Lecture Notes 10: Matrix Factorization

1 Low-rank models

1.1 Rank-1 model

Consider the problem of modeling a quantity $y[i,j]$ that depends on two indices i and j. To fix ideas, assume that $y[i,j]$ represents the rating assigned to a movie i by a user j. If we have available a data set of such ratings, how can we predict new ratings for (i,j) that we have not seen before? A possible assumption is that some movies are more popular in general than others, and some users are more generous. This is captured by the following simple model

$$y[i,j] \approx a[i]b[j]. \tag{1}$$

The features a and b capture the respective contributions of the movie and the user to the ranking. If $a[i]$ is large, movie $[i]$ receives good ratings in general. If $b[j]$ is large, then user $[i]$ is generous, they give good ratings in general.

In the model (1) the two unknown parameters $a[i]$ and $b[j]$ are combined multiplicatively. As a result, if we fit these parameters using observed data by minimizing a least-squares cost function, the optimization problem is not convex. Figure 1 illustrates this for the function

$$f(a,b) := (1-ab)^2, \tag{2}$$

which corresponds to an example where there is only one movie and one user and the rating equals one. Nonconvexity is problematic because if we use algorithms such as gradient descent to minimize the cost function, they may get stuck in local minima corresponding to parameter values that do not fit the data well. In addition, there is a scaling issue: the pairs (a,b) and $(ac,b/c)$ yield the same cost for any constant c. This motivates incorporating a constraint to restrict the magnitude of some of the parameters. For example, in Figure 1 we add the constraint $|a|=1$, which is a nonconvex constraint set.

Assume that there are m movies and n users in the data set and every user rates every movie. If we store the ratings in a matrix Y such that $Y_{ij} := y[i,j]$ and the movie and user features in the vectors $\vec{a} \in \mathbb{R}^m$ and $\vec{b} \in \mathbb{R}^n$, then equation (1) is equivalent to

$$Y \approx \vec{a}\,\vec{b}^T. \tag{3}$$

Now consider the problem of fitting the problem by solving the optimization problem

$$\min_{\vec{a}\in\mathbb{R}^m,\,\vec{b}\in\mathbb{R}^n} \left|\left|Y - \vec{a}\,\vec{b}^T\right|\right|_{\text{F}} \qquad \text{subject to} \quad \left|\left|\vec{a}\right|\right|_2 = 1. \tag{4}$$

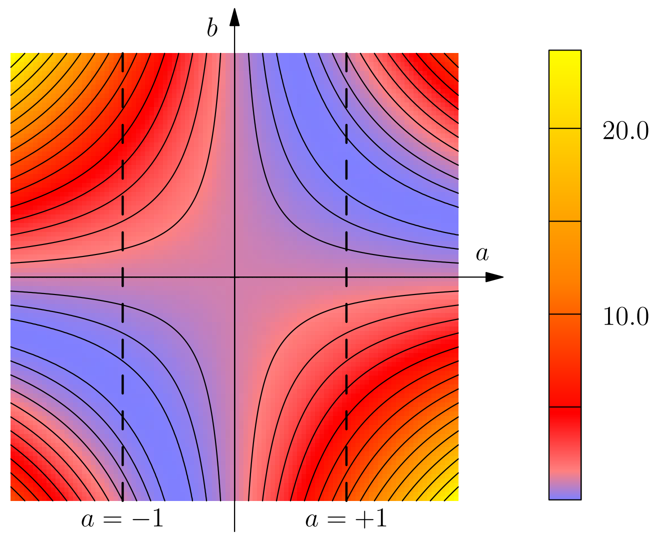

Figure 1: Heat map and contour map for the function $(1-ab)^2$. The dashed line correspond to the set $|a| = 1$.

Note that $\vec{a}\,\vec{b}^T$ is a rank-1 matrix and conversely any rank-1 matrix can be written in this form where $||\vec{a}||_2 = 1$ ($\vec{a}$ is equal to any of the columns normalized by their ℓ_2 norm). The problem is consequently equivalent to

$$\min_{X \in \mathbb{R}^{m \times n}} ||Y - X||_{\mathrm{F}} \qquad \text{subject to} \quad \operatorname{rank}(X) = 1. \tag{5}$$

By Theorem 2.10 in Lecture Notes 2 the solution $X_{\min}$ to this optimization problem is given by the truncated singular-value decomposition (SVD) of Y

$$X_{\min} = \sigma_1 \vec{u}_1 \vec{v}_1^T, \tag{6}$$

where σ_1 is the largest singular value and $\vec{u}_1$ and $\vec{v}_1$ are the corresponding singular vectors. The corresponding solutions $\vec{a}_{\min}$ and $\vec{b}_{\min}$ to problem (4) are

$$\vec{a}_{\min} = \vec{u}_1, \tag{7}$$

$$\vec{b}_{\min} = \sigma_1 \vec{v}_1. \tag{8}$$

Example 1.1 (Movies)**.** Bob, Molly, Mary and Larry rate the following six movies from 1 to 5,

$$A := \begin{array}{cccccl} & \text{Bob} & \text{Molly} & \text{Mary} & \text{Larry} & \\ \left(\right. & 1 & 1 & 5 & 4 & \left.\right) \text{The Dark Knight} \\ & 2 & 1 & 4 & 5 & \text{Spiderman 3} \\ & 4 & 5 & 2 & 1 & \text{Love Actually} \\ & 5 & 4 & 2 & 1 & \text{Bridget Jones's Diary} \\ & 4 & 5 & 1 & 2 & \text{Pretty Woman} \\ & 1 & 2 & 5 & 5 & \text{Superman 2} \end{array} \tag{9}$$

