

# Feature selection

# Feature selection

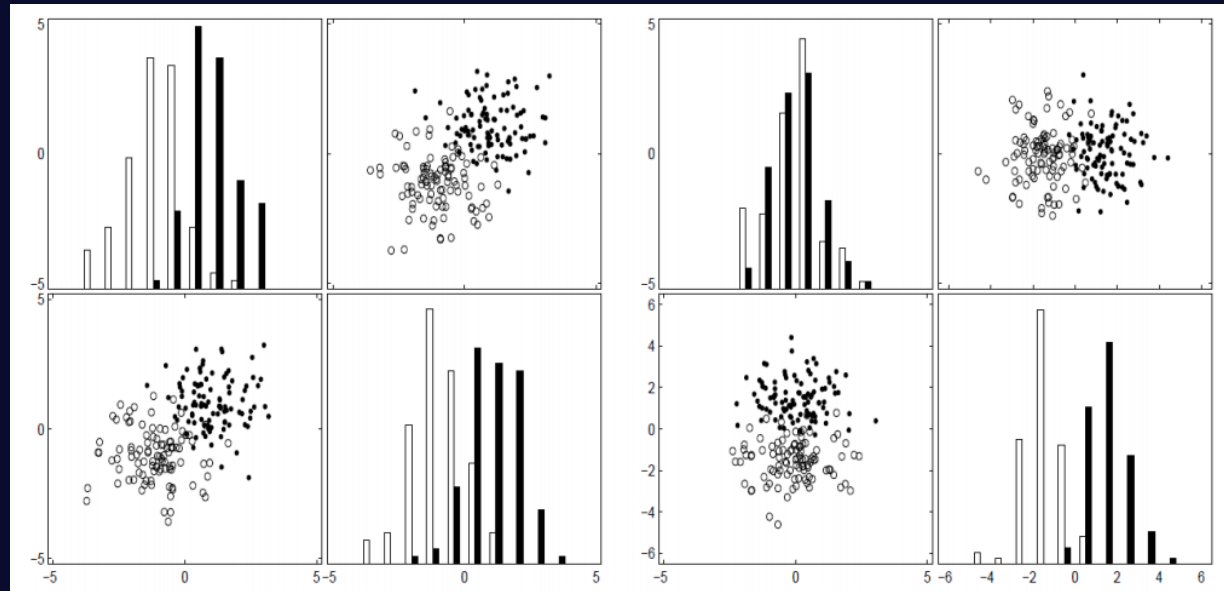
Goal: Find an informative feature subset

Methods:

- **Filter methods:** ranks features or feature subsets independently of the classifier (e.g. correlation between features, correlation between features and the label,...).
- **Wrapper methods:** uses a classifier to assess features or feature subsets
- **Embedded methods:** Specific to a given learning machine; Performs variable selection (implicitly) in the course of training (e.g. decision tree, Naïve Bayes,...)

# Filter Methods

Apparently redundant-looking variables could be non-redundant

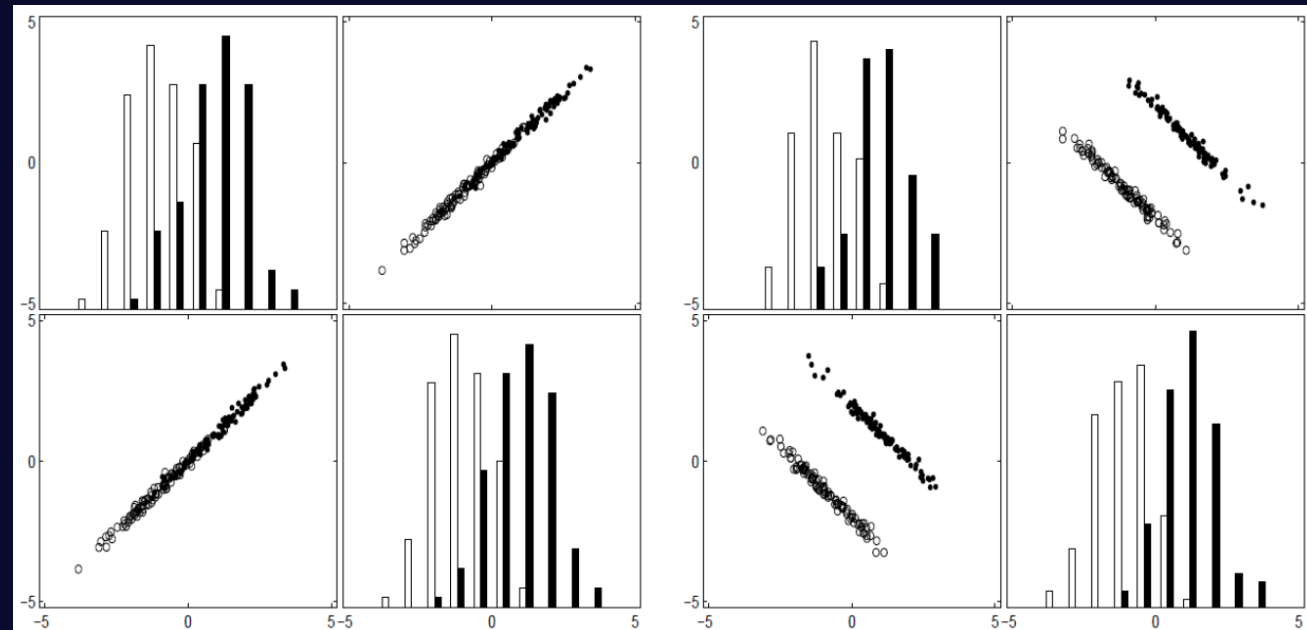


The two variables have the same distribution -> redundant?

45-degree rotation: average of the variables -> better class separation than either alone

# Filter Methods

Effect of within-class correlation on redundancy?

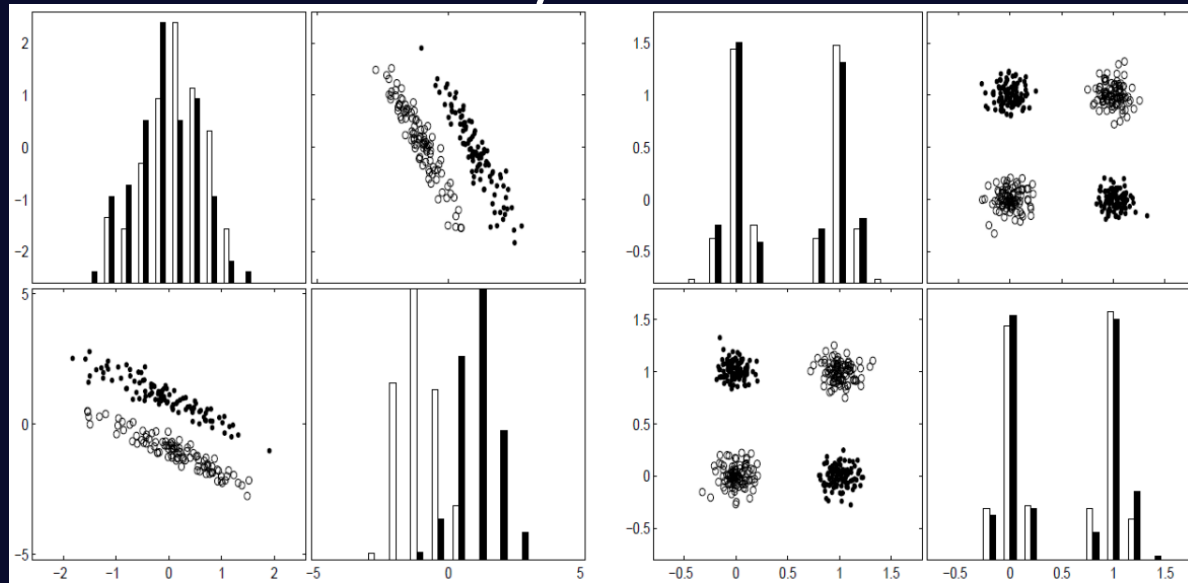


Variables highly correlated within class, correlation coincides with direction of class separation – redundant.

High within-class correlation not along class separation direction – not redundant.

# Filter Methods

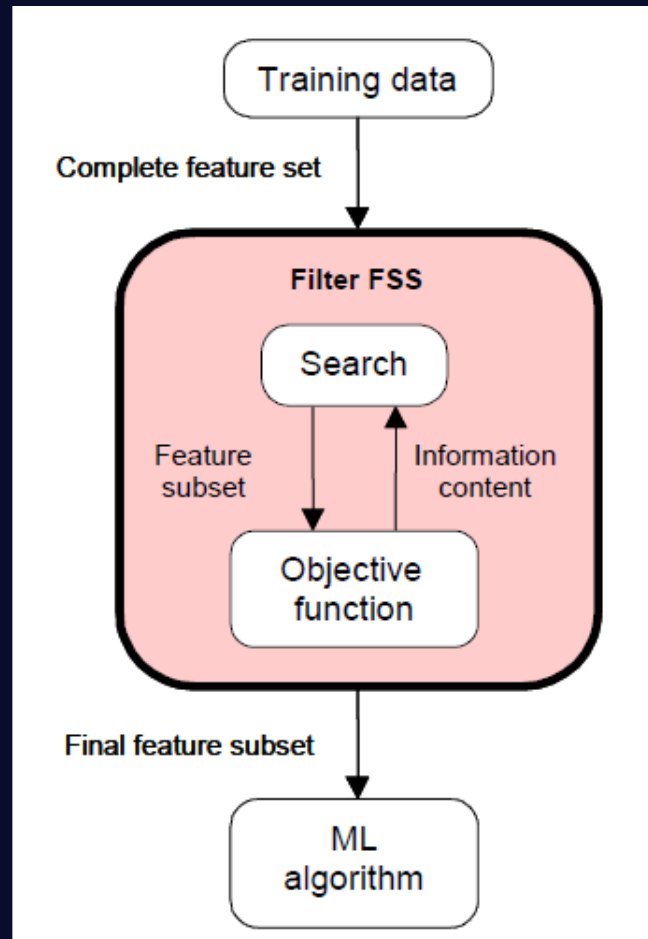
Using variable ranking to filter poor variables (to avoid overfitting to too many variables), also useful variables can be lost: a variable useless by itself can be useful with others



Top-left variable alone is unrelated to class. But together gives good separation

XOR problem: each variable by itself is useless, but together they give good separation

# Wrapper methods – general scheme



# Sequential Forward Selection

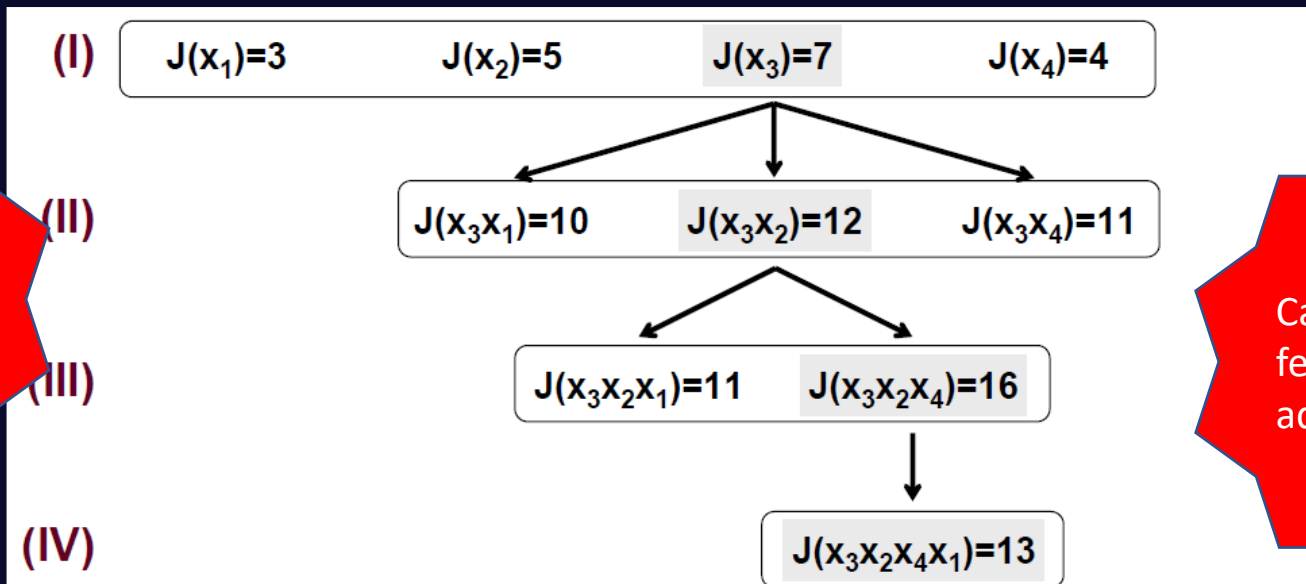
1. Start with the empty set  $Y_0 = \{\emptyset\}$
2. Select the next best feature  $x^+ = \operatorname{argmax}_{x \in Y_k} [J(Y_k + x)]$
3. Update  $Y_{k+1} = Y_k + x^+$ ;  $k = k + 1$
4. Go to 2

# Filter methods

- Let  $J(X)$  be our objective function

$$J(X) = -2x_1x_2 + 3x_1 + 5x_2 - 2x_1x_2x_3 + 7x_3 + 4x_4 - 2x_1x_2x_3x_4$$

Where  $x_i$  is an indicator function



Selected features are biased towards the classifier

Can't remove a feature already added



# Sequential Backward Selection

1. Start with the full set  $Y_0 = X$
2. Remove the worst feature  $x^- = \operatorname{argmax}_{x \in Y_k} [J(Y_k - x)]$
3. Update  $Y_{k+1} = Y_k - x^-$ ;  $k = k + 1$
4. Go to 2

# Bidirectional Search

1. Start SFS with the empty set  $Y_F = \{\emptyset\}$
2. Start SBS with the full set  $Y_B = X$
3. Select the best feature

$$x^+ = \underset{\substack{x \notin Y_{F_k} \\ x \in Y_{B_k}}}{\operatorname{argmax}} [J(Y_{F_k} + x)]$$

$$Y_{F_{k+1}} = Y_{F_k} + x^+$$

3. Remove the worst feature

$$x^- = \underset{\substack{x \in Y_{B_k} \\ x \notin Y_{F_{k+1}}}}{\operatorname{argmax}} [J(Y_{B_k} - x)]$$

$$Y_{B_{k+1}} = Y_{B_k} - x^-; \quad k = k + 1$$

4. Go to 2

# Dimensionality Reduction

# Why dimensionality reduction?

- Discover hidden correlations/topics
  - Two features that occur commonly together
- Remove redundant and noisy features
  - Not all features are useful
- Interpretation and visualization
- Easier storage and processing of the data

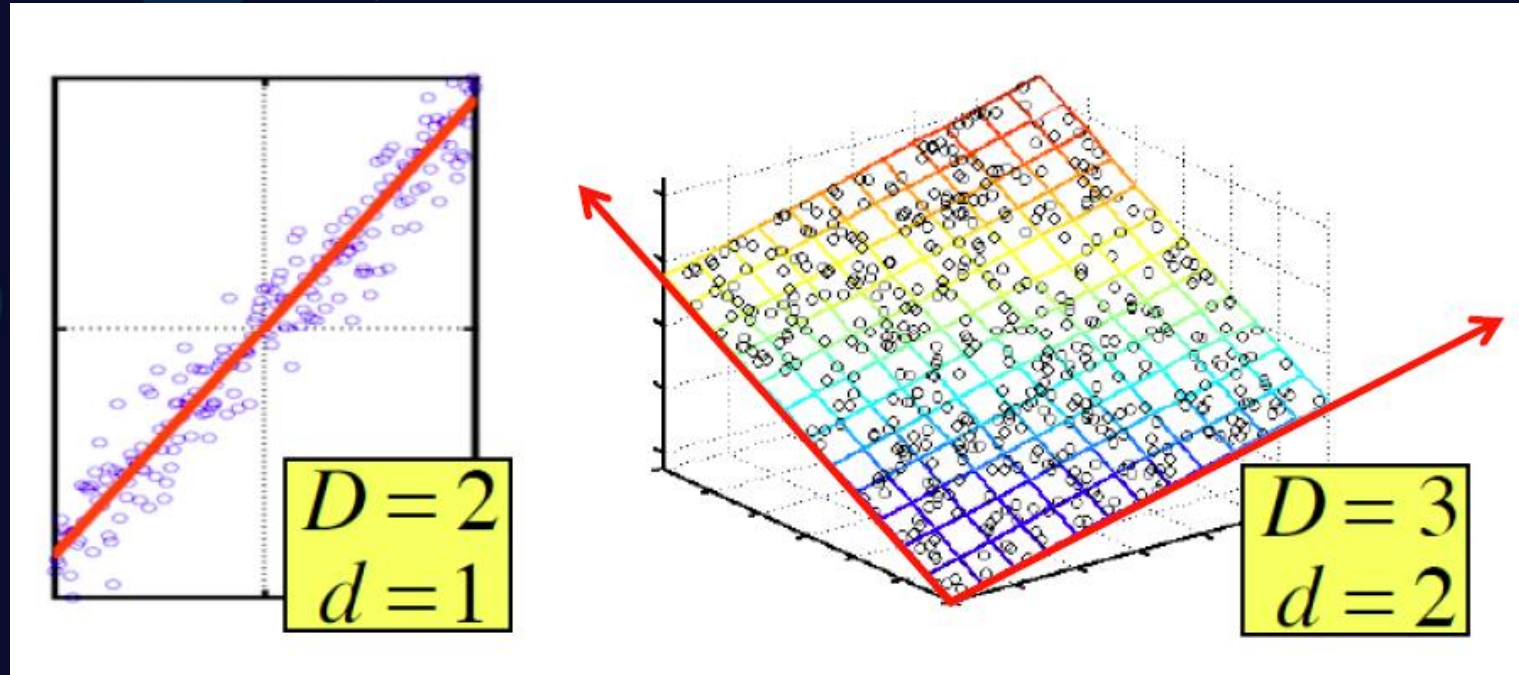
# Dimensionality Reduction

- Differs from feature selection in two ways:
  - Instead of choosing subset of features,
  - Create new features (dimensions) defined as functions over all features
  - Don't consider class labels, just the data points

# Dimensionality Reduction – main idea

- Given data points in  $d$ -dimensional space,
- Project into lower dimensional space while preserving as much information as possible
  - E.g., find best 2D approximation to 3D data
  - E.g., find best 2D approximation to 104D data
- In particular, choose projection that minimizes the squared error in reconstructing original data

# Dimensionality Reduction



- **Assumption:** Data lies on or near a low  $d$ -dimensional subspace
- Axes of this subspace are effective representation of the data

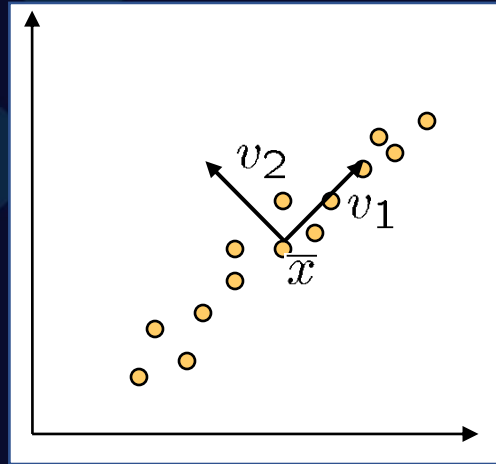
# Linear Dimensionality Reduction



PCA

# PCA – main idea

$\bar{x}$  is the mean of the orange points



How do we find  $v_1$  and  $v_2$  ?

- Consider the variation along direction  $v$  among all of the orange points:

$$var(v) = \sum_{\text{orange point } x} \|(x - \bar{x})^T \cdot v\|^2$$

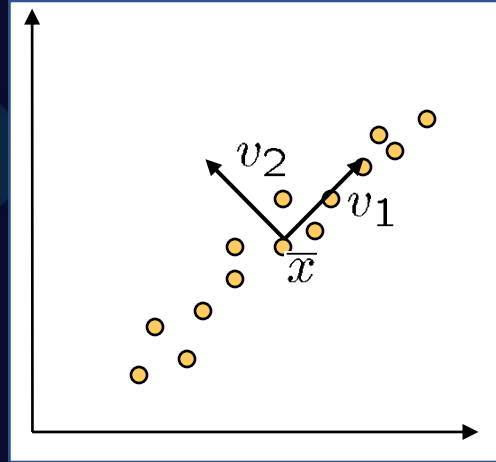
- What unit vector  $v$  minimizes  $var$ ?
- What unit vector  $v$  maximizes  $var$ ?

$$v_2 = \min_v \{var(v)\}$$

$$v_1 = \max_v \{var(v)\}$$

# PCA – main idea

$\bar{x}$  is the mean of the orange points



$$\begin{aligned} \max v^T A v \\ \text{s.t. } \|v\|=1 \end{aligned}$$

$$\begin{aligned} \text{var}(v) &= \sum_x \|(\mathbf{x} - \bar{\mathbf{x}})^T \cdot v\|^2 \\ &= \sum_x v^T (\mathbf{x} - \bar{\mathbf{x}}) (\mathbf{x} - \bar{\mathbf{x}})^T v \\ &= v^T \left[ \sum_x (\mathbf{x} - \bar{\mathbf{x}}) (\mathbf{x} - \bar{\mathbf{x}})^T \right] v \\ &= v^T A v \quad \text{where } A = \sum_x (\mathbf{x} - \bar{\mathbf{x}}) (\mathbf{x} - \bar{\mathbf{x}})^T \\ &= (X - \bar{X})(X - \bar{X})^T \end{aligned}$$

- Solution:

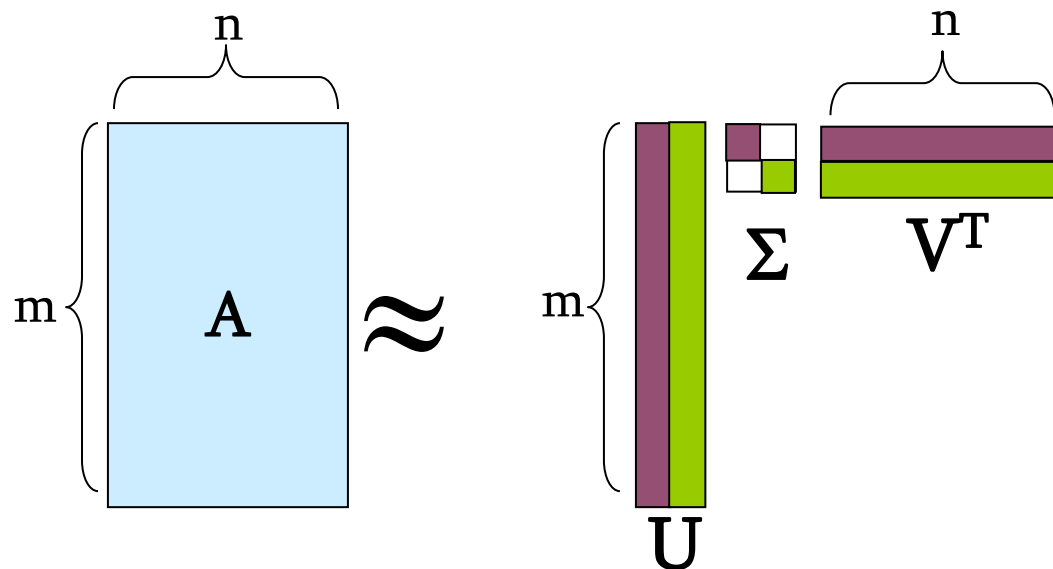
- $v_1$  is eigenvector of  $A$  with largest eigenvalue
- $v_2$  is eigenvector of  $A$  with smallest eigenvalue

# PCA Algorithm

- 1.  $X \leftarrow$  Create  $N \times d$  data matrix, with one row vector  $x_n$  per data point
- 2. Subtract mean  $\bar{x}$  from each row vector  $x_n$  in  $X$
- 3.  $\Sigma \leftarrow$  covariance matrix of  $X$
- 4. Find eigenvectors and eigenvalues of  $\Sigma$
- 5. PC's  $\leftarrow$  the  $M$  eigenvectors with largest eigenvalues

# SVD

$$\mathbf{A} \approx \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \sum_i \sigma_i \mathbf{u}_i \circ \mathbf{v}_i^T$$



# SVD – Definition

$$\mathbf{A}_{[m \times n]} = \mathbf{U}_{[m \times r]} \mathbf{\Sigma}_{[r \times r]} (\mathbf{V}_{[n \times r]})^T$$

- **A: Input data matrix**
  - $m \times n$  matrix (e.g.,  $m$  samples,  $n$  features)
- **U: Left singular vectors**
  - $m \times r$  matrix ( $m$  samples,  $r$  concepts)
- **$\Sigma$ : Singular values**
  - $r \times r$  diagonal matrix (strength of each 'concept')  
( $r$  : rank of the matrix **A**)
- **V: Right singular vectors**
  - $n \times r$  matrix ( $n$  features,  $r$  concepts)

# SVD – Definition

$$\mathbf{A}_{[m \times n]} = \mathbf{U}_{[m \times r]} \mathbf{\Sigma}_{[r \times r]} (\mathbf{V}_{[n \times r]})^T$$

- **A: Input data matrix**
  - $m \times n$  matrix (e.g.,  $m$  samples,  $n$  features)
- **U: Left singular vectors**
  - $m \times r$  matrix –  $U = eig(AA^T)$
- **$\Sigma$ : Singular values**
  - $r \times r$  diagonal matrix –  $\Sigma = diag(eigenvalues(AA^T))$
- **V: Right singular vectors**
  - $n \times r$  matrix –  $V = eig(A^T A)$

# SVD – Example: Users-to-Movies

- $A = U \Sigma V^T$  - example:

Diagram illustrating the SVD decomposition of a matrix  $A$  (Users-to-Movies) into  $U$ ,  $\Sigma$ , and  $V^T$ .

The matrix  $A$  is a 6x5 matrix with columns labeled Matrix, Alien, Serenity, Casablanca, and Amelie. The rows are labeled with movie genres: SciFi (rows 1-3) and Romance (rows 4-6). The matrix  $A$  is shown as:

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

The matrix  $A$  is decomposed into three matrices:

$$A = U \Sigma V^T = \begin{bmatrix} 0.18 & 0 \\ 0.36 & 0 \\ 0.18 & 0 \\ 0.90 & 0 \\ 0 & 0.53 \\ 0 & 0.80 \\ 0 & 0.27 \end{bmatrix} \times \begin{bmatrix} 9.64 & 0 \\ 0 & 5.29 \end{bmatrix} \times \begin{bmatrix} 0.58 & 0.58 & 0.58 & 0 & 0 \\ 0 & 0 & 0 & 0.71 & 0.71 \end{bmatrix}$$



# SVD - Interpretation #1

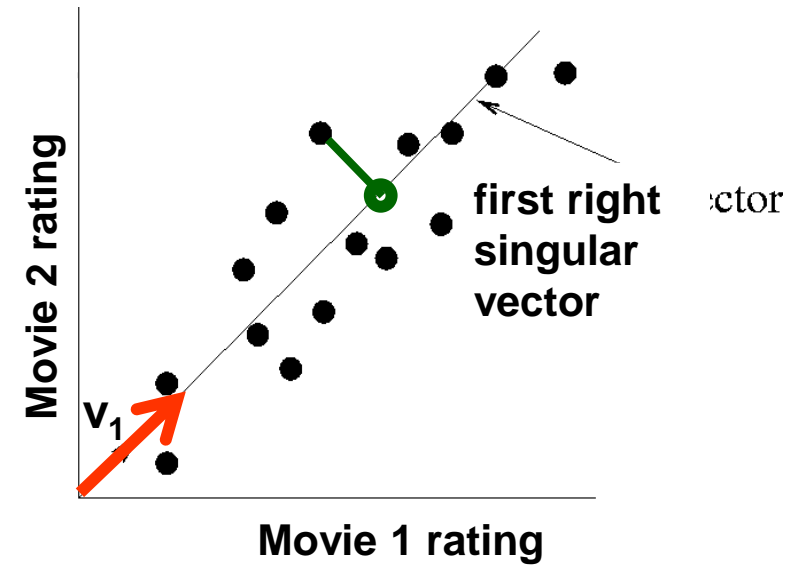
‘**movies**’, ‘**users**’ and ‘**concepts**’:

- **U**: user-to-concept similarity matrix
- **V**: movie-to-concept sim. matrix
- $\Sigma$ : its diagonal elements:  
‘strength’ of each concept

# SVD - Interpretation #2

•  $A = U \Sigma V^T$  - example:

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0.18 & 0 \\ 0.36 & 0 \\ 0.18 & 0 \\ 0.90 & 0 \\ 0 & 0.53 \\ 0 & 0.80 \\ 0 & 0.27 \end{bmatrix} \times \begin{bmatrix} 9.64 & 0 \\ 0 & 5.29 \end{bmatrix} \times \begin{bmatrix} 0.58 & 0.58 & 0.58 & 0 & 0 \\ 0 & 0 & 0 & 0.71 & 0.71 \end{bmatrix}$$



# SVD - Interpretation #2

- $A = U \Sigma V^T$  - example:

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0.18 & 0 \\ 0.36 & 0 \\ 0.18 & 0 \\ 0.90 & 0 \\ 0 & 0.53 \\ 0 & 0.80 \\ 0 & 0.27 \end{bmatrix} \times \begin{bmatrix} 9.64 & 0 \\ 0 & 5.29 \end{bmatrix} \times \begin{bmatrix} 0.58 & 0.58 & 0.58 & 0 & 0 \\ 0 & 0 & 0 & 0.71 & 0.71 \end{bmatrix}$$

variance ('spread')  
on the  $v_1$  axis

# SVD - Interpretation #2

- **Q: How exactly is dim. reduction done?**

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0.18 & 0 \\ 0.36 & 0 \\ 0.18 & 0 \\ 0.90 & 0 \\ 0 & 0.53 \\ 0 & 0.80 \\ 0 & 0.27 \end{bmatrix} \times \begin{bmatrix} 9.64 & 0 \\ 0 & 5.29 \end{bmatrix} \times \begin{bmatrix} 0.58 & 0.58 & 0.58 & 0 & 0 \\ 0 & 0 & 0 & 0.71 & 0.71 \end{bmatrix}$$

# SVD - Interpretation #2

- Q: How exactly is dim. reduction done?

• A:

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0.18 & 0 \\ 0.36 & 0 \\ 0.18 & 0 \\ 0.90 & 0 \\ 0 & 0.53 \\ 0 & 0.80 \\ 0 & 0.27 \end{bmatrix} \times \begin{bmatrix} 9.64 & 0 \\ 0 & 5.29 \end{bmatrix} \times \begin{bmatrix} 0.58 & 0.58 & 0.58 & 0 & 0 \\ 0 & 0 & 0 & 0.71 & 0.71 \end{bmatrix}$$

# SVD - Interpretation #2

- **Q:** How exactly is dim. reduction done?
- **A:** Set the smallest singular values to zero

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \sim \begin{bmatrix} 0.18 & 0 \\ 0.36 & 0 \\ 0.18 & 0 \\ 0.90 & 0 \\ 0 & 0.53 \\ 0 & 0.80 \\ 0 & 0.27 \end{bmatrix} \times \begin{bmatrix} 9.64 & 0 \\ 0 & 0 \end{bmatrix} \times \begin{bmatrix} 0.58 & 0.58 & 0.58 & 0 & 0 \\ 0 & 0 & 0 & 0.71 & 0.71 \end{bmatrix}$$

# SVD - Interpretation #2

- **Q:** How exactly is dim. reduction done?
- **A:** Set the smallest singular values to zero:

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \sim \begin{bmatrix} 0.18 & 0 \\ 0.36 & 0 \\ 0.18 & 0 \\ 0.90 & 0 \\ 0 & 0.53 \\ 0 & 0.80 \\ 0 & 0.27 \end{bmatrix} \times \begin{bmatrix} 9.64 & 0 \\ 0 & 0 \end{bmatrix} \times \begin{bmatrix} 0.58 & 0.58 & 0.58 & 0 & 0 \\ 0 & 0 & 0 & 0.71 & 0.71 \end{bmatrix}$$

*Note: In the original image, the second matrix and the third matrix are crossed out with large blue X's, indicating that the smallest singular values are being set to zero.*

# SVD - Interpretation #2

- **Q:** How exactly is dim. reduction done?
- **A:** Set the smallest singular values to zero:

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \sim \begin{bmatrix} 0.18 \\ 0.36 \\ 0.18 \\ 0.90 \\ 0 \\ 0 \\ 0 \end{bmatrix} \times \begin{bmatrix} 9.64 \end{bmatrix} \times \begin{bmatrix} 0.58 & 0.58 & 0.58 & 0 & 0 \end{bmatrix}$$



# SVD - Interpretation #2

- **Q:** How exactly is dim. reduction done?
- **A:** Set the smallest singular values to zero:

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \sim B = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

**Frobenius norm:**

$$\|M\|_F = \sqrt{\sum_{ij} M_{ij}^2}$$

$$\|A-B\|_F = \sqrt{\sum_{ij} (A_{ij}-B_{ij})^2}$$

is "small"

# Case study: How to query?

Q: Find users that like 'Matrix' and 'Alien'

$$\begin{array}{c} \uparrow \\ \text{SciFi} \\ \downarrow \\ \uparrow \\ \text{Romnce} \\ \downarrow \end{array} \begin{bmatrix} \text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \\ 1 & 1 & 1 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0.18 & 0 \\ 0.36 & 0 \\ 0.18 & 0 \\ 0.90 & 0 \\ 0 & 0.53 \\ 0 & 0.80 \\ 0 & 0.27 \end{bmatrix} \times \begin{bmatrix} 9.64 & 0 \\ 0 & 5.29 \end{bmatrix} \times \begin{bmatrix} 0.58 & 0.58 & 0.58 & 0 & 0 \\ 0 & 0 & 0 & 0.71 & 0.71 \end{bmatrix}$$

# Case study: How to query?

**Q: Find users that like 'Matrix' and 'Alien'**

**A: Map query into a 'concept space' – how?**

$$\begin{array}{c} \uparrow \\ \text{SciFi} \\ \downarrow \\ \uparrow \\ \text{Romnce} \\ \downarrow \end{array} \begin{bmatrix} \text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \\ 1 & 1 & 1 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0.18 & 0 \\ 0.36 & 0 \\ 0.18 & 0 \\ 0.90 & 0 \\ 0 & 0.53 \\ 0 & 0.80 \\ 0 & 0.27 \end{bmatrix} \times \begin{bmatrix} 9.64 & 0 \\ 0 & 5.29 \end{bmatrix} \times \begin{bmatrix} 0.58 & 0.58 & 0.58 & 0 & 0 \\ 0 & 0 & 0 & 0.71 & 0.71 \end{bmatrix}$$

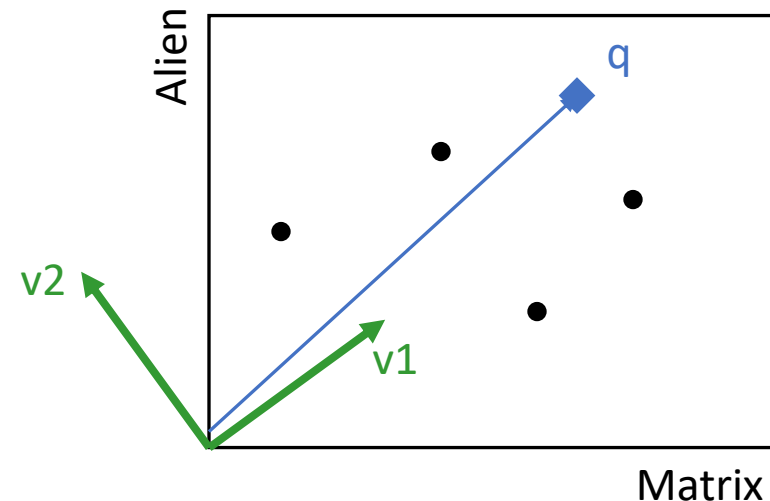
# Case study: How to query?

