Machine Learning

Lecture 10: Dimensionality Reduction & Matrix Factorization

Prof. Dr. Stephan Günnemann
Aleksandar Bojchevski

Data Analytics and Machine Learning Group
Technical University of Munich

Winter term 2020/2021

Roadmap

- Chapter: Dimensionality Reduction & Matrix Factorization

Introduction: Unsupervised Learning (I)

- Supervised learning aims to map inputs to targets with $y = f(\boldsymbol{x})$, or in a probabilistic framework it models $p(y|\boldsymbol{x})$
- Unsupervised learning can be seen as modelling $p(\boldsymbol{x})$

features

- We are trying to find the (hidden / latent) structure in the data
  - e.g. find a latent distribution $p(\boldsymbol{z})$ and a generative transformation $p(\boldsymbol{x} \mid \boldsymbol{z})$ we can then obtain $p(\boldsymbol{x}) = \int p(\boldsymbol{x} \mid \boldsymbol{z})\, p(\boldsymbol{z})\, d\boldsymbol{z}$
  - latent $\boldsymbol{z}$ usually unknown and has to be estimated

for each i

$Z_i \sim P(Z)$

$X_i \sim P(X|Z_i)$

- Examples:

cluster indicator

  - Clustering: the cluster label is the latent state
  - Anomaly detection: treat instances with low $p(\boldsymbol{x})$ as anomalies

Introduction: Unsupervised Learning (II)

- Unsupervised learning can be viewed as compression
  - compress a data point to a single label corresponding to its cluster
  - compress a data point from a higher dim. to a lower dim. latent space

- Unsupervised learning can be used ...
  - ... as a stand-alone method (e.g. to understand your data, visualization)
  - ... as a pre-processing step (e.g. use cluster label as feature for subsequent classification task; obtain small number of relevant features)
  - ... to leverage large amounts of unlabeled data for pretraining

- This lecture: Dimensionality Reduction & Matrix Factorization

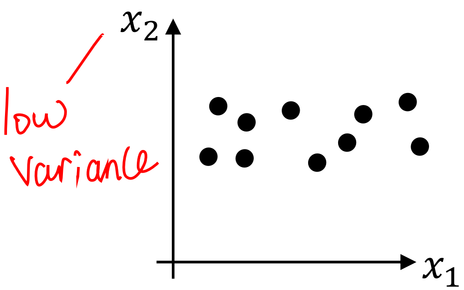
Dimensionality Reduction: Motivation

- Often data has very many features, i.e. high-dimensional data
- High-dimensional data is challenging:
  - Similarity search/computation is expensive because of high complexity of distance functions
  - Highly correlated dimension could cause trouble for some algorithms
  - Curse of dimensionality: we need exponential amounts of data to characterize the density as the dimensionality goes up
  - It is hard to visualize high-dimensional data
- Often the data lies on a low-dim. manifold, embedded in a high-dim. space

- Goal: Try to reduce the dimensionality while avoiding information loss
- Benefits:
  - Computational or memory savings
  - Uncover the intrinsic dimensionality of the data
  - (more benefits later….)

Feature (Sub-)Selection

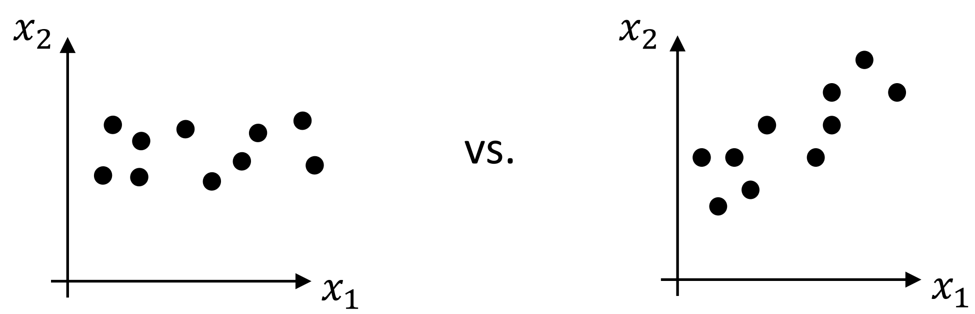
- Choose "good" dimensions using a-priori knowledge or appropriate heuristics
  - e.g. remove **low-variance** dimensions
  - Depending on the application only a few dimensions might be of interest
    - Example: shoe size interesting for shoe purchases, not so for car purchases



- Advantages:
  - No need for an intensive preprocessing or training phase to determine relevant dimensions
- Disadvantages:
  - Expert knowledge required; misjudgment possible
  - Univariate feature selection ignores correlations

Beyond Feature (Sub-)Selection

- Can we do
  - better?
  - automatic?



- Obviously: Simply discarding whole features not a good idea
  - Features are often correlated

The x_2' dimension is barely necessary to describe the data in this new coordinate system
→ can be ignored

* Ideally, we seek to capture data independent of coord sys

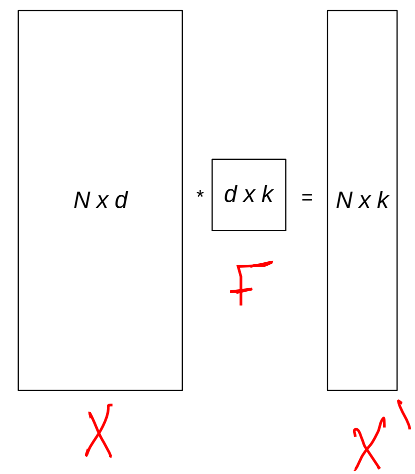
Dim. Reduction via Linear Transformations

preserving dot product
length
angle

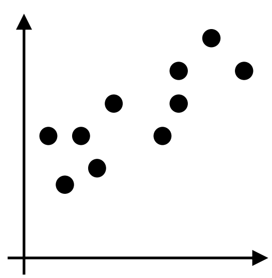
- Represent data in a different coordinate system via linear transformations
  - **change of basis** (orthogonal basis transformations)
    + potentially discarding dimensions of low variance

If such low var dim is known in advance, just remove them earlier

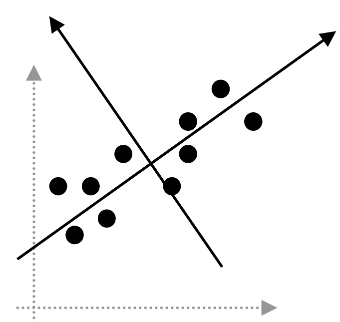
- Technical:
  - use orthonormal transformation matrix $\boldsymbol{F} \in R^{d\times k}$ k<d
  - $(\boldsymbol{x}')^T = \boldsymbol{x}^T \cdot \boldsymbol{F}$ is the transformation of (column) vector $\boldsymbol{x}$ into the new coordinate system defined by $\boldsymbol{F}$
  - $\boldsymbol{X}' = \boldsymbol{X} \cdot \boldsymbol{F}$ is the matrix containing all the transformed points $\boldsymbol{x}'_i$



$FF^T = F^TF$
$= I$



(first center to mean, then)
transformation with F



Discussion: Linear Transformations

discard 2^{nd} dim, keep the rest

let $F = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}_{3 \times 2}$

- Feature selection is a linear transformation
  - What is the matrix $\boldsymbol{F}$?

- Let $\overline{\boldsymbol{x}}$ be the mean vector (here: row vector) in the original data space, the mean vector in the transformed space is given by $\overline{\boldsymbol{x}'} = \overline{\boldsymbol{x}} \cdot \boldsymbol{F}$

- Let $\boldsymbol{\Sigma}_{\boldsymbol{X}}$ be the covariance matrix in the original data space, the covariance matrix in the transformed space is then $\boldsymbol{\Sigma}_{\boldsymbol{X}'} = \boldsymbol{F}^T \cdot \boldsymbol{\Sigma}_{\boldsymbol{X}} \cdot \boldsymbol{F}$

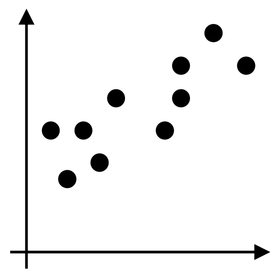
Roadmap

- Chapter: Dimensionality Reduction & Matrix Factorization
  1. Introduction
  2. **Principal Component Analysis (PCA)**
  3. Singular Value Decomposition (SVD)
  4. Matrix Factorization
  5. Neighbor Graph Methods
  6. Autoencoders (Non-linear Dimensionality Reduction)

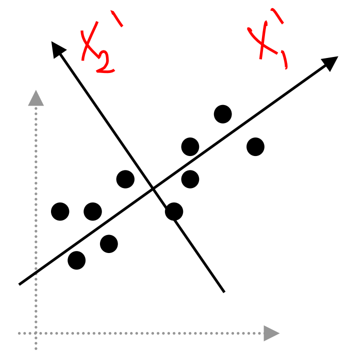
Principal Component Analysis: Motivation

- Question: Which transformation matrix $\boldsymbol{F}$ to use?
  - Is there an **optimal orthogonal transformation** (depending on the data)?
  - Optimality: Approximate the data with few coefficients as well as possible

correlated



transformation of
coordinate axes

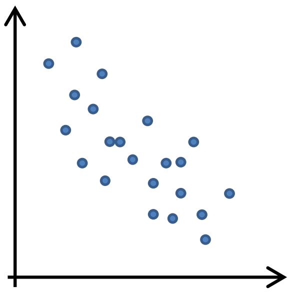


uncorrelated

- Approach: Principal Component Analysis (PCA)
  - Find a coordinate system in which the (possibly originally correlated) points are **linearly uncorrelated**
  - The dimensions with no or low variance can then be ignored

Determine the Principal Components

- Goal:
  - Transform the data, such that the **covariance between the new dimensions is 0**
  - The transformed data points are not linearly correlated any more



- Given: N d-dimensional data points: : $\{\boldsymbol{x}_i\}_{i=1}^{N}$, $\boldsymbol{x}_i \in \mathbb{R}^d \ \forall i \in \{1, \dots, N\}$
- We represent this set of points by a matrix $\boldsymbol{X} \in \mathbb{R}^{N \times d}$:

Feature matrix

$$\boldsymbol{X} = \begin{bmatrix} x_{11} & \cdots & x_{1d} \\ \vdots & \ddots & \vdots \\ x_{N1} & \cdots & x_{Nd} \end{bmatrix}$$