To fit a low-rank model, we first subtract the average rating

$$\mu := \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} A_{ij}, \tag{10}$$

from each entry in the matrix to obtain a centered matrix C and then compute its singular-value decomposition

$$A - \mu \vec{1}\,\vec{1}^T = USV^T = U \begin{bmatrix} 7.79 & 0 & 0 & 0 \\ 0 & 1.62 & 0 & 0 \\ 0 & 0 & 1.55 & 0 \\ 0 & 0 & 0 & 0.62 \end{bmatrix} V^T. \tag{11}$$

where $\vec{1} \in \mathbb{R}^4$ is a vector of ones. The fact that the first singular value is significantly larger than the rest suggests that the matrix may be well approximated by a rank-1 matrix. This is indeed the case:

$$\mu \vec{1}\,\vec{1}^T + \sigma_1 \vec{u}_1 \vec{v}_1^T = \begin{array}{c} \\ \\ \end{array} \begin{array}{cccc|l} \text{Bob} & \text{Molly} & \text{Mary} & \text{Larry} & \\ 1.34\,(1) & 1.19\,(1) & 4.66\,(5) & 4.81\,(4) & \text{The Dark Knight} \\ 1.55\,(2) & 1.42\,(1) & 4.45\,(4) & 4.58\,(5) & \text{Spiderman 3} \\ 4.45\,(4) & 4.58\,(5) & 1.55\,(2) & 1.42\,(1) & \text{Love Actually} \\ 4.43\,(5) & 4.56\,(4) & 1.57\,(2) & 1.44\,(1) & \text{Bridget Jones's Diary} \\ 4.43\,(4) & 4.56\,(5) & 1.57\,(1) & 1.44\,(2) & \text{Pretty Woman} \\ 1.34\,(1) & 1.19\,(2) & 4.66\,(5) & 4.81\,(5) & \text{Superman 2} \end{array} \tag{12}$$

For ease of comparison the values of A are shown in brackets. The first left singular vector is equal to

$$\vec{u}_1 := \begin{array}{cccccc} \text{D. Knight} & \text{Spiderman 3} & \text{Love Act.} & \text{B.J.'s Diary} & \text{P. Woman} & \text{Superman 2} \\ (\; -0.45 & -0.39 & 0.39 & 0.39 & 0.39 & -0.45 \;) \end{array}.$$

This vector allows us to cluster the movies: movies with negative entries are similar (in this case action movies) and movies with positive entries are similar (in this case romantic movies).

The first right singular vector is equal to

$$\vec{v}_1 = \begin{array}{cccc} \text{Bob} & \text{Molly} & \text{Mary} & \text{Larry} \\ (0.48 & 0.52 & -0.48 & -0.52) \end{array}. \tag{13}$$

This vector allows to cluster the users: negative entries indicate users that like action movies but hate romantic movies (Bob and Molly), whereas positive entries indicate the contrary (Mary and Larry). △

1.2 Rank-r model

Our rank-1 matrix model is extremely simplistic. Different people like different movies. In order to generalize it we can consider r factors that capture the dependence between the ratings and the movie/user

$$y[i,j] \approx \sum_{l=1}^{r} a_l[i] b_l[j]. \tag{14}$$

The parameters of the model have the following interpretation:

- $a_l[i]$: movie i is positively (> 0), negatively (< 0) or not (≈ 0) associated to factor l.
- $b_l[j]$: user j likes (> 0), hates (< 0) or is indifferent (≈ 0) to factor l.

The model learns the factors directly from the data. In some cases, these factors may be interpretable– for example, they can be associated to the genre of the movie as in Example 1.1 or the age of the user– but this is not always the case.

The model (14) corresponds to a rank-r model

$$Y \approx AB, \qquad A \in \mathbb{R}^{m\times r}, \quad B \in \mathbb{R}^{r\times n}. \tag{15}$$

We can fit such a model by computing the SVD of the data and truncating it. By Theorem 2.10 in Lecture Notes 2 the truncated SVD is the solution to

$$\min_{A\in\mathbb{R}^{m\times r}, B\in\mathbb{R}^{r\times n}} \left|\left|Y - A\,B\right|\right|_{\mathrm{F}} \qquad \text{subject to} \quad \left|\left|\vec{a}_1\right|\right|_2 = 1, \ldots, \left|\left|\vec{a}_r\right|\right|_2 = 1. \tag{16}$$

However, the SVD provides an estimate of the matrices A and B that constrains the columns of A and the rows of B to be orthogonal. As a result, these vectors do not necessarily correspond to interpretable factors.

2 Matrix completion

2.1 Missing data

The Netflix Prize was a contest organized by Netflix from 2007 to 2009 in which teams of data scientists tried to develop algorithms to improve the prediction of movie ratings. The problem of predicting ratings can be recast as that of completing a matrix from some of its entries, as illustrated in Figure 2. This problem is known as matrix completion.

At first glance, the problem of completing a matrix such as this one

$$\begin{bmatrix} 1 & ? & 5 \\ ? & 3 & 2 \end{bmatrix} \tag{17}$$

may seem completely ill posed. We can just fill in the missing entries arbitrarily! In more mathematical terms, the completion problem is equivalent to an underdetermined system of equations

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} M_{11} \\ M_{21} \\ M_{12} \\ M_{22} \\ M_{13} \\ M_{23} \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \\ 5 \\ 2 \end{bmatrix}. \tag{18}$$

