Learning signal representations

1 Introduction

In Lecture Notes 4 we described how to design representations that allow to represent signals with a small number of coefficients and how these sparse representation can be leveraged to compress and denoise signals. In these notes, we will consider the problem of learning a representation from a set of signals. In more detail, we assume that we have available a dataset of n signals $x_1, x_2, \ldots, x_n \in \mathbb{R}^d$ and our goal is to compute a small set of atoms $\Phi_1, \ldots, \Phi_k \in \mathbb{R}^d$ that allow to build a linear model for each signal

$$x_j \approx \sum_{i=1}^{k} \Phi_i \, A_{ij}, \quad 1 \leq j \leq n, \tag{1}$$

where $A_1, \ldots, A_n \in \mathbb{R}^k$ are coefficient vectors. Ideally, k should be significantly smaller than n, which means that the atoms are able to capture the common structure of the n signals. The learned atoms can be used for compression, denoising or as features for classification, whereas the coefficients in A may allow to cluster the different signals, as we will discuss later on.

If we group the signals into a matrix $X := \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}$, learning the atoms and the corresponding coefficients is equivalent to a matrix-factorization problem,

$$X \approx \begin{bmatrix} \Phi_1 & \Phi_2 & \cdots & \Phi_k \end{bmatrix} \begin{bmatrix} A_1 & A_2 & \cdots & A_n \end{bmatrix} = \Phi \, A \tag{2}$$

where $\Phi \in \mathbb{R}^{d \times k}$, $A \in \mathbb{R}^{k \times n}$. The assumption that k is smaller than n means that our goal is to approximate X with a low-rank matrix.

Learning signal representation is significantly more challenging than fitting models once the atoms are known. The reason is that any optimization problem of the form

$$\underset{\widetilde{\Phi}, \widetilde{A}}{\text{minimize}} \quad \left|\left| X - \widetilde{\Phi} \, \widetilde{A} \right|\right| + \mathcal{R}_1 \left(\widetilde{\Phi} \right) + \mathcal{R}_2 \left(\widetilde{A} \right), \tag{3}$$

where $||\cdot||$ is an arbitrary norm and $\mathcal{R}_1$ and $\mathcal{R}_2$ are regularization terms, is nonconvex due to the product between the variables. Figure 1 illustrates this by plotting the function $f(\phi, a) = (1 - \phi a)^2$ (i.e. there is only one 1D atom and one coefficient) which has minima along two separate lines: $(\alpha, 1/\alpha)$ and $(-\alpha, -1/\alpha)$ for any $\alpha > 0$. In higher dimensions, the number of local minima grows. Interestingly, if no regularization is added, it is possible to find a global minimum for the nonconvex problem by computing the singular-value decomposition, as we

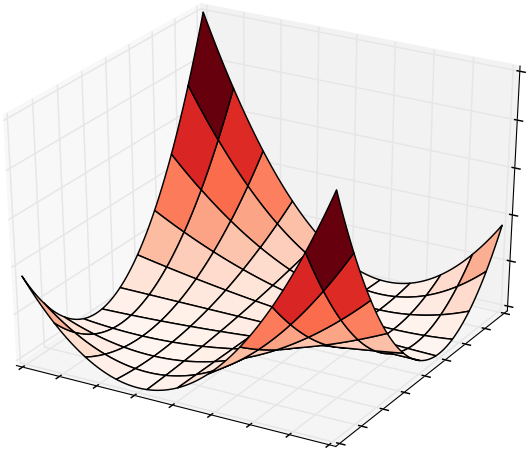

Figure 1: The function $f(\phi, a) = (1 - \phi a)^2$ has minima at $(\alpha, 1/\alpha)$ and $(-\alpha, -1/\alpha)$ for any $\alpha > 0$.

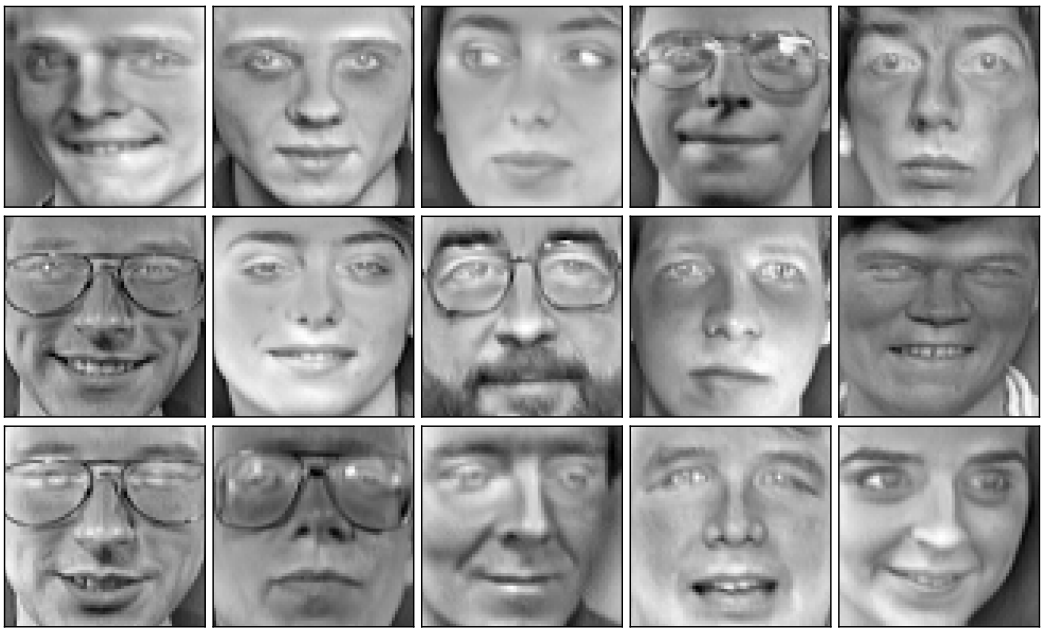

Figure 2: Examples from the face dataset. The dataset contains 10 face pictures of 40 different people.

discuss below, but this is no longer the case when we add regularization terms to the cost function.

Figure 2 shows some examples from a dataset that we will use to illustrate different models. It contains 10 face pictures of 40 different people, which were taken between April 1992 and April 1994 at AT&T Laboratories Cambridge[1].

2 K means

A possible way of representing a set of signals using a small number of atoms is to cluster the signals into several groups and assign an atom to each group. This clustering problem is typically known as k-means clustering. The aim is to learn k atoms $\Phi_1, \ldots, \Phi_k$ that minimize the cost function

$$\sum_{i=1}^{n} \left|\left|x_i - \Phi_{c(i)}\right|\right|_2^2, \tag{4}$$

$$c\left(i\right) := \arg\min_{1 \leq j \leq k} \left|\left|x_i - \Phi_j\right|\right|_2. \tag{5}$$

In words, $c\left(i\right)$ denotes the index of the atom that is assigned to the signal x_i.

After carrying out the clustering, the learned model can be expressed as factorization of the matrix X where the coefficients for each signal are restricted to just select one of the atoms,

$$X \approx \begin{bmatrix} \Phi_1 & \Phi_2 & \cdots & \Phi_k \end{bmatrix} \begin{bmatrix} e_{c(1)} & e_{c(2)} & \cdots & e_{c(n)} \end{bmatrix}, \tag{6}$$

where e_j is the standard-basis vector (all its entries are zero, except the jth entry which equals one).

Minimizing the cost function (4) over the choice of atoms and coefficients is computationally hard (NP-hard). However minimizing the cost function if the atoms or the coefficients are known is very easy. If the atoms are known, for each signal we just need to select the atom that is nearest. If the coefficients are known then the following lemma, proved in Section A.1 of the appendix, shows that the optimal atoms just correspond to the mean of each cluster.

