Factorization-Based Data Modeling
Practical Work 1

Umut Şimşekli, Simon Henriet
Télécom ParisTech, Université Paris-Saclay, Paris, France
umut.simsekli@telecom-paristech.fr

Instructions: (please read carefully)

1. This homework can be done in groups of **maximum 2** people.
2. Prepare your report as a pdf file in English by using LaTeXor a similar software (Word etc). Do not submit scanned papers.
3. Put all your files (code and/or report) in a zip file: *surname_name.zip* and submit it to `https://www.dropbox.com/request/xBmWWbEKteyQsnMxjjSI`. The deadline is **11 December 2018**. Late submissions will not be accepted.
4. One submission per group is sufficient.

1 Matrix Factorization with Alternating Least Squares and Gradient Descent

In this section, you will implement the two algorithms that we saw in the class for matrix factorization. The problem that we aim to solve is given as follows:

$$(W^\star, H^\star) = \arg\min_{W,H} \frac{1}{2}\|X - WH\|_F^2, \tag{1}$$

where $X \in \mathbb{R}^{I\times J}$ is the data matrix, and $W \in \mathbb{R}^{I\times K}$ and $H \in \mathbb{R}^{K\times J}$ are the unknown factor matrices. Here $\|A\|_F$ denotes the Frobenuis norm of a matrix A.

The Alternating Least Squares (ALS) algorithm has the following update rules:

$$W = XH^\top(HH^\top)^{-1} \tag{2}$$
$$H = (W^\top W)^{-1}W^\top X. \tag{3}$$

The Gradient Descent (GD) algorithm has the following update rules:

$$W \leftarrow W + \eta(X - WH)H^\top \tag{4}$$
$$H \leftarrow H + \eta W^\top(X - WH), \tag{5}$$

where η is a fixed step-size.

Now use the template code "matrix_factorization_template.m"

1. Implement the ALS update rules.

2. Compute the objective function values for each iteration.
3. Implement the GD update rules.
4. What is the effect of η?
5. Compute the objective function values for each iteration.
6. Play with the variable 'dataNoise'. What is the effect of this parameter on the algorithms.
7. Play with the size of the data (I, J) and the rank of the factorization K. What behavior do you observe?
8. Which algorithm do you think is better? Why?

2 Non-Negative Matrix Factorization with Multiplicative Update Rules

In some cases, the observed matrix will only contain *non-negative* values, i.e. $X \in \mathbb{R}_+^{I \times J}$. As we will see in our next lecture, a 'non-negative' model would be more appropriate for such data. In this question, we will deal with non-negative matrix factorization (NMF). The problem that we aim to solve is given as follows:

$$(W^\star, H^\star) = \underset{W \geq 0, H \geq 0}{\arg\min} \sum_{i=1}^{I} \sum_{j=1}^{J} (x_{ij} \log \frac{x_{ij}}{\hat{x}_{ij}} - x_{ij} + \hat{x}_{ij}), \tag{6}$$

where x_{ij} is an element of $X \in \mathbb{R}_+^{I \times J}$, that is the *non-negative* data matrix, and $W \in \mathbb{R}_+^{I \times K}$ and $H \in \mathbb{R}_+^{K \times J}$ are the unknown *non-negative* factor matrices. We also define $\hat{x}_{ij} = \sum_k w_{ik} h_{kj}$. The cost function that we are minimizing is called the Kullback-Leibler (KL) divergence, which turns out to be more suited for NMF than the Frobenius norm.

One of the most popular algorithms for NMF is called the multiplicative update rules (MUR). The MUR algorithm has the following update rules:

$$W \leftarrow W \circ \frac{(X/\hat{X})H^\top}{OH^\top} \tag{7}$$

$$H \leftarrow H \circ \frac{W^\top (X/\hat{X})}{W^\top O}, \tag{8}$$

where $\circ$ denotes element-wise multiplication and / and $\frac{\cdot}{\cdot}$ denote element-wise division. Here, we also define the matrix O of size $I \times J$, such that each element of O will be equal to 1, i.e. $o_{ij} = 1$ for all i and j.
Now use the template code "non_negative_matrix_factorization_template.m"

1. Implement the MUR update rules.
2. Compute the objective function values for each iteration.
3. Play with the variable 'dataNoise'. What is the effect of this parameter on the algorithms.
4. Play with the size of the data (I, J) and the rank of the factorization K. What behavior do you observe?