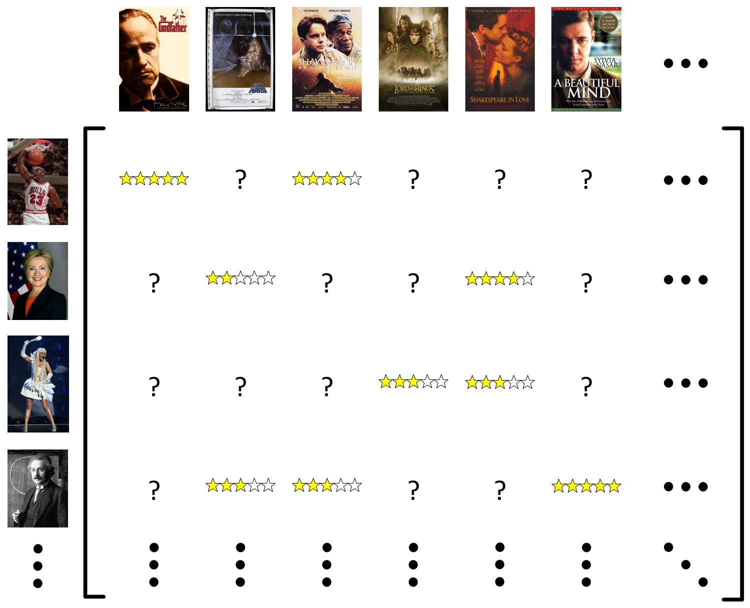

Figure 2: A depiction of the Netflix challenge in matrix form. Each row corresponds to a user that ranks a subset of the movies, which correspond to the columns. The figure is due to Mahdi Soltanolkotabi.

In order to solve the problem, we need to make an assumption on the structure of the matrix that we aim to complete. In compressed sensing we make the assumption that the original signal is sparse. In the case of matrix completion, we make the assumption that the original matrix is low rank. This implies that there exists a high correlation between the entries of the matrix, which may make it possible to infer the missing entries from the observations. As a very simple example, consider the following matrix

$$\begin{bmatrix} 1 & 1 & 1 & 1 & ? & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ ? & 1 & 1 & 1 & 1 & 1 \end{bmatrix}. \tag{19}$$

Setting the missing entries to 1 yields a rank 1 matrix, whereas setting them to any other number yields a rank 2 or rank 3 matrix.

The low-rank assumption implies that if the matrix has dimensions $m \times n$ then it can be factorized into two matrices that have dimensions $m \times r$ and $r \times n$. This factorization allows to encode the matrix using $r(m+n)$ parameters. If the number of observed entries is larger than $r(m+n)$ parameters then it may be possible to recover the missing entries. However, this is not enough to ensure that the problem is well posed.

2.2 When is matrix completion well posed?

The results of matrix completion will obviously depend on the subset of entries that are observed. For example, completion is impossible unless we observe at least one entry in each row and column.

To see why let us consider a rank 1 matrix for which we do not observe the second row,

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ ? & ? & ? & ? \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ ? \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}. \tag{20}$$

As long as we set the missing row to equal the same value, we will obtain a rank-1 matrix consistent with the measurements. In this case, the problem is not well posed.

In general, we need samples that are distributed across the whole matrix. This may be achieved by sampling entries uniformly at random. Although this model does not completely describe matrix completion problems in practice (some users tend to rate more movies, some movies are very popular and are rated by many people), making the assumption that the revealed entries are random simplifies theoretical analysis and avoids dealing with adversarial cases designed to make deterministic patterns fail.

We now turn to the question of what matrices can be completed from a subset of entries samples uniformly at random. Intuitively, matrix completion can be achieved when the information contained in the entries of the matrix is *spread out* across multiple entries. If the information is very localized then it will be impossible to reconstruct the missing entries. Consider a simple example where the matrix is sparse

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 23 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{21}$$

If we don't observe the nonzero entry, we will naturally assume that it was equal to zero.

The problem is not restricted to sparse matrices. In the following matrix the last row does not seem to be correlated to the rest of the rows,

$$M := \begin{bmatrix} 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 \\ -3 & 3 & -3 & 3 \end{bmatrix}. \tag{22}$$

This is revealed by the singular-value decomposition of the matrix, which allows to decompose it into two rank-1 matrices.

$$
\begin{aligned}
M &= U\,SV^T \qquad (23)\\
&= \begin{bmatrix} 0.5 & 0 \\ 0.5 & 0 \\ 0.5 & 0 \\ 0.5 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 8 & 0 \\ 0 & 6 \end{bmatrix} \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ -0.5 & 0.5 & -0.5 & 0.5 \end{bmatrix} \qquad (24)\\
&= 8 \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \\ 0 \end{bmatrix} \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \end{bmatrix} + 6 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} -0.5 & 0.5 & -0.5 & 0.5 \end{bmatrix} \qquad (25)\\
&= \sigma_1 U_1 V_1^T + \sigma_2 U_2 V_2^T. \qquad (26)
\end{aligned}
$$

The first rank-1 component of this decomposition has information that is very spread out,

$$
\sigma_1 U_1 V_1^T = \begin{bmatrix} 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \qquad (27)
$$

The reason is that most of the entries of V_1 are nonzero and have the same magnitude, so that each entry of U_1 affects every single entry of the corresponding row. If one of those entries is missing, we can still recover the information from the other entries.

In contrast, the information in the second rank-1 component is very localized, due to the fact that the corresponding left singular vector is very sparse,

$$
\sigma_2 U_2 V_2^T = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -3 & 3 & -3 & 3 \end{bmatrix}. \qquad (28)
$$

Each entry of the right singular vector only affects one entry of the component. If we don't observe that entry then it will be impossible to recover.

This simple example shows that sparse singular vectors are problematic for matrix completion. In order to quantify to what extent the information is spread out across the low-rank matrix we define a coherence measure that depends on the singular vectors.

