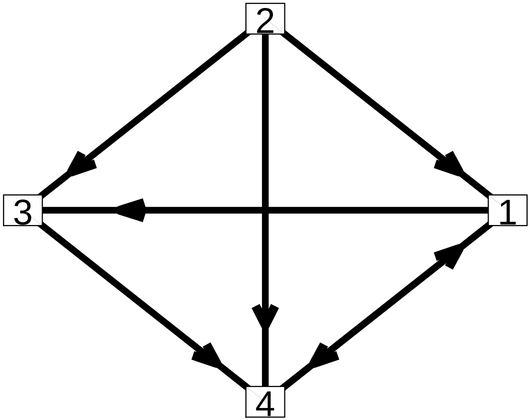
The google PageRank algorithm, session 3



Adjacency matrix.

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix}.$$

The **out-degrees** are the column sums and these are 2, 3, 1 and 1.

The out-degrees are used when constructing the conditional probabilty matrix C.

The conditional probability matrix C

Adjacency matrix.

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix}.$$

The out-degrees are the column sums and these are 2, 3, 1 and 1. The probability matrix C is

$$C = \begin{pmatrix} 0 & 1/3 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1/2 & 1/3 & 0 & 0 \\ 1/2 & 1/3 & 1 & 0 \end{pmatrix}$$

The entries in each column have been divided by the out-degree associated with the column. With this matrix this is okay as all the out-degrees are greater than 0.

A function file to get C given A – vectorised

A version of a function m-file to get the probability matrix C from a valid adjacency matrix A can be as follows.

```
function C = cmat1_no_checks(A)
% function C = cmat1_no_checks(A) determines the
% probability matrix C from an adjacency matrix A

% get the dimensions, it needs to be a square matrix
n=size(A, 1);

% get the out degrees which are the column sums
outdeg=sum(A);

% set C column-by-column
C = zeros(n, n);
for from=1:n
  C(:, from) = A(:, from)/outdeg(from);
end
```

A function file to get C given A – componentwise

```
function C = cmat1(A)

N = size(A, 1);

% get a column vector of the out-degrees
outdeg=zeros(N, 1);
for i=1:N
    % sum of elements in column i
    outdeg(i) = sum(A(:, i));
end

C = zeros(N, N);
% loop through the columns and each entry in each column
for from=1:N
    for to=1:N
        C(to, from) = A(to, from)/outdeg(from);
    end
end
```

A function file to get C given A and α – vectorised

```
function C = cmat2_mkw(A, alpha)

N = size(A, 1);

outdeg=sum(A);

C = zeros(N, N);

for from=1:N
    if outdeg(from)==0
      C(:, from)=1/(N-1);
    else
      C(:, from)=(1-alpha)*A(:, from)/outdeg(from)+...
                 alpha/(N-1);
    end
    C(from, from)=0;
end
```

A function file to get C given A and α – componentwise

```
function C = cmat2(A, alpha)

N = size(A, 1);
outdeg=zeros(N, 1);
for i=1:N
    outdeg(i) = sum(A(:, i));
end

C = zeros(N, N);
for from=1:N
    for to=1:N
        if from==to
            C(to, from) = 0;
        elseif outdeg(from)>0
            C(to, from) = alpha/(N-1) ...
                +(1-alpha)*A(to, from)/outdeg(from);
        else
            C(to, from) = 1/(N-1);
        end
    end
end
```

Plotting the paths – getting started

```
% adjacency matrix for Network 6(a)
A = [ 0 1 0 1 ;
      0 0 0 0 ;
      1 1 0 0 ;
      1 1 1 0 ];

% matrix C is computed by the function cmat2
C = cmat2(A, 0.0)

% set num of iters and starting prob vector
iter = 30;
p = [1  0  0  0]';

% ...more statements
```

Plotting the paths – figure, clf, hold on/off etc

```
figure(11);
clf
hold on;

% create the points and show something
for i=1:iter

    % 1. plot each component of the probabilities vector
    plot(i, p(1), '^r', i, p(2), 'ok', i, p(3), '+b', ...
         i, p(4), '*k', 'MarkerSize', 6);

    % 2. update to the next state
    p = C*p;
end

legend('P1', 'P2', 'P3', 'P4');
xlabel('iteration');
hold off;
```

