slido

Join at slido.com
#3844650

ⓘ Click **Present with Slido** or install our Chrome extension to display joining instructions for participants while presenting.

3844650

3844650

LECTURE 24

Principal Component Analysis I

A Dimensionality Reduction Technique for EDA

Data 100/Data 200, Spring 2025 @ UC Berkeley

Narges Norouzi and Josh Grossman

Content credit: Acknowledgments

Announcements

3844650

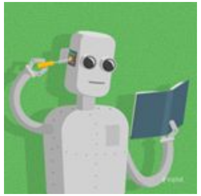
Remember that we hold catch-up hour **Fridays 3-5pm @ Haviland 12**

Format: 1/3 is a review lecture + 2/3 student questions

We're excited to see you there!

Taxonomy of Machine Learning

3844650



Labeled Data

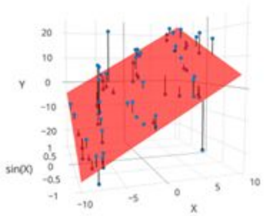
Supervised Learning

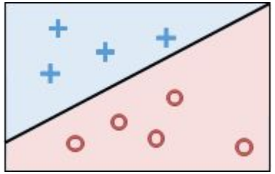
Quantitative Response

Categorical Response

Regression

Classification





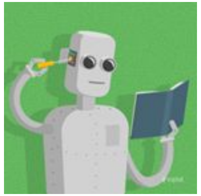
Data 8: k-Nearest Neighbors
Earlier: Logistic Regression

"Supervised Learning": Create a function that maps inputs to outputs.

- Model is trained on example input and output **pairs.** Each pair consists of:
  - Input vector **(features)**
  - Output value **(label)**.
- **Regression**: Output value is quantitative.
- **Classification**: Output value is categorical.

Taxonomy of Machine Learning

3844650



Labeled Data

Unlabeled Data

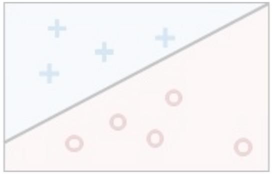
Supervised Learning

Quantitative Response

Categorical Response

Regression

Classification



Data 8: k-Nearest Neighbors
Earlier: Logistic Regression

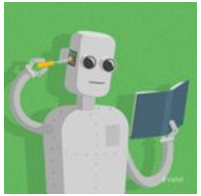
Unsupervised Learning

“Unsupervised Learning”: Identify patterns in **unlabeled** data.

- We have **features** but **no labels**
    - Sometimes we may have labels, but we choose to ignore them.

Taxonomy of Machine Learning

3844650



Labeled Data

Unlabeled Data

Supervised Learning

Unsupervised Learning

Quantitative Response

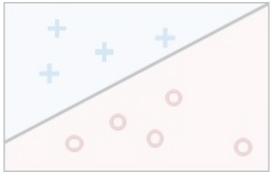
Categorical Response

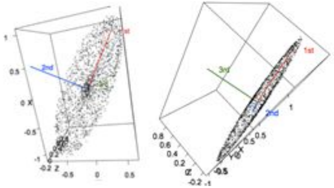
Regression

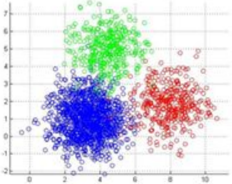
Classification

Dimensionality Reduction

Clustering





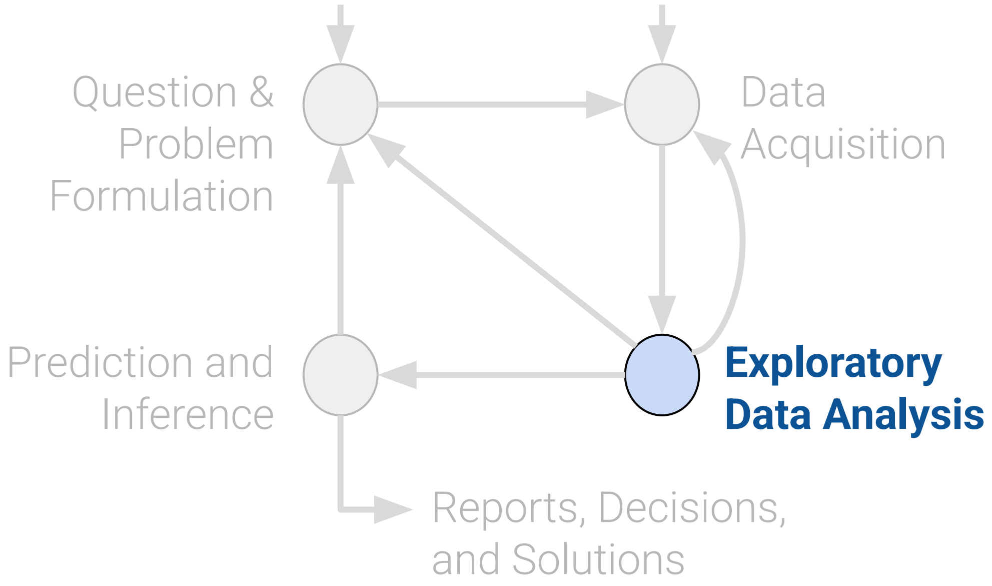


Data 8: k-Nearest Neighbors
Earlier: Logistic Regression

Today with **Principal Component Analysis (PCA)**

PCA: A Technique for High Dimensional EDA and Featurization

3844650



PCA I

today

PCA II
SVD
Applications

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

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

3844650

Today's Roadmap

Lecture 24, Data 100 Spring 2025

Visualization Revisited

Lecture 24, Data 100 Spring 2025

slido

3844650

How many dimensions can you visualize on a 2-dimensional screen?

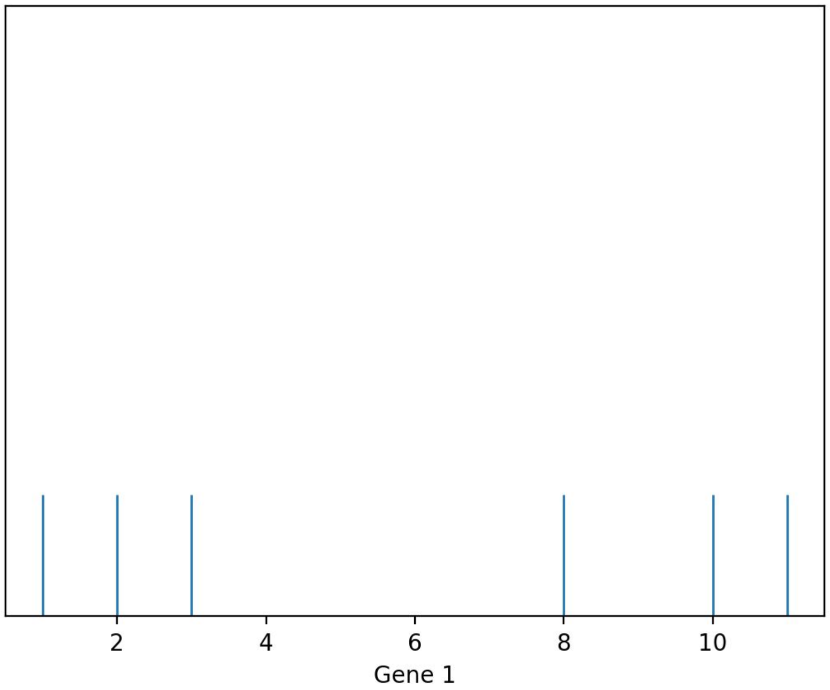
Click **Present with Slido** or install our Chrome extension to activate this poll while presenting.

Visualizing Gene Data

3844650

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

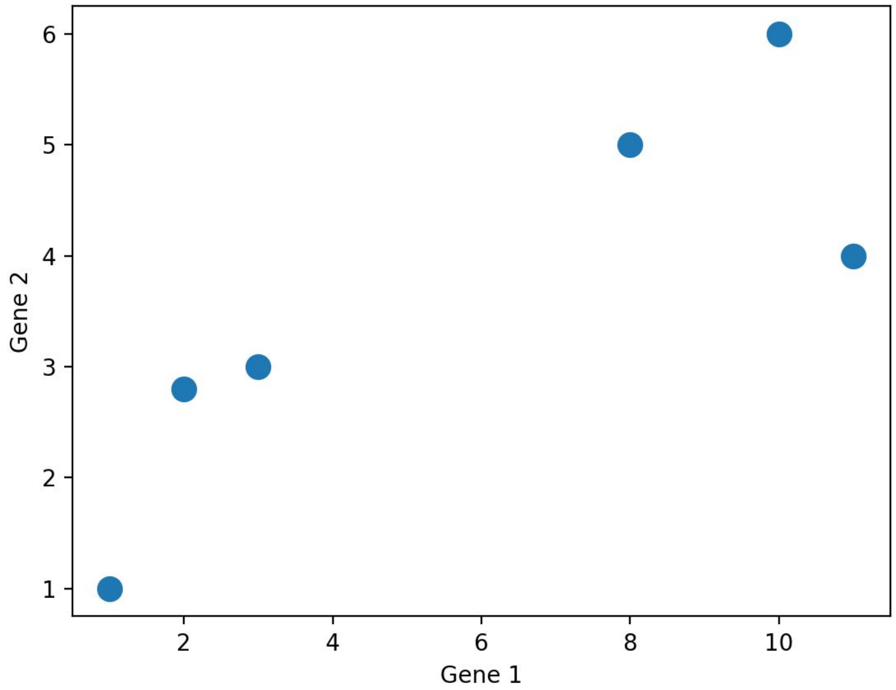


Visualizing Gene Data

3844650

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

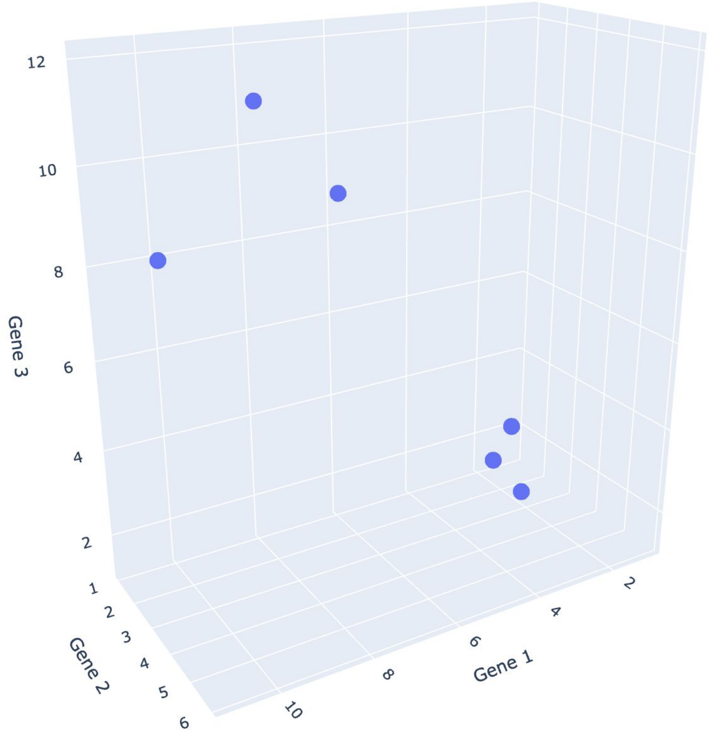


Visualizing Gene Data

Visualization can help us identify clusters in our dataset.

3844650

	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



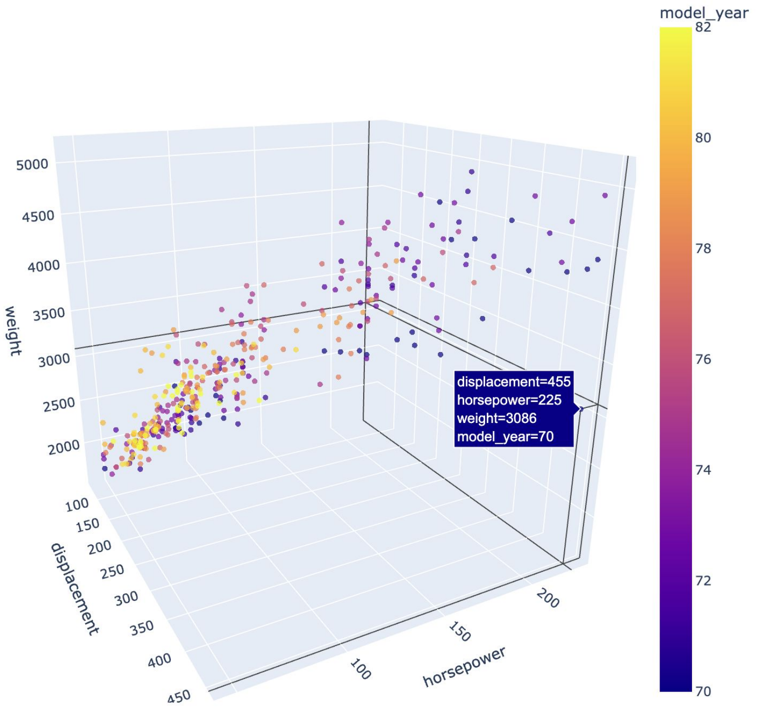
Visualizing Gene Data

3844650

Since we are 3D beings, we can't see 4D or higher. 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

Demo



3844650

Dimensionality

Lecture 24, Data 100 Spring 2025

3844650

Intrinsic Dimension of Data

3844650

Suppose we have a dataset of:

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

In Linear Algebra:

N points/row vectors in a **d**-dimension space, OR

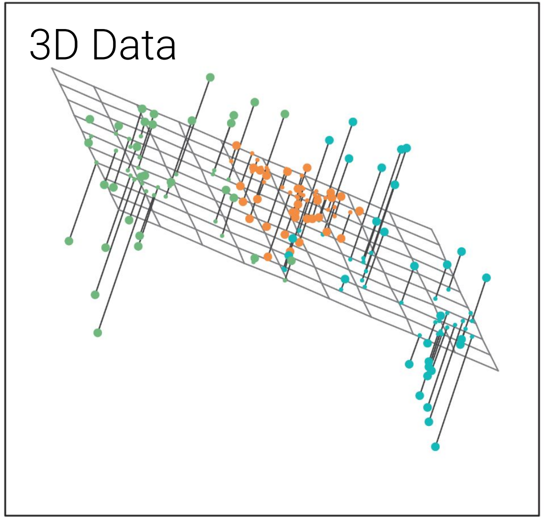
d column vectors in an **N**-dimension space

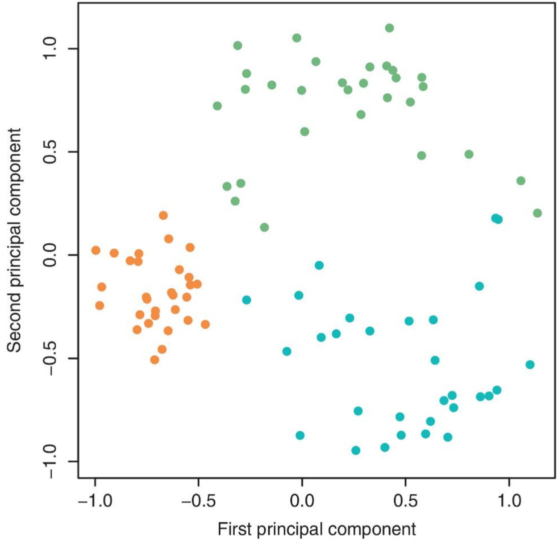
Intrinsic dimension of a dataset is the **minimal** set of dimensions needed to approximately represent the data.

Example:

- 3D dataset →
- Mostly describe by position on the 2D-plane.

Intrinsic Dimension ≈ 2





Dimensionality of the Column Space

3844650

Suppose we have a dataset of:

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

In Linear Algebra:

N points/row vectors in a **d**-dimension space, OR

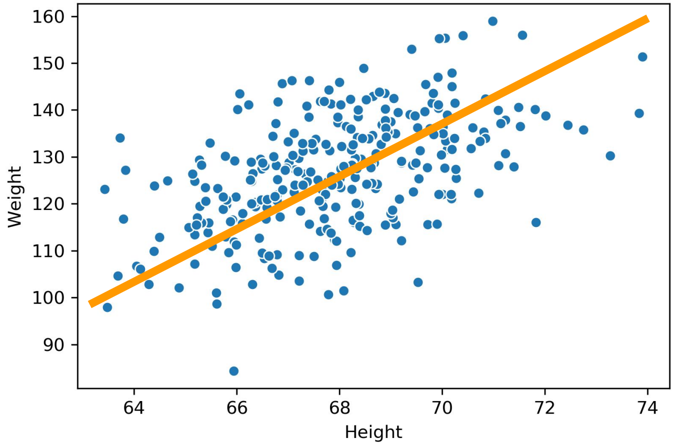
d column vectors in an **N**-dimension space.

Intrinsic dimension of a dataset is the **minimal** set of dimensions needed to approximately represent the data.

Example:

- "Somewhat" described by position on the 1D-plane (line)

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



Dimensionality of the Column Space

3844650

Suppose we have a dataset of:

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

In Linear Algebra:

N points/**row vectors** in a **d**-dimension space, OR

d column vectors in an **N**-dimension space.

Dimension of the column space of A is the **rank** of matrix A.

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

2 dimensions

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

3 dimensions

Dimensionality of the Column Space of Data?

3844650

Consider the datasets shown.

The column space of each of these datasets is:

A. 1-dimensional,
B. 2-dimensional,
C. 3-dimensional
D. >3 dimensional

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

3844650

What would you call these datasets?

ⓘ Click **Present with Slido** or install our [Chrome extension] to activate this poll while presenting.

Dimensionality of Data?

3844650

Consider the datasets shown.

The column space of each of these datasets is:

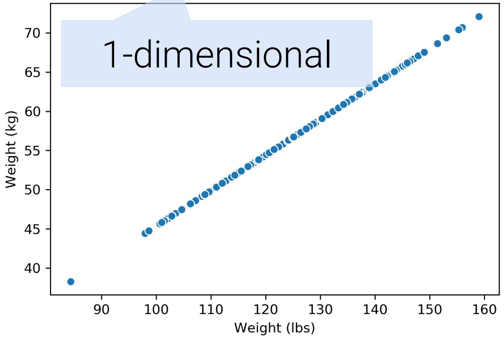
A. 1-dimensional,
B. 2-dimensional,
C. 3-dimensional
D. >3 dimensional

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?

3844650

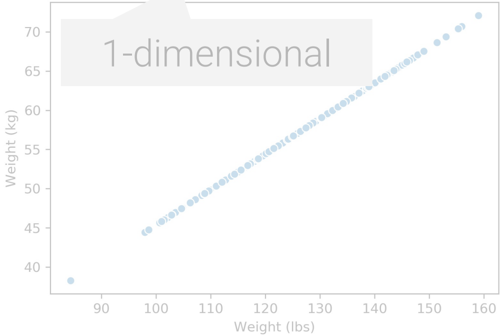
Consider the datasets shown.

The column space of each of these datasets is:

A. 1-dimensional,
B. 2-dimensional,
C. 3-dimensional
D. >3 dimensional

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.

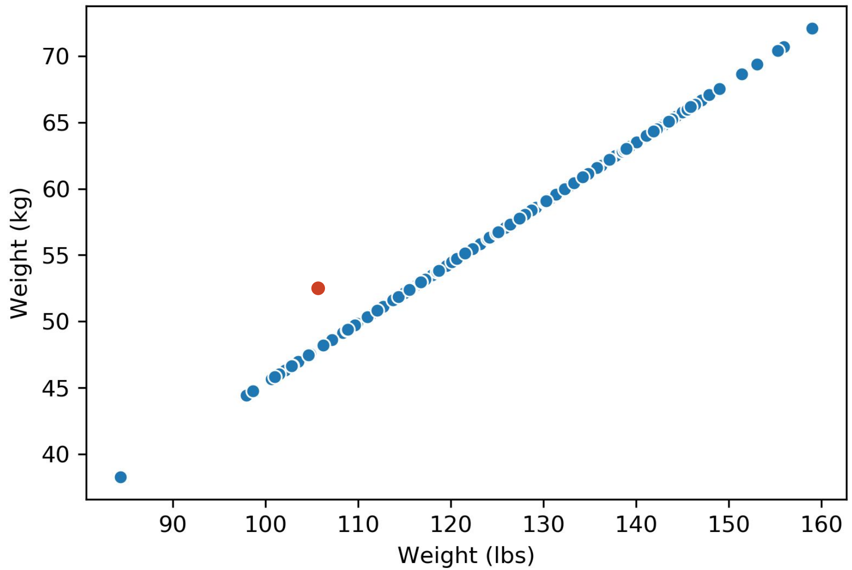
Matrix representation of this dataset has (column) **rank** 3.

Dimensionality - what does it mean…?

3844650

Note that in the dataset below, we'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 is still ***approximately*** **1-dimensional**!



Intrinsic dimension of a dataset is the **minimal** set of dimensions needed to approximately represent the data.

Dimensionality reduction is generally an **approximation** of the original data. This is often achieved through **matrix factorization**.

Matrix Decomposition (Factorization)

Lecture 24, Data 100 Spring 2025

Dimensionality Reduction as Matrix Factorization

3844650

Original Dataset

Age (days)	Height (in)	Height (ft)
182	28	2.33
399	30	2.5
725	33	2.75
630	31	2.58
124	24	2

5 x 3

≈

Reduced Dimension Dataset

Age (days)	Height (in)
182	28
399	30
725	33
630	31
124	24

5 x 2

*

2 x 3

	?	

One **linear** technique for dimensionality reduction is via **matrix decomposition**, which is closely tied to **matrix multiplication**.

Dimensionality Reduction as Matrix Factorization

3844650

Original Dataset

Reduced Dimension Dataset

$$\underset{n \times d}{\mathbf{X}} \approx \underset{n \times k}{\mathbf{Z}} * \underset{k \times d}{\mathbf{W}}$$

Today we will unpack a method to
(a) **decompose** our data matrix **X** into a
(b) **lower dimension** matrix **Z** that,
(c) when multiplied by **W,** approximately
recovers the original data.