Lemma 2.1. *For any $x_1, x_2, \ldots, x_n \in \mathbb{R}^d$ the solution to the optimization problem*

$$\underset{\phi}{\text{minimize}} \quad \sum_{i=1}^{n} \left|\left|x_i - \phi\right|\right|_2^2 \tag{7}$$

is equal to

$$\hat{\phi} = \frac{1}{n} \sum_{i=1}^{n} x_i. \tag{8}$$

[1] See http://scikit-learn.org/stable/datasets/olivetti_faces.html

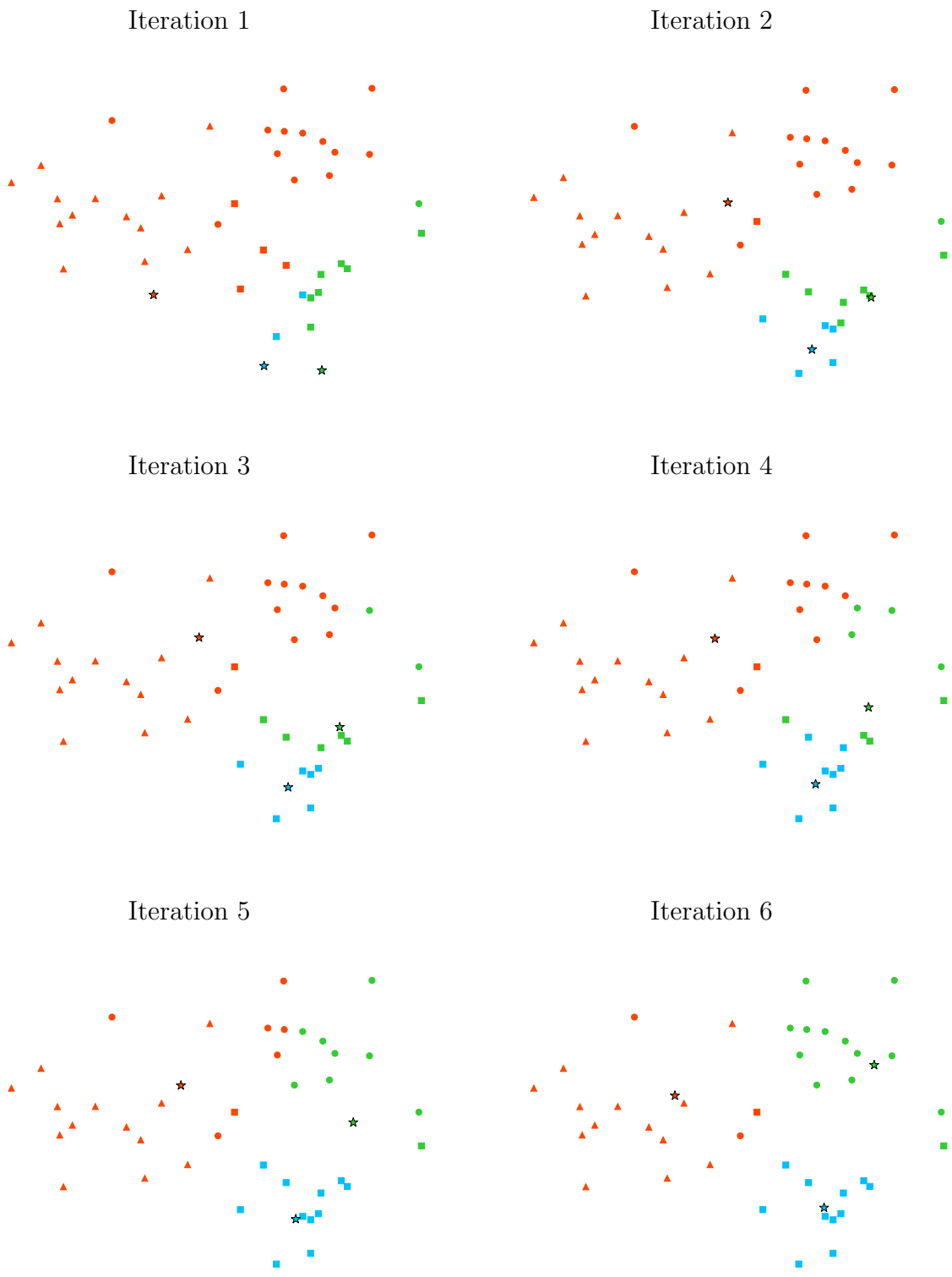

Figure 3: Iterations of Lloyd's algorithm for $k = 3$ clusters which are colored orange, blue and green. The stars indicate the atoms corresponding to each cluster. The original dataset corresponds to three clusters represented by the circles, triangles and squares.

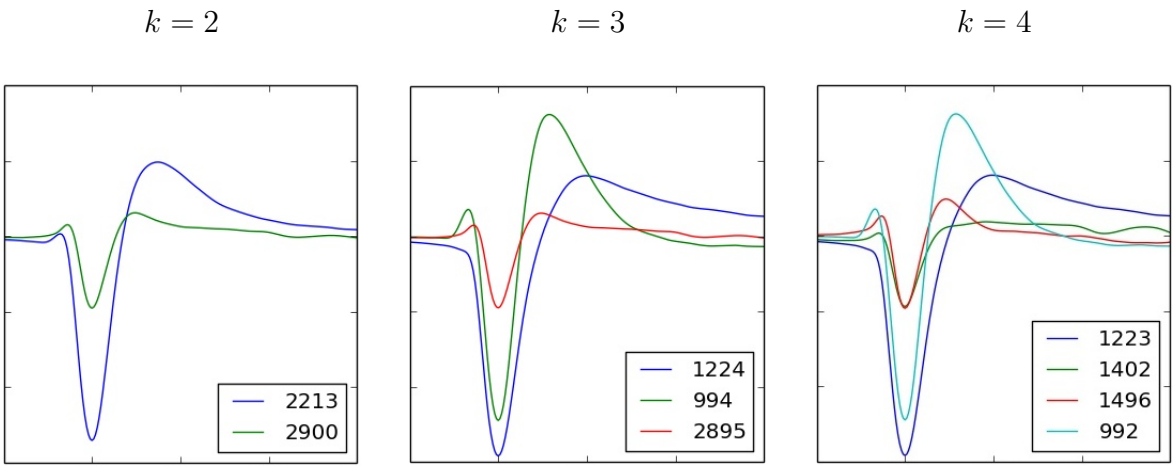

Figure 4: Results of applying k means for different values of k to neural data measured by the Chichilnisky lab at Stanford.

Lloyd's algorithm is a heuristic for the k-means problem that alternates between solving the two subproblems.

Algorithm 2.2 (Lloyd's algorithm)**.** *We initialize the atoms randomly and then alternate between updating the coefficients and the atoms.*

- *The assignment step selects the closest atom to each signal,*

$$c\left(i\right) := \arg\min_{1 \leq j \leq k} \left|\left|x_i - \Phi_j\right|\right|_2 . \tag{9}$$

- *The averaging step computes the mean of each cluster,*

$$\Phi_j := \frac{\sum_{i=1}^{n} \delta\left(c\left(j\right) = i\right)\, x_i}{\sum_{i=1}^{n} \delta\left(c\left(j\right) = i\right)}, \tag{10}$$

  where $\delta\left(c\left(j\right) = i\right)$ equals one if $c\left(j\right) = i$ and zero otherwise.

In practice, Lloyd's algorithm often finds a satisfactory solution to the k-means problem, although for some initializations it might get stuck in a bad local minimum (an atom may end up with no assignments for example). To avoid such situations we can run the method several times, using different random initializations, and select the best solution. Figure 3 illustrates Lloyd's algorithm by showing several iterations.

We end this section describing two applications of k means: one in neuroscience and another in image processing.

