DATA MINING
DIMENSIONALITY REDUCTION

SVD – PCA
Model-based recommendation systems

(Thanks to Jure Leskovec, Evimaria Terzi)
The curse of dimensionality

• Real data usually have thousands, or millions of dimensions
  • E.g., web documents, where the dimensionality is the vocabulary of words
  • Facebook graph, where the dimensionality is the number of users

• Huge number of dimensions causes problems
  • Data becomes very sparse, some algorithms become meaningless (e.g., density-based clustering)
  • The complexity of several algorithms depends on the dimensionality, and they become infeasible (e.g., nearest neighbor search).
Dimensionality Reduction

- Usually, the data can be described with fewer dimensions, without losing much of the information in the data.
- The data *reside* in a space of lower dimensionality
In this data matrix the dimension is essentially 3

There are three types of products and three types of users
Example

• Cloud of points 3D space:
  • Think of point positions as a matrix: 
    \[
    \begin{bmatrix}
    1 & 2 & 1 \\
    -2 & -3 & 1 \\
    3 & 5 & 0 \\
    \end{bmatrix}
    \]
  • 1 row per point: 
    \[
    \begin{bmatrix}
    A \\
    B \\
    C \\
    \end{bmatrix}
    \]

• 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!
Dimensionality Reduction

• Find the “true dimension” of the data
  • In reality, things are never as clear and simple as in this example, but we can still reduce the dimension.

• Essentially, we assume that some of the data is useful signal, and some data is noise, and that we can approximate the useful part with a lower dimensionality space.
  • Dimensionality reduction does not just reduce the amount of data, it often brings out the useful part of the data.
Dimensionality Reduction

Rather than representing every point with 2 coordinates we represent each point with 1 coordinate (corresponding to the position of the point on the red line).

By doing this we incur a bit of error as the points do not exactly lie on the line.

We assume that the line has the useful information, while the errors are noise.

\[
\begin{array}{c}
D = 2 \\
d = 1
\end{array}
\]

- Goal of dimensionality reduction is to discover the axis of data!
Why Reduce Dimensions?

- Discover hidden correlations/topics
  - E.g., in documents, words that occur commonly together
- Remove redundant and noisy features
  - E.g., in documents, not all words are useful
- Interpretation and visualization
- Easier storage and processing of the data
Data in the form of a matrix

• We are given \( n \) objects and \( d \) attributes describing the objects. Each object has \( d \) numeric values describing it.

• We will represent the data as a \( n \times d \) real matrix \( A \).
  • We can now use tools from linear algebra to process the data matrix.

• Our goal is to produce a new \( n \times k \) matrix \( B \) such that
  • It preserves as much of the information in the original matrix \( A \) as possible
  • It reveals something about the structure of the data in \( A \)
Example: Document matrices

- \( n \) documents
- \( d \) terms (e.g., theorem, proof, etc.)

\[ A_{ij} = \text{frequency of the } j\text{-th term in the } i\text{-th document} \]

Find subsets of terms that bring documents together
Example: Recommendation systems

\[ A_{ij} = \text{rating of } j\text{-th product by the } i\text{-th customer} \]

Find subsets of movies that capture the behavior or the customers
Linear algebra

• We assume that vectors are column vectors.
• We use $v^T$ for the transpose of vector $v$ (row vector)

• Dot product: $u^T v$ ($1 \times n, n \times 1 \rightarrow 1 \times 1$)
  • The dot product is the projection of vector $v$ on $u$ (and of $u$ on $v$)
  $$[1, 2, 3] \begin{bmatrix} 4 \\ 1 \\ 2 \end{bmatrix} = 12$$

  • $u^T v = ||v|| ||u|| \cos(u, v)$
    • If $||u|| = 1$ (unit vector) then $u^T v$ is the projection length of $v$ on $u$
    • If $||u|| = ||v|| = 1$ then $u^T v$ is the cosine similarity of $v$ and $u$

• $[-1, 2, 3] \begin{bmatrix} 4 \\ -1 \\ 2 \end{bmatrix} = 0$ : orthogonal vectors
  • Orthonormal vectors: two unit vectors that are orthogonal
Matrices

• An $n \times m$ matrix $A$ is a collection of $n$ row vectors and $m$ column vectors

$$A = \begin{bmatrix}
| & | & |
\end{bmatrix}
A = \begin{bmatrix}
- & \alpha^T_1 & - \\
- & \alpha^T_2 & - \\
- & \alpha^T_3 & -
\end{bmatrix}
$$

• Matrix-vector multiplication
  • Right multiplication $Au$:
    • projection of $u$ onto the row vectors of $A$,
    • projection of row vectors of $A$ onto $u$.
  • Left-multiplication $u^T A$:
    • projection of $u$ onto the column vectors of $A$,
    • projection of column vectors of $A$ onto $u$

• Example:

$$\begin{bmatrix}
1 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
1 \\
2 \\
3
\end{bmatrix} = \begin{bmatrix}
1
\end{bmatrix}$$
Change of basis

