Visual Analytics– Dealing with Big Data: Dimensionality Reduction

Dr. Ab Mosca (they/them)

Slides based off slides courtesy of Jordan Crouser (https://jcrouser.github.io/)

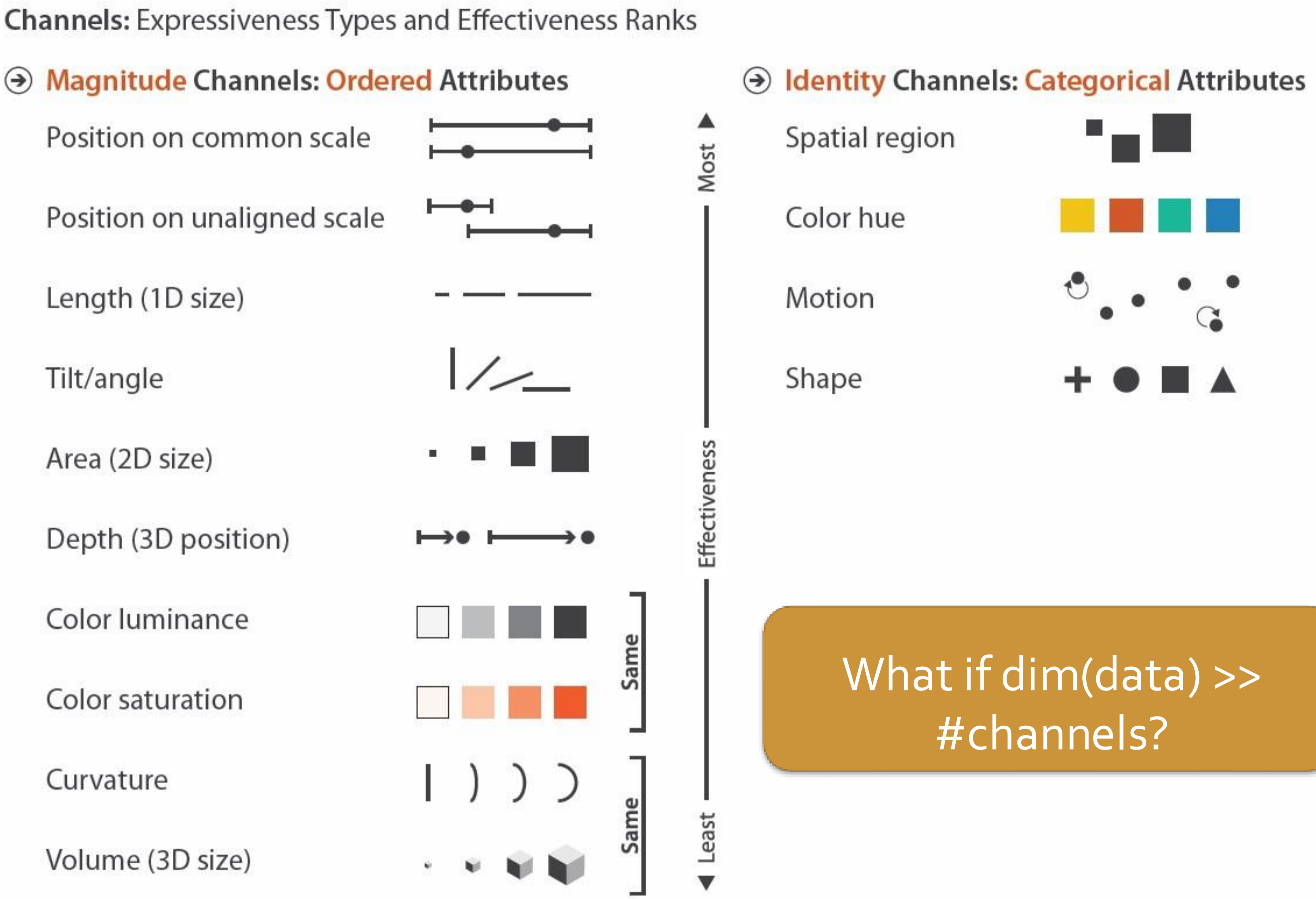
Reminder

- Be prepared for prototype testing in class on Thursday!

Plan for Today

- Dimension Reduction Techniques
  - PCA
  - MDS
  - t-SNE
  - UMAP

What we've been (mostly) worried about



What if dim(data) >> #channels?

Ideas?

Dealing with Many Dimensions

- **Current situation**: our data live in *p*-dimensional space where p >> # visual channels

- Odds are not all dimensions are equally useful

→ we can reduce the number of dimensions without loosing too much valuable information

Dimension reduction

Approach #1: Feature Elimination (i.e. Subset Selection)

- Throw out less useful/useless dimensions

Pros? Cons?

- Pros:
  - Relatively simple
  - Preserve interpretability
- Cons:
  - Don't gain any information from features you drop

Dimension reduction

Approach #2: Feature Extraction

- Create new features (dimensions) that are combinations of the old ones

Pros? Cons?

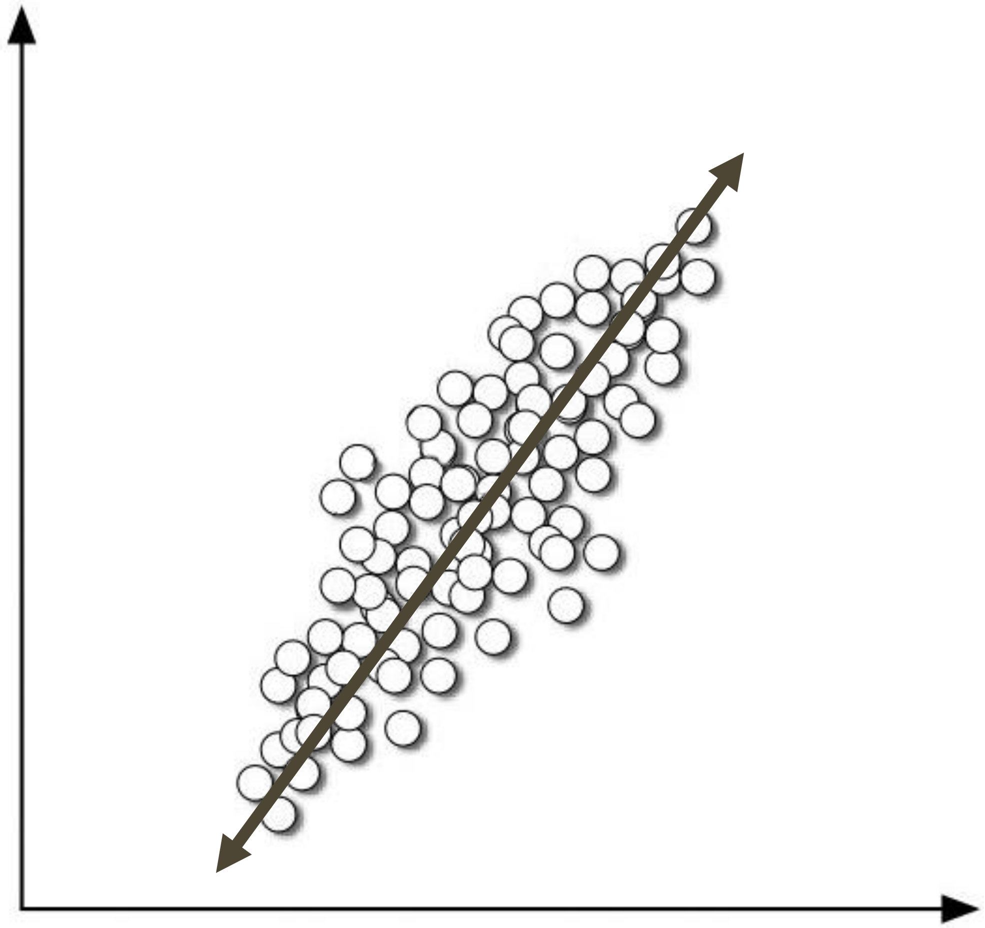
- Pros:
  - Since new features are all combinations of old features, we are not totally dropping data
- Cons:
  - Less interpretable, especially to non-experts

