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

Steffen Schmitz
University of Mannheim
stefschm@mail.uni-mannheim.de

1. COMPUTING THE LOSS FUNCTION

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

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

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

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

This loss function computes the squared error in reconstructing the revealed entries in $\mathbf{D}$, plus an L2 regularization term.

1.a Prove the Equality

Task. Prove that the above equalities hold.

In order to prove the equalities of Equation 1, 2 and 3 hold, we split them into two parts. First, we show that the local loss, $\sum_{(i,j)\in\Omega}(d_{ij}-[\mathbf{LR}]_{ij})^2$, is equivalent to the first part of the second and third equation and afterwards we will show that the reformulations of the L2 regularization, $\frac{\lambda}{2}(\|\mathbf{L}\|_F^2 + \|\mathbf{R}\|_F^2)$ are equivalent.

For the local loss we need to show that

$$(d_{ij} - [\mathbf{LR}]_{ij})^2 = (d_{ij} - \mathbf{l}_i^T \mathbf{r}_j)^2 = \left(d_{ij} - \sum_{k=1}^{r} l_{ik} r_{kj} \right)^2$$

for any $(i,j) \in \Omega$. If this equality holds for an arbitrary pair of revealed entries it holds for the complete sum. By taking the square root, subtracting d_{ij} and multiplying it by -1 we get the following statement

$$[\mathbf{LR}]_{ij} = \mathbf{l}_i^T \mathbf{r}_j = \sum_{k=1}^{r} l_{ik} r_{kj}$$

which is exactly the result of a multiplication for entry (i,j) as defined in the lecture.

Second, the equalities for the L2 regularization must hold. They are shown in Equation 4 and 5.

$$\frac{\lambda}{2}(\|\mathbf{L}\|_F^2 + \|\mathbf{R}\|_F^2) = \sum_{(i,j)\in\Omega} \left(\frac{\lambda\|\mathbf{l}_i\|_2^2}{2N_i} + \frac{\lambda\|\mathbf{r}_j\|_2^2}{2N_j} \right) \quad (4)$$

$$\frac{\lambda}{2}(\|\mathbf{L}\|_F^2 + \|\mathbf{R}\|_F^2) = \sum_{(i,j)\in\Omega} \left(\frac{\lambda\sum_{k=1}^{r} l_{ik}^2}{2N_i} + \frac{\lambda\sum_{k=1}^{r} r_{kj}^2}{2N_j} \right) \quad (5)$$

$\frac{\lambda}{2}$ is a constant in all terms and, therefore, we can cancel it out. Splitting the equations into $\mathbf{L}$ and $\mathbf{R}$ related terms, we obtain

$$\|\mathbf{L}\|_F^2 = \frac{1}{N_i} \sum_{(i,j)\in\Omega} \|\mathbf{l}_i\|_2^2 = \frac{1}{N_i} \sum_{(i,j)\in\Omega} \sum_{k=1}^{r} l_{ik}^2$$

and

$$\|\mathbf{R}\|_F^2 = \frac{1}{N_j} \sum_{(i,j)\in\Omega} \|\mathbf{r}_j\|_2^2 = \frac{1}{N_j} \sum_{(i,j)\in\Omega} \sum_{k=1}^{r} r_{kj}^2$$

The Frobenius norm, defined as $\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2}$, is the same as the two outer terms in both equations, taking the square into account. $\frac{1}{N_{i/j}}$ serves as a normalization constant that is only required, because we are looking at a subset of the rows and columns of $\mathbf{L}$ and $\mathbf{R}$, respectively.

Now the only equality left to show is that

$$\|\mathbf{L}\|_F^2 = \frac{1}{N_i} \sum_{(i,j)\in\Omega} \|\mathbf{l}_i\|_2^2$$

and

$$\|\mathbf{R}\|_F^2 = \frac{1}{N_j} \sum_{(i,j)\in\Omega} \|\mathbf{r}_j\|_2^2$$

As in the exercise for matrix completion, we can use the fact that $\|A\|_F^2 = \sum_i \|a_i\|_2^2$. Treating $\frac{1}{N_{i/j}}$ as a normalization constant again we can see that this equality, too, holds.

We, therefore, have shown that Equation 1, 2 and 3 are equivalent.

