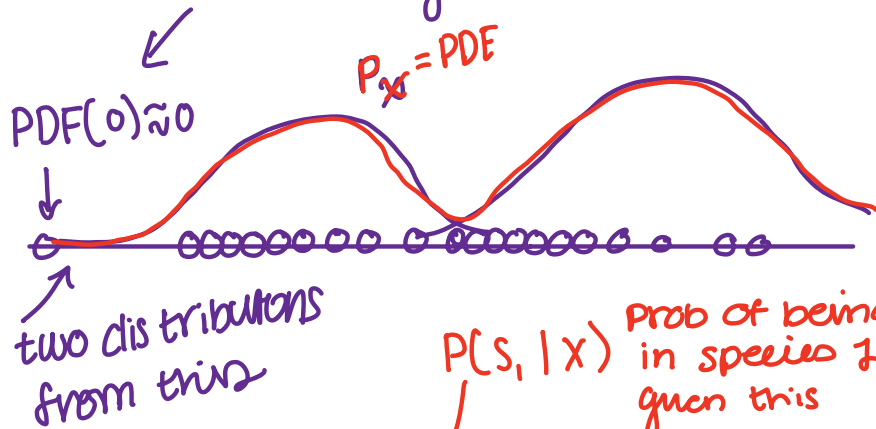

Singular Value Decomposition

— Boston University CS 506 - Lance Galletti —

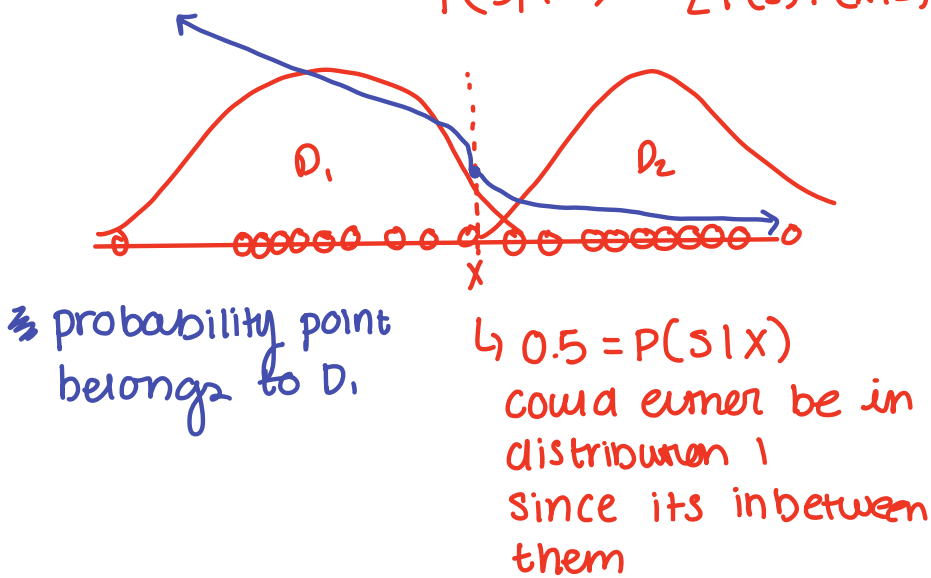
Quiz review:

probability this point comes from this dist



Quiz question about

$P(S_i | X)$ Prob of being in species i given this
 bayes rule $P(S) P(X|S)$
 $P(S_i | X) = \frac{P(S_i) P(X|S_i)}{\sum P(S_j) P(X|S_j)}$

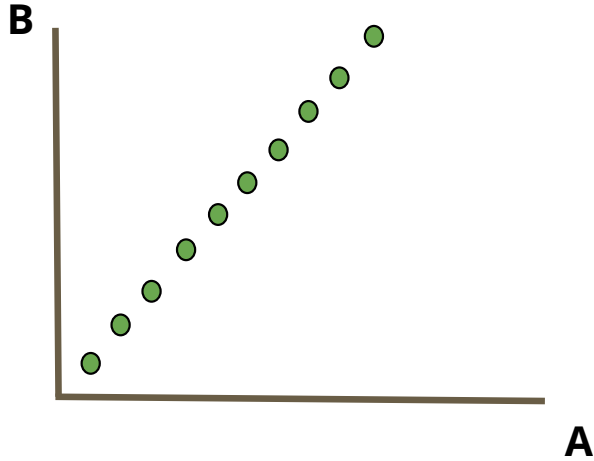


- context is important because of outliers that arent representative of overall trends

