April 2023

Assignment 4: Singular value decomposition

It's hard to do SVD with hand calculations, so there is no paper-and-pen calculation in this assignment.

1. The matrix below shows 5 viewer ratings and their preferred movies (no movie names, just genres are specified). Their ratings are values between 0 (worst) and 5 (best). The values are encoded in the matrix A. Each row represents a viewer and each column a movie.

$$A = \begin{array}{c} \begin{matrix} \text{Drama 1} & \text{Drama 2} & \text{SciFy} & \text{Documentary} \end{matrix} \\ \begin{pmatrix} 5 & 5 & 0 & 4 \\ 1 & 1 & 5 & 0 \\ 3 & 2 & 0 & 4 \\ 5 & 3 & 0 & 5 \\ 0 & 0 & 4 & 0 \end{pmatrix} \end{array} \begin{matrix} \\ \text{Ali} \\ \text{Beatrix} \\ \text{Elsa} \\ \text{Johan} \\ \text{Chandra} \end{matrix}$$

   Factoring A using the SVD offers a way to capture the relationships of how people rate movies and see if there are similarities in how people like certain movies. The matrices U and V in the decomposition can be interpreted as stereotypical viewers and stereotypical movies, and the singular values how strong the grouping are between the viewers and movies (for example how significantly strong a group of viewers like dramas **and** Scify).

   The data matrix size here is, as you can understand, not realistic. The meaning with this exercise is to be able to study the result, and to make sense of the SVD, which is hard to do with a realistic size matrix (you can't look at the result). However, the ideas can easily be transferred to large data size.

   a. Use Python (or Matlab) to compute SVD of the matrix A. You must include the SVD in the report.

   b. The largest singular values give us information about stereotypical groups/types of viewers. What types can you find if you look at $Av_1 = \sigma_1 u_1$ and $Av_2 = \sigma_2 u_2$, respectively? (These are the two strongest groups).

   c. Try to interpret the column space $\boldsymbol{C}(A)$, and the row space $\boldsymbol{C}(A^T)$ here. What dimensions and what does it mean in relation to stereotypical viewers and stereotypical movies, respectively?

2. In this task you will experiment with an algorithm commonly used for finding all eigenvalues and eigenvectors to a matrix (and as a consequence, also for finding singular values and singular vectors). The algorithm will be explained more in detail

on a lecture.
Write a function in Python (or Matlab) that take a square matrix B as input, and output B (which is overwritten in the algorithm) and Q. The function should perform the following algorithm:

Let $Q = I$ (the identity)
while not_convergence
  QR-factorize current B => Q_{temp}, R (you can use built-in QR-factorization)
  Update B: $B = R \cdot Q_{temp}$
  Update Q: $Q = Q \cdot Q_{temp}$
  print B (display on the screen)
  Choose if convergence or not, based on what B looks like
end

The idea is that you run the loop until you think B looks "good enough", i.e. when you can see the eigenvalues in B (let's say with a couple of correct decimals). You can implement the "choose if convergence"-line in the while-loop in different ways. A simple way is to input 'y' (yes) if you want to continue and 'n' (no) if you want to stop.

a. Choose a symmetric matrix, for example

$$B = \begin{pmatrix} 3 & 1 & 3 & 1 \\ 1 & 5 & 3 & 1 \\ 3 & 3 & 6 & 3 \\ 1 & 1 & 3 & 2 \end{pmatrix}$$

First, use the built-in function in Python to compute the eigenvalues and eigenvectors. "Save" them somewhere, so that you easily can compare with your results.
Run your program until you can see the eigenvalues. What seems to happen gradually when you run your algorithm?

b. Repeat the same with non-symmetric matrix, for example

$$B = \begin{pmatrix} 2 & 1 & 0 & 0 \\ 2 & 3 & 1 & 1 \\ 2 & 2 & 1 & 1 \\ 1 & 2 & 2 & 5 \end{pmatrix}$$

What did the end-result look like here?
(What you hopefully see here is the so-called Schur decomposition. You can look it up, i.e. in Wikipedia, if you haven't heard of it in linear algebra before).

c. Can you use your function to find SVD to the matrix A in task 1, without explicitly forming A^TA and AA^T? If so, try it out!
Switch off the last two lines in the algorithm ("print B" and "choose if convergence…"), and let the loop run for example 100 iterations. Try to see what kind of structure you get in the result.