Logica e Metodi Probabilistici per L'Informatica - Homework

Academic year 2021/2020

Instructor: Prof. Stefano Leonardi Teaching Assistant: Federico Fusco

Deadline: 10/05/2022

Regole generali: Potete consegnare gli esercizi in gruppi da due studenti. Tuttavia, assicuratevi di aver compreso ogni dettaglio dell'elaborato che consegnate! Non copiate da altri studenti, da internet o da altre fonti, né condividete la vostra soluzione con altri gruppi; in caso contrario l'elaborato verrà invalidato e dovrete sostenere l'esame scritto. Non ci saranno eccezioni. Se avete dubbi sull'interpretazione dell'homework potete scrivere a fuscof@diag.uniroma1.it.

Consegna in ritardo: Il punteggio degli elaborati consegnati dopo la scadenza verrà penalizzato in base ai giorni di ritardo: un giorno di ritardo corrisponde ad una penalizzazione del 10% rispetto al punteggio effettivo, due giorni al 20%, tre giorni al 30%. Elaborati inviati con più di tre giorni di ritardo non verranno corretti.

Formato consegna: Gli elaborati dovranno essere obbligatoriamente scritti in LaTex (almeno 11 pt) e riportare il nome dei due componenti del gruppo su ogni foglio. Ogni risposta dovrà essere adeguatamente motivata e riportare il numero dell'esercizio a cui si riferisce. Consegnare gli elaborati via mail a fuscof@diag.uniroma1.it usando l'indirizzo istituzionale @studenti.uniroma1.it. Le mail dovranno riportare "LMPI 2021/2022 Homework" in oggetto e contenere un solo file pdf nominato "cognome1_cognome2.pdf". Il pdf non dovrà contenere più di 4 pagine, le ulteriori pagine saranno ignorate.

Exercise 1. Supponiamo di osservare una sequenza di oggetti che arrivano uno dopo l'altro, di cui ne possiamo mantenere solo k contemporaneamente in memoria. Ogni nuovo elemento può essere scartato o salvato in memoria (eventualmente rimuovendo un elemento salvato precedentemente per fargli spazio); elementi scartati o rimossi sono persi per sempre e non possono essere più considerati. Consideriamo l'algoritmo di campionamento descritto di seguito.

Algorithm 1 Campionamento online di k elementi

```
1: S ← ∅, t ← 0
2: for ogni nuovo elemento e che arriva do
3:     t ← t + 1
4:     if |S| < k then
5:         Inserire e in S
6:     else
7:         Lanciare una moneta con probabilità di testa k/t
8:         if testa then
9:             Rimuovere un elemento u.a.c. da S ed inserire e al suo posto
```

Per ogni istante di tempo t, sia N_t l'insieme dei primi t elementi della sequenza e sia S_t la versione di S alla fine dell'iterazione del ciclo for relativo al t^o elemento della sequenza.

(a) Dimostrare che, per ogni t, l'insieme S_t è estratto uniformemente a caso tra tutti i sottinsiemi di N_t di k elementi. Formalmente, dimostrare che

$$\mathbb{P}\left(S_t = X\right) = \binom{t}{k}^{-1}, \ \forall t \geq k, \forall X \subseteq N_t, \ |X| = k.$$

(b) S_t e S_ℓ sono dipendenti o indipendenti, per $t \neq \ell$?

Exercise 2. In questo esercizio costruiremo un algoritmo randomizzato per risolvere un problema di scheduling. Sia dato una rete rappresentata da un grafo diretto G senza lacci, dove ogni nodo rappresenta un processore e gli archi rappresentano dei cavi. Il problema consiste nello scegliere i percorsi da assegnare a N pacchetti di dati. Ogni pacchetto è caratterizzato da un nodo di partenza, da uno di arrivo e dal percorso esatto (cioè dal cammino sul grafo) che il pacchetto dovrà seguire. I pacchetti si muovono in tempi discreti, in modo che ad ogni istante di tempo al più un pacchetto può attraversare ogni singolo arco. Un pacchetto può aspettare in un qualsiasi nodo ad ogni istante di tempo.

Un *piano orario* per un insieme di pacchetti specifica i tempi di movimento dei pacchetti lungo i rispettivi cammini, specificando quali pacchetti devono attraversare quali archi ad ogni istante di tempo. Il nostro obbiettivo è quello di produrre un piano orario che minimizzi il tempo totale necessario per far arrivare a destinazione tutti i pacchetti.

(a) Sia d la massima distanza percorsa da un pacchetto e sia c il numero massimo di pacchetti che devono attraversare uno stesso arco nel loro percorso. Si dimostri che il tempo di percorrenza minimo per ogni piano orario è $\Omega(c + d)$.

