

Calcolo del PageRank data una matrice di adiacenza G.

```
R = pagerank(G);  
[R, OUT, IN] = pagerank(G);
```

Argomenti di input

G - Matrice di adiacenza

G è la matrice sparsa di adiacenze relativa ad un sottoinsieme del web. Deve essere quadrata e di tipo sparse logical.

Tipo: sparse | logical

Argomenti di output

R - Vettore dei rank delle pagine

R è il vettore dei page rank associati alle pagine del sottoinsieme del web rappresentato da G.

Tipo: double

OUT - Vettore dei corrispondenti *outdegree*

OUT(i) contiene il numero di link uscenti dal nodo (i), il cui pagerank è R(i).

Tipo: double

IN - Vettore dei corrispondenti *indegree*

IN(i) contiene il numero di link entranti nel nodo (i), il cui pagerank è R(i).

Tipo: double

Algoritmo

Utilizzando un metodo iterativo, viene calcolato il vettore R dei PageRank, ovvero dei pesi numerici associati alle pagine del sottoinsieme del web analizzato, con lo scopo di classificarle in base alla loro importanza relativa all'interno dell'insieme. In particolare il calcolo di R viene effettuato attraverso la seguente formula:

$$R^{k+1} = p * G * D * R^k + e * (z^T * R^k)$$

dove k rappresenta il passo di iterazione nel calcolo di R .

Correlati

centrality

```
function [R, OUT, IN] = pagerank(G)

p = 0.85;
NMAX = 200;
TOL = 10^-7;

if isempty(G)
    error("La matrice di input è vuota.");
end

if (~issparse(G))
    error("La matrice di input non è sparsa.");
end

if (~islogical(G))
    error("La matrice di input non contiene solo elementi logici.");
end

[n m] = size(G);

if (n ~= m)
    error("La matrice di input non è quadrata.");
end

if (n < 2)
    error("Dimensioni della matrice di input non corrette.");
end

G = spdiags(zeros(n,1), 0, G);

NJ = sum(G);
index_1 = find(NJ);
index_2 = find(NJ == 0);

% lo troviamo un altro modo
temp(index_1) = 1./NJ(index_1);
temp(index_2) = 0;
D = diag(temp);

e = ones(n,1);

z(index_1,1)=(1-p)/n;
z(index_2,1)=1/n;

pGD = p*G*D;
```

```

x0 = zeros(n,1)+1/n;

R = pGD * x0 + e * (z' * x0);
niter = 1;

while ( niter < NMAX && norm(R - x0)/norm(R) >= TOL )
    x0 = R;
    R = pGD * x0 + e * (z' * x0);
    niter = niter+1;
end

if( niter == NMAX )
    warning("Numero di iterazioni massimo raggiunto.");
end

OUT = NJ;
IN = sum(G,2);

end

```