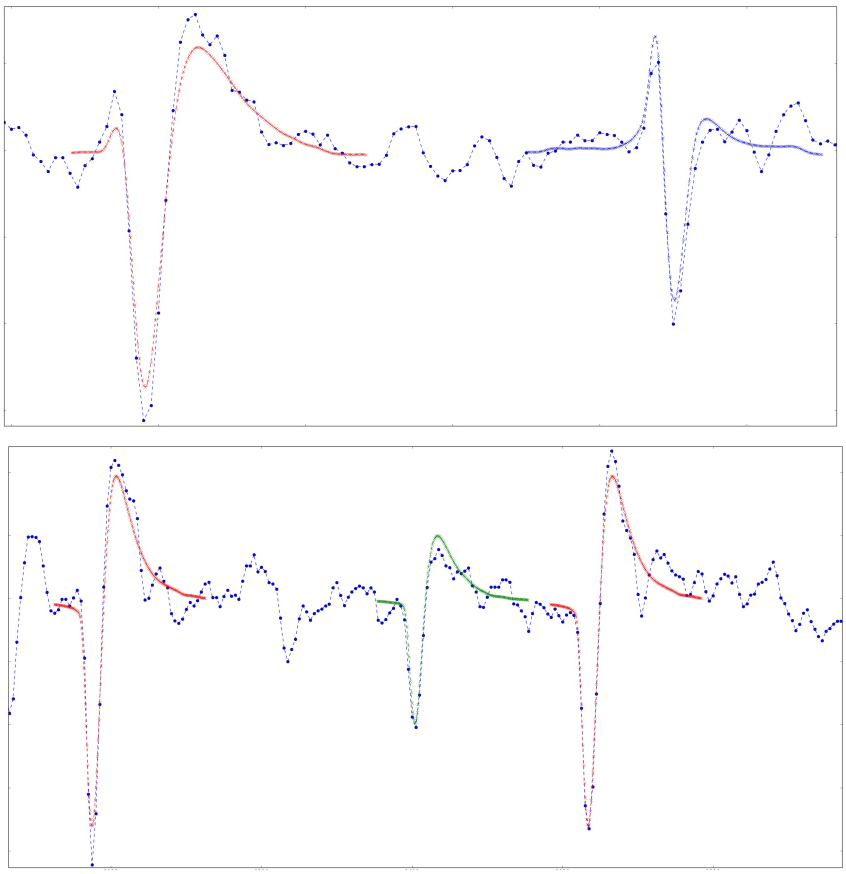

Figure 5: Atoms obtained by solving the k means problem superposed on the original neural data. The data was obtained by the Chichilnisky lab at Stanford.

In neuroscience, data is often collected by placing an array of electric sensors on tissue containing neurons. Each sensors measures the electric activity of several neurons that can be well modeled as sequences of spikes. These spikes have different shapes depending on what neuron generated them. The spike-sorting problem consists of clustering the measured spikes to assign them to the different cells. If we extract all spiking events from the data and align them adequately, we can apply solve the k means problem for these signals to sort the spikes (note that we have to fix the number of neurons k beforehand). Figure 4 shows the results of applying this procedure with different values of k to some real data measured by the Chichilnisky lab at Stanford. The atoms learnt using k means can then be used to obtain a fit to the original data as shown in Figure 5

Figures 6 and 7 show the results of solving the k means problem for the faces dataset from Figure 2 with different values of k. If k is small with respect to the number of different people in the dataset, which is equal to 40, then the atoms learnt by k means correspond to averaging the faces of different people. As we increase k, the atoms become sharper because

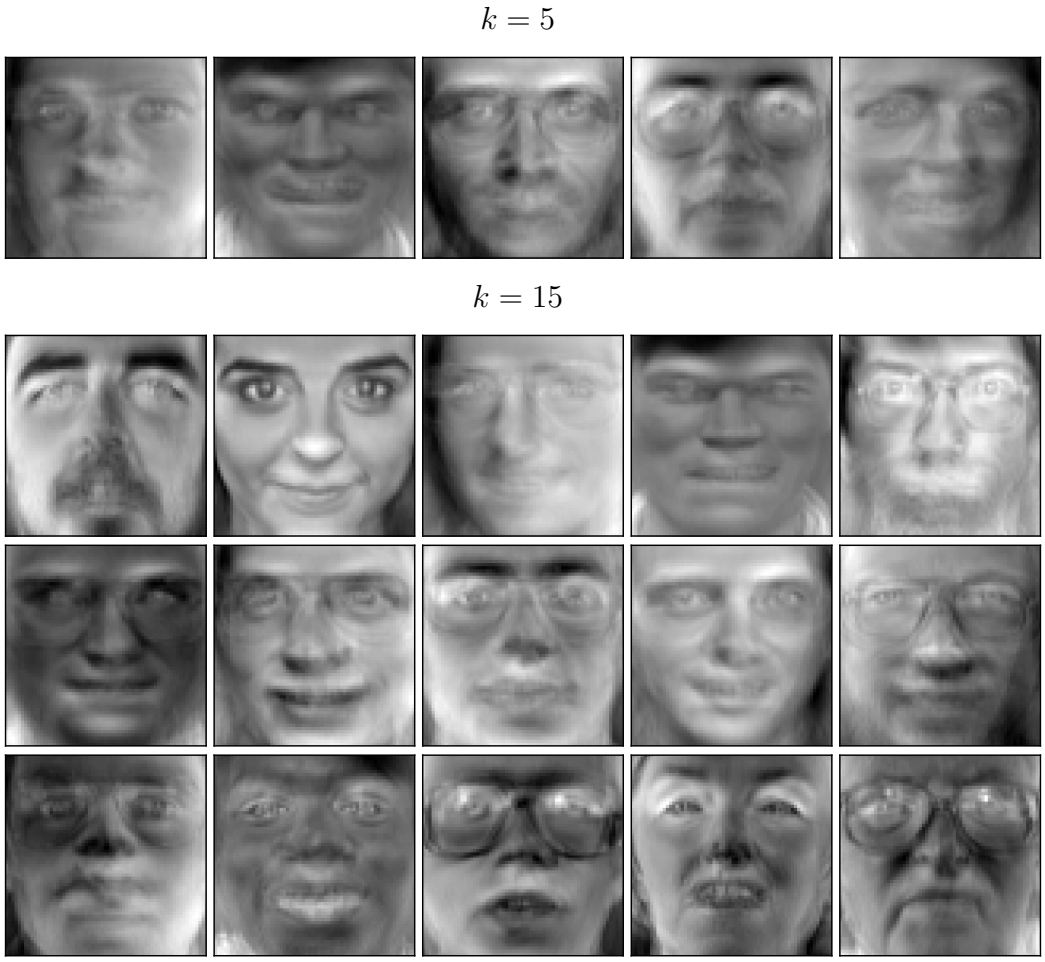

Figure 6: Atoms obtained by solving the k means problem for the faces dataset from Figure 2 with $k = 5$ and $k = 15$.

they are obtained by averaging less images, and some of them correspond to just one person.

3 Principal-component analysis

We already discussed principal-component analysis (PCA) as a dimensionality-reduction technique in Lecture Notes 5. In this section we review the method, providing a geometric and probabilistic interpretation. Then we explain how to learn signal representations using PCA and end by describing an application to collaborative filtering.

$k = 40$

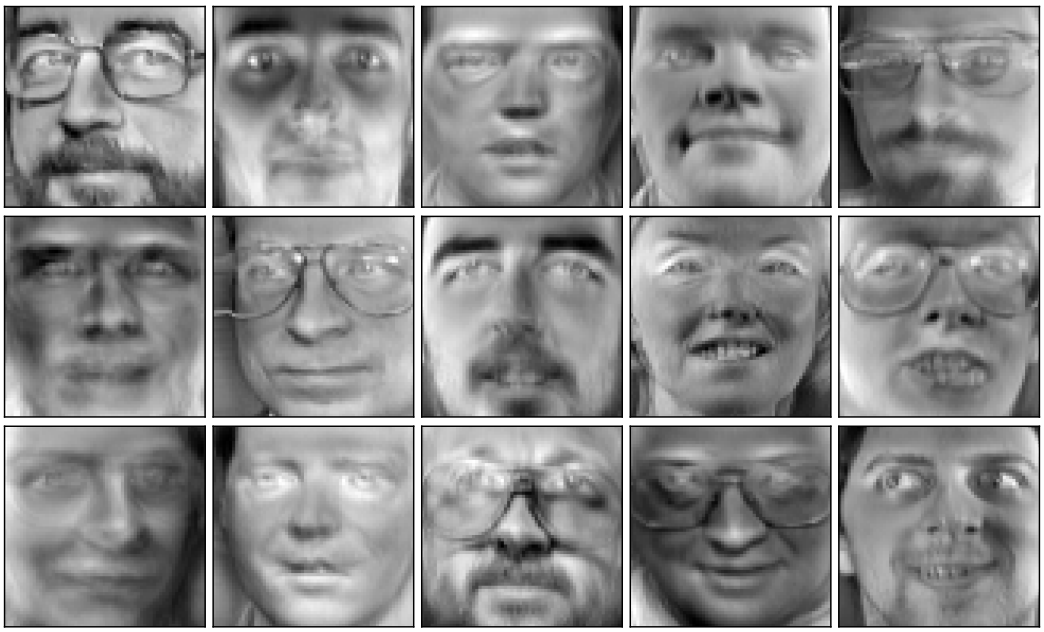

Figure 7: Atoms obtained by solving the k means problem for the faces dataset from Figure 2 with $k = 40$.

