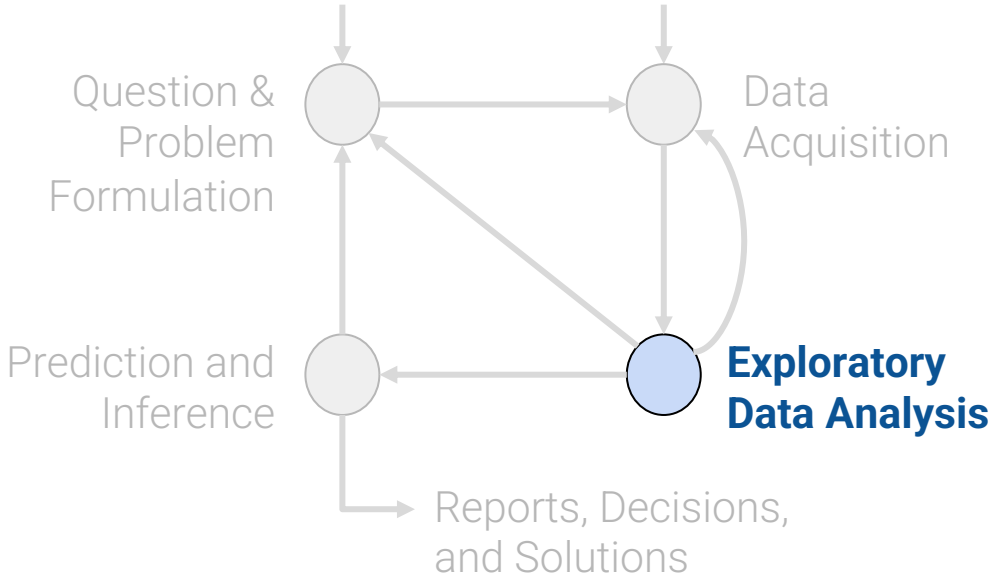
Principal Component Analysis

A New Tool for EDA using Singular Value Decomposition

PCA: A Technique for High Dimensional EDA and Featurization



SVD
PCA

today

PCA II
Applications

Principal Component Analysis (PCA) is a linear technique for dimensionality reduction.

PCA relies on a linear algebra algorithm called **Singular Value Decomposition**.

Today's Roadmap

Visualization Revisited

Dimensionality

Principal Component Analysis

Matrix as Transformation

Singular Value Decomposition

PCA with SVD

Data Variance and Centering

Visualization Revisited

Visualization Revisited

Dimensionality

Principal Component Analysis

Matrix as Transformation

Singular Value Decomposition

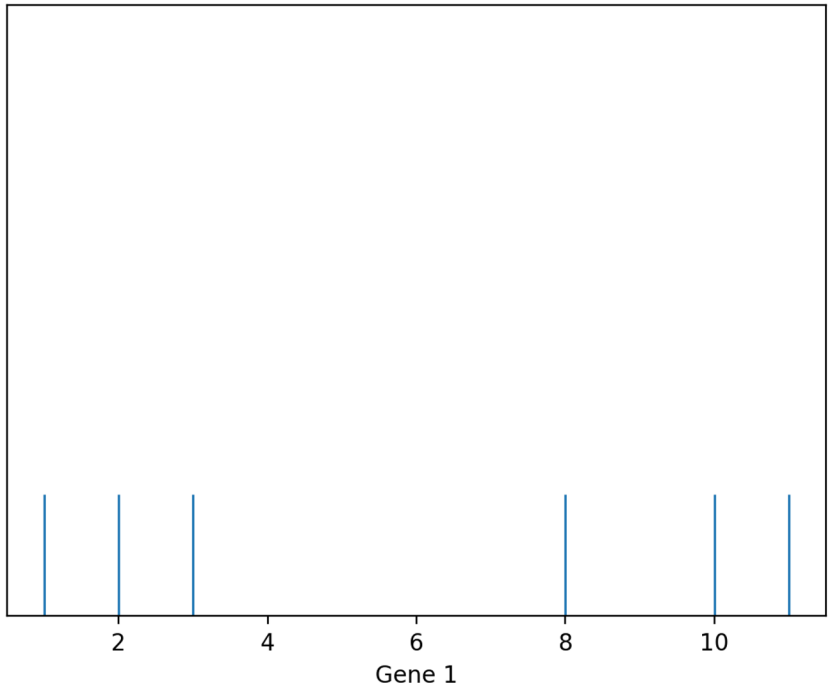
PCA with SVD

Data Variance and Centering

Visualizing Gene Data

Visualization can help us identify clusters in our dataset.

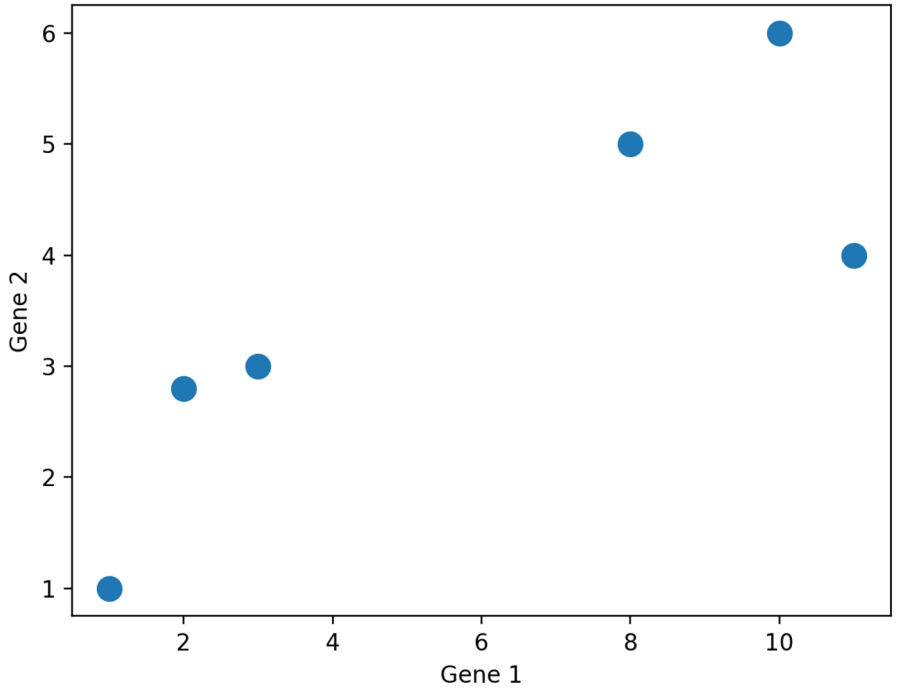
	Gene 1	Gene 2	Gene 3	Gene 4
0	10	6.0	12.0	5
1	11	4.0	9.0	7
2	8	5.0	10.0	6
3	3	3.0	2.5	2
4	2	2.8	1.3	4
5	1	1.0	2.0	7



Visualizing Gene Data

Visualization can help us identify clusters in our dataset.

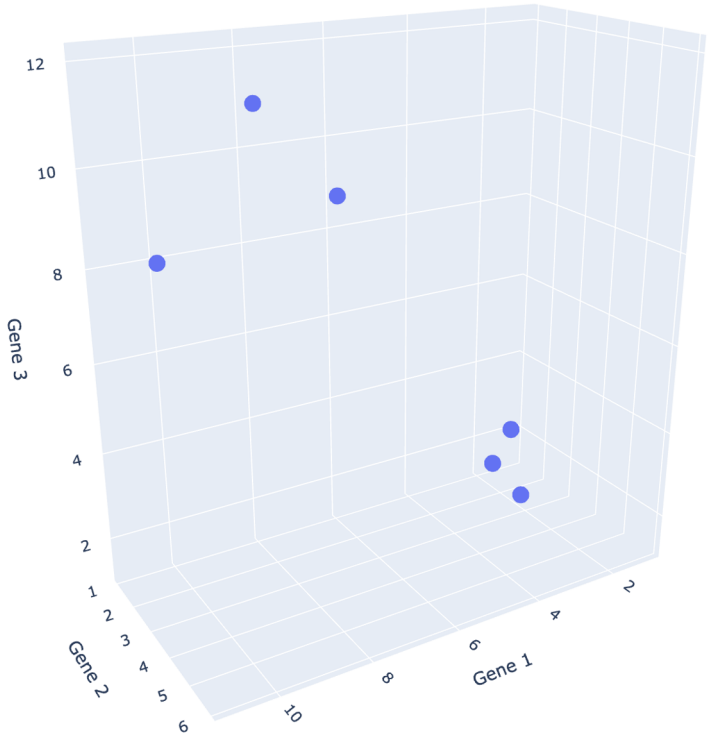
	Gene 1	Gene 2	Gene 3	Gene 4
0	10	6.0	12.0	5
1	11	4.0	9.0	7
2	8	5.0	10.0	6
3	3	3.0	2.5	2
4	2	2.8	1.3	4
5	1	1.0	2.0	7



Visualizing Gene Data

Visualization can help us identify clusters in our dataset.

	Gene 1	Gene 2	Gene 3	Gene 4
0	10	6.0	12.0	5
1	11	4.0	9.0	7
2	8	5.0	10.0	6
3	3	3.0	2.5	2
4	2	2.8	1.3	4
5	1	1.0	2.0	7



Visualizing Gene Data

Since we are all 3D beings, we can't visualize beyond three dimensions! However, many datasets come with more than three features. What can we do?

	Gene 1	Gene 2	Gene 3	Gene 4
0	10	6.0	12.0	5
1	11	4.0	9.0	7
2	8	5.0	10.0	6
3	3	3.0	2.5	2
4	2	2.8	1.3	4
5	1	1.0	2.0	7

We reduce the dataset to lower dimensions → **Dimensionality reduction**

Dimensionality

Visualization Revisited

Dimensionality

Principal Component Analysis

Matrix as Transformation

Singular Value Decomposition

PCA with SVD

Data Variance and Centering

Dimensionality of Data

Suppose we have a dataset of:

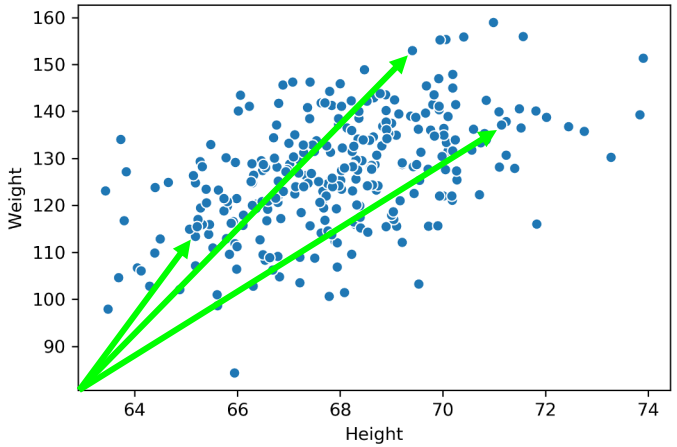
- **N** observations (datapoints)
- **d** attributes (features).

In Linear Algebra:

- **N points**/**row vectors** in a **d**-D space, OR
- **d** column vectors in an **N**-D space.

Height (in)	Weight (lbs)
65.8	113.0
71.5	136.5
69.4	153.0

Dataset 1



Dimensionality of Data

Suppose we have a dataset of:

- **N** observations (datapoints)
- **d** attributes (features).

In Linear Algebra:

- **N** points/row vectors in a **d**-D space, OR
- **d** column vectors in an **N**-D space.

From linear algebra, we know **dimension** of the column space of A is the **rank** of matrix A.

But what is the **dimensionality** of the data?
Let's try a few examples…

Height (in)	Weight (lbs)
65.8	113.0
71.5	136.5
69.4	153.0

Dataset 1

2 dimensions

Height (in)	Weight (lbs)	Age
65.8	113.0	17
71.5	136.5	21
69.4	153.0	18

Dataset 2

3 dimensions

Dimensionality of Data?

Consider the datasets shown.

What would you call these datasets?

A. 1-dimensional,
B. 2-dimensional,
C. 3-dimensional
D. Something else

Weight (lbs)	Weight (kg)
113.0	51.3
136.5	61.9
153.0	69.4

Dataset 3

Height (in)	Weight (kg)	Weight (lbs)	Age
65.8	51.3	113.0	17
71.5	61.9	136.5	21
69.4	69.4	153.0	18

Dataset 4

Dimensionality of Data?

Consider the datasets shown.

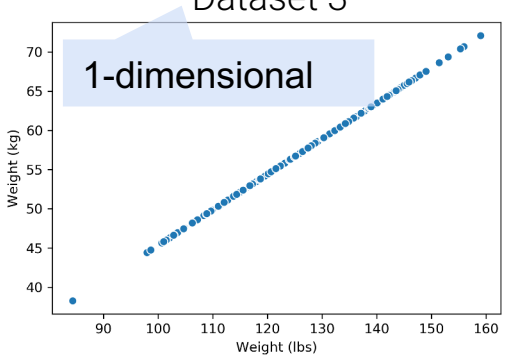
What would you call these datasets?

A. 1-dimensional, **C.** 3-dimensional

B. 2-dimensional, **D.** Something else

Weight (lbs)	Weight (kg)
113.0	51.3
136.5	61.9
153.0	69.4

Dataset 3



Height (in)	Weight (kg)	Weight (lbs)	Age
65.8	51.3	113.0	17
71.5	61.9	136.5	21
69.4	69.4	153.0	18

Dataset 4

Dimensionality of Data?

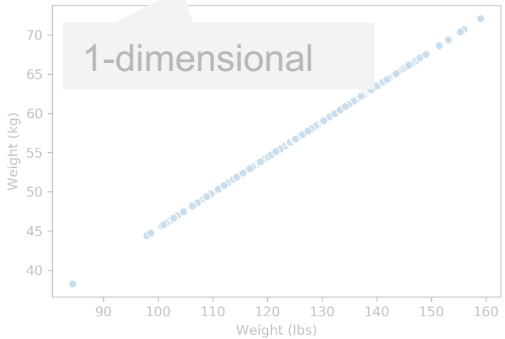
Consider the datasets shown.

What would you call these datasets?

A. 1-dimensional,
B. 2-dimensional,
C. 3-dimensional
D. Something else

Weight (lbs)	Weight (kg)
113.0	51.3
136.5	61.9
153.0	69.4

Dataset 3



Height (in)	Weight (kg)	Weight (lbs)	Age
65.8	51.3	113.0	17
71.5	61.9	136.5	21
69.4	69.4	153.0	18

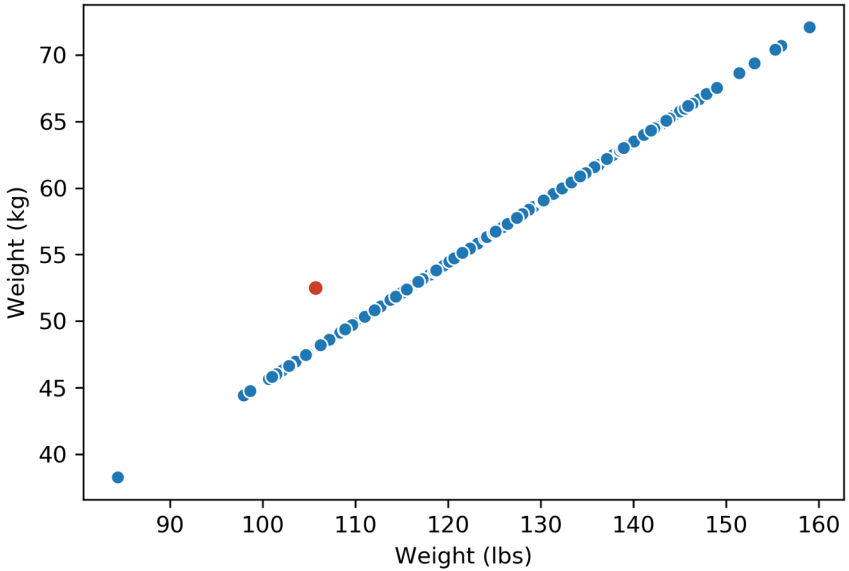
Dataset 4

- **3-dimensional**, because two weight columns are redundant.
- Notice: Matrix of dataset has (column) **rank** 3!