Interpreting Matrix multiplication

3844650

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 for an apple, \$1 for a lemon, \$4 for a melon.
- **Output** is the cost of each bowl at each store.

Number of Fruits

	# Apples	# Lemons	# Melons
Bowl 1	**2**	**2**	**2**
Bowl 2	**5**	**8**	**0**

X

Fruit Costs

	Store 1	Store 2
\$/Apple	**2**	**1**
\$/Lemon	**1**	**1**
\$/Melon	**4**	**1**

`matmul`

=

	Store 1	Store 2
Bowl 1	**14**	**6**
Bowl 2	**18**	**13**

Two ways to **interpret** matrix multiplication:
1. Linear operations per datapoint.
2. Column transformation. (Useful today!)

3844650

Multiplication View 1/2: Right matrix is Linear Operations

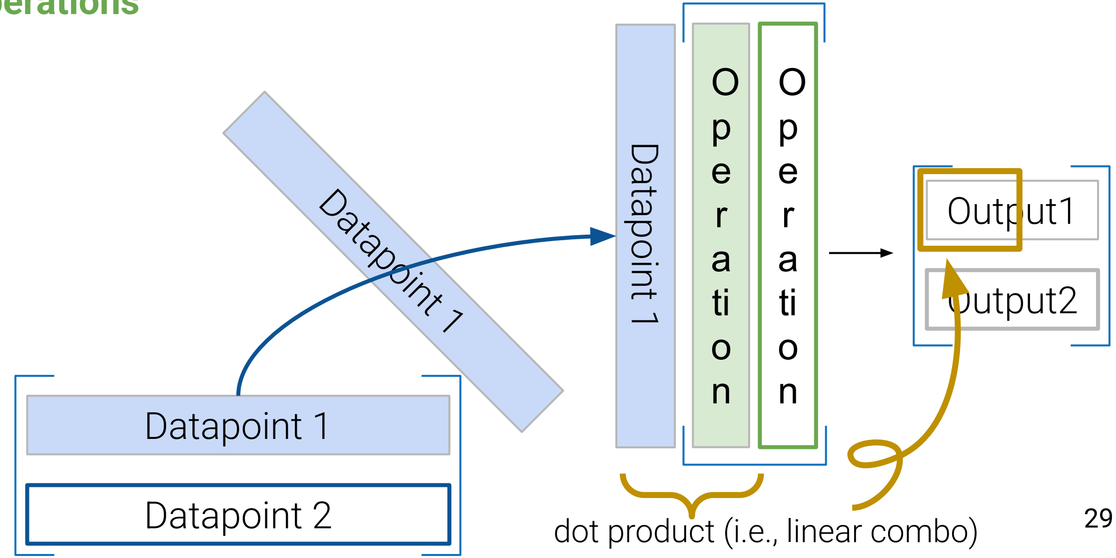
$$\begin{bmatrix} 2 & 2 & 2 \\ 5 & 8 & 0 \end{bmatrix} \times \begin{bmatrix} 2 & 1 \\ 1 & 1 \\ 4 & 1 \end{bmatrix} = \begin{bmatrix} 14 & 6 \\ 18 & 13 \end{bmatrix}$$

data

operations

View 1: Perform multiple linear operations on data.

- This is how we learn matrix multiplication.
- We use this view when building linear models.



Multiplication View 2/2: Right Matrix Transforms Features

3844650

2	2	2
5	8	0

Original columns

X

2	1
1	1
4	1

Transformation

=

14	6
18	13

New column 1

View 1: Perform multiple linear operations on data.

- We use this view when building linear models.

View 2: Multiplication is a **column transformation**. Don't think about rows.

"Recipe" to make our new column 1:

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

"2 parts col 1" "1 part col 2" "4 parts col 3"

[3blue1brown's Essence of Linear Algebra]

OLS: To get $\hat{Y}$ vector, θ_1 parts feature 1 + θ_2 parts feature 2 + …

Matrix Decomposition as a Means of Dimensionality Reduction

3844650

`matmul`

Data with reduced dimensions

*

transformation

??

=

Age (days)	Height (in)	Height (ft)
182	28	2.33
399	30	2.5
725	33	2.75

Goal **Transformation**

Matrix Decomposition (Matrix Factorization)

3844650

Matrix decomposition (a.k.a. Matrix **Factorization**) is the opposite of matrix multiplication, i.e. taking a matrix and decomposing it into two separate matrices.

- Just like with real numbers, there are **infinitely** many such decompositions.
  - 9.9 = 1.1 * 9 = 3.3 * 3.3 = 1 * 9.9 = …
- The matrix sizes aren't even unique…

Some example factorizations:

$$\underset{3\times 2}{\begin{array}{cc} \text{Age} & \text{Height (in)} \\ \hline 182 & 28 \\ 399 & 30 \\ 725 & 33 \end{array}} * \underset{2\times 3}{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1/12 \end{bmatrix}}$$

New col 1: 1 part Age
New col 2: 1 part Height (in)
New col 3: 1/12 part Height (in)

Age (days)	Height (in)	Height (ft)
182	28	2.33
399	30	2.5
725	33	2.75

Matrix Decomposition: Infinite Ways

3844650

Matrix decomposition (a.k.a. Matrix **Factorization**) is the opposite of matrix multiplication, i.e. taking a matrix and decomposing it into two separate matrices.

- Just like with real numbers, there are **infinitely** many such decompositions.
  - 9.9 = 1.1 * 9 = 3.3 * 3.3 = 1 * 9.9 = …
- The matrix sizes aren't even unique…

3x2

182	28
399	30
725	33

*

1	0	0
0	1	1/12

2x3

3x3

182	28	2.33
399	30	2.5
725	33	2.75

*

1	0	0
0	1	0
0	0	1

3x3

Age (days)	Height (in)	Height (ft)
182	28	2.33
399	30	2.5
725	33	2.75

Dimensions of possible matrix factorizations? Select all that apply.

A. (3x2) * (2x3)
B. (3x3) * (3x3)
C. (3x1) * (1x3)
D. (3x4) * (4x3)
E. Inner dimensions higher than 4

slido

3844650

What are possible matrix factorizations? Select all that apply.

ⓘ Click **Present with Slido** or install our Chrome extension to activate this poll while presenting.

Matrix Decomposition: Infinite Ways

3844650

3x2

182	28
399	30
725	33

*

1	0	0
0	1	1/12

2x3

3x3

182	28	2.33
399	30	2.5
725	33	2.75

*

1	0	0
0	1	0
0	0	1

3x3

Age (days)	Height (in)	Height (ft)
182	28	2.33
399	30	2.5
725	33	2.75

Dimensions of possible matrix factorizations? Select all that apply.

✅ (3x2) * (2x3)
✅ (3x3) * (3x3)
C. (3x1) * (1x3)
D. (3x4) * (4x3)
E. Inner dimensions higher than 4

Matrix Decomposition: Limited by Rank

3844650

3x4

182	28	2.33	0
399	30	2.5	0
725	33	2.75	0

*

1	0	0
0	1	0
0	0	1
99	31	17

4x3

Fine, but defeats the point of dimension **reduction**...

Infinite options for last row!

Age (days)	Height (in)	Height (ft)
182	28	2.33
399	30	2.5
725	33	2.75

We could keep adding 0 columns, but not useful.

We need inner dimension ≥ column rank of original data (r=2).

(A x B) * (C x D)

Dimensions of possible matrix factorizations? Select all that apply.

✅(3x2) * (2x3) **C.** (3x1) * (1x3) ✅Inner dimensions higher than 4
✅(3x3) * (3x3) ✅(3x4) * (4x3)

Matrix Decomposition: Limited by Rank

3844650

In practice, we usually construct **decompositions < rank of the original matrix.** They provide **approximate** reconstructions of the original matrix.

But, how do we find valid decompositions?

3x4

1	0	0
		0
		1
99	31	17

Fine, but defeats the point of dimension **reduction**...

Age (days)	Height (in)	Height (ft)
182	28	2.33
399	30	2.5
725	33	2.75

Impossible, because rank of original > 1!

ax/bx = a/b = 182/399
ay/by = a/b = 28/30
Contradiction!

3x1

a
b
c

*

x	y	z

1x3

Dimensions of possible matrix factorizations? Select all that apply.

✅(3x2) * (2x3) ❌(3x1) * (1x3) ✅Inner dimensions higher than 4
✅(3x3) * (3x3) ✅(3x4) * (4x3)

Automatic factorization

3844650

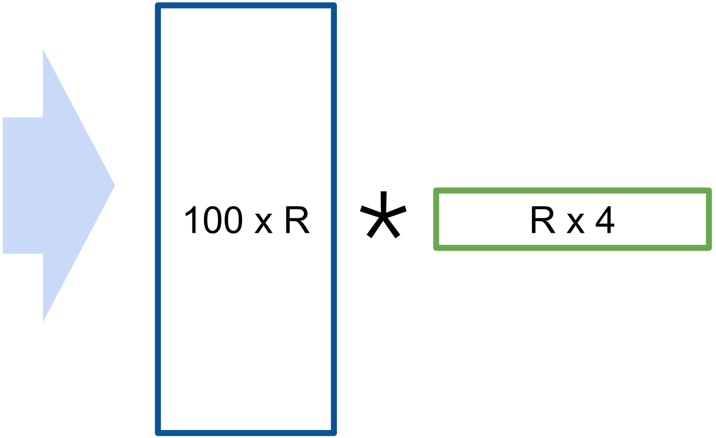
Initial goal: Find a procedure to **automatically** factorize a **rank R** matrix into an R dimensional representation multiplied by some transformation matrix.

- **Lower dimensional representation** removes redundant features.
- Imagine a 1000 dimensional dataset: If the rank is only 5, it's much easier to do EDA after this mystery procedure.

What is the rank of the matrix below? Slido!

100 x 4

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



What is the rank of the matrix?

Automatic factorization

3844650

Initial goal: Find a procedure to **automatically** factorize a **rank R** matrix into an R dimensional representation multiplied by some transformation matrix.

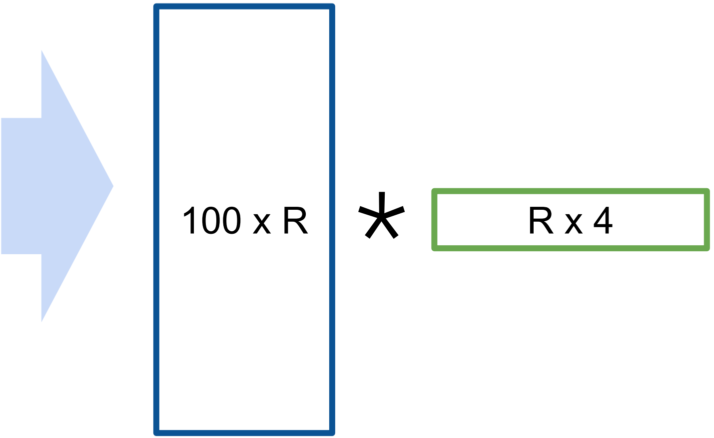
- **Lower dimensional representation** removes redundant features.
- Imagine a 1000 dimensional dataset: If the rank is only 5, it's much easier to do EDA after this mystery procedure.

Perimeter is a linear combination of width and length. **Area is not!** So, R=3.

What is the rank of the matrix below?

100 x 4

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



Automatic and Approximate factorization

3844650

What if we wanted a **2-D** representation?

- Rank of the 100x4 matrix is 3, so we can no longer **exactly** reconstruct the 100x4 matrix.

Still, some 2D matrices yield **better approximations** than others. **How well can we do?**

100 x 4

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

$\approx$

100 x **2**

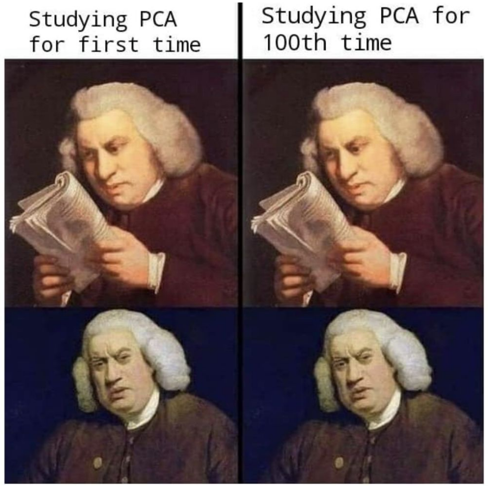
...	...

$*$

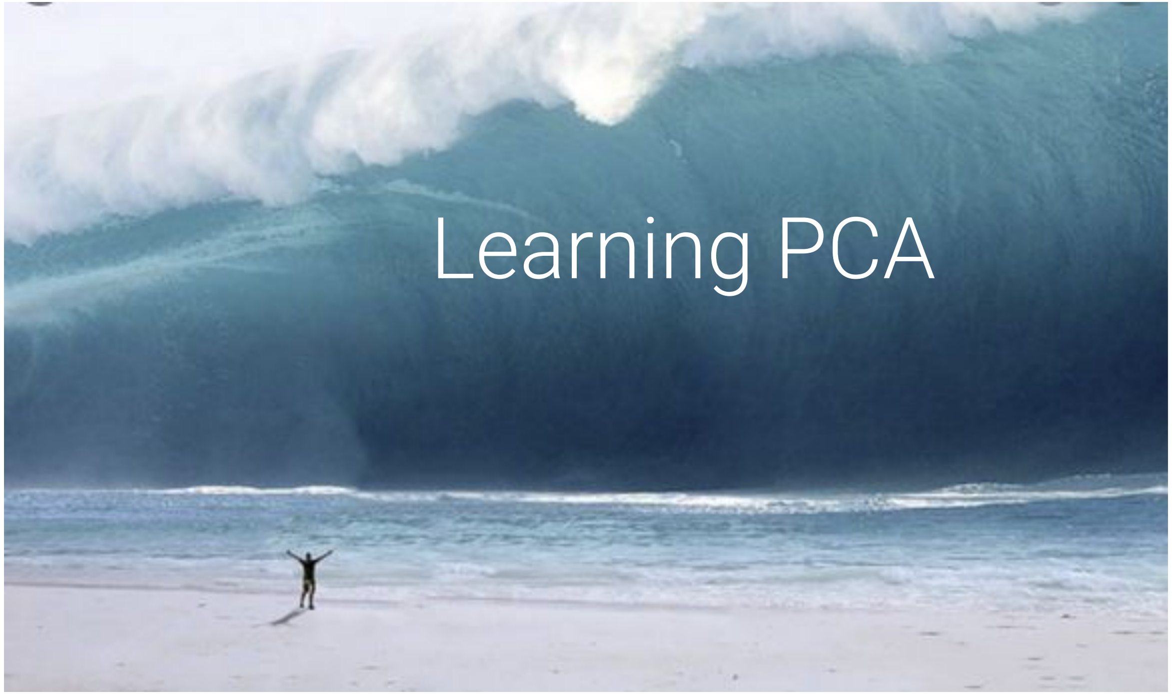
2 x 4

2-minute stretch break!

Lecture 24, Data 100 Spring 2025



Incoming Tsunami of Conceptual Challenges!



Be kind to yourself 💙 By far the biggest tsunami of Data 100!

Principal Component Analysis (PCA)

Lecture 24, Data 100 Spring 2025

3844650

Maximizing variance: A common point of confusion

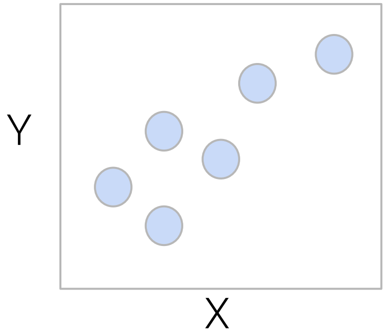
3844650

In supervised learning, we often say that **minimizing variance** is a goal.

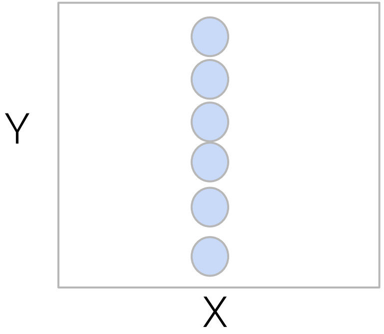
This is shorthand for minimizing the **variance of our predictions ($\hat{Y}$)**. We want similar predictions across models trained on different random samples of the same population.

In this section, we talk about **maximizing variance** captured from the original data.

We want to retain **variance of the features (X)**. Variance in the features is **information**. For example, if the features have no variance, we cannot use them to make predictions.



Var(X) > 0



Var(X) = 0

Principal Component Analysis (PCA)

3844650

Goal: Transform observations from high-dimensional data down to **low dimensions** (often 2, so we can viz!) through linear transformations.

Related Goal: Low-dimension representation should capture the **variability** of the original data.

(to define later)

100 x 4

width	length	area	perimeter
20	20	400	80
16	12	192	56
10	10	100	40
...	...	...	...
24	12	288	72

*

Principal Components (PCs) (cols)

4x2

First PC (PC1)

Second PC (PC2)

Transformation Matrix

Latent Factors/Features (cols)

100 x 2

Latent Feature 1

Latent Feature 2

...

...

<ins>Latent</ins>: Hidden or underlying

Principal Component Analysis (PCA)

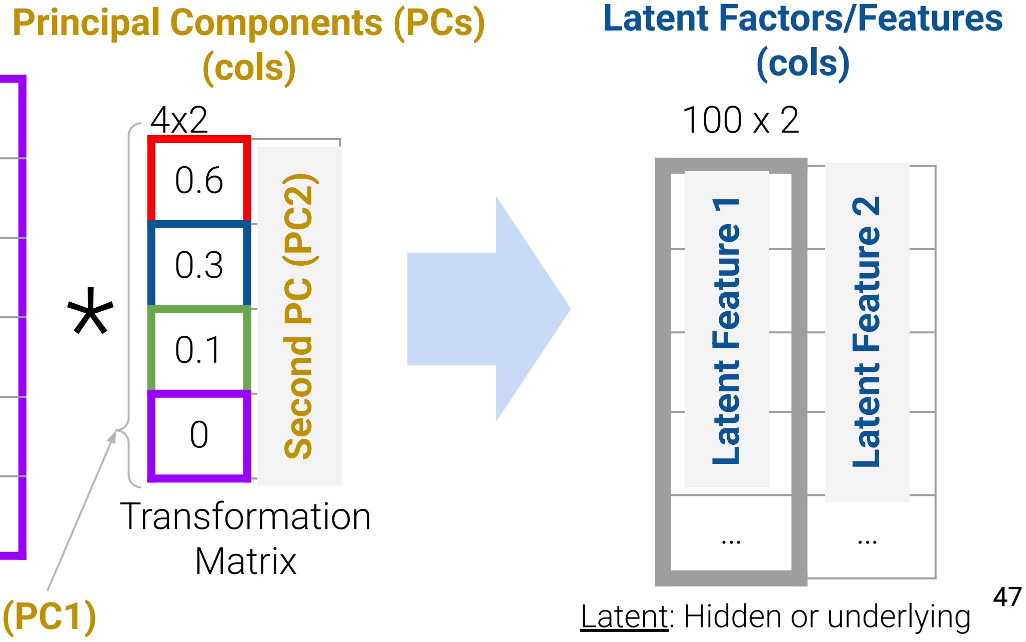
3844650

The PCs are **recipes** for constructing the **latent features**.

"To make our new+improved **Latent Feature 1**, combine **PC1[0]** parts width, **PC1[1]** parts length, **PC1[2]** parts area, and **PC1[3]** parts perimeter."

100 x 4

width	length	area	perimeter
20	20	400	80
16	12	192	56
10	10	100	40
...	...	...	...
24	12	288	72



Latent: Hidden or underlying

Why perform PCA?

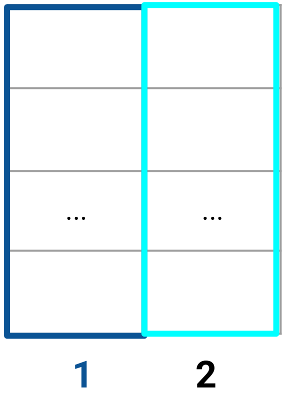
3844650

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

Related Goal: Low-dimension representation should capture the **variability** of the original data.

Often work with Latent Factors

100 x 2



1. **Visually** identify clusters of similar high-dimensional observations.

- Most visualizations are 2-D, so often construct 2 dimensions.

2. You believe the data are inherently low rank, e.g., **just a few features** could approximately determine the rest through linear associations.

3. Some models benefit from decorrelated features (e.g., Naive Bayes).

- PCA **eliminates correlations** between features.