Saving the points and then plotting curves

```
% ..earlier statements not shown
n=size(A, 1);
pp=zeros(n, iter);
pp(:, 1)=p;

% create the points and show something
for i=2:iter
    pp(:, i) = C*pp(:, i-1);
end

figure(12);
x=1:iter;
plot(x, pp(1, :), x, pp(2, :), x, pp(3, :), x, pp(4, :));
legend('P1', 'P2', 'P3', 'P4');
xlabel('iteration');
```

Saving the points and then plotting as points

```
% ..earlier statements not shown
n=size(A, 1);
pp=zeros(n, iter);
pp(:, 1)=p;

% create the points and show something
for i=2:iter
    pp(:, i) = C*pp(:, i-1);
end

figure(14);
x=1:iter;
plot(x, pp(1, :), '^r', x, pp(2, :), 'ok', ...
     x, pp(3, :), '+b', x, p(4, :), '*k', ...
     'MarkerSize', 6);
legend('P1', 'P2', 'P3', 'P4');
xlabel('iteration');
```

Saving the points and using hold

```
% ..statements as before
figure(15);
clf
hold on
x=1:iter;
plot(x, pp(1, :), '^r', 'MarkerSize', 6);
plot(x, pp(2, :), 'ok', 'MarkerSize', 6);
plot(x, pp(3, :), '+b', 'MarkerSize', 6);
plot(x, pp(4, :), '*k', 'MarkerSize', 6);
legend('P1', 'P2', 'P3', 'P4');
xlabel('iteration');
hold off
```

`figure`, `clf`, `hold on`, `hold off` mechanism

```
figure(15);
clf
hold on
% .. plotting statements
hold off
```

With this set-up several plots are put on top of each other.

`clf` ensures that the figure window is clear at the start.

Remember to have a `hold on` if you have used a `hold off`.

The steady state – an eigenvector

Mathematically we get each new probability by multiplying by C and in the Matlab program we have

```
p=C*p;
```

We overwrite the old `p` with the next `p`.

The examples suggest that there is a limit as the number of iterations tends to ∞. Mathematically this vector is such that

$$C\underline{p} = \underline{p}$$

with $\underline{p} = (p_i)$, $p_i \ge 0$ and

$$p_1 + p_2 + \cdots + p_n = 1.$$

$\underline{p}$ is an **eigenvector** of C with **eigenvalue** 1 normalised in the above way.

Other ways of writing things

$$C\underline{p} = \underline{p} \quad \text{is} \quad (C - I)\underline{p} = \underline{0}.$$

The last case is a homogeneous system with a non-trivial solution.

Once way to getting the specific solution that we want is to construct the augmented system.

$$\begin{pmatrix} -1 & c_{12} & \cdots & c_{1N} \\ c_{21} & -1 & \cdots & c_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & -1 \\ 1 & 1 & \cdots & 1 \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_N \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}.$$

The augmented matrix has size $(N + 1) \times N$. Although there are more equations than unknowns it can be shown that there is a unique solution.

Group homework

Network 10 has 3 nodes and the following is the adjacency matrix.

$$A = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

1. Construct the matrix C in the basic model.
2. Write down the homogeneous system

$$(C - I)\underline{p} = \underline{0}.$$

3. Check that

$$\det(C - I) = ??$$

   What is ??

Group homework continued

4. Get the general solution to

$$(C - I)\underline{x} = \underline{0}.$$

5. If the general solution is written as $\underline{x} = \underline{x}(t)$ for a free parameter t then what value of t is such that $\underline{x}(t) = \underline{p}$ where $\underline{p}$ is the probability vector?
6. Replace one of the 3 equations in

$$(C - I)\underline{p} = \underline{0}.$$

with

$$p_1 + p_2 + p_3 = 1.$$

Check that the solution is unchanged.