# Lectures 1-2-3

Gianna M. Del Corso

PhD-course Dipartimento di Informatica, Università di Pisa, Italy

January 16-18th 2019

$$Ax = b$$

Let $A \in \mathbb{R}^{n\times n}$ be a nonsingular matrix then for any vector $b \in \mathbb{R}^n$ there is a vector $x \in \mathbb{R}^n$ such that $Ax = b$

- In many applications $A$ is non square or is singular.
- The typical situation is $A \in \mathbb{R}^{m\times n}, m > n$ (overdetermined). The problem $Ax = b$ has in general no solution.
- Let $r = b - Ax$, we want to find vector $x$ such that the residual is made as small as possible.
- If we take the Euclidean norm we get the *linear least square problem*

$$\min_x \|b - Ax\|_2$$

# Least Squares

- With geometrical considerations it is intuitive that the residual should be orthogonal to the column space of $A$, the range space of $A$, denoted by $R(A)$.
- Imposing $r \perp R(A)$ we get $\forall z \in \mathbb{R}^n$

$$0 = (Az)^T(b - Ax) = z^T A^T b - z^T A^T Ax = z^T(A^T b - A^T Ax)$$

- A vector $x \in \mathbb{R}^n$ minimizes the residual norm $\|r\|_2 = \|b - Ax\|_2$ iff $r \perp R(A)$ or equivalently

$$A^T Ax = A^T b$$

$A^T A$ is nonsingular and the *normal equations* have a unique solution iff $A$ has full rank.

# Normal equations

$A$ full rank then $x = (A^T A)^{-1} A^T b$ is the unique solution of the normal equations

$$A^+ = (A^T A)^{-1} A^T, \quad A^+ \in \mathbb{R}^{n \times m}$$

is called the Moore-Penrose pseudoinverse

To solve normal equations

- In general we do not work with pseudoinverse!
- use factorizations instead
  - Cholesky
  - $QR$
  - Singular Value decomposition

Each factorization requires $O(n^2 m)$ to compute the factors and $O(nm)$ for the right hand side

# Cholesky

if $A$ is full rank, $A^T A$ is symmetric and positive definite, then we can compute the Cholesky factorization of $A^T A = LL^T$ with $L$ lower triangular, then

$$x = A^+ b = (LL^T)^{-1} A^T b = L^{-T}(L^{-1}(A^T b))$$

- Compute $L$ such that $A^T A = LL^T$
- Let $y = A^T b$, solve $Lz = y$
- Solve $L^T x = z$

# QR

In situations where it is important to separate informations from noise is better to work with orthogonal vectors
$Q \in \mathbb{R}^{n\times n}$ is orthogonal if $Q^TQ = I$, i.e. the columns of $Q$ are such that $q_i^Tq_j = 0, i \neq j$ and $q_i^Tq_i = 1$.

- $P, Q$ orthogonal $\implies$ $PQ$ orthogonal
- $QQ^T = I$
- $Q_1 \in \mathbb{R}^{n\times k}$ with orthogonal columns, there exists $Q_2 \in \mathbb{R}^{n\times(n-k)}$ such that $Q = [Q_1|Q_2]$ is orthogonal.
- $Q$ orthogonal then $\|Qx\|_2 = \|x\|_2$
- $U \in \mathbb{R}^{m\times m}$, $V \in \mathbb{R}^{n\times n}$ orthogonal $\|UAV\|_2 = \|A\|_2$ and $\|UAV\|_F = \|A\|_F$.

# QR

Let $A = QR$ where $Q = [Q_1|Q_2]$ with $Q_1 \in \mathbb{R}^{m\times n}$ and $R = \begin{bmatrix} R_1 \\ O \end{bmatrix} \in \mathbb{R}^{m\times n}$. $A = Q_1 R_1$, we have

$$\|r\|_2^2 = \|b - QRx\|_2^2 =$$

$$= \left\| \begin{bmatrix} Q_1^T b \\ Q_2^T b \end{bmatrix} - \begin{bmatrix} R_1 x \\ O \end{bmatrix} \right\|_2^2 = \|Q_1^T b - R_1 x\|_2^2 + \|Q_2^T b\|_2^2$$

If $A$ is full rank $x = R_1^{-1} Q_1^T b$ minimizes $\|r\|^2$.

## Backward stability

Let $A \in \mathbb{R}^{m\times n}$, $m \geq n$ be full rank, solving the least square problem with with QR factorization (Householder transformations) the computed solution $\hat{x}$ is the exact least square solution of

$$\min_x \|(A + \Delta A)\hat{x} - (b + \delta b)\|_2$$

where $\|\Delta A\|_F \leq c_1 m\, n\, \varepsilon \|A\|_F, \quad \|\delta b\|_2 \leq c_2 m\, n\, \|b\|_2 + O(\varepsilon^2)$.

# SVD

- QR based method provides an orthogonal basis only for $R(A)$.
- with SVD we have an orthonormal basis also for the row space of $A$.

Let $A \in \mathbb{R}^{m \times n}$, a singular value decomposition is a factorization

$$A = U \Sigma V^T$$

where $U \in \mathbb{R}^{m \times m}$, $V \in \mathbb{R}^{n \times n}$ are orthogonal matrices, $\Sigma = diag(\sigma_1, \sigma_2, \dots, \sigma_{\min(n,m)})$ is a "diagonal" matrix, with $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_{\min(n,m)} \geq 0$.

- $\sigma_i$ are singular values
- $u_i$ are the left singular vectors, and $v_i$ are the right singular vectors, $Av_i = \sigma_i u_i$.

You might be already come across SVD with different names: PCA or Karhunen-Loewe expansion.