Two Equivalent Framings of PCA

3844650

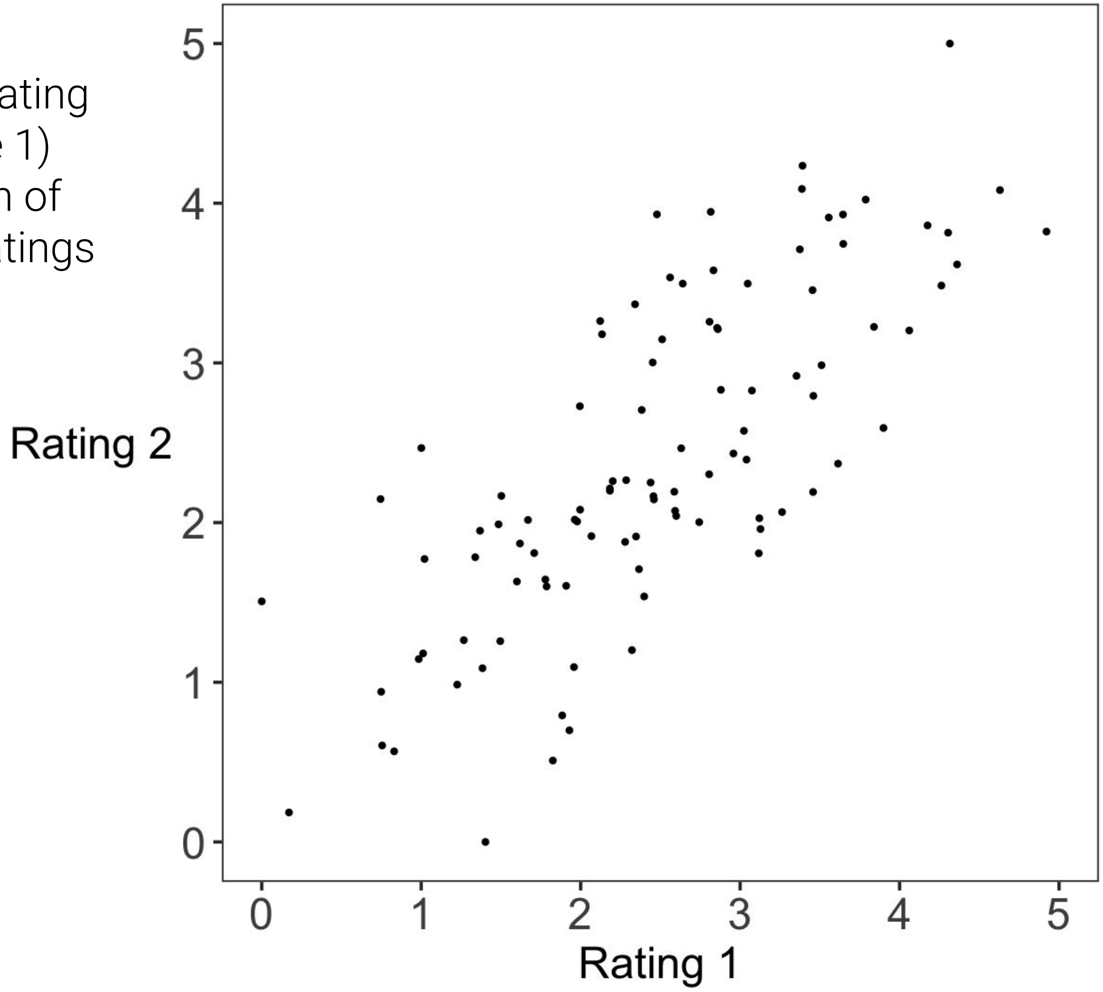
There are two equivalent ways to frame PCA:

1. Finding the directions of **maximum variability** in the data
2. Finding the low dimensional (rank) matrix factorization that best **approximates** the data**.**

We will start with the **variance maximization** framing (more common) and then return to the **best approximation** framing (more general).

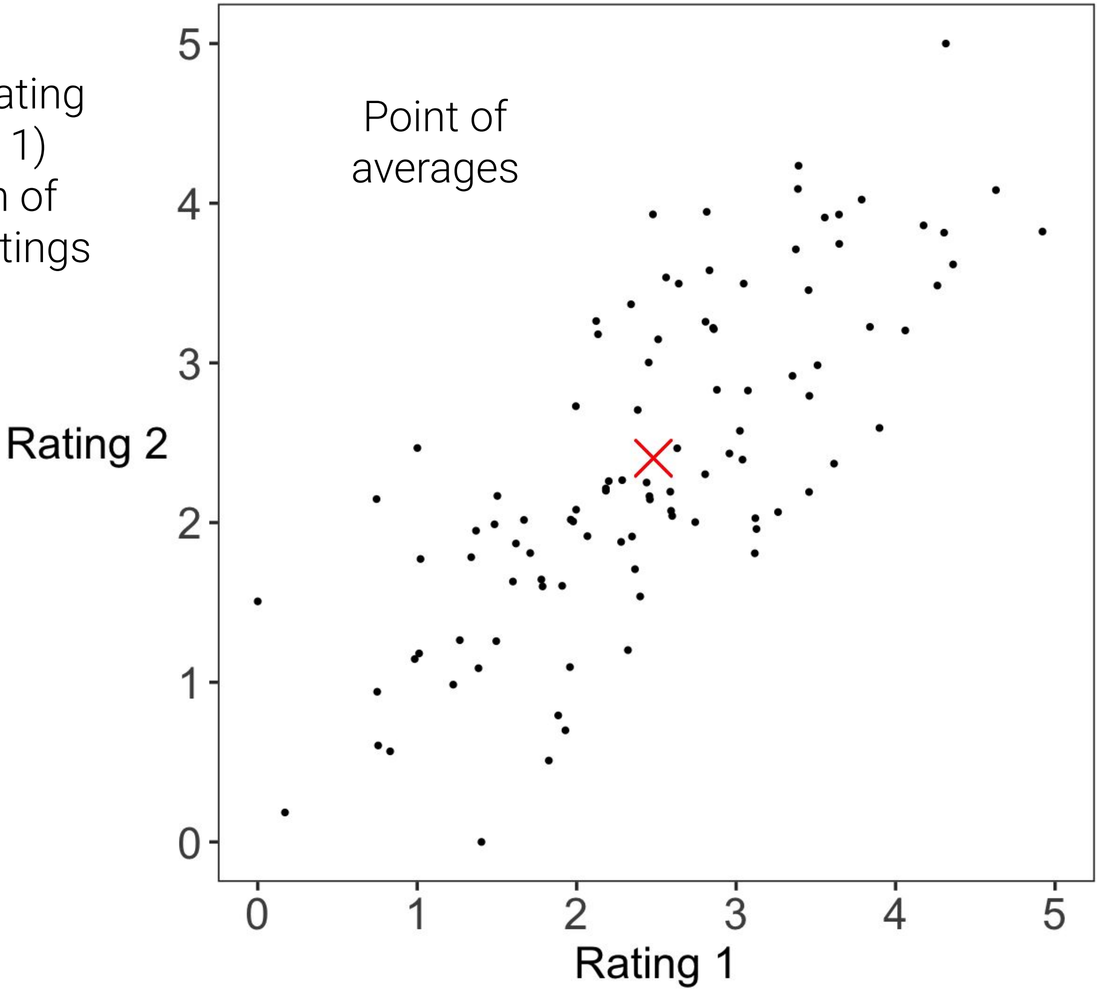
As you explore more advanced dimensionality reduction techniques, they will often seek to find “simplified representations” of data from which we can still approximately recover the original data, following framing 2.

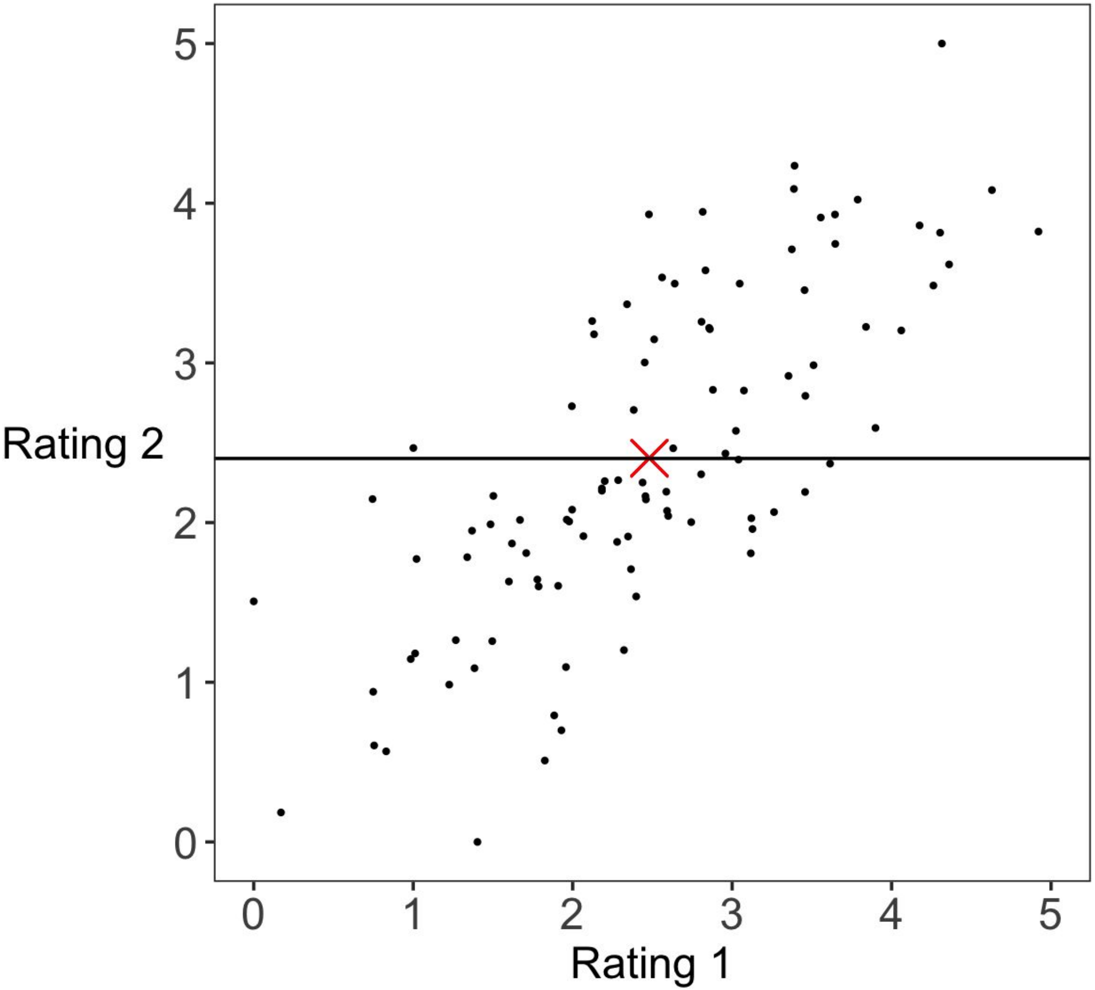
Base case: Can we construct a "new" rating (i.e., Latent Feature 1) that captures much of the variability of Ratings 1 and 2?



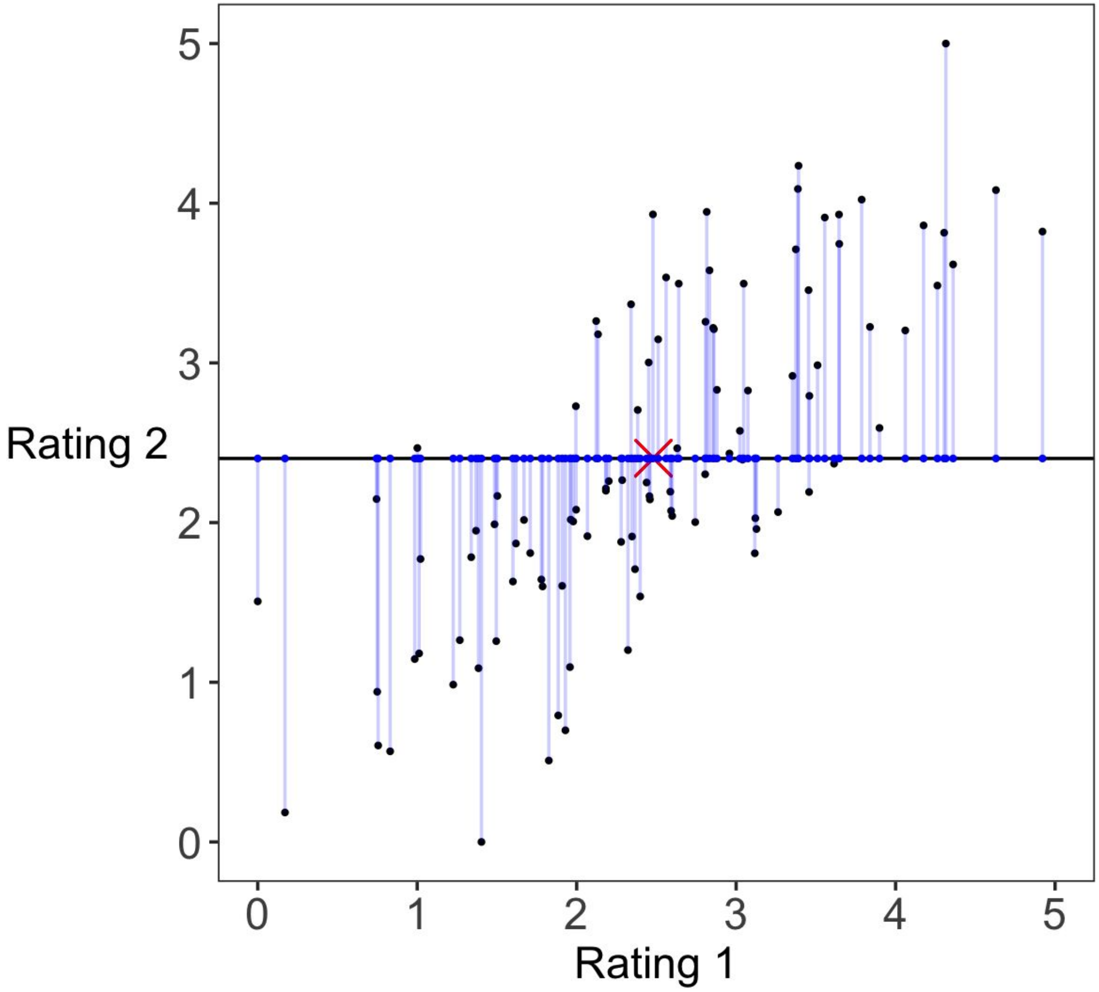
3844650

Base case: Can we construct a "new" rating (i.e., Latent Feature 1) that captures much of the variability of Ratings 1 and 2?