Definition 2.1 (Coherence). *Let $U\,SV^T$ be the singular-value decomposition of an $n \times n$ matrix M with rank r. The coherence μ of M is a constant such that*

$$
\max_{1 \le j \le n} \sum_{i=1}^{r} U_{ij}^2 \le \frac{n\mu}{r} \qquad (29)
$$

$$
\max_{1 \le j \le n} \sum_{i=1}^{r} V_{ij}^2 \le \frac{n\mu}{r}. \qquad (30)
$$

This condition was first introduced in [1]. Its exact formulation is not too important. The point is that matrix completion from uniform samples only makes sense for matrices which are incoherent, and therefore do not have spiky singular values. There is a direct analogy with the super-resolution problem, where sparsity is not a strong enough constraint to make the problem well posed and the class of signals of interest has to be further restricted to signals with supports that satisfy a minimum separation.

2.3 Minimizing the nuclear norm

We are interested in recovering low-rank matrices from a subset of their entries. Let $\vec{y}$ be a vector containing the revealed entries and let Ω be the corresponding entries. Ideally, we would like to select the matrix with the lowest rank that corresponds to the measurements,

$$\min_{X \in \mathbb{R}^{m \times n}} \operatorname{rank}(X) \quad \text{such that } X_\Omega = \vec{y}. \tag{31}$$

Unfortunately, this optimization problem is computationally hard to solve. Substituting the rank with the nuclear norm yields a tractable alternative:

$$\min_{X \in \mathbb{R}^{m \times n}} ||X||_* \quad \text{such that } X_\Omega = \vec{y}. \tag{32}$$

The cost function is convex and the constraint is linear, so this is a convex program. In practice, the revealed entries are usually noisy. They do not correspond exactly to entries from a low-rank matrix. We take this into account by removing the equality constraint and adding a data-fidelity term penalizing the ℓ_2-norm error over the revealed entries in the cost function,

$$\min_{X \in \mathbb{R}^{m \times n}} \frac{1}{2} ||X_\Omega - \vec{y}||_2^2 + \lambda ||X||_*, \tag{33}$$

where $\lambda > 0$ is a regularization parameter.

Example 2.2 (Collaborative filtering (simulated)). We now apply this method to the following completion problem:

$$\begin{array}{cccccc} & \text{Bob} & \text{Molly} & \text{Mary} & \text{Larry} & \\ \left(\right. & 1 & ? & 5 & 4 & \left.\right) \text{The Dark Knight} \\ & ? & 1 & 4 & 5 & \text{Spiderman 3} \\ & 4 & 5 & 2 & ? & \text{Love Actually} \\ & 5 & 4 & 2 & 1 & \text{Bridget Jones's Diary} \\ & 4 & 5 & 1 & 2 & \text{Pretty Woman} \\ & 1 & 2 & ? & 5 & \text{Superman 2} \end{array} \tag{34}$$

In more detail we apply the following steps:

1. We compute the average observed rating and subtract it from each entry in the matrix. We denote the vector of centered ratings by y.

2. We solve the optimization problem (32).

3. We add the average observed rating to the solution of the optimization problem and round each entry to the nearest integer.

The result is pretty good,

$$\begin{array}{cccccl} & \text{Bob} & \text{Molly} & \text{Mary} & \text{Larry} & \\ \left(\right. & 1 & 2\ (1) & 5 & 4 & \left.\right) \text{The Dark Knight} \\ & 2\ (2) & 1 & 4 & 5 & \text{Spiderman 3} \\ & 4 & 5 & 2 & 2\ (1) & \text{Love Actually} \\ & 5 & 4 & 2 & 1 & \text{Bridget Jones's Diary} \\ & 4 & 5 & 1 & 2 & \text{Pretty Woman} \\ & 1 & 2 & 5\ (5) & 5 & \text{Superman 2} \end{array} \tag{35}$$

For comparison the original ratings are shown in brackets. △

2.4 Algorithms

In this section we describe a proximal-gradient method to solve Problem 33. Recall that proximal-gradient methods allow to solve problems of the form

$$\text{minimize} \quad f(x) + g(x), \tag{36}$$

where f is differentiable and we can apply the proximal operator prox_g efficiently.

Recall that the proximal operator norm of the ℓ_1 norm is a soft-thresholding operator. Analogously, the proximal operator of the nuclear norm is applied by soft-thresholding the singular values of the matrix.

Theorem 2.3 (Proximal operator of the nuclear norm). *The solution to*

$$\min_{X \in \mathbb{R}^{m \times n}} \frac{1}{2} \left|\left|Y - X\right|\right|_{\text{F}}^2 + \tau \left|\left|X\right|\right|_* \tag{37}$$

is $\mathcal{D}_\tau(Y)$, obtained by soft-thresholding the singular values of $Y = U\, SV^T$

$$\mathcal{D}_\tau(Y) := U\, \mathcal{S}_\tau(S)\, V^T, \tag{38}$$

$$\mathcal{S}_\tau(S)_{ii} := \begin{cases} S_{ii} - \tau & \text{if } S_{ii} > \tau, \\ 0 & \text{otherwise.} \end{cases} \tag{39}$$

Proof. Due to the Frobenius norm term, the cost function is strictly convex. This implies that any point at which there exists a subgradient that is equal to zero is the solution to the optimization problem. The subgradients of the cost function at X are of the form,

$$X - Y + \tau G, \tag{40}$$

where G is a subgradient of the nuclear norm at X. If we can show that

$$\frac{1}{\tau}\left(Y - D_\tau(Y)\right) \tag{41}$$

is a subgradient of the nuclear norm at $D_\tau(Y)$ then $D_\tau(Y)$ is the solution.