Dimensionality - what does it mean…?

Note that in the dataset below, I've added one **outlier** point to Dataset 3

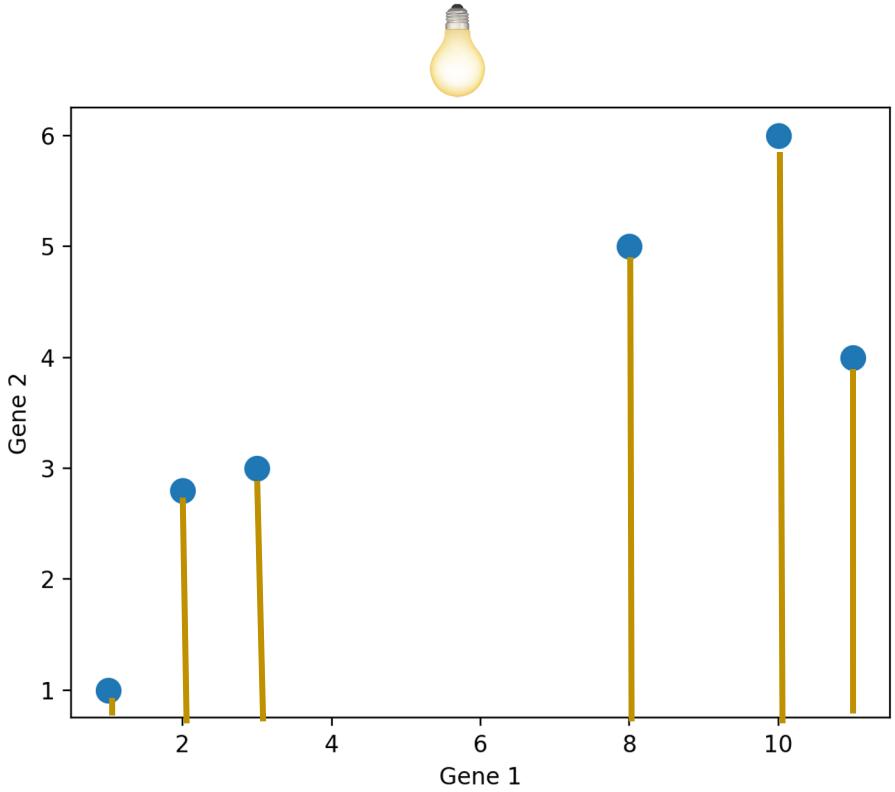
- Just this one outlier is enough to change the **rank** of the matrix to 2.
- But the data (in my opinion) is still **1-dimensional**!



Dimensionality reduction is generally an **approximation of the original data**.
This is achieved through projecting the data onto a desired dimension.

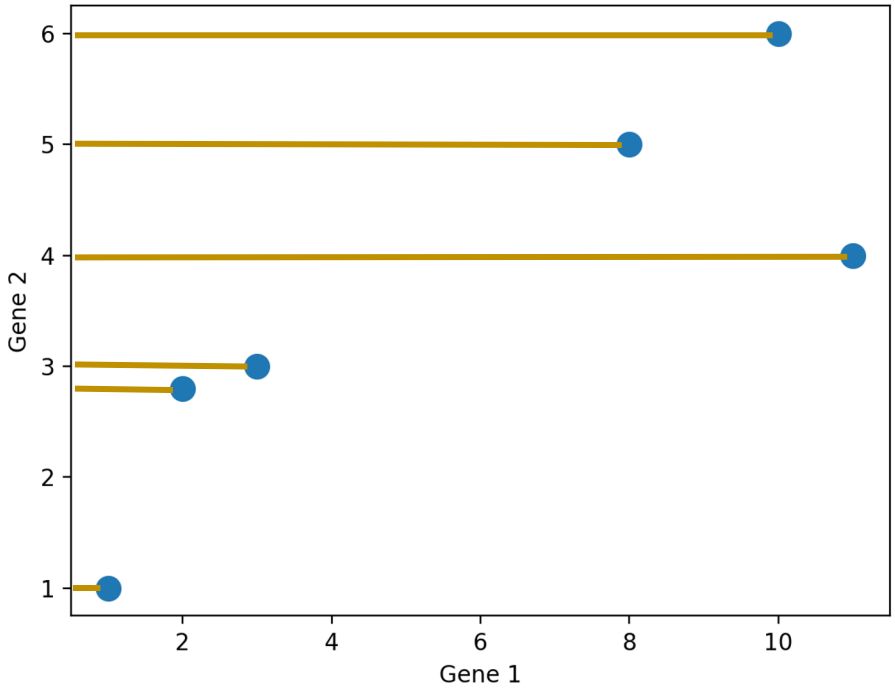
Reducing the Dimensions of Gene Data

How do we project a dataset onto a lower dimension? There are many ways to do it.



Reducing the Dimensions of Gene Data

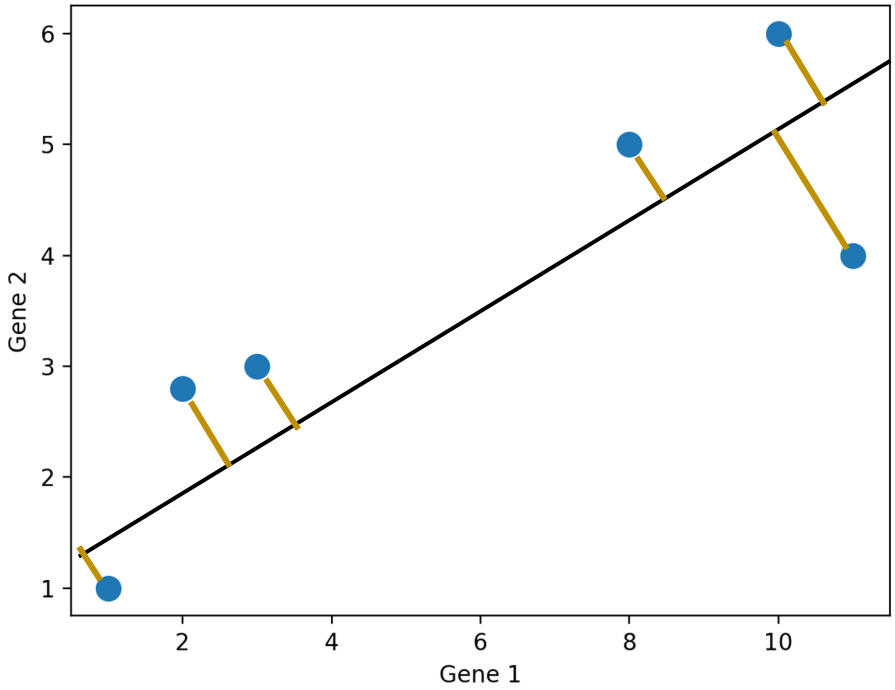
How do we project a dataset onto a lower dimension? There are many ways to do it.



Reducing the Dimensions of Gene Data

How do we project a dataset onto a lower dimension? There are many ways to do it.

How do we know which projection to choose?



In general, we want the projection that is the **best approximation** for the original data.

In other words, we want the projection that capture the **most variance** of the original data.

Principal Component Analysis

Visualization Revisited

Dimensionality

Principal Component Analysis

Matrix as Transformation

Singular Value Decomposition

PCA with SVD

Data Variance and Centering

Rectangle Dataset

We will consider a dataset containing measurements of rectangles: width, height, area, perimeter.

width	**height**	**area**	**perimeter**
20	20	400	80
16	12	192	56
...	...	...	...
24	12	288	72

What's the dimension of this dataset?

- Perimeter is 2(width + height) → a linear combination of columns
- Area is width x height → not a *linear* combination

The rank of the data matrix is 3, and I would say the dimension of the dataset is also 3.

Capturing Total Variance

We define the **total variance** of a data matrix as the sum of variances of the attributes.

width	**height**	**area**	**perimeter**
20	20	400	80
16	12	192	56
...	...	...	...
24	12	288	72

Total Variance: **402.56** = 7.69 5.35 50.79 338.73

Capturing Total Variance, Approach 1

We define the **total variance** of a data matrix as the sum of variances of attributes.

Total Variance: **402.56**

width	height	area	perimeter
20	20	400	80
16	12	192	56
...	...	...	...
24	12	288	72

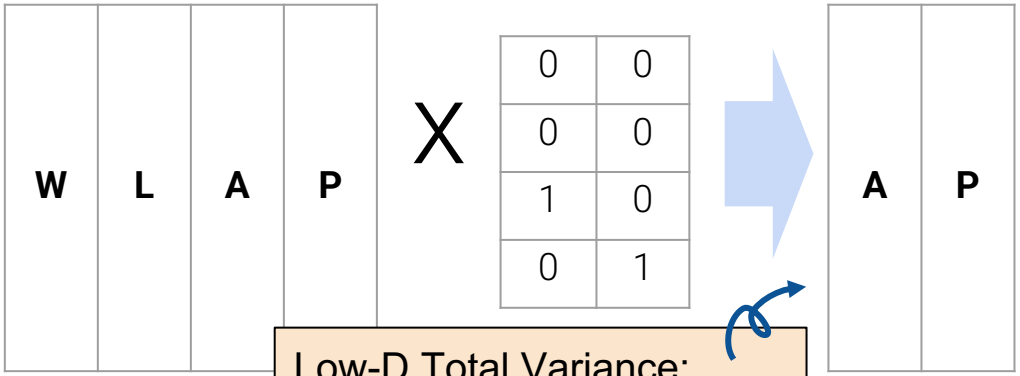
Reasonable **Approach 1**:

1. Find variances of each attribute

```
np.var(rectangle,axis=0).sort_values()
```

```
height          5.3475
width           7.6891
perimeter      50.7904
area          338.7316
dtype: float64
```

2. Keep the two attributes with highest variance.



Capturing Total Variance: PCA's approach

Reasonable **Approach 1**:

1. Find variances of each attribute

```
np.var(rectangle,axis=0).sort_values()

height          5.3475
width           7.6891
perimeter      50.7904
area          338.7316
dtype: float64
```

2. Keep the two attributes with highest variance.

W	L	A	P

X

0	0
0	0
1	0
0	1

A	P

Total Variance: **389.52**.

Approach 2: Principal Components Analysis

It turns out that the 2-D approximation that captures the most variance is the following:

-26.4	0.163
17.0	-2.18
…	…
11.8	-1.61
389.62	7.53

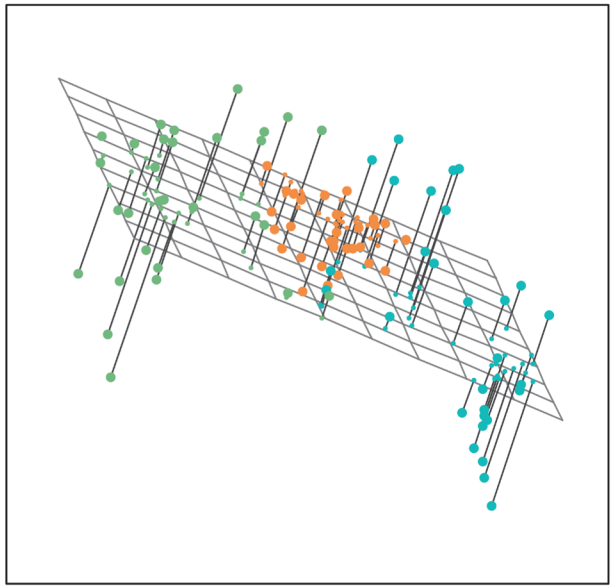
These **principal components** are also orthogonal vectors.

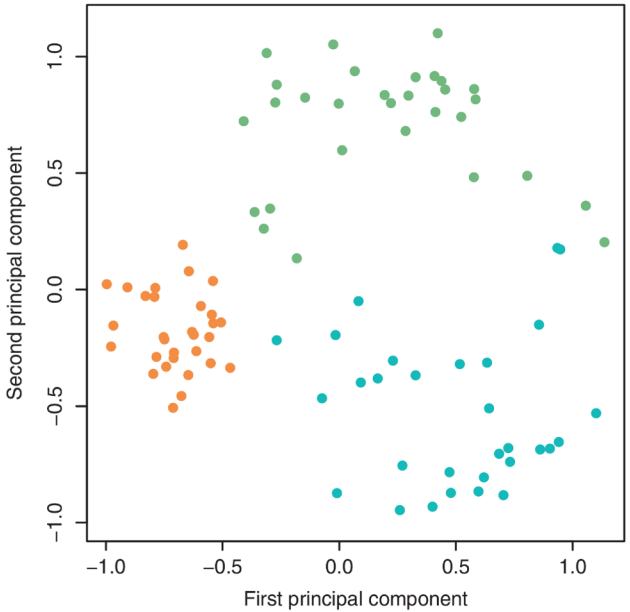
Total Variance: **397.15**.

Principal Component Analysis (PCA)

Goal: Transform observations from high-dimensional data down to **low dimensions** (often 2) through linear transformations.

Related Goal: Find a linear transformation that creates a low-dimension representation which captures as much of the original data's **total variance** as possible.





Why perform PCA?

We often perform PCA during the **Exploratory Data Analysis** (EDA) stage of our data science lifecycle (if we already know what to model, we probably don't need PCA); it helps us with

- Visually identifying clusters of similar observations in high dimensions.
- Removing irrelevant dimensions if we suspect that the dataset is inherently low rank. For example, if the columns are collinear: there are many attributes but only a few mostly determine the rest through linear associations.
- Finding a small basis for representing variations in complex things, e.g., images, genes.
- Reducing the number of dimensions to make some computation cheaper.

Why **two** dimensions?

- Most visualizations are 2-D! Choose the two axes on which to plot datapoints.

Principal Component Analysis: A Procedural View

1. Center the data matrix by subtracting the mean of each attribute column.

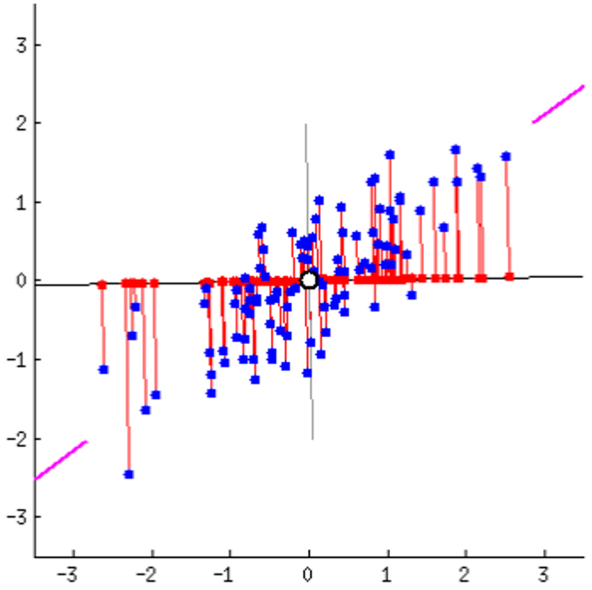
1. To find the i-th **principal component** $\boldsymbol{v_i}$:
   - $\boldsymbol{v}$ is a **unit vector** that linearly combines the attributes.
   - $\boldsymbol{v}$ gives a one-dimensional projection of the data.
   - $\boldsymbol{v}$ is chosen to minimize the sum of squared distances between each point and its projection onto $\boldsymbol{v}$.
   - Choose $\boldsymbol{v}$ such that it is orthogonal to all previous principal components.

k principal components capture the **most variance** of any k-dimensional reduction of the data matrix.