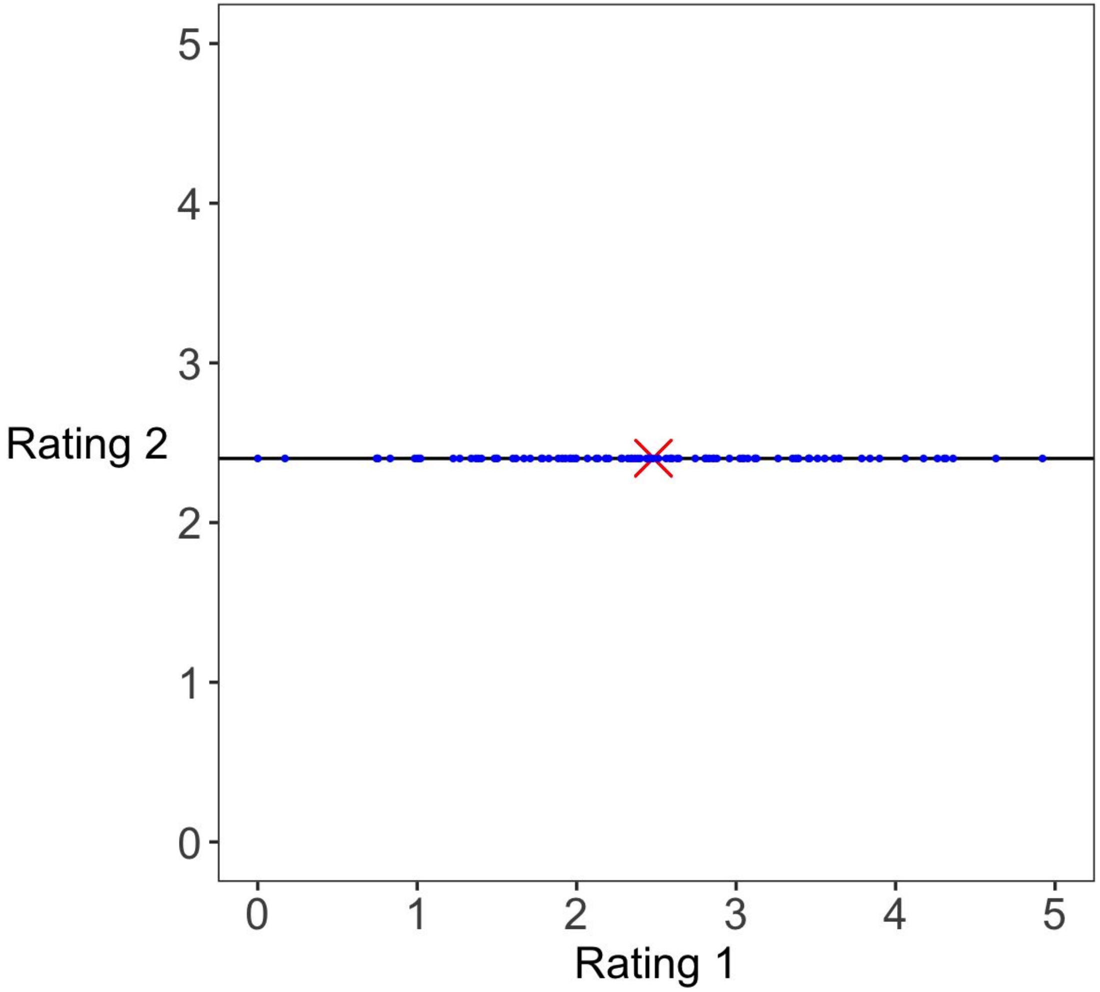




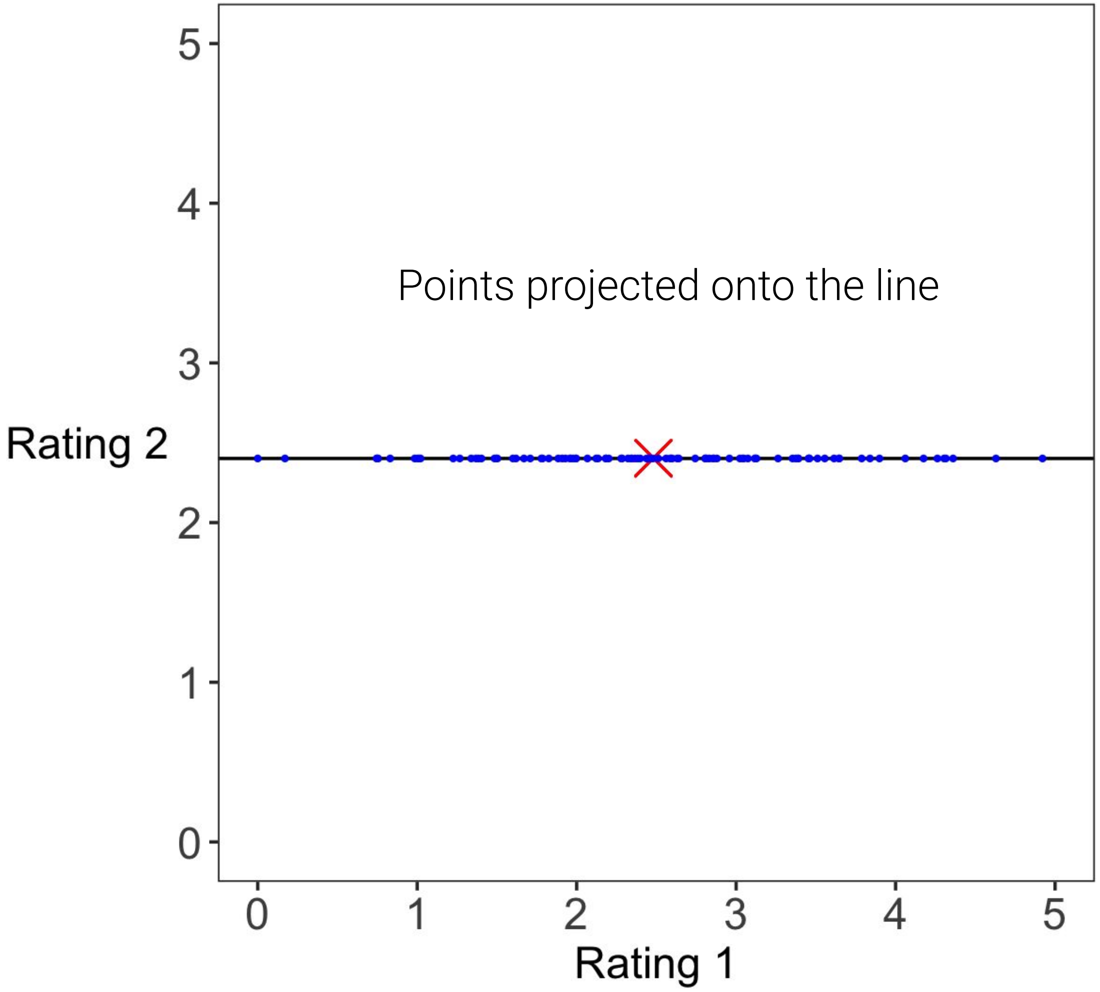
3844650



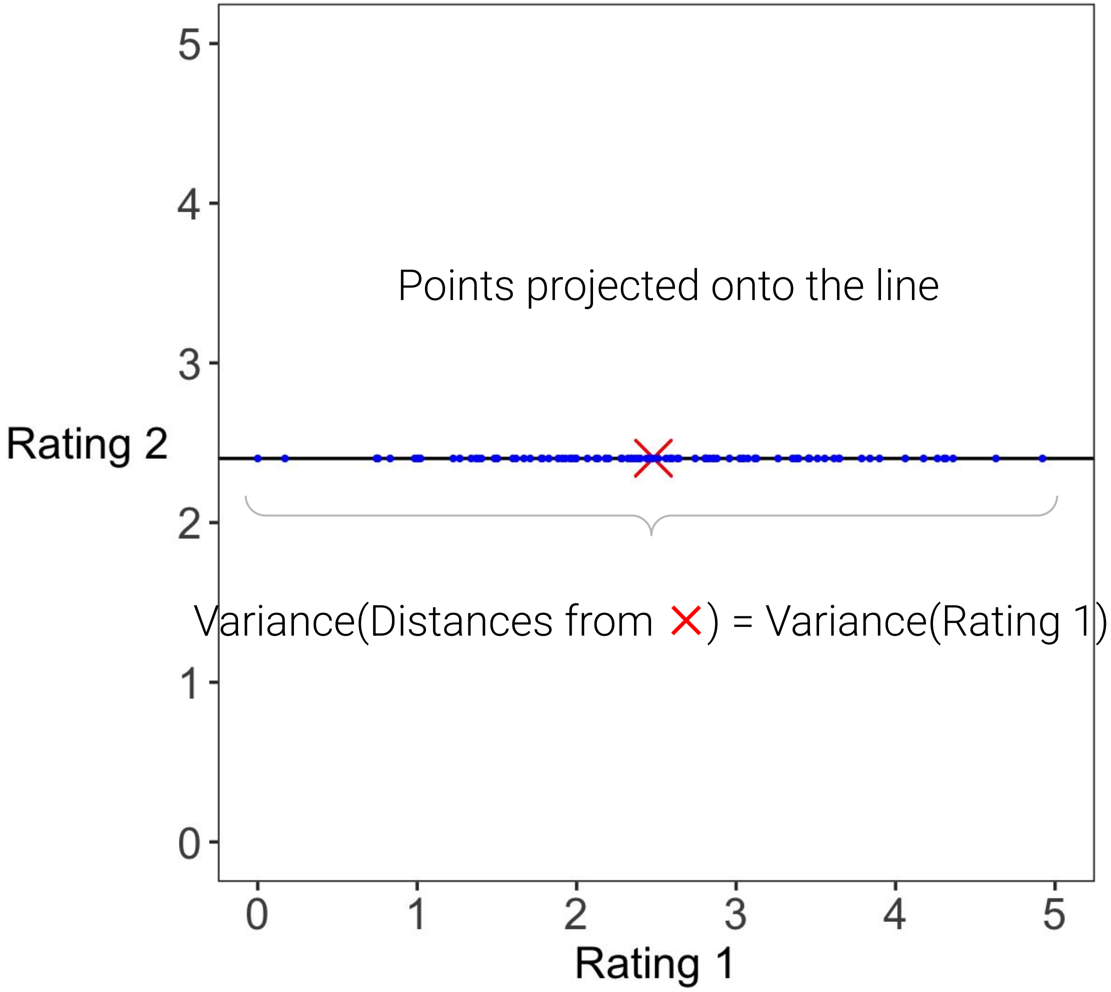
3844650



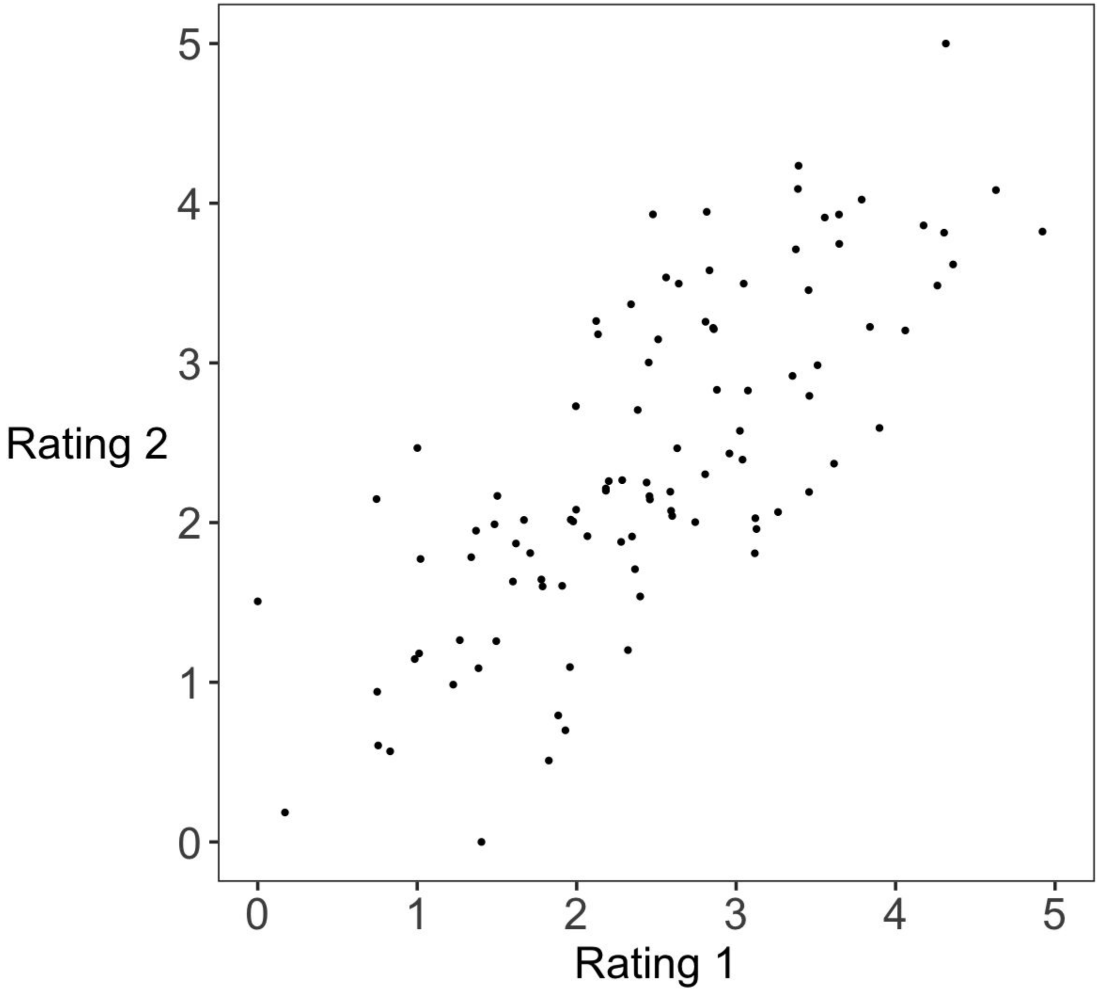
3844650



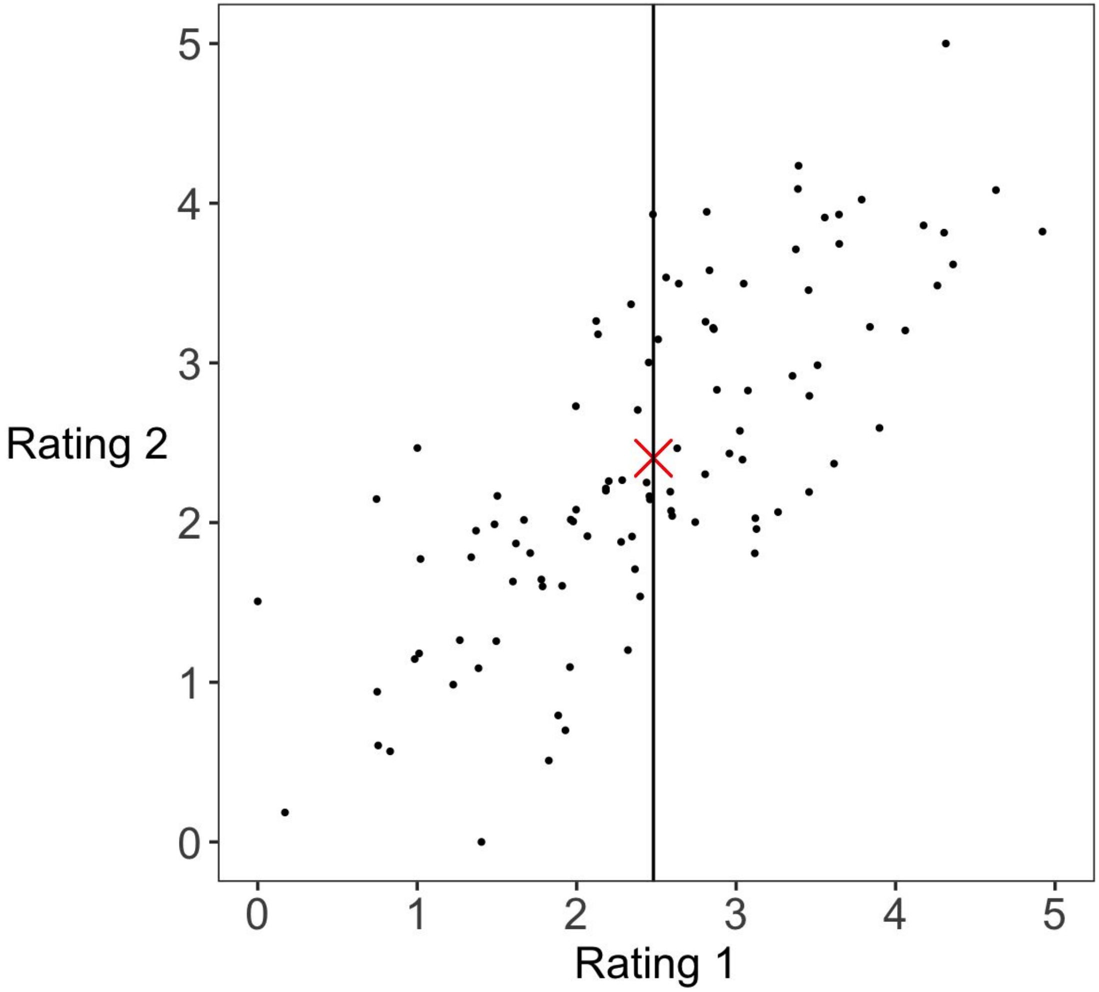
3844650



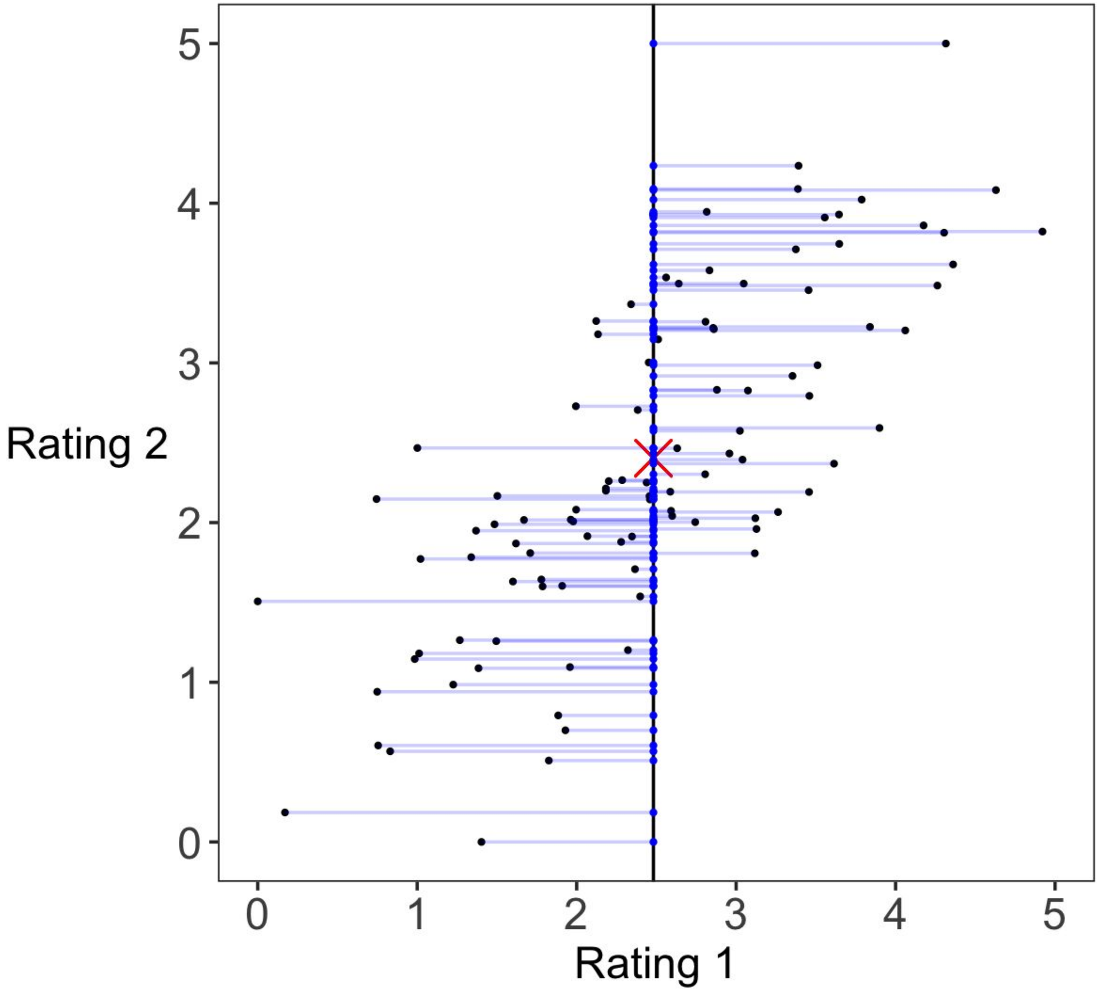
Points projected onto the line
Variance(Distances from ×) = Variance(Rating 1)
Rating 2
Rating 1
0
1
2
3
4
5



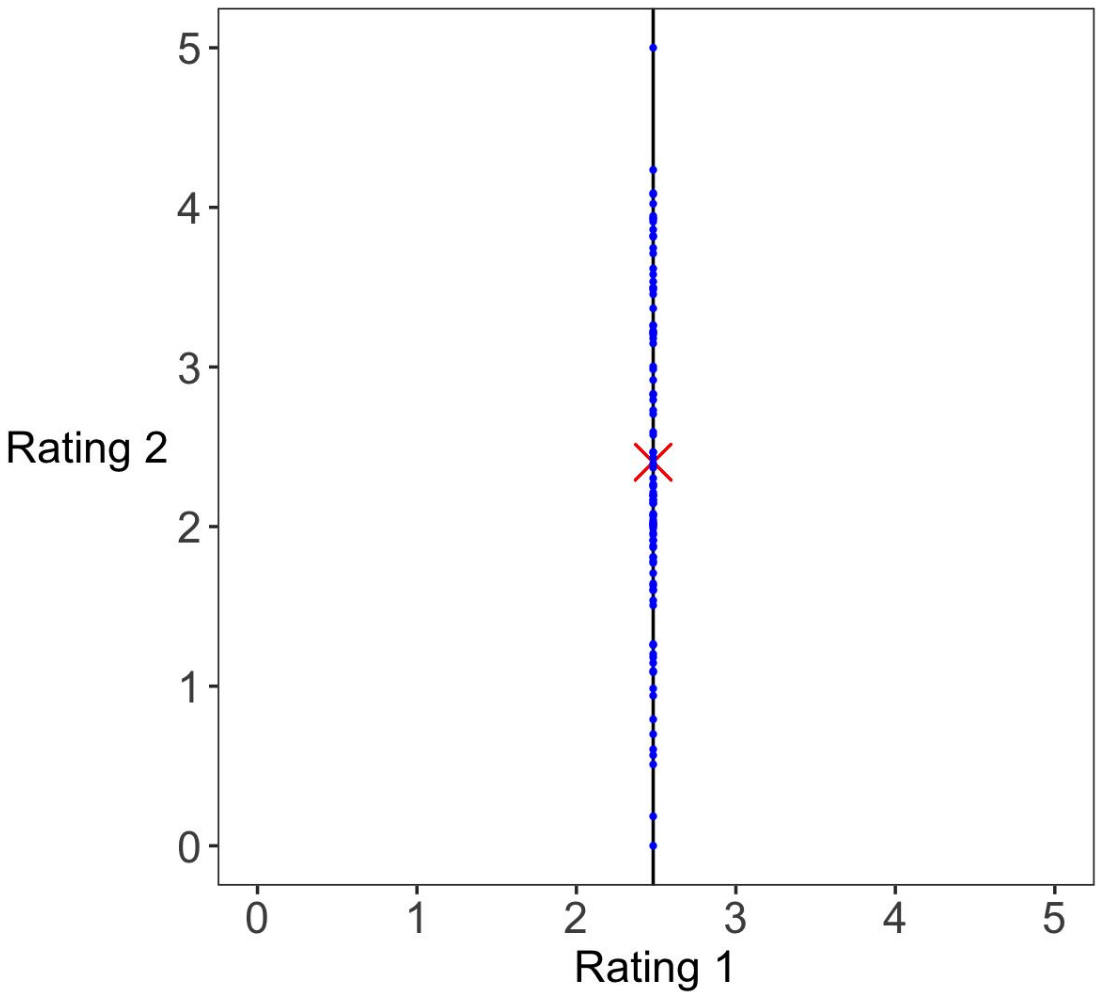
3844650



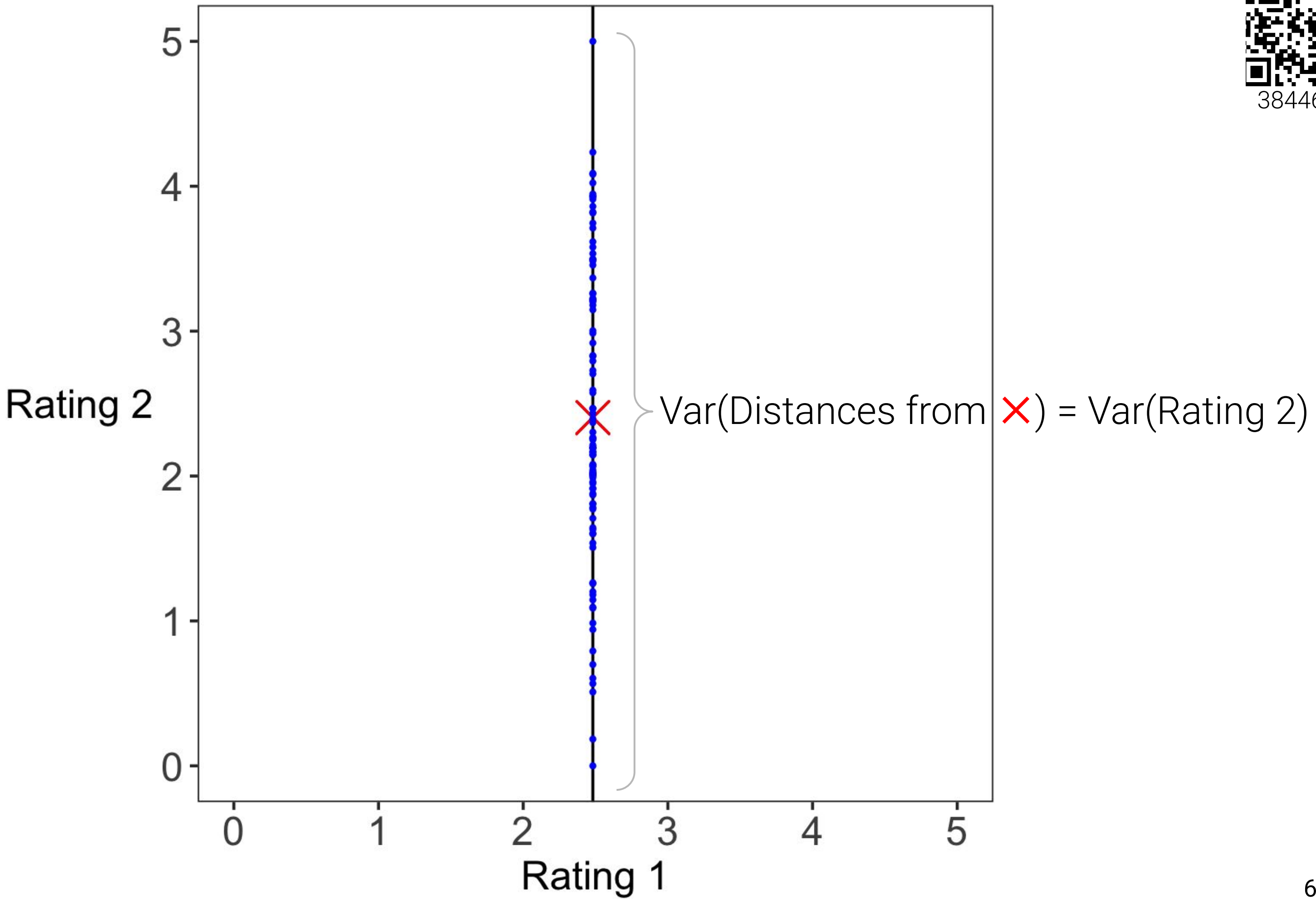
3844650



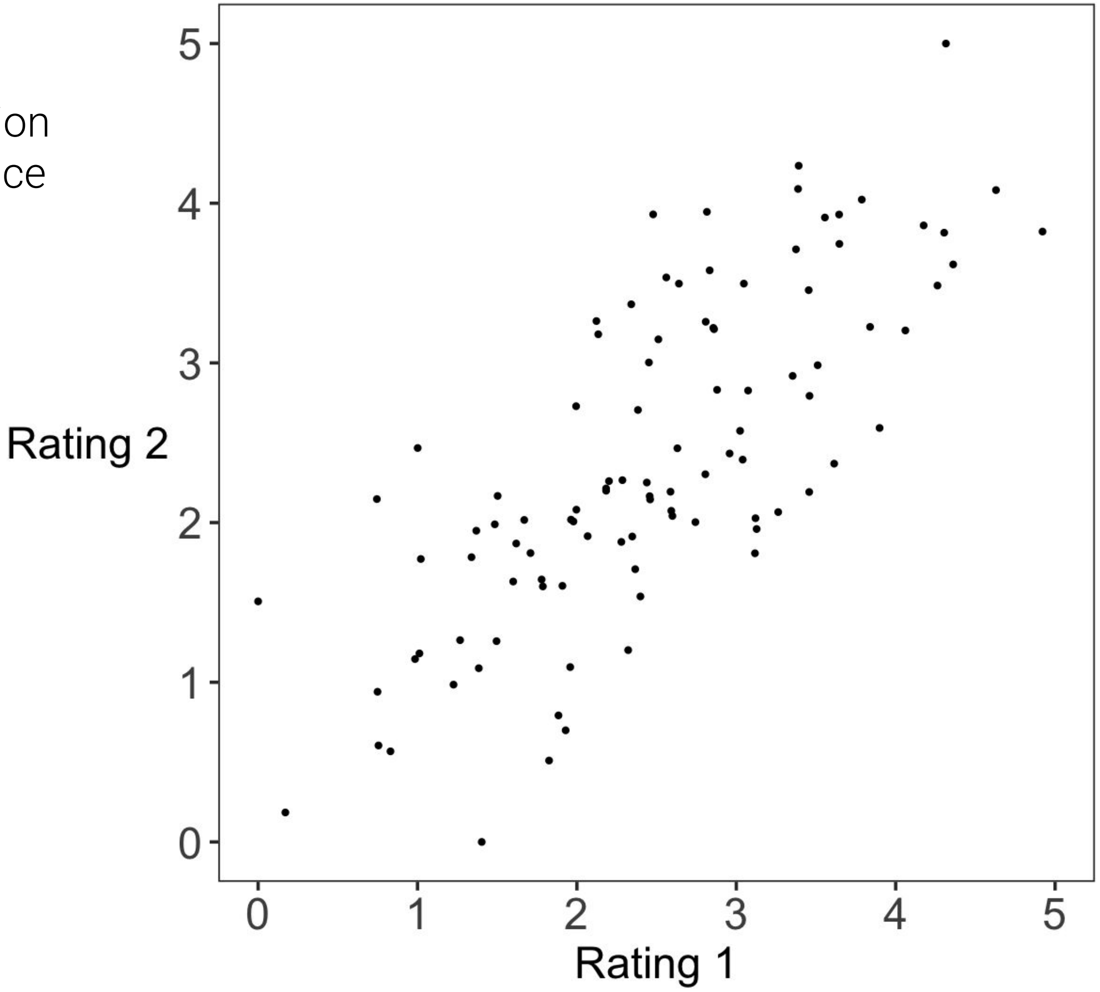
Rating 2
Rating 1
0
1
2
3
4
5

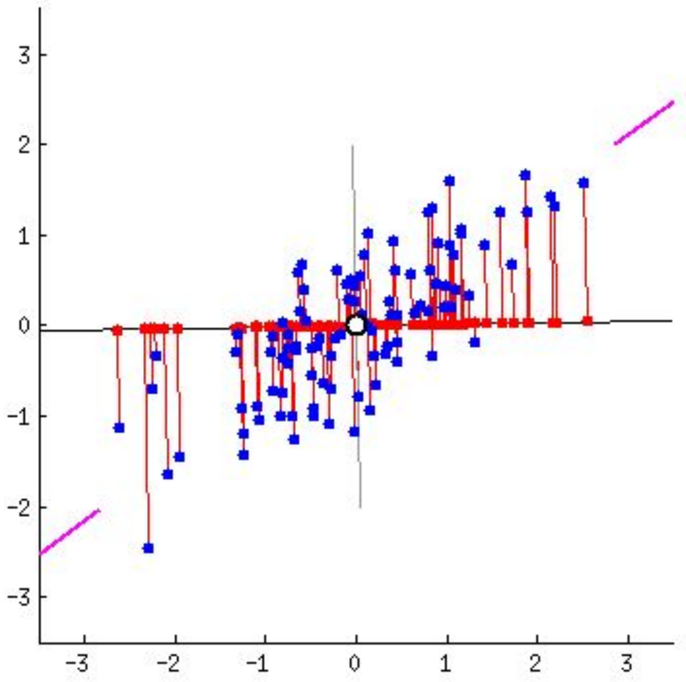


3844650


Rating 2
5
4
3
2
1
0
0
1
2
3
4
5
Rating 1
Var(Distances from ✕) = Var(Rating 2)

What is the direction of maximal variance of the data?