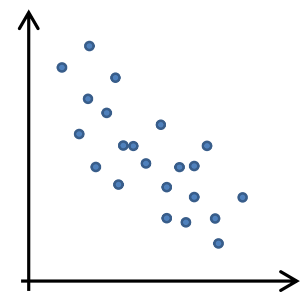
instance

  - The row $\boldsymbol{x}_i = \{x_{i1}, \dots, x_{id}\} \in \mathbb{R}^d$ denotes the i-th point and the column $\boldsymbol{X}_{:,j}$ denotes the vector containing all values from the j-th dimension

feature

Determine the Principal Components

- Goal:
  - Transform the data, such that the **covariance between the new dimensions is 0**
  - The transformed data points are not linearly correlated any more



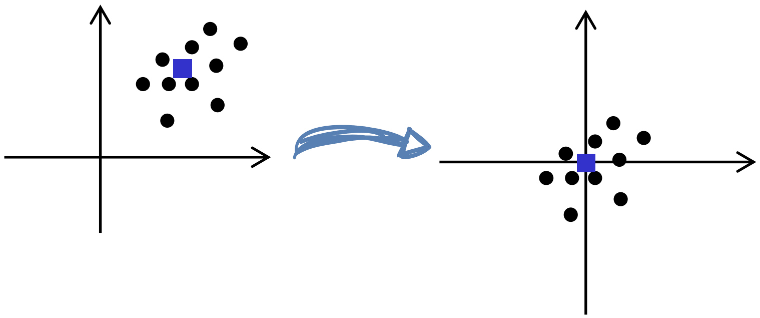
- General approach
  1. Center the data   i.e zero mean
  2. Compute the covariance matrix
  3. Use the Eigenvector decomposition to transform the coordinate system

$\boxed{X \quad N \times D} \longrightarrow \boxed{\mathrm{Cov} \quad D \times D} \longrightarrow V \cdot W \cdot V^T$

V: Matrix of eigenvectors

W: Diagonal matrix of eigenvalues

Determine the Principal Components



- Given: $\boldsymbol{X} \in \mathbb{R}^{N\times d}$: $\boldsymbol{X} = \begin{bmatrix} x_{11} & \cdots & x_{1d} \\ \vdots & \ddots & \vdots \\ x_{N1} & \cdots & x_{Nd} \end{bmatrix}$
- Shift the points by their mean $\overline{\boldsymbol{x}} \in \mathbb{R}^d$ (centralized data): $\widetilde{\boldsymbol{x}}_i = \boldsymbol{x}_i - \overline{\boldsymbol{x}}$

Statistics:

Zero order statistic : number of points N

First order statistic: the mean of the N points, the vector $\overline{\boldsymbol{x}} \in \mathbb{R}^d$:

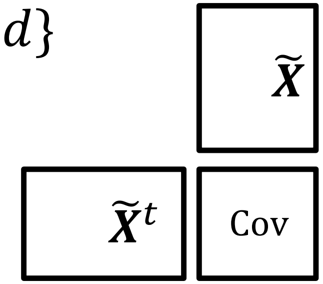
$$\overline{\boldsymbol{x}} = \begin{bmatrix} \bar{x}_1 \\ \vdots \\ \bar{x}_d \end{bmatrix} = \frac{1}{N} \cdot \boldsymbol{X}^T \cdot \mathbf{1}_N$$

where $\mathbf{1}_N$ is an N-dimensional vector of ones

Determine the Principal Components

- Determine the variances $\mathrm{Var}\left(\widetilde{\boldsymbol{X}}_j\right)$ for each dimension $j \in \{1, \dots d\}$
- Determine the covariance $\mathrm{Cov}(\widetilde{\boldsymbol{X}}_{j_1}, \widetilde{\boldsymbol{X}}_{j_2})$ between dimensions j_1 and j_2, $\forall\, j_1 \neq j_2 \in \{1, \dots d\}$

➢ Leads to the covariance matrix $\boldsymbol{\Sigma}_{\widetilde{\boldsymbol{X}}} \in \mathbb{R}^{d \times d}$



<u>Statistics:</u>

Second order statistic: variance and covariance

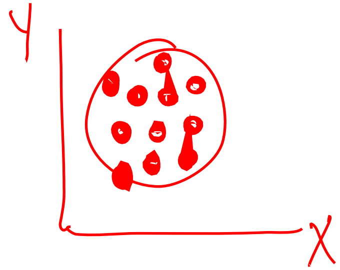
The variance within the j-th dimension in $\boldsymbol{X}$ is:

$$\mathrm{Var}(\boldsymbol{X}_j) = \frac{1}{N} \sum_{i=1}^{N} \left(\boldsymbol{x}_{ij} - \overline{\boldsymbol{x}}_j\right)^2 = \frac{1}{N} \cdot \boldsymbol{X}_j^T \boldsymbol{X}_j - \overline{\boldsymbol{x}}_j^2$$

The covariance between dimension j_1 and j_2 is:

$$\mathrm{Cov}(\boldsymbol{X}_{j_1}, \boldsymbol{X}_{j_2}) = \frac{1}{N} \sum_{i=1}^{N} \left(\boldsymbol{x}_{ij_1} - \overline{\boldsymbol{x}}_{j_1}\right) \cdot \left(\boldsymbol{x}_{ij_2} - \overline{\boldsymbol{x}}_{j_2}\right) = \frac{1}{N} \cdot \boldsymbol{X}_{j_1}^T \boldsymbol{X}_{j_2} - \overline{\boldsymbol{x}}_{j_1} \overline{\boldsymbol{x}}_{j_2}$$

$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$



Variation in X equal Variation in Y $\Rightarrow$ Zero Correlation

Statistics (continued):

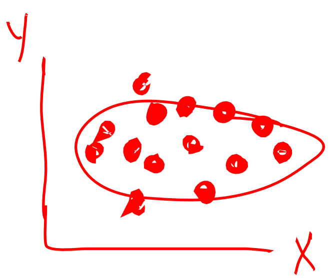
For the set of points contained in $\boldsymbol{X}$ the corresponding covariance matrix is defined as:

$$\boldsymbol{\Sigma}_{\boldsymbol{X}} = \begin{bmatrix} \mathrm{Var}(\boldsymbol{X}_1) & \mathrm{Cov}(\boldsymbol{X}_1, \boldsymbol{X}_2) & \cdots & \mathrm{Cov}(\boldsymbol{X}_1, \boldsymbol{X}_d) \\ \mathrm{Cov}(\boldsymbol{X}_2, \boldsymbol{X}_1) & \mathrm{Var}(\boldsymbol{X}_2) & & \\ \vdots & & \ddots & \vdots \\ \mathrm{Cov}(\boldsymbol{X}_d, \boldsymbol{X}_1) & \cdots & & \mathrm{Var}(\boldsymbol{X}_d) \end{bmatrix} = \frac{1}{N}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X} - \overline{\boldsymbol{x}}\,\overline{\boldsymbol{x}}^{T}$$

zero mean

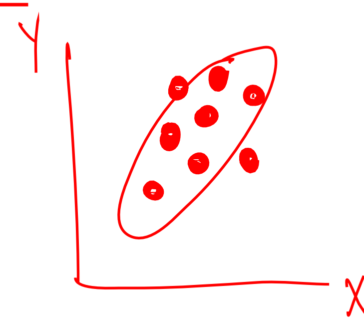
- Remark: Covariance matrices are symmetric

$\begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix}$



$Var_X = 4\, Var_Y$

$\begin{bmatrix} 1 & 1.5 \\ 1.5 & 1 \end{bmatrix}$



X & Y directly correlate

non-zero off-diag

Our original data is usually correlated i.e non Zero values everywhere

$$Cov' = \begin{pmatrix} Var(1)' & 0 & \cdots & 0 \\ 0 & Var(2)' & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & Var(D)' \end{pmatrix}$$

- **Goal of PCA:** Transformation of the coordinate system such that the covariances between the new axes are 0
- **Approach:**
  - Diagonalization by changing the basis (= adapt the coordinate system)
  - According to the spectral theorem, the eigenvectors of a symmetric matrix form an orthogonal basis

➢ **Eigendecomposition of the covariance matrix:** $\boldsymbol{\Sigma}_{\widetilde{X}} = \boldsymbol{\Gamma} \cdot \boldsymbol{\Lambda} \cdot \boldsymbol{\Gamma}^T$

Eigendecomposition (spectral decomposition) is the factorization of $\boldsymbol{A} \in \mathbb{R}^{d\times d}$:

$$\boldsymbol{A} = \boldsymbol{\Gamma} \cdot \boldsymbol{\Lambda} \cdot \boldsymbol{\Gamma}^T$$

cols are γ_i of covariance matrix

→ matrices $\boldsymbol{\Gamma}, \boldsymbol{\Lambda} \in \mathbb{R}^{d\times d}$ with columns of $\boldsymbol{\Gamma}$ being the normalized eigenvectors $\boldsymbol{\gamma}_i$

→ $\boldsymbol{\Gamma}$ is an orthonormal matrix: $\boldsymbol{\Gamma} \cdot \boldsymbol{\Gamma}^T = \boldsymbol{\Gamma}^T \cdot \boldsymbol{\Gamma} = \mathrm{Id}$ $(\boldsymbol{\Gamma}^T = \boldsymbol{\Gamma}^{-1})$

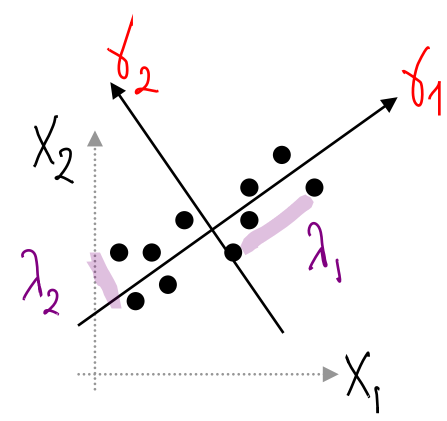
→ $\boldsymbol{\Lambda}$ is a diagonal matrix with eigenvalues λ_i as the diagonal elements

they don't need to have unit length

γ_i : the vec that stayed the same, only scaled by λ_i
when applying matrix A to some vec
placing γ_i as cols for some matrix M $\Rightarrow$ M will be orthogonal

