Foundations of Data Science
Lecture 4, Module 1
Fall 2022

Rumi Chunara, PhD

*<u>Fine Print</u>: these slides are, and always will be a work in progress. The material presented herein is original, inspired, or borrowed from others' work (mostly from professors **Rumi Chunara, Brian d'Alessandro, and Juliana Freire**). Where possible, attribution and acknowledgement will be made to content's original source. Do not distribute without the instructor's permission.*

Data Preprocessing

Major Tasks in Data Preprocessing

- Data sampling covered
- Data cleaning covered
- Data integration covered
- Data reduction

Major Tasks in Data Preprocessing

- Data sampling
- Data cleaning
- Data integration
- Data reduction

- Entropy
- Conditional entropy
- Comparing the entropy of a target (Y) conditioned to different features (X) to perform feature selection

- If metrics such as Pearson Correlation or Conditional Entropy are used for pairs (x_i, y), where x_i is a feature in the dataset and y is the target, they can help in feature ranking and selection

- Decision Tree Algorithms
  - We'll cover them later in the course

Data Reduction: Beyond Pairwise Relations

- **Goal**: obtain a reduced representation of the data set that is *much smaller* in volume but yet leads to very similar analytical results
  - Pearson correlation and conditional entropy can also be used to remove features and thus reduce datasets, but usually they are not as extreme
  - Pairwise metrics do not take more complex interactions across features and target into account

Although this is not a linear algebra course, this equation happens so often in data analysis that it is worth learning.

$$\mathbf{X} = \mathbf{U}\,\mathbf{\Sigma}\,\mathbf{V}^{\mathbf{T}}$$

$$MxN \qquad MxR \qquad RxR \qquad (NxR)^T$$

Goal: understand it intuitively and be able to use it as a tool for solving Data Science problems.

Let's go through this piece by piece:

$$\mathbf{U}$$

U holds the left singular vectors. Each row contains *r* elements that correspond to the latent factors of each row of X (note that they are ***not*** the original features anymore). U is orthonormal.

$$\boldsymbol{\Sigma}$$

Σ is a diagonal matrix that holds the singular values (in descending order). The singular values are essentially weights that determine how much each latent factor contributes to the matrix.

$$\mathbf{V}^{\mathbf{T}}$$

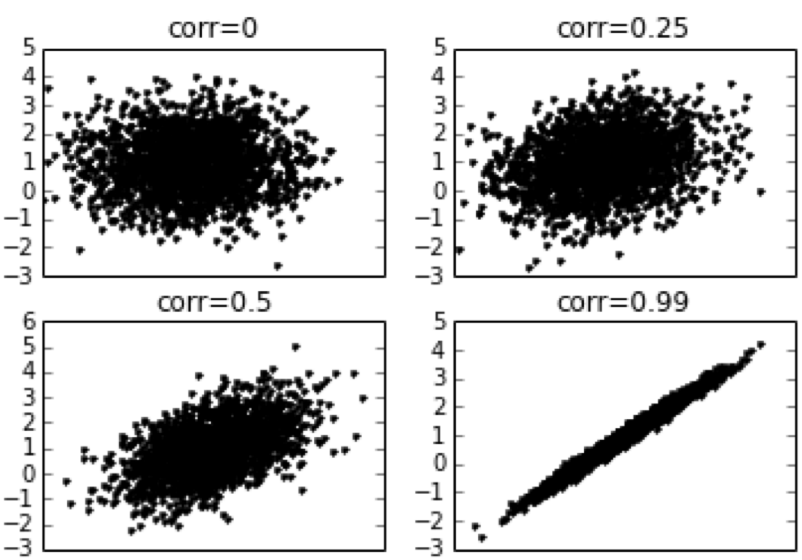
V holds the right singular vectors. Each row contains *r* elements that correspond to the latent factors of each column of X. V is orthonormal.

