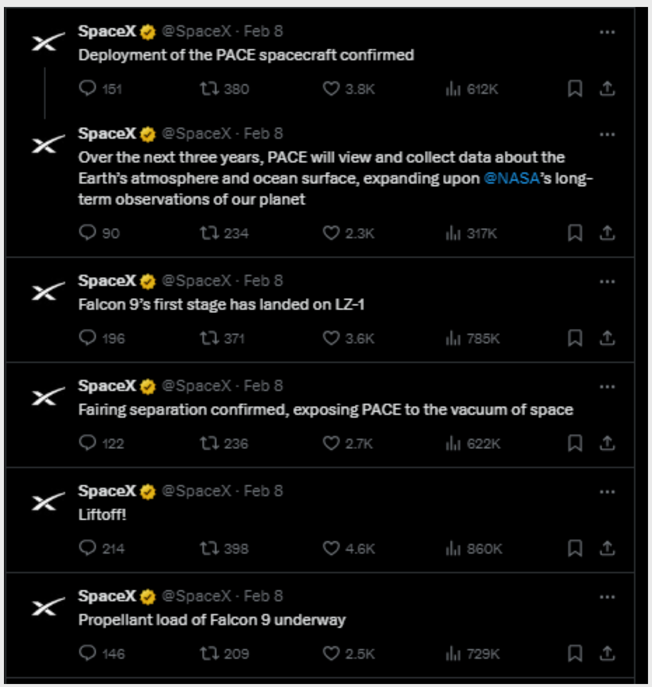
Clustering

Compressing Data

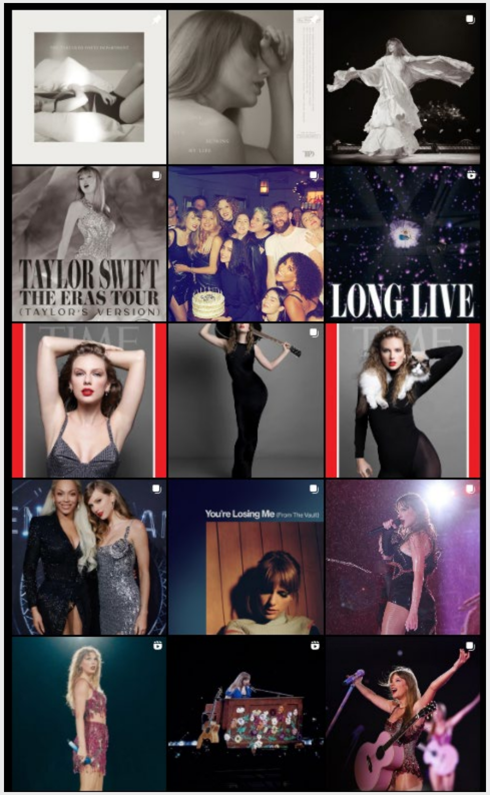
- Can we describe these tweets with fewer bits of information?



- Space travel tweets
- DOGE tweets
- Edgelord tweets

Compressing Data

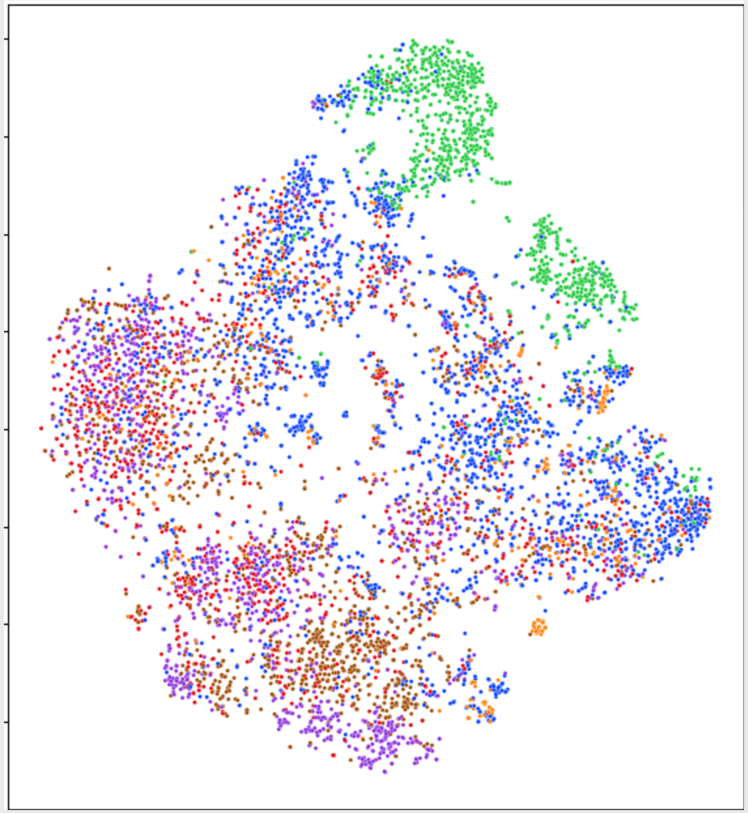
- **Can we describe these images with fewer bits of information?**



- **Album cover images**
- **Concert images**
- **Cat images**
- **Friends images**