Let us separate the singular-value decomposition of Y into the singular vectors corresponding to singular values greater than τ, denoted by U_0 and V_0 and the rest

$$Y = U\,SV^T \tag{42}$$

$$= \begin{bmatrix} U_0 & U_1 \end{bmatrix} \begin{bmatrix} S_0 & 0 \\ 0 & S_1 \end{bmatrix} \begin{bmatrix} V_0 & V_1 \end{bmatrix}^T. \tag{43}$$

Note that $D_\tau(Y) = U_0(S_0 - \tau I)V_0^T$, so that

$$\frac{1}{\tau}(Y - D_\tau(Y)) = U_0 V_0^T + \frac{1}{\tau} U_1 S_1 V_1^T. \tag{44}$$

By construction all the singular values of $U_1 S_1 V_1^T$ are smaller than τ, so

$$\left|\left| \frac{1}{\tau} U_1 S_1 V_1^T \right|\right| \leq 1. \tag{45}$$

In addition, by definition of the singular-value decomposition $U_0^T U_1 = 0$ and $V_0^T V_1 = 0$. As a result, (44) is a subgradient of the nuclear norm at $D_\tau(Y)$ and the proof is complete. □

Algorithm 2.4 (Proximal-gradient method for nuclear-norm regularization)**.** *Let Y be a matrix such that $Y_\Omega = y$ and let us abuse notation by interpreting $X_\Omega^{(k)}$ as a matrix which is zero on Ω^c. We set the initial point $X^{(0)}$ to Y. Then we iterate the update*

$$X^{(k+1)} = \mathcal{D}_{\alpha_k \lambda}\left(X^{(k)} - \alpha_k \left(X_\Omega^{(k)} - Y\right)\right), \tag{46}$$

where $\alpha_k > 0$ is the step size.

Example 2.5 (Collaborative filtering (real))**.** The Movielens data set contains ratings from 671 users for 300 movies. The ratings are between 1 and 10. Figure 3 shows the results of applying algorithm 2.4 (as implemented by this package) using a training set of 9 135 ratings and evaluating the model on 1 016 test ratings. For large values of λ the model is too low rank and is not able to fit the data, so both the training and test error is high. When λ is too small, the model is not low rank, which results in overfitting: the observed entries are approximated by a high-rank model that is not able to predict the test entries. For intermediate values of λ the model achieves an average error of about 2/10. △

2.5 Alternating minimization

Minimizing the nuclear norm to recover a low-rank matrix is an effective method but it has a drawback: it requires repeatedly computing the singular-value decomposition of the matrix, which can be computationally heavy for large matrices. A more efficient alternative is to parametrize the matrix as AB where $A \in \mathbb{R}^{m\times k}$ and $B \in \mathbb{R}^{k\times n}$, which requires fixing a value for the rank of the matrix k (in practice this can be set by cross validation). The two components A and B can then be fit by solving the optimization problem

$$\min_{\widetilde{A}\in\mathbb{R}^{m\times k}, \widetilde{B}\in\mathbb{R}^{k\times n}} \left|\left| \left(\widetilde{A}\widetilde{B}\right)_\Omega - \vec{y} \right|\right|_2. \tag{47}$$

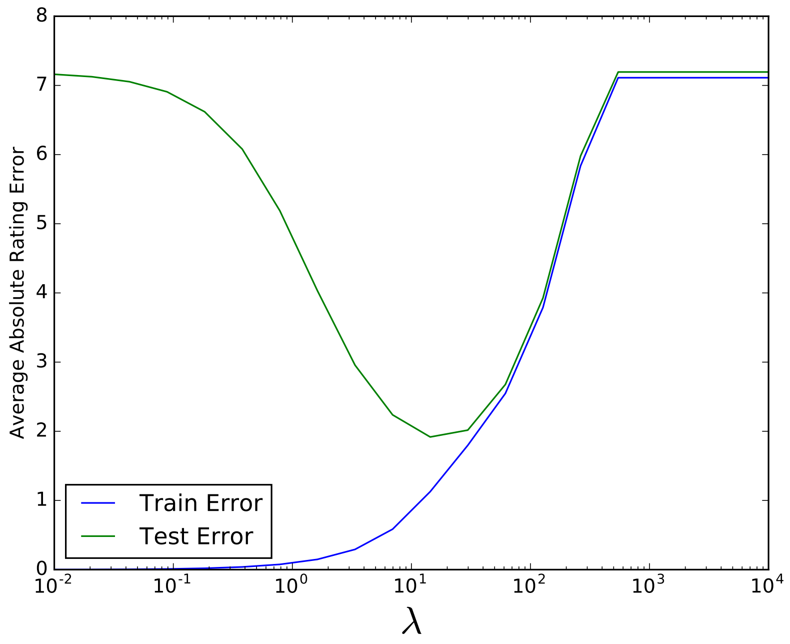

Figure 3: Training and test error of nuclear-norm-based collaborative filtering for the data described in Example 2.5.

This nonconvex problem is usually tackled by alternating minimization. Indeed, if we fix $\widetilde{B} = B$ the optimization problem over $\widetilde{A}$ is just a least-squares problem

$$\min_{\widetilde{A} \in \mathbb{R}^{m \times k}} \left|\left| \left(\widetilde{A} B \right)_{\Omega} - \vec{y} \right|\right|_2 \tag{48}$$

and the same is true for the optimization problem over $\widetilde{B}$ if we fix $\widetilde{A} = A$. Iteratively solving these least-squares problems allows to find a local minimum of the cost function. Under certain assumptions, one can even show that a certain initialization coupled with this procedure guaranteees exact recovery, see [3] for more details.

3 Structured low-rank models

3.1 Nonnegative matrix factorization

As we discussed in Lecture Notes 2, PCA can be used to compute the main principal directions of a dataset, which can be interpreted as basis vectors that capture as much of the energy as possible. These vectors are constrained to be orthogonal. Unfortunately, as a result they are often not necessarily interpretable. For example, when we compute the principal directions and components of a data set of faces, they both may have negative values, so it is difficult to interpret the directions as *face atoms* that can be added to form a face. This suggests computing a decomposition where both atoms and coefficients are nonnegative, with the hope that this will allow us to learn a more interpretable model.

