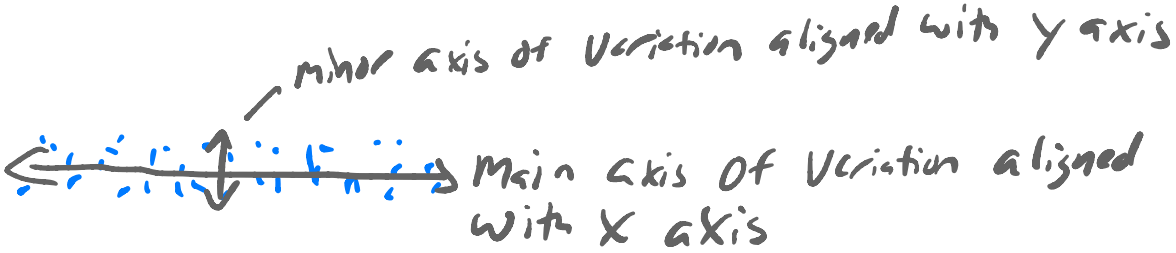
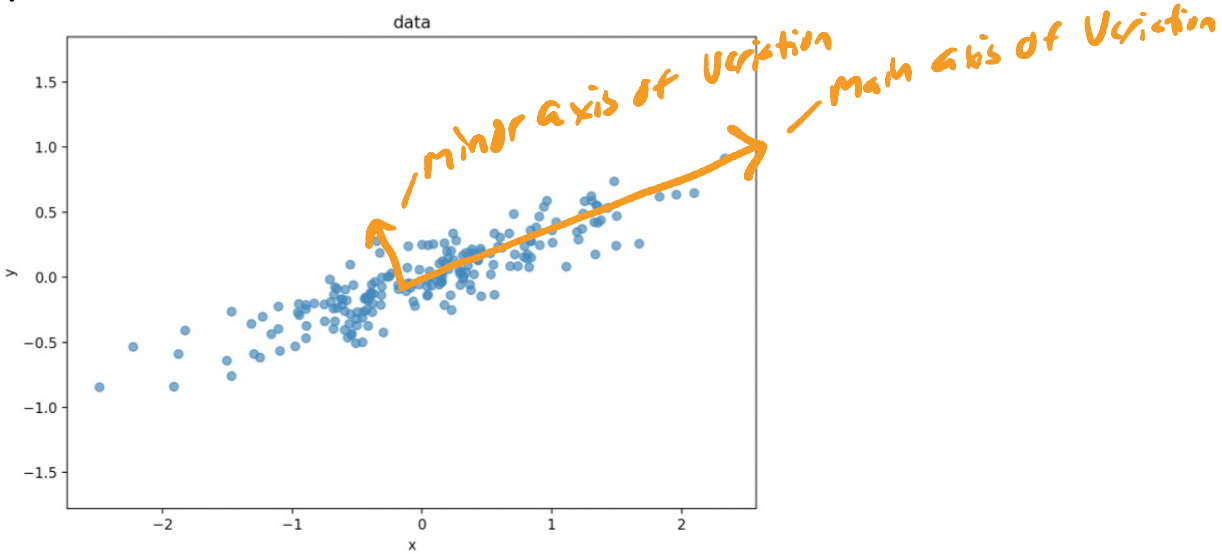
Principal Component Analysis (PCA)

Recall the example where we could toss out a variable (y) without losing much information:

In this case, the main axes of variation of data are aligned with the coordinate axes



⇒ Clearly this is not true in general! Axes of variation do not need to correspond to variables in dataset — e.g. x, y, etc:



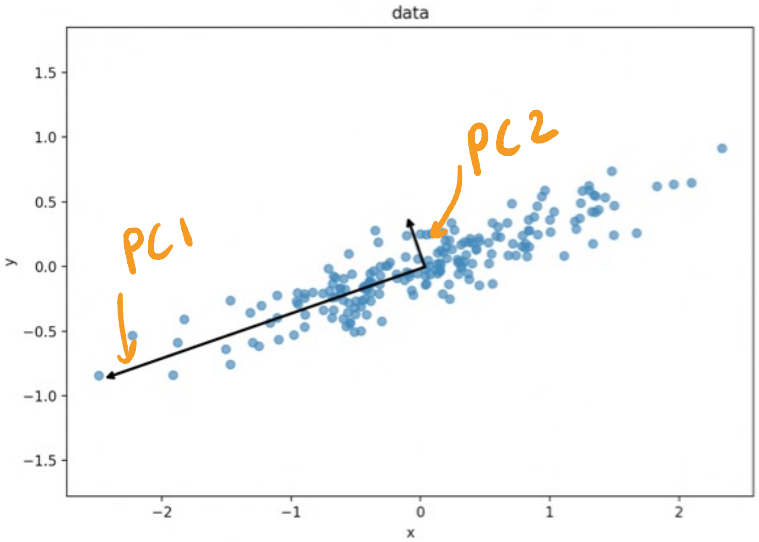
Above main variation axis != x axis!
minor variation axis != y axis!

We call these intrinsic axes of variation in data

principal components (PCs)

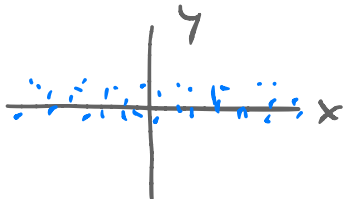
⇒ we number them in order based on the amount of variation along each intrinsic axis

⇒ larger variation → smaller number.



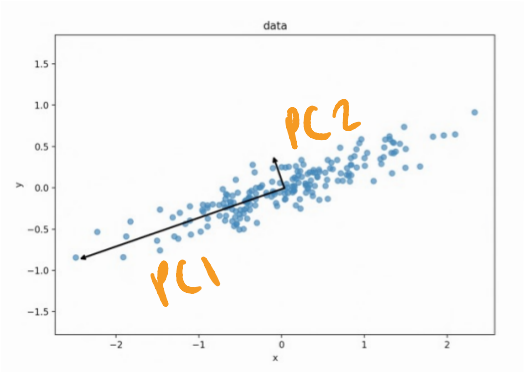
Goal of PCA Reduce dimensionality of dataset (e.g. M = 500 → 30) by dropping intrinsic axes with little variation (least amount of info lost)

e.g. PC2 above, y variable in

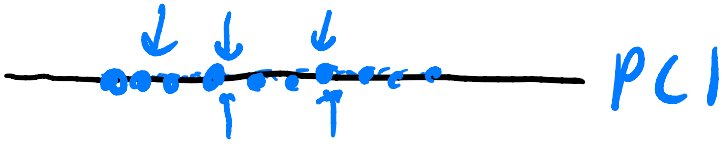


⚠ We do not want to drop data variables

e.g. ~~example~~ drop PC2.



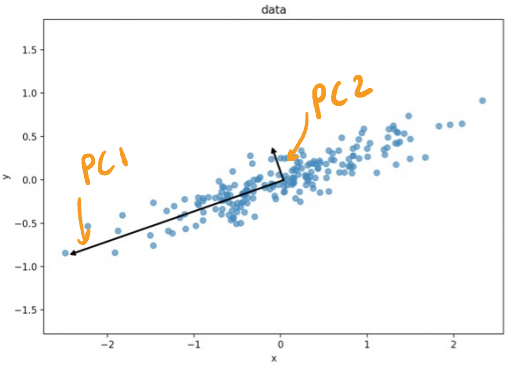
→



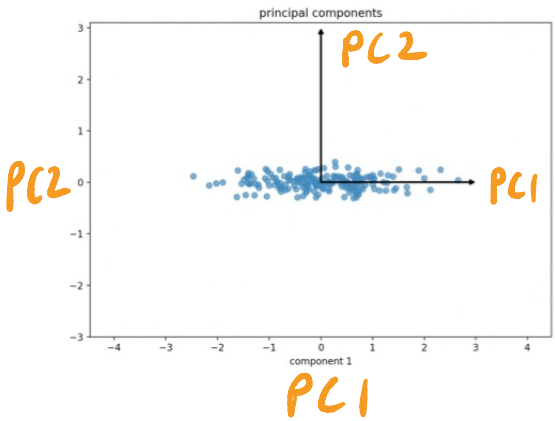
How do we "access" the PCs? — they can be oriented in any direction (but they are assumed to be orthogonal to each other).

