

---

---

# Singular Value Decomposition

— Boston University CS 506 - Lance Galletti —

---

---

# Recall

$$\begin{array}{c} \text{n data} \\ \text{points} \end{array} \left\{ \begin{pmatrix} x_{11} & \dots & x_{1j} & \dots & x_{1m} \\ \vdots & \ddots & \vdots & & \vdots \\ x_{i1} & \dots & x_{ij} & \dots & x_{im} \\ \vdots & & \vdots & \ddots & \vdots \\ x_{n1} & \dots & x_{nj} & \dots & x_{nm} \end{pmatrix} \right.$$

$\underbrace{\hspace{10em}}$   
m features

# Goal

Examine this matrix and uncover its linear algebraic properties to:

1. Approximate  $A$  with a smaller matrix  $B$  that is easier to store but contains similar information as  $A$
2. Dimensionality Reduction / Feature Extraction
3. Anomaly Detection & Denoising

# Linear Algebra Review

**Definition:** The vectors in a set  $\mathbf{V} = \{ \vec{v}_1, \dots, \vec{v}_n \}$  are **linearly independent** if

$$a_1 \vec{v}_1 + \dots + a_n \vec{v}_n = \vec{0}$$

can only be satisfied by  $a_i = 0$

**Note:** this means no vector in that set can be expressed as a **linear combination** of other vectors in the set.

# Linear Algebra Review

## Definition:

The **determinant** of a square matrix  $A$  is a scalar value that encodes properties about the **linear mapping** described by  $A$ .

2x2:

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

$$\det(A) = ad - bc$$

# Linear Algebra Review

## Definition:

The **determinant** of a square matrix  $A$  is a scalar value that encodes properties about the **linear mapping** described by  $A$ .

3x3:

$$A = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \quad \det(A) = a \cdot \det \begin{pmatrix} e & f \\ h & i \end{pmatrix} - b \cdot \det \begin{pmatrix} d & f \\ g & i \end{pmatrix} + c \cdot \det \begin{pmatrix} d & e \\ g & h \end{pmatrix}$$

# Linear Algebra Review

## Definition:

The **determinant** of a square matrix  $A$  is a scalar value that encodes properties about the **linear mapping** described by  $A$ .

$n \times n$ :

Can recursively compute it. How?

# Linear Algebra Review

## Property:

**n** vectors  $\{ \vec{v}_1, \dots, \vec{v}_n \}$  in an **n**-dimensional space are **linearly independent** iff the matrix **A**:

$$\mathbf{A} = [ \vec{v}_1, \dots, \vec{v}_n ] \text{ (n x n)}$$

has non-zero determinant.

**Q:** Can **m** > **n** vectors in an **n**-dimensional space be linearly independent?



# Linear Algebra Review

## Definition:

The **rank** of a matrix **A** is the dimension of the vector space spanned by its column space. This is equivalent to the maximal number of linearly independent columns / rows of **A**.

## Definition:

A matrix **A** is **full-rank** iff  $\text{rank}(\mathbf{A}) = \min(m, n)$

**Note:** Get the rank of a matrix through the **Gram-Schmidt process**

# Matrix Factorization

Any matrix **A** of rank **k** can be factored as

$$\mathbf{A} = \mathbf{UV}$$

where

**U** is **n x k**

**V** is **k x m**

# Matrix Factorization

To store an  $\mathbf{n} \times \mathbf{m}$  matrix  $\mathbf{A}$  requires storing  $\mathbf{m} \cdot \mathbf{n}$  values.

However, if the rank of the matrix of  $\mathbf{A}$  is  $\mathbf{k}$ , since  $\mathbf{A}$  can be factored as

$$\mathbf{A} = \mathbf{UV}$$

which requires storing  $\mathbf{k}(\mathbf{m} + \mathbf{n})$  values.

# In Practice

Most datasets are full rank despite containing a lot of redundant /similar information...

But we might be able to approximate the dataset with a lower rank one that contains similar information.

# Approximation

## Goal:

Approximate  $\mathbf{A}$  with  $\mathbf{A}^{(k)}$  (low-rank matrix) such that

1.  $\mathbf{d}(\mathbf{A}, \mathbf{A}^{(k)})$  is small
2.  $k$  is small compared to  $m$  &  $n$

# Frobenius Distance

$$d_F(A, B) = \|A - B\|_F = \sqrt{\sum_{i,j} (a_{ij} - b_{ij})^2}$$

i.e. the pairwise sum of squares difference in values of A and B

# Approximation

## Definition:

When  $k < \text{rank}(\mathbf{A})$ , the **rank- $k$  approximation** of  $\mathbf{A}$  (in the least squares sense) is

$$A^{(k)} = \arg \min_{\{B | \text{rank}(B)=k\}} d_F(A, B)$$

# Matrix Factorization Improved

Not only can we factorize a matrix **A** of rank **k** as **A = UV**. But we can factorize **A** using a process called Singular Value Decomposition where:

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



# Approximation

## Definition:

The **Singular Value Decomposition** of a rank- $r$  matrix  $A$  has the form

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

where

$U$  is  $n \times r$

The **columns** of  $U$  are orthogonal & unit length ( $U^T U = I$ )

$V$  is  $m \times r$

The **columns** of  $V$  are orthogonal & unit length ( $V^T V = I$ )

# Approximation

## Definition:

The **Singular Value Decomposition** of a rank-r matrix  $A$  has the form

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

where

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma_1 & & & 0 \\ & \sigma_2 & & \\ & & \ddots & \\ 0 & & & \sigma_r \end{pmatrix}$$

with  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > 0$

$\sigma_i$  is the square root of the eigenvalues of  $\mathbf{A}^T\mathbf{A}$  and are called **singular values**

# Approximation

Find  $\mathbf{A}^{(k)}$  by decomposing  $\mathbf{A}$ :

$$A = \begin{pmatrix} U_1 & U_2 \end{pmatrix} \begin{pmatrix} \Sigma_1 & \\ & \Sigma_2 \end{pmatrix} \begin{pmatrix} V_1 & V_2 \end{pmatrix}$$

$$\mathbf{A}^{(k)} = \mathbf{U}_1 \mathbf{\Sigma}_1 \mathbf{V}_1^T$$

Where

$\mathbf{U}_1$  is  $\mathbf{n} \times \mathbf{k}$

$\mathbf{\Sigma}_1$  is  $\mathbf{k} \times \mathbf{k}$

$\mathbf{V}_1$  is  $\mathbf{m} \times \mathbf{k}$

# Approximation

1	1	1	0	0
2	2	2	0	0
1	1	1	0	0
5	5	5	0	0
0	0	0	2	2
0	0	0	3	3
0	0	0	1	1

=

0.18	0
0.36	0
0.18	0
0.90	0
0	0.53
0	0.80
0	0.27

**x**

9.64	0
0	5.29

**x**

0.58	0.58	0.58	0	0
0	0	0	0.71	0.71

# Approximation

1	1	1	0	0
2	2	2	0	0
1	1	1	0	0
5	5	5	0	0
0	0	0	2	2
0	0	0	3	3
0	0	0	1	1

~

0.18	0
0.36	0
0.18	0
0.90	0
0	0.53
0	0.80
0	0.27

**x**

9.64	0
0	0

**x**

0.58	0.58	0.58	0	0
0	0	0	0.71	0.71

# Approximation

1	1	1	0	0
2	2	2	0	0
1	1	1	0	0
5	5	5	0	0
0	0	0	2	2
0	0	0	3	3
0	0	0	1	1

~

0.18	0
0.36	0
0.18	0
0.90	0
0	0.53
0	0.80
0	0.27

**x**

9.64	0
0	0

**x**

0.58	0.58	0.58	0	0
0	0	0	0.71	0.71

# Approximation

1	1	1	0	0
2	2	2	0	0
1	1	1	0	0
5	5	5	0	0
0	0	0	2	2
0	0	0	3	3
0	0	0	1	1

~

1	1	1	0	0
2	2	2	0	0
1	1	1	0	0
5	5	5	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

# Approximation

The  $i^{\text{th}}$  **singular vector** represents the direction of the  $i^{\text{th}}$  most variance.

$$\Sigma = \begin{pmatrix} \sigma_1 & & & 0 \\ & \sigma_2 & & \\ & & \ddots & \\ 0 & & & \sigma_r \end{pmatrix}$$



Singular Values express the importance / significance of a singular vector



# Approximation

**Property:**

$$d_F(A, A^{(k)})^2 = \sum_{i=k+1}^r \sigma_i^2$$

**Note:** the larger **k** is, the smaller the distance.

# Approximation

To find the right **k** you can:

1. Look at the singular value plot to find the elbow point
2. Look at the residual error of choosing different **k**

# Related to Principal Component Analysis (PCA)

SVD and PCA are related

See demo

# Anomaly Detection

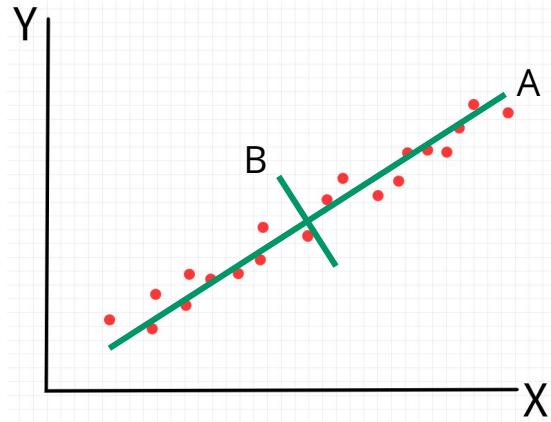
Define  $\mathbf{O} = \mathbf{A} - \mathbf{A}^{(k)}$

The largest rows of  $\mathbf{O}$  could be considered anomalies

# Dimensionality Reduction

**Idea:** project the data onto a subspace generated from a subset of singular vectors / principal components.

We want to project onto the components that capture most of the variance / information in the data.



Which principal component should we project on?