Validazione

es. supponiamo 2 classi con due numerosità di elementi: $W_1(n_1)$ e $W_2(n_2)$

- approccio è algoritmo di feature selection su tutti gli elementi delle due classi → creo il classificatore → ottengo il risultato migliore possibile per il classificatore
  e vado a vedere il risultato della classificazione degli elementi che ho usato per costruirlo
  → uso la percentuale di elementi classificati scorrettamente per identificare le prestazioni

		vero classificato	W_1	W_2
W_1	$n_1 = 105$	W_1	80	20
W_2	$n_2 = 90$	W_2	15	70

(se uso gli stessi esempi per costruire e validare il classificatore ottengo la prestazione massima)

- Però voglio anche provare il classificatore in una situazione + realistica, con altri dati (dati)
  ⇒ suddividi gli esempi in due campioni: uno parte per costruire il classificatore e una parte per verificare le prestazioni → prestazioni + basse ma + realistiche
  → come divido i due gruppi? va bene solo se ho tante numerosità

- leave-one-out: ne lascio uno fuori (1 es per costruire, 1 per verificare). Ripeto però questa operazione più volte per tutta la numerosità → riesco a ricreare la tabella sopra

FILTER FILTER: non tengono conto del predittore utilizzato
Ci sono algoritmi che prendono e insieme di caratteristiche, creano un subset e verificano la bontà dell'insieme.

WRAPPER
I wrapper invece vengono per valutare le prestazioni: utilizzano le performance di predizione di un classificatore per determinare l'importanza di un sottinsieme di variabili

RETI NEURALI

DATI → SISTEMA → INFORMAZIONE CONOSCENZA

Conoscenza di base, sui pazienti, legata ad altri dati
È altrettanto l'informazione che viene presa la decisione

Sviluppare un SISTEMA in modo tale che la creazione di informazione venga facilitata.
↳ dobbiamo inserire nel nostro sistema la conoscenza.
- in modo implicito: es. un algoritmo è una rappresentazione di conoscenza.
  ↳ la conoscenza è "nascosta" nell'algoritmo
- " " esplicito: la conoscenza è separata dal sistema (es. regole
  (un pezzo del sistema è dato dalla conoscenza)

Ci sono metodi diversi di rappresentazione della conoscenza

Fase di APPRENDIMENTO
→ necessità di avere degli ESEMPI (tanti) diversi
- apprendimento supervisionato
- " non supervisionato → attraverso il meccanismo del raggruppamento

es. per la classificazione gli esempi sono un vettore X_p con associato un valore C_p (classe di appartenenza)
↳ sottopongo esempi diversi (+ esempi più volte)
↳ numerosità degli esempi per ognuna classe deve essere equivalente

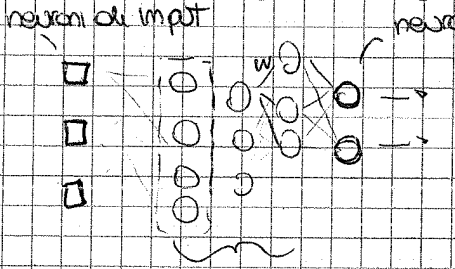
TRAINING SET = insieme di esempi
↳ sufficientemente numeroso e completo

È però difficile garantire la rappresentatività (esempi che servono)

Apprendimento che continua nel tempo ⇝ normalmente invece i sistemi hanno una fase limitata di apprendimento (devi sì)

Nelle max parte delle reti neurali non siamo in grado di fare una verifica dell'apprendimento
↳ i pesi contengono la conoscenza.

Una rete neurale si basa sul concetto di grafo: nodi = neuroni (elemento che elabora il dato e la conoscenza)
→ è un'elaborazione distribuita



STRUTTURA FEED-FORWARD

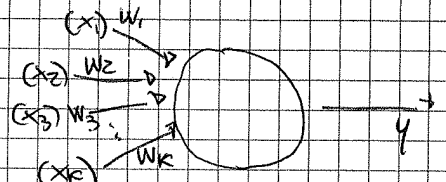
⇐ è la struttura della rete

↳ stati di neuroni: ognuno non è collegato allo stesso, ma con i neuroni dello stato successivo

Non è detto che questa struttura tutti i neuroni siano collegati
(normalmente varia da 0 a 1)
Ciascun arco ha associato un valore che è il peso (valori diversi per gli archi)
Durante la fase di apprendimento i pesi si modificano per ottenere il valore corretto.
↳ ho un input, viene elaborato dai vari neuroni ed ho un output
Se modifico ogni volta i pesi però non arrivo mai alla fine:
devo modificarli in modo intelligente

