FSAN/ELEG815: Statistical Learning

Gonzalo R. Arce

Department of Electrical and Computer Engineering
University of Delaware

2. Eigen Analysis, SVD, PCA, and Matrix Completion

Outline

Eigen Analysis

Objective: Utilize tools from linear algebra to characterize and analyze matrices, especially the correlation matrix

- The correlation matrix plays a large role in statistical characterization and processing.
- Previously result: $\mathbf{R}$ is Hermitian.
- Further insight into the correlation matrix is achieved through eigen analysis
  - Eigenvalues and vectors
  - Matrix diagonalization
  - Application: Optimum filtering problems

Objective: For a Hermitian matrix $\mathbf{R}$, find a vector $\mathbf{q}$ satisfying

$$\mathbf{R}\mathbf{q} = \lambda\mathbf{q}$$

- Interpretation: Linear transformation by $\mathbf{R}$ changes the scale, but not the direction of $\mathbf{q}$
- Fact: A $M \times M$ matrix $\mathbf{R}$ has M eigenvectors and eigenvalues

$$\mathbf{R}\mathbf{q}_i = \lambda_i\mathbf{q}_i \quad i = 1, 2, 3, \cdots, M$$

To see this, note

$$(\mathbf{R} - \lambda\mathbf{I})\mathbf{q} = \mathbf{0}$$

For this to be true, the row/columns of $(\mathbf{R} - \lambda\mathbf{I})$ must be linearly dependent,

$$\Rightarrow \det(\mathbf{R} - \lambda\mathbf{I}) = 0$$

Note: $\det(\mathbf{R}-\lambda\mathbf{I})$ is a Mth order polynomial in λ

- The roots of the polynomial are the eigenvalues $\lambda_1, \lambda_2, \cdots, \lambda_M$

$$\mathbf{R}\mathbf{q}_i = \lambda_i \mathbf{q}_i$$

- Each eigenvector $\mathbf{q}_i$ is associated with one eigenvalue λ_i
- The eigenvectors are not unique

$$\begin{aligned} \mathbf{R}\mathbf{q}_i &= \lambda_i \mathbf{q}_i \\ \Rightarrow \mathbf{R}(a\mathbf{q}_i) &= \lambda_i (a\mathbf{q}_i) \end{aligned}$$

Consequence: eigenvectors are generally normalized, e.g., $|\mathbf{q}_i| = 1$ for $i = 1, 2, \ldots, M$

Example (General two dimensional case)

Let $M = 2$ and

$$\mathbf{R} = \begin{bmatrix} R_{1,1} & R_{1,2} \\ R_{2,1} & R_{2,2} \end{bmatrix}$$

Determine the eigenvalues and eigenvectors.

Thus

$$\begin{aligned} \det(\mathbf{R} - \lambda\mathbf{I}) &= 0 \\ \Rightarrow \begin{vmatrix} R_{1,1} - \lambda & R_{1,2} \\ R_{2,1} & R_{2,2} - \lambda \end{vmatrix} &= 0 \\ \Rightarrow \lambda^2 - \lambda(R_{1,1} + R_{2,2}) + (R_{1,1}R_{2,2} - R_{1,2}R_{2,1}) &= 0 \end{aligned}$$

$$\Rightarrow \lambda_{1,2} = \frac{1}{2}\left[(R_{1,1} + R_{2,2}) \pm \sqrt{4R_{1,2}R_{2,1} + (R_{1,1} - R_{2,2})}\right]$$

Back substitution yields the eigenvectors:

$$\begin{bmatrix} R_{1,1}-\lambda & R_{1,2} \\ R_{2,1} & R_{2,2}-\lambda \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

In general, this yields a set of linear equations. In the $M=2$ case:

$$\begin{aligned} (R_{1,1}-\lambda)q_1 + R_{1,2}q_2 =& 0 \\ R_{2,1}q_1 + (R_{2,2}-\lambda)q_2 =& 0 \end{aligned}$$

- Solving the set of linear equations for a specific eigenvalue λ_i yields the corresponding eigenvector, $\mathbf{q}_i$

Example (Two–dimensional white noise)

Let $\mathbf{R}$ be the correlation matrix of a two–sample vector of zero mean white noise

$$\mathbf{R} = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix}$$

Determine the eigenvalues and eigenvectors.

Carrying out the analysis yields eigenvalues

$$\begin{aligned} \lambda_{1,2} &= \frac{1}{2}\left[(R_{1,1}+R_{2,2}) \pm \sqrt{4R_{1,2}R_{2,1}+(R_{1,1}-R_{2,2})}\right] \\ &= \frac{1}{2}\left[(\sigma^2+\sigma^2) \pm \sqrt{0+(\sigma^2-\sigma^2)}\right] = \sigma^2 \end{aligned}$$

and eigenvectors

$$\mathbf{q}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \text{and} \quad \mathbf{q}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Note: The eigenvectors are unit length (and orthogonal)

Eigen Properties

Property (eigenvalues of $\mathbf{R}^k$)

If $\lambda_1, \lambda_2, \cdots, \lambda_M$ are the eigenvalues of $\mathbf{R}$, then $\lambda_1^k, \lambda_2^k, \cdots, \lambda_M^k$ are the eigenvalues of $\mathbf{R}^k$.

Proof: Note $\mathbf{R}\mathbf{q}_i = \lambda_i \mathbf{q}_i$.Multiplying both sides by $\mathbf{R}$ $k-1$ times,

$$\mathbf{R}^k \mathbf{q}_i = \lambda_i \mathbf{R}^{k-1} \mathbf{q}_i = \lambda_i^k \mathbf{q}_i$$

Property (linear independence of eigenvectors)

The eigenvectors $\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_M$, of $\mathbf{R}$ are linearly independent, i.e.,

$$\sum_{i=1}^{M} a_i \mathbf{q}_i \neq \mathbf{0}$$

for all nonzero scalars $a_1, a_2, \cdots, a_M$.

Property (Correlation matrix eigenvalues are real & nonnegative)

The eigenvalues of $\mathbf{R}$ are real and nonnegative.

Proof:

$$\begin{aligned} \mathbf{R}\mathbf{q}_i &= \lambda_i \mathbf{q}_i \\ \Rightarrow \mathbf{q}_i^H \mathbf{R}\mathbf{q}_i &= \lambda_i \mathbf{q}_i^H \mathbf{q}_i \qquad \text{[pre–multiply by } \mathbf{q}_i^H\text{]} \\ \Rightarrow \lambda_i &= \frac{\mathbf{q}_i^H \mathbf{R}\mathbf{q}_i}{\mathbf{q}_i^H \mathbf{q}_i} \geq 0 \end{aligned}$$

Follows from the facts: $\mathbf{R}$ is positive semi-definite and $\mathbf{q}_i^H \mathbf{q}_i = |\mathbf{q_i}|^2 > 0$

Note: In most cases, $\mathbf{R}$ is positive definite and

$$\lambda_i > 0, \qquad i = 1, 2, \cdots, M$$

Property (Unique eigenvalues $\Rightarrow$ orthogonal eigenvectors)

If $\lambda_1, \lambda_2, \cdots, \lambda_M$ are unique eigenvalues of $\mathbf{R}$, then the corresponding eigenvectors, $\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_M$, are orthogonal.

Proof:

$$\begin{aligned} \mathbf{R}\mathbf{q}_i &= \lambda_i \mathbf{q}_i \\ \Rightarrow \mathbf{q}_j^H \mathbf{R}\mathbf{q}_i &= \lambda_i \mathbf{q}_j^H \mathbf{q}_i \qquad\qquad (*) \end{aligned}$$

Also, since λ_j is real and $\mathbf{R}$ is Hermitian

$$\begin{aligned} \mathbf{R}\mathbf{q}_j &= \lambda_j \mathbf{q}_j \\ \Rightarrow \mathbf{q}_j^H \mathbf{R} &= \lambda_j \mathbf{q}_j^H \\ \Rightarrow \mathbf{q}_j^H \mathbf{R}\mathbf{q}_i &= \lambda_j \mathbf{q}_j^H \mathbf{q}_i \end{aligned}$$

Substituting the LHS from $(*)$

$$\Rightarrow \lambda_i \mathbf{q}_j^H \mathbf{q}_i = \lambda_j \mathbf{q}_j^H \mathbf{q}_i$$

Thus

$$\begin{aligned} \lambda_i \mathbf{q}_j^H \mathbf{q}_i &= \lambda_j \mathbf{q}_j^H \mathbf{q}_i \\ \Rightarrow (\lambda_i - \lambda_j) \mathbf{q}_j^H \mathbf{q}_i &= 0 \end{aligned}$$

Since $\lambda_1, \lambda_2, \cdots, \lambda_M$ are unique

$$\mathbf{q}_j^H \mathbf{q}_i = 0 \qquad i \neq j$$

$\Rightarrow \mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_M$ are orthogonal.

QED

Diagonalization of $\mathbf{R}$

Objective: Find a transformation that transforms the correlation matrix into a diagonal matrix.

Let $\lambda_1, \lambda_2, \cdots, \lambda_M$ be unique eigenvectors of $\mathbf{R}$ and take $\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_M$ to be the M orthonormal eigenvectors

$$\mathbf{q}_i^H \mathbf{q}_j = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

Define $\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_M]$ and $\mathbf{\Omega} = \mathrm{diag}(\lambda_1, \lambda_2, \cdots, \lambda_M)$. Then consider

$$\mathbf{Q}^H \mathbf{R} \mathbf{Q} \;=\; \begin{bmatrix} \mathbf{q}_1^H \\ \mathbf{q}_2^H \\ \vdots \\ \mathbf{q}_M^H \end{bmatrix} \mathbf{R} [\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_M]$$

$$
\begin{aligned}
\mathbf{Q}^H\mathbf{R}\mathbf{Q} &= \begin{bmatrix} \mathbf{q}_1^H \\ \mathbf{q}_2^H \\ \vdots \\ \mathbf{q}_M^H \end{bmatrix} \mathbf{R}[\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_M] \\
&= \begin{bmatrix} \mathbf{q}_1^H \\ \mathbf{q}_2^H \\ \vdots \\ \mathbf{q}_M^H \end{bmatrix} [\lambda_1\mathbf{q}_1, \lambda_2\mathbf{q}_2, \cdots, \lambda_N\mathbf{q}_M] \\
&= \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_M \end{bmatrix} \\
\Rightarrow \mathbf{Q}^H\mathbf{R}\mathbf{Q} &= \mathbf{\Omega} \quad \text{(eigenvector diagonalization of } \mathbf{R}\text{)}
\end{aligned}
$$

Property ($\mathbf{Q}$ is unitary)

$\mathbf{Q}$ is unitary, i.e., $\mathbf{Q}^{-1} = \mathbf{Q}^H$

Proof: Since the $\mathbf{q}_i$ eigenvectors are orthonormal

$$\begin{aligned} \mathbf{Q}^H\mathbf{Q} &= \begin{bmatrix} \mathbf{q}_1^H \\ \mathbf{q}_2^H \\ \vdots \\ \mathbf{q}_M^H \end{bmatrix} [\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_M] = \mathbf{I} \\ \Rightarrow \mathbf{Q}^{-1} &= \mathbf{Q}^H \end{aligned}$$

Property (Eigen decomposition of $\mathbf{R}$)

The correlation matrix can be expressed as

$$\mathbf{R} = \sum_{i=1}^{M} \lambda_i \mathbf{q}_i \mathbf{q}_i^H$$

Proof: The correlation diagonalization result states

$$\mathbf{Q}^H\mathbf{R}\mathbf{Q} = \mathbf{\Omega}$$

Isolating $\mathbf{R}$ and expanding,

$$\begin{aligned}
\mathbf{R} &= \mathbf{Q}\mathbf{\Omega}Q^H = [\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_M]\mathbf{\Omega}\begin{bmatrix} \mathbf{q}_1^H \\ \mathbf{q}_2^H \\ \vdots \\ \mathbf{q}_M^H \end{bmatrix} \\
&= [\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_M]\begin{bmatrix} \lambda_1\mathbf{q}_1^H \\ \lambda_2\mathbf{q}_2^H \\ \vdots \\ \lambda_M\mathbf{q}_M^H \end{bmatrix} = \sum_{i=1}^{M}\lambda_i\mathbf{q}_i\mathbf{q}_i^H
\end{aligned}$$

Aside (trace & determinant for matrix products)

Note $\text{trace}(\boldsymbol{A}) \triangleq \sum_i A_{i,i}$. Also,

$$\text{trace}(\boldsymbol{AB}) = \text{trace}(\boldsymbol{BA}) \qquad \text{similarly} \qquad \det(\boldsymbol{AB}) = \det(\boldsymbol{A})\det(\boldsymbol{B})$$

Property (Determinant–Eigenvalue Relation)

The determinant of the correlation matrix is related to the eigenvalues as follows:

$$\det(\mathbf{R}) = \prod_{i=1}^{M} \lambda_i$$

Proof: Using $\mathbf{R} = \boldsymbol{Q\Omega Q}^H$ and the above,

$$\begin{aligned}\det(\mathbf{R}) &= \det(\boldsymbol{Q\Omega Q}^H) \\ &= \det(\mathbf{Q})\det(\mathbf{Q}^H)\det(\boldsymbol{\Omega}) = \det(\boldsymbol{\Omega}) = \prod_{i=1}^{M} \lambda_i\end{aligned}$$

Property (Trace–Eigenvalue Relation)

The trace of the correlation matrix is related to the eigenvalues as follows:

$$\text{trace}(\mathbf{R}) = \sum_{i=1}^{M} \lambda_i$$

Proof: Note

$$\begin{aligned} \text{trace}(\mathbf{R}) &= \text{trace}(\boldsymbol{Q}\boldsymbol{\Omega}\boldsymbol{Q}^H) \\ &= \text{trace}(\mathbf{Q}^H\boldsymbol{Q}\boldsymbol{\Omega}) \\ &= \text{trace}(\boldsymbol{\Omega}) \\ &= \sum_{i=1}^{M} \lambda_i \end{aligned}$$

QED

Matrix-Vector Multiplication

Example in 2D:

$$\mathbf{A} = \begin{bmatrix} 2 & 1 \\ -1 & 1 \end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

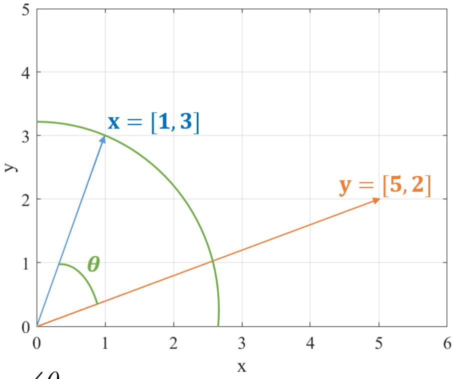
and,

$$\mathbf{y} = \mathbf{A}\mathbf{x} = \begin{bmatrix} 2 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \end{bmatrix} = \begin{bmatrix} 5 \\ 2 \end{bmatrix}$$

What is the geometrical meaning of the matrix-vector multiplication?