**Q: Find users that like 'Matrix'**

**A: map query vectors into 'concept space' – how?**

$$q = \begin{bmatrix} \text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \\ 5 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Project into concept space:  
Inner product with each  
'concept' vector  $v_i$



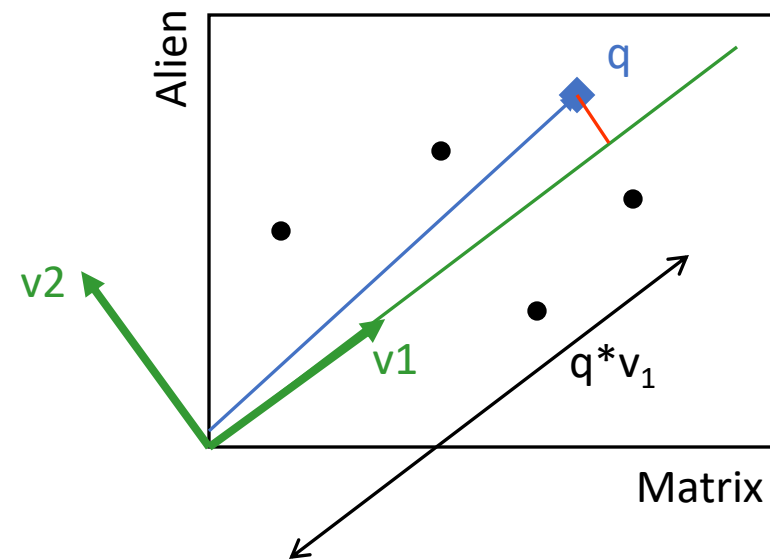
# Case study: How to query?

**Q: Find users that like 'Matrix'**

**A: map the vector into 'concept space' – how?**

$$q = \begin{bmatrix} \text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \\ 5 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Project into concept space:  
Inner product with each  
'concept' vector  $v_i$



# Case study: How to query?

Compactly, we have:

$$q_{\text{concept}} = q V$$

E.g.:

$$q = \begin{bmatrix} \text{Matrix} \\ 5 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \begin{matrix} \text{Alien} \\ \text{Serenity} \\ \text{Casablanca} \\ \text{Amelie} \end{matrix}$$
$$\begin{bmatrix} 0.58 & 0 \\ 0.58 & 0 \\ 0.58 & 0 \\ 0 & 0.71 \\ 0 & 0.71 \end{bmatrix} \quad \begin{matrix} \text{SciFi-concept} \\ \downarrow \\ 2.9 & 0 \end{matrix}$$

movie-to-concept similarities

# Case study: How to query?

How would the user  $d$  that rated ('Alien', 'Serenity') be handled?

$$d_{\text{concept}} = d \mathbf{V}$$

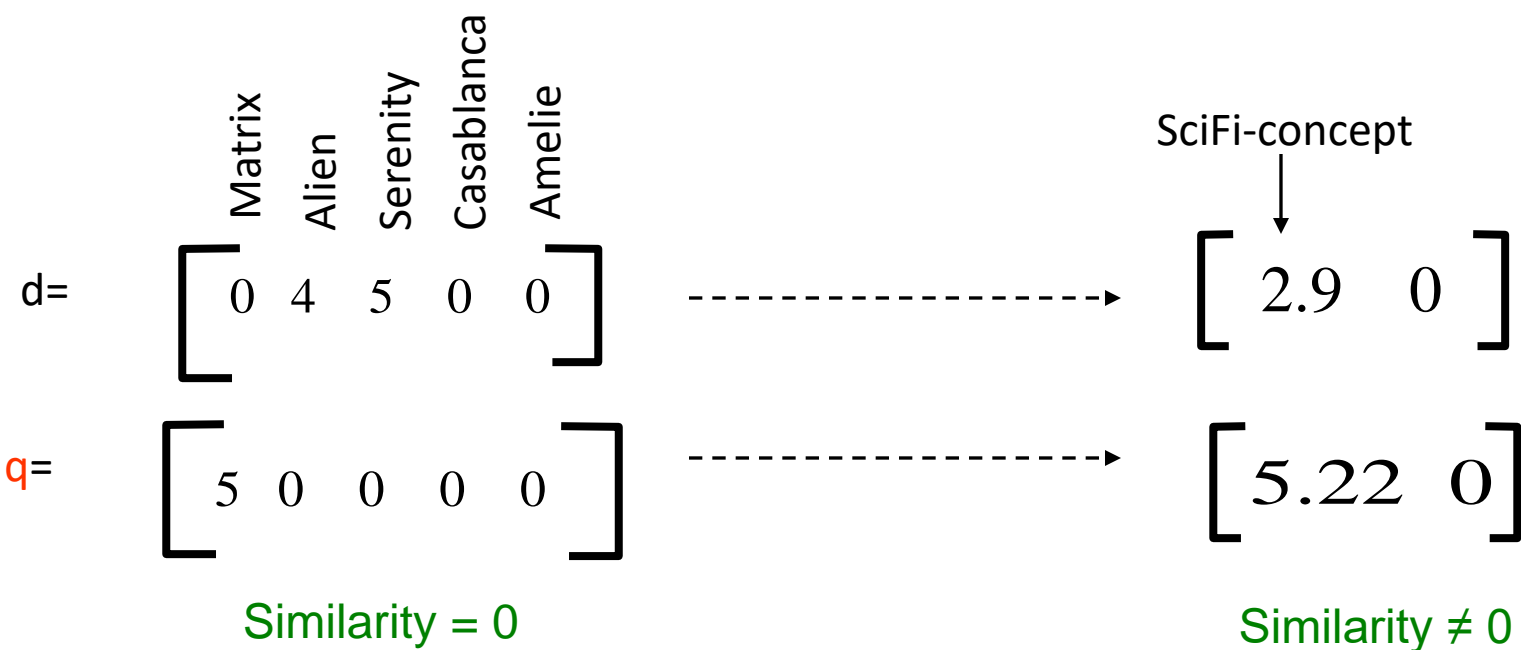
E.g.:

$$d = \begin{bmatrix} \text{Matrix} \\ 0 & 4 & 5 & 0 & 0 \end{bmatrix} \begin{matrix} \text{Alien} \\ \text{Serenity} \\ \text{Casablanca} \\ \text{Amelie} \end{matrix} = \begin{bmatrix} 0.58 & 0 \\ 0.58 & 0 \\ 0.58 & 0 \\ 0 & 0.71 \\ 0 & 0.71 \end{bmatrix} = \begin{bmatrix} \text{SciFi-concept} \\ 5.22 & 0 \end{bmatrix}$$

movie-to-concept similarities

# Case study: How to query?

**Observation:** User  $d$  that rated ('Alien', 'Serenity') will be similar to query "user"  $q$  that rated ('Matrix'), although  $d$  did not rate 'Matrix'!





# PCA consideration

## +Optimal reconstruction error:

- in Frobenius norm

## - Interpretability problem:

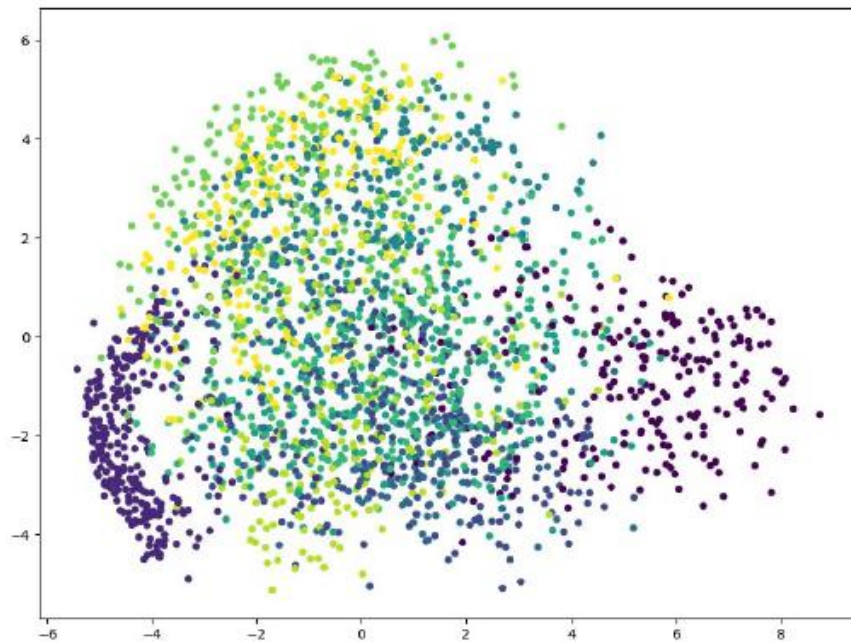
- A singular vector specifies a linear combination of all input columns or rows

# When will PCA work?

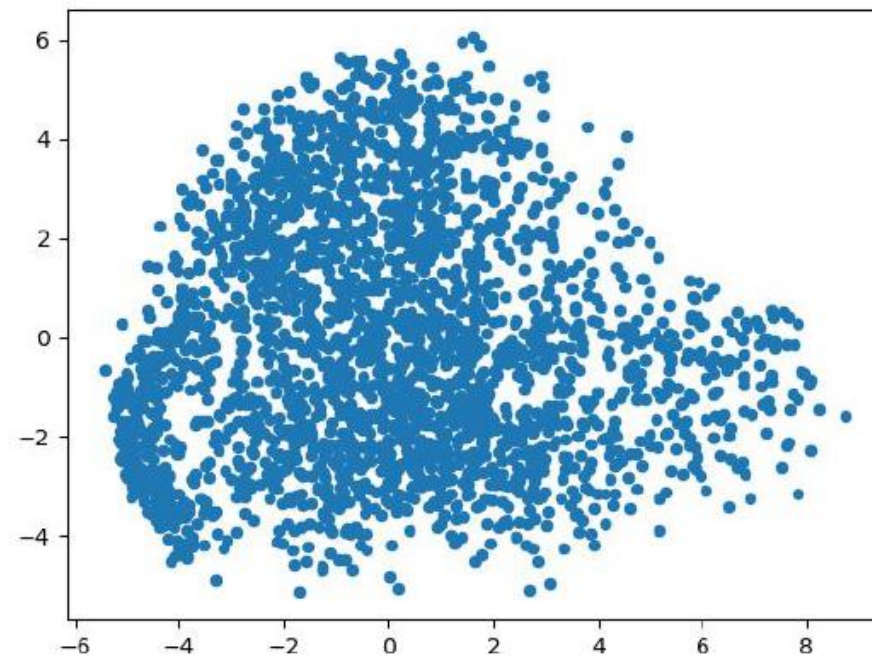
- Assumes relationships among variables are LINEAR
- Cloud of points in p-dimensional space has linear dimensions that can be effectively summarized by the principal axes.
- If the structure in the data is NONLINEAR (the cloud of points twists and curves its way through p-dimensional space), the principal axes will not be an efficient and informative summary of the data.

# Non-Linear Dimensionality Reduction

# PCA on MNIST



Visualization with labels



Visualization without labels

# PCA on MNIST

Why PCA fails to properly reduce dimensions of MNIST?

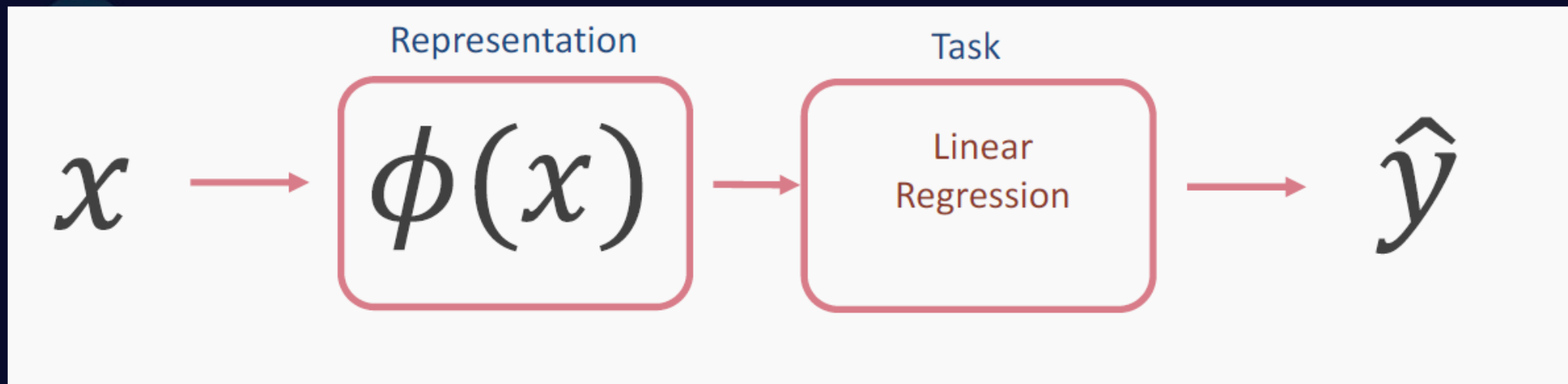
PCA is good, but it is a linear algorithm, meaning that it cannot represent complex relationship between features

t-SNE is non-linear dimensionality reduction technique that has better performance. It is designed mainly for visualization purposes.

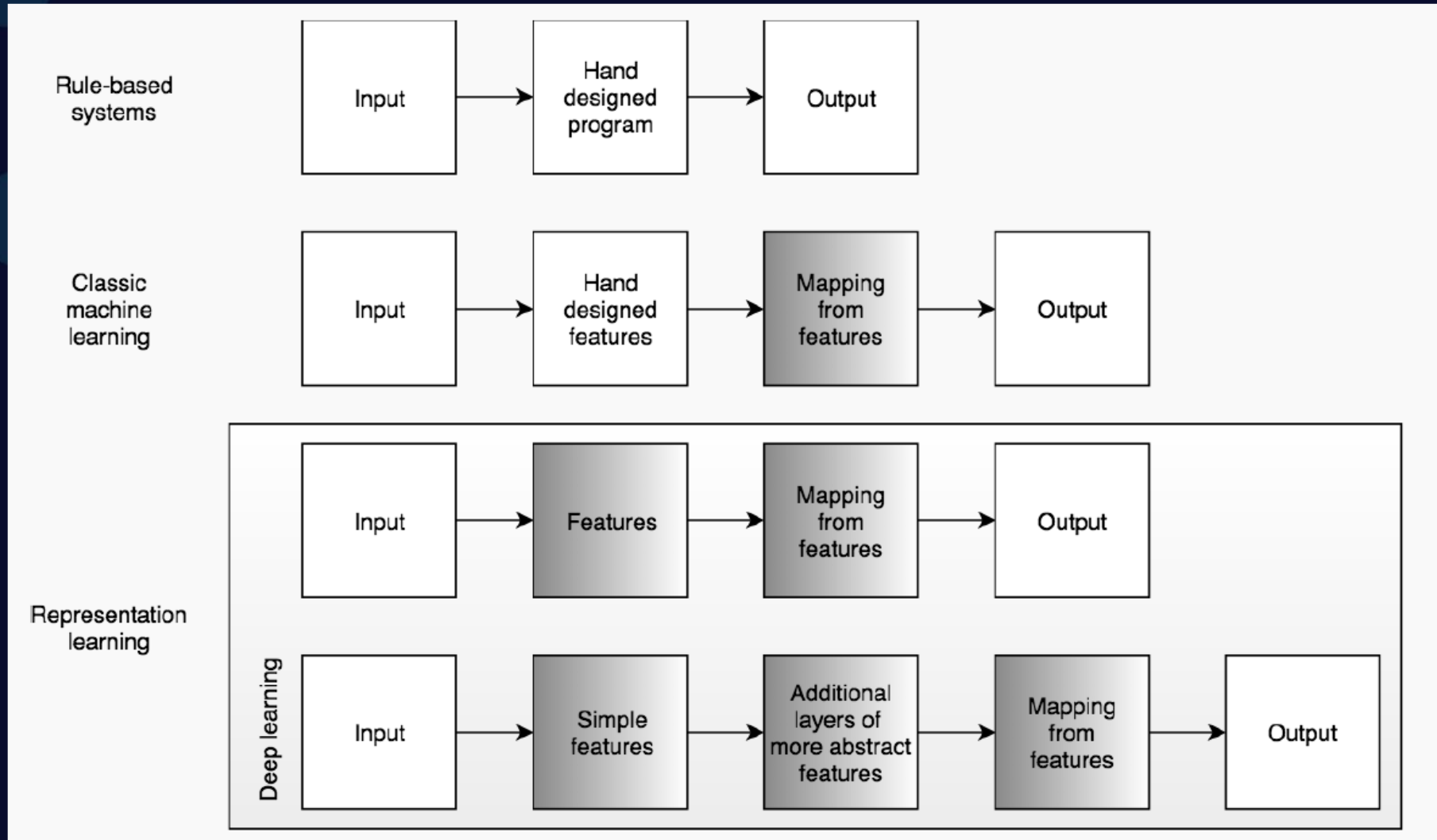
Why not use Neural Networks?

There is a dimensionality reduction technique based on Neural Network called Autoencoder!

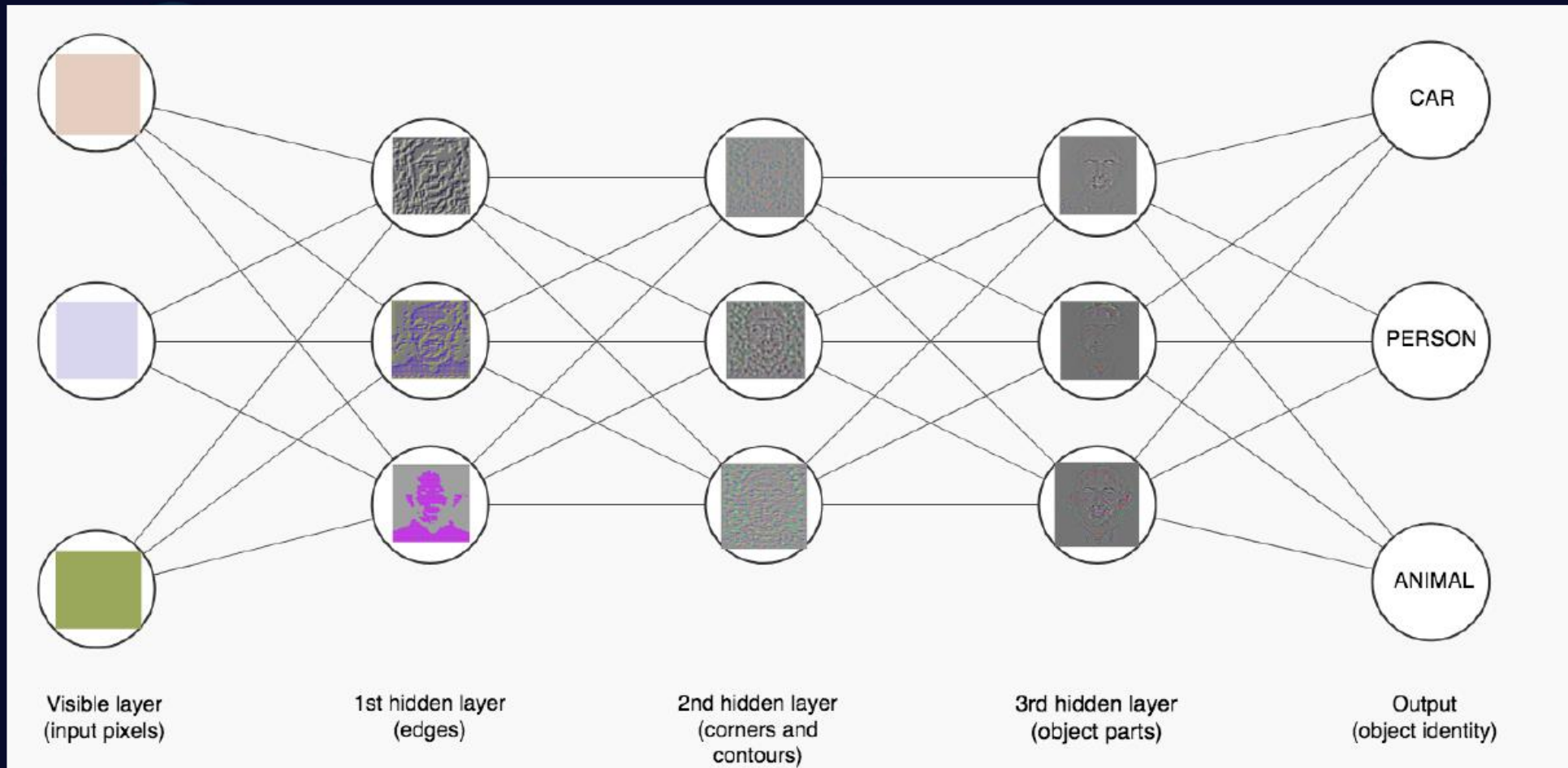
# Representational Learning



# Representational Learning



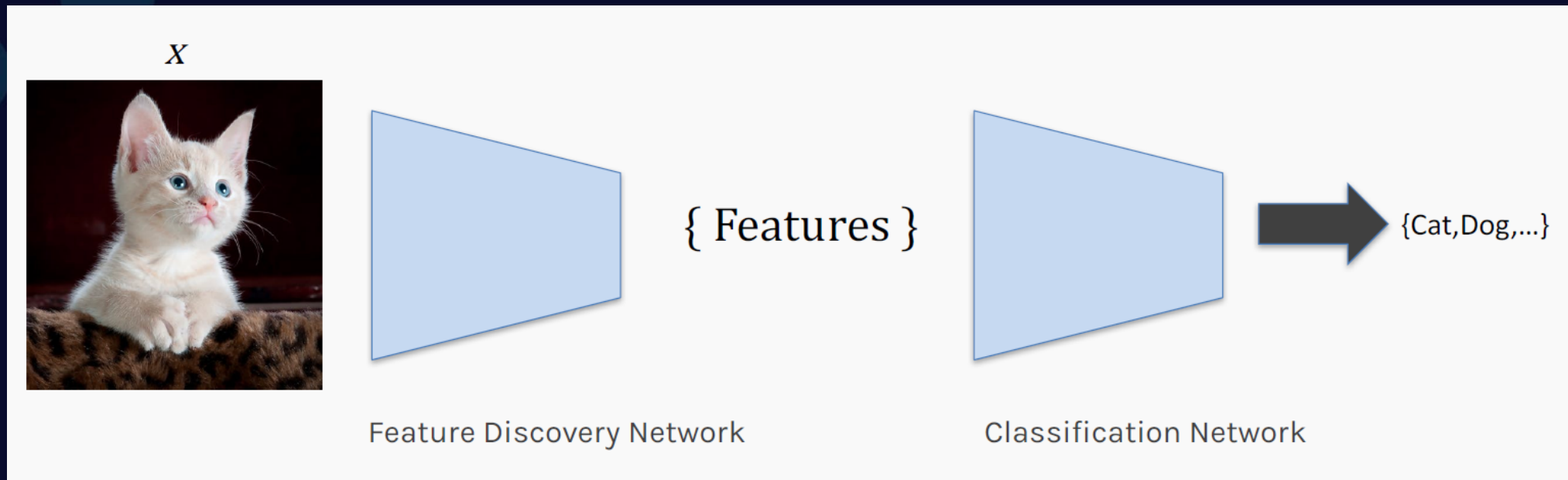
# Representational Learning





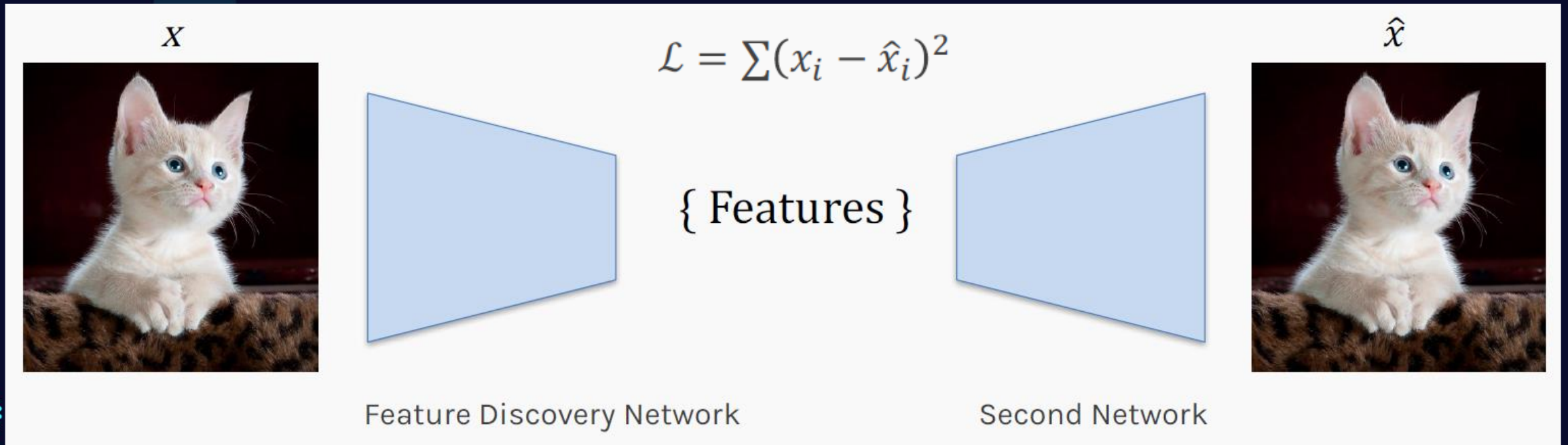
# Representational

- We train the two networks by minimizing the loss function (cross entropy loss)



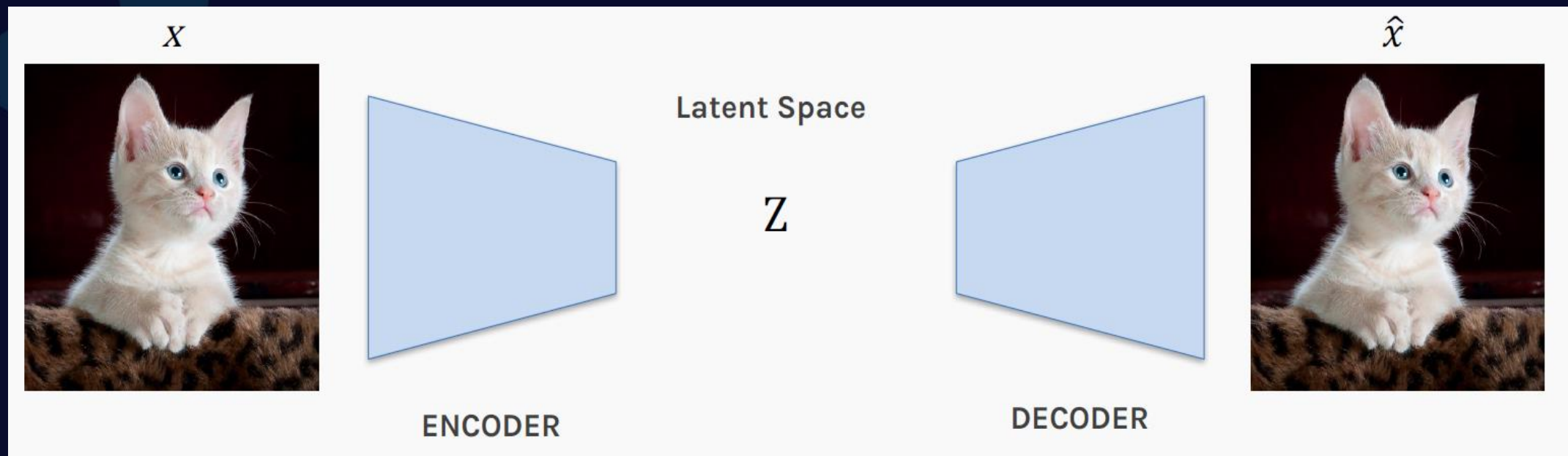
# Representational – No labels

- We train the two networks by minimizing the reconstruction loss function



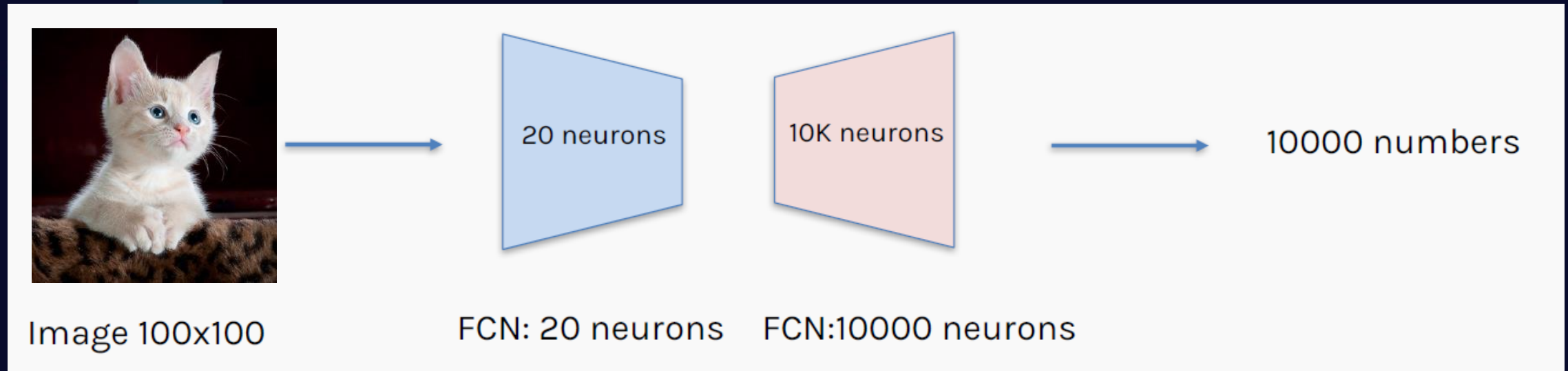
# Representational – Autoencoders

- Autoencoders

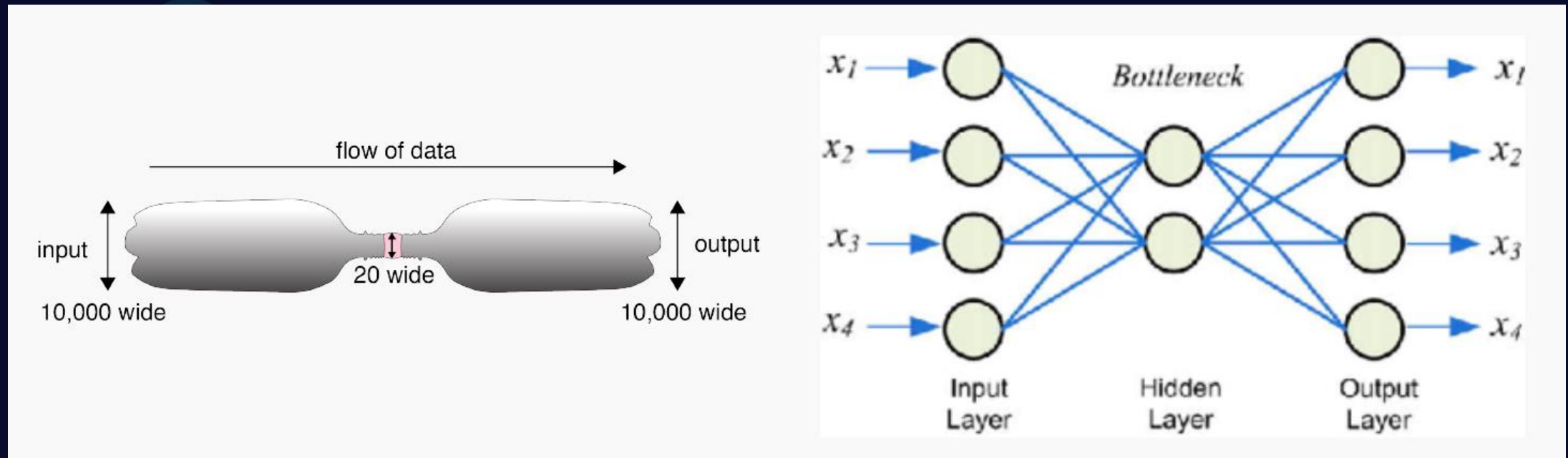


# Autoencoders

- Simplest example
















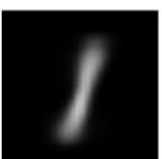


# Autoencoders – Bottleneck layer



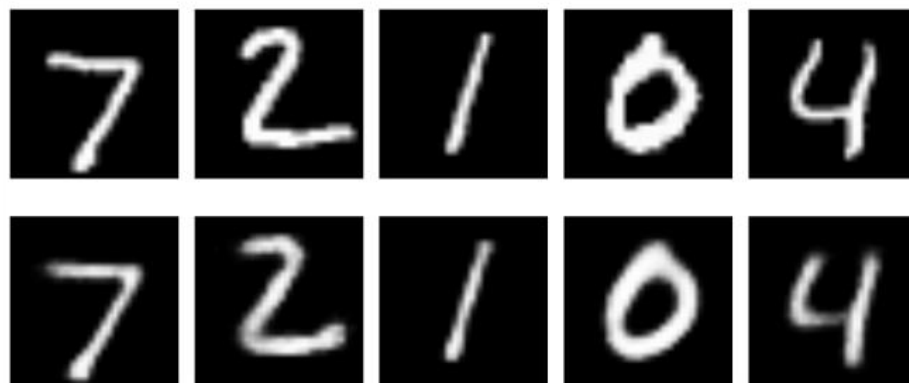
# Autoencoders – Bottleneck layer

- We say that an autoencoder is an example of semi-supervised or self-supervised learning.
- It sort-of is supervised learning because we give the system explicit goal data (the output should be the same as the input), and it sort-of isn't supervised learning because we don't have any manually determined labels or targets on the inputs.

