METODE NUMERICE: Tema #2
Let's Google!

Termen de predare: 30 APRILIE 2016

Titulari curs: *Florin Pop, George Popescu*

Obiectivele Temei

Obiectivele generale ale acestei teme de casă sunt:

- Familiarizarea cu metoda PageRank folosită de motoatele de cătare pentru calculul relevanţei unei pagini web;
- Implementatarea algoritmilor `Iterative`, `Algebraic` şi `Power Method` pentru problema PageRank;
- Să calculeze inversa unei matrici folosind o metoda stabilă numeric.
- Calculul Gradului de Apartenenţă.

Descriere generala

PageRank este un algoritm de analiză a hiperlegăturilor din Internet, folosit de motorul de căutare Google pentru a acorda o pondere fiecărui element dintr-o mulţime de documente interconectate prin hiperlegături, cu scopul măsurării importanţei relative în cadrul mulţimii. Fie un set de N resurse (pagini web). Fiecare pagină din acest set poate conţine 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 (conţin informaţii mai multe, sunt lucrări ştiinţifice 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 informaţii, în funcţie de cuvintele-cheie tastate de acesta) *în carul cu fân* (mulţimea 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 importanţa unei resurse din cadrul unui set, şi de un algoritm de calcul a acestei importanţe. 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 înţelege cum funcţionează acest algoritm, să ne imaginăm că vrem să calculăm cât de importantă este pagina A, de exemplu. Să notăm cu $M(A)$ mulţimea 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 deţinut 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 şi celelalte pagini din $M(A)$, precum şi probabilitatea ca un utilizator să continue navigatul pe Internet (această probabilitate este dată de un coeficent d), atunci formula de a calcula $PR(A)$ (considerând că se ştie $PR(B)$, $\forall B \in M(A)$) este:

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

La adresa web [1] (la secţiunea Computation) veţi găsi pseudocodul pentru diferiţi algoritmi de a determina coeficenţii PR.

O metoda importantă folosită in algoritmul de PageRank este bazată pe funcţiile membru din logica fuzzy. *Logica fuzzy* ([3]) este o logică care extinde logica clasică, booleană. Astfel, dacă în logica booleană o propoziţie ia valori dintr-o mulţime a cărei cardinal este 2, în logica fuzzy valoarea de adevăr a unei propoziţii $\in [0, 1]$. Aceaste valori sunt date de aşa-zise *funcţii membru* ([5]).

Cerinta 1. Algoritmul *Iterative* (30p)

Să presupunem existenţa unui program care primeşte ca date de intrare o colecţie de N resurse web şi determină un graf, reprezentat printr-o listă de adiacenţă. Acest graf va fi afişat într-un fişier, 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 val_1 şi val_2 (câte una pe linie; le veţi folosi pentru a rezolva cerinţele următoare alte temei). Pentru a rezolva cerinţele temei, va trebui să construiţi matricea de adiacenţă a acestui graf (notată cu A: $A(i, j) = 0$, dacă nodul i nu se învecinează cu nodul j, şi 1 altfel).

Pentru calcularea indicelui PageRank folosind algoritmul Iterative se alege iniţial $PR(p_i, 0) = \frac{1}{N}$. La paşii următori se foloseşte relaţia de recurenţa:

$$PR(p_i, t+1) = \tfrac{1-d}{N} + d \textstyle\sum_{p_j \in M(p_i)} \tfrac{PR(p_j,t)}{L(p_j)}$$

Sau se poate folosi forma matriceală:

$$R(t+1) = d\mathcal{M}R(t) + \frac{1-d}{N}\mathbf{1}. \quad (2)$$

unde $R_i(t) = PR(p_i, t)$, $\mathbf{1}$ este vectorul coloană de lungime N care conţine numai 1.

Matricea $\mathcal{M}$ este definită astfel:

$$\mathcal{M}_{ij} = \begin{cases} \frac{1}{L(p_j)} & \text{dacă j are link către i} \\ 0 & \text{altfel} \end{cases}$$

Sau $\mathcal{M} = (K^{-1}A)^T$, unde A este matricea de adiacenţă, iar K este o matrice diagonală cu gradele de ieşire pe diagonală.

Algoritmul se opreşte când se atinge pentru un ϵ:

$|R(t+1) - R(t)| < \epsilon$

Să se implementeze algoritmul *Iterative* descris mai sus. Observaţii:

1. Orice pagină web analizată conţine cel puţin un link spre o altă pagină web. Asta înseamnă că matricea K (din algoritmul *Iterative*) este inversabilă.
2. Sunt unele pagini mai lungi care au link-uri spre ele însele (pentru a permite o navigare mai uşoară), deci nu toate elementele de pe diagonala principală a matricii A sunt 0. În analiza efectuată, aceste link-uri nu au nicio semnificaţie, deci acestea nu vor intra în calcul. Deci $A(i, i) = 0, \forall i \in \{1, 2, \ldots, N\}$.

