



Introduction

Linear Algebra

Department of Computer Engineering
Sharif University of Technology

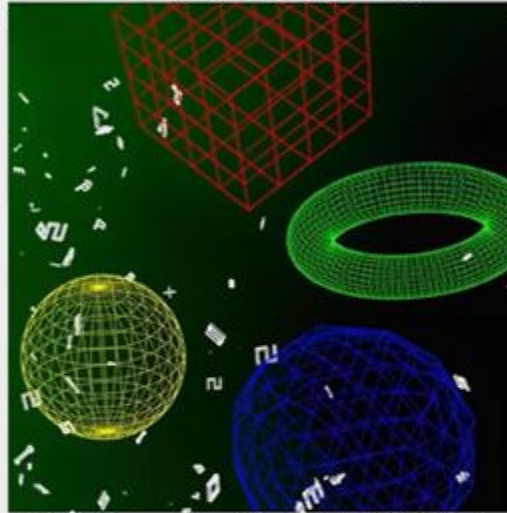
Hamid R. Rabiee rabiee@sharif.edu

Maryam Ramezani maryam.ramezani@sharif.edu



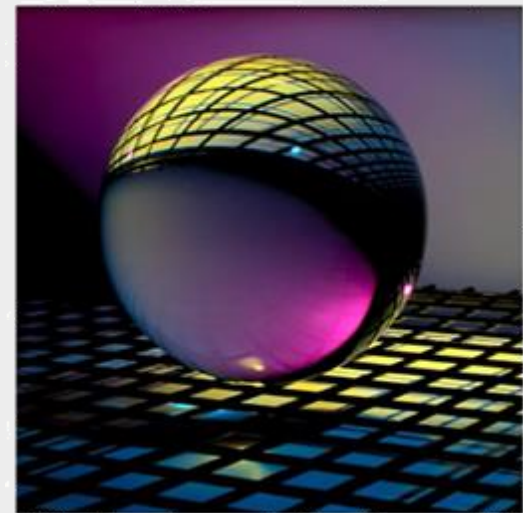
Data Representations

Use basic ideas of linear algebra to represent data in a way that computers can understand: vectors.



Vector Embeddings

Learn ways to choose these representations wisely via matrix factorizations.



Dimensionality Reduction

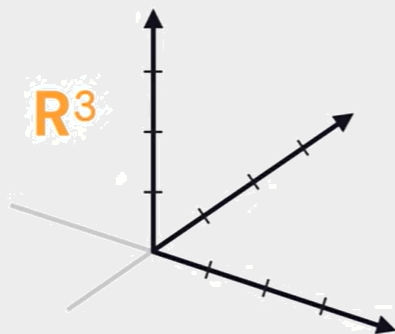
Deal with large-dimensional data using linear maps and their eigenvectors and eigenvalues.



- How can we represent data (images, text, user preferences, etc.) in a way that computers can understand?
 - Organize information into a vector!
- A **vector** is a 1-dimensional array of numbers.
 - It has both a magnitude (length) and a direction
- The totality of a vectors with n entries is an **n-dimensional vector space**.

$$V = \begin{matrix} \nearrow \\ \end{matrix} = \begin{bmatrix} -3 \\ 0.7 \\ 2 \end{bmatrix}$$

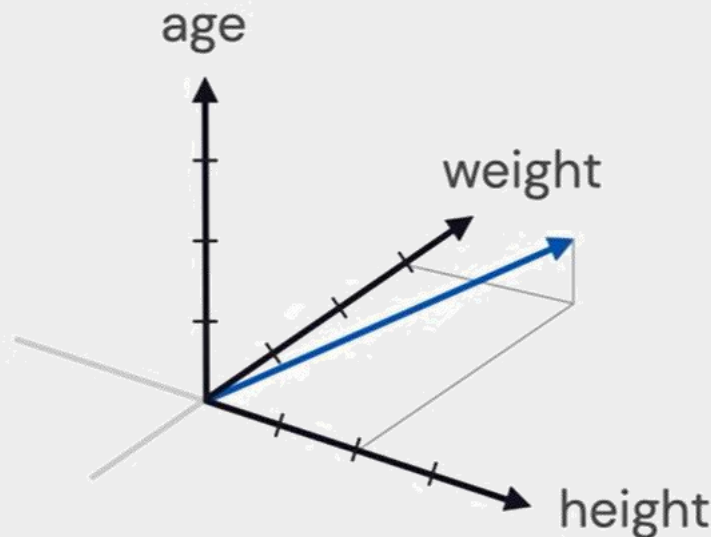
“3-dimensional space” consists
of all vectors with 3 entries:





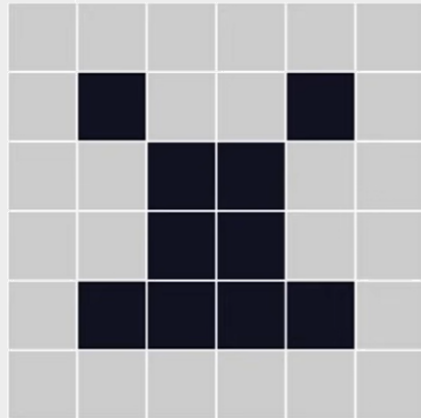
- ❑ A **feature vector** is a vector whose entries represent the “features” of an object.
- ❑ The vector space containing them is called **feature space**.

$$P = \begin{bmatrix} 64 \\ 131 \\ 24 \end{bmatrix} \begin{array}{l} \text{height} \\ \text{weight} \\ \text{age} \end{array}$$



Data Representation Applications

- In black and white images, black and white pixels correspond to 0s and 1s.



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



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

- In grayscale pixels are numbers between 0 and 255.