Quick example

- **X** is a document matrix with **m** documents and **n** words
  - every row of **X** is a document *i*
  - every column of **X** is a word *j*
  - If the word *j* is present in the document *i*, **X[*i, j*] = 1**; else **X[*i, j*] = 0**
- **U** is a matrix with **m** documents and **r** latent factors per document -- think of these latent factors as "concepts". You can think of **U** as a **"document-to-concept" matrix**
- **Σ** is a diagonal matrix with **r** entries, such that the *k*th entry of **Σ** is the weight of the *k*th latent factor/concept
- **Vt** is a matrix that associates each of the **n** words with the **r** concepts -- it can be seen as a **"word-to-concept" matrix**

How does this relate to the global structure of the dataset?

Consider these scatter plots. Each plot can be expressed as an Nx2 Matrix. The SVD is a tool that can help us understand how much information or structure is actually present in the matrix.

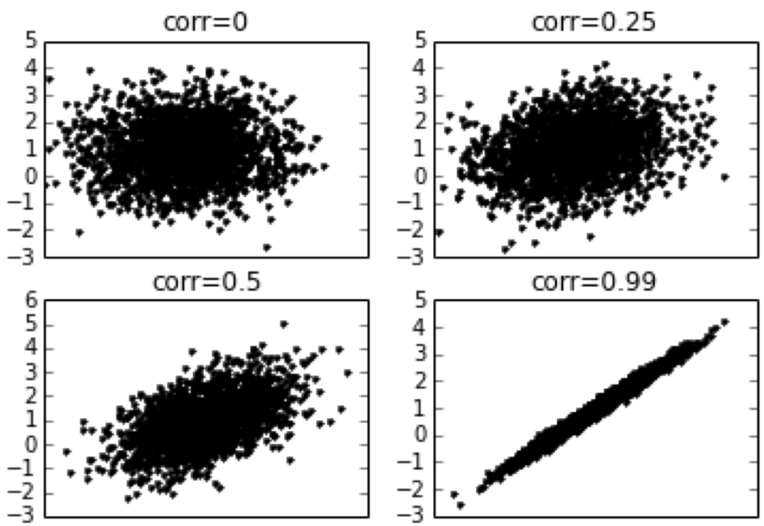


More independence of columns = more information.

Less independence of columns = less information.

Singular value decomposition

Using python we can easily compute the SVD:
U,Sig,Vt = np.linalg.svd(matrix)