3. Algoritmul va fi implementat în fişierul `Iterative.m`; funcţia `Iterative` va primi ca argumente, în această ordine: numele fişierului din care va citi graful, parametrul d (vezi descrierea lui mai sus), parametrul *eps* (eroarea care apare în calculul vectorului PR). Va avea ca dată de ieşire vectorul PR.

```
function R = Iterative(nume, d, eps)
  % Functia care calculeaza matricea R folosind algoritmul iterativ.
  % Intrari:
  % -> nume: numele fisierului din care se citeste;
  % -> d: coeficentul d, adica probabilitatea ca un anumit navigator sa
    continue navigarea (0.85 in cele mai multe cazuri)
  % -> eps: eroarea care apare in algoritm.
  % Iesiri:
  % -> R: vectorul de PageRank-uri acordat pentru fiecare pagina.
```

Listing 1: Iterative.m

4. Toate fişierele de intrare vor conţine pe prima linie numărul N, apoi pe următoarele N linii matricea de adiacenţă.

Cerinta 2. Algoritmul *Algebraic* (25p)

Când $t \to \infty$ ecuaţia 2 devine

$$R = d\mathcal{M}R + \frac{1-d}{N}\mathbf{1}. \tag{3}$$

Aceasta are soluţia $R = (\mathcal{I} - d\mathcal{M})^{-1}\frac{1-d}{N}\mathbf{1}$, unde $\mathcal{I}$ este matricea identitate.

Să se implementeze algoritmul *Algebraic* descris mai sus, folosind fişierul `Algebraic.m`; funcţia `Algebraic` va primi ca argumente, în această ordine: numele fişierului de intare (care are structura fişierului de intrare de la Cerinţa 1), parametrul d (care are aceeaşi interpretare ca la Cerinţa 1). Va avea ca dată de ieşire vectorul PR.

```
function R = Algebraic(nume, d)
  % Functia care calculeaza vectorul PageRank folosind varianta algebrica de
    calcul.
  % Intrari:
  % -> nume: numele fisierului din care se citeste;
  % -> d: probabilitatea ca un anumit utilizator sa continue navigarea la o pagina
     urmatoare.
  % Iesiri:
  % -> R: vectorul de PageRank-uri acordat pentru fiecare pagina.
```

Listing 2: Algebraic.m

Observaţie. Pentru a calcula inversa unei matrici, se va folosi algoritmul **Gram-Schmidt**: fie T o matrice (cu n linii şi n coloane) inversabilă pentru care se cere să se determine T^{-1}. Avem: $T = \begin{bmatrix} t_1 & t_2 & \dots & t_n \end{bmatrix}$, iar $T^{-1} = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}$ şi $T \cdot T^{-1} = I_n$. Deci,

$$T \cdot \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix} = \begin{bmatrix} e_1 & e_2 & \dots & e_n \end{bmatrix} \tag{4}$$

$$T \cdot x_i = e_i. \tag{5}$$

unde e_i este coloana i din matricea unitate I_n.

Pentru a afla T^{-1} se va rezolva sistemul 5 pentru fiecare i în parte, folosind algoritmul Gram-Schmidt optimizat.

Indicaţie. Puteţi aplica algoritmul Gram-Schmidt o singură dată, pentru a afla Q şi R astfel încât $T = Q \cdot R$; pe baza matricilor Q şi R veţi rezolva apoi cele n sisteme de ecuaţii.

```
function B = GramSchmidt(A)
  % Functia care calculeaza inversa matricii A folosind factorizari Gram-Schmidt
  % Se va inlocui aceasta linie cu descrierea algoritmului de inversare
```

Listing 3: GramSchmidt.m

Cerinta 3. *Power Method* (25p)

Dacă matricea $\mathcal{M}$ este stochastică "pe coloane" (suma elementelor de pe fiecare coloană este 1) şi vectorul R este o distribuţie de probabilitate, ecuaţia 3 este echivalentă cu

$$R = \left(d\mathcal{M} + \frac{1-d}{N}\mathbf{E} \right) R = \mathcal{P}R, \tag{6}$$

unde $\mathbf{E} = \mathbf{1} \cdot \mathbf{1}^T$ (o matrice de dimensiune $N \times N$, plină de 1).

Matricea $\mathcal{P}$ va fi de asemenea stochastică. Astfel, valoarea sa proprie cu magnitudinea maximă va fi $\lambda = 1$ şi va avea un unic vector propriu corespunzător. Detalii asupra acestor proprietăţi pot fi găsite la [7]. Astfel că R va fi de fapt vectorul propriu principal al matricii $\mathcal{P}$.

Se poate verifica faptul că matricea M şi vectorul R verifică proprietăţile menţionate mai sus. În consecinţă, să se calculeze, folosind metoda puterii, vectorul Page Rank R ca vector propriu al matricii $\mathcal{P}$, folosind o condiţie de oprire similară cu cea de la Cerinţa 1, pentru o valoare ϵ dată.