PERCEPTRON

Questa struttura è classica delle reti supervisionate:

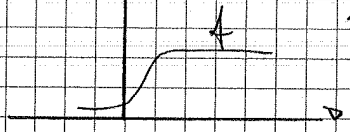


un neurone ha un certo n° di input ed una sola uscita (che poi viene propagata)

$y = f(w, x)$

$y = f\left(\sum_{i=1}^{k} w_i x_i\right)$ se il peso (w_i) è $\underset{\simeq 0}{1}$ → x_i molto influente / x_i non è influente

Il meccanismo dei pesi agisce facendo in modo che gli input diventino + o - influenti.
→ questo mi permette di differenziare le classi.



⇝ esistono funzioni predefinite

Per ogni neurone devo decidere la funzione ⇝ ma è troppo complicato
⇒ normalmente i neuroni dello stesso strato hanno tutti la stessa funzione.

→ devo decidere quanti sono gli strati e quanti neuroni metto per ogni strato

es.  funzione a gradino $\underset{1 = B}{0 = \overset{\text{classe}}{A}}$

- due neuroni

A	B
0	1
1	0

⇒ posso anche avere i mon classificati (00, 11)

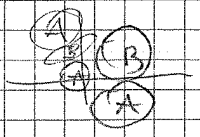
- un neurone

[0-0.5[	A
[0.5-1]	B

⇒ per classificare tra 2 classi è inutile avere più di due neuroni.

Più aggiungo strati meglio riesco a classificare situazioni complesse

Posso avere zone non ben definite di separazione tra le classi



→ non è detto che se metto tanti strati però, miglioro anche in situazioni semplici ⇒ peggioro

⇒ devo fare a tentativi: non posso fare tutti (problema di tempo)

epoca: do tutti gli input e modifico i pesi in base alle uscite (resetto tutto e poi modifico o uno per volta e modifico dipende dalle precedenti)

↳ nella nuova epoca ripeto cambiando l'ordine degli input

CLUSTER

06-11-2012

Conosciamo le classi, sappiamo quante sono, quali sono → classificazione

Le classi non sono conosciute → clustering → reti non supervisionate

si apprende tramite degli esempi già classificati

Costruire dei sottogruppi di elementi simili → è importante avere tanti esempi

↳ sottogruppi diversi tra loro, elementi simili in un sottogruppo (distanza, somiglianza interno)

CLUSTERING = trovare uno strumento di divisione

→ massimizziamo la distanza, minimizziamo la varianza

(per minimizzare la varianza basterebbe fare un cluster per elemento)

Il processo con cui vado a formare i sottogruppi è iterativo

⇒ si usa per acquisire la conoscenza che non avevo prima

es. uso di cluster → decomposizione del segnale emg: per ricostruire i trend di potenziale d'azione nelle varie unità motorie

K-MEANS

Si basa sul valor medio: prendo gli esempi, fa un'ipotesi sul n° di classi, associo a ciascun cluster un prototipo (lo faccio casualmente di solito) ↳ non è indicativo: serve per partire

es. 400 esempi ↳ 7 classi → 7 prototipi

suddivido gli esempi nelle classi, e costruisco il valor medio trovando un nuovo prototipo

Creo una nuova suddivisione in cluster (a partire dai nuovi prototipi)

↳ ho una suddivisione diversa (assegnazione al cluster: confronto l'elemento coi prototipi e lo assegno a quello + simile)

⇒ faccio molte iterazioni finché non ho + spostamenti: ogni elemento rimane nel cluster

es. 7 3 1 6 12 11 24 31 27 21 17

C1(7): 12 11 24 31 27 21 17 → C1(20): 8 24 31 27 21 17

C2(4): 7 3 1 6 → C2(4): 7 3 1 6 11

C1(7): 7 6 12 11 24 31 27 21 17

C2(3): 3 1

Un cluster con una varianza troppo grande → 2 cluster ; e viceversa

⇒ potrei aumentare/diminuire il n° cluster e ripetere il K-means.

Questo lavoro lo fa l'algoritmo 'isodata' automaticamente

↳ richiede dei parametri

- n° minimo di esempi per cluster
- n° approssimativo di cluster
- max varianza in un cluster
- max n° di cluster che possono essere uniti ad ogni iterazione

20-11-2012

DENDROGRAMMA

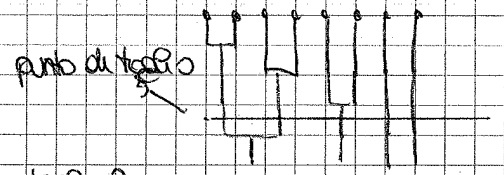
Usare un tipo di aggregazione gerarchico.

