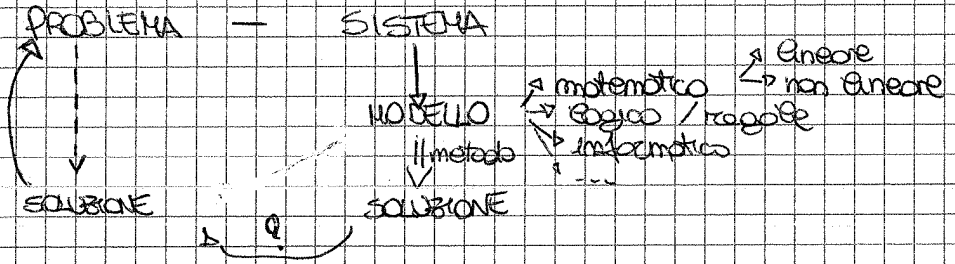
ESERCIZIO:

Struttura sanitaria accreditata (privata). Si vuole aggiungere una prestazione ambulatoriale → voglio ottimizzare il numero di medici, massimizzare i profitti

- avere liste d'attesa brevi
- più possibilità di scegliere/mantenere il medico

⇒ devo far scegliere ai pazienti questa struttura rispetto alla sanità pubblica

08-10-2017



Applicando un metodo al modello otteniamo una soluzione; è anche la soluzione del problema?

Si cercano di costruire modelli + flessibili → rappresentare bene il sistema senza allontanarsi troppo dalla realtà (è difficile farlo con modelli matematici)

OTTIMIZZARE → cosa voglio ottimizzare, rispetto al modello → qual è la funzione obiettivo

Dato lo stesso modello si possono utilizzare metodi diversi

→ se non funziona si modifica qualcosa e si ripete il procedimento.

OTTIMIZZAZIONE

+ funzione obiettivo

+ vincoli.

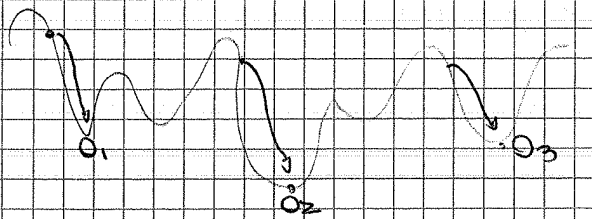
se la funz. obiettivo è lineare, è + facile → non c'è un sistema complesso.

↳ ha un solo ottimo. (quella non lineare ne ha + di uno)

↳ è fondamentale da quale punto parto

il problema è che quando arrivo ad un ottimo non so se è il migliore

↳ potrei da + punti.



Se soddisfo i vincoli la soluzione è accettabile, se no no.

↳ posso inserirli nella funzione obiettivo.

↳ possono essere "invalicabili", oppure in altri casi posso "rilassarli"

es. ho diverse possibilità di utilizzare un budget per acquistare apparecchiature

$BUDGET = C = C_1 + C_2 + C_3 + \dots + C_{nc}$ → n° apparecchiature, tipo apparecchiature

ci sono situazioni in cui il budget è assolutamente finito (non riesco a spenderlo giusto $C \geq C_1 + C_2 + C_{nc} \dots$

una funzione obiettivo potrebbe essere: $\max(m) + T$
(m: n° apparecchiature, T: tempo)

$$\max(m) + T + \beta\left(C - \sum_{i=1}^{n} C_i\right)$$

↳ per pesare il fatto che se supero C voglio o prendere + o −

...

Metodi euristici basati su ricerca locale

↳ si sono sviluppati fino a rendere facile rappresentare problemi complessi (non sempre facile con equazioni matematiche)

↳ non garantiscono di ottenere ottimo globale, ma ad esempio sono utili perché sono + "veloci" e − "costosi"

⇒ parto da una soluzione, da questa individuo un insieme di altre soluzioni (VICINATO) ⇒ da questo vicinato ci scelgo una soluzione → si ripete
↳ attraverso un operazione elementare iterativamente

→ metodi che creano algoritmi ispirati ad un modello tratto dalla realtà: metaeuristiche

1. Inizializzazione: trovare una soluzione
2. Generazione del vicinato
3. Test di accettazione → per trovare una nuova soluzione
4. Test di terminazione → continuare o fermare il ciclo

↳ nasce dal raffreddamento dei metalli

→ Simulated annealing: [al punto 3, se metodo classico va sempre nella direzione del miglioramento] Questo metodo invece accetta anche una soluzione peggiore ⇒ mi muovo molto di +, − vincolato dalla potenza.

↳ ho dei limiti (funzione di probabilità) che lo blocca
↳ riduco nel tempo la probabilità di accettare un peggioramento
// riduzione della temperatura → è un altro parametro

es. posso scegliere una soluzione random tra quelle del vicinato, non la migliore

(confronto con $e^{-\delta/t}$ → minore, con t = temperatura)
in questo modo il limite diventa + stringente, xke diminuisce!

in questo algoritmo ci sono un certo n° di decisioni basate sulla probabilità

EVOLUTIONARY COMPUTING ⇒ GLI ALGORITMI GENETICI

Siamo partiti dalla soluzione di partenza → la facciamo mutare da + soluz

andiamo avanti iterando finché non siamo più in grado di trovare miglioramenti o raggiungiamo il n° max di iterazioni che ci siamo prefissati

