1565862

slido

Join at slido.com #1565862

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

1565862

LECTURE 25

Principal Component Analysis II

PCA: A technique for EDA and feature generation.

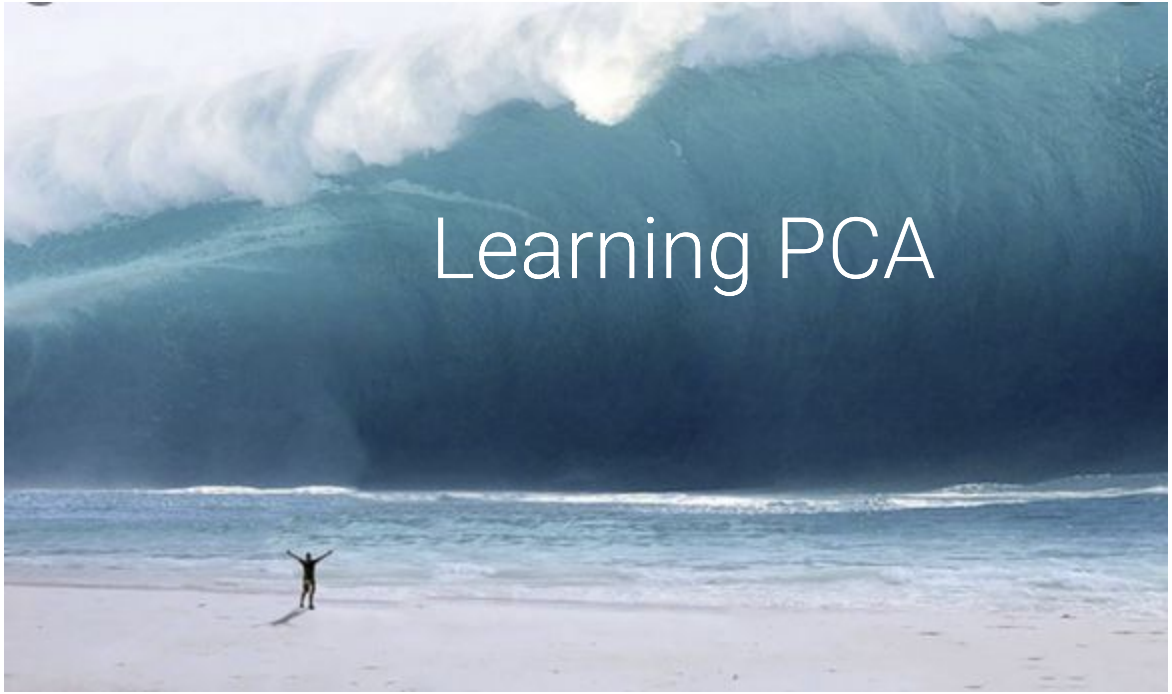
Data 100/Data 200, Spring 2025 @ UC Berkeley

Narges Norouzi and Josh Grossman

Content credit: Acknowledgments

Incoming Tsunami of Conceptual Challenges!

1565862



Be kind to yourself 💙

1565862

PCA as Loss Minimization

Singular Value Decomposition

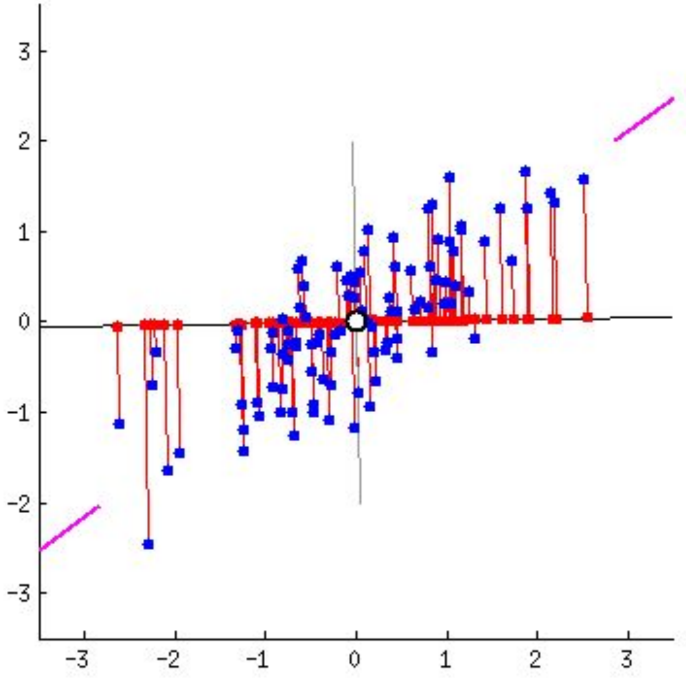
PCA with SVD

Centering Data and Computing Variance

PCA as Loss Minimization

Lecture 25, Data 100 Spring 2025

Useful reference for this lecture:
stats.stackexchange.com/questions/134282/relationship-between-svd-and-pca-how-to-use-svd-to-perform-pca