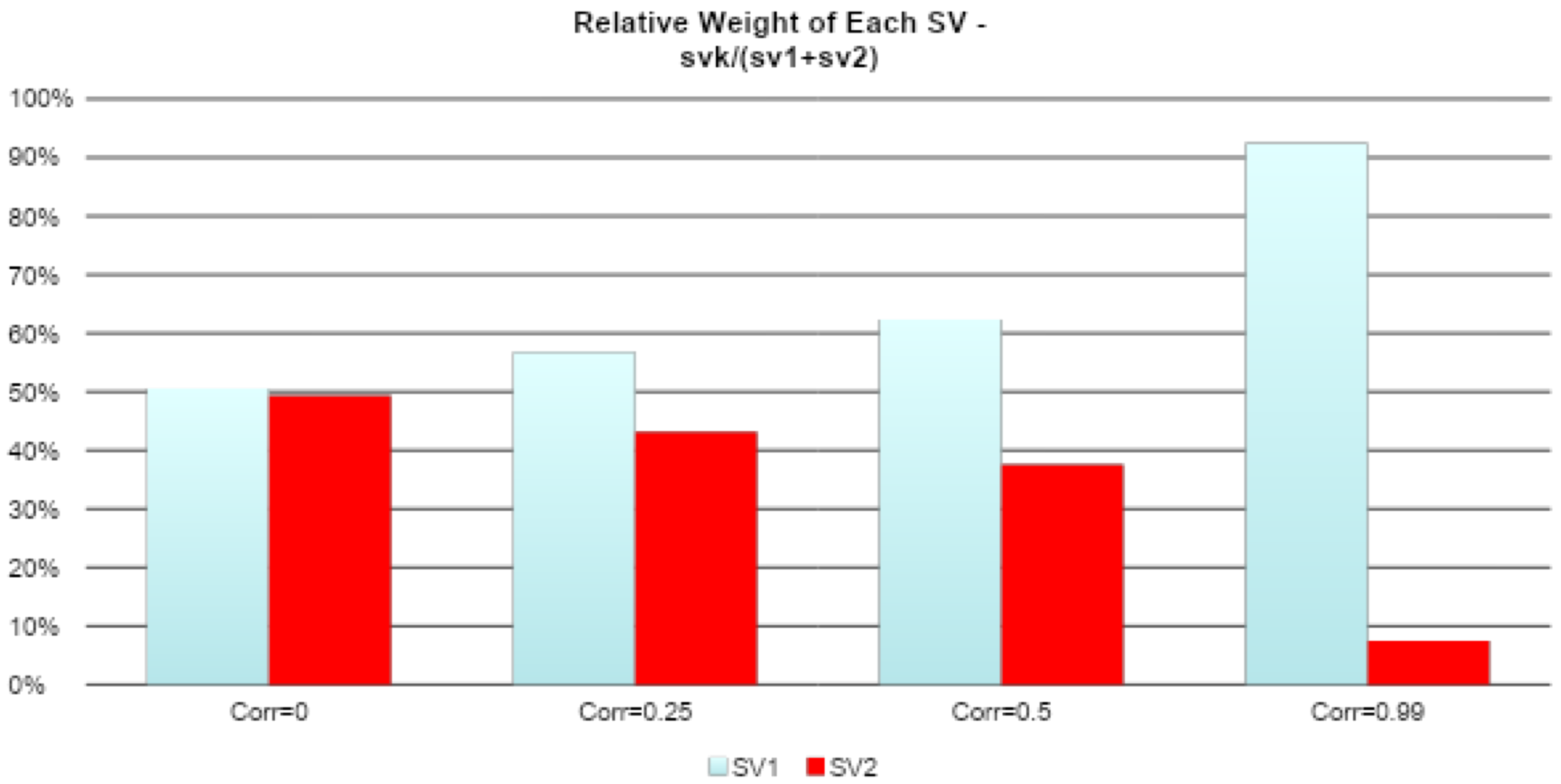
$$\Sigma = \begin{bmatrix} 45 & \\ & 44 \end{bmatrix} \qquad \Sigma = \begin{bmatrix} 50 & \\ & 38 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} 53 & \\ & 32 \end{bmatrix} \qquad \Sigma = \begin{bmatrix} 62 & \\ & 5 \end{bmatrix}$$

The magnitude of the singular values is generally determined by the magnitude of the values in X, so **normalize the matrix first!**

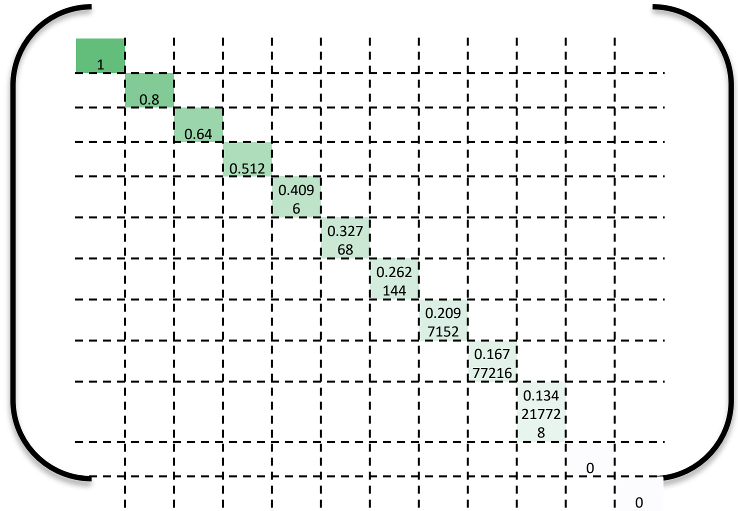
The relative difference between singular values is a function of the independence of the columns.

Less independence of the columns leads to singular values with more skew from each other.

