

# 95-865 Unstructured Data Analytics

## Lecture 8: Clustering (cont'd)

Slides by George H. Chen

# Clustering on Text Demo

- We're clustering on the 20 Newsgroups dataset (preprocessed by lemmatizing every token)
- We filtered out some documents that are likely not in English
- We filtered out vocab words that showed up in too many or too few documents
  - Resulting 2D table of feature vectors: `filtered_tf`
  - Resulting 1D table of vocabulary words: `filtered_vocab`
- After filtering, text documents still varied wildly in length
  - Convert each row of `filtered_tf` into a probability vector
    - Resulting 2D table of probability vectors: `prob_vectors`

# One-Hot Vectors are Probability Vectors

- Imagine a document that mentions only a single word "learn" (assume that "learn" is in the vocabulary)
- The raw counts (term frequency) feature vector of this document would be a one-hot vector (all 0s except for a 1 at the word index for "learn")

$[0, 0, 0, 0, \dots, 0, 1, 0, \dots, 0]$

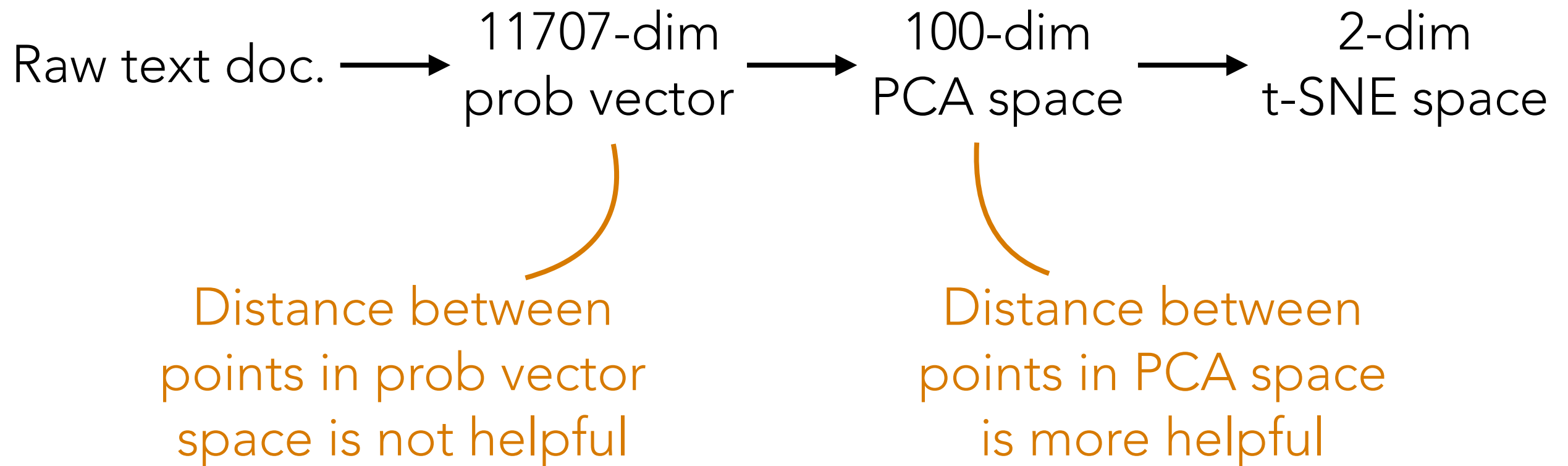


at index of vocabulary word "learn"

- A one-hot vector has entries that are nonnegative and that sums to 1

one-hot vectors are valid probability vectors

# Clustering on Text Demo



We show clustering results in 100-dim PCA space

We also show clustering results in 2-dim t-SNE space

# Clustering on Text

Resuming the demo from last time...

# (Flashback) Learning a GMM

Step 0: Guess  $k$

Step 1: Guess cluster probabilities, means, and covariances  
(often done using  $k$ -means)

Repeat until convergence:

Step 2: Compute probability of each point being in each of the  $k$  clusters

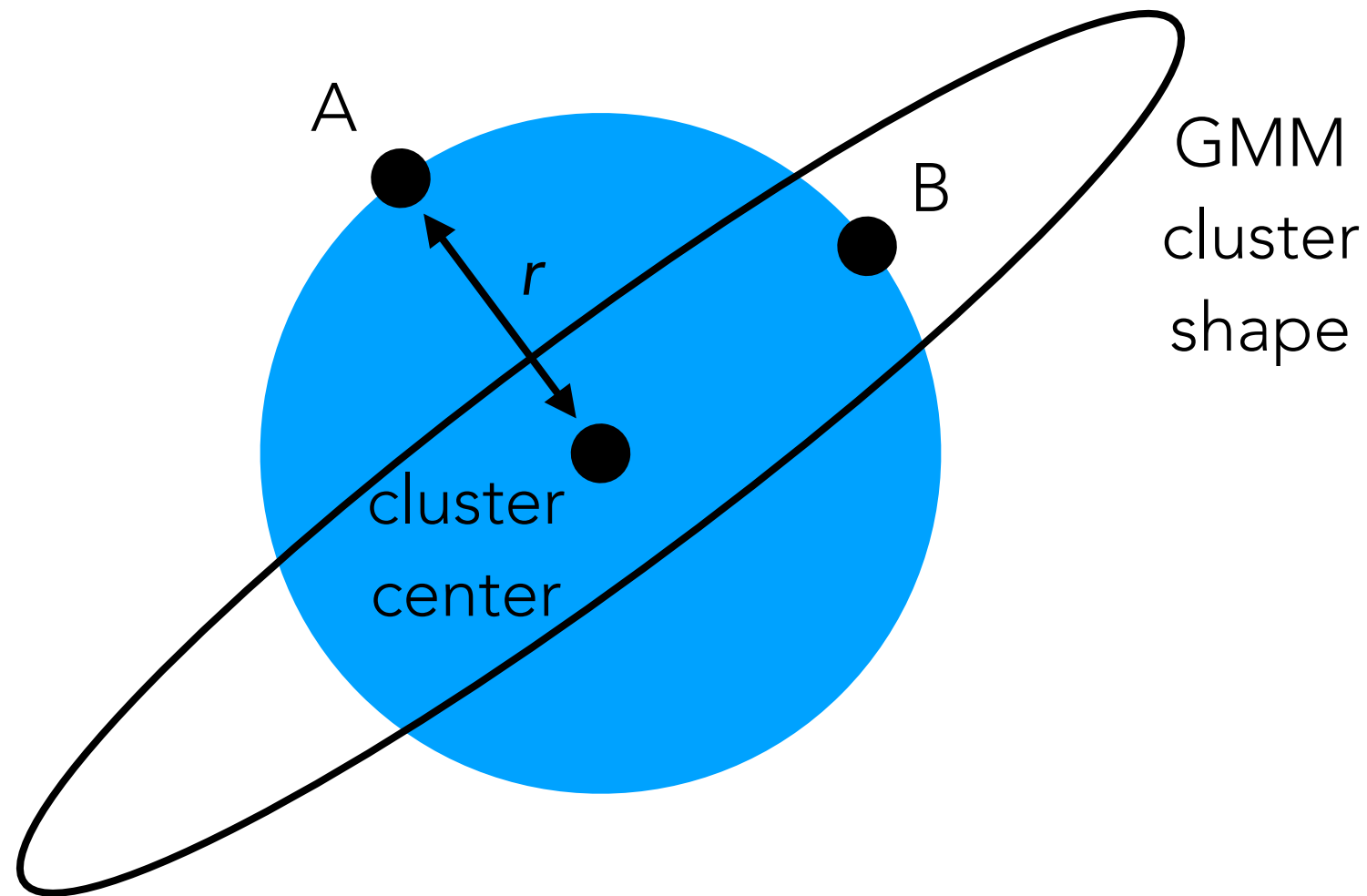
Step 3: Update cluster probabilities, means, and covariances accounting for probabilities of each point belonging to each of the clusters

This algorithm is called the **Expectation-Maximization (EM)** algorithm for GMMs (and approximately does maximum likelihood)

(Note: EM by itself is a general algorithm not just for GMMs)