(b) Si consideri il seguente piano orario che però ignora la congestione degli archi e permette a più pacchetti di percorrere lo stesso arco durante un singolo istante di tempo. Il piano orario associa ad ogni pacchetto un ritardo casuale, scelto uniformemente ed indipendentemente dall'intervallo $[1, \lceil \frac{\alpha c}{\log(Nd)} \rceil]$, dove α è una costante. Un pacchetto che riceve un ritardo x aspetta nel suo nodo di partenza per x istanti di tempo, poi percorre il suo cammino un arco alla volta, ignorando gli altri pacchetti e senza mai fermarsi. Si dia un

limite superiore alla probabilità che più di $O(\log(Nd))$ pacchetti usino un certo arco e ad un certo tempo t.

(c) Si consideri ancora il piano orario che ignora la congestione descritto al punto (b). Si dimostri che esiste una costante α tale per cui la probabilità che più di $O(\log(Nd))$ pacchetti passino attraverso un qualsiasi arco sia al più $\frac{1}{Nd}$.

(d) Si usi il piano orario che ignora la congestione per creare un semplice algoritmo randomizzato che, con alta probabilità (in N), produca un piano orario di lunghezza $O(c + d\log(Nd))$ con code di lunghezza $O(\log(Nd))$ e rispettando il vincolo che al più un pacchetto può attraversare un arco per volta.

Exercise 3. Si consideri l'algoritmo randomizzato per l'estrazione del k^o elemento in una lista non ordinata di numero distinti presentato di seguito e rispondere alle seguenti domande:

1. Si dimostri che l'algoritmo restituisce la risposta corretta.
2. Sia N il numero (casuale) di confronti operato dall'algoritmo. Quanto vale N nel caso peggiore?
3. Dimostrare che $\mathbb{E}[N] \in O(n)$.
4. **Bonus:** Dimostrare che $N \in O(n)$ con alta probabilità. *Suggerimento*: abbiamo visto a lezione un algoritmo simile...

Algorithm 2 Algoritmo randomizzato per l'estrazione del k^o elemento

Input: Una lista $S = \{x_1, x_2, \ldots, x_n\}$ di n numeri distinti
Output: il k^o numero più piccolo in S
Seleziona un elemento x di S uniformemente a caso. ▷ x è il pivot!
Confronta ogni elemento di S con x e dividi $S \setminus \{x\}$ in due liste:
    (a) S_1 contiene tutti gli elementi in S più piccoli di x
    (b) S_2 tutti quelli maggiori di x
Sia $i = |S_1|$
if $i = k - 1$ **then**
    Restituisci x
else if $i \geq k$ **then**
    Richiama ricorsivamente l'algoritmo per trovare il k^o elemento di S_1
else
    Richiama ricorsivamente l'algoritmo per trovare il $(k - i - 1)^o$ elemento di S_2

Exercise 4 (Stima del coefficiente di clustering)**.** Sia $G = (V, E)$ un grafo non diretto. Per ogni vertice $v \in V$, denotiamo con $N(v)$ il suo vicinato, i.e. $N(v) = \{u \in V | \{u, v\} \in E\}$. Il coefficiente di clustering C_v di un vertice v di G è definito come la probabilità che due vicini scelti a caso siano connessi da un arco:

$$C_v = \frac{|\{\{u, w\} \in E : u \in N(v) \text{ and } w \in N(v)\}|}{\binom{|N_v|}{2}}.$$

Se $|N(v)| < 2$, adottiamo la convenzione che $C_v = 0$. Il coefficiente di clustering C_G di un grafo G è la media tra i coefficienti di clustering dei suoi vertici:

$$C_G = \frac{1}{|V|} \sum_{v \in V} C_v$$

1. Descrivere un algoritmo deterministico che calcoli il coefficiente di clustering in tempo polinomiale (in $n = |V|$ ed $m = |E|$). Dimostrare la sua correttezza ed analizzare il suo costo computazionale.
2. Si consideri ora l'algoritmo randomizzato descritto di seguito. Dimostrare che, in valore atteso, restituisce una stima corretta del coefficiente di clustering, i.e. $\mathbb{E}[X_i] = C_G$.
3. Si dimostri che, per ogni $\varepsilon > 0$, $\ell \geq \frac{3}{\varepsilon^2 C_G} \ln(\frac{2}{\delta})$ campionamenti sono sufficienti ad ottenere, con probabilità almeno $1 - \delta$, una $(1 \pm \varepsilon)$ approssimazione di C_G, cioè:

$$(1 - \varepsilon)C_G \leq X \leq (1 + \varepsilon)C_G.$$

Algorithm 3 Algoritmo randomizzato per il calcolo del coefficiente di clustering

Input: Un grafo $G = (V, E)$, il numero ℓ di campionamenti da eseguire
Output: X, una stima di C_G
for $i = 1, \ldots, \ell$ **do**
    Sia u un nodo estratto uniformemente a caso
    if $|N(u)| > 1$ **then**
        Estrarre una coppia di nodi v, w da $N(u)$ uniformemente a caso (senza sostituzione)
        if $\{v, w\} \in E$ **then**
            $X_i \leftarrow 1$
        else
            $X_i \leftarrow 0$
Restituire $X \leftarrow \frac{1}{\ell} \sum_{i=1}^{\ell} X_i$