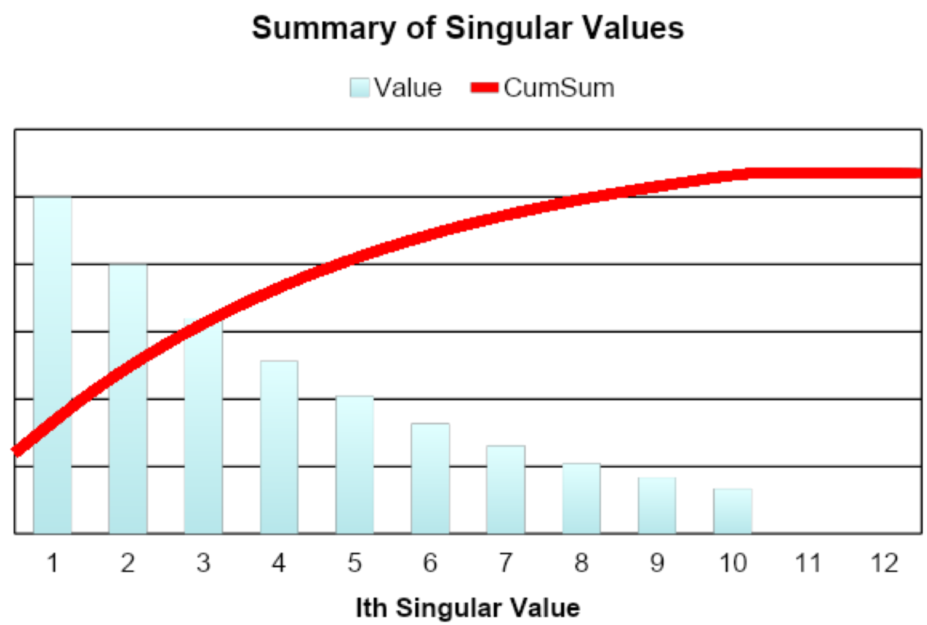


This idea generalizes to more dimensions, which is where the SVD is extremely useful.

$\Sigma =$



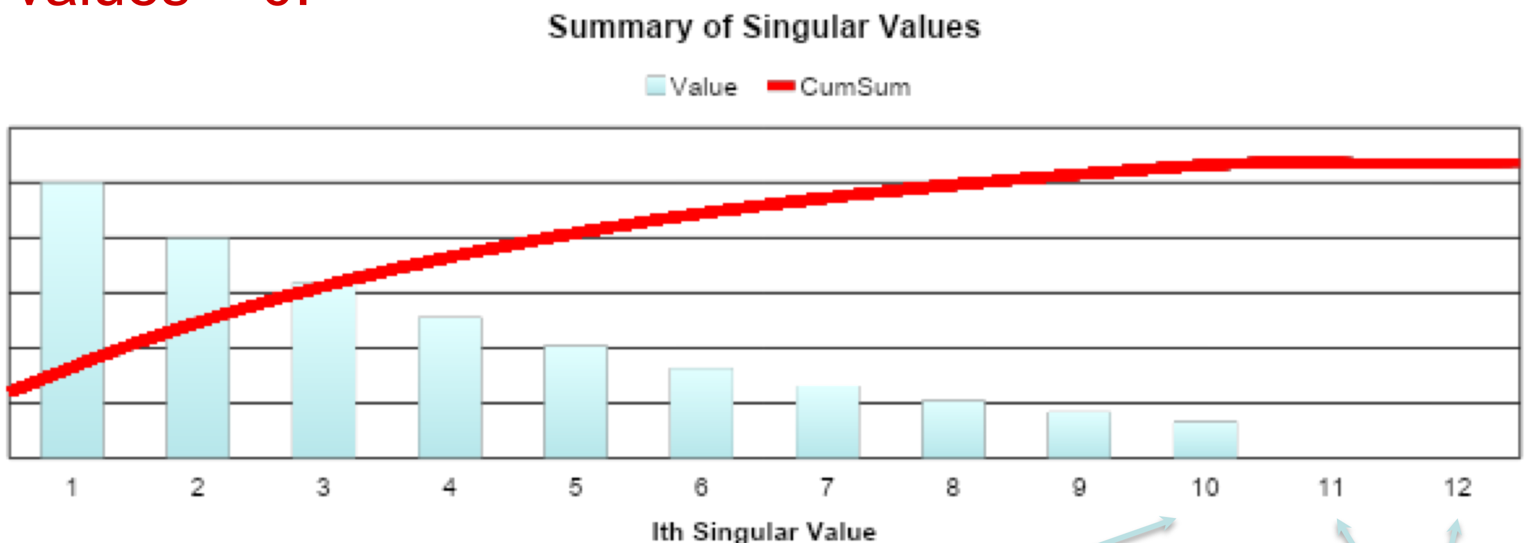
The skew of the singular values, and the shape of the cumulative sum curve can give us a sense of the degree of independence in a multidimensional matrix.



