B)}{P(Y=0 | X=B)} = \frac{M_{B1}}{M_{B0}}$

e sostituendo gli odds nello f. delle regressione:

$$\ln(\text{Odds}(Y_i | X_i)) = \beta_0 + \beta_1 X_{i1} + \dots + \beta_d X_{id}$$

e rischia con la nozione del libro:

$$\hat{P}(x) = \frac{\exp(w \cdot x - t)}{\exp(w \cdot x - t) + 1} = \frac{e^{w \cdot x - t}}{1 + e^{-(w \cdot x - t)}}$$

e la distribuzione di Bernoulli:  $P(y_i | x_i) = \hat{P}(x_i)^{y_i} (1 - \hat{P}(x_i))^{(1-y_i)}$

Dove:  $-t$  è il termine errore ( $\beta_0$ )

- $w$  vettore dei pesi
- $x$  " " " dati
- $-t + w \cdot x$  è la regressione lineare

Risolviamo anche le formule della likelihood e la sua forma logaritmica:

$$CL(w, t) = \prod_i P(y_i | x_i) = \prod_i \hat{P}(x_i)^{y_i} (1 - \hat{P}(x_i))^{(1-y_i)}$$

$$LCL(w, t) = \left( \sum_i y_i \ln \hat{P}(x_i) + (1 - y_i) \ln (1 - \hat{P}(x_i)) \right)$$

1: Oggetto appartiene alla classe 1  
0: oggetto non

$$= \sum_{x_i \in T^+} \ln \hat{P}(x_i) + \sum_{x_i \in T^-} \ln (1 - \hat{P}(x_i))$$

- Si cercano i parametri delle leggi di distribuzione che rende verosimile il dataset: confrontare a 0 le d

derivate parziali rispetto a  $w$  e trovare così il  
MASSIMO DEL LOGARITMO DELLA LIKELIHOOD:

$$\nabla_w LCL(w, \tau) = 0 \quad \text{e} \quad \frac{\partial}{\partial \tau} LCL(w, \tau) = 0$$

$$\textcircled{1} \quad \sum_{x_i \in T^+} (\hat{p}(x_i) - 1) + \sum_{x_i \in T^-} \hat{p}(x_i) = \sum_{x_i \in T} (\hat{p}(x_i) - y_i)$$

Ora, l'errore comeva = errore alla classe positiva dell'esempio della predizione!

↪ Se è 0: in media la  $\hat{p}(x_i)$  predetta è uguale alle proporzioni degli esempi positivi nel dataset.

L'altra derivata parziale della LC è rispetto ai suoi pesi:  $\frac{\partial}{\partial w_j} = \sum_{x_i \in T} (y_i - \hat{p}(x_i)) x_{ij}$

La sua derivata sarà uguale a zero (valore ottimo) esattamente quando  $y_i = \hat{p}(x_i) \forall x_i \rightarrow$  se una feature non esiste, non viene perciò pesata...

Ponendo a zero queste 2 derivate parziali si ricava  
man mano i valori dei parametri  $w$  e  $\tau$ :

$$w^*, \tau^* = \arg \max_{w, \tau} CLL(w, \tau) = \arg \max_{w, \tau} LCL(w, \tau)$$

↳ Questo è un problema di ottimizzazione convesso: ho un solo massimo e a cui posso applicare un metodo di risalita come quello del percorso e il suo metodo di aggiornamento dei pesi:

$$w = w + \eta(y_i - \hat{p}_i) k_i$$

• Algoritmo di EM (expectation - maximization):

- Algoritmo di clustering (k-means probabilistico)
- Produce una legge generativa dei cluster
- Associa ad ogni esempio una serie di probabilità di appartenenza al cluster.
- Idea: guardare gli esempi e trovare la legge generativa del cluster con max likelihood.

Dallo 2 step, niente iterazione:

1) EXPECTATION STEP: vengono fatti i parametri del cluster e determinato a quale cluster ci sia più probabilità di appartenere.

2) MAXIMIZATION STEP: una volta associata a ciascuno esempio il peso di appartenenza ad un cluster, si

mao i per far maggiore i parametri del cluster.      funzione di dist.      Probabilità di prob.

$$\uparrow \text{Expectation step: } \hat{P}(C_i | x_i) = \frac{\underline{g}(x_i | C_i) P(C_i)}{\underline{g}(x_i)} = w_{i,j}$$

Attribuisce ad ogni esempio  $x_i$  la sua prob. di appartenere al cluster  $C_i$ , dove,

$$g(x_j | C_i) = g(x_j; \mu_i; \sigma_i) = \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{(x_j - \mu_i)^2}{2\sigma_i^2}}$$

→ NOTA: nelle formule si moltiplica una funz. di distribuzione da prob. per una prob.: le formule funzione dove che al denom. abbiano una f. ed esse si annullino.

$w_{i,j}$  = probabilità per l'esempio  $j$ -esimo di appartenere al cluster  $i$ -esimo.

↑ Maximization step: si usa il principio della maximum likelihood estimation. Essendo la likelihood definita dalle formule:

$$L(\chi, \mu_i, \sigma_i) = \prod_j g(x_j) = \prod_j \sum_i P(C_i) \frac{1}{\sqrt{2\pi}\sigma_i} e^{\left(\frac{-(x_j - \mu_i)^2}{2\sigma_i^2}\right)}$$

Per minimizzare  $L(x, \mu_i, \tau_i)$  dobbiamo trovare ed applicare i 3 stimatori fatti:

$$1) \text{Probabilità del cluster } P(C_i) = \frac{1}{m} \sum_{j=1}^m w_{ij}$$

$$2) \text{Centroide per il cluster } i\text{-esimo: } \mu_i = \frac{w_{i1}x_1 + w_{i2}x_2 + \dots + w_{im}x_m}{w_{i1} + \dots + w_{im}}$$

$$3) \text{Lo scarto quadratico del cluster: } \tau_i^2 = \frac{w_i(x_i - \mu_i)^2 + \dots}{w_{i1} + w_{i2} + \dots}$$

↑ Condizione di terminazione: si calcola il log. della likelihood attuale e quello precedente: se non c'è miglioramento ci fermiamo.