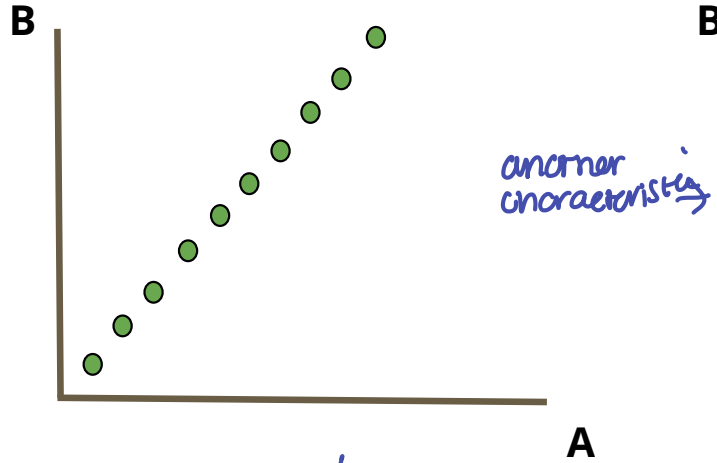
GMV estimates 3h components

maximum disagreement: $\frac{n \cdot (n-1)}{2} = \frac{5(4)}{2} = \frac{20}{2} = 10$

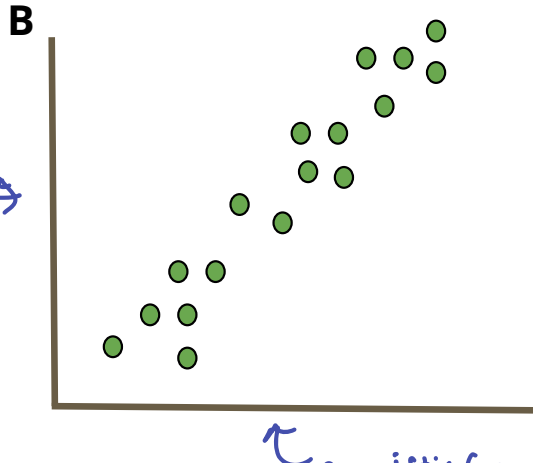
Characteristics of a dataset to look for



Characteristics of a dataset to look for



another characteristic →



positive change / positive trend

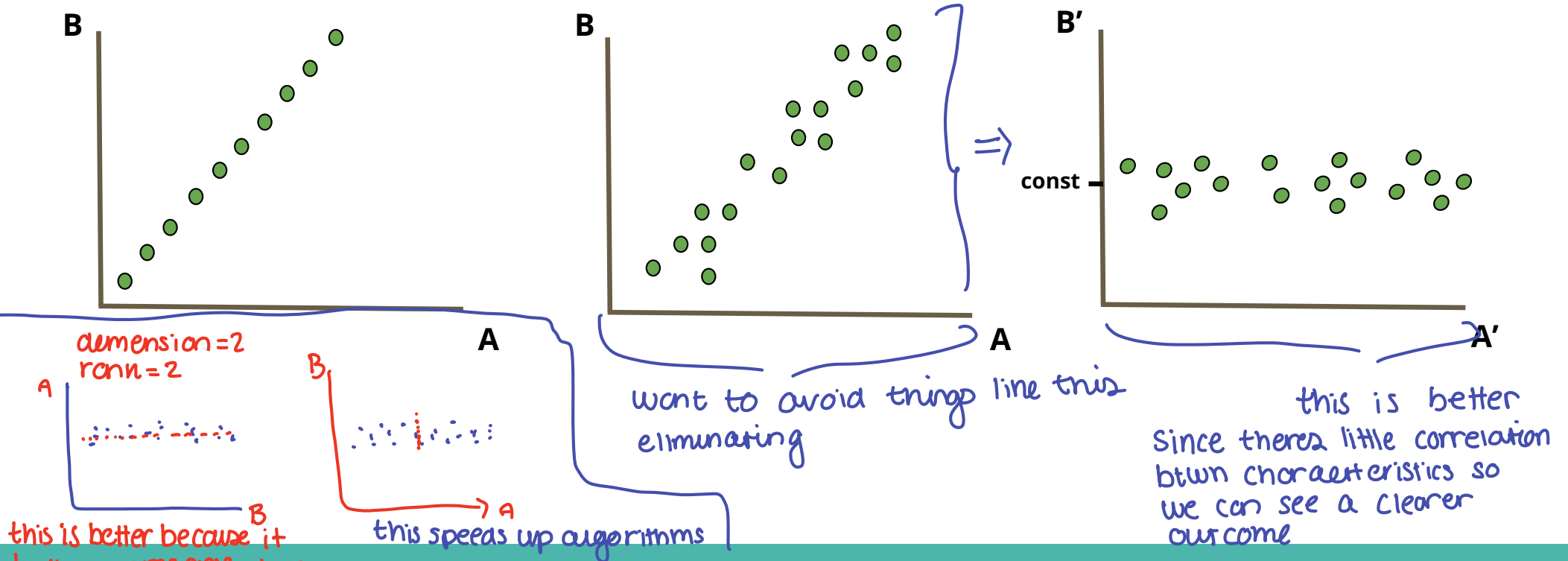
↳ increase in characteristics → increase in positive outcome