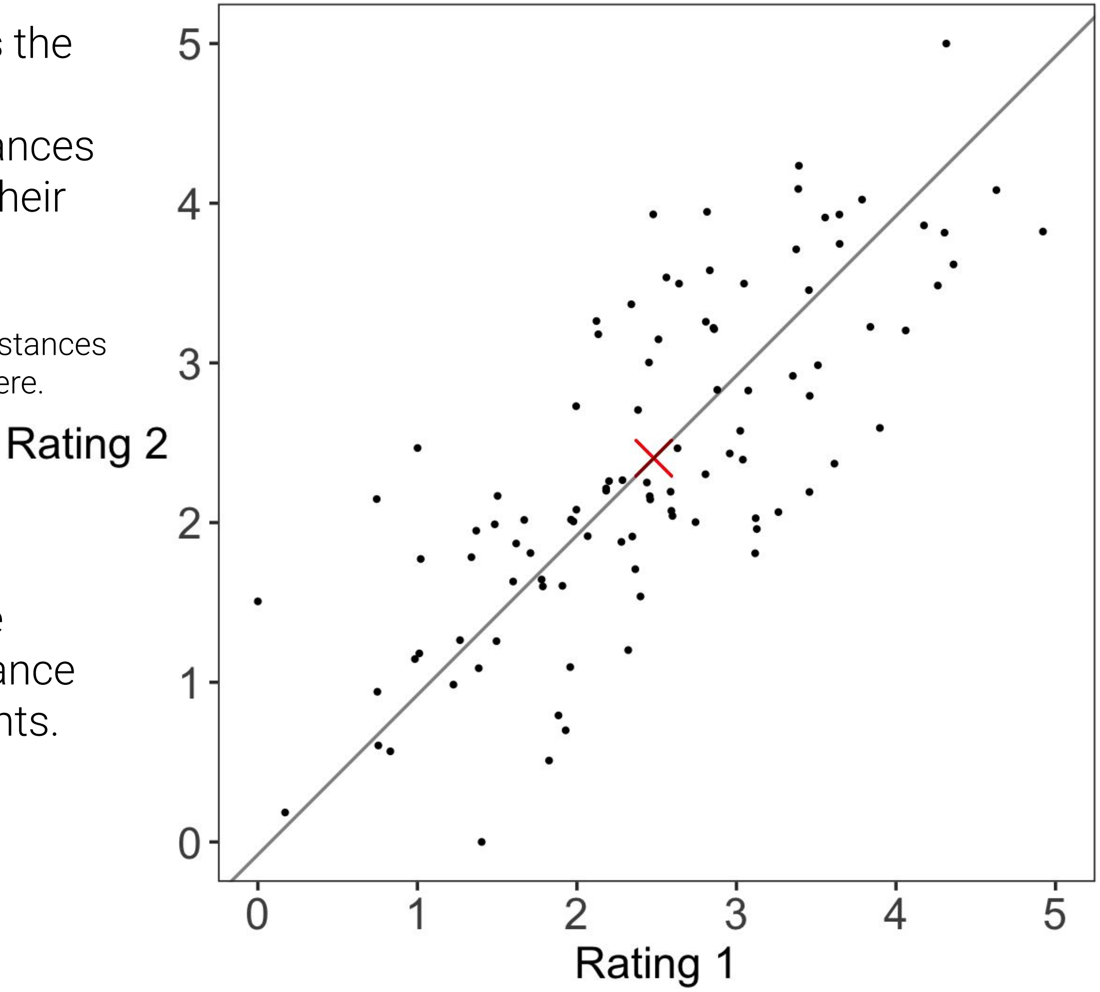
Maximizing variance = **Spreading out red dots**

Equivalent: Minimize sum of squared **perpendicular** distances from points to projected point

[StackExchange]

This line **minimizes** the sum of squared **perpendicular** distances from the points to their projection

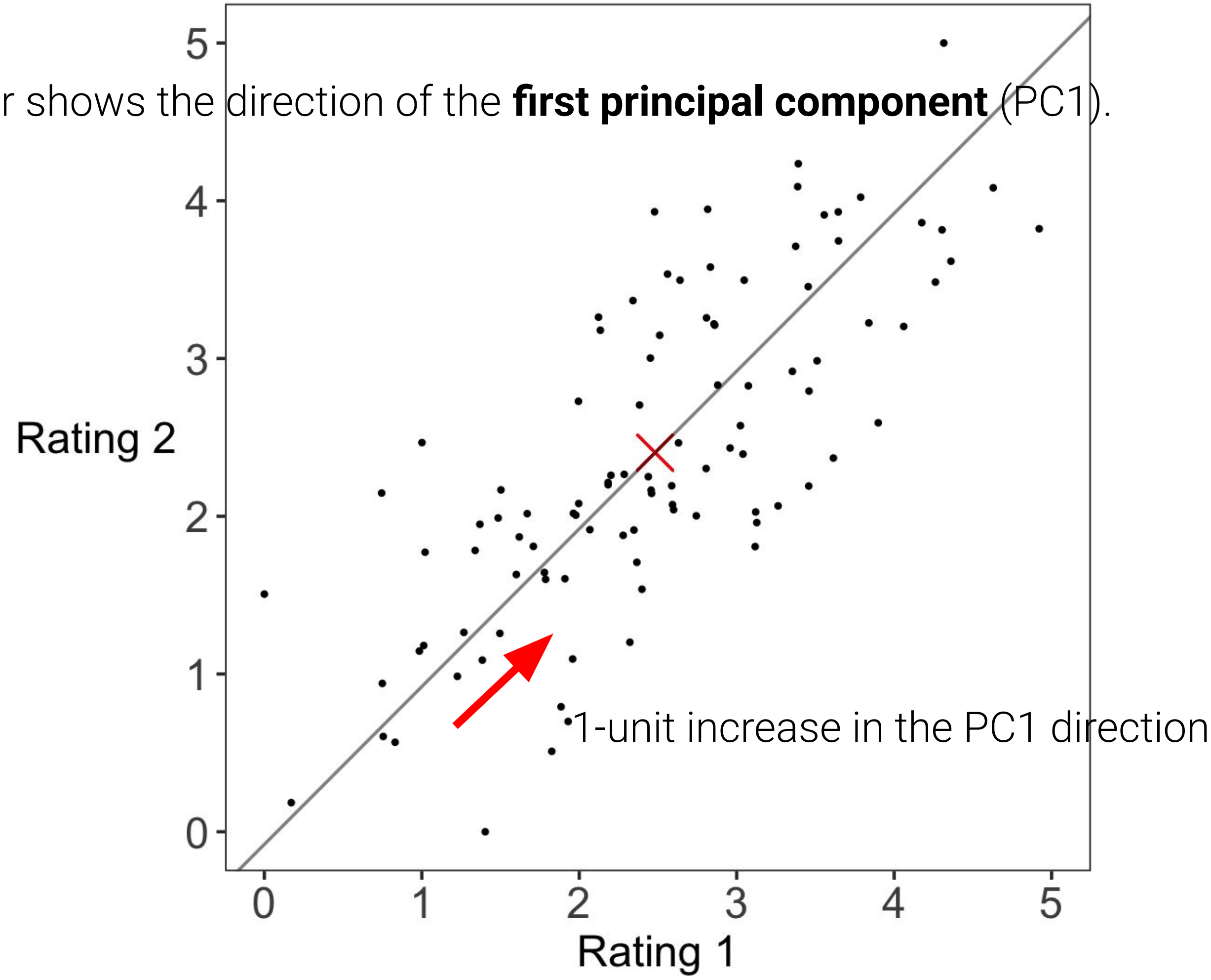
OLS minimizes **vertical** distances to outcomes (Y). No Y's here.



Equivalently, the line **maximizes** the variance of the projected points.

3844650

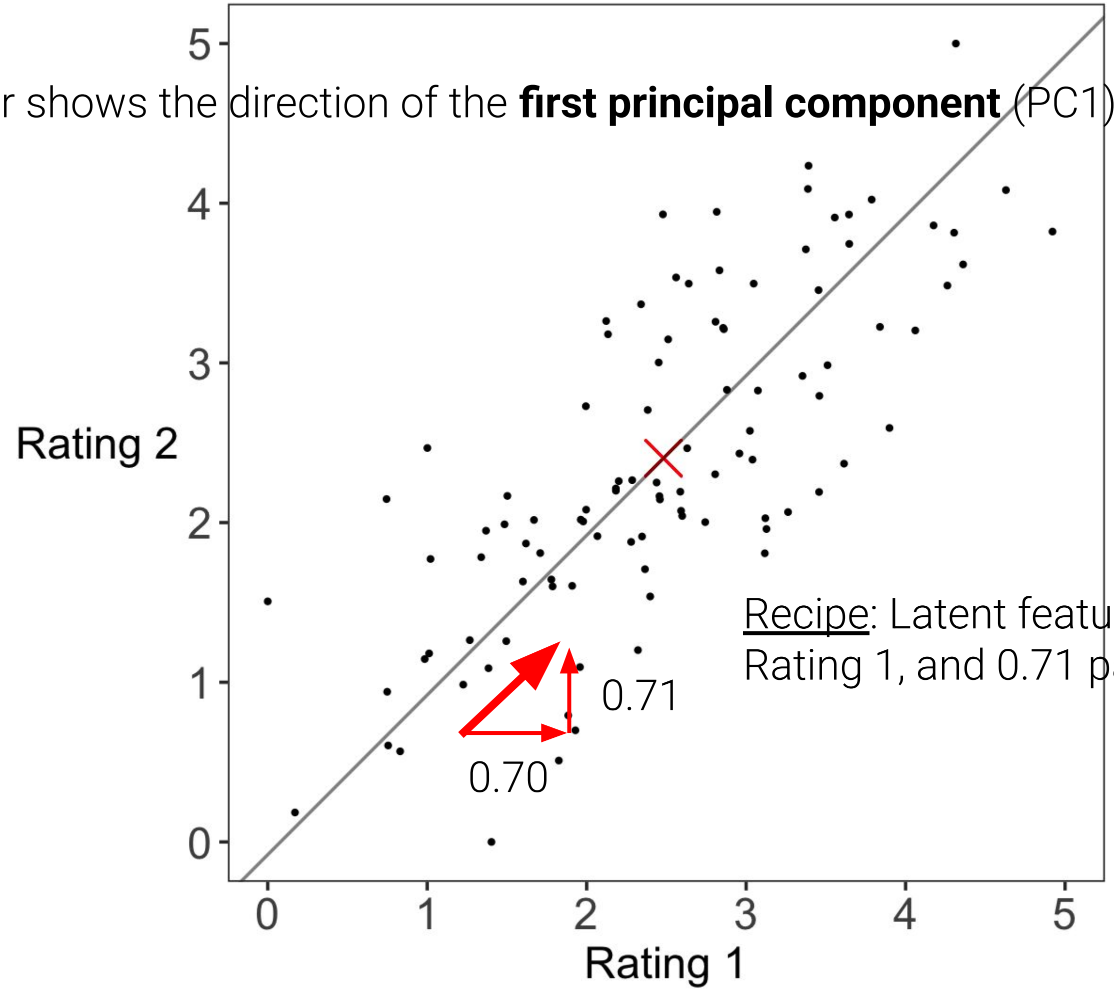
This length 1 vector shows the direction of the **first principal component** (PC1).



3844650

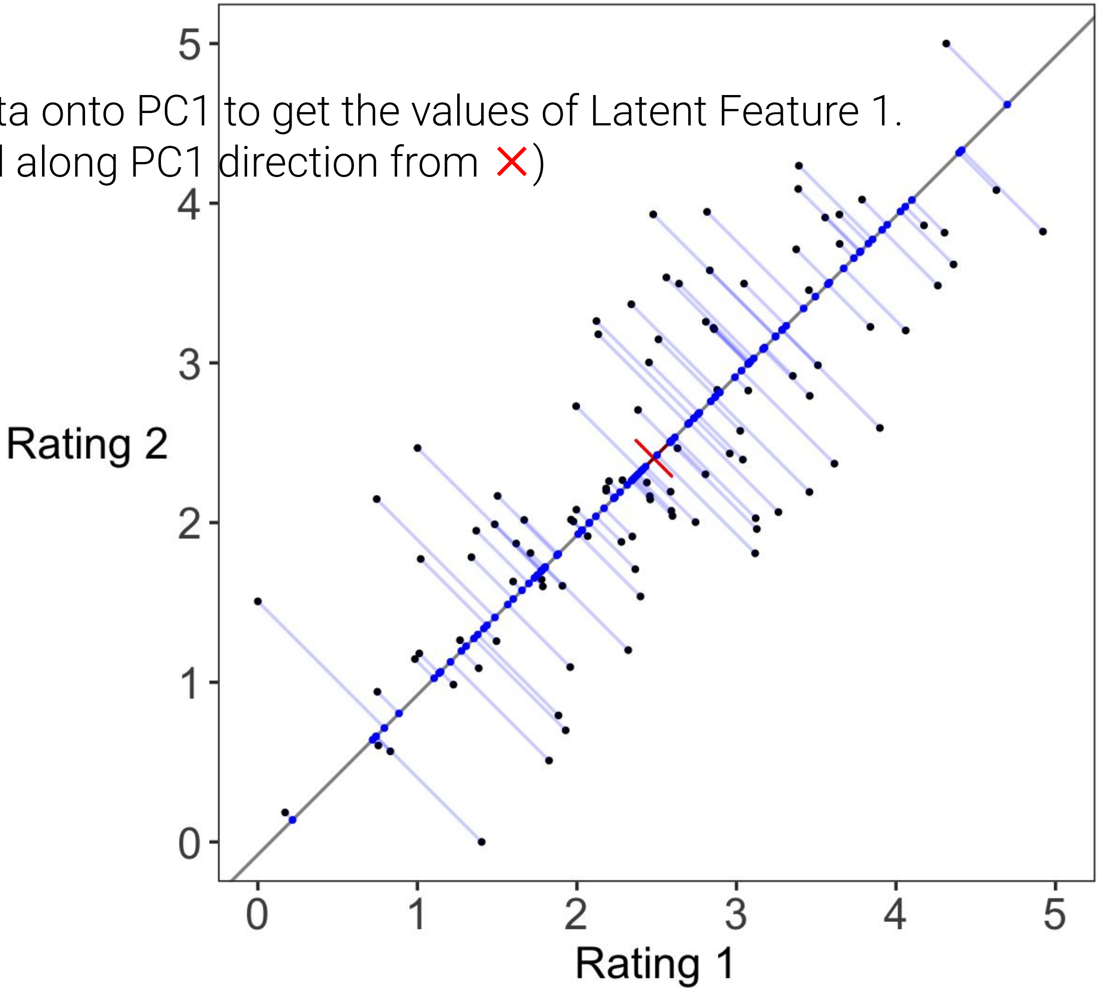
This length 1 vector shows the direction of the **first principal component** (PC1).

3844650



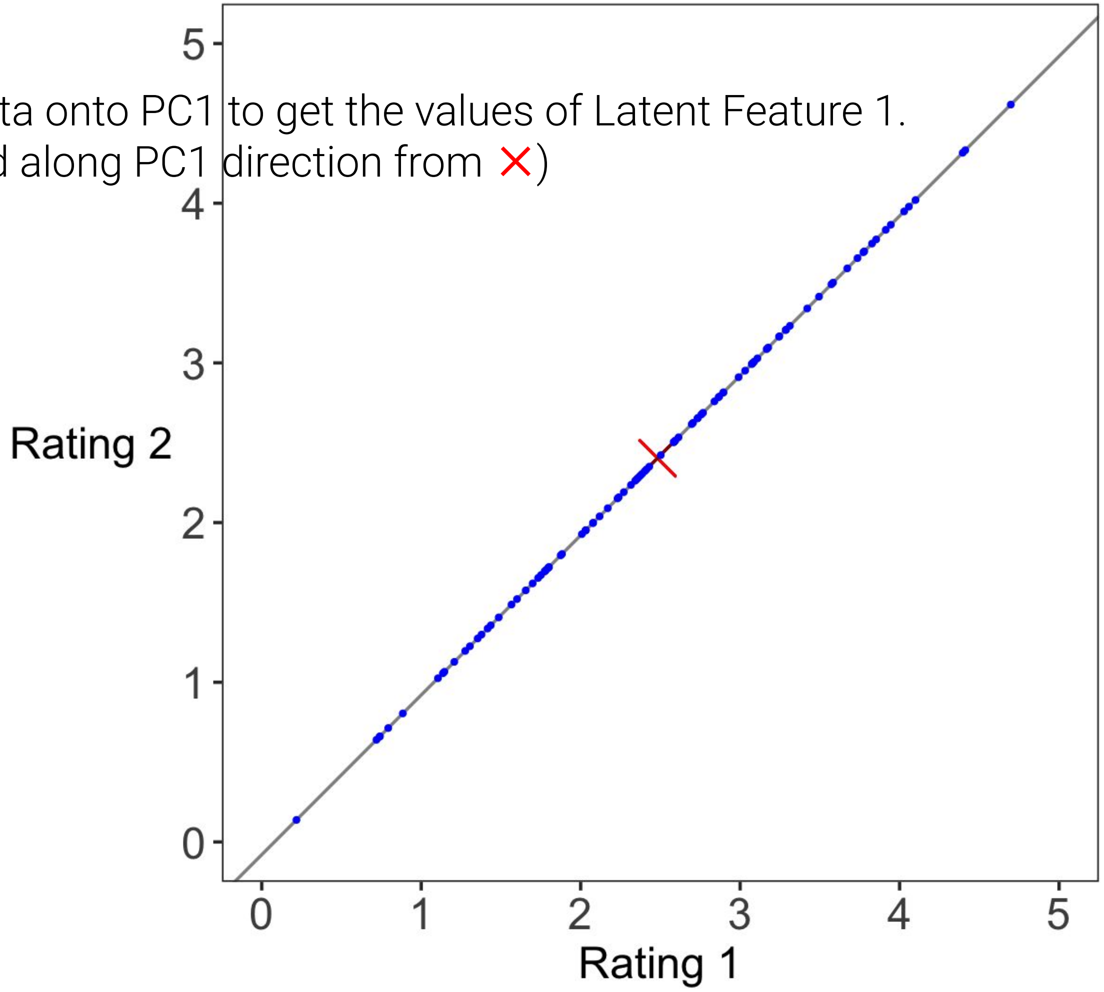
Recipe: Latent feature 1 is 0.70 parts Rating 1, and 0.71 parts Rating 2.

