Sperimentazione sui tempi di CPU del metodo PageRank
Corso di LSMC, a.a. 2017-2018

Davide Gori
550282

April 4, 2018

1 Obiettivi e descrizione della sperimentazione

Vogliamo valutare come i tempi di cpu impiegati dal metodo di PageRank crescono al crescere della dimensione n della matrice di adiacenza. In particolare calcoleremo il tempo medio di esecuzione di un'iterazione dell'algoritmo al variare di n e di γ.
Per questo realizzeremo due sperimentazioni

2 Prima sperimentazione

In questo esercizio è stato necessario abbassare i valori di dimensione della matrice di adiacenza a causa della poca potenza e memoria del calcolatore. Di seguito il procedimento eseguito:

- Modifichiamo la funzione `PageRank` in `PageRank2`, che restituisce un vettore contenente rispettivamente il ranking, il tempo totale impiegano nelle iterazioni e il numero di iterazioni.
- per ogni $n = 100, 1000, 10000, 100000, 1000000$ generiamo tre matrici sparse (col comando `sprand`) con un numero di elementi non nulli pari a 0.1n, n, 10n.
- Invochiamo il comando `PageRank2` e prendiamo il quoziente tra tempo e numero di passi ad ogni chiamata.

3 Lo script

Questa è la funzione usata:

```
function [y, tempo, it] = PageRank2(H, v, gamma, itmax)
tic
n = size(H,1);
usn = 1/n;
e = ones(n,1);
d = H*e;
d = d';
dang = d==0;
dh = d + dang*n;
dh = 1./dh;
x = rand(1,n);
x = x/sum(x);
v = v/sum(v);
for it=1:itmax
   y = x.*dh;
   y = y*H + usn*sum(dang.*x);
   y = y*gamma+(1-gamma)*v;
   err = max(abs(x-y));
   x = y;
   %disp([it,err])
   if err<1.e-13*max(x)
      break
   end
end
tempo=toc;
it;
```

Questo è lo script che realizza la sperimentazione:

```
for k=1:5
    n=10*(10^k);
    disp(['caso ', num2str(n)])
    for g=1:3
        disp(['sottocaso densita ', num2str(10^(g-1)),'n'])
        d=10^(g-1) /n;
        H=sprand(n, n, d)~= 0;
        [y, t, it] =PageRank2(H, ones(1,n), 0.85, 1000);
        t
        it
        rapp=t/it
    end
end
```

4 Risultati

La tabella seguente riporta i valori in output:

n	el	t	it	$ratio$
100	n	0.0033152	40	8.2880e-05
100	10n	0.0026560	25	1.0624e-04
100	100n	8.0705e-04	2	4.0352e-04
1000	n	0.024854	174	1.4284e-04
1000	10n	0.0072720	25	2.9088e-04
1000	100n	0.023525	14	0.0016804
10000	n	0.040196	54	7.4437e-04
10000	10n	0.056684	26	0.0021802
10000	100n	0.24388	14	0.017420
100000	n	20.442	179	0.11420
100000	10n	14.392	26	0.55354
100000	100n	107.16	14	7.6546

Dove con el si intende il numero di elementi non nulli della matrice, con t il tempo di esecuzione della funzione `PageRank2` e con $ratio$ il rapporto tempo su numero di iterazioni.
Possiamo notare che per il caso $n = 100$ non ha senso la densità $100n$, infatti questo significa che la matrice è completamente piena di 1: non è un caso reale. Notiamo che all'aumentare della densità diminuisce il numero di passi impiegati dall'algoritmo per raggiungere una certa precisione, ma questo non impedisce al tempo di esecuzione di aumentare: infatti come possiamo notare dal caso $n = 10000$ (che è sicuramente il più rappresentativo dell'intera sperimentazione) il tempo impiegato mediamente per ogni singolo passo nel caso $el = 100n$ è circa 7.5 secondi.