- The **new coordinate system** is defined by the eigenvectors $\boldsymbol{\gamma}_i$:
  - Transformed data: $\boldsymbol{Y} = \widetilde{\boldsymbol{X}} \cdot \boldsymbol{\Gamma}$
  - $\boldsymbol{\Lambda}$ is the covariance matrix in this new coordinate system
  - New system has variance λ_i in dimension i
  - $\forall i_1 \neq i_2 : \mathrm{Cov}\left(\boldsymbol{Y}_{i_1}, \boldsymbol{Y}_{i_2}\right) = 0$

no correlation
bet dim



Dimensionality Reduction with PCA

- Approach
  - The coordinates with low variance (hence low λ_i) can be ignored
  - W.l.o.g. let us assume $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_\mathrm{d}$

➢ Truncation of $\mathbf{\Gamma}$

- **Keep only columns (i.e. eigenvectors) of $\mathbf{\Gamma}$ corresponding to the largest $\mathbf{k}$ eigenvalues $\boldsymbol{\lambda_1, \ldots, \lambda_k}$**
- $\boldsymbol{Y}_{\text{reduced}} = \widetilde{\boldsymbol{X}} \cdot \mathbf{\Gamma}_{\text{truncated}}$

applying linear transformation to get the new mean

- How to pick k?
  - Frequently used: 90% rule; the *k* variances should explain 90% of the energy
  - k = smallest value ensuring $\sum_{i=1}^{k} \lambda_i \geq 0.9 \cdot \sum_{i=1}^{d} \lambda_i$

➢ The modified points (transformed and truncated) contain most of the information of the original points and are low dimensional

GOAL

* So far it's zero mean data, sometimes we reserve this; seeking more info if lost due to subtracting the mean

Complexity

- Complexity of PCA:

$$O(N \cdot d^2) \quad + \quad O(d^3) \quad + \quad O(N \cdot d \cdot k) = O(N \cdot d^2 + d^3)$$

Compute covariance matrix | Eigenvalue decomposition | Project data onto the k-dimensional space

- Remarks on eigenvalue decomposition:
  - Usually we are interested in the reduced data only
  - ➢ **Only the *k* largest eigenvectors required** (i.e. not all of them)
  - Use iterative approaches (next slide) for finding eigenvectors
    - Complexity: $O(\#\text{it} \cdot d^2)$ // #it = number of iterations
    - For sparse data even faster: $O(\#\text{it} \cdot \#\text{nz})$ // #nz = number of nonzero elements in the matrix

directly compute K largest, no need to compute all

How to Compute Eigenvectors?

- Eigenvalues are important for many machine learning/data mining tasks
  - PCA, Ranking of Websites, Community Detection, … // see our other lecture!
  - How to compute them efficiently?

- Power iteration (a.k.a. Von Mises iteration)
  - Iterative approach to compute **a single** eigenvector
- Let $\boldsymbol{A}$ be a matrix and $\boldsymbol{v}$ be an arbitrary (normalized) vector
  - Iteratively compute $\boldsymbol{v} \leftarrow \frac{\boldsymbol{A} \cdot \boldsymbol{v}}{\|\boldsymbol{A} \cdot \boldsymbol{v}\|}$ until convergence
    - in each step, $\boldsymbol{v}$ is simply multiplied with $\boldsymbol{A}$ and normalized
  - **$\boldsymbol{v}$ converges to the eigenvector of $\boldsymbol{A}$ with largest absolute value**
  - Highly efficient for sparse data

How to Compute Eigenvectors?

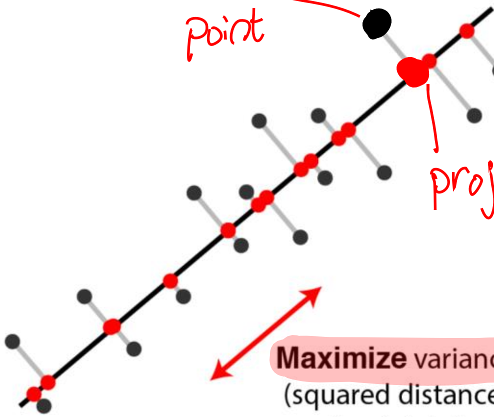
- Convergence: # iter depend on
  - Linear convergence with rate $|\lambda_2/\lambda_1|$
  - Fast convergence if first and second eigenvalue are dissimilar
- How to find **multiple (the k largest) eigenvectors**?
  - Let us focus on symmetric matrices $\boldsymbol{A}$
  - Eigenvalue decomposition leads to: $\boldsymbol{A} = \boldsymbol{\Gamma} \cdot \boldsymbol{\Lambda} \cdot \boldsymbol{\Gamma}^T = \sum_{i=1}^{d} \lambda_i \cdot \boldsymbol{\gamma}_i \cdot \boldsymbol{\gamma}_i^T$
  - Define deflated matrix: $\widehat{\boldsymbol{A}} = \boldsymbol{A} - \lambda_1 \cdot \boldsymbol{\gamma}_1 \cdot \boldsymbol{\gamma}_1^T$
    - $\widehat{\boldsymbol{A}}$ has the same eigenvectors as $\boldsymbol{A}$ except the first one has become zero
  - ➢ Apply power iteration on $\widehat{\boldsymbol{A}}$ to find the second largest eigenvector of $\boldsymbol{A}$

NB Largest λ_1 corresponds to the dim which have the largest variance in the transformed space

Alternative views of PCA

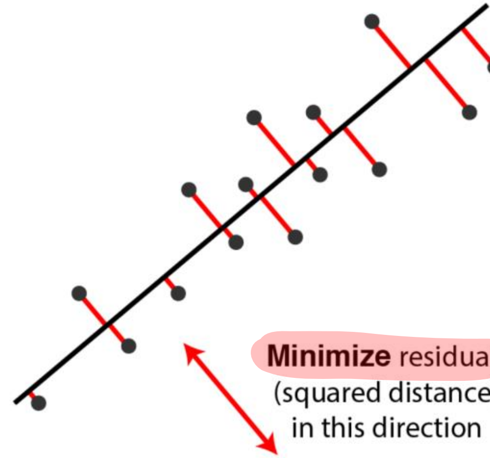
Eigenvector also called principal component

data point



proj data

Maximize variance
(squared distance)
of red dots in
this direction



proj dist

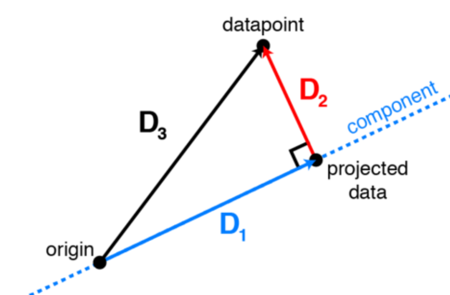
Minimize residuals
(squared distance)
in this direction

$$D_3^2 = D_1^2 + D_2^2$$

fixed; variance of original space

max

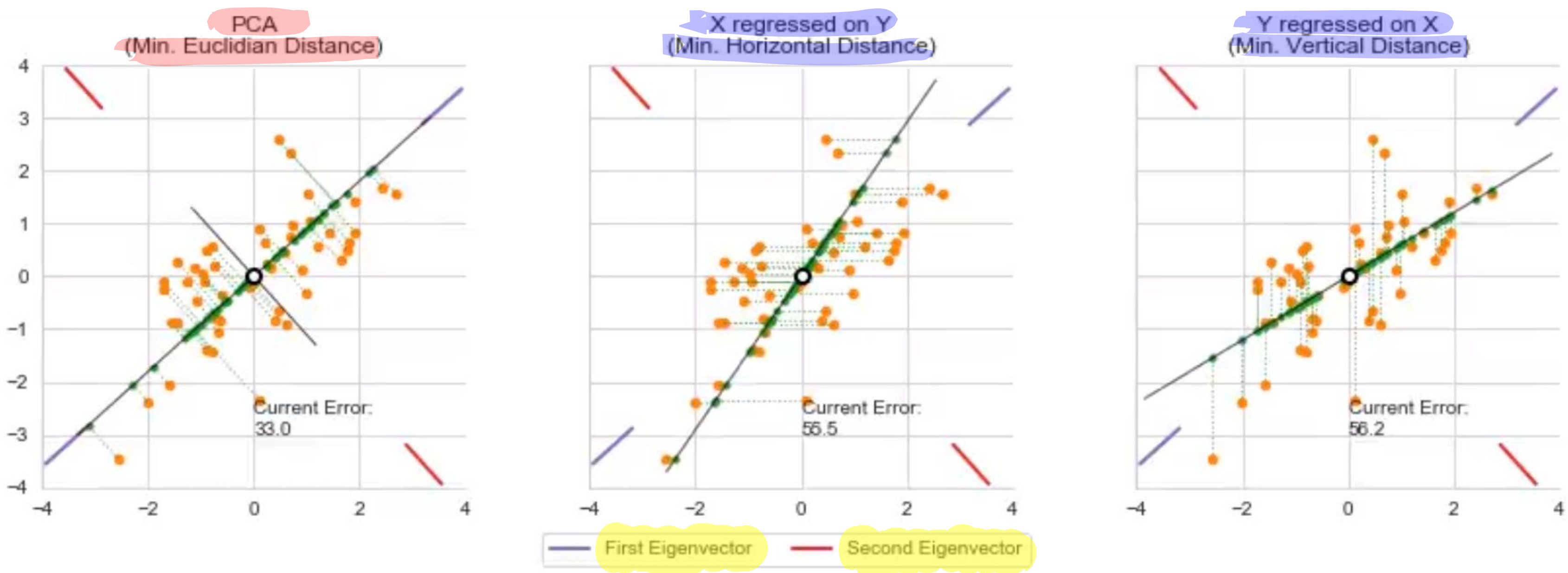
min



Images adapted from Alexh Williams

PCA vs. Regression

Y given



trying to min both Simultaneously

X-dir only is to be min

Y_1 is not aligned

Image adapted from Quentin André

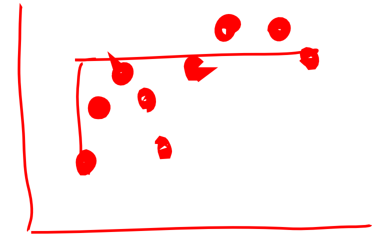
PCA: Summary

- PCA finds the optimal transformation by deriving uncorrelated dimensions
  - Exploits eigendecomposition
- Dimensionality reduction
  - After transformation simply remove dimensions with lowest variance (or use only the k largest eigenvectors for transformation)
- Limitations
  - Only captures linear relationships (one solution: Kernel PCA)