2. COMPUTING THE LOCAL GRADIENTS

2.a Gradient per Entry

Task. Set

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

as in (2) and (3). Compute the gradients w.r.t. to each entry in the factor matrices, i.e., compute

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

First, we compute the gradient with regards to l_{ik} which we can write as

$$\begin{aligned}\nabla_{l_{ik}} L_{ij}(\mathbf{l}_i, \mathbf{r}_j) = \nabla_{l_{ik}} &\left(d_{ij} - \sum_{k=1}^{r} l_{ik} r_{kj}\right)^2 \\ &+ \nabla_{l_{ik}} \frac{\lambda \sum_{k=1}^{r} l_{ik}^2}{2N_i} \\ &+ \nabla_{l_{ik}} \frac{\lambda \sum_{k=1}^{r} r_{kj}^2}{2N_j}\end{aligned}$$

using the sum-rule. The last term does not depend on l_{ik}, can, therefore, be treated as a constant, and becomes 0 in the derivative. In the second summand we can factor out $\frac{\lambda}{2N_i}$ and are left with $\nabla_{l_{ik}} \sum_{k=1}^{r} l_{ik}^2$ which is equal to $2 \cdot \sum_{k=1}^{r} l_{ik}$. This leaves us with

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

Finally, we apply the chain rule and get

$$\nabla_{l_{ik}} L_{ij}(\mathbf{l}_i, \mathbf{r}_j) = -2 \cdot (d_{ij} - \sum_{k=1}^{r} l_{ik} r_{kj}) \cdot (\sum_{k=1}^{r} r_{kj}) + \frac{\lambda}{N_i} \sum_{k=1}^{r} l_{ik}$$

Similarly we get as the result for $\nabla_{r_{kj}} L_{ij}(\mathbf{l}_i, \mathbf{r}_j)$

$$\nabla_{r_{kj}} L_{ij}(\mathbf{l}_i, \mathbf{r}_j) = -2 \cdot (\sum_{k=1}^{r} l_{ik}) \cdot (d_{ij} - \sum_{k=1}^{r} l_{ik} r_{kj}) + \frac{\lambda}{N_j} \sum_{k=1}^{r} r_{kj}$$

2.b Gradient per Row/Column

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

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

For this subtask we can reuse the results from Section 2.a and replace the sums over k values with a vectorized computation. This results in the following equation for $\nabla_{\mathbf{l}_i^T} L_{ij}(\mathbf{l}_i, \mathbf{r}_j)$

$$\nabla_{\mathbf{l}_i^T} L_{ij}(\mathbf{l}_i, \mathbf{r}_j) = -2 \cdot (d_{ij} - \mathbf{l}_i^T \mathbf{r}_j) \cdot \mathbf{r}_j^T + \frac{\lambda}{N_i} \mathbf{l}_i^T$$

and for $\nabla_{\mathbf{r}_j^T} L_{ij}(\mathbf{l}_i, \mathbf{r}_j)$

$$\nabla_{\mathbf{r}_j^T} L_{ij}(\mathbf{l}_i, \mathbf{r}_j) = -2 \cdot \mathbf{l}_i^T \cdot (d_{ij} - \mathbf{l}_i^T \mathbf{r}_j) + \frac{\lambda}{N_j} \mathbf{r}_j^T$$

3. IMPLEMENTING GRADIENT DESCENT

Task. Complete the function gdepoch. 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.

The goal of our implementation is to compute an estimation of the full matrix, based on the revealed entries. In contrast to the Singular Value Decomposition, we only take revealed entries into account and ignore missing values completely.

We try to minimize the loss function from Section 1 by the application of Gradient Descent. Therefore, we compute the gradient at a random point and walk downhill towards a minimum. After each step (epoch) we are closer to the minimum and our approximation of the matrix should become better. The first approximation after 5 epochs is shown in Figure 1. We can only see a few grey dots on a black background without the possibility to identify any shapes in the picture. At this point we are still far away from the best factorization and observe a high error between the original matrix $\mathbf{D}$ and the rank r factorization $\mathbf{LR}$.

