

Programmazione dinamica (VI parte)

Progettazione di Algoritmi a.a. 2023-24

Matricole congrue a 1

Docente: Annalisa De Bonis

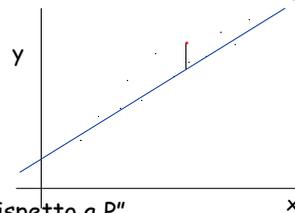
91

91

Segmented Least Squares

- **Minimi quadrati.**
 - Problema fondamentale in statistica e calcolo numerico.
 - Dato un insieme P di n punti del piano $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$.
 - Trovare una linea L di equazione $y = ax + b$ che minimizza la somma degli errori quadratici.

$$\text{Error}(L,P) = \sum_{i=1}^n (y_i - ax_i - b)^2$$



- Chiameremo questa quantità "Errore di L rispetto a P "
- Chiameremo "Errore minimo per P ", il minimo valore di $\text{Error}(L,P)$ su tutte le possibili linee L
- **Soluzione.** Analisi \Rightarrow il **minimo errore** per un dato insieme P di punti si ottiene usando la linea di equazione $y = ax + b$ con a e b dati da

$$a = \frac{n \sum_i x_i y_i - (\sum_i x_i)(\sum_i y_i)}{n \sum_i x_i^2 - (\sum_i x_i)^2}, \quad b = \frac{\sum_i y_i - a \sum_i x_i}{n}$$

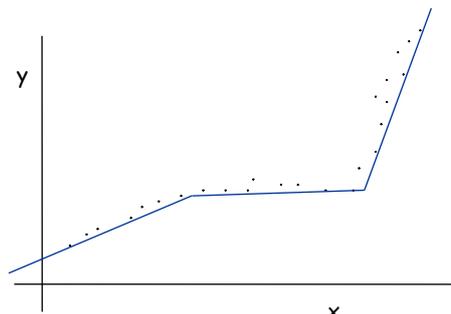
92

92

Segmented Least Squares

- L'errore minimo per alcuni insiemi input di punti puo` essere molto alto a causa del fatto che i punti potrebbero essere disposti in modo da non poter essere ben approssimati usando un'unica linea.

Esempio: i punti in figura non possono essere ben approssimati usando un'unica linea. Se pero` usiamo tre linee riusciamo a ridurre di molto l'errore.



Progettazione di Algoritmi A.A. 2023-24
A. De Bonis

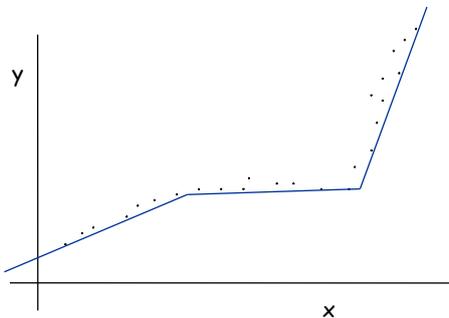
93

93

Segmented Least Squares

Segmented least squares.

- In generale per ridurre l'errore avremo bisogno di una sequenza di linee intorno alle quali si distribuiscono sottoinsiemi di punti di P .
- Ovviamente se ci fosse concesso di usare un numero arbitrariamente grande di segmenti potremmo ridurre a zero l'errore:
 - Potremmo usare una linea per ogni coppia di punti consecutivi.
- **Domanda.** Qual e` la misura da ottimizzare se vogliamo trovare un giusto compromesso tra accuratezza della soluzione e parsimonia nel numero di linee usate?



Il problema e` un caso particolare del problema del change detection che trova applicazione in data mining e nella statistica: data una sequenza di punti, vogliamo identificare alcuni punti della sequenza in cui avvengono delle variazioni significative (in questo caso quando si passa da un'approssimazione lineare ad un'altra)

Progettazione di Algoritmi A.A. 2023-24
A. De Bonis

94

94

Segmented Least Squares

Formulazione del problema Segmented Least Squares.

- Dato un insieme P di n punti nel piano $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ con $x_1 < x_2 < \dots < x_n$, **vogliamo partizionare P in un certo numero m di sottoinsiemi P_1, P_2, \dots, P_m in modo tale che**
- Ciascun P_i e' costituito da punti contigui lungo l'asse delle ascisse
 - P_i viene chiamato segmento
- La sequenza di linee L_1, L_2, \dots, L_m **ottime** rispettivamente per P_1, P_2, \dots, P_m minimizzi la **somma delle 2 seguenti quantita'**:
 - 1) La somma E degli m errori minimi per P_1, P_2, \dots, P_m (l'errore minimo per il segmento P_i e' ottenuto dalla linea L_i)

$$E = \text{Error}(L_1, L_2, \dots, L_m; P_1, P_2, \dots, P_m) = \sum_{j=1}^m \sum_{(x_i, y_i) \in P_j} (y_i - a_j x_i - b_j)^2$$

- 2) Il numero m di linee (pesato per una certa costante input $C > 0$)

La quantita' da minimizzare e' quindi $E + Cm$ (penalita').