- Given a collection of documents (e.g. Wikipedia articles), assign to every word a vector whose i^{th} entry is the number of times the word appears in the i^{th} document.
- These vectors can be assemble into a large matrix, useful for latent semantic analytics.

$$\text{dog} = \begin{bmatrix} 0 \\ 7 \\ 0 \\ 0 \\ 51 \\ \vdots \\ 0 \end{bmatrix} \begin{matrix} \text{Wiki \#1} \\ \text{Wiki \#2} \\ \text{Wiki \#3} \\ \text{Wiki \#4} \\ \text{Wiki \#5} \\ \\ \text{Wiki \#54,000,000} \end{matrix}$$



- ❑ In the sub-field of machine learning for working with text data called natural language processing (**NLP**), it is common to represent documents as large matrices of word occurrences.

	Quick	Brown	Fox	Jumps	Over	Lazy	Dog
The quick brown fox jumps over the lazy dog	1	1	1	1	1	1	1
If the fox is quick he can jump over the dog.	1	0	1	0	1	0	1
Foxes are quick. Dogs are lazy.	0	1	1	0	0	1	1
Can a fox jump over a dog?	0	0	1	1	1	0	1

- ❑ Matrix factorization methods, such as the singular-value decomposition can be applied to this sparse matrix. Documents processed in this way are much easier to compare, query, and use as the basis for a supervised machine learning model.



- ❑ Given users and items (e.g. movies), vectors can indicate if a user has interacted with the item (yes=1, no=0).
- ❑ User's rating a number between 0 and 5.

$$\text{user1} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 1 \\ 0 \end{bmatrix} \quad \begin{array}{l} \text{No} \\ \text{Yes} \\ \text{No} \\ \text{No} \\ \\ \text{Yes} \\ \text{No} \end{array}$$

$$\text{user2} = \begin{bmatrix} 0 \\ 5 \\ 0 \\ 3 \\ \vdots \\ 0 \\ 2 \end{bmatrix} \quad \begin{array}{l} ? \\ \text{Love} \\ ? \\ \text{Like} \\ \\ ? \\ \text{Dislike} \end{array}$$

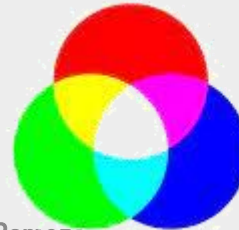
- ❑ Sometimes you work with categorical data in machine learning.
- ❑ It is common to encode categorical variables to make them easier to work with and learn by some techniques. A popular encoding for categorical variables is the one hot encoding.
- ❑ A one hot encoding is:

Example

$$\text{red} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$\text{green} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\text{blue} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$





- ❑ One-Hot Encodings (standard basis vector)
 - Assign to each word a vector with one 1 and 0s elsewhere.
 - Suppose our language only has four words:

$$\text{apple} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\text{cat} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$\text{house} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\text{tiger} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$



Drawbacks

- ❖ Very sparse vectors.
- ❖ Are never similar!





❑ Dot Product

- The product of numbers is another number.
- The dot product of vectors is not another vector! It is a number!!

$$\begin{array}{c} 2 \times 5 = 10 \\ \uparrow \quad \uparrow \quad \uparrow \\ \text{Numbers} \end{array}$$

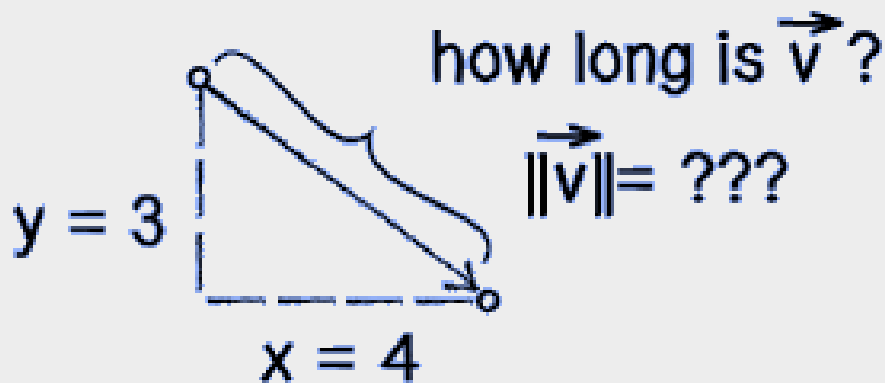
vs

$$\begin{array}{c} \begin{bmatrix} 2 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 5 \end{bmatrix} = 7 \\ \uparrow \quad \uparrow \quad \uparrow \\ \text{Vectors} \quad \text{A number} \end{array}$$

$$\begin{bmatrix} 1 \\ 0 \\ 3 \end{bmatrix} \cdot \begin{bmatrix} 7 \\ 2 \\ -1 \end{bmatrix} = (1)(7) + (0)(2) + (3)(-1) = 4$$



- Dot product between a vector and itself: magnitude-squared, the **length** squared, or the squared-norm, of the vector.



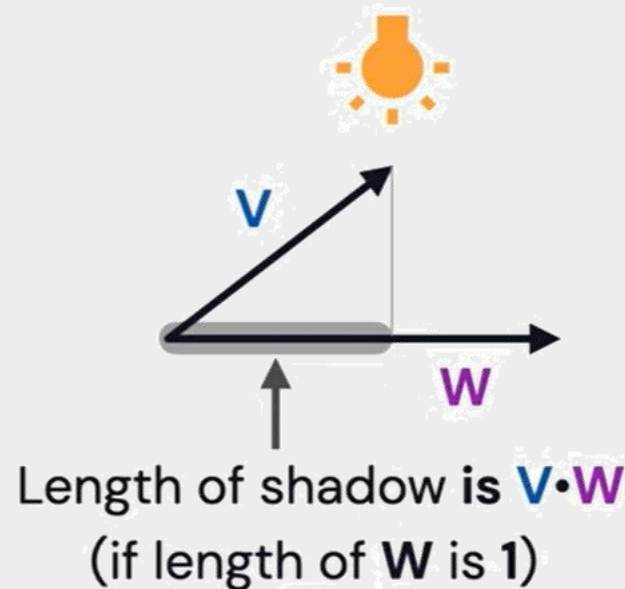
$$\mathbf{v} \cdot \mathbf{v} = \begin{bmatrix} 4 \\ 3 \end{bmatrix} \cdot \begin{bmatrix} 4 \\ 3 \end{bmatrix} = 16 + 9 = 25$$

Length(\mathbf{v})=5

$$\mathbf{a}^T \mathbf{a} = \|\mathbf{a}\|^2 = \sum_{i=1}^n a_i a_i = \sum_{i=1}^n a_i^2$$



- ❑ Represents the length of the “shadow” of one vector along another.
- ❑ This indicates how similar the two vectors are.