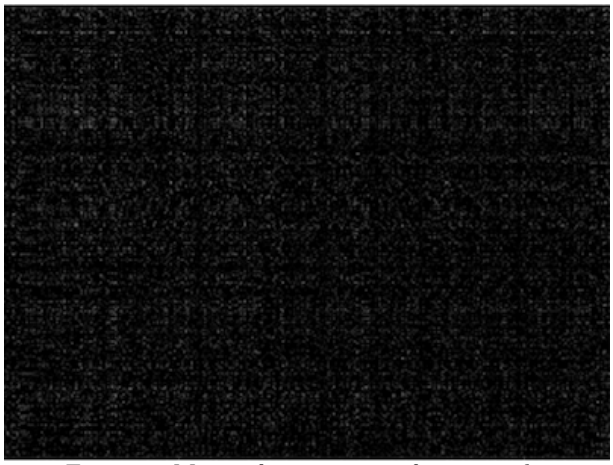
Figure 1: Matrix factorization after 5 epochs

Figure 2 visualizes the factorization after 15 epochs of gradient descent. It includes shapes that are clearly distinguishable and, with having in mind, that the original image shows three persons with sunglasses and a label at the bottom we can make out the basic shapes of all of these. The factorization after 15 epochs is much better than after 5. It reconstructs the basic structure of the image without showing any details.

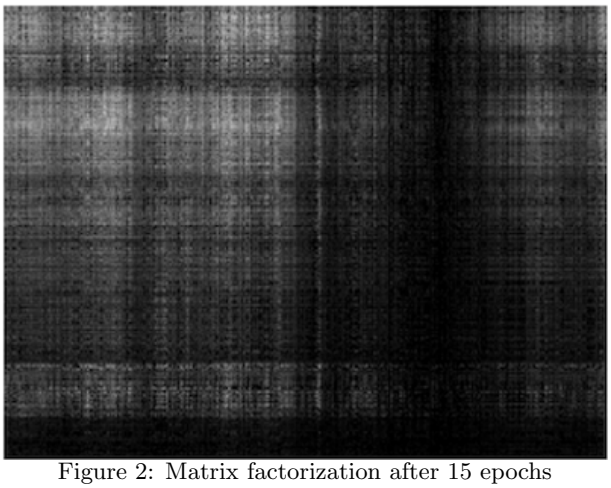
Figure 2: Matrix factorization after 15 epochs

Finally, we look at the factorization after running 50 epochs of gradient descent. The image in Figure 3 clearly resembles

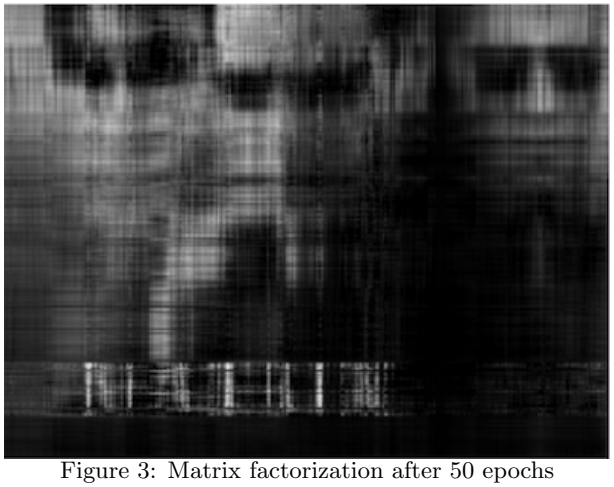

Figure 3: Matrix factorization after 50 epochs

the original image with unrevealed entries. We can clearly identify three persons and the "Matrix" text in the bottom left. The difference between the original image with unrevealed entries and the factorization **LR** seems very small.

Yet, there still is a lot of noise in the image. Running gradient descent for image completion is a suboptimal method, because it does not take the relation between neighbouring pixels into account. Another factor is that we only compute a rank r approximation instead of a perfect factorization.

Looking at the computed error in each epoch we can see that it sharply falls in the beginning, but the improvement continues to decline with each epoch. Approaching the minimum, it becomes harder to optimize the image even further without increasing the loss elsewhere.

All in all, we draw the conclusion that the gradient descent approach produces a good factorization for the sparse matrix **D**.

4. IMPLEMENTING STOCHASTIC GRADIENT DESCENT

Task. 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.

The resulting factorization **LR** for matrix **D** is shown in Figure 4. As in Figure 3, we see three persons with sunglasses and the "Matrix" label in the bottom left of the image. Both images look very similar, which means that both methods found the same minimum of the error function.

From the error per epoch in the notebook we also derive that they start with the same error and finish of at a similar error that is around 415. The errors of corresponding epochs are closely together.