- By default, a vector is expressed in the axis-aligned basis.
  - For example, for vector \([-1,2]\) we have:
    \[
    \begin{bmatrix}
    -1 \\
    2
    \end{bmatrix} = -1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + 2 \begin{bmatrix} 0 \\ 1 \end{bmatrix}
    \]
  - With a projection we can change the basis over which the vector is expressed.

\[
\begin{bmatrix}
\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\
-\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2}
\end{bmatrix}\begin{bmatrix}
-1 \\
2
\end{bmatrix} = \begin{bmatrix}
\frac{3\sqrt{2}}{2} \\
\frac{\sqrt{2}}{2}
\end{bmatrix}
\]

The rows of the matrix are the new basis vectors.
Row and Column space

• **Row space** of A: The set of vectors that can be written as a linear combination of the rows of A
  • All vectors of the form \( \mathbf{v} = \mathbf{u}^T A \)
  • the coordinates of \( \mathbf{u} \) are the coefficients of the linear combination

\[
\begin{bmatrix}
    a_{11} & \cdots & a_{1i} & \cdots & a_{1m} \\
    a_{21} & \cdots & a_{2i} & \cdots & a_{2m} \\
    \vdots & & \vdots & & \vdots \\
    a_{n1} & \cdots & a_{ni} & \cdots & a_{nm}
\end{bmatrix}
\begin{bmatrix}
    u_1 \\
    \vdots \\
    u_i \\
    \vdots \\
    u_n
\end{bmatrix}
= \begin{bmatrix}
    u_1 a_{11} \\
    \vdots \\
    u_1 a_{ni} \\
    \vdots \\
    u_1 a_{nm}
\end{bmatrix} + \begin{bmatrix}
    u_2 a_{21} \\
    \vdots \\
    u_2 a_{ni} \\
    \vdots \\
    u_2 a_{nm}
\end{bmatrix} + \cdots + \begin{bmatrix}
    u_n a_{n1} \\
    \vdots \\
    u_n a_{ni} \\
    \vdots \\
    u_n a_{nm}
\end{bmatrix}
\]

• **Column space** of A: The set of vectors that can be written as a linear combination of the columns of A
  • All vectors of the form \( \mathbf{v} = A \mathbf{u} \)
  • the coordinates of \( \mathbf{u} \) are the coefficients of the linear combination
Rank

- **Rank** of A: the number of *linearly independent* row (or column) vectors
  - These vectors define a *basis* for the row (or column) space of A
    - All vectors in the row (column) space are linear combinations of the basis vectors

**Example**

- Matrix \( \mathbf{A} = \begin{bmatrix} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{bmatrix} \) 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.
Rank-1 matrices

- In a rank-1 matrix, all columns (or rows) are multiples of the same column (or row) vector

\[ A = \begin{bmatrix} 1 & 2 & -1 \\ 2 & 4 & -2 \\ 3 & 6 & -3 \end{bmatrix} \]

- All rows are multiples of \( r^T = [1, 2, -1] \)

- All columns are multiples of \( c = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \)
Rank-1 matrices

• A rank-1 matrix is the result of the outer product of two vectors

• Outer product: $uv^T$ ($n \times 1, 1 \times m \rightarrow n \times m$)

$$uv^T = \begin{bmatrix} u_1 \\ \vdots \\ u_i \\ \vdots \\ u_n \end{bmatrix} \begin{bmatrix} v_1 & \ldots & v_j & \ldots & v_m \end{bmatrix} = \begin{bmatrix} u_1v_1 & \cdots & u_1v_j & \cdots & u_1v_m \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ u_iv_1 & \cdots & u_iv_j & \cdots & u_iv_m \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ u_nv_1 & \cdots & u_nv_j & \cdots & u_nv_m \end{bmatrix}$$

• The resulting $n \times m$ matrix has rank 1: all rows (or columns) are linearly dependent

• $A = cr^T$

$$A = \begin{bmatrix} 1 & 2 & -1 \\ 2 & 4 & -2 \\ 3 & 6 & -3 \end{bmatrix}, c = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, r^T = [1, 2, -1]$$
Eigenvectors

- (Right) Eigenvector of matrix $A$: a vector $v$ such that $Av = \lambda v$
  - $\lambda$: eigenvalue of eigenvector $v$
- A symmetric matrix $A$ of rank $r$, has $r$ orthonormal eigenvectors $u_1, u_2, ..., u_r$ with eigenvalues $\lambda_1, \lambda_2, ..., \lambda_r$.
- Eigenvectors define an orthonormal basis for the column and row spaces of $A$
- We can write:
  \[
  A = U \Lambda U^T
  \]
  \[
  A = \lambda_1 u_1 u_1^T + \lambda_2 u_2 u_2^T + \cdots + \lambda_r u_r u_r^T
  \]
- Any vector $Ax$ (resp. $x^T A$) can be written as a linear combination of the vectors $u_1, u_2, ..., u_r$ (resp. $u_1^T, u_2^T, ..., u_r^T$)
SINGULAR VALUE DECOMPOSITION (SVD)
Singular Value Decomposition (SVD)

\[ A = U \Sigma V^T = [u_1, u_2, \ldots, u_r] \begin{bmatrix} \sigma_1 & 0 & \ldots & 0 \\ 0 & \sigma_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \sigma_r \end{bmatrix} [v_1^T, v_2^T, \ldots, v_r^T] \]