Principal Component Analysis: If you're curious

1. Center the data matrix by subtracting the mean of each attribute column.

1. To find the i-th **principal component** $\boldsymbol{v}_i$:
   - $\boldsymbol{v}$ is a **unit vector** that linearly combines the attributes.
   - $\boldsymbol{v}$ gives a one-dimensional projection of the data.
   - $\boldsymbol{v}$ is chosen to minimize the sum of squared distances between each point and its projection onto $\boldsymbol{v}$.
   - Choose $\boldsymbol{v}$ such that it is orthogonal to all previous principal components.



Maximizing variance = spreading out red dots
Minimizing error (i.e., projection)

=

making red lines short

[StackExchange]

k principal components capture the **most variance** of any k-dimensional reduction of the data matrix.

Principal Component Analysis: A Procedural View

1. Center the data matrix by subtracting the mean of each attribute column.

1. To find the i-th **principal component** $\boldsymbol{v}_i$:
   - $\boldsymbol{v}$ is a **unit vector** that linearly combines the attributes.
   - $\boldsymbol{v}$ gives a one-dimensional projection of the data.
   - $\boldsymbol{v}$ is chosen to minimize the sum of squared distances between each point and its projection onto $\boldsymbol{v}$.
   - Choose $\boldsymbol{v}$ such that it is orthogonal to all previous principal components.

In practice, we don't carry out this procedure.

Instead, we use **singular value decomposition (SVD)** to find all principal components efficiently.

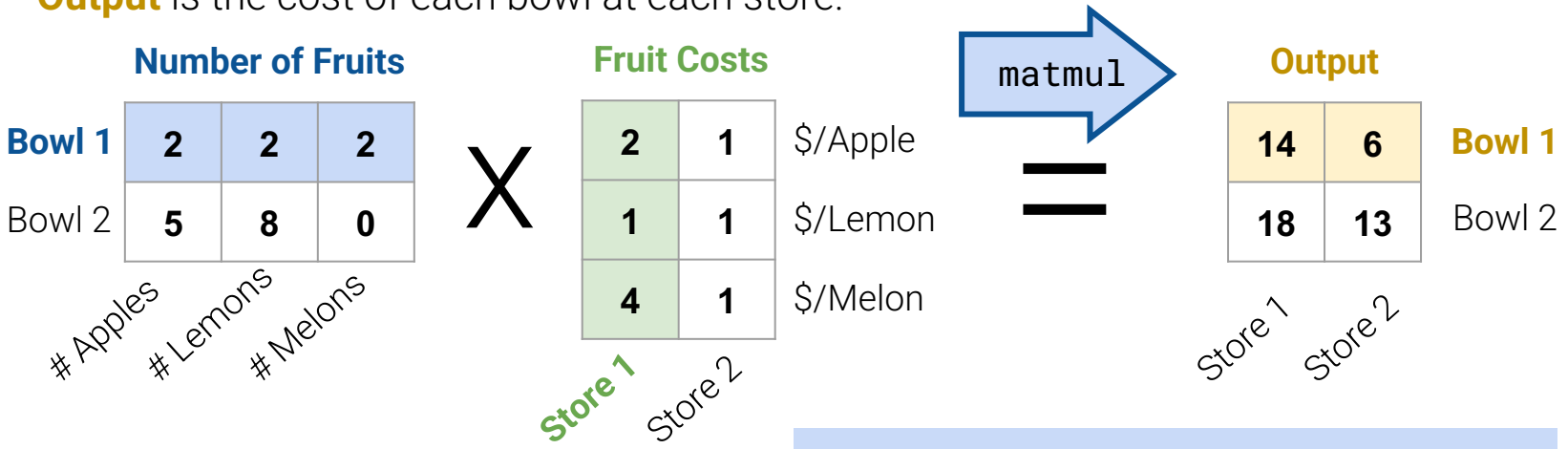
k principal components capture the **most variance** of any k-dimensional reduction of the data matrix.

Matrix as Transformation

[Linear Algebra] Interpreting Matrix multiplication

Consider the matrix multiplication example below.

- Each **row** of the **fruits matrix** represents one bowl of fruit.
  - First bowl: 2 apples, 2 lemons, 2 melons.
- Each **column** of the **dollars matrix** represents the cost of fruit at a store.
  - First store: 2 dollars for an apple, 1 dollar for a lemon, 4 dollars for a melon.
- **Output** is the cost of each bowl at each store.



Two ways to **interpret** matrix multiplication:
1. Row dot column to get each datapoint
2. Linear transformation of columns

[Linear Algebra] Multiplication View 1: Right matrix is Linear Operations

2	2	2
5	8	0

data

X

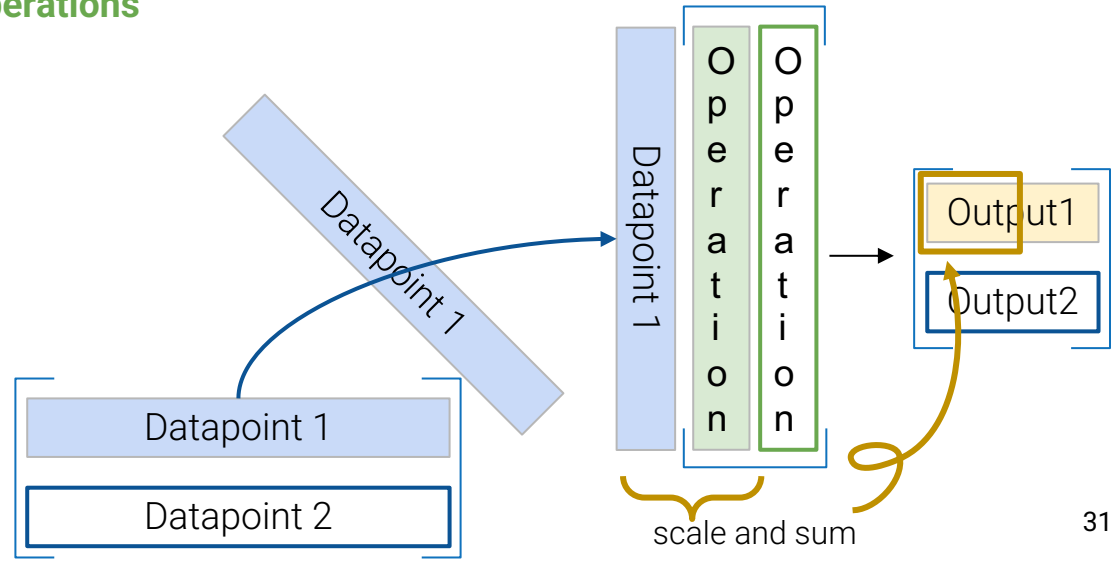
2	1
1	1
4	1

operations

=

14	6
18	13

View 1: Perform multiple linear operations on data.



[Linear Algebra] Multiplication View 2: Right Matrix Transforms Features

2	2	2
5	8	0

Original columns

X

2	1
1	1
4	1

Transformation

=

14	6
18	13

New column

View 2: Multiplication is a column transformation.

In this example, the right matrix transforms the left matrix from 3D to 2D!

$$\begin{bmatrix} 2 & 2 & 2 \\ 5 & 8 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ 4 \end{bmatrix}$$

$$= 2\begin{bmatrix} 2 \\ 5 \end{bmatrix} + 1\begin{bmatrix} 2 \\ 8 \end{bmatrix} + 4\begin{bmatrix} 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 14 \\ 18 \end{bmatrix}$$

3blue1brown's Essence of Linear Algebra

PCA Through Matrix Transformation

Goal: Transform observations from high-dimensional data down to **low dimensions** (often 2) through linear transformations.

Related Goal: Find a linear transformation that creates a low-dimension representation which captures as much of the original data's **total variance** as possible.

100 x 4

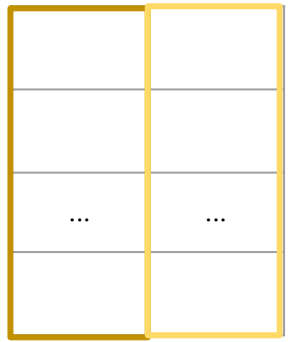
width	**length**	**area**	**perimeter**
20	20	400	80
16	12	192	56
...	...	...	...
24	12	288	72

X

4x2

Transformation

100 x 2



Principal Component 1 2

Singular Value Decomposition

SVD Pep Talk

Singular value decomposition (**SVD**) is an important concept in linear algebra.

- We assume you have taken (or are taking) a linear algebra course.
- We will not explain SVD in its entirety—in particular, we will not prove:
  - Why SVD is a valid decomposition of rectangular matrices, and
  - Why PCA is an application of SVD.

Today, we will not go much into the theory and details of SVD.

Instead, we will only cover what is needed for a data science interpretation.

Singular Value Decomposition

Singular value decomposition (SVD) describes a matrix decomposition into three matrices:

$$\underset{X \in \mathbb{R}^{n \times d}}{X} = \underset{U \in \mathbb{R}^{n \times d}}{U} \quad \underset{\Sigma \in \mathbb{R}^{d \times d}}{\Sigma} \quad \underset{V \in \mathbb{R}^{d \times d}}{V^\top}$$

columns of U are orthonormal

Columns of U are **left singular vectors**

diagonal matrix Σ

diagonal values (**singular values**), are ordered from **greatest to least**

r non-zero singular values

columns of V are orthonormal

Rows of V^T are **right singular vectors**

rank r ≤ d

There are infinite possible decompositions! SVD chooses a special (but non-unique) one with these nice properties

*note 1: assume d < n.

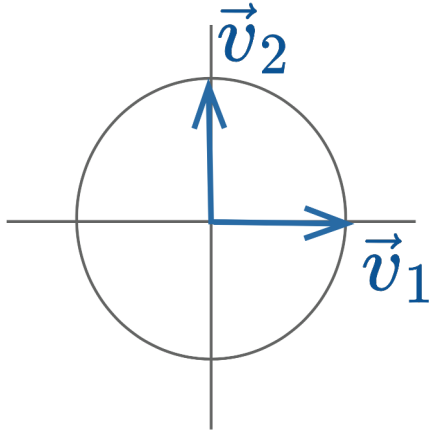
**note 2: many texts define U as square (n x n)

[Linear Algebra] Orthonormality

Orthonormal is a combination of two words: orthogonal and normal.

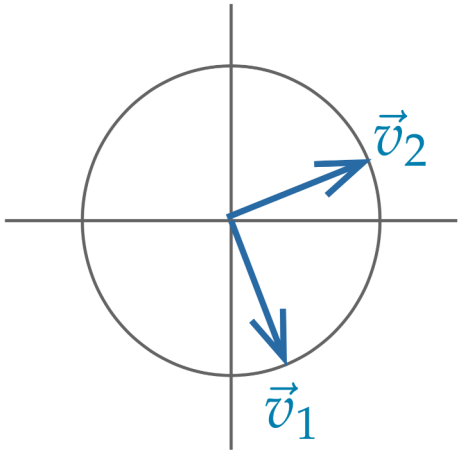
When we say the columns of a matrix are orthonormal, we are saying:

- The columns are all **orthogonal** to each other. I.e. All pairs of columns have a dot product of zero.
- All columns are **unit vectors**. The length of each column vector is 1.



Properties of orthonormal matrices:

- **Orthonormal inverse**: If an m x n matrix Q has orthonormal columns, $QQ^T = I_m$ and $Q^TQ = I_n$
- **Rotation of coordinates**: the linear transformation represented by an orthonormal matrix is often a **rotation** (and less often a reflection)
  - Imagine columns of the matrix as where the unit vectors of the original space will land.



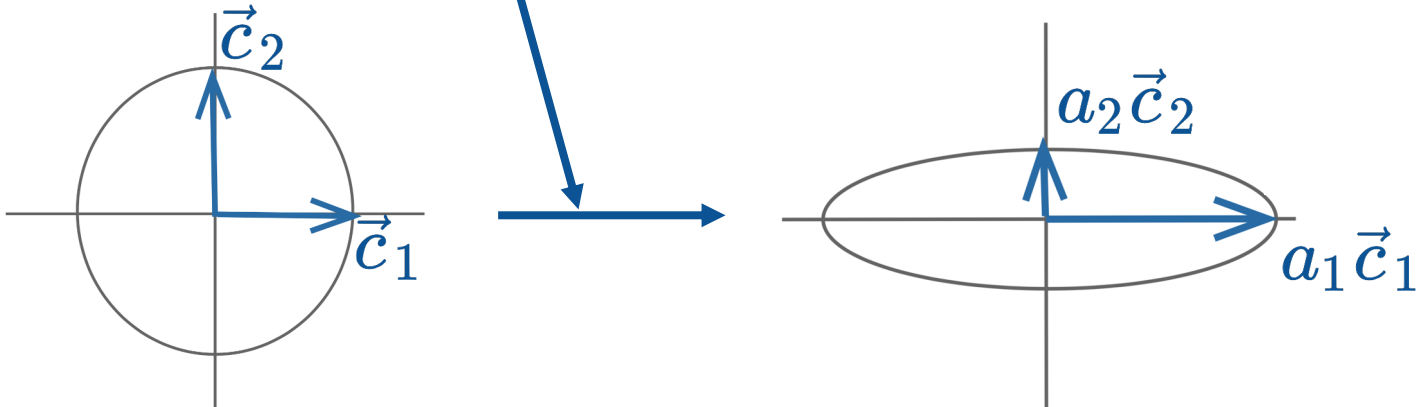
[Linear Algebra] Diagonal Matrices

Diagonal matrices are square matrices with non-zero values on the diagonal axis and zero everywhere else.

Right-multiplied diagonal matrices **scale** each column up or down by a constant factor.

$$\begin{bmatrix} | & | & | \\ \vec{c}_1 & \vec{c}_2 & \vec{c}_3 \\ | & | & | \end{bmatrix} \begin{bmatrix} a_1 & 0 & 0 \\ 0 & a_2 & 0 \\ 0 & 0 & a_3 \end{bmatrix} = \begin{bmatrix} | & | & | \\ a_1\vec{c}_1 & a_2\vec{c}_2 & a_3\vec{c}_3 \\ | & | & | \end{bmatrix}$$

Geometrically, this transformation can be viewed as scaling the coordinate system.

