# Programmeren (FEB22012)

## 6. Exercise

Deadline for submission: 2011-09-25 23:59 CET

This exercise is to be done **in pairs**. Submit your solution in a standard way through BB, but now *only* with the account of one of the pair members. Include in the source files the names and student numbers of *both* of the pair. If you have problems finding a pair or suffer from fear of social contact, you are also allowed to make the exercise individually.

## Multidimensional scaling

Data analysts typically face the problem of making sense of their data at hand, yet many statistical techniques are available to them in order to alleviate this issue. One of these techniques, Multidimensional Scaling (MDS), is typically applied in order to visualize data in such a way that it allows for exploring similarities and dissimilarities between items in the data set.

When applying MDS, items are visualized in a low-dimensional space in such a way that the distances between the items represent the associated dissimilarities as closely as possible. The extent to which distances represent dissimilarities is typically reflected in the solution's (normalized) stress value, which should be minimized.

A well-known algorithm for finding the optimal coordinates is the Scaling by MAjorizing a COmplicated Function (SMACOF) approach. The main algorithm for the SMACOF approach is detailed in pseudocode in Algorithm 1.

In this exercise, you will be performing MDS in MATLAB on sales data by means of the SMACOF approach as well as by using some built-in MATLAB functions. The data is available from `http://smaa.fi/tommi/courses/prog2/data/clothes.xls`. This data set contains a table with the number of times any pair of 2 out of 12 clothing items has been bought in one purchase.

***Note:*** the SMACOF algorithm involves many computations. Implementing these computations by using matrix operations and matrix algebra is *crucial* for execution speed and for successful completion of this exercise.

**Algorithm 1**: High-level pseudocode for the SMACOF algoritm.

```
initialize iteration at 0
initialize previous stress at ∞
calculate stress and distances for initial random coordinates
visualize initial random coordinates
while (previous stress - stress) > threshold do
   update iteration
   update previous stress
   update coordinates
   calculate stress and distances for updated coordinates
   produce error if stress goes up
   visualize coordinates
end
```

# Exercise

1. In this exercise, we will use the `clothes.xls` data set. Our goal is to explore similarities and dissimilarities between items in this data set. MDS is a very helpful technique here. However, in order to be able to apply this statistical technique to our data set, we first need to transform the co-occurrence frequencies in our data set into dissimilarities. Write a function that applies the so-called gravity model to convert co-occurrence frequencies into dissimilarities.

   (a) The input of this function should be an $n \times n$ matrix $\mathbf{F}$ with co-occurrence frequencies of $n$ items.

   (b) This function should produce an $n \times n$ matrix $\mathbf{\Delta}$ with dissimilarities of $n$ items as output.

   (c) According to the gravity model,

$$\mathbf{\Delta}_{ij} = \sqrt{\frac{\mathbf{F}_{ii}\mathbf{F}_{jj}}{\mathbf{F}_{ij}}}, \qquad \forall i, j \in \{1, \ldots, n\}\,. \tag{1}$$

2. Create a function that applies the SMACOF algorithm for MDS in order to assign items coordinates such that the distances between the items reflect their dissimilarities as closely as possible.

   (a) Your function should take as input:

   - An $n \times n$ matrix $\mathbf{\Delta}$ with dissimilarities of $n$ items.
   - Descriptions of all $n$ items.
   - An $n \times p$ matrix $\mathbf{X}$ with initial coordinates for $n$ items in $p$ dimensions.
   - A threshold $\epsilon$ for the algorithm's accuracy.
   - A binary variable for toggling the visualization on and off.

(b) The function should produce as output:

- An $n \times p$ matrix $\mathbf{X}$ with the optimized coordinates for $n$ items in $p$ dimensions.
- The total stress value $\sigma$ of coordinates of all $n$ items.

(c) The contents of this function should reflect Algorithm 1. The function should make use of three other functions, i.e., one function for calculating the stress of a set of coordinates, another function for updating the coordinates, and finally a function for visualizing a set of coordinates. We will implement these functions in the next steps.

(d) If $\sigma_{i-1} - \sigma_i > -\epsilon$ at iteration $i$, something must be wrong in your SMACOF implementation, which should result in your function to produce an error message. You can use the MATLAB function `error` to do this.

3. Write a function that computes the stress of a set of coordinates with respect to the dissimilarities of the associated items.

(a) The inputs of this function should be:

- An $n \times n$ matrix $\mathbf{\Delta}$ with dissimilarities of $n$ items.
- An $n \times p$ matrix $\mathbf{X}$ with coordinates for $n$ items in $p$ dimensions.

(b) The outputs of this function should be:

- The total stress value $\sigma$ of coordinates of all $n$ items.
- An $n \times n$ matrix $\mathbf{D}$ with the Euclidean distances of $n$ items to one another in $p$ dimensions.

(c) This function should first compute the Euclidean distances between all $n$ items in $p$ dimensions, given their coordinates $\mathbf{X}$. Then, the stress $\sigma$ of these distances $\mathbf{D}$ with respect to their associated dissimilarities $\mathbf{\Delta}$ can be computed as

$$\sigma = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} \left(\mathbf{\Delta}_{ij} - \mathbf{D}_{ij}\right)^2}{\sum_{i=1}^{n} \sum_{j=1}^{n} \mathbf{\Delta}_{ij}^2} \tag{2}$$

(d) ***Hint:*** remember from last week that given an $n \times p$ matrix $\mathbf{A}$ with $n$ coordinates in $p$-dimensional space and an $m \times p$ matrix $\mathbf{B}$ with $m$ $p$-dimensional coordinates, the $n \times m$ matrix of their Euclidean distances $\mathbf{D}$ can be computed as

$$\mathbf{D} = \sqrt{\mathbf{x}\mathbf{1}_m + \mathbf{1}_n\mathbf{y}' - 2\mathbf{A}\mathbf{B}'}, \tag{3}$$

with $\mathbf{x}$ an $n \times 1$ vector where the elements represent the sum of squared coordinates of $\mathbf{A}$, $\mathbf{y}$ an $m \times 1$ vector with the sum of squared coordinates of $\mathbf{B}$, $\mathbf{1}_n$ an $n \times 1$ vector of ones, and $\mathbf{1}_m$ a $1 \times m$ vector of ones.