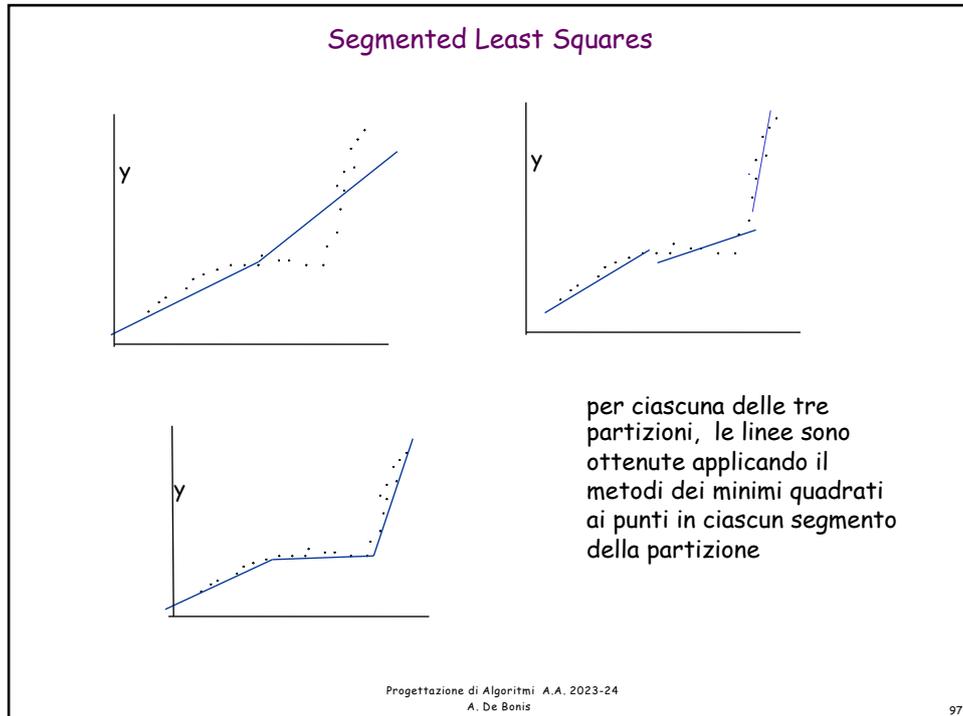
Segmented Least Squares

Formulazione del problema Segmented Least Squares.

- **Input:** insieme P di n punti nel piano $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ con $x_1 < x_2 < \dots < x_n$, e una costante $C > 0$
- **Obiettivo:** Trovare una partizione P_1, P_2, \dots, P_m di P tale che
 1. ciascun P_i e' costituito da punti contigui lungo l'asse delle ascisse
 2. la penalita' $E + Cm$ sia la piu' piccola possibile
- dove E e' la somma degli m errori minimi per P_1, P_2, \dots, P_m

$$E = \sum_{j=1}^m \sum_{(x_i, y_i) \in P_j} (y_i - a_j x_i - b_j)^2$$

Per ogni j nella sommatoria a_j e b_j sono ottenuti applicando il metodo dei minimi quadrati ai punti di P_j



97

Segmented Least Squares

- Il numero di partizioni in segmenti dei punti in P è esponenziale \rightarrow ricerca esaustiva e inefficiente
- La programmazione dinamica ci permette di progettare un algoritmo efficiente per trovare una partizione di penalità minima
- A differenza del problema dell'Interval Scheduling Pesato in cui utilizzavamo una ricorrenza basata su due possibili scelte, per questo problema utilizzeremo una ricorrenza basata su un numero polinomiale di scelte (lo stesso avveniva per il problema della moltiplicazione di una catena di matrici).

Progettazione di Algoritmi A.A. 2023-24
A. De Bonis

98

Approccio basato sulla programmazione dinamica

Notazione

- Sia p_j un qualsiasi punto input,
 $OPT(j)$ = costo minimo della penalità per i punti p_1, p_2, \dots, p_j .
- Siano p_i e p_j due dei punti input, con $i \leq j$,
 $e(i, j)$ = minimo errore per l'insieme di punti $\{p_i, p_{i+1}, \dots, p_j\}$.

$$e(i, j) = \sum_{k=i}^j (y_k - a_{ij}x_k - b_{ij})^2$$

dove a_{ij} e b_{ij} sono ottenuti applicando il metodo dei minimi quadrati ai punti p_i, p_2, \dots, p_j

Per computare $OPT(j)$, osserviamo che

- se l'ultimo segmento nella partizione di $\{p_1, p_2, \dots, p_j\}$ è costituito dai punti p_i, p_{i+1}, \dots, p_j per un certo i , allora
- **penalità** = $OPT(i-1) + e(i, j) + C$.
- Il valore della **penalità** cambia in base alla scelta di i
- Il valore $OPT(j)$ è ottenuto in corrispondenza dell'indice i che minimizza $OPT(i-1) + e(i, j) + C$.