Matrix-Vector Multiplication

$$\mathbf{y} = \mathbf{A}\mathbf{x} = \begin{bmatrix} 2 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \end{bmatrix} = \begin{bmatrix} 5 \\ 2 \end{bmatrix}$$

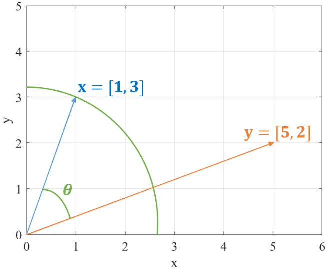


- Rotates the vector $\angle\theta$
- Stretches the vector

Matrix-Vector Multiplication

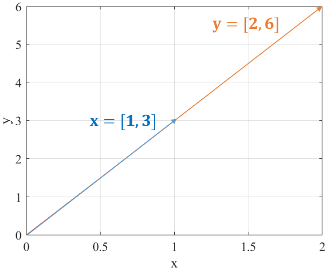
To rotate **x** by an angle θ, we pre-multiply by

$$\mathbf{A} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$



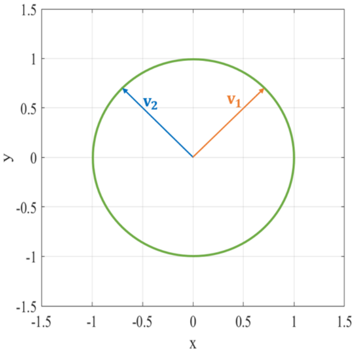
Stretch **x** by factor α, pre-multiply by

$$\mathbf{A} = \begin{bmatrix} \alpha & 0 \\ 0 & \alpha \end{bmatrix}$$



Matrix-Vector Multiplication

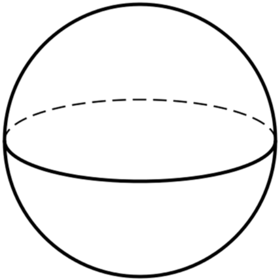
Consider the vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ depicting a circle. What happens to the circle under matrix multiplication?



2-D Circle

$\mathbf{A}[\mathbf{v}_1 \quad \mathbf{v}_2]$

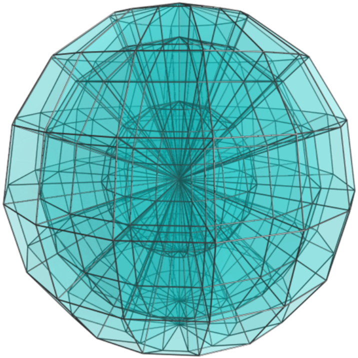
$\mathbf{v}_i \in \mathbb{C}^2$



3-D Sphere

$\mathbf{A}[\mathbf{v}_1 \quad \mathbf{v}_2 \quad \mathbf{v}_3]$

$\mathbf{v}_i \in \mathbb{C}^3$



n-D Hypersphere

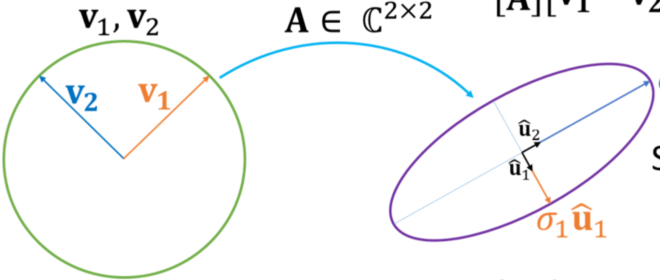
$\mathbf{A}[\mathbf{v}_1 \quad \cdots \quad \mathbf{v}_n]$

$\mathbf{v}_i \in \mathbb{C}^n$

Matrix-Vector Multiplication

What happens to the 2D circle under matrix multiplication?

$$[\mathbf{A}][\mathbf{v_1} \quad \mathbf{v_2}] = [\hat{\mathbf{u}}_1 \quad \hat{\mathbf{u}}_2]\begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix}$$



Principal axis

Singular values

$\hat{\mathbf{u}}_1, \hat{\mathbf{u}}_2$ Unitary orthonormal vectors

σ_1, σ_2 "Stretching" constant

Note: Ortogonality holds since they are all rotated by the same angle.

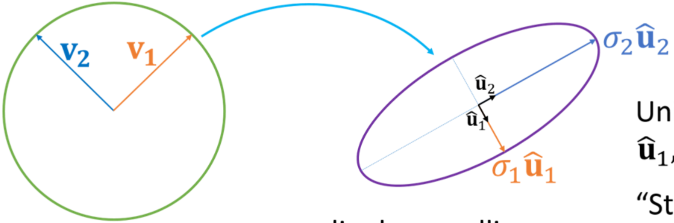
Matrix-Vector Multiplication

What happens to the n-D hyper-sphere under matrix multiplication?

$$\mathbf{A}\mathbf{v}_{\mathrm{j}} = \sigma_j \hat{\mathbf{u}}_j$$

$$[\mathbf{A}][\mathbf{v}_1 \quad \dots \quad \mathbf{v}_{\mathrm{n}}] = [\hat{\mathbf{u}}_{\mathrm{j}} \quad \dots \quad \hat{\mathbf{u}}_{\mathrm{j}}]\begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_n \end{bmatrix}$$

$\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n$ $\quad$ $\mathbf{A} \in \mathbb{C}^{m \times n}$



n-dim hyper-ellipse

Unitary orthonormal vectors
$\hat{\mathbf{u}}_1, \hat{\mathbf{u}}_2, \dots, \hat{\mathbf{u}}_n$

“Stretching” constant
$\sigma_1, \sigma_2, \dots, \sigma_{\mathrm{n}}$

n-dim Hyper-Sphere Mapping to n-dim Hyper-Ellipsoid

The mapping can be written as

$$\begin{array}{rcl} \mathbf{A}\mathbf{v}_1 & = & \sigma_1 \hat{\mathbf{u}}_1 \\ \vdots & & \vdots \\ \mathbf{A}\mathbf{v}_n & = & \sigma_j \hat{\mathbf{u}}_n \end{array}$$

Expressed in matrix form as

$$\underbrace{\begin{bmatrix} \\ \mathbf{A} \\ \\ \end{bmatrix}}_{\mathbf{A} \in \mathbb{C}^{m\times n}} \underbrace{\begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \ldots \mathbf{v}_n \end{bmatrix}}_{\mathbf{V}\ \mathbb{C}^{n\times n}} = \underbrace{\begin{bmatrix} \hat{\mathbf{u}}_1 & \hat{\mathbf{u}}_2 & \ldots \hat{\mathbf{u}}_n \end{bmatrix}}_{\hat{\mathbf{U}}\ \mathbb{C}^{m\times n}} \underbrace{\begin{bmatrix} \sigma_1 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & \sigma_n \end{bmatrix}}_{\hat{\mathbf{\Sigma}}\ \mathbb{C}^{n\times n}}$$

$$\mathbf{A}\mathbf{V} = \hat{\mathbf{U}}\hat{\mathbf{\Sigma}}$$

n-dim Hyper-Sphere Mapping to n-dim Hyper-Ellipsoid

Let $\mathbf{v}_1, \ldots, \mathbf{v}_n$ be unitary orthonormal vectors, then $\mathbf{V} = [\mathbf{v}_1 \; \mathbf{v}_2 \; \ldots \; \mathbf{v}_n]$ is a unitary transformation matrix, that is

$$\mathbf{V}^{-1} = \mathbf{V}^H.$$

Let $\hat{\mathbf{u}}_1, \ldots, \hat{\mathbf{u}}_n$ be unitary orthonormal vectors, then $\hat{\mathbf{U}} = [\hat{\mathbf{u}}_1 \; \hat{\mathbf{u}}_2 \; \ldots \; \hat{\mathbf{u}}_n]$ is a unitary transformation matrix, that is

$$\mathbf{U}^{-1} = \hat{\mathbf{U}}^H.$$

Reduced Singular Value Decomposition

The mapping is thus given by,

$$\mathbf{AV} = \hat{\mathbf{U}}\hat{\Sigma}$$

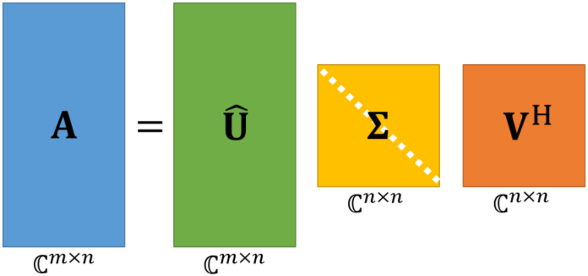
Multiply both sides by $\mathbf{V}^{-1}$ we obtain:

$$\begin{aligned} \mathbf{AVV}^{-1} &= \hat{\mathbf{U}}\hat{\Sigma}\mathbf{V}^{-1} \\ \mathbf{AVV}^{H} &= \hat{\mathbf{U}}\hat{\Sigma}\mathbf{V}^{H} \\ \mathbf{AI} &= \hat{\mathbf{U}}\hat{\Sigma}\mathbf{V}^{H} \\ \mathbf{A} &= \hat{\mathbf{U}}\hat{\Sigma}\mathbf{V}^{H} \end{aligned}$$

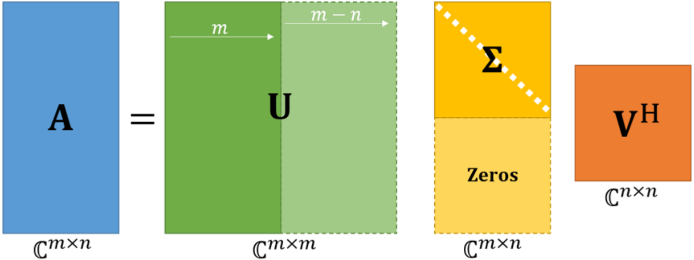
where $\mathbf{\Sigma} = \text{diag}\left([\sigma_1, \sigma_2, \ldots, \sigma_n]\right)$, such that $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_p \geq 0$.

Singular Value Decomposition

- Reduced SVD



- SVD



Theorem 1

Every matrix $\mathbf{A} \in \mathbb{C}^{m\times n}$ has a singular value decomposition (SVD).

- Singular values σ_j are uniquely determined.
- If $\mathbf{A}$ is square σ_j are distinct.
- $\mathbf{u}_j$ and $\mathbf{v}_j$ are also unique up to a complex sign. (unique if the complex sign is ignored)

SVD calculation

Start with $\mathbf{A}^\mathsf{T}\mathbf{A}$:

$$\begin{aligned}
\mathbf{A}^\mathsf{H}\mathbf{A} &= \left(\mathbf{U}\Sigma\mathbf{V}^{H}\right)^\mathsf{H}\left(\mathbf{U}\Sigma\mathbf{V}^\mathsf{H}\right) \\
&= \mathbf{V}\Sigma\mathbf{U}^\mathsf{H}\mathbf{U}\Sigma\mathbf{V}^{H} \\
\mathbf{A}^\mathsf{H}\mathbf{A}\mathbf{V} &= \mathbf{V}\Sigma^2\mathbf{V}^\mathsf{H}\mathbf{V} \\
\mathbf{A}^\mathsf{H}\mathbf{A}\mathbf{V} &= \mathbf{V}\Sigma^2
\end{aligned}$$

Reduces to an eigenvalue decomposition problem of the form:

$$\underbrace{\mathbf{A}^\mathsf{T}\mathbf{A}}_{\mathbf{B}}\mathbf{V} = \mathbf{V}\underbrace{\Sigma^2}_{\mathbf{\Lambda}},$$

where $\mathbf{\Lambda}$ is a diagonal matrix with the eigenvalues of $\mathbf{B}$ and $\mathbf{V}$ corresponds to the eigenvectors of $\mathbf{B}$.