5 Seconda sperimentazione

Vogliamo ora studiare come cambia il numero di iterazioni al variare del parametro γ (Google consiglia 0.85). Fissiamo la dimensione della matrice di adiacenza a 10000 e prendiamo un vettore di personalizzazione tale che $v(i) = 1$ per ogni $1 \leq i \leq n$. Facciamo variare γ e registriamo i tempi di esecuzione dell'algoritmo `PageRank`. Ripetiamo la procedura con tre matrici di adiacenza con numero di elementi rispettivamente n, $10n$ e $100n$.

- Usiamo la funzione `PageRank2` usata nella precedente sperimentazione, per misurare i tempi.
- Per ogni $\gamma = 0.5 : 0.01 : 0.99$ memorizziamo il numero di iterazioni effettuate dall' algoritmo `PageRank2` con i parametri descritti sopra e matrice di adiacenza di densità n.
- ripetiamo il punto precedente anche per le altre due densità.
- Disegnamo i tre grafici che hanno in ascissa i valori di γ e ordinata il numero di iterazioni effettuate.

6 Lo script

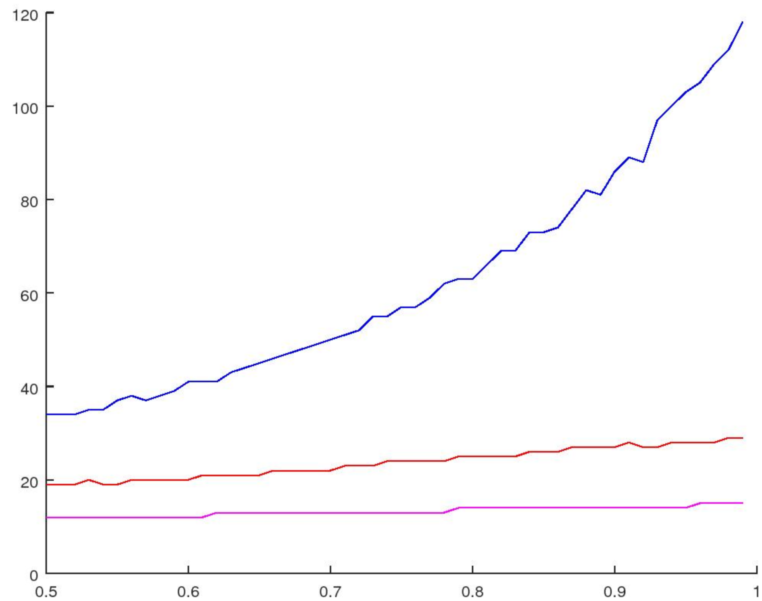
Questo eé lo script che realizza la sperimentazione:

```
q=[0.5:0.01:0.99];
n=10000;
A=sprand(n, n, 1/n)~= 0;
B=sprand(n, n, 10/n)~= 0;
C=sprand(n, n, 100/n)~= 0;
for i=1:length(q)
  [ya, ta, ita]=PageRank2(A, ones(1,n), q(i), 1000);
  [yb, tb, itb]=PageRank2(B, ones(1,n), q(i), 1000);
  [yc, tc, itc]=PageRank2(C, ones(1,n), q(i), 1000);
  a(i)=ita;
  b(i)=itb;
  c(i)=itc;
end
hold on
plot(q,a, "b")
plot(q,b, "r")
plot(q,c, "m")
```

Dove la funzione `PageRank2` è la stessa usata per la prima esperienza.

7 Risultati

Riportiamo il grafico in output:



Dove i grafici blu, rosso e magenta sono relativi alle matrici di adiacenza con un numero di elementi non nulli pari a n, $10n$, $100n$.
Si nota subito che più la matrice è sparsa e maggiore è il numero di iterazioni effettuate (in accordo con i risultati dell'esperimento precedente).
Le tre curve sono "grossomodo" crescenti, questo vuol dire che più γ è alto (e quindi si ha un'influenza del vettore di personalizzazione minore) più è necessario un maggior numero di iterazioni per ottenere una buona approssimazione.
Guardando invece la derivata delle tre curve si può evincere che questa è più piccola più la densità della matrice di adiacenza è grossa. Sembra inoltre che a densità molto basse la derivata della funzione cresca velocemente e quindi la derivata seconda sia positiva.
Queste osservazioni derivano esclusivamente da una mera osservazione dei dati nel grafico.