One of the most powerful applications of the SVD is creating a low-rank (reduced) approximation of a data matrix. Many of the following methods are based on using the SVD to get a low-rank approximation:

- Data compression
- Dimensionality reduction
- Recommender systems

- The rank of a matrix is the size of the largest number of independent columns of a matrix.
- The rank can be found by counting the number of singular values > 0.



These SVs are greater than 0, but they are small. The add very little information to this matrix, and could even represent noise. What would happen if we set them to zero?

These SVs are zero so the rank of this matrix is 10

The Low-Rank Approximation

We can build a matrix X_k that approximates our original matrix by doing the following:

1. Compute the SVD of X
2. Truncate U, Σ and V to take only the ***k*** highest columns & singular values
3. Multiply back together to get X_k

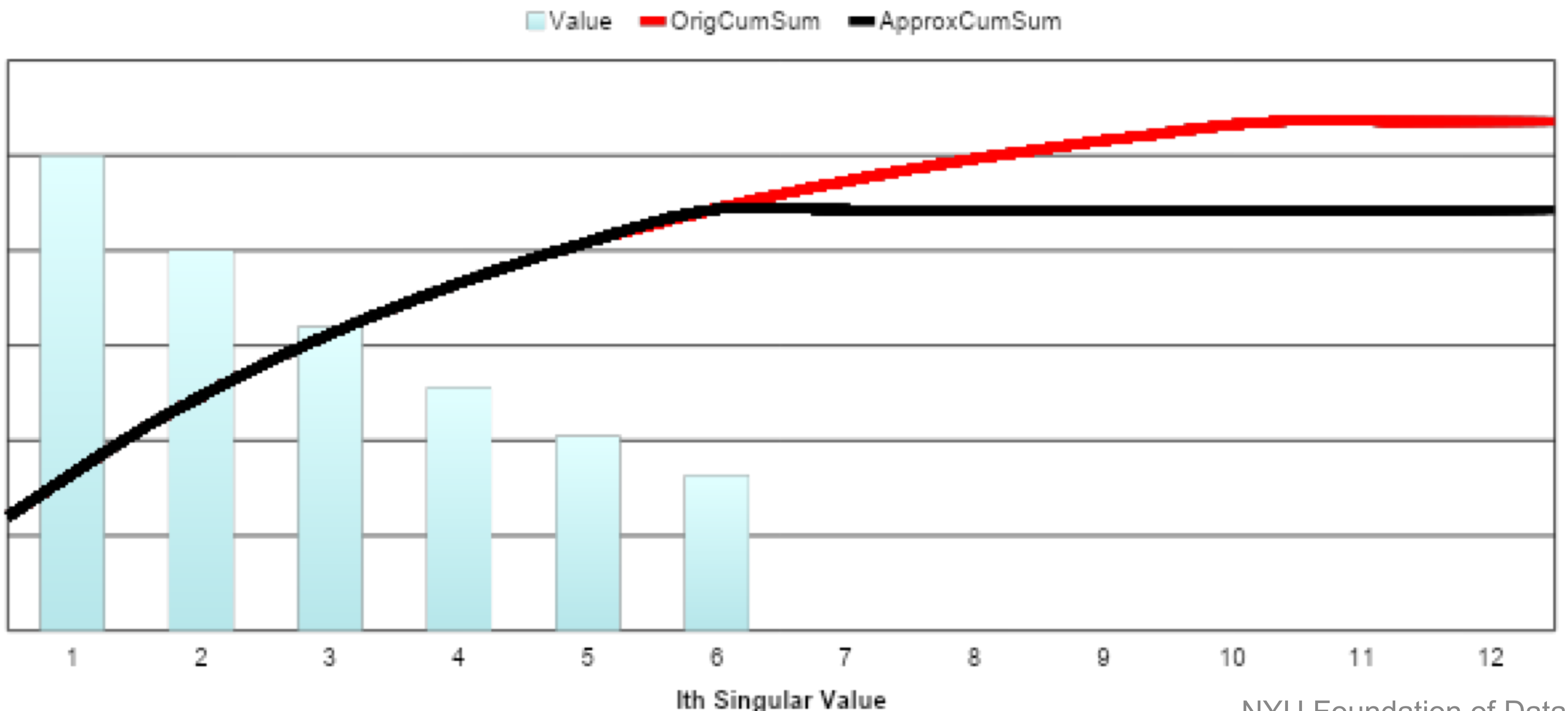
Low Rank Approximation – K<<R

$$\mathbf{X_k = U_k \, \Sigma_k \, V_k{}^T}$$

MxN *MxK* *KxK* *(NxK)*^{T}

Our original matrix had 12 columns with a rank of 10. In our approximation, we decided to use k = 6. The cumulative sum of singular values is related to the amount of information contained in the matrix. By using k = 6, we lost some information, but not half.