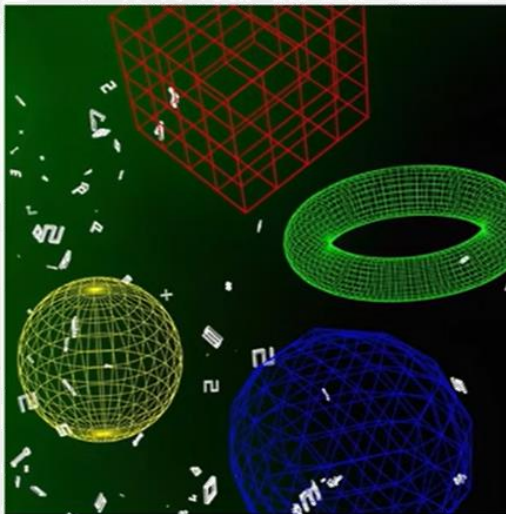
$$\text{apple} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \text{cat} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad \text{house} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad \text{tiger} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\text{apple} \cdot \text{cat} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \mathbf{0} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \text{tiger} \cdot \text{cat}$$



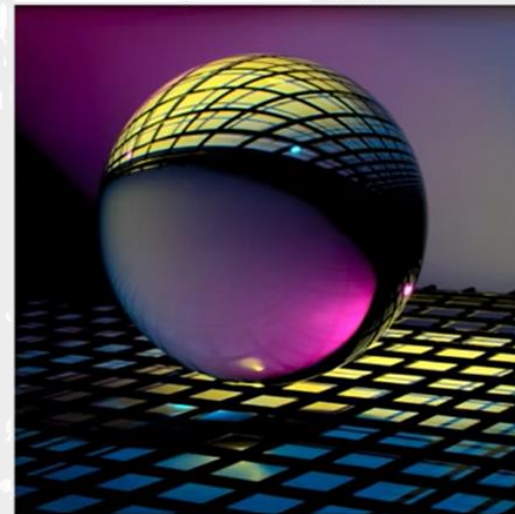
Data Representations

Use basic ideas of linear algebra to represent data in a way that computers can understand: vectors.



Vector Embeddings

Learn ways to choose these representations wisely via matrix factorizations.



Dimensionality Reduction

Deal with large-dimensional data using linear maps and their eigenvectors and eigenvalues.



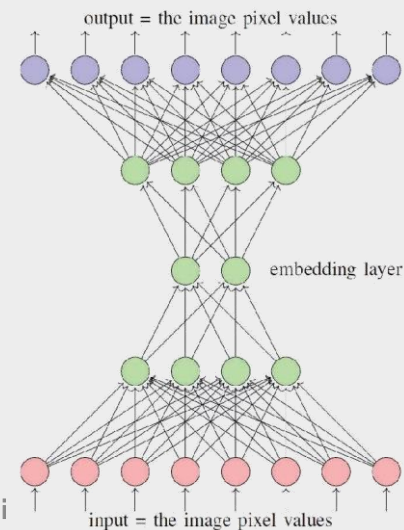
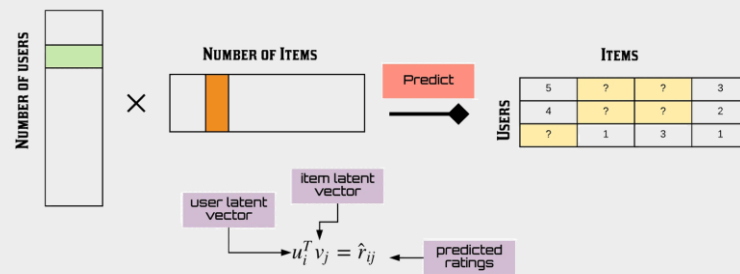
- An embedding of a vector is another vector in a smaller dimensional space.

Replace $\begin{bmatrix} * \\ * \\ * \end{bmatrix}$ with $\begin{bmatrix} * \\ * \end{bmatrix}$

Matrix
Factorization

Neural
Networks

MATRIX FACTORIZATION





- ❑ A **matrix** is a 2-dimensional array of numbers.
- ❑ **Matrix is a linear transformation**
 - It represents a particular process of turning one vector into another: stretching, rotating, scaling or something more complex.

$$\begin{bmatrix} 3 & 0 & 7 \\ 1 & -5 & 9 \\ 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix}$$

input output

Image Rotation



(a)



(b)

Image Scaling

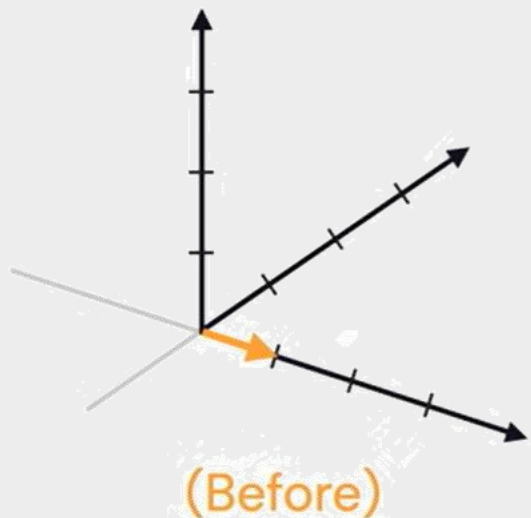


(a)