* PCA is usually the first thing to be done, when analyzing data

Roadmap

PCA is not efficient in case of non-linear data



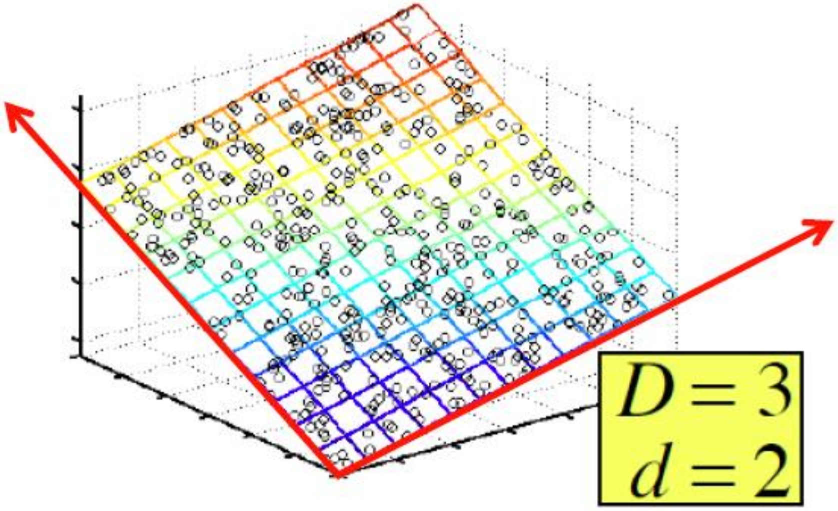
- Chapter: Dimensionality Reduction & Matrix Factorization
  1. Introduction
  2. Principal Component Analysis (PCA)
  3. **Singular Value Decomposition (SVD)**
     - **Idea: Low Rank Approximation**
     - SVD & Latent Factors
     - Dimensionality Reduction
  4. Matrix Factorization
  5. Neighbor Graph Methods
  6. Autoencoders (Non-linear Dimensionality Reduction)

equivalent

→ aims to find the best low rank approximation for agiven matrix

Low-Dimensional Manifold

- Data often lies on a low-dimensional manifold embedded in higher dimensional space



- How can me measure the dimensionality of this manifold?
  – put differently how to measure the intrinsic dimensionality of the data
- How can we find this manifold?

Rank of a Matrix

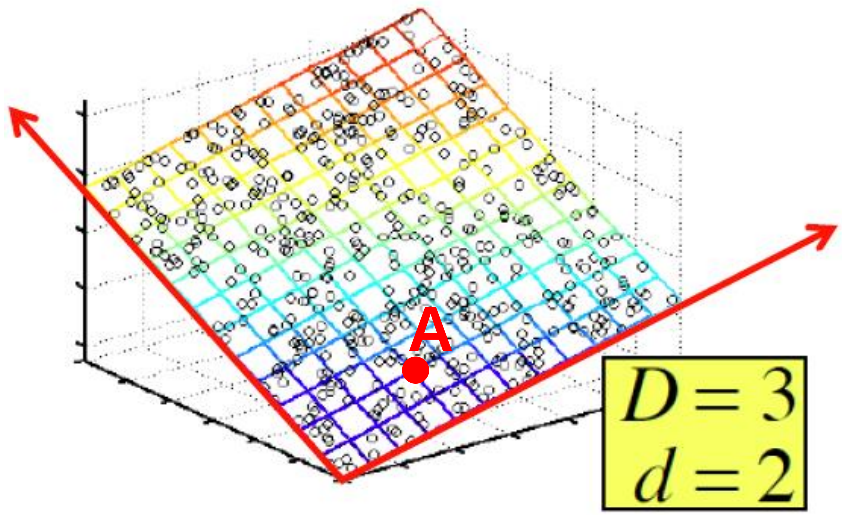
- Q: What is the rank of a matrix A?
- A: Number of linearly independent columns/rows of A
- Example:
  - Matrix A = $\begin{bmatrix} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{bmatrix}$ has rank r=2
    - Why? The first two rows are linearly independent, so the rank is at least 2, but all three rows are linearly dependent (the first is equal to the sum of the second and third) so the rank must be less than 3.

- Why do we care about low rank?
  - We can write A as two "basis" vectors: [1 2 1] [-2 -3 1]
  - And new coordinates of: [1 0] [0 1] [1 -1]

Rank is "Dimensionality"

- Cloud of points in 3D space:
  - Think of point positions as a matrix:

1 row per point: $\begin{array}{c} \\ \end{array}\begin{bmatrix} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{bmatrix}\begin{array}{l} \mathbf{A} \\ \mathbf{B} \\ \mathbf{C} \end{array}$



- **We can rewrite coordinates more efficiently!**
  - Old basis vectors: [1 0 0] [0 1 0] [0 0 1]
  - **New basis vectors: [1 2 1] [-2 -3 1]**
  - Then A has new coordinates: [1 0], B: [0 1], C: [1 -1]
    - **Notice: We reduced the number of coordinates!**

Low Rank Approximation

- Idea: approximate original data A by a low rank matrix B

A being noisy, its rows are no longer dependent

all its row are linearly independent

$$\mathbf{A} = \begin{bmatrix} 1.01 & 2.05 & 0.9 \\ -2.1 & -3.05 & 1.1 \\ 2.99 & 5.01 & 0.3 \end{bmatrix} \approx \begin{bmatrix} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{bmatrix} = \mathbf{B}$$

$\text{rank}(\boldsymbol{A}) = 3$

we need 3 coordinates to describe each point

$\text{rank}(\boldsymbol{B}) = 2$

we need only 2 coordinates per point

Low Rank Approximation

- Idea: approximate original data A by a low rank matrix B

$$\mathbf{A} = \begin{bmatrix} 1.01 & 2.05 & 0.9 \\ -2.1 & -3.05 & 1.1 \\ 2.99 & 5.01 & 0.3 \end{bmatrix} \approx \begin{bmatrix} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{bmatrix} = \mathbf{B}$$

- Important: Even though both A and B are $\in R^{n \times d}$ we need only two coordinates per point to describe B
  - rank(A)=3 vs. rank(B)=2 (3 vs. 2 coordinates per point)

- **Goal: Find the best low rank approximation**
  - best = minimize the sum of reconstruction error
  - Given matrix $\boldsymbol{A} \in \mathbb{R}^{n \times d}$, find $\boldsymbol{B} \in \mathbb{R}^{n \times d}$ with $\operatorname{rank}(\boldsymbol{B}) = k$ that minimizes

$$\|\boldsymbol{A} - \boldsymbol{B}\|_F^2 = \sum_{i=1}^{N} \sum_{j=1}^{D} \left(a_{ij} - b_{ij}\right)^2$$

$$\|M\|_F^2 = \sum_{ij} M_{ij}^2$$

We use SVD to solve it

Roadmap

- Chapter: Dimensionality Reduction & Matrix Factorization
  1. Introduction
  2. Principal Component Analysis (PCA)
  3. **Singular Value Decomposition (SVD)**
     – Idea: Low Rank Approximation
     – **SVD & Latent Factors**
     – Dimensionality Reduction
  4. Matrix Factorization
  5. Neighbor Graph Methods
  6. Autoencoders (Non-linear Dimensionality Reduction)

Singular Value Decomposition (SVD): Definition

- Each real matrix $\boldsymbol{A} \in \mathbb{R}^{n\times d}$ can be decomposed into $\boldsymbol{A} = \boldsymbol{U}\cdot\boldsymbol{\Sigma}\cdot\boldsymbol{V}^{\mathrm{T}}$ (note: **exact representation**, no approximation), where
  - $\boldsymbol{U} \in \mathbb{R}^{n\times r}$, $\boldsymbol{\Sigma} \in \mathbb{R}^{r\times r}$, $\boldsymbol{V} \in \mathbb{R}^{\mathrm{d}\times \mathrm{r}}$
  - $\boldsymbol{U}, \boldsymbol{V}$: column orthonormal
    - i.e. $\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I}$; $\boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{I}$ (I: identity matrix)
    - $\boldsymbol{U}$ are called the left singular vectors, $\boldsymbol{V}$ the right singular vectors
  - $\boldsymbol{\Sigma}$: diagonal
    - $r \times r$ diagonal matrix (r: rank of matrix $\boldsymbol{A}$)
    - entries (called singular values) are positive, and sorted in decreasing order ($\sigma_1 \geq \sigma_2 \geq \ \dots \geq \sigma_r > 0$)