3.1 Algorithm

PCA allows to find directions in this space along which the data have a high variation. This is achieved by centering the data and then extracting the singular vectors corresponding to the largest singular values.

Algorithm 3.1 (Principal component analysis)**.** *Given n data vectors $\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n \in \mathbb{R}^d$, we apply the following steps.*

1. *Center the data,*

$$x_i = \tilde{x}_i - \frac{1}{n} \sum_{i=1}^{n} \tilde{x}_i, \qquad 1 \leq i \leq n. \tag{11}$$

2. *Group the centered data in a data matrix $X \in \mathbb{R}^{d \times n}$*

$$X = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}. \tag{12}$$

3. *Compute the singular-value decomposition (SVD) of X and extract the left singular vectors corresponding to the k largest singular values. These are the first k principal components.*

3.2 PCA: Geometric interpretation

Once the data are centered, the energy of the projection of the data points onto different directions in the ambient space reflects the variation of the dataset along those directions. PCA selects the directions that maximize the ℓ_2 norm of the projection and are mutually orthogonal. The sum of the squared ℓ_2 norms of the projection of the centered data $x_1, x_2, \ldots, x_n$ onto a 1D subspace spanned by a unit-norm vector u can be expressed as

$$\sum_{i=1}^{n} \left|\left|\mathcal{P}_{\text{span}(u)}\, x_i\right|\right|_2^2 = \sum_{i=1}^{n} u^T x_i x_i^T u \tag{13}$$

$$= u^T X X^T u \tag{14}$$

$$= \left|\left|X^T u\right|\right|_2^2. \tag{15}$$

If we want to maximize the energy of the projection onto a subspace of dimension k, an option is to choose orthogonal 1D projections sequentially. First we choose a unit vector U_1 that maximizes $\left|\left|X^T u\right|\right|_2^2$ and is consequently the 1D subspace that is better adapted to the data. Then, we choose a second unit vector U_2 *orthogonal* to the first which maximizes $\left|\left|X^T u\right|\right|_2^2$ and hence is the 1D subspace that is better adapted to the data while being in

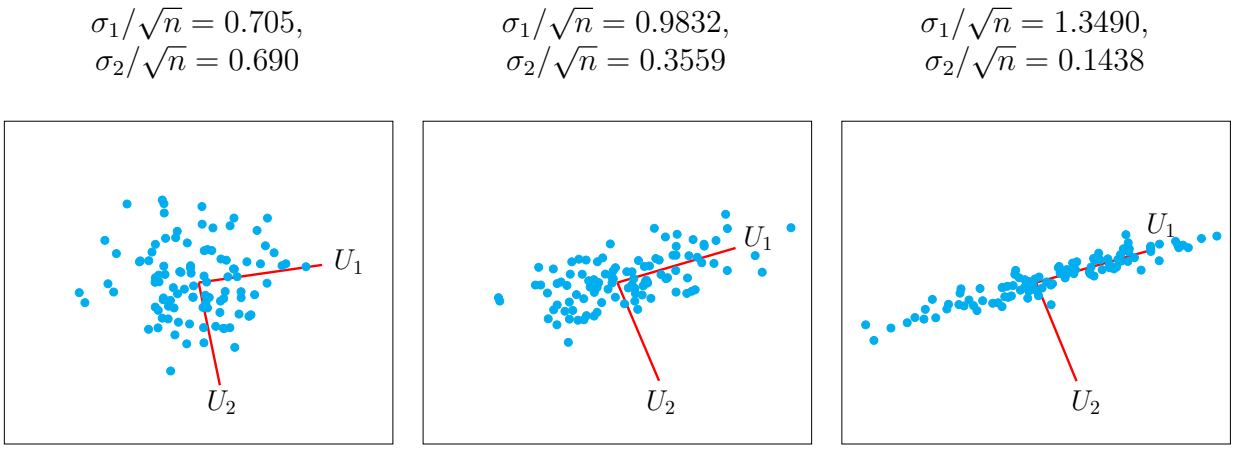

Figure 8: PCA of a dataset with $n = 100$ 2D vectors with different configurations. The two first singular values reflect how much energy is preserved by projecting onto the two first principal components.

the orthogonal complement of U_1. We repeat this procedure until we have k orthogonal directions, which are the first k principal components.

More formally, the left singular vectors $U_1, U_2, \ldots, U_k$ and the corresponding singular values $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_k$, are given by

$$\sigma_1 = \max_{||u||_2=1} \left|\left|X^T u\right|\right|_2, \tag{16}$$

$$U_1 = \arg\max_{||u||_2=1} \left|\left|X^T u\right|\right|_2, \tag{17}$$

$$\sigma_j = \max_{\substack{||u||_2=1 \\ u \perp u_1,\ldots,u_{j-1}}} \left|\left|X^T u\right|\right|_2, \quad 2 \leq j \leq k, \tag{18}$$

$$U_j = \arg\max_{\substack{||u||_2=1 \\ u \perp u_1,\ldots,u_{j-1}}} \left|\left|X^T u\right|\right|_2, \quad 2 \leq j \leq k. \tag{19}$$

This is established in Lemma A.1 of Lecture Notes 5.

Figure 8 provides an example in 2D. Note how each singular value is proportional to the energy that lies in the direction of the corresponding principal component.

PCA is equivalent to choosing the *best* (in terms of ℓ_2 norm) k 1D subspaces following a *greedy* procedure, since at each step we choose the best 1D subspace orthogonal to the previous ones. A natural question to ask is whether this method produces the best k-dimensional subspace. A priori this is not necessarily the case; many greedy algorithms produce suboptimal results. However, in this case the greedy procedure is indeed optimal: the subspace spanned by the first k principal components is the *best* subspace we can choose in terms of the ℓ_2-norm of the projections. The following result is borrowed from Lecture Notes 5 (Theorem 2.5).

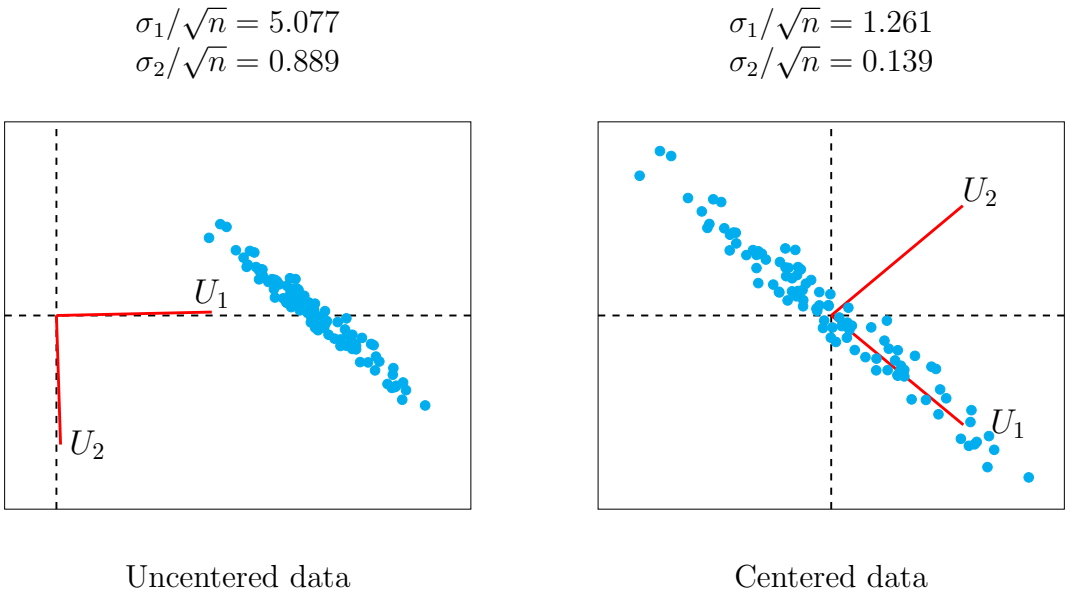

Figure 9: PCA applied to $n = 100$ 2D data points. On the left the data are not centered. As a result the dominant principal component U_1 lies in the direction of the mean of the data and PCA does not reflect the actual structure. Once we center, U_1 becomes aligned with the direction of maximal variation.