Algoritmul va fi implementat în fişierul `Power.m`; funcţia `Power` va primi ca argumente, în această ordine: numele fişierului de intare (care are structura fişierului de intrare de la cerinţele anterioare), parametrul d (care are aceeaşi interpretare ca la cerinţele anterioare), parametrul *eps* (eroarea care apare în calculul vectorului PR). Va avea ca dată de ieşire vectorul PR.

```
function R = Power(nume, d, eps)
  % Functia care calculeaza vectorul PageRank folosind metoda puterii.
  % Intrari:
  % -> nume: numele fisierului din care se citeste;
  % -> d: probabilitatea ca un anumit utilizator sa continue navigarea la o pagina
     urmatoare.
  % -> eps: eroarea care apare in algoritm.
  % Iesiri:
  % -> R: vectorul de PageRank-uri acordat pentru fiecare pagina.
```

Listing 4: Power.m

Cerinta 4. Gradul de Apartenenta (20p)

Fie următoarea funcţie membru:

$$u(x) = \begin{cases} 0, & x \in [0, val_1); \\ a \cdot x + b, & x \in [val_1, val_2]; \\ 1, & x \in (val_2, 1] \end{cases} \tag{7}$$

unde val_1 şi val_2 sunt date în fişierul de intrare, aşa cum este descris la Cerinţa 1; a şi b sunt valori calculate de voi astfel încât $u(x)$ să fie o funcţie continuă (conforma cu 7). Această funcţie indică gradul de apartenenţă al paginii a cărui PageRank este x la mulţimea *paginilor importante*.

```
function y = Apartenenta(x, val1, val2)
  % Functia care primeste ca parametrii x, val1, val2 si care calculeaza valoarea
    functiei membru in punctul x.
  % Stim ca 0 <= x <= 1
```

Listing 5: Apartenenta.m

Să se scrie fişierul `PageRank.m`; funcţia `PageRank` primeşte ca date de intrare, în această ordine, un nume de fişier, parametrul d, parametrul eps. Toţi aceşti parametrii au interpretarea de mai sus.

```
function [R1 R2 R3] = PageRank(nume, d, eps)
  % Calculeaza indicii PageRank pentru cele 3 cerinte
  % Scrie fisierul de iesire nume.out
```

Listing 6: PageRank.m

`PageRank.m` va scrie un nou fişier, a cărui nume este dat de numele fişierului primit ca parametru, la care se concatenează şirul `.out`. Va scrie în noul fişier, pe prima linie, numărul N (numărul de pagini web analizate), urmat de un rând liber; va calcula vectorii PR rezultaţi din folosirea celor 3 algoritmi şi îi va scrie în fişierul .out, fiecare pe câte N linii, urmaţi de câte un rând liber. După acest pas, se va ordona descrescător vectorul PR calculat de cel de-al doilea algoritm (folosind orice algoritm de sortare, se va nota PR_1 acest vector sortat). Se vor afişa în fişierul de ieşire N linii de forma:

```
i j F
```

unde i reprezintă indici în vectorul PR_1 (se vor afişa în ordinea: $1, 2, 3, \ldots, N$), j reprezintă nodul a cărui *PageRank* este $PR_1(i)$, iar $F = u(PR_1(i))$. Practic, se va face un clasament al celor mai *importante pagini*, clasament în care interesează locul obţinut (adică numărul i), numărul paginii care a obţinut acest loc, şi gradul de apartenenţă a acestei pagini la mulţimea *paginilor importante*.

Exemple de date de intrare si de rezultate

Exemplu 1

$d = 0.85$, $eps = 0.001$; Conţinutul fişierului `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
```

Funcţia `PageRank.m` va scrie fişierul `graf1.out`:

```
7

0.137326
```

```
0.142411
0.156197
0.176649
0.147704
0.119857
0.119857

0.137419
0.142396
0.156189
0.176532
0.147754
0.119855
0.119855

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 6 0.125242
7 7 0.125242
```

Exemplu 2

$d = 0.85$, $eps = 0.001$; Fişierul `graf2` conţine:

```
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
```

Funcţia `PageRank.m` va scrie fişierul `graf2.out`:

```
8

0.185595
0.218098
0.179923
0.077482
```

```
0.087307
0.031945
0.078322
0.141328

0.185572
0.218095
0.179872
0.077464
0.087308
0.031919
0.078385
0.141387

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
```

Detalii de implementare si redactare

Tema de casă va implementa funcţiile menţionate la fiecare cerinţă în parte. Pentru implementarea temei puteţi folosi si alte funcţii definite de voi, dar cele menţionate mai sus sunt obligatorii. Trebuie să ţineţi cont de următoarele aspecte:

- codul sursă va conţine comentarii semnificative şi sugestive cu privire la implementarea algoritmilor;
- existenţa unui fişier `readme.txt` care va prezenta detaliile legate de implementarea ţi testarea temei;
- fişierele care compun tema de casa vor fi incluse într-o arhivă `.zip` care respectă specificaţiile din regulamentul cursului;
- tema se va implementa in Maltab şi va fi testată în mediul Octave.

Resurse Web

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
6. https://en.wikipedia.org/wiki/Perron-Frobenius-theorem

7. http://citeseerx.ist.psu.edu/viewdoc