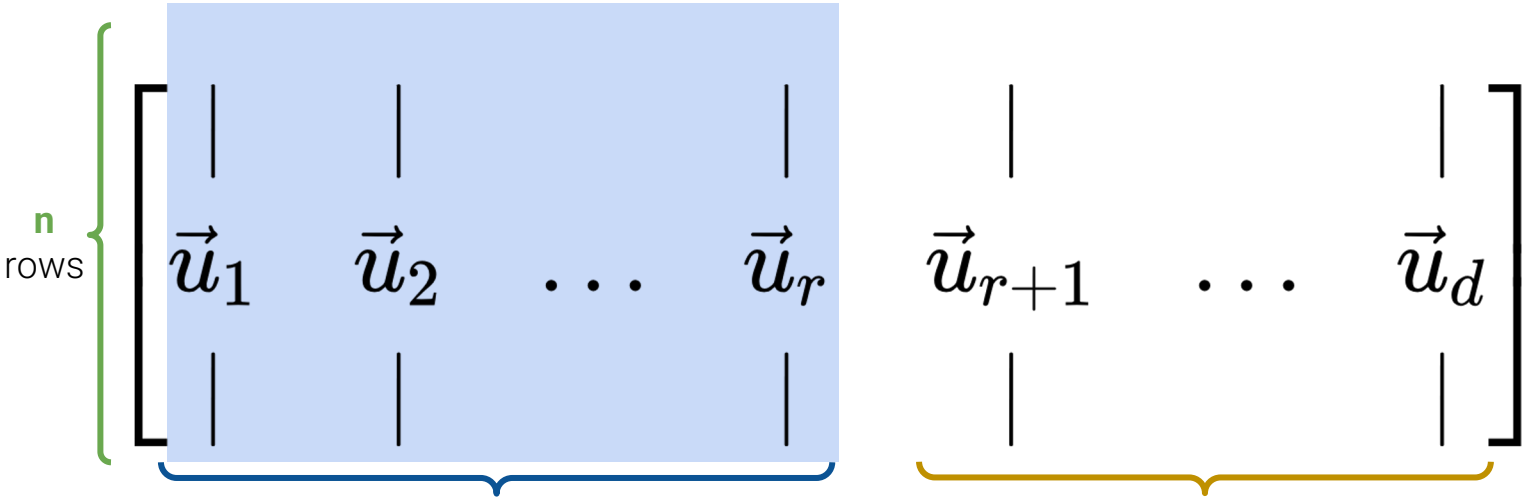


SVD One-by-One: U

$$U$$

$$U \in \mathbb{R}^{n \times d}$$

- Columns of U are **orthonormal**: $\vec{u}_i^\top \vec{u}_j = 0$ for all i, j and all vectors $\vec{u}_i$ are unit vectors
- Columns of U are called the **left singular vectors**
- $\boldsymbol{U}\boldsymbol{U}^T = I_n$ and $\boldsymbol{U}^T\boldsymbol{U} = I_d$
- Can think of $\boldsymbol{U}$ as a rotation



SVD One-by-One: Σ

$$\Sigma$$

$$\Sigma \in \mathbb{R}^{d \times d}$$

- Diagonal values (**singular values**), are ordered from **greatest to least**
- r **non-zero** singular values

$$\begin{bmatrix} \sigma_1 & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_r & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 \end{bmatrix}$$

d (rows) $\times$ d (columns)

- r is the rank of X
- The majority of the matrix is zero
- Useful information: the **singular values** $\sigma_1, \sigma_2, \dots, \sigma_r$
- Singular values are **sorted**: $\sigma_1 > \sigma_2 > \cdots > \sigma_r$
- Can think of Σ as a scaling operation

SVD One-by-One: V^T

$$V^\top$$

$$V \in \mathbb{R}^{d \times d}$$

- Columns of V are orthonormal → rows of V^T are orthonormal
- Columns of V are called the **right singular vectors**

$$\begin{bmatrix} — & \vec{v}_1^\top & — \\ — & \vec{v}_2^\top & — \\ & \cdots & \\ — & \vec{v}_d^\top & — \end{bmatrix}$$

d rows

d columns

- $VV^T = V^TV = I_d$
- Can think of V as a rotation

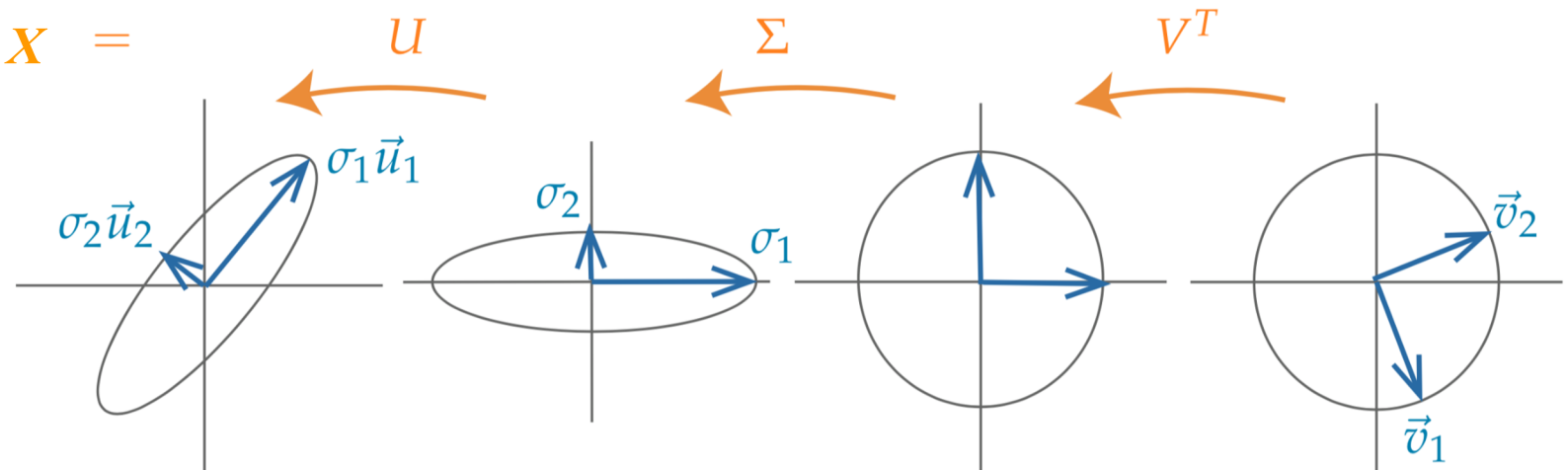
Singular Value Decomposition: Geometric Perspective

(out of scope)

Singular value decomposition (SVD) describes a matrix decomposition into three matrices:

$$\underset{X \in \mathbb{R}^{n \times d}}{X} = \underset{U \in \mathbb{R}^{n \times d}}{U} \quad \underset{\Sigma \in \mathbb{R}^{d \times d}}{\Sigma} \quad \underset{V \in \mathbb{R}^{d \times d}}{V^\top}$$

We've seen that $\boldsymbol{U}$ and $\boldsymbol{V}$ represent rotations, and $\boldsymbol{\Sigma}$ represents scaling. Therefore, SVD says that any matrix can be decomposed into a rotation, then a scaling, and another rotation.



SVD in NumPy

```
U, S, Vt = np.linalg.svd(X, full_matrices = False)
```

$$U \in \mathbb{R}^{n \times d}$$

$$X = U \Sigma V^\top$$

	width	height	area	perimeter
0	8	6	48	28
1	2	4	8	12
2	1	3	3	8
3	9	3	27	24
4	9	8	72	34

Demo Slides

Check the notebook for details

Demo Slides

Check the notebook for details

SVD in NumPy

```
U, S, Vt = np.linalg.svd(X, full_matrices = False)
```

$$U \in \mathbb{R}^{n \times d}$$

$$X = U \Sigma V^\top$$

0	1	2	3
-0.155151	0.064830	-0.029935	0.833096
-0.038370	-0.089155	0.062019	-0.454464
-0.020357	-0.081138	0.058997	0.004259
-0.101519	-0.076203	-0.148160	-0.019153
-0.218973	0.206423	0.007274	-0.062668

363
63
26
0

0	1	2	3
-0.146436	-0.129942	-0.810020	-0.552756
-0.192736	-0.189128	0.586348	-0.763727
-0.704957	0.709155	0.007952	0.008396
-0.666667	-0.666667	0.000000	0.333333

X is **rank 3**

From SVD to PCA

1. Center the data matrix by subtracting the mean of each attribute column.

(we'll come back to why centering is important)

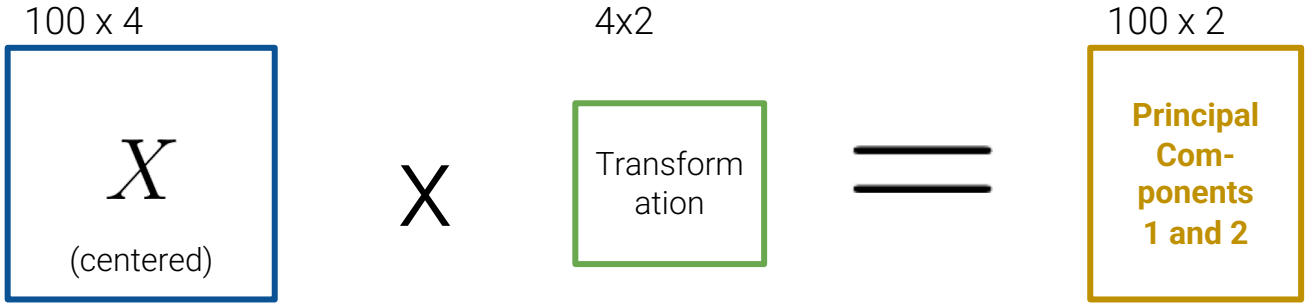
1. To find $\mathbf{v_i}$, the i-th **principal component**:
   - v is a **unit vector** that linearly combines the attributes.
   - v gives a one-dimensional projection of the data.
   - v is chosen to minimize the sum of squared distances between each point and its projection onto v.
   - Choose v such that it is orthogonal to all previous principal components.

Let's now use SVD to get us **principal components**.

PCA with SVD

PCA with SVD

Recall PCA applies a linear transformation (matrix multiplication) for dimensionality reduction:



SVD decomposes X into these matrices, where V has orthonormal columns and is square:

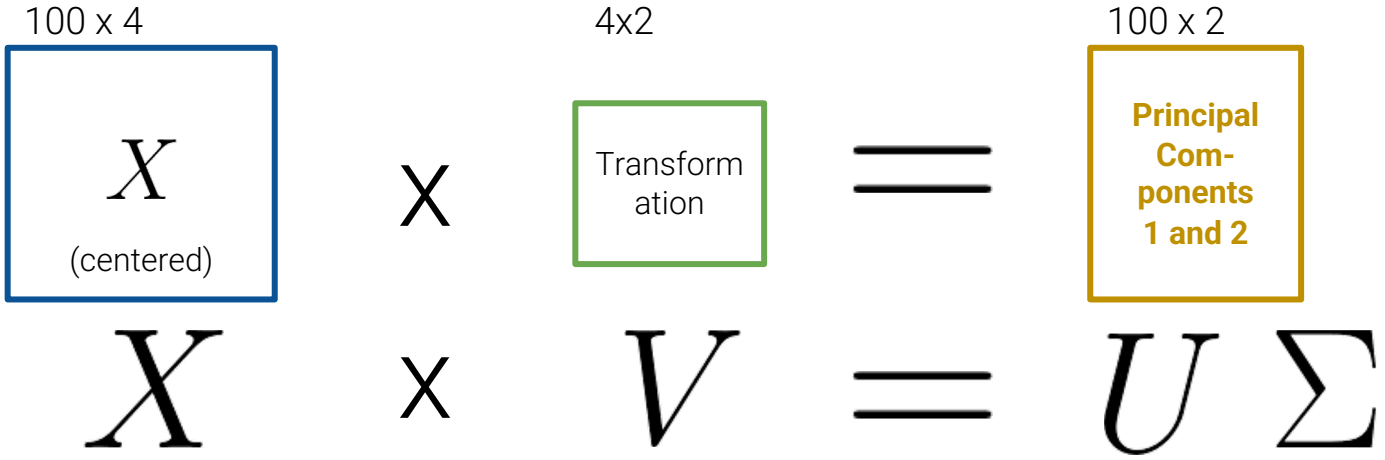
$$X = U\Sigma V^T$$

If we right-multiply both sides by V:

$$XV = U\Sigma V^\top V$$

PCA with SVD

To get the first **two principal components**, first find the SVD $X = U\Sigma V^T$.



Take the first 2 **columns of $U\Sigma$** (or XV). These are **PC1** and **PC2**.

Principal Component Analysis

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

100 x 4 | 100 x 4 | 4 x 4 | 4 x 4

$$\begin{bmatrix} | & | & & | & | & & | \\ \vec{u}_1 & \vec{u}_2 & \dots & \vec{u}_r & \vec{u}_{r+1} & \dots & \vec{u}_d \\ | & | & & | & | & & | \end{bmatrix}$$

$$\begin{bmatrix} \sigma_1 & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_r & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 \end{bmatrix}$$

$$\begin{bmatrix} - & \vec{v}_1^\top & - \\ - & \vec{v}_2^\top & - \\ & \dots & \\ - & \vec{v}_d^\top & - \end{bmatrix}$$

Two questions:

1. Based on the above SVD, what are PC1 and PC2?
2. Based on the above SVD, what is a rank 1 approximation of X?

Q1: What is PC1? Approach 1 of 2

$$XV = U \qquad \Sigma$$

$$= \begin{bmatrix} | & | & & | \\ \vec{u}_1 & \vec{u}_2 & \dots & \vec{u}_d \\ | & | & & | \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & \dots & 0 & 0 \\ 0 & \sigma_2 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & \sigma_r & 0 \\ 0 & 0 & \dots & 0 & 0 \end{bmatrix}$$

PC1 = (**1st left singular vector**) x (**1st singular value**)

PC2 = (**2nd left singular vector**) x (**2nd singular value**)

Q1: What is PC1? Approach 2 of 2

$$X \times V = U \Sigma$$

-0.13	0.01	0.03	-0.30
0.09	-0.08	0.01	0.71
...	...	...	...
0.06	-0.06	-.11	-0.07

X

(centered)

=

-.10	0.67	0.31	0.67
-.07	-0.37	-0.64	0.67
-0.93	-0.26	0.26	0
-0.34	0.59	-0.65	-0.33

x

PC1

(**right singular vector**)

=

-26.4
17.0
...
11.8

Summary

How to get Principal Components?

Given an n x d matrix $\boldsymbol{X}$, here's how we get the first k PCs:

1. **Center** $\boldsymbol{X}$ by subtracting the mean of each column
2. Get the **Singular Value Decomposition** of centered $\boldsymbol{X}$: $\boldsymbol{U}, \boldsymbol{\Sigma}$ and $\boldsymbol{V}^T$
3. **Multiply** either $\boldsymbol{U\Sigma}$ or $\boldsymbol{XV}$
4. Take the **first k columns** of $\boldsymbol{U\Sigma}$ (or $\boldsymbol{XV}$). These are the first k principal components of $\boldsymbol{X}$.

Onto the demo!

The Rank 1 Approximation of X?

$$X = U \quad \Sigma \quad V^T$$

100 x 4 | 100 x 4 | 4 x 4 | 4 x 4

-0.13	0.01	0.03	-0.30
0.09	-0.08	0.01	0.71
...	...	...	...
0.06	-0.06	-.11	-0.07

197.4	0	0	0
0	27.43	0	0
0	0	23.26	0
0	0	0	0

-0.10	-0.07	-0.93	-0.34
0.67	-0.37	0.26	0.59
0.31	-0.64	0.26	-0.65
0.67	0.67	0	-0.33

Q2: What is the Rank 1 Approximation of X?

$$X = U \quad \Sigma \quad V^T$$

100 x 4 | 100 x 4 | 4 x 4 | 4 x 4

-0.13	0.01	0.03	-0.30
0.09	-0.08	0.01	0.71
...	...	...	...
0.06	-0.06	-.11	-0.07

197.4	0	0	0
0	27.43	0	0
0	0	23.26	0
0	0	0	0

-0.10	-0.07	-0.93	-0.34
0.67	-0.37	0.26	0.59
0.31	-0.64	0.26	-0.65
0.67	0.67	0	-0.33

$$\sigma_1 u_1 v_1^T \in \mathbb{R}^{100 \times 4}$$

Rank 1 approximation

Q2. What is the Rank 1 Approximation of X?

$$X = U \quad \Sigma \quad V^T$$

$$\sigma_1 u_1 v_1^T \in \mathbb{R}^{100 \times 4}$$

W	L	A	P
2.97	1.35	24.78	8.64
-3.03	-0.65	-15.22	-7.36
...	...	...	...
-3.03	1.35	-11.22	-3.36

(centered)

$\approx$

W	L	A	P
2.61	1.92	24.61	9.07
-1.68	-1.24	-15.87	-7.98
...	...	...	...
-1.16	-0.86	-11.00	-4.05

Rank 1 approximation

In this case, our Rank 1 approximation is pretty close! Why?