Theorem 3.2. *For any matrix $X \in \mathbb{R}^{d \times n}$ with left singular vectors $U_1, U_2, \ldots, U_n$ corresponding to the singular values $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n$, if we fix any $k \leq \min\{d, n\}$*

$$\sum_{i=1}^{n} \left|\left|\mathcal{P}_{\text{span}(U_1, U_2, \ldots, U_k)}\, x_i\right|\right|_2^2 \geq \sum_{i=1}^{n} ||\mathcal{P}_{\mathcal{S}}\, x_i||_2^2 \,, \tag{20}$$

for any subspace $\mathcal{S}$ of dimension k.

Figure 9 illustrates the importance of centering before applying PCA. Theorems **??** and 3.2 still hold if the data are not centered. However, the norm of the projection onto a certain direction no longer reflects the variation of the data. In fact, if the data are concentrated around a point that is far from the origin, the first principal component will tend be aligned in that direction. This makes sense as projecting onto that direction captures more energy. As a result, the principal components do not capture the directions of maximum variation *within* the cloud of data.

3.3 PCA: Probabilistic interpretation

Let us interpret our data, $x_1, x_2, \ldots, x_n$ in $\mathbb{R}^m$, as samples of a random vector $\mathbf{x}$ of dimension m. Recall that we are interested in determining the directions of maximum variation of the

data in ambient space. In probabilistic terms, we want to find the directions in which the data have *higher variance*. The covariance matrix of the data provides this information. In fact, we can use it to determine the variance of the data in any direction.

Lemma 3.3. *Let u be a unit vector,*

$$\operatorname{Var}\left(\boldsymbol{x}^T u\right) = u^T \Sigma_{\boldsymbol{x}} u. \tag{21}$$

Proof.

$$\begin{aligned}
\operatorname{Var}\left(\mathbf{x}^T u\right) &= \mathrm{E}\left(\left(\mathbf{x}^T u\right)^2\right) - \mathrm{E}^2\left(\mathbf{x}^T u\right) && (22)\\
&= \mathrm{E}\left(u\mathbf{x}\mathbf{x}^T u\right) - \mathrm{E}\left(u^T \mathbf{x}\right)\mathrm{E}\left(\mathbf{x}^T u\right) && (23)\\
&= u^T\left(\mathrm{E}\left(\mathbf{x}\mathbf{x}^T\right) - \mathrm{E}\left(\mathbf{x}\right)\mathrm{E}\left(\mathbf{x}\right)^T\right) u && (24)\\
&= u^T \Sigma_{\mathbf{x}} u. && (25)
\end{aligned}$$

□

Of course, if we only have access to samples of the random vector, we do not know the covariance matrix of the vector. However we can approximate it using the empirical covariance matrix.

Definition 3.4 (Empirical covariance matrix)**.** *The empirical covariance of the vectors $x_1, x_2, \ldots, x_n$ in $\mathbb{R}^m$ is equal to*

$$\begin{aligned}
\overline{\Sigma}_n &:= \frac{1}{n}\sum_{i=1}^{n}\left(x_i - \overline{x}_n\right)\left(x_i - \overline{x}_n\right)^T && (26)\\
&= \frac{1}{n} X X^T, && (27)
\end{aligned}$$

where $\overline{x}_n$ is the sample mean, as defined in Definition 1.3 of Lecture Notes 4, and X is the matrix containing the centered data as defined in (12)*.*

If we assume that the mean of the data is zero (i.e. that the data have been centered using the true mean), then the empirical covariance is an unbiased estimator of the true covariance matrix:

$$\begin{aligned}
\mathrm{E}\left(\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i\mathbf{x}_i^T\right) &= \frac{1}{n}\sum_{i=1}^{n}\mathrm{E}\left(\mathbf{x}_i\mathbf{x}_i^T\right) && (28)\\
&= \Sigma_{\mathbf{x}}. && (29)
\end{aligned}$$

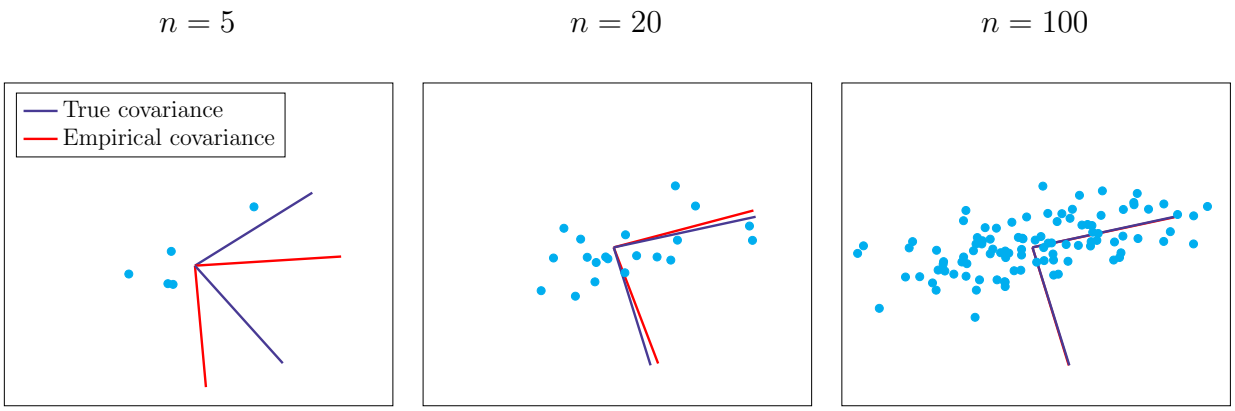

Figure 10: Principal components of n data vectors samples from a 2D Gaussian distribution. The eigenvectors of the covariance matrix of the distribution are also shown.

If the higher moments of the data $\mathrm{E}\left(X_i^2 X_j^2\right)$ and $\mathrm{E}\left(X_i^4\right)$ are finite, by Chebyshev's inequality the entries of the empirical covariance matrix converge to the entries of the true covariance matrix. This means that in the limit

$$\begin{aligned} \mathrm{Var}\left(\mathbf{x}^T u\right) &= u^T \Sigma_{\mathbf{x}} u & (30) \\ &\approx \frac{1}{n} u^T X X^T u & (31) \\ &= \frac{1}{n} \left|\left| X^T u \right|\right|_2^2 & (32) \end{aligned}$$

for any unit-norm vector u. In the limit the principal components correspond to the directions of maximum variance of the underlying random vector. These directions correspond to the eigenvectors of the true covariance matrix, which maximize the quadratic form $u^T \Sigma_{\mathbf{x}} u$. Figure 10 illustrates how the principal components converge to the eigenvectors of $\Sigma_{\mathbf{x}}$.

3.4 PCA as a matrix-approximation method

Let $U \Sigma V^T$ denote the SVD of the data matrix X, which we assume to be centered for simplicity. PCA allows us to obtain a rank k decomposition of the form (2) which is optimal in a certain sense, as shown in the following proposition which is proved in Section A.2 of the appendix.