Progettazione di Algoritmi A.A. 2023-24
A. De Bonis

99

99

Approccio basato sulla programmazione dinamica

- Da quanto detto nella slide precedente, si ottiene la seguente formula per $OPT(j)$:

$$OPT(j) = \begin{cases} 0 & \text{se } j=0 \\ \min_{1 \leq i \leq j} \{ e(i, j) + C + OPT(i-1) \} & \text{altrimenti} \end{cases}$$

Progettazione di Algoritmi A.A. 2023-24
A. De Bonis

100

100

Segmented Least Squares: Algorithm

```

INPUT:  $n, p_1, \dots, p_n, c$ 

Segmented-Least-Squares() {
  M[0] = 0
  for j = 1 to n
    for i = 1 to j
      compute the least square error  $e(i, j)$  for
      the segment  $p_i, \dots, p_j$ 

  for j = 1 to n
    M[j] =  $\min_{1 \leq i \leq j} (e(i, j) + C + M[i-1])$ 

  return M[n]
}

```

Tempo di esecuzione. $O(n^3)$.

- Collo di bottiglia = dobbiamo computare il valore $e(i, j)$ per $O(n^2)$ coppie i, j . Usando la formula per computare la minima somma degli errori quadratici, ciascun $e(i, j)$ è computato in tempo $O(n)$

101

Algoritmo che produce la partizione

```

Find-Segments(j)
  If  $j = 0$  then
    Output nothing
  Else
    Find an  $i$  that minimizes  $e_{i,j} + C + M[i-1]$ 
    Output the segment  $\{p_i, \dots, p_j\}$  and the result of
      Find-Segments( $i-1$ )
  Endif

```

102

Esercizio

- Per il saggio di fine anno gli alunni di una scuola saranno disposti in fila secondo un ordine prestabilito e **non modificabile**. La fila sarà suddivisa in gruppi contigui e ciascun gruppo dovrà intonare una parte dell'inno della scuola. Il maestro di canto ha inventato un dispositivo che permette di valutare come si fondono le voci di un gruppo tra di loro. Più **basso** è il punteggio assegnato dal dispositivo ad un gruppo, migliore è l'armonia delle voci. Il maestro vuole ripartire la fila di alunni in gruppi contigui in modo da ottenere entrambi i seguenti obiettivi
 - la somma dei punteggi dei gruppi sia la più piccola possibile
 - il numero totale di gruppi non sia troppo grande per evitare che l'inno debba essere suddiviso in parti troppo piccole. Ogni gruppo fa aumentare il costo della soluzione di un valore costante $g > 0$.

NB: Il dispositivo computa per ogni coppia di posizioni i e j con $i < j$, il valore $f(i,j)$ dove $f(i,j)$ = punteggio assegnato al gruppo che parte dall'alunno in posizione i e termina con l'alunno in posizione j .

continua nella slide
successiva

103

Esercizio

- Si formuli il suddetto problema sotto forma di problema computazionale specificando in cosa consistono un'istanza del problema (input) e una soluzione del problema (output). Occorre definire una funzione costo di cui occorre ottimizzare il valore.
- Si fornisca una formula ricorsiva per il calcolo del valore della soluzione ottima del problema basata sul principio della programmazione dinamica. **Si spieghi in modo chiaro come si ottiene la suddetta formula.**
- Si scriva lo pseudocodice dell'algoritmo che trova il valore della soluzione ottima per il problema.

104

Sottosequenza crescente piu` lunga elementi non necessariamente contigui

- Vogliamo individuare la sottosequenza (strettamente) crescente piu` lunga in una sequenza di numeri. La sottosequenza non deve essere necessariamente formata da elementi contigui nella sequenza input. La sequenza input è memorizzata in un array A .
- Indichiamo con $A[0, \dots, i]$ il segmento di A degli elementi con indice da 0 a i .
- Esempio: $A = \langle 3 \ 12 \ 9 \ 4 \ 12 \ 5 \ 8 \ 11 \ 6 \ 13 \ 10 \rangle$
- La sottosequenza crescente piu` lunga è $\langle 3 \ 4 \ 5 \ 8 \ 11 \ 13 \rangle$

Progettazione di Algoritmi A.A. 2023-24
A. De Bonis

105

Sottosequenza crescente piu` lunga elementi non necessariamente contigui

- $OPT(i)$ = lunghezza della sottosequenza crescente più lunga che termina in i (l'ultimo elemento della sottosequenza è $A[i]$)
- Una volta che abbiamo calcolato $OPT(i)$ per ogni i , andiamo a calcolare il massimo di tutti i valori $OPT(i)$ per ottenere la lunghezza della sottosequenza crescente piu` lunga.