A nonnegative matrix factorization of the data matrix may be obtained by solving the optimization problem,

$$\begin{aligned}
\text{minimize} \quad & \left|\left|X - \tilde{A}\,\tilde{B}\right|\right|_{\mathrm{F}}^2 && (49)\\
\text{subject to} \quad & \tilde{A}_{i,j} \geq 0, && (50)\\
& \tilde{B}_{i,j} \geq 0, \qquad \text{for all } i,j && (51)
\end{aligned}$$

where $\tilde{A} \in \mathbb{R}^{d\times r}$ and $\tilde{B} \in \mathbb{R}^{r\times n}$ for a fixed r. This is a nonconvex problem which is computationally hard, due to the nonnegative constraint. Several methods to compute local optima have been suggested in the literature, as well as alternative cost functions to replace the Frobenius norm. We refer interested readers to [4].

Figure 4 shows the columns of the left matrix A, which we interpret as atoms that can be combined to form the faces, obtained by applying this method to the faces dataset from Lecture Notes 2 and compares them to the principal directions obtained through PCA. Due to the nonnegative constraint, the atoms resemble portions of faces (the black areas have very small values) which capture features such as the eyebrows, the nose, the eyes, etc.

Example 3.1 (Topic modeling)**.** Topic modeling aims to learn the thematic structure of a text corpus automatically. We will illustrate this application with a simple example. We take six newspaper articles and compute the frequency of a list of words in each of them. Our final goal is to separate the words into different clusters that hopefully correspond to different topics. The following matrix contains the counts for each word and article. Each entry contains the number of times that the word corresponding to column j is mentioned in the article corresponding to row i.

$$A = \begin{array}{c} \\ \\ \\ \\ \\ \\ \end{array}
\begin{array}{ccccccccc|c}
\text{singer} & \text{GDP} & \text{senate} & \text{election} & \text{vote} & \text{stock} & \text{bass} & \text{market} & \text{band} & \textit{Articles}\\
6 & 1 & 1 & 0 & 0 & 1 & 9 & 0 & 8 & \text{a}\\
1 & 0 & 9 & 5 & 8 & 1 & 0 & 1 & 0 & \text{b}\\
8 & 1 & 0 & 1 & 0 & 0 & 9 & 1 & 7 & \text{c}\\
0 & 7 & 1 & 0 & 0 & 9 & 1 & 7 & 0 & \text{d}\\
0 & 5 & 6 & 7 & 5 & 6 & 0 & 7 & 2 & \text{e}\\
1 & 0 & 8 & 5 & 9 & 2 & 0 & 0 & 1 & \text{f}
\end{array}$$

Computing the singular-value decomposition of the matrix– after subtracting the mean of each entry as in (11)– we determine that the matrix is approximately low rank

$$A = USV^T = U \begin{bmatrix} 23.64 & 0 & 0 & 0 & & \\ 0 & 18.82 & 0 & 0 & 0 & 0\\ 0 & 0 & 14.23 & 0 & 0 & 0\\ 0 & 0 & 0 & 3.63 & 0 & 0\\ 0 & 0 & 0 & 0 & 2.03 & 0\\ 0 & 0 & 0 & 0 & 0 & 1.36 \end{bmatrix} V^T \qquad (52)$$

Unfortunately the singular vectors do not have an intuitive interpretation as in Section ??. In

