universität ulm Universität Ulm

Timo Tonn
Tobias Born
Summer term 2023
Points: 40
Submit until: 12.05.2023

Exercise Sheet 3 to NMDS

1. [**Complete Golub-Kahan-Reinsch SVD algorithm**, 4+4+8+2+2 points]

Finally, the time has come and we will put our work from the previous two sheets together and will obtain the Golub-Kahan-Reinsch SVD algorithm. The rough procedure is clear:

Input: $A \in \mathbb{R}^{m\times n}$ with $m \geq n$
$B_0 = \text{bidiagonalization}(A)$
for $i = 1, 2, \ldots$ **do**
  $B_i = \text{golub\_kahan\_svd\_step}(B_{i-1})$
end for

Unfortunately, we still face two problems:

- On sheet 2 we learned that the *implicit Q-theorem* ensures that one iteration of the GKR-algorithm equals one iteration of the QR-algorithm up to signs. However, this theorem requires the irreducibility of the bidiagonal matrix, what means that all bidiagonal entries and all diagonal entries, except for the one at the very bottom right, are not allowed to be zero. This is not yet taken into account.
- What do we choose as stopping criterion?

Therefore, we divide the bidiagonal matrix into three submatrices in every iteration.

$$B = \begin{pmatrix} B_{11} & & \\ & B_{22} & \\ & & B_{33} \\ p & n-p-q & q \end{pmatrix} \begin{matrix} p \\ n-p-q \\ q \end{matrix} \tag{1}$$

B_{33} is diagonal, B_{22} bidiagonal with a non-zero superdiagonal and B_{11} the remaining part of B. B_{33} already contains q singular values and does not have to be processed any more. B_{22} is irreducible (is it?) and therefore the next run of *golub_kahan_svd_step(·)* will be applied to it. B_{11} waits for B_{22} to be diagonalized. To get this structure, the following (implicit stopping criterion) is applied to the bidiagonal matrix at the beginning of each iteration:

$$\text{if } |B(i,i{+}1)| \leq \varepsilon(|B(i,i)| + |B(i{+}1,i{+}1)|) \text{ then set } B(i,i+1) = 0,\ i = 1,\ldots,n{-}1$$

If we repeat this until $q = n$, then we should be done. But unfortunately something is still missing. As indicated above, B_{22} is not necessarily irreducible. We ensured that all bidiagonal entries are non-zero, but not that all diagonal entries (except for the one at the very bottom right) are zero. So we need a procedure for the case that

$$B_{22} = \begin{pmatrix} d_1 & f_1 & & & & \\ & d_2 & f_2 & & & \\ & & \ddots & \ddots & & \\ & & & 0 & f_j & \\ & & & & \ddots & \ddots \\ & & & & & d_n \end{pmatrix}$$

with $1 \leq j \leq n{-}1$. What one does is to apply a sequence of Givens rotations such that the j-th row of B_{22} is zero. Then B_{22} decouples into two bidiagonal matrices. By redetermining p and q, one gets

a new B_{22} which is irreducible (the lower part of the old B_{22}). The upper part of the old B_{22} will then be included in the new B_{11}. The sequence of Givens rotations pursues the following pattern:

$$B = \begin{pmatrix} + & + & & & \\ & 0 & \star & & \\ & & + & + & \\ & & & + & + \\ & & & & + \end{pmatrix} \xrightarrow{G^T_{j+1,j}} \begin{pmatrix} + & + & & & \\ & 0 & 0 & \star & \\ & & + & + & \\ & & & + & + \\ & & & & + \end{pmatrix} \xrightarrow{G^T_{j+2,j}} \begin{pmatrix} + & + & & & \\ & 0 & 0 & 0 & \star \\ & & + & + & \\ & & & + & + \\ & & & & + \end{pmatrix}$$

$$\xrightarrow{G^T_{j+3,j}} \begin{pmatrix} + & + & & & \\ & 0 & 0 & 0 & 0 \\ & & + & + & \\ & & & + & + \\ & & & & + \end{pmatrix} = \left(\begin{array}{cc:ccc} + & + & & & \\ & 0 & & & \\ \hdashline & & + & + & \\ & & & + & + \\ & & & & + \end{array}\right)$$

One wants to chase the non-zero element along the j-th row until it vanishes. What is the only possibility to do this without introducing a second non-zero element? It is the Givens rotation $G^T_{j+1,j}$ which zeroes $B(j, j+1)$, gives $B(j, j+2)$ a non-zero value and changes the value of $B(j+1, j+1)$ and $B(j+1, j+2)$. So the non-zero element moves from $B(j, j+1)$ to $B(j, j+2)$, but no further non-zero element is introduced. Of course, one could also choose any $G^T_{i\neq j,j}$ to zero $B(j, j+1)$. But then always a second non-zero element would be introduced. Think it through!
After applying the first rotation, one wants to zero $B(j, j+2)$. With the same logic as in the first step, this only can be done using $G^T_{j+2,j}$. Continuing like this, $G^T_{n,j}$ will finally yield a bidiagonal matrix with a zero row j.

a) Create a MATLAB program *[B] = zero_row(B,j)* which deals with the problem described directly above (square bidiagonal matrix, all bidiagonal elements are non-zero, one diagonal entry is zero). Use your programs for Givens rotations from the previous exercise sheet.