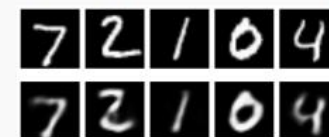
# Autoencoders – Bottleneck layer

20 latent variables	original					
	reconstructed					
10 latent variables	original					
	reconstructed					



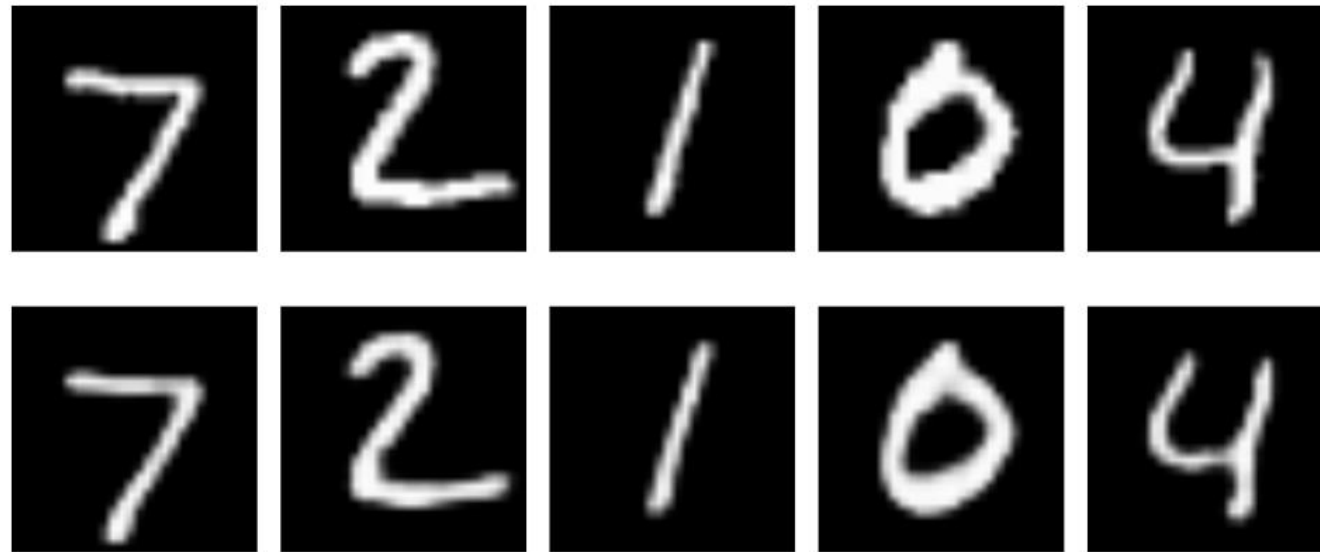
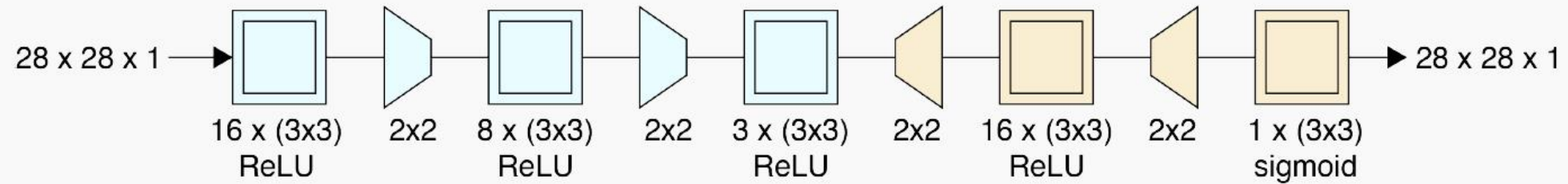


Shallow 20 latent variables

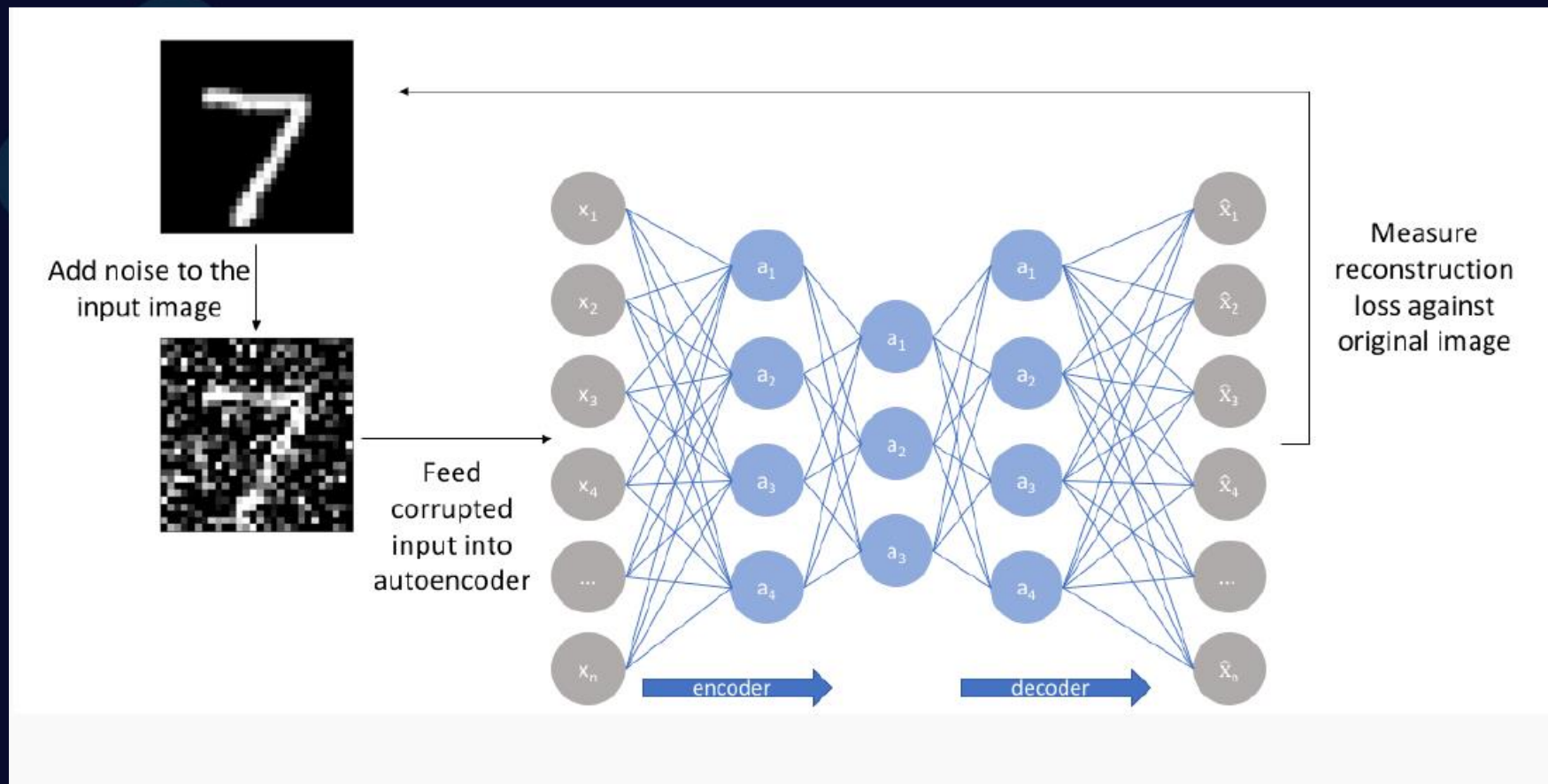




# Convolutional Autoencoders

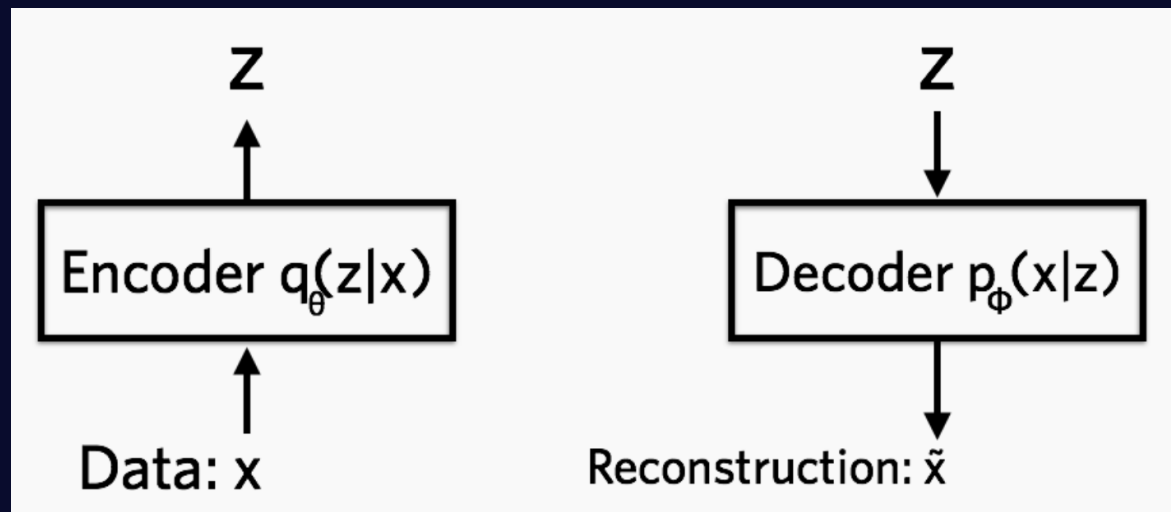


# Denoising Autoencoders



# Variational Autoencoder (VAE)

- Key idea: make both the encoder and the decoder probabilistic.
- I.e., the latent variables,  $z$ , are drawn from a probability distribution depending on the input,  $x$ , and the reconstruction is chosen probabilistically from  $z$ .

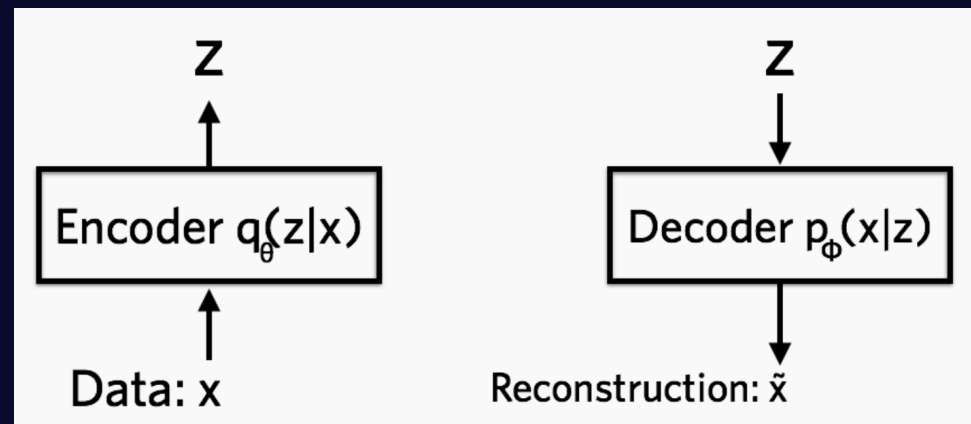


# VAE Encoder

- The encoder takes input and returns parameters for a probability density (e.g., Gaussian): I.e.,  $q_{\theta}(z | x)$  gives the mean and co-variance matrix.
- We can sample from this distribution to get random values of the lower-dimensional representation  $z$ .
- Implemented via a neural network: each input  $x$  gives a vector mean and diagonal covariance matrix that determine the Gaussian density

# VAE Decoder

- The decoder takes latent variable  $z$  and returns parameters for a distribution. E.g.,  $p_{\phi}(x|z)$  gives the mean and variance for each pixel in the output.
- Reconstruction is produced by sampling.
- Implemented via neural network, the NN parameters  $\phi$  are learned.



# VAE loss function

- Loss function for autoencoder:  $L_2$  distance between output and input (or clean input for denoising case)
- For VAE, we need to learn parameters of two probability distributions. For a single input,  $x_i$ , we maximize the expected value of returning  $x_i$  or minimize the expected negative log likelihood.

$$-\mathbb{E}_{z \sim q_\theta(z|x_i)}[\log p_\phi(x_i | z)]$$

- This takes expected value wrt  $z$  over the current distribution  $q_\theta(z|x_i)$  of the loss  $-\log p_\phi(x_i|z)$

# VAE loss function

- **Problem:** the weights may adjust to memorize input images via  $z$ . I.e., input that we regard as similar may end up very different in  $z$  space.
- We prefer continuous latent representations to give meaningful parameterizations. E.g., smooth changes from one digit to another.
- **Solution:** Try to force  $q_{\theta}(z|x_i)$  to be close to a standard normal (or some other simple density).

# VAE loss function

- For a single data point  $x_i$  we get the loss function

$$l_i(\theta, \phi) = -\mathbb{E}_{z \sim q_\theta(z|x_i)} [\log p_\phi(x_i | z)] + \text{KL}(q_\theta(z | x_i) || p(z))$$

- The first term promotes recovery of the input.
- The second term keeps the encoding continuous – the encoding is compared to a fixed  $p(z)$  regardless of the input, which inhibits memorization.
- With this loss function the VAE can (almost) be trained using gradient descent on minibatches.



# VAE loss function

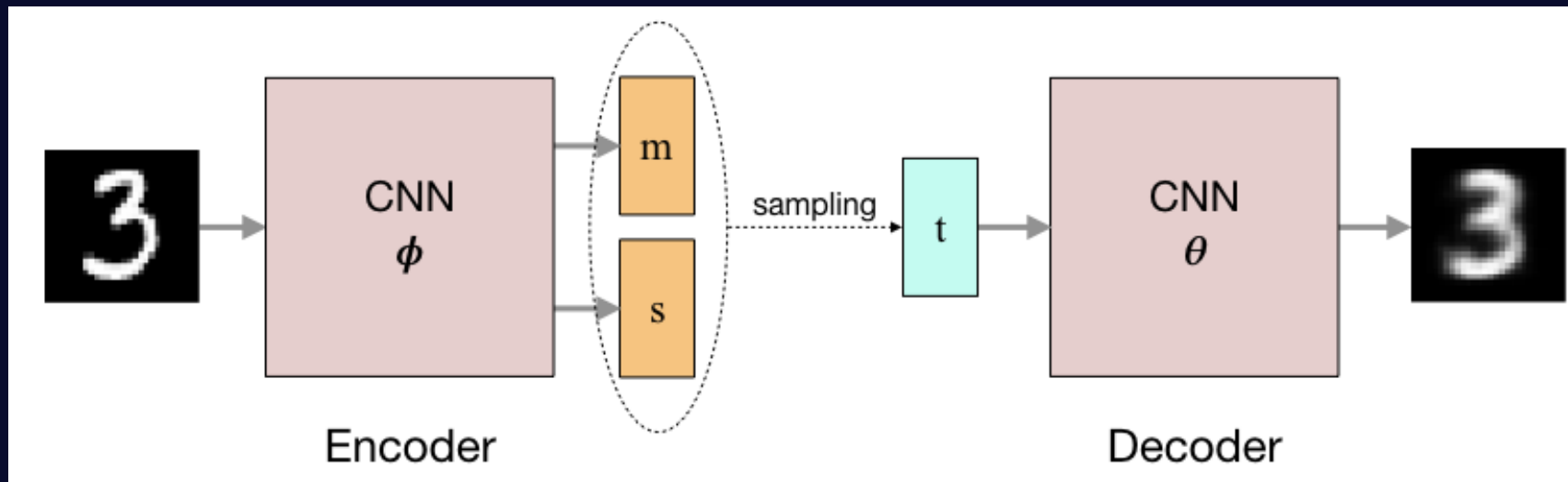
- For a single data point  $x_i$  we get the loss function

$$l_i(\theta, \phi) = -\mathbb{E}_{z \sim q_\theta(z|x_i)} [\log p_\phi(x_i | z)] + \text{KL}(q_\theta(z | x_i) || p(z))$$

- **Problem:** The expectation would usually be approximated by choosing samples and averaging. This is not differentiable wrt  $\theta$  and  $\phi$ .

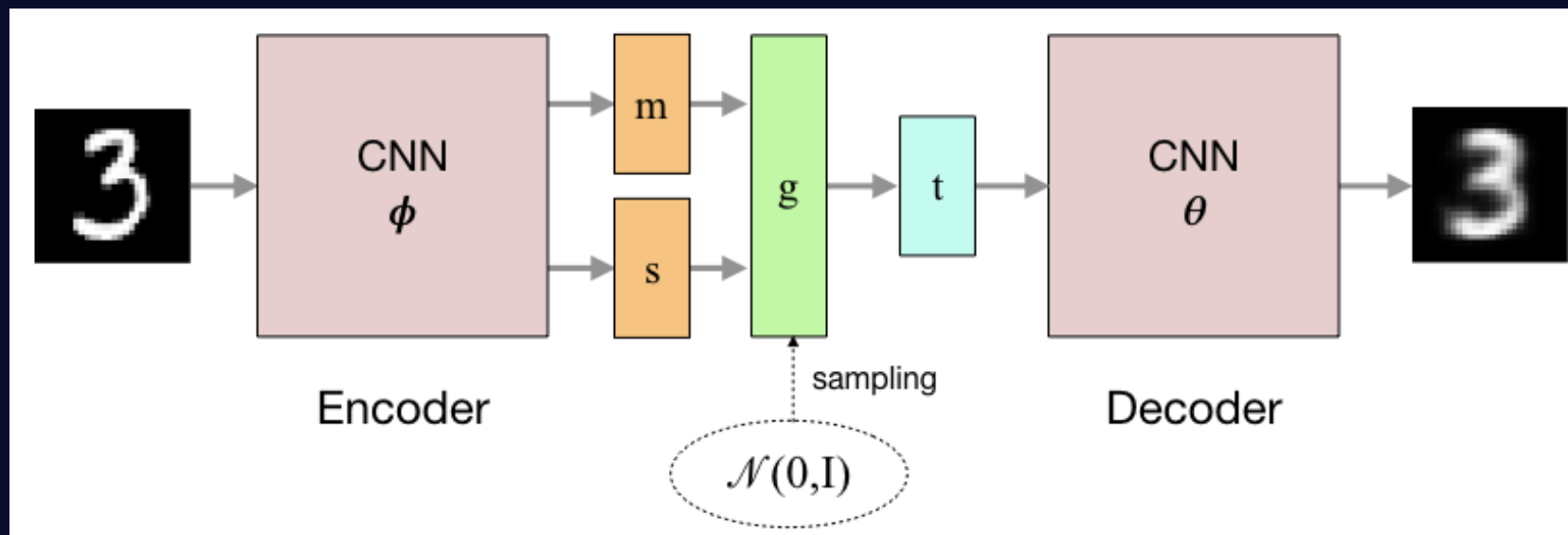
# VAE loss function

- **Problem:** The expectation would usually be approximated by choosing samples and averaging. This is not differentiable wrt  $\theta$  and  $\phi$ .



# VAE loss function

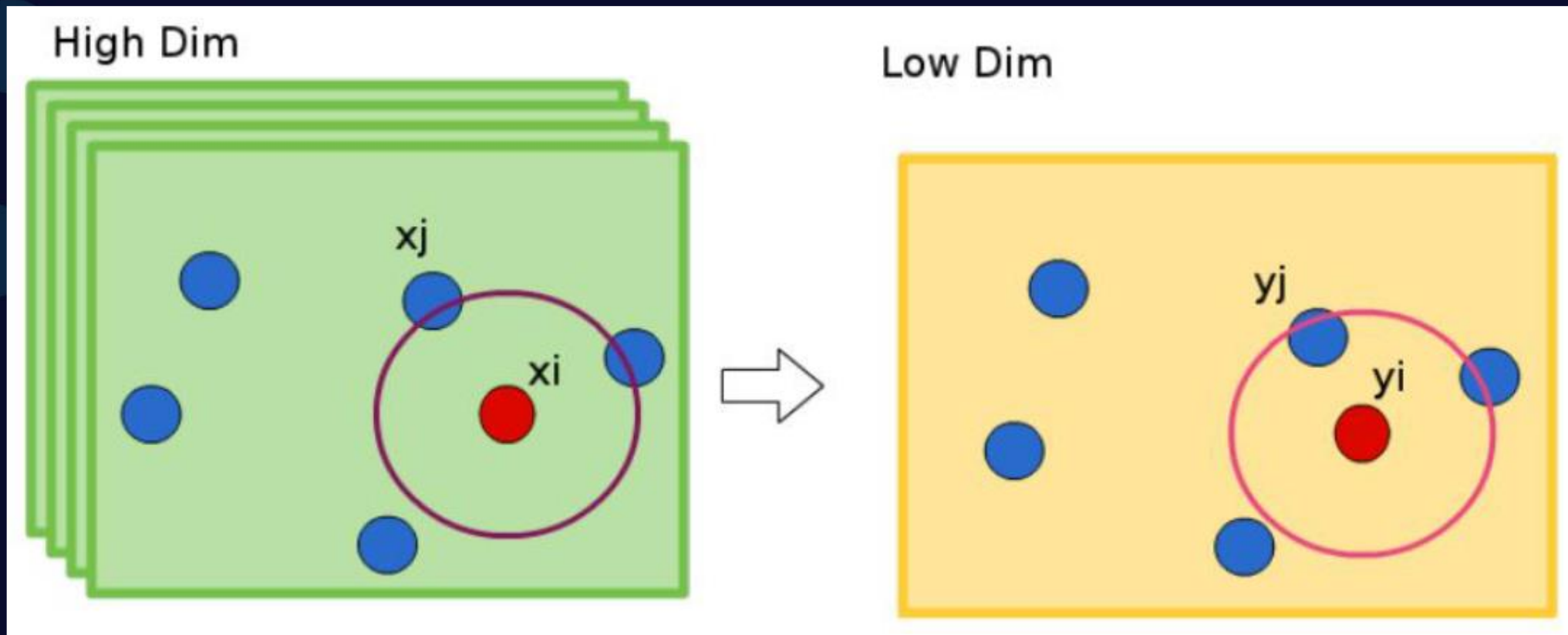
- **Reparameterization:** If  $z$  is  $N(\mu(x_i), \Sigma(x_i))$ , then we can sample  $z$  using  $z = \mu(x_i) + \sqrt{\Sigma(x_i)} \epsilon$ , where  $\epsilon$  is  $N(0,1)$ . So we can draw samples from  $N(0,1)$ , which doesn't depend on the parameters.



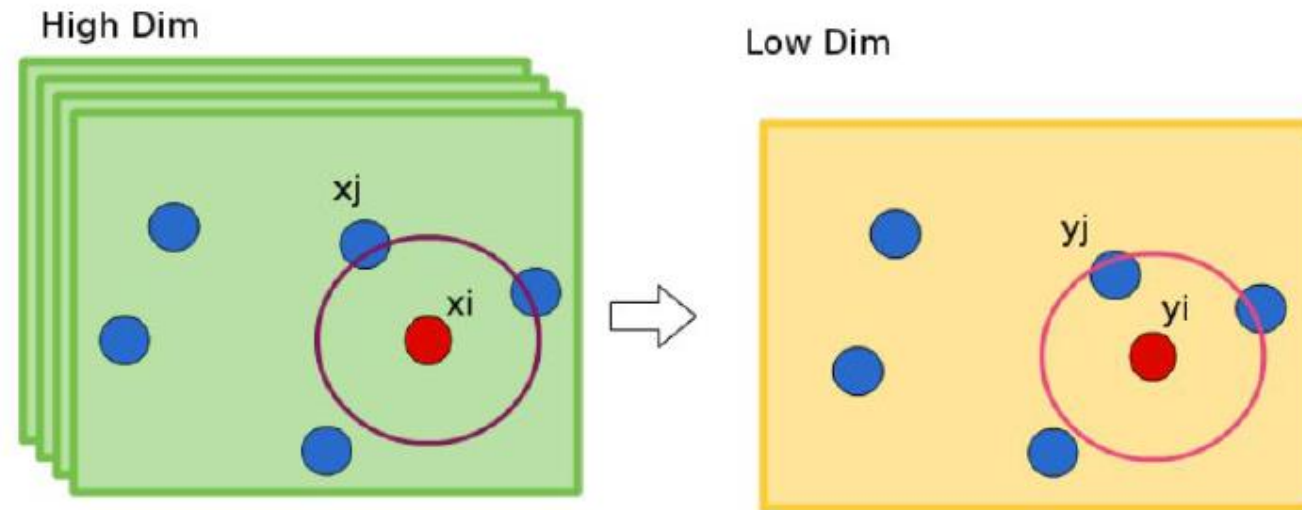
# VAE generative model

- After training,  $q_{\theta}(z|x_i)$  is close to a standard normal,  $N(0,1)$  – easy to sample.
- Using a sample of  $z$  from  $q_{\theta}(z|x_i)$  as input to sample from  $p_{\phi}(x|z)$  gives an approximate reconstruction of  $x_i$ , at least in expectation.
- If we sample any  $z$  from  $N(0,1)$  and use it as input to sample from then we can approximate the entire data distribution  $p(x)$ . I.e., we can generate new samples that look like the input but aren't in the input.

# SNE



# SNE



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

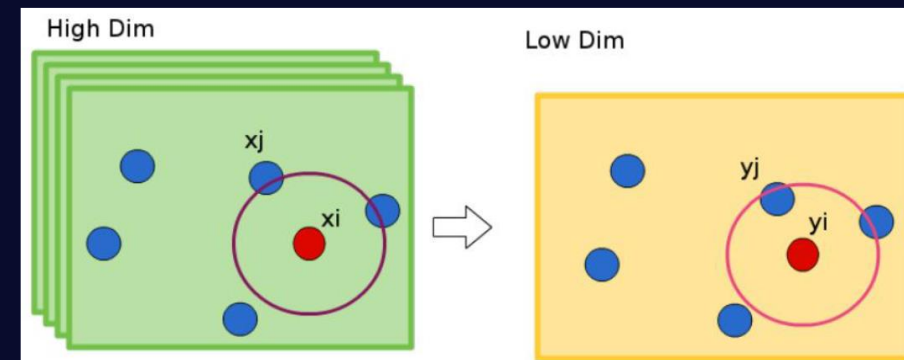
# SNE

Similarities in higher dim.

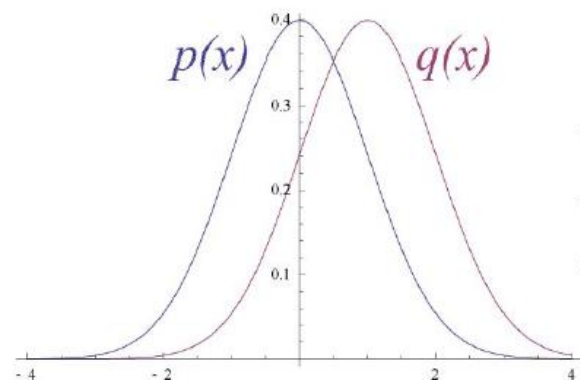
$$p_{j|i} = \frac{\exp(-||x_i - x_j||^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-||x_i - x_k||^2 / 2\sigma_i^2)}$$

Similarities in lower dim.

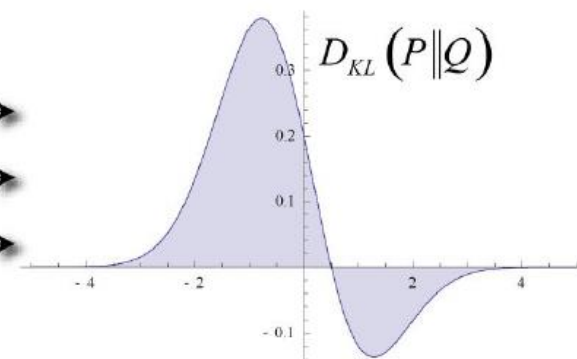
$$q_{j|i} = \frac{\exp(-||y_i - y_j||^2)}{\sum_{k \neq i} \exp(-||y_i - y_k||^2)}$$



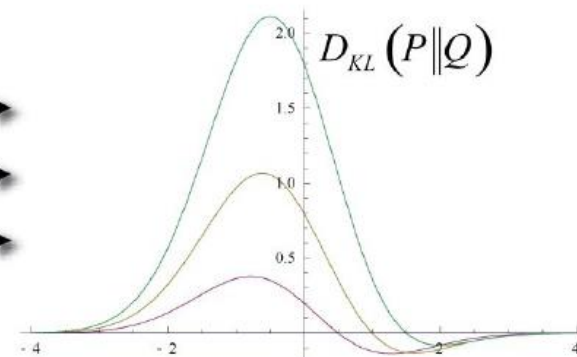
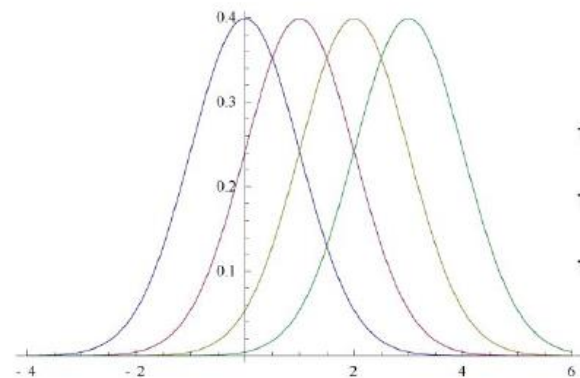
# SNE



Original Gaussian PDF's



KL Area to be Integrated



$$D_{KL}(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}.$$



# SNE - Loss function

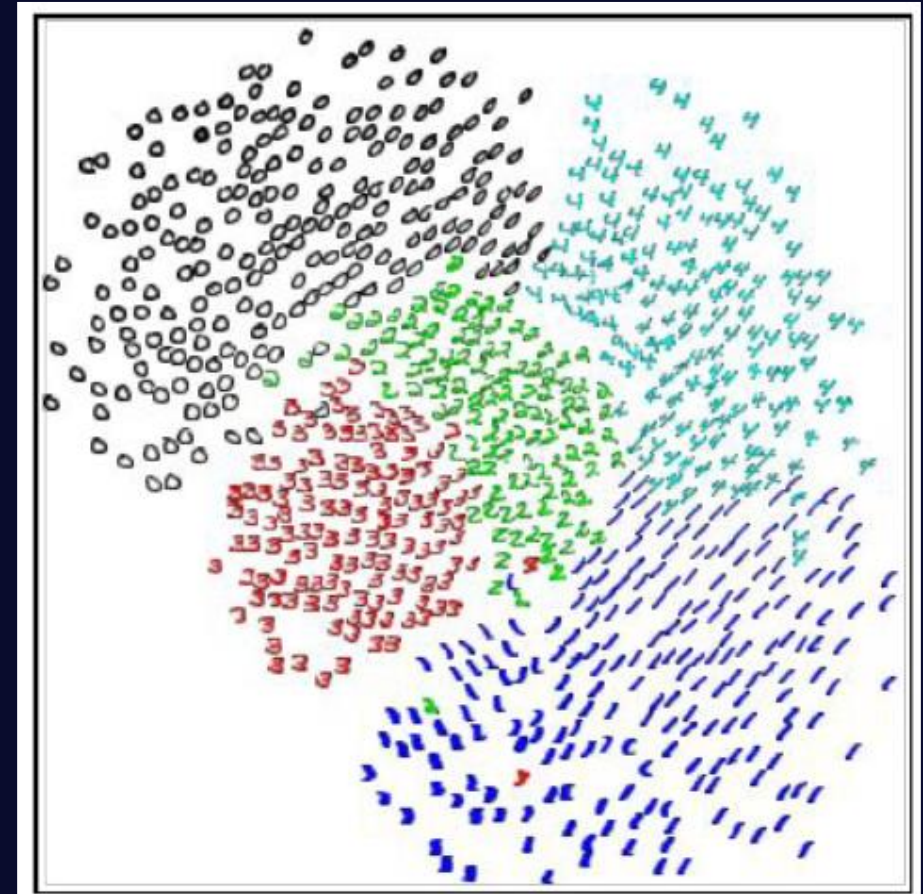
$$C = \sum_i KL(P_i || Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

$$\frac{\partial C}{\partial y_i} = \sum_{j \neq i} (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j)$$

# SNE

The result of running the SNE algorithm on 3000 256-dimensional grayscale images of handwritten digits. Pictures of the original data vectors  $x_i$  (scans of handwritten digit) are shown at the location corresponding to their low-dimensional images  $y_i$  as found by SNE.

The classes are quite well separated even though SNE had no information about class labels. Furthermore, within each class, properties like orientation, skew and stroke thickness tend to vary smoothly across the space.