- \( r \): rank of matrix \( A \)
- \( \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r \): singular values of matrix \( A \)
- \( u_1, u_2, \ldots, u_r \): left singular vectors of \( A \)
- \( v_1, v_2, \ldots, v_r \): right singular vectors of \( A \)

\[ A = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_r u_r v_r^T \]

SVD can be computed for any matrix (not necessarily symmetric or square)
Singular Value Decomposition

• The left singular vectors $U$ are an orthonormal basis for the column space of $A$.
• The right singular vectors $V$ are an orthonormal basis for the row space of $A$.
• If $A$ has rank $r$, then $A$ can be written as the sum of $r$ rank-1 matrices

- There are $r$ “linear components/trends” (axes) in $A$.
  - Linear component: vector $v$ along which the row vectors of $A$ tend to be aligned
  - Strength of the i-th linear trend: $||Av_i|| = |\sigma_i|$

• Property of SVD:
  
  $$v_1 = \arg \max_{v:||v||=1} ||Av||$$
Symmetric matrices

- Special case: $A$ is symmetric positive definite matrix

$$A = \lambda_1 u_1 u_1^T + \lambda_2 u_2 u_2^T + \cdots + \lambda_r u_r u_r^T$$

- $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_r \geq 0$: Eigenvalues of $A$ are also the singular values

- $u_1, u_2, \ldots, u_r$: Eigenvectors of $A$ are also the left and right singular vectors
Singular Values and Eigenvalues

• Singular Value Decomposition
  \[ A = U\Sigma V^T \]

• The left singular vectors of \( A \) are also the eigenvectors of the (symmetric) matrix \( AA^T \)
  \[ AA^T = U\Sigma^2 U^T \]

• The right singular vectors of \( A \) are also the eigenvectors of the (symmetric) matrix \( A^T A \)
  \[ A^T A = V\Sigma^2 V^T \]

• The singular values of matrix \( A \) are also the square roots of eigenvalues of \( AA^T \) and \( A^T A \)
  \[ \lambda_i(A^T A) = \lambda_i(AA^T) = \sigma_i^2 \]
SVD properties

- Singular Value Decomposition has three useful properties that we will study now:
  - It provides the important (principal) directions (dimensions) in the data – Principal Component Analysis
  - It provides the best low rank approximation for our matrix
  - It minimizes the reconstruction error (squared distance between real data points and their estimates)
PRINCIPAL COMPONENT ANALYSIS (PCA)
Principal Component Analysis

• Goal: reduce the dimensionality while preserving the “information in the data”.

• In the new space we want to:
  • Maximize the amount of information
  • Minimize redundancy – remove the redundant dimensions
  • Minimize the noise in the data.
Variability

• Information in the data: \textit{variability} in the data
  • We measure variability using the \textit{covariance matrix}.
  • Sample variance for variable $X$:
    $$\sigma_X^2 = \frac{1}{N} \sum_i (x_i - \mu_X)(x_i - \mu_X) = \frac{1}{N} (x - \mu_X)^T (x - \mu_X)$$
  • Sample covariance of variables $X$ and $Y$
    $$\sigma_{XY}^2 = \frac{1}{N} \sum_i (x_i - \mu_X)(y_i - \mu_Y) = \frac{1}{N} (x - \mu_X)^T (y - \mu_Y)$$

• High variance $\sigma_X^2$ means high information in dimension (attribute) $X$
  • We want to maximize the signal-to-noise ratio $\frac{\sigma_{signal}^2}{\sigma_{noise}^2}$

• High co-variance $\sigma_{XY}^2$ means high correlation between attributes $X,Y$, and thus high redundancy.
  • Ideally, we want $\sigma_{XY}^2 = 0$ for all pairs $X,Y$
Example

• In the data below the data are essentially one-dimensional, but what is the axis we should use?
  • The direction in which the variance is maximized.

The variance along the direction orthogonal to the main direction is small and captures the noise in the data.
Example

- Which direction is best to project?
Example

- Which direction is best to project?
Example

- Which direction is best to project?
Which direction is best to project?
• Which direction is best to project?