↳ characteristics shouldn't depend on each other because we want to isolate characteristics effects on overall outcome

↑ characteristic (study hrs, extracurriculars, family income)

Characteristics of a dataset to look for

goal of single value decomp. : eliminate correlations



better represents data
set

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

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

Goal

extracting / removing linear relationships

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
← dataset

Rank of matrix will be smaller
but similar

$$\begin{array}{c}
 \mathbf{A} \quad \dots \quad \mathbf{J} \quad \dots \quad \mathbf{M} \\
 \left. \begin{array}{c} \mathbf{n} \text{ data} \\ \text{points} \end{array} \right\} \begin{pmatrix} x_{11} & \dots & x_{1j} & \dots & x_{1m} \\ \vdots & \ddots & \vdots & & \vdots \\ x_{i1} & \dots & x_{ij} & \dots & x_{im} \\ \vdots & & \vdots & \ddots & \vdots \\ x_{n1} & \dots & x_{nj} & \dots & x_{nm} \end{pmatrix} \\
 \underbrace{\hspace{10em}}_{\mathbf{m} \text{ features}}
 \end{array}$$

$$\begin{array}{c}
 \mathbf{A}' \quad \dots \quad \mathbf{J}' \quad \dots \quad \mathbf{M}' \\
 \begin{pmatrix} x_{11} & \dots & x_{1j} & \dots & x_{1m} \\ \vdots & \ddots & \vdots & & \vdots \\ x_{i1} & \dots & x_{ij} & \dots & x_{im} \\ \vdots & & \vdots & \ddots & \vdots \\ x_{n1} & \dots & x_{nj} & \dots & x_{nm} \end{pmatrix} \\
 \underbrace{\hspace{10em}}_{\mathbf{m} \text{ features}}
 \end{array}$$

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

Projecting

$$\begin{array}{c}
 \mathbf{A} \quad \dots \quad \mathbf{J} \quad \dots \quad \mathbf{M} \\
 \left. \begin{array}{c} \mathbf{n} \text{ data} \\ \text{points} \end{array} \right\} \left(\begin{array}{ccccc} x_{11} & \dots & x_{1j} & \dots & x_{1m} \\ \vdots & \ddots & \vdots & & \vdots \\ x_{i1} & \dots & x_{ij} & \dots & x_{im} \\ \vdots & & \vdots & \ddots & \vdots \\ x_{n1} & \dots & x_{nj} & \dots & x_{nm} \end{array} \right) \\
 \underbrace{\hspace{10em}}_{\mathbf{m} \text{ features}}
 \end{array}$$

$$\begin{array}{c}
 \mathbf{A}'' \quad \dots \quad \mathbf{J}'' \\
 \left(\begin{array}{ccc} x_{11} & \dots & x_{1j} \\ \vdots & \ddots & \vdots \\ x_{i1} & \dots & x_{ij} \\ \vdots & & \vdots \\ x_{n1} & \dots & x_{nj} \end{array} \right) \\
 \underbrace{\hspace{10em}}_{\mathbf{j} \text{ features}}
 \end{array}$$

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
 - ↳ extracting outliers

Specify anomalous set of points

$$\begin{array}{c} \text{n data points} \left\{ \begin{array}{c} \left(\begin{array}{ccccc} x_{11} & \dots & x_{1j} & \dots & x_{1m} \\ \vdots & \ddots & \vdots & & \vdots \\ x_{i1} & \dots & x_{ij} & \dots & x_{im} \\ \vdots & & \vdots & \ddots & \vdots \\ x_{n1} & \dots & x_{nj} & \dots & x_{nm} \end{array} \right) \end{array} \right. \\ \underbrace{\hspace{10em}} \\ \text{m features} \end{array}$$

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:

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \qquad \det(A) = ad - bc$$

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

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


where

U is **n x k**

V is **k x m**

useful cuz you only need to store
a particular amount of data

Matrix Factorization

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

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

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

which requires storing $k(m + n)$ values.