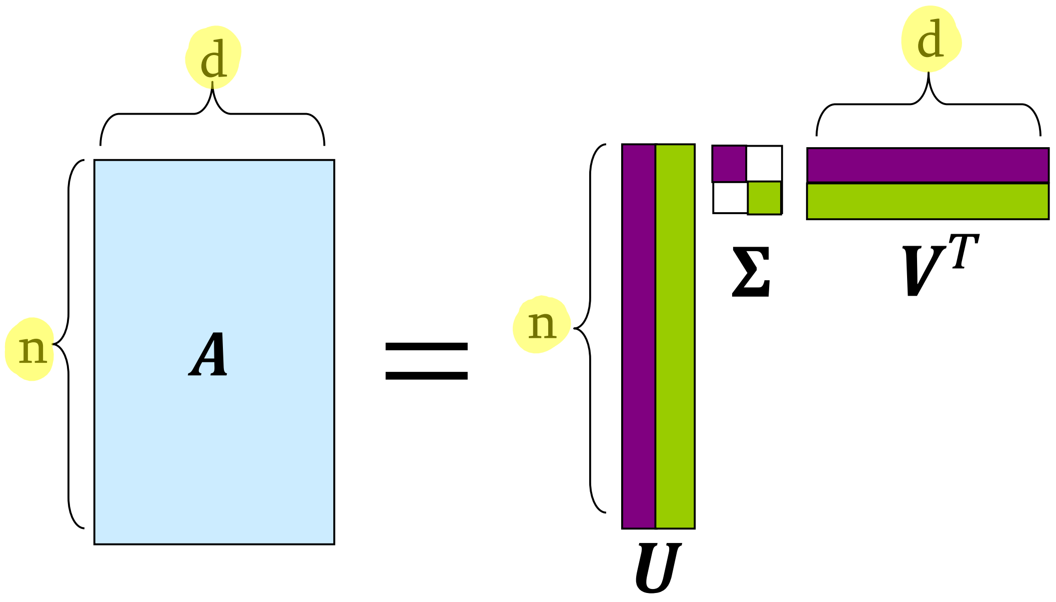
$r \leq \min(n, d)$

- Remark: The decomposition is (almost) unique
  - see e.g. multiplication by -1

* Orthonormal → Zero dot product
→ unit length

Singular Value Decomposition

$$\boldsymbol{A} = \boldsymbol{U\Sigma V}^T$$



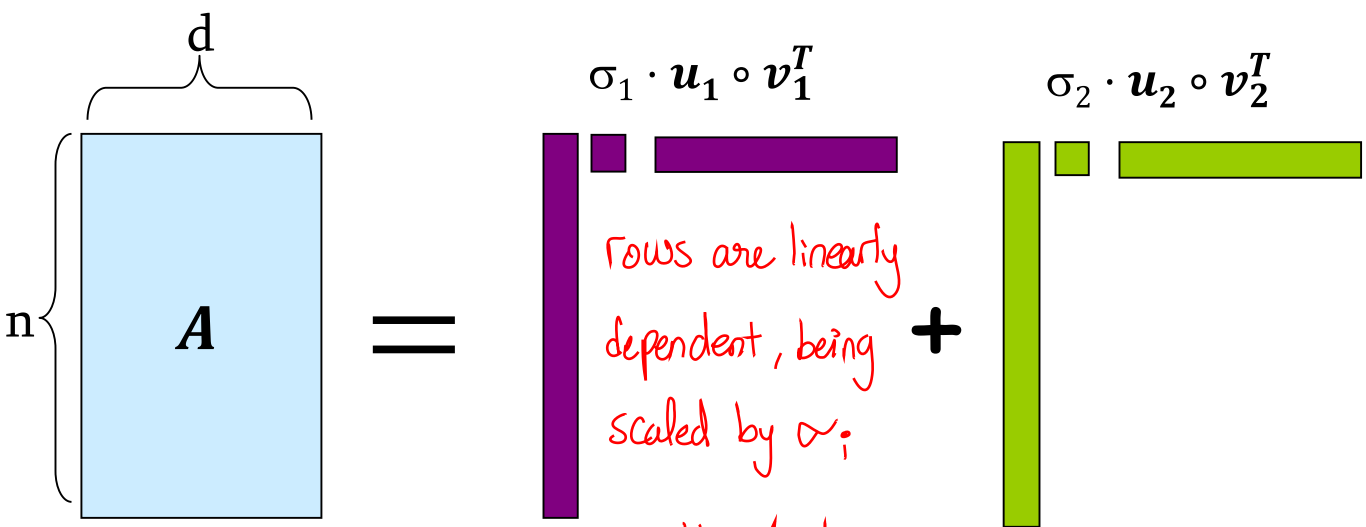
here: $r = 2$

Singular Value Decomposition

As summation

outer prod

$$\boldsymbol{A} = \boldsymbol{U\Sigma V}^T = \sum_{i=1}^{r} \sigma_i \cdot \boldsymbol{u}_i \circ \boldsymbol{v}_i^T$$



rows are linearly dependent, being scaled by α_i

rank = 1 by construction

here: $r = 2$

SVD Example: Users-to-Movies

- $A = U\Sigma V^T$ - example: Users to Movies

* Assuming A is face image, U shall be called eigenfaces where rows express features

→ every col of A is reconstructed as linear combination of eigenfaces scaled by α_i

SciFi-concept

Romance-concept

$$
\begin{array}{c c}
 & \begin{array}{ccccc} \text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \end{array} \\
\text{users} & \begin{bmatrix} \boxed{\begin{matrix} 1 & 1 & 1 \\ 3 & 3 & 3 \\ 4 & 4 & 4 \\ 5 & 5 & 5 \end{matrix}} & \begin{matrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{matrix} \\ \begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{matrix} & \begin{matrix} 4 & 4 \\ 5 & 5 \\ 2 & 2 \end{matrix} \end{bmatrix}_{7\times 5}
\end{array}
=
\begin{bmatrix} 0.14 & 0 \\ 0.42 & 0 \\ 0.56 & 0 \\ 0.70 & 0 \\ 0 & 0.59 \\ 0 & 0.74 \\ 0 & 0.29 \end{bmatrix}_{7\times 2}
\times
\begin{bmatrix} 12.36 & 0 \\ 0 & 9.48 \end{bmatrix}_{2\times 2}
\times
\begin{bmatrix} 0.57 & 0.57 & 0.57 & 0 & 0 \\ 0 & 0 & 0 & 0.70 & 0.70 \end{bmatrix}_{2\times 7}
$$

rank = 2

Users
Movies
Concepts

SVD Example: Users-to-Movies

- $\boldsymbol{A} = \boldsymbol{U\Sigma V}^T$ - example: Users to Movies

Unique Decomposition

SciFi-concept

Romance-concept (note the multiplication by -1)

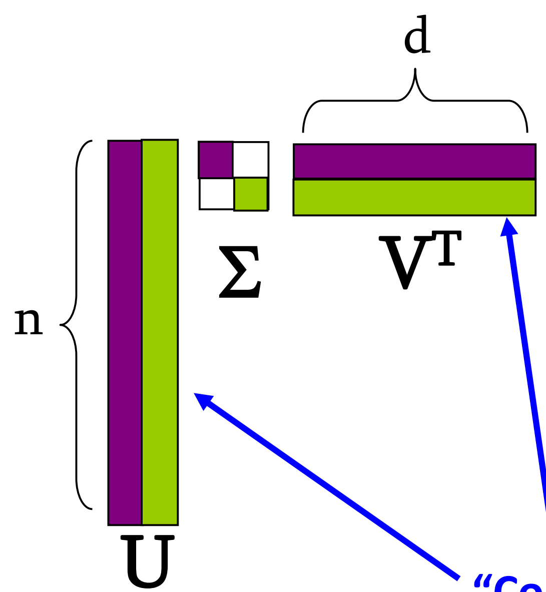
$$\begin{array}{c} \\ \text{users} \end{array} \begin{array}{c} \begin{array}{ccccc} \text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 0 & 0 & 2 & 2 \end{bmatrix} \end{array} = \begin{bmatrix} 0.14 & 0 \\ 0.42 & 0 \\ 0.56 & 0 \\ 0.70 & 0 \\ 0 & -0.59 \\ 0 & -0.74 \\ 0 & -0.29 \end{bmatrix} \times \begin{bmatrix} 12.36 & 0 \\ 0 & 9.48 \end{bmatrix} \times \begin{bmatrix} 0.57 & 0.57 & 0.57 & 0 & 0 \\ 0 & 0 & 0 & -0.70 & -0.70 \end{bmatrix}$$

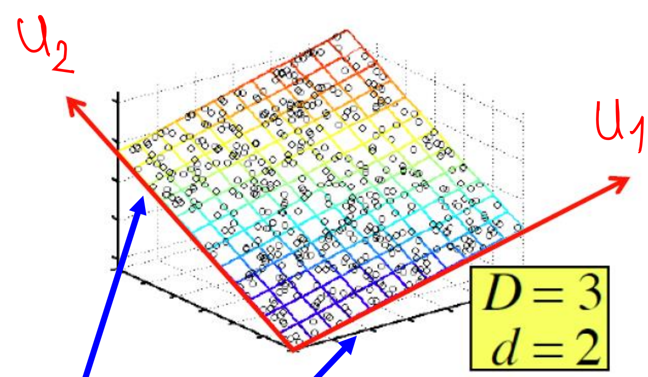
SVD Example: Latent Factors

- $\boldsymbol{A} = \boldsymbol{U\Sigma V}^T$ - example: Users to Movies

the colors

$$\begin{array}{cc} & \begin{matrix}\text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie}\end{matrix} \\ \text{users} & \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 0 & 0 & 2 & 2 \end{bmatrix} \end{array} =$$





"Concepts"
a.k.a. Latent dimensions
a.k.a. Latent factors

SVD Example: Beyond Blocks

Assuming we changed A

Values of U changed alittle

Value of Σ & V^T changed, a new row introduced

$$
\underset{7\times5}{\begin{array}{c}
\begin{array}{ccccc} \text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \end{array} \\
\begin{bmatrix}
\mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\
\mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\
\mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\
\mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\
0 & \mathbf{2} & 0 & \mathbf{4} & \mathbf{4} \\
0 & 0 & 0 & \mathbf{5} & \mathbf{5} \\
0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2}
\end{bmatrix}
\end{array}}
=
\underset{7\times3}{\begin{bmatrix}
\mathbf{0.13} & -0.02 & -0.01 \\
\mathbf{0.41} & -0.07 & -0.03 \\
\mathbf{0.55} & -0.09 & -0.04 \\
\mathbf{0.68} & -0.11 & -0.05 \\
0.15 & \mathbf{0.59} & \mathbf{0.65} \\
0.07 & \mathbf{0.73} & \mathbf{-0.67} \\
0.07 & \mathbf{0.29} & \mathbf{0.32}
\end{bmatrix}}
\times
\underset{3\times3}{\begin{bmatrix}
\mathbf{12.4} & 0 & 0 \\
0 & \mathbf{9.5} & 0 \\
0 & 0 & \mathbf{1.3}
\end{bmatrix}}
\times
\underset{3\times5}{\begin{bmatrix}
\mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\
-0.12 & 0.02 & -0.12 & \mathbf{0.69} & \mathbf{0.69} \\
0.40 & \mathbf{-0.80} & 0.40 & 0.09 & 0.09
\end{bmatrix}}
$$

