

Temă de casă 2 - Sisteme de ecuații liniare. Factorizări ortogonale

Responsabili:

Andrei Sfrent (andreisfrent@gmail.com)

Bogdan Știrbăț (bogdan.stirbat@gmail.com)

Mihaela Vasile (mihaela.a.vasile@gmail.com)

Data publicării: 14.03.2011

Data predării: 28.03.2011

Obiective:

În urma rezolvării acestei teme studentul va fi capabil să scrie programe în limbajul de programare Matlab, să utilizeze metode numerice pentru rezolvarea sistemelor de ecuații liniare, să recunoască sisteme de ecuații în situații practice.

Introducere

Fie un set de N resurse (pagini web). Fiecare pagină din acest set poate conține link-uri spre alte pagini. Spre unele pagini vor fi "îndreptate" mai multe link-uri decât spre altele. Cu alte cuvinte, un utilizator (care navighează pe internet, accesând diverse pagini întâmplător) va accesa cu o probabilitate mai mare unele resurse, iar alte resurse vor fi accesate cu o probabilitate mai mică. Putem spune că paginile care vor fi vizitate cu o probabilitate mai mare sunt mai "importante" decât celelalte (conțin informații mai multe, sunt lucrări științifice mai citate decât altele etc).

Un motor de căutare va trebui să redirecteze în prim-plan paginile cele mai importante (astfel încât dacă un utilizator caută un ac (anumite informații, în funcție de cuvintele-cheie tastate de acesta) în carul cu fân (mulțimea tuturor paginilor web), atunci această sarcină să nu fie proverbial de imposibilă). Astfel, un motor de căutare are nevoie de un mod de a măsura importanța unei resurse din cadrul unui set, și de un algoritm de calcul a acestei importanțe. Un algoritm care calculează acest factor este PageRank, iar "unitatea" de măsură folosită este indicele PageRank. Vom nota $PR(R)$ indicele PageRank al resursei R .

Pentru a înțelege cum funcționează acest algoritm, să ne imaginăm că vrem să calculăm cât de importantă este pagina A , de exemplu. Să notăm cu $M(A)$ mulțimea tuturor paginilor din care se poate ajunge la pagina A printr-un singur clic. Fie $B \in M(A)$. Evident, cu cât probabilitatea ca un utilizator să ajungă la pagina B este mai mare, cu atât probabilitatea de a ajunge la A este mai

mare. De asemenea, cu cât numărul de link-uri deținut de pagina B este mai mare (vom nota cu $L(B)$ acest număr) este mai mare, cu atât probabilitatea ca următoarea pagină vizitată să fie A este mai mică. Dacă luăm în considerare și celelalte pagini din $M(A)$, precum și probabilitatea ca un utilizator să continue navigatul pe internet (această probabilitate este dată de un coeficient d), atunci formula de a calcula $PR(A)$ (considerând că se știe $PR(C)$, $\forall C \in M(A)$) este:

$$PR(A) = \frac{1-d}{N} + d \sum_{B \in M(A)} \frac{PR(B)}{L(B)}$$

La adresa [1] veți găsi pseudocodul pentru diferiți algoritmi de a determina coeficienții PR (la secțiunea "Computation").

Logica fuzzy ([3]) este o logică care extinde logica clasică, booleană. Astfel, dacă în logica booleană o propoziție ia valori dintr-o mulțime a cărei cardinal este 2, în logica fuzzy valoarea de adevăr a unei propoziții $\in [0, 1]$. Aceste valori sunt date de așa-zise funcții membru ([5]).

Cerințe

Task1(30 p)

Să presupunem existența unui program care primește ca date de intrare o colecție de N resurse web și determină un graf, reprezentat printr-o listă de adiacență. Acest graf va fi afișat într-un fișier, astfel: pe prima linie va fi dat numărul N , iar pe următoarele linii vor fi date listele de vecini: o linie va începe cu numărul nodului (fie i acest parametru) pentru care se dau vecinii, va urma numărul de noduri cu care se învecinează nodul i , iar următoarele numere reprezintă nodurile cu care se învecinează i . Pe ultimele 2 linii sunt date valorile $val1$ și $val2$ (câte una pe linie; le veți folosi pentru a rezolva Task3). Pentru a rezolva cerințele temei, va trebui să construiți matricea de adiacență a acestui graf (notată cu A : $A(i, j) = 0$, dacă nodul i nu se învecinează cu nodul j , și 1 altfel.) Să se implementeze algoritmul "Iterative" de la adresa [1].

Observații:

1) Orice pagină web analizată conține cel puțin un link spre o altă pagină web. Asta înseamnă că matricea K este inversabilă.

2) Sunt unele pagini mai lungi care au link-uri spre ele însele (pentru a permite o navigare mai ușoară), deci nu toate elementele de pe diagonala principală a matricii A sunt 0. În analiza efectuată, aceste link-uri nu au nicio semnificație, deci acestea nu vor intra în calcul. Deci $A(i, i) = 0$, $\forall i \in \{1, 2, \dots, N\}$.

3) Algoritmul va fi implementat în fișierul `Iterative.m`; funcția `Iterative` va primi ca argumente, în această ordine: numele fișierului din care va citi graful, parametrul d (vezi descrierea lui mai sus), parametrul eps (eruaarea care apare în calculul vectorului PR). Va avea ca dată de ieșire vectorul PR .