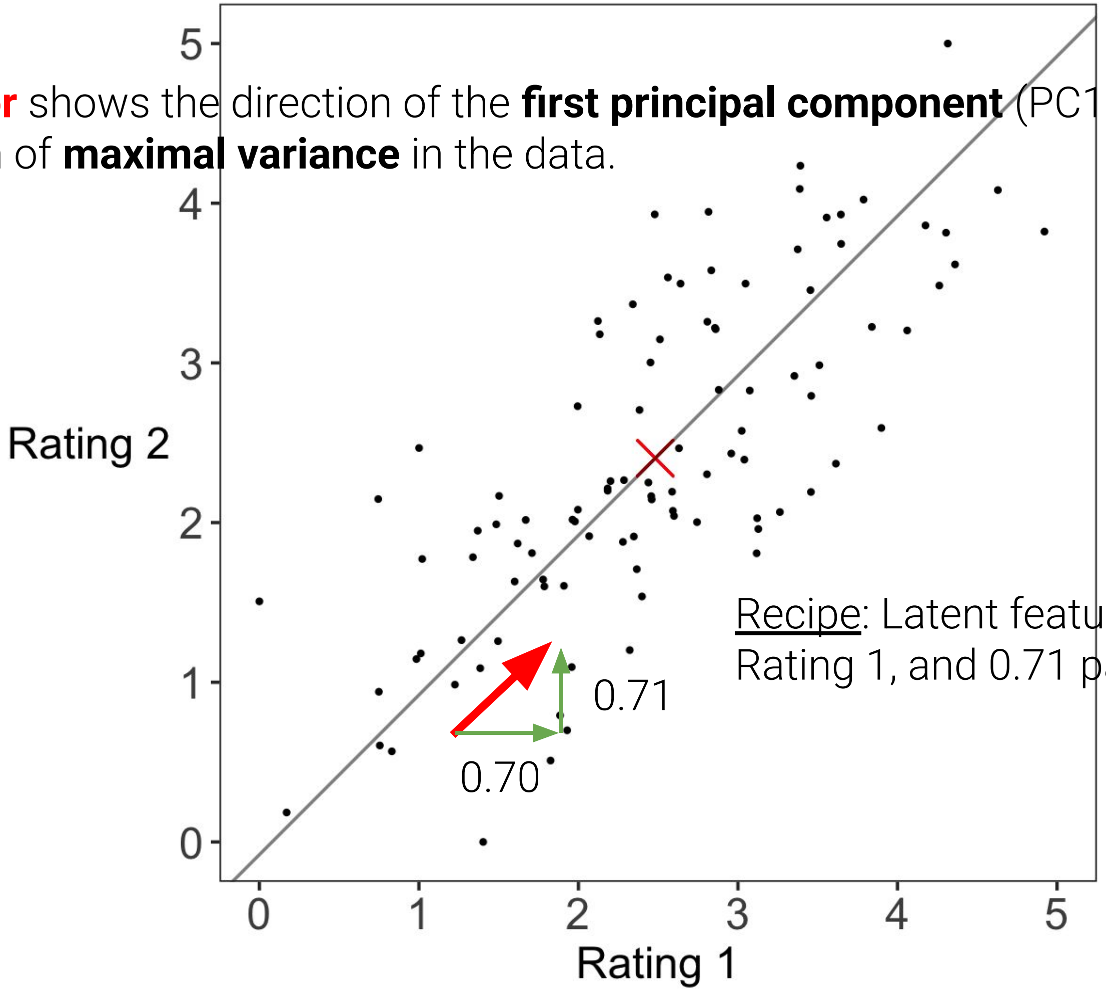


Maximizing variance = **Spreading out red dots**

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

[StackExchange]

This **length 1 vector** shows the direction of the **first principal component** (PC1).
This is the direction of **maximal variance** in the data.

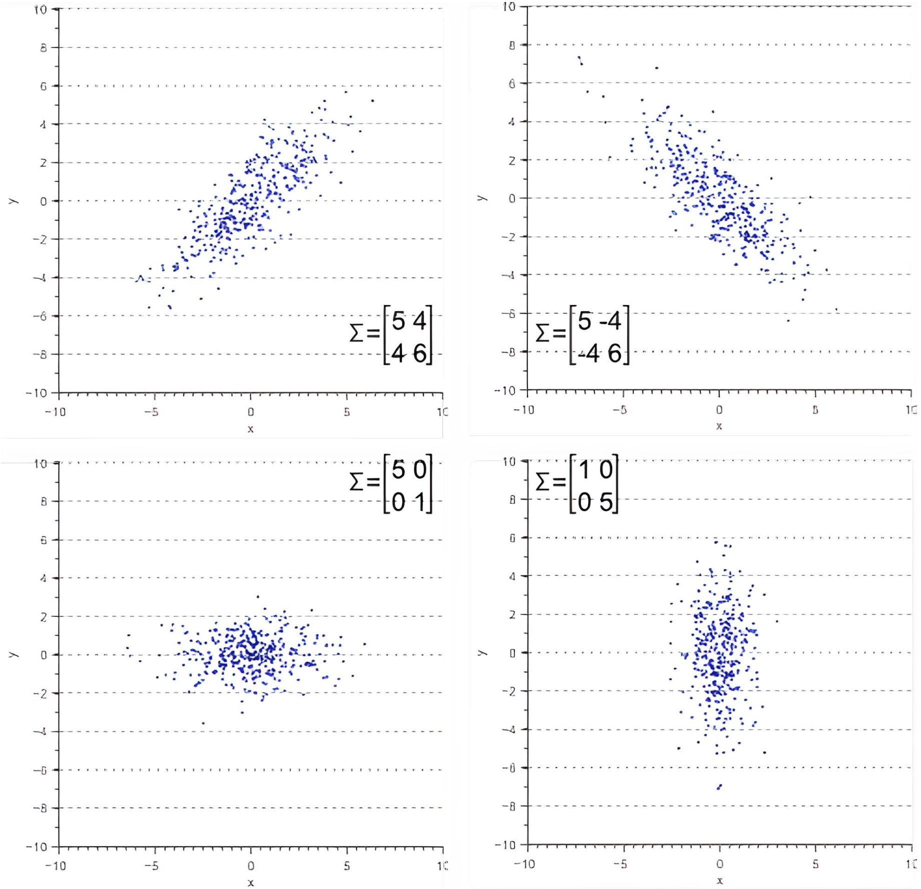


Recipe: Latent feature 1 is 0.70 parts Rating 1, and 0.71 parts Rating 2.

1565862

Visual guide to covariance matrix

1565862



<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

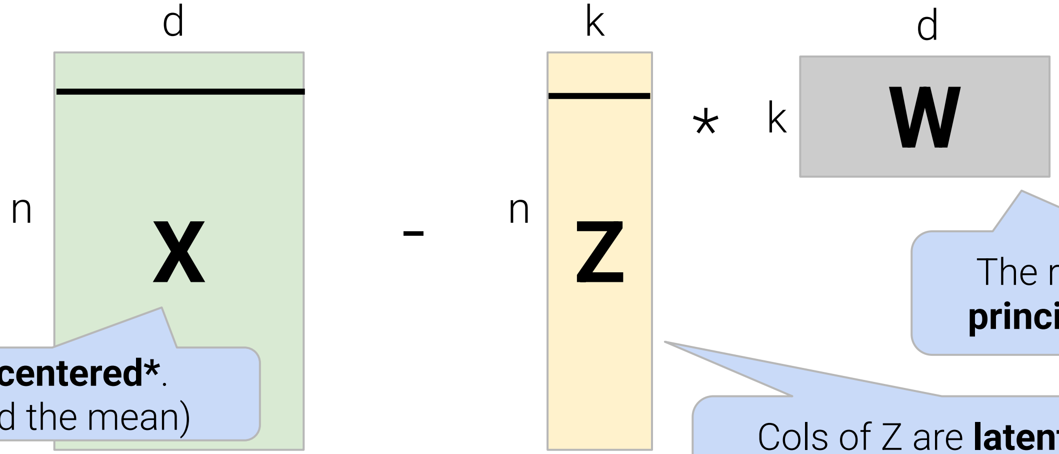
1565862

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

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

Row Vector

Row Vector



The rows of W are the **principal components.**

Assume **centered***.
(subtracted the mean)

Cols of Z are **latent features**.
Rows are "latent" observations.

