Linear Algebra

Due at 4:00pm on Monday 7 October 2019

What you need to get

- `YOU_a2-q1q2q3.ipynb`: Jupyter notebook for Q1, Q2, and Q3

If you want to typeset your solutions to Q4, Q5 and Q6 in LaTeX, then you're welcome to use:

- **https://www.overleaf.com/read/vztpkwrpmtcn**: (optional) Overleaf template

What to do

Google Page Rank

The objective of this task is to develop Python code that computes the Page Rank of a set of web pages based on the network adjacency graph. The adjacency graph is represented by a matrix G, where

$$G_{ij} = \begin{cases} 1 & \text{if } \exists \text{ a link from } j \text{ to } i \\ 0 & \text{otherwise} \end{cases}$$

1. **[2 marks]** Complete the Python function

   ```
   y = SparseMatMult(G, x)
   ```

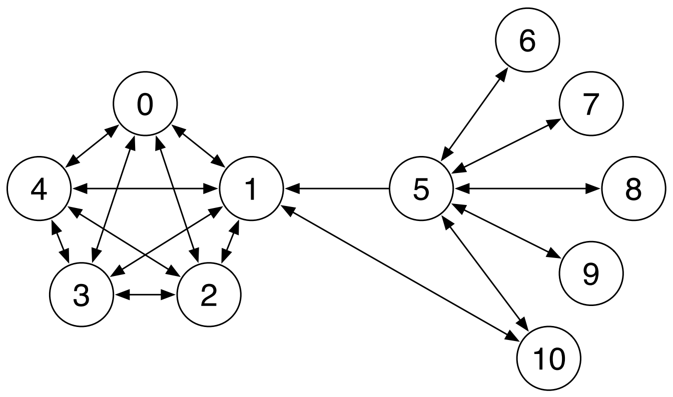
   which multiplies the sparse matrix `G` by the vector `x`. The matrix is stored using the *dictionary-of-keys* method. You can get the key-value pairs using the built-in call `G.nonzero()`. See the function's documentation for full details.

2. **[5 marks]** Complete the Python function

   ```
   p, iters = PageRank(G, alpha)
   ```

   that finds the steady-state solution (eigenvector for $\lambda = 1$) of the Page Rank problem using the iterative method discussed in class. The output `p` is an $R \times 1$ vector (where R is the number of nodes) containing the node scores, and `iters` is the number of iterations the method took to converge. Your method should use your implementation of `SparseMatMult` from question 1, and must **not** form a full $R \times R$ matrix at any time (even as an intermediate while evaluating an expression). To input the initial adjacency matrix, you can use `dok_matrix` from the `scipy.sparse` module (see class demonstration `Randy_demo`). Your iterations should start with a uniform distribution in `p`, and terminate once the solution is found to within a tolerance of 10^{-8} (ie. none of the elements in `p` changes by more than the tolerance).

3. **[5 marks]** Consider the small web shown in the figure below, where each bi-directional arrow indicates **two** links, one in each direction.



(a) Edit the notebook to create the corresponding sparse matrix G. Again, you should create a `dok_matrix` using the `scipy.sparse` module.

(b) Run your `PageRank` function on the network with an α-value of 0.85. Create a stem plot of the node scores.

(c) Run `PageRank` for α-values in the range $[0, 1]$, in increments of 0.05. Plot the number of iterations it takes to converge over that range of α values (remember to label your axes). What do you notice about the relationship between α and the number of iterations?

(d) Create a stem plot of the node scores for $\alpha = 0.05$, and another plot for $\alpha = 0.95$? (Do I need to mention axis labels again?) How does a low α value affect the node scores?

LU Factorization

4. **[6 marks]** Consider the system of equations:

$$\begin{aligned} 12x_1 + 12x_2 - 24x_3 - 48x_4 &= -60 \\ 4x_1 + 2x_3 + 8x_4 &= -4 \\ 6x_1 + 4x_2 - 11x_3 - 16x_4 &= -30 \\ -4x_1 - 12x_2 + 20x_3 + 40x_4 &= 36 \end{aligned}$$

(a) Write the coefficient (system) matrix A and the right-hand-side vector b so that $Ax = b$.

(b) By manual calculation (showing your work), compute the LU factorization (with row pivoting) of the system matrix in part (a). That is, find a permutation matrix P, a unit-diagonal, lower-triangular matrix L, and an upper-triangular matrix U such that $PA = LU$.

(c) Solve the system manually using the LU factorization above. Show your work.

5. **[7 marks]** Consider the linear system of equations

$$\begin{bmatrix} 2 & 3 & 1 \\ 6 & 8.9999 & 6 \\ 4 & 8 & 11 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ -2 \end{bmatrix}. \tag{1}$$

(a) Evaluate an accurate solution to (1). You may use Python's `numpy.linalg.solve` function. You do not have to show how you got the solution, but write the solution in your answer, and call it $\mathbf{x}^{(\text{true})}$.

(b) Simulating 5-significant-digit rounding by hand, solve (1) using Gaussian elimination **without row pivoting**, followed by back substitution. Call your answer $\mathbf{x}^{(\text{no-pivot})}$. Show your work.

(c) Simulating 5-significant-digit rounding by hand, solve (1) using Gaussian elimination **with row pivoting**, followed by back substitution. Call your answer $\mathbf{x}^{(\text{pivot})}$. Show your work.

(d) Compute the relative error for each of $\mathbf{x}^{(\text{pivot})}$ and $\mathbf{x}^{(\text{no-pivot})}$ using the L^2 (Euclidean) norm, and $\mathbf{x}^{(\text{true})}$ as the true solution.

6. **[3 marks]** Suppose A and B are both $N \times N$ nonsingular matrices. Give an algorithm to solve the $N \times N$ linear system $ABx = y$ for x in $\mathcal{O}(N^2)$ flops, given the LU factorizations $L_A U_A = P_A A$ and $L_B U_B = P_B B$.

   In your algorithm, you may use matrix-vector type products, as well as the functions:

   ForwardSub(L, y): Solves $Lx = y$ for x using the forward substitution method, and returns the solution vector.

   BackwardSub(U, y): Solves $Ux = y$ for x using the backward substitution method, and returns the solution vector.

What to submit

You must submit a series of PDF documents to Crowdmark. For each coding question, export your jupyter notebook to PDF. When a proof or manual calculation is requested, you can typeset your solution (eg. using LaTeX or Word), or write your solution by hand. In either case, your solution should also be submitted as PDF. **If you submit photos of handwritten work, it is your responsibility to ensure that they are legible.**

Submit the following PDF documents:

1. A **single** PDF of the notebook containing solutions for all of Q1, Q2, and Q3
2. A PDF for Q4
3. A PDF for Q5
4. A PDF for Q6