Proposition 3.5 (Best rank-k approximation)**.** *Let $U \Sigma V^T$ be the SVD of M. We denote by $U_{1:k}$ the matrix that contains the first k left singular vectors of M, $\Sigma_{1:k}$ a $k \times k$ diagonal matrix containing the k largest singular values of X and $V_{1:k}$ the matrix containing the first k right singular vectors. $U_{1:k} \Sigma_{1:k} V_{1:k}^T$ is the best rank-k approximation of M in Frobenius*

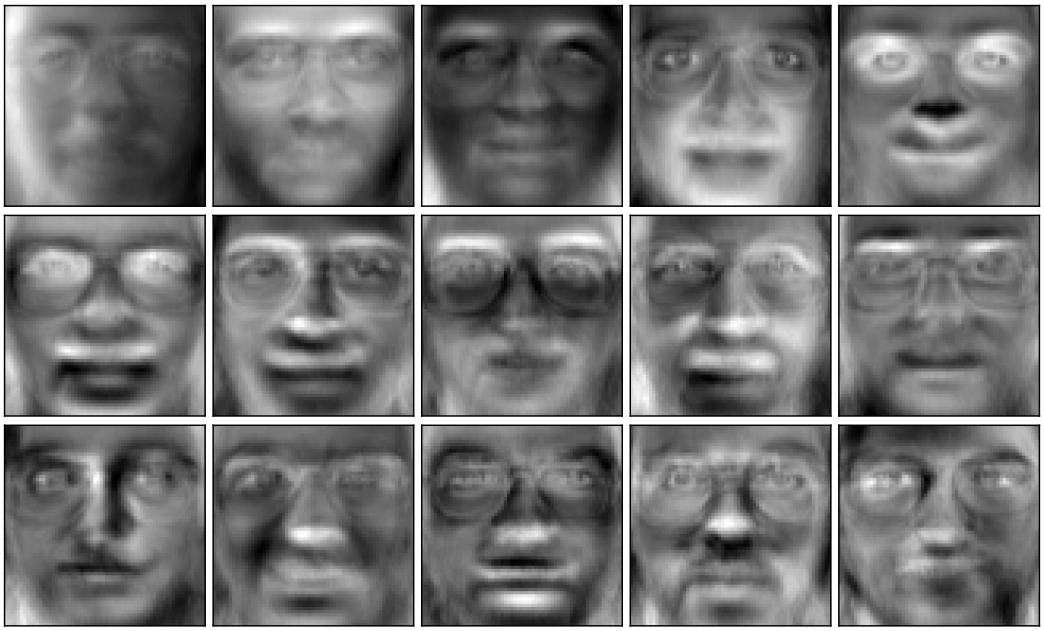

Figure 11: Principal components corresponding to the faces dataset from Figure 2.

norm,

$$U_{1:k}\Sigma_{1:k}V_{1:k}^T = \arg \min_{\left\{\tilde{M} \mid \text{rank}\left(\tilde{M}\right)=k\right\}} \left|\left|M - \tilde{M}\right|\right|_{\text{F}}^2. \tag{33}$$

We can interpret the principal components $\Phi = U_{1:k}$ as orthogonal atoms, whereas $A = \Sigma_{1:k}V_{1:k}^T$ corresponds to the coefficient matrix. Figure 11 shows the result of applying PCA to the faces dataset from Figure 2.

3.5 Collaborative filtering

We now describe an application of PCA to collaborative filtering, where the aim is to pool together information from many users to obtain a model of their behavior. In particular, for movie data, we consider the ratings given by a set of users to a set of movies. If the some of the users have similar tastes, then the ratings will be correlated. PCA allows to uncover this low-rank structure in the data. We demonstrate this through a simple example. Bob,

Molly, Mary and Larry rate the following six movies from 1 to 5,

$$A := \begin{array}{c} \begin{array}{cccc} \text{Bob} & \text{Molly} & \text{Mary} & \text{Larry} \end{array} \\ \begin{pmatrix} 1 & 1 & 5 & 4 \\ 2 & 1 & 4 & 5 \\ 4 & 5 & 2 & 1 \\ 5 & 4 & 2 & 1 \\ 4 & 5 & 1 & 2 \\ 1 & 2 & 5 & 5 \end{pmatrix} \end{array} \begin{array}{l} \text{The Dark Knight} \\ \text{Spiderman 3} \\ \text{Love Actually} \\ \text{Bridget Jones's Diary} \\ \text{Pretty Woman} \\ \text{Superman 2} \end{array} \tag{34}$$

We subtract the average rating,

$$\mu := \frac{1}{n} \sum_{i=1}^{m} \sum_{j=1}^{n} A_{ij}, \tag{35}$$

$$\tag{36}$$

from each entry in the matrix and then compute its singular value decomposition

$$A - \bar{A} = USV^T = U \begin{bmatrix} 7.79 & 0 & 0 & 0 \\ 0 & 1.62 & 0 & 0 \\ 0 & 0 & 1.55 & 0 \\ 0 & 0 & 0 & 0.62 \end{bmatrix} V^T, \tag{37}$$

where

$$\bar{A} := \begin{bmatrix} \mu & \mu & \cdots & \mu \\ \mu & \mu & \cdots & \mu \\ \cdots & \cdots & \cdots & \cdots \\ \mu & \mu & \cdots & \mu \end{bmatrix}. \tag{38}$$

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 the case (for ease of comparison the values of A are shown in brackets):

$$\bar{A} + \sigma_1 U_1 V_1^T = \begin{array}{c} \begin{array}{cccc} \text{Bob} & \text{Molly} & \text{Mary} & \text{Larry} \end{array} \\ \begin{pmatrix} 1.34\,(1) & 1.19\,(1) & 4.66\,(5) & 4.81\,(4) \\ 1.55\,(2) & 1.42\,(1) & 4.45\,(4) & 4.58\,(5) \\ 4.45\,(4) & 4.58\,(5) & 1.55\,(2) & 1.42\,(1) \\ 4.43\,(5) & 4.56\,(4) & 1.57\,(2) & 1.44\,(1) \\ 4.43\,(4) & 4.56\,(5) & 1.57\,(1) & 1.44\,(2) \\ 1.34\,(1) & 1.19\,(2) & 4.66\,(5) & 4.81\,(5) \end{pmatrix} \end{array} \begin{array}{l} \text{The Dark Knight} \\ \text{Spiderman 3} \\ \text{Love Actually} \\ \text{Bridget Jones's Diary} \\ \text{Pretty Woman} \\ \text{Superman 2} \end{array} \tag{39}$$

The first left singular vector is equal to

$$U_1 = \begin{pmatrix} \text{D. Knight} & \text{Spiderman 3} & \text{Love Act.} & \text{B.J.'s Diary} & \text{P. Woman} & \text{Superman 2} \\ -0.45 & -0.39 & 0.39 & 0.39 & 0.39 & -0.45 \end{pmatrix}.$$

It can be interpreted as an atom in the sense that centered scores for each person are proportional to U_1. Alternatively, it can be interpreted as coefficients corresponding to the atoms V_1 below, which allow to cluster the movies into action (+) and romantic (-) movies.

The first right singular vector is equal to

$$V_1 = \begin{pmatrix} \text{Bob} & \text{Molly} & \text{Mary} & \text{Larry} \\ 0.48 & 0.52 & -0.48 & -0.52 \end{pmatrix}. \tag{40}$$

It can be interpreted as an atom in the sense that centered scores for each movie are proportional to V_1. Alternatively, it can be interpreted as coefficients corresponding to the atom U_1, which allow to cluster the users that have similar tastes (Bob and Molly vs Mary and Larry).

This example is obviously very simple, but it illustrates the interest of low-rank models in collaborative filtering. These models reveal the correlation structure of the data, uncovering hidden factors that determine users' preferences and can be used for clustering. Of course, in practice there will be more than one factor.