⇒ Rotate the data so that the PCs become coordinate axes:

New coordinate axes: The axes of primary variation in the data



Rotate →



How is this done? A translation then a rotation matrix applied:

rotation matrix

centered data matrix:

$A_c = A - \vec{\mu}$ ← translation

See plots above both centered at (0,0)

Rotated, centered data

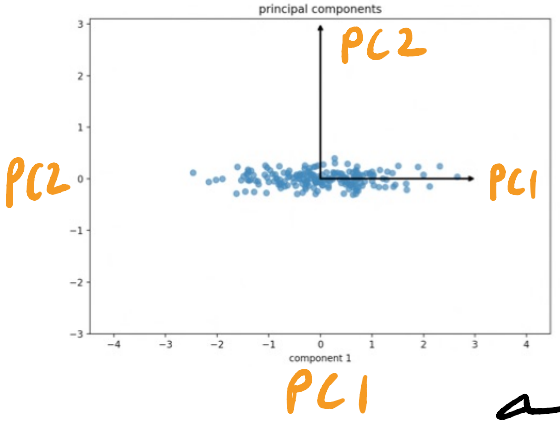
$$\hat{A}_c = (P.T \,@\, A_c).T$$

Same thing

or more concisely:

$$\hat{A}_c = A_c @ P$$

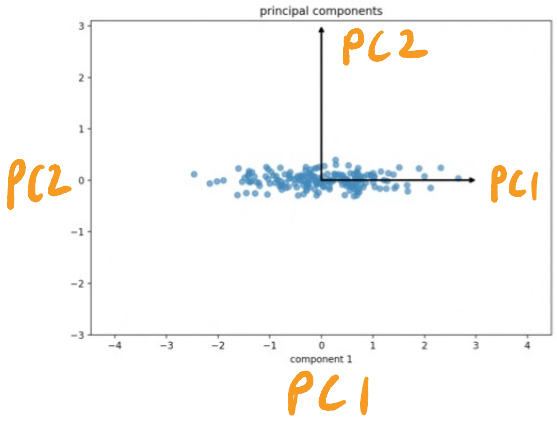
$\hat{A}_c$



$\hat{A}_c$ is centered data in PCA Space: Coordinate axes are PC1 & PC2, not X, Y

... and to drop the insignificant PCs, we can project onto the large meaningful ones (e.g. PC1)

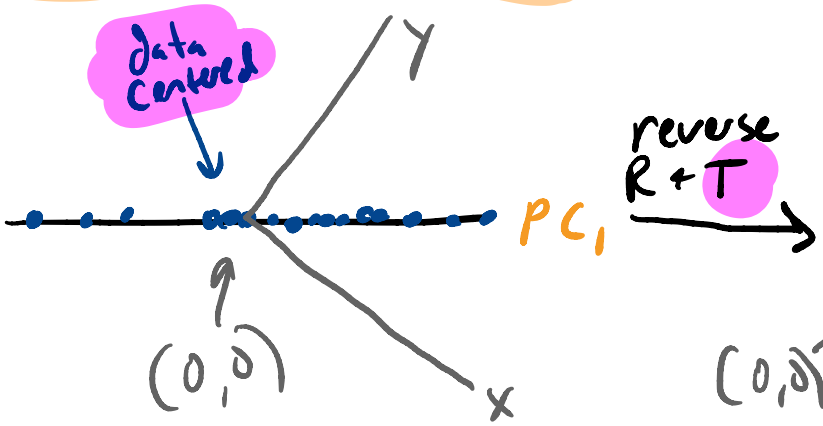
Remember: project = index vars — e.g. $\hat{A}_c[:, 0]$ (PC1 in index 0)