What do we gain with the low rank approximation?
What do we lose?



SVD Example: Taste in Movies

	Matrix	Alien	Serenity	Casablanca	Amelie
U1	**1**	**1**	**1**	0	0
U2	**3**	**3**	**3**	0	0
U3	**4**	**4**	**4**	0	0
U4	**5**	**5**	**5**	0	0
U5	0	**2**	0	**4**	**4**
U6	0	0	0	**5**	**5**
U7	0	**1**	0	**2**	**2**

Let's decompose it to discover broader taste trends

Note that the ratings are on the same scale!

Adapted from https://www.youtube.com/watch?v=P5mlg91as1c

SVD Example: Taste in Movies

***k* = 3**

	Matrix	Alien	Serenity	Casablanca	Amelie
U1	**1**	**1**	**1**	0	0
U2	**3**	**3**	**3**	0	0
U3	**4**	**4**	**4**	0	0
U4	**5**	**5**	**5**	0	0
U5	0	**2**	0	**4**	**4**
U6	0	0	0	**5**	**5**
U7	0	**1**	0	**2**	**2**

=

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**

U

X

Σ

12.4	0	0
0	**9.5**	0
0	0	**1.3**

X

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

Vt

** Note how you have two very clear groups: one that likes scifi and another that prefers romance

Adapted from https://www.youtube.com/watch?v=P5mlg91as1c

SVD Example: Taste in Movies

	Matrix	Alien	Serenity	Casablanca	Amelie
U1	**1**	**1**	**1**	0	0
U2	**3**	**3**	**3**	0	0
U3	**4**	**4**	**4**	0	0
U4	**5**	**5**	**5**	0	0
U5	0	**2**	0	**4**	**4**
U6	0	0	0	**5**	**5**
U7	0	**1**	0	**2**	**2**