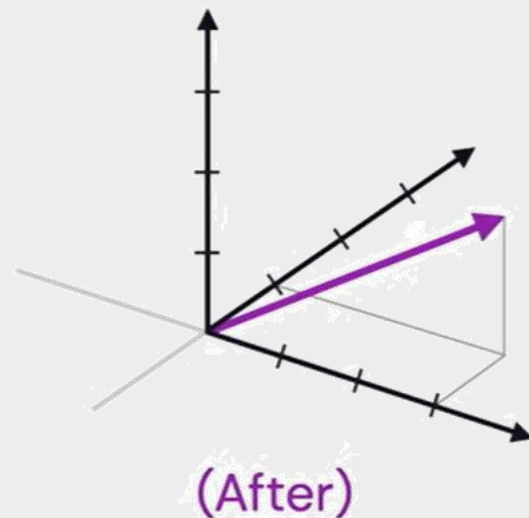


(b)

- A **matrix** represents a **transformation** of an entire vector space to another (possibly of different dimensions)



$$\begin{bmatrix} 3 & 0 & 7 \\ 1 & -5 & 9 \\ 2 & 0 & 0 \end{bmatrix}$$





- ❑ We can multiply **numbers** and get **number**.
- ❑ We can multiply **vectors** by dot product and get **number**.
- ❑ We can multiply **matrices** and get a **matrix**.

$$\begin{bmatrix} 3 & 0 & 7 \\ 1 & -5 & 9 \\ 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 2 \\ 1 & 0 \\ -1 & 0 \end{bmatrix} = \begin{bmatrix} -7 & 6 \\ -14 & 2 \\ 0 & 4 \end{bmatrix}$$

3×3 3×2 3×2

Sizes must match

- ❑ **Factorization?**



In general factorization is HARD!





- Fundamental **Theorem** in Linear Algebra:
 - **Every matrices can be factored!**

Theorem

Singular Value Decomposition (SVD)

Every $n \times m$ matrix can be written as a product of three smaller matrices as below:

$$\begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix}_{n \times m} = \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix}_{n \times k} \begin{bmatrix} \cdot & & & & \\ & \cdot & & & \\ & & \cdot & & \\ & & & \cdot & \\ & & & & \cdot \end{bmatrix}_{k \times k} \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix}_{k \times m}$$

Columns are "orthonormal" (orange arrow pointing to U)

Diagonal matrix (blue arrow pointing to D)

Rows are "orthonormal" (purple arrow pointing to V^T)

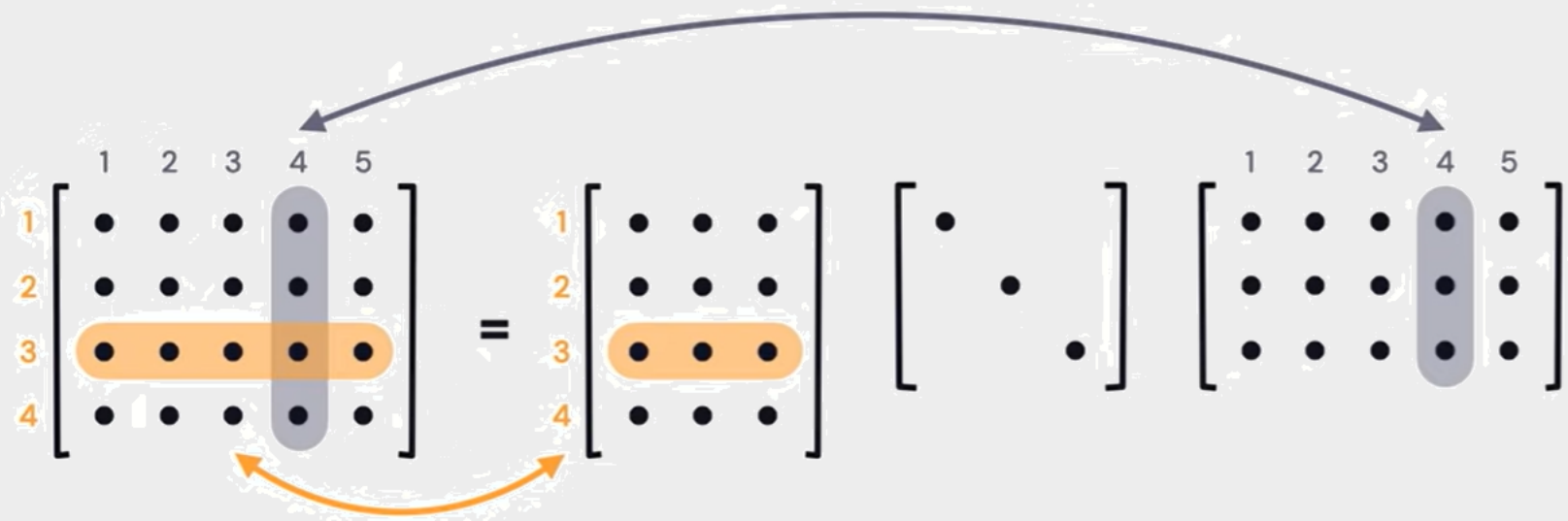
M U D V^T

$n \times m$ $n \times k$ $k \times k$ $k \times m$

($k = \text{rank of } M$)












- ❑ It has wide use in linear algebra and can be used directly in applications such as feature selection, visualization, noise reduction, and more.
- ❑ The columns/rows of the factors are candidates for embeddings.