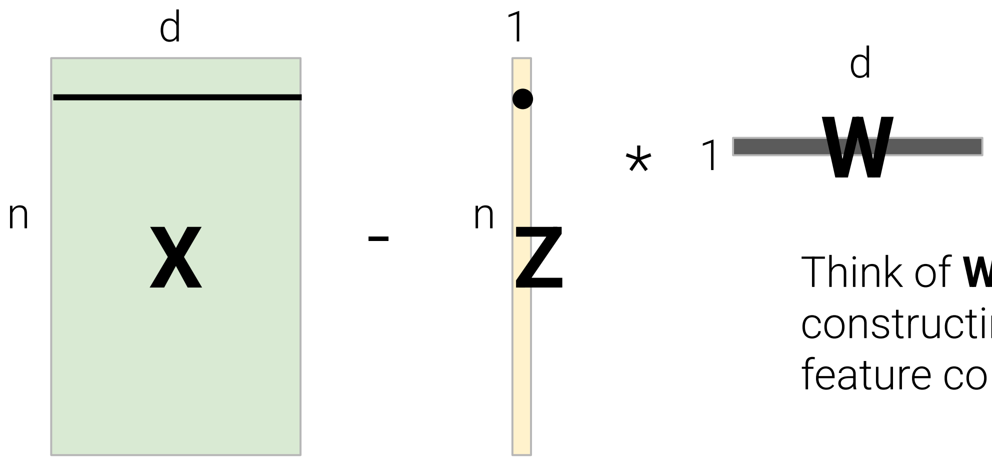
*or **standardized**, if units of features differ a lot!

Simplified Derivation: consider (k=1)

1565862

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

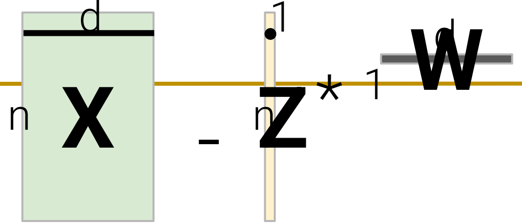
$$L(z, w) = \frac{1}{n} \sum_{i=1}^{n} \left(X_i - z_i w\right) \left(X_i - z_i w\right)^T$$



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: Optimizing for w



1565862

Minimize the loss with respect to w:

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

i.e., make w length 1

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 the partial derivative with respect to λ and set equal to 0 to minimize L, we will recover the constraint:

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

See skipped slides of Lecture 24 to see how the covariance matrix (Σ) shows up!

Simplified Derivation: Optimizing for w

1565862

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

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: $-2\Sigma w^T + 2\lambda w^T = 0$

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

What is this?

Remember that λ is a scalar!

1565862

What is this?

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

Simplified Derivation: Optimizing for w

1565862

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

Eigenvalue of **Σ**

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

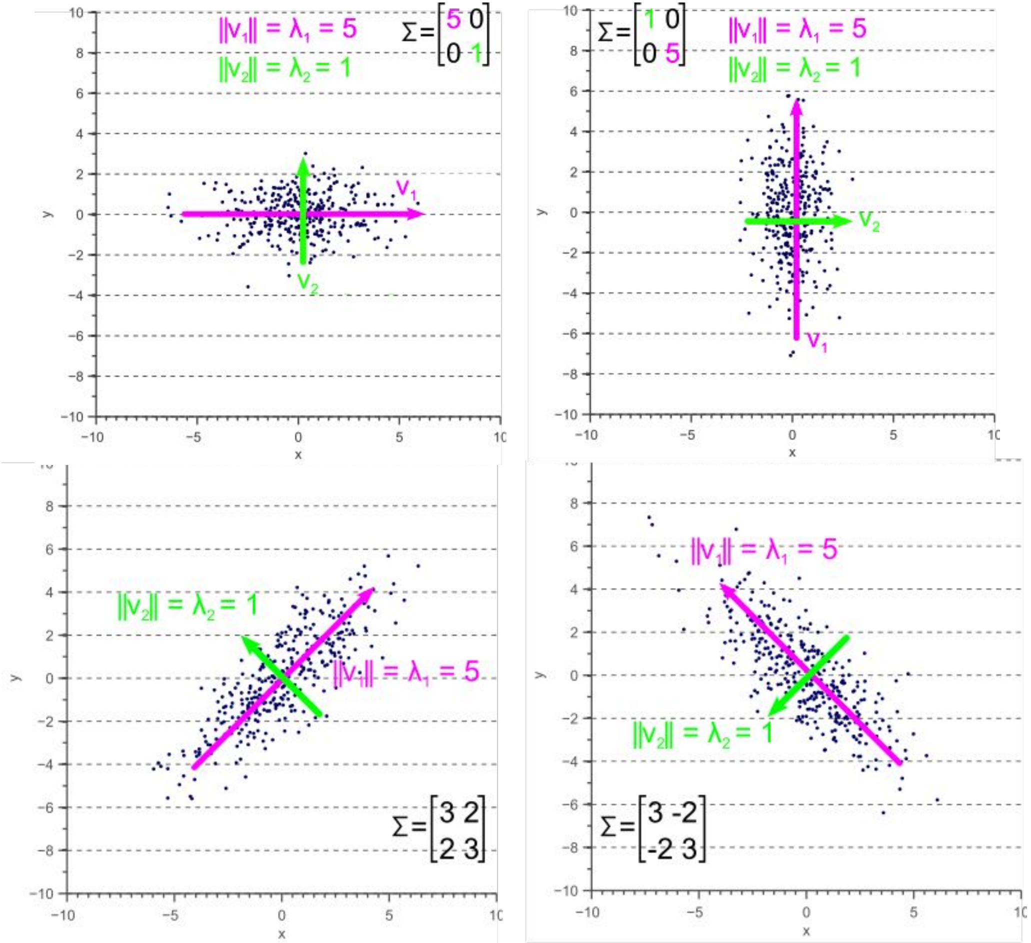
Eigenvector of **Σ**

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 $w\Sigma w^T = w\lambda w^T = \lambda ww^T = \boldsymbol{\lambda}$ **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



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

1565862

Singular Value Decomposition

Lecture 25, Data 100 Spring 2025

1565862

Useful reference for this lecture:
stats.stackexchange.com/questions/134282/relationship-between-svd-and-pca-how-to-use-svd-to-perform-pca

1565862

Singular value decomposition (**SVD**) decomposes a matrix into a product of three matrices.