- Modelli basati sulla comprensione:

Sai che sappiamo che l'entropia rappresenta il limite teorico di compressibilità di un contenuto informativo, poniamo di avere questa quantità dell'interno del logaritmo della KAP:

$$J_{\text{KAP}} = \text{argmax}_y -\log P(x=x | y=y) - \log P(y=y)$$

$$\text{IC}(x|y) = -\log_2 P(x|y) \quad \text{and} \quad \text{IC}(y) = -\log_2 P(y)$$

- esercizio delle applicazioni su slides / appunti

→ Vogliano applicare questo principio ai modelli:

Dati:

- $L(m)$ : complessità del modello ( $\ln$  bit)
- $L(D|m)$ : errori commessi sul dataset  $D$  dal modello  $m$  (lunghezza in bit della descrizione di un dato  $D$  in un modello  $m$ )

Allora il modello, scelto con la sua minima descrizione length, sarà:

$$m_{\text{min}} = \arg \min_m (L(m) + L(D|m))$$

## FEATURES E MODIFICAZIONE

- Definizione: una feature è un mapping  $f_i : X \rightarrow F_i$  dallo spazio delle istanze  $X$  al Feature Space  $F_i$ .
- Distinguiamo tra loro le features da:
  - DOMINIO  $X$ : numeri reali, interi, colori, booleani...
  - POSSIBILI OPERAZIONI che faccio effettuare: esistono grande varietà di scelte di misurazione!
- Manipolare le feature, ovvero:
  - Cambiare il dominio: flessibili, discrittive
  - Partizionare set biglii e lavorare su quelli piccoli

• Statistica sulle Features: si divide in 3 parti:

- 1) Statistiche di central tendency (tendenza di una distribuzione verso il suo centro)
- 2) statistiche di dispersione
- 3) statistiche di forma (shape)

### 1) Statistiche di central tendency:

- La media
- la mediana (il middle value)
- la moda (il valore più frequente)

Tipo	Mode	Median	Mean
Categorial	YES	NO	NO
Ordinal	YES	YES	NO
Quantitative	YES	YES	YES

- ⇒ La media può essere computata differentemente a seconda delle sue scole:
- a) Lineare: media aritmetica
  - b) Reciproche: media aritmetica
  - c) Frequenze / uscite: media aritmetica

## 2) Statistiche di dispersione:

- Varianza (che è additiva per v.a. non correlate)
- Deviazione standard (esprime sulle stesse scale delle feature)
- Range / midrange:  $(b-a)$  e  $\left(\frac{b-a}{2}\right)$
- Percentili: (ad es. il p-esimo perc. è il valore t.c. il p-percentile dei valori ricadono sotto se) → 25° sono quantili, 10° e molti più sono decili.



### Ricordiamo 2 formule:

- Varianza ( $x$ ) =  $\sigma^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \mu)^2$
- Deviazione standard ( $x$ ) =  $\sqrt{\sigma^2} = \sigma$

↳ Per la diseguaglianza di Chebysev la differenza tra le media e la mediana non è mai + grande di  $\sigma\sqrt{2}$ .

- Percentile Plot: plot dei percentili che può avere lo stesso aspetto di una distribuzione cumulativa di probabilità.

### 3) Statistiche di forma (shape):

→ Skewness:  $\frac{m_3}{\sqrt{3}}$  → (+) coda a dx > coda a sx

→ Kurtosis:  $\frac{m_4}{m_2^2}$  → se + si usa la  $k^2$

→ Momento centrale k-esimo:  $m_k = \frac{1}{n} \sum_{i=1}^n (x_i - \mu)^k$   
dove  $\mu_i$  è la media dell'esempio.

a) Primo momento centrale  $= m_1 = 0$

b) Secondo " "  $= m_2 = \text{Varianza}$  (sempre +)

c) Terzo " "  $= (+)$  se coda a dx,  $(-)$  coda a sx.

$k^2$ : Excess Kurtosis  $= \frac{m_4}{m_2^2} - 3 = (+)$  se la distribut.  
è più appuntita della normale.

Date le statistiche facendo un recap sui dati/features:

Tipo	Ordine	Soda	Tendenza	Dispersione	Shape
Categor.	x	x	moda	u/a	u/a
Ordinale	✓	x	median	quartili	u/a
Quantitativa	✓	✓	media	range, interq., var, std dev.	skew, kurt.

• Pari i modelli vediamo con cui trattano le feature:

- Modelli probabilistici trattano ogni feature come ORDINALE
  - Modelli ad albero ignorano le quantitative e trattano tutto come ORDINALE.
  - Modelli geometrici parlano solo modellare feature quantitative.
  - Structured Feature: variabili costruire prima del learning model (sono i literali di Prolog, bruh)
- 

### Trasformazione di Feature:

At Da	Quantitativa	Ordinale	Categorica	Boolea
Quant.	NORMAIZZAZ.	CAUBNZAIS	CAUBNIS	CAUBNIS
Ordinale	DISCRETEZZ.	DISCRETEZZ.	DISCRETEZZ.	DISCRETEZZ.
Categorica	DISCRETEZZ.	DISCRETEZZ.	DISCRETEZZ.	DISCRETEZZ.
boolea	THRESHOLDING	THRESHOLDING	THRESHOLDING	THRESHOLDING

↳ **NORM.** e **CAUB**: adattano le scate delle feature quant.  
o aggugno una scata e feature che non ne ha una.