Vector Embedding Applications

❑ User – Movie Matrix

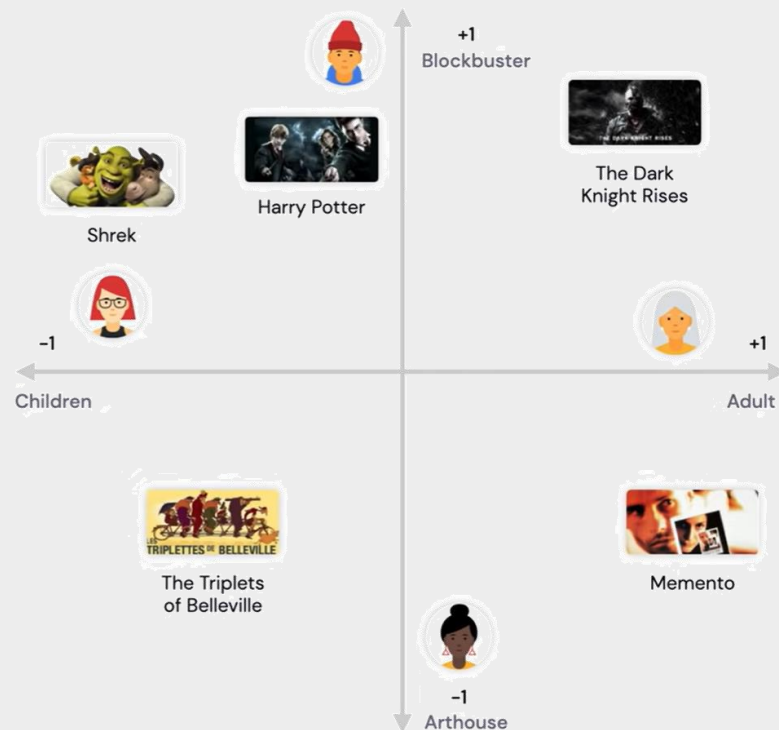
- Checkmarks = watched movie
- Empty cells = not watched movie

					
	Harry Potter	The Triplets of Belleville	Shrek	The Dark Knight Rises	Memento
	✓		✓	✓	
		✓			✓
	✓	✓	✓		
				✓	✓

$$\text{User 3} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$\text{Shrek} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

- ❑ We don't know the features!
 - Example: 2-dimensional “latent” feature space!
- ❑ We want to find the new, smaller dimensional vector representations that capture these features.












Recommender Systems



					
	Harry Potter	The Triplets of Belleville	Shrek	The Dark Knight Rises	Memento
	✓		✓	✓	
		✓			✓
	✓	✓	✓		
				✓	✓
	4×5				

$$\approx \begin{matrix} & U \\ & \begin{bmatrix} \bullet & \bullet \\ \bullet & \bullet \\ \bullet & \bullet \\ \bullet & \bullet \end{bmatrix} \\ 4 \times 2 \end{matrix} \quad \begin{matrix} & V^T \\ & \begin{bmatrix} \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \end{bmatrix} \\ 2 \times 5 \end{matrix}$$

					
	Harry Potter	The Triplets of Belleville	Shrek	The Dark Knight Rises	Memento
	✓		✓	✓	
		✓			✓
	✓	✓	✓		
				✓	✓

\approx

1	.1
-1	0
.2	-1
.1	1

U is a **user** × feature matrix

V^T is a feature × **movie** matrix

.9	-1	1	1	-.9
-.2	-.8	-1	.9	1

0.88	-1.08	0.9	1.09	-0.8
-0.9	1.0	-1.0	-1.0	0.9
0.38	0.6	1.2	-0.7	-1.18
-0.11	-0.9	-0.9	1.0	0.91

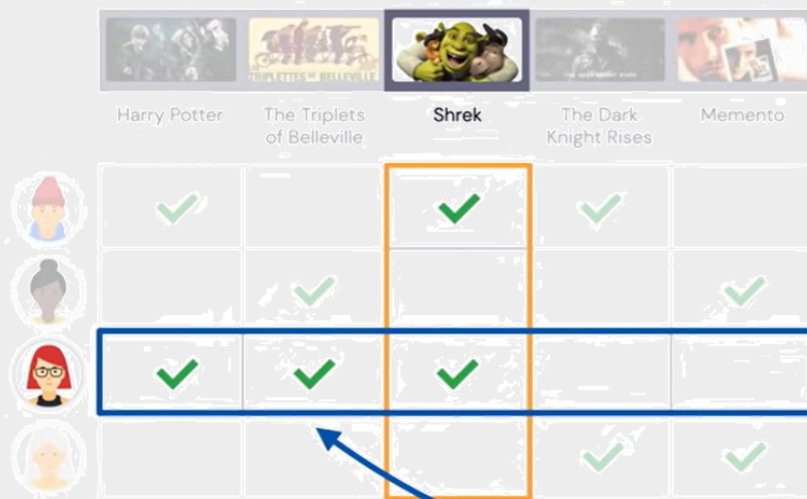
Recommender Systems



$$\text{User 3} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$\text{Shrek} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