• Note that in the case of blue and green directions we have points that are falling on each other.
• The red maximizes the variance, which gives more information
Covariance matrix

- We are given the data matrix $A$, with $n$ rows and $m$ columns, where the rows correspond to data samples over a set of features defined by the columns.
- Remove the mean of each column from the column vectors to get the centered matrix $\bar{A}$.
- The matrix $C = \bar{A}^T \bar{A}$ is proportional to the covariance matrix of the column vectors of $\bar{A}$.
- We want to change the basis of the data so that the matrix becomes diagonal.
  - All the values are in the diagonal and the off-diagonal entries are zero.
  - Covariances now become zero, so there is no redundancy.
PCA: Principal Component Analysis

• We will project the rows of matrix $\tilde{A}$ onto a new set of attributes (dimensions) such that:
  • Each attribute captures the most remaining variance in the data, while orthogonal to the existing attributes
    • The first attribute should capture the most variance in the data
    • The attributes have zero covariance to each other (they are orthogonal)

• For matrix $A$, the variance of the rows of $A$ when projected to vector $v$ is proportional to $\sigma^2 = \|\tilde{A}v\|^2$
  • The first right singular vector of $\tilde{A}$ maximizes $\sigma^2$!
PCA and SVD

• PCA is a special case of SVD on the centered matrix.
• After projecting the centered matrix $\bar{A}$ to the singular vectors in $V$ we have that the covariance matrix of the new dataset $\bar{A}V$ is:

$$(\bar{A}V)^T(\bar{A}V) = \Sigma$$

• We have achieved to make the matrix diagonal!

• Dimensionality reduction: Don’t keep all the singular vectors in $V$ just the $k$ first ones.
**PCA**

**Input:** 2-d dimensional points

**Output:**

1st (right) singular vector: direction of maximal variance,

2nd (right) singular vector: direction of maximal variance, after removing the projection of the data along the first singular vector.
Singular values

$\sigma_1$: measures data variance along the first singular vector.

$\sigma_2$: measures how much of the data variance is explained by the second singular vector.
Singular values tell us something about the variance

• The variance in the direction of the $k$-th principal component is given by the corresponding singular value $\sigma_k^2$

• Singular values can be used to estimate how many components to keep

• **Rule of thumb:** keep enough to explain 85% of the variation:

\[
\frac{\sum_{j=1}^{k} \sigma_j^2}{\sum_{j=1}^{n} \sigma_j^2} \approx 0.85
\]
Example

\[ A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} \]

\(a_{ij}\): usage of student \(i\) of drug \(j\)

- First right singular vector \(v_1\)
  - More or less same weight to all drugs
  - Discriminates heavy from light users
- Second right singular vector
  - Positive values for legal drugs, negative for illegal

\[ A = U\Sigma V^T \]
LOW-RANK APPROXIMATION
SVD and Rank-\(k\) approximations

\[
A = U \Sigma V^T
\]

We keep the \(k\) most important singular vectors.

The matrix \(U_k \Sigma_k V_k^T\) is a rank-\(k\) approximation of \(A\).

The idea is that this is the part that has the useful information and noise is removed.

This is also the best rank-\(k\) approximation (closest to the original \(A\)).
Rank-k approximations \((A_k)\)

\[
\begin{pmatrix}
A_k \\
n \times d
\end{pmatrix} = 
\begin{pmatrix}
U_k \\
n \times k
\end{pmatrix} \cdot 
\begin{pmatrix}
\Sigma_k \\
k \times k
\end{pmatrix} \cdot 
\begin{pmatrix}
V_k^T \\
k \times d
\end{pmatrix}
\]

\(U_k\): orthogonal matrix containing the top \(k\) left singular vectors of \(A\).

\(V_k\): orthogonal matrix containing the top \(k\) right singular vectors of \(A\).

\(\Sigma_k\): diagonal matrix containing the top \(k\) singular values of \(A\)

\(A_k\) is a rank-k approximation of \(A\)

\(A_k\) is the best approximation of \(A\)
**SVD as an optimization**

- The **rank-k approximation** matrix $A_k$ produced by the top-k singular vectors of $A$ **minimizes** the **Frobenious norm** of the difference with the matrix $A$

\[
A_k = \arg \min_{B: \text{rank}(B)=k} \|A - B\|_F^2
\]

\[
\|A - B\|_F^2 = \sum_{i,j} (A_{ij} - B_{ij})^2
\]

**Explanation:** The $(i, j)$ cell in $A_k$ is close on average with the $(i, j)$ cell of $A$
What does this mean?

• We can project the row (and column) vectors of the matrix $A$ into a $k$-dimensional space and preserve most of the information.

• (Ideally) The $A_k$ approximation of matrix $A$, contains all the useful information, and what is discarded is noise.