↳ **ORDENING**: aggiunge o adatta l'ordine delle feature sulla sua referenza ed una scelta.

↳ **UNDERRMING, BINARIALIZATION**: ottengono dettagli sui meccanismi del modello DEDUTTIVA

↳ **TRANSFORMATIONS, DISCRETIZATION**: ottengono dettagli sui meccanismi introducendo nuove informazioni.

(UNSUPERVISED)

- Thresholding: discretizzazione del dataset, suddividendolo in intervalli (bins): trasforma sue feature quantitative in una ordinale.
- Discretization: (UNSUPERVISED)

1) Equal-width (distance) partitioning: divide il range in  $N$  intervalli di stessa grandezza. La via più facile ma sensibile ad outliers e una suddivisione bene dati ad alta svarianza!

2) Equal-frequency (frequency) partitioning: divide il range in  $N$  intervalli che contengono approssimativamente lo stesso numero di esempi, ha un buon dato scaling ma è più complesso.

• Thresholding: (SUPERVISED)

Potremo usare le feature come ranker per contrassegno

curve di copertura e nelle curve individuare i punti dove meglio "tagliare" le nostre features, sotto stress nelle Conca salvare le curve.

### • Discretizzatore: (SUPERVISED - RECURSIVE PARTITIONING)

Punto di partenza: istante - valore - classe, ci ritroviamo con una classifica come la seguente:

$$(- + \text{---} + + [-+] | \text{---})$$

↑ dove inizia a mettere

split come questi volta per volta...

$$(- + \text{---} | + + [-+] | \text{---})$$

E così via, tracciamo i simboli split con colori di impurità (entropia) e alle fine costruiamo una coverage curve con quelli che ci interessa!

↳ Algoritmo RecPart (S, f, Q): (recursice DIVISION PARTITIONING)

Idee:

$S$  = set di istante pre-labeled

$f$  = feature values  $f(x)$

$Q$  = funzione di scelta

if stopping-criterion () :

return  $\delta$

$S_1, S_r = \text{split}(S, \text{threshold} = \tau, \text{optimality} = Q)$

$T_1 = \text{RecPart}(S_1, g, Q)$

$T_r = \text{RecPart}(S_r, g, Q)$

return  $T_1 + \tau + T_r$

ci basano su information GAIN

↳ Algorithm AggloMerge( $S, g, Q$ ): RECURSIVE AGGLOMERATIVE CLUSTERING

bins = data-points (with = scorescore)

bins. merge-consecutive-pair-bins()

while ( bins. is-mergeable() ):

best\_q = bins. evaluate(Q)

bins. merge(best\_q)

return bins. thresholds

Score calcolati

con il chi quadro

•  $\chi^2$ : esercizio con ... [-+] | --- ] ↴

LEFT BIN		MIDDLE BIN	
+	1	0	1
-	1	3	4
=	2	3	5

} Valori di  $\chi^2$  bassi sono  
merge molto buoni, aver  
conservato una buona  
dipendenza tra le  
caratteristiche

↳ Poi si confronta con le attese...

- Normalizzazione:

1) Standardizzazione: se le feature sono distribuite normalmente posso convertirle in z-scores  $z = \frac{x-\mu}{\sigma}$  sottraendo la media e dividendo per l'asd. dev.

2) MIN-MAX: escludo le feature da una scala [0,1], se conosco il min e max della feature posso applicare il linear scaling.  
 $f = (\frac{f - \text{MIN}}{\text{MAX} - \text{MIN}})$ , se ci sono valori di  $f > \text{max}$  ad uno, speso li trasduco a 1.

- Calibrazione: trasformazione di feature supervised aggiungendo una scala che fornisca informazioni di classe a features arbitrarie.

Dato una feature  $F: X \rightarrow F$ , si considera una feature calibrata  $F^c: X \rightarrow [0,1]$  t.c.  $F^c(x) \approx P(+|v)$  dove  $v = F(x)$  originale.

↳ Per feature continue la calibrazione è semplicemente l'atto di collezionare frequenze relative al training set. (esempio su appunti - pag)

→ Se la distr. di classe non è UNIFORME, dobbiamo evitare di dare importanza alle classi prime

delle calibrazione delle feature che meglio rappresenta.  
 Introduciamo allora i prior odds  $c = P(+)/P(-)$   
 e calcoliamo gli odds a POSTERIORI come segue:

$$P.O. = \frac{P(+|f)}{P(-|f)} = \underbrace{\frac{P(f|+)}{P(f|-)}}_{(O)} \cdot \underbrace{\frac{P(+)}{P(-)}}_{(C)} \quad \text{dove } f = \text{feature value}$$

↓                          ↓                          ↓  
 $O = p/(1-p)$        $\ell = O/c$        $P = \frac{m}{m+n} \quad I = \frac{m}{m+n}$

↑ *positive*

← utilizziamo la likelihood ratio per togliere l'effetto delle p. a posteriori delle p. a priori.

$$\text{calibrated\_feature}(x) = \frac{m}{m+c(n-m)}$$

$$\text{calibrated\_feature\_laplace}(x) = \frac{m+1}{m+1+c(n-m+1)}$$

es: se sappiamo che le prior odds di avere il diabete è  $(1/48 = c)$ ,

↪ Per un  $x$  alto:  $F^c(x) = \frac{1}{1 + \frac{1}{48} \frac{(18-1)}{n-m}} = \underbrace{0.74}_{> 0.5} \quad \underline{\text{ALTO RISCHIO}}$

↪ Per un  $x$  basso:  $F^c(x) = \frac{1}{1 + 1/48(55-1)} = 0.47 (< 0.5) \quad \underline{\text{BASSO RISCHIO}}$

## Calibrazione Per feature ordinali e quantitative:

→ Primo si discretizzano, poi si calibra con se stesse feature categòriche.