We project the data onto PC1 to get the values of Latent Feature 1.
(Values measured along PC1 direction from ✕)



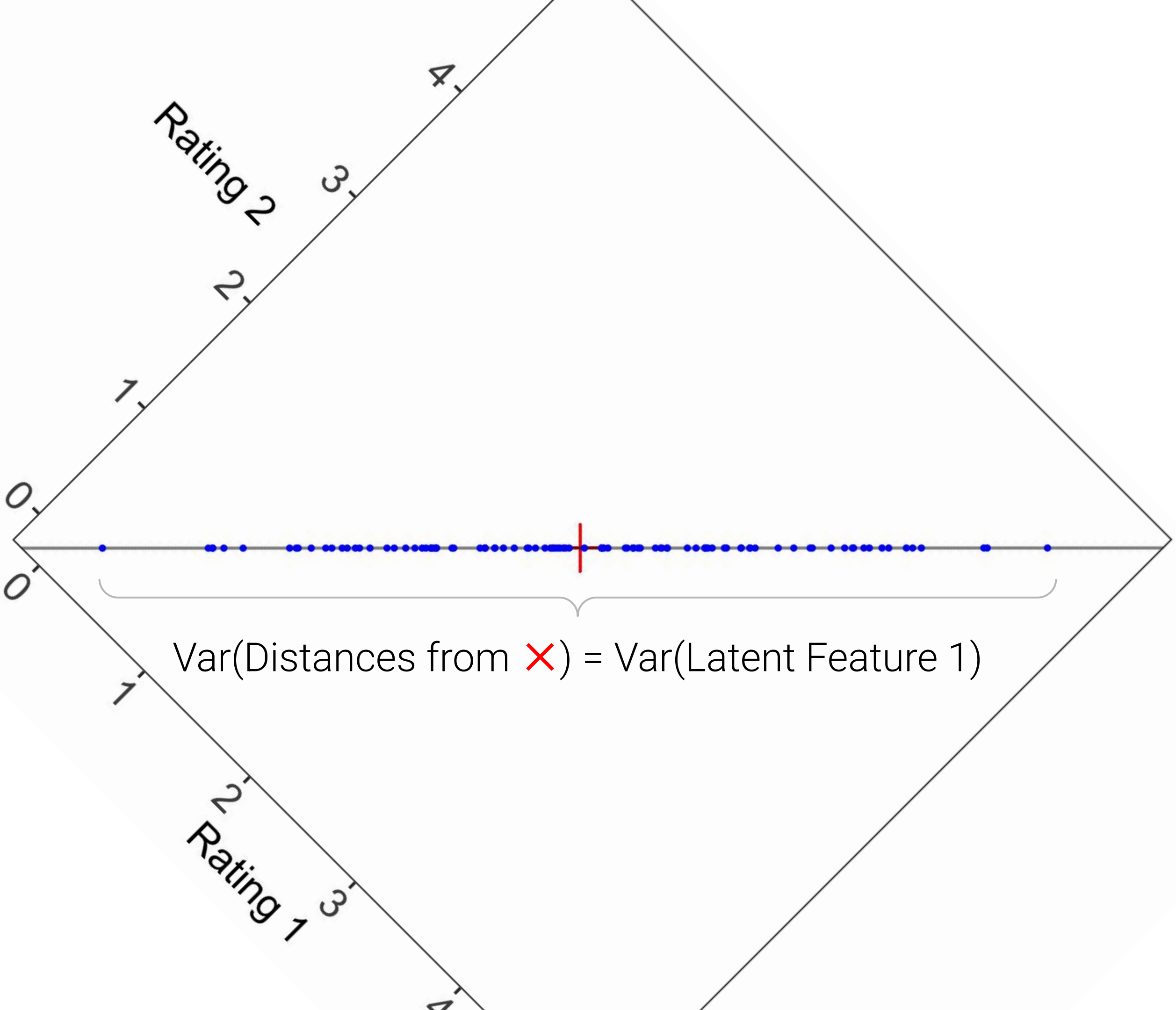
3844650

We project the data onto PC1 to get the values of Latent Feature 1.
(Values measured along PC1 direction from ×)

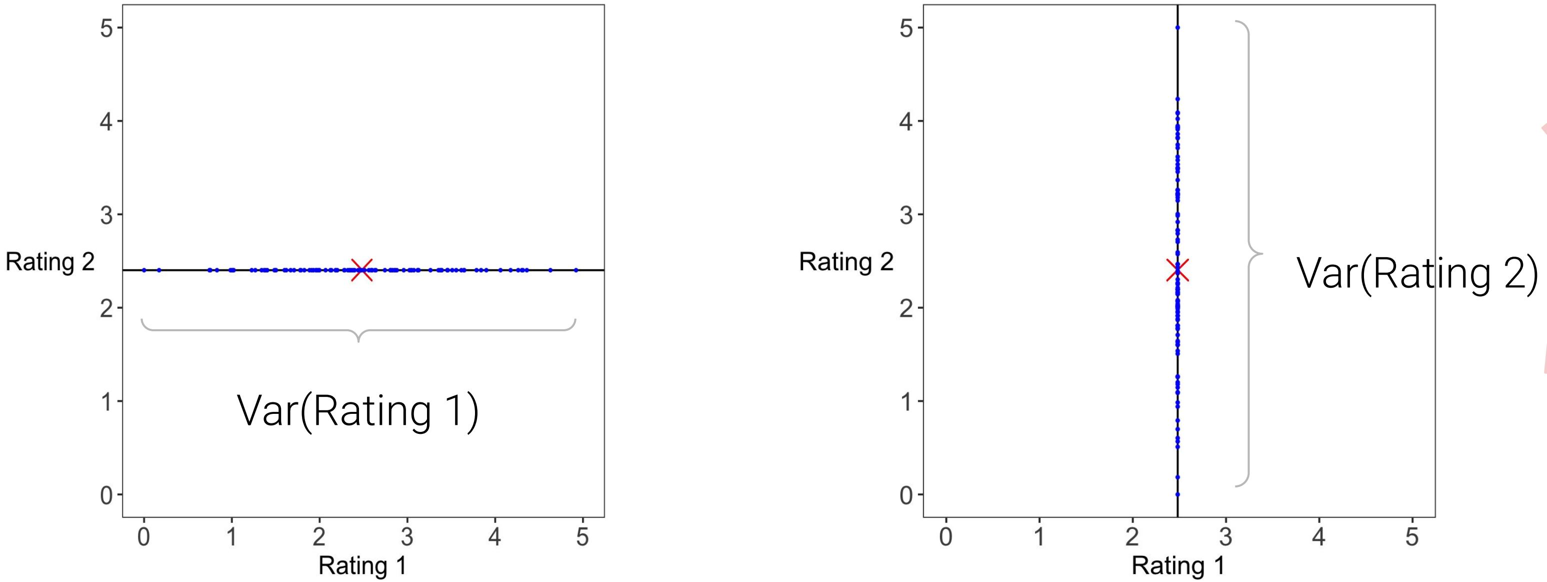


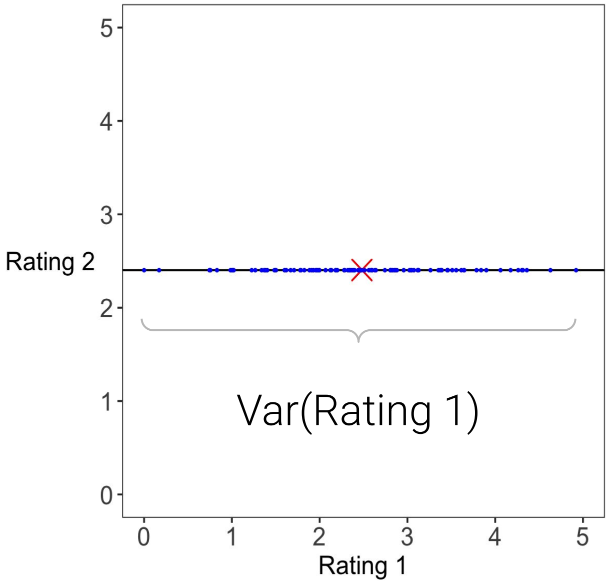
3844650

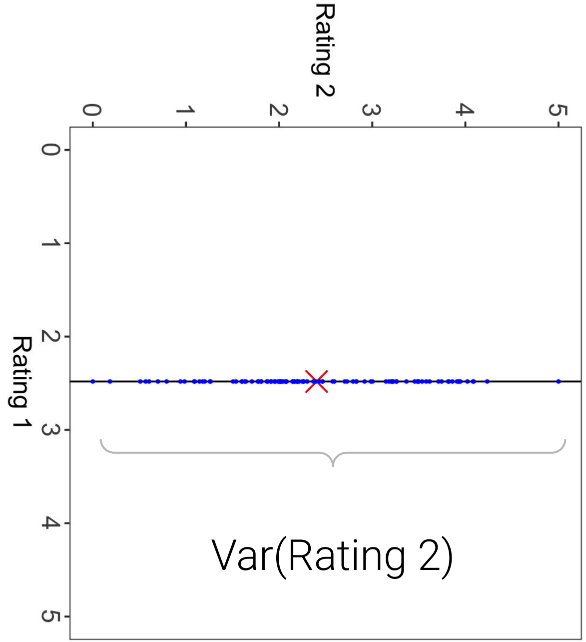
3844650


Rating 2
0
1
2
3
4
Rating 1
0
1
2
3
4
Var(Distances from ✕) = Var(Latent Feature 1)

3844650


5
4
3
2
1
0
Rating 2
Var(Rating 1)
0 1 2 3 4 5
Rating 1
5
4
3
2
1
0
Rating 2
Var(Rating 2)
0 1 2 3 4 5
Rating 1

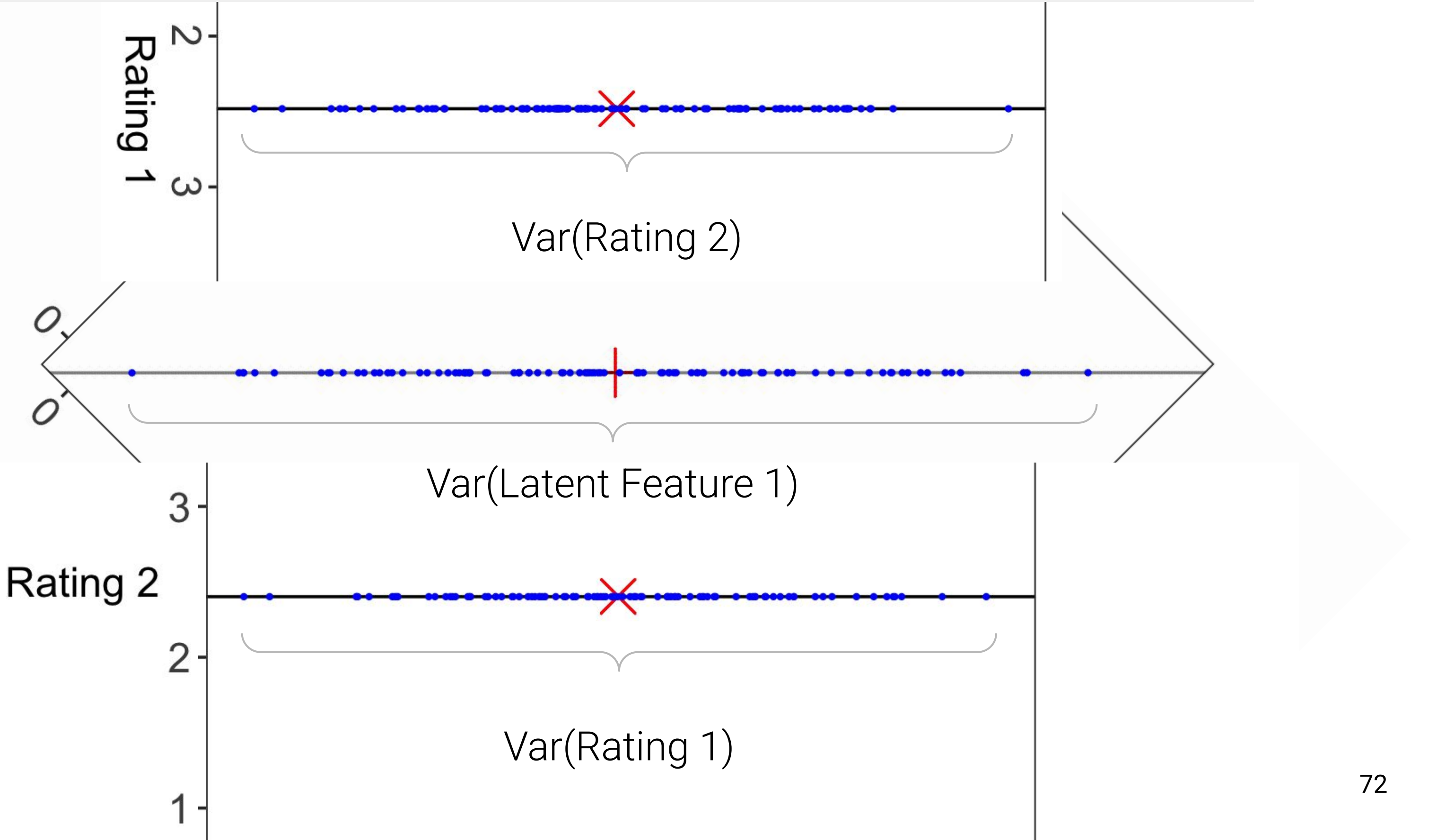

5
4
3
Rating 2
2
1
Var(Rating 1)
0
0
1
2
3
4
5
Rating 1


Rating 2
Rating 1
Var(Rating 2)

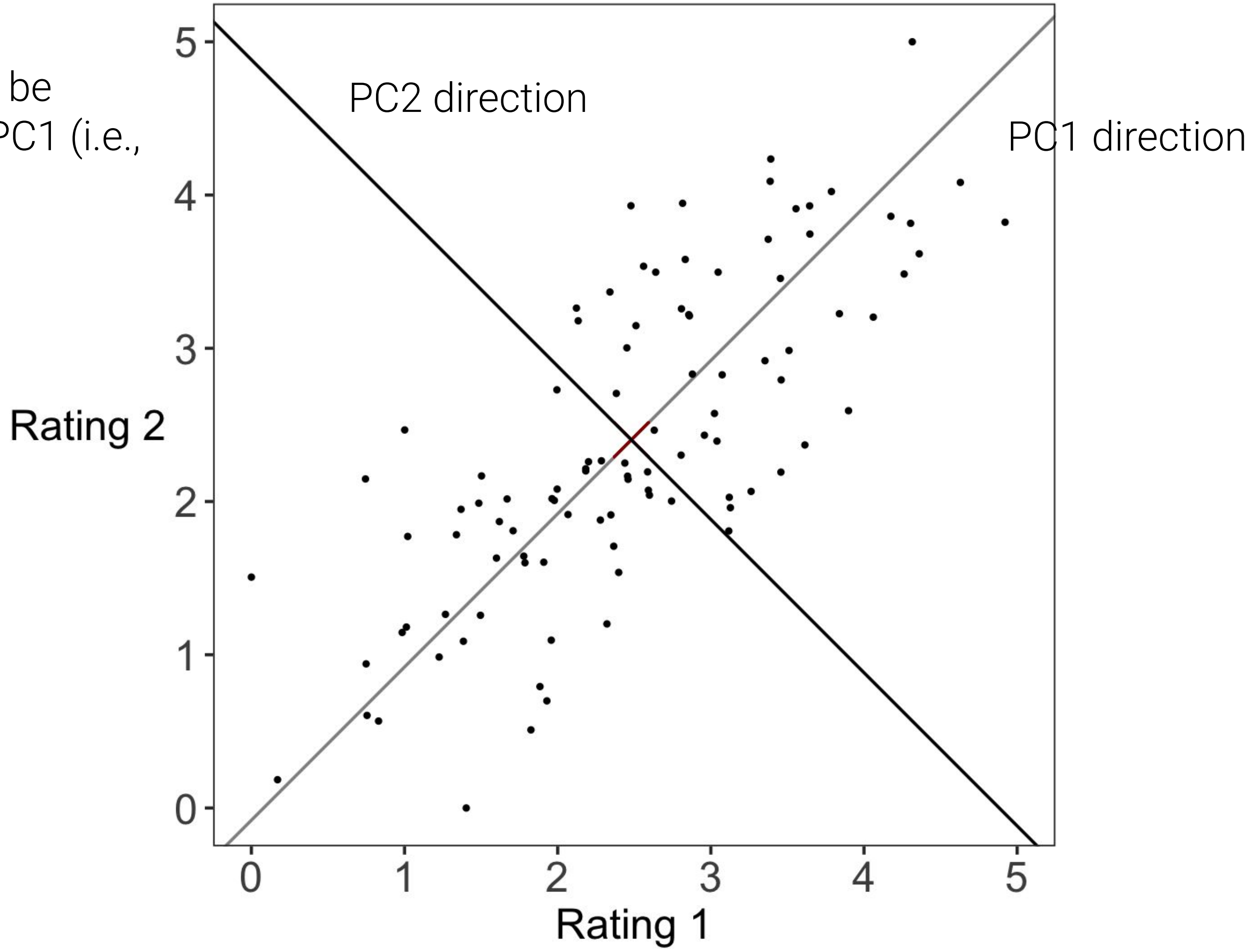
The PC1 dimension has **greater variance** than either of the original dimensions.

Coarsely, it contains **more information** than either dimension alone.

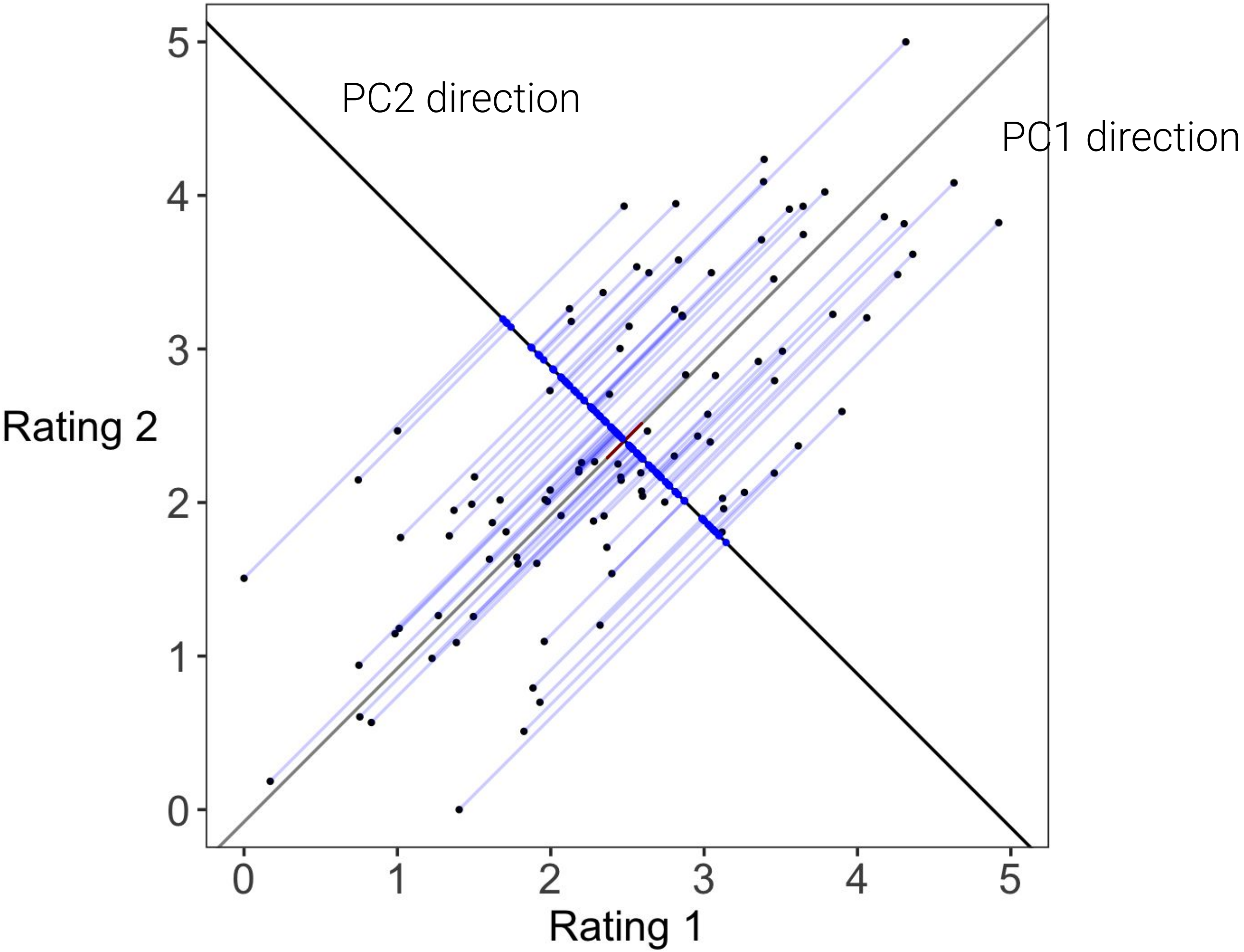
3844650



PC2 direction must be **uncorrelated** with PC1 (i.e., orthogonal)