# Symetric-SNE

$$C = KL(P||Q) = \sum_i \sum_{j \neq i} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

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

$$q_{ij} = \frac{\exp(-||y_i - y_j||^2)}{\sum_{k \neq i} \exp(-||y_i - y_k||^2)}$$

# Symetric-SNE

This results in

$$p_{ij} = p_{ji}, q_{ij} = q_{ji}$$

The main advantage is a simplified form of the gradient

$$\frac{\partial \mathcal{C}}{\partial y_i} = 2 \sum_j (p_{ij} - q_{ij})(y_i - y_j)$$

# t-SNE

- In high dimension we have more room, points can have a lot of different neighbors. In 2D a point can have a few neighbors at distance one all far from each other - what happens when we embed in 1D?
- This is the "crowding problem" - we don't have enough room to accommodate all neighbors. This is one of the biggest problems with SNE.

# t-SNE

Instead of using the same distribution in both dimensions we will use heavy tailed distribution at the lower dim. (t-distribution)

$$q_{ij} \propto (1 + ||y_i - y_j||^2)^{-1}$$

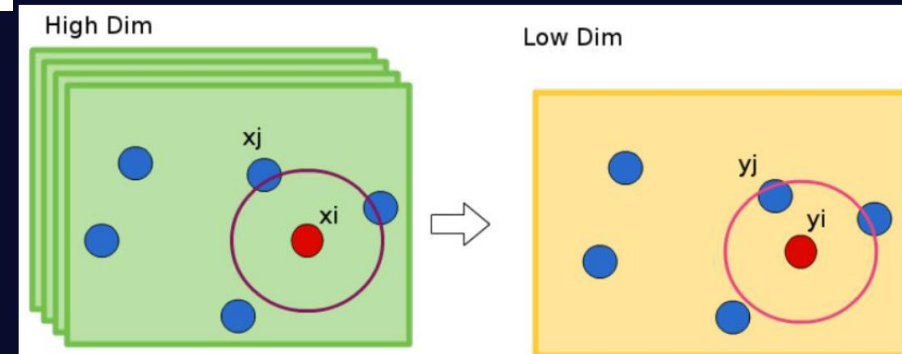
# t-SNE

Similarities in higher dim.

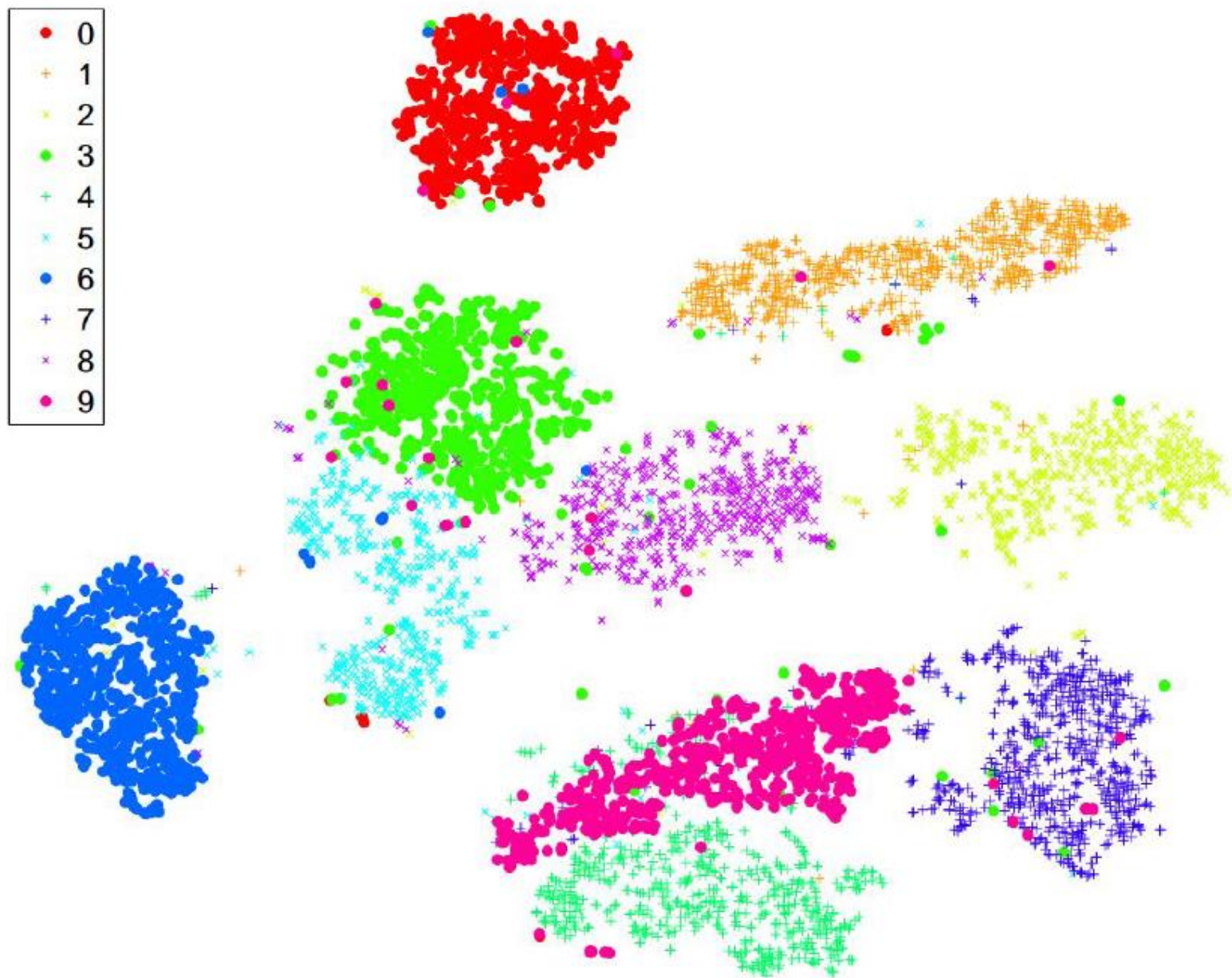
$$p_{ij} = \frac{\exp(-||x_i - x_j||^2 / 2\sigma^2)}{\sum_{k \neq i} \exp(-||x_i - x_k||^2 / 2\sigma^2)}$$

Similarities in lower dim.

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









# Clustering

# Unsupervised learning: clustering

Raw data



extract  
features

features

$f_1, f_2, f_3, \dots, f_n$

$f_1, f_2, f_3, \dots, f_n$

$f_1, f_2, f_3, \dots, f_n$

$f_1, f_2, f_3, \dots, f_n$

$f_1, f_2, f_3, \dots, f_n$

group into  
classes/clust  
ers

Clusters

**No “supervision”, we’re only given data and want to find natural groupings**

# Unsupervised learning: modeling

Most frequently, when people think of unsupervised learning they think clustering

Another category: learning probabilities/parameters for models without supervision

- Learn a translation dictionary
- Learn a grammar for a language
- Learn the social graph

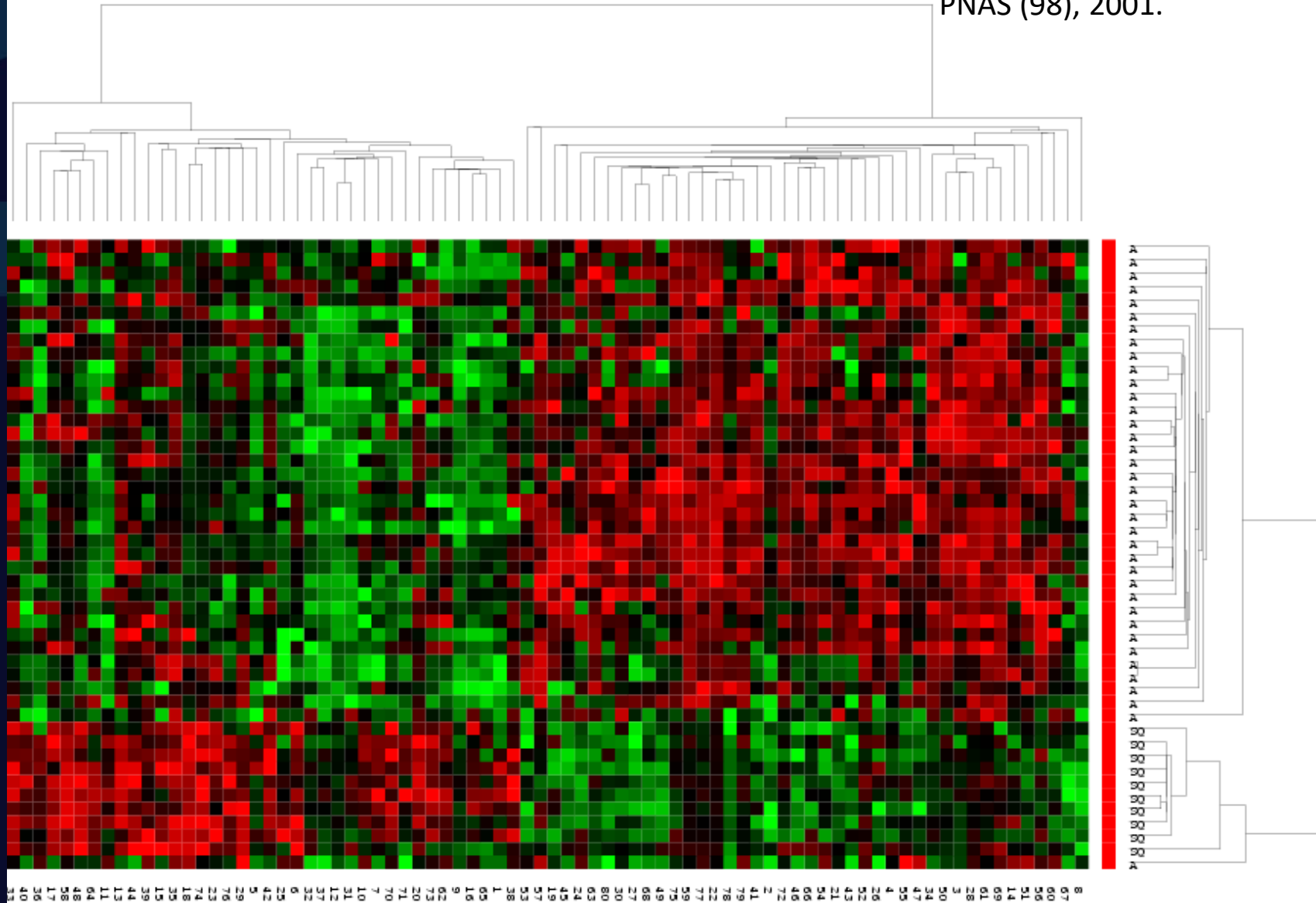
# Clustering

**Clustering:** the process of grouping a set of objects into classes of similar objects

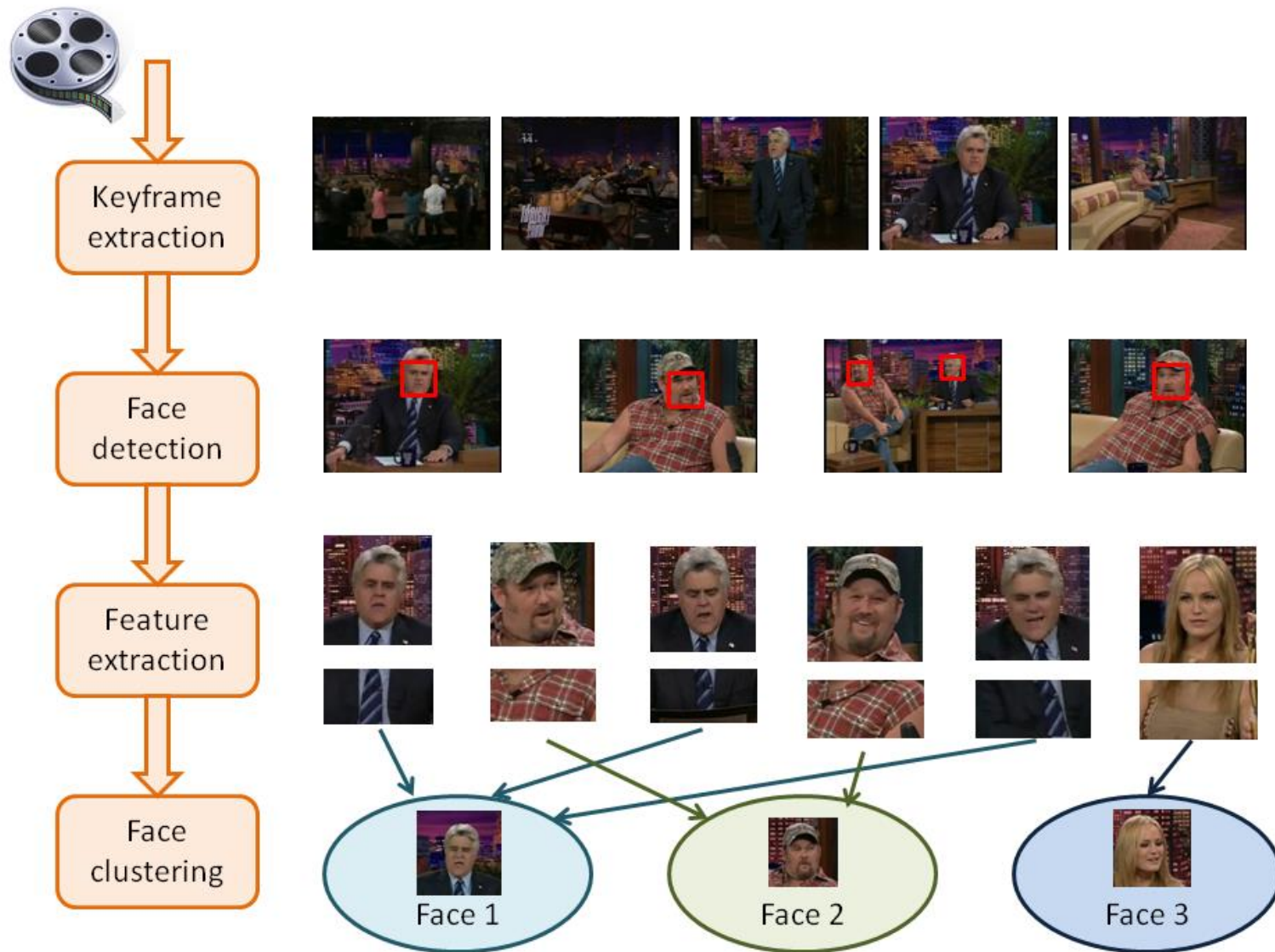
**Applications?**

# Gene expression data

Data from Garber et al.  
PNAS (98), 2001.



# Face Clustering

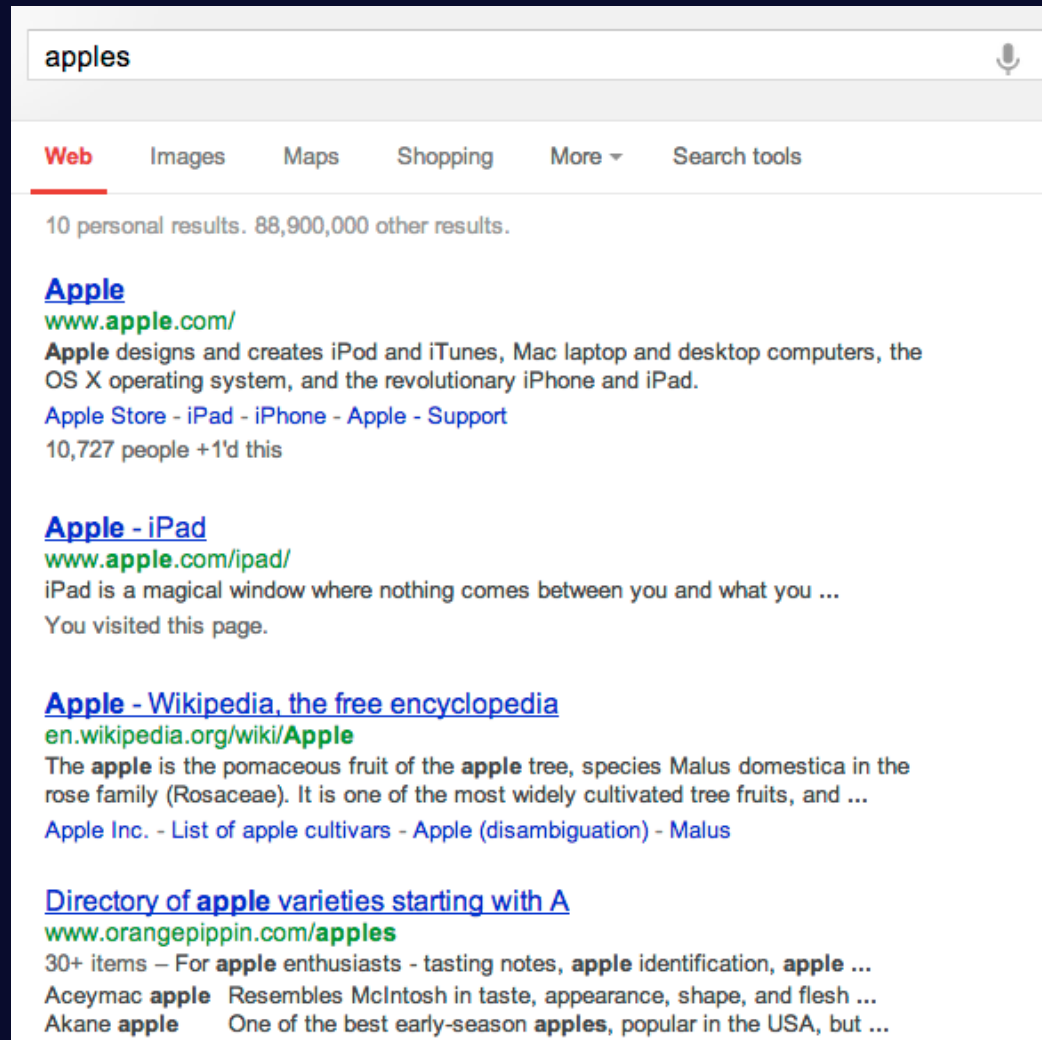




# Face clustering



# Search result clustering



apples

Web Images Maps Shopping More Search tools

10 personal results. 88,900,000 other results.

[Apple](#)  
[www.apple.com/](http://www.apple.com/)  
Apple designs and creates iPod and iTunes, Mac laptop and desktop computers, the OS X operating system, and the revolutionary iPhone and iPad.  
[Apple Store](#) - [iPad](#) - [iPhone](#) - [Apple](#) - [Support](#)  
10,727 people +1'd this

[Apple - iPad](#)  
[www.apple.com/ipad/](http://www.apple.com/ipad/)  
iPad is a magical window where nothing comes between you and what you ...  
You visited this page.

[Apple - Wikipedia, the free encyclopedia](#)  
[en.wikipedia.org/wiki/Apple](http://en.wikipedia.org/wiki/Apple)  
The **apple** is the pomaceous fruit of the **apple** tree, species *Malus domestica* in the rose family (Rosaceae). It is one of the most widely cultivated tree fruits, and ...  
[Apple Inc.](#) - [List of apple cultivars](#) - [Apple \(disambiguation\)](#) - [Malus](#)

[Directory of apple varieties starting with A](#)  
[www.orangeippin.com/apples](http://www.orangeippin.com/apples)  
30+ items – For **apple** enthusiasts - tasting notes, **apple** identification, **apple** ...  
**Aceymac apple** Resembles McIntosh in taste, appearance, shape, and flesh ...  
**Akane apple** One of the best early-season **apples**, popular in the USA, but ...



# Google News

Google

News

## Top Stories

Iran  
Xbox One  
Tarun Tejpal  
Manny Pacquiao  
Ukraine  
Kabul  
New England Patriots  
Latvia  
Derrick Rose  
Doctor Who

+ Xbox One



E! Online

See realtime coverage

## Console Wars 2013: Microsoft's Xbox One vs. Sony's PlayStation 4

E! Online - 1 hour ago

The future is now! Last week, Sony released its next generation console, PlayStation 4. This weekend, Microsoft drops the much touted all-in-one media device, Xbox One. We've been geeking out over the two new systems, and compiling a report on the new ...

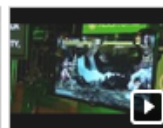
[Xbox One sales exceed one million in first 24 hours](#) Joystiq - by David Hinkle  
[Xbox One vs. PS4: A Guide to Making the Toughest Gaming Decision in Years](#)

ABC News - by Joanna Stern

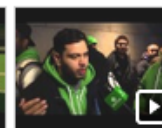
Related  
[Microsoft »](#)



Wall Street Journal



YouTube



YouTube



Washingto...



RedOrbit



Guardian ...



The Guardian

## Xbox One and Microsoft websites marred by problems on launch day

The Guardian  
9 hours ago



Written by  
Jemima Kiss

Microsoft's Xbox One launch was marred by problems with its online services early on Friday which took down the official website Xbox.

## Consumers line up for Xbox One

USA TODAY - Nov 23, 2013

Eager video game players lined up at stores across the country awaiting the arrival of Microsoft's Xbox One, a week to the day after rival Sony introduced its PlayStation 4. The console, available for sale tonight at 12:01 a.m.



Product Re

## Here are all the Xbox One voice commands

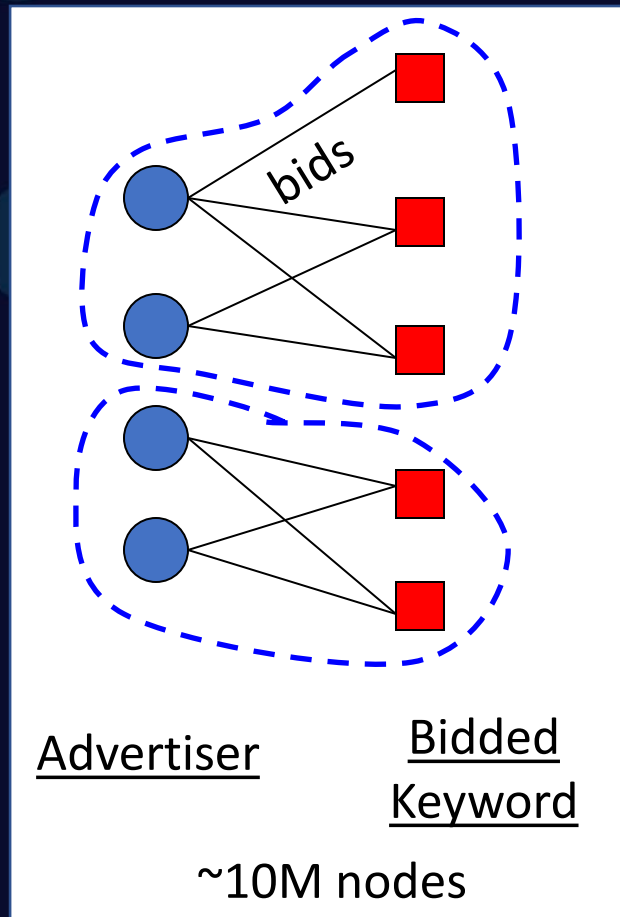
Polygon  
9 hours ago



Written by  
Megan Farokhmanesh

Microsoft posted a guide to Xbox One voice commands, including how to navigate menus, control volume and multitask, on its Tumblr.

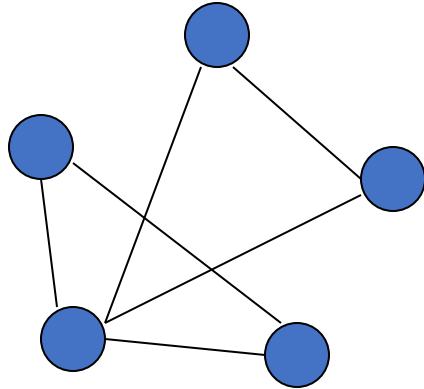
# Clustering in search advertising



Find clusters of advertisers and keywords

- Keyword suggestion
- Performance estimation

# Clustering applications



Who-messages-who IM/text/twitter graph

~100M nodes

## Find clusters of users

- Targeted advertising
- Exploratory analysis

## Clusters of the Web Graph

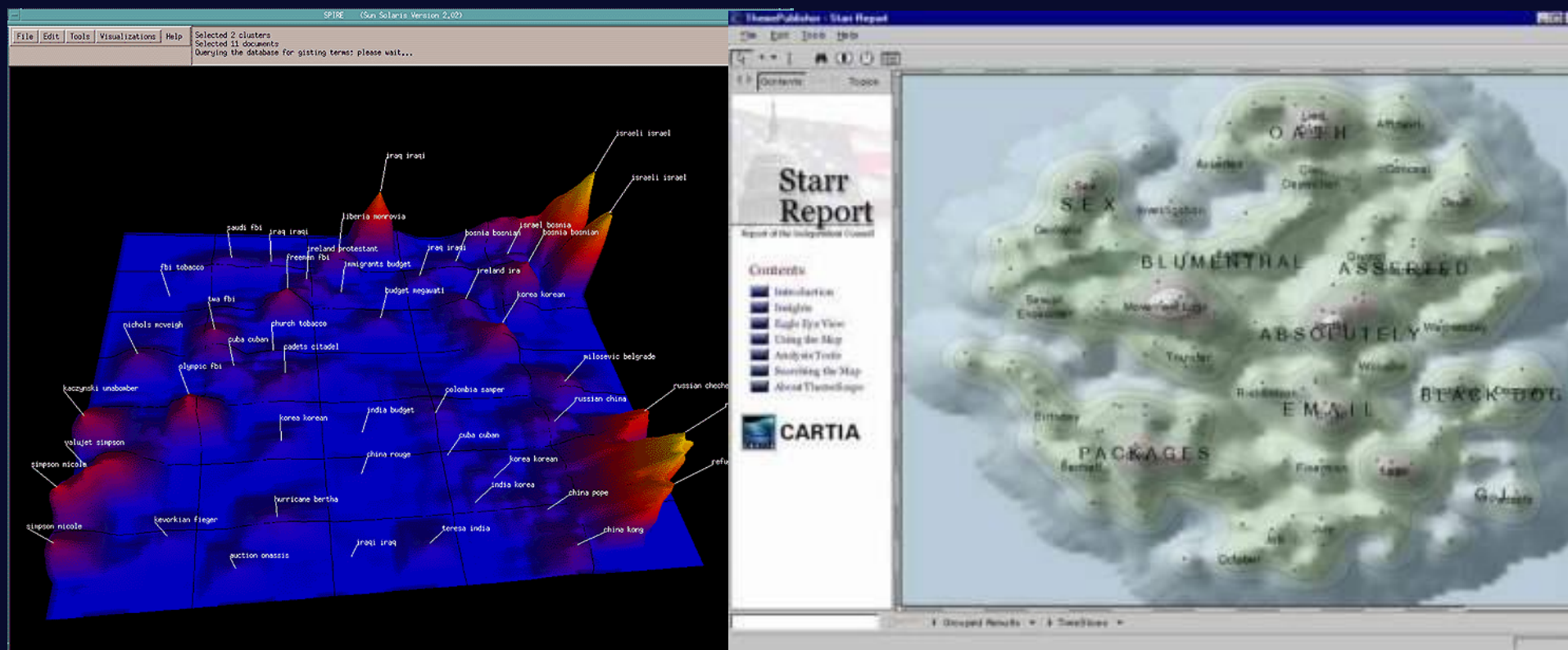
- Distributed pagerank computation

# Data visualization

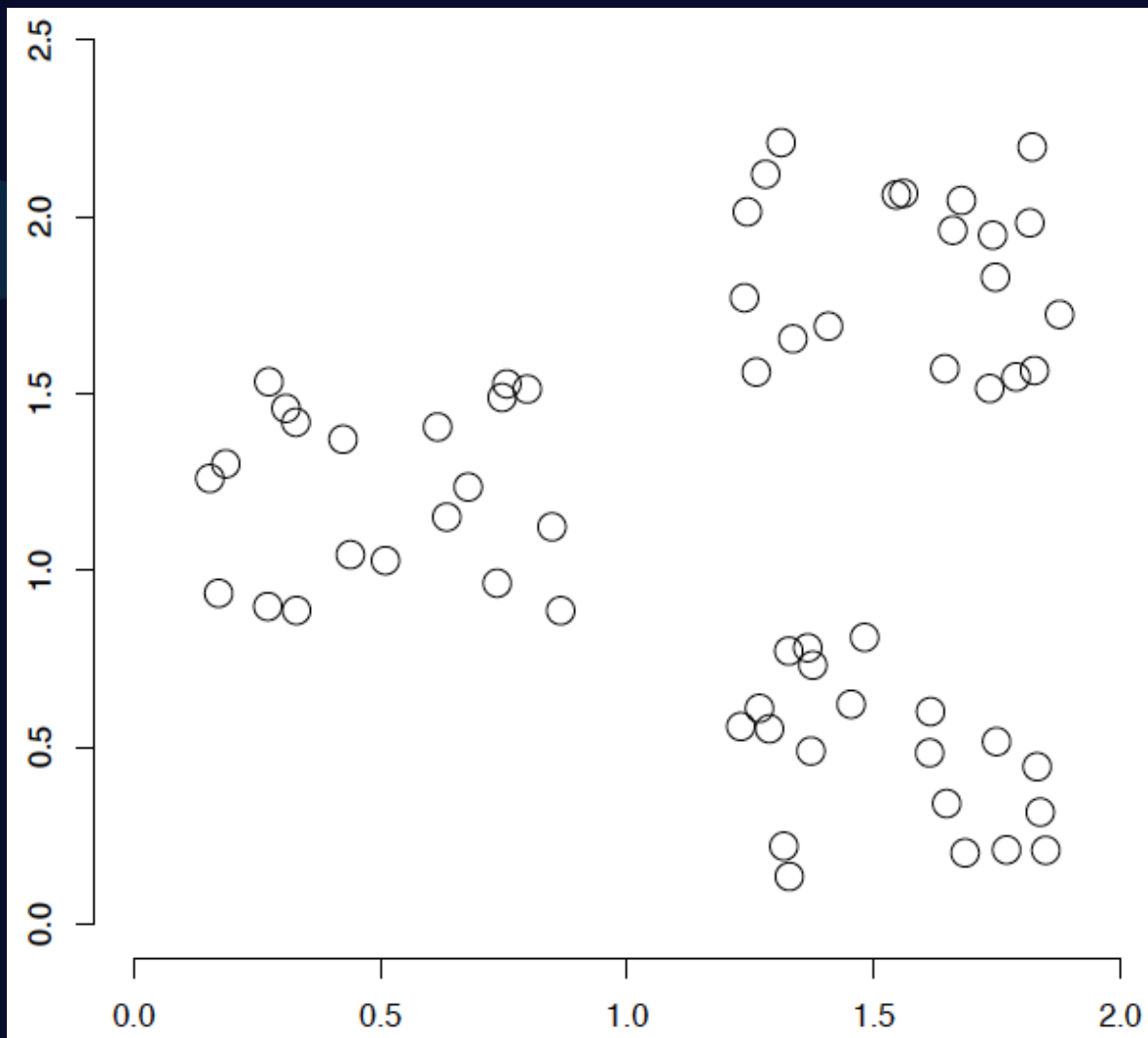
## Wise et al, “Visualizing the non-visual” PNNL

# ThemeSapes, Cartia

- [Mountain height = cluster size]



# A data set with clear cluster structure



# Issues for clustering

## Representation for clustering

- How do we represent an example
  - features, etc.
- Similarity/distance between examples

## Flat clustering or hierarchical

## Number of clusters

- Fixed a priori
- Data driven?

