Data Mining and Matrices (FSS18)
Assignment 2: Matrix Completion

The archive provided to you contains this assignment description, the dataset, as well as the R code fragments for you to complete. Comments and documentation in the code provide further information.

It suffices to fill out the "holes" that are marked in the code fragments provided to you, but feel free to modify the code to your liking. You need to stick with R though.

Please adhere to the following guidelines in all the assignments. If you do not follow those guidelines, we may grade your solution as a FAIL.

Provide a single ZIP archive with name `dmm18-<assignment number>-<your ILIAS login>.zip`. The archive needs to contain:

- A single PDF report that contains answers to the tasks specified in the assignment, including helpful figures and a high-level description of your approach. Do not simply convert your R code to a PDF! Write a separate document, stay focused and brief. As a guideline, try to stay below 10 pages.
- All the code that you created and used in its original format.

Make sure that your report is self-explanatory and follows standard scientific practice. Use the tasks numbers of the assignments as your section and subsection numbers. Label all figures (and refer to figure labels in your write-up). Include references if you used additional sources or material.

Hand-in your solution via ILIAS until the date specified there. This is a hard deadline.

Preliminaries

The goal of this assignment is to implement and experiment with the matrix completion methods discussed in the lecture. (Don't worry, you will have to implement only the core parts of gradient descent and stochastic gradient descent.) We will provide you with a sample of the points of an image; your goal is to reconstruct the image under various settings. Note that matrix completion is generally not a good method to complete images because it does not account for correlations between neighboring pixels. Since images and their completions can be visually understood, we nevertheless perform matrix completion on an image in this exercise.

The image is given as a text file named `image_sample.mmc` in the Matrix-Market coordinate format (see `http://math.nist.gov/MatrixMarket/formats.html#coord` for details). We also provide an R script named `a02-mc.R`, which contains code to load and visualize this image, as well as other pieces of code that you can use when working through this assignment. Familiarize yourself with the data and the provided code.

R is designed for "vectorized" operations. The completion algorithms that we implement here are not vectorized (i.e., they make use for `for` loops), which in practice means that they will generally be orders of magnitude slower than, say, an equivalent C implementation (R supports calls to C). Our datasets are small so that runtimes will still be acceptable.

To facilitate grading, your document should show the completed matrices visually.

1 Computing the Loss Function

We use the same notation as in the lecture. Let $\boldsymbol{D}$ be an $m \times n$ data matrix and Ω be the indexes of the revealed entries. Entry (i, j) of $\boldsymbol{D}$ is revealed if $(i, j) \in \Omega$, otherwise it is unknown. Denote by $N = |\Omega|$ the number of revealed entries, by N_i the number of revealed entries in row i, and by N_j the number of revealed entries in column j. Denote by r the desired rank of the factorization, and by $\boldsymbol{L}_{m\times r}$ and $\boldsymbol{R}_{r\times n}$ the (current) factor matrices.

In this assignment, we will make use of the following loss function:

$$L(\boldsymbol{L}, \boldsymbol{R}) = \sum_{(i,j)\in\Omega} (d_{ij} - [\boldsymbol{LR}]_{ij})^2 + \frac{\lambda}{2}(\|\boldsymbol{L}\|_F^2 + \|\boldsymbol{R}\|_F^2) \tag{1}$$

$$= \sum_{(i,j)\in\Omega} \left[(d_{ij} - \boldsymbol{l}_i^T \boldsymbol{r}_j)^2 + \frac{\lambda \|\boldsymbol{l}_i\|_2^2}{2N_i} + \frac{\lambda \|\boldsymbol{r}_j\|_2^2}{2N_j} \right] \tag{2}$$

$$= \sum_{(i,j)\in\Omega} \left[\left(d_{ij} - \sum_{k=1}^{r} l_{ik} r_{kj} \right)^2 + \frac{\lambda \sum_{k=1}^{r} l_{ik}^2}{2N_i} + \frac{\lambda \sum_{k=1}^{r} r_{kj}^2}{2N_j} \right] \tag{3}$$

This loss function computes the squared error in reconstructing the revealed entries in $\boldsymbol{D}$, plus an L2 regularization term. Here (1) has the form discussed in the lecture, (2) has a form that we assumed when discussing (stochastic) gradient descent, and (3) has a form that may help you to compute derivatives.

a) Prove that the above equalities hold.

b) Complete the function called `loss` in the provided R script. The script provides an example function call and its output; verify that your implementation is correct.

c) **Optional.** The element-wise product of two conforming matrices is called the *Hadamard product*, denoted $[\boldsymbol{A} \circ \boldsymbol{B}]_{ij} = a_{ij} b_{ij}$. Let $\boldsymbol{X}$ be an $m \times n$ matrix such that $x_{ij} = 1$ if $(i, j) \in \Omega$ and $x_{ij} = 0$ otherwise. Use $\boldsymbol{X}$ and the Hadamard product to write the loss function in matrix terms (i.e., without an explicit summation).