SVD calculation

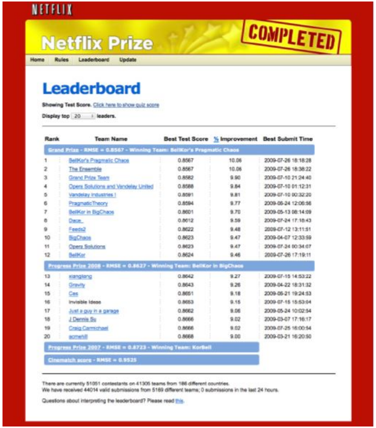
How do we calculate **U**:

$$\begin{aligned}
\mathbf{A}\mathbf{A}^{\mathsf{H}} &= \left(\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathsf{H}}\right)\left(\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathsf{H}}\right)^{\mathsf{H}} \\
&= \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathsf{H}}\mathbf{V}\boldsymbol{\Sigma}\mathbf{U}^{\mathsf{H}} \\
\mathbf{A}\mathbf{A}^{\mathsf{H}}\mathbf{U} &= \mathbf{U}\boldsymbol{\Sigma}^2\mathbf{U}^{\mathsf{H}}\mathbf{U} \\
\underbrace{\mathbf{A}\mathbf{A}^{\mathsf{H}}}_{\mathbf{B}}\mathbf{U} &= \mathbf{U}\underbrace{\boldsymbol{\Sigma}^2}_{\boldsymbol{\Lambda}}
\end{aligned}$$

Eigenvalue problem where $\boldsymbol{\Lambda}$ is a diagonal matrix with the eigenvalues of **B** and **U** corresponds to the eigenvectors of **B**.

Netflix Movie Challenge

- Dataset: $n = 17,770$ movies (columns) and $m = 480,189$ customers (rows).
- Customers rated movies on a scale from 1 to 5. Matrix is very sparse with "only" 100 million of the ratings present in the training set.
- Goal: Predict the ratings for unrated movies.



NETFLIX

Netflix Prize COMPLETED

Home Rules Leaderboard Update

Leaderboard

Showing Test Score. Click here to show quiz score

Display top 20 leaders.

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	0.8582	9.90	2009-07-10 21:24:40
4	Opera Solutions and Vandelay United	0.8588	9.84	2009-07-10 01:12:31
5	Vandelay Industries !	0.8591	9.81	2009-07-10 00:32:20
6	PragmaticTheory	0.8594	9.77	2009-06-24 12:06:56
7	BellKor in BigChaos	0.8601	9.70	2009-05-13 08:14:09
8	Dace_	0.8612	9.59	2009-07-24 17:18:43
9	Feeds2	0.8622	9.48	2009-07-12 13:11:51
10	BigChaos	0.8623	9.47	2009-04-07 12:33:59
11	Opera Solutions	0.8623	9.47	2009-07-24 00:34:07
12	BellKor	0.8624	9.46	2009-07-26 17:19:11
Progress Prize 2008 - RMSE = 0.8627 - Winning Team: BellKor in BigChaos				
13	xiangliang	0.8642	9.27	2009-07-15 14:53:22
14	Gravity	0.8643	9.26	2009-04-22 18:31:32
15	Ces	0.8651	9.18	2009-06-21 19:24:53
16	Invisible Ideas	0.8653	9.15	2009-07-15 15:53:04
17	Just a guy in a garage	0.8662	9.06	2009-05-24 10:02:54
18	J Dennis Su	0.8666	9.02	2009-03-07 17:16:17
19	Craig Carmichael	0.8666	9.02	2009-07-25 16:00:54
20	acmehill	0.8668	9.00	2009-03-21 16:20:50
Progress Prize 2007 - RMSE = 0.8723 - Winning Team: KorBell				
Cinematch score - RMSE = 0.9525				

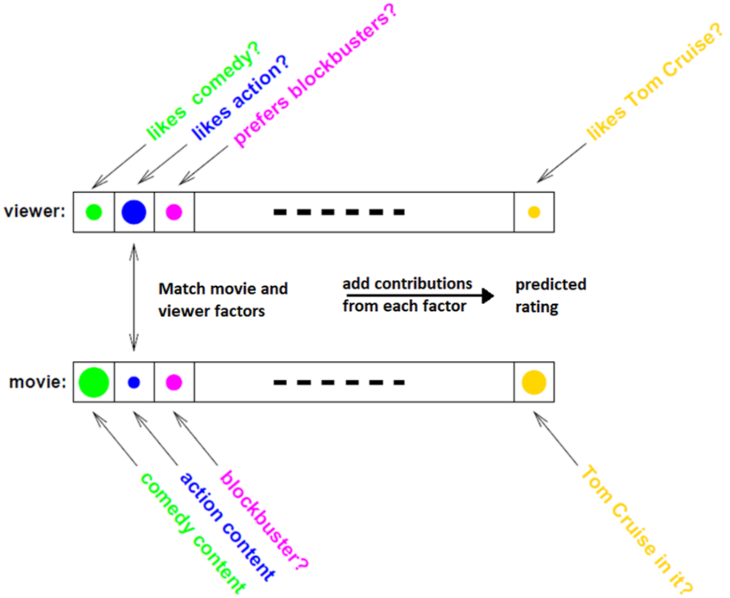
There are currently 51051 contestants on 41305 teams from 186 different countries.
We have received 44014 valid submissions from 5169 different teams; 0 submissions in the last 24 hours.

Questions about interpreting the leaderboard? Please read this.

- (2006) "Cinematch" algorithm used by Netflix RMSE=0.9525 over a large test set.
- Competition started in 2006, winner should improve this RMSE by at least 10%.
- 2009 "Bellkor's Pragmatic Chaos," uses a combination of many statistical techniques to win.

Movie Rating - A Solution

- Describe a movie as an array of factors, e.g. comedy, action...
- Describe each viewer using same factors, e.g. likes comedy, likes action, etc
- Rating based on match/mismatch
- More factors $\rightarrow$ better prediction



Singular Value Decomposition Solution

Viewers rated movies on a scale from 1 to 5. 0 for movies that were not rated by the user.

- Each column j is a different movie
- Each row i is a different viewer
- Each element $a_{i,j}$ represents the rating of movie j by viewer i

	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5
Viewer 1	0	1	0	0	5
Viewer 2	4	2	0	0	0
Viewer 3	0	0	3	3	0
Viewer 4	4	2	0	0	0
Viewer 5	0	0	0	0	5
Viewer 6	0	0	3	3	0
Viewer 7	1	0	0	0	4
Viewer 8	2	1	0	0	4
Viewer 9	1	0	0	0	4

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & \cdots & \cdots & a_{1,n} \\ \vdots & \ddots & \ddots & \vdots \\ a_{m,1} & \cdots & \cdots & a_{m,n} \end{bmatrix}$$

Goal: Use SVD to predict unobserved data or the rating of a movie that hasn't come out yet.

Singular Value Decomposition Solution

We want to classify Movies and Viewers:

$$Movies = \begin{cases} \text{Category 1} \\ \text{Category 2} \\ \text{Category 3} \\ \vdots \end{cases}$$

Intuitively, if $Movie_1 \approx Movie_2$, these movies are similar (same category).

	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5
Viewer 1	0	1	0	0	5
Viewer 2	4	2	0	0	0
Viewer 3	0	0	3	3	0
Viewer 4	4	2	0	0	0
Viewer 5	0	0	0	0	5
Viewer 6	0	0	3	3	0
Viewer 7	1	0	0	0	4
Viewer 8	2	1	0	0	4
Viewer 9	1	0	0	0	4

Categories are determined by matrix $\mathbf{A}$ and SVD algorithm.

Singular Value Decomposition Solution

Now, consider that each movie belongs to more than one category e.g. half comedy and half action. This can be written as:

$$Movie_j = \; v_1\text{Cat1} + v_2\text{Cat2} + \cdots + v_n\text{Catn}$$
$$\text{s.t.}||\mathbf{v}||_2 = 1$$

where the set of categories $\{\text{Cat}j \in \mathbb{R}^{n\times 1}\}$ forms an orthonormal basis.

$$\text{Cat} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & 1 \end{bmatrix}_{n\times n}$$

	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5
Viewer 1	0	1	0	0	5
Viewer 2	4	2	0	0	0
Viewer 3	0	0	3	3	0
Viewer 4	4	2	0	0	0
Viewer 5	0	0	0	0	5
Viewer 6	0	0	3	3	0
Viewer 7	1	0	0	0	4
Viewer 8	2	1	0	0	4
Viewer 9	1	0	0	0	4

Singular Value Decomposition Solution

In the case of $Viewers$, we use the same $Movies$' categories:

$$Movies = \left\{ \begin{array}{c} \text{Category 1} \\ \text{Category 2} \\ \text{Category 3} \\ \vdots \end{array} \right\} = Viewers.$$

	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5
Viewer 1	0	1	0	0	5
Viewer 2	4	2	0	0	0
Viewer 3	0	0	3	3	0
Viewer 4	4	2	0	0	0
Viewer 5	0	0	0	0	5
Viewer 6	0	0	3	3	0
Viewer 7	1	0	0	0	4
Viewer 8	2	1	0	0	4
Viewer 9	1	0	0	0	4

E.g. a viewer that loves comedy is represented with the same unit vector of the comedy category movies (Cat$i \in \mathbb{R}^{1\times n}$) .
Each $Viewer$ is represented as:

$$Viewer_i = \quad u_1\text{Cat1} + u_2\text{Cat2} + \cdots + u_n\text{Catn}$$
$$\text{s.t.}||\mathbf{u}||_2 = 1$$

Singular Value Decomposition Solution

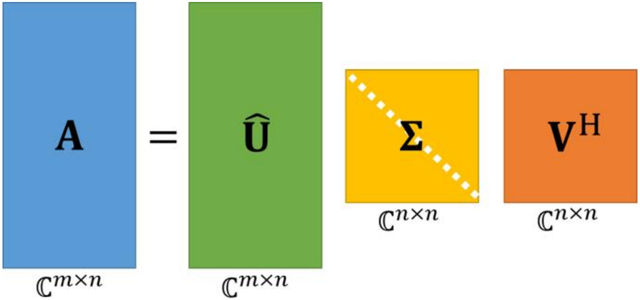
If $m > n$ i.e # of Viewers $>$ # of Movies, each $Viewer$ is represented as:

$$Viewer_i = \quad u_1\text{Cat1} + u_2\text{Cat2} + \cdots + u_n\text{Catn} + \cdots + u_m\text{Catm}$$
$$\text{s.t.}||\mathbf{u}||_2 = 1$$

where Cat$i \in \mathbb{R}^{1\times m}$. Thus, useless categories vectors with zero rating value are added.

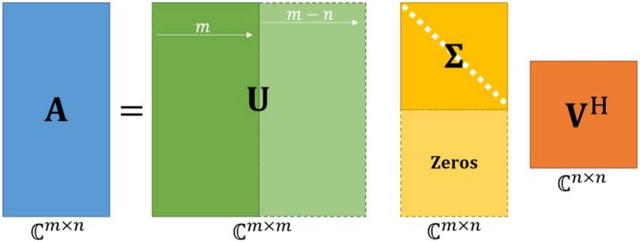
Singular Value Decomposition Solution

From Theorem 1:
There exist a unique decomposition into categories. Every matrix $\mathbf{A} \in \mathbb{C}^{m \times n}$ can be factorized as $\mathbf{A} = \hat{\mathbf{U}}\Sigma\mathbf{V}^H$ where:



Singular Value Decomposition Solution

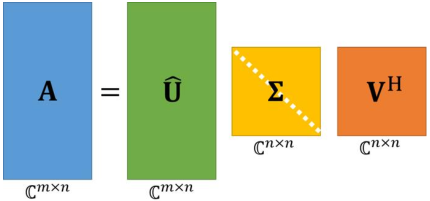
We have more viewers than movies:



New categories are created. The new vectors are still unit vectors orthonormal to all the basis vectors but the ratings of these useless categories are zero.

Note: consider reduced SVD i.e. consider only useful categories.