Clusters and Compression

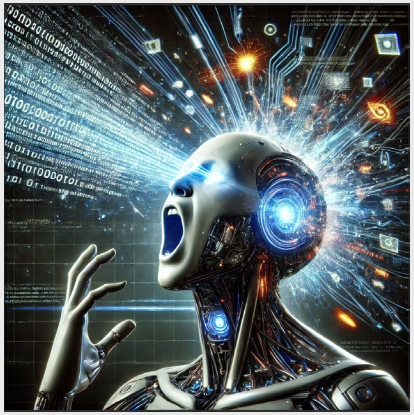
- Clustering is a form of data compression
- Many times data exists in distinct clusters
- If we can find these clusters, we can summarize the data in terms of the clusters



Clusters and Generative AI

- We give the AI data for analysis

- For moderate amounts of data, we can put it all in the prompt
  - Moderate = 128,000 tokens of text
  - Moderate = 250 images

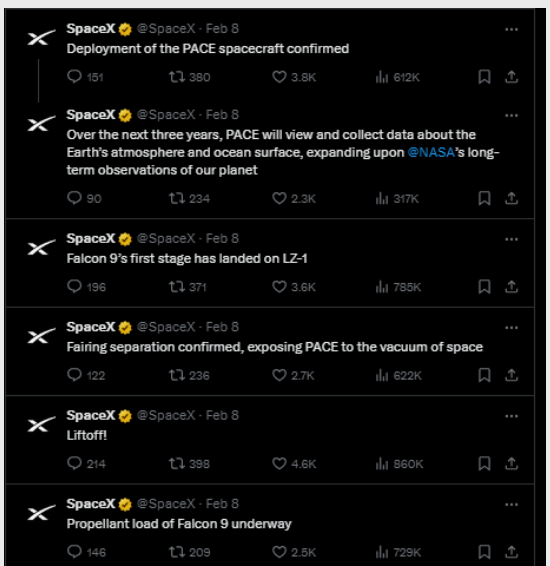


- For large amounts of data, the AI can’t handle it
  - We need to compress the data for the AI
  - Clustering allows us to compress the data

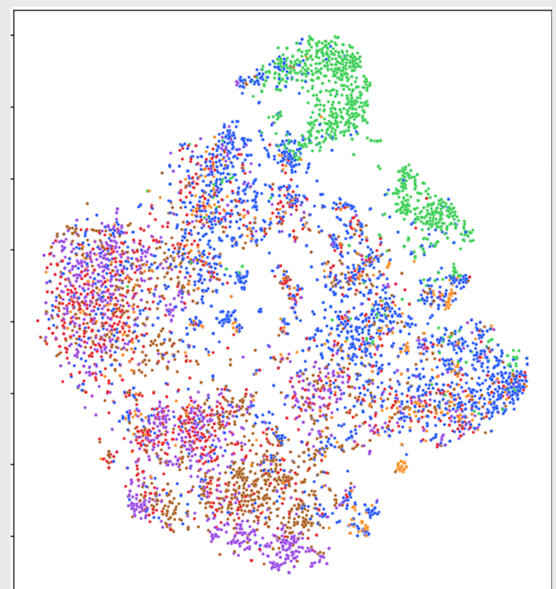
Clusters and Embeddings

- Good embeddings of data map similarity into geometry
- Finding clusters then reduces to finding geometrically clustered data points

Data



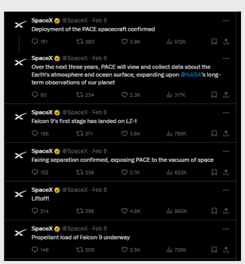
Embeddings



Clustering Pipeline

- To cluster data we will use the following steps
  - Embed with a neural network in high dimensions (transformer or ResNet)
  - Embed in low dimensions (PCA)
  - Cluster data (K-means)

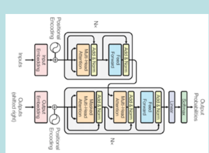
Data

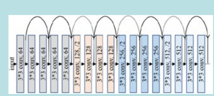


Embed

Text - transformers

Images - ResNet



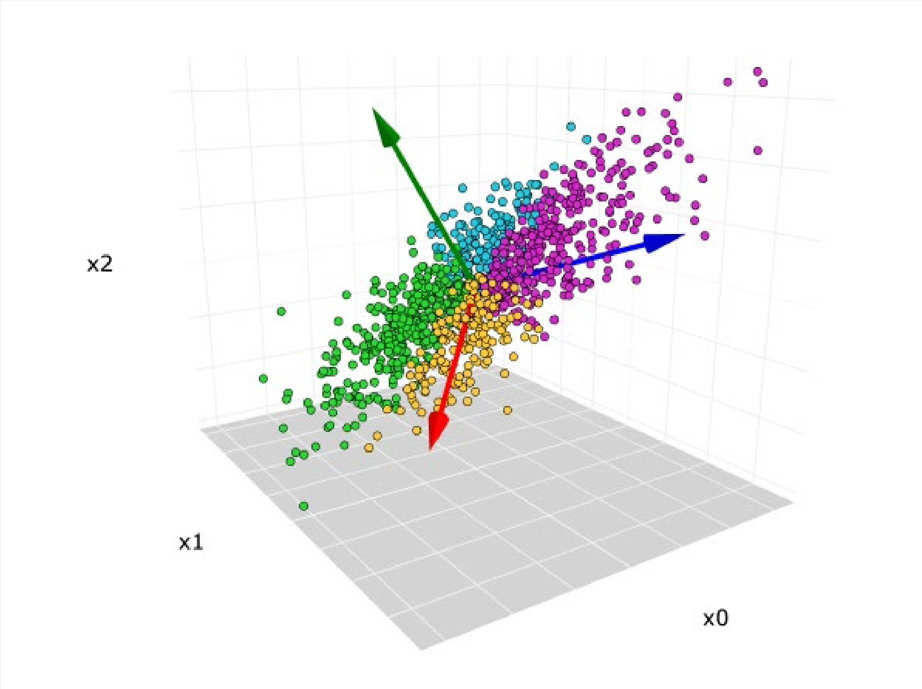


Embed in low dimensions

PCA

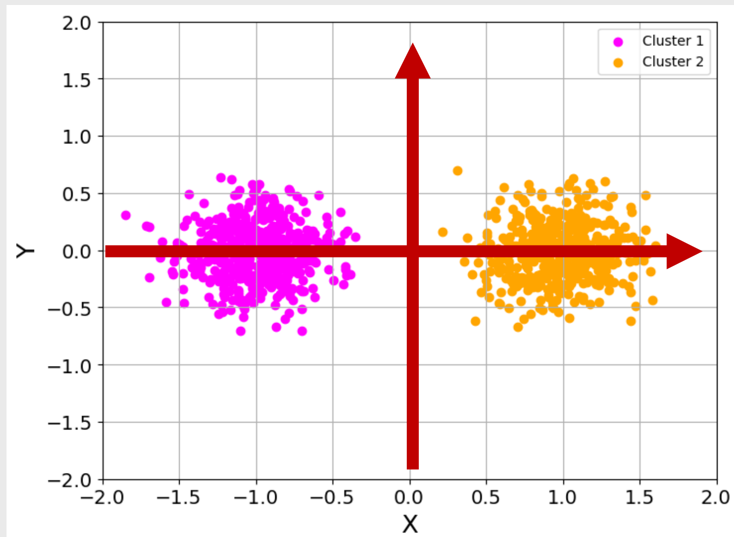
Cluster

K-means

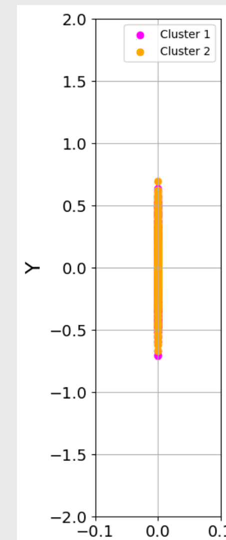


PRINCIPLE COMPONENT ANALYSIS (PCA)

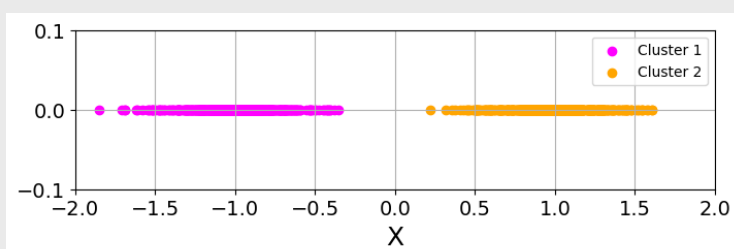
Embedding Data in 1 Dimension



Project onto y

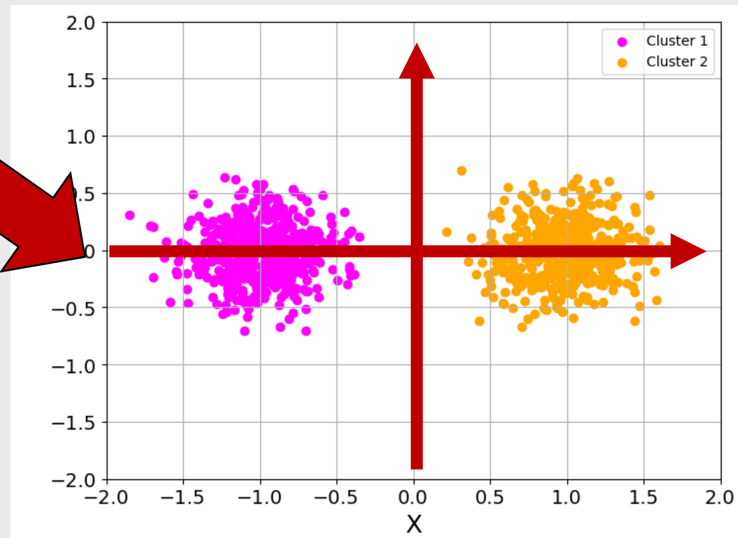


Project onto x

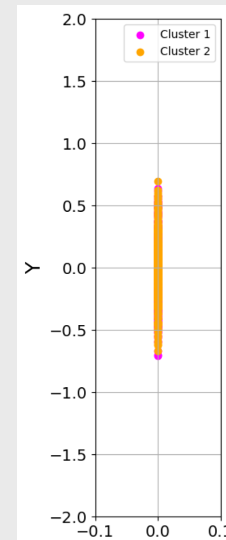


Embedding Data in 1 Dimension

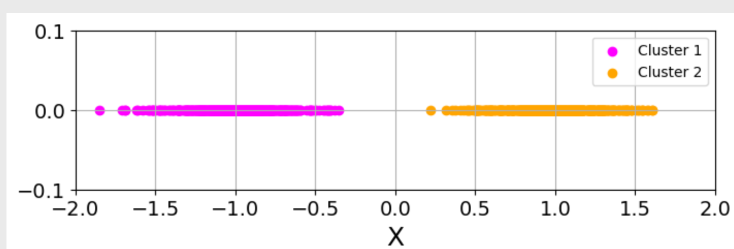
Direction of maximum variance



Project onto y



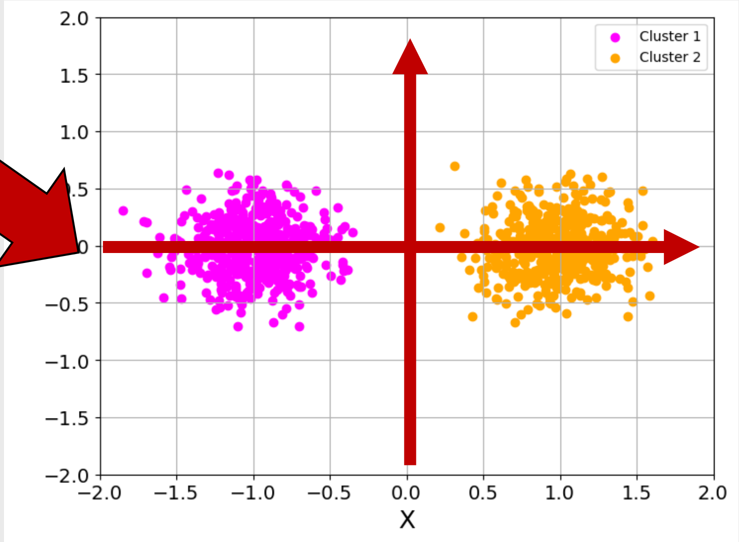
Project onto x



Direction of Maximum Variance

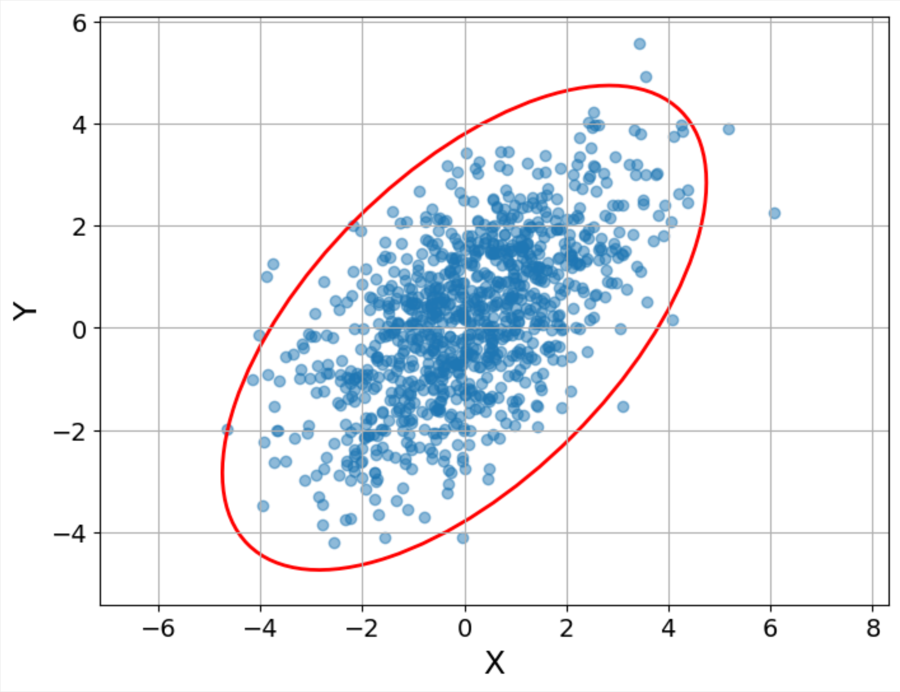
- If we project the data onto the direction of maximum variance, we can separate clusters
- How do we find this direction?

Direction of maximum variance



Covariance Matrices

- We can describe data with dimension d with a d x d covariance matrix
- This matrix encodes the direction of maximum variance



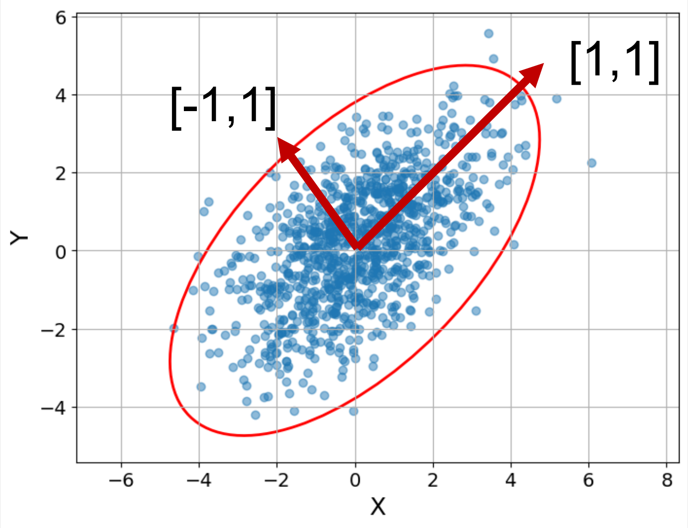
d = 2

$$\Sigma = \begin{bmatrix} E[XX] & E[XY] \\ E[YX] & E[YY] \end{bmatrix}$$

Principal Components

- The ellipse around the data is encoded in the covariance matrix
- The axes of the ellipse are the principal components
- The length of the axes are the standard deviations

$$\Sigma = \begin{bmatrix} 1.0 & 0.55 \\ 0.55 & 1.0 \end{bmatrix}$$



$$\Sigma = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 3.76 & 0 \\ 0 & 1.08 \end{bmatrix} \left(\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \right)^{-1}$$

Principal Component Analysis (PCA)

- PCA lets us compute the projection of data onto the principal components (PCs) of its covariance matrix very quickly

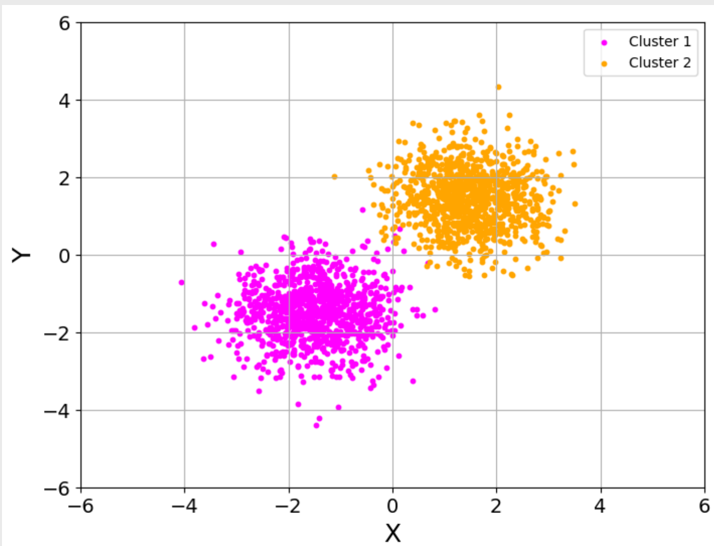
- PCA was invented in 1901

- Has many names depending on the field you are in
  - Karhunen–Loève transform
  - Hotelling transform
  - eigenvalue decomposition
  - singular value decomposition
  - proper orthogonal decomposition
  - factor analysis
  - spectral decomposition
  - empirical orthogonal functions

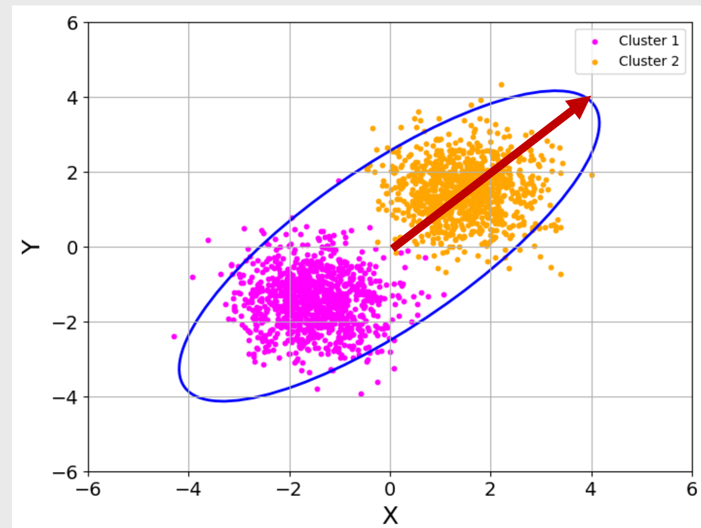
PCA Algorithm

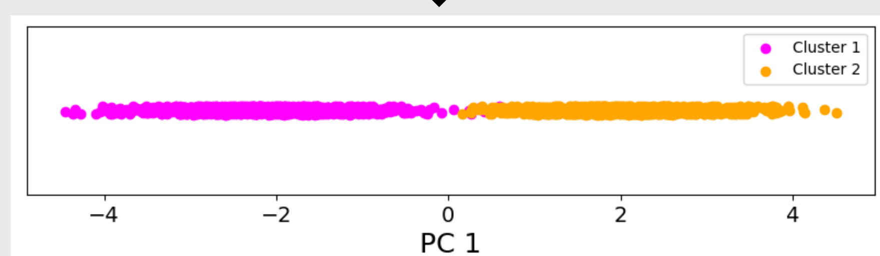
1. Center data at zero (subtract the means)
2. Make data variance 1 in each dimension
3. Compute covariance matrix
4. Compute the principal components (PCs) with the largest variance (*fit*)
5. Project the data onto each PC to obtain the PCA embedding (*transform*)

PCA Example



- Compute covariance matrix and PC 1



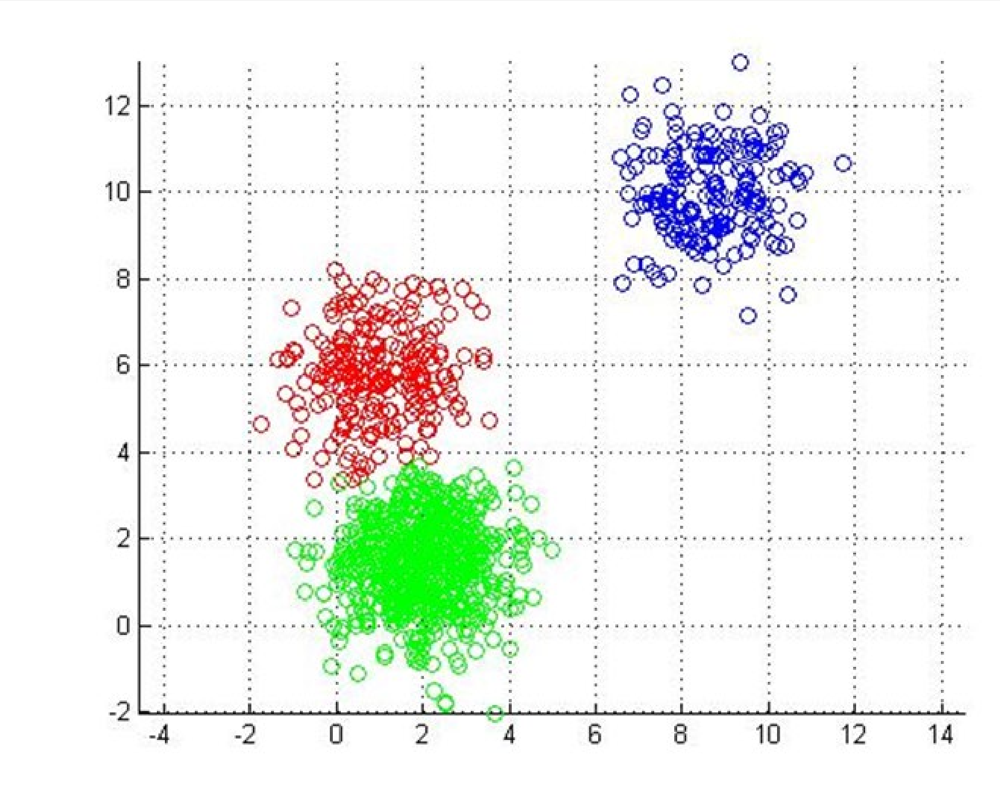


- Project data to PC 1

Pros and Cons of PCA

- Pros
  – Extremely fast algorithm (good for big datasets)
  – Easy to interpret (each PC is a direction of high variance, probably due to some data property)

- Cons
  – Does not give the best embedding for some high-dimensional datasets



K-MEANS CLUSTERING

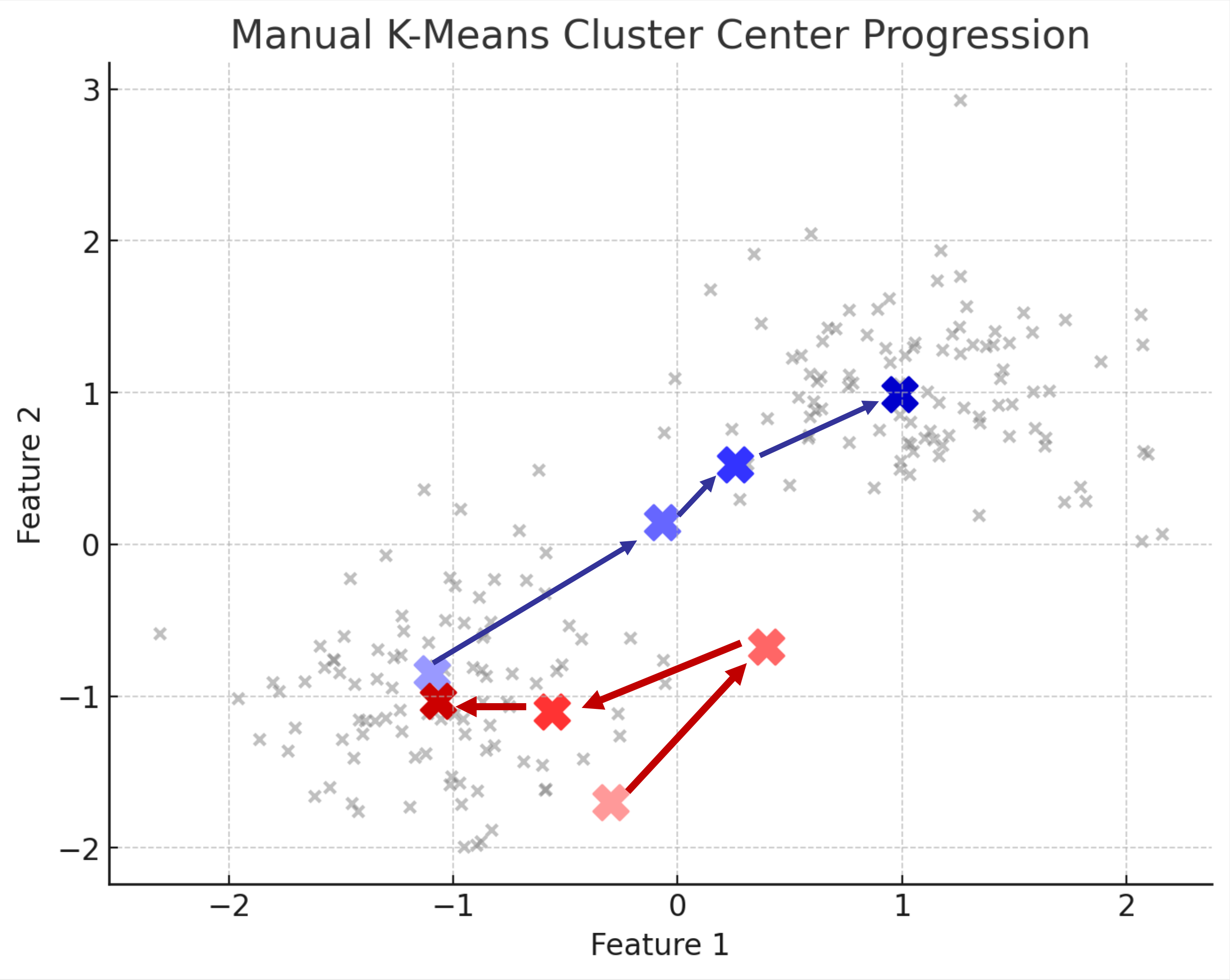
K-Means Clustering

- K-means clustering is a simple algorithm for clustering any kind of data (once it is embedded in a vector space)
- Invented in 1957
- Still one of the most popular ways to cluster data

K-Means Clustering Algorithm

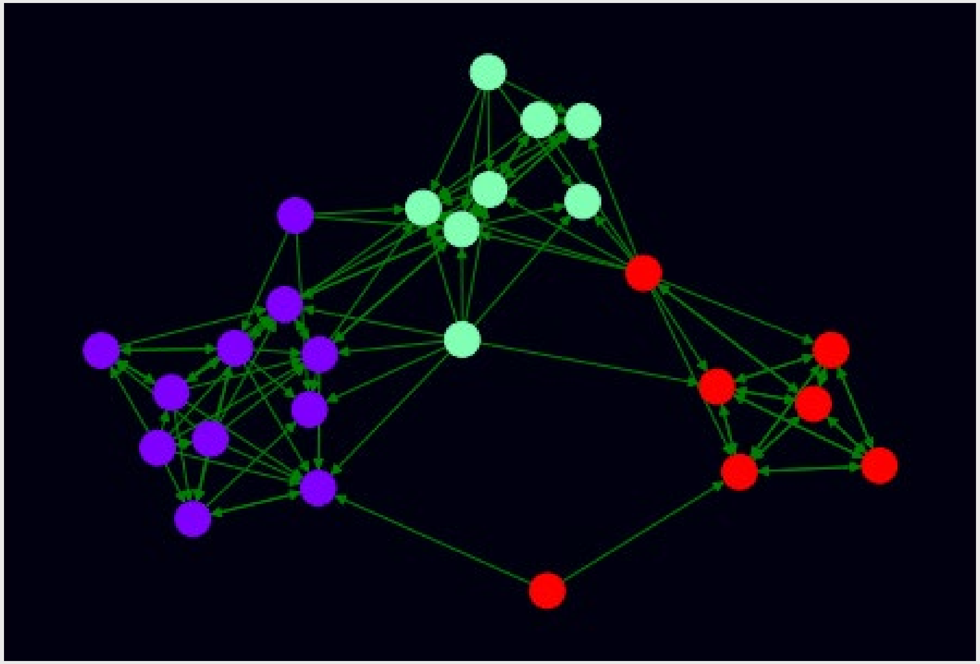
1. Choose the number of clusters k
2. Initialize the cluster centers randomly
3. Repeat this iteration
   1. Assign each data point to the cluster whose center it is closest to
   2. Set the center of a cluster equal to the center of mass of the data points assigned to it
4. Stop when the cluster centers stop changing

K-Means Example



Community Detection

- An important problem in social networks is finding communities – clusters of people in a network
- Data = network structure
- Clustering algorithm = spectral clustering (K-means on spectral embeddings)