Data Variance and Centering

Capturing Total Variance

(from before)

We define the **total variance** of a data matrix as the sum of variances of attributes.

width	**length**	**area**	**perimeter**
20	20	400	80
16	12	192	56
...	...	...	...
24	12	288	72

Total Variance: **402.56** = 7.69 5.35 50.79 338.73

Variance and Singular Values

We define the **total variance** of a data matrix as the sum of variances of attributes.

width	length	area	perimeter
20	20	400	80
16	12	192	56
...	...	...	...
24	12	288	72
Total Variance: **402.56** =	7.69	5.35	50.79	338.73

Formally, the ith singular value tells us the **component score**, i.e., how much of the variance is captured by the ith principal component. n is # of datapoints.

$$i\text{-th component score} = \frac{(i\text{-th singular value})^2}{n}$$

197.4	0	0	0
0	27.43	0	0
0	0	23.26	0
0	0	0	0

→ $197.39^2/100$ = 389.63

→ $27.43^2/100$ = 7.52

→ $23.26^2 / 100$ = 5.41

Variance and Singular Values

We define the **total variance** of a data matrix as the sum of variances of attributes.

width	length	area	perimeter
20	20	400	80
16	12	192	56
...	...	...	...
24	12	288	72

Total Variance: **402.56** = 7.69 5.35 50.79 338.73

Formally, the ith singular value tells us the **component score**, i.e., how much of the variance is captured by the ith principal component. N is # of datapoints.

$$\text{i-th component score} = \frac{(\text{ith singular value})^2}{N}$$

197.4	0	0	0
0	27.43	0	0
0	0	23.26	0
0	0	0	0

Variance captured by **PC1**

→ $197.39^2/100$ = **389.63**

→ $27.43^2/100$ = 7.52

→ $23.26^2 / 100$ = 5.41

Sum = **402.56**. ✅

PCA needs Data Centering

Recall the steps to obtain Principal Components via SVD:

1. Center the data matrix by subtracting the mean of each attribute column.

1. To find the p **principal component**s:
   a. Compute SVD of $X = U\Sigma V^T$.
   b. The first p columns of $U\Sigma$ (or equivalently, XV) contain the p principal components of X.

- The principal components are a low-dimension representation that capture as much of the original data's total variance as possible.
- The component scores sum to total variance only if we **center our data**.

Why? Proof out of scope, but see the next slide if you're interested.

SVD and Variance: Proof Intuition

(proof out of scope)

$$(X^T X)_{ii} = x_i^T x_i = \sum_{j=1}^{n} x_{ij}^2$$

Define $X = \begin{bmatrix} | & \dots & | & \dots & | \\ x_1 & \vdots & v_i & \vdots & v_d \\ | & \dots & | & \dots & | \end{bmatrix}$

Covariance matrix

$$(\frac{1}{n}\tilde{X}^T\tilde{X})_{ii} = \frac{1}{n}(x_i - \bar{x}_i)^T(x_i - \bar{x}_i)$$

Define $\tilde{X}$ As a centered version of X

$$= \frac{1}{n}\sum_{j=1}^{n}(x_{ij} - \overline{x}_i)^2$$

Variance of feature i

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

Suppose the following SVD of X

$$X^T X = (U\Sigma V^T)^T(U\Sigma V^T)$$

$$= (V\Sigma U^T)(U\Sigma V^T)$$

Transpose properties

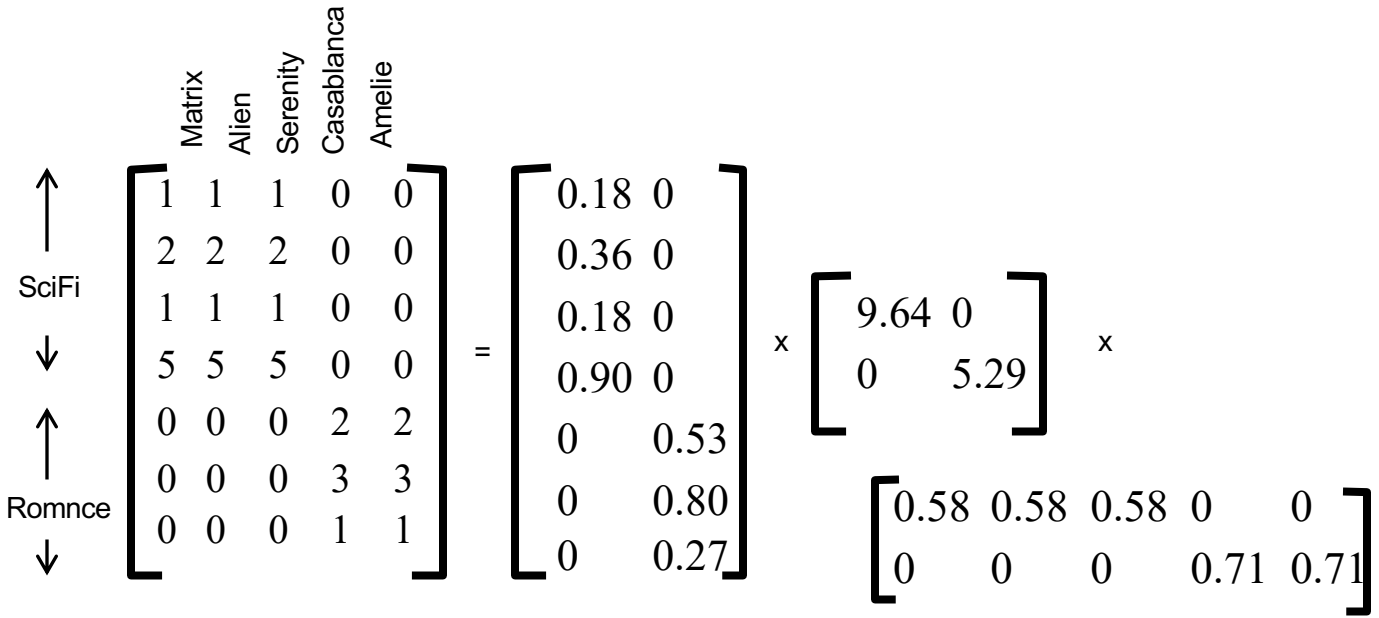
$$= V\Sigma^2 V^T$$

U has orthonormal columns, so $U^T U = I$

[StackExchange]

Case study: How to query?

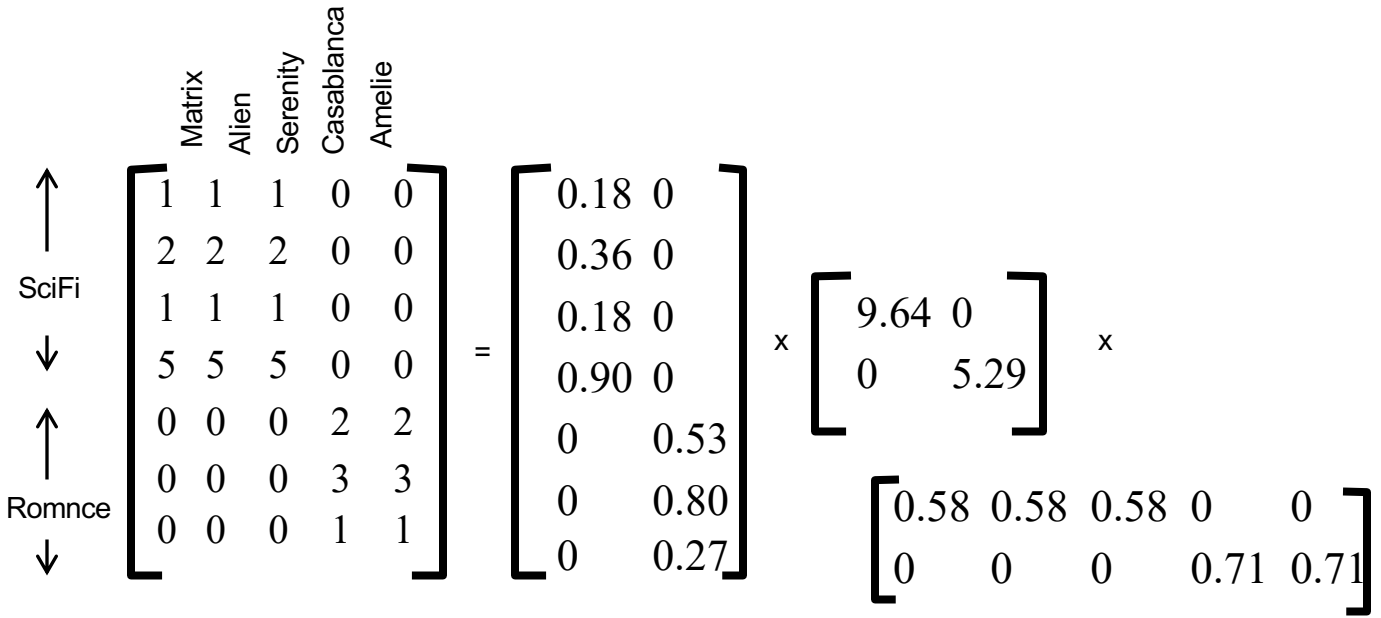
Q: Find users that like ‘Matrix’ and ‘Alien’



Case study: How to query?

Q: Find users that like ‘Matrix’

A: Map query into a ‘concept space’ – how?



Case study: How to query?

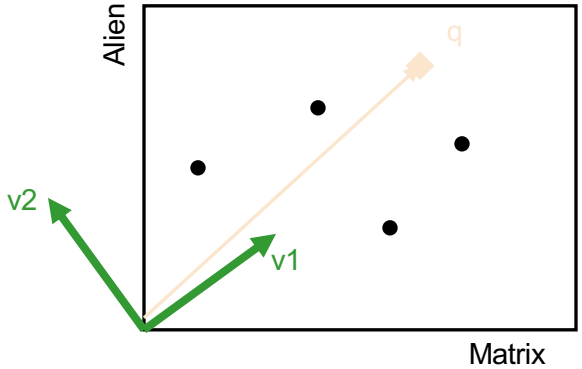
Q: Find users that like ‘Matrix’

A: map query vectors into ‘concept space’ – how?

q=

Matrix	Alien	Serenity	Casablanca	Amelie
5	0	0	0	0

Project into concept space: Inner product with each ‘concept’ vector v_i



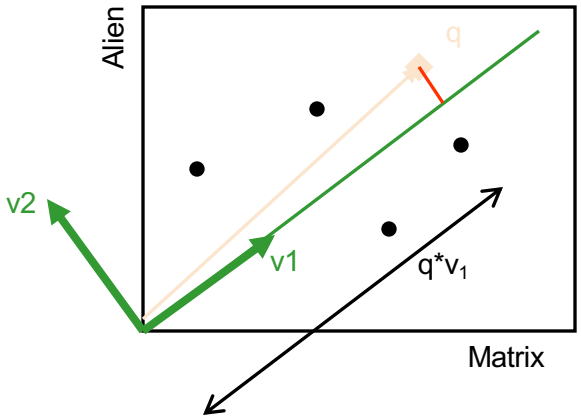
Case study: How to query?

Q: Find users that like 'Matrix'

A: map the vector into 'concept space' – how?

$q =$ [Matrix: 5, Alien: 0, Serenity: 0, Casablanca: 0, Amelie: 0]

Project into concept space:
Inner product with each
'concept' vector v_i

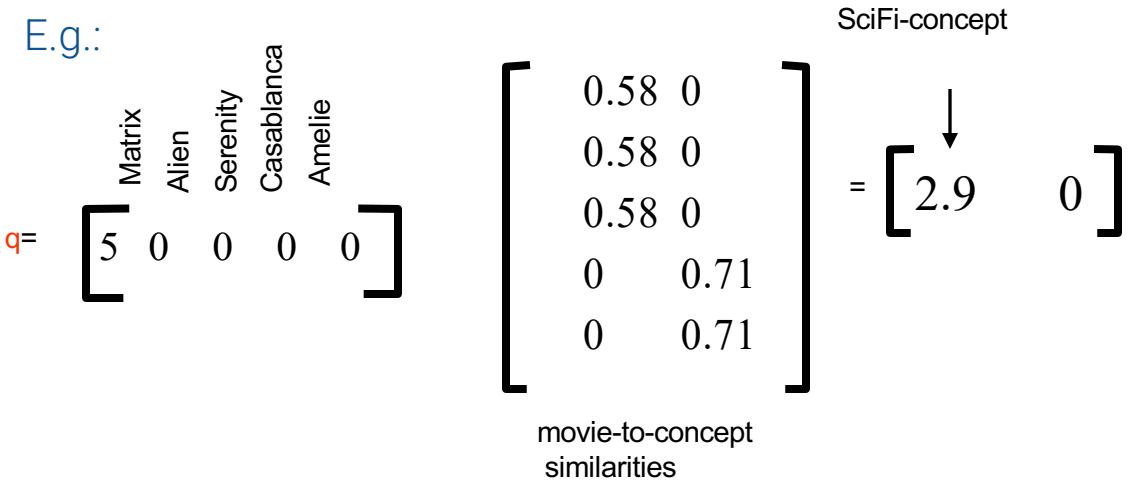


Case study: How to query?

Compactly, we have:

$q_{concept} = q\ V$

E.g.:

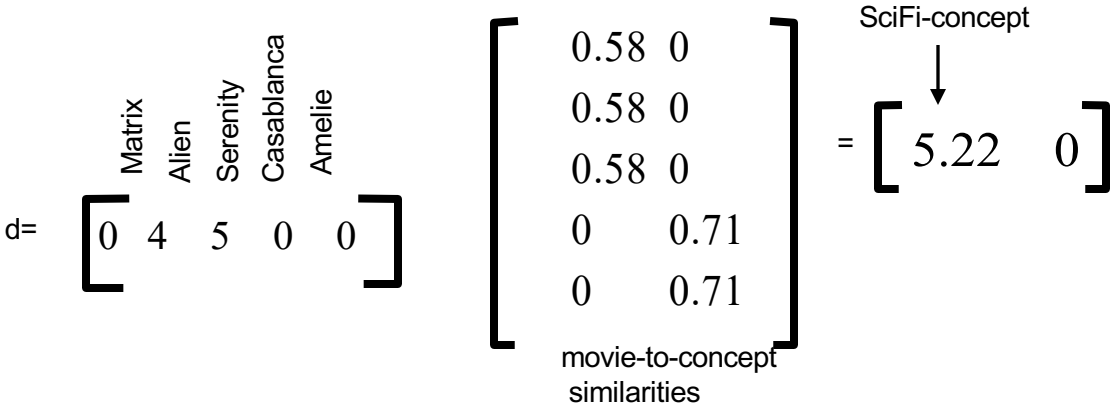


Case study: How to query?

How would the user *d* that rated ('Alien', 'Serenity') be handled?

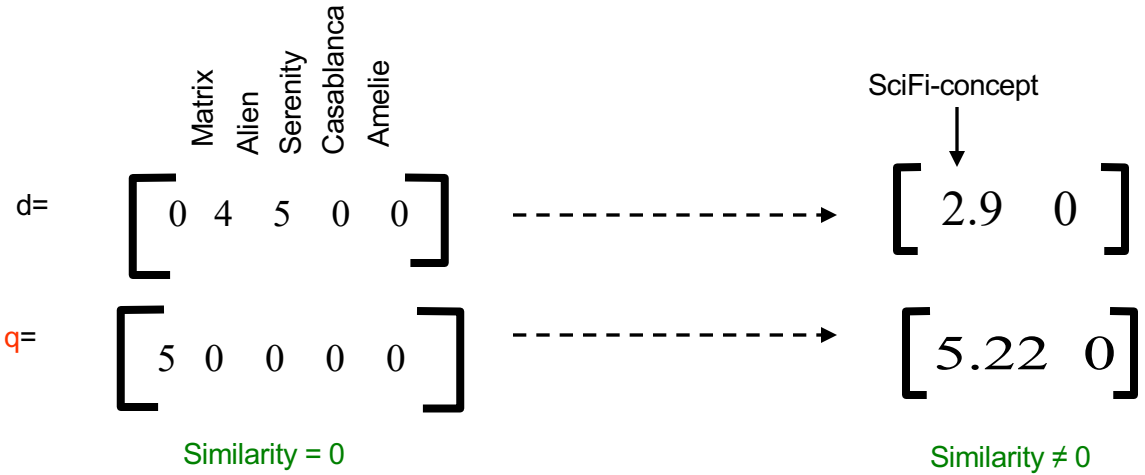
$d_{concept} = d\ V$

E.g.:



Case study: How to query?