This result is expected, because the stochastic gradient descent should approximate converge to the same optimum as

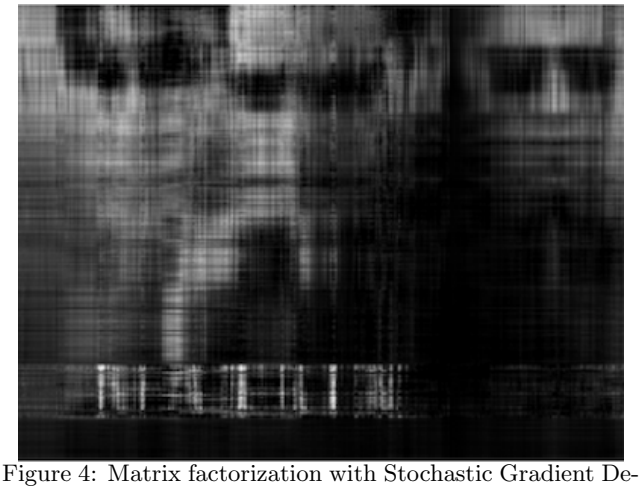

Figure 4: Matrix factorization with Stochastic Gradient Descent

the gradient descent, especially when the Loss function is convex.

5. IMPACT OF PARAMETER CHOICES

Task. 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.

For this section we compute all results with stochastic gradient descent.

The first parameter combination we look at are the parameters that we have used across all previous sections. It uses rank $r = 10$ and $\lambda = 2$, which means that the factorization is mildly regularized. This again means that we apply a small penalty if the magnitude of the matrices **L** and **R** becomes large. The result in Figure 5 demonstrates, as expected, a similar image to the ones in the previous chapters.

In Figure 6 we remove the regularization and only minimize the error between the original matrix and our factorization. In the picture we observe brighter spots than before and a greater variation of colors. Especially the more prominent features of the left two faces, like the noses and the cheekbones have a color closer to white than in the mildly regularized image.

As the regularization imposes a penalty on large values this is expected. The absence of this regularisation allows the

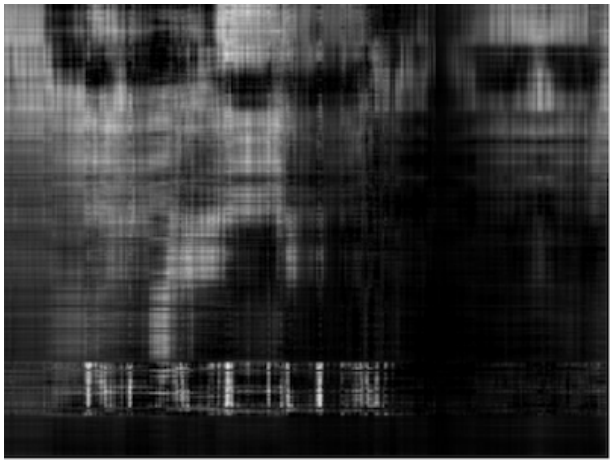

Figure 5: Mildly regularized matrix factorization

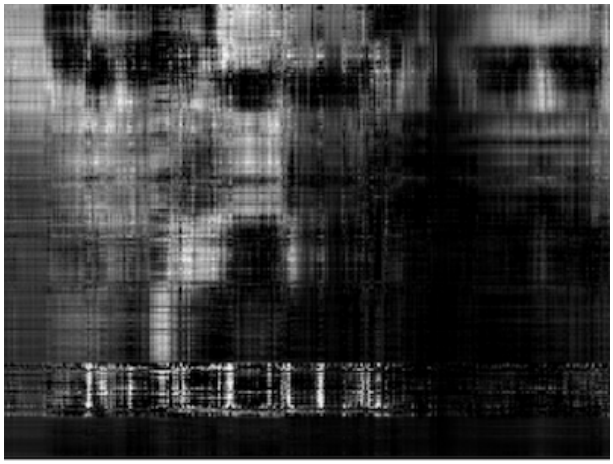

Figure 6: Unregularized matrix factorization

factorization to minimize the error even further by increasing the values in some spots. This is a common consequence of overfitting the dataset and one of the reasons why regularization is applied [1, cf. p.227].