# Existance

$$A = \begin{bmatrix} \vert & \vert & \vert & \vert & \vert & \vert & \vert \\ \vert & \vert & \vert & U & \vert & \vert & \vert \\ \vert & \vert & \vert & \vert & \vert & \vert & \vert \end{bmatrix} \begin{bmatrix} \sigma_1 & & & \\ & \ddots & & O \\ & & \ddots & \\ & O & & \sigma_p \\ & & & 0 \end{bmatrix} \begin{bmatrix} \text{——————} \\ \text{——————} \\ \text{———} V^T \text{———} \\ \text{——————} \end{bmatrix}$$

# Solving LSP by SVD

$$m \geq n,\ A = [U_1|U_2] \left[ \begin{array}{c} \Sigma \\ 0 \end{array} \right] V^T,\ U_1 \in \mathbb{R}^{m \times n}$$

$$\|r\|_2^2 = \|b - U\Sigma V^T x\|_2^2 = \left\| \begin{pmatrix} U_1^T b \\ U_2^T b \end{pmatrix} - \begin{pmatrix} \Sigma \\ 0 \end{pmatrix} V^T x \right\|_2^2$$

The least square solution is given by

$$x = V\Sigma^{-1} U_1^T b = \sum_{i=1}^{n} \frac{u_i^T b}{\sigma_i} v_i$$

If $A$ is full rank $\sigma_i > 0$ and the solution is unique.

# Fundamental Subspaces

The SVD gives orthogonal bases of the four fundamental subspaces of a matrix. Assume that $A$ has rank $r$, then

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > \sigma_{r+1} = \cdots = \sigma_n = 0$$

- $R(A)$ The left singular vectors $u_1, u_2, \ldots, u_r$ are an orthogonal basis for $R(A)$. $R(A) = span(u_1, \ldots, u_r)$ and $rank(A) = r$.
- $N(A)$ The right singular vectors $v_{r+1}, v_{r+2}, \ldots, v_n$ are an orthonormal basis for $N(A)$, i.e. $dim(N(A)) = n - r$.
- $R(A^T)$ The right singular vectors $v_1, v_2, \ldots, v_r$ are an orthogonal basis for $R(A^T)$.
- $N(A^T)$ Le left singular vectors $u_{r+1}, u_{r+2}, \ldots, u_m$ are an orthonormal basis for $N(A^T)$.

Changing basis $b' = U^T b, x' = V^T x$ we have

$$b = Ax \Leftrightarrow U^T b = U^T Ax \Leftrightarrow b' = \Sigma x'$$

$A$ reduces to diagonal form when the range is expressed in the basis of columns of $U$ and the domain in the basis of the column of $V$.

# Matrix properties

If $A \in \mathbb{R}^{n\times n}$ and there exists a complete set of eigenvectors then

$$A = S\Lambda S^{-1}.$$

The change of basis corresponds to $b' = S^{-1}b$, $x' = S^{-1}x$ and $b' = \Lambda x'$.

- SVD uses two different basis while eigendecompostion only one
- SVD orthonormal basis while only normal matrices are diagonalizable by an orthogonal matrix
- All the matrices have SVD decomposition while only diagonalizable matrices have an eigendecomposition.

# Useful properties

- $\|A\|_2 = \sigma_1, \|A\|_F = \sqrt{\sigma_1^2 + \sigma_2^2 + \cdots + \sigma_r^2}$
- $0 < \sigma_i = \sqrt{\lambda(A^T A)} = \sqrt{\lambda(AA^T)}$, for $\lambda(A^T A) \neq 0$.
- If $A = A^T$, $\sigma = |\lambda(A)|$
- $A \in \mathbb{R}^{n \times n}$, $|\det(A)| = \prod_{i=1}^n \sigma_i$
- $A = \sum_{i=1}^r \sigma_i u_i v_i^T = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots \sigma_r u_r v_r^T$

# Low rank approximation

## Eckart-Young-Mirsky Theorem

For any $k, 0 \leq k \leq r$ let

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$$

then

$$\|A - A_k\|_2 = \min_{B:rank(B)\leq k} \|A - B\|_2 = \sigma_{k+1}$$

and

$$\|A - A_k\|_F = \min_{B:rank(B)\leq k} \|A - B\|_F = \sqrt{\sigma_{k+1}^2 + \cdots + \sigma_r^2}$$

# Rank-deficient or underdetermined systems

$$A = [U_1 | U_2] \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix}$$

where $\Sigma_1 = diag(\sigma_1, \ldots, \sigma_r) \in \mathbb{R}^{r \times r}$. Setting $y = V^T x$, and $\begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} U_1^T b \\ U_2^T b \end{bmatrix}$

$$\|r\|_2^2 = \|Ax - b\|_2^2 = \left\| \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} - \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \right\|_2^2$$

$$= \|\Sigma_1 y_1 - b_1\|_2^2 + \|b_2\|_2^2$$

The solutions are $y = \begin{bmatrix} \Sigma_1^{-1} b_1 \\ y_2 \end{bmatrix}$, where $y_2$ is arbitrary. The minimal norm solution is given setting $y_2 = 0$, and hence $x = A^+ b$, $A^+ = V \begin{bmatrix} \Sigma_1^{-1} & 0 \\ 0 & 0 \end{bmatrix} U^T$. Same for undetermined systems.

# Computing the SVD

There are stable algorithms for computing the SVD also when $A$ is rank-deficient.

- First reduce the matrix to bidiagonal form with Householder transformations
- Find the SVD decomposition of the bidiagonal matrix

Cost $O(mn^2)$ flops.
The truncated SVD can be computed using a process employing Lanczos method.

References


- Lloyd N. Trefethen, David Bau, III, *Numerical Linear Algebra*, 1997
- J. Demmel, *Applied Numerical Linear Algebra*, 1997.