4. Create a function that updates a set of coordinates of items by exploiting the difference between their dissimilarities and their distances.

   (a) This function should take as input:
   - An $n \times p$ matrix $\mathbf{X}$ with coordinates for $n$ items in $p$ dimensions.
   - An $n \times n$ matrix $\mathbf{\Delta}$ with dissimilarities of $n$ items.
   - An $n \times n$ matrix $\mathbf{D}$ with the Euclidean distances of $n$ items to one another in $p$ dimensions.

   (b) Your function should produce as output an updated $n \times p$ matrix $\mathbf{X}$ with coordinates for $n$ items in $p$ dimensions.

   (c) This function should first construct an $n \times n$ matrix $\mathbf{Z}$ with

$$\mathbf{Z}_{ij} = \begin{cases} \frac{-\mathbf{\Delta}_{ij}}{\mathbf{D}_{ij}} & \text{if } i \neq j \text{ and } \mathbf{D}_{ij} > 0, \\ 0 & \text{if } i \neq j \text{ and } \mathbf{D}_{ij} = 0, \end{cases} \tag{4}$$

   for $i, j \in \{1, \ldots, n\}$. The elements on the diagonal of $\mathbf{Z}$ should then be computed as the negated sum of the elements on the rows, i.e.,

$$\mathbf{Z}_{ii} = -\sum_{j=1, j \neq i}^{n} \mathbf{Z}_{ij}, \qquad \forall i \in \{1, \ldots, n\}. \tag{5}$$

   Finally, using $\mathbf{Z}$, the coordinates $\mathbf{X}$ can be updated as

$$\mathbf{X} = n^{-1}\mathbf{Z}\mathbf{X}. \tag{6}$$

5. Create a function that can be used to visualize up to the first three dimensions of a set of coordinates of items over time.

   (a) Depending on your implementation, inputs for this function may be:
   - An $n \times p$ matrix $\mathbf{X}$ with coordinates for $n$ items in $p$ dimensions.
   - Descriptions of all $n$ items.
   - A title for your plot.

   (b) The visualization of an iteration of the SMACOF algorithm may look like the snapshot presented in Figure 1. At any rate, include labels in your plot, use square axes, and incorporate the stress value in your visualization.

   (c) ***Hint:*** in case the number of dimensions $p = 1$, you can plot the items by using the `plot` function in MATLAB, with the coordinates for the second dimension set to 0. In case $p = 2$, you can obviously use the `plot` function as well. In case $p \geq 3$, you may use the `plot3` function in order to plot the first three dimensions.

   (d) ***Hint:*** you can add labels to your plot by using MATLAB's `text` function.

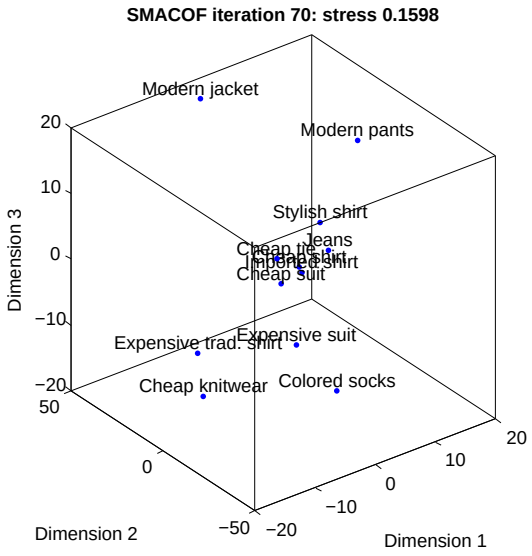


Figure 1: Example of a visualization of an iteration of the SMACOF algorithm.

(e) ***Hint:*** prevent your animation from cluttering and slowing down by clearing the figure before plotting an iteration. You can use the MATLAB command `cla` for this.

(f) ***Hint:*** in order to be able to actually see your animation, you may need to briefly `pause` the animation after plotting an iteration.

6. Write a script that:

(a) Retrieves co-occurrence frequencies and labels from `clothes.xls` by using the MATLAB function `xlsread`.

(b) Transforms the co-occurrence frequencies into dissimilarities by applying the gravity model.

(c) Generates an initial set of random coordinates for all items in 3 dimensions.

(d) Optimizes the initial 3-dimensional coordinates by using your SMACOF function and visualizes each iteration.

(e) Optimizes the initial 3-dimensional coordinates by using MATLAB's built-in `fminunc` function to minimize your stress function. As options for this `fminunc` function, you can set both `'LargeScale'` and `'Display'` to `'off'`, whereas `'TolFun'` can simply be your $\epsilon$ parameter. Visualize the optimized coordinates in a new figure by using your visualization function.

(f) Assesses stress values and execution times for SMACOF and `fminunc` for dimensionalities ranging from 1 to the number of variables in `clothes.xls` (no MDS visualizations needed). Plot stress values per number of dimensions for SMACOF and `fminunc` in one figure. Make another figure that shows the execution times of both algorithms as a function of the dimensionality.

Which number of dimensions yields the best trade-off between representative power (i.e., low stress) and interpretability (i.e., low dimensionality)? Which method is typically faster for performing MDS?