- 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:
  - **How** the SVD is computed
  - **Why** SVD is a valid decomposition of rectangular matrices

Today, we will only cover **what** is needed for PCA.

Checkout the EE16 notes for a deeper study of SVD:
https://eecs16b.org/notes/sp24/note15.pdf

Singular Value Decomposition

1565862

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

$$X = U S V^T$$

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

rank r ≤ d

Assume d < n.

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

columns of U are **orthonormal** (i.e., unit length and orthogonal)

Columns of U are eigenvectors of XX^T

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

diagonal matrix of **singular values**, ordered from **largest to smallest.** **r** positive singular values. Remaining **d-r** singular values are 0.

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

columns+rows of V are **orthonormal**

Columns of V (rows of V^T) are **eigenvectors of X^TX**

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

Looks like covariance matrix without 1/n!

There are technically infinite possible factorizations, but if singular values are distinct + in decreasing order then solution space is constrained to shuffling columns of U and V.

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**
- $UU^T = I_n$ and $U^TU = I_d$
- Can think of U as a rotation

Eigenvectors of XX^T:

$$XX^T = USV^T (USV^T)^T = USV^T V S^T U^T$$
$$= USS^T U^T$$

Right multiply by U:

$$XX^T U = US^T S U^T U$$
$$= USS^T$$

SVD One-by-One: S

1565862

$$S$$

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

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

$$d\left\{\underbrace{\begin{bmatrix} s_1 & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & s_2 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & s_r & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 \end{bmatrix}}_{d}\right.$$

- r is the rank of X
- The majority of the matrix is zero
- The **singular values s_1, s_2, …, s_r** are **non-negative** and **sorted $s_1 \geq s_2 \geq \dots \geq s_r$**
- Can think of S 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
- Rows of V are also orthonormal
- Columns of V are called the **right singular vectors**
- $VV^T = V^TV = I_d$
- Can think of V as a rotation

Eigenvectors of X^TX:

$$X^TX = (USV^T)^T USV^T = VS^TU^TUSV^T$$

$$X^TX = VS^TSV^T$$

Right multiply by V: ⇨

$$X^TXV = VS^TSV^TV$$

$$= VS^TS$$

NumPy SVD

1565862

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

[documentation]

$\underbrace{\texttt{full\_matrices = False}}_{U \in \mathbb{R}^{n \times d}}$

"economy" or "compact" SVD

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

width	height	area	Perim.
2.97	1.35	24.78	8.64
-3.03	-0.65	-15.22	-7.36
-4.03	-1.65	-20.22	-11.36
3.97	-1.65	3.78	4.64
3.97	3.35	48.78	14.64
-2.03	-3.65	-20.22	-11.36
-1.03	-2.65	-15.22	-7.36
0.97	0.35	6.78	2.64
1.97	-3.65	-16.22	-3.36
2.97	-2.65	-7.22	0.64
...	...	...	...

n x d

-0.13	0.01	0.03	-0.21
0.09	-0.08	0.01	0.56
0.12	-0.13	0.09	-0.07
-0.03	0.18	0.01	-0.05
-0.26	-0.09	0.09	-0.06
0.12	-0.05	0.17	-0.05
0.09	0	0.1	-0.08
-0.04	0.01	0	-0.08
0.08	0.18	0.04	-0.05
0.03	0.19	0.02	-0.05
...	...	...	...

n x d

197.39			
	27.43		
		23.26	
			0

d x d

-0.1	-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

d x d

X is therefore **rank 3**.

From SVD to PCA

1565862

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 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 compute the **principal components**.

PCA with SVD

Lecture 25, Data 100 Spring 2025

1565862

PCA from SVD

1565862

Assume we have constructed the Singular Value Decomposition (SVD) of X:

$$X = USV^T$$

If X is **centered**, the **covariance matrix (Σ)** of X is:

PCA from SVD

1565862

Assume we have constructed the Singular Value Decomposition (SVD) of X:

$$X = USV^T$$

If X is **centered**, the **covariance matrix (Σ)** of X is:

$$\Sigma = (1/n)\ X^T X$$

$$= (1/n)\ (USV^T)^T USV^T$$ Substitution.

$$= (1/n)\ VS^T U^T USV^T$$ Matrix transpose rule. $(AB)^T = B^T A^T$

$$= (1/n)\ VS^T SV^T$$ U is orthonormal. $U^T U = \mathbf{I}$

$$= (1/n)\ VS^2 V^T$$ S is diagonal. Square diagonal elements.

PCA from SVD

1565862

Assume we have constructed the Singular Value Decomposition (SVD) of X:

$$X = USV^T$$

If X is **centered**, the **covariance matrix (Σ)** of X is:

PCA from SVD

1565862

Assume we have constructed the Singular Value Decomposition (SVD) of X:

$$X = USV^T$$

If X is **centered**, the **covariance matrix (Σ)** of X is:

$$\Sigma = (1/n)\ VS^2V^T$$

Where we left off.

$$\Sigma V = (1/n)\ VS^2V^TV$$

Right multiply by V.

$$\Sigma V = \frac{S^2}{n}V$$

V is orthonormal. $V^TV = \mathbf{I}$. S^2 can be moved.

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

Looks like our loss minimization solution! ✅

Principal Components (PCs) are the Eigenvectors of the Covariance Matrix

1565862

Assume we have constructed the Singular Value Decomposition (SVD) of X:

$$X = USV^T$$

$$\Sigma V = \frac{S^2}{n} V$$

The **columns of V** (rows of V^T) are the **eigenvectors** of the **covariance matrix** (**Σ**) and therefore the **PCs.** Orthogonal **directions** of greatest variance of the data.

The (**singular values**)^2 **/ n** are the **eigenvalues** of **Σ.** **Variance** of the data in each of the directions given by the PCs.