↳ Nella calibrazione però bisogna mantenere l'ordine dei valori, così da avere probabilità crescenti con i loro pesi (detti dai rank).

1) **Calibrazione logistica**: come una distribuzione parabolica dei valori, fa le seguenti cose =

a) Stima le medie  $\mu^+$   $\mu^-$  delle classi e la  $\sigma$ .

b) Transforma i valori delle features  $F(x)$  in z-score

$$z = \frac{x - \mu}{\sigma} \quad || \quad \mu = \frac{\mu^+ + \mu^-}{2}$$

c) Riscola gli z-score a  $F^d(x) = d'z(x)$  con

$$d' = \frac{\mu^+ - \mu^-}{\sigma}$$

d) Applica la sigmoida a  $F^c(x) = d'z(x)$  per ottenere delle prob. calibrate:

$$F^c(x) = \frac{\exp(F^d(x))}{1 + \exp(F^d(x))}$$

2) Calibrazione Isotonica: mappare monotonicamente le crescenze delle prob. originali ai valori ri-scalati.

- Ordina le istante di Training a seconda dei valori delle feature originali e crea la curva Roc.
- Crea il convex hull e per ogni seguente calcola il numero di elementi positivi e il numero totale di istante
- Discretizza le feature a seconda del seguente del convex hull. I singoli valori delle feature calibrate saranno calcolati A seguendo i -esimi così:

$$V_i^c = \frac{m_i + 1}{m_i + \ell + C(m_i - m_i + 1)}$$

- Se una scelta additiva è richiesta, usa:

$$V_i^d = \ln\left(\frac{V_i^c}{1 - V_i^c}\right) = \ln(V_i^c) - \ln(1 - V_i^c)$$

- Discretizzatore e feature riuonose: la discretizzazione può essere utile perché sostituisce una serie di valori che ricadono in un solo bin con un solo valore discreto. Questo può essere utilizzato per smussare i valori riuonosi quando valori incerti sono presenti.

Quando troviamo del rumore sotto forma di un valore mancante dobbiamo trovare quel valore mancante:

- a) Un problema di classificazione parcoo trovare media, mediana o moda per calcolare ed utilizzare il loro valore per quello mancante
- b) Possiamo addestrare un modello per gli ambienti dei valori mancanti e usare il modello predittivo per trovare i valori.

• Costruire nuove features:

- Divene parole poche e rare combinarle in bag o n-grammi.
- Prendere il decisio di 2 features e fare il prodotto cartesiano
- Prendere le combinazioni aritmetiche / polinomiali di feature quantitative (Metodi Kernel)
- Apprendere su sotto-grafso con subgroup discovery e applicarci una feature booleana:  
$$\begin{cases} \text{length} = [3, 5] \wedge \text{gills} = \text{no} \rightarrow B = \text{true} \\ \text{length} > [3, 5] \vee \text{gills} = \text{yes} \rightarrow B = \text{false} \end{cases}$$

• Selezionare nuove features: per velocizzare il learning, per evitare overfitting, ridurre la curva di dimensionality

scegliano di selezionare le feature in 2 modi:

a) Filter approach: seleziona le feat. guardando gli scores con una metrica (spesso supervisionata) come  $\chi^2$  e IG.

Problemi coi:

→ Ridondanza delle features

→ Non considera la dipendenza tra features

b) Relief approach: rivede l'estrazione di  $x$  dal training set. Se trova il più vicino HIT h (più vicino ègualo delle stescole) e il più vicino MISS m.

Utilizza m ed h per calcolare uno score  $f_i$  in modo che:

→ Si alzi con  $\text{Dist}(x, m_i)^2$

→ Diminuisce con  $\text{Dist}(x, h_i)^2$

Dopo m iterazioni, dividiamo il risultato per m

E così ottieno il relevance score per le feature

• Algorithm Relief(D):

$$f = [0, \dots, 0]$$

for i in range(0, m):

$X_i = \text{extract\_random}(D)$

$h_i = f_i \cup \text{nearest\_hit}(D, \text{same\_class} = X_i)$

$$M_{ij} = \text{final-means} - \text{miss}(D, \text{opposite-class} = k_j)$$

Fr  $\delta_T$  in  $\delta_i$ :

$$\delta_T = \delta_T - (x_{ij} - h_{ij})^2 + (k_{ij} - m_{ij})^2$$

Retrui  $f/m$

c) Wrapper approach: rileva i set di features nel quale una feature è utile per il learning model, considerando il contesto.

Problema: il set di feature cresce esponenzialmente con il numero totale di feature possibili.

→ Forward Selection: parte da un singolo feature e aggiunge al set singole features

→ Backward Elimination: parte da un set con tutte le possibili features e ne toglie una a mano.

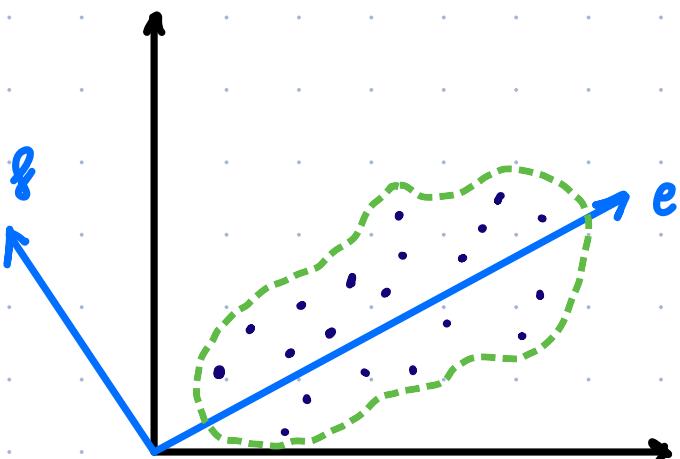
• PCA: Principal Component Analysis: metodo di costruzione e selezione di features.