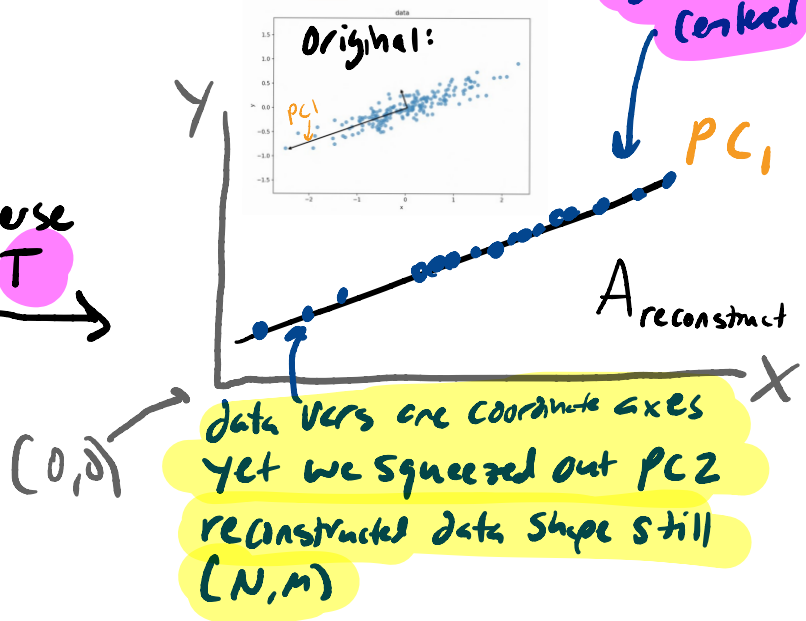
$\hat{A}_c[:, 0]$ →

This achieves what we wanted, but the coordinate axes are PCs — not data variables, which might not what we want.

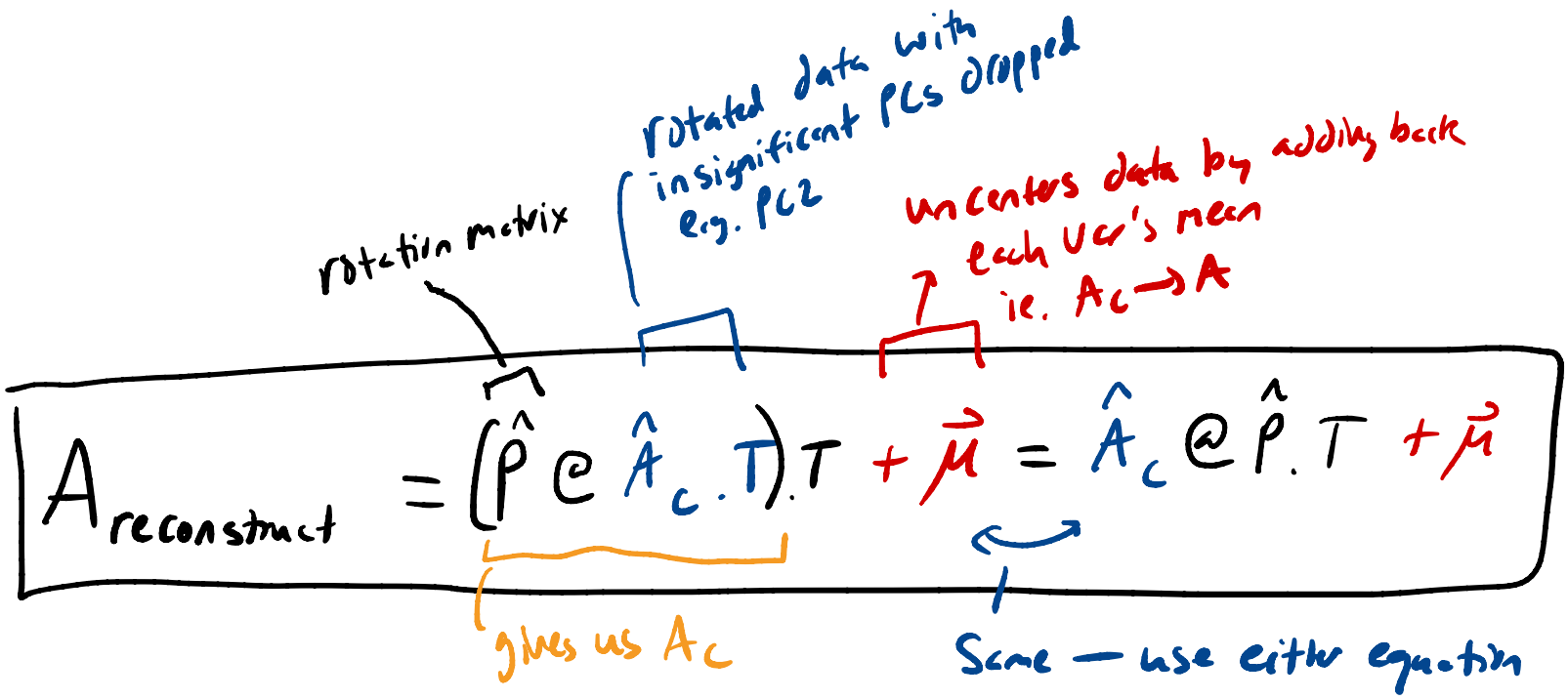
We can express this result in terms of the original data variables (reconstruct the data) by undoing the rotation + translation!