• (Ideally) The $k$ dimensions reveal latent features/aspects/topics of the term (document) space.
LATENT FACTOR MODELS AND RECOMMENDATION SYSTEMS
Latent factor model

- Rows (columns) are linear combinations of $k$ latent factors
  - E.g., in our extreme document example there are two factors
- Some noise is added to this rank-k matrix resulting in higher rank
- SVD retrieves the latent factors (hopefully).
An (extreme) example

- User-Movie matrix
  - Blue and Red rows (columns) are linearly dependent

- There are two prototype users (vectors of movies): blue and red
  - To describe the data is enough to describe the two prototypes, and the projection weights for each row

- $A$ is a rank-2 matrix

$$A = [w_1, w_2] \begin{bmatrix} d_1^T \\ d_2^T \end{bmatrix}$$
SVD – Example: Users-to-Movies

\[ A = U \Sigma V^T \] - example: Users to Movies
SVD – Example: Users-to-Movies

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

\[
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
3 & 3 & 3 & 0 & 0 \\
4 & 4 & 4 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 0 & 0 & 4 & 4 \\
0 & 0 & 0 & 5 & 5 \\
0 & 0 & 0 & 2 & 2 \\
\end{bmatrix}
= \begin{bmatrix}
0.14 & 0.00 \\
0.42 & 0.00 \\
0.56 & 0.00 \\
0.70 & 0.00 \\
0.00 & 0.60 \\
0.00 & 0.75 \\
0.00 & 0.30 \\
\end{bmatrix}
\begin{bmatrix}
12.4 & 0 \\
0 & 9.5 \\
\end{bmatrix}
\begin{bmatrix}
0.58 & 0.58 & 0.58 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.71 & 0.71 \\
\end{bmatrix}
\]
SVD – Example: Users-to-Movies

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

\[
\begin{pmatrix}
1 & 1 & 1 & 0 & 0 \\
3 & 3 & 3 & 0 & 0 \\
4 & 4 & 4 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 0 & 0 & 4 & 4 \\
0 & 0 & 0 & 5 & 5 \\
0 & 0 & 0 & 2 & 2 \\
\end{pmatrix}
\]

\[
\begin{pmatrix}
0.14 & 0.00 \\
0.42 & 0.00 \\
0.56 & 0.00 \\
0.70 & 0.00 \\
0.00 & 0.60 \\
0.00 & 0.75 \\
0.00 & 0.30 \\
\end{pmatrix}
\]

\[
\begin{pmatrix}
12.4 & 0 \\
0 & 9.5 \\
\end{pmatrix}
\]

\[
\begin{pmatrix}
0.58 & 0.58 & 0.58 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.71 & 0.71 \\
\end{pmatrix}
\]

\( U \) is “user-to-concept” similarity (or importance) matrix
SVD – Example: Users-to-Movies

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

\[
\begin{pmatrix}
1 & 1 & 1 & 0 & 0 \\
3 & 3 & 3 & 0 & 0 \\
4 & 4 & 4 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 0 & 0 & 4 & 4 \\
0 & 0 & 0 & 5 & 5 \\
0 & 0 & 0 & 2 & 2 \\
\end{pmatrix}
\begin{pmatrix}
0.14 & 0.00 \\
0.42 & 0.00 \\
0.56 & 0.00 \\
0.70 & 0.00 \\
0.00 & 0.60 \\
0.00 & 0.75 \\
0.00 & 0.30 \\
\end{pmatrix}
\begin{pmatrix}
12.4 & 0 \\
0 & 9.5 \\
\end{pmatrix}
\]

\[X \times \begin{pmatrix}
0.58 & 0.58 & 0.58 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.71 & 0.71 \\
\end{pmatrix}
\]

\(V\) is “movie to concept” similarity (or importance) matrix.
SVD – Example: Users-to-Movies

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

Matrix

$\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
3 & 3 & 3 & 0 & 0 \\
4 & 4 & 4 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 0 & 0 & 4 & 4 \\
0 & 0 & 0 & 5 & 5 \\
0 & 0 & 0 & 2 & 2 \\
\end{bmatrix}$

Sci-Fi-concept

Romance-concept

$\Sigma$ is the “concept strength” matrix

$\begin{bmatrix}
0.14 & 0.00 \\
0.42 & 0.00 \\
0.56 & 0.00 \\
0.70 & 0.00 \\
0.00 & 0.60 \\
0.00 & 0.75 \\
0.00 & 0.30 \\
\end{bmatrix}$

$x \begin{bmatrix}
12.4 & 0 \\
0 & 9.5 \\
\end{bmatrix} x$
SVD – Example: Users-to-Movies

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

$$
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
3 & 3 & 3 & 0 & 0 \\
4 & 4 & 4 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 0 & 0 & 4 & 4 \\
0 & 0 & 0 & 5 & 5 \\
0 & 0 & 0 & 2 & 2
\end{bmatrix}
= 
\begin{bmatrix}
0.14 & 0.00 \\
0.42 & 0.00 \\
0.56 & 0.00 \\
0.70 & 0.00 \\
0.00 & 0.60 \\
0.00 & 0.75 \\
0.00 & 0.30
\end{bmatrix}
$$