# Application of SVD

- Image compression
- Text mining
- Face recognition
- Recommender systems

# Image compression with SVD

An $m$-by-$n$ image is an $m$-by-$n$ matrix $X$ where $X_{ij}$ represents the brightness of pixel $(i, j)$. Storing $X$ requires storing the $mn$ entries

$$X = U\Sigma V^T$$

then

$$X_k = U(:, 1:k)\Sigma(1:k, 1:k)V^T(1:k, :)$$

is the best rank-$k$ approximation. To store $X_k$ only $(n+m)k$ values needed.

# Text mining with SVD

- Extract information from large collections of texts
- Examples: find relevant information from the web, from large collection of scientific papers, etc
- Usually we have a query

# The vector space model

- Documents and query are represented as vectors in $\mathbb{R}^m$, $m$ is the size of our vocabulary
- Document preparation
  - Elimination of the *stop words*
  - Stemming: take only the roots of the words
- construction of the term-by-document matrix
- Find document vector close to query

# The vector space model

The term-by-document matrix $A$, is as follows

$$A_{i,j} \neq 0 \quad \text{if document } j\text{−th contains term } i\text{−th}$$

Usually a weighting technique is used

$$A_{i,j} = f_{ij} \log(n/n_i)$$

where $f_{ij}$ is the frequency of term $i$ in doc $j$, and $n_i$ is the number of documents in which term $i$ appears.

# The vector space model

Typically a very sparse matrix

A toy example [Lars Eldén 2007]

| | | | |
|---|---|---|---|
| $T_1$ | eigenvalue | $D_1$ | The Google matrix $P$ is a model of the Internet |
| $T_2$ | England | | |
| $T_3$ | FIFA | $D_2$ | $P_{ij}$ is nonzero if there is a link from Web page $j$ to $i$ |
| $T_4$ | Google | | |
| $T_5$ | Internet | $D_3$ | The Google matrix is used to rank all Web pages |
| $T_6$ | link | | |
| $T_7$ | matrix | $D_4$ | The ranking is done by solving a matrix eigenvalue problem |
| $T_8$ | page | | |
| $T_9$ | rank | $D_5$ | England dropped out of the top 10 in the FIFA ranking |
| $T_{10}$ | Web | | |

# The vector space model

Term-by-Document matrix

$$A = \begin{array}{r c} & \begin{array}{ccccc} D_1 & D_2 & D_3 & D_4 & D_5 \end{array} \\ \begin{array}{r} \text{eigenvalue} \\ \text{England} \\ \text{FIFA} \\ \text{Google} \\ \text{Internet} \\ \text{link} \\ \text{matrix} \\ \text{page} \\ \text{rank} \\ \text{Web} \end{array} & \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

$D_1$: The Google matrix $P$ is a model of the Internet

$D_3$: The Google matrix is used to rank all Web pages

# The vector space model

Term-by-Document matrix

$$A = \begin{array}{r} \\ \text{eigenvalue} \\ \text{England} \\ \text{FIFA} \\ \text{Google} \\ \text{Internet} \\ \text{link} \\ \text{matrix} \\ \text{page} \\ \text{rank} \\ \text{Web} \end{array} \begin{array}{c} \begin{array}{ccccc} D_1 & D_2 & D_3 & D_4 & D_5 \end{array} \\ \left[ \begin{array}{ccccc} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \end{array} \right] \end{array}$$

$D_1$: The Google matrix $P$ is a model of the Internet

$D_3$: The Google matrix is used to rank all Web pages

# Vector space model

Normalizing by column, i.e. dividing by $\|A(:,i)\|_2$ we get

$$A = \begin{bmatrix} 0 & 0 & 0 & 0.5774 & 0 \\ 0 & 0 & 0 & 0 & 0.5774 \\ 0 & 0 & 0 & 0 & 0.5774 \\ 0.5774 & 0 & 0.4472 & 0 & 0 \\ 0.5774 & 0 & 0 & 0 & 0 \\ 0 & 0.5774 & 0 & 0 & 0 \\ 0.5774 & 0 & 0.4472 & 0.5774 & 0 \\ 0 & 0.5774 & 0.4472 & 0 & 0 \\ 0 & 0 & 0.4472 & 0.5774 & 0.5774 \\ 0 & 0.5774 & 0.4472 & 0 & 0 \end{bmatrix}$$

# Query matching

Each query is represented as well as a vector in the vector space. In our example, queries are 10-entries vectors since we have 10 terms. Query= *Ranking of Web pages*, that we represent with the vector

$$q = (0, 0, 0, 0, 0, 0, 0, 1, 1, 1)^T \,,$$

normalizing

$$q = (0, 0, 0, 0, 0, 0, 0, 0.5774, 0.5774, 0.5774)^T$$

Which are the documents close to the query?

# Query matching

A simple idea is to use cosine similarity which measures the cosine of the angle between two vectors.

We compute the cosine similarity as $q^T A(:, i)$, obtaining the following values

$$cos(\theta) = (0, 0.6667, 0.7746, 0.3333, 0.3333)$$

Query= *Ranking of Web pages*,

$D_1$: The Google matrix $P$ is a model of the Internet

$D_2$: $P_{ij}$ is nonzero if there is a link from Web page $j$ to $i$

$D_3$: The Google matrix is used to rank all Web pages

$D_4$: The ranking is done by solving a matrix eigenvalue problem

$D_5$: England dropped out of the top 10 in the FIFA ranking

# Latent Semantic Indexing

We can have poor results or miss important documents

- Choice of the terms used in queries
- polysemy/synonymy
- errors

Texts seems to have latent semantic structure which is destroyed by the variety of terms and synonyms in the text.