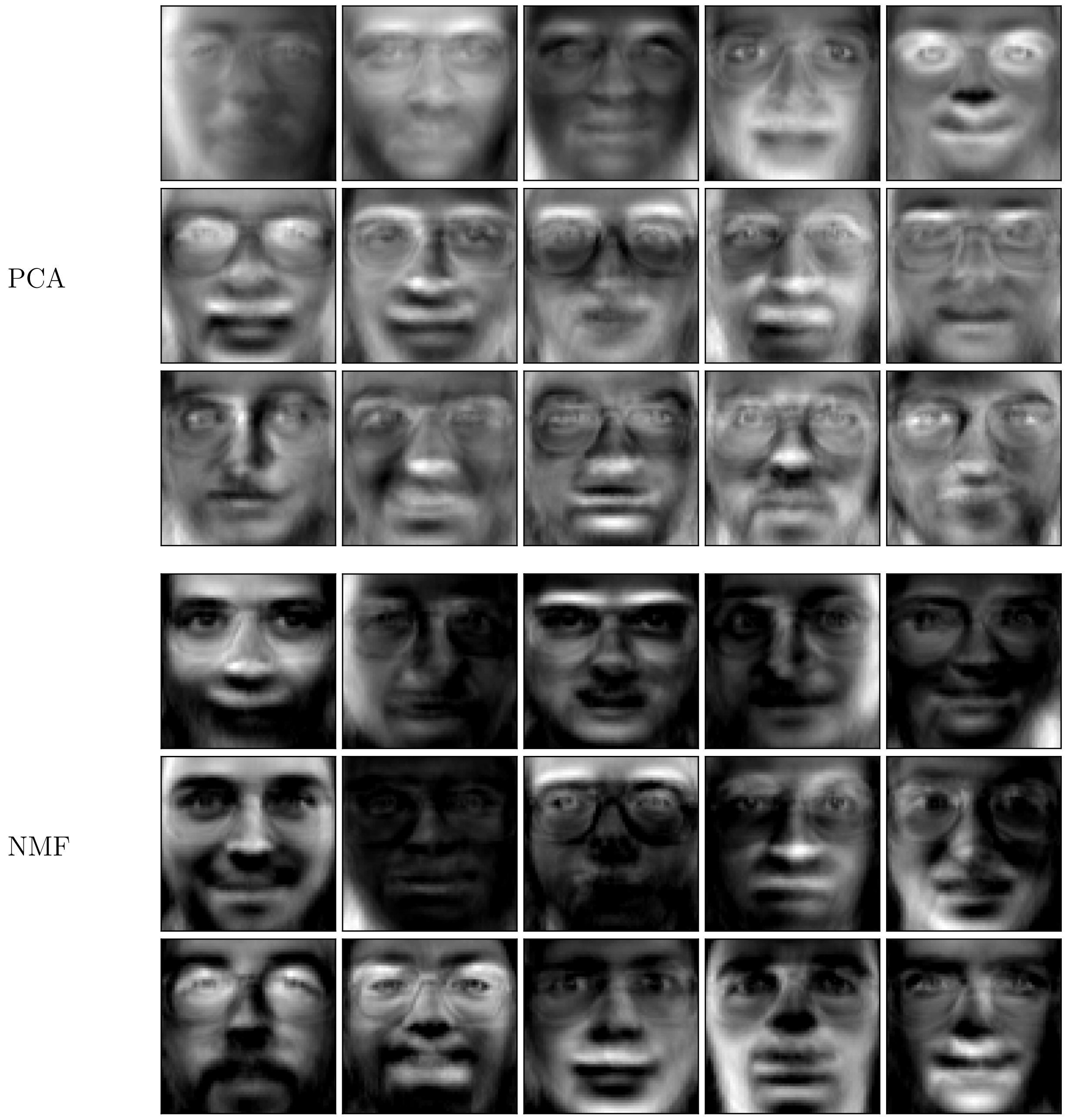

Figure 4: Atoms obtained by applying PCA and nonnegative matrix factorization to the faces dataset in Lecture Notes 2.

particular, they do not allow to cluster the words

$$\begin{array}{rcllllll}
 & & \text{a} & \text{b} & \text{c} & \text{d} & \text{e} & \text{f} \\
U_1 & = & (-0.24 & -0.47 & -0.24 & -0.32 & -0.58 & -0.47) \\
U_2 & = & (\ 0.64 & -0.23 & 0.67 & -0.03 & -0.18 & -0.21) \\
U_3 & = & (-0.08 & -0.39 & -0.08 & 0.77 & 0.28 & -0.40)
\end{array} \tag{53}$$

or the articles

$$\begin{array}{rcccccccccc}
 & & \text{singer} & \text{GDP} & \text{senate} & \text{election} & \text{vote} & \text{stock} & \text{bass} & \text{market} & \text{band} \\
V_1 & = & (-0.18 & -0.24 & -0.51 & -0.38 & -0.46 & -0.34 & -0.2 & -0.3 & -0.22) \\
V_2 & = & (\ 0.47 & 0.01 & -0.22 & -0.15 & -0.25 & -0.07 & 0.63 & -0.05 & 0.49\ ) \\
V_3 & = & (-0.13 & 0.47 & -0.3 & -0.14 & -0.37 & 0.52 & -0.04 & 0.49 & -0.07)
\end{array} \tag{54}$$

A problem here is that the singular vectors have negative entries that are difficult to interpret. In the case of rating prediction, negative ratings mean that a person does not like a movie. In contrast articles either are about a topic or they are not: it makes sense to add atoms corresponding to different topics to approximate the word count of a document but not to subtract them. Following this intuition, we apply nonnegative matrix factorization to obtain two matrices $W \in \mathbb{R}^{m\times k}$ and $H \in \mathbb{R}^{k\times n}$ such that

$$M \approx WH, \quad W_{i,j} \geq 0, \qquad 1 \leq i \leq m,\ 1 \leq j \leq k, \tag{55}$$

$$H_{i,j} \geq 0, \qquad 1 \leq i \leq k,\ 1 \leq i \leq n. \tag{56}$$

In our example, we set $k = 3$. H_1, H_2 and H_3 can be interpreted as word-count atoms, but also as coefficients that weigh the contribution of W_1, W_2 and W_3.

$$\begin{array}{rcccccccccc}
 & & \text{singer} & \text{GDP} & \text{senate} & \text{election} & \text{vote} & \text{stock} & \text{bass} & \text{market} & \text{band} \\
H_1 & = & (\ 0.34 & 0 & 3.73 & 2.54 & 3.67 & 0.52 & 0 & 0.35 & 0.35\ ) \\
H_2 & = & (\ 0 & 2.21 & 0.21 & 0.45 & 0 & 2.64 & 0.21 & 2.43 & 0.22\ ) \\
H_3 & = & (\ 3.22 & 0.37 & 0.19 & 0.2 & 0 & 0.12 & 4.13 & 0.13 & 3.43\ )
\end{array} \tag{57}$$

The latter interpretation allows to cluster the words into topics. The first topic corresponds to the entries that are not zero (or very small) in H_1: senate, election and vote. The second corresponds to H_2: GDP, stock and market. The third corresponds to H_3: singer, bass and band.

The entries of W allow to assign the topics to articles. b, e and f are about politics (topic 1), d and e about economics (topic 3) and a and c about music (topic 3)

$$\begin{array}{rcllllll}
 & & \text{a} & \text{b} & \text{c} & \text{d} & \text{e} & \text{f} \\
W_1 & = & (0.03 & 2.23 & 0 & 0 & 1.59 & 2.24) \\
W_2 & = & (\ 0.1 & 0 & 0.08 & 3.13 & 2.32 & 0\ ) \\
W_3 & = & (2.13 & 0 & 2.22 & 0 & 0 & 0.03)
\end{array} \tag{58}$$