4 Nonnegative matrix factorization

4.1 Optimization problem

As explained in the previous section, PCA computes the best low-rank approximation to the data matrix in Frobenius norm. However, depending on the application the atoms obtained from the decomposition are not very interpretable. For example, in Figure 11 the atoms may have negative pixels and the coefficients negative values, so it is difficult to interpret them 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

Figure 12: Atoms obtained by applying nonnegative matrix factorization to the faces dataset from Figure 2.

optimization problem,

$$\begin{aligned} \text{minimize} \quad & \left|\left| X - \tilde{\Phi}\ \tilde{A} \right|\right|_{\text{F}}^2 && (41) \\ \text{subject to} \quad & \tilde{\Phi}_{i,j} \geq 0, && (42) \\ & \tilde{A}_{i,j} \geq 0, \qquad \text{for all } i, j && (43) \end{aligned}$$

where $\tilde{\Phi} \in \mathbb{R}^{d \times k}$ and $\tilde{A} \in \mathbb{R}^{k \times n}$ for a fixed k. 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 [2]. Figure 12 shows the atoms obtained by applying this method to the faces dataset from Figure 2. 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.

4.2 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.

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

Computing the singular-value decomposition of the matrix– after subtracting the mean of each entry as in (37)– we determine that the matrix is approximately low rank

$$A - \bar{A} = USV^T = U \begin{bmatrix} 19.32 & 0 & 0 & 0 & & \\ 0 & 14.46 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4.99 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2.77 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.67 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.93 \end{bmatrix} V^T. \tag{44}$$

Unfortunately the singular vectors do not have an intuitive interpretation as in Section **??**. In particular, they do not allow to cluster the words

$$\begin{array}{rcccccccc} & & \text{a} & \text{b} & \text{c} & \text{d} & \text{e} & \text{f} \\ U_1 & = & (-0.51 & -0.40 & -0.54 & -0.11 & -0.38 & -0.38) \\ U_2 & = & (\ 0.19 & -0.45 & -0.19 & -0.69 & -0.2 & -0.46) \\ U_3 & = & (\ 0.14 & -0.27 & -0.09 & -0.58 & -0.69 & -0.29) \end{array} \tag{45}$$

or the articles

$$\begin{array}{rcccccccccc} & & \text{singer} & \text{GDP} & \text{senate} & \text{election} & \text{vote} & \text{stock} & \text{bass} & \text{market} & \text{band} \\ V_1 & = & (-0.38 & 0.05 & 0.4 & 0.27 & 0.4 & 0.17 & -0.52 & 0.14 & -0.38) \\ V_2 & = & (\ 0.16 & -0.46 & 0.33 & 0.15 & 0.38- & 0.49 & 0.1 & -0.47 & 0.12\ ) \\ V_3 & = & (-0.18 & -0.18 & -0.04 & -0.74 & -0.05 & 0.11 & -0.1 & -0.43 & -0.43) \end{array} \tag{46}$$

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

$$M \approx WH, \quad W_{i,j} \geq 0, \qquad 1 \leq i \leq m,\ 1 \leq j \leq k, \tag{47}$$

$$H_{i,j} \geq 0, \qquad 1 \leq i \leq k,\ 1 \leq i \leq n. \tag{48}$$

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.

$$\begin{array}{cccccccccccc} & & \text{singer} & \text{GDP} & \text{senate} & \text{election} & \text{vote} & \text{stock} & \text{bass} & \text{market} & \text{band} & \\ H_1 & = & (\ 0.34 & 0 & 3.73 & 2.54 & 3.67 & 0.52 & 0 & 0.35 & 0.35\ ) & \\ H_2 & = & (\ 0 & 2.21 & 0.21 & 0.45 & 0 & 2.64 & 0.21 & 2.43 & 0.22\ ) & \\ H_3 & = & (\ 3.22 & 0.37 & 0.19 & 0.2 & 0 & 0.12 & 4.13 & 0.13 & 3.43\ ) & \end{array} \tag{49}$$

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)

$$\begin{array}{cccccccc} & & \text{a} & \text{b} & \text{c} & \text{d} & \text{e} & \text{f} \\ W_1 & = & (0.03 & 2.23 & 0 & 0 & 1.59 & 2.24) \\ W_2 & = & (\ 0.1 & 0 & 0.08 & 3.13 & 2.32 & 0\ ) \\ W_3 & = & (2.13 & 0 & 2.22 & 0 & 0 & 0.03) \end{array} \tag{50}$$

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

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

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

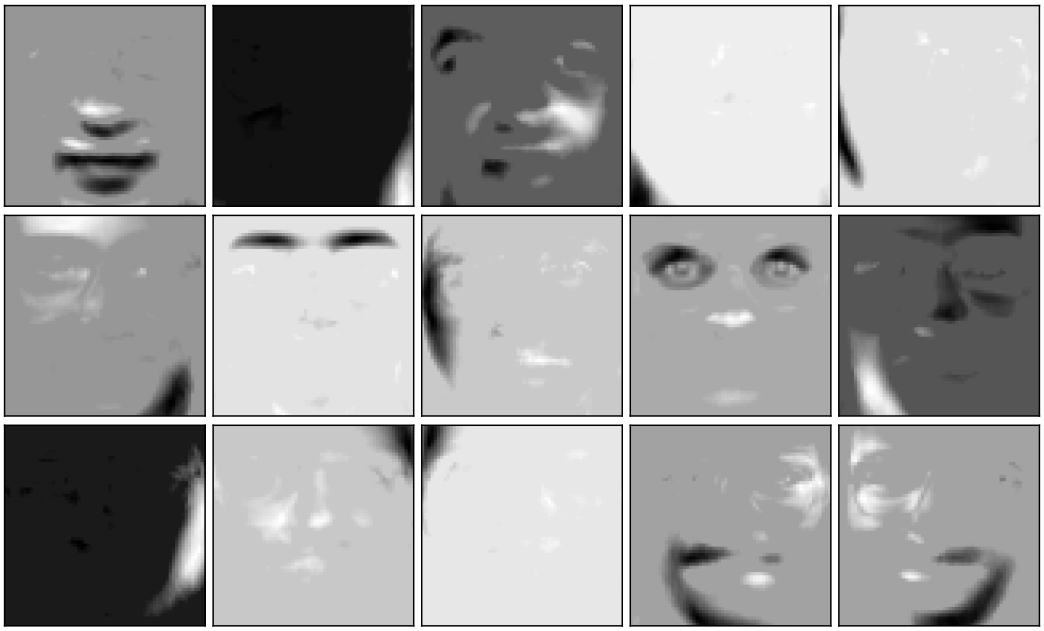

Figure 13: Atoms obtained by applying sparse PCA to the faces dataset from Figure 2.

5 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

$$\text{minimize} \quad \left|\left|X - \tilde{\Phi}\ \tilde{A}\right|\right|_2^2 + \lambda \sum_{i=1}^{k} \left|\left|\tilde{\Phi}_i\right|\right|_1 \tag{51}$$

$$\text{subject to} \quad \left|\left|\tilde{\Phi}_i\right|\right|_2 = 1, \qquad 1 \leq i \leq k. \tag{52}$$

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 13 shows the atoms obtained by applying this method to the faces dataset from Figure 2. The model indeed learns very localized atoms that represent face features.

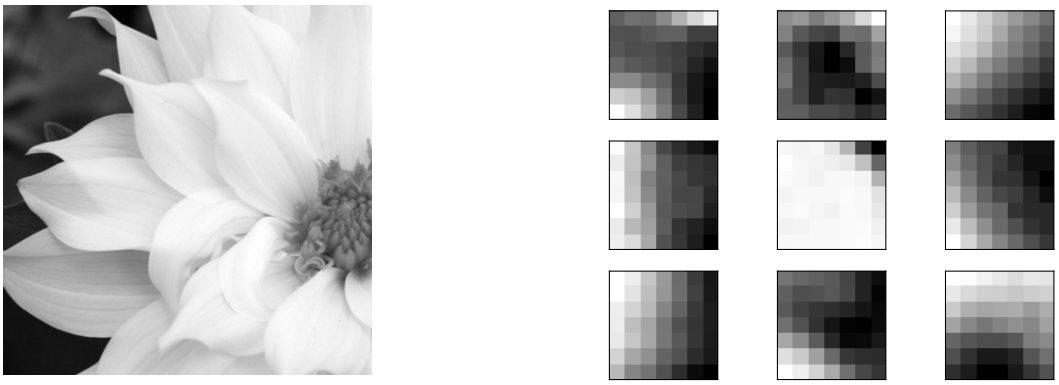

Figure 14: Atoms (right) learnt from patches extracted from a natural image (left).