# (Rough Intuition) How Shape is Encoded by a GMM

For this ellipse-shaped Gaussian, point B is considered more similar to the cluster center than point A



$k$ -means would think that point A and point B are equally similar to the cluster center (since both points are distance  $r$  away from the center)

# Relating $k$ -means to GMMs

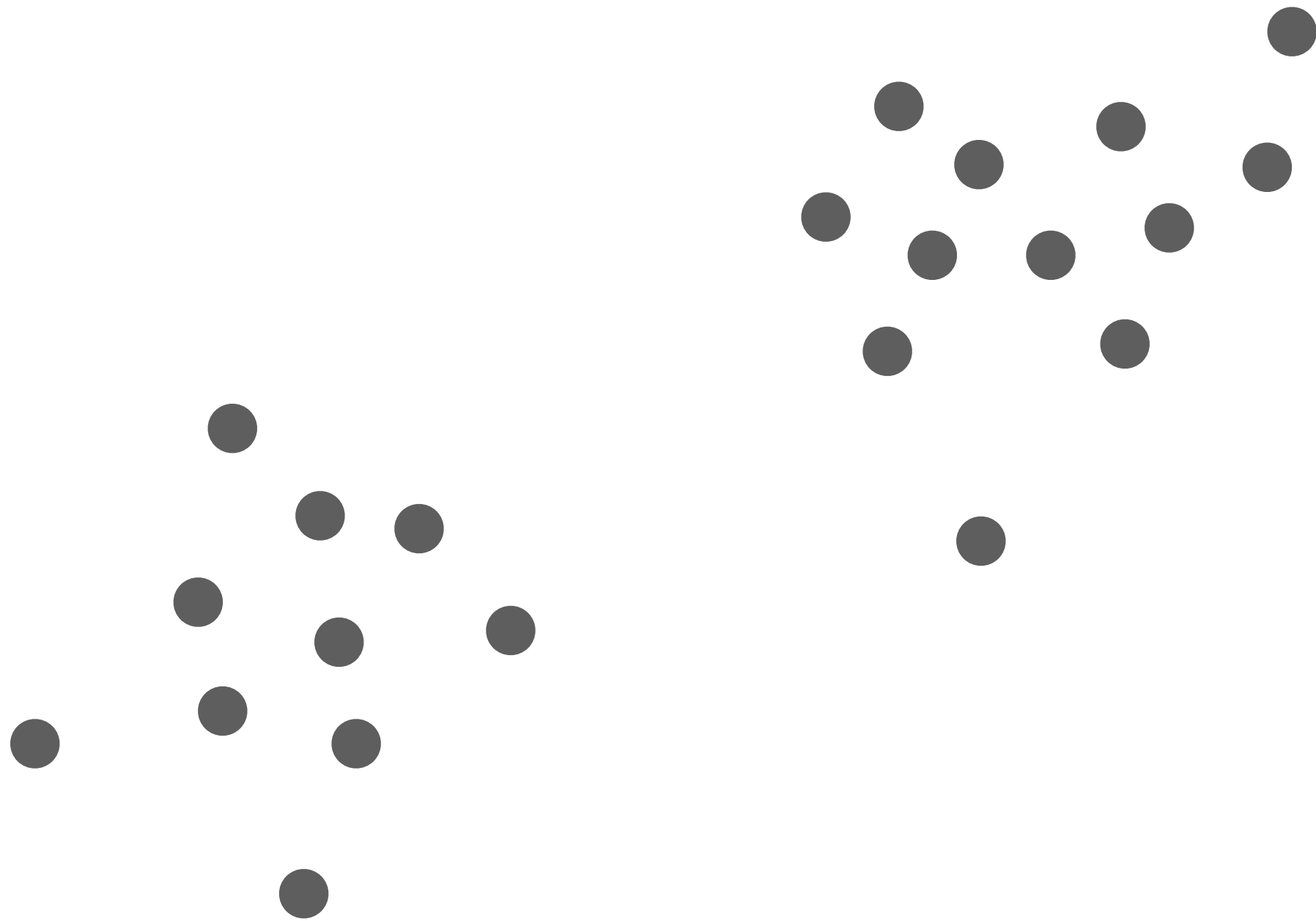
If the ellipses are all circles and have the same "skinniness" (e.g., in the 1D case it means they all have same variance):

- $k$ -means approximates the EM algorithm for GMMs (as there is no need to keep track of cluster shape)
- $k$ -means does a "hard" assignment of each point to a cluster, whereas the EM algorithm does a "soft" (probabilistic) assignment

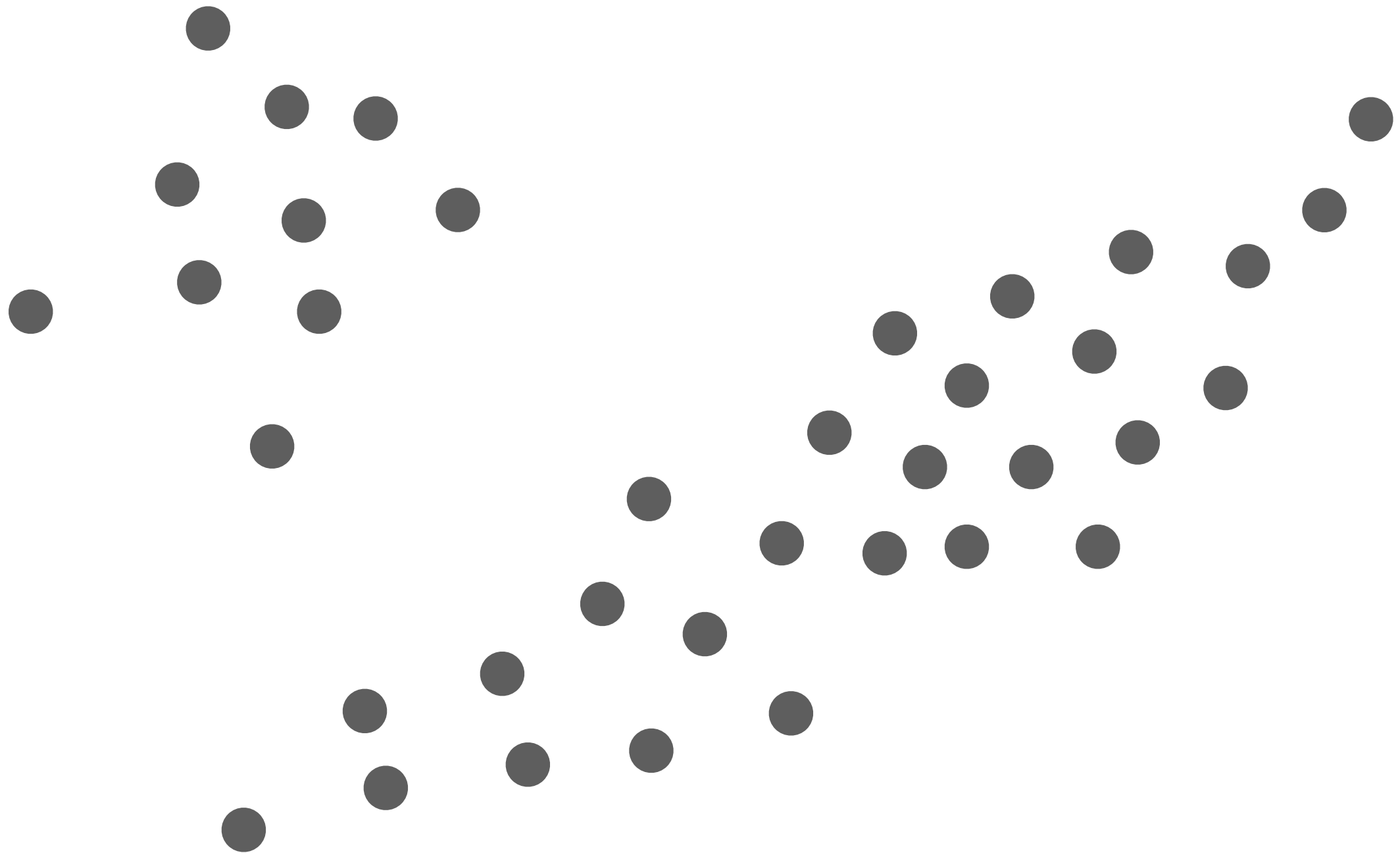
*Interpretation: When the data appear as if they're from a GMM with true clusters that "look like circles of equal size", then  $k$ -means should work well*