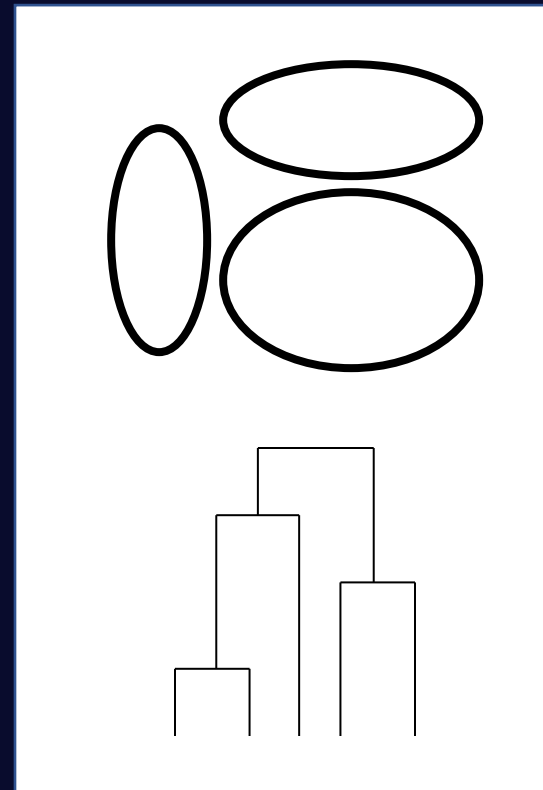
# Clustering Algorithms

## Flat algorithms

- Usually start with a random (partial) partitioning
- Refine it iteratively
  - $K$  means clustering
  - Model based clustering
- Spectral clustering

## Hierarchical algorithms

- Bottom-up, agglomerative
- Top-down, divisive



# Hard vs. soft clustering

Hard clustering: Each example belongs to exactly one cluster

Soft clustering: An example can belong to more than one cluster (probabilistic)

- Makes more sense for applications like creating browsable hierarchies
- You may want to put a pair of sneakers in two clusters: (i) sports apparel and (ii) shoes



# K-means

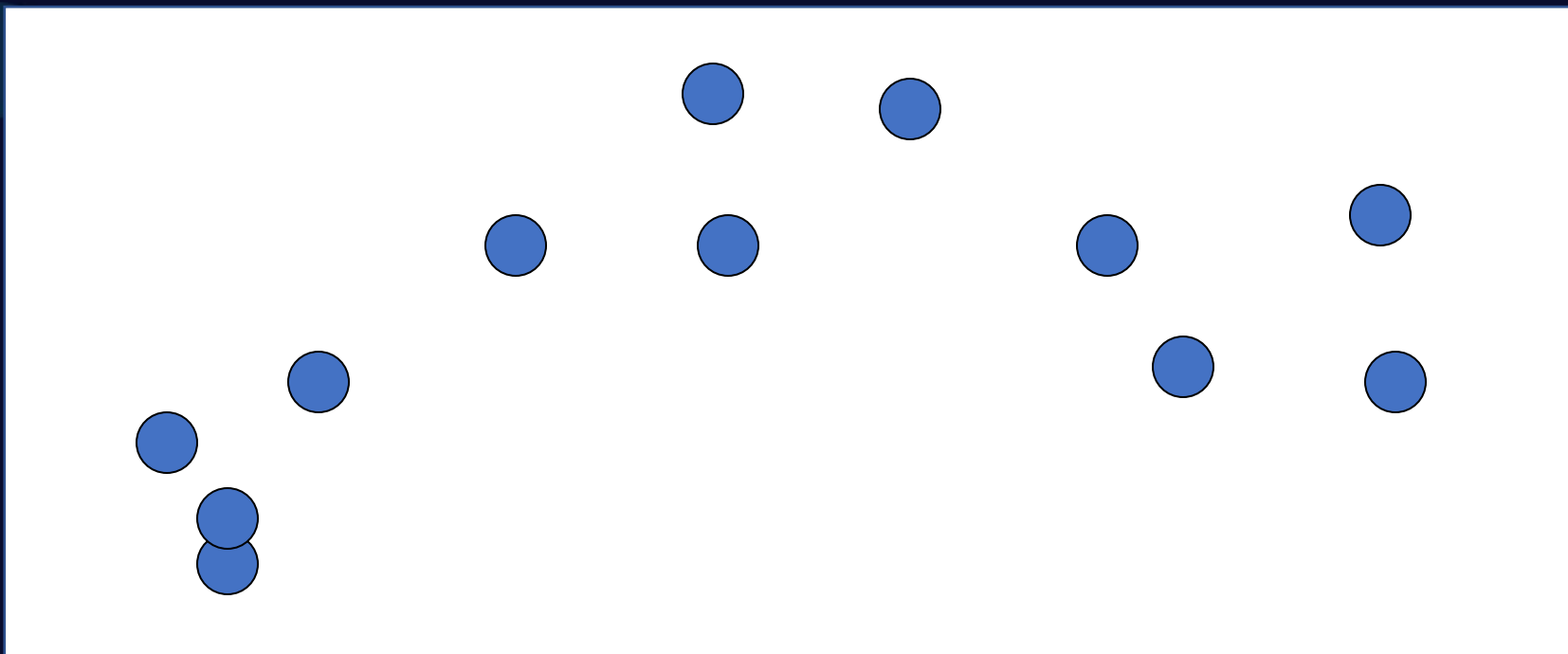
Most well-known and popular clustering algorithm:

Start with some initial cluster centers

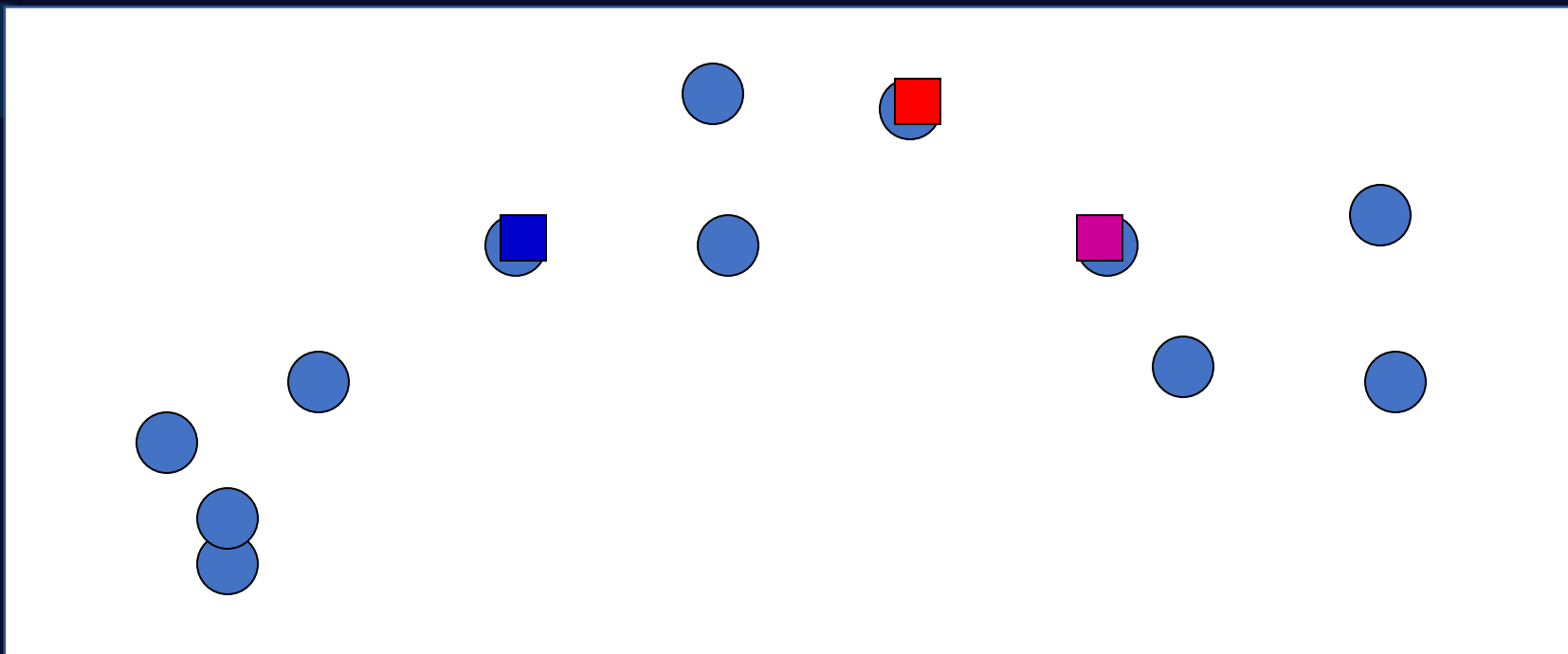
Iterate:

- Assign/cluster each example to closest center
- Recalculate centers as the mean of the points in a cluster

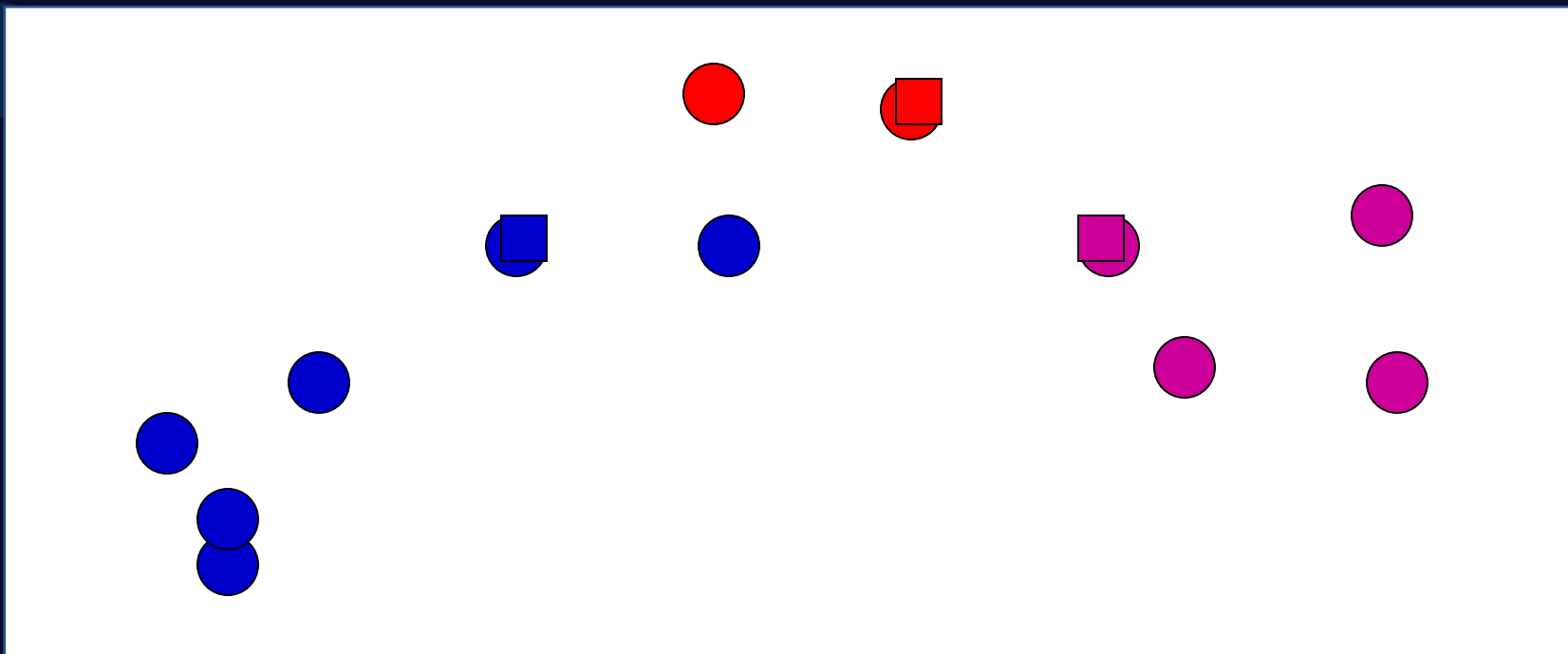
# K-means: an example



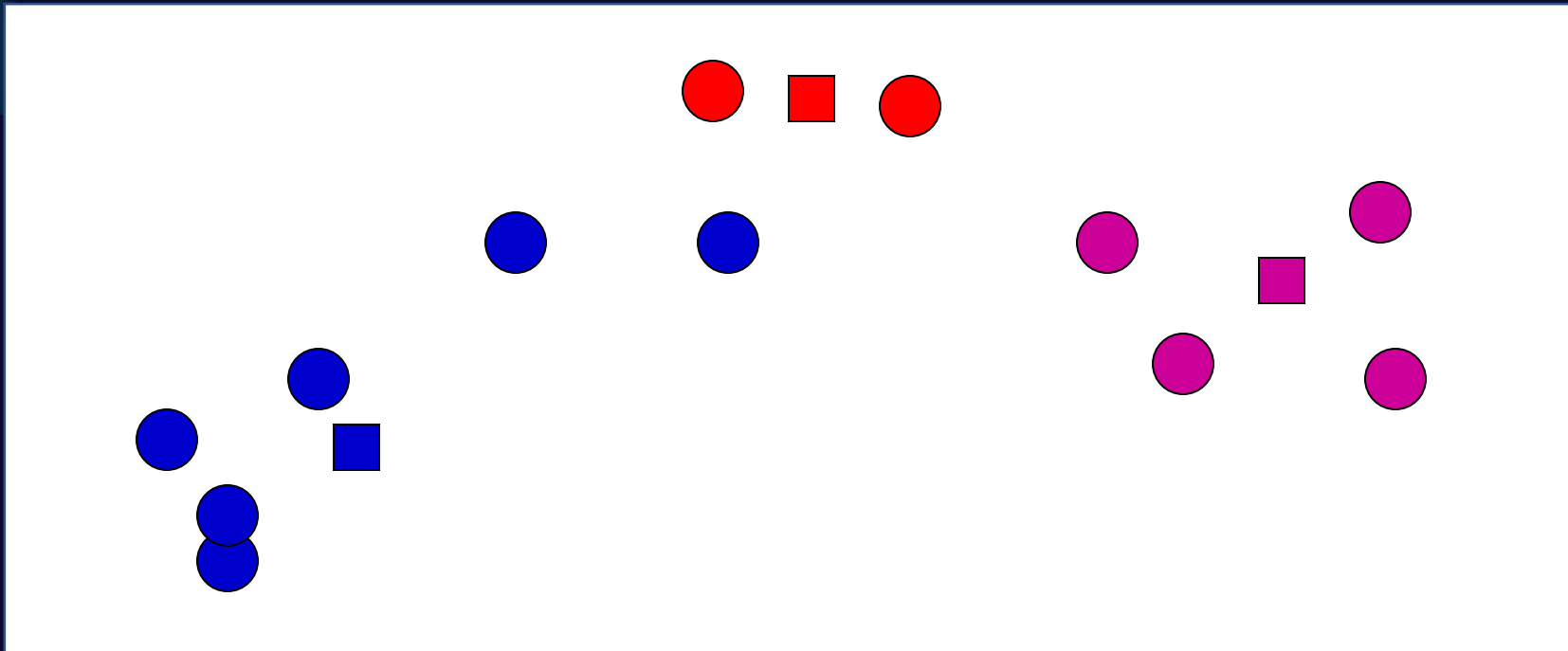
# K-means: Initialize centers randomly



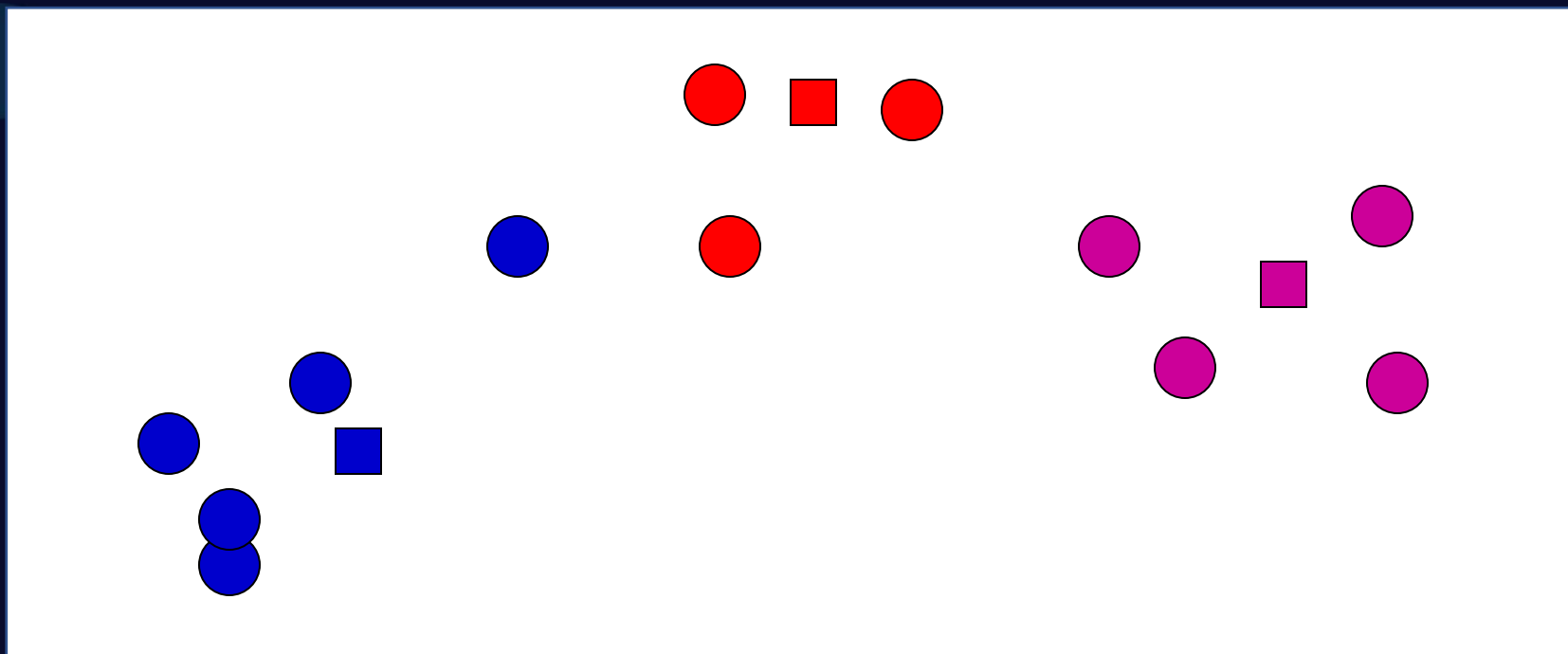
# K-means: assign points to nearest center



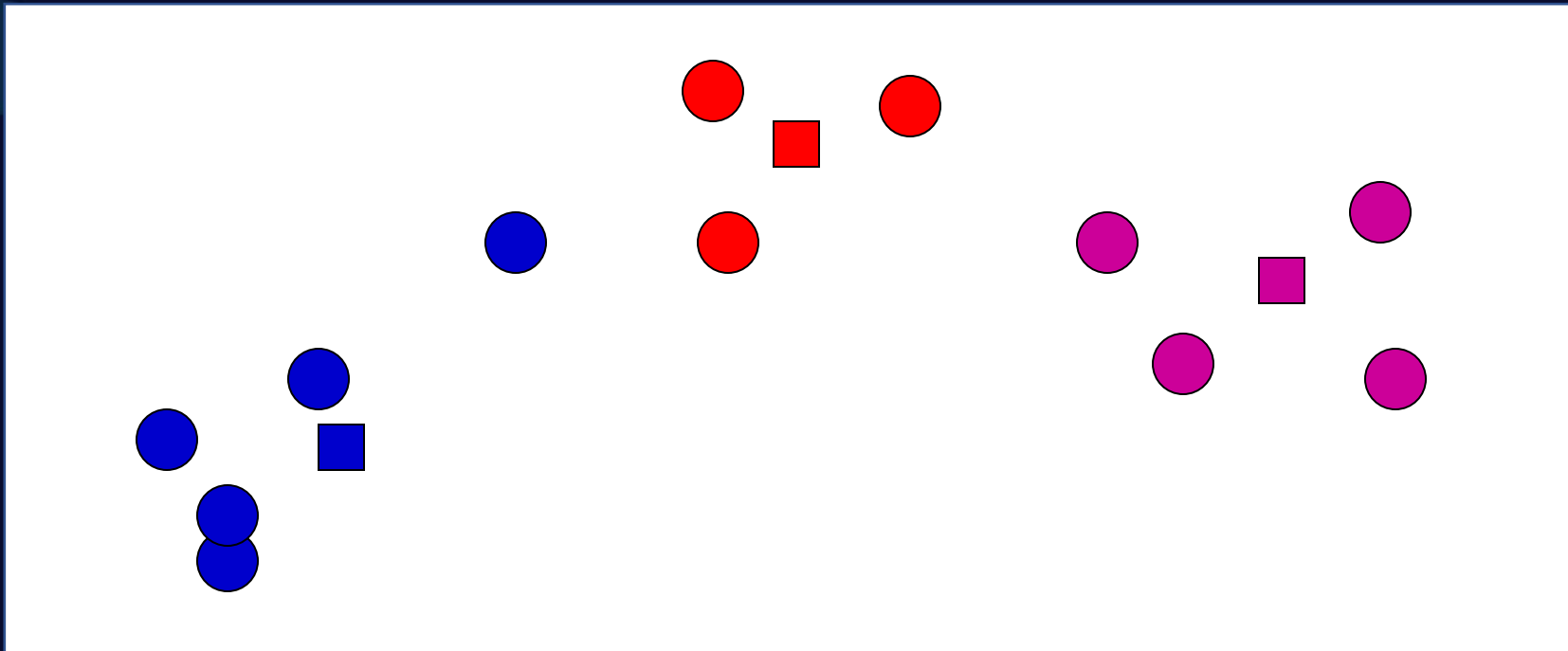
# K-means: readjust centers



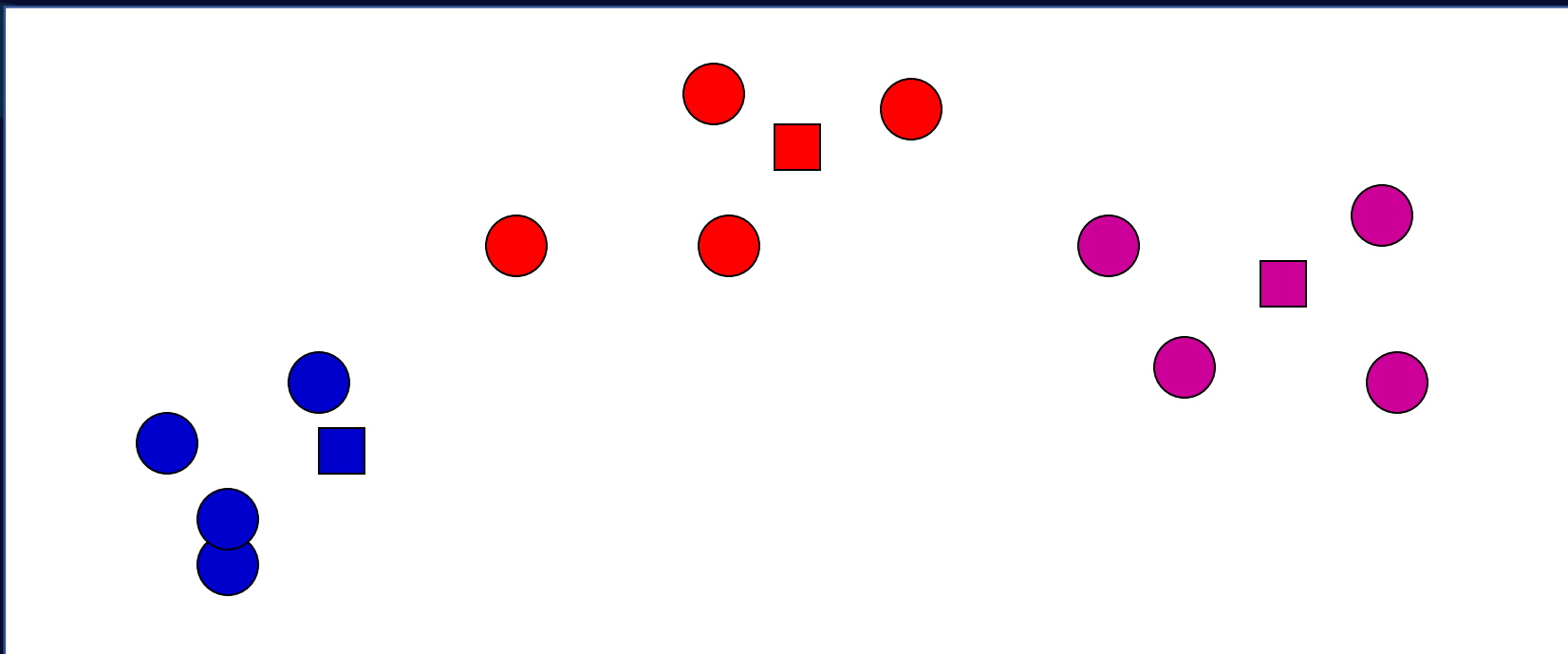
# K-means: assign points to nearest center



# K-means: readjust centers

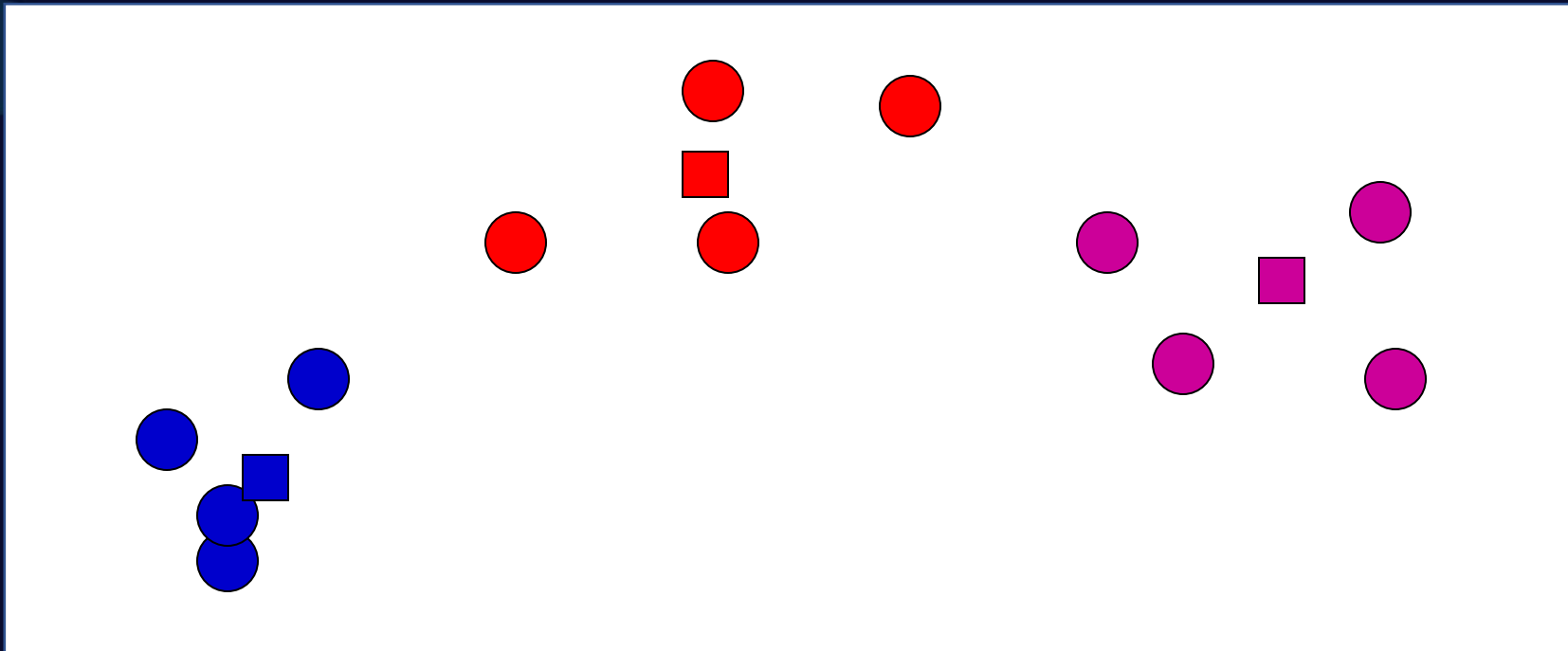


# K-means: assign points to nearest center

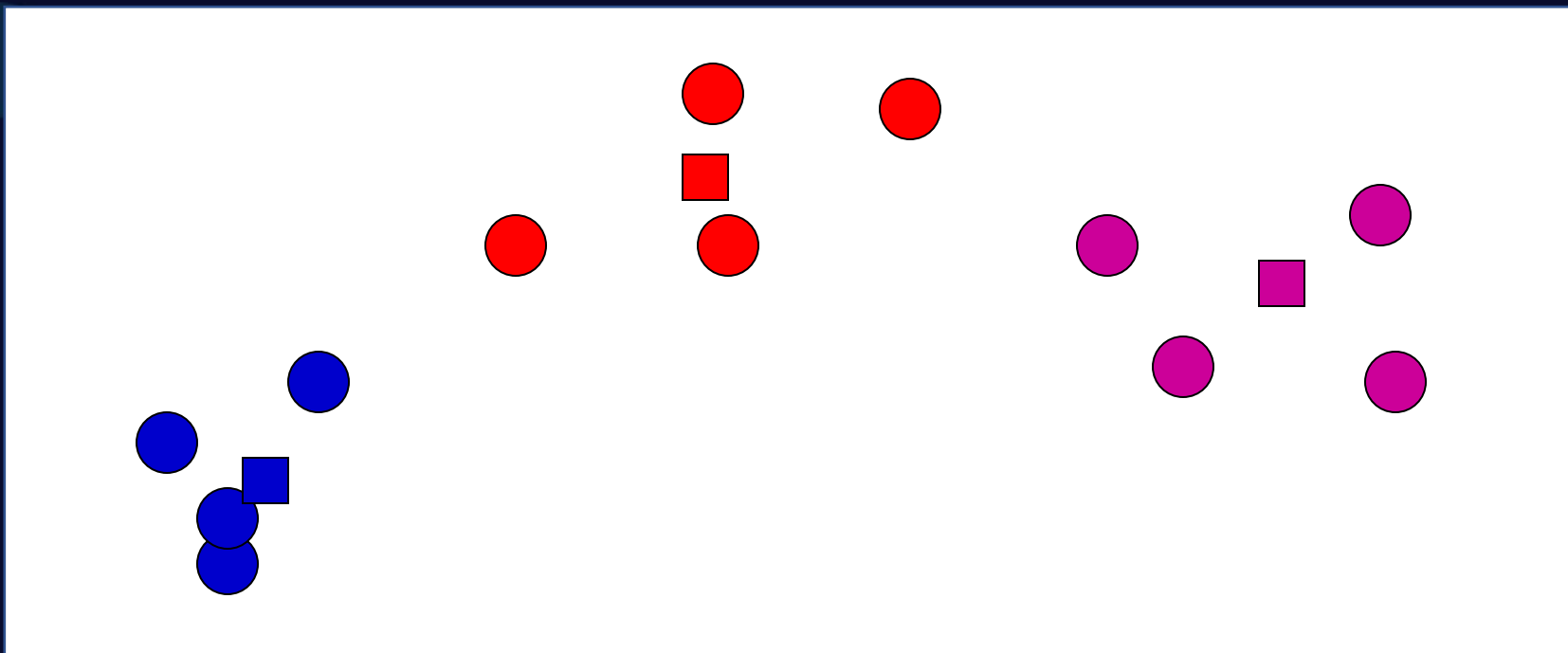




# K-means: readjust centers



# K-means: assign points to nearest center

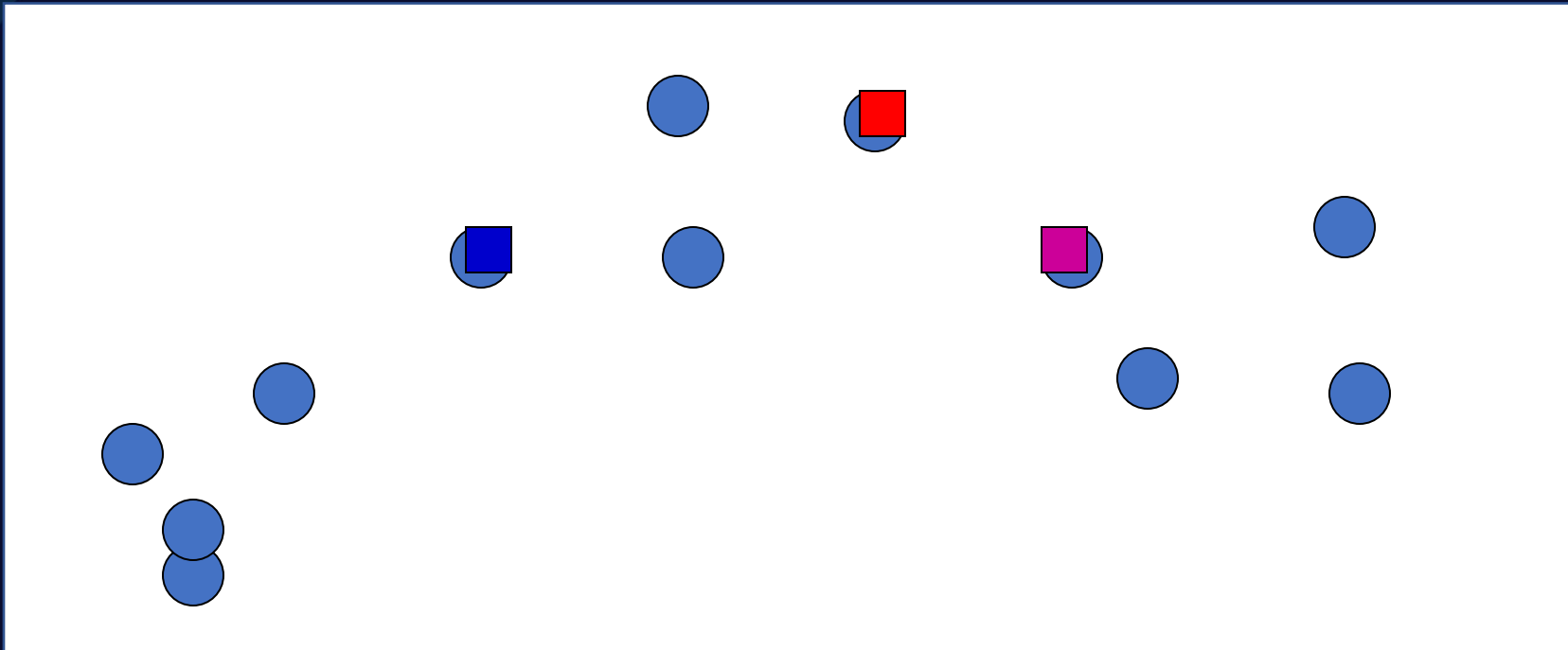


No changes: Done

# K-means

Iterate:

- **Assign/cluster each example to closest center**
- Recalculate centers as the mean of the points in a cluster



How do we do this?