Progettazione di Algoritmi A.A. 2023-24
A. De Bonis

106

Sottosequenza crescente piu` lunga elementi non necessariamente contigui

- Osserviamo che nella sottosequenza crescente piu` lunga che termina in i , l'elemento $A[i]$ e` preceduto dalla piu` lunga sottosequenza crescente che termina in un certo j tale che $j < i$ e $A[j] < A[i]$.
 - Quale di questi j bisogna prendere?
 - Quello per cui la lunghezza della sottosequenza è massima, cioe` l'indice j per cui si ottiene il massimo valore $OPT(j)$ in questo insieme: $\{OPT(j): 0 \leq j \leq i-1 \text{ e } A[j] < A[i]\}$

Si ha quindi

$$OPT(i) = \max\{OPT(j)+1: 0 \leq j \leq i-1 \text{ e } A[j] < A[i]\}$$

Progettazione di Algoritmi A.A. 2023-24
A. De Bonis

107

Sottosequenza crescente piu` lunga elementi non necessariamente contigui

- Esempio: $A = \langle 3 \ 12 \ 9 \ 4 \ 12 \ 5 \ 8 \ 11 \ 6 \ 13 \ 10 \rangle$
- per calcolare $OPT(10)$ consideriamo $j=0, j=2, j=3, j=5, j=6, j=8$
- per calcolare $OPT(9)$ consideriamo $j=0, j=2, j=3, j=4, j=5, j=6, j=7, j=8$
- per calcolare $OPT(8)$ consideriamo $j=0, j=3, j=5$
- per calcolare $OPT(7)$ consideriamo $j=0, j=2, j=3, j=5, j=6$
- per calcolare $OPT(6)$ consideriamo $j=0, j=3, j=5$,
- per calcolare $OPT(5)$ consideriamo $j=0, j=3$
- per calcolare $OPT(4)$ consideriamo $j=0, j=2, j=3$
- per calcolare $OPT(3)$ consideriamo $j=0$
- per calcolare $OPT(2)$ consideriamo $j=0$
- per calcolare $OPT(1)$ consideriamo $j=0$
- $OPT(0)=1 \rightarrow OPT(1)=OPT(2)=OPT(3)=2$
- $OPT(0)=1$ e $OPT(2)=OPT(3)=2 \rightarrow OPT(4)=3$
- $OPT(0)=1$ e $OPT(3)=2 \rightarrow OPT(5)=3$
- $OPT(0)=1, OPT(3)=2, OPT(5)=3 \rightarrow OPT(6)=4$
- $OPT(0)=1, OPT(2)=OPT(3)=2, OPT(5)=3, OPT(6)=4 \rightarrow OPT(7)=5$
- $OPT(0)=1, OPT(3)=2, OPT(5)=3 \rightarrow OPT(8)=4$
- $OPT(0)=1, OPT(2)=OPT(3)=2, OPT(4)=3, OPT(5)=3, OPT(6)=4, OPT(7)=5 \rightarrow$
 $OPT(9)=6$
- $OPT(0)=1$ e $OPT(3)=2, OPT(5)=3, OPT(6)=OPT(8)=4 \rightarrow OPT(10)=5$

Progettazione di Algoritmi A.A. 2023-24
A. De Bonis

108

Sottosequenza crescente piu` lunga elementi non necessariamente contigui

Questo algoritmo trova la lunghezza della sottosequenza crescente piu` lunga che termina in i

```
LIS(A,i):
  if i<0 return 0
  if i=0
    P[0]=-1, M[0]=1, return 1
  if M[i]!=empty return M[i]
  M[i]=1 //restera` 1 se non ci sono j<i con A[j]<A[i]
  P[i]=-1 //serve nel caso alla fine M[i]=1
  for( j=0; j<i; j++)
    m=LIS(A,j)
    if (A[j]<A[i])
      if(M[j]<m+1)
        M[i]=m+1
        P[i]=j
  return M[i]
```

tempo $O(n^2)$

M globale
P globale: mi serve per la stampa
P[i]= indice dell'elemento che precede i
nella sottosequenza crescente piu` lunga
che termina in i

Per computare il valore della sottosequenza crescente
piu` lunga dell'array occorre prima invocare LIS(A,n-1) e
poi trovare il massimo dell'array M.