New vector for Shrek



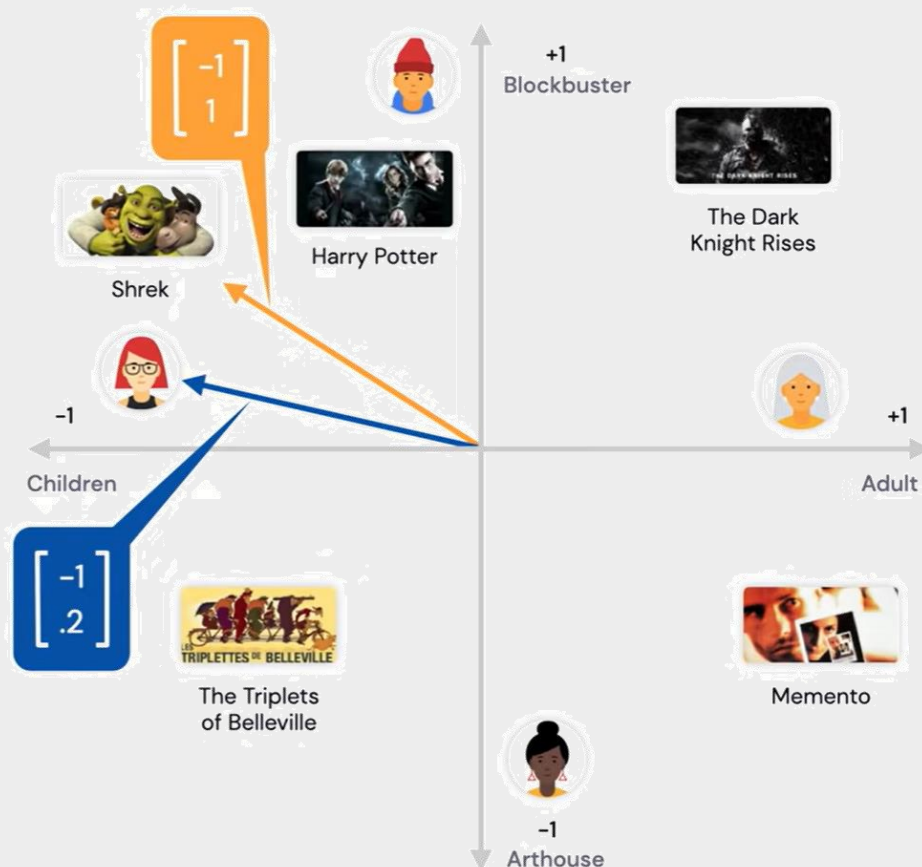
≈

1	.1
-1	0
.2	-1
.1	1

.9	-1	1	1	-.9
-.2	-.8	-1	.9	1

0.88	-1.08	0.9	1.09	-0.8
-0.9	1.0	-1.0	-1.0	0.9
0.38	0.6	1.2	-0.7	-1.18
-0.11	-0.9	-0.9	1.0	0.91

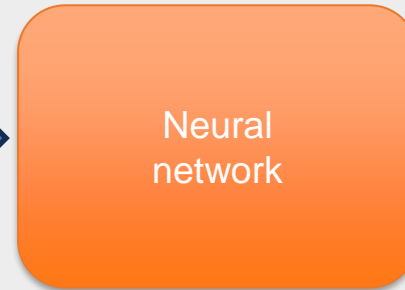
- ❑ These two vectors are close!
- ❑ The shadow of orange vector onto blue vector is pretty large!



Feed data vector into a Neural network. The output is vector embedding.

Under the hood:
Matrix multiplication *plus* more.

Movies viewed by user

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


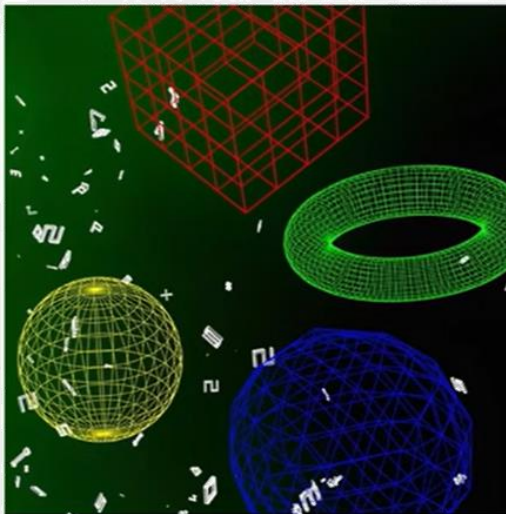
Vector embedding

$$\begin{bmatrix} * \\ * \\ * \\ * \end{bmatrix}$$



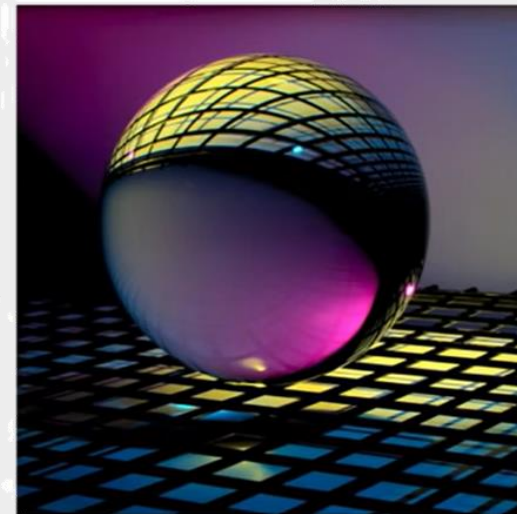
Data Representations

Use basic ideas of linear algebra to represent data in a way that computers can understand: vectors.



Vector Embeddings

Learn ways to choose these representations wisely via matrix factorizations.

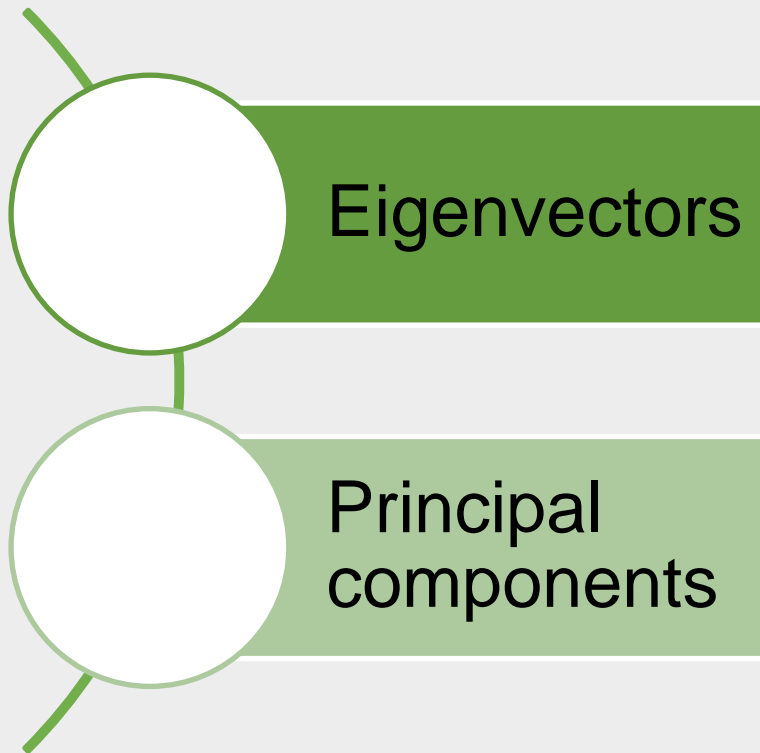


Dimensionality Reduction

Deal with large-dimensional data using linear maps and their eigenvectors and eigenvalues.