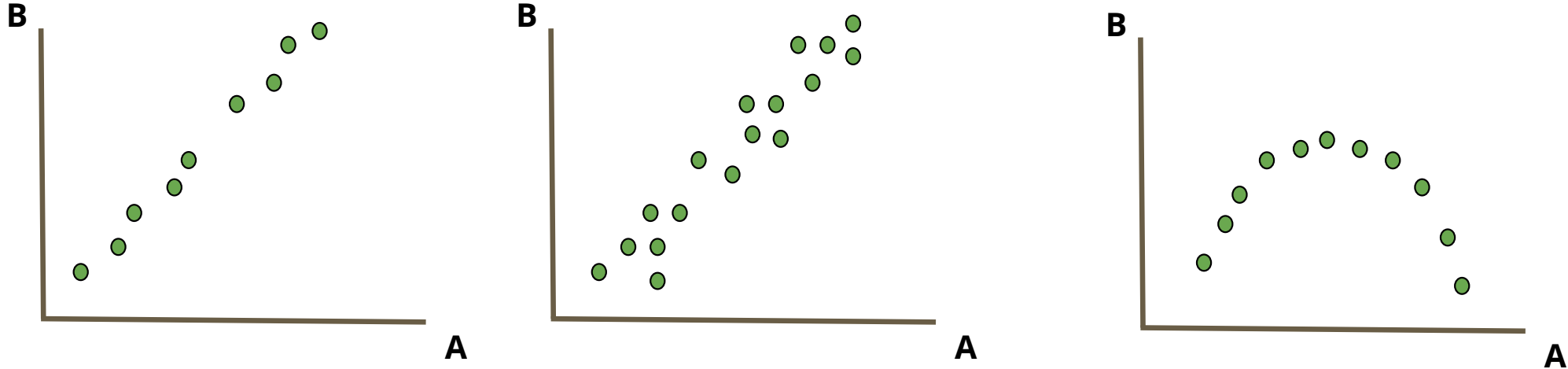
In Practice

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

datasets realistically have a high rank because of small variations that are difficult to detect

In Practice

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



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. $d(\mathbf{A}, \mathbf{A}^{(k)})$ 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 $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > 0$

σ_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} :

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

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

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	~	0.18	0	X	<table><tr><td>9.64</td><td>0</td></tr><tr><td>0</td><td>0</td></tr></table>	9.64	0	0	0	X	<table><tr><td>0.58</td><td>0.58</td><td>0.58</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0.71</td><td>0.71</td></tr></table>					0.58	0.58	0.58	0	0	0	0	0	0.71	0.71
9.64	0																												
0	0																												
0.58	0.58	0.58	0	0																									
0	0	0	0.71	0.71																									
2	2	2	0	0		0.36	0																						
1	1	1	0	0		0.18	0																						
5	5	5	0	0		0.90	0																						
0	0	0	2	2	0	0.53																							
0	0	0	3	3	0	0.80																							
0	0	0	1	1	0	0.27																							

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^{th} **singular vector** represents the direction of the i^{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)

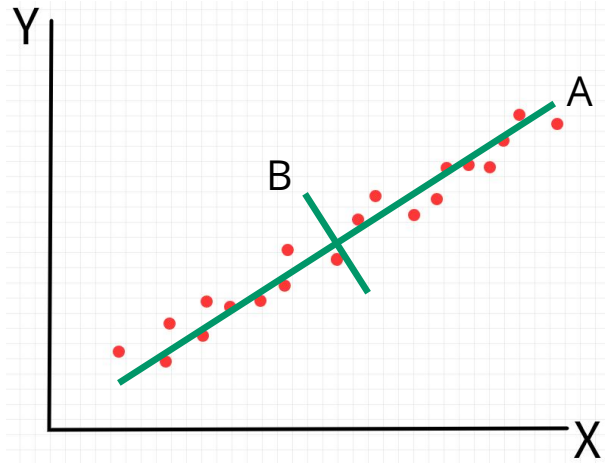
SVD and PCA are related

See demo

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?

Anomaly Detection

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

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