Observation: User *d* that rated ('Alien', 'Serenity') will be similar to query "user" q that rated ('Matrix'), although *d* did not rate 'Matrix'!



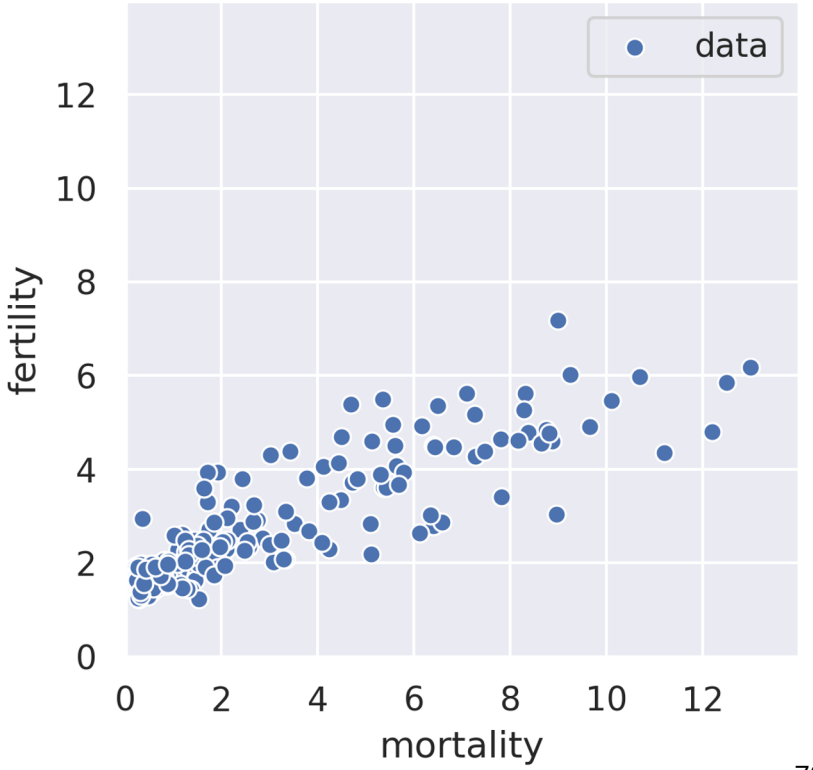
Extra: PCA vs. Regression

- Recap and Goals
- Singular Value Decomposition
- Low Rank Approximation
- Singular Value Decomposition Theory
- Principal Components
- Principal Components and Variance
- Principal Component Analysis Example

Regression: The Big Idea

Suppose we know the child mortality rate of a given country.

- Linear regression tries to predict the fertility rate from the mortality rate.
- For example, if the mortality is 6, we might guess the fertility is near 4.

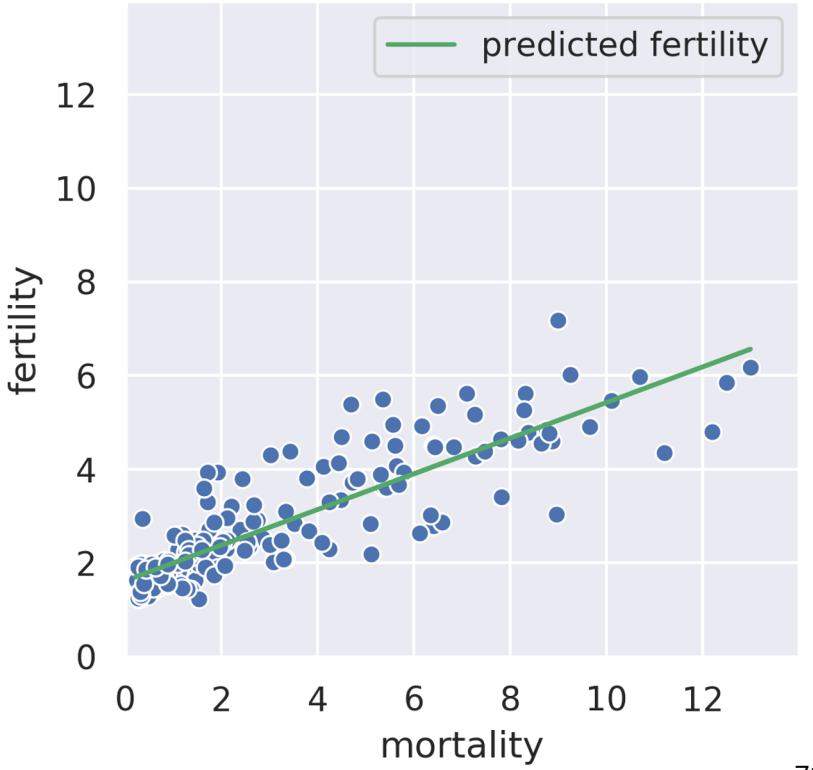


Regression: The Big Idea

Suppose we know the child mortality rate of a given country.

- Linear regression tries to predict the fertility rate from the mortality rate.
- For example, if the mortality is 6, we might guess the fertility is near 4.

The regression line tells us the “best” prediction of fertility given all possible mortality values.



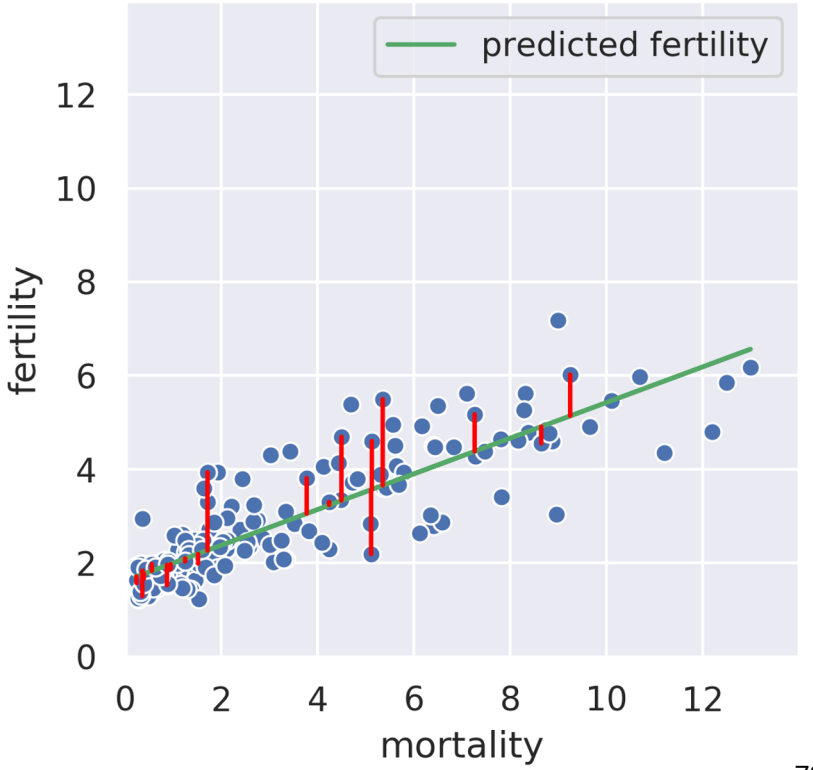
Regression: The Big Idea

Suppose we know the child mortality rate of a given country.

- Linear regression tries to predict the fertility rate from the mortality rate.
- For example, if the mortality is 6, we might guess the fertility is near 4.

The regression line tells us the “best” prediction of fertility given all possible mortality values.

- Minimizes the root mean squared error [see vertical red lines, only some shown].

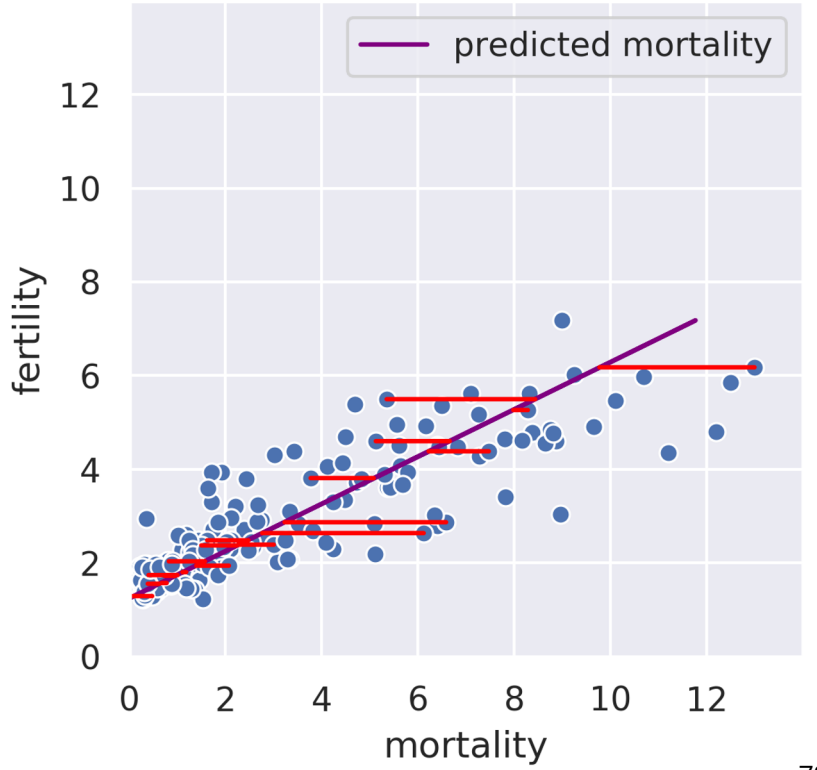


Regression: The Big Idea

We can also perform a regression in the reverse direction.

That is, given the fertility, we try to predict the mortality.

In this case, we get a different regression line which minimizes the root mean squared length of the *horizontal* lines.



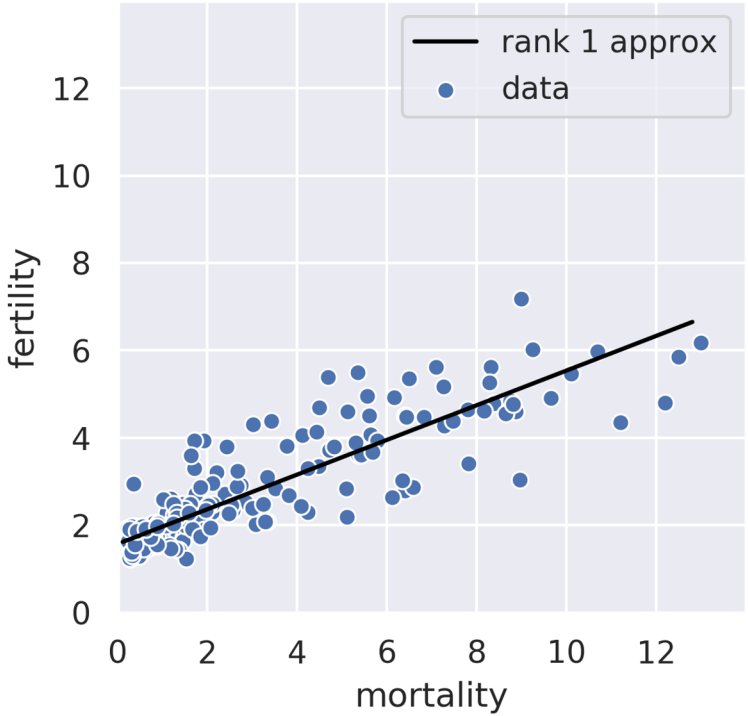
Rank 1 Approximation of Fertility / Mortality Data

Below we see our data and the rank 1 approximation.

Rank 1 Approximation of Fertility / Mortality Data

If we make a line plot instead, this starts to look a lot like a regression line.

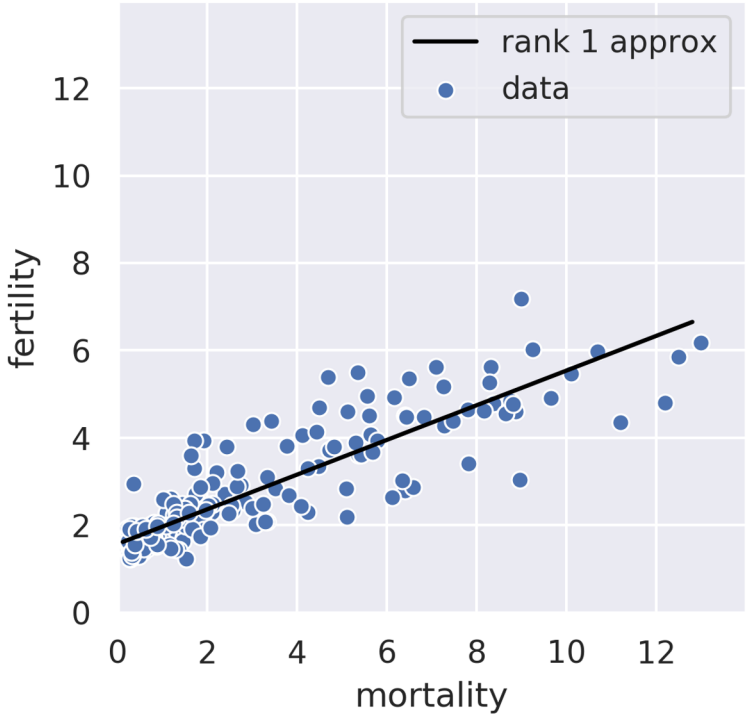
- Note: The approximation is just the data projected on to this line.

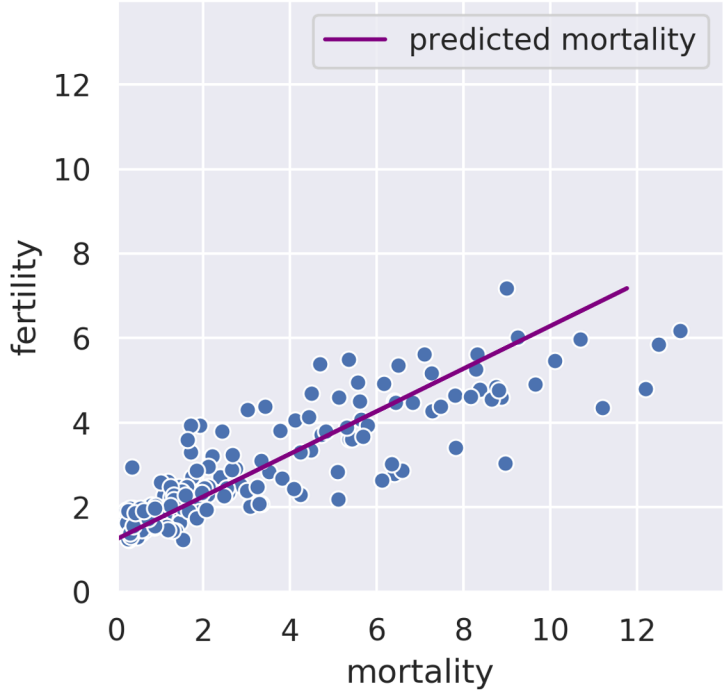


Rank 1 Approximation vs. Predicted Mortality Regression Line

The rank 1 approximation is not the same as the mortality regression line.

- They're vaguely close, but not the same.