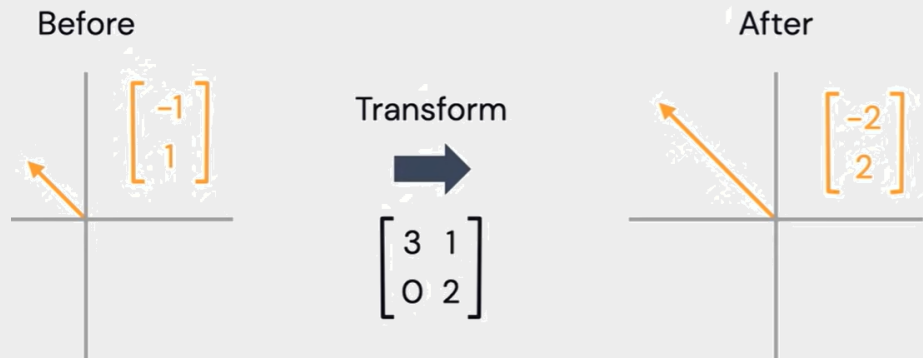


- ❑ “Compress” high-dimensional data into a smaller-dimensional, more meaningful subspace.
- ❑ This should be done in a way that doesn’t lose too much information.





- ❑ Matrix is a **transformation** between vector spaces
- ❑ There are some transformations for which some vectors never change direction, but are only scaled.



These special vectors are called **eigenvectors**
The scaling factor is called an **eigenvalue**.

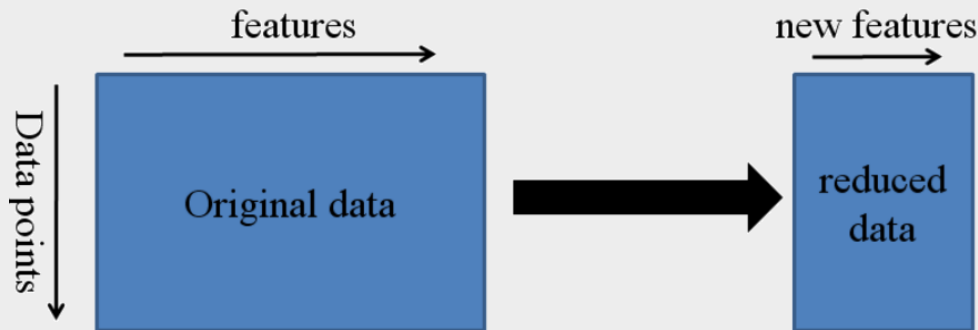
$$\begin{bmatrix} 3 & 1 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -2 \\ 2 \end{bmatrix} = 2 \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