Now, rank(A) = 3

SciFi-concept

Romance-concept

$$
\begin{array}{c}
\begin{array}{ccccc} \text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \end{array} \\
\begin{bmatrix}
\mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\
\mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\
\mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\
\mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\
0 & \mathbf{2} & 0 & \mathbf{4} & \mathbf{4} \\
0 & 0 & 0 & \mathbf{5} & \mathbf{5} \\
0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2}
\end{bmatrix}
\end{array}
=
\begin{bmatrix}
\mathbf{0.13} & -0.02 & -0.01 \\
\mathbf{0.41} & -0.07 & -0.03 \\
\mathbf{0.55} & -0.09 & -0.04 \\
\mathbf{0.68} & -0.11 & -0.05 \\
0.15 & \mathbf{0.59} & \mathbf{0.65} \\
0.07 & \mathbf{0.73} & \mathbf{-0.67} \\
0.07 & \mathbf{0.29} & \mathbf{0.32}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{12.4} & 0 & 0 \\
0 & \mathbf{9.5} & 0 \\
0 & 0 & \mathbf{1.3}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\
-0.12 & 0.02 & -0.12 & \mathbf{0.69} & \mathbf{0.69} \\
0.40 & \mathbf{-0.80} & 0.40 & 0.09 & 0.09
\end{bmatrix}
$$

Matrix A < users / movies

0.13 reflects the amount of similarity bet
- 1st user
- 1st concept

***U* is "user-to-concept" similarity matrix**

$$
\begin{array}{c}
\begin{array}{ccccc} \text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \end{array} \\
\begin{bmatrix}
\mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\
\mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\
\mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\
\mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\
0 & \mathbf{2} & 0 & \mathbf{4} & \mathbf{4} \\
0 & 0 & 0 & \mathbf{5} & \mathbf{5} \\
0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2}
\end{bmatrix}
\end{array}
=
\begin{array}{c}
\begin{array}{ccc} \text{SciFi-concept} & \text{Romance-concept} & \end{array} \\
\begin{bmatrix}
\mathbf{0.13} & -0.02 & -0.01 \\
\mathbf{0.41} & -0.07 & -0.03 \\
\mathbf{0.55} & -0.09 & -0.04 \\
\mathbf{0.68} & -0.11 & -0.05 \\
0.15 & \mathbf{0.59} & \mathbf{0.65} \\
0.07 & \mathbf{0.73} & \mathbf{-0.67} \\
0.07 & \mathbf{0.29} & \mathbf{0.32}
\end{bmatrix}
\end{array}
\times
\begin{bmatrix}
12.4 & 0 & 0 \\
0 & \mathbf{9.5} & 0 \\
0 & 0 & \mathbf{1.3}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\
-0.12 & 0.02 & -0.12 & \mathbf{0.69} & \mathbf{0.69} \\
0.40 & \mathbf{-0.80} & 0.40 & 0.09 & 0.09
\end{bmatrix}
$$

* when interpreting values, don't forget reflecting on all the multiplications $U\Sigma V^T$

SciFi-concept

“strength” of the SciFi-concept

$$
\begin{array}{c}
\begin{array}{ccccc} \text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \end{array} \\
\begin{bmatrix}
\mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\
\mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\
\mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\
\mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\
0 & \mathbf{2} & 0 & \mathbf{4} & \mathbf{4} \\
0 & 0 & 0 & \mathbf{5} & \mathbf{5} \\
0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2}
\end{bmatrix}
\end{array}
=
\begin{bmatrix}
\mathbf{0.13} & -0.02 & -0.01 \\
\mathbf{0.41} & -0.07 & -0.03 \\
\mathbf{0.55} & -0.09 & -0.04 \\
\mathbf{0.68} & -0.11 & -0.05 \\
0.15 & \mathbf{0.59} & \mathbf{0.65} \\
0.07 & \mathbf{0.73} & \mathbf{-0.67} \\
0.07 & \mathbf{0.29} & \mathbf{0.32}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{12.4} & 0 & 0 \\
0 & \mathbf{9.5} & 0 \\
0 & 0 & \mathbf{1.3}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\
-0.12 & 0.02 & -0.12 & \mathbf{0.69} & \mathbf{0.69} \\
0.40 & \mathbf{-0.80} & 0.40 & 0.09 & 0.09
\end{bmatrix}
$$

V is "movie-to-concept" similarity matrix

SciFi-concept

$$
\begin{array}{ccccc}
\text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \\
\end{array}
$$

$$
\begin{bmatrix}
\mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\
\mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\
\mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\
\mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\
0 & \mathbf{2} & 0 & \mathbf{4} & \mathbf{4} \\
0 & 0 & 0 & \mathbf{5} & \mathbf{5} \\
0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2}
\end{bmatrix}
=
\begin{bmatrix}
\mathbf{0.13} & -0.02 & -0.01 \\
\mathbf{0.41} & -0.07 & -0.03 \\
\mathbf{0.55} & -0.09 & -0.04 \\
\mathbf{0.68} & -0.11 & -0.05 \\
0.15 & \mathbf{0.59} & \mathbf{0.65} \\
0.07 & \mathbf{0.73} & \mathbf{-0.67} \\
0.07 & \mathbf{0.29} & \mathbf{0.32}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{12.4} & 0 & 0 \\
0 & \mathbf{9.5} & 0 \\
0 & 0 & \mathbf{1.3}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\
-0.12 & 0.02 & -0.12 & \mathbf{0.69} & \mathbf{0.69} \\
0.40 & \mathbf{-0.80} & 0.40 & 0.09 & 0.09
\end{bmatrix}
$$

SciFi-concept

SVD: Interpretation

* Such decomposition allows for unvealing the true latent dimensionality/rank, allowing for better interpretability of the factors U, Σ & V^T.

- $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$
- 'movies', 'users' and 'concepts':
  - $\boldsymbol{A}$ original data: movies-to-users
  - $\boldsymbol{U}$: user-to-concept similarity matrix
  - $\boldsymbol{V}$: movie-to-concept similarity matrix
  - $\boldsymbol{\Sigma}$: its diagonal elements: 'strength' of each concept

- **Benefits of SVD (or in general matrix decomposition):**
  - Discover hidden correlations/topics
    - Words that occur commonly together; movies of the same genre; …
  - Interpretation and visualization

Roadmap

- Chapter: Dimensionality Reduction & Matrix Factorization
  1. Introduction
  2. Principal Component Analysis (PCA)
  3. **Singular Value Decomposition (SVD)**
     - Idea: Low Rank Approximation
     - SVD & Latent Factors
     - **Dimensionality Reduction**
  4. Matrix Factorization
  5. Neighbor Graph Methods
  6. Autoencoders (Non-linear Dimensionality Reduction)

Recap: Dim. Reduction by Low Rank Approx.

- Idea: approximate original data $\boldsymbol{A}$ by a low rank matrix $\boldsymbol{B}$

$$\boldsymbol{A} = \begin{bmatrix} 1.01 & 2.05 & 0.9 \\ -2.1 & -3.05 & 1.1 \\ 2.99 & 5.01 & 0.3 \end{bmatrix} \approx \begin{bmatrix} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{bmatrix} = \boldsymbol{B}$$

- Important: Even though both $\boldsymbol{A}$ and $\boldsymbol{B}$ are $\in \mathbb{R}^{n\times d}$ we need only two coordinates per point to describe $\boldsymbol{B}$
  - $\mathrm{rank}(\boldsymbol{A}) = 3$ vs. $\mathrm{rank}(\boldsymbol{B}) = 2$ (3 vs. 2 coordinates per point)

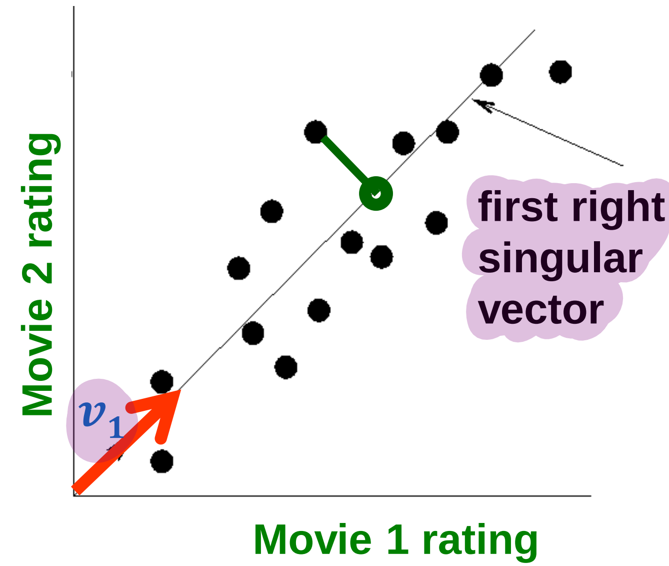
- **Goal: Find the best low rank approximation**
  - best = minimize the sum of reconstruction error
  - Given matrix $\boldsymbol{A} \in \mathbb{R}^{n\times d}$, find $\boldsymbol{B} \in \mathbb{R}^{n\times d}$ with $\mathrm{rank}(\boldsymbol{B}) = k$ that minimizes

$$\|\boldsymbol{A} - \boldsymbol{B}\|_F^2 = \sum_{i=1}^{N}\sum_{j=1}^{D}\left(a_{ij} - b_{ij}\right)^2$$

GOAL → finding 2 basis vectors (2, because rank=2)
& for each, getting the new coord [slide 28]