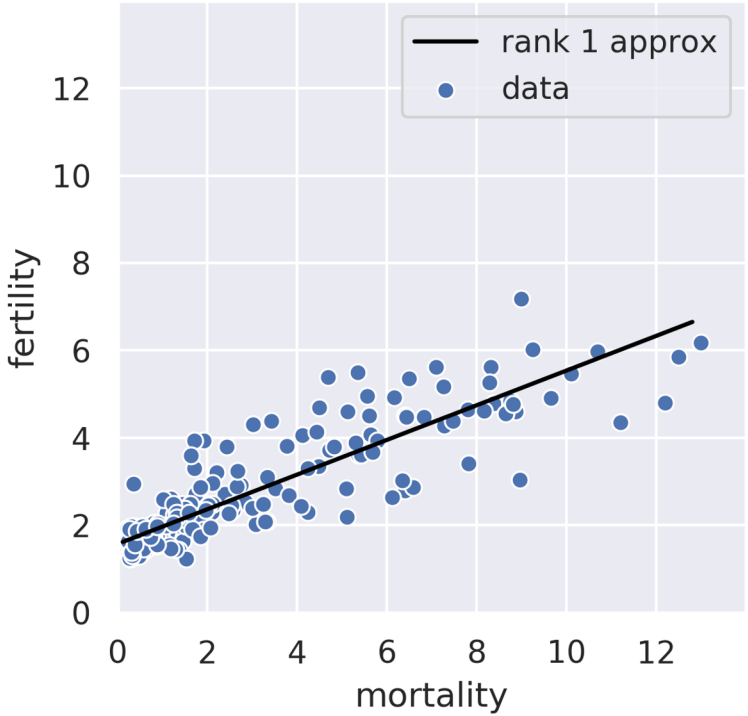


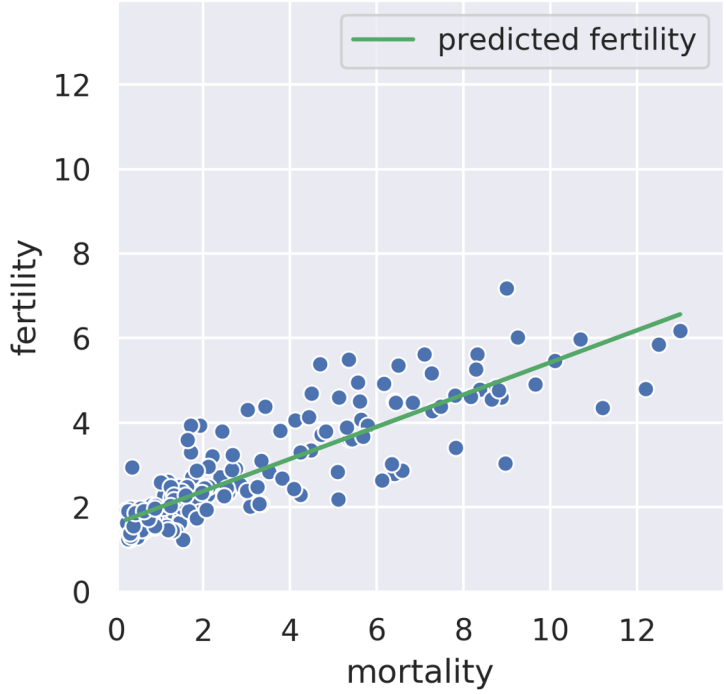


Rank 1 Approximation vs. Predicted Fertility Regression Line

The fertility regression line is pretty close, but is also different.

- The closeness is just a coincidence.

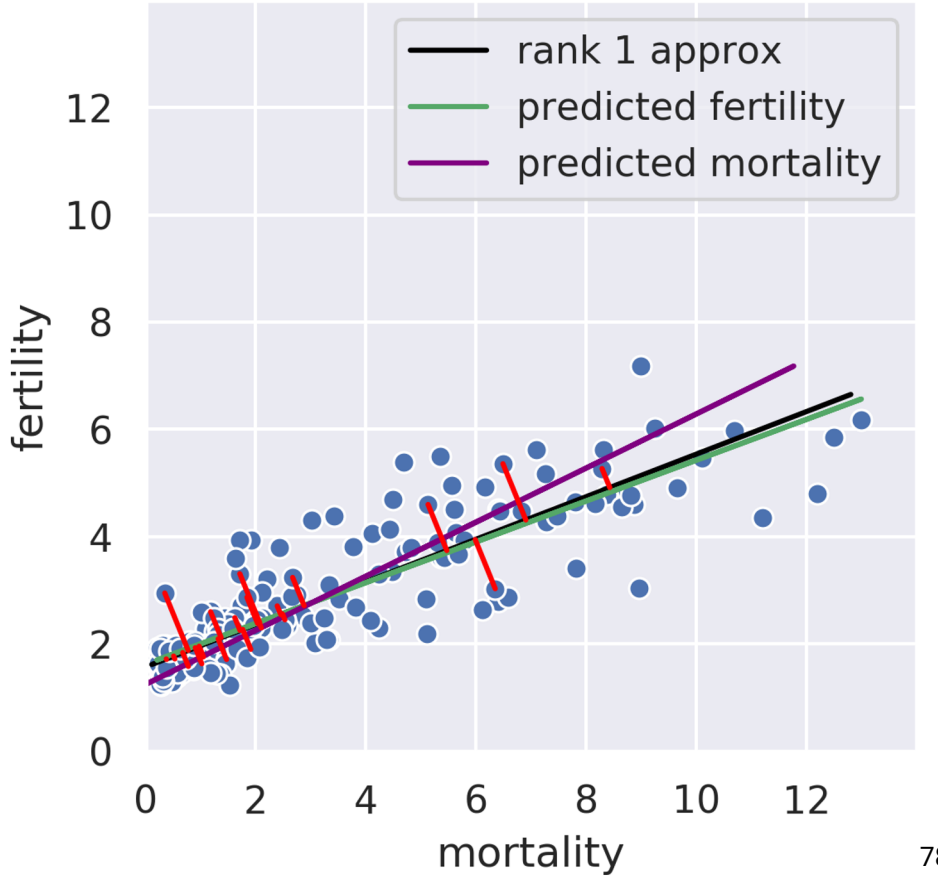




SVD: Minimizing the Perpendicular Error

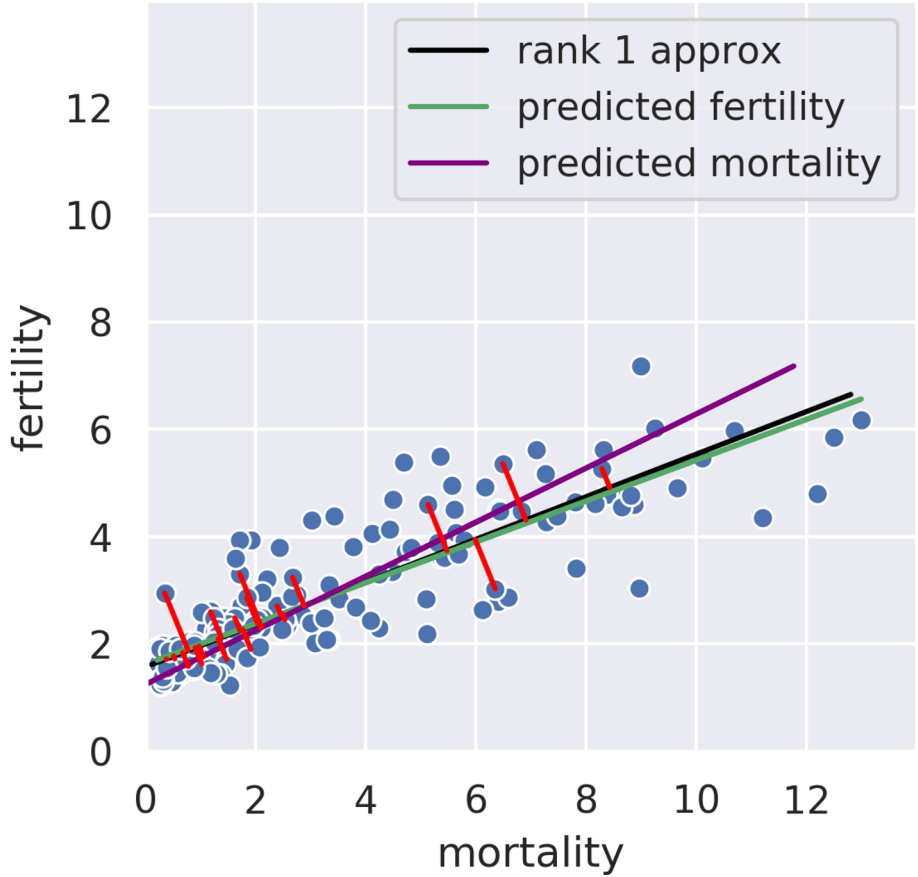
Instead of minimizing *horizontal* or *vertical* error, our rank 1 approximation minimizes the error ***perpendicular*** to the subspace onto which we're projecting.

That is, SVD finds the line such that if we project our data onto that line, the error between the projection and our original data is minimized.



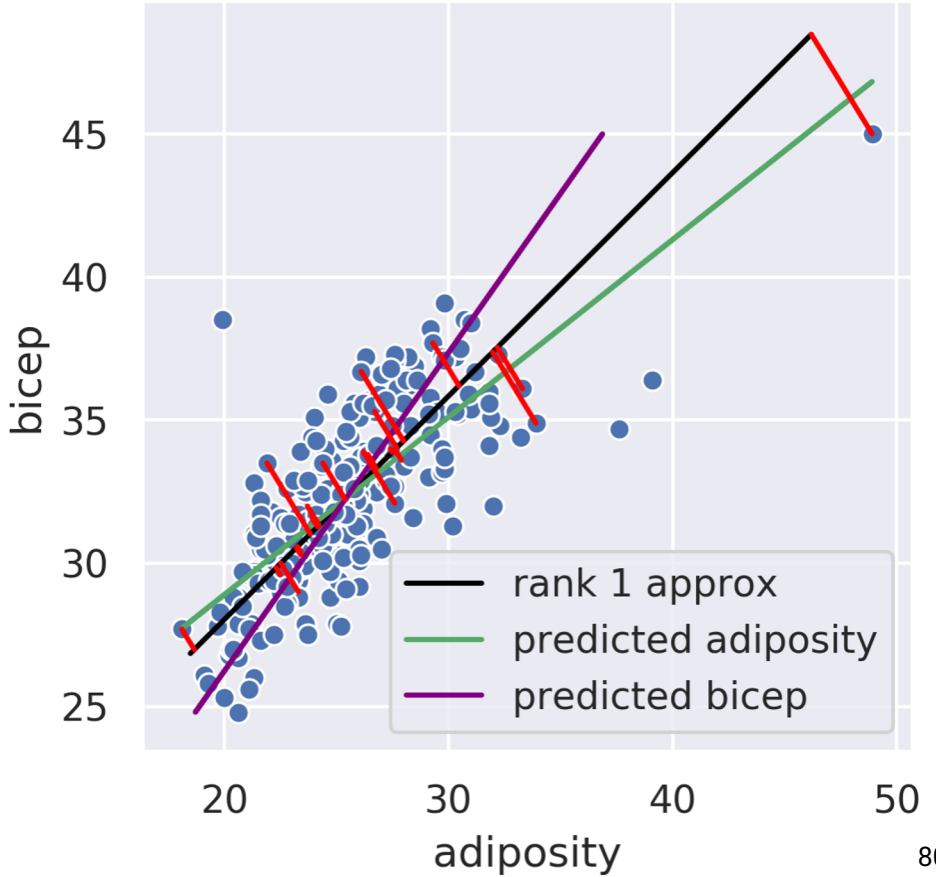
SVD: Minimizing the Perpendicular Error

Reminder: The similarity of the rank 1 approximation and the fertility was just a coincidence.



SVD: Minimizing the Perpendicular Error

Looking at adiposity and bicep size from our body measurements dataset, we see the 1D subspace onto which we are projecting is between the two regression lines.



Beyond 1D/2D

In higher dimensions the idea behind principal components is just the same!

Example: Suppose we have 30 dimensional data and decide to use the first 5 principal components.

- Our procedure minimizes the error between:
  - The original 30 dimensional data.
  - The projection of that 30 dimensional data on to the “best” 5 dimensional subspace.

Principal Component Analysis II

PCA: An alternate technique for EDA and feature generation.

Today's Roadmap

PCA Review

Interpreting PCA

Applications of PCA

PCA Review

[Recap] How to obtain Principal Components

a. Compute SVD on **centered** X:

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

b. Take the first k columns of XV or $U\Sigma$

.

These are the first k **principal components**.

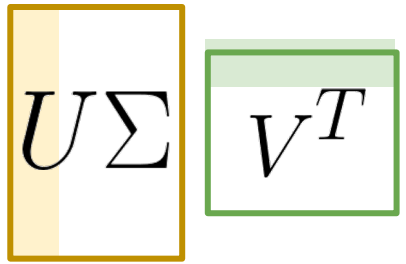
c. If we wanted to get a **rank-k approximation** of X:
(σ_j: j-th singular value, $\mathbf{u}_j$: j-th col of U, v_j: j-th col of V)

This effectively gives us k "features" from our d original features.

For most applications, we directly analyze the principal components themselves.


```
U[:, 0:k] @ np.diag(S[0:k]) @ Vt[0:k, :]
```

$$X_p = \sum_{j=1}^{p} \sigma_j u_j v_j^T$$



[Recap] How much variance is captured

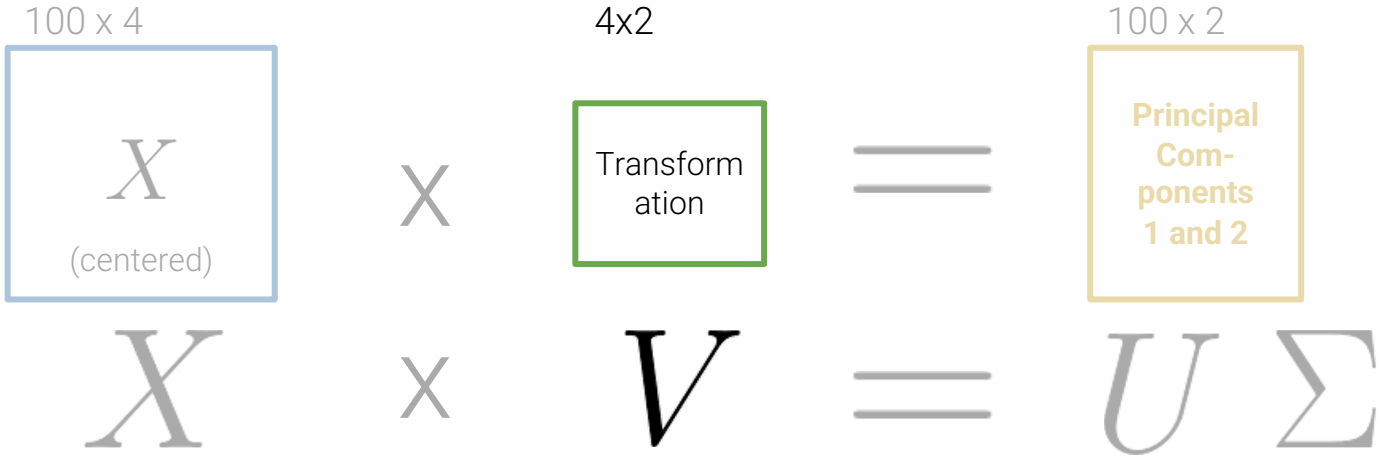
- The principal components are a low-dimension representation that capture as much of the original data's **total variance** as possible.
- **Component scores** tell us how much variance each principal component captures:

$$\text{score}_i = \frac{(\text{singular value}_i)^2}{n}$$

- The component scores sum to total variance if we **center our data**.

PCA with SVD

To get the first **two principal components**, first find the SVD $X = U\Sigma V^T$.