reverse R+T →



To reverse R+T and reconstruct data from $\hat{A}_c$:



What is the rotation matrix P?

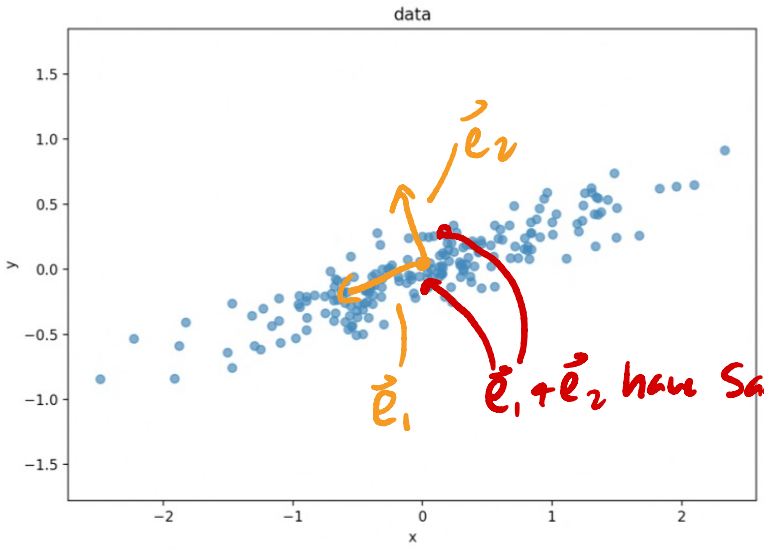
P is the **eigenvectors** of the **covariance matrix** Σ of the data.

Recall: $\Sigma = \frac{1}{N-1} A_c.T @ A_c$

Shape: (M, M)

M of them — one per data variable

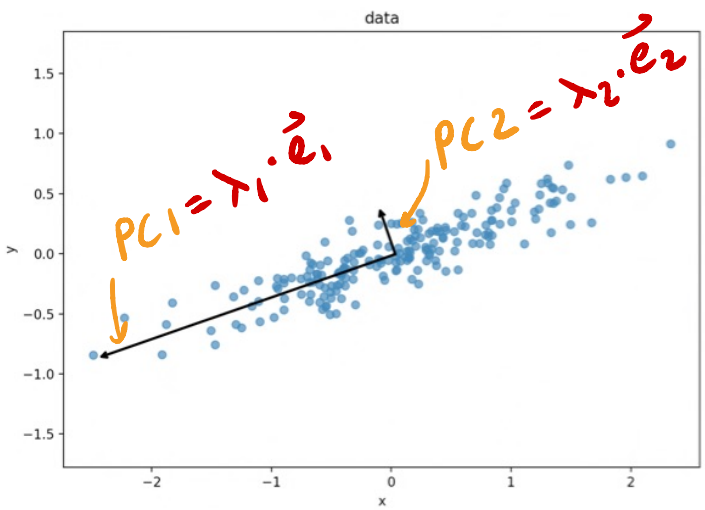
Eigen vectors $P = [\vec{e}_1, \vec{e}_2, \ldots, \vec{e}_m]$ tell us the **direction** but not amount of the corresponding PCs.