6 Sparse coding

In Lecture Notes 4 we studied several sparsifying transforms that allow to decompose data into a small number of atoms in order to perform compression or denoising. Dictionary-learning or sparse-coding techniques allow to learn these transforms directly from the data. This is very useful in situations where a dataset with a large number of signals cannot be compactly represented in any predefined dictionary. The aim is to learn a dictionary $\Phi \in \mathbb{R}^{d \times k}$ such that $X \approx \Phi A$, where the matrix of coefficients $A \in \mathbb{R}^{k \times n}$ is very sparse. Following the heuristic that penalizing the ℓ_1 norm promotes sparse solutions, this may be achieved by solving the following optimization program,

$$\min_{\widetilde{\Phi},\, \widetilde{A}} \left|\left| X - \widetilde{\Phi}\widetilde{A} \right|\right|_{\mathrm{F}}^2 + \lambda \left|\left| \widetilde{A} \right|\right|_1 \quad \text{such that } \left|\left| \widetilde{\Phi}_i \right|\right|_2 = 1, \quad 1 \leq i \leq k. \tag{53}$$

Note that the formulation is very similar to sparse PCA, with the crucial difference that we are promoting sparsity in the coefficient matrix, as opposed to in the atoms. An efficient method to compute a local minimum of the nonconvex problem is to apply techniques based on stochastic gradient descent [4].

Figure 14 shows patches learnt from a natural image. The corresponding dictionary can be used to denoise other images quite effectively, as shown in Figure 15. See [3] for other applications of dictionary learning in image processing. Finally, we note that interestingly sparse coding was first proposed in neuroscience, as an explanation of the Gabor-like receptive fields of neurons in the visual cortex [5].

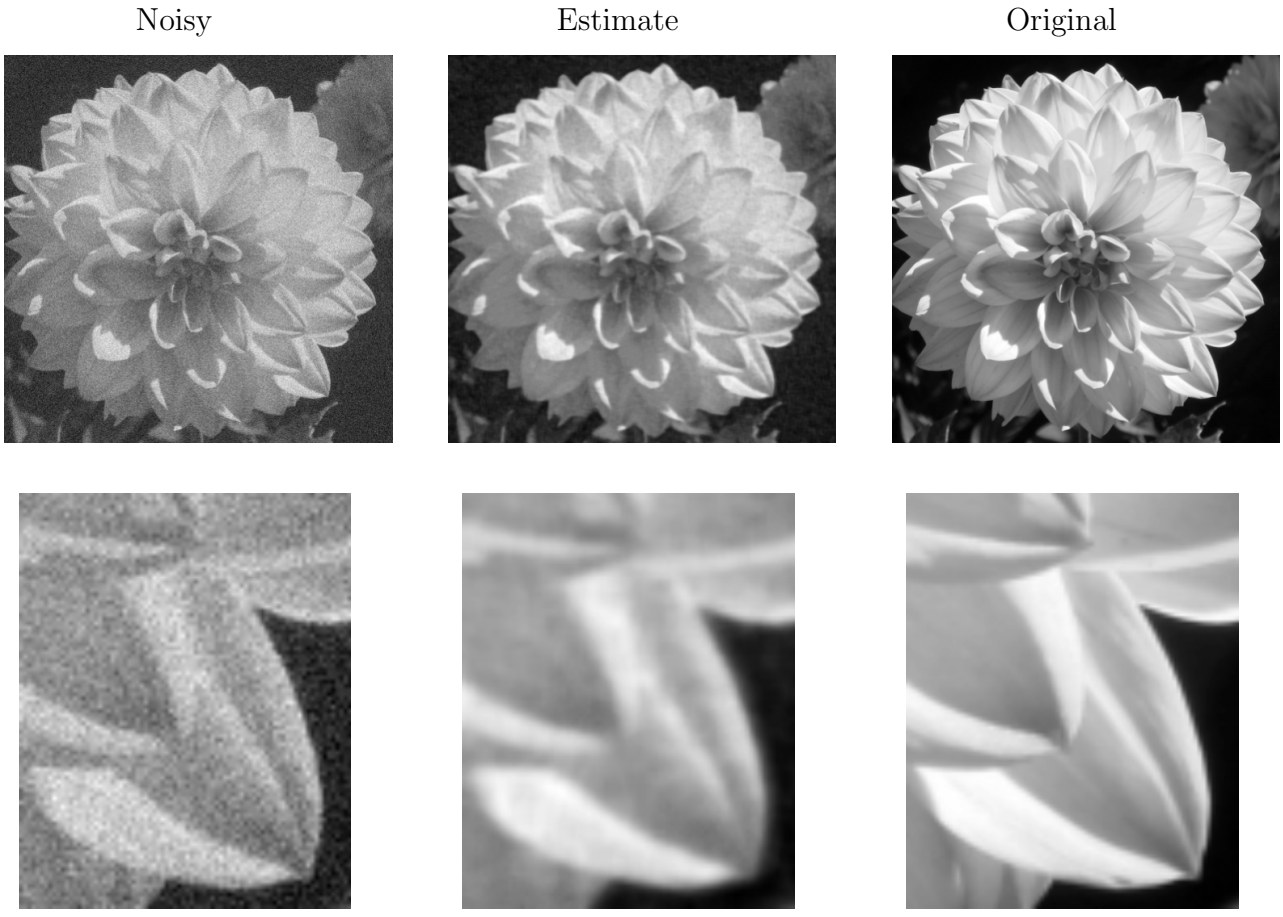

Figure 15: Denoising results using the dictionary learnt from the image shown in Figure 14.

References

The tutorial [3] is an excellent reference on the application of matrix-decomposition techniques in machine learning and image processing. Chapters 7 and 8 of [1] 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] T. Hastie, R. Tibshirani, and M. Wainwright. *Statistical learning with sparsity: the lasso and generalizations*. CRC Press, 2015.

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

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

[4] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *The Journal of Machine Learning Research*, 11:19–60, 2010.

[5] B. A. Olshausen and D. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

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

A Proofs

A.1 Proof of Lemma 2.1

The function

$$f(\phi) = \frac{1}{2}\sum_{i=1}^{n} ||x_i - \phi||_2^2 \tag{54}$$

is convex and its gradient is equal to

$$\nabla f(\phi) = \sum_{i=1}^{n} \phi - x_i \tag{55}$$

$$= n\phi - \sum_{i=1}^{n} x_i. \tag{56}$$

Setting the gradient to zero yields the result.

A.2 Proof of Proposition 3.5

Let M be an arbitrary rank k matrix with singular value decomposition $U_M \Sigma_M V_M^T$. By Pythagoras' Theorem

$$||X - M||_{\mathrm{F}}^2 = \left|\left|X - U_M U_M^T X\right|\right|_{\mathrm{F}}^2 + \left|\left|M - U_M U_M^T X\right|\right|_{\mathrm{F}}^2$$

because the column space of $X - U_M U_M^T X$ is orthogonal to the column space of M. Now, recall that by Theorem 3.2,

$$\left|\left|U_{1:k} U_{1:k}^T X\right|\right|_{\mathrm{F}}^2 \leq \left|\left|U_M U_M^T X\right|\right|_{\mathrm{F}}^2$$

since $U_M U_M^T$ represents a projection onto a k-dimensional subspace. We conclude that

$$\begin{aligned} ||X - M||_{\mathrm{F}}^2 &\geq \left|\left|X - U_M U_M^T X\right|\right|_{\mathrm{F}}^2 \\ &\geq \left|\left|U_{1:k} U_{1:k}^T X\right|\right|_{\mathrm{F}}^2 . \end{aligned}$$