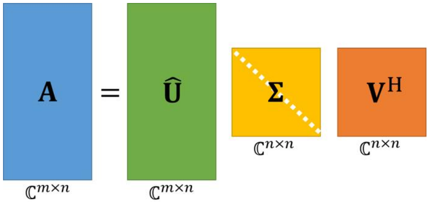
Singular Value Decomposition Solution



- Each row vector ($\mathbf{u}_i$) in $\hat{\mathbf{U}}$ represents the taste of a $Viewer_i$ on the corresponding categories.

$$\hat{\mathbf{U}} = \begin{bmatrix} u_{1,1} & \cdots & \cdots & u_{1,n} \\ \vdots & \ddots & \ddots & \vdots \\ u_{m,1} & \cdots & \cdots & u_{m,n} \end{bmatrix} = \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_m \end{bmatrix}$$

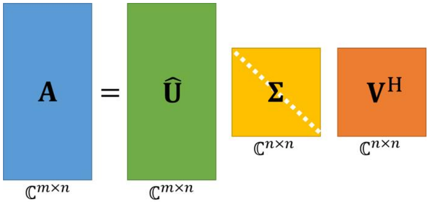
Singular Value Decomposition Solution



- Each column ($\mathbf{v}_j$) in $\mathbf{V}^H$ represents the content of a $Movie_j$ on the corresponding categories.

$$\mathbf{V}^H = \begin{bmatrix} v_{1,1} & \cdots & \cdots & v_{1,n} \\ \vdots & \ddots & \ddots & \vdots \\ v_{n,1} & \cdots & \cdots & v_{n,n} \end{bmatrix} = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \ldots & \mathbf{v}_n \end{bmatrix}$$

Singular Value Decomposition Solution



- Each singular value σ_{ii} in Σ computes how a viewer of category i rates a movie of the same category i.

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_{1,1} & 0 & \cdots & 0 \\ \vdots & \sigma_{2,2} & \ddots & \vdots \\ 0 & 0 & \ldots & \sigma_{n,n} \end{bmatrix}$$

Singular Value Decomposition Solution

The representation of each movie can be obtained by

$$
\begin{aligned}
Movie_j &= v_{1,j}\text{Cat1} + v_{2,j}\text{Cat2} + \cdots + v_{n,j}\text{Catn} \qquad \text{s.t.} ||\mathbf{v}_j||_2 = 1 \\
&= v_{1,j}\begin{bmatrix} \sqrt{\sigma_{1,1}} \\ 0 \\ \vdots \\ 0 \end{bmatrix} + v_{2,j}\begin{bmatrix} 0 \\ \sqrt{\sigma_{2,2}} \\ \vdots \\ 0 \end{bmatrix} + \cdots + v_{n,j}\begin{bmatrix} 0 \\ 0 \\ \vdots \\ \sqrt{\sigma_{n,n}} \end{bmatrix} \\
&= \sqrt{\Sigma}\mathbf{v}_j \qquad \in \mathbb{C}^{n\times 1}
\end{aligned}
$$

Singular Value Decomposition Solution

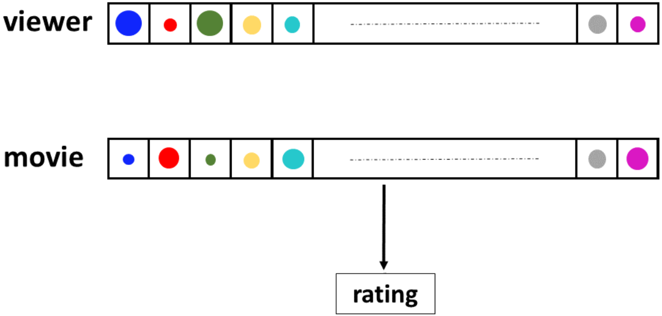
The representation of each viewer can be obtained by

$$
\begin{aligned}
Viewer_i &= u_{i,1}\text{Cat1} + u_{i,2}\text{Cat2} + \cdots + u_{i,n}\text{Catn} + \cdots + u_{i,m}\text{Catm} \\
&\qquad \text{s.t.} ||\mathbf{u}_i||_2 = 1, \qquad \text{Cat}j\text{=0 for } j > n \rightarrow \text{useless categories} \\
&= u_{i,1}\begin{bmatrix} \sqrt{\sigma_{1,1}} \\ 0 \\ \vdots \\ 0 \end{bmatrix}^H + u_{i,2}\begin{bmatrix} 0 \\ \sqrt{\sigma_{2,2}} \\ \vdots \\ 0 \end{bmatrix}^H + \cdots + u_{i,n}\begin{bmatrix} 0 \\ 0 \\ \vdots \\ \sqrt{\sigma_{n,n}} \end{bmatrix}^H \\
&= \mathbf{u}_i\sqrt{\boldsymbol{\Sigma}}^H \qquad \in \mathbb{C}^{1\times n}
\end{aligned}
$$

Singular Value Decomposition Solution

Given the decomposition of a movie and a viewer, the rating is estimated by:

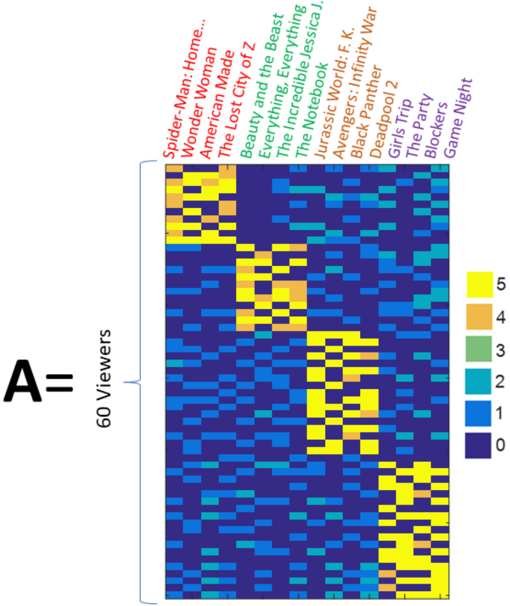
$$\begin{aligned} Viewer_i Movie_j &= u_{i,1}v_{1,j}\sigma_{1,1} + u_{i,2}v_{2,j}\sigma_{2,2} + \cdots + u_{i,n}v_{n,j}\sigma_{n,n} \\ &= (\mathbf{u}_i\sqrt{\mathbf{\Sigma}}^H)(\sqrt{\mathbf{\Sigma}}\mathbf{v}_j) \\ &= \mathbf{u}_i\mathbf{\Sigma}\mathbf{v}_j \end{aligned}$$



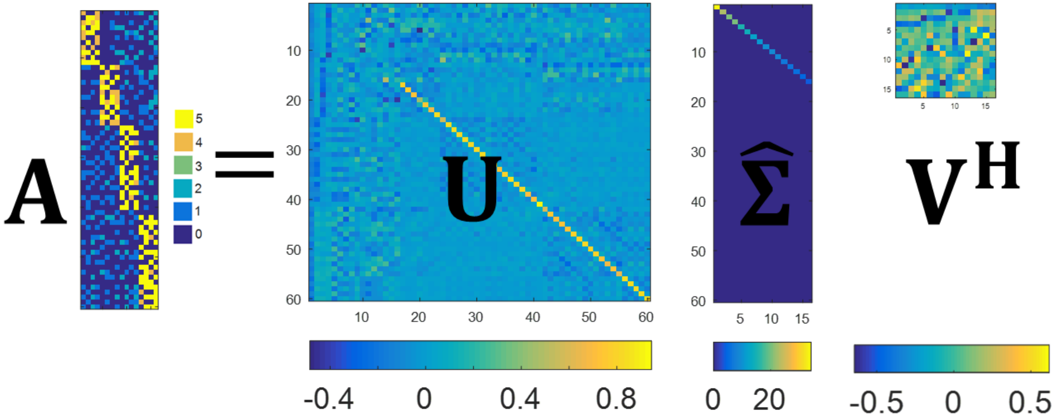
Singular Value Decomposition - Example

Considering the rating from 60 viewers to 16 movies of 4 different genres(action, romance, sci-fi, comedy), we generate $\mathbf{A} \in \mathbb{R}^{60\times 16}$

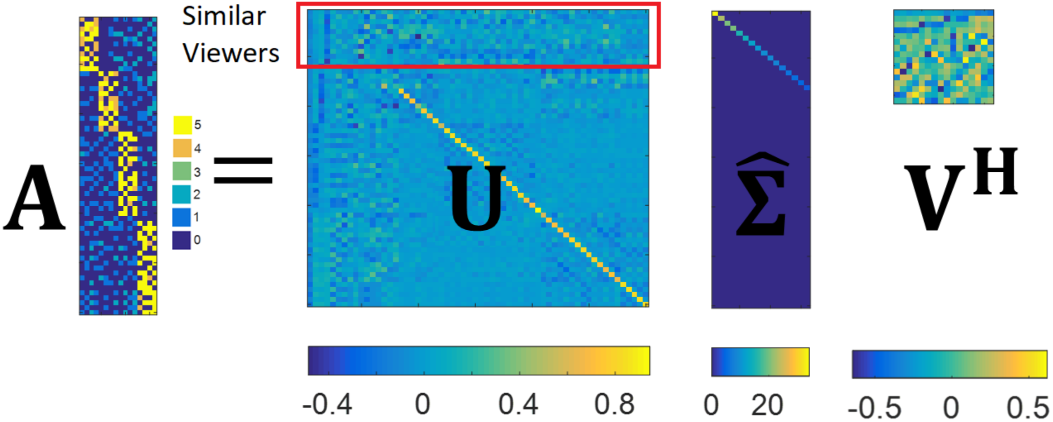
- Viewers rated movies on a scale from 1 to 5, 0 for movies that were not rated by the user.
- Observe the same 4 categories of viewers.



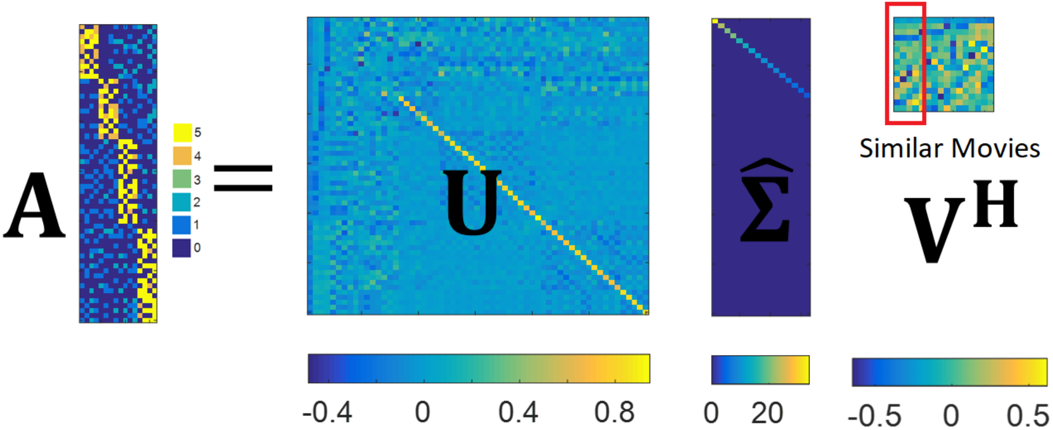
Singular Value Decomposition - Example



Singular Value Decomposition - Example



Singular Value Decomposition - Example



Singular Value Decomposition - Example

To estimate not rated movies (zero entries in $\mathbf{A}$), we use additional information: $\mathbf{A}$ is known to be low-rank or approximately low-rank.

Thus, we are going to use the k-rank approximation of the matrix $\mathbf{A}$ that is:

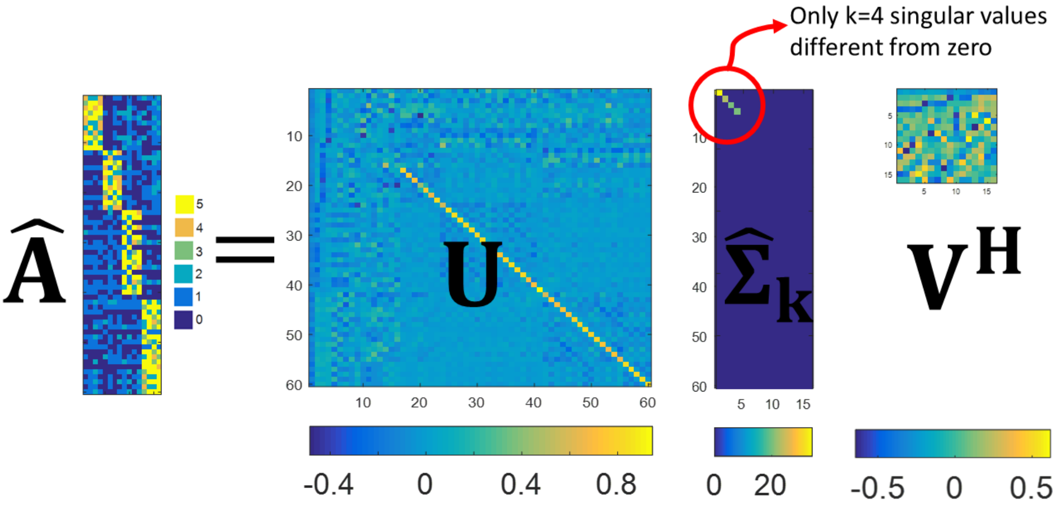
$$\hat{\mathbf{A}} = \mathbf{U}\hat{\Sigma}_k\mathbf{V}^H$$

where $\hat{\Sigma}_k$ has all but the first k singular values σ_{ii} set to zero.