→ GOAL: trovare una proiezione che catturi le più grande variazioni nei dati; serve a ridurre le dimensioni di un vettore perdendo meno info possibili.

→ A cosa serve: alle volte non ho senso tenere tutte le feature (fuorileggi, duplicati...) che hanno tante dimensioni, dunque me riducono le dimensioni anzim (il numero di classi per feature).

→ PROCEDIMENTO:

- 1) Sposta i dati tale che la media di ogni dimensione sia a zero
- 2) Trova gli autovettori della matrice di scatter S
- 3) Gli autovettori definiscono il nuovo spazio di rappresentazione delle istanze dei dati.



•  $e_1, e_2$  nuove feature  
costruite su cui calcoliamo  
la varianza (e scegliono  
le istanze su  $e$ )

↳ sono i nuovi "x, y"  
Piani delle rotazioni

del piano per cercare la max varianza.  
utilizzando:

- Matrici di scatter (servono per calcolare le z.f.)
- Autovettori (e autovettori)
- Guardare video su slides

## ENSEMBLE LEARNING

- Mischie di tecniche per le risoluzione di tecniche di classificazione / regressione combilandone risultati di più algoritmi di apprendimento.
- Apprendimento forte: Su una certa classe di problemi si riesce a raggiungere quasi una precisione di 1.
- Apprendimento debole: Su una certa classe di problemi si riesce a far meglio di una predizione casuale pari a 0.5 ( $\text{prec} > 0.5$ ).

Dato un alg. debole si può riuscire a costruire un fusore? Se fusore possibile riuscire sempre a risolvere in modo forte qualsiasi problema.

• Come funziona l'app. ensemble:

- 1) Dato un dataset di paranza  $D$  si ottengono  $T$  nuovi dataset, dove  $T$  arbitrario.
- 2)  $\forall T$  dovrà essere imposto dell'alg. di apprendimento debole che genererà  $T$  modelli
- 3)  $\forall T$  modelli: vengono così combinati in un  $H$  finale con il quale risolviamo il problema di apprendimento.

- Come combinare diversi diversi modelli?

$$H(x) = f\left(\sum_{i=1}^T w_i H_i(x)\right)$$

Dove:

- $x$  è un esempio del dataset
- $H_i(x)$  è la classificazione singola di  $x$  dal class. i-class.
- $w_i$  è il peso associato al modello  $H_i$
- $w_i > w_j$  vuol dire che  $H_i$  è più affidabile del modello  $H_j$ .
- $w_i > 0$  solitamente

- Come generare diversi modelli di base?

- 1) BAGGING (bootstrap aggregation)
- 2) BOOSTING (sopracorso AdaBoost)
- 3) ECOC (error-correcting output codes)
- 4) STACKING

- Bagging:

dataset dimensione  
↓ ↓ algoritmo apprendimento d-

- 1) Algoritmo Bagging ( $D, T, A$ ): base

models = []

for  $t$  in range(1, T):

Costruisce repliche di bootstrap ( $D$ )

$$\text{models}[\bar{i}] = A(D)$$

Hillme combine (models, method = "majority")

// insieme di modelli le cui predizioni

verranno combinate in un solo modello finale

## 2) Spiegazione intuitiva:

→ Si costruiscono dei nuovi dataset per addestrare  $T$  modelli

→ Si utilizzerà il metodo di bootstrap sul dataset da fare con  $D$  (cambiando le stesse righe)

→ Verranno combinati alle fine i modelli per meccanismo di maggioranza, cioè se avranno tutti i per  $w_i = 1$ , percepirà il modello risultante:

$$M(x) = f\left(\sum_{i=1}^T M_i(x)\right)$$

Dove  $f$  è la funzione di segno:

$$\begin{cases} +1 & \text{se somma positiva} \\ 0 & \text{se somma nulla} \\ -1 & \text{se somma negativa} \end{cases}$$

→ Si usano repliche di bootstrap diversificate dall'estrazione, senza riutilizzare delle replicate precedenti.

↳ lo percezionale di essere che venga esclusa in ogni replica è del 37%, a dire che cosa segue:

la p che in  $\chi$  esce più non faccia parte della terza è  $(1 - \frac{1}{m})^m$ , il suo limite per  $m \rightarrow \infty$  è  $\frac{1}{e} = 0.37$  cca.

3) Random Forest: algoritmo di apprendimento (A) per il bagging con alberi di decisione non forti come algoritmo di apprendimento debole.

- Noto sull'apprendimento: una classe di concetti e' FONDAMENTALMENTE APPRENDIBILE se  $\exists$  un algoritmo che in TEMPO POLINOMIALE ottenga in BASSO ERRORE con alta certezza. I modelli ottimi ottengono in processo DEBOLI al contrario si caratterizza di raggiungere risultati poco migliori di scelte casuali.
- Adaptive Boosting: ab. permette di ottenere che una classe di concetti C può essere appresa debolmente se questa può essere appresa forteamente!  
↳ Dato un alg. A in grado di fornire una sol. migliore di quella casuale, allora esiste un algoritmo di boosting in grado di fornire una soluzione accurata a piacimento che si basa su A!  
↓
- AdaBoost:  
→ Asseguire pesi a tutti gli esempi (modificare

le distribuzioni originali, non care nel bagging)  
 → si cerca di fare in modo che tutti gli esempi vengano classificati correttamente dal voto dei modelli: ottenuti da A.