Ho n elementi. → ad ogni iterazione unisco i cluster con i due elementi più vicini
↳ ho n-1 cluster

e così via fino a costruire un unico cluster che contiene tutti gli elementi

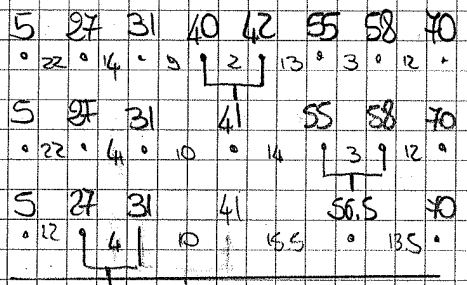
↳ devo trovare un "punto di taglio"

es. quando la distanza che calcolo è maggiore (molto) delle precedenti mi fermo



Automaticamente non è così banale capire quando tagliare

esempio:



→ dovrei fare le linee verticali + lunghe a seconda dell'aumento di distanza

↳ un buon punto per tagliare è dove la distanza aumenta molto

(algoritmo: mettere uno soglia)

⇒ i dendrogrammi sono presenti in letteratura, ma non sono molto usati

RETI NEURALI NON SUPERVISIONATE

Apprendimento non supervisionato = non sappiamo a quale classe il training set appartiene. (non è classificato)

→ non posso usare apprendimento supervisionato su training set non classificato

ma posso fare il contrario (non supervisionato su classificato)

↳ si fa per ricerca: vedere se applicando il supervisionato o il non supervisionato ottengo cluster diversi, cose diverse, ...

COMPETITIVE LEARNING:

normalmente c'è un solo layer nascosto. Uso i pesi come prototipo.

Prendo i pesi collegati al neurone e li confronto con l'input: il neurone che ho i pesi più vicini

es. ogni neurone in questo esempio ha tre pesi $\vec{w}_1 = \{w_{11}, w_{12}, w_{13}\}$

$\vec{x}_1 = \{x_{11}, x_{12}, x_{13}\}$ tre valori che caratterizzano l'elemento.

Confronto $\vec{x}_1$ e $\vec{w}_1$ ↳ è l'input

faccio le distanze d_{x_1,w_1} d_{x_1,w_2}, d_{x_1,w_3}, d_{x_1,w_4} d_{x_1,w_5}.

$$d_{x_i,w_i} = \sum_{j=1}^{3} |x_{ij} - w_{ij}|$$

tra queste 5 distanze, ne avrò una + piccola che individua il neurone che associo all'elemento

(è già simile) → cerco di renderlo ancora + simile (ma non uguale)

↳ gli sommo di peso Δw_{ij} $\left\{ \begin{array}{l} \alpha (x_i - w_{ij}) \\ 0 \end{array} \right.$

↳ potrebbe anche essere una funzione + complicata

⇒ cioè avvicino il neurone che è già + vicino e non vado gli altri pesi (degli altri neuroni)

→ con molte iterazioni ciascun neurone si specializza con certi input.

Al termine dell'apprendimento ciascun neurone rappresenta degli elementi

→ i pesi rappresentano le nostre caratteristiche (cosa che non era con la rete supervisionata)

↳ è + potente la non supervisionata per capire quanto ragionevole è stata la osservazione

↳ posso valutare le classi valutando i pesi

Il problema è lo stesso del k-mean: definire un n° di neuroni della rete

↳ se pochi: classi troppo vaste

se tanti: troppe classi, risultati dispersi.

Non c'è nessun rapporto tra i vari neuroni, a parte quello legato al confronto della distanza minima per trovare il vincitore

(i neuroni vicini non hanno legami, possono essere vicini come lontani)

⇒ <u>mappe di Kohonen</u>: i neuroni vicini sono legati → quando modifico i pesi, non modifico solo il vincitore, ma anche gli altri (avvicino i vicini e allontano i lontani)

27-11-2017

Vediamo la bontà del clustering:

- <u>omogeneità del singolo cluster</u>
- <u>distanza tra cluster</u>

convergenza → i pesi non variano più da un'iterazione all'altra
↳ piccolissime modifiche

Al termine dell'addestramento i pesi associati a ciascun neurone sono il prototipo della classe

se ho più neuroni di classi → più neuroni che fanno riferimento alla stessa classe, con pesi abbastanza simili tra loro

→ se ne metto troppi, diventa complicato

⇒ fare un post-processing per individuare veramente le classi (es. quando la distanza tra classi cluster è troppo bassa)
↳ può significare che ho usato troppi cluster ↳ posso riaddestrare la rete con meno neuroni
posso aggregare i cluster (con k-mean rete neurale, ...)