★ Rotating data by P aligns coordinate axes with PCs

How do we know the amount of variation in each PC direction? **Eigen values** $= [\lambda_1, \lambda_2, \ldots, \lambda_m]$

M scalars/floats
One per PC/eigenvector/data variable.



← λ_1 is much bigger than λ_2!

Computing eigenvectors and eigenvalues

In numpy:

covariance matrix

```
e_vals, e_vecs = np.linalg.eig(Σ)
```

(e_vecs = P)

↑ Computes the **spectral decomposition** of Σ

P is an orthogonal matrix (forms orthogonal basis)

Strategy for removing insignificant PCs:

e_vals tell us amount of variation in a PC

⇒ if small, we can remove.

e.g. e_vals = [0.1, 3, 0.7, 5]

⇒ Sort high → low:

e_vals = [5, 3, 0.7, 0.1]

these account for small part of overall data variation, might be good candidate to remove PC3 & PC4 here.

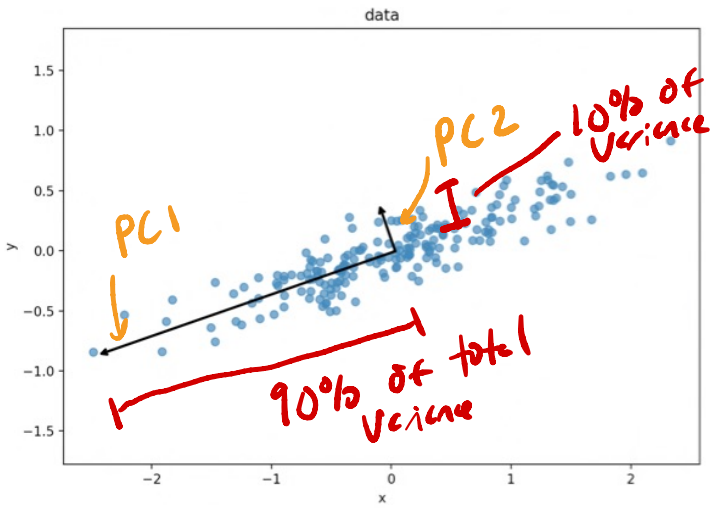
☆ How many PCs to drop / what is insignificant is up to us to decide.

⇒ useful to determine the proportion of total variance accounted for by each PC

these numbers have little meaning on their own

e_vals = [5, 3, 0.7, 0.1]

↳ expressing each as a fraction of the total variance "pot" more helpful:



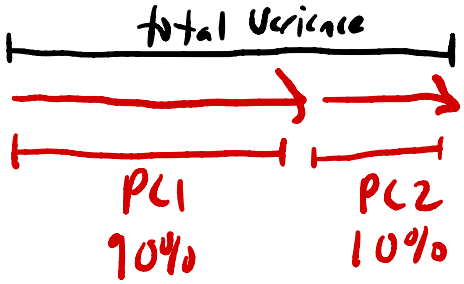
e_vals = [5, 3, 0.7, 0.1]

total var = 5 + 3 + 0.7 + 0.1
= 8.8

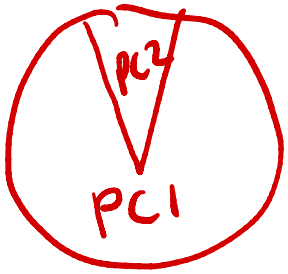
prop_var = [5/8.8, 3/8.8, 0.7/8.8, 0.1/8.8]

57% 34% 8% 1%

Sum = 100%



☆ prop_var must sum to 100% (or 1).



total variance

Often helpful to define a Variance-kept Cut-off (threshold) — e.g. toss out PCs to keep at least 90% of the total variance in the data.

⇒ Compute Cumulative Variance accounted for:

Cum_Var =

$$\left[\frac{5}{8.8}, \frac{3}{8.8}+\frac{5}{8.8}, \frac{0.7}{8.8}+\frac{3}{8.8}+\frac{5}{8.8}, \frac{0.1}{8.8}+\frac{0.7}{8.8}+\frac{3}{8.8}+\frac{5}{8.8}\right]$$

$$=[0.57, 0.9, 0.99, 1.0]$$

discard

Keep

after accounting for all PCs including this smallest one, total var must be 1.0 i.e. 100%.