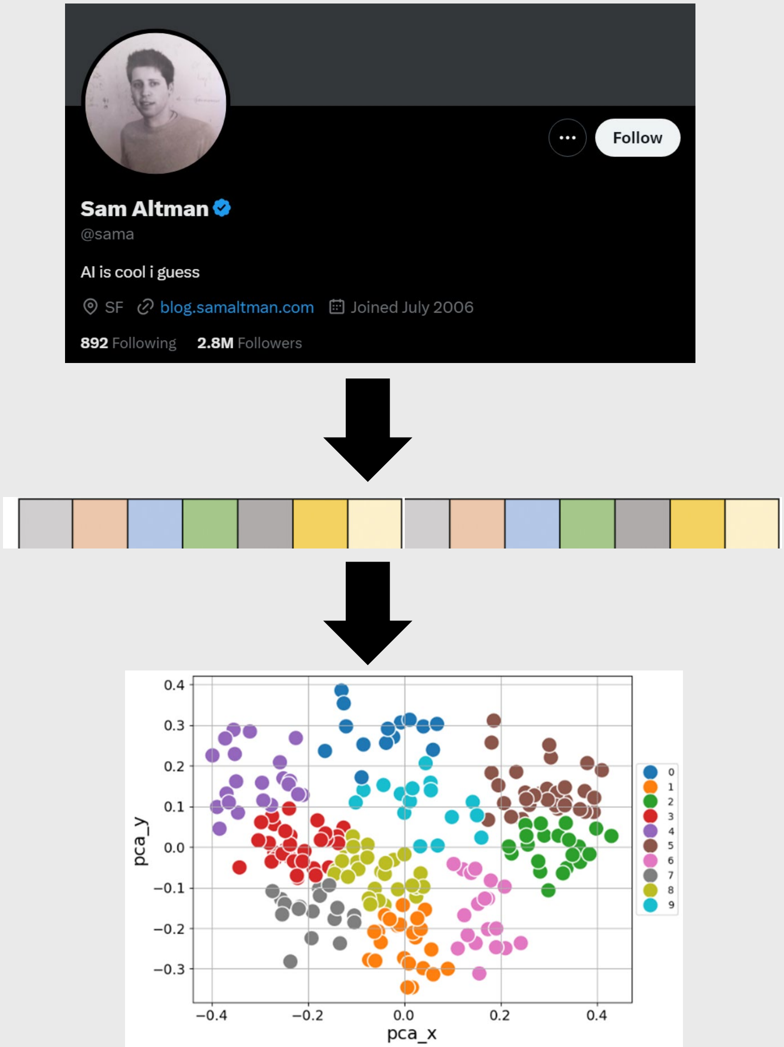
Challenges with Community Detection

- Today network data is hard to collect ☹
  - We can use Chrome plugins like TwFollow

- It's ok, we can use AI to find communities based on user (non-network) data ☺

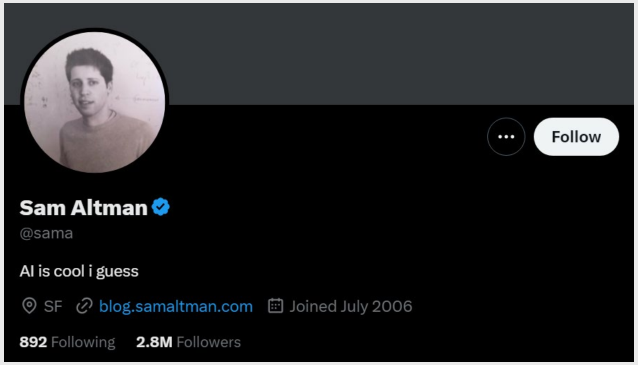
Community Detection With Embeddings and K-Means

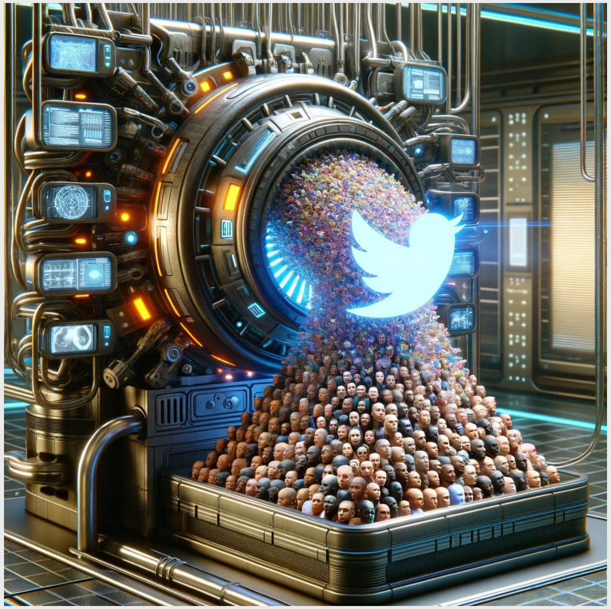
- User profile has a username, name, location, and bio
- Embed this data with a transformer, then use K-means to find the communities



Community Detection With AI

- User profile has a username, name, location, and bio
- We can feed all this raw data to the AI and ask it to give us the communities



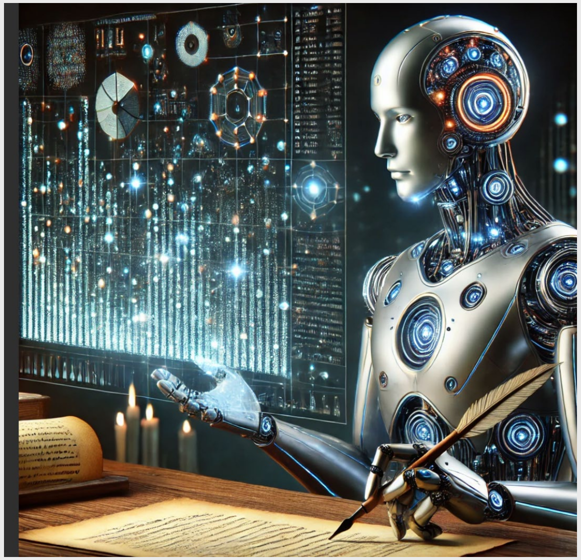


AI Enhanced Clustering

- In the old days, the end goal of clustering was to assign each data point to a cluster

- Today with generative AI, we can go further and understand the clusters
  - Title
  - Description of underlying theme
  - Representative examples

- We can feed these cluster summaries to the AI for use in other tasks



Coding Session

- Cluster tweets
  - OpenAI transformer embeddings and K-means clustering
  - ChatGPT to describe clusters
- Cluster images
  - ResNet embeddings and and K-means clustering
  - ChatGPT to describe clusters
- Find communities in a social network using no network data
  - OpenAI transformer embeddings and K-means clustering
  - ChatGPT
  - ChatGPT to describe communities