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

← Same as definitions of corresponding terms (earlier slide)

Computing Latent Vectors Using X * V

1565862

Constructing a 2 PC approximation (k=2). PCs as a "recipe" to make Z from features in X.

$$X \quad * \quad V \quad = \quad Z$$

width	height	area	Perim.
2.97	1.35	24.78	8.64
-3.03	-0.65	-15.22	-7.36
-4.03	-1.65	-20.22	-11.36
3.97	-1.65	3.78	4.64
3.97	3.35	48.78	14.64
-2.03	-3.65	-20.22	-11.36
…	…	…	…

PC1	PC2		
-0.1	0.67	0.31	0.67
-0.07	-0.37	-0.64	0.67
-0.93	-0.26	0.26	0
-0.34	0.59	-0.65	-0.33

-26.43	0.16
17.05	-2.18
23.25	-3.54
-5.38	5.03
-51.09	-2.59
23.19	-1.45
…	…

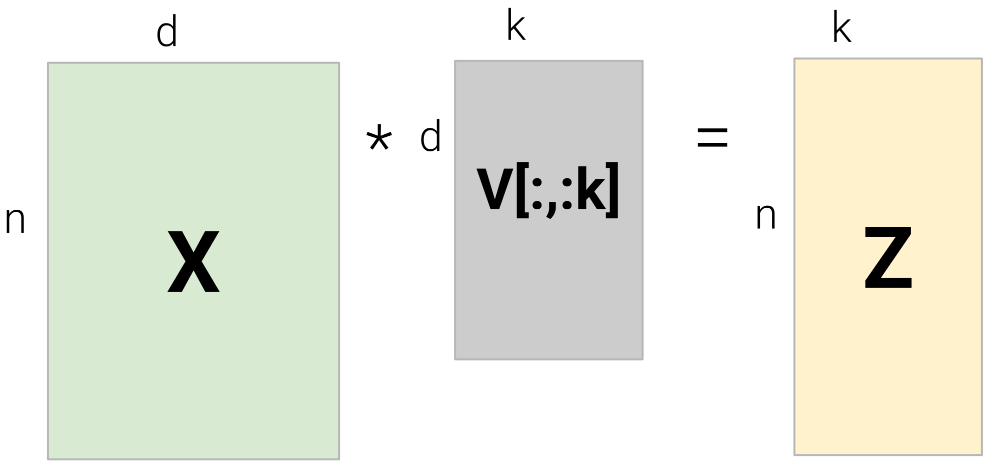
From SVD to PCA

1565862

We have now shown that if we construct the singular value decomposition of X:

$$X = USV^T$$

The **first k** columns of V (rows of V^T) are the **first k PCs**. We construct the **latent features (Z)** of X by projecting X onto the **k PCs**. We choose k! Often, k=2.



There is another way to compute Z**:**

PC "Recipe" of features

$$Z = XV \quad = USV^T V$$

$$= US$$

U contains normalized columns of Z. Scale up by S.

Dot products can be interpreted as projections → See end of Lecture 24.

Computing Latent Vectors Using U * S

1565862

Equivalent construction of Z! **U cols are normalized Z cols**. S "scales up" U cols to Z cols.

$$U \quad * \quad S \quad = \quad Z$$

-0.13	0.01	0.03	-0.21
0.09	-0.08	0.01	0.56
0.12	-0.13	0.09	-0.07
-0.03	0.18	0.01	-0.05
-0.26	-0.09	0.09	-0.06
0.12	-0.05	0.17	-0.05
...	...	...	...

197.39			
	27.43		
		23.26	
			0

-26.43	0.16
17.05	-2.18
23.25	-3.54
-5.38	5.03
-51.09	-2.59
23.19	-1.45
...	...

Recovering the Data

1565862

Given the entire Z matrix we can recover the **centered X** by **multiplying by V^T:**

$$ZV^T = XVV^T = USV^T = X$$

This is like **inverting** our PC recipe → How do we combine our latent features (Z) to get back our original features (X)?

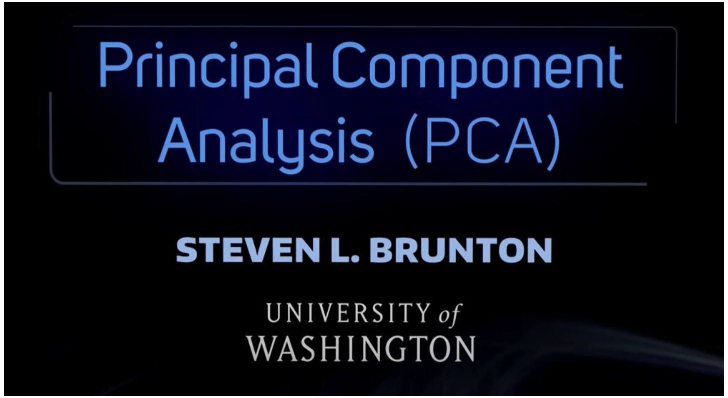
If you choose **k < r,** where r is rank(X), you will only recover X **approximately.**

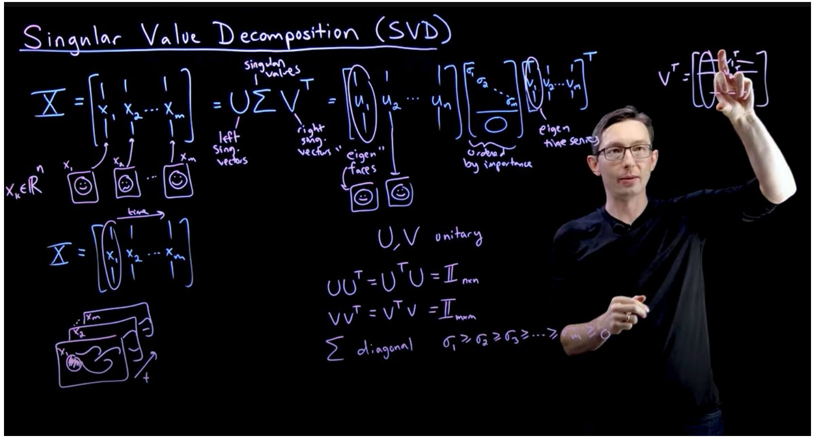
To recover the original (uncentered) X we would also need to add back the mean.

A Great Resource on This Topic: Steve Brunton's SVD Video Series (UW Prof)

1565862

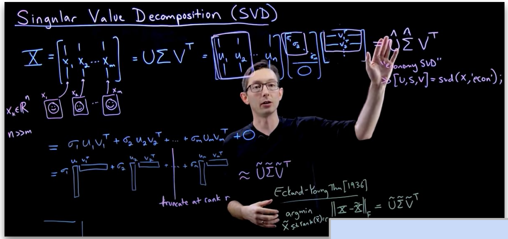
[Singular Value Decomposition Playlist]



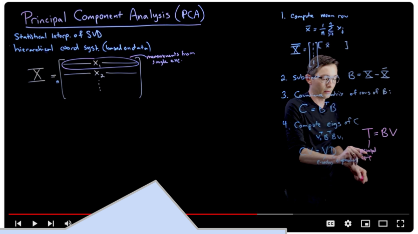


- Extraordinarily clear.
- Videos cover the foundations, applications, linear algebra, stats and code.
- A masterpiece of teaching.
- Note: our *d* is his *m* (# of features).