*k*-means should do well on this



But not on this



# Relating $k$ -means to GMMs

If the ellipses are all circles and have the same "skinniness" (e.g., in the 1D case it means they all have same variance):

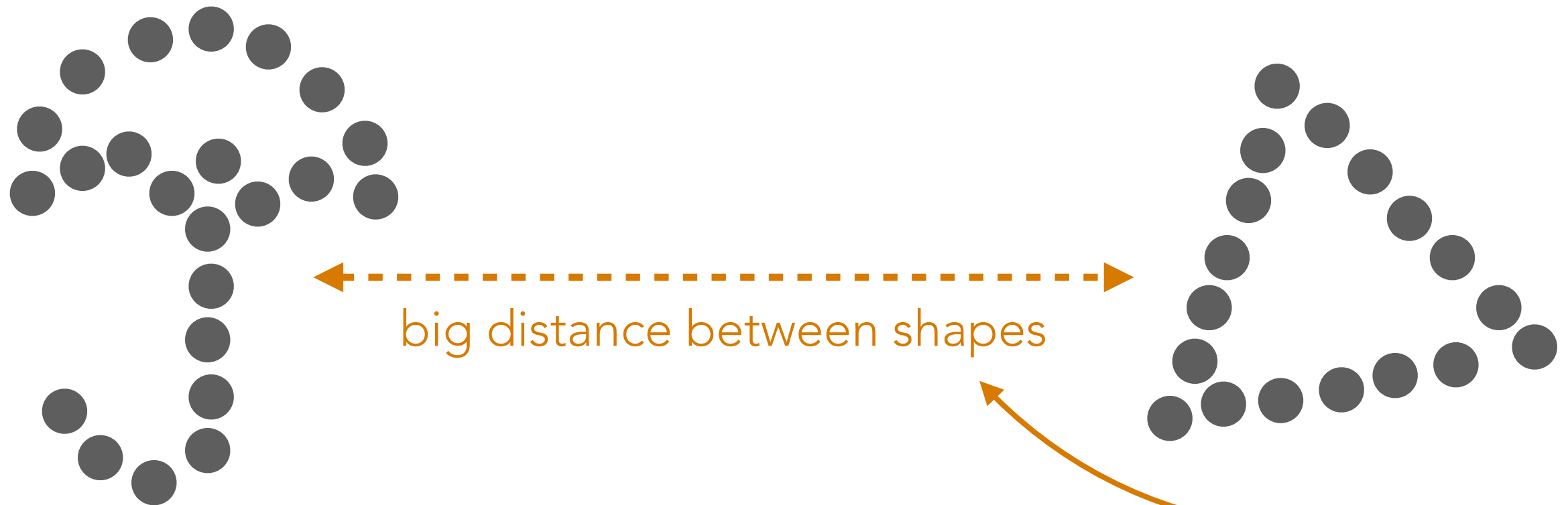
- $k$ -means approximates the EM algorithm for GMMs (as there is no need to keep track of cluster shape)
- $k$ -means does a "hard" assignment of each point to a cluster, whereas the EM algorithm does a "soft" (probabilistic) assignment

*Interpretation: When the data appear as if they're from a GMM with true clusters that "look like circles of equal size", then  $k$ -means should work well*

This is *not* the only scenario in which  $k$ -means should work well

Even if data aren't generated  
from a GMM, *k*-means and  
GMMs can still cluster correctly

This dataset obviously doesn't appear to be generated by a GMM



$k$ -means with  $k = 2$ , and 2-component GMM will both work well in identifying the two shapes as separate clusters

Key idea: the clusters are very *well-separated*  
(so that *many* clustering algorithms will work well in this case!)

# Automatically Choosing the Number of Clusters $k$

For  $k = 2, 3, \dots$  up to some user-specified max value:

Fit model ( $k$ -means or GMM) using  $k$

Compute a **score** for the model

But what score function should we use?

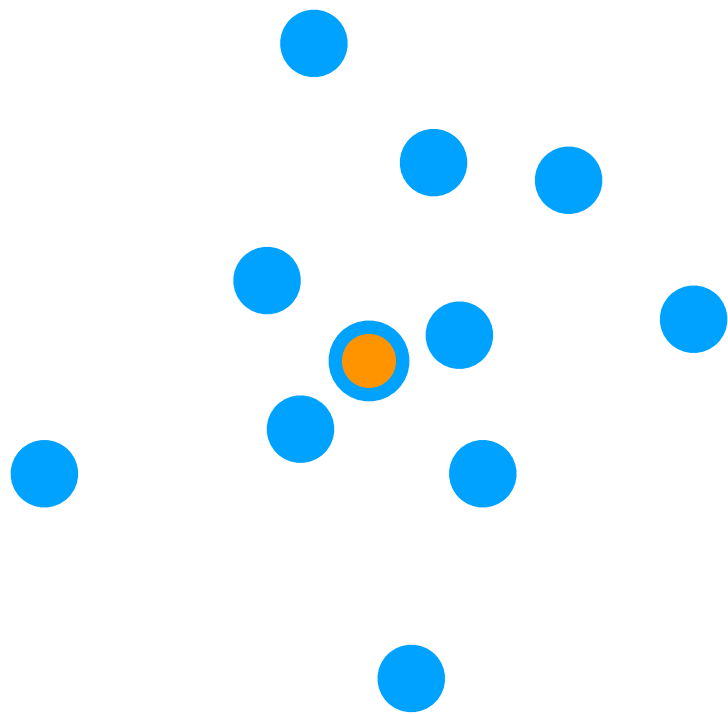
Use whichever  $k$  has the **best** score

No single way of choosing  $k$  is the "best" way

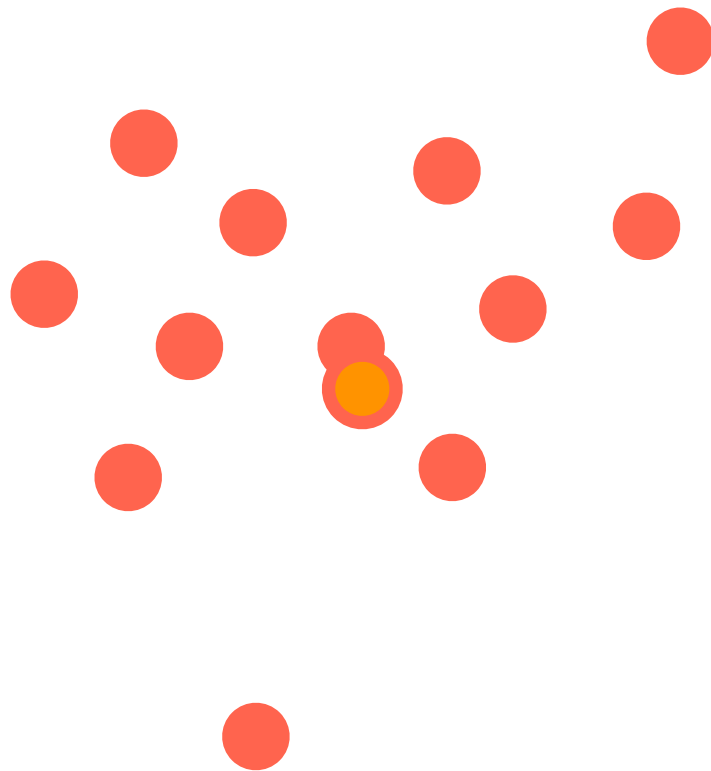
Here's an example of a score function you don't want to use

# Residual Sum of Squares

Look at one cluster at a time



Cluster 1

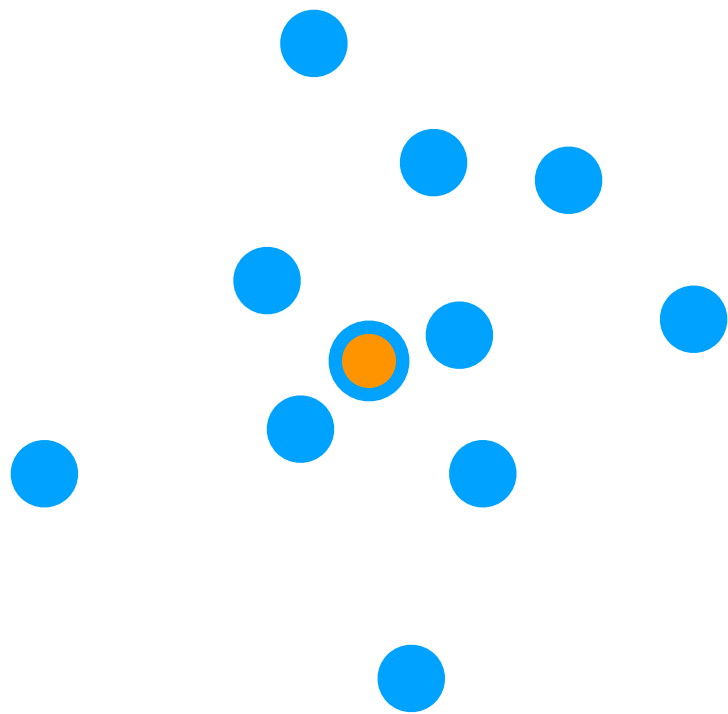


Cluster 2

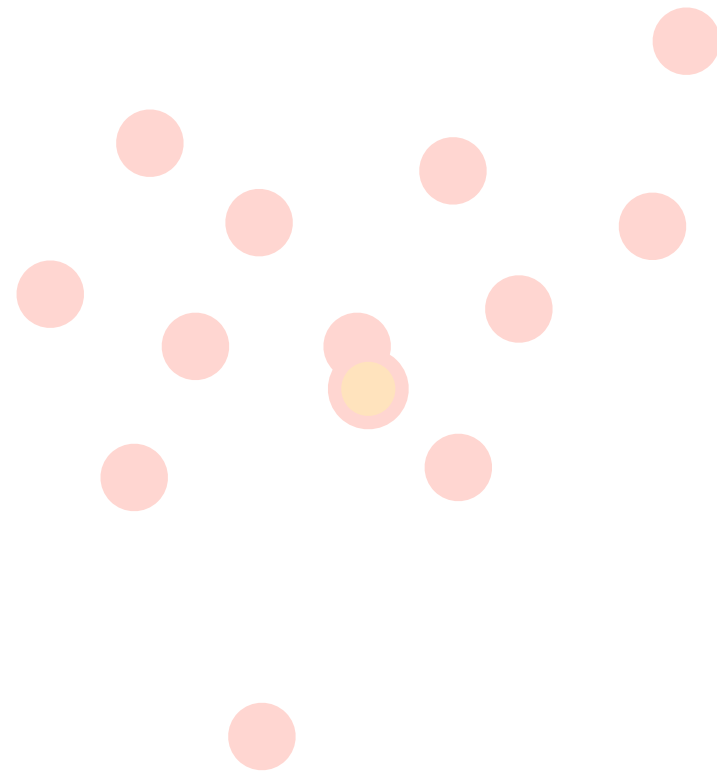


# Residual Sum of Squares

Look at one cluster at a time



Cluster 1

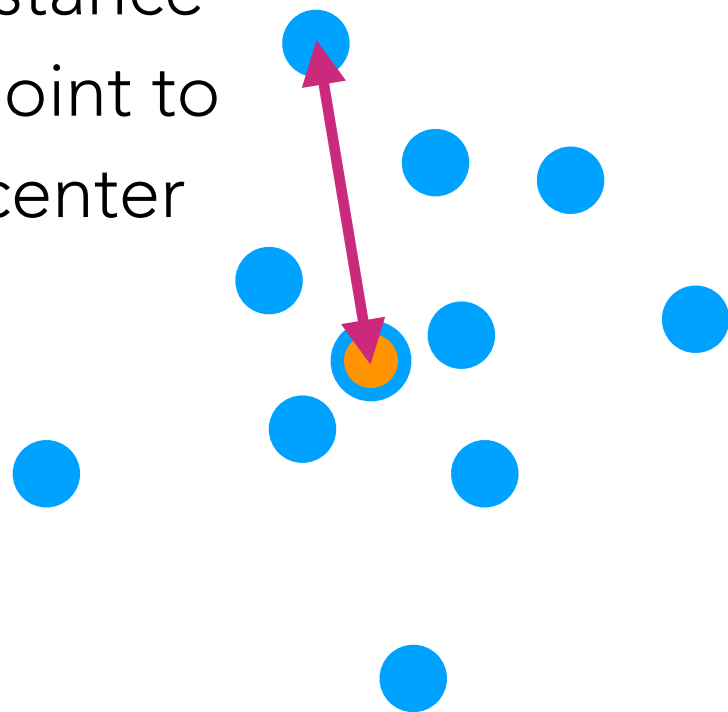


Cluster 2

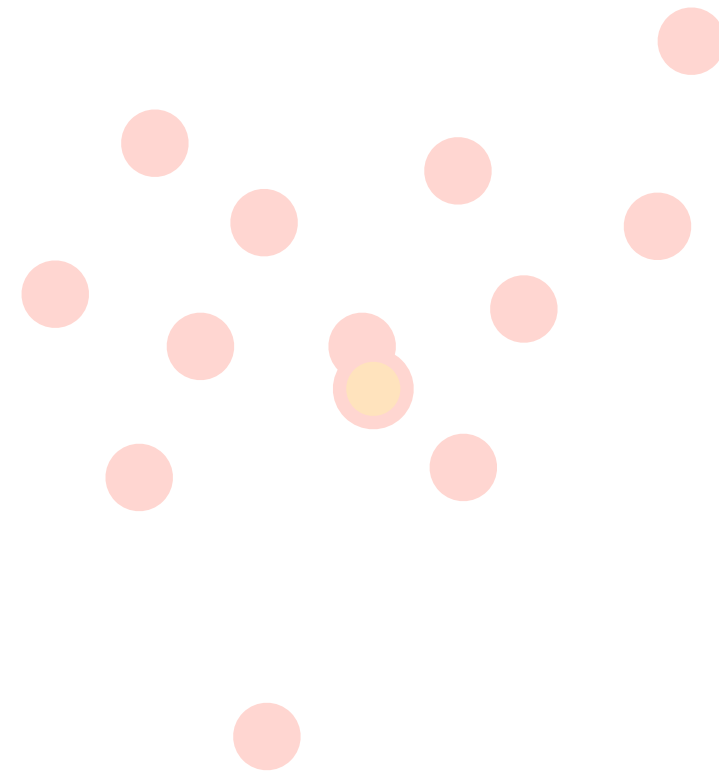
# Residual Sum of Squares

Look at one cluster at a time

Measure distance  
from each point to  
its cluster center



Cluster 1

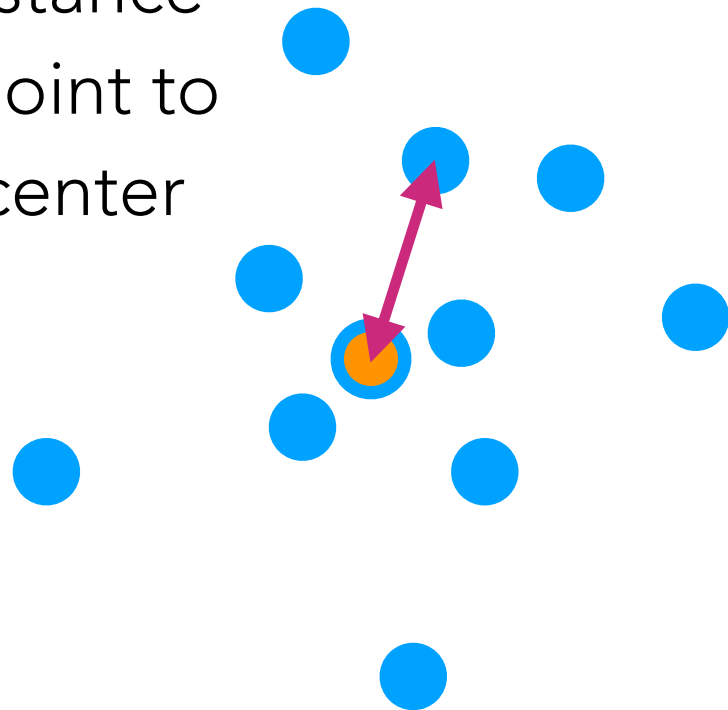


Cluster 2

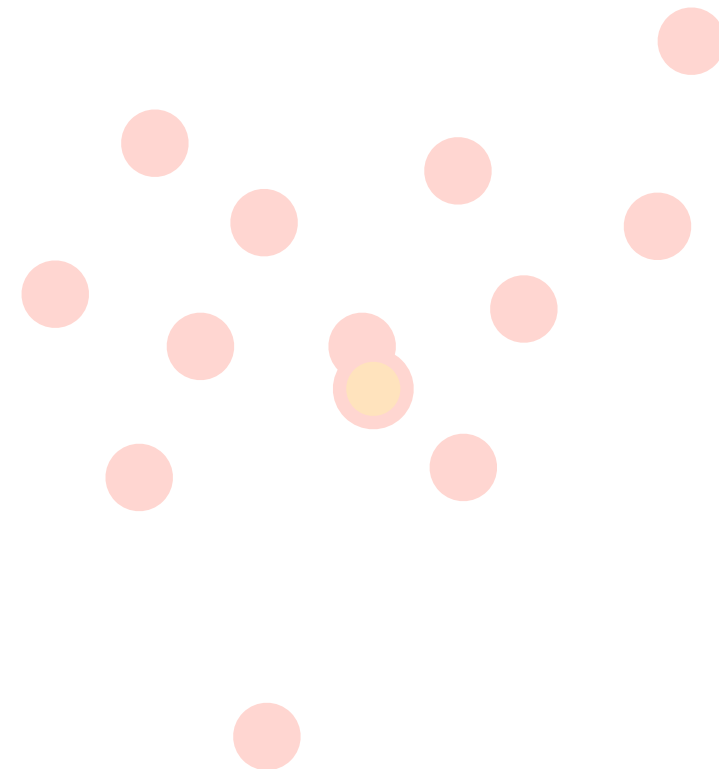
# Residual Sum of Squares

Look at one cluster at a time

Measure distance  
from each point to  
its cluster center



Cluster 1

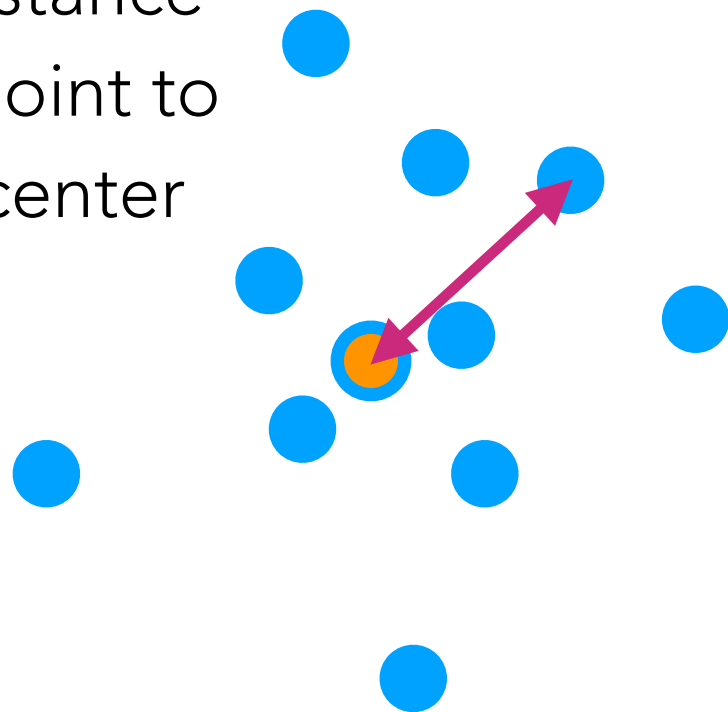


Cluster 2

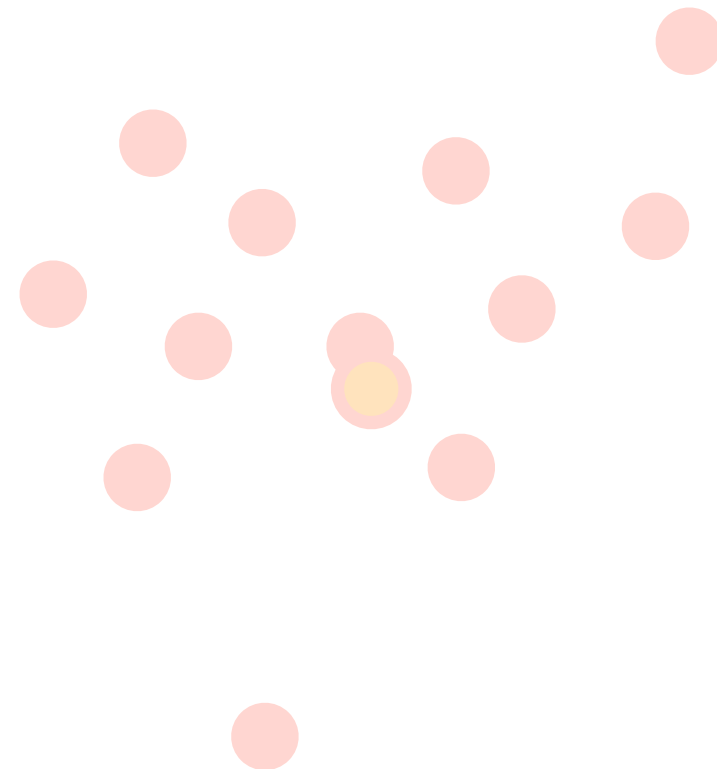
# Residual Sum of Squares

Look at one cluster at a time

Measure distance  
from each point to  
its cluster center



Cluster 1

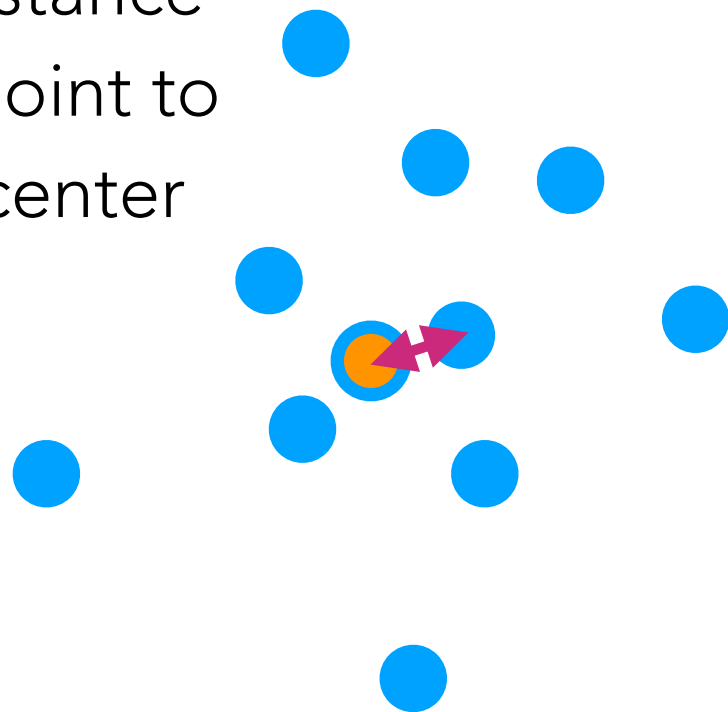


Cluster 2

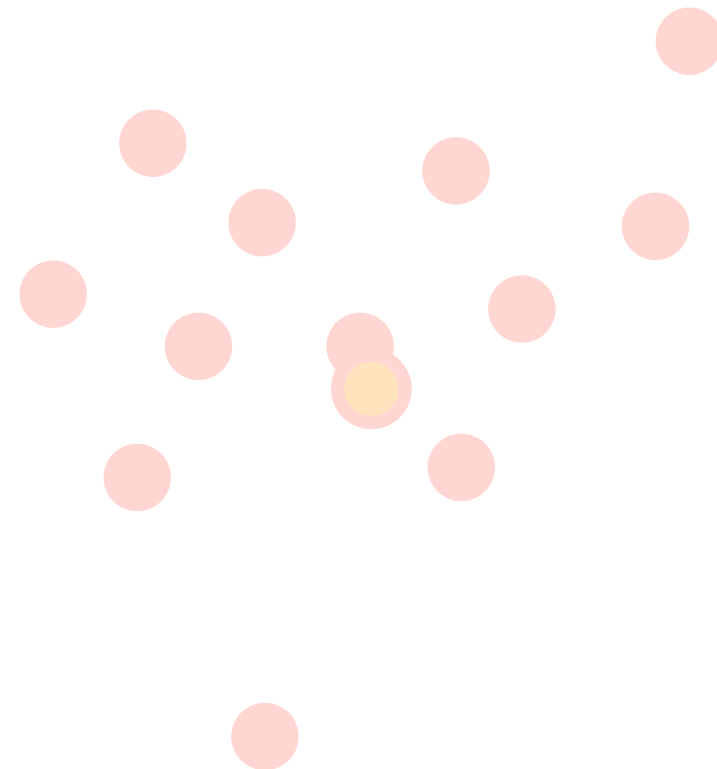
# Residual Sum of Squares

Look at one cluster at a time

Measure distance  
from each point to  
its cluster center



Cluster 1

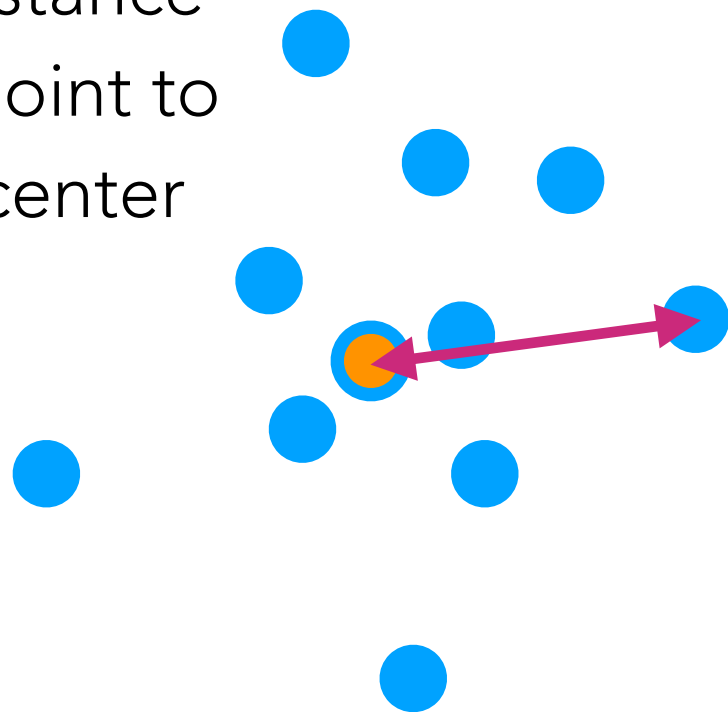


Cluster 2

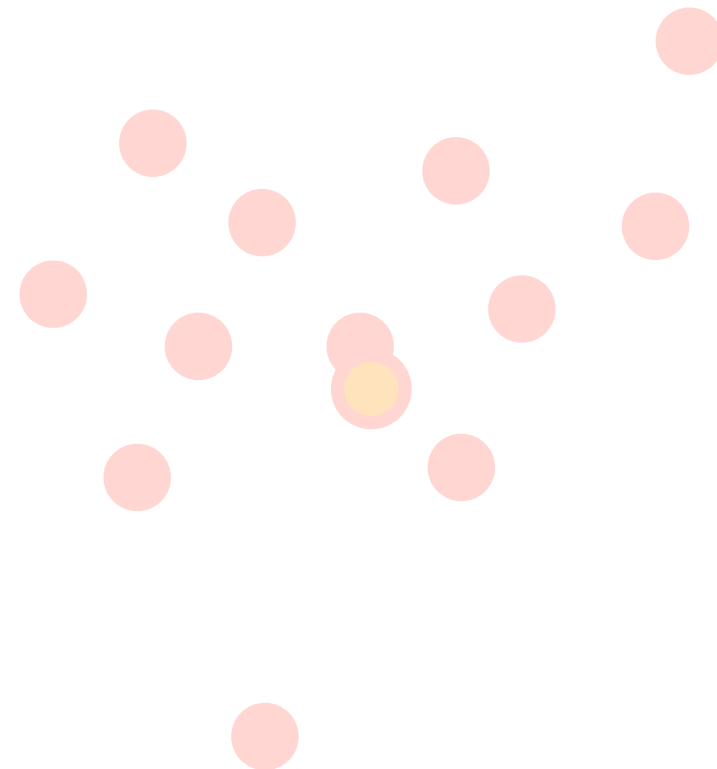
# Residual Sum of Squares

Look at one cluster at a time

Measure distance  
from each point to  
its cluster center



Cluster 1

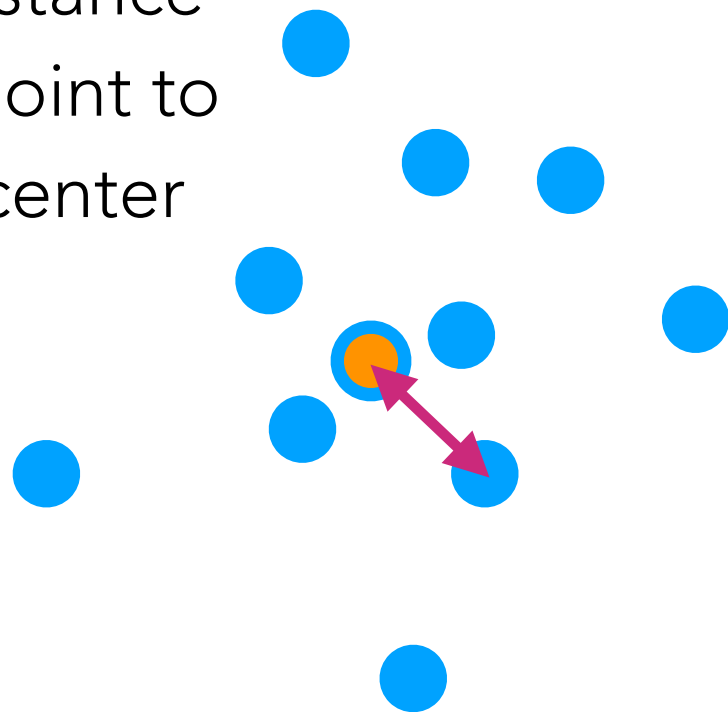


Cluster 2

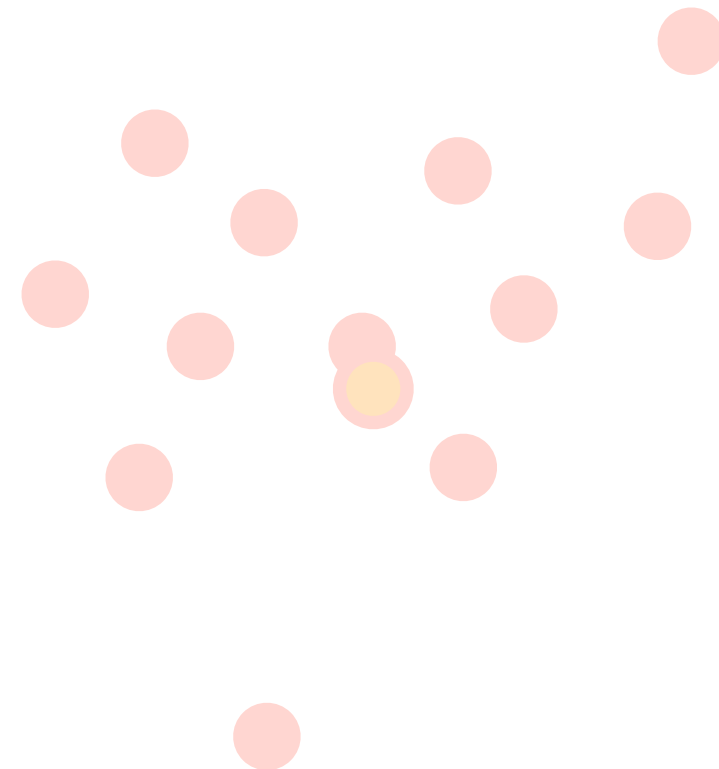
# Residual Sum of Squares

Look at one cluster at a time

Measure distance  
from each point to  
its cluster center



Cluster 1

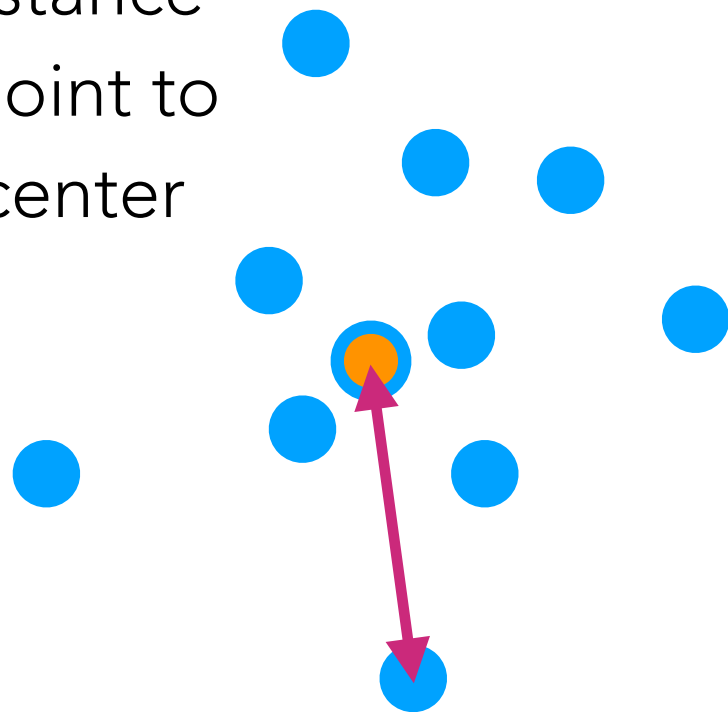


Cluster 2

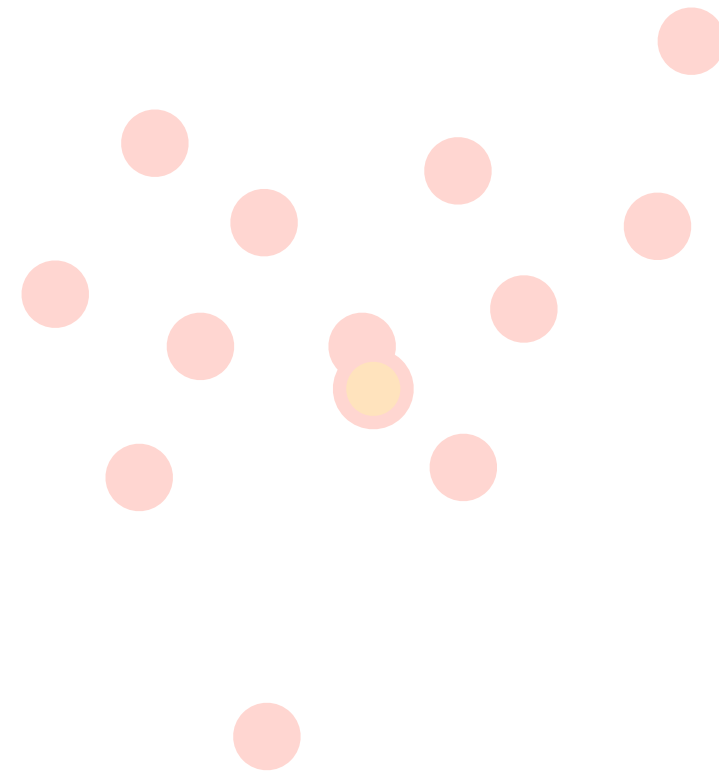
# Residual Sum of Squares

Look at one cluster at a time

Measure distance  
from each point to  
its cluster center



Cluster 1



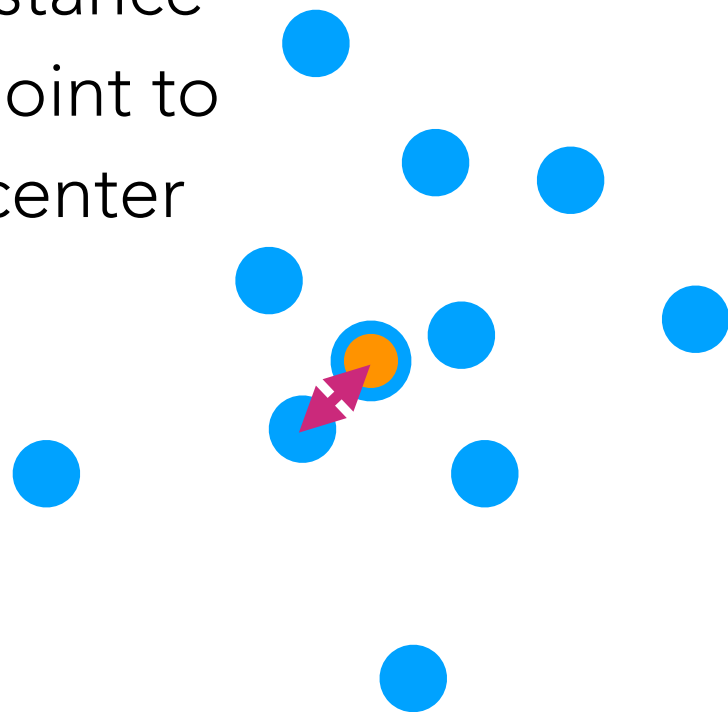
Cluster 2



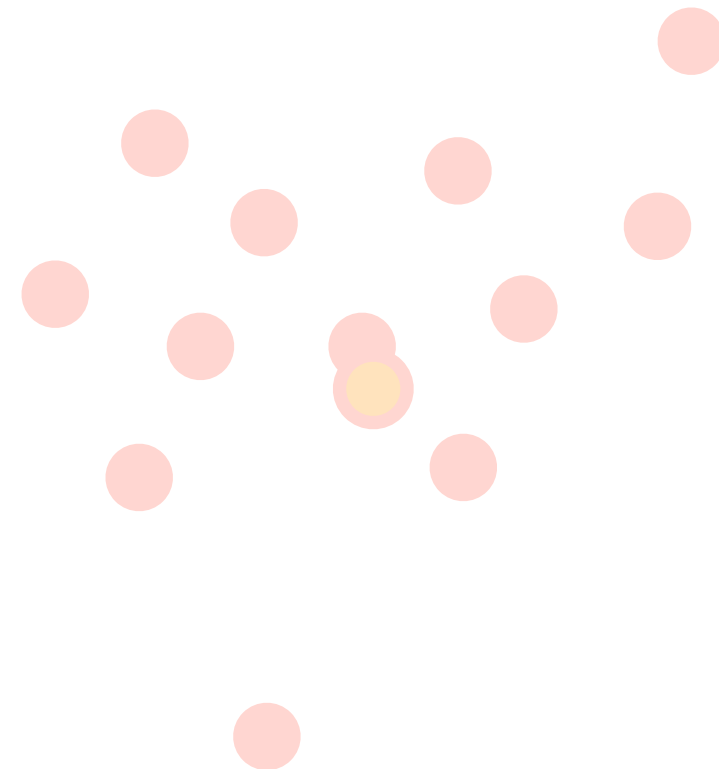
# Residual Sum of Squares

Look at one cluster at a time