• Note Adaboost:

- 1) I pesi iniziali sono tutti settati a  $\frac{1}{n}$
- 2) Essi vengono aggiornati nel seguente modo:
  - Diminuiti nel caso ci sia una pred. corretta
  - Aumentati " " " " + " " scometa.
- 3) Se A debole, ci si aspetta un errore < 0.5

• Algoritmo Adaboost (D, T, A):

$$W^t = \left[ \frac{1}{|D|}, \dots, \frac{1}{|D|} \right] // \text{Pesi iniziali associati agli esempi}$$

for  $t$  in range(1, T):

$$M_t = A(D, W[t]) // \text{l'alg. debole genera un modello}$$

//  $M_t$  misurando l'errore privato con il voto  $W[t]$

$$\epsilon_t = \sum_{i=1}^n W_i^t I[g_i \neq M_t(x_i)] // \epsilon_t \text{ è una misura}$$

// dell'errore complessivo del modello  $M_t$  appena

// addestrato: somma dei pesi associati ad esempi

// classificati NON correttamente.

$$\alpha_t = \frac{1}{2} \ln \left( \frac{1 - \epsilon_t}{\epsilon_t} \right) // \alpha_t: peso associato al t-esimo$$

// modello che verrà utilizzato quando alla fine  
// si dovranno combinare tutti i  $T$  in uno unico  
for  $i$  in range( $1, n$ ):

$$w_i^{t+1} = w_i^t \cdot \exp(-\alpha_t y_i m_t(x_i))$$

// Vengono  $w$ -pesati gli esempi del dataset.

// A peso  $w$ : del vettore della successiva iterazione

//  $t+1$  la formula riduce i pesi degli esempi

// classificati correttamente e aumenta gli altri

$$w^{t+1} = \text{Normalizza}(w^{t+1}) // normalizzazione ad 1$$

$$\text{fusione } M(x) = \text{sigm}(\sum_{t=1}^T \alpha_t m_t(x))$$

// Adaboost restituisce un modello che prende il

// segno della combinazione di tutti i modelli  $m_t$

// pesati con i loro pesi associati  $\alpha_t$

### Correttezza di Adaboost:

→ Adaboost può avere le più care in minimizzazione  
di una funzione di penalizzazione esponentiale pesata  
dell'intero sistema ensemble.  $\hookrightarrow M_T$

→ Algoritmo è greedy: ad ogni iterazione seleziona  
le ipotesi deboli e i pesi  $\alpha_t$  che hanno lo scopo di  
minimizzare la funzione  $M_T$ .

Si assume ora di aver già costruito fatto dei modelli  
e di essere arrivati a costruirne il  $T-1$ :

$$M_{T-1}(x) = \sum_{t=1}^{T-1} \alpha_t M_t(x)$$

Bisogna a questo punto focalizzare l'attenzione sulle scelte di  $M_T$  e  $\alpha_T$ . In particolare l'errore complessivo dell'ultimo modello, avendo  $H_T(x) = M_{T-1}(x) + K_T M_T(x)$ :

$E = \sum_i e^{-y_i} H_T(x_i)$  || errore totale calcolato come la somma dei singoli errori, calcolati come  $e^{-y_i} M_T(x_i)$ . Se la predizione è corretta, ovvero  $y_i H_T(x_i)$  sono concordi la segue, allora l'errore sarà neutro (dato che lo ignoriamo) e il valore totale sarà compreso tra 0 e 1 per una penalizzazione molto bassa. Al contrario a classificazione non OK la penalizzazione che viene eseguita viene esponentiale = multe (diventa  $> 1$ ) dato che abbiamo un errore positivo.

$= \sum_i e^{-y_i} (M_{T-1}(x_i) + K_T M_T(x_i))$  || Qui escludiamo l'errore applicando la def. di  $H_T(x)$  da prima

$= \sum_i e^{-y_i} M_{T-1}(x_i) - y_i \alpha_T M_T(x_T)$  || Distribuiamo  $y_i$  su entrambe le parti interne!

$= \sum_i e^{-y_i} M_{T-1}(x_i) \cdot e^{-y_i \alpha_T M_T(x_T)}$  || Applichiamo proprietà delle potenze isolando i 2  $e^{(\dots)}$

$= \sum_i w_i^T e^{-y_i \alpha_T M_T(x_i)}$  || Sostituendo il primo termine con  $w_i^T$  secondo la sua definizione

$= \sum_{\{i | y_i = M_T(x_i)\}} w_i^T e^{-\alpha_T} + \sum_{\{i | y_i \neq M_T(x_i)\}} w_i^T e^{\alpha_T}$  || Vengono separate

nelle due sommazioni gli esempi classificati corretti sono meno di quelli mali (...).

Nella prima sommatoria il prodotto di  $y_i m_T(x_i)$  è +1  
duque lasciamo  $-x_T$ , -1 nella seconda e quindi  $x_T$ .

$$= \sum_{\{i | y_i = m_T(x_i)\}} w_i^T e^{-\alpha_T} + \sum_{\{i | y_i > m_T(x_i)\}} w_i^T e^{\alpha_T} + \underbrace{\sum_{\{... \neq ... \}} w_i^T e^{-\alpha_T} - \sum_{\{... \neq ... \}} w_i^T e^{\alpha_T}}$$

Il Aggiungendo e sottraendo un'identica quantità  
che ci servirà per dividere + fruire come  
minimizzare le fct. err. dell'err.

$$= \sum_i w_i^T e^{-\alpha_T} + \sum_{\{i | y_i \neq m_T(x_i)\}} w_i^T (e^{\alpha_T} - e^{-\alpha_T})$$

con  $\begin{cases} e^{-y_i m_{T-1}(x_i)} & \text{se } T > 1 \\ 1 & \text{se } T = 1 \end{cases}$

Il raggruppando il 1° ed il 3° elemento: l'argomento identico  
e gli indici sono complementari. Anche le altre  
somma (2° e 4°) potranno essere duque raggruppate.  
Ci accorgiamo ora di cosa l'obiettivo sia solo più  
minimizzare l'errore, che si vede dal secondo termine.  
Come sempre per minimizzare, facciamo la deriva  
partiale (sta volta sul per  $x_T$ ) e lo mettiamo a