[S. Brunton's video on SVD]



[S. Brunton's video on PCA]



Unfortunately, his definition of the **Principal Components** is wrong (~8:30).

Demo
Computing PCA using SVD

1565862

Centering Data and Computing Variance

Lecture 25, Data 100 Spring 2025

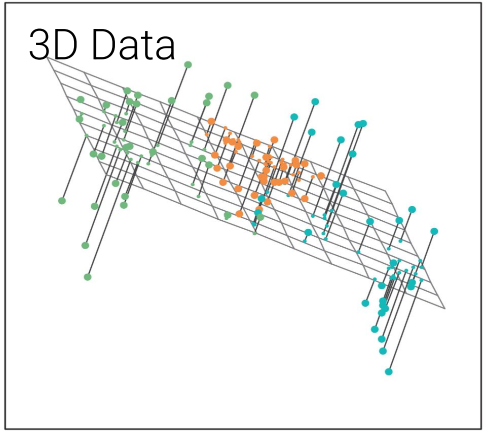
1565862

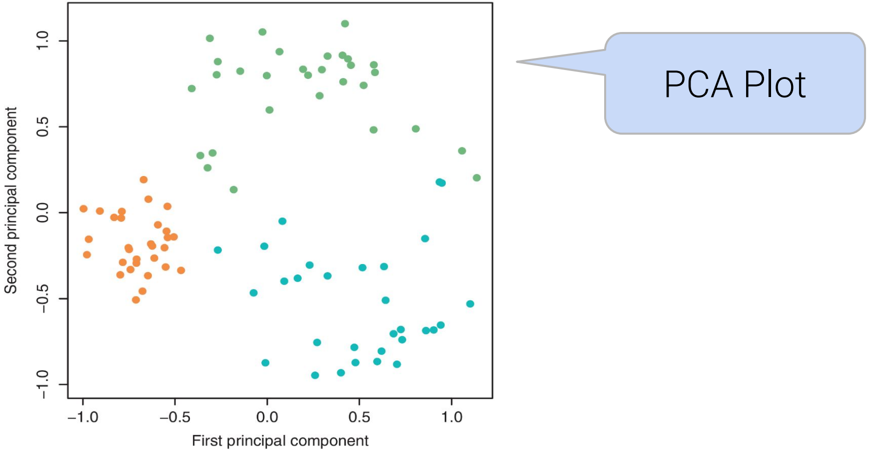
PCA Plot

1565862

We often construct a scatter plot of the data projected onto the **first two principal components**. This is often called a **PCA plot**.

- PCA plots allow us to visually assess similarities between our data points and if there are any clusters in our dataset.





If PC1+PC2 explain a large % of the **total variance**, then the PCA plot is a good representation.

If not, then a **PCA plot** is omitting lots of information.

Capturing Total Variance

(from before)

1565862

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

1565862

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
7.69	5.35	50.79	338.73

$$\Sigma V = \boxed{\frac{S^2}{n}} V$$

Total Variance: **402.56** =

The **component score** tells us the **variance** captured by the i^{th} principal component. The component scores are the **eigenvalues of the covariance matrix**. N is # of datapoints.

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

Variance captured by **PC1**

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

$\rightarrow 197.39^2/100 =$ **389.63**

$\rightarrow 27.43^2/100 = 7.52$

$\rightarrow 23.26^2/100 = 5.41$

Sum = **402.56**. ✅

Variance Ratios

1565862

$$X = USV^T$$

```
u, s, vt = np.linalg.svd(X, full_matrices = False)
```

How do we compute an array of **variance ratios**, where each element is the **fraction** that each PC contributes to total data variance?

A. `s / n          # n is len(X), num features`
B. `s ** 2 / n`
C. `s / sum(s)`
D. `s**2 / sum(s**2)`
E. Something else

i-th component score = Variance captured by i-th PC = $\dfrac{(i\text{-th singular value})^2}{n}$

1565862

slido

How do we compute an array of variance ratios?

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

Variance Ratios

1565862

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

$$X = USV^T$$

$$\text{total variance} = \text{sum of all the component scores} = \sum_{i=1}^{k} \frac{s_i^2}{N}$$

$$\text{variance ratio of principal component j} = \frac{\text{component score j}}{\text{total variance}} = \frac{s_j^2/N}{\sum_{i=1}^{k} s_i^2/N}$$

= s**2 / sum(s**2)

How do we compute an array of **variance ratios**, where each element is the **fraction** that each principal component contributes to total data variance?

```
u, s, vt = np.linalg.svd(X, full_matrices = False)
```

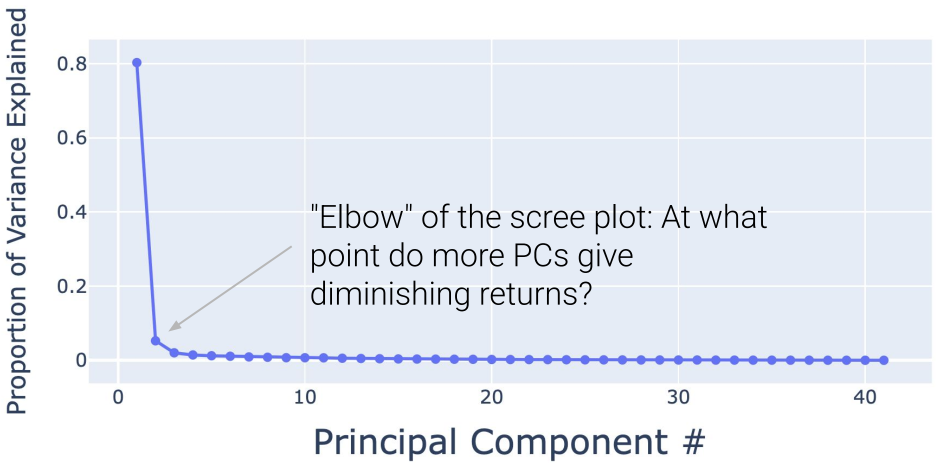
A. `s / n          # n is len(X), num features`
B. `s ** 2 / n`
C. `s / sum(s)`
✅ `s**2 / sum(s**2)`
E. `Something else`

Scree Plot

1565862

If PC1+PC2 explain a large % of the total variance, then the PCA plot is a good representation. If not, then a **PCA plot** is omitting lots of information.