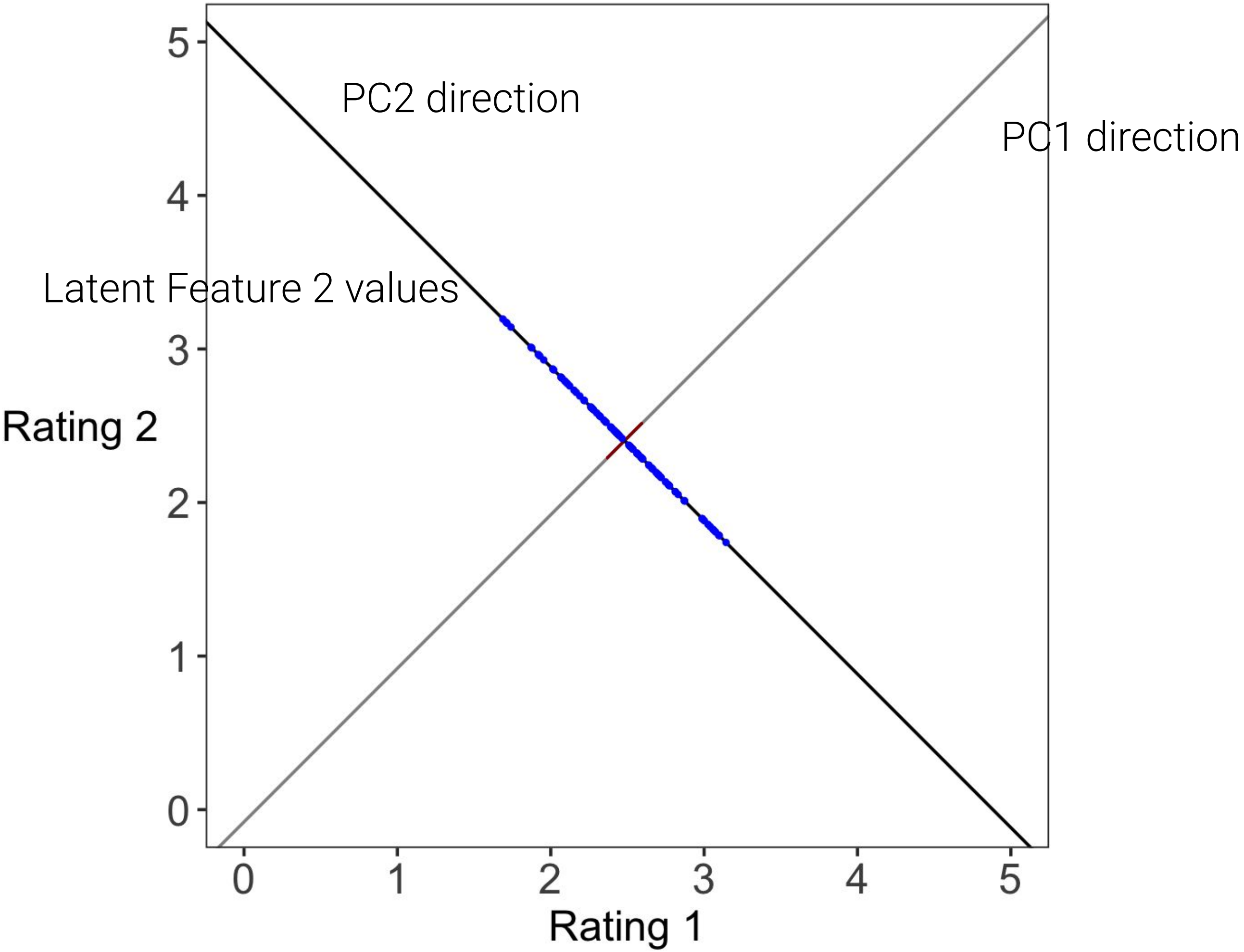


3844650


PC2 direction
PC1 direction
Rating 2
Rating 1
5
4
3
2
1
0
0
1
2
3
4
5

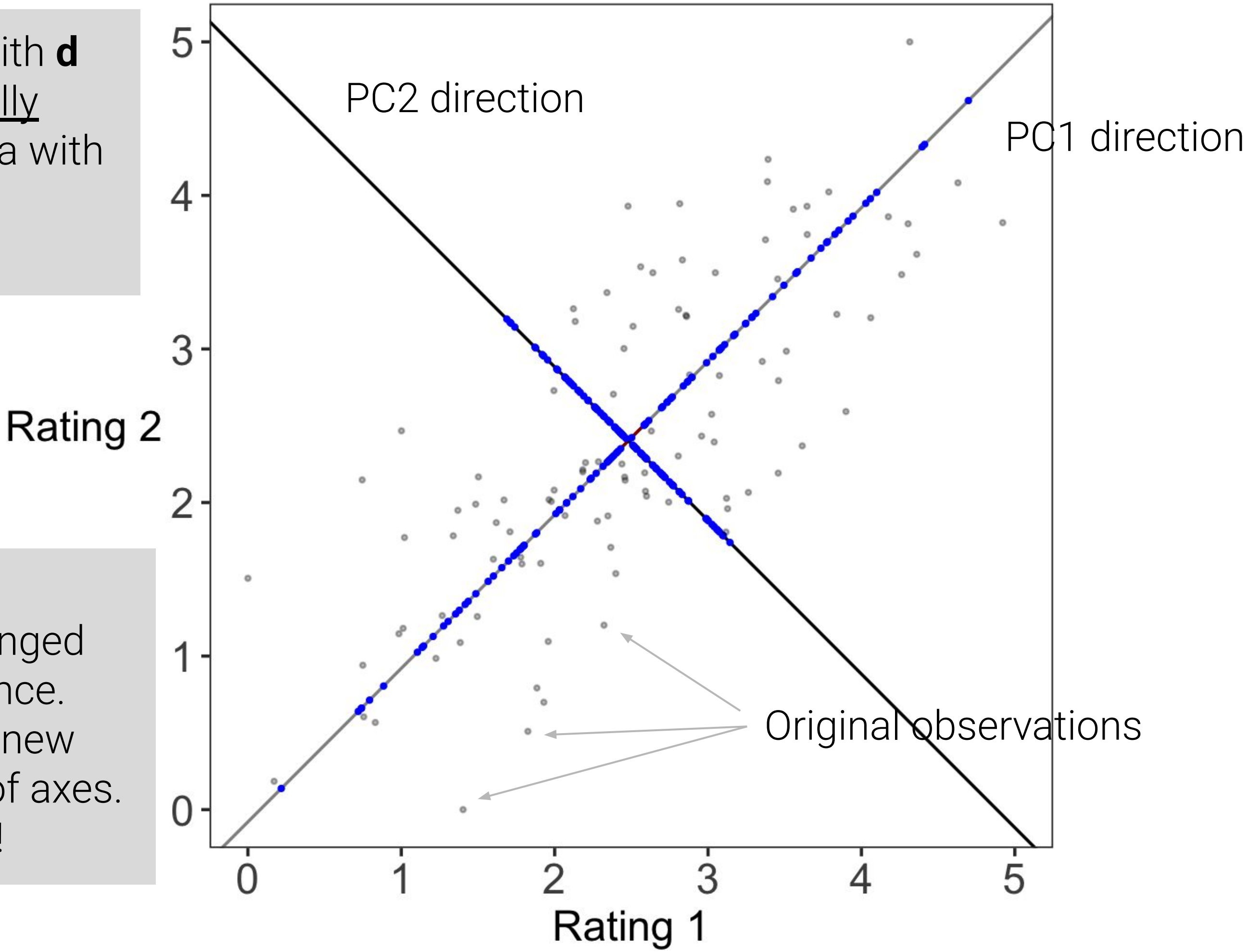
3844650

3844650



PC2 direction
PC1 direction
Latent Feature 2 values
Rating 2
Rating 1
0
1
2
3
4
5

For **n** data points with **d features**, we can fully reconstruct the data with **min(n,d)** principal components.



At this point, we've essentially just changed our frame of reference. PC1 and PC2 are a new (uncorrelated) set of axes. Turn your head 45^{o}!

3844650

Capturing Total Variance

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

Goal of PCA, restated:

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

3844650

Capturing Total Variance, Approach 1

3844650

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**

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        338.7731
dtype: float64
```

2. Keep the two attributes with highest variance.

$$\begin{bmatrix} W & L & A & P \end{bmatrix} \times \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} A & P \end{bmatrix}$$

Low-D Total Variance: 389.52. *Can we do better?*

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: PCA

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 **latent factors** (feature columns) were constructed by a **linear combinations of features** (using PCA)**.**

Total Variance of Original Data: **402.56**

Total Variance: 397.15.

Principal Component Analysis: A Procedural View

3844650

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

2. 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 **maximize the variance** along the projection onto v.
   - Choose 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.

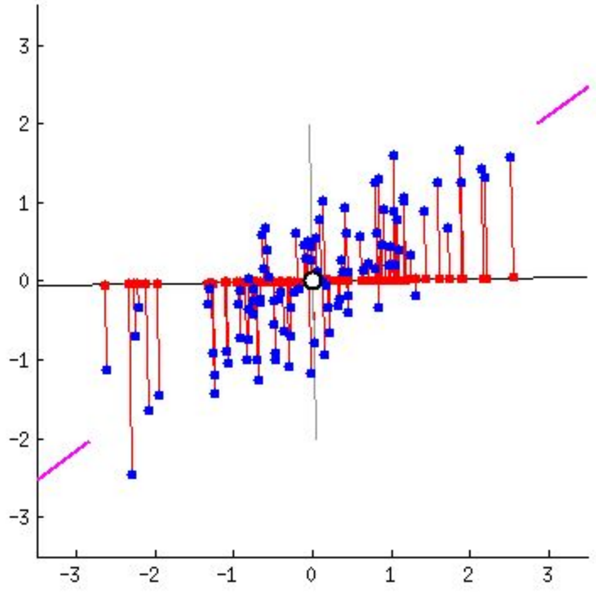
Principal Component Analysis: If you're curious

(out of scope)

3844650

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

2. 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 **maximize the variance** along the projection onto v.
   - Choose 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.

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

[StackExchange]

Principal Component Analysis: A Procedural View

3844650

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

2. 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 maximize the variance along the projection onto v.
   - Choose 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.

3844650

Deriving PCA as Error Minimization

Lecture 24, Data 100 Spring 2025

You are not expected to be able to redo the derivation in this section. However, understanding the derivation will make PCA more intuitive.

Recall: Covariance

3844650

Recall the definition of **covariance** (remember, very similar to **correlation**):

$$\mathrm{Cov}(X, Y) = E[(X - E[X])(Y - E[Y])]$$

Sample covariance is the estimated covariance of X and Y computed with a sample of size n:

$$\frac{1}{n}\sum_{i=1}^{n}(X_i - \mu_x)(Y_i - \mu_y)$$

The sample covariance if X and Y have mean=0:

$$\frac{1}{n}\sum_{i=1}^{n} X_i Y_i$$

If we **center** a vector (i.e., subtract the mean from each element), its new mean is 0.

3844650

The covariance matrix (**Σ**) of a feature matrix **X** where the features are **centered**:

d x 1   1 x d

$$\Sigma = \frac{1}{n}\sum_{i=1}^{n} X_i^T X_i$$

= **d x d matrix**!

Sample covariance of centered feature 1 and centered feature d

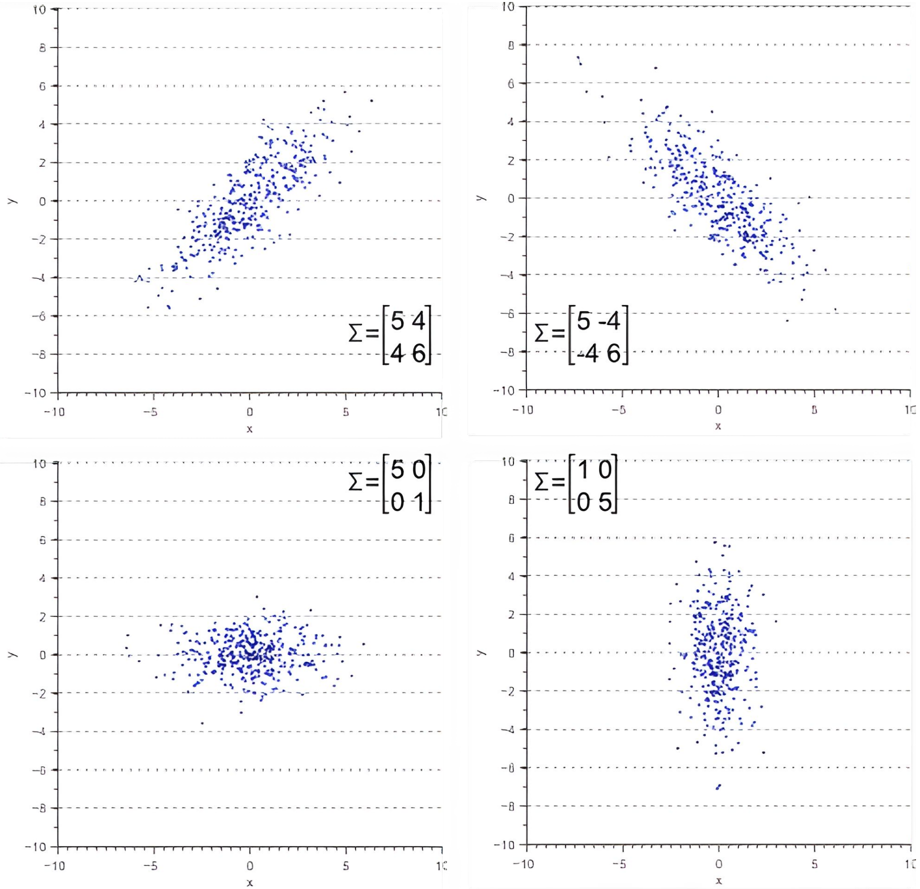
Symmetric matrix! $\Sigma^T = \Sigma$

Variance of centered feature 1

$$\frac{1}{n}\sum_{i=1}^{n}\begin{bmatrix} X_{i1}X_{i1} & X_{i1}X_{i2} & \cdots & X_{i1}X_{id} \\ X_{i2}X_{i1} & X_{i2}X_{i2} & \cdots & X_{i2}X_{id} \\ \vdots & \vdots & \ddots & \vdots \\ X_{id}X_{i1} & X_{id}X_{i2} & \cdots & X_{id}X_{id} \end{bmatrix}$$

Visual guide to covariance matrix

3844650



<u>Diagonal elements</u>: Variance of data along each dimension.

<u>Off-diagonal elements</u>: Covariance of different dimensions. Think "unscaled" correlation!

visiondummy.com/2014/04/geometric-interpretation-covariance-matrix

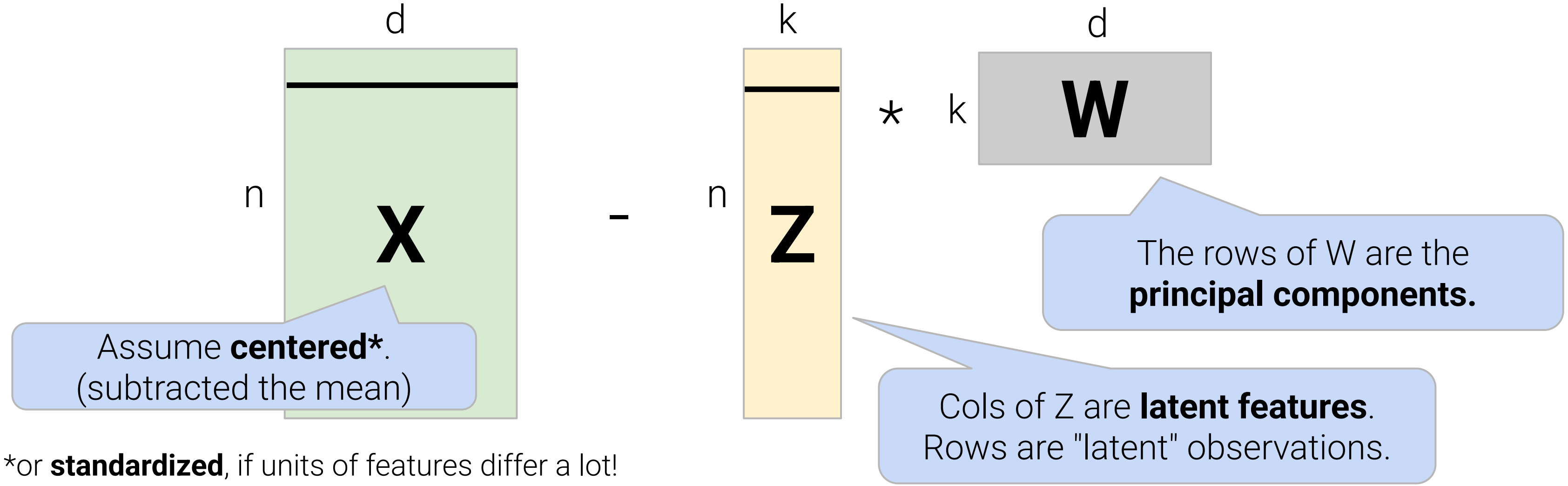
Derive PCA using Loss Minimization

Goal: Minimize the **reconstruction loss** of our **matrix factorization model**:

1 x d   1 x k   k x d

$$L(Z, W) = \frac{1}{n} \sum_{i=1}^{n} \|X_i - Z_i W\|^2$$

Row Vector   Row Vector



*or **standardized**, if units of features differ a lot!

Derive PCA using Loss Minimization

3844650

Goal: Minimize the **reconstruction loss** for our **matrix factorization model**:

$$L(Z, W) = \frac{1}{n}\sum_{i=1}^{n} \|X_i - Z_i W\|^2$$

$$= \frac{1}{n}\sum_{i=1}^{n} (X_i - Z_i W)(X_i - Z_i W)^T$$

Row Vector

Column Vector

Derive PCA using Loss Minimization

Goal: Minimize the reconstruction loss for our matrix factorization model:

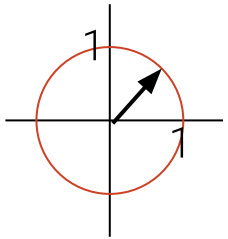
$$L(Z,W) = \frac{1}{n}\sum_{i=1}^{n}\left(X_i - Z_iW\right)\left(X_i - Z_iW\right)^T$$

Recall there are many solutions so **we constrain our model**:

- **W** is a row-orthonormal matrix **(i.e., $WW^T=I$)** where the **rows of W** are our **Principal Components (PCs)**. This ensures our PCs are uncorrelated (i.e., not redundant).

$$\boxed{\mathbf{W}} * \boxed{\mathbf{W}^T} = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & \ldots & \\ & & & & 1 \end{bmatrix}$$

$$\|w\|^2 = ww^T = 1$$



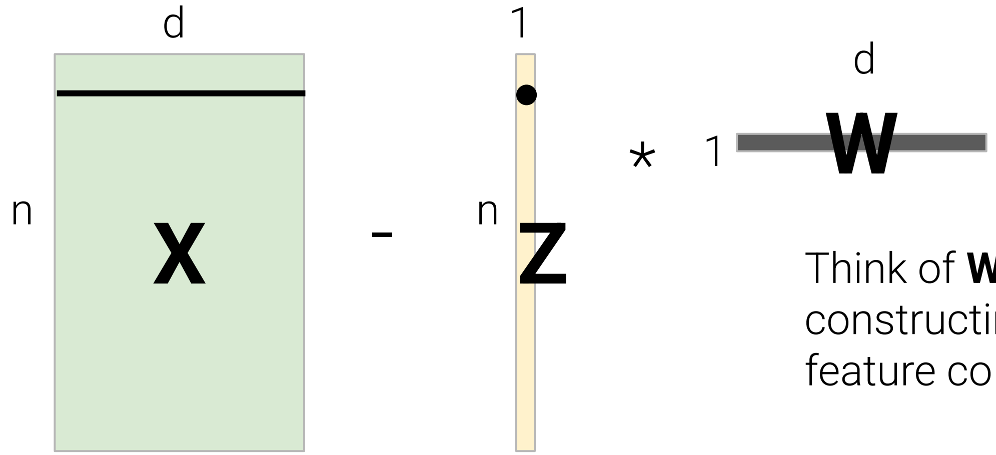
Row-orthonormal: Each row vector is length 1 and orthogonal to other rows

Simplified Derivation: consider (k=1)

3844650

Let consider the situation when k=1 (i.e., we construct only one latent feature):

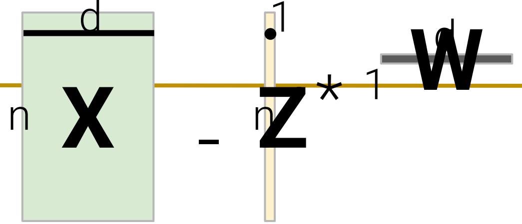
$$L(z, w) = \frac{1}{n} \sum_{i=1}^{n} (X_i - z_i w)(X_i - z_i w)^T$$



Think of **W** as the recipe for constructing **Z** from the **d** feature cols in **X**.

Derivation based on Kevin Murphy's derivation in the excellent PML Textbook.

Simplified Derivation: Simplify the Loss



3844650

Let consider the situation when k=1:

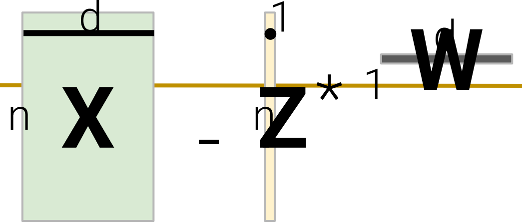
$$L(z,w) = \frac{1}{n}\sum_{i=1}^{n}\left(X_i - z_i w\right)\left(X_i - z_i w\right)^T$$

Expanding the loss:

$$L(z,w) = \frac{1}{n}\sum_{i=1}^{n}\left(\underbrace{X_i X_i^T}_{\text{Constant (ignore)}} - 2 z_i X_i w^T + z_i^2 \underbrace{w w^T}_{=1 \text{ by orthonormality}}\right)$$

$$= \frac{1}{n}\sum_{i=1}^{n}\left(-2 z_i X_i w^T + z_i^2\right)$$

Simplified Derivation: Optimizing for z



3844650

Let consider the situation when k=1:

$$L(z, w) = \frac{1}{n} \sum_{i=1}^{n} \left(-2 z_i X_i w^T + z_i^2\right)$$

Taking the derivative with respect to z_i:

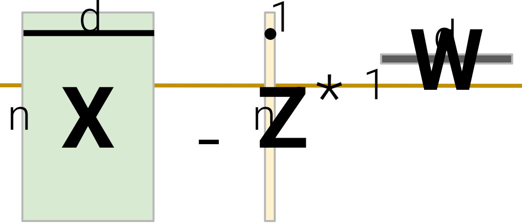
$$\frac{\partial}{\partial z_i} L(z, w) = \frac{1}{n} \left(-2 X_i w^T + 2 z_i\right)$$

Setting the derivative equal to 0 and solving for z_i:

$$z_i = X_i w^T$$

We can compute z by **projecting onto w**

Simplified Derivation: Substituting soln for z



Substituting the solution for z: $z_i = X_i w^T$

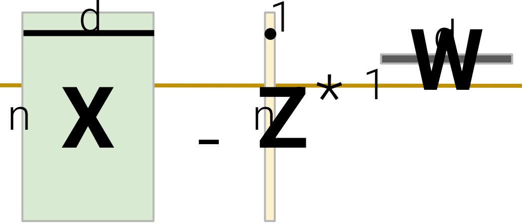
$$L(z, w) = \frac{1}{n} \sum_{i=1}^{n} \left(-2 z_i X_i w^T + z_i^2\right)$$

$$L(z = Xw^T, w) = \frac{1}{n} \sum_{i=1}^{n} \left(-2 X_i w^T X_i w^T + \left(X_i w^T\right)^2\right)$$

Algebra: $$= \frac{1}{n} \sum_{i=1}^{n} \left(-X_i w^T X_i w^T\right) = \frac{1}{n} \sum_{i=1}^{n} \left(-w X_i^T X_i w^T\right)$$

Definition of Cov (Σ): $$= -w \frac{1}{n} \sum_{i=1}^{n} \left(X_i^T X_i\right) w^T = -w \Sigma w^T$$

Simplified Derivation: Substituting soln for z



3844650

Substituting the solution for z: $z_i = X_i w^T$

$$L(z, w) = \frac{1}{n}\sum_{i=1}^{n}\left(-2 z_i X_i w^T + z_i^2\right)$$

$$L(z = Xw^T, w) = \frac{1}{n}\sum_{i=1}^{n}\left(-2 X_i w^T X_i w^T + \left(X_i w^T\right)^2\right)$$

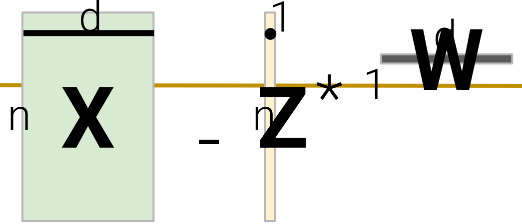
Algebra: $$= \frac{1}{n}\sum_{i=1}^{n}\left(-X_i w^T X_i w^T\right) = \frac{1}{n}\sum_{i=1}^{n}\left(-w X_i^T X_i w^T\right)$$

Definition of Cov (Σ): $$= -w\frac{1}{n}\sum_{i=1}^{n}\left(X_i^T X_i\right) w^T = -w\Sigma w^T$$

Lecture 24 ended here!

(We finished 10 minutes early)

Simplified Derivation: Optimizing for w



3844650

Minimize the loss with respect to w:

$$L(w) = -w\Sigma w^T$$

Make **w really big** (to infinity) … but we have the **orthonormality constraint ww^T=1**

Use [Lagrange multiplier λ]() to introduce the constraint **ww^T=1** to our optimization problem:

$$L(w, \lambda) = -w\Sigma w^T + \lambda \left(ww^T - 1 \right)$$

Intuition for Lagrange multiplier: When we take partial derivative with respect to λ and set equal to 0 to minimize L, we will recover the constraint:

$$\frac{\partial}{\partial \lambda} L(w, \lambda) = ww^T - 1 = 0$$

Simplified Derivation: Optimizing for w

3844650

$$L(w, \lambda) = -w\Sigma w^T + \lambda\left(ww^T - 1\right)$$

Take **derivative with respect to w** (vector calculus – out of scope for Data 100)**:**

$$\frac{\partial}{\partial w}\left(-w\Sigma w^T + \lambda\left(ww^T - 1\right)\right) = -2\Sigma w^T + 2\lambda w^T$$

Think of ww^T like squaring → Derivative is $2w^T$

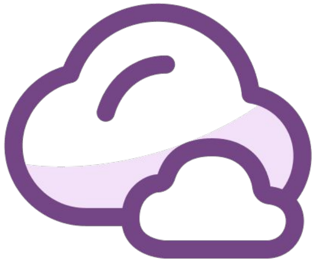
Setting equal to zero: $-2\Sigma w^T + 2\lambda w^T = 0$

$$\boxed{\Sigma w^T = \lambda w^T}$$

What is this?

Remember that λ is a scalar!

3844650



What is this?

ⓘ Click **Present with Slido** or install our Chrome extension to activate this poll while presenting.

Simplified Derivation: Optimizing for w

3844650

$$L(w, \lambda) = -w\Sigma w^T + \lambda \left(ww^T - 1\right)$$

Eigenvalue of **Σ**

$$\Sigma w^T = \lambda w^T$$

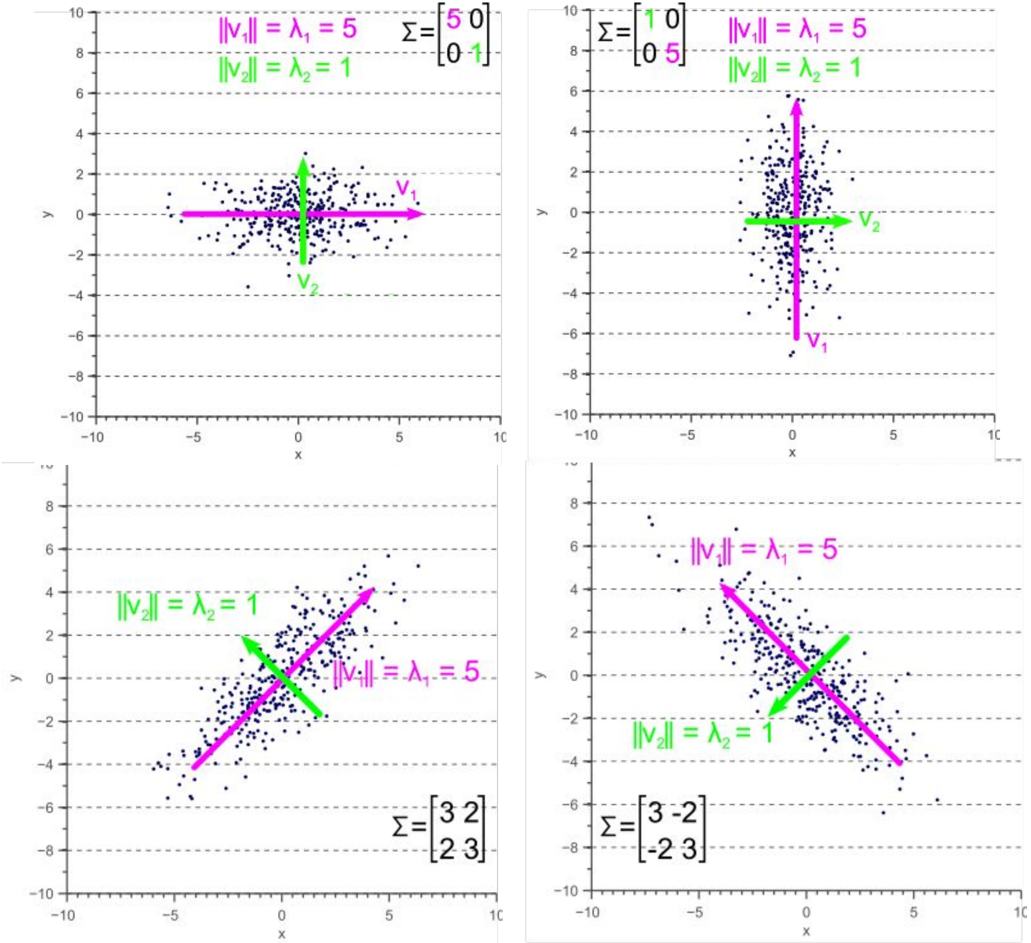
Eigenvector of **Σ**

w is a **unit** (i.e., length 1) **eigenvector** of the **covariance matrix.** When we multiply a matrix by one of its eigenvectors, it's equivalent to multiplying the eigenvector by its (scalar) eigenvalue.

L(w,λ) is **minimized** when $w\Sigma w^T = w\lambda w^T = \lambda ww^T = \lambda$ **is big**. So, the **optimal w** is the eigenvector with the **largest eigenvalue λ.**

In other words, the **optimal w** points in the direction of the **greatest variance** of the data (PC1!), and λ is the variance in that direction (so long as X is centered!).

Eigenvectors and eigenvalues of covariance matrix



3844650

Same data rotated four ways.

$v_1/5$ is PC1. **Unit-length** eigenvector with largest eigenvalue points in direction of maximum variance. Eigenvalue (λ_1=5) is variance.

$v_2/1$ is PC2. Unit-length eigenvector with second largest eigenvalue points in direction of maximum variance that is **orthogonal** to first eigenvector.

visiondummy.com/2014/04/geometric-interpretation-covariance-matrix

Extending the Derivation to the Second PC (Bonus)

3844650

We can extend the derivation inductively to the next principal component:

$$\frac{\partial}{\partial w_2} L(w_2, \lambda_2, \lambda_{12}) = -w_2 \Sigma w_2^T + \lambda_2 \left(w_2 w_2^T - 1\right) + \underbrace{\lambda_{12} \left(w_1 w_2^T - 0\right)}_{\textbf{Orthogonality Constraint}}$$

Taking the derivative with respect to w_2:

$$\frac{\partial}{\partial w_2} L(w_2, \lambda_2, \lambda_{12}) = -2\Sigma w_2^T + 2\lambda_2 w_2^T + \lambda_{12} w_1^T$$

Set equal to 0 and left multiply by w_1:

$$-2\underbrace{\underbrace{w_1 \Sigma}_{\lambda w_1} w_2^T}_{\mathbf{0}} + 2\lambda_2 \underbrace{w_1 w_2^T}_{\mathbf{0}} + \lambda_{12} \underbrace{w_1 w_1^T}_{\mathbf{1}} = 0$$

$$\Rightarrow \lambda_{12} = 0$$

Extending the Derivation to the Second PC (Bonus)

3844650

We can extend the derivation inductively to the next principal component:

$$\frac{\partial}{\partial w_2} L(w_2, \lambda_2, \lambda_{12}) = -w_2 \Sigma w_2^T + \lambda_2 \left(w_2 w_2^T - 1 \right) + \underbrace{\lambda_{12} \left(w_1 w_2^T - 0 \right)}_{\textbf{Orthogonality Constraint}}$$

Taking the derivative with respect to w_2:

$$\frac{\partial}{\partial w_2} L(w_2, \lambda_2, \lambda_{12}) = \boxed{-2\Sigma w_2^T + 2\lambda_2 w_2^T} + \cancel{\lambda_{12} w_1^T}$$

Set equal to 0 and left multiply by w_1:

$$-2\underbrace{\underbrace{w_1 \Sigma}_{\lambda w_1} w_2^T}_{0} + 2\lambda_2 \underbrace{w_1 w_2^T}_{0} + \lambda_{12} \underbrace{w_1 w_1^T}_{1} = 0$$

$$\Rightarrow \lambda_{12} = 0$$

Discussing the Main Result

3844650

$$\Sigma w^T = \lambda w^T$$

This implies that:

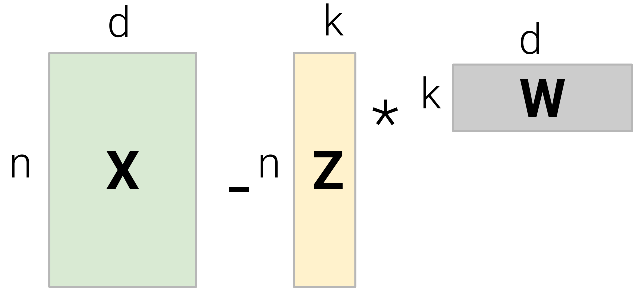
1. w is a **unit eigenvector** of the **covariance matrix** and
2. the **error is minimized** when w is the eigenvector with the **largest eigenvalue** $\boldsymbol{\lambda}$

Covariance Matrix for X

Eigenvalue

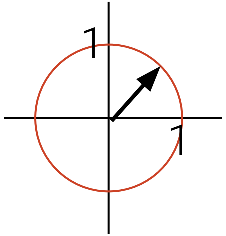
Eigenvalue Equation: $\Sigma w^T = \lambda w^T$

Eigenvector



Unitary constraint:

$$\|w\|^2 = ww^T = 1$$

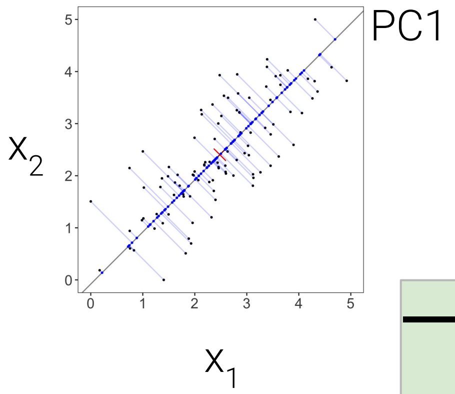


Recall: w is a row vector in our current derivation.

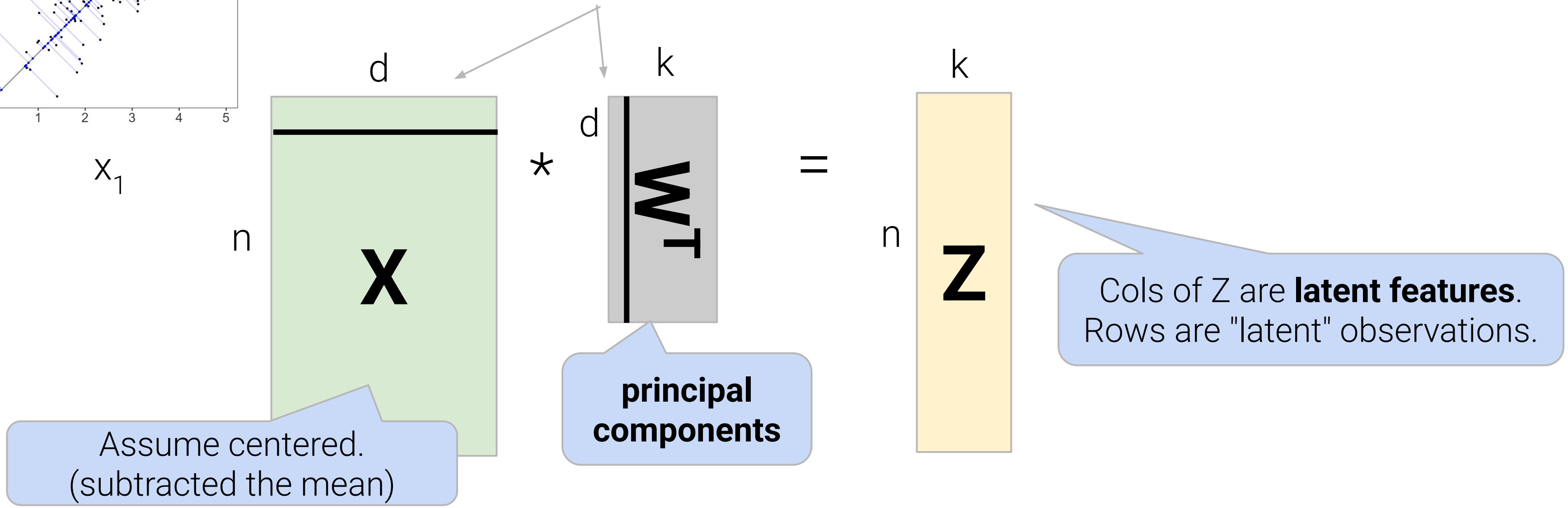
Take Away from the Optimization Framing

3844650

The **principal components (PCs)** are the **unit eigenvectors** of the **covariance matrix** with the **largest eigenvalues.** These are the directions of **maximum variance** in the data.

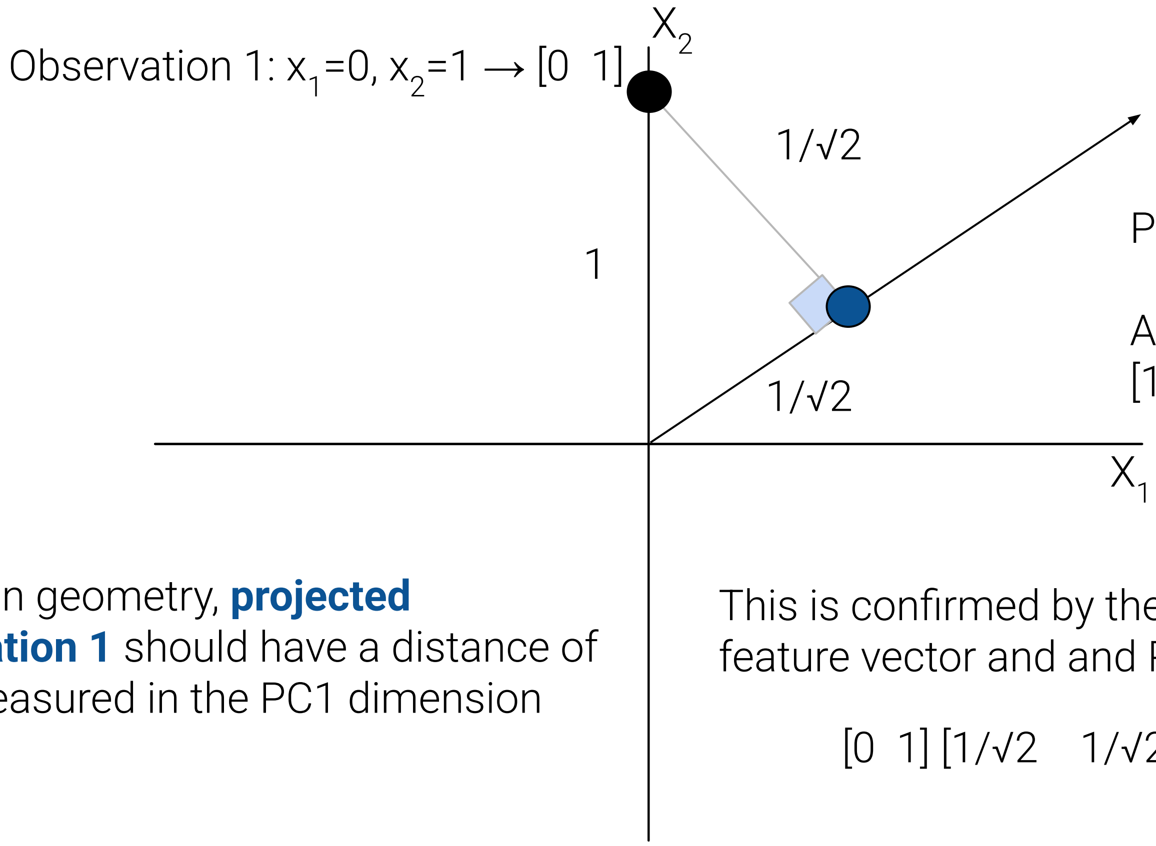


The dot product of each row in X and column in W^T represents a projection of a $(X_1,\ldots,X_d)$ data point onto the PC dimension given by the column.



*or standardized, if units of features differ a lot!

Geometry of projecting a point onto a PC dimension



PC1 direction: $[1\ 1]^T$

As a unit vector:
$[1/\sqrt{2} \quad 1/\sqrt{2}]^T$

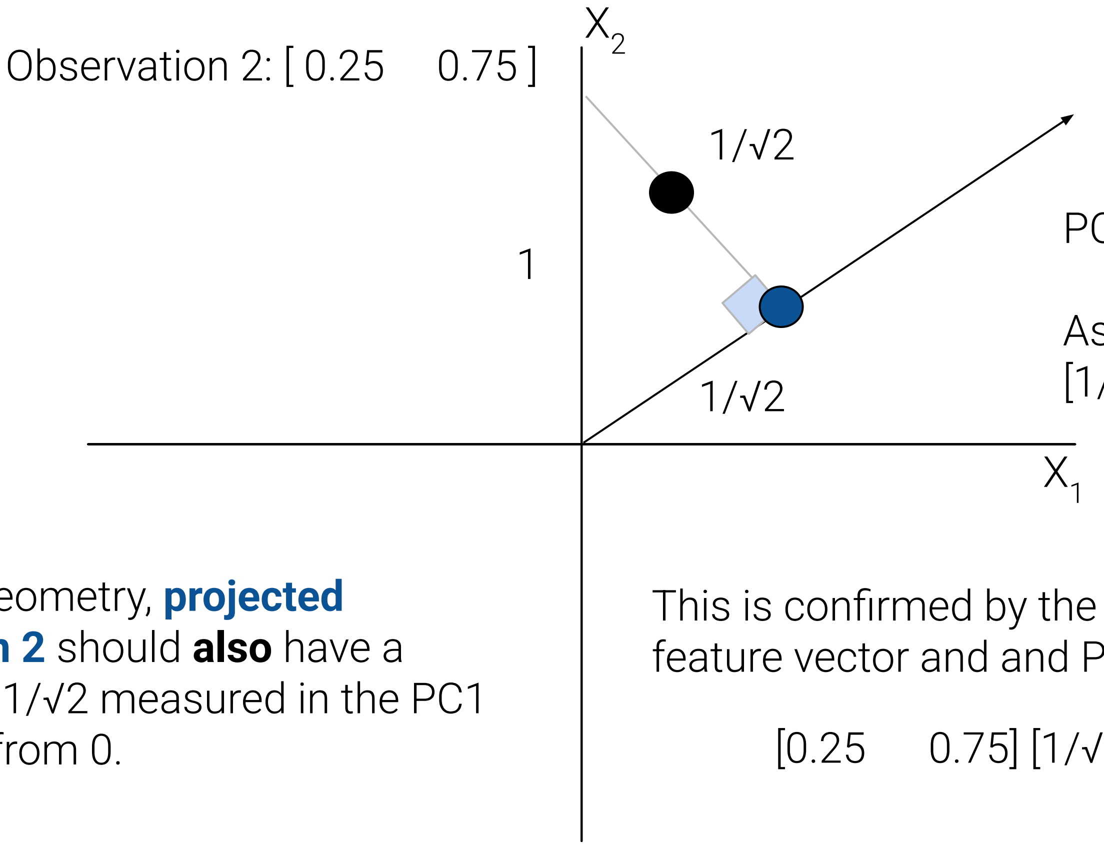
Based on geometry, **projected observation 1** should have a distance of $1/\sqrt{2}$ measured in the PC1 dimension from 0.

This is confirmed by the **dot product** of the feature vector and and PC1 unit vector.

$$[0\ \ 1]\,[1/\sqrt{2} \quad 1/\sqrt{2}]^T = 1/\sqrt{2}$$

Intuition for why XW^T projects the observations onto the PC vectors given by W^T.

Geometry of projecting a point onto a PC dimension



PC1 direction: $[1 \; 1]^T$

As a unit vector:
$[1/\sqrt{2} \quad 1/\sqrt{2}]^T$

Based on geometry, **projected observation 2** should **also** have a distance of $1/\sqrt{2}$ measured in the PC1 dimension from 0.

This is confirmed by the **dot product** of the feature vector and and PC1 unit vector.

$$[0.25 \quad 0.75] \, [1/\sqrt{2} \quad 1/\sqrt{2}]^T = 1/\sqrt{2}$$

Intuition for why XW^T projects the observations onto the PC vectors given by W^T.

3844650

LECTURE 24

PCA I

Content credit: Acknowledgments