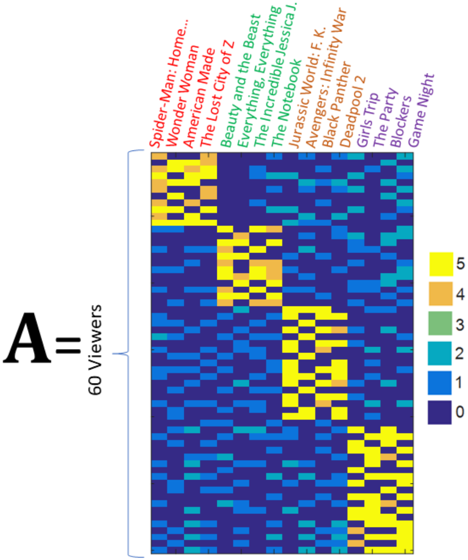
The ratings different from zero in $\mathbf{A}$ are set to its original value.

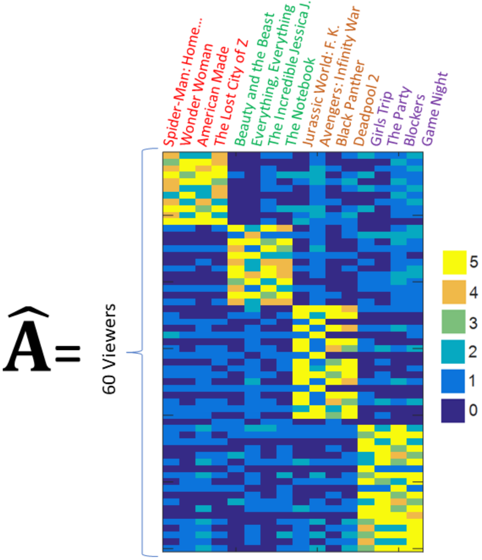
Note: The ratings matrix $\mathbf{A}$ is expected to be low-rank since user preferences can be described by a few categories (k), such as the movie genres.

Singular Value Decomposition - Example



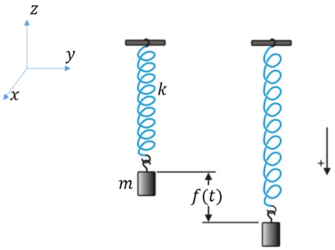
Singular Value Decomposition - Example





Principal Component Analysis (PCA)

- Simple, method for extracting relevant information from confusing data sets.
- How to reduce a complex data set to a lower dimension?
- Consider a mass attached to a spring which oscillates as shown below.

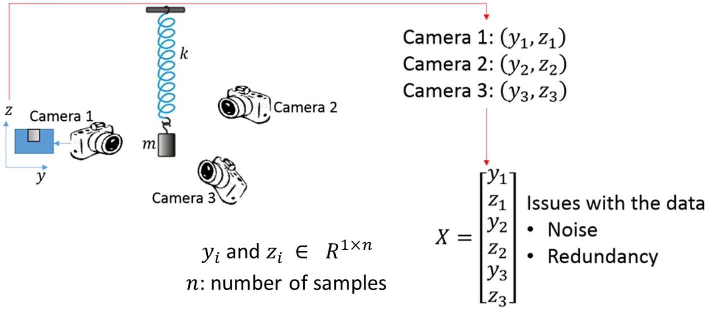


$$F = ma$$
$$-w^2 f = m\frac{d^2 f}{dt}$$
$$f(t) = A\cos(wt + w_0)$$

What if we did not know that $F = ma$?

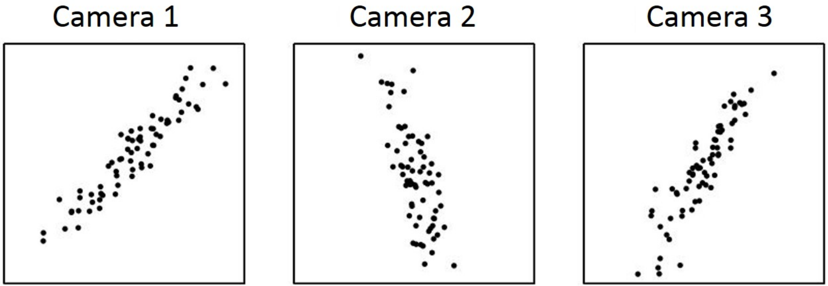
PCA - Motivation: Toy example

- Since we live in a 3D world $\rightarrow$ use three cameras to capture data from the system.
- No information about the real x,y, and z axes $\rightarrow$ camera positions are chosen arbitrarily.
- How do we get from this data set to a simple equation of z ?



PCA - Motivation: Toy example

- Three cameras give redundant information.
- Only one camera at a specific angle necessary to describe the system behavior.
- PCA is used to avoid redundancy.



Change of Basis

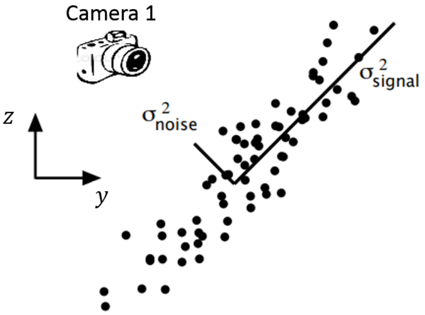
- PCA: Is there another basis, which is a linear combination of the original basis, that best respresents the data set?
- Let $\mathbf{X}$ be the original data set, where each column is a single measurements set.
- Let $\mathbf{Y}$ be a linear transformation by $\mathbf{P}$, i.e. $\mathbf{Y} = \mathbf{PX}$, where $\mathbf{X} = [\mathbf{x}_1 | \ldots | \mathbf{x}_n]$ and $\mathbf{x}_i \in \mathbb{R}^{m\times 1}$ represents a sampled vector.

Implications:

- Geometrically $\mathbf{P}$ is a rotation and a stretch which transforms $\mathbf{X}$ into $\mathbf{Y}$.
- The rows of $\mathbf{P}$, $\{\mathbf{p}_1, \ldots, \mathbf{p}_m\}$ are a set of new basis vectors for expressing the columns of $\mathbf{X}$.

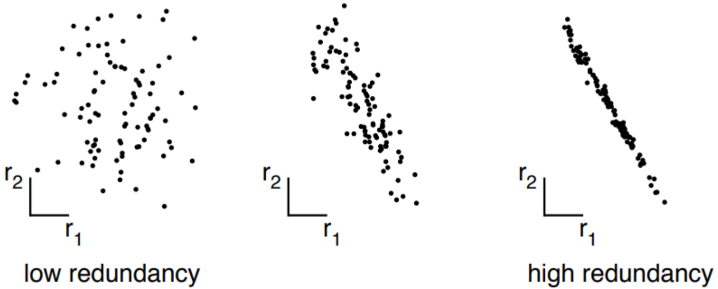
What is the best way to re-express $\mathbf{X}$?, what is a good choice for $\mathbf{P}$?

Noise



- Signal and noise variances are depicted as σ^2_{signal} and σ^2_{noise}.
- The largest direction of variance is not along the natural basis but along the best-fit line.
- The directions with largest variances contain the dynamics of interest.
- Intuition: Find the direction indicated by σ_{signal}.

Redundancy



- Figures depict possible plots between two arbitrary measurement types r_1 and r_2.
- Low redundancy $\rightarrow$ uncorrelated recordings
- High redundancy$\rightarrow$ correlated recordings, e.g. the sensors are too close or the measured variables are equivalent.
- If recordings are highly correlated it is not necessary to measure both of them.

PCA - Basic concepts

Let **a** $= [a_1, a_2, \ldots, a_n]$ and **b** $= [b_1, b_2, \ldots, b_n]$ be two sets of measurements. Are they related?
If the mean of a and b is zero, then:

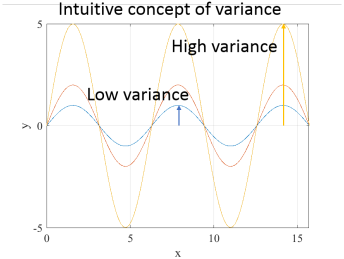
- Variance: How large the change is in each vector.

$$\sigma_a^2 = \frac{1}{n}\mathbf{a}\mathbf{a}^T = \frac{1}{n}\sum_i a_i^2$$

$$\sigma_b^2 = \frac{1}{n}\mathbf{b}\mathbf{b}^T = \frac{1}{n}\sum_i b_i^2$$

- Covariance: Statistical relationship between data in **a** and **b**.

$$\sigma_{ab}^2 = \frac{1}{n}\mathbf{a}\mathbf{b}^T = \frac{1}{n}\sum_i a_i b_i$$

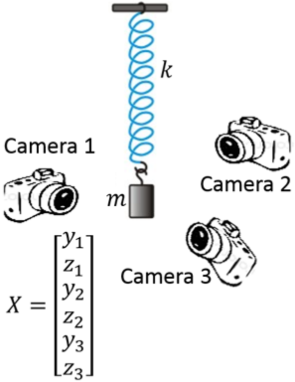


Variance and Covariance

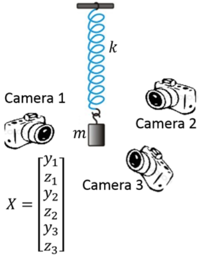
Let $\mathbf{X}$ be defined as $\mathbf{X} = [\mathbf{x}_1^T | \ldots | \mathbf{x}_m^T]$, where $\mathbf{x}_i \in \mathbb{R}^{n\times 1}$ is a column vector that corresponds to all measurements of a particular type. Then the covariance matrix is defined as:

$$\mathbf{C}_\mathbf{X} = \frac{1}{n}\mathbf{X}\mathbf{X}^T$$

The covariance values reflect the noise and redundancy in the measurements.



Variance and Covariance



Recall $\mathbf{C_X}$ is the covariance matrix of $\mathbf{X}$ defined as $\mathbf{C_X} = \frac{1}{n}\mathbf{X}\mathbf{X}^T$.

- Covariance matrix in the spring example is $\mathbf{C_X} \in \mathbb{R}^{6\times 6}$:

$$\mathbf{C_X} = \begin{bmatrix} \sigma^2_{y_1y_1} & \sigma^2_{y_1z_1} & \sigma^2_{y_1y_2} & \sigma^2_{y_1z_2} & \sigma^2_{y_1y_3} & \sigma^2_{y_1z_3} \\ \sigma^2_{z_1y_1} & \sigma^2_{z_1z_1} & \sigma^2_{z_1y_2} & \sigma^2_{z_1z_2} & \sigma^2_{z_1y_3} & \sigma^2_{z_1z_3} \\ \sigma^2_{y_2y_1} & \sigma^2_{y_2z_1} & \sigma^2_{y_2y_2} & \sigma^2_{y_2z_2} & \sigma^2_{y_2y_3} & \sigma^2_{y_2z_3} \\ \sigma^2_{z_2y_1} & \sigma^2_{z_2z_1} & \sigma^2_{z_2y_2} & \sigma^2_{z_2z_2} & \sigma^2_{z_2y_3} & \sigma^2_{z_2z_3} \\ \sigma^2_{y_3y_1} & \sigma^2_{y_3z_1} & \sigma^2_{y_3y_2} & \sigma^2_{y_3z_2} & \sigma^2_{y_3y_3} & \sigma^2_{y_3z_3} \\ \sigma^2_{z_3y_1} & \sigma^2_{z_3z_1} & \sigma^2_{z_3y_2} & \sigma^2_{z_3z_2} & \sigma^2_{z_3y_3} & \sigma^2_{z_3z_3} \end{bmatrix}$$

- Diagonal: Variance measures; Off-diagonal: covariance between all pairs.
- $\mathbf{C_X}$ is hermitian and symmetric, i.e. $\mathbf{C_X} = \mathbf{C_X^T}^{\,*} = \mathbf{C_X^T}$.

Covariance Matrix Interpretation

$$\mathbf{C_X} = \begin{bmatrix} \sigma^2_{y_1y_1} & \sigma^2_{y_1z_1} & \sigma^2_{y_1y_2} & \sigma^2_{y_1z_2} & \sigma^2_{y_1y_3} & \sigma^2_{y_1z_3} \\ \sigma^2_{z_1y_1} & \sigma^2_{z_1z_1} & \sigma^2_{z_1y_2} & \sigma^2_{z_1z_2} & \sigma^2_{z_1y_3} & \sigma^2_{z_1z_3} \\ \sigma^2_{y_2y_1} & \sigma^2_{y_2z_1} & \sigma^2_{y_2y_2} & \sigma^2_{y_2z_2} & \sigma^2_{y_2y_3} & \sigma^2_{y_2z_3} \\ \sigma^2_{z_2y_1} & \sigma^2_{z_2z_1} & \sigma^2_{z_2y_2} & \sigma^2_{z_2z_2} & \sigma^2_{z_2y_3} & \sigma^2_{z_2z_3} \\ \sigma^2_{y_3y_1} & \sigma^2_{y_3z_1} & \sigma^2_{y_3y_2} & \sigma^2_{y_3z_2} & \sigma^2_{y_3y_3} & \sigma^2_{y_3z_3} \\ \sigma^2_{z_3y_1} & \sigma^2_{z_3z_1} & \sigma^2_{z_3y_2} & \sigma^2_{z_3z_2} & \sigma^2_{z_3y_3} & \sigma^2_{z_3z_3} \end{bmatrix}$$

Off-diagonal terms

- If covariance is large then components are statistically dependent.
- If covariance is small then components are statistically independent.