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

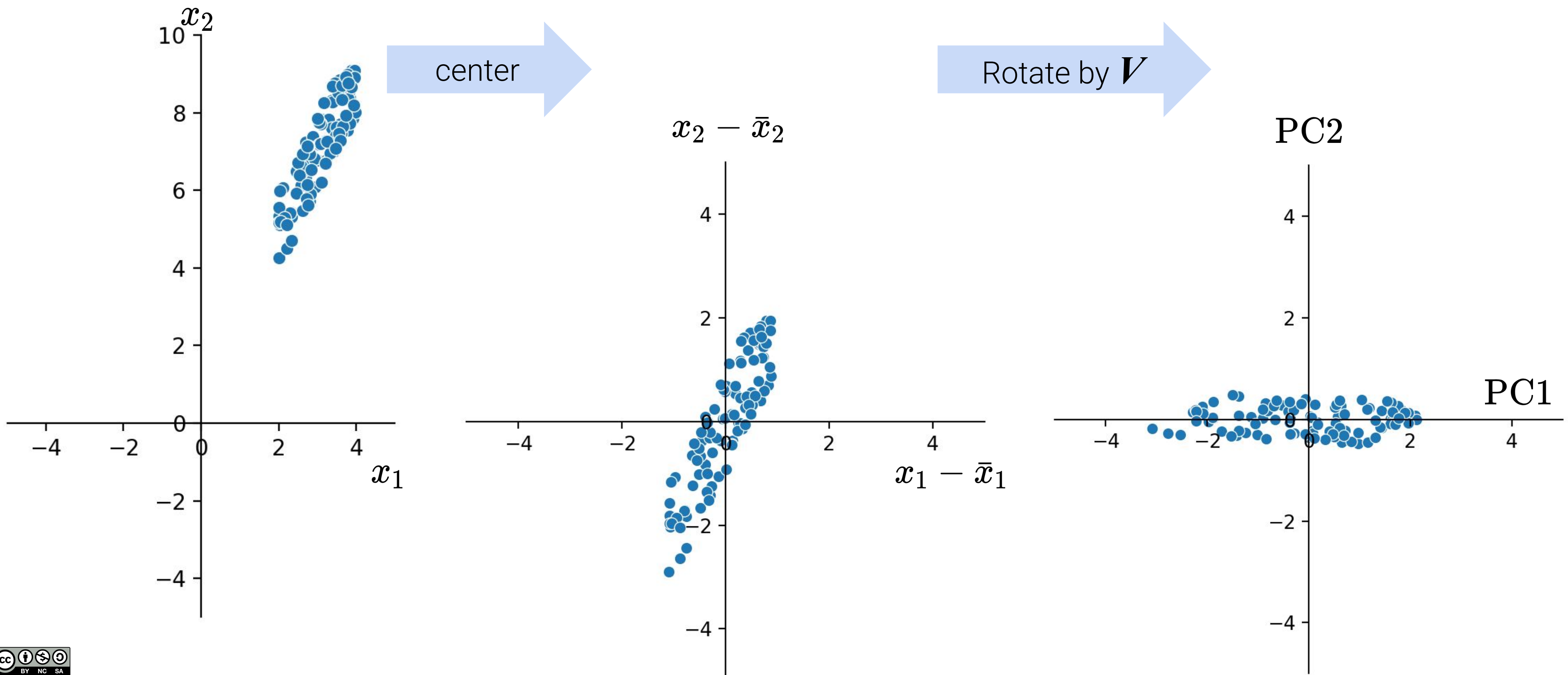


Scree [wikipedia]

How PCA Transforms Data, Visually

1565862

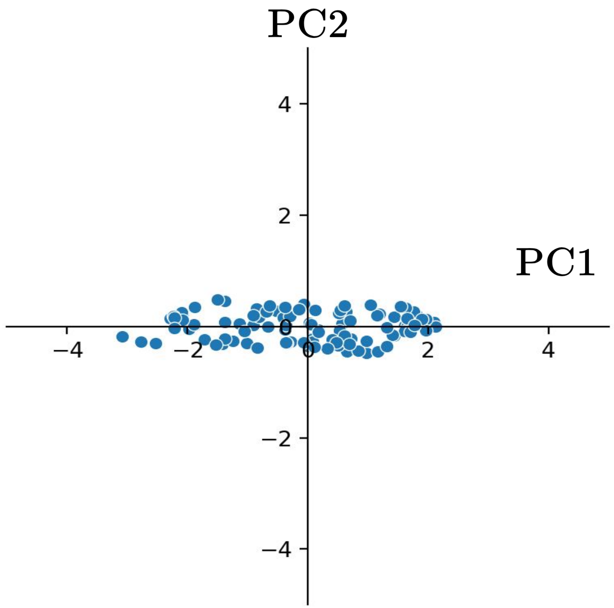
PCA first centers the data matrix, then rotates it so that the direction with the **greatest variance** is aligned with the x-axis. [We saw this "changing frame of reference" last lecture.]



Principal Components

1565862

- Principal components are all **orthogonal** to each other
  - Why? Recall that the **columns of V 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
- The **latent factors** are obtained by **projecting** X onto the principal components



Centering and Standardizing the Data

When running PCA it is important to **center the data** (ensure data is centered at 0):

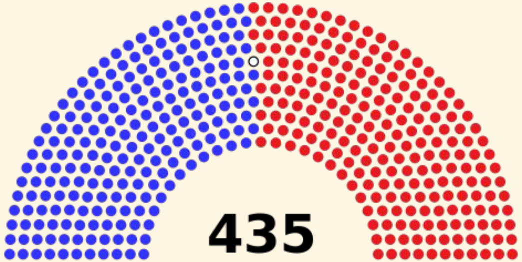
```
X = X - np.mean(X, axis = 0)
```

- The `sklearn` PCA code does this **automatically!**

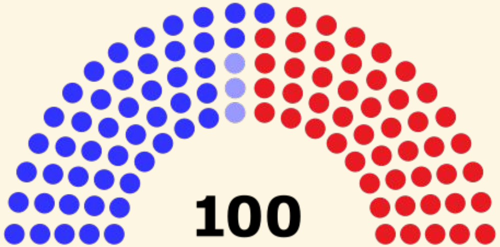
You should also *consider* **standardizing** your data:

```
X = (X - np.mean(X, axis = 0)) / np.std(X, axis = 0)
```

- The `sklearn` PCA code does <u>not</u> do this automatically
- Standardizing puts all cols on same scale (e.g., X=0.5 has same meaning across cols)
- You should **not** standardize your data if the units are **all the same** (e.g., all 0/1 cols)



House of representatives



Senate

Demo

Congressional Vote Data

1565862

Let's examine how the **House of Representatives** (of the 116th Congress, 1st session) voted in the month of **September 2019**.