This hidden structure can be discovered by means of SVD, i.e. projecting on a “reduced” space that can filter out some of the “noise” of the language: this is the Latent semantic indexing

# Low rank approximation

Using SVD we can project terms and document in a smaller space. Let $A$ be the term-by-document matrix, take SVD $A = U\Sigma V^T$, and choosing a small $k$

$$A \approx A_k = U_k \Sigma_k V_k^T,$$

$U_k \approx$document space, $V_k \approx$ terms space

# Query matching

Compare **q** with documents projected in a $k$ dimensional space, i.e. the columns of $A_k$.
$q^T A_k = q^T U_k \Sigma_k V_k^T = (U_k^T q)^T H_k, H_k = \Sigma_k V_k^T$.

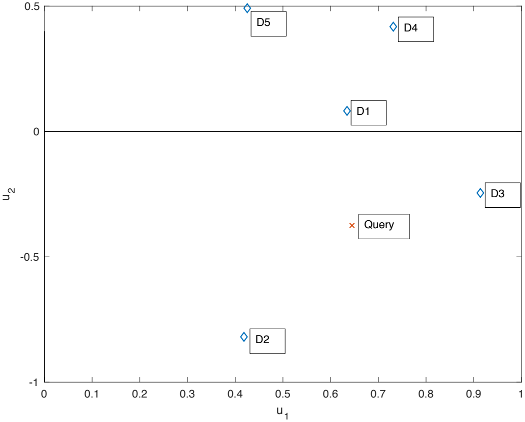
$$\cos(\theta_j) = \frac{q_k^T h_j}{\|q_k\| \, \|h_j\|}, \quad q_k = U_k^T q.$$

We don't need to explicitly compute $A_k$!
Vectors $\mathbf{h}_k$ can be computed once and re-used for different queries.
With $k = 2$ in our toy example the cosines are

$$cos(\theta) = (0.7928, 0.8408, 0.9652, 0.5011, 0.1852).$$

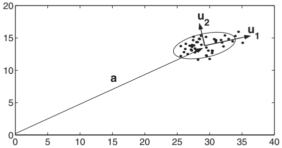
Query= *Ranking of Web pages*
D1: The Google matrix $P$ is a model of the Internet
D2: $P_{ij}$ is nonzero if there is a link from Web page $j$ to $i$
D3: The Google matrix is used to rank all Web pages
D4: The ranking is done by solving a matrix eigenvalue problem
D5: England dropped out of the top 10 in the FIFA ranking

# Low rank approximation



# SVD and Covariance

Let $x_1, \ldots, x_n \in \mathbb{R}^m$ $n$ vectors with zero average,
$a = \frac{1}{n}\sum_{j=1}^{n} x_j = 0$
How are those vectors correlated?
We seek for a direction that best approximates the distribution of the vectors/ we look for the direction of greater variation




find $u$ such that

$$\mu = \max_{\|u\|_2=1} \frac{1}{n}\sum_{j=1}^{n}(u^T x_j)^2.$$

# SVD and Covariance

Introducing the covariance matrix

$$C = \frac{1}{n}\sum_{j=1}^{n} x_j x_j^T,$$

we have that $C = U_c \Lambda_c U_c^T$ and

$$\mu = \max_{\|u\|_2=1}(u^T C u) = \max_{\|y\|_2=1}(y^T \Lambda_c y) = \max_{\|y\|_2=1}\sum_{j=1}^{n}\lambda_j |y_j|^2.$$

We get $\mu = \lambda_1$ and $y = e_1$, meaning that $u = U_c y = U_c e_1 = u_1$ the dominant eigenvector of the covariance matrix.
Let $X = \frac{1}{\sqrt{n}}[x_1, \ldots, x_n]$, we have

$$X = U\Sigma V^T$$

and $U_c = U$ and $\Sigma^2 = \Lambda_c$.

# Eigenfaces

Extract relevant information contained in facial images.

Compare the representation of the faces rather than the faces directly!

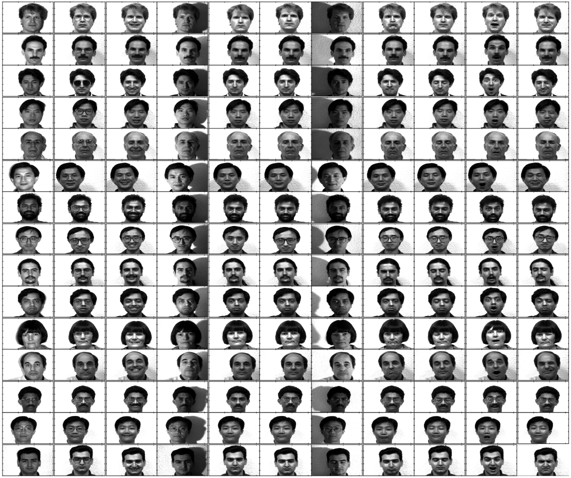
- Do facial images occupy some lower-dimensional subspaces?
- Eigenfaces is a simple algorithm proposed in the '80s

# Eigenfaces

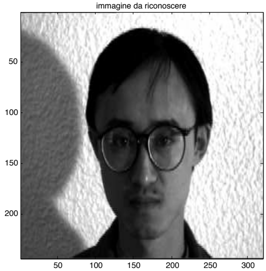
Objective: fast, simple and accurate method

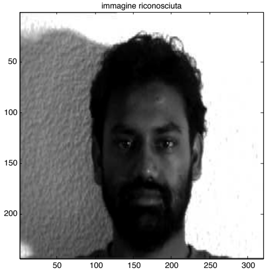
- Previous approach: concentrate on facial characteristics such as shape of eyes, mouth, nose
- The naïve approach is not working! Heads can be flexed, shades and lights, glasses...
- faces should be normalized with respect to position, size and intensity

# Distance between images



# Distance between images




Idea: we need encoding/decoding techniques to reveal the informative content emphasizing local and global features of a face.