# K-means

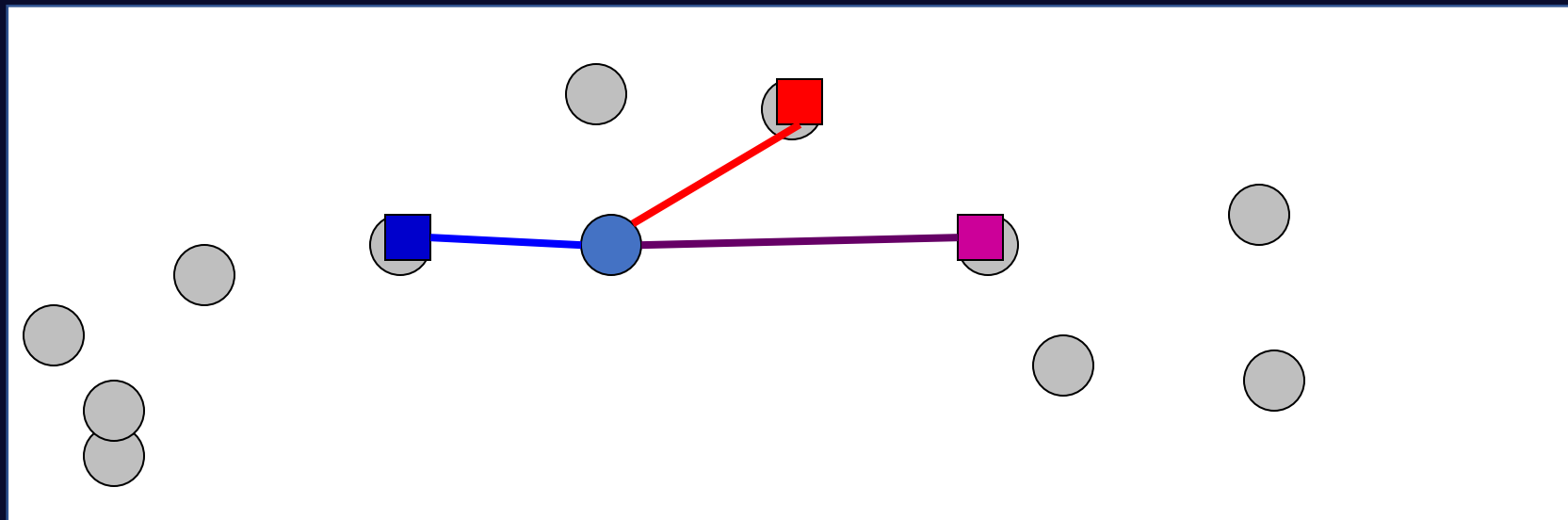
Iterate:

- **Assign/cluster each example to closest center**

iterate over each point:

- get distance to each cluster center
- assign to closest center (hard cluster)

- Recalculate centers as the mean of the points in a cluster



# K-means

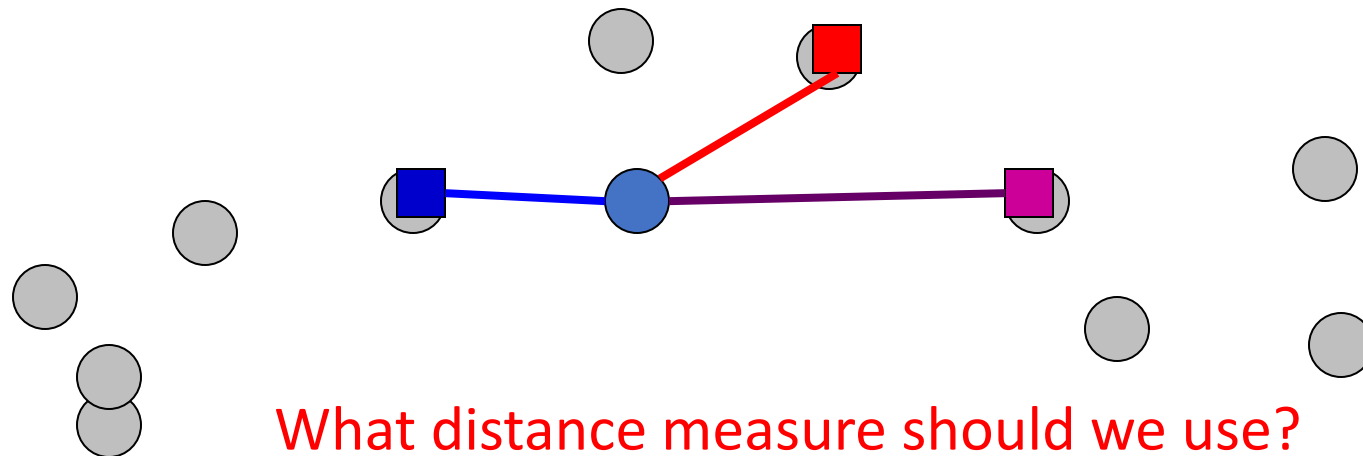
Iterate:

- **Assign/cluster each example to closest center**

iterate over each point:

- get **distance** to each cluster center
- assign to closest center (hard cluster)

- Recalculate centers as the mean of the points in a cluster



# Distance measures

Euclidean:

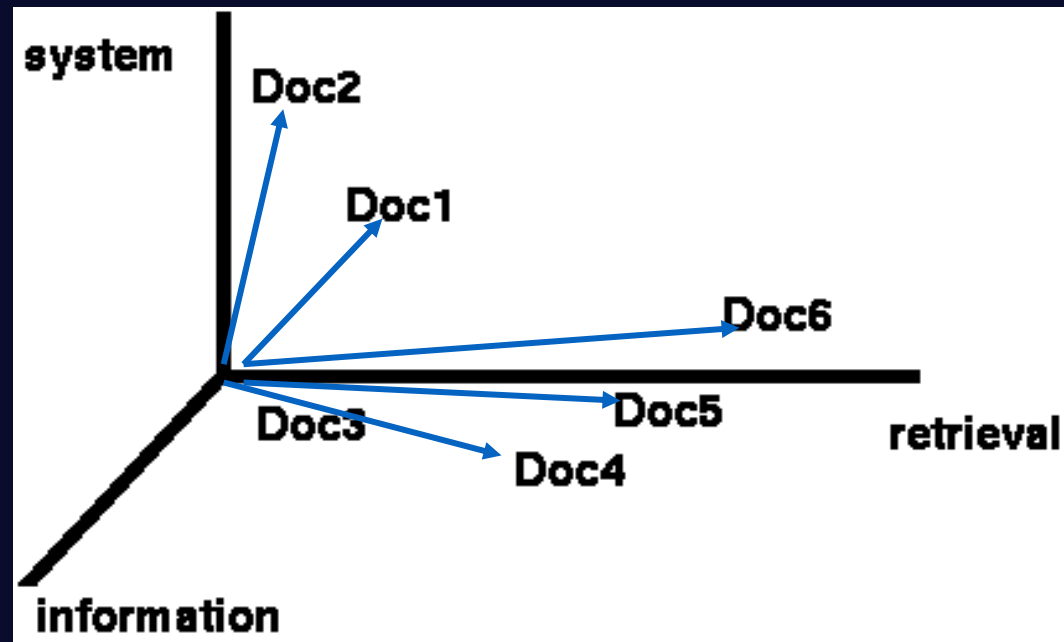
$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

good for spatial data

# Clustering documents

One feature for each word. The value is the number of times that word occurs.

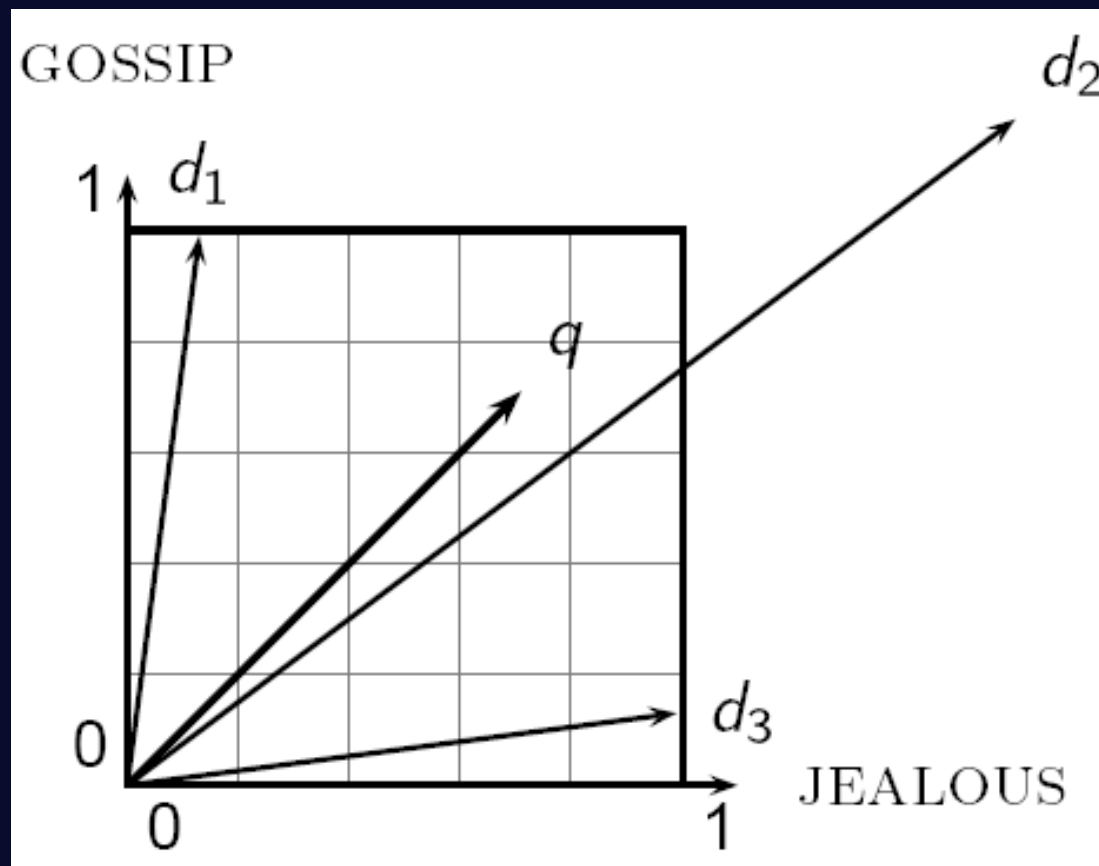
Documents are points or vectors in this space



# When Euclidean distance doesn't work

Which document is closest to  $q$  using Euclidean distance?

Which do you think should be closer?



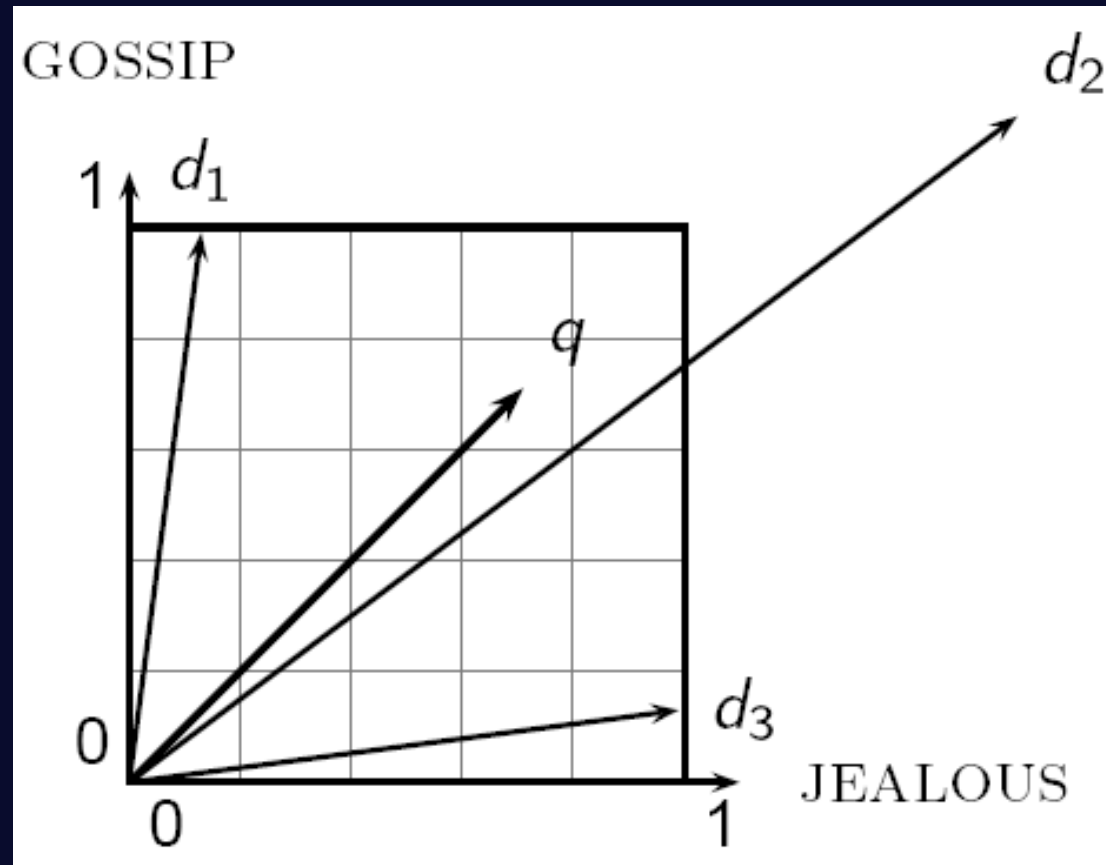


# Issues with Euclidian distance

the Euclidean distance between  $q$  and  $d_2$  is large

but, the distribution of terms in the query  $q$  and the distribution of terms in the document  $d_2$  are very similar

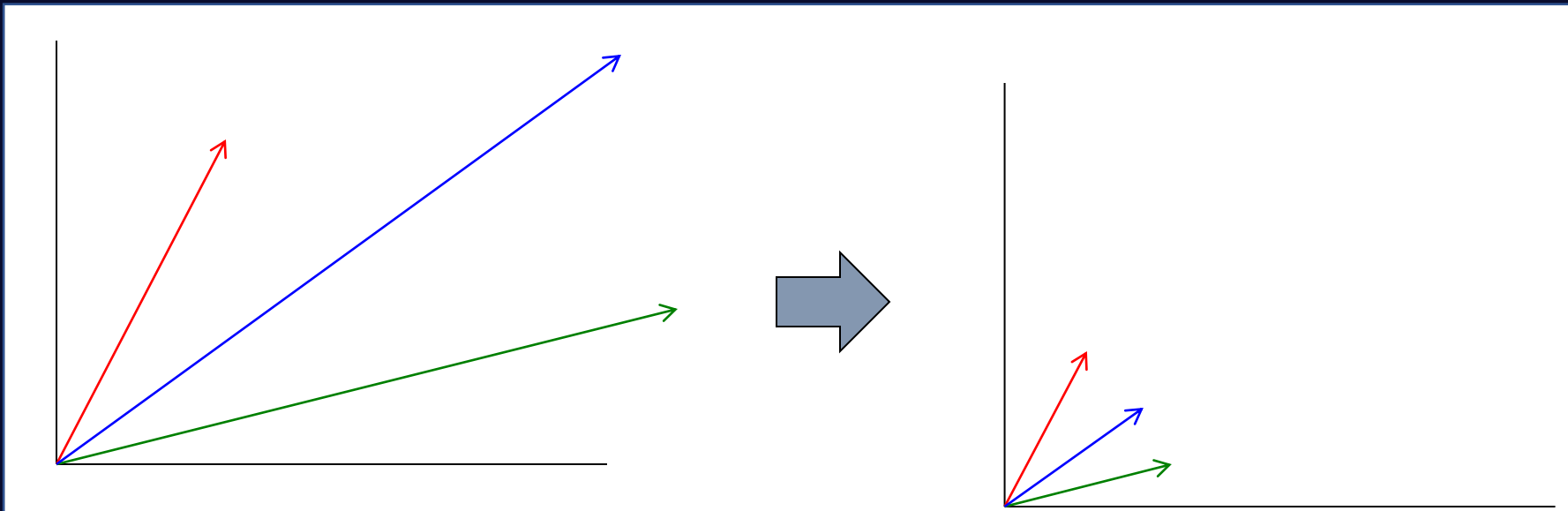
This is not what we want!



# cosine similarity

$$\text{sim}(x, y) = \frac{x \cdot y}{|x||y|} = \frac{x}{|x|} \cdot \frac{y}{|y|} = \frac{\sum_{i=1}^n x_i y_i}{\sqrt{\sum_{i=1}^n x_i^2} \sqrt{\sum_{i=1}^n y_i^2}}$$

correlated with the angle between two vectors



# cosine distance

cosine similarity is a similarity between 0 and 1, with things that are similar 1 and not 0

We want a distance measure, cosine distance:

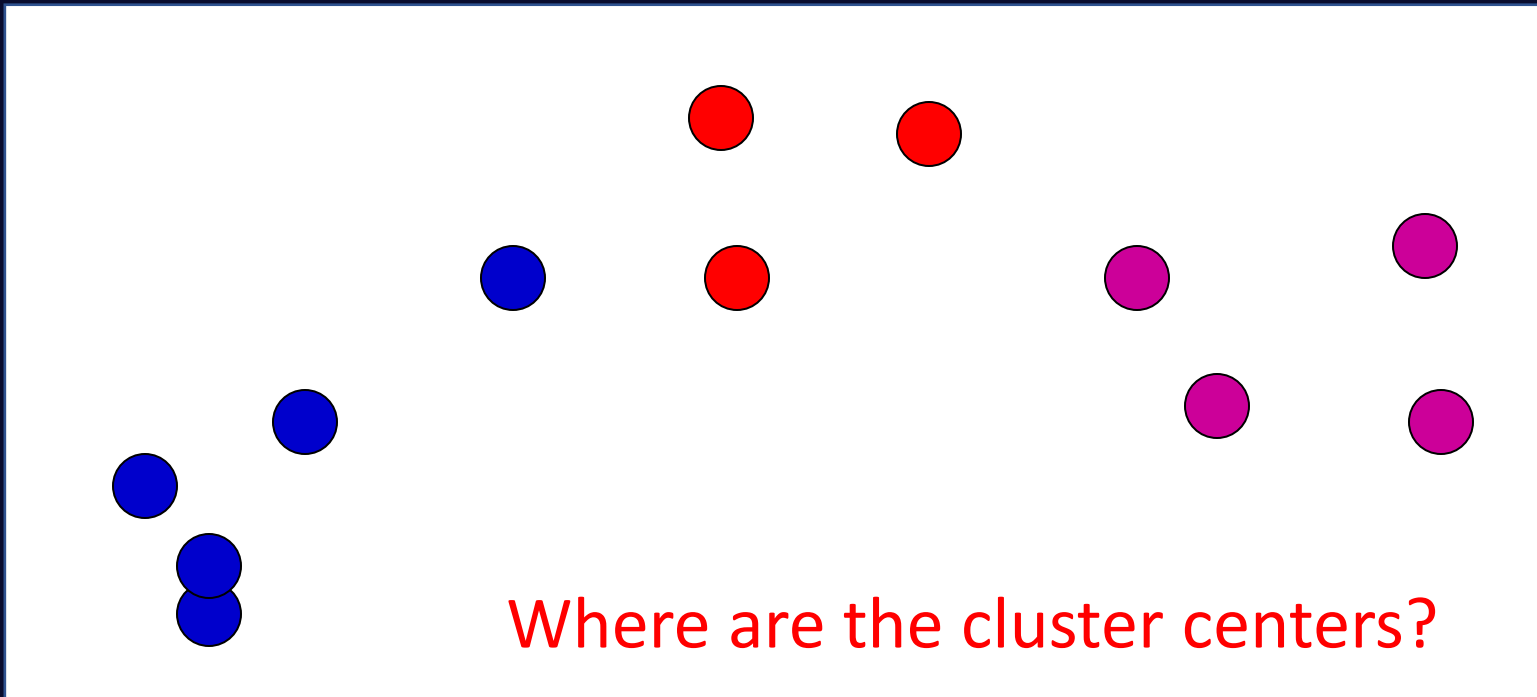
$$d(x, y) = 1 - \text{sim}(x, y)$$

- good for text data and many other “real world” data sets
- is computationally friendly since we only need to consider features that have non-zero values **both** examples

# K-means

Iterate:

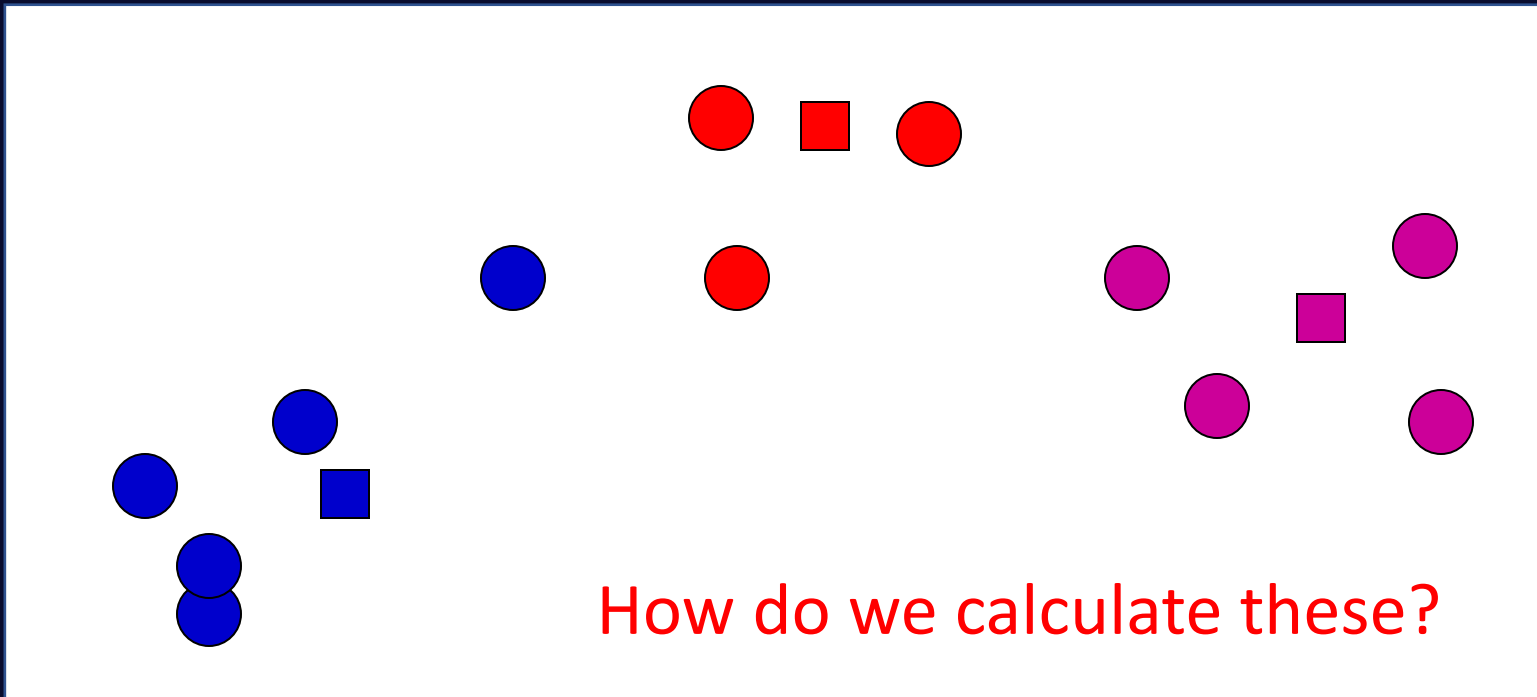
- Assign/cluster each example to closest center
- Recalculate centers as the mean of the points in a cluster



# K-means

Iterate:

- Assign/cluster each example to closest center
- Recalculate centers as the mean of the points in a cluster

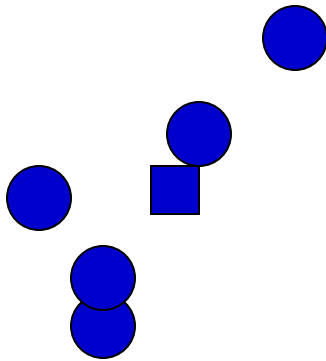


# K-means

Iterate:

- Assign/cluster each example to closest center
- Recalculate centers as the mean of the points in a cluster

Mean of the points in the cluster:



$$m(C) = \frac{1}{|C|} \sum_{x \in C} x$$

where:

$$x + y = \sum_{i=1}^n x_i + y_i \quad \frac{x}{|C|} = \sum_{i=1}^n \frac{x_i}{|C|}$$

# K-means loss function

K-means tries to minimize what is called the “k-means” loss function:

$$loss = \sum_{i=1}^n d(x_i, m_k)^2 \quad \text{where } m_k \text{ is cluster center for } x_i$$

that is, the sum of the squared distances from each point to the associated cluster center

# Minimizing k-means loss

Iterate:

1. Assign/cluster each example to closest center
  2. Recalculate centers as the mean of the points in a cluster
- 

$$loss = \sum_{i=1}^n d(x_i, m_k)^2 \quad \text{where } m_k \text{ is cluster center for } x_i$$

Does each step of k-means move towards reducing this loss function (or at least not increasing)?



# Minimizing k-means loss

Iterate:

1. Assign/cluster each example to closest center
  2. Recalculate centers as the mean of the points in a cluster
- 

$$loss = \sum_{i=1}^n d(x_i, m_k)^2 \quad \text{where } m_k \text{ is cluster center for } x_i$$

This isn't quite a complete proof/argument, but:

1. Any other assignment would end up in a larger loss
1. The mean of a set of values minimizes the squared error

# Minimizing k-means loss

Iterate:

1. Assign/cluster each example to closest center
  2. Recalculate centers as the mean of the points in a cluster
- 

$$loss = \sum_{i=1}^n d(x_i, m_k)^2 \quad \text{where } m_k \text{ is cluster center for } x_i$$

Does this mean that k-means will always find the minimum loss/clustering?

# Minimizing k-means loss

Iterate:

1. Assign/cluster each example to closest center
  2. Recalculate centers as the mean of the points in a cluster
- 

$$loss = \sum_{i=1}^n d(x_i, m_k)^2 \quad \text{where } m_k \text{ is cluster center for } x_i$$

NO! It will find *a minimum*.

Unfortunately, the k-means loss function is generally not convex and for most problems has many, many minima

We're only guaranteed to find one of them

# K-means variations/parameters

Start with some initial cluster centers

Iterate:

- Assign/cluster each example to closest center
- Recalculate centers as the mean of the points in a cluster

What are some other  
variations/parameters we haven't  
specified?

# K-means variations/parameters

Initial (seed) cluster centers

Convergence

- A fixed number of iterations
- partitions unchanged
- Cluster centers don't change

# Seed choice

Results can vary drastically based on random seed selection

Some seeds can result in poor convergence rate, or convergence to sub-optimal clusterings

## Common heuristics

- Random centers in the space
- Randomly pick examples
- Points least similar to any existing center (furthest centers heuristic)
- **Try out multiple starting points**
- Initialize with the results of another clustering method

# Furthest centers heuristic

$\mu_1$  = pick random point

for  $i = 2$  to  $K$ :

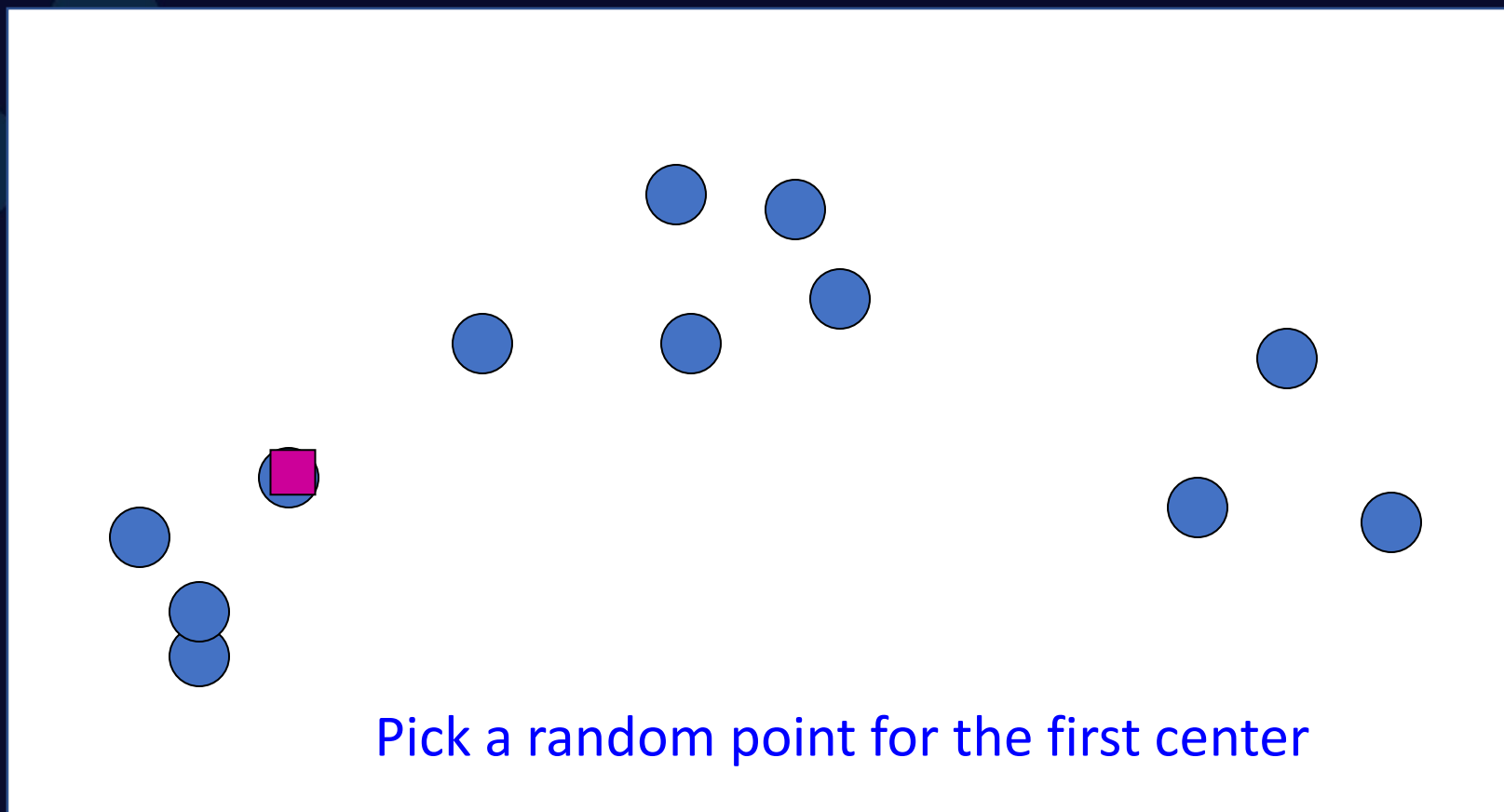
$\mu_i$  = point that is furthest from **any** previous centers

$$m_i = \underbrace{\arg \max_x}_{\text{point with the largest distance to any previous center}} \underbrace{\min_{m_j : 1 < j < i} d(x, m_j)}_{\text{smallest distance from } x \text{ to any previous center}}$$

point with the largest distance  
to any previous center

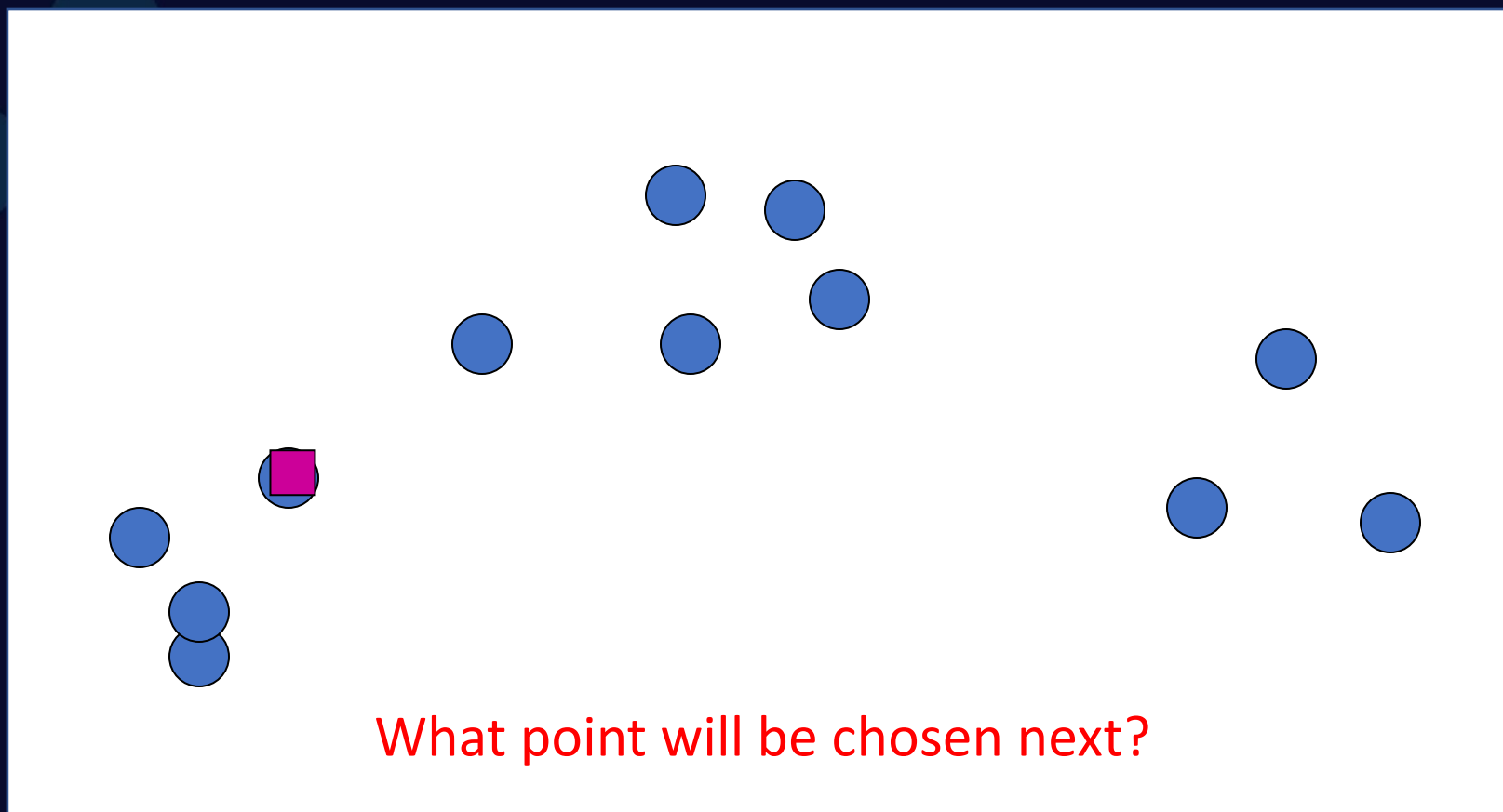
smallest distance from  $x$  to  
any previous center