$\theta$  per trovare il punto di minimo.

$$\frac{\partial E}{\partial \alpha_T} = \sum_{\{i \dots \neq \dots i\}} w_i^r e^{\alpha_T} - \sum_{\{i \dots = \dots i\}} w_i^r e^{-\alpha_T}$$

e ricordando che  $E_T = \sum_{\{i \dots \neq \dots i\}} w_i^r$ , sostituendo e mettendo la derivata a 0,  $\theta$ .

$$\left( e^{\alpha_T} \sum_{\{i \dots \neq \dots i\}} w_i^r \right) - \left( e^{-\alpha_T} \sum_{\{i \dots = \dots i\}} w_i^r \right) = 0$$

$$e^{\alpha_T} \cdot E_T - e^{-\alpha_T} \cdot (1 - E_T) = 0$$

$$e^{\alpha_T} E_T = e^{-\alpha_T} (1 - E_T)$$

$$\frac{e^{\alpha_T}}{e^{-\alpha_T}} \frac{E_T}{1 - E_T} = \frac{e^{-\alpha_T}}{e^{\alpha_T}} \frac{1 - E_T}{E_T}$$

$$e^{\alpha_T - (-\alpha_T)} = \frac{1 - E_T}{E_T}$$

$$e^{2\alpha_T} = //$$

$$2\alpha_T = \ln(//)$$

$$\alpha_T = \frac{1}{2} \ln(//)$$

(le spiegazione algebrica viene riportata a pagina 240)

### • Data set Resampling:

- IDEA: dal dataset originale viene fatta una copia dove gli esempi vengono ritirati in proporzionale al loro peso

w associato. Questa operazione viene svolta ogni volta che vengono aggiornati

- Algorithm Resample ( $D, w, m$ ):

$D' = []$

for  $i$  in range( $1, m$ ):

$v = \text{random}(0, 1)$

$D'.append(K_w) // T.C. \sum_{i=1}^{n-1} w_i < v \leq \sum_{i=1}^n w_i$

ritorna  $D'$

Per  $\textcircled{m}$  volte inserisce con una prob. data dal peso di ciascun esempio l'esempio nel dataset.  
Con un  $m$  grande oppure si metterà bene le distrib. originali di  $D$ .

$m$  determina:

→ le dimensioni del nuovo dataset  $D'$

→ Più  $D'$  grande più sarà lo sforzo computazionale

→ Più  $m$  grande, maggiore è la precisione con cui  $D'$  rispecchia le distrib. determinante dei pesi.

↓

Quindi molto spesso  $|D'| = |D|$  scelto.

Analisi del funti quale è degli algoritmi ensemble:

1) Decomposizione Bias-Variance: gli alg. ensemble risultano efficaci nel ridurre sia il bias che la varianza. Il bagging tra questi è il migliore nel ridurre la varianza, tenendo però conto di applicarlo ad alberi che soffrono di alta varianza.

Se  $X$  è v.a. che conta il num. di errori commessi dai  $T$  modelli nel Bagging:

$$X \sim \text{Bin}(p, T) \quad (\text{dove } p \text{ è l'errore} < 0.5)$$

Ora vediamo la p. che la maggioranza dei modelli classifici correttamente:

$$P(X > \frac{T}{2}) = \sum_{x=\frac{T}{2}+1}^T P(X=x)$$

Così il maggior numero dei modelli diventa meno prob. che la maggioranza di questi si sbagli!

Per quanto riguarda AdaBoost millezma bene e traibi ed è molto efficace sui modelli lineari.

- 2) **Benefici di carattere statistico**: facciamo riferimento ad una approssimazione del classificatore ottimale Bayes: si prendano  $k$  classificazioni e ne si fa la media, ma è esattamente quello che facciamo con l'ensemble!
- 3) **Benefici nelle rappresentazioni dei concetti**: già trattati.
- 4) **Benefici computazionali**: si può scegliere il miglior algoritmo con confronti statistici soliti.
- 

## MISURAZIONE DEGLI ESPERIMENTI

- **Accuracy vs. Avg. Rec.**: noi sappiamo che l'accuracy è data da  $(TP/TOTAU)$  e che l'avg. recall è la media tra il CPr e il TMr; quale è migliore soluzioe?
  - ↳ Accuracy quando ci si aspetta una distribuzione affidabile per il dominio e non solo per il test set.
  - ↳ Avg. Recall quando non sappiamo nulla circa le freq. degli esempi nelle altre sottosezioni di dominio.

→ Per un classificatore è difficile mantenere una buona precisione sia una buona recall, solitamente i due si polarizzano verso uno delle due:

↳ Alta precisione: tende a predire tutti esempi come positivi al costo di sbagliarsi (-Recall...)

↳ Alta Recall: cliccando molti esempi positivi sbaglio e abbasso la precisione.

↳ Entrambi non considerano gli esempi correttamente predetti negativi

↳ F-measure: media aritmetica Recall / Precision, a scopo di dar merito all'algoritmo di classificare correttamente i negativi!

• Accuracy di un ranker:  $A = \frac{m}{m+n} \cdot \frac{2 \cdot AUC}{4} + \frac{1}{2}$

Dove AUC: area sotto la curva

↳ TPR e AUC sono utili se siano interessati al ranking!

• Misurare la varianza delle misurazioni: doce che abbiano capito che non esiste una sola misura della performance adeguata ad ogni problema quello che vogliono fare è misurare la varianza (e minimizzarla) delle diverse misurazioni.

Per farlo però il nostro dataset dovrà essere grande

a suff. per dividerlo in  $k$  parti e calcolarne la varianza. Admesso, in cas di dataset piccolo, applichiamo la Cross-Validation: si formano  $k$ -partizioni random,  $1$  viene usata come Test set,  $k-1$  come Training. I training non sono indipendenti dato che coincidono  $k-2$  parti di, mentre il Test set lo è: utilizziamo la C.V. per valutare modelli ed overfitting.