# The algorithm

Inizialization: acquire the training set :Faces Space

$$I_1, I_2, \cdots, I_M \quad \text{training set}$$

In our case 15 persons wearing 11 different facial expressions, $M = 165$.

# Centering

$$\Psi = \frac{1}{M}\sum_{i=1}^{M} I_i \quad \text{“average” face}$$

$$\Phi_i = I_i - \Psi \quad \text{difference with the average face}$$

All the “shifted faces” have zero mean.
Vectorize each image and form the matrix

$$X = [\Phi_1(:), \Phi_2(:), \cdots, \Phi_M(:)]$$

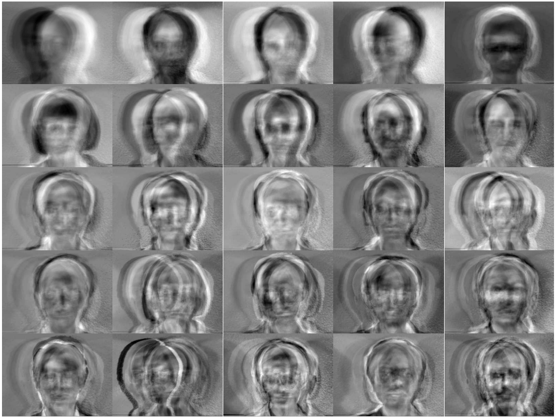
We want to construct and orthogonal basis for the space spanned by the faces

# After centering

The “shifted” training set becomes

# Eigenfaces

The eigenvalues of the covariance matrix $C = XX^T$ are called eigenfaces



# Computational issues

- $C$ has size $N^2 \times N^2$, if images are $N \times N$....it can be a huge full matrix
  In our example, $N^2 = 77760$
- Consider instead $L = A^T A$ which is only $M \times M$, in our example $M = 165$.
- The eigenvectors $C$ are related to those of $L$

$$(A^T A) v_i = \mu_i v_i,$$

$$A \, (A^T A) \, v_i = \mu_i A \, v_i,$$

$$(AA^T) A v_i = \mu_i A \, v_i,$$

substituting $A v_i = u_i$, the unscaled eigenvectors of $C$.
$u_i = \sum_{k=1}^{M} v_{ik} \Phi_k$.

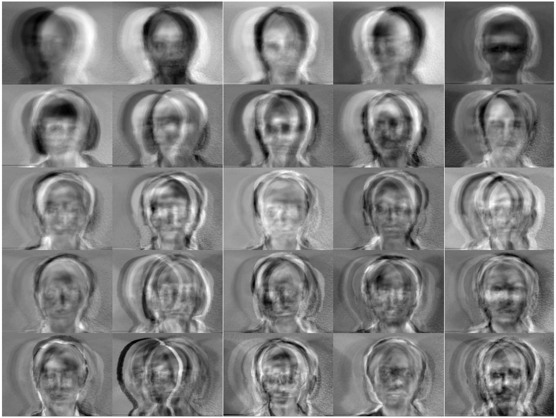
# The algorithm

- Compute the $M$ eigenvectors $v_i$ of $L = A^T A$ and then compute the eigenvectors $u_i$
- Actually, find directly $u_i$ and $v_i$ with SVD of $A$!!
- $$A = U \Sigma V^H,$$

  and we can even consider the reduced SVD. $U \in \mathbb{R}^{N^2 \times M}, \Sigma \in \mathbb{R}^{M \times M}, V \in \mathbb{R}^{M \times M}$.

# The algorithm

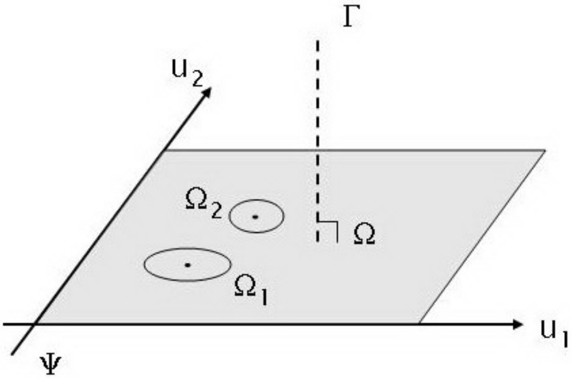
The columns of $U$ are the Eigenfaces



# The algorithm

We can consider only $k << M$ eigenfaces corresponding to the dominant $k$ singular values $\sigma_i$!

span$\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k\}$ is the face space



# Facial recognition

$\Omega_i = U(:, 1:k)^T \, \Phi_i$ is the projection of the $i$-th face in the reduced face space.

Recognition: Given a query image $I_Q$ not in our training set,

- Compute the "shifted face" $\Gamma = I_Q - \Psi$
- Project $\Gamma$ on the space of the faces, we get $\Omega = U(:, 1:k)^T \, \Gamma$.

# Classifier

Given $\Omega$ how can I find its closest image $I_i$?

- Compute the distance to all the projected images
  $\varepsilon_i = \|\Omega - \Omega_i\|^2$
- If $\min_i \varepsilon_i < \theta_\varepsilon$ then $I_Q$ is similar to $I_j$, $j : \min_i \varepsilon_i = \varepsilon_j$
- If, for every $i$, $\varepsilon_i > \theta_\varepsilon$, the image is classified as "unknown"

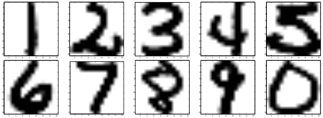
# Summary

Recap:

- Collect a set of $M$ faces. More effective if we have a number of images for each person, with variation in expression and lighting
- Build the matrix with the vectorized images as columns, and subtract to it the average face
- Compute reduced SVD of $A$ taking $k << M$ terms
- The $k$ left singular vectors represent the eigenfaces
- Project each image into the face space
- If $\min_i \|\Omega - \Omega_i\|^2$ is sufficiently small we recognize the query image.