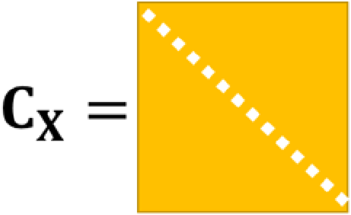
Diagonal terms:

- If variance is large it contains a lot of information about the system.
- If variance is small it does not provide significant information about the system.

PCA

Goal: Change basis such that the covariance matrix of the data is diagonal.

- If off-diagonal terms ≈ 0, the redundancies are eliminated.
- Diagonal terms represent the variance of each component.
- Components with large variance are the most representative.



Looks like the SVD!

PCA and Eigenvalue Decomposition

How to solve the problem?

- Data set: $\mathbf{X} \in \mathbb{R}^{m \times n}$, where m is the number of measurement types and n is the number of samples.
- PCA : Find an orthonormal matrix $\mathbf{P}$ in $\mathbf{Y} = \mathbf{P}\mathbf{X}$ such that $\mathbf{C_Y} = \frac{1}{n}\mathbf{Y}\mathbf{Y}^T$ is a diagonal matrix.
- The rows of $\mathbf{P}$ are the principal components of $\mathbf{X}$

PCA and Eigenvalue Decomposition

We begin rewriting $\mathbf{C_Y}$ in terms of the unknown variable.

$$
\begin{aligned}
\mathbf{C_Y} &= \frac{1}{n}\mathbf{Y}\mathbf{Y}^T \\
&= \frac{1}{n}(\mathbf{PX})(\mathbf{PX})^T \\
&= \frac{1}{n}\mathbf{P}\mathbf{X}\mathbf{X}^T\mathbf{P}^T \\
&= \mathbf{P}\left(\frac{1}{n}\mathbf{X}\mathbf{X}^T\right)\mathbf{P}^T \\
&= \mathbf{P}\mathbf{C_X}\mathbf{P}^T
\end{aligned}
$$

PCA and Eigenvalue Decomposition

$\mathbf{C_X}$ can be diagonalized by an orthogonal matrix of its eigenvectors since it is a symmetric matrix. Let $\mathbf{P} = \mathbf{Q}^T$, where $\mathbf{Q}$ is a matrix with the eigenvectors of $\frac{1}{n}\mathbf{X}\mathbf{X}^T$, then:

$$\begin{aligned}
\mathbf{C_Y} &= \mathbf{P}\mathbf{C_X}\mathbf{P}^T \\
&= \mathbf{P}\left(\mathbf{Q}\boldsymbol{\Omega}\mathbf{Q}^T\right)\mathbf{P}^T \\
&= \mathbf{P}\left(\mathbf{P}^T\boldsymbol{\Omega}\mathbf{P}\right)\mathbf{P}^T \\
&= \left(\mathbf{P}\mathbf{P}^{-1}\right)\boldsymbol{\Omega}\left(\mathbf{P}\mathbf{P}^{-1}\right) \\
&= \boldsymbol{\Omega}
\end{aligned}$$

The transformation $\mathbf{Y} = \mathbf{P}\mathbf{X}$ diagonalizes the system. Covariance of $\mathbf{Y}$ is a diagonal matrix with the eigenvalues of $\frac{1}{n}\mathbf{X}\mathbf{X}^T$.

PCA and SVD

The SVD of $\mathbf{X}$ is given by $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T$. Let $\mathbf{P} = \mathbf{U}^T$, then:

$$\mathbf{Y} = \mathbf{U}^T\mathbf{X},$$

The covariance matrix of $\mathbf{Y}$ is given by:

$$\begin{aligned}
\mathbf{C}_{\mathbf{Y}} &= \frac{1}{n}\mathbf{Y}\mathbf{Y}^T \\
&= \frac{1}{n}\mathbf{U}^T\mathbf{X}\mathbf{X}^T\mathbf{U} \\
&= \frac{1}{n}\mathbf{U}^T\mathbf{U}\Sigma\mathbf{V}^T\mathbf{V}\Sigma\mathbf{U}^T\mathbf{U} \\
&= \frac{1}{n}\Sigma^2
\end{aligned}$$

PCA

- The transformation $\mathbf{Y} = \mathbf{U}^T\mathbf{X}$ diagonalized the system. Covariance of $\mathbf{Y}$ is a diagonal matrix with the squared singular values of $\mathbf{X}$ multiplied by a factor of $\frac{1}{n}$.
- It can be concluded that $\boldsymbol{\Sigma}^2 = \boldsymbol{\Omega}$, and $\sigma_i^2 = \lambda_i$.
- The principal components of the data matrix are given by $\mathbf{U}^T$.

Application: Face Recognition

- PCA in face recognition $\triangleq$ Eigenfaces
- Intuition: Figure out the correlation between the rows/ colums of $\mathbf{A}$ from the SVD.

$$\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T \tag{1}$$

- How important each direction is: $\boldsymbol{\Sigma}$
- Principal Directions: $\mathbf{U}$
- How each individual component (row/column) projects onto the principal components: $\mathbf{V}$.

Data in Face Recognition

The data matrix is constructed by vectorizing the face images as shown below, i.e. $\mathbf{A} = [\mathbf{A}_1^T | \mathbf{A}_2^T | \dots | \mathbf{A}_N^T]^T$. The matrix will be $N \times M$, where N is the number of images in the data base and M is the number of pixels of each image.

Vectorized Image

Discretized Image

$$\mathbf{A} = \begin{bmatrix} \text{image 1} \\ \text{image 2} \\ \text{image 3} \\ \vdots \\ \text{image N} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \mathbf{A}_3 \\ \vdots \\ \mathbf{A}_N \end{bmatrix}$$

Data Matrix

Example - Celebrity Images

Example, take 5 images of each celebrity: George Clooney, Bruce Willis, Margaret Thatcher and Matt Damon. In the example, $M = 240 * 160 = 38400$ and $N = 20$.

$$\mathbf{A} = \begin{bmatrix} \text{-------- Image 1 --------} \\ \text{-------- Image 2 --------} \\ \text{-------- Image 3 --------} \\ \\ \\ \text{-------- Image 20 --------} \end{bmatrix}_{20 \times 38400}$$

Average Faces

How do the average of the faces of these celebrities look like?

$$\bar{\mathbf{a}}_i = \frac{1}{5}\sum_{j=1}^{5}\mathbf{A}_j \quad \text{where} \quad \mathbf{A}_j \in \mathbb{R}^{1\times M}$$

Average Faces

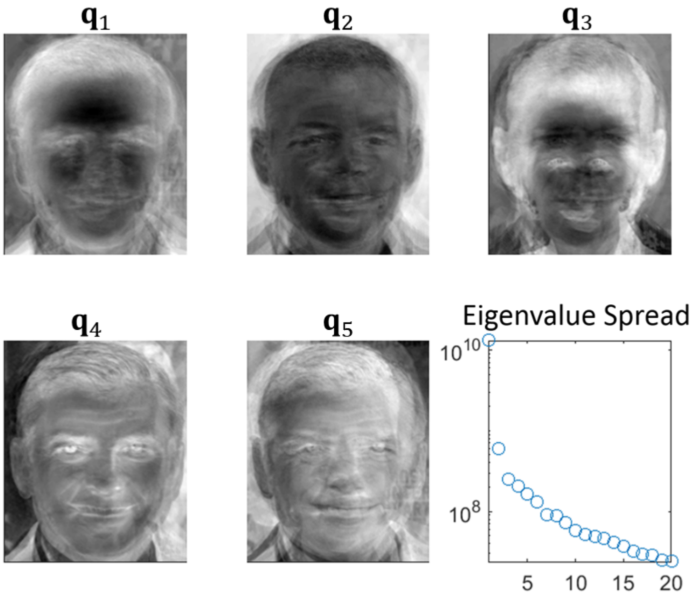
What defines George Clooney's face?

- Data matrix $\mathbf{A} \in \mathbb{R}^{N \times M}$ with the images of the example.
- Compute the correlation matrix of the features of the dataset, i.e. the pixels.
- The correlation matrix is $\mathbf{C} = \mathbf{A}^T\mathbf{A} \in \mathbb{R}^{M \times M}$, here $M = 38400$.
- High correlation values $\rightarrow$ everybody has eyes, a nose and a mouth.
- Correlations between images of the same person will be higher.

Average Face

Eigendecomposition

- Obtain the eigenvalue decomposition of $\mathbf{C} = \mathbf{A}^T\mathbf{A}$. That is $\mathbf{C} = \mathbf{Q}\mathbf{\Omega}\mathbf{Q}^{-1}$.
- First eigenvectors $\mathbf{q}_i \in \mathbb{R}^{M\times 1}$ are called the principal components (eigenfaces).
- One can reconstruct each face as a weighted sum of the eigenvectors.



Representing Faces onto Basis

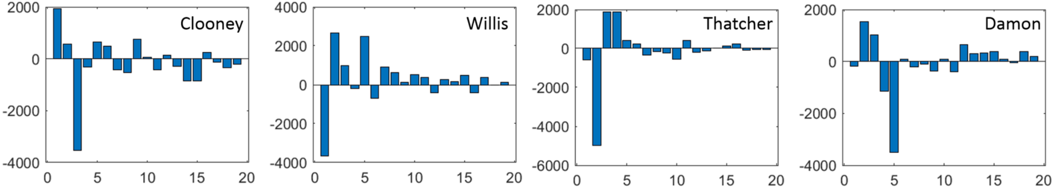
Each face $\mathbf{A}_i \in \mathbb{R}^{1\times M}$ in the data set $\mathbf{A} = [\mathbf{A}_1^T | \mathbf{A}_2^T | \ldots | \mathbf{A}_N^T]^T$, can be represented as a linear combination of the best K eigenvectors:

$$\mathbf{A}_i^T = \sum_{j=1}^{K} w_j \mathbf{q}_j, \text{ where } w_j = \mathbf{q}_j^T \mathbf{A}_i^T \tag{2}$$

$K = 1$ $K = 5$ $K = 10$ $K = 15$ $K = 20$

Projection of the Average faces into the K=20 largest Eigenvectors

- $\mathbf{Q}$ is $M \times M$, from now on let $\mathbf{V}$ be the matrix formed by the first K=20 eigenvectors, i.e. $\mathbf{V} \in \mathbb{R}^{M \times K}$.
- Project the average faces $\bar{\mathbf{a}}_i \in \mathbb{R}^{1 \times M}$ onto the reduced eigenvector space, i.e. $\mathbf{p}_{\bar{\mathbf{a}}_i} = \bar{\mathbf{a}}_i \mathbf{V} \quad \in \mathbb{R}^{1 \times K}$
- Projections for each face are characteristic of each average face and could be used for classification purposes.



Projection of new images

- Test set: New image of Margaret Thatcher, Maryl Streep as Margaret Thatcher in "The Iron Lady", Betty White.
- Project test images onto eigenvector space, $\mathbf{p} = \mathbf{x}\mathbf{V} \in \mathbb{R}^{1\times K}$, where $\mathbf{x} \in \mathbb{R}^{1\times M}$ is the new vectorized image and $\mathbf{V} \in \mathbb{R}^{M\times K}$ is the matrix with the first 20 eigenvectors of the database.
- Reconstruct images as $\hat{\mathbf{x}} = \mathbf{V}\mathbf{p}^T$.
- Error defined as the difference between the projection of the new image and the projection of the original Margaret Thatcher images $\mathbf{o}_j\mathbf{V}$ where $j = 1, \ldots, 5$, that is

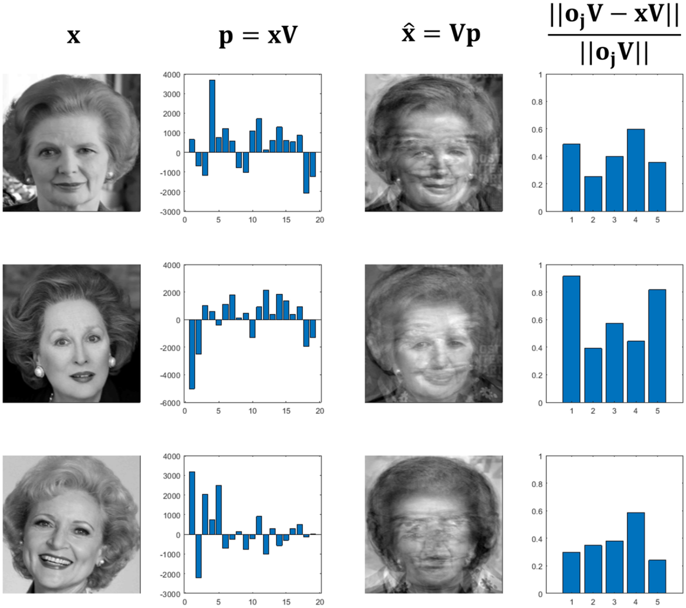
$$E_j = \frac{||\mathbf{o}_j\mathbf{V} - \mathbf{x}\mathbf{V}||}{||\mathbf{o}_j\mathbf{V}||},$$

where $\mathbf{o}_j$ are the original images of the database, in this case the 5 images of Margareth Thatcher.

Projection of new images

Image depicts, from left to right

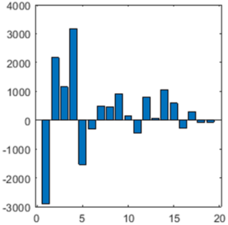
- Test images.
- Projection of the test images onto the eigenvector space $\mathbf{p} = \mathbf{x}\mathbf{V}$.
- Reconstructed images using the first 20 eigenvectors of the database $\hat{\mathbf{x}} = \mathbf{V}\mathbf{p}^T$.
- Error of the projection with respect to each original Margareth Thatcher Image $\mathbf{o}_j$ for $j = 1, ..., 5$.



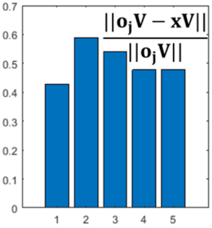
Projection of new images

$\mathbf{x}$

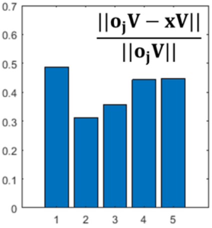
$\mathbf{p} = \mathbf{x}\mathbf{V}$



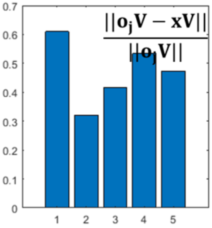
$\hat{\mathbf{x}} = \mathbf{V}\mathbf{p}$



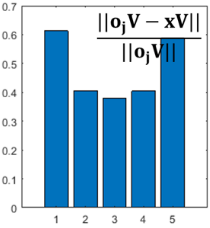
Clooney



Willis



Thatcher



Damon

Matrix Approximations and Completion

Given an $m \times n$ matrix $\mathbf{Z} = \{z_{ij}\}$, find a matrix $\hat{\mathbf{Z}}$ that approximates $\mathbf{Z}$.

- $\hat{\mathbf{Z}}$ may have simpler structure.
- Missing entries in $\mathbf{Z}$, a problem known as *matrix completion*.

Approach based on optimization:

$$\hat{\mathbf{Z}} = \text{arg} \min_{\mathbf{M} \in \mathbb{R}^{m \times n}} ||\mathbf{Z} - \mathbf{M}||_F^2 \text{ subject to } \Phi(\mathbf{M}) \leq c \qquad (3)$$

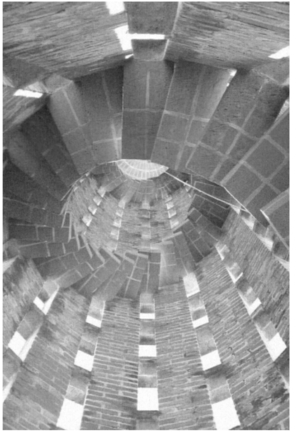
where $||\mathbf{A}||_F^2 = \sum\sum_{i,j} |a_{ij}|^2$ is the Frobenius Norm, and $\Phi(\cdot)$ is a constraint function that encourages $\hat{\mathbf{Z}}$ to be sparse in some sense.

Constraint $\Phi(\mathbf{Z})$	Resulting method
(a) $\|\|\hat{\mathbf{Z}}\|\|_{\ell_1} \leq c$	Sparse matrix approximation
(b) rank$(\hat{\mathbf{Z}}) \leq k$	Singular value decomposition
(c) $\|\|\hat{\mathbf{Z}}\|\|_* \leq c$	Convex matrix approximation

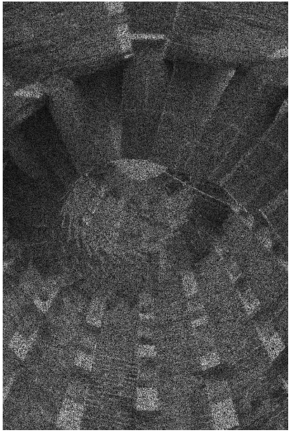
- (a) ℓ_1-norm of all entries of $\hat{\mathbf{Z}}$. Leads to a soft-thresholding $\hat{z}_{ij} = \text{sign}(z_{ij})(|z_{ij}| - \gamma)_+$, where $\gamma > 0$ is such that $\sum_{i=1}^{m}\sum_{j=1}^{n} |\hat{z_{ij}}| = c$.
- (b) Bounds the rank of $\hat{\mathbf{Z}}$, or the number of nonzero singular values in $\hat{\mathbf{Z}}$. Approximation is non-convex, but solution found by computing the SVD and truncating it to its top k components.
- (c) Relaxes the rank constraint to a *nuclear norm* ($||\mathbf{A}||_* = \sum_{i=1}^{min\{m,n\}} \sigma_i$). Solved by computing the SVD and soft-thresholding its singular values.

Motivation: Image Reconstruction from Incomplete Data

Reconstructed image



Incomplete image 50% of the pixels



Matrices with missing elements can be solved exactly using method (c), whereas methods based on (b) are more difficult to solve in general.

The Singular Value Decomposition

Given an $m \times n$ matrix $\mathbf{Z}$ with $m \geq n$, its *singular value decomposition* takes the form

$$\mathbf{Z} = \mathbf{U}\mathbf{D}\mathbf{V}^T \tag{4}$$

- $\mathbf{U}$ is an $m \times n$ orthogonal matrix ($\mathbf{U}^T\mathbf{U} = \mathbf{I}_n$) whose columns $\mathbf{u}_j \in \mathbb{R}^m$ are the *left singular vectors*.
- $\mathbf{V}$ is an $n \times n$ orthogonal matrix ($\mathbf{V}^T\mathbf{V} = \mathbf{I}_n$) whose columns $\mathbf{v}_j \in \mathbb{R}^n$ are the *right singular vectors*.
- The $n \times n$ matrix $\mathbf{D}$ is diagonal, with $d_1 \geq d_2 \geq \cdots \geq d_n \geq 0$ known as the *singular values*.

The Singular Value Decomposition

- If columns of $\mathbf{Z}$ are centered (zero mean), then the right singular vectors $\{\mathbf{v}_j\}_{j=1}^n$ define the *principal components* of $\mathbf{Z}$.
- The unit vector $\mathbf{v}_1$ yields the linear combination $\mathbf{s}_1 = \mathbf{Z}\mathbf{v}_1$ with highest sample variance among all possible choices of unit vectors.
- $\mathbf{s}_1$ is the *first principal component* of $\mathbf{Z}$, and $\mathbf{v}_1$ is the corresponding *direction* or *loading* vector.

The Singular Value Decomposition

Suppose $r \leq \text{rank}(\mathbf{Z}) = 800$, and let $\mathbf{D}_r$ be a diagonal matrix with all but the first r diagonal entries of $\mathbf{D}$ set to zero. The optimization problem

$$\hat{\mathbf{Z}}_r = \min_{\text{rank}(M)=r} ||\mathbf{Z} - \mathbf{M}||_F \qquad (5)$$

has a closed form solution $\hat{\mathbf{Z}}_r = \mathbf{U}\mathbf{D}_r\mathbf{V}^T \triangleq$ the rank-r SVD. $\hat{\mathbf{Z}}_r$ is sparse in the sense that all but r singular values are zero.

800 Singular Values 164 Singular Values 24 Singular Values 12 Singular Values

Matrix Completion

Problem Formulation: Recover an $m \times n$ matrix $\mathbf{Z}$ when we only get to observe $p \ll mn$ of its entries.

- Impossible without additional information!
- Assumption: Matrix is known to be low-rank or approximately low-rank.
- Matrix Completion: Fill the missing entries.
- Used in: machine learning, computer vision...

Optimization Problem

- Observe the entries of the $m \times n$ matrix $\mathbf{Z}$ indexed by the subset $\Omega \subset \{1, \cdots, m\} \times \{1, \cdots, n\}$.
- Seek the lowest rank approximating matrix $\hat{\mathbf{Z}}$ that interpolates the entries of $\mathbf{Z}$

$$\begin{aligned} &\text{minimize} && \operatorname{rank}(\mathbf{M}) \\ &\text{subject to} && m_{ij} = z_{ij},\ (i,j) \in \Omega, \end{aligned} \tag{6}$$

- Rank minimization problem is NP-hard.
- Forcing interpolation leads to overfitting.

Optimization Problem

- Better to allow **M** to make some errors on the observed data:

$$\begin{aligned} \text{minimize} \quad & \operatorname{rank}(\mathbf{M}) \\ \text{subject to} \quad & \sum_{(i,j)\in\Omega} (z_{ij} - m_{ij})^2 \leq \delta\,, \end{aligned} \tag{7}$$

or equivalently

$$\underset{\operatorname{rank}(\mathbf{M})\leq r}{\text{minimize}} \quad \sum_{(i,j)\in\Omega} (z_{ij} - m_{ij})^2\,, \tag{8}$$

- Both problems are non-convex, and exact solutions are generally not available.

Matrix Completion Using the Nuclear Norm

- Nuclear norm of $\mathbf{M}_{m\times n}$:

$$||\mathbf{M}||_* = \sum_{k=1}^{n} \sigma_k(\mathbf{M}) \tag{9}$$

- Convex relaxation of the rank minimization problem:

$$\begin{aligned} &\text{minimize} && ||\mathbf{M}||_* \\ &\text{subject to} && m_{ij} = z_{ij},\ (i,j) \in \Omega\ , \end{aligned} \tag{10}$$

- Whereas the rank counts the number of nonzero singular values, the nuclear norm sums their amplitude.
- Analogous to the ℓ_1 norm as a relaxation for the ℓ_0 norm as sparsity measure.

Notation

Given an observed subset Ω of matrix entries, define the projection operator as:

$$[P_\Omega(\mathbf{Z})]_{i,j} = \begin{cases} z_{ij} & if \quad (i,j) \in \Omega \\ 0 & \text{otherwise} \end{cases}$$

P_Ω replaces the missing entries in $\mathbf{Z}$ with zeros, and leaves the observed entries alone.
The optimization criterion is then :

$$\sum_{(i,j)\in\Omega} (z_{ij} - m_{ij})^2 = ||P_\Omega(\mathbf{Z}) - P_\Omega(\mathbf{M})||_F \tag{11}$$

where $||\cdot||_F$ is the Frobenius norm of a matrix defined as the element-wise sum of squares.

Singular Value Thresholding for Matrix Completion,[+]

- Solves the optimization problem:

$$\begin{array}{ll} \text{minimize} & ||\mathbf{M}||_* \\ \text{subject to} & P_\Omega(\mathbf{M}) = P_\Omega(\mathbf{Z}) \,, \end{array} \tag{12}$$

- The SVD of a matrix **M** of rank r is:

$$\mathbf{M} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T \,,\ \boldsymbol{\Sigma} = \mathrm{diag}(\{\sigma_i\}_{1\le i\le r}) \tag{13}$$

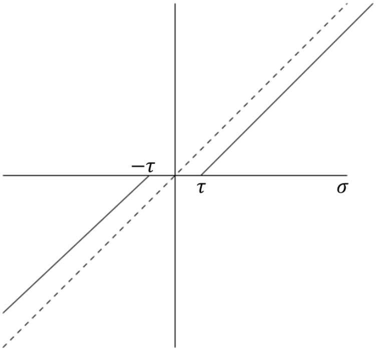
[+]Cai et al. (2010), SIAM Journal on Optimization, Vol. 20, No. 4

Singular Value Thresholding (SVT)

- For each $\tau \geq 0$, the soft-thresholding operator D_τ is defined as:

$$D_\tau(\mathbf{M}) = \mathbf{U} D_\tau(\boldsymbol{\Sigma}) \mathbf{V}^T \ , \ D_\tau(\boldsymbol{\Sigma}) = \mathrm{diag}(\mathsf{sgn}(\sigma_i)\{|\sigma_i| - \tau\}_+) \qquad (14)$$

where t, $t_+ = \max(0, t)$. Operator applies soft-thresholding to the singular values of **M**, effectively shrinking them towards zero.



SVT Algorithm - Shrinkage Iterations

Fix $\tau > 0$ and a sequence $\{\delta_k\}$ of positive step sizes. Starting with $\mathbf{Y}^0 = \mathbf{0}$, inductively define for $k = 1, 2, \ldots$,

$$\begin{cases} \mathbf{M}^k = D_\tau(\mathbf{Y}^{k-1}) \\ \mathbf{Y}^k = \mathbf{Y}^{k-1} + \delta_k P_\Omega(\mathbf{Z} - \mathbf{M}^k) \end{cases}$$

$$\begin{aligned} \mathbf{M}^1 &= D_\tau(\mathbf{Y}^0) = 0 \\ \mathbf{Y}^1 &= 0 + \delta_1 P_\Omega(\mathbf{Z} - 0) \\ &= \delta_1 P_\Omega(\mathbf{Z}) \end{aligned}$$

$$\begin{aligned} \mathbf{M}^2 &= D_\tau(\mathbf{Y}^1) = D_\tau(\delta_1 P_\Omega(\mathbf{Z})) \\ \mathbf{Y}^2 &= \delta_1 P_\Omega(\mathbf{Z}) + \delta_2 P_\Omega(\mathbf{Z} - D_\tau(\delta_1 P_\Omega(\mathbf{Z}))) \end{aligned}$$

until a stopping criterion is reached. At each step, we only need to compute an SVD and perform elementary matrix operations.