$\Sigma$ is the “spread (variance)” matrix

1st singular vector

SciFi

Romance

Movie 1

Movie 2

Origin

Vectors

$X$
SVD – Example: Users-to-Movies

• The axes on which we project the users (rows) are defined by the movies-to-concepts matrix $V$
• Similarly for movies the axes are defined by matrix $U$

\[
\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 \\
0 & 0 & 0 & 4 & 4 \\
0 & 0 & 0 & 5 & 5 \\
0 & 0 & 0 & 2 & 2
\end{bmatrix}
\begin{bmatrix}
0.58 & 0.00 \\
0.58 & 0.00 \\
0.58 & 0.00 \\
0.00 & 0.71 \\
0.00 & 0.71 \\
0.00 & 0.71
\end{bmatrix}
\begin{bmatrix}
12.4 & 0 \\
12.4 & 0 \\
12.4 & 0 \\
12.4 & 0 \\
12.4 & 0 \\
12.4 & 0
\end{bmatrix}
= 
\begin{bmatrix}
0.14 & 0.00 \\
0.42 & 0.00 \\
0.56 & 0.00 \\
0.70 & 0.00 \\
0.70 & 0.00 \\
0.70 & 0.00
\end{bmatrix}
\begin{bmatrix}
12.4 & 0 \\
0 & 9.5
\end{bmatrix}
An (more realistic) example

• User-Movie matrix

\[ A = \]

• There are two prototypes of users and movies, but they are noisy
  • Missing ratings
  • Ratings “out of character”

• This is the usual case for real data (lots of missing entries)
SVD – Example: Users-to-Movies

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

\[
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
3 & 3 & 3 & 0 & 0 \\
4 & 4 & 4 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 2 & 0 & 4 & 4 \\
0 & 0 & 0 & 5 & 5 \\
0 & 1 & 0 & 2 & 2 \\
\end{bmatrix}
\]

\( \Sigma \)  

\( V^T \)  

\( U \)  

“Concepts”  
AKA Latent dimensions  
AKA Latent factors  

SciFi  
Romance
SVD – Example: Users-to-Movies

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

Matrix

<table>
<thead>
<tr>
<th>SciFi</th>
<th>Romance</th>
<th>Alien</th>
<th>Serenity</th>
<th>Casablanca</th>
<th>Amelie</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

\[
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
3 & 3 & 3 & 0 & 0 \\
4 & 4 & 4 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 2 & 0 & 4 & 4 \\
0 & 0 & 0 & 5 & 5 \\
0 & 1 & 0 & 2 & 2 \\
\end{bmatrix}
= 
\begin{bmatrix}
0.13 & -0.02 & -0.01 \\
0.41 & -0.07 & -0.03 \\
0.55 & -0.09 & -0.04 \\
0.68 & -0.11 & -0.05 \\
0.15 & 0.59 & 0.65 \\
0.07 & 0.73 & -0.67 \\
0.07 & 0.29 & 0.32 \\
\end{bmatrix}
\begin{bmatrix}
12.4 & 0 & 0 \\
0 & 9.5 & 0 \\
0 & 0 & 1.3 \\
\end{bmatrix}
\begin{bmatrix}
0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\
-0.12 & 0.02 & -0.12 & 0.69 & 0.69 \\
0.40 & -0.80 & 0.40 & 0.09 & 0.09 \\
\end{bmatrix}
\]
### SVD – Example: Users-to-Movies

- **A = U Σ VT - example: Users to Movies**

<table>
<thead>
<tr>
<th>SciFi</th>
<th>SciFi-concept</th>
<th>Romance-concept</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SciFi</td>
<td>1 1 1 0 0</td>
<td></td>
</tr>
<tr>
<td>Rom</td>
<td>3 3 3 0 0</td>
<td></td>
</tr>
<tr>
<td>SciFi</td>
<td>4 4 4 0 0</td>
<td></td>
</tr>
<tr>
<td>Rom</td>
<td>5 5 5 0 0</td>
<td></td>
</tr>
<tr>
<td>SciFi</td>
<td>0 2 4 4 0</td>
<td></td>
</tr>
<tr>
<td>Rom</td>
<td>0 0 5 5 0</td>
<td></td>
</tr>
<tr>
<td>SciFi</td>
<td>0 1 0 2 2</td>
<td></td>
</tr>
</tbody>
</table>

The first two vectors are more or less unchanged.
SVD – Example: Users-to-Movies

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

\[
\begin{pmatrix}
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{pmatrix}
\begin{pmatrix}
0.13 & -0.02 & -0.01 \\
0.41 & -0.07 & -0.03 \\
0.55 & -0.09 & -0.04 \\
0.68 & -0.11 & -0.05 \\
0.15 & 0.59 & 0.65 \\
0.07 & 0.73 & -0.67 \\
0.07 & 0.29 & 0.32
\end{pmatrix}
\begin{pmatrix}
12.4 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0
\end{pmatrix}
\begin{pmatrix}
1.3 \\
0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\
-0.12 & 0.02 & -0.12 & 0.69 & 0.69 \\
0.40 & -0.80 & 0.40 & 0.09 & 0.09
\end{pmatrix}

The third vector has a very low singular value.
SVD - Interpretation

‘movies’, ‘users’ and ‘concepts’:

- $U$: user-to-concept similarity matrix
- $V$: movie-to-concept similarity matrix
- $\Sigma$: its diagonal elements: ‘strength’ of each concept
Rank-k approximation

• In this User-Movie matrix

\[
A = \begin{bmatrix}
\vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots \\
\end{bmatrix}
\]

• We have more than two singular vectors, but the strongest ones are still about the two types.
  • The third models the noise in the data
• By keeping the two strongest singular vectors we obtain most of the information in the data.
  • This is the rank-2 approximation of the matrix A
Example

More details

• **Q:** How exactly is dim. reduction done?
• **A:** Compute SVD

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

More details

• **Q:** How exactly is dim. reduction done?
• **A:** Set smallest singular values to zero
Example

More details

• **Q:** How exactly is dim. reduction done?
• **A:** Set smallest singular values to zero
Example

More details

• **Q:** How exactly is dim. reduction done?
• **A:** Keep the new representation of the users-movies

\[
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
3 & 3 & 3 & 0 & 0 \\
4 & 4 & 4 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 2 & 0 & 4 & 4 \\
0 & 0 & 0 & 5 & 5 \\
0 & 1 & 0 & 2 & 2
\end{bmatrix}
\approx
\begin{bmatrix}
0.13 & -0.02 \\
0.41 & -0.07 \\
0.55 & -0.09 \\
0.68 & -0.11 \\
0.15 & 0.59 \\
0.07 & 0.73 \\
0.07 & 0.29
\end{bmatrix}
\times
\begin{bmatrix}
12.4 & 0 \\
0 & 9.5
\end{bmatrix}
\times
\begin{bmatrix}
0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\
-0.12 & 0.02 & -0.12 & 0.69 & 0.69
\end{bmatrix}
\]
Example

More details

• **Q:** How exactly is dim. reduction done?
• **A:** Compute new matrix that approximates the original (w/o noise)

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

Frobenius norm:
\[
\|M\|_F = \sqrt{\sum_{ij} M_{ij}^2}
\]

**is “small”**
SVD for matrix reconstruction

• We will now see how we can use the fact that SVD gives the best rank-k approximation for a data matrix A.
• The idea is that we assume that the “true” matrix is rank-k, and rank is increased due to noise
• We use SVD to find the best rank-k approximation for A, and thus the best approximation of the “true” matrix
Model-based Recommendation Systems

- **Data**: An \( n \times m \) matrix \( D \) of users rating movies
  - Sparse and often noisy
- **Assumption**: There are \( k \) basic user profiles, and each user is a linear combination of these profiles
  - E.g., action, comedy, drama, romance
- **Implication**: There exists an ideal matrix \( M \) with the true preferences of all the users for all movies that is a rank-\( k \) matrix
- The matrix \( D \) that we observe is a noisy, and incomplete version of the matrix \( M \)
- Given matrix \( D \) we want to construct a matrix \( \tilde{D} \) that is as close as possible to the “true” matrix \( M \)
- **Model-based** collaborative filtering
Model-based Recommendation Systems

- Use as $\tilde{D}$ the rank-k approximation matrix $D_k$ of matrix $D$.
- The matrix $D_k$ is a full matrix (not sparse). Use the entries of $D_k$ to predict unseen ratings.

**Algorithm:**
- Compute the rank-k approximation $D_k$ of matrix $D$.
- For a user $u$ and movie $m$, predict the value $D_k[m, u]$.

- We know that matrix $D_k$ is the rank-k matrix that is closest to the observed data $D$.
- Under some assumptions for the missing and noisy entries, it can also be shown that it is the closest matrix to the ideal matrix $M$. 
## Example

### Missing ratings and noise

\[
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
0 & 3 & 3 & 0 & 0 \\
4 & 4 & 0 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 2 & 0 & 4 & 4 \\
0 & 0 & 0 & 5 & 5 \\
0 & 1 & 0 & 2 & 2
\end{bmatrix}
\begin{bmatrix}
0.14 & -0.06 & -0.04 \\
0.30 & -0.11 & -0.61 \\
0.43 & -0.16 & 0.76 \\
0.74 & -0.31 & -0.18 \\
0.15 & 0.53 & 0.02 \\
0.07 & 0.70 & -0.03 \\
0.07 & 0.27 & 0.01
\end{bmatrix}
\times
\begin{bmatrix}
12.4 & 0 & 0 \\
0 & 9.5 & 0 \\
0 & 0 & 1.3
\end{bmatrix}
\times
\begin{bmatrix}
0.51 & 0.66 & 0.44 & 0.23 & 0.23 \\
-0.24 & -0.13 & -0.21 & 0.66 & 0.66 \\
0.59 & 0.08 & -0.80 & 0.01 & 0.01
\end{bmatrix}
\]
Example