# Classification

A different application: Handwritten digits



- Assume we have low resolution $s \times s$ images of handwritten digits
- Vectorizing all the images corresponding to the same digit and stacking them, we have for each digit $d$ a matrix $A^{(d)} \in \mathbb{R}^{m \times n_d}$, $m = s^2$, $m < n_d$.
- We expect $rank(A^{(d)}) < m$, since otherwise the subspaces of the different digits would intersect (all subsets of $R^{s^2}$)

$$A^{(d)} = \sum_{i=1}^{m} \sigma_i u_i v_i^T$$

hence $a_j^{(d)} = \sum_{i=1}^{m} (\sigma_i v_{ij}) u_i$.

# Classification

If an unknown digit can be better approximated in one particular basis (say the basis of 3's) of singular images than in the other classes, then it is likely to be a 3.

Let $z$ be the unknown digit, for each $d = 0, 1, \dots, 9$ compute the least square solution of

$$\min_{\alpha^{(d)}} \|z - U_k^{(d)} \alpha^{(d)}\|_2$$

that is $\alpha^{(d)} = (U_k^{(d)})^T z$, the solution is obtained projecting $z$ onto the image space of each digit.

$$\min_{\alpha^{(d)}} \|z - U_k^{(d)} \alpha^{(d)}\|_2 = \|(I - U_k^{(d)} (U_k^{(d)})^T) z\|_2.$$

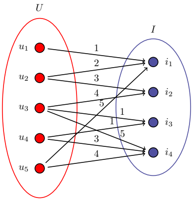
# Recommender systems

“A recommender system is a system which seeks to predict the "rating" or "preference" a user would give to an item.”

Latent factor models: Explain the ratings by characterizing both items and users on a few factors inferred from the ratings patterns G.

# Recommender systems

- Product-by-user matrix instead of the term-by document




Associate the Utility matrix

$$A = \begin{bmatrix} 1 & ? & ? & ? \\ 2 & 3 & ? & ? \\ ? & 4 & 1 & 5 \\ ? & ? & 1 & 3 \\ 5 & ? & ? & 4 \end{bmatrix}$$

The goal of a recommender system is to predict some of the ?

# A Toy example

$$
A = \begin{array}{r} \\ \text{Alice} \\ \text{Bob} \\ \text{Carol} \\ \text{David} \end{array}
\begin{array}{c} \begin{array}{ccccccc} HP1 & HP2 & HP3 & TW & SW1 & SW2 & SW3 \end{array} \\
\begin{bmatrix} 4 & ? & 5 & 1 & ? & ? & 1 \\ 5 & 5 & 4 & ? & ? & ? & ? \\ ? & ? & ? & 2 & 4 & 5 & ? \\ ? & 3 & ? & ? & ? & ? & 3 \end{bmatrix} \end{array}
$$

Would Alice like SW2?

# A Toy example

Fill in all of the empty cells with the average rating for that movie

$$\tilde{A} = \begin{array}{r} \\ \text{Alice} \\ \text{Bob} \\ \text{Carol} \\ \text{David} \end{array} \begin{array}{c} \begin{array}{ccccccc} HP1 & HP2 & HP3 & TW & SW1 & SW2 & SW3 \end{array} \\ \begin{bmatrix} 4 & 4 & 5 & 1 & 4 & 5 & 1 \\ 5 & 5 & 4 & 1.5 & 4 & 5 & 2 \\ 4.5 & 4 & 4.5 & 2 & 4 & 5 & 2 \\ 4.5 & 3 & 4.5 & 1.5 & 4 & 5 & 3 \end{bmatrix} \end{array}$$

Compute SVD of $\tilde{A}$

# A Toy example

Looking at the singular values of $A$, we can approximate $A$ on the ? with what is predicted by the $A_k$.

$$A_1 = \begin{bmatrix} 4.3853 & 3.9052 & 4.3694 & 1.4632 & 3.8910 & 4.8638 & 1.9473 \\ 4.6798 & 4.1673 & 4.6628 & 1.5615 & 4.1523 & 5.1903 & 2.0780 \\ 4.5378 & 4.0410 & 4.5214 & 1.5141 & 4.0263 & 5.0329 & 2.0150 \\ 4.4179 & 3.9341 & 4.4018 & 1.4741 & 3.9199 & 4.8998 & 1.9617 \end{bmatrix}$$

# A Toy example

Substituting in $A$ the unknown values with the predicted ones (rounded) we get

$$\tilde{A} = \begin{array}{r c} & \begin{array}{ccccccc} HP1 & HP2 & HP3 & TW & SW1 & SW2 & SW3 \end{array} \\ \begin{array}{r} \text{Alice} \\ \text{Bob} \\ \text{Carol} \\ \text{David} \end{array} & \begin{bmatrix} 4 & 4 & 5 & 1 & 4 & 5 & 1 \\ 5 & 5 & 4 & 2 & 4 & 5 & 2 \\ 5 & 4 & 5 & 2 & 4 & 5 & 2 \\ 4 & 3 & 4 & 1 & 4 & 5 & 3 \end{bmatrix} \end{array}$$

# A Toy example

Substituting in $A$ the unknown values with the predicted ones (rounded) we get

$$\tilde{A} = \begin{array}{r c} & \begin{array}{ccccccc} HP1 & HP2 & HP3 & TW & SW1 & SW2 & SW3 \end{array} \\ \begin{array}{r} \text{Alice} \\ \text{Bob} \\ \text{Carol} \\ \text{David} \end{array} & \begin{bmatrix} 4 & 4 & 5 & 1 & 4 & 5 & 1 \\ 5 & 5 & 4 & 2 & 4 & 5 & 2 \\ 5 & 4 & 5 & 2 & 4 & 5 & 2 \\ 4 & 3 & 4 & 1 & 4 & 5 & 3 \end{bmatrix} \end{array}$$

- Q:Would Alice like SW2?