b) Create a MATLAB program *[p,q] = determine_indexes(B)* which calculates p and q as described in (1). Make sure that your program works for all possible indexes p and q.

c) Finally, implement the whole Golub-Kahan-Reinsch algorithm for an arbitrary matrix $A \in \mathbb{R}^{m\times n}$ in a MATLAB program *[S] = gkr_algorithm(A,tol)*. Use all necessary programs you developed in the last three exercise sheets. Your program only has to compute a diagonal matrix containing the singular values, you can omit the orthogonal matrices U and V. If you need an overview of the algorithm, see page 30 in the lecture notes.
Hint: At the point where you check if an entry of B_{22} is zero, you better check if there are very small entries instead of zero entries (floating point numbers rarely become exactly zero).

d) Test your program appropriately.

e) Make a comparison of the runtime of your program and the MATLAB command *svd(·)*. Consider several cases. You can use the commands *tic* and *toc*.

Remark 1. *This is a potentially infinite iteration. If one chooses the shifts as we did on sheet 2 (it is called Wilkinson-Shift) and uses an appropriate criterion to consider diagonal and bidiagonal elements as zero, then heuristically it turns out that usually every singular value needs two or three iterations to converge. So usually* $2n$ *to* $3n$ *iterations are necessary.*
Recall that the matrices of the singular vectors U *and* V *can be obtained by multiplying all orthogonal matrices that are applied in the bidiagonalization and the iteration. As you can imagine, this is a big effort.*

2. [**Application: least-squares fitting**, 3+3+3+3 points]

Besides image compression we now come to a second application example of the SVD. Consider an overdetermined linear system of equations:

$$Ax = b \quad \text{with } A \in \mathbb{R}^{m\times n},\ x \in \mathbb{R}^n,\ b \in \mathbb{R}^m \quad \text{and} \quad m > n\ . \tag{2}$$

Of course, we cannot solve this system, but we can compute

$$x \ = \ \underset{x\in\mathbb{R}^n}{\operatorname{argmin}} \|Ax - b\|_2\ . \tag{3}$$

This can be done in many ways, one of them is to use a so-called *Moore-Penrose pseudoinverse*. Every real matrix $A \in \mathbb{R}^{m\times n}$ has a unique Moore-Penrose pseudoinverse $A^+ \in \mathbb{R}^{n\times m}$ which is defined by four conditions (if you are interested, look them up!). It is a generalization of an inverse and for a regular $B \in \mathbb{R}^{n\times n}$ it holds that $B^+ = B^{-1}$. One can show that (3) is solved by

$$\bar{x} \;=\; A^+ b \;.$$

For motivation, note that in case of a regular $B \in \mathbb{R}^{n\times n}$ one gets that

$$\bar{x} = B^{-1}b \;\text{ and }\; Bx = BB^{-1}b = b \;.$$

Fortunately for us, the pseudoinverse of $A = U\Sigma V^T$ is

$$A^+ \;=\; V\Sigma^+ U^T$$

and if A has full rank (in the sense that all singular values are bigger than zero) it holds that

$$\Sigma^+ = \begin{pmatrix} \frac{1}{\sigma_1} & & & \vdots \\ & \ddots & & 0 \\ & & \frac{1}{\sigma_n} & \vdots \end{pmatrix} \qquad (\text{for } m > n) \;.$$

a) Consider random data points in $\mathbb{R}^2$, i.e. $(a_1, b_1), \ldots, (a_m, b_m) \in \mathbb{R}^2$ with $m > 2$. We want to calculate a simple linear regression, so find $x_1, x_2 \in \mathbb{R}$ such that the affine function

$$y(a) \;=\; x_1 a + x_2$$

minimizes

$$\sum_{i=1}^{m} |y(a_i) - b_i|^2 \;.$$

Put this problem in the form of (2), i.e. formulate the suitable A and b.

b) Realize this in MATLAB. Create random data points using *create_data_points_regression.m*. Then compute the affine function y and plot the data points and y.
Hint: You can of course use MATLAB's *svd(·)*.

c) Now, we again have random data points in $\mathbb{R}^2$, but this time we want to approximate them by a circle. Find the corresponding A and b.
Hint: Note that by Pythagoras it holds that $(a_i - x_m)^2 + (b_i - y_m)^2 = r^2$ for a circle with radius r and centre (x_m, y_m). Multiply this out and notice that you have three unknowns.

d) Also realize this in MATLAB and plot the data points and the resulting circle.
Use *create_data_points_circle.m*.

3. [**Error of best approximation in Frobenius norm**, 8 points]

Finally, we come back to the best rank-k-approximation A_k. You already know that

$$A_k \;=\; \underset{\substack{B \in \mathbb{R}^{m\times n} \\ \operatorname{rank}(B) \le k}}{\operatorname{argmin}} \|A - B\|_2 \;=\; \underset{\substack{B \in \mathbb{R}^{m\times n} \\ \operatorname{rank}(B) \le k}}{\operatorname{argmin}} \|A - B\|_F$$

and that

$$\|A - A_k\|_2 = \sigma_{k+1} \;.$$

What is still missing is the error in the Frobenius norm.

So let $A \in \mathbb{R}^{m\times n}$ with its rank-k-approximation A_k, $k \le \operatorname{rank}(A)$. Prove that

$$\|A - A_k\|_F \;=\; \sqrt{\sum_{i=k+1}^{\min(m,n)} \sigma_i^2} \;.$$