↳ Con meno di  $100$  esempi dobbiamo utilizzare la C.V. "leave one out" in cui il numero delle partizioni =  $M^o$  esempi. Qui ulta che misuro l'accuracy ottieno uno/zero mediano sulle  $k$  partizioni: molto buon se la dist. è normale.

↳ Con dataset sbilanciati: C.V. "stratificata". La selezione degli esempi per part. è corretta ma il numero contenuto è rappresentativo del dominio.

• Interpretare le Misure stoc (intervalli di confidenza): prendendo una stima  $\hat{\alpha}$  dalla Cv. e ne studiando la distribuzione rispetto ad un valore che è l'osservabilità della stima.

Dagli intervalli di confidenza sappiamo che surroundo di conoscere la stima dell'accuracy e std. dev.  $\sigma$ ,

l'attendibilità della stima è data dalla sua appartenenza ad un intervallo di confidenza.

→ Per calcolare l'intervallo di confidenza usciamo la Varianza, tralire l'applicazione della bionomiale con  $\hat{a}(\hat{a}-\hat{a}')$ , se il n° di esami è alto, approssimiamo la bionomiale con la curva di una normale (è simmetrico e quindi la preferiamo) → Poi vale che  $n \cdot \hat{a}(1-\hat{a}) \geq 5$ , altrimenti meglio la bionomiale.

↳

Esempio: se 80 esami sono correttamente classificati su 100 tot. l'accuracy stimata sarà  $\hat{a} = 0.8$ .

La varianza sarà  $\text{Var} = \hat{a}(1-\hat{a})/n = 0.8 \cdot 0.2 / 100 = 0.0016$  la cui dev. std.  $\sqrt{0.0016} = 0.04$

Per questo caso c'è verificata la condizione  $n\hat{a}(1-\hat{a}) \geq 5$  ( $= 16$ , **OK!**) Quindi possiamo calcolare l'intervallo di confidenza con una Gaussiana.

- $[0.76, 0.84]$  con distante  $\sigma$
- $[0.72, 0.88]$  " " 2σ: "assumendo una accuracy di 0.8, la P. che la misurazione ricade in  $[0.72, 0.88]$  è 0.95.  
Ha se fiduciano  $n : (50 \text{ tot}, 40 \text{ correttamente classificati, accuracy superiore di } 80\% \text{ ma } \Gamma = 0.06)$
- $[0.74, 0.86]$  con distante  $\sigma$

- $[0.68, 0.92]$  " " 2 $\sigma$  e confidence di 95%.
- Ipotesi nulle: un'affermazione sulla distribuzione di probabilità di una o più V.A.: vuol dire affermare che non ci sia differenza/relazione tra due fenomeni misurati/associazione tra due gruppi.

Nel nostro caso la sfruttiamo su a (i) valore vero del parametro: supponiamo che la nostra  $H_0$  sia che l'accorta reale sia 0.5 e che  $\sigma = \sqrt{0.5(1-0.5)/100} = 0.05$ . Data le stime di 0.8 calcoleremo il P-VALUE (P. di avere una misurazione  $\geq 0.8$  dada  $H_0$ ): P-VALUE verrà poi confrontata con una soglia (solitamente molto bassa, tipo  $\alpha = 0.05$  con 95% di confidence)  $\rightarrow H_0$  sarà rifiutata se p-value < soglia (qui  $p = 1.97 \cdot 10^{-9}$ ), qui si.

- Paired T-Test: quando non disponiamo di  $\sigma$  non posiamo facilmente costruire una gaussiana, ma una T-distribuzione.  
Per una coppia di oggetti viene calcolata la differenza di accoratozza in ciascuna partizione.  
 $H_0$  è che questa differenza sia 0.  
Si calcola lo p-value con una tavola e si rifiuta.

Ho sse p-value <  $\alpha$ . Con le T di Student decidiamo i gradi di libertà.

- Wilcoxon's signed-rank test: il T-test non va bene con dataset multipli, quindi utilizziamo il  $\alpha$ : l'idea è di fare ranking sulle differenze di perf. in valore assoluto, si calcola quindi la somma di rank possibili e negativi e si considera la somma minore. Il test assume che un'ampia differenza di performance è migliore rispetto ad una minore. Il test è NON-PARAMETRICO: ma si assume le distrib. e quindi è meno sensibile agli outliers

- Friedman Test: test per confrontare k algoritmi su m dataset escludendo che la considero decada nel tempo. Esso è:

→ Basato su RANKING

→ NON-PARAMETRICO

$$\cdot R_j = \sum_i \left( \frac{R_{ij}}{J} \right) : \text{ranking medio dell'algoritmo } j\text{-esimo}$$

$$\cdot \bar{R} = \frac{1}{mk} \sum_{i,j} R_{ij} = \frac{k+1}{2} : \text{rank medio}$$

$$\cdot m \sum_j (R_j - \bar{R})^2 : \text{la somma degli scarti quad ranki}$$

che è la distanza da un rank "centroide"

- $\frac{1}{m(k-1)} \sum_{ij} (R_{ij} - \bar{R})^2$ : le somme degli scarti quadratici ma che è la distanza di Pruema su tutti i rank

↓ La statistica di Friedman è il rapporto tra il secondo ed il terzo valore

- Post-hoc Test: Test finale, non parametrico la cui idea è calcolare le differenze critiche date dalla formula del Newman Test:

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6n}}, \text{ dove } q_\alpha \text{ dipende da } \alpha \text{ e } n.$$

- Bonferroni-Dunn Test: Variante del precedente in cui si fanno  $k-1$  confronti invece che  $k(k-1)/2$  (risparmiano tempo + spazio)

Appunti scritti male da: Federico Torrielli: