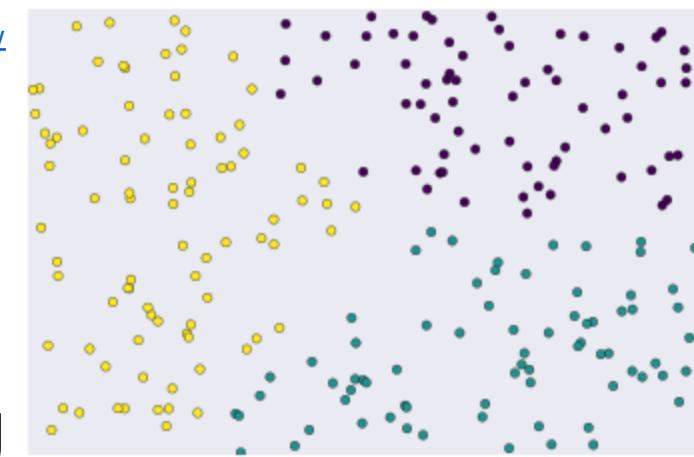


# Unsupervised Learning

Tuesday, Lecture 2  
FASE ML Bootcamp

Based on material from:  
Elements of Statistical Learning by Hastie, Tibshirani, Friedman,  
Nina Balca, CMU 10-601  
Google's ML course  
Elena Sugi's slides  
Derek Hoeim's slides

# Clustering

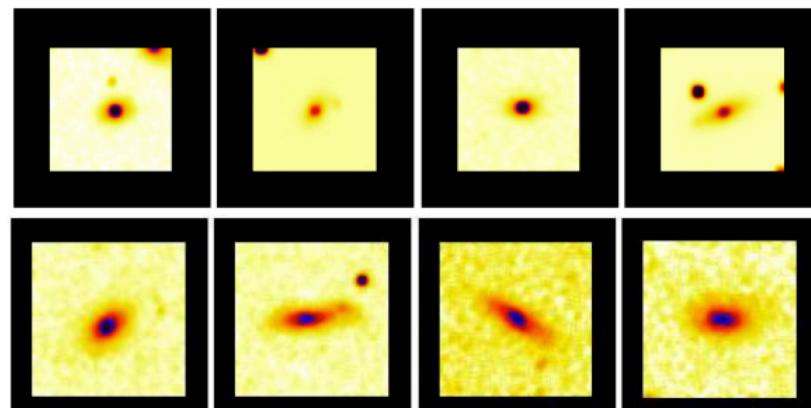


The most common type of **unsupervised** learning

Goal: group “**similar**” data points together

**Unsupervised** because we don’t label the data as we did in classification/regression: let the features speak for themselves!

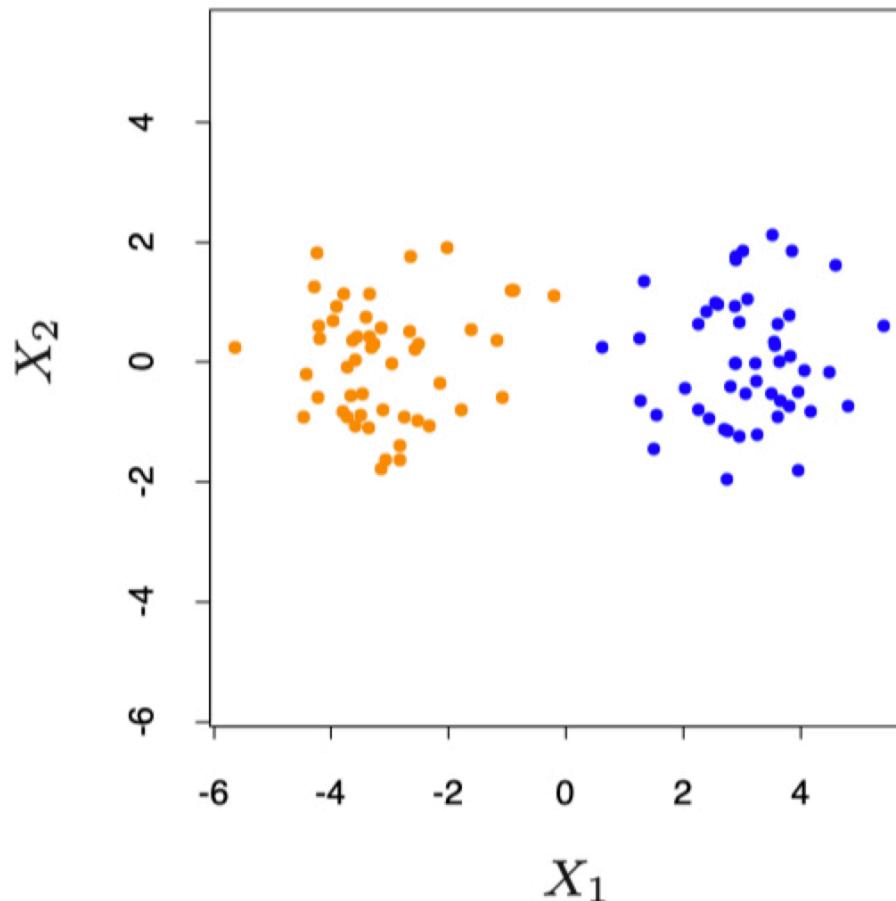
Clustering galaxies, from Miller et al. (2005)



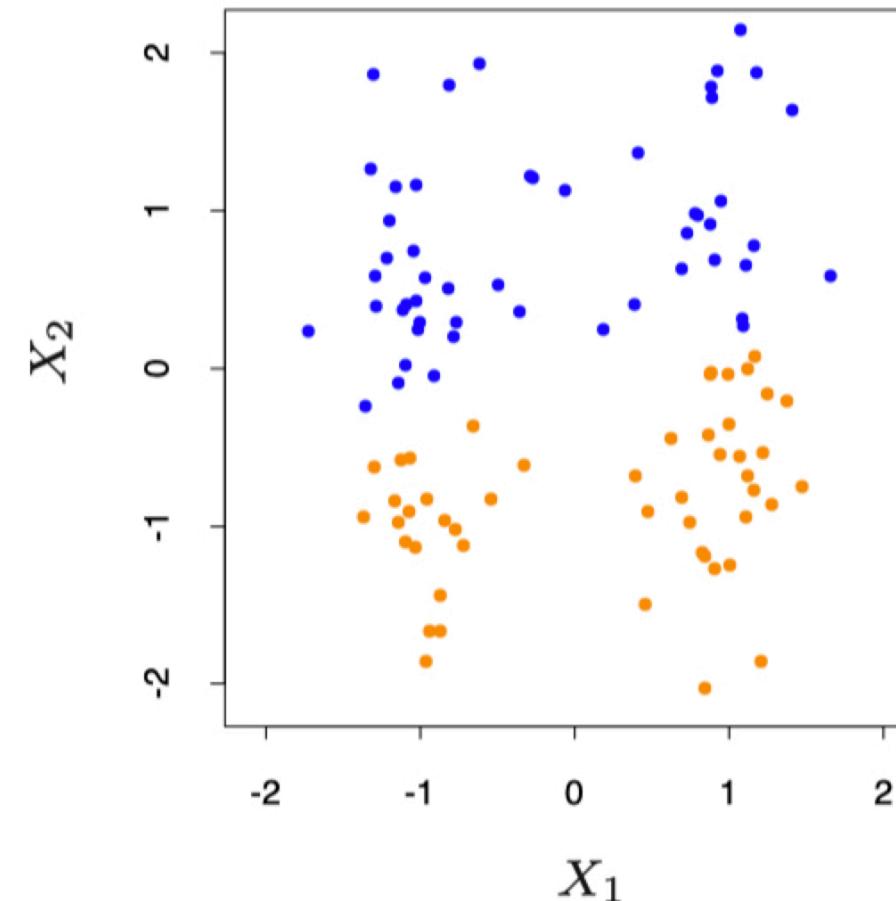
# Data Preparation for Clustering

The points are colored by a clustering algorithm

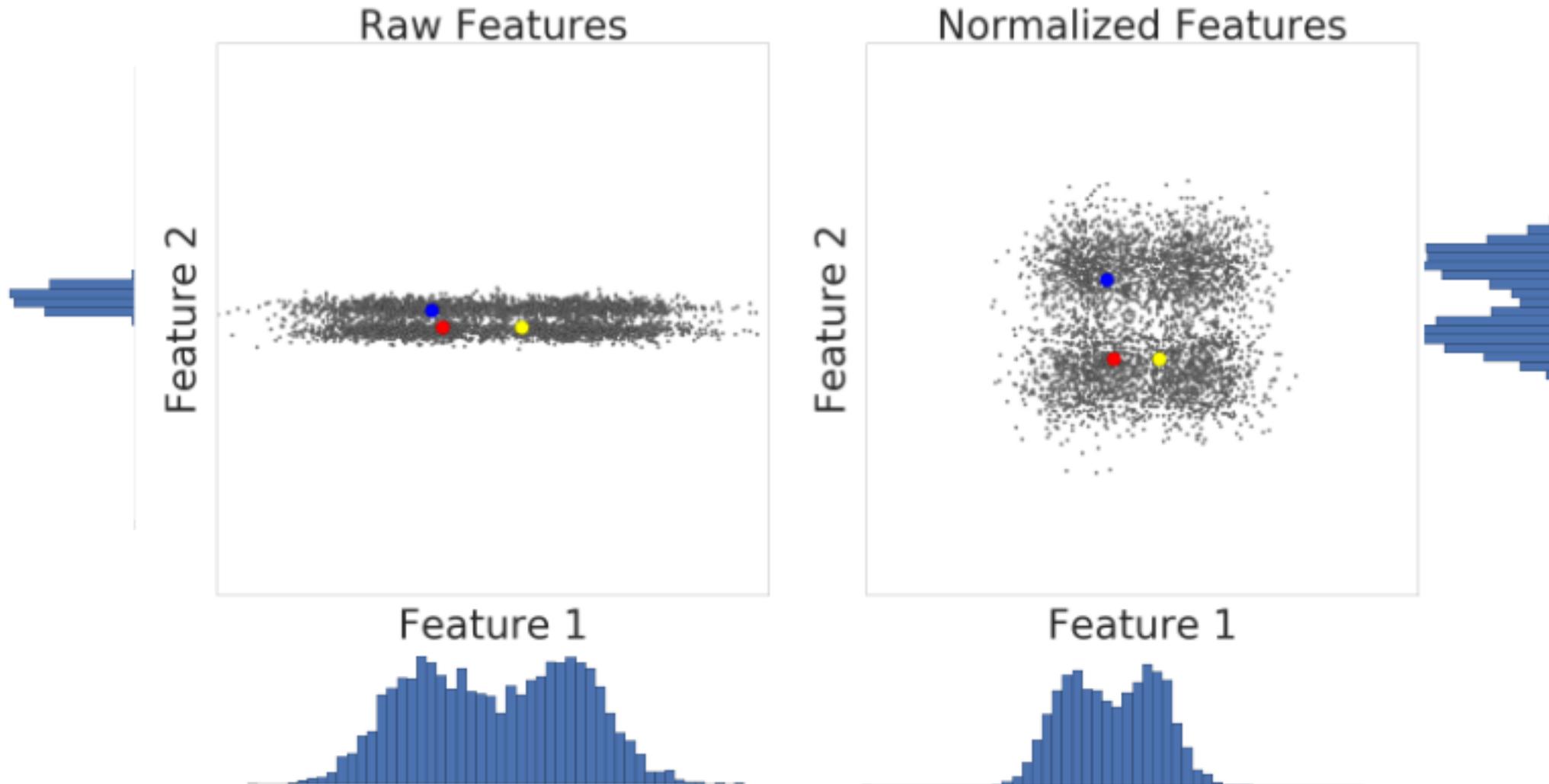
Raw data



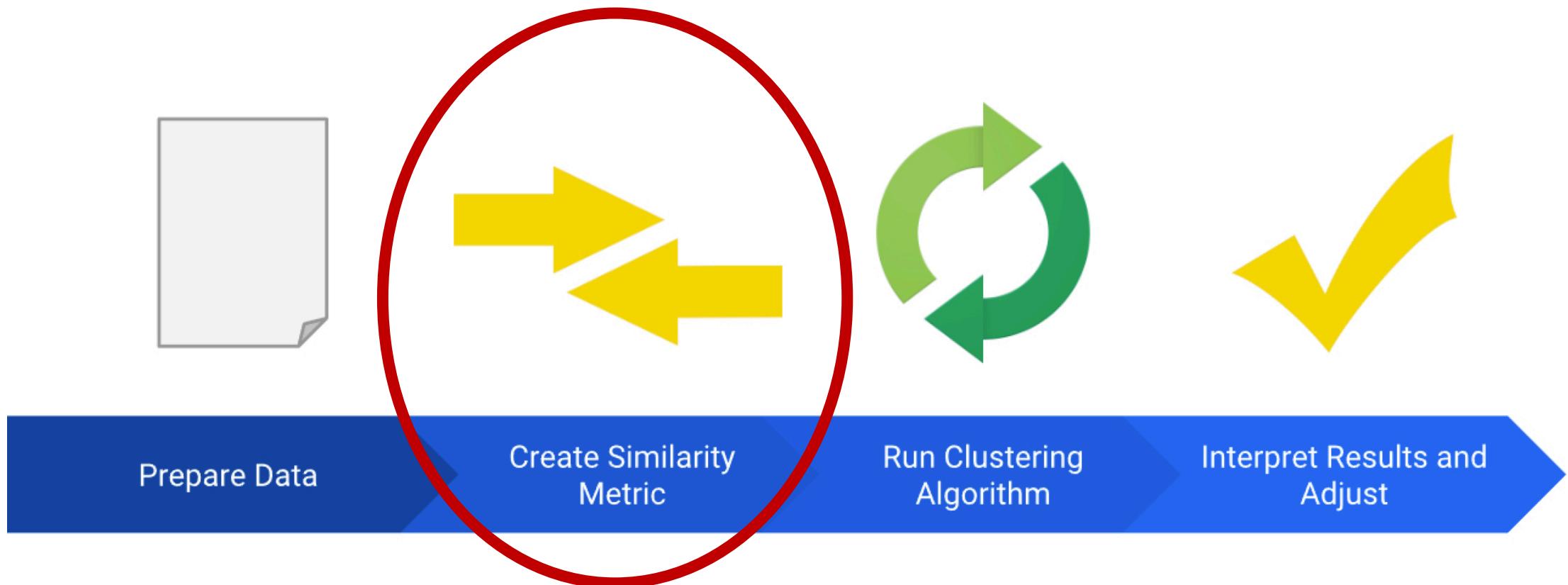
Standardized data



# Data Preparation for Clustering



# Clustering workflow



From Google's Clustering lesson: <https://developers.google.com/machine-learning/clustering/>

# Distance Metrics

The right distance metric depends on your application!

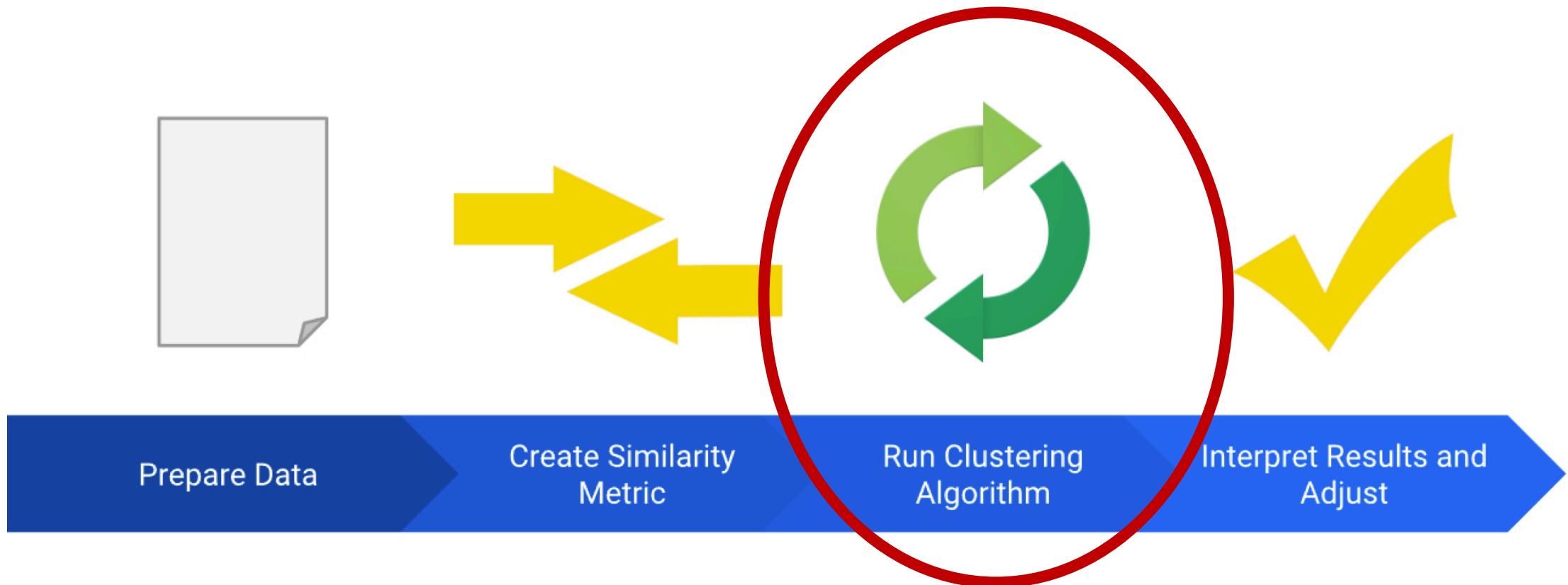
**Distance of vectors**  $x = (x_1, \dots, x_n)$  and  $y = (y_1, \dots, y_n)$

- **Euclidean distance**  $d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$
- **Manhattan distance**  $d(x, y) = \sum_{i=1}^n |x_i - y_i|$
- **Correlation distance**  $d(x, y) = 1 - r(x, y)$   $r(x, y)$  is Pearson correlation coefficient

**Distance of sequences** ACCTTG and TACCTG

- **Hamming distance** ACCTTG => 3  
TACCTG
- **Levenshtein distance** .ACCTTG => 2  
TACC\_.TG

# Clustering workflow

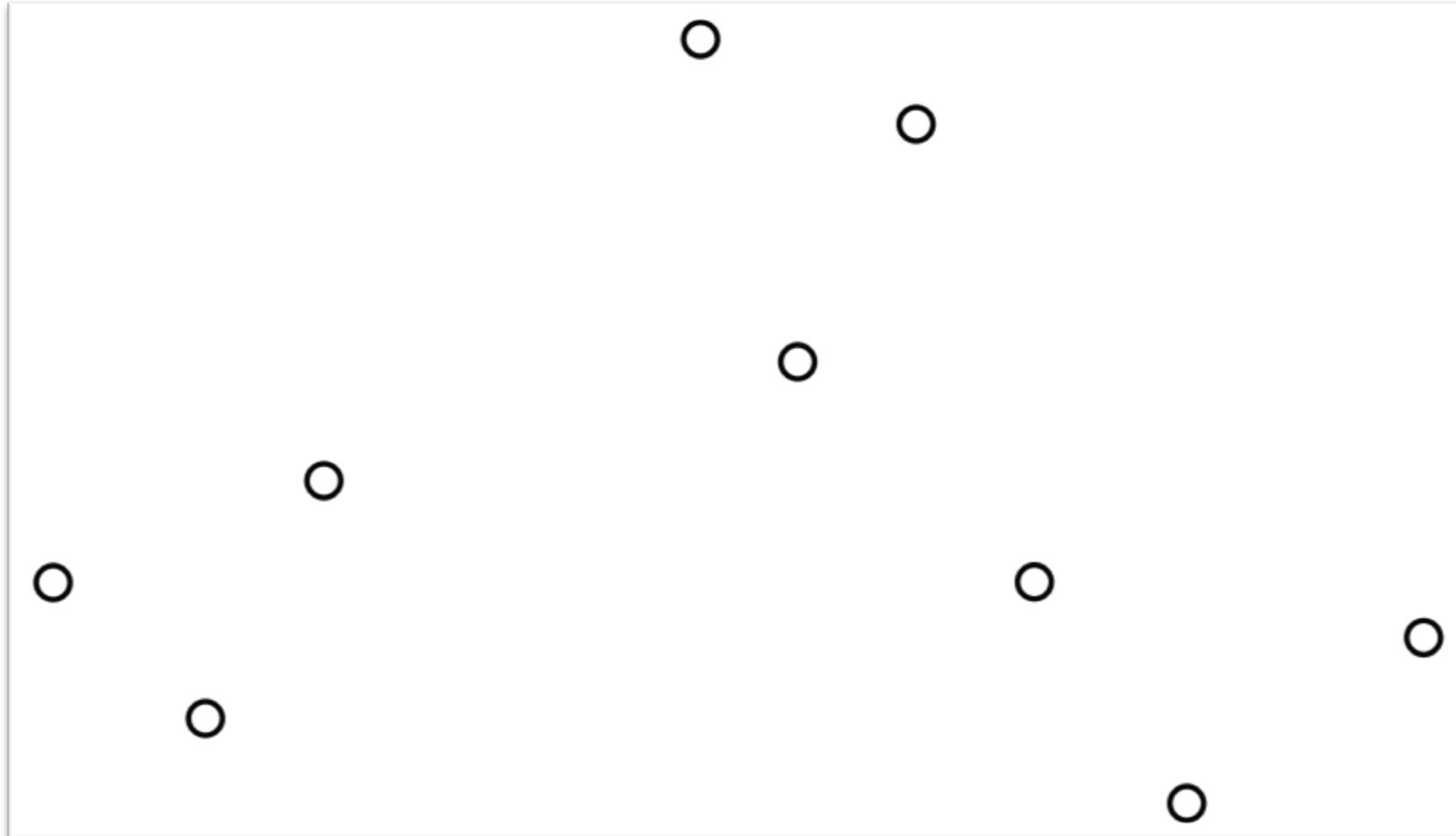


From Google's Clustering lesson: <https://developers.google.com/machine-learning/clustering/>

# K-means Clustering

Euclidean distance metric

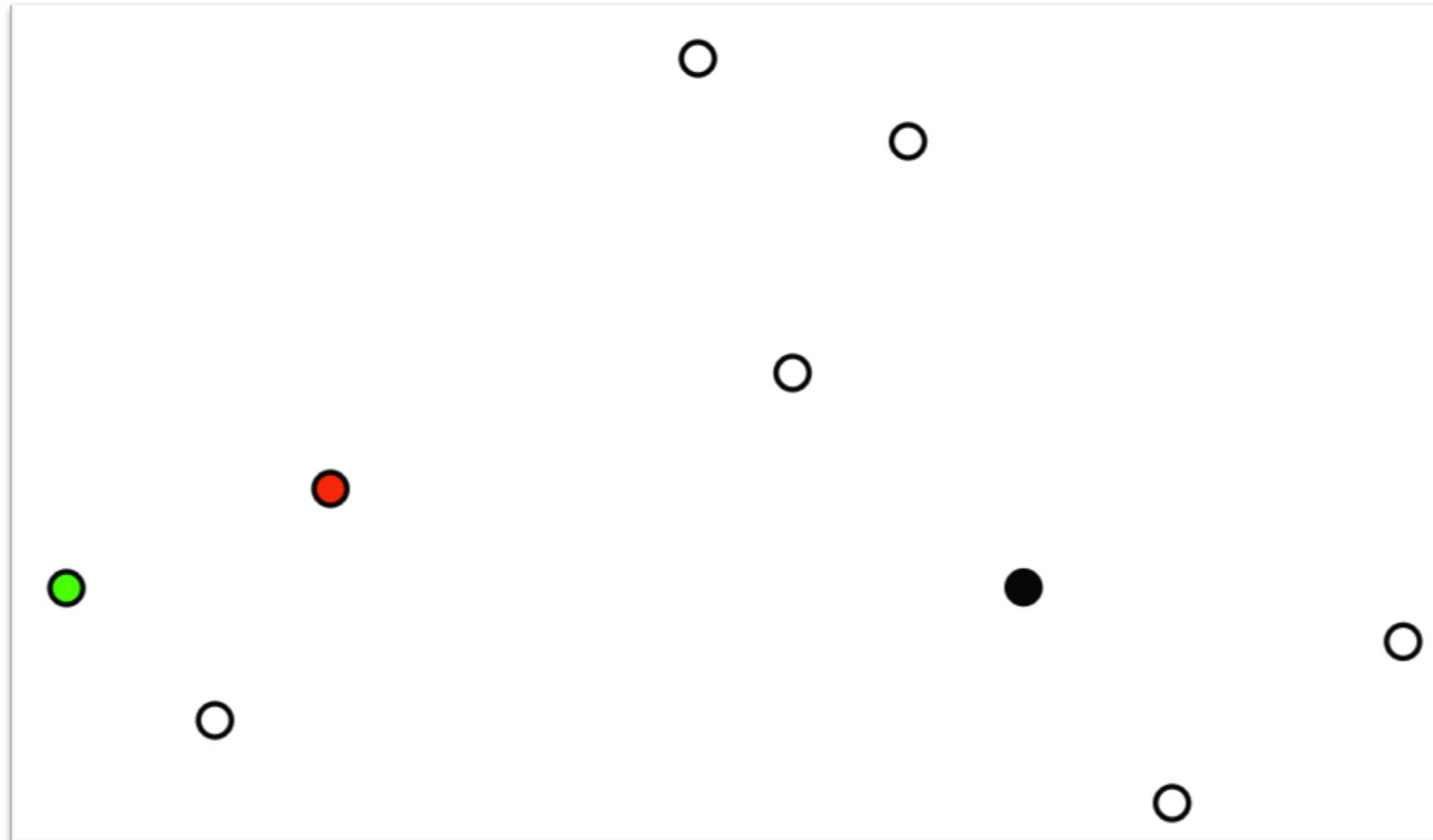
Given a set of data points...



# K-means Clustering

Euclidean distance metric

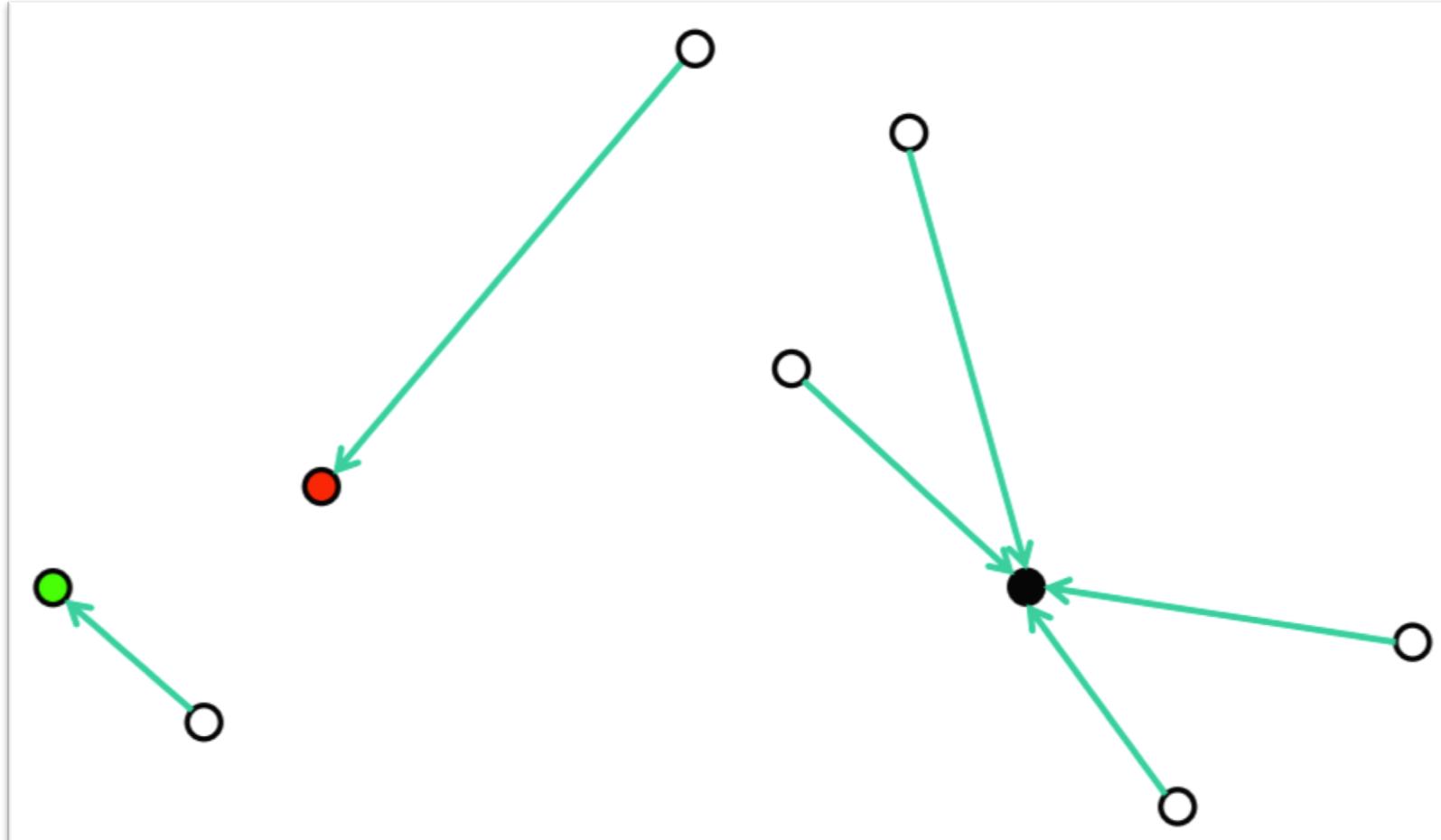
Select  $k=3$  **initial centers** at random



# K-means Clustering

Euclidean distance metric

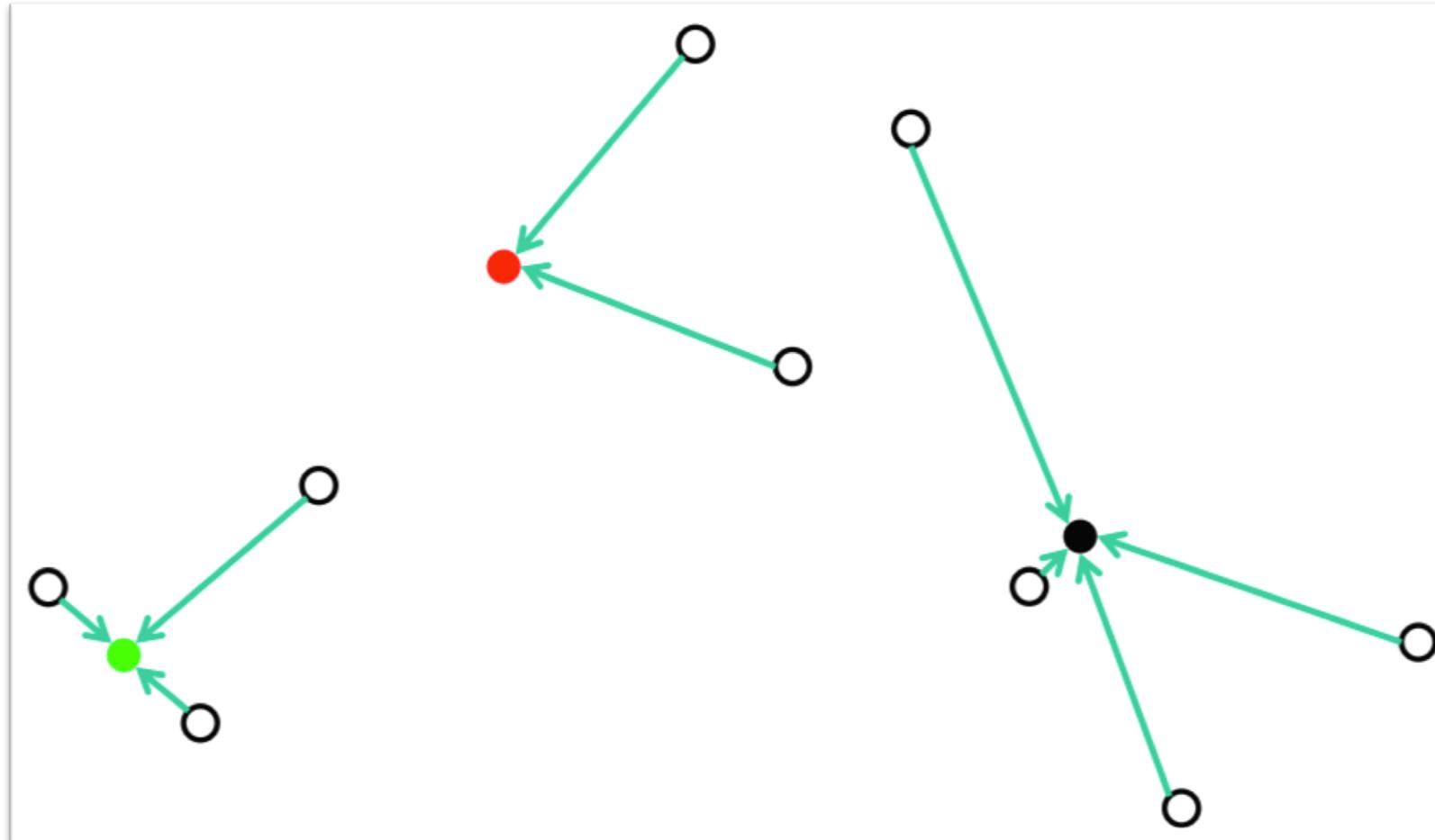
Recompute optimal centers given a fixed clustering



# K-means Clustering

Euclidean distance metric

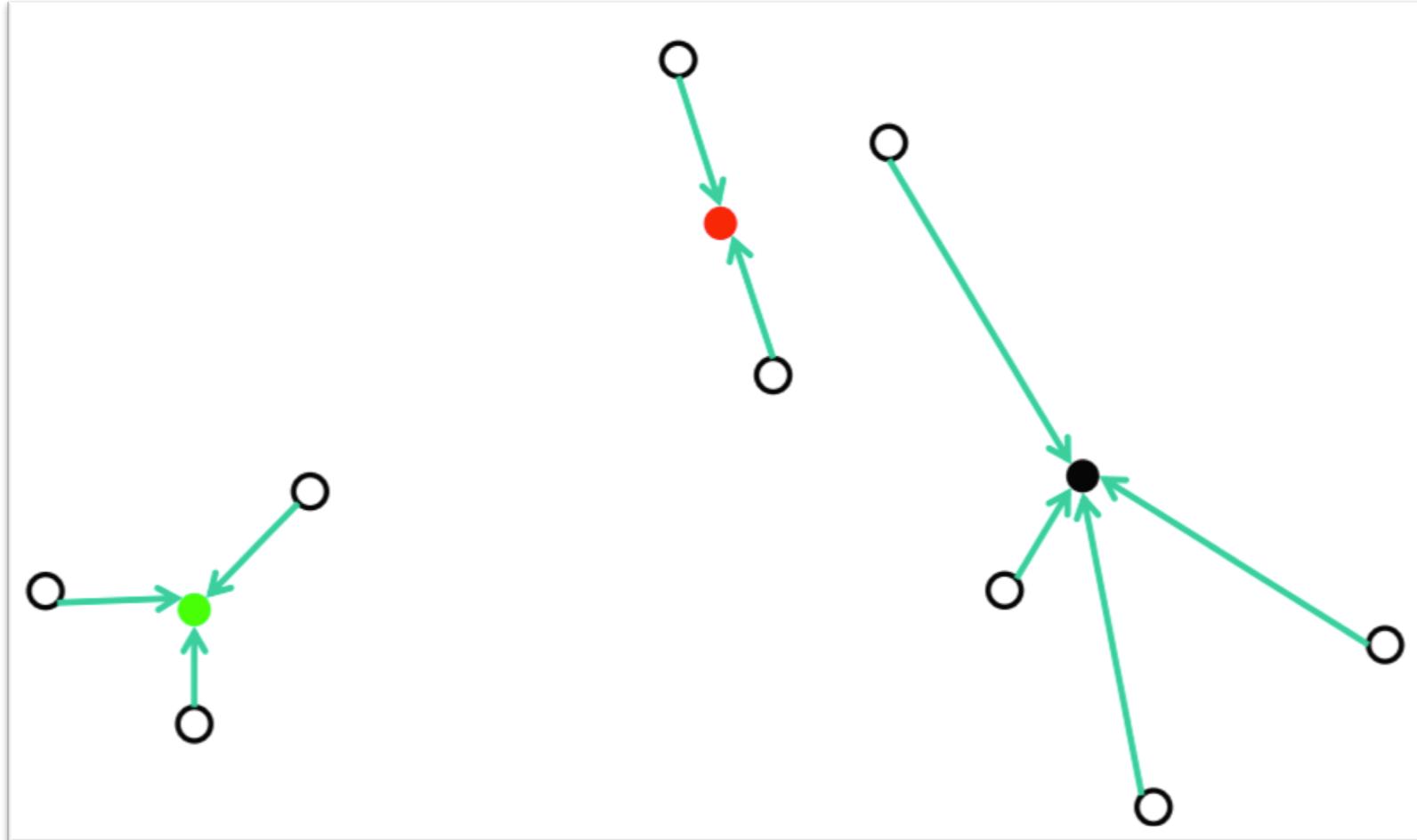
Assign each point to its nearest center



# K-means Clustering

Euclidean distance metric

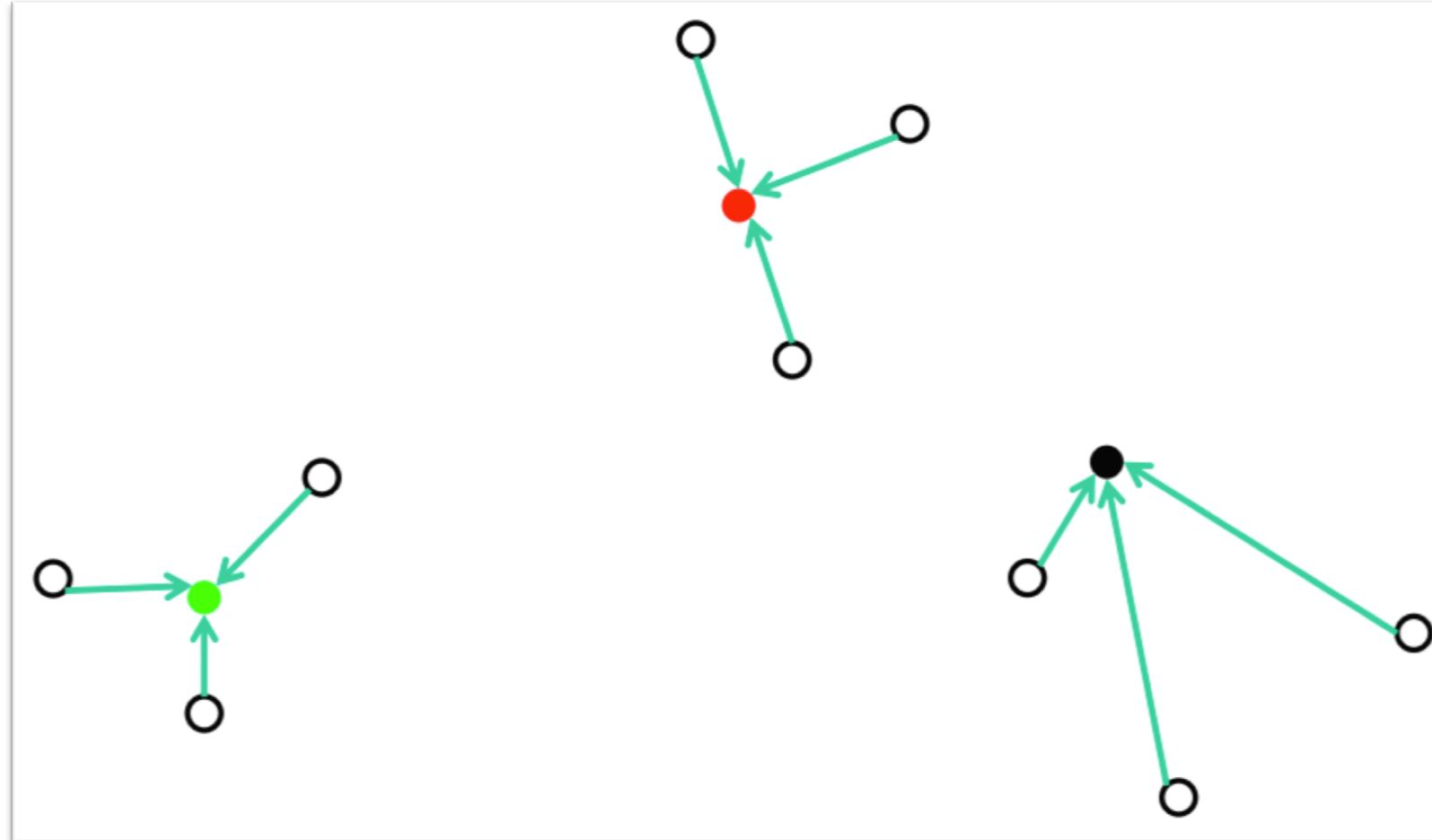
Recompute optimal centers given a fixed clustering



# K-means Clustering

Euclidean distance metric

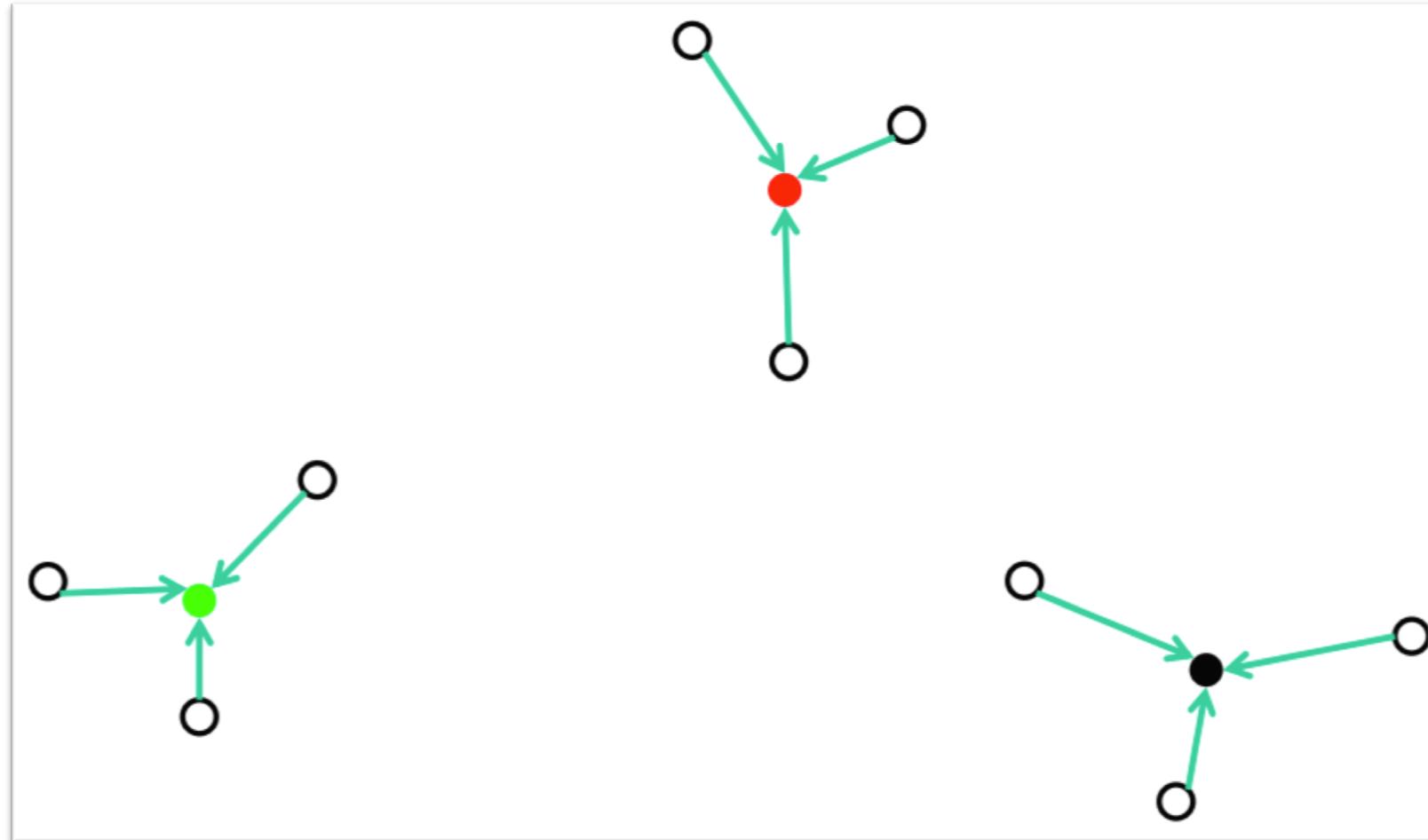
Assign each point to its nearest center



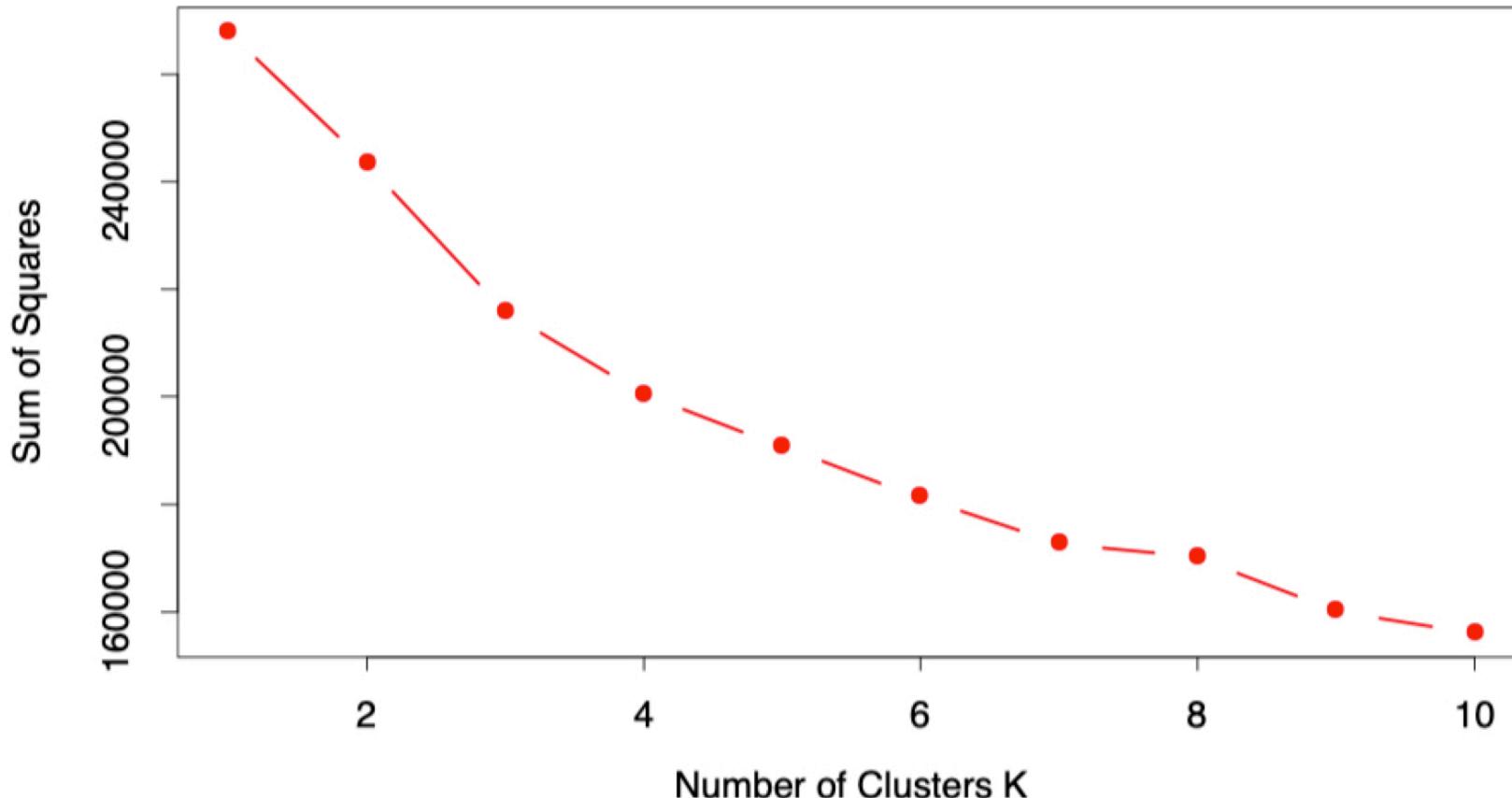
# K-means Clustering

Euclidean distance metric

Recompute optimal centers given a fixed clustering

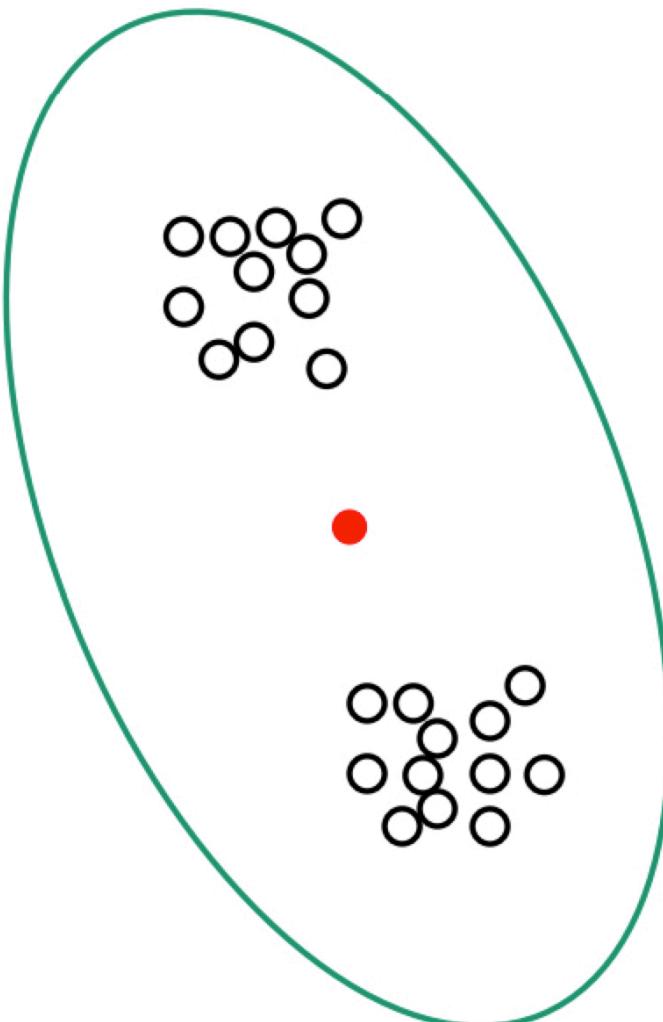


# Selecting the number of clusters

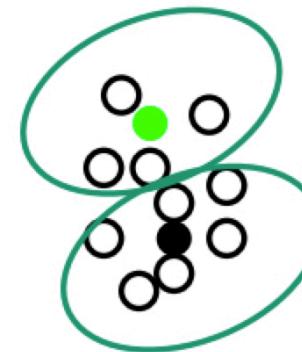


**FIGURE 14.8.** Total within-cluster sum of squares for  $K$ -means clustering applied to the human tumor microarray data.

# K-means can fail

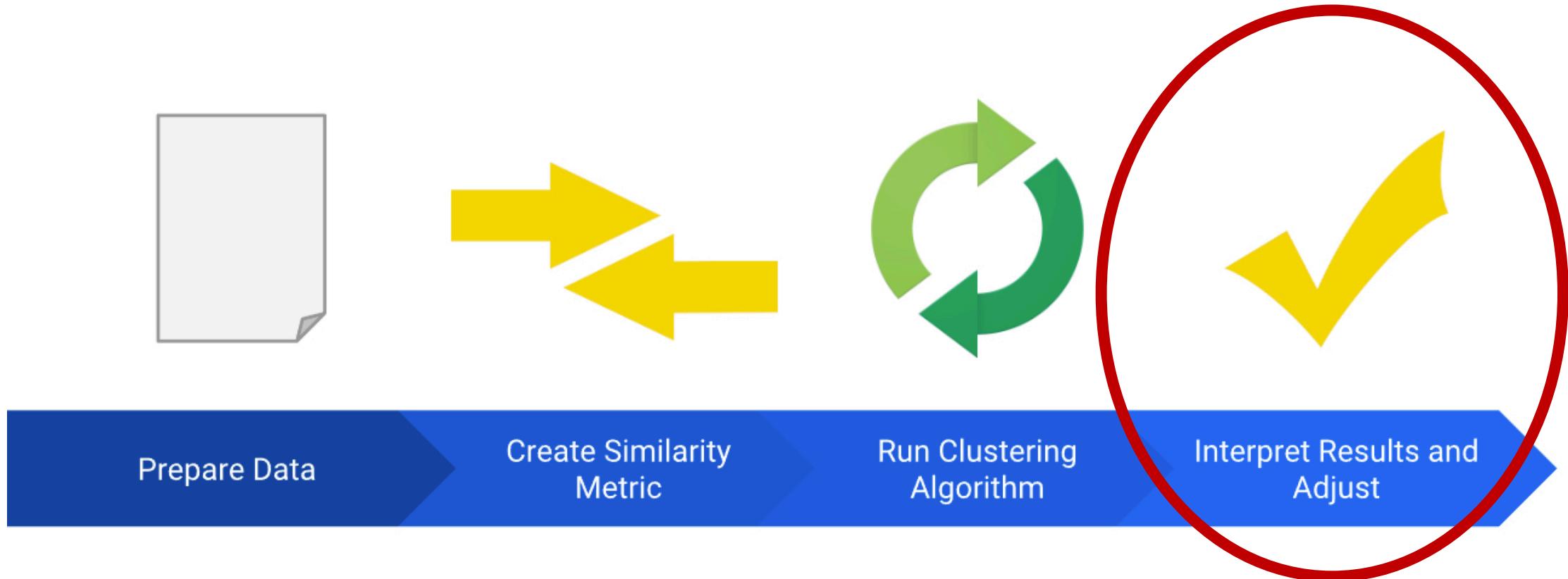


This is a heuristic!  
No guarantees it'll find optimum

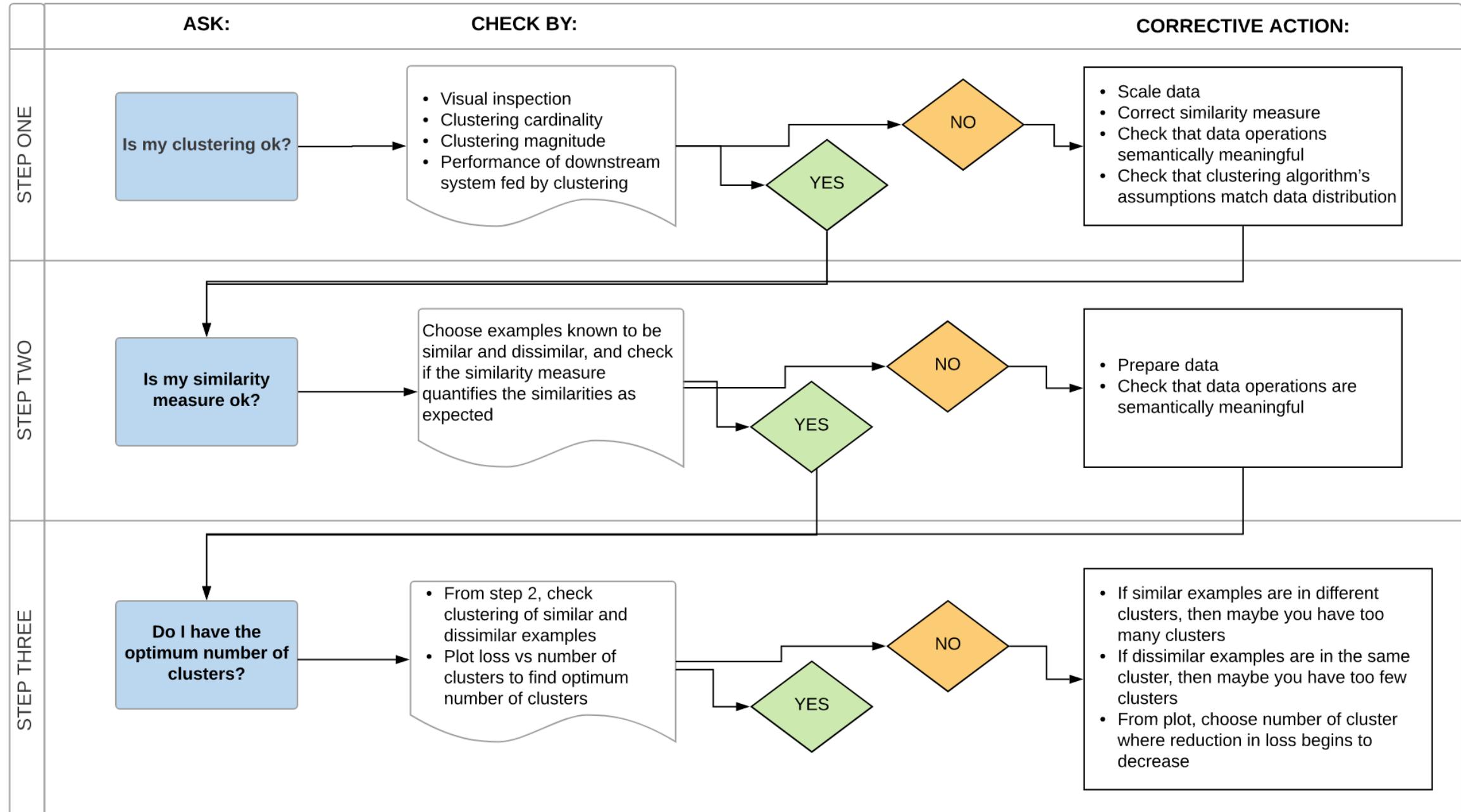


In practice, smarter centroid initialization solves this

# Clustering workflow

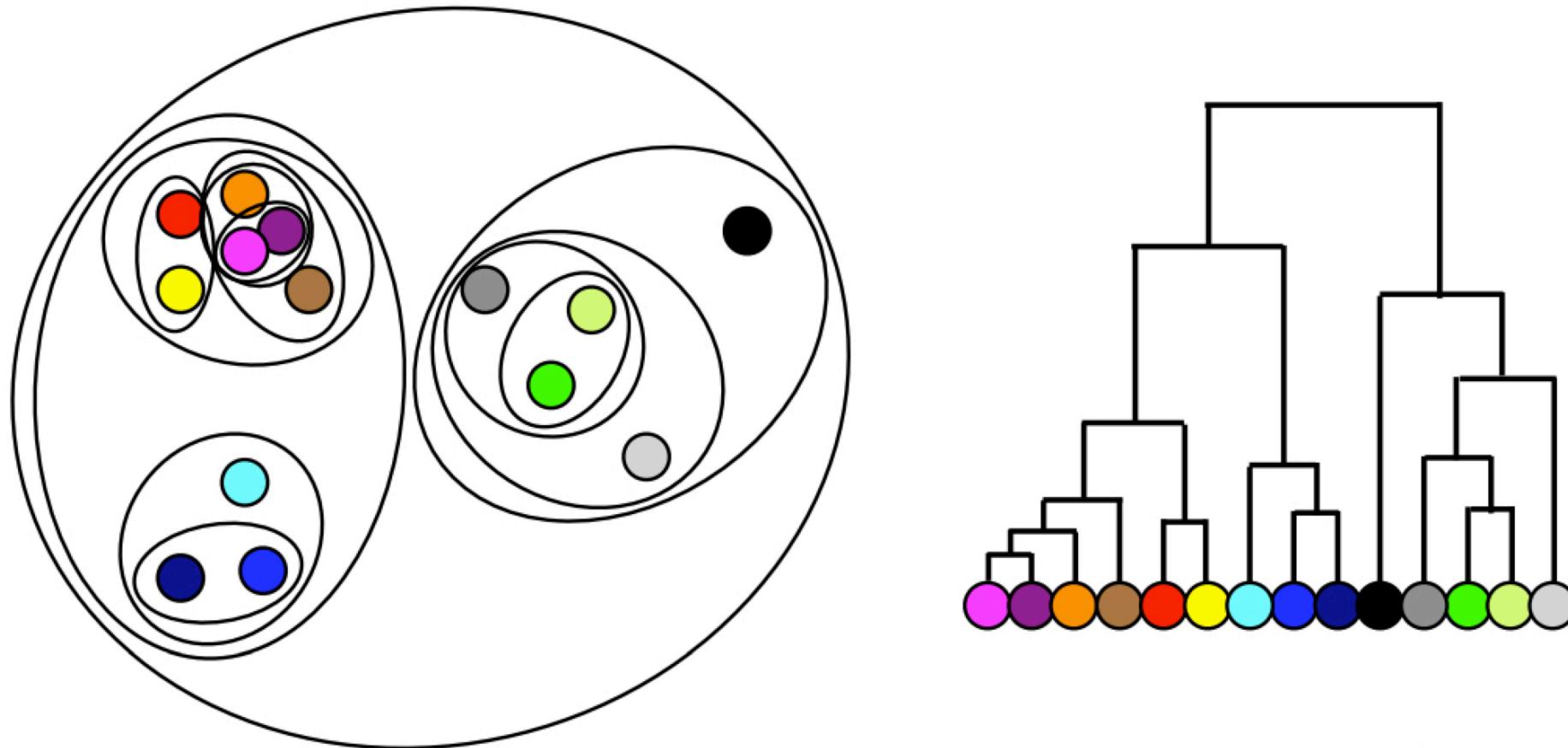


From Google's Clustering lesson: <https://developers.google.com/machine-learning/clustering/>



# Hierarchical Clustering

High-level idea: build a tree (hierarchy) of clusters

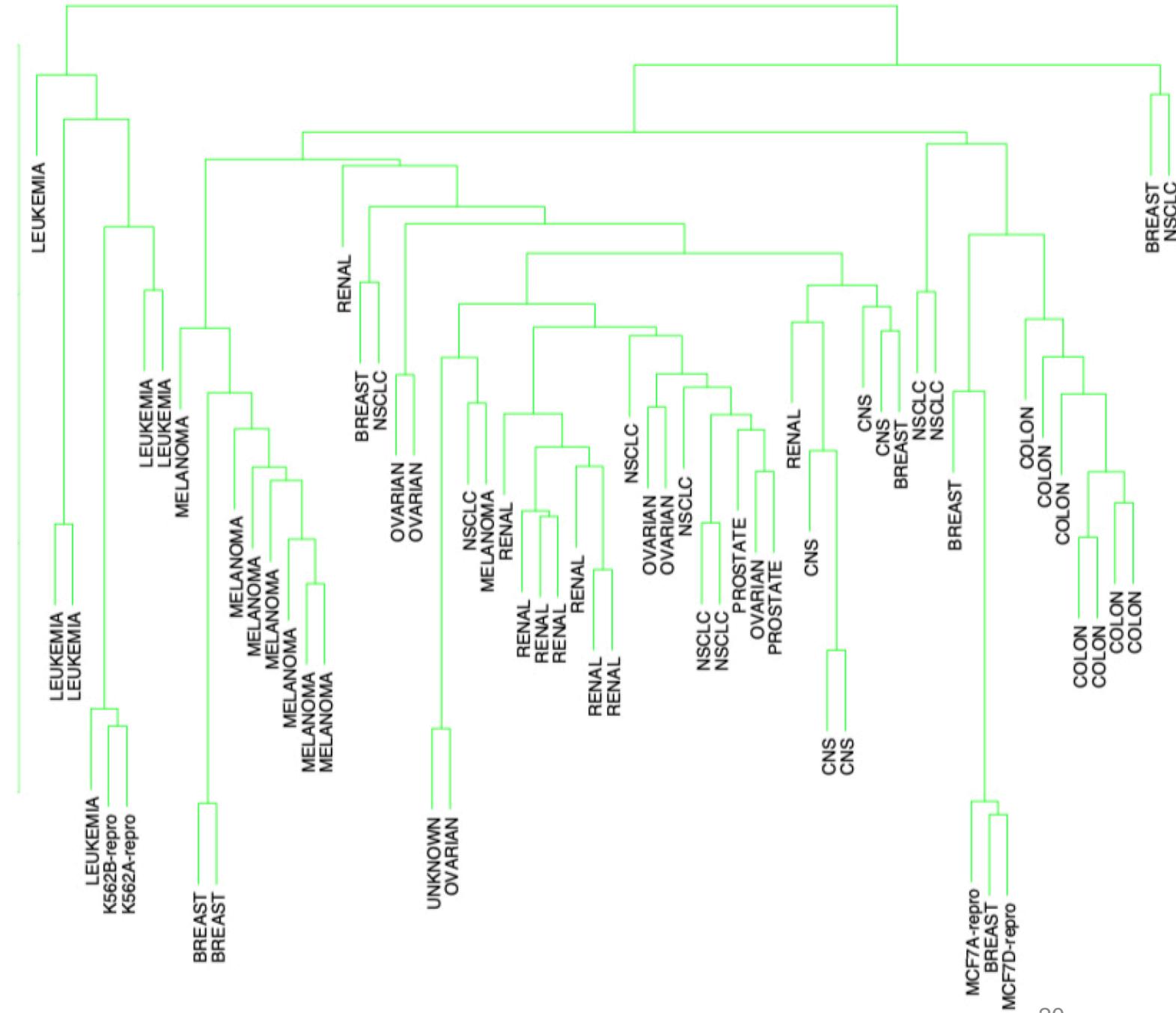


# Hierarchical Clustering

Human Tumor Microarray Data

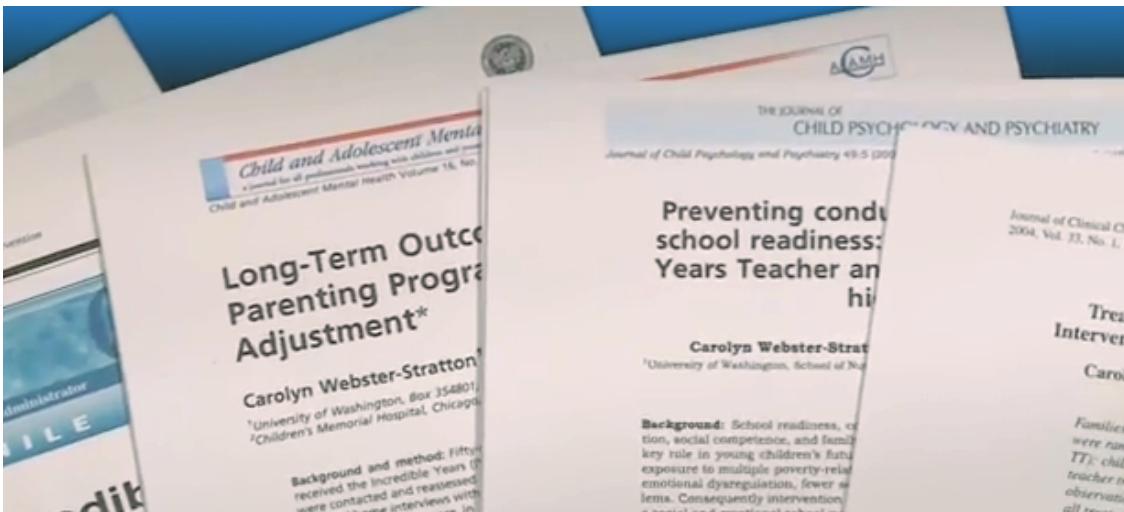
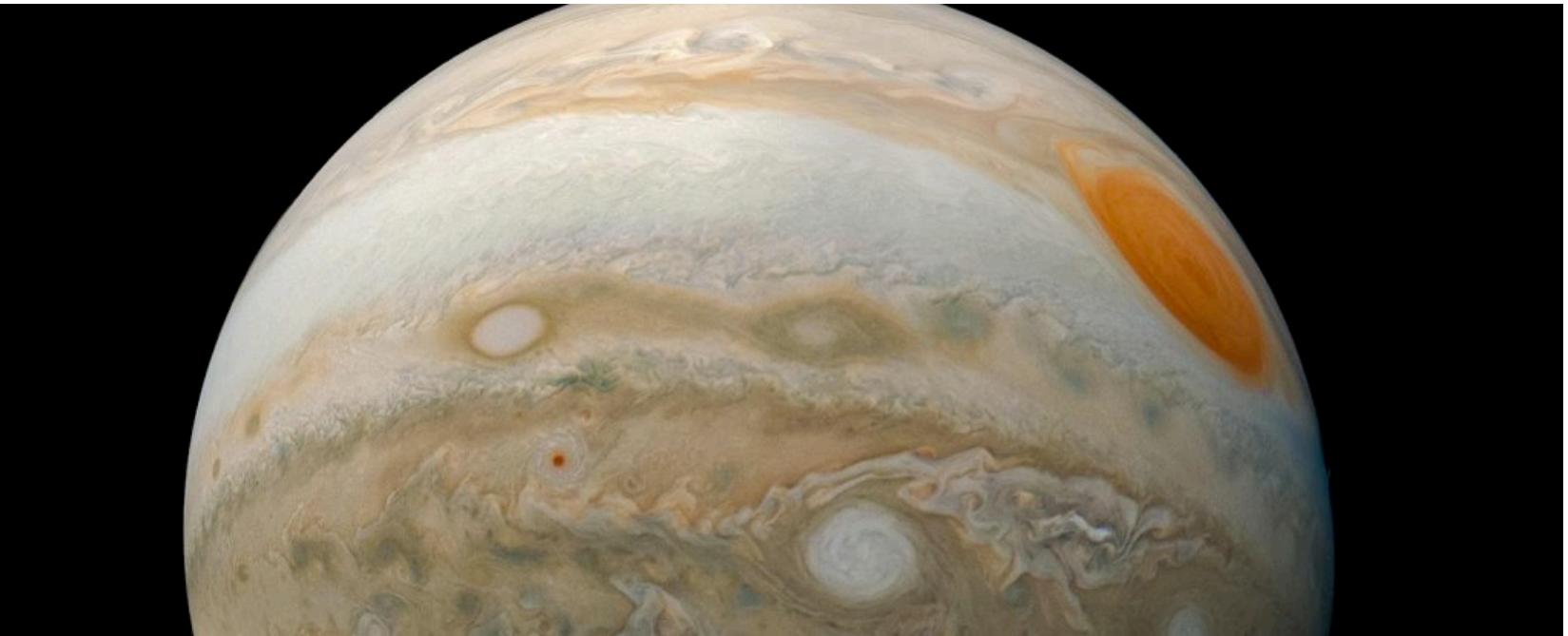
64 samples, 6830 features  
(gene expression levels)

Elements of Statistical Learning,  
Chapter 14.3.8



# Dimensionality Reduction

or dealing with very high-dimensional data



# Dimensionality Reduction



Examples: Principal Component Analysis (PCA), t-SNE, ...

Powerful unsupervised learning techniques for extracting **hidden** (potentially **lower dimensional**) structure from high dimensional datasets.

Useful for:

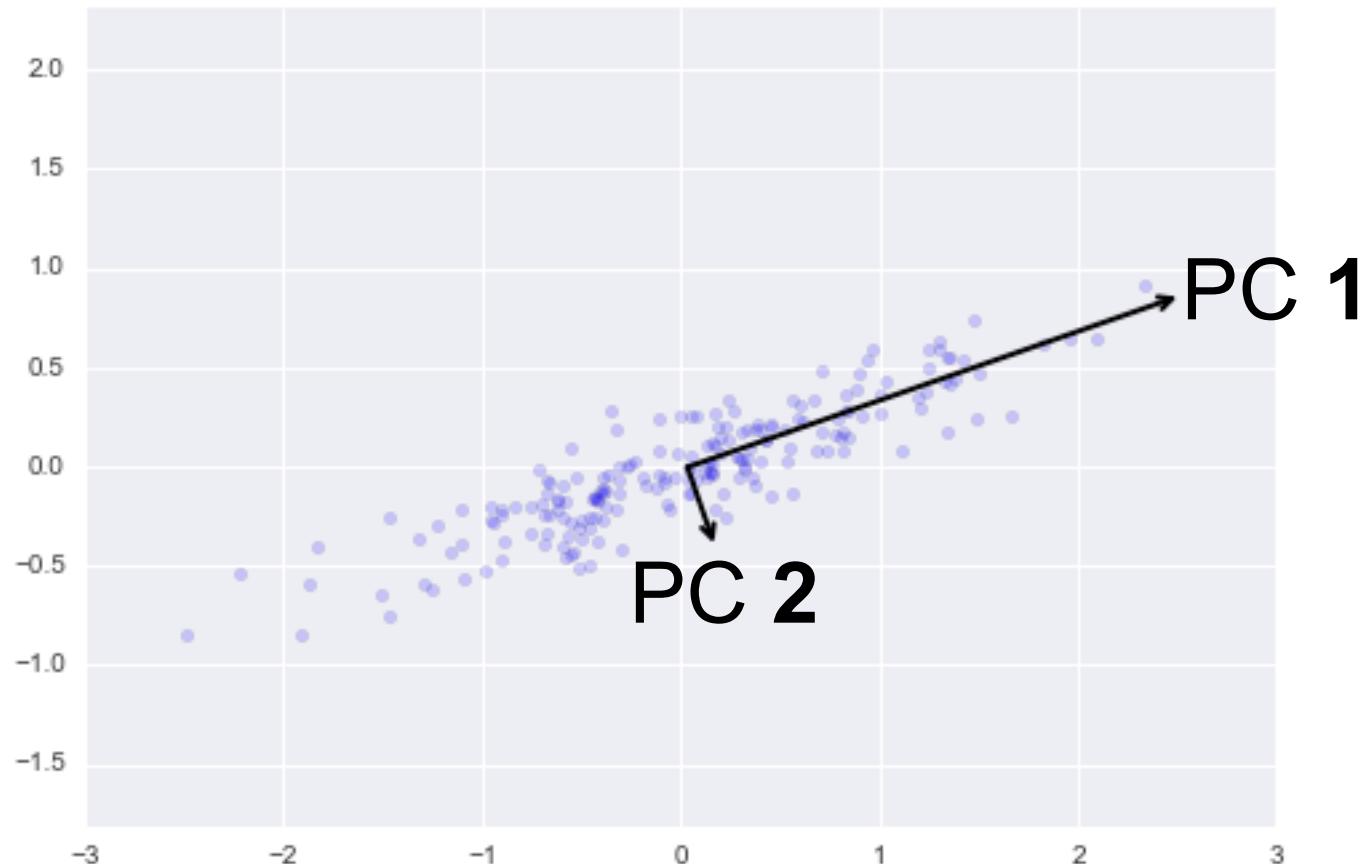
- **Visualization**
- Data **compression** for faster supervised learning
- **Noise removal**



Based on slide by Nina Balcan

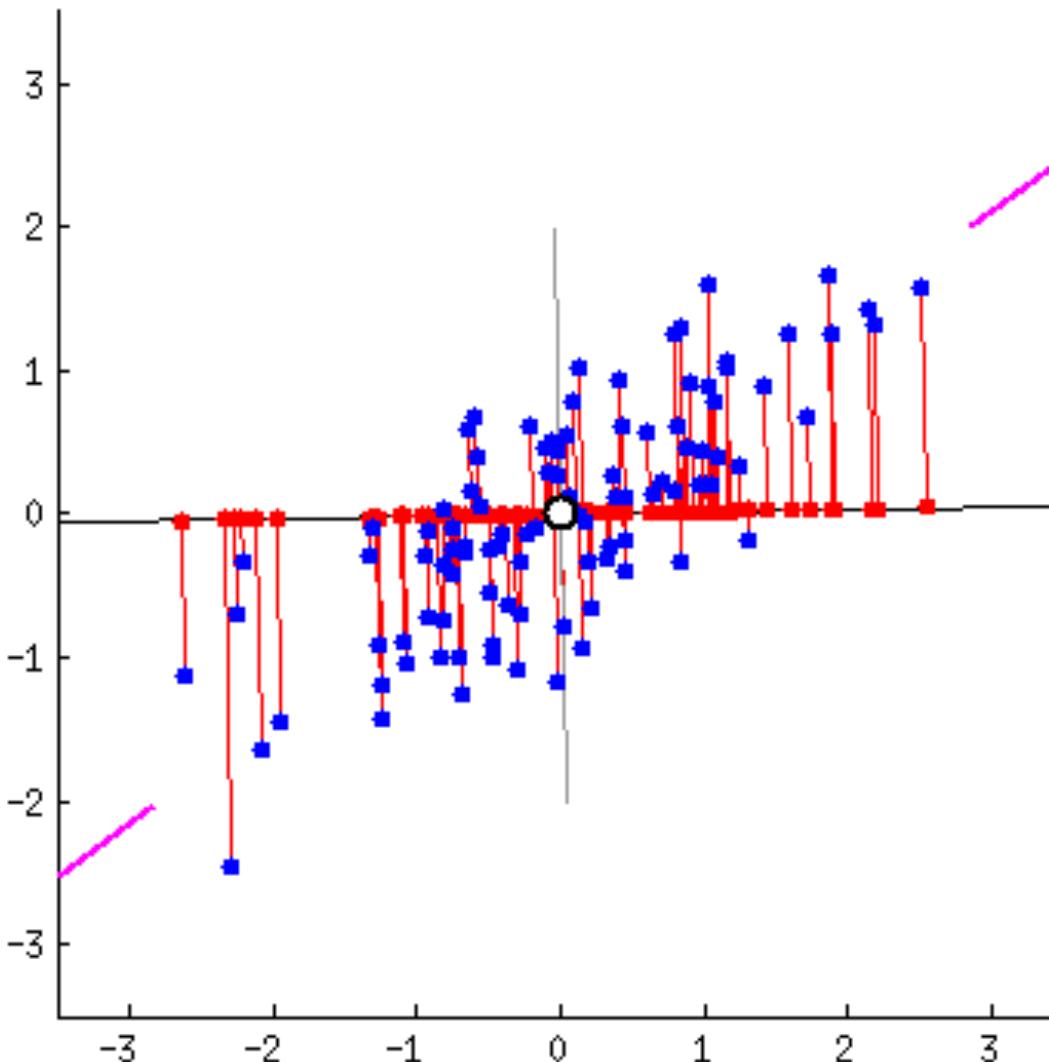
# Principal Component Analysis (PCA)

- Principal Components (PC) are **orthogonal directions** that **capture most of the variance** in the data.



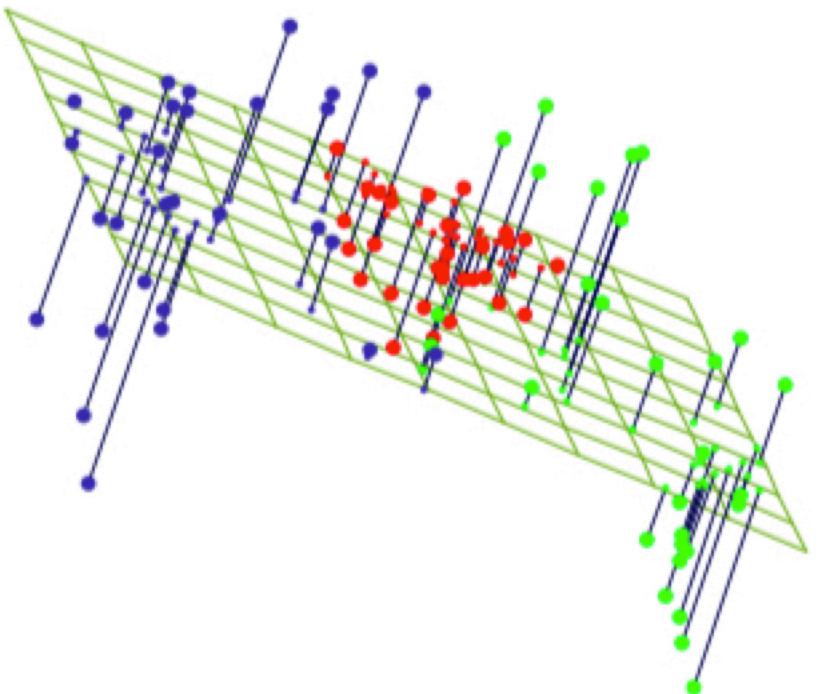
# Principal Component Analysis (PCA)

From <https://stats.stackexchange.com/a/140579>

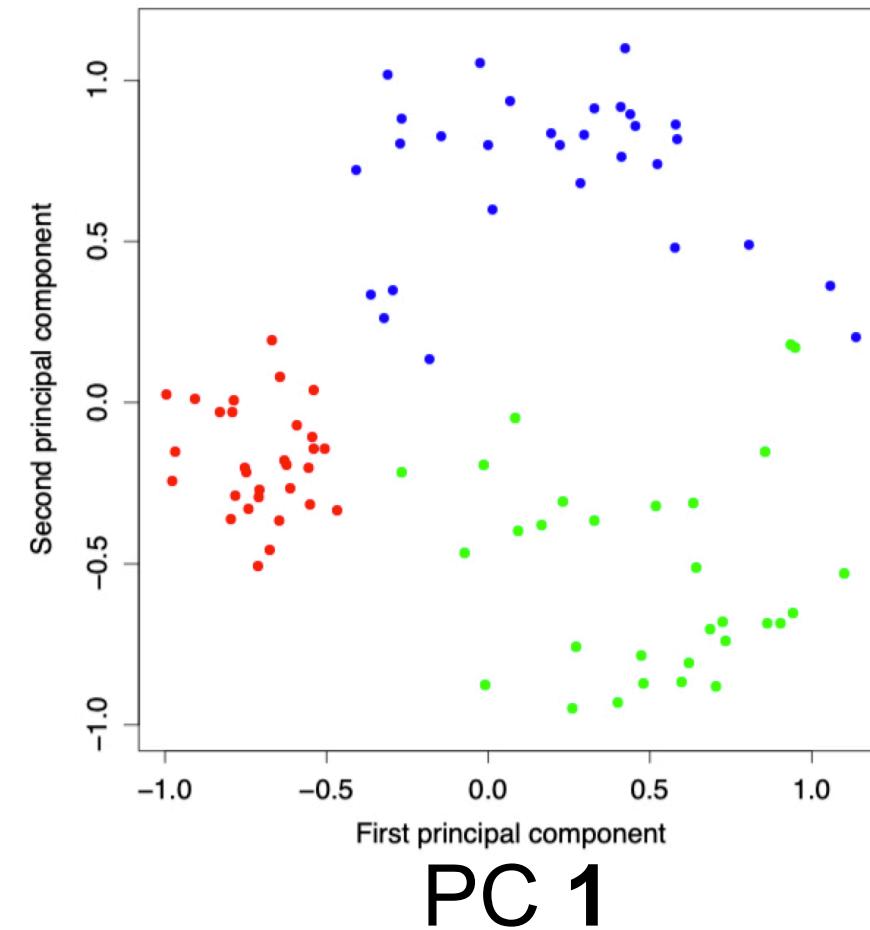


# Principal Component Analysis (PCA)

- Principal Components (PC) are **orthogonal directions** that **capture most of the variance** in the data.



PC 2



PC 1

# Principal Component Analysis (PCA)

Dataset  $S = \{x_i\}_{i=1,\dots,n}$

- $x_i$ : data example with  $d$  attributes
- $\mu$ : mean data point

**PCA approximates  $x$  with  $\hat{x}_k$  by finding projection directions  $v_0, v_1, \dots, v_k$  such that:**

$$\hat{x}_k = \mu + ((x - \mu) \cdot v_0)v_0 + ((x - \mu) \cdot v_1)v_1 + \cdots + ((x - \mu) \cdot v_k)v_k$$

and the reconstruction error  $(x - \hat{x}_k)^2$  is minimized

$v_0, v_1, \dots, v_k$  are **orthogonal unit vectors**.

# PCA for Face Reconstruction

Eigenfaces, slide based on Derek Hoeim's, UIUC CS543

Image dataset,  $\{x_i\}_{i=1,\dots,n}$

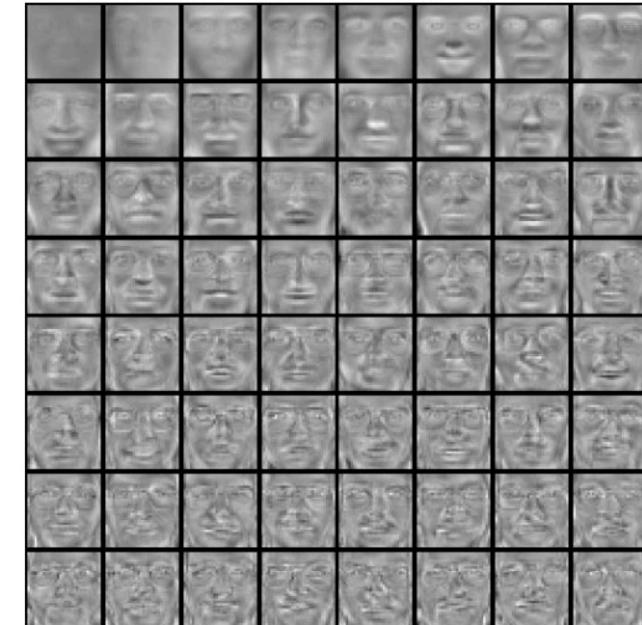


PCA algorithm



64 Principal Components

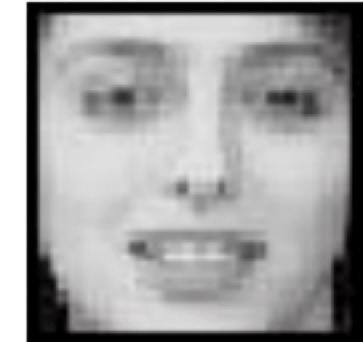
$v_0, v_1, \dots, v_{63}$



# PCA for Face Reconstruction

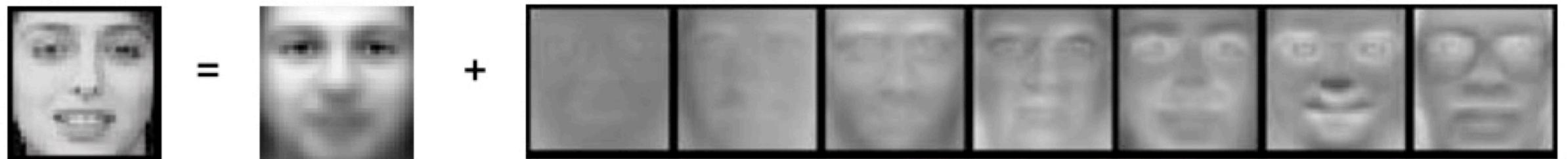
Eigenfaces, slide based on Derek Hoeim's, UIUC CS543

Face Reconstruction using the Principal Components



$$\hat{x}_8 = \mu + ((x - \mu) \cdot v_0)v_0 + \dots + ((x - \mu) \cdot v_7)v_7$$

The equation illustrates the reconstruction of a face image  $x$  using PCA. It shows the face  $x$  being approximated as the mean face  $\mu$  plus a weighted sum of principal components  $v_0, \dots, v_7$ . Blue arrows point from the terms  $((x - \mu) \cdot v_0)v_0$  and  $((x - \mu) \cdot v_7)v_7$  to the corresponding eigenvectors in the matrix below.

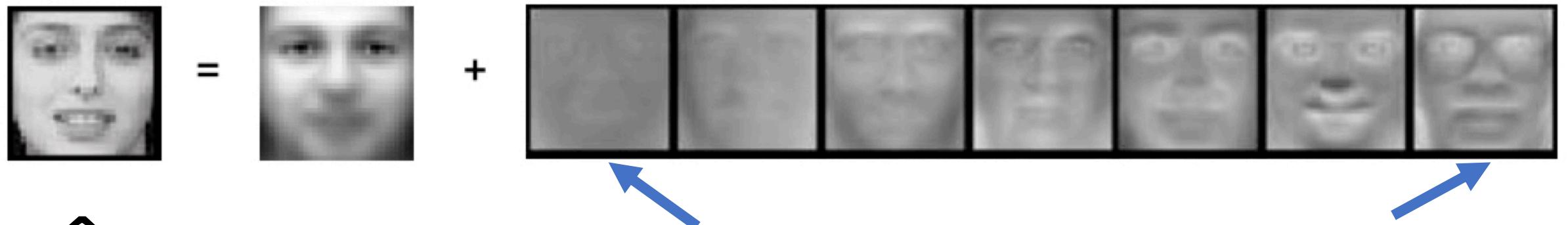
A horizontal row of seven grayscale images representing the principal components  $v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7$ .

# PCA for Face Reconstruction

Eigenfaces, slide based on Derek Hoeim's, UIUC CS543

Face Recognition using PCA:

- 1- Given face image datasets, **extract Principal Components**  $v_0, v_1, \dots, v_k$ .
- 2- Given new image, **project** onto PCs.
- 3- **Find closest** (projected) image in training dataset

$$\hat{x}_8 = \mu + ((x - \mu) \cdot v_0)v_0 + \dots + ((x - \mu) \cdot v_7)v_7$$


[PDF] [Face recognition using eigenfaces](#)

[MA Turk, AP Pentland](#) - ... on Computer Vision and Pattern Recognition, 1991 - [cin.ufpe.br](#)

We present an approach to the detection and identification of human faces and describe a working, near-real-time face recognition system which tracks a subject's head and then recognizes the person by comparing characteristics of the face to those of known ...

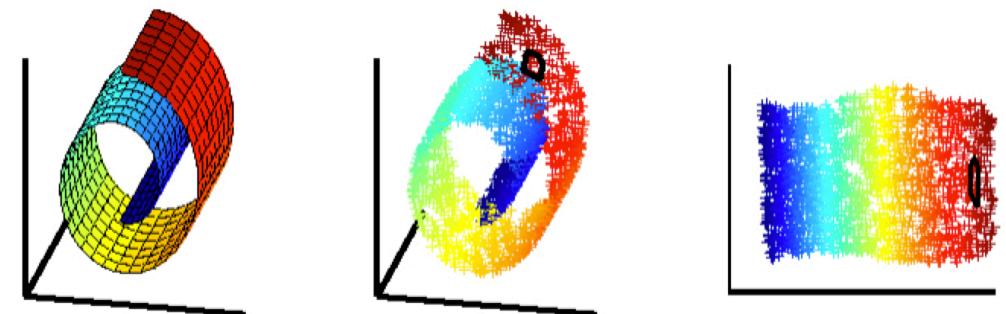
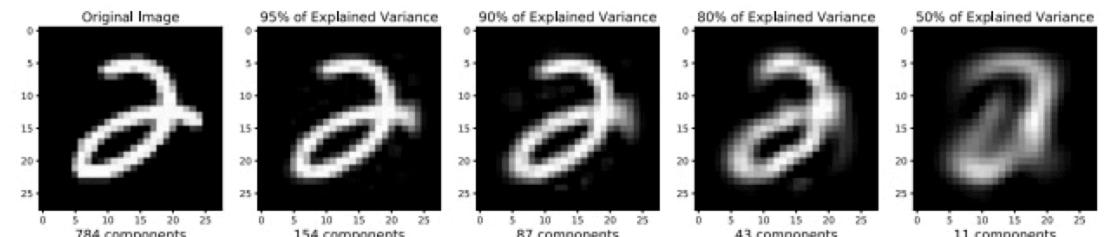
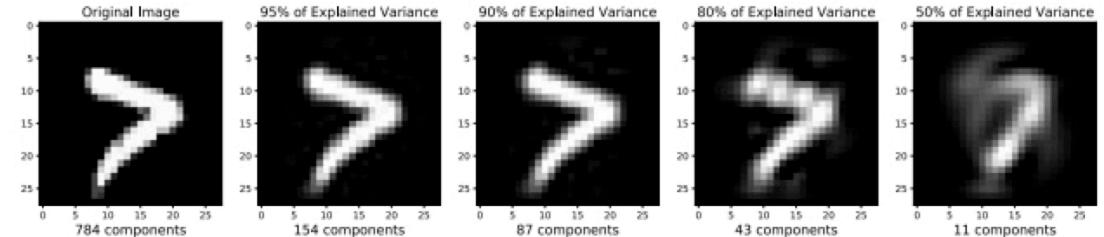
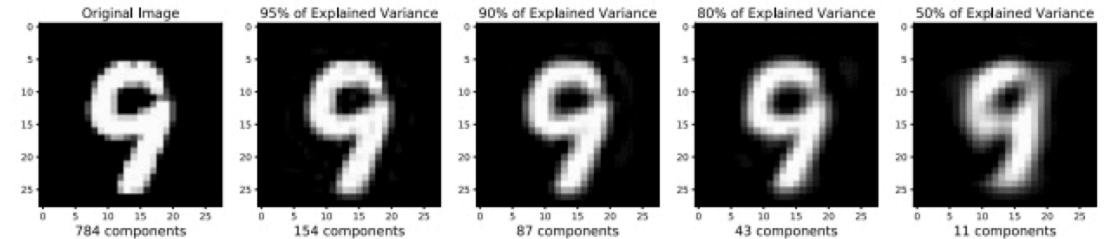
# Final words on PCA

## Advantages

- **Fast** to compute an optimal solution: an eigenvector problem
- **No hyper-parameters** to tune

## Caveats

- Limited to **second-order** dependencies (variance is the limit)
- Limited to **linear** projections



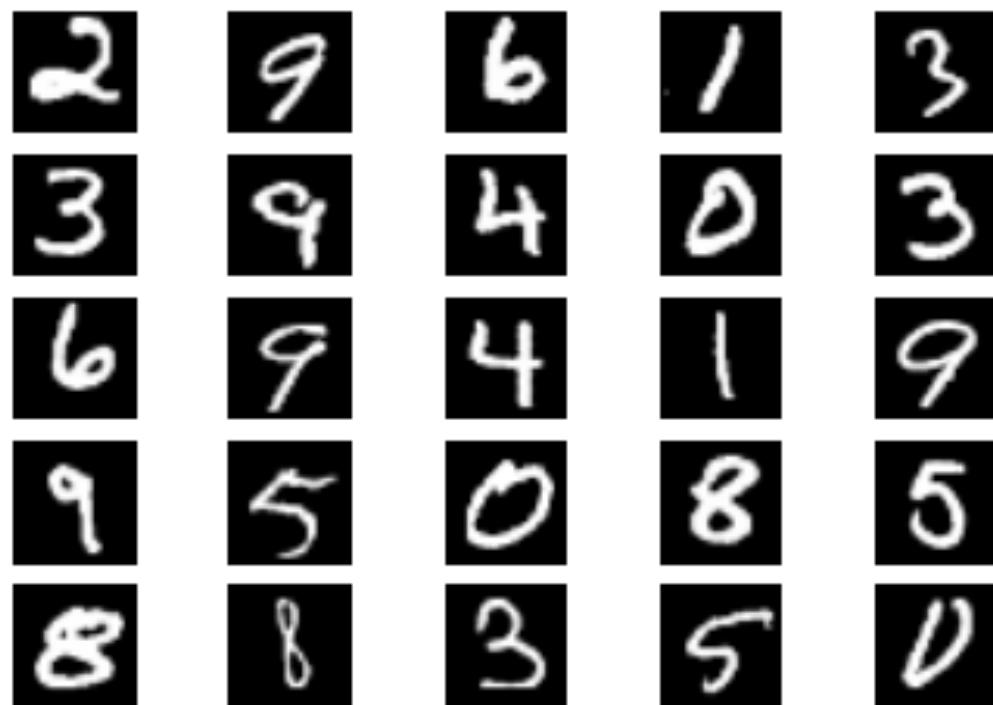
From Michael Guerzhoy's slides, UofT CSC320

# t-SNE

**t-Distributed Stochastic Neighbor Embedding**

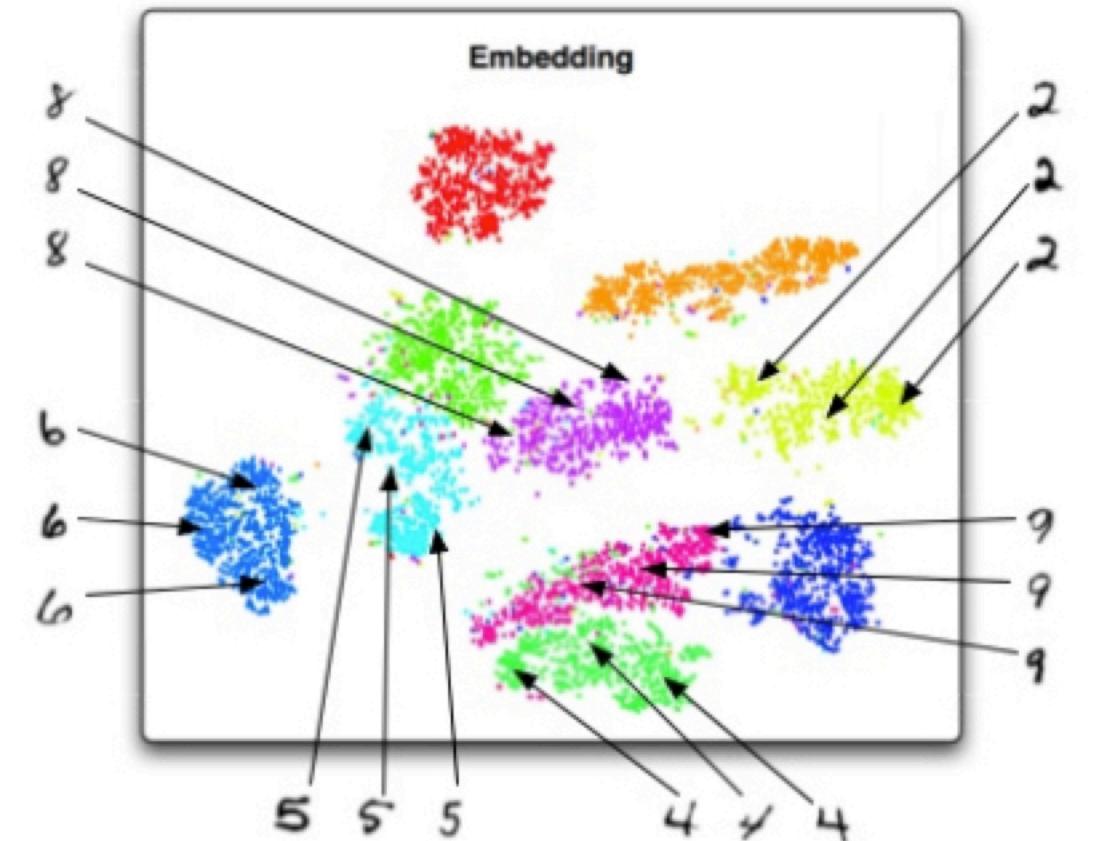
**784 dimensions = 28x28 pixels**

Random Sampling of MNIST



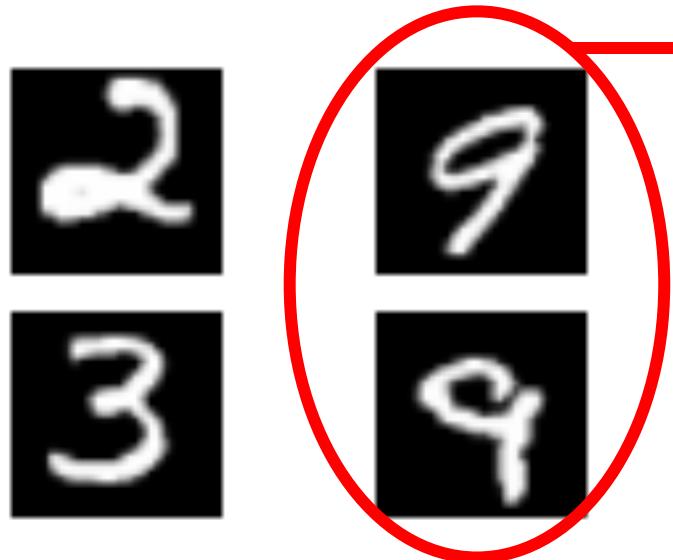
t-SNE

**2 dimensions**



# t-SNE

## t-Distributed Stochastic Neighbor Embedding



t-SNE finds a  $y_i$  for each  $x_i$   
such that the two  
**probability distributions**  
**are similar**

high probability

$$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2 / 2\sigma_i^2)}$$

Probability that point  $x_i$  “selects”  $x_j$  as “neighbor”.  
Based on transforming Euclidean distance to a probability via  
a Gaussian distribution.

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l} (1 + \|y_k - y_l\|^2)^{-1}}$$

$y_i$  is the low-dimensional mapping of  $x_i$   
 $q_{ij}$  is the probability that  $y_i$  “selects”  $y_j$  as “neighbor”.

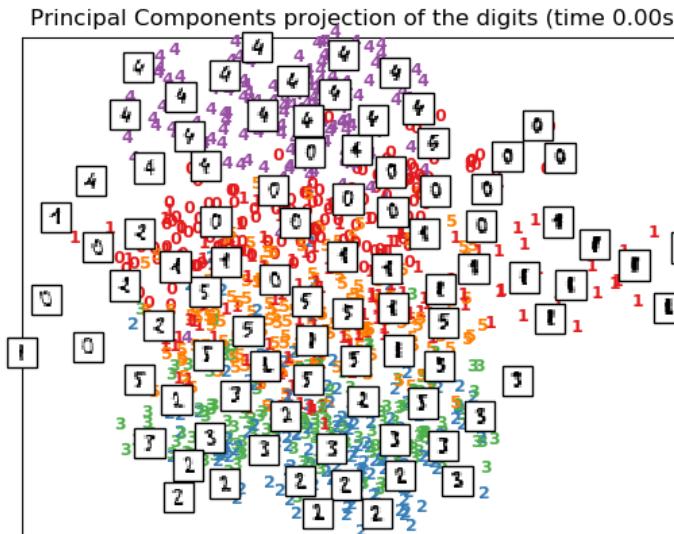
# t-SNE

# t-Distributed Stochastic Neighbor Embedding

A selection from the 64-dimensional digits dataset

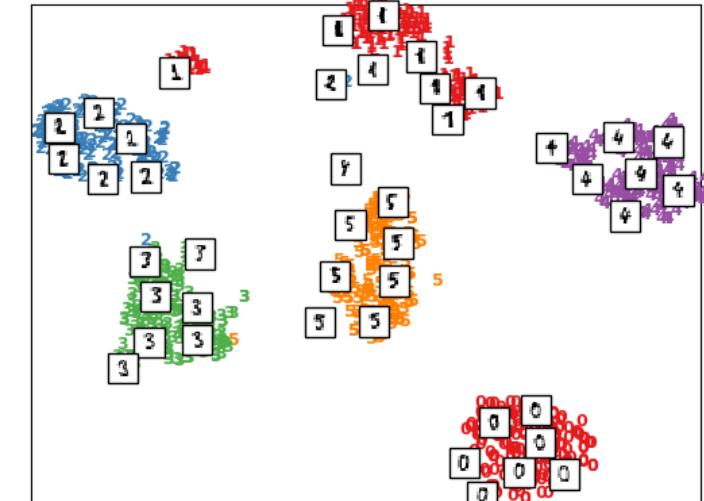
0	1	2	3	4	5	0	1	2	3	4	5	0	5
5	5	0	4	1	3	5	1	0	0	2	2	0	1
4	4	1	5	0	5	2	4	0	0	1	3	1	1
3	4	4	0	5	3	1	5	4	4	2	2	5	5
2	3	4	5	0	1	2	3	4	5	0	5	5	5
0	4	4	3	5	1	0	0	2	2	1	0	4	4
1	5	0	5	2	2	0	0	1	3	2	1	3	4
0	5	3	4	4	1	1	2	1	3	4	4	3	1
5	0	4	2	3	4	4	1	2	3	3	3	4	4
5	4	0	0	2	2	2	0	4	2	3	3	4	4
5	2	2	0	0	4	3	2	4	4	3	4	3	0
3	5	5	4	2	2	2	5	5	4	0	3	0	1
0	1	2	3	4	5	0	1	2	3	4	5	0	1
5	1	0	0	1	2	2	0	1	2	3	3	3	4
1	2	0	0	1	3	2	1	4	3	1	4	0	5
1	5	4	4	2	2	2	1	2	5	4	4	0	0
2	3	4	5	0	1	2	3	4	5	0	5	5	0
0	0	2	1	2	0	1	3	3	3	4	4	4	1
0	0	1	3	2	1	4	3	1	3	4	5	0	5
4	4	2	2	1	5	5	4	4	0	0	1	2	3

## PCA



## t-SNE

t-SNE embedding of the digits (time 4.60s)



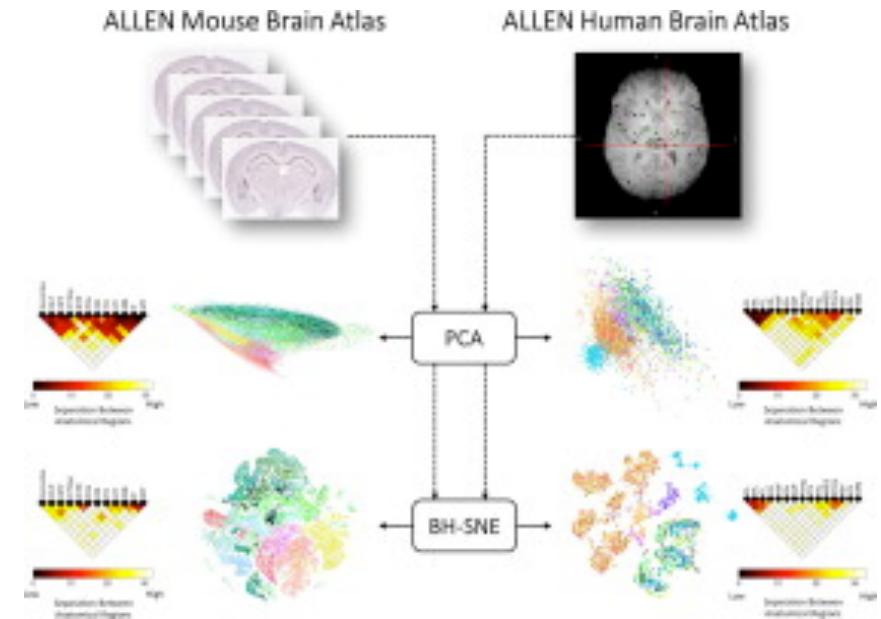
# Final words on t-SNE

## Advantages

- Conserves **local and global patterns** in the data
- Handles highly **non-linear** data

## Caveats

- **Slow** to compute, but **PCA** preprocessing speeds it up!
- Requires careful hyper-parameter **tuning**
- **Cannot project a new point**



Visualizing the spatial gene expression organization in the brain through non-linear similarity embeddings. Mahfouz et al. (2015)

# Interactive dimensionality reduction

- <https://projector.tensorflow.org/>
  - You can use the “LOAD” button to upload your own data and interactively visualize the results of PCA or t-SNE, **in the browser**
- <https://distill.pub/2016/misread-tsne/>
  - t-SNE has some tricky hyper-parameters that must be tuned to the dataset you care about. This interactive study looks at how the hyper-parameters behave and gives guidelines for tuning them to get the best outcomes