## K-means: Initialize furthest from centers

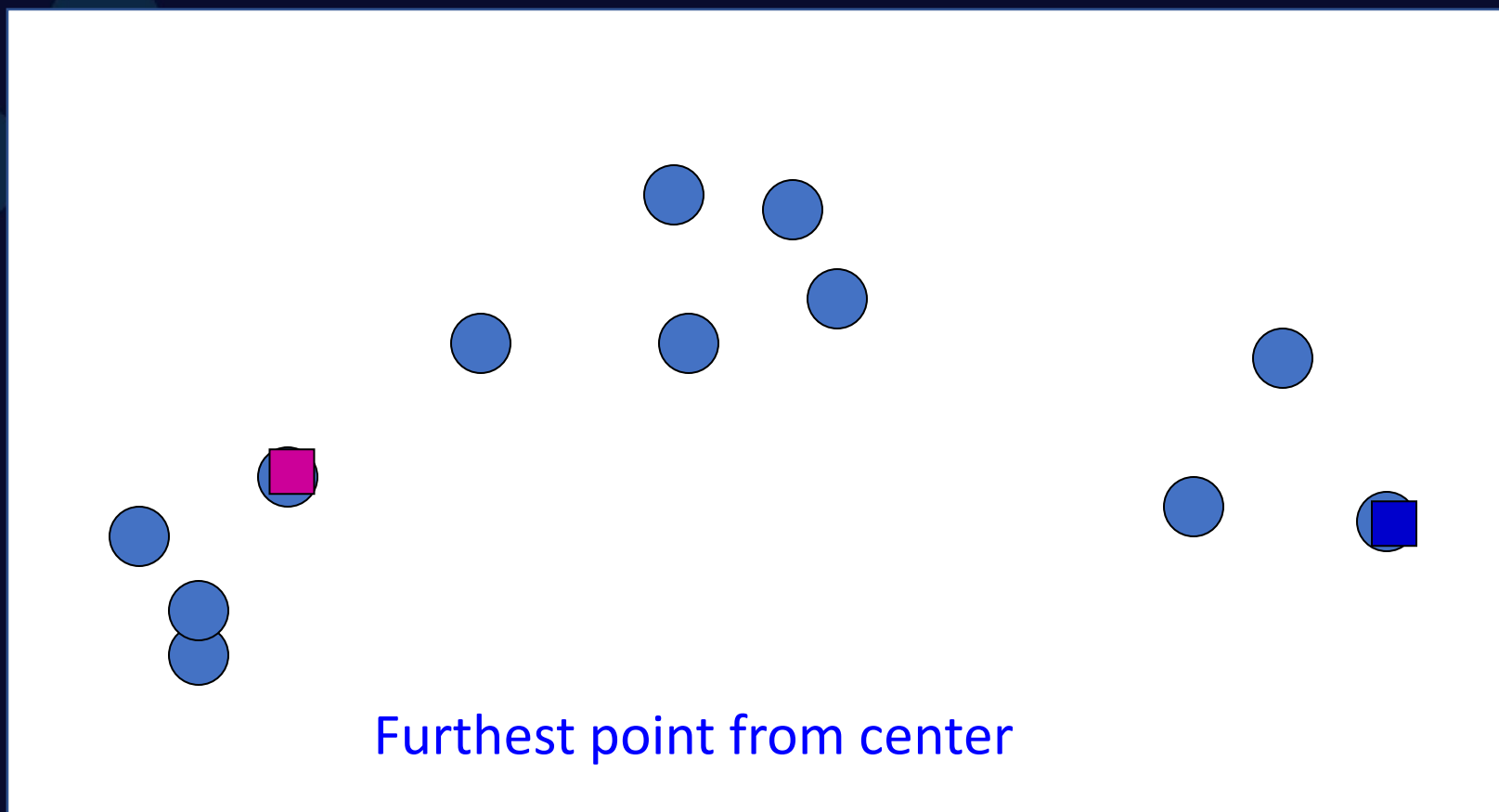




## K-means: Initialize furthest from centers

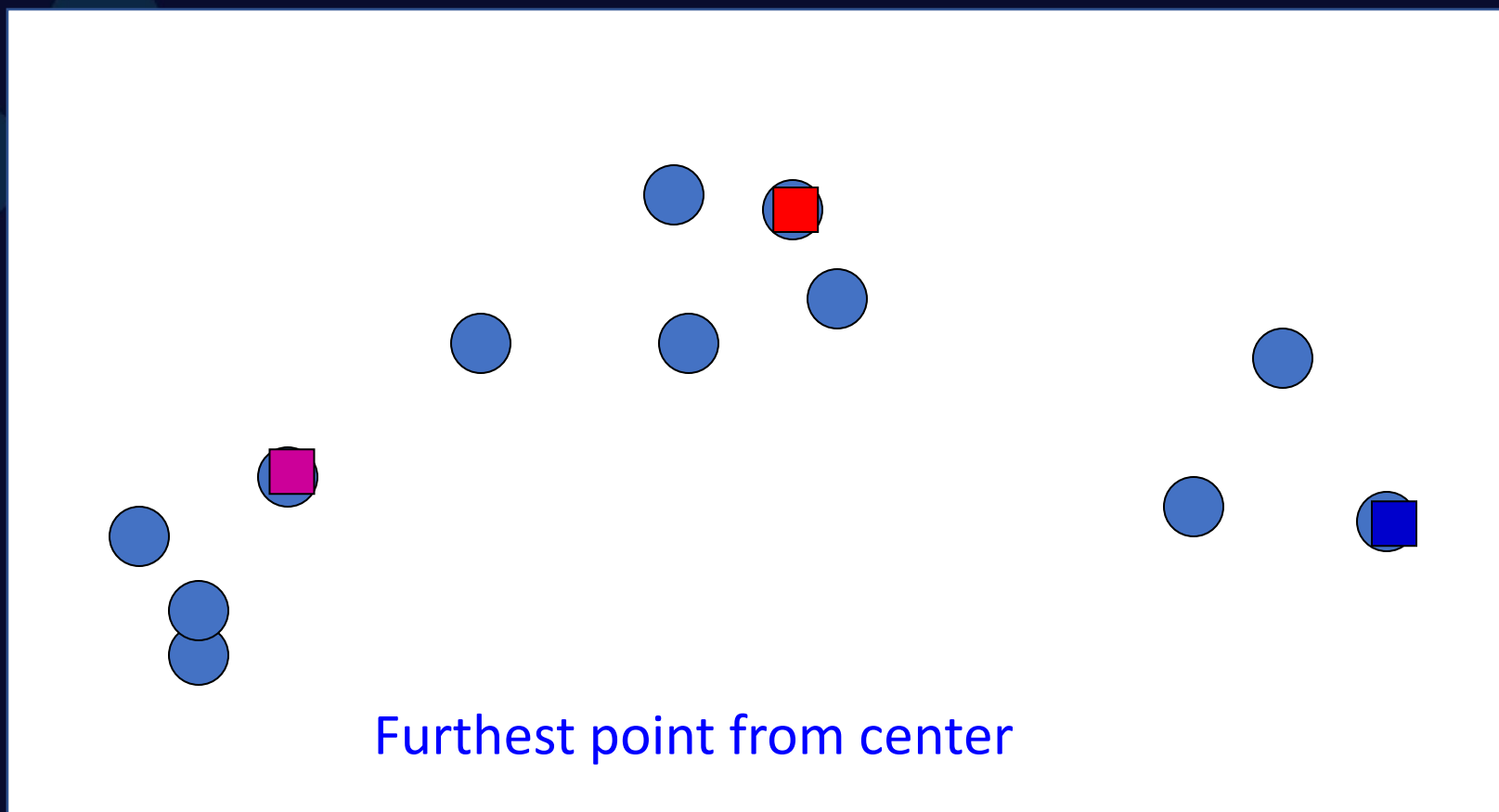


## K-means: Initialize furthest from centers



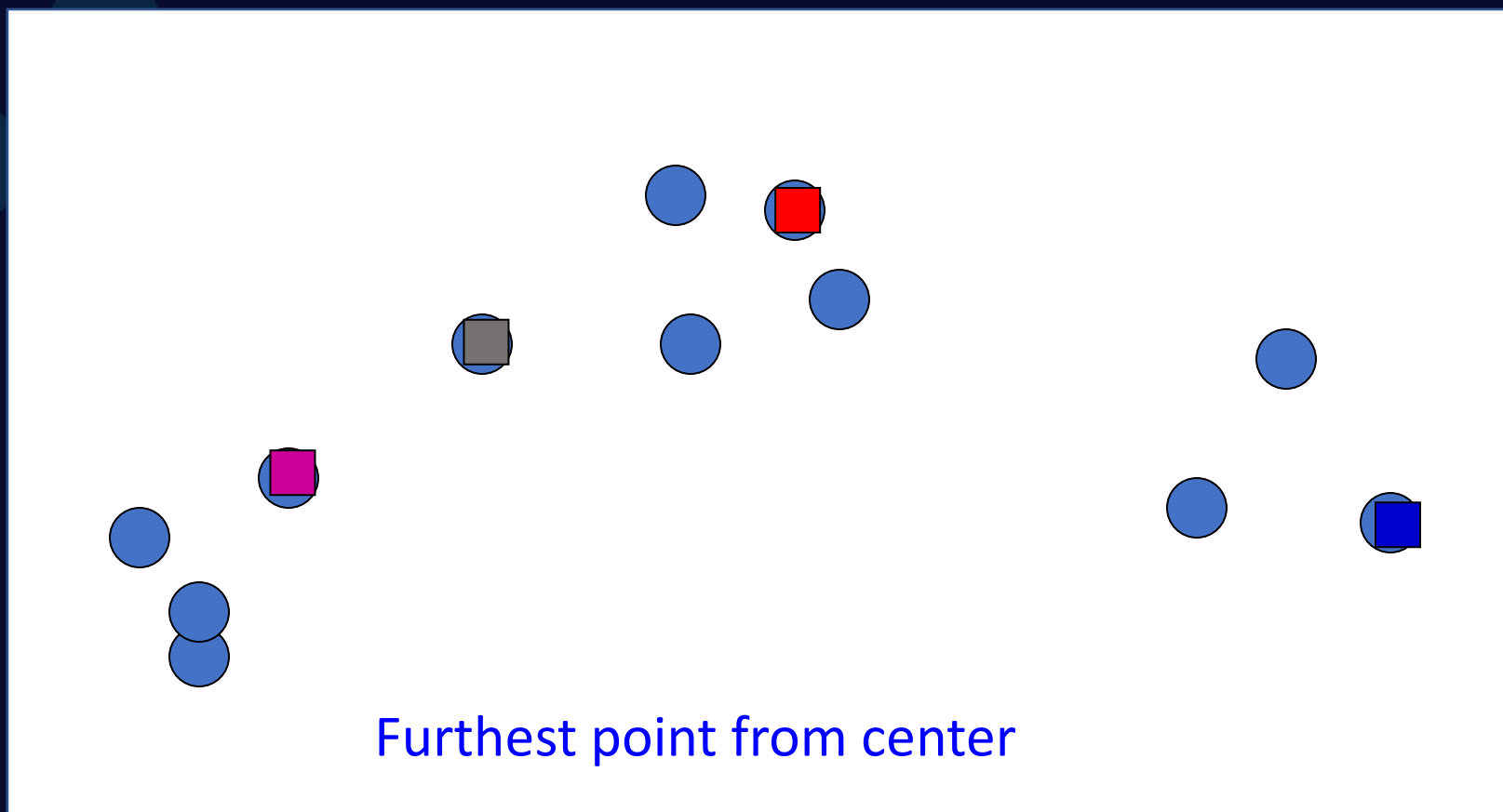
What point will be chosen next?

## K-means: Initialize furthest from centers



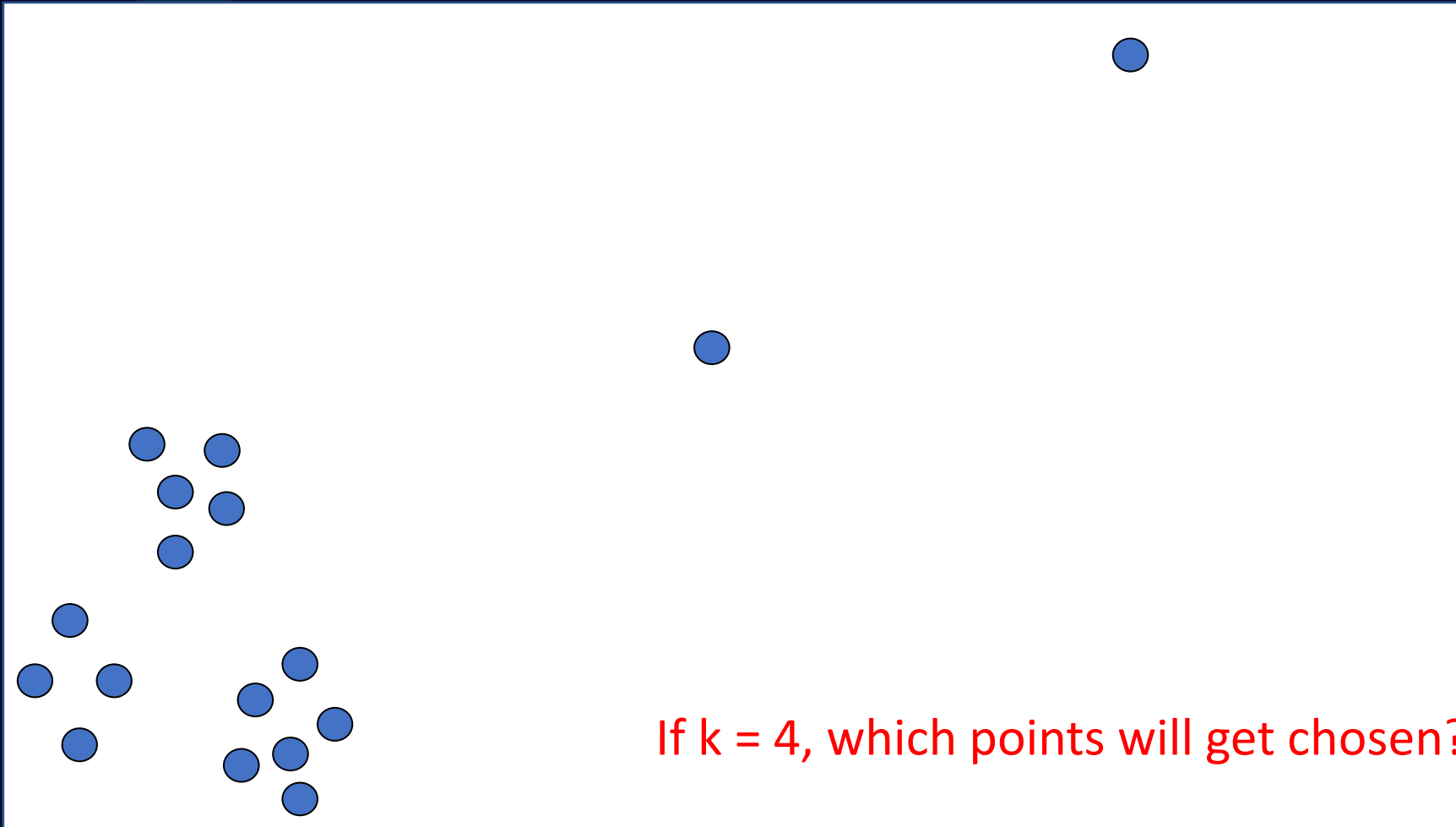
What point will be chosen next?

## K-means: Initialize furthest from centers

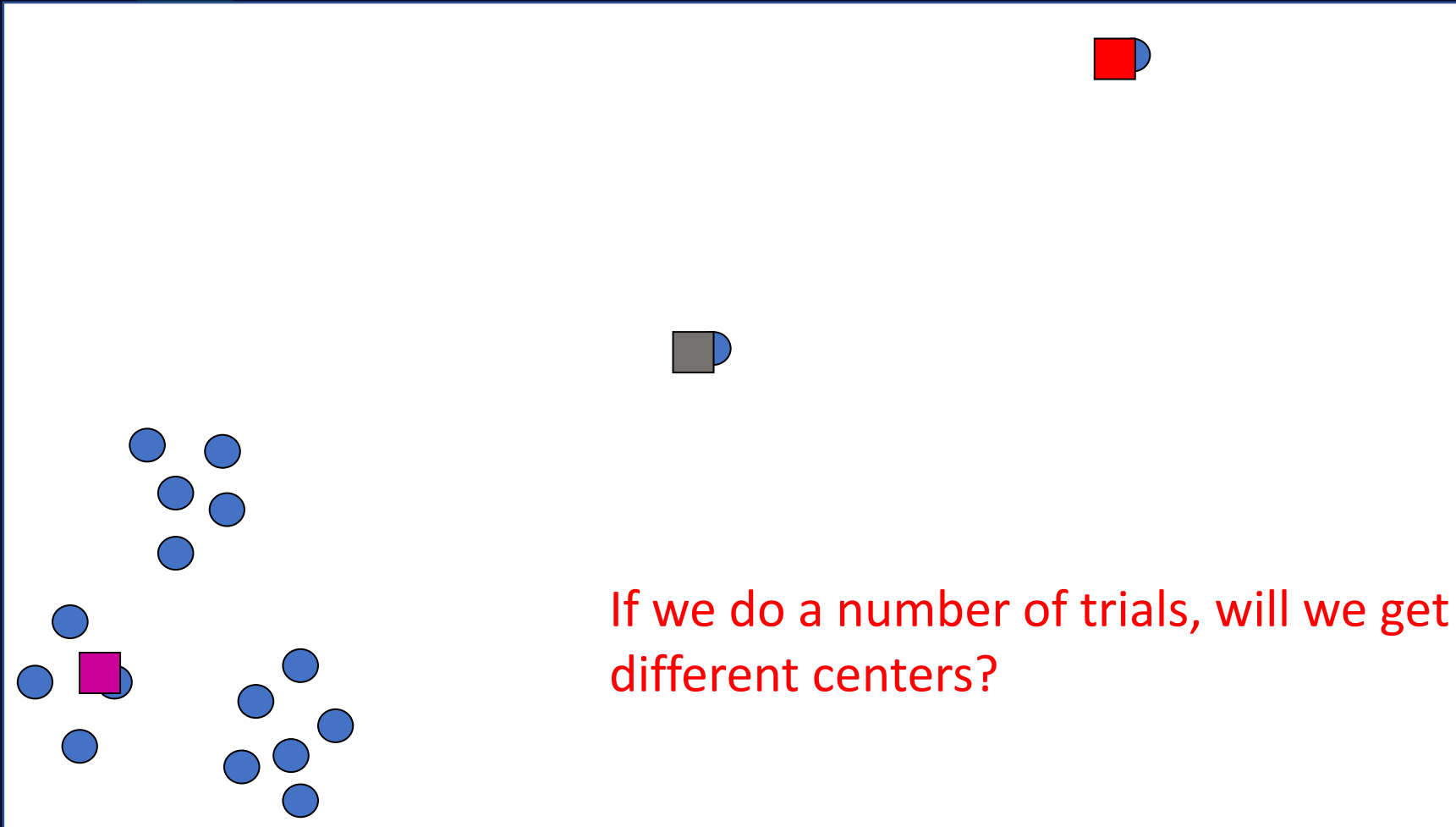


Any issues/concerns with this approach?

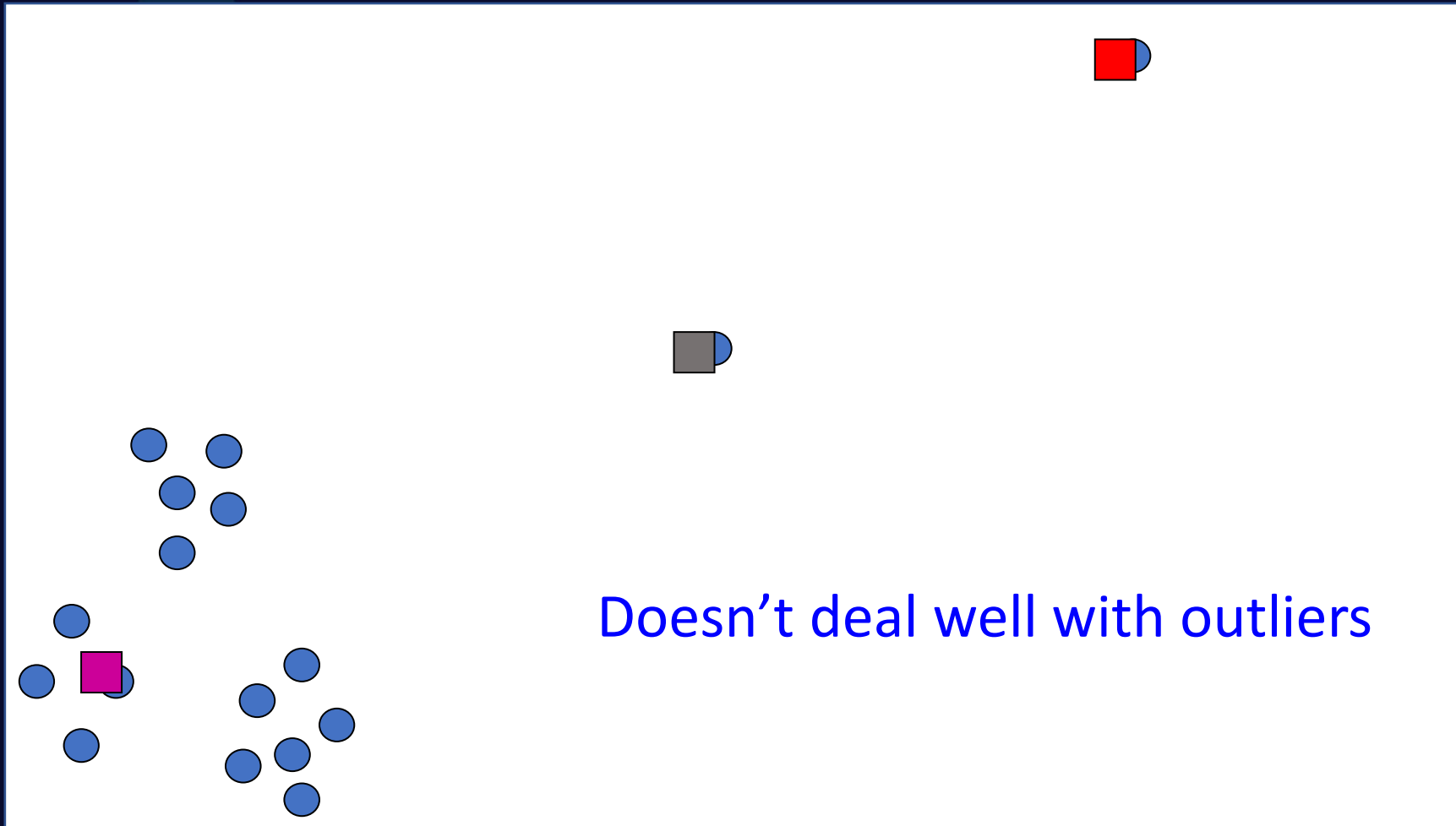
# Furthest points concerns



# Furthest points concerns



# Furthest points concerns



# K-means++

$\mu_1$  = pick random point

for  $k = 2$  to  $K$ :

for  $i = 1$  to  $N$ :

$s_i = \min d(x_i, \mu_{1\dots k-1})$  // smallest distance to any center

$\mu_k$  = randomly pick point *proportionate* to  $s$

How does this help?



# K-means++

$\mu_1$  = pick random point

for  $k = 2$  to  $K$ :

for  $i = 1$  to  $N$ :

$s_i = \min d(x_i, \mu_{1\dots k-1})$  // smallest distance to any center

$\mu_k$  = randomly pick point *proportionate* to  $s$

- Makes it possible to select other points
  - if  $\#points \gg \#outliers$ , we will pick good points
- Makes it non-deterministic, which will help with random runs
- Nice theoretical guarantees!

# K-Medoids

The k-Means algorithm is sensitive to outliers because an object with an “extremely large value” may substantially distort the distribution. The effect is particularly exacerbated due to the use of the SSE (sum-of-squared error) objective function. The k-Medoids algorithm aims to diminish the effect of outliers.

# K-Medoids

This algorithm selects an object as a cluster center (one representative object per cluster) instead of taking the mean value of the objects in a cluster (as in k-Means algorithm).

We call this cluster representative as a cluster medoid or simply medoid.

Initially, it selects a random set of  $k$  objects as the set of medoids.

Then at each step, all objects from the set of objects, which are not currently medoids are examined one by one to see if they should be medoids.

# K-Medoids

- The k-Medoids algorithm **determines** whether there is an object that should replace one of the current medoids.
- This is accomplished by looking all pair of medoid, non-medoid objects, and then choosing a pair that improves the objective function of clustering the best and exchange them.
- The sum-of-absolute error (SAE) function is used as the objective function.

$$SAE = \sum_{i=1}^k \sum_{x \in C_i, x \notin M \text{ and } c_m \in M} |x - c_m|$$

Where  $c_m$  denotes a medoid

$M$  is the set of all medoids at any instant

$x$  is an object belongs to set of non-medoid object, that is,  $x$  belongs to some cluster and is not a medoid. i.e.  $x \in C_i, x \notin M$

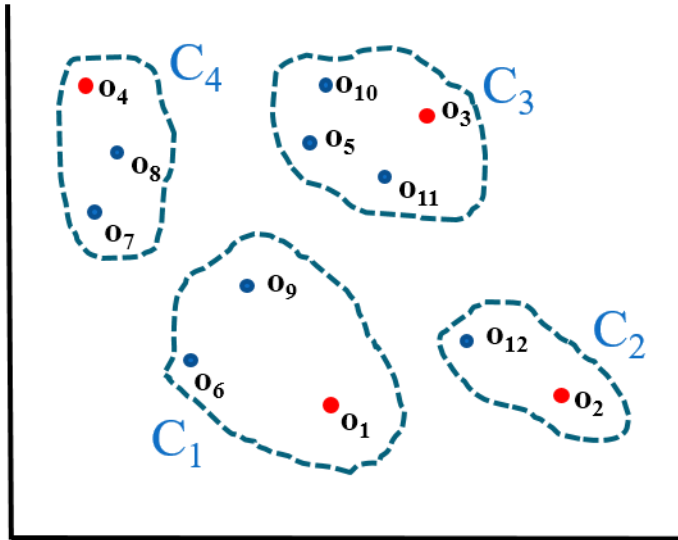
# K-Medoids

For a given set of medoids, at any iteration, it select that exchange which has minimum SAE.

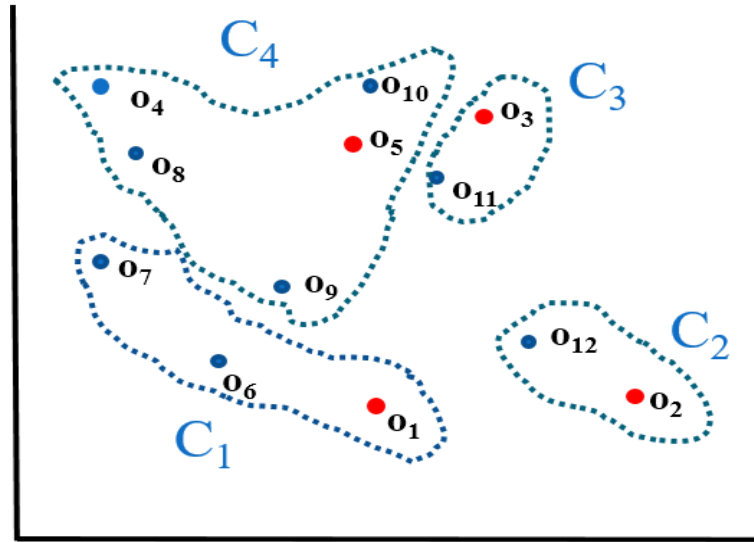
The procedure terminates, if there is no any change in SAE in successive iteration (i.e. there is no change in medoid).

This k-Medoids algorithm is also known as PAM (Partitioning around Medoids).

# PAM (Partitioning around Medoids)



(a) Cluster with  $o_1$ ,  $o_2$ ,  $o_3$ , and  $o_4$  as medoids



(b) Cluster after swapping  $o_4$  and  $o_5$  ( $o_5$  becomes the new medoid).

# PAM (Partitioning around Medoids)

## 1. Comparing k-Means with k-Medoids:

- Both algorithms need to fix  $k$ , the number of clusters prior to the algorithms. Also, both algorithms arbitrarily choose the initial cluster centroids.
- The k-Medoid method is more robust than k-Means in the presence of outliers, because a medoid is less influenced by outliers than a mean.

## 2. Time complexity of PAM:

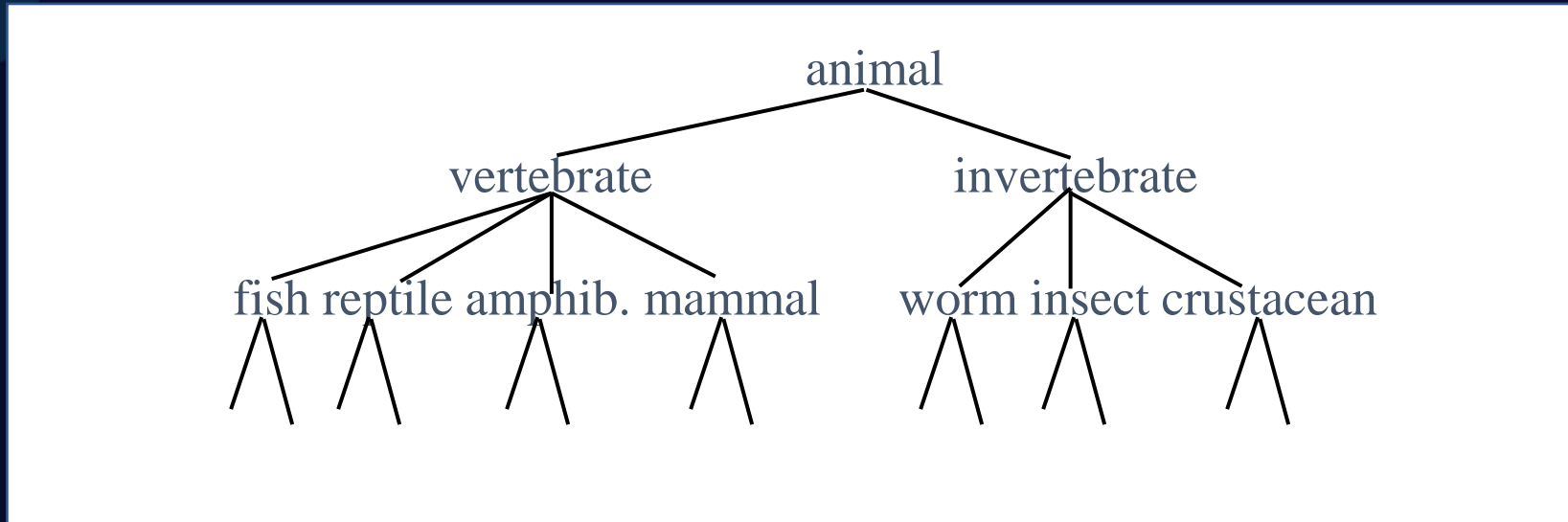
- For each iteration, PAM considers  $k(n - k)$  pairs of objects  $o_i, o_j$  for which a cost  $cost(o_i, o_j)$  is determined. Calculating the cost during each iteration requires that the cost be calculated for all other non-medoids  $o_j$ . There are  $n - k$  of these. Thus, the total time complexity per iteration is  $n(n - k)^2$ . The total number of iterations may be quite large.

## 3. Applicability of PAM:

- PAM does not scale well to large databases because of its computational complexity.

# Hierarchical Clustering

- Build a tree-based hierarchical taxonomy (*dendrogram*) from a set of documents.



How could you do this with k-means?



# Hierarchical Clustering algorithms

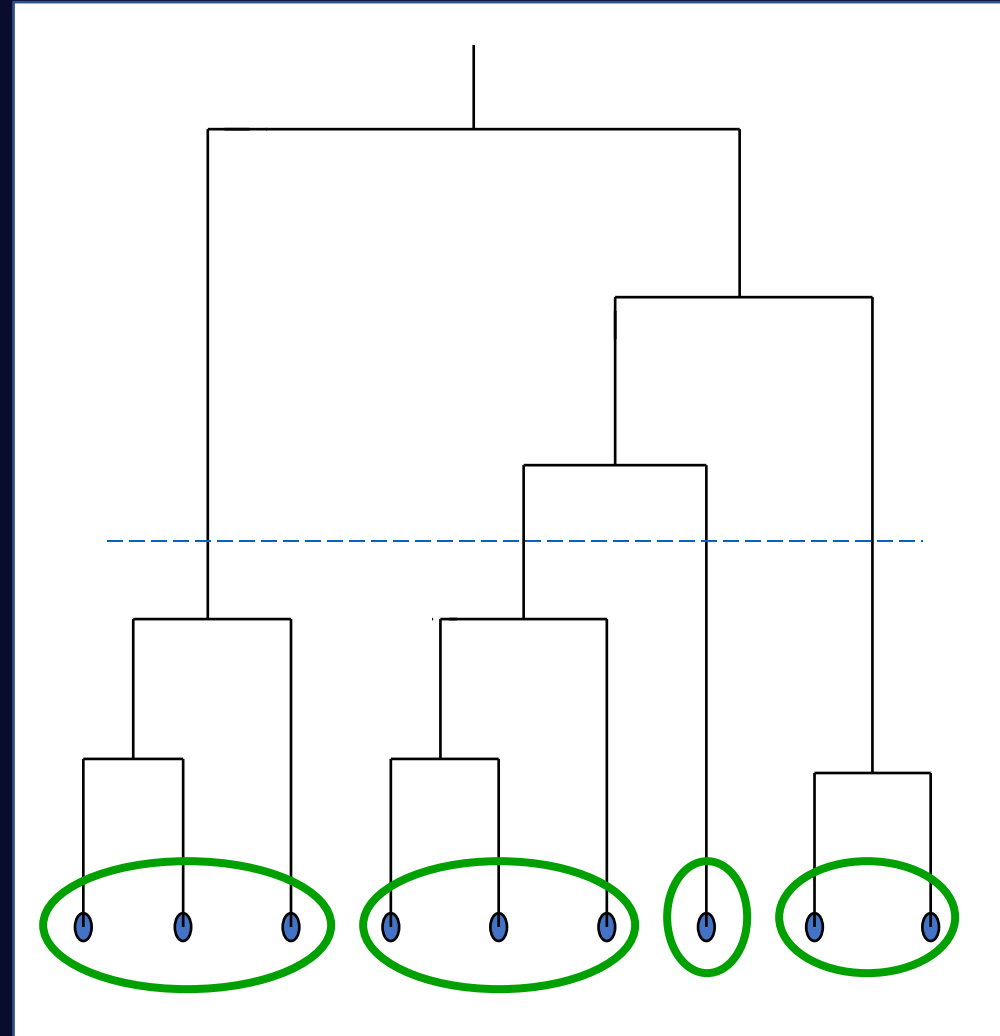
- **Agglomerative (bottom-up):**
  - Start with each document being a single cluster.
  - Eventually all documents belong to the same cluster.
- **Divisive (top-down):**
  - Start with all documents belong to the same cluster.
  - Eventually each node forms a cluster on its own.
  - Could be a recursive application of k-means like algorithms
- Does not require the number of clusters  $k$  in advance
- Needs a termination/readout condition

# Hierarchical Agglomerative Clustering (HAC)

- Assumes a similarity function for determining the similarity of two instances.
- Starts with all instances in a separate cluster and then repeatedly joins the two clusters that are most similar until there is only one cluster.
- The history of merging forms a binary tree or hierarchy.