=

Scifi taste **Romance taste**

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**

U

U: "user-to-taste" matrix

X

Σ: "strength" of each taste matrix

Σ

Scifi **Romance**

12.4	0	0
0	**9.5**	0
0	0	**1.3**

X

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

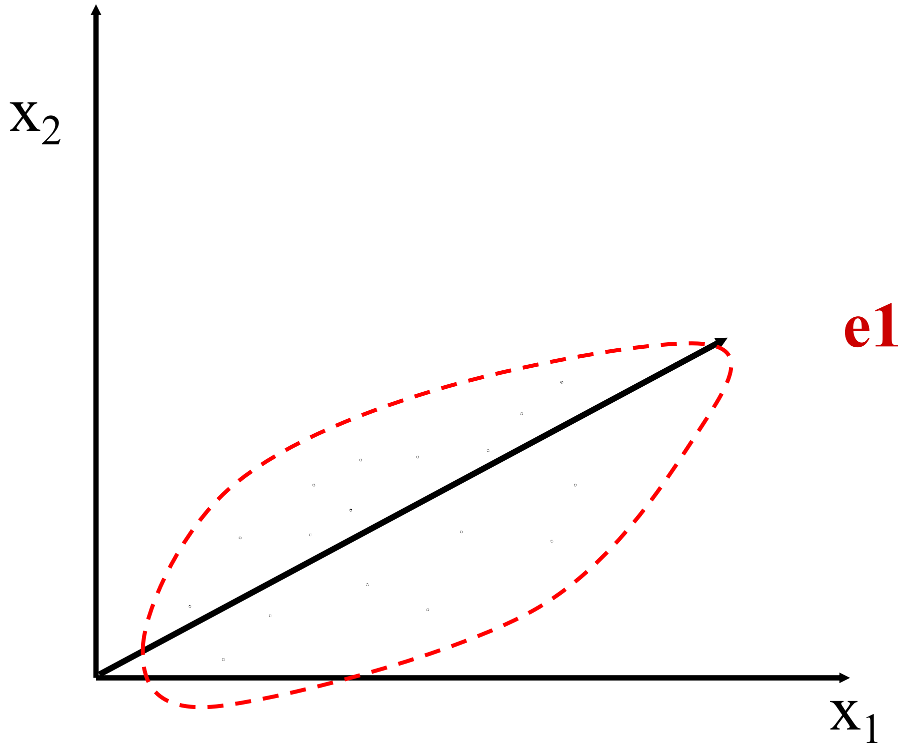
Vt

Vt: "movie-to-taste" matrix

Third "taste" (concept) has very low strength; does not explain users or movies very much.

Principal Component Analysis (PCA)

- Find a projection that captures the largest amount of variation in data (largest variance)
- The original data is projected onto a much smaller space, resulting in dimensionality reduction. We find the eigenvectors of the covariance matrix, and these eigenvectors define the new space



Principal Component Analysis (Steps)

- Given *N* data vectors from *n*-dimensions, find $k \leq n$ orthogonal vectors (*principal components*) that can be best used to represent data
    1. Normalize input data: each attribute falls within the same range
    2. Compute *k* orthonormal (unit) vectors, i.e., *principal components*
    3. Each input data (vector) is a linear combination of the *k* principal component vectors
    4. The principal components are sorted in order of decreasing "significance" or strength
    5. Since the components are sorted, the size of the data can be reduced by eliminating the *weak components*, i.e., those with low variance (i.e., using the strongest principal components, it is possible to reconstruct a good approximation of the original data)
- Works for numeric data only

Summary

- Methods that identify pairwise dependencies between features and the target (e.g., Pearson correlation, X^2 test, conditional entropy) can be used for feature ranking and selection
  - Note that **the meaning of the selected features does not change if you use these methods**
- Methods that identify the overall structure of information of the dataset can also be used to reduce it
  - Here, **by projecting the dataset onto a different system of coordinates, we "lose" the original features** (think of the "concepts" in the document matrix example)

Feature Selection Overview

Filter methods	**Wrapper methods**	**Embedded methods**
Generic set of methods which do not incorporate a **specific machine learning algorithm.**	Evaluates on a **specific machine learning algorithm** to find optimal features.	Embeds (fix) features during **model building process.** Feature selection is done by observing each iteration of model training phase.
Much **faster** compared to Wrapper methods in terms of time complexity	**High computation time** for a dataset with many features	Sits **between Filter methods and Wrapper methods** in terms of time complexity
Less prone to **over-fitting**	High chances of **over-fitting** because it involves training of machine learning models with different combination of features	Generally used to reduce **over-fitting** by **penalizing** the coefficients of a model being too large.
Examples – **Correlation, Chi-Square test, ANOVA, Information gain** etc.	Examples - **Forward Selection, Backward elimination, Stepwise selection** etc.	Examples - **LASSO, Elastic Net, Ridge Regression** etc.

https://www.analyticsvidhya.com/blog/2020/10/a-comprehensive-guide-to-feature-selection-using-wrapper-methods-in-python/