MATH10017 Assessed Coursework 3 (2023/24)

Matteo Fasiolo

Context

Consider the task of ranking academic articles using their bibliographies. In particular, papers that are cited by many other articles can be seen as having high **impact** on academic research. For example the paper:

- Tibshirani, R., 1996. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society Series B: Statistical Methodology, 58(1), pp.267-288.

was the first to propose lasso regression (nowadays widely use in statistics) and has over 50 thousand citations in February 2024. On the other hand, papers that refer to many other papers (i.e., papers that have long bibliographies) can be seen as useful **knowledge** bases on existing research.

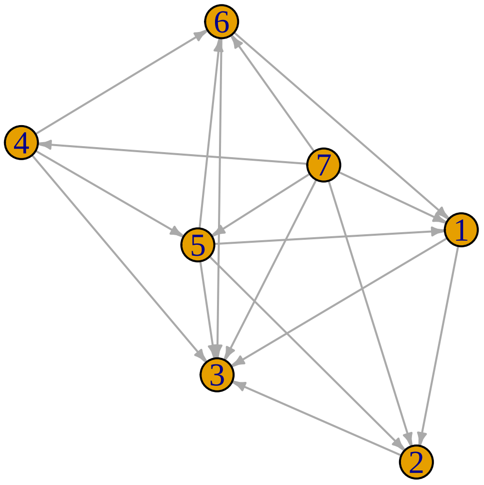
Assume that we want to compute the **impact** and the **knowledge** scores of a set of papers, based on their bibliographies. Citation data can be represented via a graph, where nodes represent papers and edges citations. In particular, consider the matrix A with elements:

- $A_{i,j} = 1$ if paper i cites j.
- $A_{i,j} = 0$ if paper i does not cite j.
- $A_{i,i} = 0$ as self-citations do not exist.

In this project we will consider the following example of the matrix A:

```
0 1 1 0 0 0 0
0 0 1 0 0 0 0
0 0 0 0 0 0 0
0 0 1 0 1 1 0
1 1 1 0 0 1 0
1 0 1 0 0 0 0
1 1 1 1 1 1 0
```

The matrix is 7×7, the i-th row showing which papers are being cited by the i-th article. Note that the matrix can be represented by the following graph:



Just by looking at the graph, we see that paper 3 is quite impactful, while paper 7 is a knowledge base (see also column 3 and row 7 of matrix A).

The algorithm

Recall that we want to quantify the value of a set of papers on the basis of their impact and usefulness as knowledge bases, and that all the information we have is the matrix A. Assume that we are dealing with n articles, so that A is $n \times n$. Let $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ be vectors of length n and let their i-th elements represent, respectively, the impact and the knowledge-base score of the i-th paper. Let ε be a small positive constant (e.g., 10^{-6}) and M a positive integer (e.g., 100). All the elements of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are initialised at 1 and updated via the following iterative algorithm:

- Repeat the following steps for up to M iterations

    1. Assign $\boldsymbol{\alpha}^{\text{old}} \leftarrow \boldsymbol{\alpha}$ and $\boldsymbol{\beta}^{\text{old}} \leftarrow \boldsymbol{\beta}$.
    2. Set $\alpha_i \leftarrow \sum_{j=1}^{n} A_{j,i}\beta_j$, for $i = 1, \dots, n$.
    3. Set $\beta_i \leftarrow \sum_{j=1}^{n} A_{i,j}\alpha_j$, for $i = 1, \dots, n$.
    4. Normalise by doing
    $$\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha}/||\boldsymbol{\alpha}||, \quad \boldsymbol{\beta} \leftarrow \boldsymbol{\beta}/||\boldsymbol{\beta}||,$$
    where $||\boldsymbol{\alpha}|| = \sqrt{\sum_{i=1}^{n} \alpha_i^2}$ (Euclidean norm).
    5. If $\max\{|\alpha_1 - \alpha_1^{old}|, \dots, |\alpha_n - \alpha_n^{old}|\} > \varepsilon$ or $\max\{|\beta_1 - \beta_1^{old}|, \dots, |\beta_n - \beta_n^{old}|\} > \varepsilon$ go back to step 1, otherwise terminate.

The reasoning behind the algorithm above is the following. We iteratively update the impact and the knowledge scores, in particular the impact of a paper is computed using the knowledge scores of the papers that are citing it, while the knowledge score of a paper is computed using the impact scores that the paper is citing. Normalisation is used to prevent the scores from getting bigger and bigger with the iterations. If the scores remain roughly unchanged between two iterations we assume that the algorithm has converged and we terminate.

Your Task

Your code is to implement the algorithm above and to apply it to the matrix A given at the beginning of this document. The details of the implementation are up to you but your code should contain at least one class, e.g. something like:

```
class Algorithm{
  // ... Your code here
public:
  // ... Your code here
  void run(){
    // Runs the algorithm
  }
  void print_scores(){
    // Prints out the scores (in a nice format)
  }
  // More methods...
}
```

Note that:

- Having run your algorithm, you want to check whether it converged or whether it reached the maximum number of iterations, M.
- It would be useful to reuse the `matrix` class we used many times in the labs.
- You should NOT use any complex C++ object or class that we have not seen in class. For example, if you use std::vector you will lose points. But see the hints below.
- You should make sure that your code does not have memory leaks.

Additional hints

These could help you to write your solution:

- In terms of libraries, it should be sufficient to include:

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
```

- You can compute the square root of a double using `sqrt` and its absolute value using `abs` as in:

  ```
  double x = 4.0;
  sqrt(x);
  abs(x);
  ```

  where `sqrt` and `abs` are defined in `math.h`.

Marking criteria

The marking scheme is indicative and is intended only as a guide to the relative weighting of the different parts of the project. So the final mark is roughly determined by:

1. Submitting correct code (10%):
   - Submitting a single `.cpp` file with the **correct name**. E.g., if your email is `sn10022@bristol.ac.uk` your should submit a file named `sn10022.cpp`.
   - The code compiles and runs **without errors, crashes or infinite loops**. The code will be tested with `g++` from the lab pack.
2. Writing a correct implementation of the algorithm above (40%):
   - Your code should **print out the impact and knowledge scores of each paper** using the matrix A given above.
   - You should **check whether the algorithm converged**.
3. Good algorithm design (30%):
   - Using OOP (encapsulation, constructors, destructors, memory management, etc).
4. Good coding practice (20%):
   - Appropriate comments, variable names and code formatting.

Plagiarism policy

While you can discuss the general strategy of your code with your classmates, you must code independently. Code sharing and/or co-developing will be treated as academic collusion. This is a serious offence, see here.

You should be able to write the code for this project using only the functions and libraries that we have used so far in the course. See also the hints above. The use of code found on the internet is discouraged. But, if you use do use code found on the web or elsewhere, it should be limited to a very small section of your code and you will need to disclose its the source in your comments. Otherwise, it will be regarded as plagiarism.