The next parameter combination applies a heavy penalty for large matrix norms of $\mathbf{L}$ and $\mathbf{R}$. The resulting image is displayed in Figure 7. We are able to single out the basic

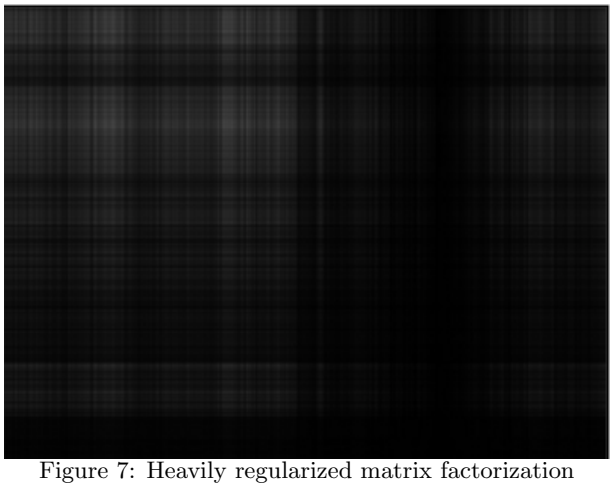

Figure 7: Heavily regularized matrix factorization

outlines of the three characters in the image, but, overall, very little gets revealed. Instead of overfitting the image like in Figure 6, we underfit it, which means that the penalty on the matrices is so high that they are not flexible enough to capture the essence of the original matrix $\mathbf{D}$.

The last two parameter combinations focus on the rank of the matrix instead of the regularization parameter λ. A low rank corresponds to a rather simple matrix, because all rows and columns have to be linear combinations of another, while a high rank allows the matrix to be more complex.

The matrix factorization for rank 1 is visualized in Figure 8. The shapes look similar to the gradient descent result after 15 epochs from Figure 2. We can also observe that

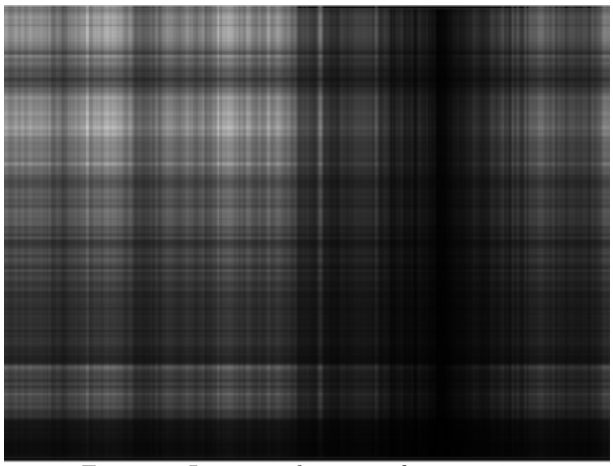

Figure 8: Lower rank matrix factorization

the image consists exclusively of straight lines without any clearly distinguishable pixels. This is the case, because every

row and column is a linear combination and, thus, it is not possible to create any shapes, except through intersections of weighted lines.

On the opposite side, we have the high rank factorization in Figure 9, which captures many details. The image looks sim-

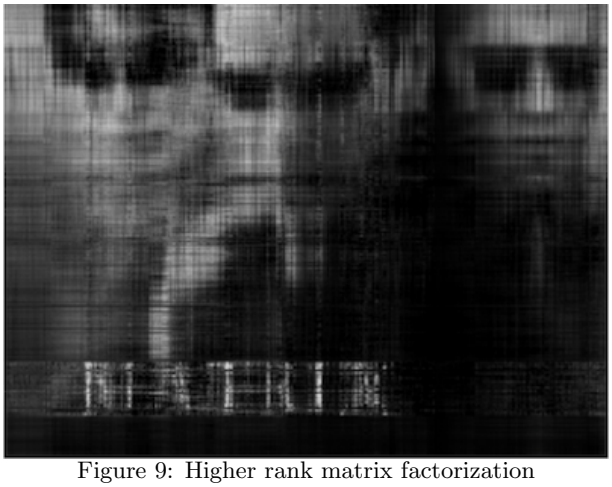

Figure 9: Higher rank matrix factorization

ilar to the mildly regularized image in Figure 5, but makes some minor improvements around the "Matrix" label on the bottom left. With the additional complexity that is allowed through the higher rank it improves on some structures, but is still similar to the rank 10 image.

All in all, the choice of the parameters $r = 10$ and $\lambda = 2$ seems the most reasonable, as the resulting image contains all important structures and, at the same time, is not very complex.

6. REFERENCES

[1] K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.