M V = 2 V

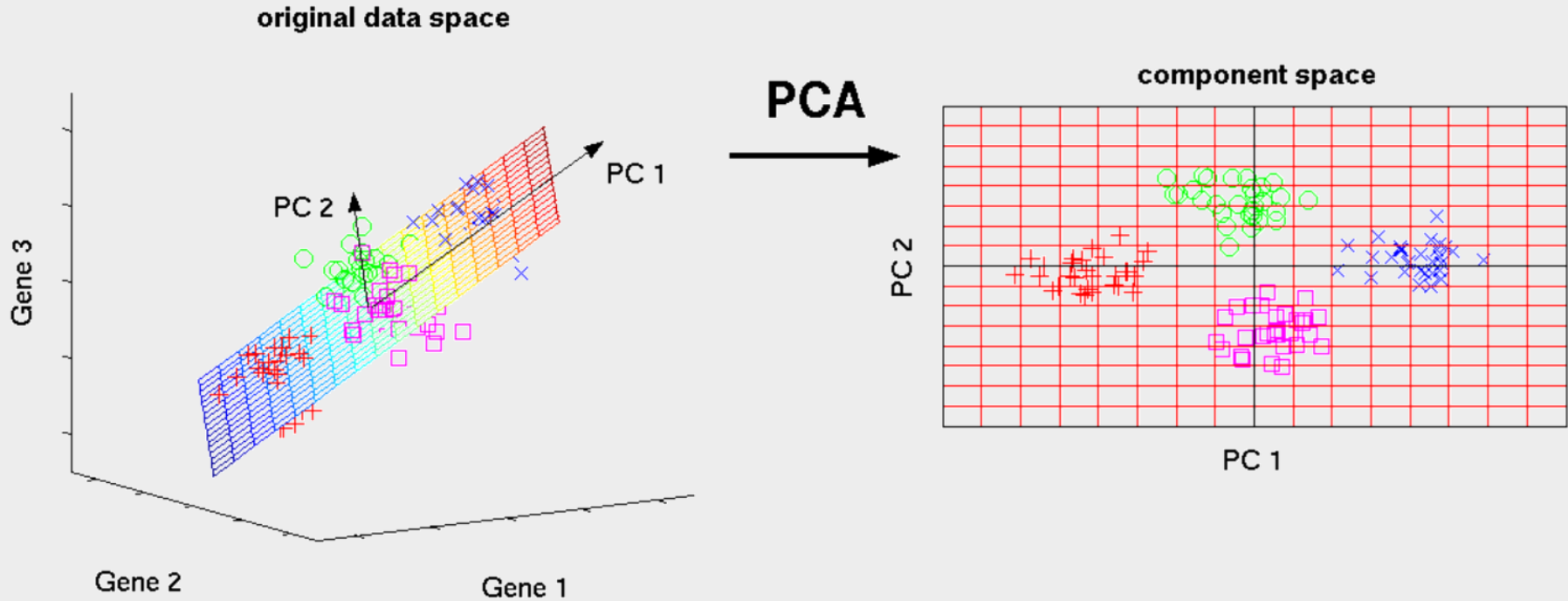


□ Principal Component Analysis

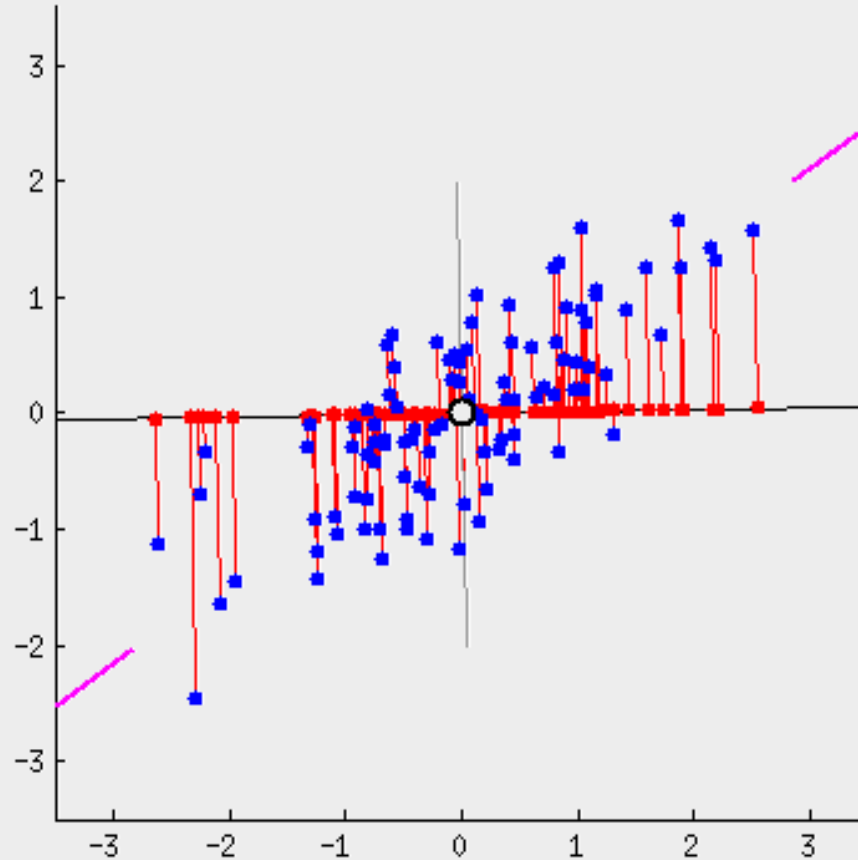
- Often, a dataset has many columns, perhaps tens, hundreds, thousands, or more.
- Methods for automatically reducing the number of columns of a dataset are called dimensionality reduction, and perhaps the most popular method is called the principal component analysis, or PCA for short
- The core of the PCA method is a matrix factorization method from linear algebra.



Principal Component Analysis (PCA)



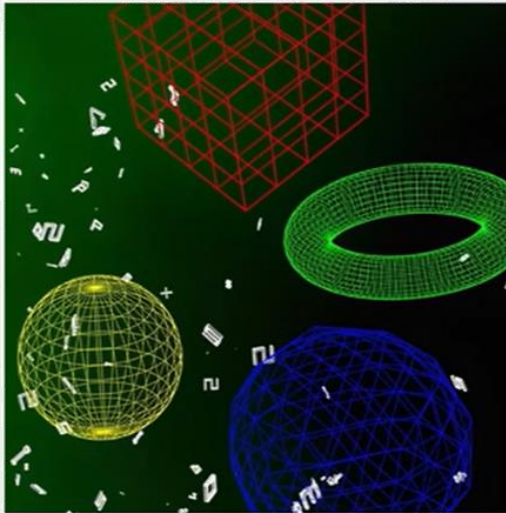
Principal Component Analysis (PCA)





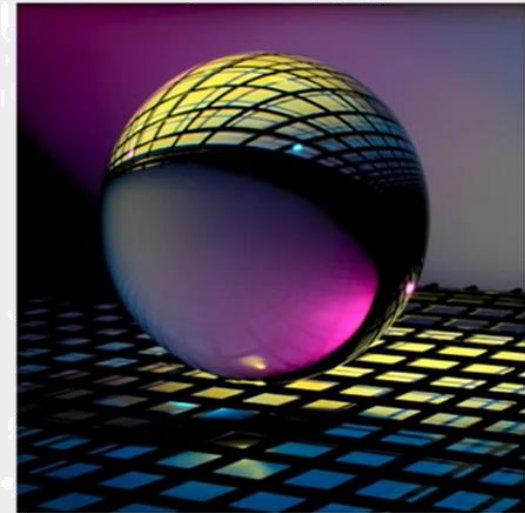
Data Representations

Use basic ideas of linear algebra to represent data in a way that computers can understand: vectors.



Vector Embeddings

Learn ways to choose these representations wisely via matrix factorizations.



Dimensionality Reduction

Deal with large-dimensional data using linear maps and their eigenvectors and eigenvalues.



- ❑ A friendly introduction to linear algebra for ML (ML Tech Talks) by TensorFlow
- ❑ Introduction to Applied Linear Algebra Vectors, Matrices, and Least Squares
- ❑ Linear Algebra and its applications
- ❑ Linear algebra A Modern Introduction David Poole