Task2(40 p)

Să se implementeze algoritmul "Algebraic" de la adresa [1]. Algoritmul se va implementa în fișierul Algebraic.m; funcția Algebraic va primi ca argumente, în această ordine: numele fișierului de intrare (care are structura fișierului de intrare de la Task1), parametrul d (care are aceeași interpretare ca la Task1). Va avea ca dată de ieșire vectorul PR .

Observații. Pentru a calcula inversa unei matrici, se va folosi algoritmul Gram-Schmidt:

fie T o matrice (cu n linii și n coloane) inversabilă pentru care se cere să se determine T^{-1} ; $T = [t_1 \ t_2 \ \dots \ t_n]$, iar $T^{-1} = [x_1 \ x_2 \ \dots \ x_n]$. Deci,

$$T \cdot T^{-1} = I_n$$

$$T \cdot [x_1 x_2 \dots x_n] = [e_1 e_2 \dots e_n]$$

$$T \cdot x_i = e_i(*)$$

unde e_i este coloana care are 0 peste tot, în afară de poziția i (unde are 1).

Pentru a afla T^{-1} se va rezolva sistemul (*) pentru fiecare i în parte, folosind algoritmul Gram-Schmidt.

(Indicație) Puteți aplica algoritmul Gram-Schmidt o singură dată, pentru a afla Q și R astfel încât $T = Q \cdot R$; pe baza matricilor Q și R veți rezolva apoi cele n sisteme de ecuații.

Task3(30 p)

Fie următoarea funcție membru:

$$u(x) = \begin{cases} 0, & x \in [0, val1); \\ a \cdot x + b, & x \in [val1, val2]; \\ 1, & x \in [val2, 1] \end{cases}$$

unde $val1$ și $val2$ sunt date în fișierul de intrare, așa cum este descris la Task1; a și b sunt valori calculate de voi astfel încât $u(x)$ să fie o funcție continuă. Această funcție indică gradul de apartenență al paginii a cărui PageRank este x la mulțimea "paginilor importante."

Să se scrie fișierul tema2.m; funcția "tema2" primește ca date de intrare, în această ordine, un nume de fișier, parametrul d , parametrul eps . Toți acești parametri au interpretarea de mai sus. tema2.m va scrie un nou fișier, a cărui nume este dat de numele fișierului primit ca parametru, la care se concatenează șirul ".out". Va scrie în noul fișier, pe prima linie, numărul N (numărul de pagini web analizate), va calcula vectorul PR folosind primul algoritm și-l va scrie în fișierul .out pe N linii, apoi va calcula vectorul PR folosind al doilea algoritm și-l va scrie în fișierul .out; se va lăsa un rând gol între N și primul vector, și un rând gol între cei doi vectori. După acest pas, se va ordona descrescător vectorul PR calculat de cel de-al doilea algoritm (folosind orice algoritm de sortare, se va nota $PR1$ acest vector sortat). Se va lăsa în fișierul de ieșire un spațiu liber, apoi se vor afișa N linii de forma:

$i \ j \ F$

unde i reprezintă indici în vectorul $PR1$ (se vor afișa în ordinea: 1, 2, 3, ... N), j reprezintă nodul a cărui PageRank este $PR1(i)$, iar $F = u(PR1(i))$. Practic, se va face un clasament al celor mai "importante" pagini, clasament în care interesează locul obținut (adică numărul i), numărul paginii care a obținut acest loc, și gradul de apartenență a acestei pagini la mulțimea paginilor "importante".

Exemple de date de intrare și de rezultate

Exemplu1:

$d = 0.85$, $eps = 0.001$;

Conținutul fișierului "graf1" este:

```
7
1 3 2 3 4
2 3 3 4 5
3 5 1 2 5 6 7
4 5 1 2 3 6 7
5 3 4 6 7
6 3 1 4 5
7 4 1 2 3 5
0.001
0.95
```

Funcția tema2.m va scrie fișierul "graf1.out":

```
7
0.137730
0.142355
0.156163
0.176215
0.147967
0.119785
0.119785
0.137419
0.142396
0.156189
0.176532
0.147754
0.119855
0.119855
1 4 0.184965
2 3 0.163529
3 5 0.154641
4 2 0.148994
5 1 0.143750
6 7 0.125242
7 6 0.125242
```

Exemplu2:

$d = 0.85$, $eps = 0.001$;

Fișierul "graf2" conține:

```
8
1 2 2 3
2 5 1 2 4 5 8
3 4 1 2 7 8
4 5 1 2 3 5 6
5 6 1 2 3 4 7 8
6 3 5 7 8
7 5 1 2 3 7 8
8 3 1 2 3
0.08
0.85
```

Funcția tema2.m va scrie fișierul "graf2.out":

```
8
0.185532
0.218111
0.179646
0.077618
0.087414
0.031813
0.078387
0.141480
0.185572
0.218095
0.179872
0.077464
0.087308
0.031919
0.078385
0.141387
1 2 0.179344
2 1 0.137106
3 3 0.129704
4 8 0.079723
5 5 0.009490
6 7 0.000000
7 4 0.000000
8 6 0.000000
```

Resurse:

- 1) <http://en.wikipedia.org/wiki/PageRank>
- 2) <http://www.cs.huji.ac.il/~csip/CSIP2007-intro.pdf>
- 3) http://en.wikipedia.org/wiki/Fuzzy_logic
- 4) <http://www.seattlerobotics.org/encoder/mar98/fuz/flindex.html>
- 5) <http://www.atp.ruhr-uni-bochum.de/rt1/syscontrol/node117.html>