SVD: Alternative Interpretation

- A = $U\Sigma V^T$ example:



variance ('spread') on the v_1 axis

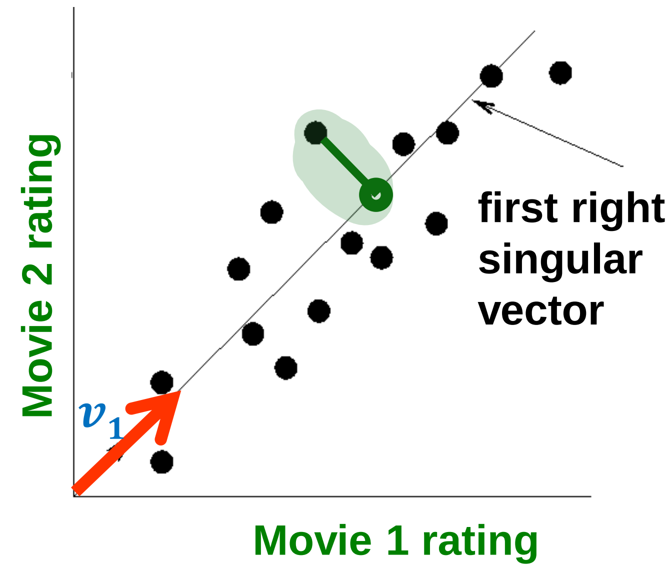
$$
\begin{bmatrix}
\mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\
\mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\
\mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\
\mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\
0 & \mathbf{2} & 0 & \mathbf{4} & \mathbf{4} \\
0 & 0 & 0 & \mathbf{5} & \mathbf{5} \\
0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2}
\end{bmatrix}
=
\begin{bmatrix}
\mathbf{0.13} & 0.02 & -0.01 \\
\mathbf{0.41} & 0.07 & -0.03 \\
\mathbf{0.55} & 0.09 & -0.04 \\
\mathbf{0.68} & 0.11 & -0.05 \\
0.15 & \mathbf{-0.59} & \mathbf{0.65} \\
0.07 & \mathbf{-0.73} & \mathbf{-0.67} \\
0.07 & \mathbf{-0.29} & \mathbf{0.32}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{12.4} & 0 & 0 \\
0 & \mathbf{9.5} & 0 \\
0 & 0 & \mathbf{1.3}
\end{bmatrix}
\times
\begin{array}{c}
v_1 \\ v_2 \\ v_3
\end{array}
\begin{bmatrix}
\mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\
0.12 & -0.02 & 0.12 & \mathbf{-0.69} & \mathbf{-0.69} \\
0.40 & \mathbf{-0.80} & 0.40 & 0.09 & 0.09
\end{bmatrix}
$$

New axes

- **U Σ: Gives the coordinates of the points in the projection axis**

"new coord"

Projection of users on the "Sci-Fi" axis U Σ:



$$
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
3 & 3 & 3 & 0 & 0 \\
4 & 4 & 4 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 2 & 0 & 4 & 4 \\
0 & 0 & 0 & 5 & 5 \\
0 & 1 & 0 & 2 & 2
\end{bmatrix}
\begin{bmatrix}
\boxed{1.61} & 0.19 & -0.01 \\
5.08 & 0.66 & -0.03 \\
6.82 & 0.85 & -0.05 \\
8.43 & 1.04 & -0.06 \\
1.86 & -5.60 & 0.84 \\
0.86 & -6.93 & -0.87 \\
0.86 & -2.75 & 0.41
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\
0.12 & -0.02 & 0.12 & \mathbf{-0.69} & \mathbf{-0.69} \\
0.40 & \mathbf{-0.80} & 0.40 & 0.09 & 0.09
\end{bmatrix}
$$

8.43 → 5th user enjoys sci-fi

SVD: Best Approximation

- How to find the best approximation?

$$\begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\ \mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\ \mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\ \mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\ 0 & \mathbf{2} & 0 & \mathbf{4} & \mathbf{4} \\ 0 & 0 & 0 & \mathbf{5} & \mathbf{5} \\ 0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2} \end{bmatrix} = \begin{bmatrix} \mathbf{0.13} & 0.02 & -0.01 \\ \mathbf{0.41} & 0.07 & -0.03 \\ \mathbf{0.55} & 0.09 & -0.04 \\ \mathbf{0.68} & 0.11 & -0.05 \\ 0.15 & \mathbf{-0.59} & \mathbf{0.65} \\ 0.07 & \mathbf{-0.73} & \mathbf{-0.67} \\ 0.07 & \mathbf{-0.29} & \mathbf{0.32} \end{bmatrix} \times \begin{bmatrix} \mathbf{12.4} & 0 & 0 \\ 0 & \mathbf{9.5} & 0 \\ 0 & 0 & \mathbf{1.3} \end{bmatrix} \times \begin{bmatrix} \mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & \mathbf{-0.69} & \mathbf{-0.69} \\ 0.40 & \mathbf{-0.80} & 0.40 & 0.09 & 0.09 \end{bmatrix}$$

- How to find the best approximation?

Set smallest singular values to zero!

$$
\begin{bmatrix}
\mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\
\mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\
\mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\
\mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\
0 & \mathbf{2} & 0 & \mathbf{4} & \mathbf{4} \\
0 & 0 & 0 & \mathbf{5} & \mathbf{5} \\
0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2}
\end{bmatrix}
=
\begin{bmatrix}
\mathbf{0.13} & 0.02 & -0.01 \\
\mathbf{0.41} & 0.07 & -0.03 \\
\mathbf{0.55} & 0.09 & -0.04 \\
\mathbf{0.68} & 0.11 & -0.05 \\
0.15 & \mathbf{-0.59} & \mathbf{0.65} \\
0.07 & \mathbf{-0.73} & \mathbf{-0.67} \\
0.07 & \mathbf{-0.29} & \mathbf{0.32}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{12.4} & 0 & 0 \\
0 & \mathbf{9.5} & 0 \\
0 & 0 & \cancel{\mathbf{1.3}}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\
0.12 & -0.02 & 0.12 & \mathbf{-0.69} & \mathbf{-0.69} \\
0.40 & \mathbf{-0.80} & 0.40 & 0.09 & 0.09
\end{bmatrix}
$$

NB Interpretation of 1.3 : how spread data around V_3

low variance, hence to be ignored

$$
\begin{bmatrix}
\mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\
\mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\
\mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\
\mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\
0 & \mathbf{2} & 0 & \mathbf{4} & \mathbf{4} \\
0 & 0 & 0 & \mathbf{5} & \mathbf{5} \\
0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2}
\end{bmatrix}
\approx
\begin{bmatrix}
\mathbf{0.13} & 0.02 & \cancel{-0.01} \\
\mathbf{0.41} & 0.07 & \cancel{-0.03} \\
\mathbf{0.55} & 0.09 & \cancel{-0.04} \\
\mathbf{0.68} & 0.11 & \cancel{-0.05} \\
0.15 & \mathbf{-0.59} & \cancel{\mathbf{0.65}} \\
0.07 & \mathbf{-0.73} & \cancel{\mathbf{-0.67}} \\
0.07 & \mathbf{-0.29} & \cancel{\mathbf{0.32}}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{12.4} & 0 & 0 \\
0 & \mathbf{9.5} & 0 \\
0 & 0 & \cancel{\mathbf{1.3}}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\
0.12 & -0.02 & 0.12 & \mathbf{-0.69} & \mathbf{-0.69} \\
\cancel{0.40} & \cancel{\mathbf{-0.80}} & \cancel{0.40} & \cancel{0.09} & \cancel{0.09}
\end{bmatrix}
$$

$$
\begin{bmatrix}
\mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\
\mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\
\mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\
\mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\
0 & \mathbf{2} & 0 & \mathbf{4} & \mathbf{4} \\
0 & 0 & 0 & \mathbf{5} & \mathbf{5} \\
0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2}
\end{bmatrix}
\approx
\underset{7X2}{\begin{bmatrix}
\mathbf{0.13} & 0.02 \\
\mathbf{0.41} & 0.07 \\
\mathbf{0.55} & 0.09 \\
\mathbf{0.68} & 0.11 \\
0.15 & \mathbf{-0.59} \\
0.07 & \mathbf{-0.73} \\
0.07 & \mathbf{-0.29}
\end{bmatrix}}
\times
\underset{2X2}{\begin{bmatrix}
\mathbf{12.4} & 0 \\
0 & \mathbf{9.5}
\end{bmatrix}}
\times
\underset{2X7}{\begin{bmatrix}
\mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\
0.12 & -0.02 & 0.12 & \mathbf{-0.69} & \mathbf{-0.69}
\end{bmatrix}}
$$

* think of A as sum over rank 1 matrices

$$A = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T$$

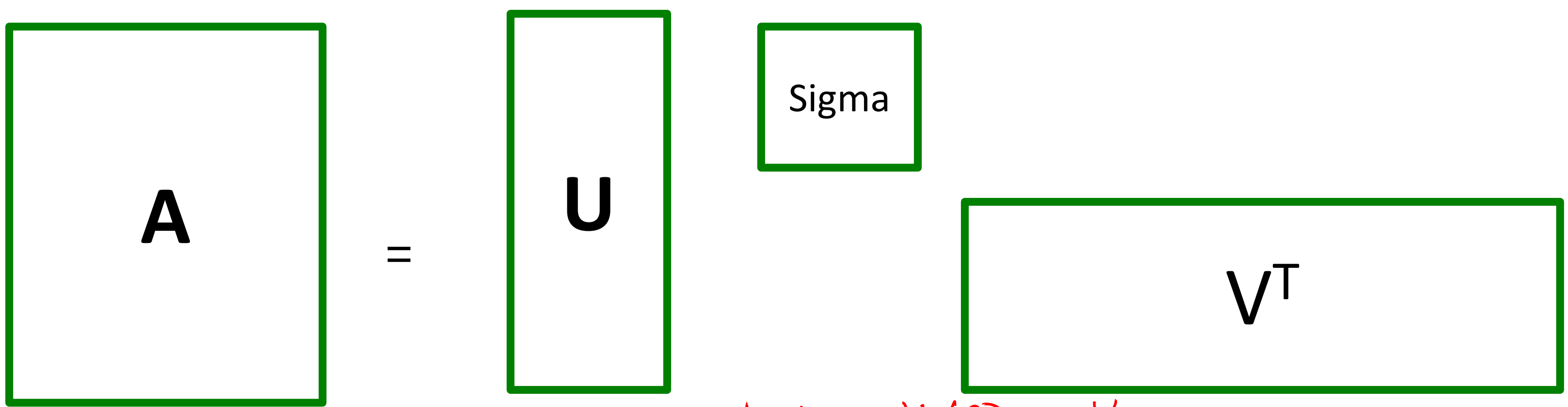
by construction, the rank of the new matrix is 2

$$\begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\ \mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\ \mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\ \mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\ 0 & \mathbf{2} & 0 & \mathbf{4} & \mathbf{4} \\ 0 & 0 & 0 & \mathbf{5} & \mathbf{5} \\ 0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2} \end{bmatrix} \approx \begin{bmatrix} \mathbf{0.92} & \mathbf{0.95} & \mathbf{0.92} & 0.01 & 0.01 \\ \mathbf{2.91} & \mathbf{3.01} & \mathbf{2.91} & -0.01 & -0.01 \\ \mathbf{3.90} & \mathbf{4.04} & \mathbf{3.90} & 0.01 & 0.01 \\ \mathbf{4.82} & \mathbf{5.00} & \mathbf{4.82} & 0.03 & 0.03 \\ 0.70 & \mathbf{0.53} & 0.70 & \mathbf{4.11} & \mathbf{4.11} \\ -0.69 & 1.34 & -0.69 & \mathbf{4.78} & \mathbf{4.78} \\ 0.32 & \mathbf{0.23} & 0.32 & \mathbf{2.01} & \mathbf{2.01} \end{bmatrix}$$