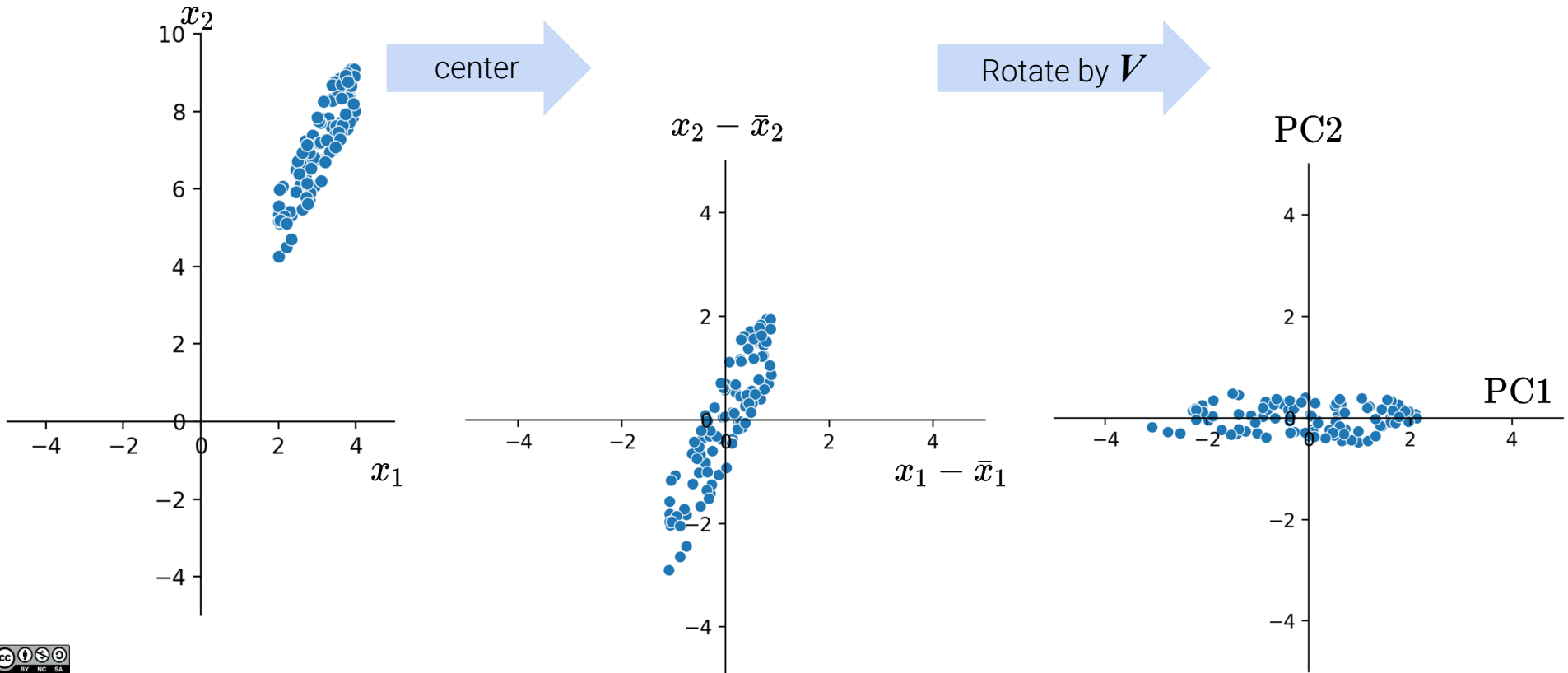


Take the first 2 **columns of UΣ** (or XV). These are PC1 and PC2.

The first n **rows of V^T** are **directions** for the n principal components.

How PCA transforms data, visually

PCA first centers the data matrix, then rotates it such that the direction with the most variation (i.e. the direction that's the most spread-out) is aligned with the x-axis.



Columns of V are the directions

$$XV = \begin{bmatrix} | & & | & & | \\ \vdots & & \vdots & & \vdots \\ x_1 & \dots & x_j & \dots & x_d \\ \vdots & & \vdots & & \vdots \\ | & & | & & | \end{bmatrix} \begin{bmatrix} v_{11} & & | & & | \\ \vdots & & & & \\ v_{j1} & \dots & v_k & \dots & v_d \\ \vdots & & & & \\ v_{d1} & & | & & | \end{bmatrix}$$

$X \in \mathbb{R}^{n \times d}$

$V \in \mathbb{R}^{d \times d}$

transformation

$$v_{11} \cdot \begin{bmatrix} | \\ \vdots \\ x_1 \\ \vdots \\ | \end{bmatrix} + \cdots + v_{j1} \cdot \begin{bmatrix} | \\ \vdots \\ x_j \\ \vdots \\ | \end{bmatrix} + \cdots + v_{d1} \cdot \begin{bmatrix} | \\ \vdots \\ x_d \\ \vdots \\ | \end{bmatrix} = \text{PC1}$$

scalar 1

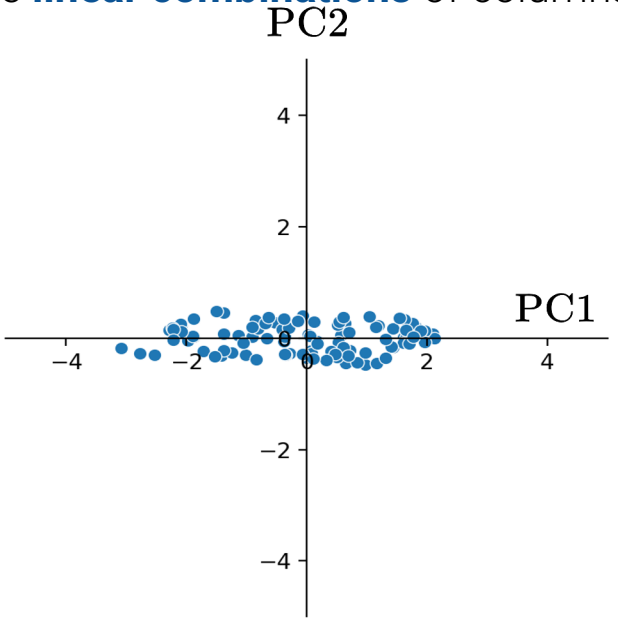
feature 1

The elements of each **column of V (row of V^T)** **rotate** the original feature vectors into a principal component.

The first column of V indicates **how each feature contributes** (e.g., positively, negatively, etc.) to PC1.

Principal Components

- Principal components are all **orthogonal** to each other
  - Why? Recall that columns of U are orthonormal!
- Principal Components are **axis-aligned**
  - If we plot two PCs on a 2D plane, one will lie on the x-axis, the other on the y-axis
- Principal Components are **linear combinations** of columns in our data $\boldsymbol{X}$



Interpreting PCA

[From last time] Why perform PCA?

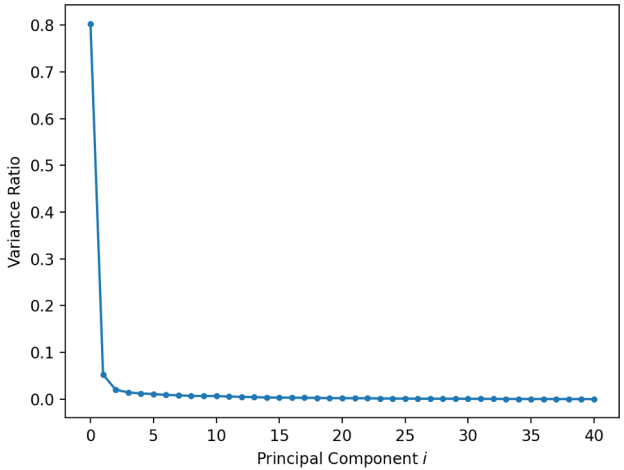
We often perform PCA during the Exploratory Data Analysis (EDA) stage of our data science lifecycle (if we already know what to model, we probably don't need PCA); it helps us with

- Visually identifying **clusters** of similar observations in high dimensions.
- Removing irrelevant dimensions if we suspect that the dataset is inherently low rank. For example, if the columns are **collinear**: there are many attributes but only a few mostly determine the rest through linear associations.
- Finding a small basis for representing variations in complex things, e.g., images, genes.
- Reducing the number of dimensions to make some computation cheaper.

Scree Plot

If the first two singular values are large and all others are small, then **two dimensions are enough** to describe most of what distinguishes one observation from another. If not, then a PCA scatter plot is omitting lots of information.

A **scree plot** shows the variance ratio captured by each principal component, largest first.



Scree [wikipedia]

Biplot

$$v_{11} \cdot \begin{bmatrix} | \\ \vdots \\ x_1 \\ \vdots \\ | \end{bmatrix} + \cdots + v_{j1} \cdot \begin{bmatrix} | \\ \vdots \\ x_j \\ \vdots \\ | \end{bmatrix} + \cdots + v_{d1} \cdot \begin{bmatrix} | \\ \vdots \\ x_d \\ \vdots \\ | \end{bmatrix} = \text{PC1}$$

scalar 1

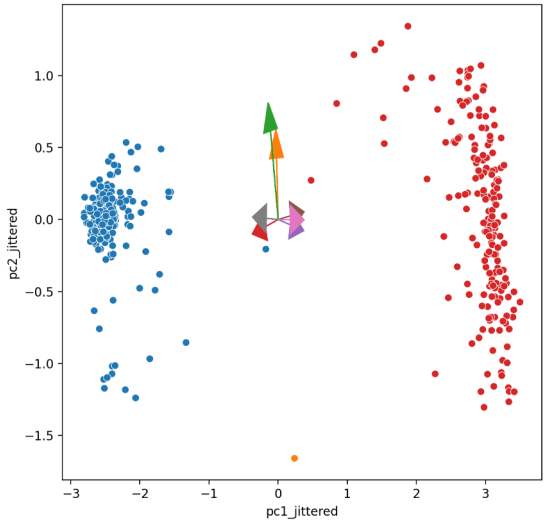
The i-th column of $\boldsymbol{V}$ indicates **how each feature contributes** to PC i.

Biplots superimpose the **directions** onto the plot of PC1 vs. PC2.

Vector j corresponds to the direction for feature j, e.g., $(\mathbf{v}_{1j}, \mathbf{v}_{2j})$.

- There are several ways to scale biplots vectors; in this course we plot the direction itself.
- For other scalings, which can lead to more interpretable directions/loadings, see [SAS biplots].

Through biplots, we can interpret how features correlate with the principal components shown: positively, negatively, or not much at all.

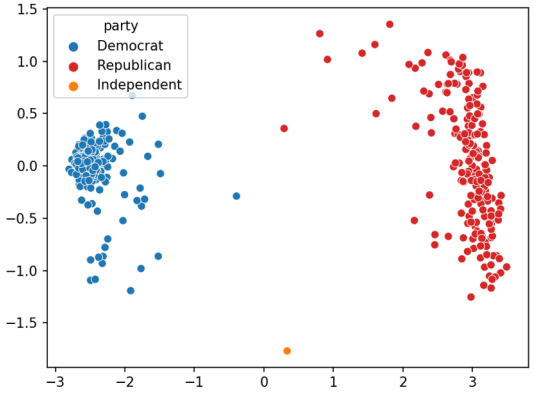


Summary: Plots based on PCA

PCA Plot

Scatter plot of PC1 against PC2

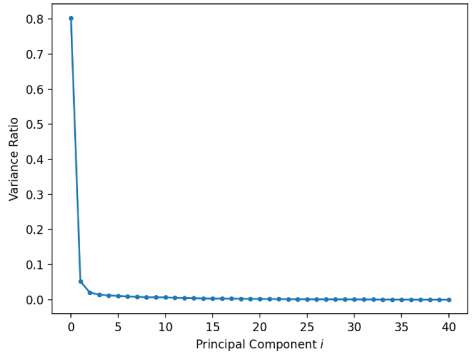
- Helps us assess similarities between our data points and if there are any clusters in our dataset.



Scree Plot

Line plot showing the **variance ratio** captured by each principal component, largest first.

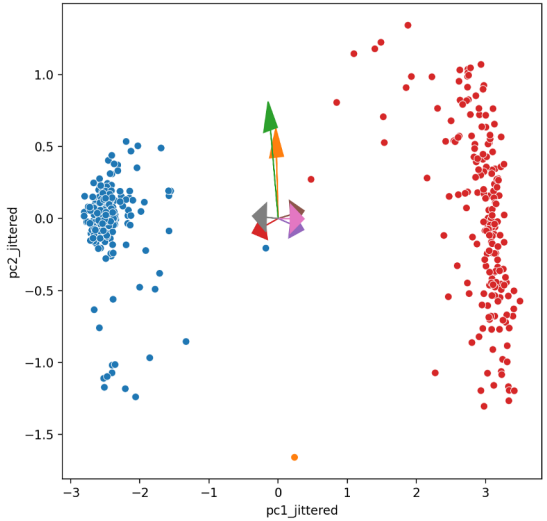
- If first two is large enough, we know PCA plot is good representation of data
- "Elbow" method to assess how many PCs to use



Biplot

PCA plot + **directions** of feature importance for PC1 and PC2

- All benefits from PCA plot, and
- Shows how much some features contribute to PC1/2

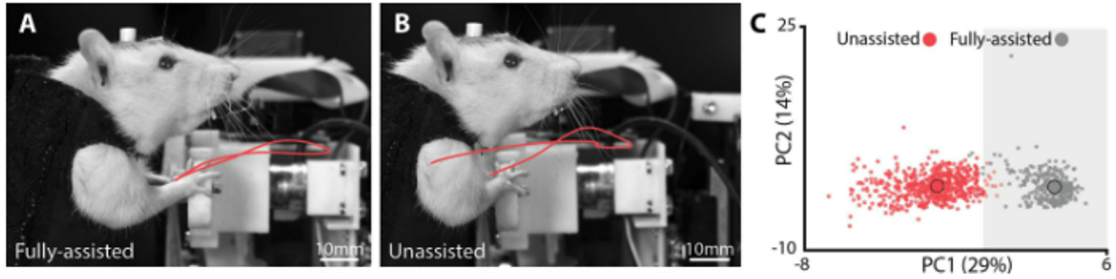


Applications of PCA

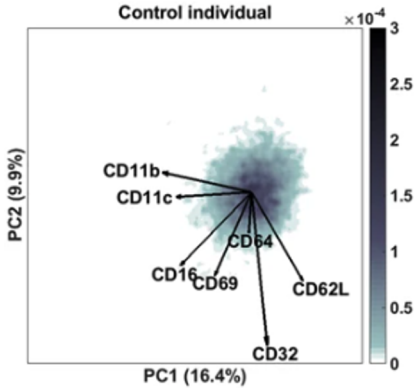
PCA in Biology

PCA is commonly used in biomedical contexts, which have many named variables!

1. To cluster data (Paper 1, Paper 2)



2. To identify correlated variables (interpret rows of V^T as linear coefficients) (Paper 3). Uses biplots.



Why perform PCA?

We often perform PCA during the **Exploratory Data Analysis** (EDA) stage of our data science lifecycle (if we already know what to model, we probably don't need PCA); it helps us with

- Visually identifying clusters of similar observations in high dimensions.
- Removing irrelevant dimensions if we suspect that the dataset is inherently low rank. For example, if the columns are collinear: there are many attributes but only a few mostly determine the rest through linear associations.
- Finding a small basis for representing variations in complex things, e.g., images, genes.
- Reducing the number of dimensions to make some computation cheaper.

Image Classification

Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. Han Xiao, Kashif Rasul, Roland Vollgraf. arXiv:1708.07747
https://github.com/zalandoresearch/fashion-mnist

Summary

In **machine learning**, **PCA** is often used as a **preprocessing step** prior to training a supervised model..

Lecture 23

Principal Component Analysis II

Content credit: Acknowledgments