To keep 90% of the information in data/variance, only keep top 2 PCs.

Summary of PCA

1) [optional] Normalize data if appropriate so that each data value in range $[0, 1]$.

Note: Can be done in vectorized way without matrix multiplication with one line of code.

- Normalize per-variable (separately) may be appropriate if data variables have many different units (e.g. miles & cms)
  - ⇒ You lose relative scales among vars
- normalize globally (together) may be appropriate if variables have similar units (e.g. inches)
  - ⇒ won't distort relative scales among vars

2) Center data by subtracting each variable by its mean $\vec{\mu}$

$$A_c = A - \vec{\mu}$$

3) Compute covariance matrix: $\Sigma = \frac{1}{N-1} A_c.T \circledast A_c$

4) Compute eigenvectors P and eigenvalues via spectral decomposition of Σ — via numpy.

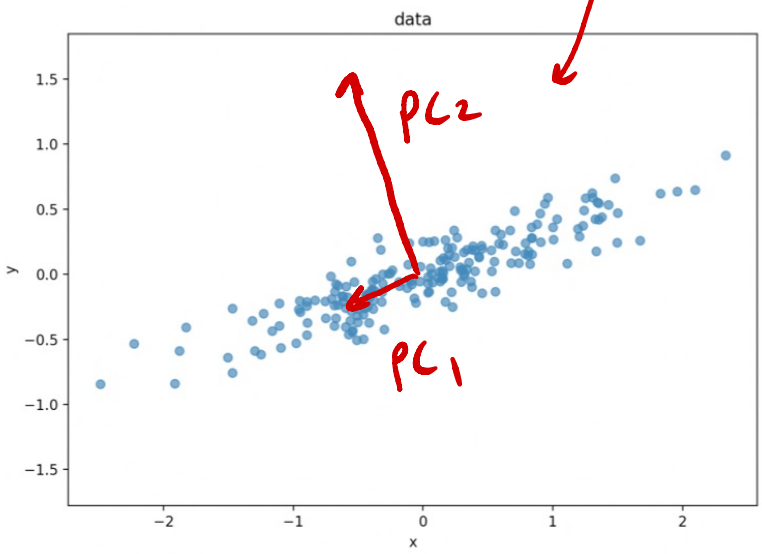
Shape: (M, M)

5) Sort e_vals high → low

e.g. $[3.2, 1.1, 0.03]$

Sort P (columns) in the same order otherwise PC lengths & directions get scrambled!

problem if we don't sort P & evals the same way!



6a) Compute proportion variance accounted for by each PC

e.g total = $3.2 + 1.1 + 0.03 = 4.33$

$\Rightarrow \left[3.2/4.33,\ 1.1/4.33,\ 0.03/4.33\right]$

6b) Compute cumulative variance accounted for by PC_i:

e.g $\left[3.2/4.33,\ (3.2+1.1)/4.33,\ (3.2+1.1+0.03)/4.33\right]$

6c) Threshold PCs based on desired amount of info/variance to keep. We keep K PCs.

e.g. ≥90%. If PC1 + PC2 = 90%, toss out PC3.

7) project data into PCA space (rotating data to align coordinate axes to PCs)

$\hat{P}$ is P with certain PCs/cols dropped

$$\underbrace{\hat{A}_C}_{(N,K)} = \underbrace{A_C}_{(N,M)} @ \underbrace{\hat{P}}_{(M,K)}$$

keep top K PCs

8) [optional] Reconstruct data [rotating data after projecting away insignificant PCs to align coordinate axes with original data variables]

if you didn't normalize:

$$\underbrace{A_{reconstut}}_{(N,M)} = \underbrace{\hat{A}_C}_{(N,K)} @ \underbrace{\hat{P}.T}_{(K,M)} + \underbrace{\vec{\mu}}_{(1,M)}$$

if you normalized:

original data range in each variable

$$A_{reconstut} = \vec{S} \cdot \left(\hat{A}_C @ \hat{P}.T\right) + \vec{\mu}$$