we started with rank 2

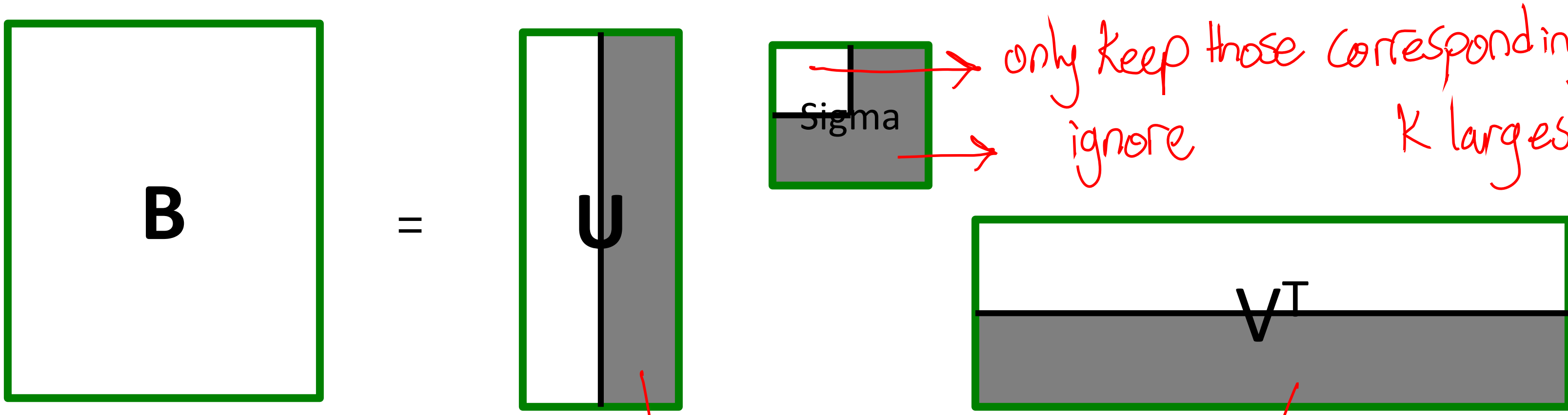
the reason for such good approximation is the value we removed is too small

SVD: Best Low Rank Approximation



best rank(B) = K

B is best approximation of A



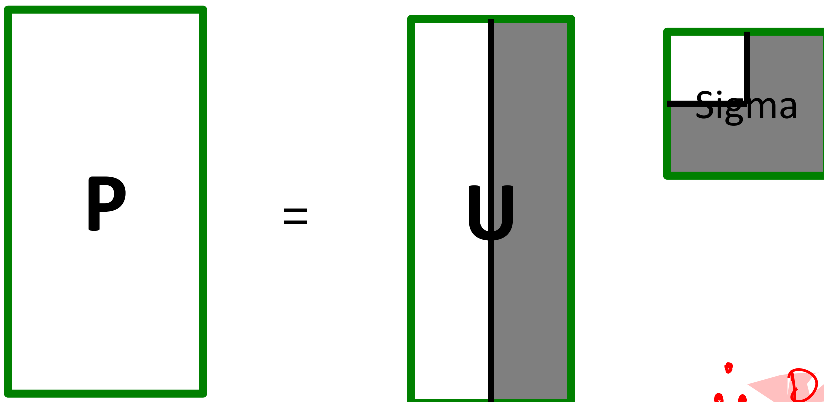
only keep those corresponding to K largest

ignore

also ignore corresponding dim in U & V^T

SVD: Projection

- Note: The actual projected/reduced data can be obtained by computing



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

$\therefore P = U \Sigma V^T V = U \Sigma$

- Or equivalently: $P = A \cdot V$ (since V is orthonormal)

Proj data

original data

right singular values

* No need to calc everything for U, Σ & V^T, just largest K values

Best Approximation – Intuitive Explanation

- Recap: Vectors $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$ are of unit length
- W.l.o.g.: $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \ldots \geq 0$

$$\begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\ \mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\ \mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\ \mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\ 0 & \mathbf{2} & 0 & \mathbf{4} & \mathbf{4} \\ 0 & 0 & 0 & \mathbf{5} & \mathbf{5} \\ 0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2} \end{bmatrix} = \begin{bmatrix} | & | \\ \boldsymbol{u}_1 & \boldsymbol{u}_2 \\ | & | \end{bmatrix} \times \begin{bmatrix} \sigma_1 & \emptyset \\ \emptyset & \sigma_2 \end{bmatrix} \times \begin{bmatrix} — & \boldsymbol{v}_1 & — \\ — & \boldsymbol{v}_2 & — \end{bmatrix}$$

* Even having larges of $u_i \cdot v_i^T$ would make no diff in terms of error
Having both vectors u_i & v_i on the same scale, the resulting matrix after multiplication will also be on the same scale.

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} = \underbrace{\boldsymbol{\sigma}_1 \; \boldsymbol{u}_1 \; \boldsymbol{v}^T{}_1 + \boldsymbol{\sigma}_2 \; \boldsymbol{u}_2 \; \boldsymbol{v}^T{}_2 + \ldots}_{r \text{ terms}}$$

Q: How many $\boldsymbol{\sigma_i}$ to pick?
A: Rule of thumb:
keep 90% of 'energy'
$\sum_{i=1}^{k} \sigma_i^2 \geq 0.9 \sum_{i=1}^{r} \sigma_i^2$

* σ_i scales the terms $\boldsymbol{u}_i \cdot \boldsymbol{v}_i^T$
* Zeroing small σ_i introduces less error

Small $\sigma \rightarrow$ Small error

NB σ_i corresponds to the variance

SVD: Best Low Rank Approximation - Proof

- Theorem: Let A = $U\Sigma V^T$ ($\sigma_1 \geq \sigma_2 \geq \cdots$, rank(A)=r) and B = USV^T with S being a diagonal $r \times r$ matrix where
  - $s_i = \sigma_i$ for i=1…k and $s_i = 0$ else

  Then B is a best rank-k approximation to A regarding Frobenius norm, i.e. B is a solution to $\min_B \|A - B\|_F$ where rank(B)=k

$$\begin{bmatrix} \sigma_1 & & & & & \\ & \ddots & & & & \\ & & \sigma_k & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{bmatrix}$$

flatten the vec,

calc euclidean dist

- We have uploaded a detailed proof to the web
  - Note: Many proofs on the web and on other lecture slides are incorrect!

- Remark: B is also an optimal low-rank approximation regarding the spectral norm (operator 2-norm): $\min_B \|A - B\|_2$
  - $\|X\|_2$ = largest singular value of X

SVD: Best Low Rank Approximation - Proof

- Some facts:
  - $\|X\|_F = \|X^T\|_F$
    - obvious from the definition
  - $\|X\|_F^2 = \text{trace}(X^T X)$ // trace = sum of diagonal entries
    - easy homework
  - Frobenius norm is invariant to orthonormal transformations U
    - Note: If $U^T U = I$ then also $UU^T = I$
    - $\|UX\|_F^2 = \text{trace}\big((UX)^T(UX)\big) = \text{trace}\big(X^T U^T U\text{X}\big) = \text{trace}\big(\text{X}^T X\big) = \|X\|_F^2$
    - $\|XU\|_F^2 = \left\|(XU)^T\right\|_F^2 = \text{trace}\big(((XU)(XU)^T\big)$
      $= \text{trace}\big(XUU^T X^T\big) = \text{trace}\big(XX^T\big) = \|X\|_F^2$
- Let $\text{A} = U\Sigma V^T$ then $\|A\|_F^2 = \|\Sigma\|_F^2 = \sum_i^r \sigma_i^2$
  - follows from above results

SVD: Complexity

- To compute SVD:
  - $O(n \cdot d^2)$ or $O(n^2 \cdot d)$ (whichever is less)

    #rows #cols
- But:
  - Less work, if we just want singular values
  - or if we want first k singular vectors
  - or if the matrix is sparse
- Implemented in linear algebra packages like
  - LINPACK, Matlab, SPlus, Mathematica ...

* SVD is used → in dimensionality reduction
→ solving linear sys of equations

$Ax = b$ A being non-square, → underdetermined
→ overdetermined

SVD & PCA: Comparison

- Given data X (let's assume it is already centered)
- SVD gives us:
  - X = $U\Sigma V^T$
  - Projected data obtained by $X \cdot V$ (or truncated V)
- PCA computes the eigendecomposition of the covariance matrix
  - Covariance matrix: $X^T X$ assuming X already centered
  - Eigendecomposition leads to $X^T X = \Gamma \cdot \Lambda \cdot \Gamma^T$
  - Projected data obtained by $\mathrm{X} \cdot \Gamma$ (or truncated Γ)
- Let us calculate:
  - $X^T X = \left(U\Sigma V^T\right)^T U\Sigma V^T = V\Sigma^T\, U^T\, (U\Sigma V^T) = V\Sigma\Sigma^T\, V^T = V\Sigma^2\, V^T$
  - Γ=V **PCA and SVD are equivalent!**
  - $\Sigma^2 = \Lambda$ **squared singular values are variances in new space!**

V: eigen vec of $X^T X$

Γ: right singular vec of X

SVD & PCA: Comparison

transform the data such that dimensions of new space are uncorrelated + discard (new) dimensions with smallest variance

PCA

=

find optimal low-rank approximation

(regarding Frobenius norm)

SVD

Remark: Computation of SVD

- We can use the eigendecomposition to calculate the singular value decomposition

- $X^T X = \left(U\Sigma V^T\right)^T U\Sigma V^T = V\Sigma^T U^T (U\Sigma V^T) = V\Sigma\Sigma^T V^T = V\Sigma^2 V^T$
  - V = eigenvectors of $X^T X$ right singular vec
- $XX^T = U\Sigma V^T (V\Sigma^T U^T) = U\Sigma\Sigma^T U^T$
  - U = eigenvectors of XX^T left singular vec

- Drawback: Numerically instable
  - better to use specialized algorithms