Se invece ho meno neuroni di quanti ne servono, ho alta variabilità tra i cluster (classi) che crea

<u>Variante di KOHONEN</u> → a differenza del COMPETITIVE LEARNING di base, associo + cluster

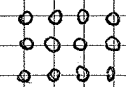
<u>a più neuroni</u>

↳ i neuroni che sono fisicamente vicini vengono coinvolti: si modificano i pesi (in modo minore) dei neuroni vicini al neurone vincitore → devo definire una regola per capire succede se un neurone vin

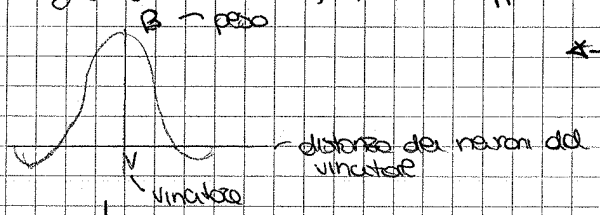
↳ modifico i pesi solo del neurone vincitore ciascun neurone fa per sé, è scollegato da quelli vicini

⇒ crea delle zone che rappresentano degli input simili tra di loro

⇒ in questo caso è necessario che ci sia sovrabbondanza di neuroni!



- COMPETITIVE LEARNING BASE → n° neuroni ≈ n° CLUSTER!
- MAPPE DI KOHONEN → n° neuroni >> n° cluster!
  - può essere quadrata, rettangolare, esagonale, ... ⇒ dimensione della mappa
  - bisogna sovrabbondare, ma non troppo ⇒ dimensione dell'intorno
    ⇒ topologia dell'intorno
    ← PESI CHE DO ALL'INTORNO DEL VINCITORE



  - allontano i neuroni + distanti : caso una barriera (non è molto usata)

C'è anche il parametro di inizializzazione dei pesi

le mappe di Kohonen (e la max parte delle reti non supervisionate) sono single layer

(per sapere se sono addestrate si guarda quanto si modificano i pesi)

batch = sottoinsieme di neuroni con pesi simili tra loro, che classificano lo stesso cluster

LOGICA FUZZY

07-01-2013

Tiene conto delle <u>incertezze</u> → i nostri dati e i nostri concetti sono affetti da incertezza

es. analizzare il concetto di età = giovani - adulti - anziani

i confini di queste tre classi non sono definiti

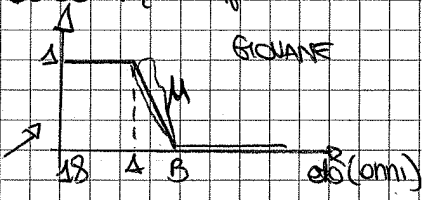
↳ permette di rappresentare l'incertezza

si rappresentano i concetti come delle classi → a ciascuna classe è associata una funzione d'appartenenza (valori compresi tra 0 e 1 ⌜ appartenenza certa ⌞ non appartenenza certa)

→ Fun. di appartenenza

$\mu_{GIOVANE}(19) = 1$

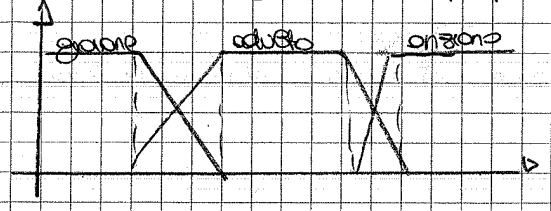
→ è un modello della mia informazione (l'appartenenza alle classi)

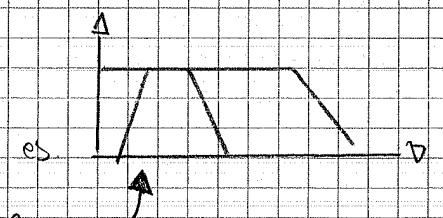


esempio di funzione

Si definisce un grado di appartenenza alle mie classi per ciascuno paradigma

Si rappresentano le membership function delle varie classi in un unico grafico





È possibile avere l'appartenenza certa a più di una classe

es. per l'età ho senso che quando finisce l'appartenenza certa ad una classe inizia un altro classe

Le funzioni di appartenenza normalmente sono trapezoidali / triangolari (ma possono essere come voglio)

Come faccio a definire A, B, ... → le funzioni?

- chiedo ad un esperto
- faccio un'indagine su un certo n° di persone