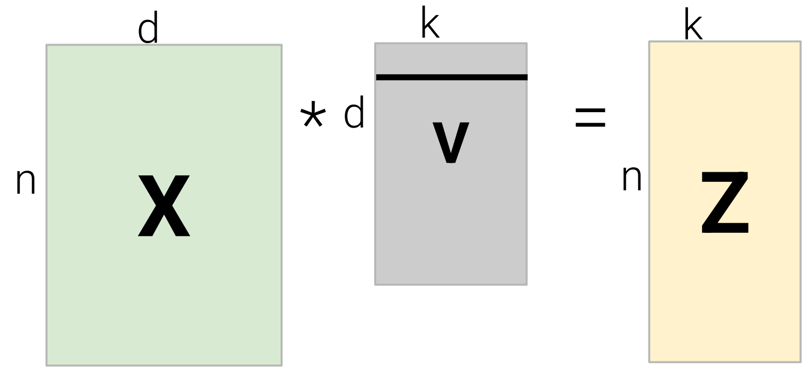
Specifically, we'll look at the records of Roll call votes. From the U.S. Senate (link):

> *Roll call votes occur when a representative or senator votes "yea" or "nay," so that the names of members voting on each side are recorded.*

Do legislators' roll call votes show a relationship with their political party?

Biplot

1565862



The i-th row of V indicates **how much feature *i* contributes** to each PC. Cols of V are the PCs ("recipes").

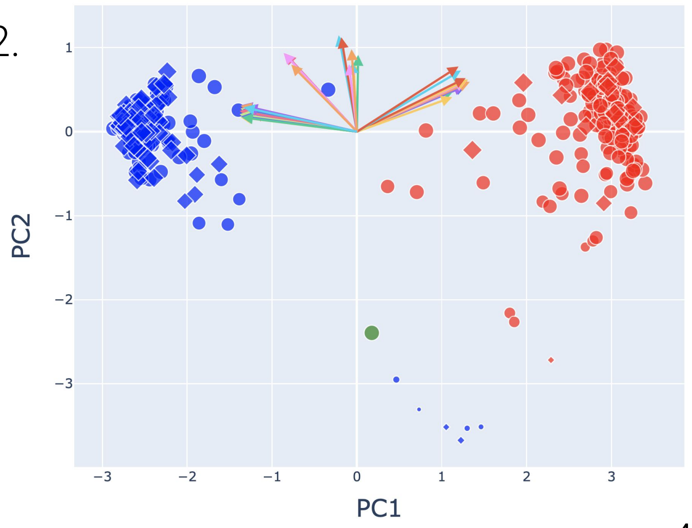
"First row of V = [0.9,-0.44] → PC1 linear combo is 0.9 parts feature 1, and PC2 has -0.44 parts feature 1."

Biplots superimpose **feature influence** on plot of PC1 vs. PC2.

Biplots help us interpret how features influence the PCs: positively, negatively, or not much at all.

Simplest biplot: Plot the rows of V with no scaling.

- For other scalings, which can lead to more interpretable directions/loadings, see [SAS biplots].



Demo

Time permitting!

Image Classification

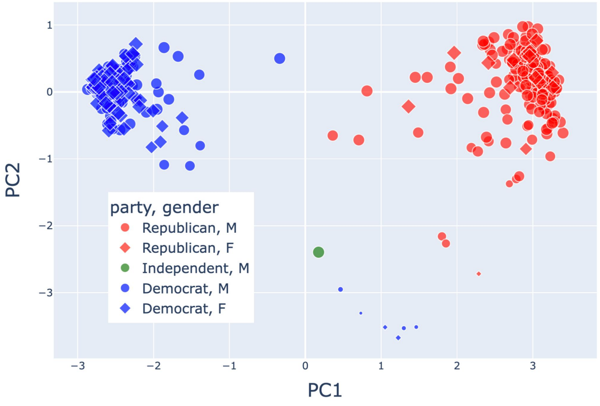
Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. Han Xiao, Kashif Rasul, Roland Vollgraf. arXiv:1708.07747
https://github.com/zalandoresearch/fashion-mnist

Summary: Plots Based on PCA

1565862

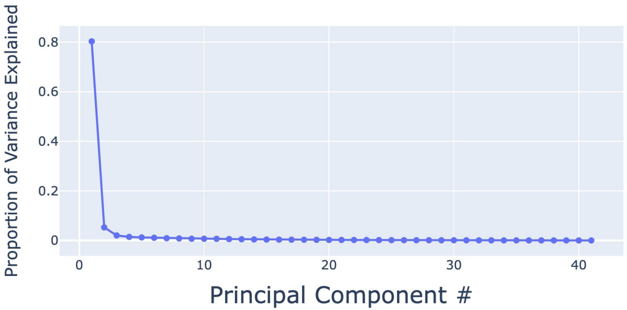
PCA Plot

Scatter plot of PC1 against PC2
- Similarities of data points; identifying clusters.



Scree Plot

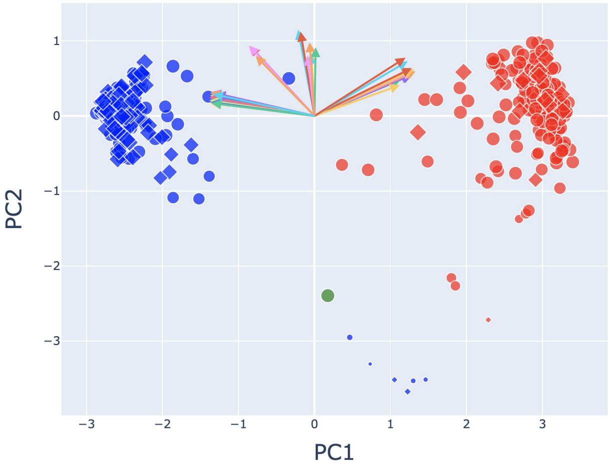
Variance ratio captured by each PC, with largest first.
- If first two ratios large, PCA plot is good representation
- “Elbow” method to assess how many PCs to use



Biplot

PCA plot + **influence of features** on PC1 and PC2.

Each vector is a bill.



LECTURE 25

Principal Component Analysis II

Data 100/Data 200, Spring 2025 @ UC Berkeley

Narges Norouzi and Josh Grossman

Content credit: Acknowledgments