Matrices and Statistics with Applications
Computer Exercise: SVD, PCA, Generalized inverses

SeSE, LU, SLU, UU

March, 2016

The tasks within this Lab are related to SVD, PCA and Pseudoinverses. You are welcome to extend some of the exercises with additional tasks, relevant to the theme of this lab.
At the end, the results of the exercise should be sketched and shown to the lab-consultant.
The tasks require programs and data files, which can be downloaded from
`http://user.it.uu.se/~maya/Courses/NGSSC/index_Stat.html`.
Make a copy of these files in some of your directories.

Exercise 1 (Easy, optional)
Download the files `yeastarray_t2.txt` and `Gene_test.m`.
You have at your disposal the test for the Gene analysis data. You could reproduce the tests and try to analyse the results from the SVD decomposition.

Exercise 2 (Large data sets and covariance matrices)
Load the file `formaya.mat`. It contains four files

```
 Name            Size    Bytes     Class     Attributes

  covmatrix    2500x2500   50000000  double
  covmatrix2   2500x2500   50000000  double
  cutout         50x50x50766   253830000  int16
  cutout2        50x50x50766   253830000  int16
```

We will work only with `covmatrix2` and `cutout2`.

Tasks:

1. Consider `covmatrix2`. Find all its eigenvalues and study them.
   Explain the very small eigenvalues, some of which are even negative.
   How many dominant components are observed? Can you illustrate them graphically?

2. The number of nonzero components in the matrix is 5621641 and the number of entries is $2500^2 = 6250000$, thus, the matrix is full. (Note that there are negative entries in the covariance matrix.)

   A study of the size of the entries shows that some of them are very small by absolute value.

   (a) What would be the impact on the PCA if we delete some of the entries, say, that are by absolute value less than 0.005 or 0.05? (This effect can be achieved during the computation of the covariance matrix - we can neglect some small entries upon computation.)

(b) What properties of the matrix have to be preserved?

3. Consider the data set `cutout2`. Note that it is provided in single precision and if we want to perform algebraic operations with it, we have to convert it to double.

   (a) Estimate the computer resources needed to compute the SVD on the matrix? Note that each cut has very few nonzero elements and the total number of nonzero elements is not $50 * 50 * 50766 = 126915000$ but 26392058.

   (b) Reshape the data as (50766x2500) and (try to) compute its SVD. In Matlab try to create the matrix as sparse.

   (c) Do a performance study: check the time needed to compute SVD on parts of the reshaped data, say 2500x2500, 5000x2500 etc. What is the increase factor in computing time?

Exercise 3 (Variable selection)
Consider the data set `Oxigen.dat`. The data contains measurements, made on men involved in a physical fitness course at North Carolina State University. The variables are
Age (years), Weight (kg), Oxygen intake rate (ml per kg body weight per minute), time to run 1.5 miles (minutes), heart rate while resting, heart rate while running (same time Oxygen rate measured), and maximum heart rate recorded while running.
Aerobic fitness (measured by the ability to consume oxygen) should be predicted, in this case, from six background variables. The goal is to develop an equation to predict fitness based on the exercise tests rather than on expensive and cumbersome oxygen consumption measurements.

1. Perform variable selection in order to find the best choice of variables that describe the data in a best way. You should run several variants and then come up with some kind of conclusion.

2. How would you organize the numerical computations? Note that this is a very small example of a problem for variable selection. In practice we would have 20000 observations and 10000 background variables of which some are maybe strongly related. Thus, all tests on variable selection must be done in a computationally very efficient way.

Exercise 4 (Generalized inverses)
The task with this exercise is to compare two methods to compute the generalized inverse A^+ and discuss computational cost, measured by time and accuracy of the computed inverse.
Let $A \in \mathbb{R}^{m,n}$. We pose the question to compute A^+.
Method 1: via SVD
Compute the thin SVD of A, $A = U\Sigma V^T$. Then $A^+ = V\sigma^{-1}U^T$.
Method 2: via iterations

Theorem 1 (See ref. [1]) *Let the real scalar α satisfy*

$$0 < \alpha < \frac{2}{\lambda_{max}(A^*A)}.$$

Then the sequence $\{Y_k\}$, obtained as

$$Y_0 = \alpha A^*$$
$$Y_{k+1} = Y_k(2 * I - A * Y_k)$$

converges to A^+ when $k \to \infty$.

The iterative procedure depends on a parameter, α, that is related to the maximum eigenvalue of A^*A. As A^*A is symmetric and positive definite, all its eigenvalues are real and positive.
For not very large matrices, it can be viable to form A^*A explicitly and then compute its maximum eigenvalue. If A is large, the latter option would be too expensive, both in terms of time and memory. Therefore, one would prefer to compute an estimate (and upper bound) of $\lambda_{max}(A^*A)$ and use that to compute α. To this end, we recall Gershgorin's theorem, that states that the largest absolute row-sum (or column-sum) of a symmetric matrix is an upper bound of its largest eigenvalue.

Tasks:
Write a function in Matlab or R, that performs the following:

1. Computes the generalized inverse using SVD. Time the computation.
2. Implements the iterative algorithm, computing explicitly both A^*A and using the exact $\lambda_{max}(A^*A)$. Store the preparation time (to compute α, the time needed for the iterative method to converge and the number of iterations. As a stopping tolerance use $norm(Y_{k+1} - Y_k) \leq 10^{-8}$.
3. Implements the iterative algorithm, computing an approximation of $\lambda_{max}(A^*A)$ without explicitly forming A^*A. Store the preparation time (to compute α, the time needed for the iterative method to converge and the iterations. Use the same stopping tolerance.
4. Output: the three generalized inverses, the times to compute them and the iterations in the latter two cases.

Run the code with the following matrices:
- $A1 = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{bmatrix}$
- the data matrix from `yeastarray_t2.txt`
- $A = randn(1000, 250)$

Explain the results. Which is the fastest? Will the conclusion remain the same for larger matrices? How sensitive is the iterative method with respect to α?

References

[1] Adi Ben-Israel and Dan Cohen, On Iterative Computation of Generalized Inverses and Associated Projections *SIAM Journal on Numerical Analysis*, 3 (1966), 410-419.