# A Toy example

Substituting in $A$ the unknown values with the predicted ones (rounded) we get

$$\tilde{A} = \begin{array}{r} \\ \text{Alice} \\ \text{Bob} \\ \text{Carol} \\ \text{David} \end{array} \begin{array}{c} \begin{array}{ccccccc} HP1 & HP2 & HP3 & TW & SW1 & SW2 & SW3 \end{array} \\ \begin{bmatrix} 4 & 4 & 5 & 1 & 4 & 5 & 1 \\ 5 & 5 & 4 & 2 & 4 & 5 & 2 \\ 5 & 4 & 5 & 2 & 4 & 5 & 2 \\ 4 & 3 & 4 & 1 & 4 & 5 & 3 \end{bmatrix} \end{array}$$

- Q:Would Alice like SW2?
- A:Mostly likely, yes! The value predicted is 5

# References


- N. Muller, L.Magaia, B.M. Herbst. *Singular Value Decomposition, Eigenfaces, and 3D Reconstructions*. SIAM Review 2004.
- Lars Eldén. *Matrix Methods in Data Mining and Pattern Recognition* 2007

# Other factorizations

We seek to approximatelly factor data matrix $A \in \mathbb{R}^{m \times n}$ as

$$A \approx LMR, \quad L \in \mathbb{R}^{m \times r}, M \in \mathbb{R}^{r \times r}, R \in \mathbb{R}^{r \times n},$$

- symmetric eidecomposition $A = Q \Lambda Q^T$
- Singular value decomposition $A = U \Sigma V^T$
- Pivoted QR method, $A\Pi = QR$, $r_{11} \geq r_{22} \geq \cdots \geq r_{nn}$
- Interpolative decomposition $A\Pi \approx C[I|T]$, where $C$ is a subset of the columns of $A$, and $|t_{ij}| \leq 2$.
- CUR decomposition $A \approx CUR$ where $C$ and $R$ are subsets of the columns and rows of $A$.

ID and CUR are easier to interpret because are expressed in terms of factors in the original data set.

# CUR factorization

Find $C, U, R$, such that $\min \|A - CUR\|_F$

- $C$ is a subset of the columns of $A$
- $R$ is a subset of the rows of $A$
- $U$ is a small rank matrix

Motivations

$$\left( \quad \text{user-by-movie} \quad \right) = \left( \quad \text{users} \quad \right) U \left( \text{ movies } \right)$$

- $C$ contains the most "important" users
- $R$ contains the most "important" movies

Sparsity is preserved

Fundamental questions

- Which columns of $A$ should go in $C$? Which rows of $A$ should go in $R$?
- Given $C$ and $R$, how construct the best $U$?

# Leverage scores:

Starting from $A = U\Sigma V^T$ compute the row leverage score

$$\ell_{r,j} = \|U(j,:)\|_2$$

$R$ is composed by the rows of $A$ with the highest leverage score

To rank the importance of the columns, take the 2-norm of each row of $V$:

$$\ell_{c,j} = \|V(j,:)\|_2$$

$C$ is composed by the columns of $A$ with the highest leverage score
We can use some some random selection strategy based on probability distribution $\{\ell_{r,j} / \sum_{j=1}^{m} \ell_{r,j}^2\}$.

# Constructing $U$

Once constructed $C = A(:, q)$ and $R = A(p, :)$,

- $U = (A(p, q))^{-1}$. This choice recovers perfectly entries of $A$, since
$$(CUR)(p, q) = A(p, q).$$
- $U = C^{+}AR^{+}$, optimal choice for the Frobenius norm

Given $k < rank(A)$, and $\varepsilon > 0$, find
$C \in \mathbb{R}^{m \times c}, R \in \mathbb{R}^{r \times n}, U \in \mathbb{R}^{c \times r}$ such that

$$\|A - CUR\|_F^2 \leq (1+\varepsilon)\|A - A_k\|_F^2,$$

with $c, r$ and $rank(U)$ being as small as possible

Not always possible to have $c, r = k$.

## Lower bound

For any $A$ and CUR factorization such that

$$\|A - CUR\|_F^2 \leq (1+\varepsilon)\|A - A_k\|_F^2.$$

Then for any $k \geq 1$ and for any $\varepsilon < 1/3$,

$$c = \Omega(k/\varepsilon)$$

$$r = \Omega(k/\varepsilon)$$

and

$$rank(U) \geq k/2.$$

Use randomizaed algorithm to have faster algorithms than SVD
Tipically $c, r = O(k/eps)$,

# References


- Drineas, Kannan, FOCS 2003
- Drineas, Kannan, , Mahoney, SIAM J on Computing 2006.

# NMF

If $A \in \mathbb{R}_+^{m\times n}$ we seek for two nonnegative factors $W \in \mathbb{R}^{m\times k}$ an $H \in \mathbb{R}^{k\times n}$ such that

$$A \approx WH.$$

Each column of $A$ is a weigthed sum of the columns of $W$ with positive weights.