Feature Extraction

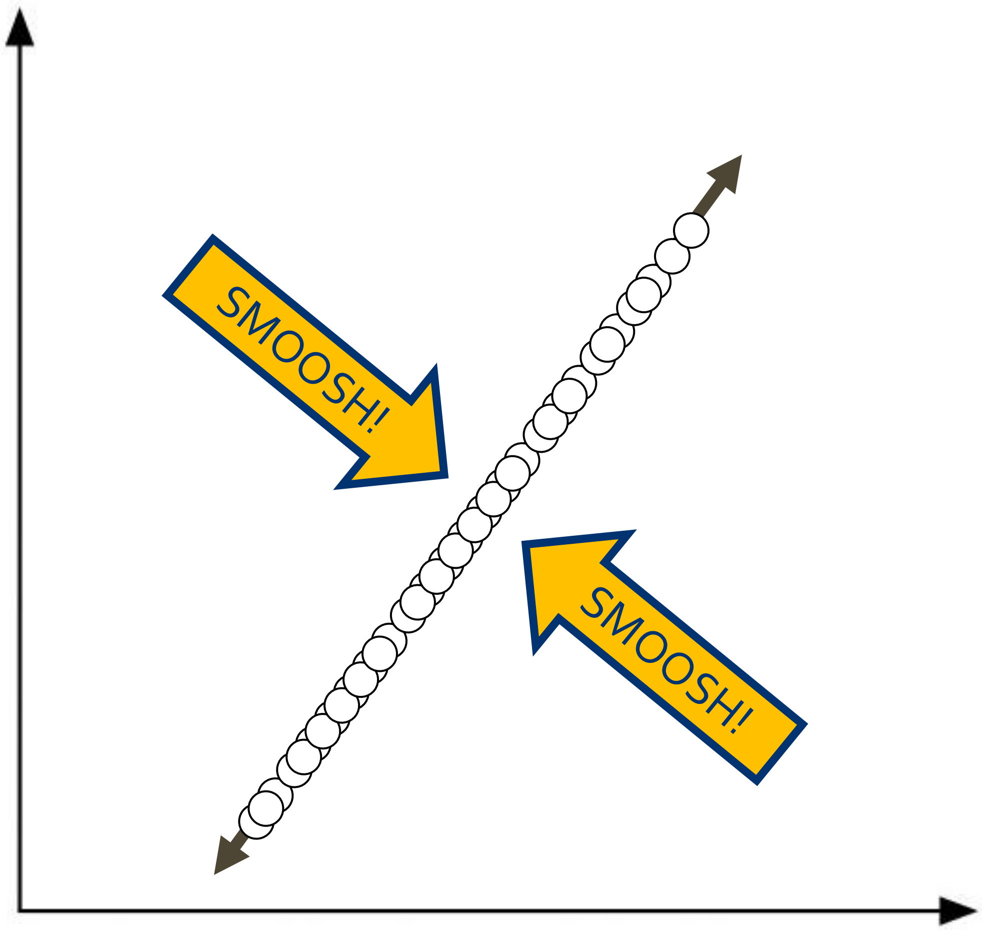
Principal Component Analysis

- Project data into a smaller space composed of most important (informative) components