⇒ COMPUTATIONAL INTELLIGENCE

popolazione iniziale di soluzioni → selezione di un sottoinsieme → operatori → nuove soluzioni

FITNESS = funzione obiettivo → prestazioni

le soluzioni devono essere codificate → ottengo una stringa di 0 e 1

↳ passo rappresentato come presente (1) o assente (0)

individuare quanti bit servono per rappresentare una variabile

Se una variabile non assume valori interi → cambio algoritmo
↘ decido quante cifre decimali utilizzare

→ regola di codifica e regola di decodifica
VALORE → STRINGA STRINGA → VALORE

Andare a verificare che la soluzione sia ammissibile ↝ mi capita + facilmente che non sono ammissibili (es. uso + bit di quanti servono)

$|m|$ = n° soluzioni popolazione → vado per "esplorare", ad ogni iterazione

n° iterazioni

m e n iterazioni sono collegati: se aumento uno diminuisce l'altro, e viceversa (non è una legge precisa però)

↳ generazione casuale (verifica dell'ammissibilità)

REGOLA DI SELEZIONE
H = n° di soluzioni selezionate

m + H nuove soluzioni → V: nuova popolazione (m)

OPERATORI GENETICI:

- CROSSOVER
all'interno delle soluzioni se ne prendono due e se ne scambiano pezzi (uno con l'altro e viceversa)
↳ normalmente si prendono le soluzioni a coppie
- MUTAZIONI
cambia casualmente uno o più bit ↝ probabilità di mutazione

Per noi è importante che l'algoritmo sia stabile ↝ le soluzioni sono simili

TABU SEARCH — sono consentite mosse peggiorative. Per non cadere in loop, le ultime 9-10-20 mosse fatte sono tabù, non posso essere ripetute ⇒ uso della MEMORIA
↳ possono esserci eccezioni

Le poche regole non richiedono l'uso delle probabilità

è un metodo deterministico

→ idea di base di qualcosa che non posso fare → può essere infranto solo se ho ottenuto vantaggio

È basato sulla memoria

es. problema della location di un'apparecchiatura costosa

es. tecnico che deve fare interventi riparativi in posti diversi → trovare il percorso ottimale

$T_1, T_2, \ldots, T_8$

f = tempo di spostamento

$T_2, T_8, T_7, T_6, T_1, T_4, T_3, T_5$ soluzione attuale

per costruire il vicinato di questa soluzione, posso ad esempio scambiare due coppie
es. T_8 con tutte le altre

$T_8, T_2, T_7, T_6, \ldots$

$T_2, T_7, T_8, \ldots$

...

Ho tutte queste (il vicinato) viene selezionata come soluzione successiva la migliore (anche se peggiore della soluzione prima, ...)

se itero nuovamente, è possibile che ricompaiano le soluzioni di prima → queste vengono escluse (concetto di memoria) per evitare di finire in un loop

Posso però effettuare uno scambio "tabù" (da evitare) se mi porta in una soluzione migliore di tutte le precedenti

[es. cerco di costruire un materiale resistente con 4 diversi strati]

Lo scambio tabù ha un "tempo", una durata (n° iterazioni)

tornando all'esempio degli interventi: vogliamo minimizzare ~~[illegible]~~ i chilometri

→ ci serve una matrice con le distanze tra i vari posti

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8
T_1		12	3	24	1	7	-	-
T_2			-	-	-	-	-	-
T_3				-	-	-	-	-
					-	-	-	-
						-	-	-
							-	-
								-

parto da una soluzione qualsiasi

ant colony organization/optimization

ACO = ottimizzazione che si basa sull'osservazione del comportamento delle formiche

→ in qualche modo comunicano tra loro: rilasciano feromoni durante il percorso
↳ si orientano nei vari percorsi da + feromoni

ma + il percorso è breve, + volte lo stesso formica passa → + feromoni

12-10-2017

FEATURE SELECTION

Non è detto che avere tanti dati aumenti la nostra capacità di fornire info

↳ alcuni dati possono essere una sorta di rumore → peggiorano la classificazione

Da quando raccogliere dati in modo elettronico è diventato facile → molto usato (anche se ad es. in ospedale molti dati clinici sono ancora cartacei)

La validità del dato è importantissima

→ gli outlier sono i valori che raccolgo di un dato e si discostano molto dalla media: potrebbero essere un errore, un'anomalia

Dati che provengono da test, o dal clinico (personale sanitario)

Trial = studio controllato (in ambito clinico)
↳ devo dimostrare l'efficacia dopo gli studi teorici } es. nuovi farmaci

- feature construction
- feature selection

avere uno strumento per intercettare gli errori di classificazione

es. preso un paziente, decidere se trattare o no l'ipertensione / l'ipercolesterolemia, e anche come trattarlo

APPROCCIO

- forza bruta: prendo tutte le possibili combinazioni
  ↳ non è fattibile, ho dei tempi troppo lunghi
- euristico: si generano uno o + sottinsiemi, si testano → si ripete iterativamente finché non si ottiene un buon risultato
  ↳ ci sono diversi metodi
  → è difficile da trovare

si richiede una funzione che ci permetta di valutare i vari sottinsiemi.

i criteri di stop sono vari: fermarsi con buone caratteristiche, dopo n iterazioni, ...

non è detto che ci sia un solo sottinsieme di caratteristiche ma ch buoni risultati

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