SVT Algorithm - Shrinkage Iterations

Image Inpainting - Convex Optimization Solver

With 70% of the Information.

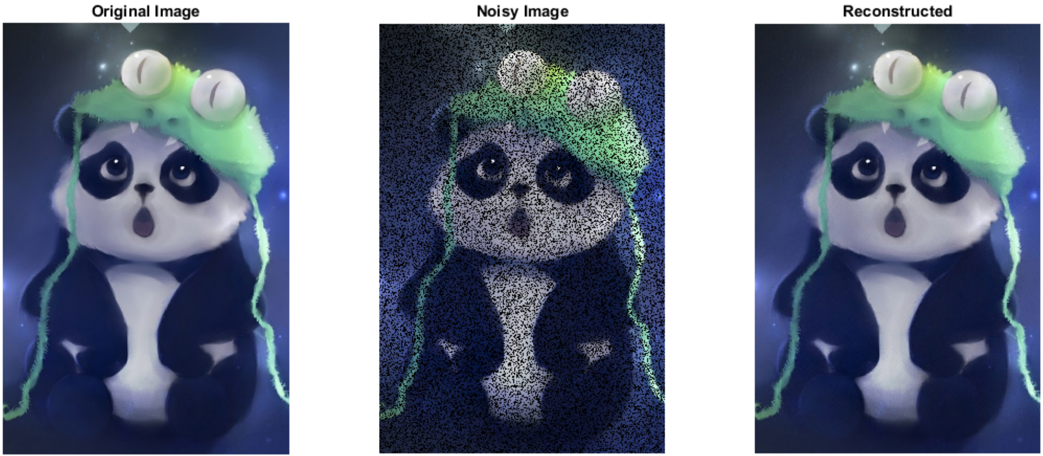


Image Inpainting - Convex Optimization Solver

With 50% of the Information. And multiple columns missing.

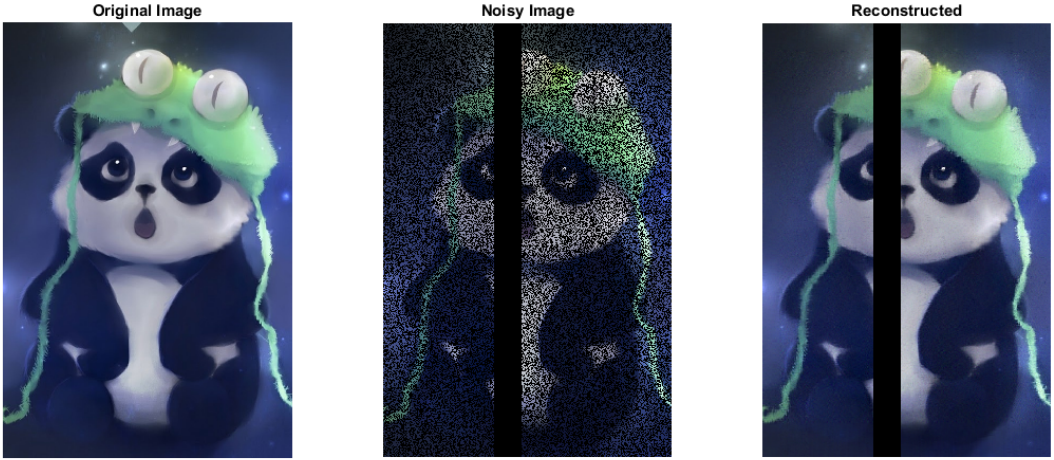


Image Inpainting - Convex Optimization Solver

With 50% of the Information. PSNR=35.9 dB.

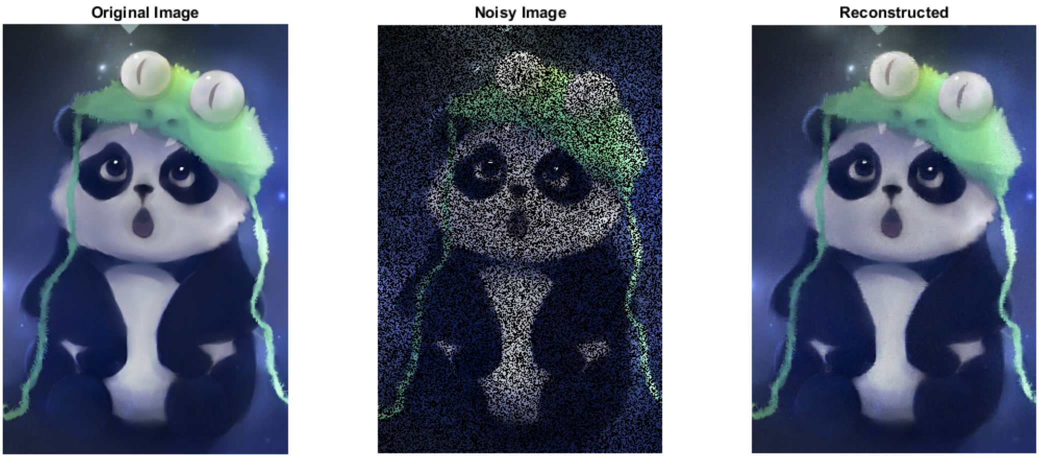
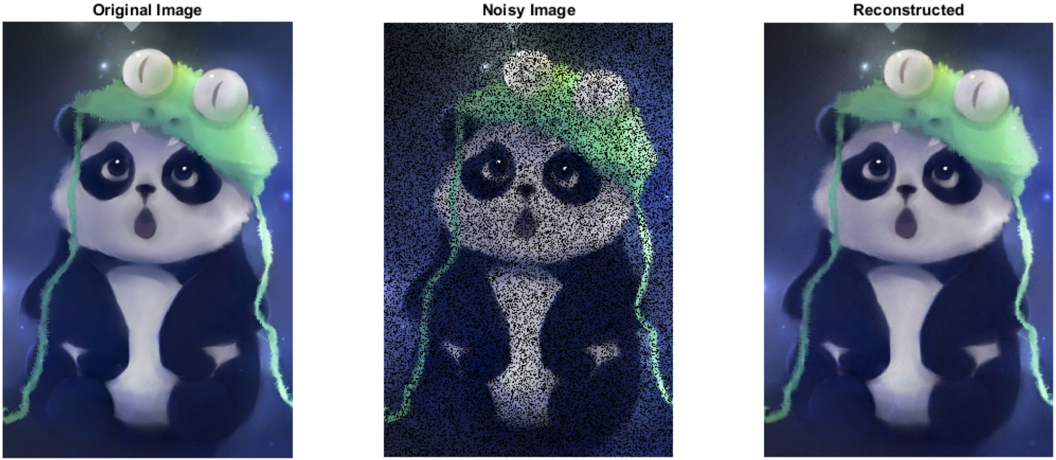


Image Inpainting - SVT Algorithm[+]

With 50% of the Information. PSNR=38.1 dB.



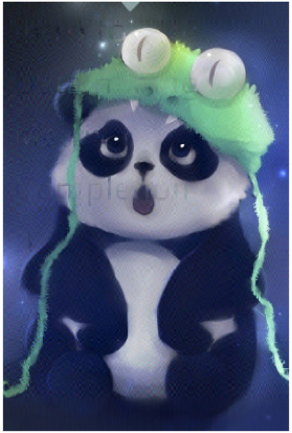
[+]Cai et al. (2010), SIAM Journal on Optimization, Vol. 20, No. 4

Text Removal - Convex Optimization Solver

Original Image

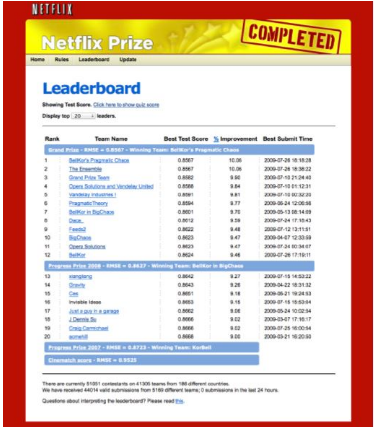
Noisy Image

Reconstructed



Netflix Movie Challenge - Revisited

- Dataset: $n = 17{,}770$ movies (columns) and $m = 480{,}189$ customers (rows).
- Customers rated movies on a scale from 1 to 5. Matrix is very sparse with "only" 100 million of the ratings present in the training set.
- Goal: Predict the ratings for unrated movies.



NETFLIX

Netflix Prize COMPLETED

Home Rules Leaderboard Update

Leaderboard

Showing Test Score. Click here to show quiz score

Display top 20 leaders.

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	0.8582	9.90	2009-07-10 21:24:40
4	Opera Solutions and Vandelay United	0.8588	9.84	2009-07-10 01:12:31
5	Vandelay Industries !	0.8591	9.81	2009-07-10 00:32:20
6	PragmaticTheory	0.8594	9.77	2009-06-24 12:06:56
7	BellKor in BigChaos	0.8601	9.70	2009-05-13 08:14:09
8	Dace_	0.8612	9.59	2009-07-24 17:18:43
9	Feeds2	0.8622	9.48	2009-07-12 13:11:51
10	BigChaos	0.8623	9.47	2009-04-07 12:33:59
11	Opera Solutions	0.8623	9.47	2009-07-24 00:34:07
12	BellKor	0.8624	9.46	2009-07-26 17:19:11
Progress Prize 2008 - RMSE = 0.8627 - Winning Team: BellKor in BigChaos				
13	xiangliang	0.8642	9.27	2009-07-15 14:53:22
14	Gravity	0.8643	9.26	2009-04-22 18:31:32
15	Ces	0.8651	9.18	2009-06-21 19:24:53
16	Invisible Ideas	0.8653	9.15	2009-07-15 15:53:04
17	Just a guy in a garage	0.8662	9.06	2009-05-24 10:02:54
18	J Dennis Su	0.8666	9.02	2009-03-07 17:16:17
19	Craig Carmichael	0.8666	9.02	2009-07-25 16:00:54
20	acmehill	0.8668	9.00	2009-03-21 16:20:50
Progress Prize 2007 - RMSE = 0.8723 - Winning Team: KorBell				
Cinematch score - RMSE = 0.9525				

There are currently 51051 contestants on 41305 teams from 186 different countries.
We have received 44014 valid submissions from 5169 different teams; 0 submissions in the last 24 hours.

Questions about interpreting the leaderboard? Please read this.

- (2006) "Cinematch" algorithm used by Netflix RMSE=0.9525 over a large test set.
- Competition started in 2006, winner should improve this RMSE by at least 10%.
- 2009 "Bellkor's Pragmatic Chaos," uses a combination of many statistical techniques to win.

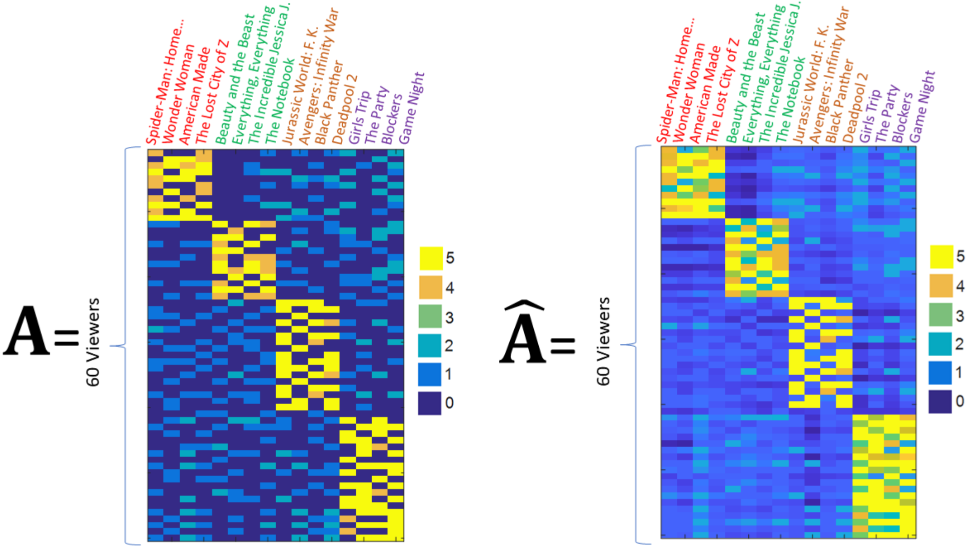
SVT Algorithm Solution

Use SVT Algorithm to estimate not rated movies (zero entries in $\mathbf{A}$), solving the optimization problem:

$$\begin{aligned} &\text{minimize} \quad ||\hat{\mathbf{A}}||_* \\ &\text{subject to} \quad P_\Omega(\hat{\mathbf{A}}) = P_\Omega(\mathbf{A}) \ , \end{aligned}$$

Note: The ratings matrix $\mathbf{A}$ is expected to be low-rank since user preferences can be described by a few categories (k), such as the movie genres.

SVT Algorithm Solution



SVT Algorithm Solution