2 Computing the Local Gradients

a) Set

$$L_{ij}(\boldsymbol{l}_i, \boldsymbol{r}_j) = (d_{ij} - \boldsymbol{l}_i^T \boldsymbol{r}_j)^2 + \frac{\lambda \left\|\boldsymbol{l}_i\right\|_2^2}{2N_i} + \frac{\lambda \left\|\boldsymbol{r}_j\right\|_2^2}{2N_j}$$
$$= \left(d_{ij} - \sum_{k=1}^{r} l_{ik} r_{kj}\right)^2 + \frac{\lambda \sum_{k=1}^{r} l_{ik}^2}{2N_i} + \frac{\lambda \sum_{k=1}^{r} r_{kj}^2}{2N_j}$$

as in (2) and (3). Then

$$L(\boldsymbol{L}, \boldsymbol{R}) = \sum_{(i,j)\in\Omega} L_{ij}(\boldsymbol{l}_i, \boldsymbol{r}_j).$$

Compute the gradients w.r.t. to each entry in the factor matrices, i.e., compute

$$\nabla_{l_{ik}} L_{ij}(\boldsymbol{l}_i, \boldsymbol{r}_j),$$
$$\nabla_{r_{kj}} L_{ij}(\boldsymbol{l}_i, \boldsymbol{r}_j).$$

b) Compute the gradients w.r.t. to each row/column in the factor matrices, i.e., compute

$$\nabla_{\boldsymbol{l}_i^T} L_{ij}(\boldsymbol{l}_i, \boldsymbol{r}_j),$$
$$\nabla_{\boldsymbol{r}_j^T} L_{ij}(\boldsymbol{l}_i, \boldsymbol{r}_j).$$

c) Complete the function called `dlossp` in the provided R script. The values of N_i and N_j have already been computed and are accessible via the vectors `Nis` and `Njs`, respectively. As before, the script provides an example function call and its output; verify that your implementation is correct.

d) **Optional.** Use $\boldsymbol{X}$ as defined in Task 2c) and the Hadamard product to write the gradient descent update rule of the loss function L for $\boldsymbol{L}$ in matrix terms (i.e., without an explicit summation). Do the same for the gradient rule for $\boldsymbol{R}$.

3 Implementing Gradient Descent

Most of the implementation of gradient descent is already provided in the R script. Complete the function `gdepoch`: first compute the derivate matrices $\boldsymbol{L}^\nabla$ and $\boldsymbol{R}^\nabla$, then perform a gradient step with the step size provided to the function. The script provides an example function call and its output; verify that your implementation is correct.

Once this function is implemented, you are ready to factorize the provided image. The default choice of parameters in the R script is $r = 10$ and $\lambda = 2$. Run gradient descent by executing the commands provided in the R script. Visualize the result after 5, 15, and 50 epochs and discuss.

4 Implementing Stochastic Gradient Descent

Implement stochastic gradient descent by completing the `sgdepoch` function; here you need to implement only a single SGD step. The script provides an example function call and its output; verify that your implementation is correct.

Factorize the matrix again, this time using stochastic gradient descent (as before, set $r = 10$ and $\lambda = 2$). Compare the result with the result obtained by gradient descent and discuss.

5 Impact of Parameter Choices

Factorize the input matrix with the following choice of parameters (using either gradient descent or stochastic gradient descent):

- $r = 10$, $\lambda = 2$ (mildly regularized),
- $r = 10$, $\lambda = 0$ (unregularized),
- $r = 10$, $\lambda = 20$ (heavily regularized),
- $r = 1$, $\lambda = 0$ (lower rank),
- $r = 20$, $\lambda = 2$ (higher rank).

Compare the resulting completed matrices visually. What is the effect of the various parameters? Discuss.

Optional. Can you envision a technique that chooses the hyperparameters r and λ automatically? Try it to determine a suitable choice.