# Dendrogram: Hierarchical Clustering

- Clustering obtained by cutting the dendrogram at a desired level: each connected component forms a cluster.



# Hierarchical Agglomerative Clustering (HAC)

- Starts with each doc in a separate cluster
  - then repeatedly joins the closest pair of clusters, until there is only one cluster.
- The history of merging forms a binary tree or hierarchy.

How to measure distance of clusters??

# *Closest pair* of clusters

Many variants to defining closest pair of clusters

- **Single-link**

- Distance of the “*closest*” points (single-link)

- **Complete-link**

- Distance of the “furthest” points

- **Centroid**

- Distance of the centroids (centers of gravity)

- **(Average-link)**

- Average distance between pairs of elements

# Single Link Agglomerative Clustering

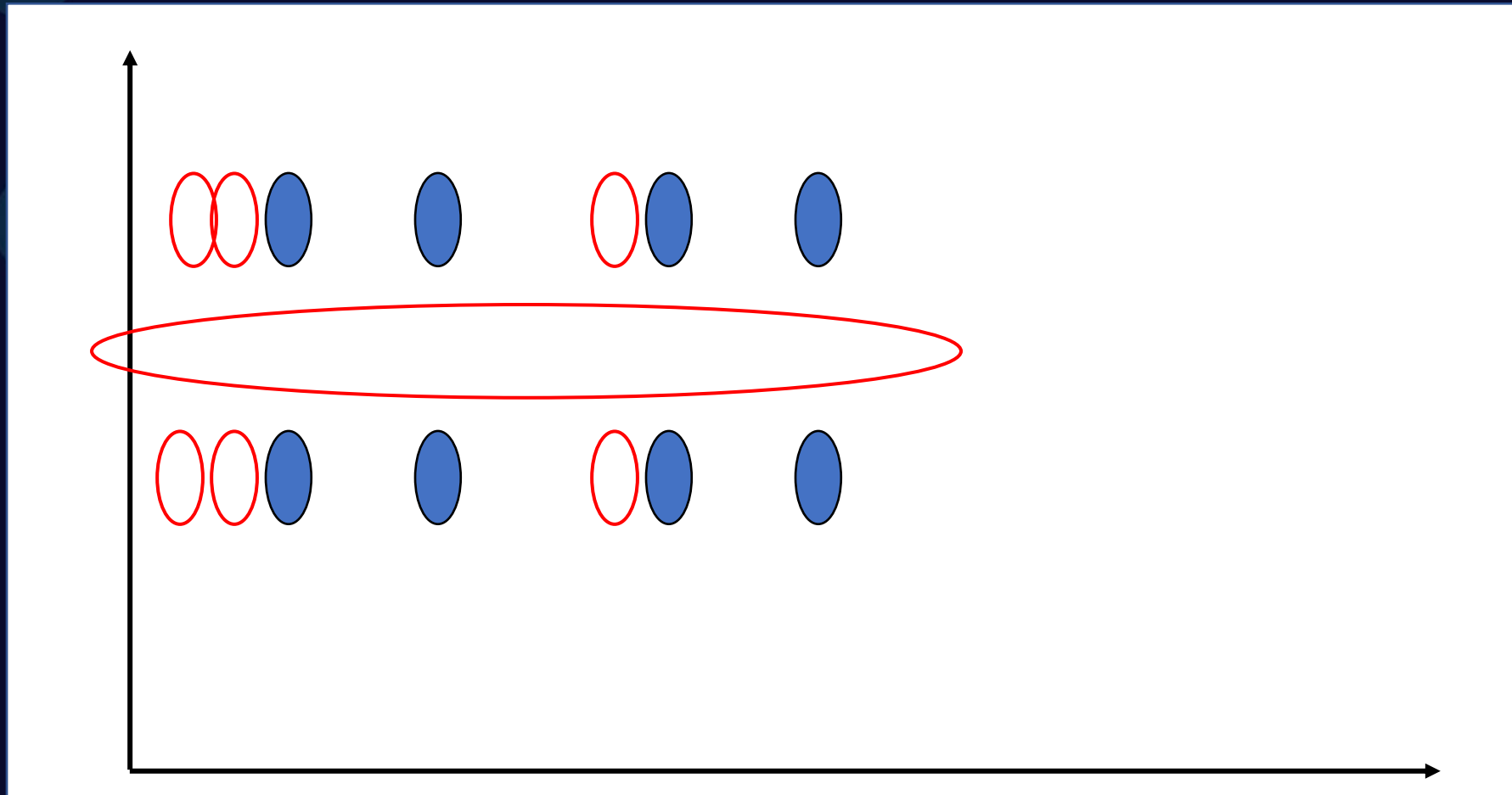
- Use maximum similarity of pairs:

$$\text{sim}(c_i, c_j) = \max_{x \in c_i, y \in c_j} \text{sim}(x, y)$$

- Can result in “straggly” (long and thin) clusters due to chaining effect.
- After merging  $c_i$  and  $c_j$ , the similarity of the resulting cluster to another cluster,  $c_k$ , is:

$$\text{sim}((c_i \cup c_j), c_k) = \max(\text{sim}(c_i, c_k), \text{sim}(c_j, c_k))$$

# Single Link Example



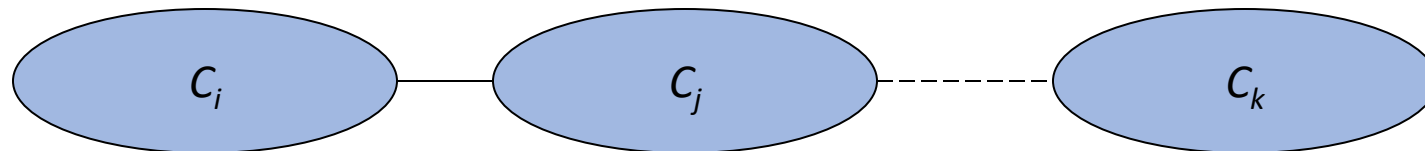
# Complete Link Agglomerative Clustering

- Use minimum similarity of pairs:

$$\text{sim}(c_i, c_j) = \min_{x \in c_i, y \in c_j} \text{sim}(x, y)$$

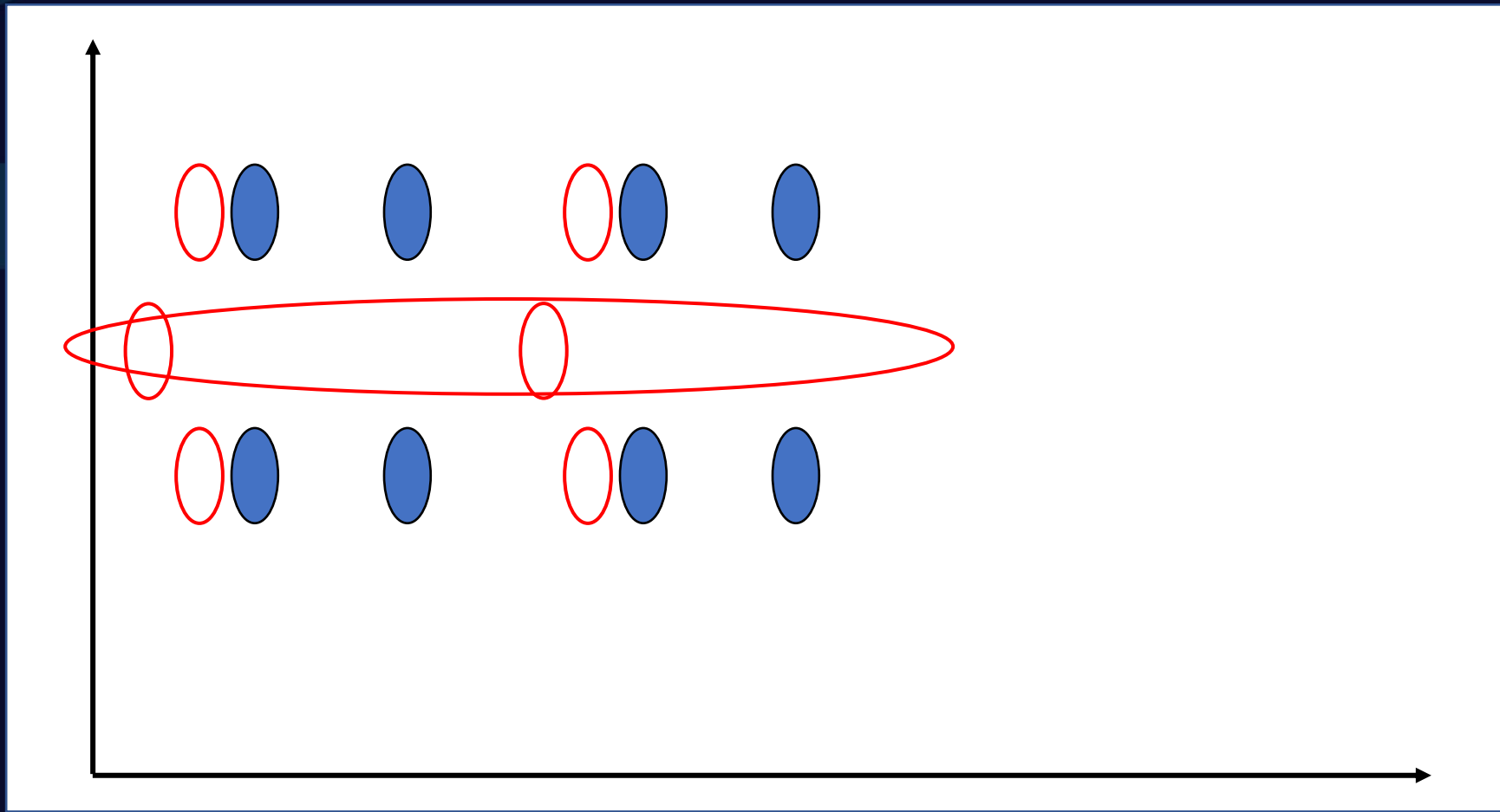
- Makes “tighter,” spherical clusters that are typically preferable.
- After merging  $c_i$  and  $c_j$ , the similarity of the resulting cluster to another cluster,  $c_k$ , is:

$$\text{sim}((c_i \cup c_j), c_k) = \min(\text{sim}(c_i, c_k), \text{sim}(c_j, c_k))$$





# Complete Link Example



# Key notion: *cluster representative*

- We want a notion of a representative point in a cluster
- Representative should be some sort of “typical” or central point in the cluster, e.g.,
  - point inducing smallest radii to docs in cluster
  - smallest squared distances, etc.
  - point that is the “average” of all docs in the cluster
    - Centroid or center of gravity

# Centroid-based Similarity

- Always maintain average of vectors in each cluster:

$$\vec{s}(c_j) = \frac{\sum_{\vec{x} \in c_j} \vec{x}}{|c_j|}$$

- Compute similarity of clusters by:

$$\text{sim}(c_i, c_j) = \text{sim}(s(c_i), s(c_j))$$

- For non-vector data, can't always make a centroid

# Computational Complexity

- In the first iteration, all HAC methods need to compute similarity of all pairs of  $n$  individual instances which is  $O(mn^2)$ .
- In each of the subsequent  $n-2$  merging iterations, compute the distance between the most recently created cluster and all other existing clusters.
- Maintaining of heap of distances allows this to be  $O(mn^2 \log n)$

# Major issue - labeling

- After clustering algorithm finds clusters - how can they be useful to the end user?
- Need pithy label for each cluster
  - In search results, say “Animal” or “Car” in the *jaguar* example.
  - In topic trees, need navigational cues.
    - Often done by hand, a posteriori.

How would you do this?

# How to Label Clusters

- Show titles of typical documents
  - Titles are easy to scan
  - Authors create them for quick scanning!
  - But you can only show a few titles which may not fully represent cluster
- Show words/phrases prominent in cluster
  - More likely to fully represent cluster
  - Use distinguishing words/phrases
    - Differential labeling
  - But harder to scan

# Labeling

- Common heuristics - list 5-10 most frequent terms in the centroid vector.
  - Drop stop-words; stem.
- Differential labeling by frequent terms
  - Within a collection “Computers”, clusters all have the word **computer** as frequent term.
  - Discriminant analysis of centroids.
- Perhaps better: distinctive noun phrase

# Efficient large-scale clustering

- 17M biomedical papers in Medline
  - Each paper contains ~20 citations
  - Clustering 340M citations
- 
- $\sim 10^{17}$  distance calculations for naïve HAC



# Expensive Distance Metric for Text

- String edit distance
- Compute with dynamic programming
- Costs for character:
  - insertion
  - deletion
  - substitution
  - ...

	S	e	c	a	t	
S	0.0	0.7	1.4	2.1	2.8	3.5
c	0.7	0.0	0.7	1.1	1.4	1.8
o	1.4	0.7	1.0	0.7	1.4	1.8
t	2.1	1.1	1.7	1.4	1.7	2.4
t	2.8	1.4	2.1	1.8	2.4	1.7
t	3.5	1.8	2.4	2.1	2.8	2.4

# String edit (Levenstein) distance

- Distance is **shortest sequence of edit commands** that transform  $s$  to  $t$ .
- Simplest set of operations:
  - Copy character from  $s$  over to  $t$
  - Delete a character in  $s$  (cost 1)
  - Insert a character in  $t$  (cost 1)
  - Substitute one character for another (cost 1)

# Levenstein distance - example

- distance("William Cohen", "William Cohon")

<i>s</i>	W	I	L	L	I	A	M	_	C	O	H	E		N
					/	/	/	/	/	/	/	/		
						<i>alignment</i>								
<i>t</i>	W	I	L	L	L	I	A	M	_	C	O	H	O	N
<i>op</i>	C	C	C	C	I	C	C	C	C	C	C	C	S	C
<i>cost</i>	0	0	0	0	1	1	1	1	1	1	1	1	2	2

# Levenstein distance - example

- distance("William Cohen", "William Cohon")

<i>s</i>	W	I	L	L	gap	I	A	M	_	C	O	H	E	N
<i>t</i>	W	I	L	L	L	I	A	M	_	C	O	H	O	N
<i>op</i>	C	C	C	C	I	C	C	C	C	C	C	C	S	C
<i>cost</i>	0	0	0	0	1	1	1	1	1	1	1	1	2	2

# Computing Levenstein distance

$D(i,j)$  = score of **best** alignment from  $s1..si$  to  $t1..tj$

$$= \min \left\{ \begin{array}{ll} D(i-1,j-1), \text{ if } s_i=t_j & //copy \\ D(i-1,j-1)+1, \text{ if } s_i \neq t_j & //substitute \\ D(i-1,j)+1 & //insert \\ D(i,j-1)+1 & //delete \end{array} \right.$$

# Computing Levenstein distance

	C	O	H	E	N
M	1	2	3	4	5
C	1	2	3	4	5
C	2	2	3	4	5
O	3	2	3	4	5
H	4	3	2	3	4
N	5	4	3	3	3

=  $D(s, t)$

# Computing Levenshtein distance

A *trace* indicates where the min value came from, and can be used to find edit operations and/or a best *alignment* (may be more than 1)

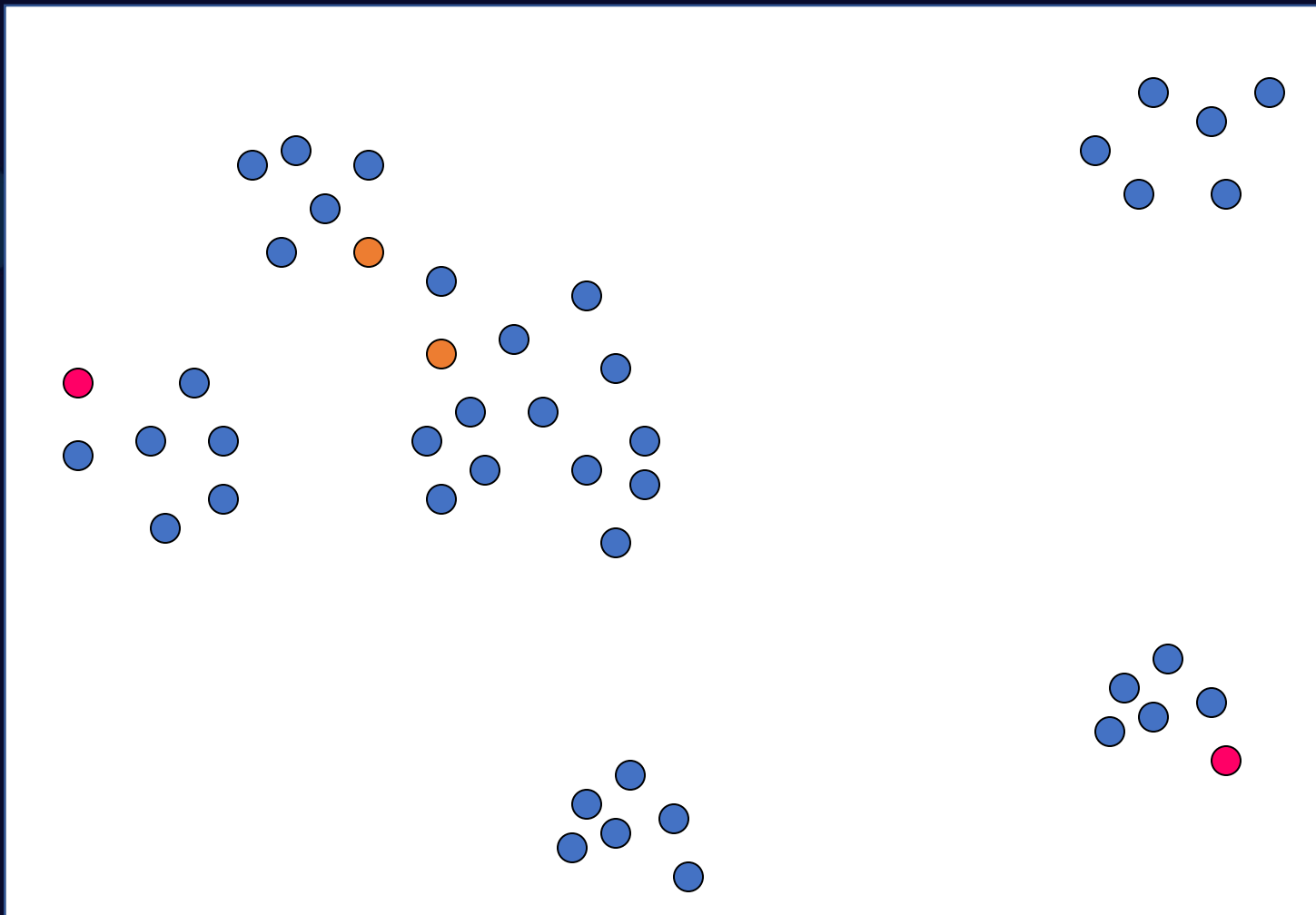
	C	O	H	E	N
M	1	2	3	4	5
C	1	2	3	4	5
C	2	3	3	4	5
O	3	2	3	4	5
H	4	3	2	3	4
N	5	4	3	3	3

# The Canopies Approach

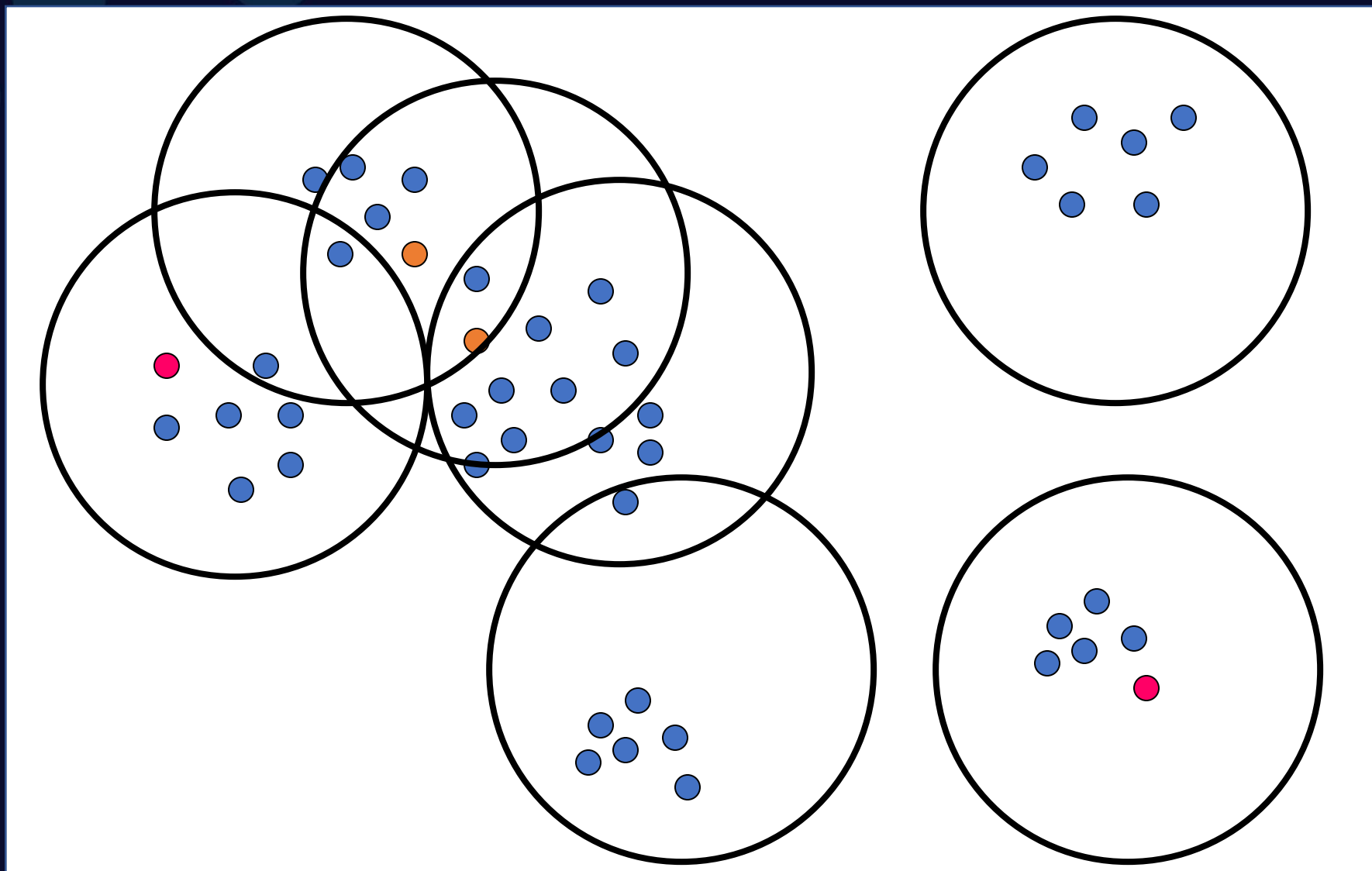
- Two distance metrics: cheap & expensive
- First Pass
  - very inexpensive distance metric
  - create overlapping canopies
- Second Pass
  - expensive, accurate distance metric
  - canopies determine which distances calculated



# Illustrating Canopies

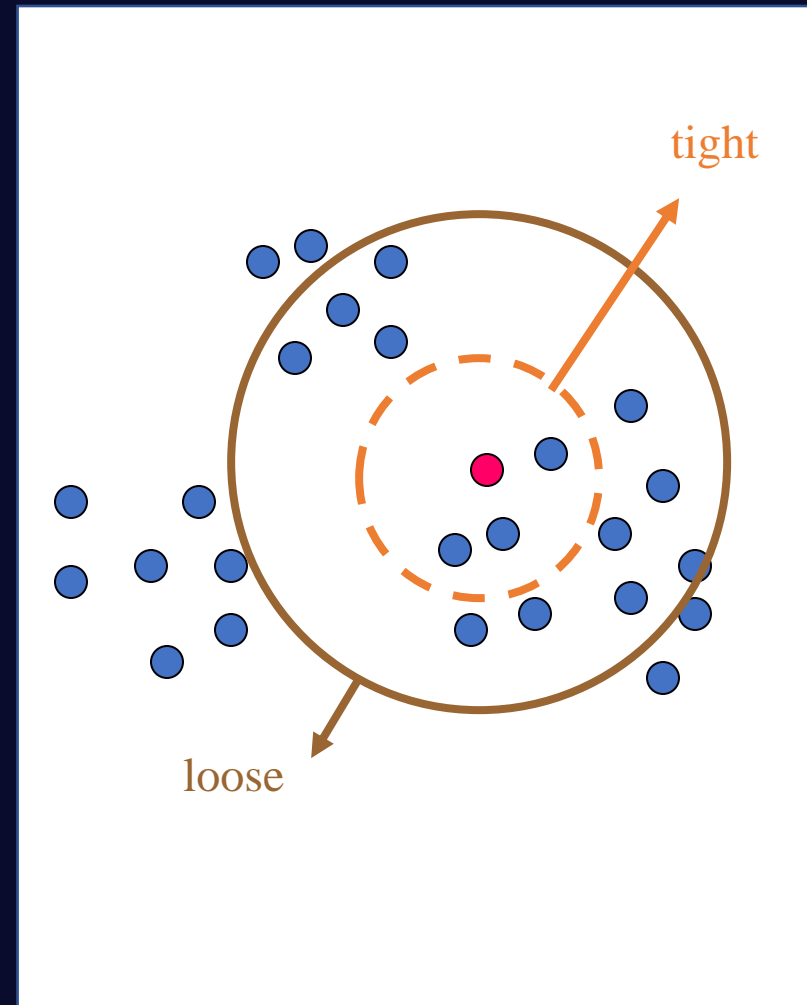


# Overlapping Canopies



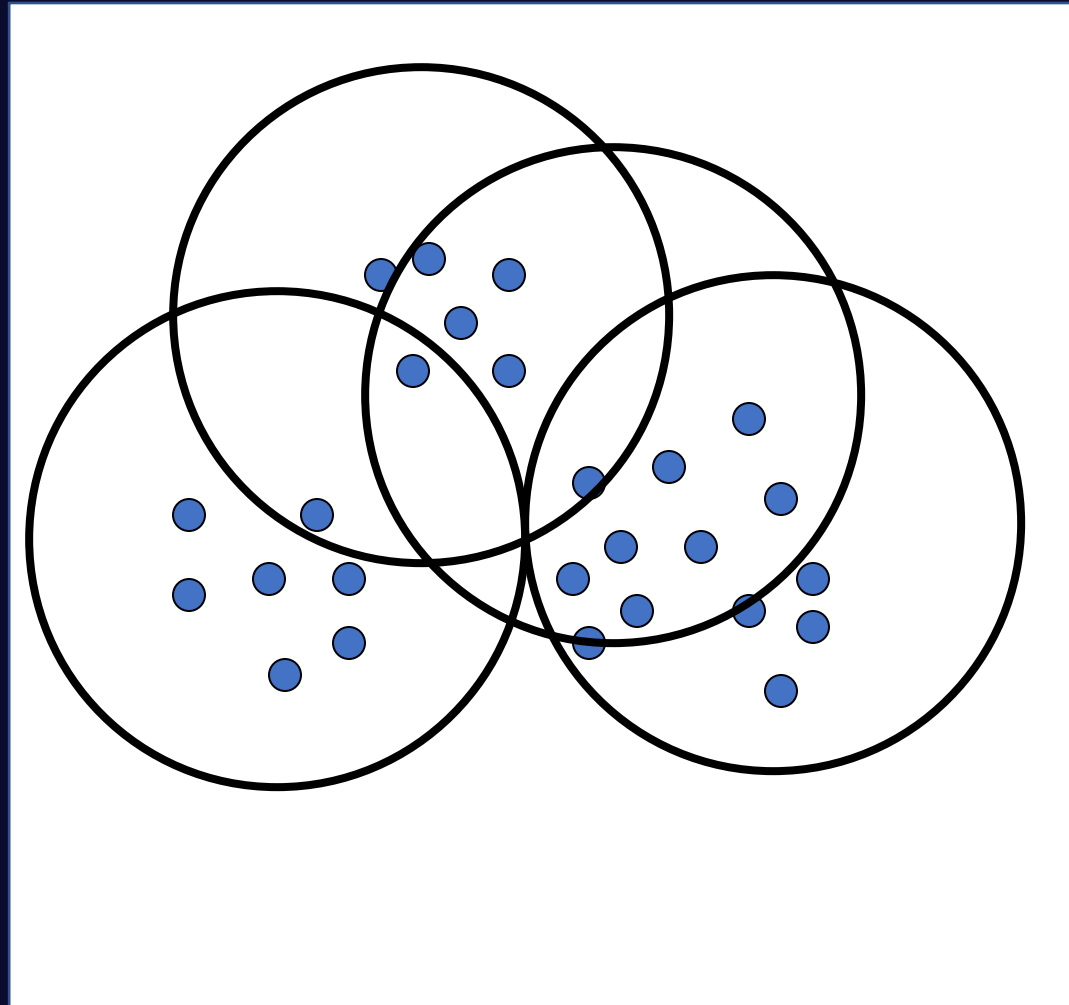
# Creating canopies with two thresholds

- Put all points in D
- Loop:
  - Pick a point X from D
  - Put points within  $K_{loose}$  of X in canopy
  - Remove points within  $K_{tight}$  of X from D



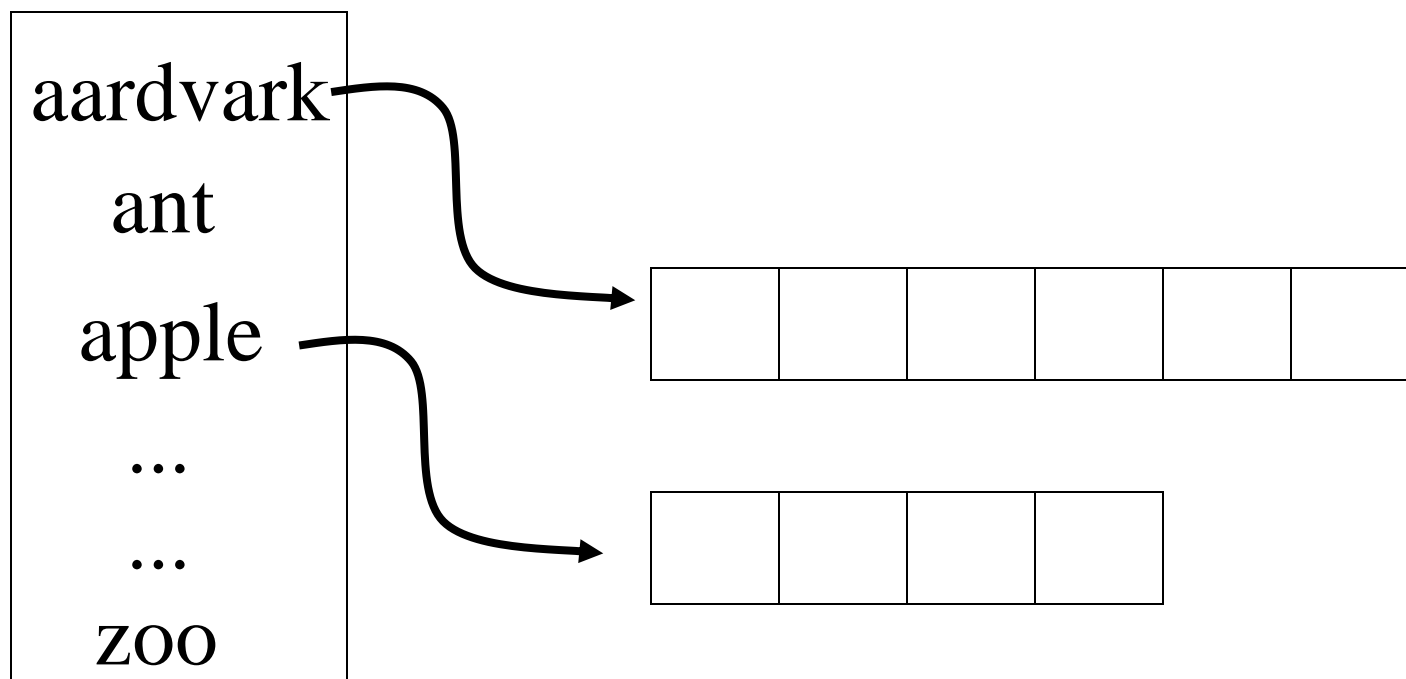
# Using canopies with HAC

- Calculate expensive distances between points in the same canopy
- All other distances default to infinity
- Use finite distances and iteratively merge closest



# Inexpensive Distance Metric for Text

- Word-level matching (TFIDF)
- Inexpensive using an inverted index



# Expensive Distance Metric for Text

- String edit distance
- Compute with dynamic programming
- Costs for character:
  - insertion
  - deletion
  - substitution
  - ...

	S	e	c	a	t	
S  c  o  t  t	0.0	0.7	1.4	2.1	2.8	3.5
	0.7	0.0	0.7	1.1	1.4	1.8
	1.4	0.7	1.0	0.7	1.4	1.8
	2.1	1.1	1.7	1.4	1.7	2.4
	2.8	1.4	2.1	1.8	2.4	1.7
	3.5	1.8	2.4	2.1	2.8	2.4

# Computational Savings

- inexpensive metric  $\ll$  expensive metric
  - Canopy creation nearly for free
- Number of canopies:  $c$  (large)
- Number of canopies per point:  $f$  (small but  $> 1$ )
- $fn/c$  points per canopy (if evenly spread)
- $O(c(fn/c)^2)$  distance calculations initially

- Complexity reduction:  $O(f^2/c)$

# Canopies

- Two distance metrics
  - cheap and approximate
  - expensive and accurate
- Two-pass clustering
  - create overlapping canopies
  - full clustering with limited distances