Example [Bindel 2018]

| Meaning of columns of $A$ | Meaning of columns of $W$ |
|---|---|
| Word in documents | word in topics |
| images of faces | images of facial features |
| connection of friends | communities |

More difficult to compute than SVD and with some problems

- Choice of $k$
- the optimization problem $\min_{W\geq 0, H\geq 0} \frac{1}{2}\|A - WH\|_F^2$ is non-convex
- NMF is not incremental

# NMF: Some algorithms

The problem is non convex $\implies$ many local minima, in general an iterative approach

- Coordinate descent method. $R = A - WH$, compute $w_{ij} = w_{ij} + s \geq 0$ where $s$ minimizes the quadratic form

$$\frac{1}{2}\|A - (W + se_i e_j^T)H\|_F^2$$

- ALS: Given $A$ and $W_i$ find

$$H_i = \operatorname*{argmin}_{X \in \mathbb{R}^{k \times n}} \|A - W_i X\|_F^2, \quad \text{make } H_i \geq 0$$

$$W_{i+1} = \operatorname*{argmin}_{X \in \mathbb{R}^{m \times k}} \|A - XH_i\|_F^2 \quad \text{make } W_{i+1} \geq 0.$$

- ANLS:

$$W_i = \operatorname*{argmin}_{X \geq 0} \|A - XH_i\|_F^2, H_{i+1} = \operatorname*{argmin}_{X \geq 0} \|A - W_i X\|_F^2.$$

Independent convex problems but with nonnegative constraints.

# ANLS

Every limit point generated from the ANLS framework is a stationary point for the non-convex original problem

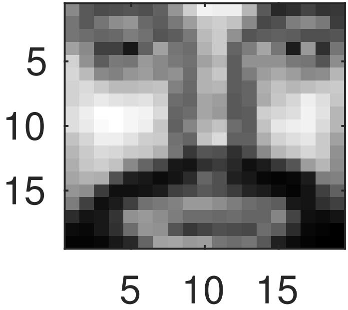
To solve the least square problems inside the ANLS we can use methods such as

- Active-set
- the projected gradient
- the projected quasi-Newton
- greedy coordinate descent

We can solve the problem also with normal equations

# NMF: Facial Features extractions

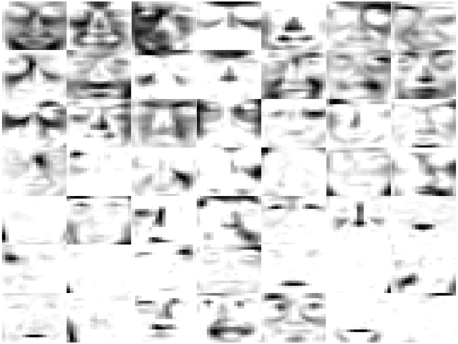
As before a dataset of facial images




We compute NMF of the matrix, $A \approx WH$

# NMF: Facial Features extractions

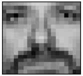
If we reshape and plot the columns of $W$

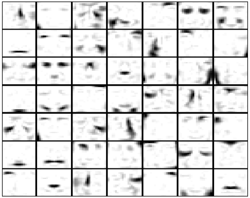


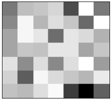
We can identify facial features such as eyes, mounts, mustaches, etc.

# NMF: Facial Features extractions

$$\underbrace{X(:,j)}_{j\text{th facial image}} \approx \sum_{k=1}^{r} \underbrace{W(:,k)}_{\text{facial features}} \underbrace{H(k,j)}_{\substack{\text{importance of features}\\ \text{in } j\text{th image}}} = \underbrace{WH(:,j)}_{\substack{\text{approximation}\\ \text{of } j\text{th image}}} .$$







[Gillis 2014]

# NMF for clustering

If $A$ is symmetric as when $a_{ij}$ represent similarity between item $i$ and item $j$

If $X = (x_1, \dots, x_n)$ is the data matrix, the $A = X^T X$ is a similarity matrix.

A more used similarity measure is the following

$$e_{ij} = exp(-\frac{\|x_i - x_j\|^2}{\mu\sigma}), \quad \mu = \max_{ij} \|x_i - x_j\|^2.$$

$$a_{ij} = d_i^{-1/2} e_{ij} d_j^{-1/2}, \quad d_i = \sum_{j=1}^{n} e_{ij}.$$

- $A$ is invariant under rotations or change of scale.

# NMF for clustering

The symmetric NMF problem is

$$\min_{W \geq 0} \|A - WW^T\|_F^2$$

can be solver with a penalty technique

$$\min_{W,H \geq 0} \|A - WH\|_F^2 + \lambda \|W - H^T\|_F^2.$$

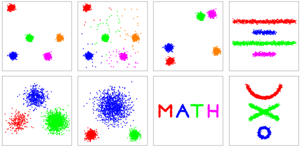
In the ideal case

$$A = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}, \quad w = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}.$$

# NMF for clustering

To extract clusters:

- normalize the rows of $W$ with the infinity norm, $\tilde{W} = WD^{-1}, D = diag(\|W(i,:)\|_\infty)$
- $B_{ij} = 0$ if $\tilde{W}_{ij} < 1$, $B_{ij} = 1$ if $\tilde{W}_{ij} \geq 1$, element $i$ is set in cluster $j$.
  - if $W(i,:) = 0$ element $i$ cannot be assigned to any cluster (very unlikely)
  - $\sum_{j=1}^{N} B(i,j) \neq 1$ element $i$ can be assigned to more than a cluster (very unlikely)
  - If all the columns of $B$ have at least a nonzero entry then the algorithm produced the correct number of clusters.
  - if some columns of $B$ are zero, the algorithm produced a lower number of clusters



# References


- Lars Eldén *Matrix Methods in Data Mining and Pattern Recognition* 2007
- N.Gillis *The Why and How of Nonnegative Matrix Factorization*, 2014. [arXiv]