Missing ratings and noise

\[
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
0 & 3 & 3 & 0 & 0 \\
4 & 4 & 0 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 2 & 0 & 4 & 4 \\
0 & 0 & 0 & 5 & 5 \\
0 & 1 & 0 & 2 & 2 \\
\end{bmatrix}
\begin{bmatrix}
0.14 & -0.06 & -0.04 \\
0.30 & -0.11 & -0.61 \\
0.43 & -0.16 & 0.76 \\
0.74 & -0.31 & -0.18 \\
0.15 & 0.53 & 0.02 \\
0.07 & 0.70 & -0.03 \\
0.07 & 0.27 & 0.01 \\
\end{bmatrix}
\begin{bmatrix}
12.4 & 0 & 0 \\
0 & 9.5 & 0 \\
0 & 0 & 1.3 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.51 & 0.66 & 0.44 & 0.23 & 0.23 \\
-0.24 & -0.13 & -0.21 & 0.66 & 0.66 \\
0.59 & 0.08 & -0.80 & 0.01 & 0.01 \\
\end{bmatrix}
\]
Example

• Reconstruction of missing ratings

\[
\begin{bmatrix}
0.96 & 1.14 & 0.82 & -0.01 & -0.01 \\
1.94 & 2.32 & 1.66 & 0.07 & 0.07 \\
2.77 & 3.32 & 2.37 & 0.08 & 0.08 \\
4.84 & 5.74 & 4.14 & -0.08 & 0.08 \\
0.40 & 1.42 & 0.33 & 4.06 & 4.06 \\
-0.42 & 0.63 & -0.38 & 4.92 & 4.92 \\
0.20 & 0.71 & 0.16 & 2.03 & 2.03
\end{bmatrix}
\]

• This is the rank-2 approximation of the input matrix
Example

- Reconstruction of missing ratings

\[
\begin{bmatrix}
0.96 & 1.14 & 0.82 & -0.01 & -0.01 \\
1.94 & 2.32 & 1.66 & 0.07 & 0.07 \\
2.77 & 3.32 & 2.37 & 0.08 & 0.08 \\
4.84 & 5.74 & 4.14 & -0.08 & 0.08 \\
0.40 & 1.42 & 0.33 & 4.06 & 4.06 \\
-0.42 & 0.63 & -0.38 & 4.92 & 4.92 \\
0.20 & 0.71 & 0.16 & 2.03 & 2.03 \\
\end{bmatrix}
\]

This is our prediction for the cell \((u, m)\):

\[
r(u, m) = \sum_{k=1,2} U[u, k] \sigma_k V[k, m]
\]

This is essentially the dot product of the \(u\)-row of matrix \(U\), with the \(m\)-row of matrix \(V\), where the entries are weighted by the \(\Sigma\) values.

- This is the rank-2 approximation of the input matrix
Latent Factor Models

• SVD also considers entries that are missing!
• Use specialized methods to find $P$, $Q$
  
  $$\min_{P,Q} \sum_{(i,x) \in R} (r_{xi} - q_i^T \cdot p_x)^2$$

  $$\hat{r}_{xi} = q_i^T \cdot p_x$$

• Note:
  • We don’t require cols of $P$, $Q$ to be orthogonal/unit length
  • $P$, $Q$ map users/movies to a latent space
Latent factors

• We can define latent factor models that apply the same idea in different ways
  - Probabilistic/Generative models.
• These models are also described as Matrix Factorization techniques

• The latent factor methods work well in practice, and they are employed by most sophisticated recommendation systems

• Today there is also a lot of deep learning involved
Another Application

• Latent Semantic Indexing (LSI):
  • Apply PCA on the document-term matrix, and index the k-dimensional vectors
  • When a query comes, project it onto the k-dimensional space and compute cosine similarity in this space
  • Principal components capture main topics, and enrich the document representation
OTHER PROPERTIES OF SVD
Another property of PCA/SVD

• The chosen vectors are such that minimize the sum of square differences between the data vectors and the low-dimensional projections
SVD is “the Rolls-Royce and the Swiss Army Knife of Numerical Linear Algebra.”*
*Dianne O’Leary, MMDS ’06
Other dimensionality reduction techniques

• **Multidimensional Scaling**
  • You are given pairwise distances between objects
  • Find a low-dimensional representation of the objects (2D for visualization) such that you approximate the distances as well as possible

• **(Deep) Neural Network Embeddings**: 
  • Use a very simplistic, high-dimensional representation of the data (e.g., one-hot encoding)
  • Consider a prediction problem (e.g., predict the context words) and pass the data through a multi-layer neural network.
  • Use the input of the last layer as the representation
Computation of eigenvectors

- Consider a symmetric square matrix $M$
- Power-method:
  - Start with the vector $v$ of all 1’s
  - Compute $v = Mv$
  - Normalize by the length of $v$
  - Repeat until the vector does not change
- This will give us the first eigenvector.
- The first eigenvalue is $\lambda = v^T M v$
- For the second one, compute the first eigenvector of the matrix $M^* = M - \lambda v v^T$
Computing singular vectors

• Compute the eigenvectors and eigenvalues of the matrices $MM^T$ and $M^TM$