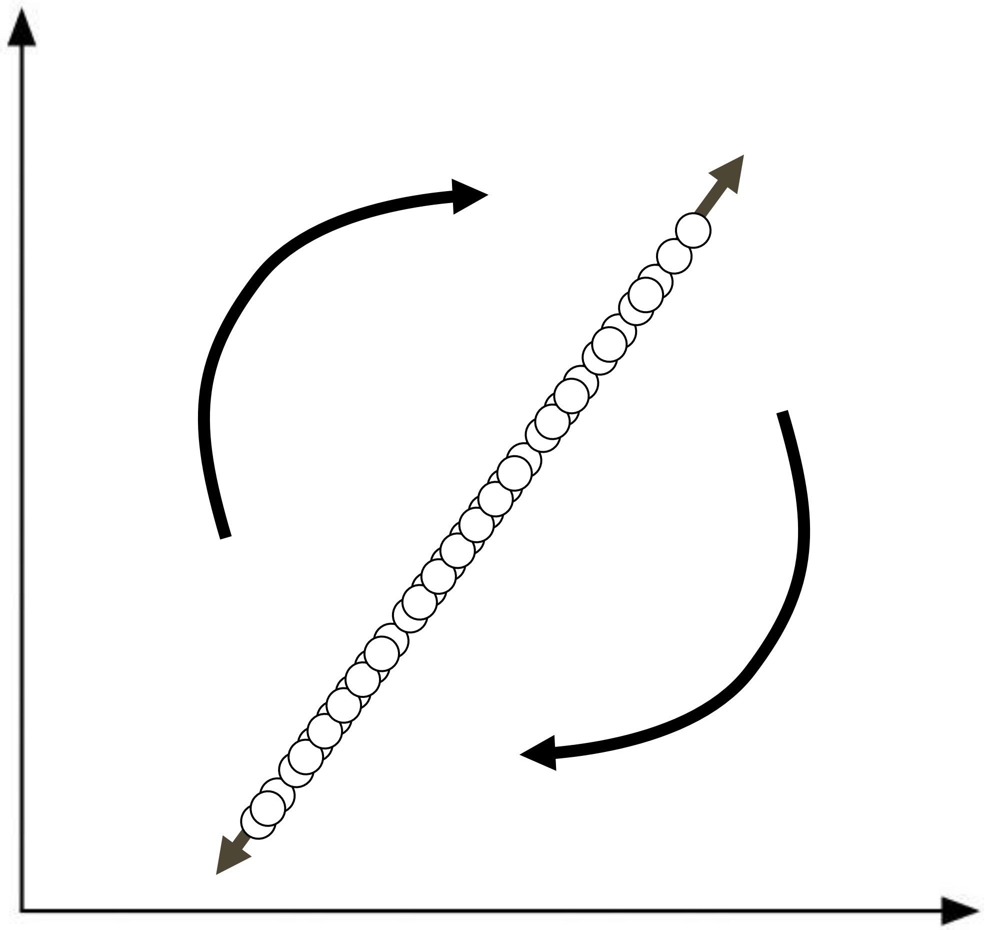
Projection



Projection



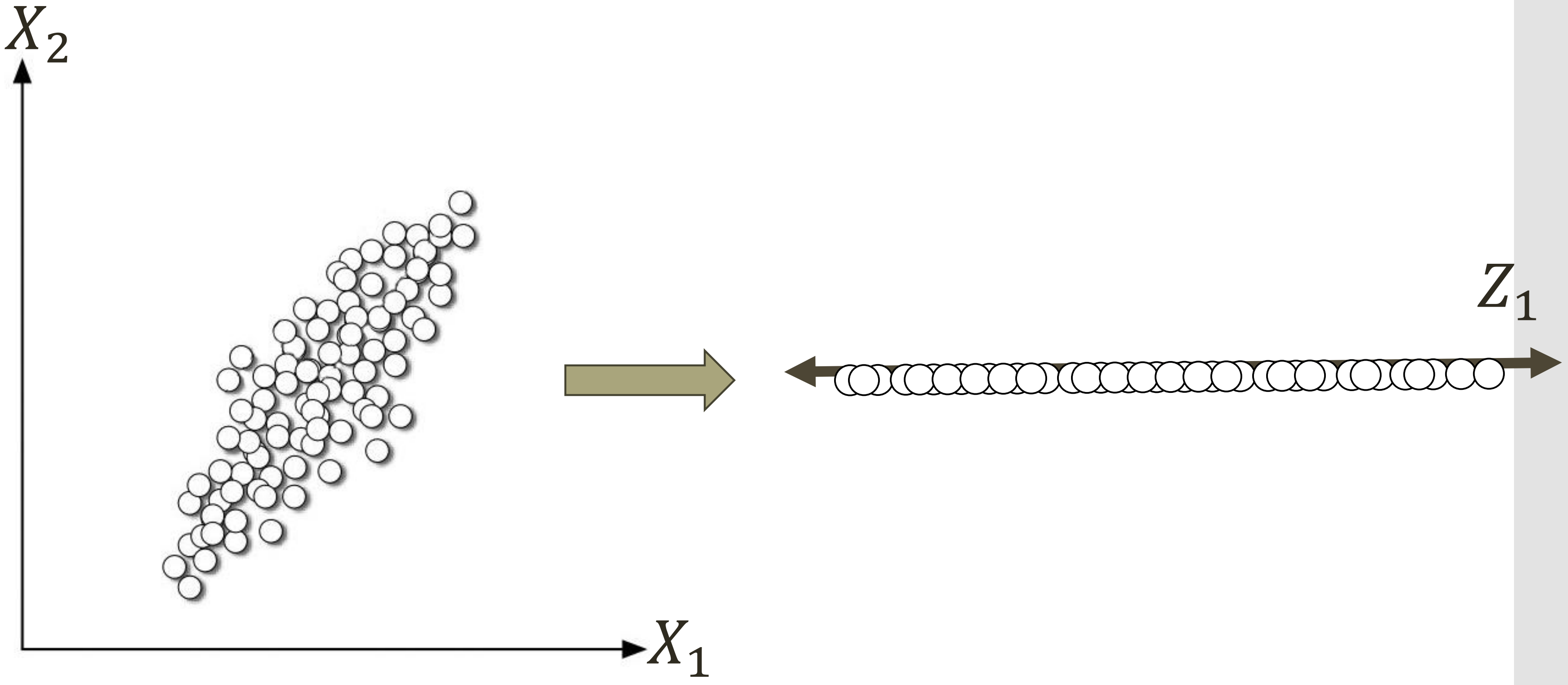
Projection



Projection

Big idea: *transform* the data into a new space

$$[X_1 \quad X_2] \mapsto [Z_1]$$



$$[X_1 \quad X_2 \quad X_3 \quad X_4 \quad X_5] \mapsto [Z_1 \quad Z_2]$$

Linear projection

- New features are **linear combinations** of original data:

$$Z_j = \sum_{i}^{m} \theta_{ij} X_i$$

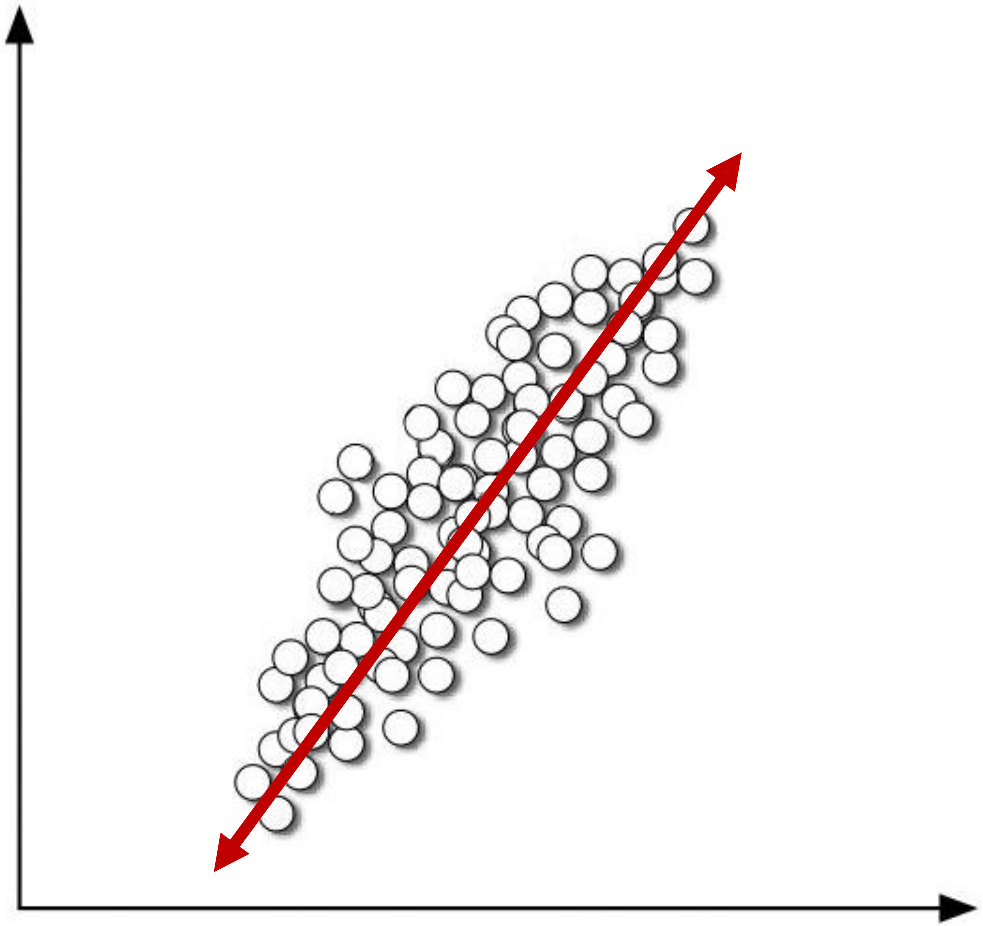
- We get them by multiplying the *data matrix* by a *projection matrix*

$$\begin{bmatrix} Z_1 & Z_2 \end{bmatrix} = \begin{bmatrix} X_1 & X_2 & X_3 & X_4 & X_5 \end{bmatrix} \boxed{\begin{bmatrix} \varphi_{1,1} & \varphi_{1,2} \\ \varphi_{2,1} & \varphi_{2,2} \\ \varphi_{3,1} & \varphi_{3,2} \\ \varphi_{4,1} & \varphi_{4,2} \\ \varphi_{5,1} & \varphi_{5,2} \end{bmatrix}}$$

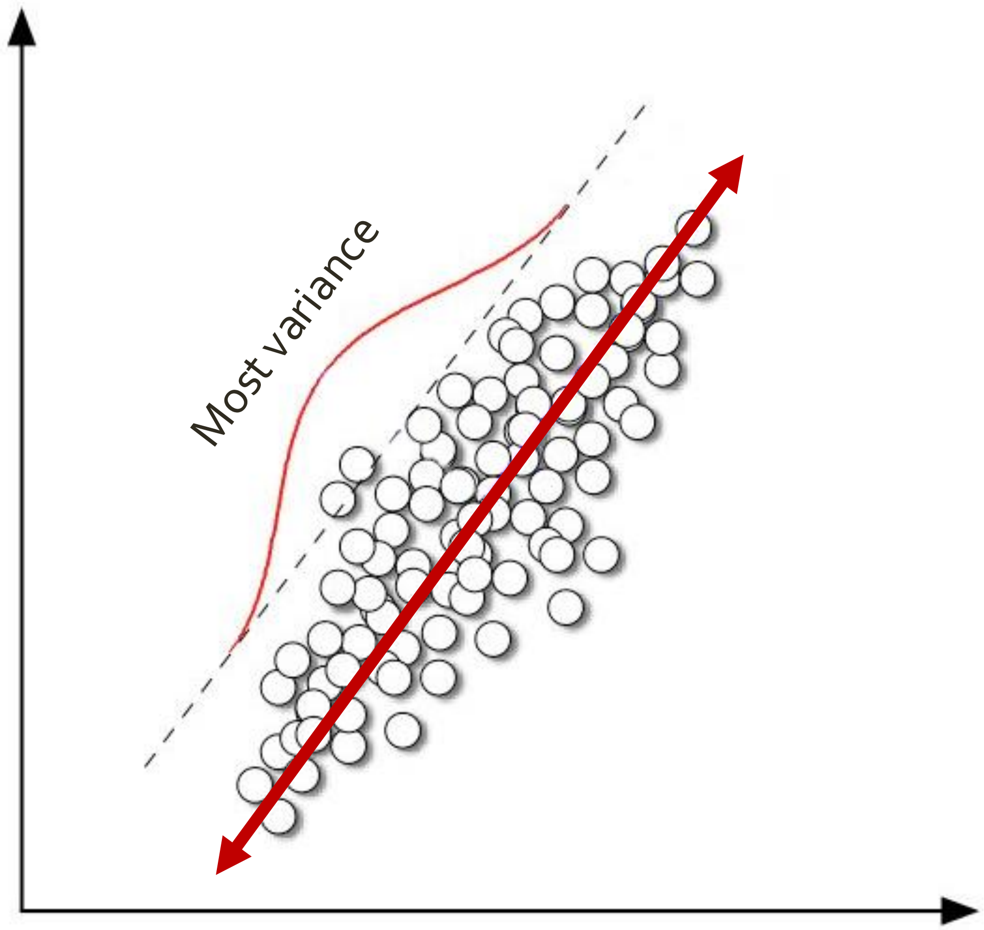
Why is projection helpful?

- Data can be rotated, scaled, and translated without changing the **underlying relationships**
- This means you're allowed to look at the data from whatever angle makes your life easier…
- Because new dimensions are combinations of old ones, we do not lose any data!

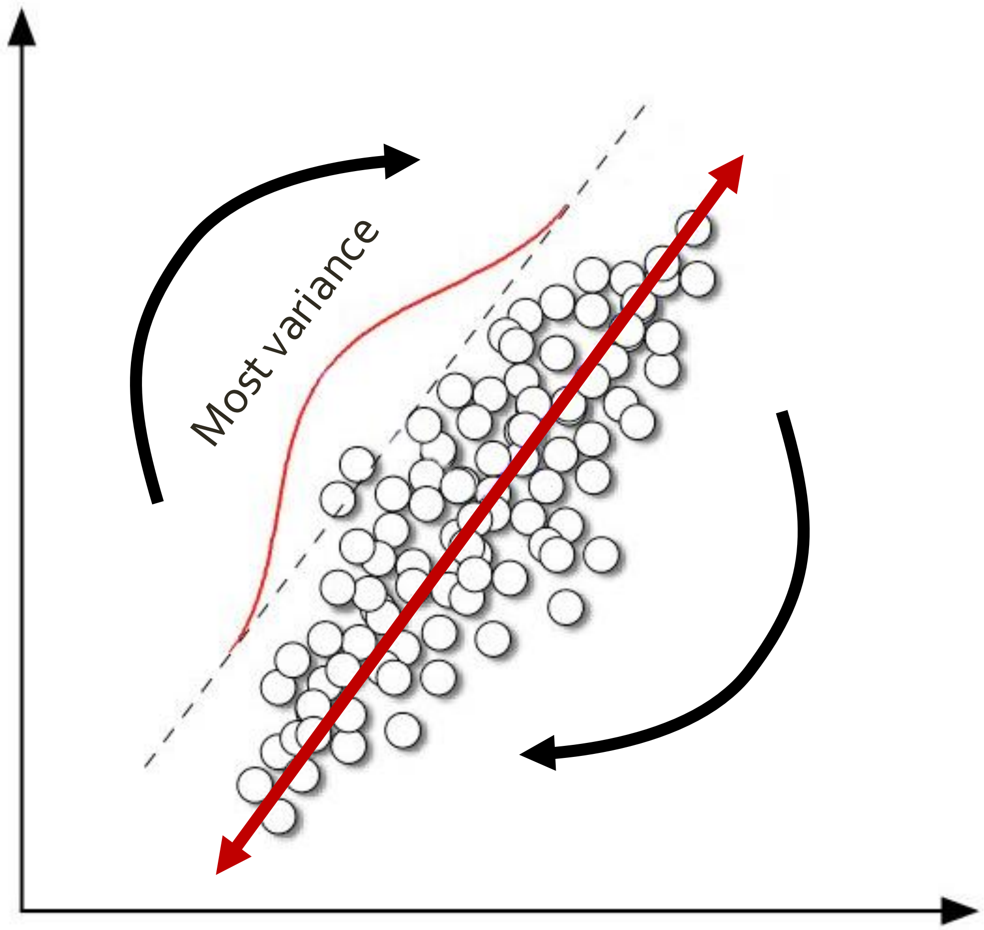
Flashback: why did we pick this line?



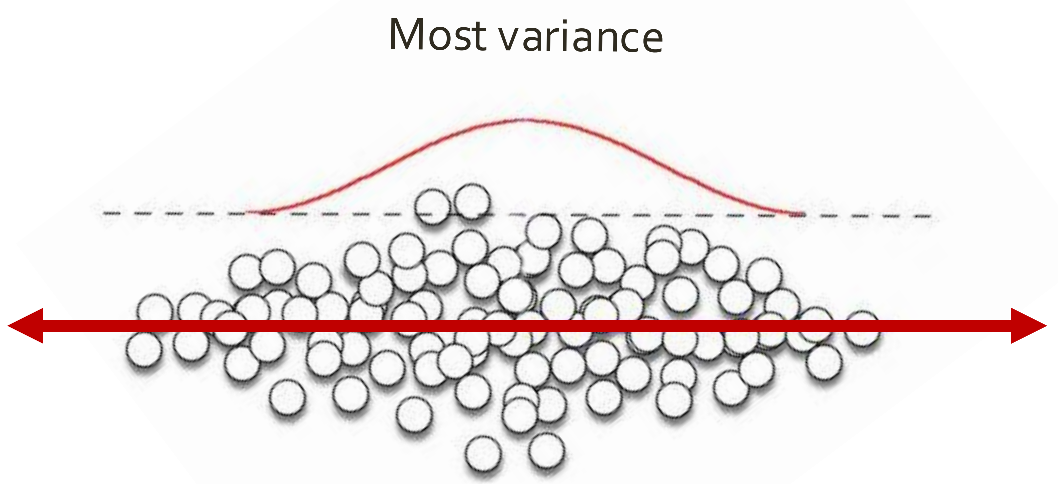
Explains the most **variance** in the data



Imagine this line as a new dimension…



“Principal component”



Mathematically

- The 1^{st} **principal component** is the normalized* linear combination of features:

$$Z_1 = \phi_{11} X_1 + \phi_{21} X_2 + \cdots + \phi_{p1} X_p$$

that has the largest variance

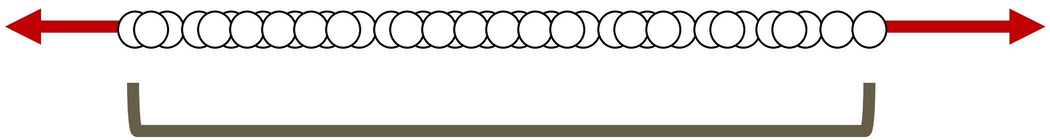
- $\phi_{11}, \dots, \phi_{p1}$: the **loadings** of the 1^{st} principal component

* By **normalized** we mean: $\sum_{j=1}^{p} \phi_{j1}^2 = 1$

Using loadings to project

Multiply by loading vector to project ("smoosh") each observation onto the line:

$$z_{i1} = \phi_{11}x_{i1} + \phi_{21}x_{i2} + \cdots + \phi_{p1}x_{ip}$$



These values are called the **scores** of the 1st principal component

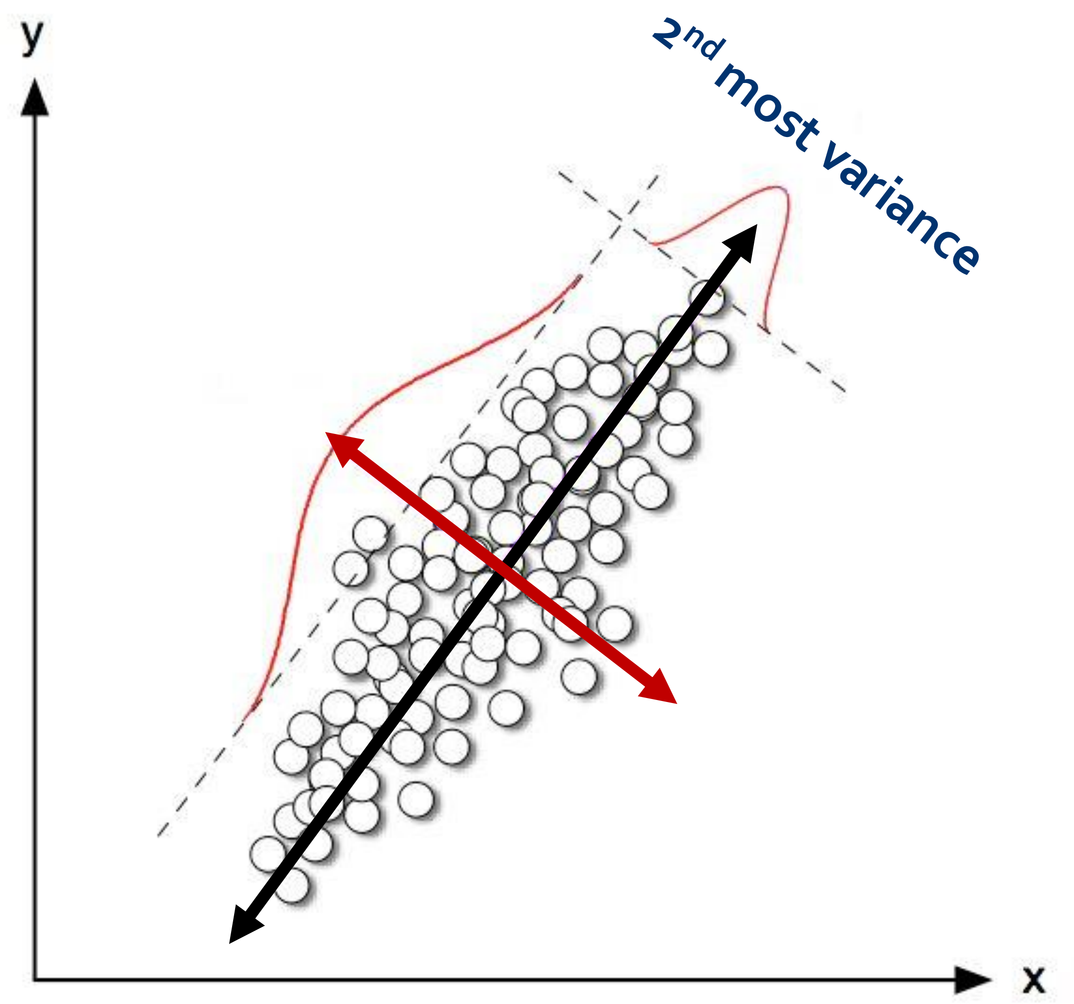
Additional principal components

- The **2nd principal component** is the normalized linear combination of the features

$$Z_2 = \phi_{12}X_1 + \phi_{22}X_2 + \cdots + \phi_{p2}X_p$$

that has maximal variance out of all linear combinations that are **uncorrelated** with Z_1

Principal components are orthogonal



Generating additional principal components

- We can think of this recursively
- To find the M^{th} principal component . . .
  - Find the first $(M-1)$ principal components
  - Subtract the projection into that space
  - Maximize the variance in the remaining *complementary* space

Principal Component Analysis

- Conduct analysis using *n* principal components instead of the original data

How do we choose *n*?

https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c

Principal Component Analysis

- Conduct analysis using *n* principal components instead of the original data

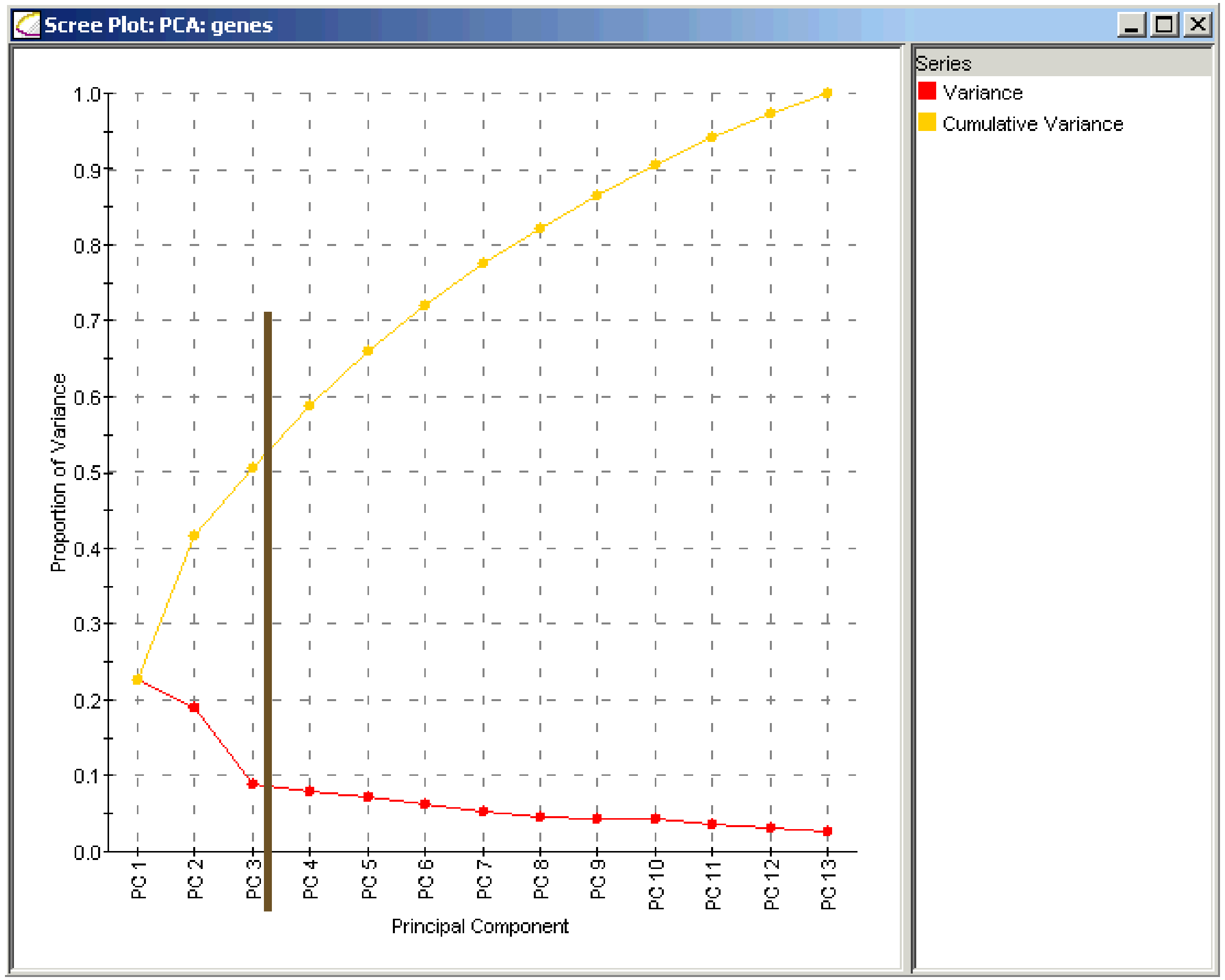
Choosing *n:*

- **Option 1:** Arbitrarily choose
  - Ex. Will your audience only understand 2-d data?
- **Option 2:** Choose a proportion of data variance that must be captured by your principal components. Keep adding principal components until the threshold is hit.
- **Option 3:** Plot cumulative proportion of data variance captured by your principle components. Look for "elbow" where reduction in variance drops off.

https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c

Principal Component Analysis

- **Option 3:** Plot cumulative proportion of data variance captured by your principle components. Look for "elbow" where reduction in variance drops off.



https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c

PCA Exploration

- Take 10 minutes to explore using this applet:

https://setosa.io/ev/principal-component-analysis/

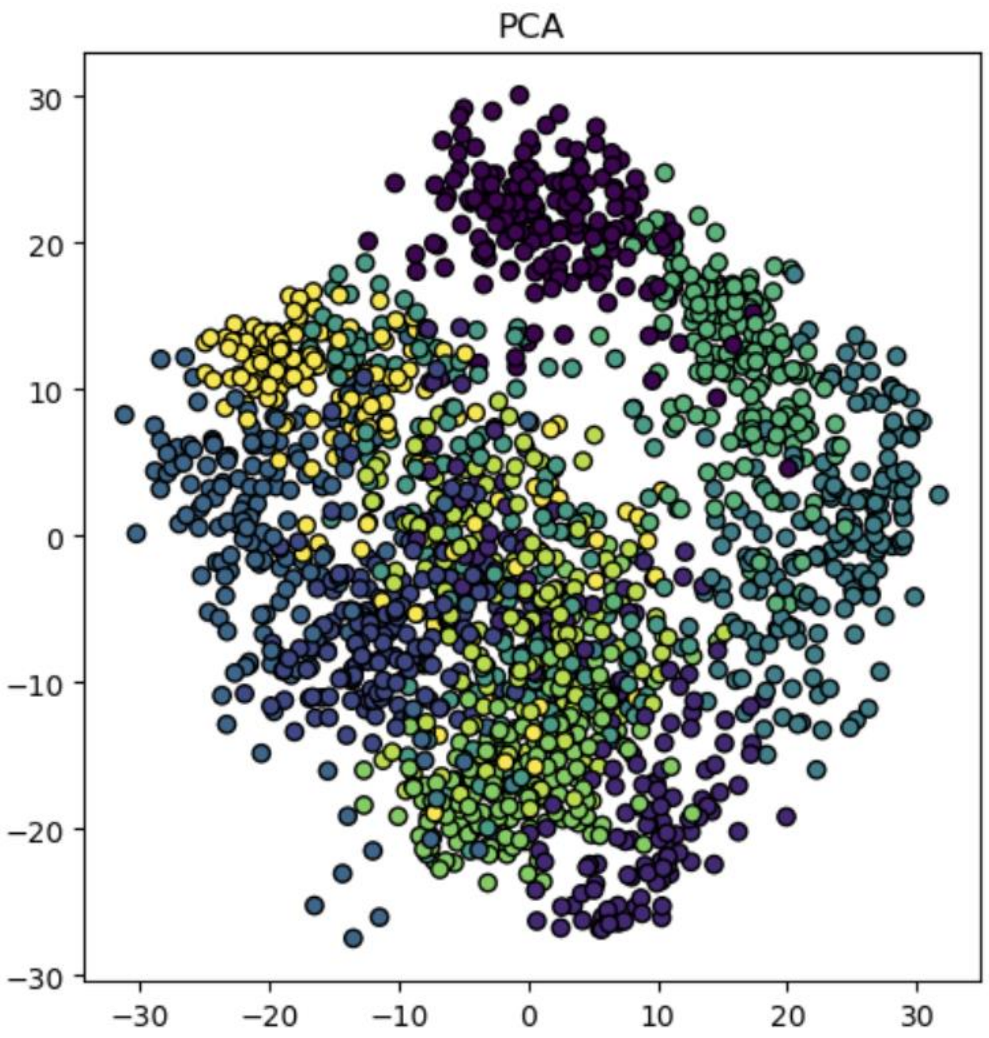
Example

- Digits Dataset

Each datapoint is a 8x8 image of a digit.

Classes	10
Samples per class	~180
Samples total	1797
Dimensionality	64
Features	integers 0-16

Example: 1st two PCs from PCA



https://medium.com/@conniezhou678/unveiling-dimensionality-reduction-a-comparative-analysis-of-9-methods-part-10-8594f8cd4336

Potential issues with PCA?

New idea

- Preserve high-dimensional pairwise distance in projection
  - “similar” stuff stays close together
  - “different” stuff can move apart

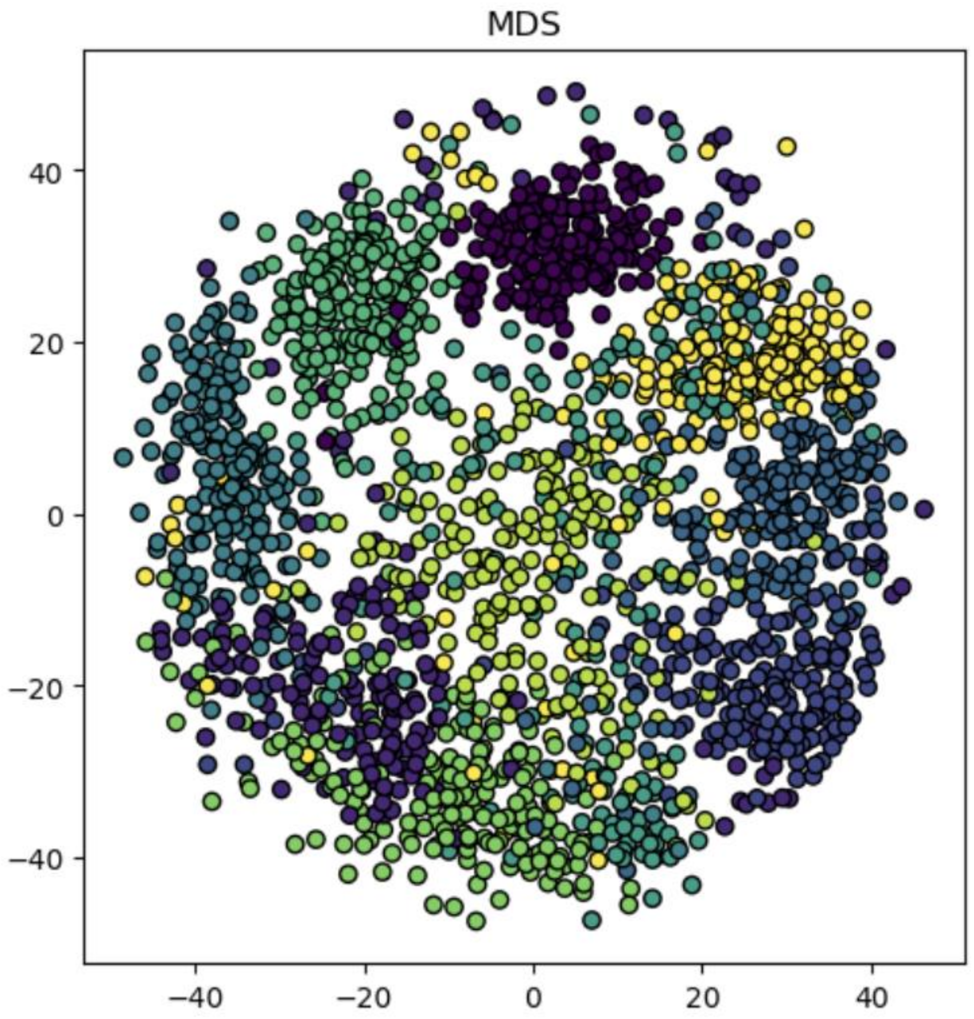
MDS (Multidimensional Scaling)

1. Choose a good distance metric
2. Compute a pairwise distance matrix
3. Find a 2D embedding that preserves those distances- (the distance matrix kind of acts like a stress tensor, so you can think of MDS kind of like a "force directed" layout but without visible edges)

Fun (but also sad) fact:

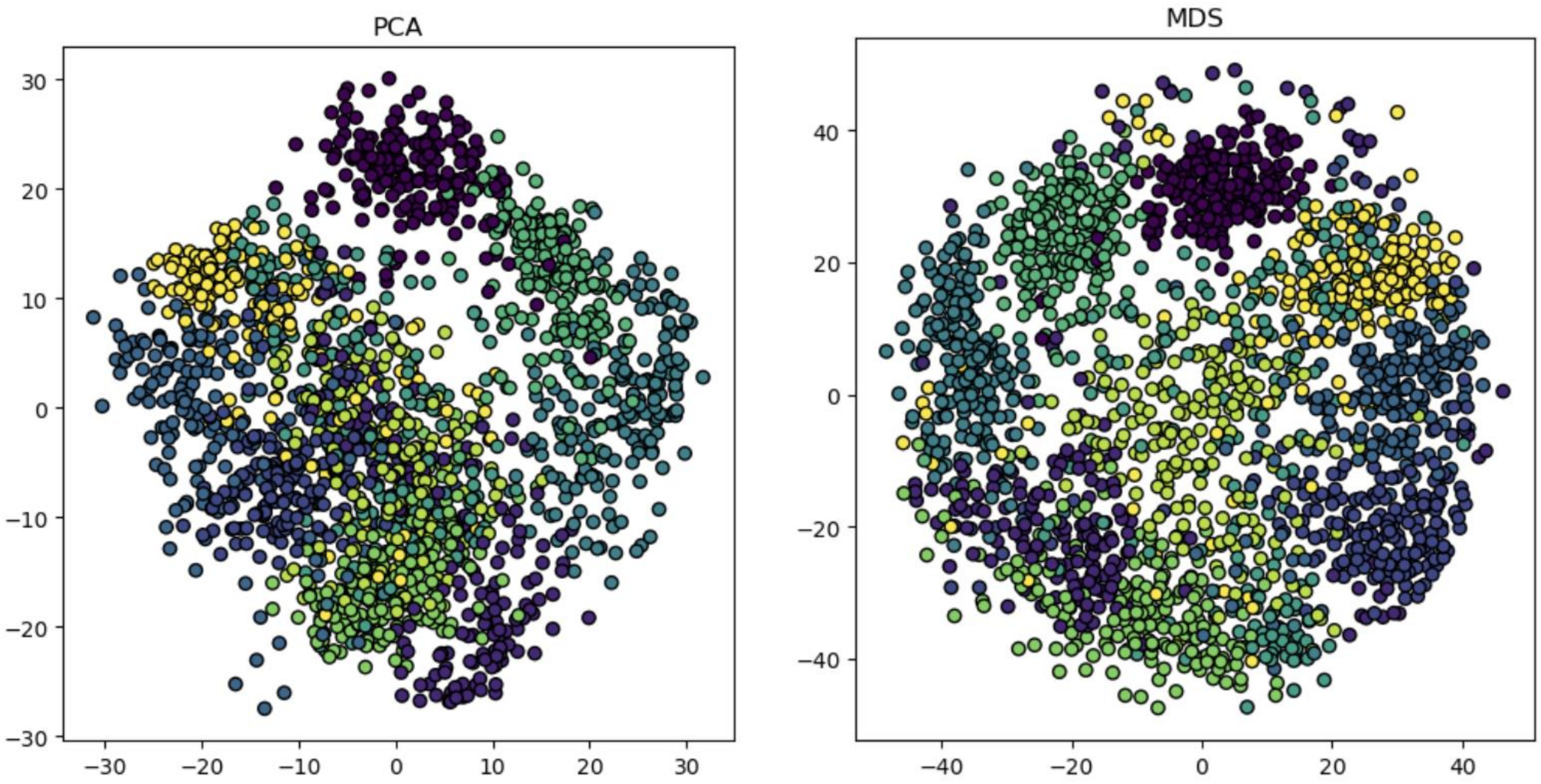
- PCA is just a special case of MDS (if we use Euclidean distance and choose the 1st two components)

Example: 1st two PCs from MDS



https://medium.com/@conniezhou678/unveiling-dimensionality-reduction-a-comparative-analysis-of-9-methods-part-10-8594f8cd4336

Example: 1st two PCs from PCA vs MDS



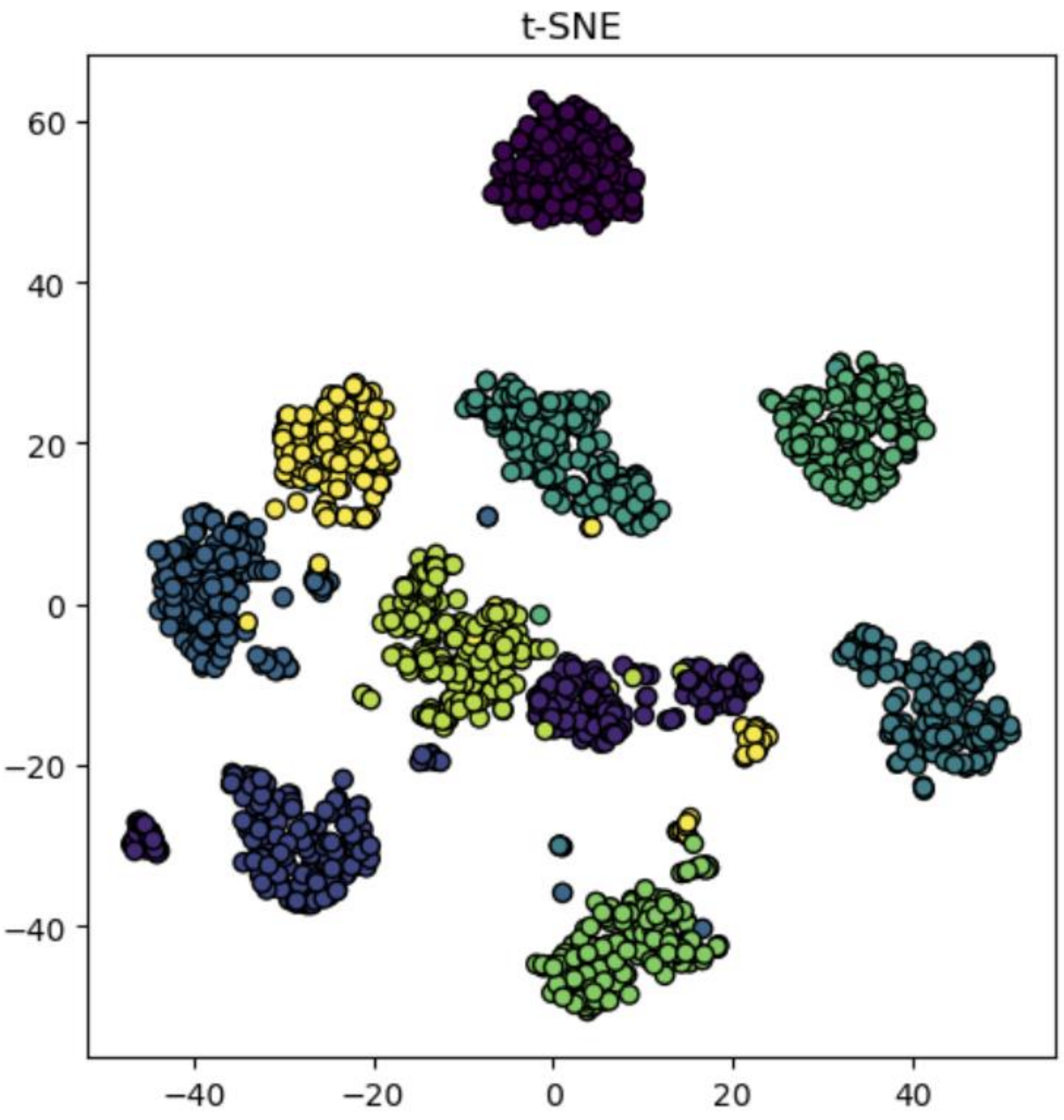
https://medium.com/@conniezhou678/unveiling-dimensionality-reduction-a-comparative-analysis-of-9-methods-part-10-8594f8cd4336

Potential issues with general MDS?

t-SNE (t-Distributed Stochastic Neighbor Embedding)

- Same objective: preserve pairwise distances
- Different approach:
  - Similarity of points is determined by a probability distribution (t-distribution) and the conditional probability the point A will pick point B as it's "neighbor"
  - Density of points is taken into account via "perplexity"
  - Stochastic process (element of randomness)

Example: 1st two PCs from t-SNE



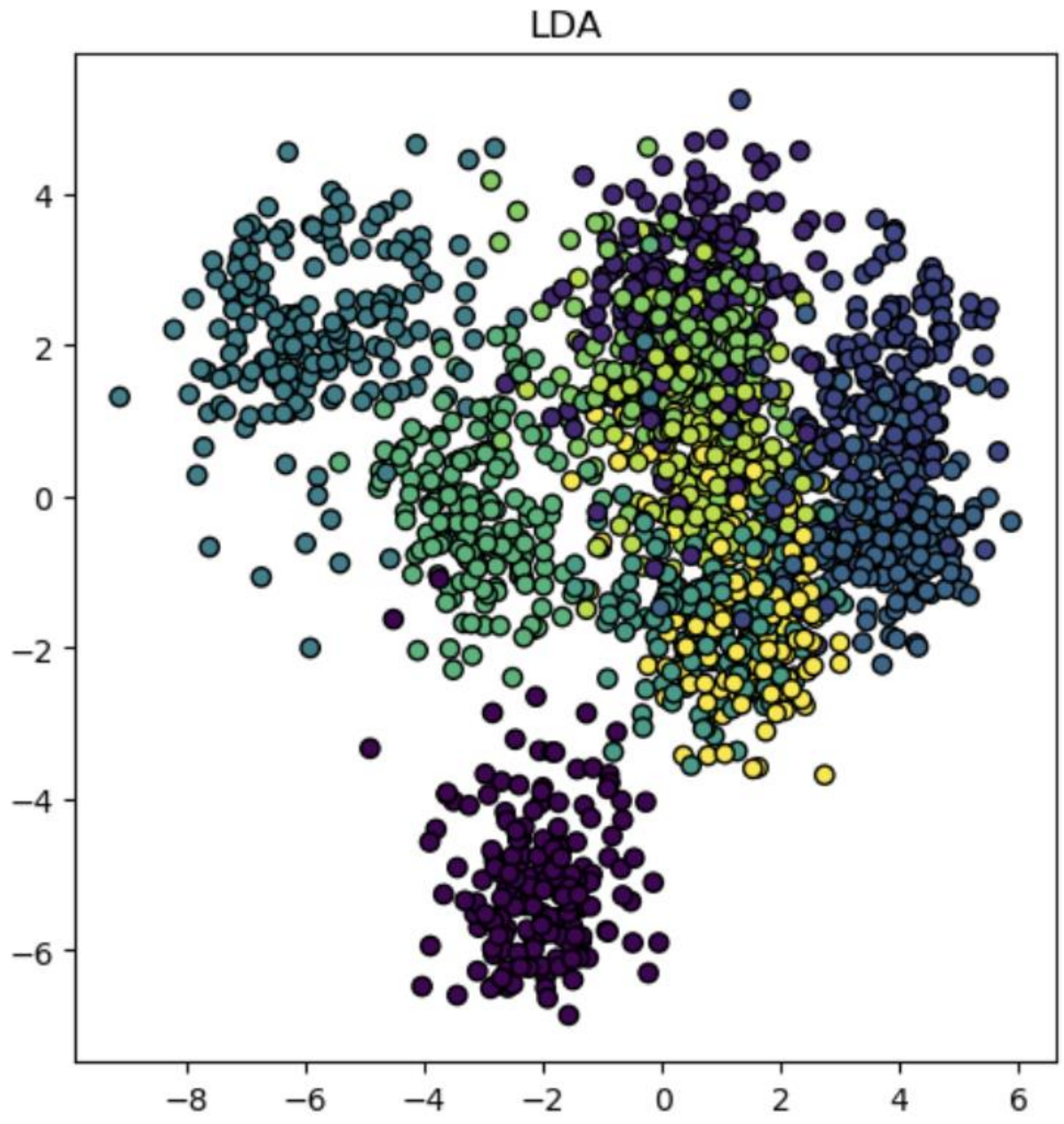
https://medium.com/@conniezhou678/unveiling-dimensionality-reduction-a-comparative-analysis-of-9-methods-part-10-8594f8cd4336

Potential issues with t-SNE?

LDA (Linear Discriminant Analysis)

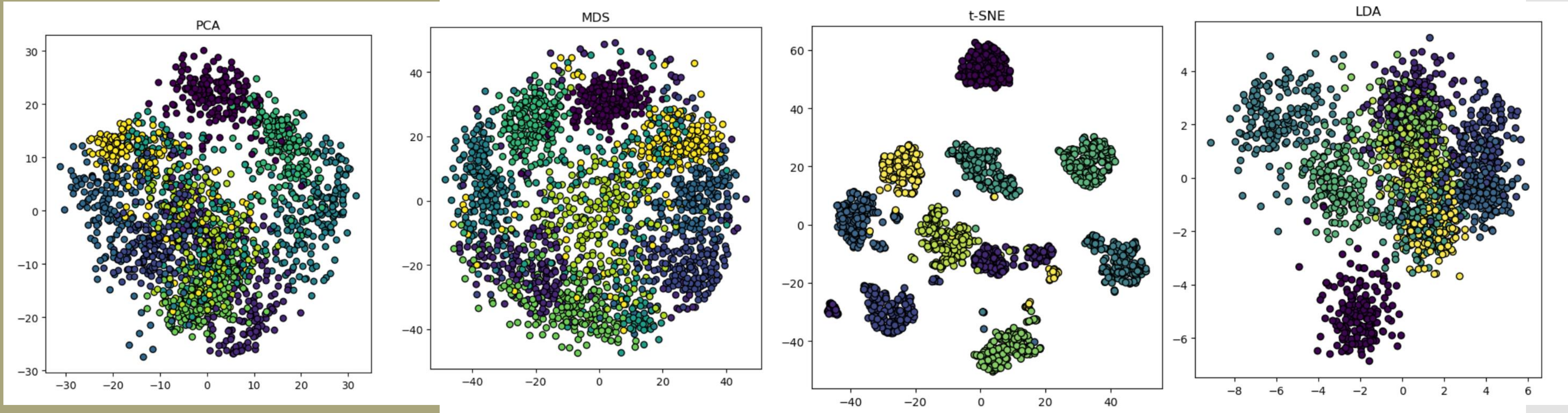
- Prioritizes class separability
- From a dataset of d independent features, extracts k new independent features that separate the classes the most
  - Note: you need to know classes to use LDA
- Algorithm is similar to PCA, but with the added constraints of minimizing inter-class spread and maximizing intra-class spread

Example: 1st two PCs from LDA



https://medium.com/@conniezhou678/unveiling-dimensionality-reduction-a-comparative-analysis-of-9-methods-part-10-8594f8cd4336

Potential issues with LDA?



https://medium.com/@conniezhou678/unveiling-dimensionality-reduction-a-comparative-analysis-of-9-methods-part-10-8594f8cd4336

Takeaways

- There's no "one right answer"
- Each technique has benefits and drawbacks:
  - MDS (or PCA) is great if you eventually need to be able to relate the result back to the original dimensions
  - t-SNE does a great job preserving local similarity, but sometimes at the expense of global structure (and it can get really slow)
  - LDA is fast and relatively accurate