Finally, we check that the factorization provides a good fit to the data. The product WH is equal to

$$\begin{array}{c} \begin{array}{ccccccccc} \text{singer} & \text{GDP} & \text{senate} & \text{election} & \text{vote} & \text{stock} & \text{bass} & \text{market} & \text{band} \end{array} & \textit{Art.} \\ \begin{pmatrix} 6.89\ (6) & 1.01\ (1) & 0.53\ (1) & 0.54\ (0) & 0.10\ (0) & 0.53\ (1) & 8.83\ (9) & 0.53\ (0) & 7.36\ (8) \\ 0.75\ (1) & 0\ (0) & 8.32\ (9) & 5.66\ (5) & 8.18\ (8) & 1.15\ (1) & 0\ (0) & 0.78\ (1) & 0.78\ (0) \\ 7.14\ (8) & 0.99\ (1) & 0.44\ (0) & 0.47\ (1) & 0\ (0) & 0.47\ (0) & 9.16\ (9) & 0.48\ (1) & 7.62\ (7) \\ 0\ (0) & 7\ (6.91) & 0.67\ (1) & 1.41\ (0) & 0\ (0) & 8.28\ (9) & 0.65\ (1) & 7.60\ (7) & 0.69\ (0) \\ 0.53\ (0) & 5.12\ (5) & 6.45\ (6) & 5.09\ (7) & 5.85\ (5) & 6.97\ (6) & 0.48\ (0) & 6.19\ (7) & 1.07\ (2) \\ 0.86\ (1) & 0.01\ (0) & 8.36\ (8) & 5.69\ (5) & 8.22\ (9) & 1.16\ (2) & 0.14\ (0) & 0.79\ (0) & 0.9\ (1) \end{pmatrix} & \begin{array}{c} \text{a} \\ \text{b} \\ \text{c} \\ \text{d} \\ \text{e} \\ \text{f} \end{array} \end{array}$$

For ease of comparison the values of A are shown in brackets. △

4 Sparse principal-component analysis

In certain cases, it may be desirable to learn sparse atoms that are able to represent a set of signals. In the case of the faces dataset, this may force the representation to isolate specific face features such as the mouth, the eyes, etc. In order to fit such a model, we can incorporate a sparsity constraint on the atoms by using the ℓ_1 norm

$$\text{minimize} \quad \left|\left|X - \tilde{A}\,\tilde{B}\right|\right|_2^2 + \lambda \sum_{i=1}^{k} \left|\left|\tilde{A}_i\right|\right|_1 \tag{59}$$

$$\text{subject to} \quad \left|\left|\tilde{A}_i\right|\right|_2 = 1, \qquad 1 \leq i \leq k. \tag{60}$$

Due to the sparsity-inducing constraint, this problem is computationally hard, as in the case of nonnegative matrix factorization. We refer the interested reader to [6] for algorithms to compute local minima. Figure 5 shows the atoms obtained by applying this method to the faces dataset from Figure 4. The model indeed learns very localized atoms that represent face features.

References

The tutorial [5] is an excellent reference on the application of matrix-decomposition techniques in machine learning and image processing. Chapters 7 and 8 of [2] describe low-rank models in statistics. The numerical experiments shown in these notes were implemented using scikit-learn, which is available online at `http://scikit-learn.org`. In particular, a script to apply different matrix-decomposition techniques to the faces dataset is available here.

[1] E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717–772, 2009.

[2] T. Hastie, R. Tibshirani, and M. Wainwright. *Statistical learning with sparsity: the lasso and generalizations*. CRC Press, 2015.

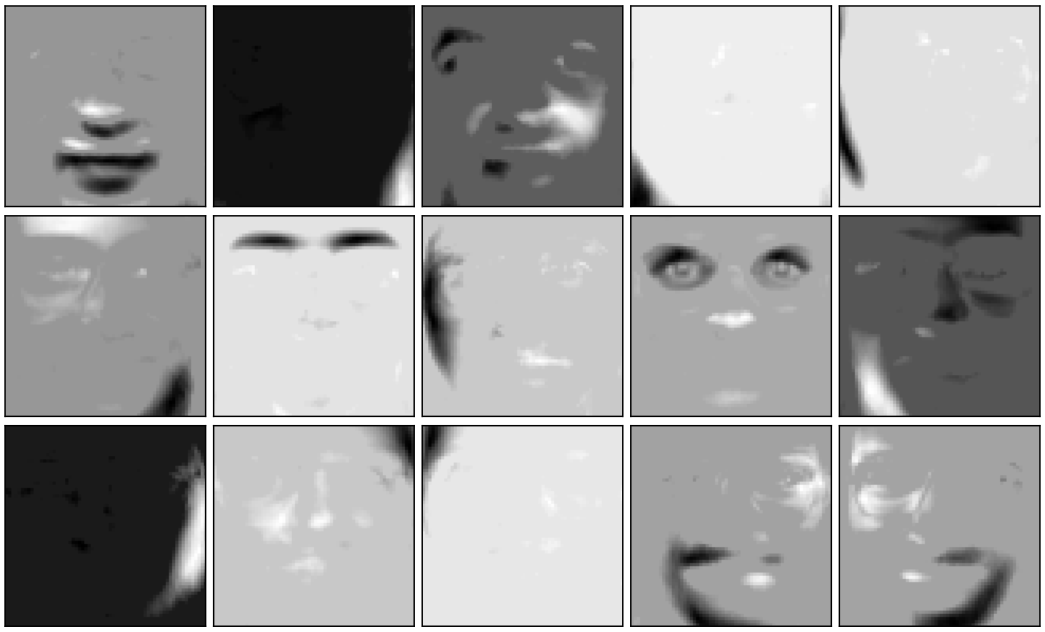

Figure 5: Atoms obtained by applying sparse PCA to the faces dataset from Figure 4.

[3] P. Jain, P. Netrapalli, and S. Sanghavi. Low-rank matrix completion using alternating minimization. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 665–674. ACM, 2013.

[4] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562, 2001.

[5] J. Mairal, F. Bach, and J. Ponce. Sparse modeling for image and vision processing. *arXiv preprint arXiv:1411.3230*, 2014.

[6] H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis. *Journal of computational and graphical statistics*, 15(2):265–286, 2006.