Measure distance  
from each point to  
its cluster center



Cluster 1

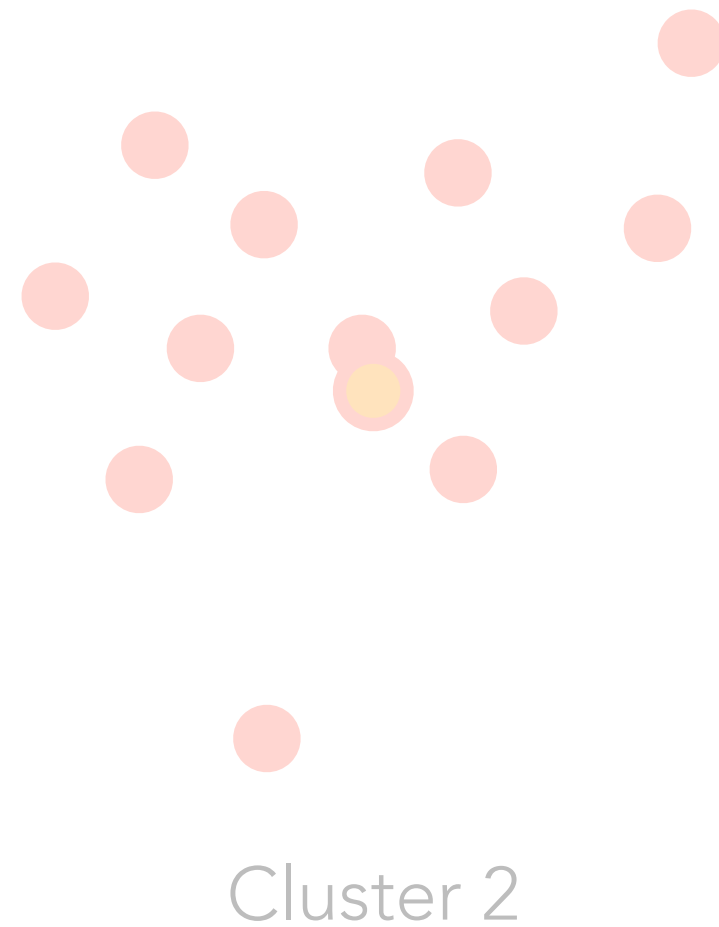
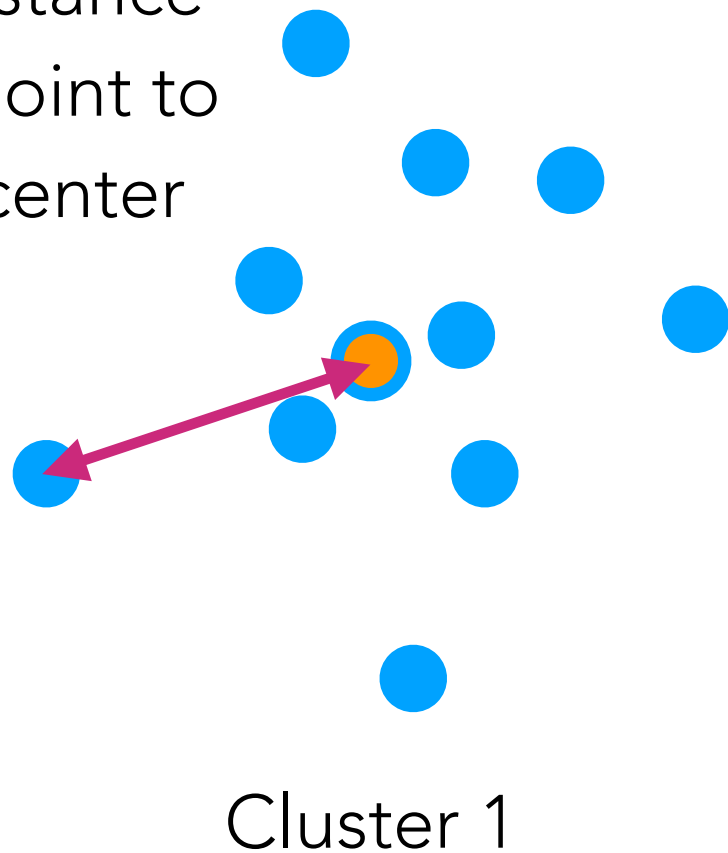


Cluster 2

# Residual Sum of Squares

Look at one cluster at a time

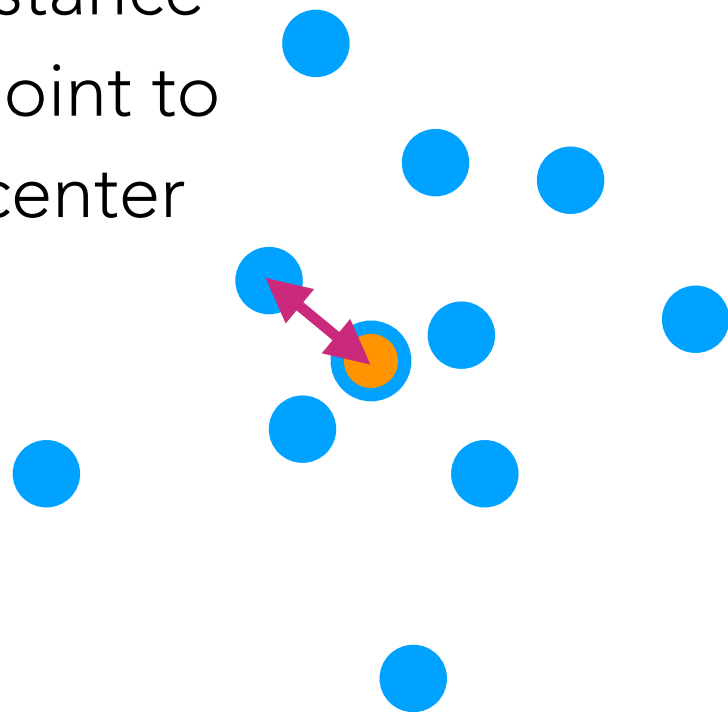
Measure distance  
from each point to  
its cluster center



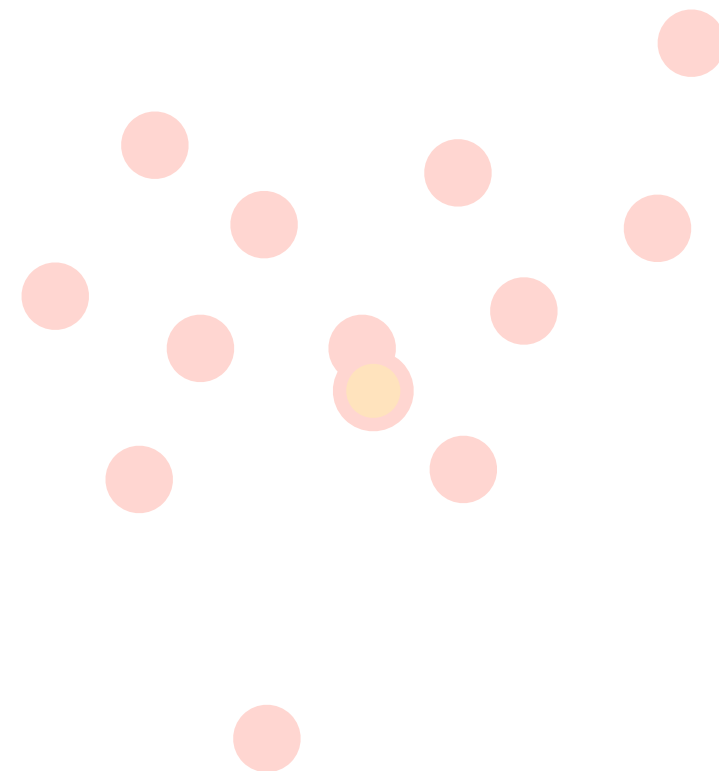
# Residual Sum of Squares

Look at one cluster at a time

Measure distance  
from each point to  
its cluster center



Cluster 1

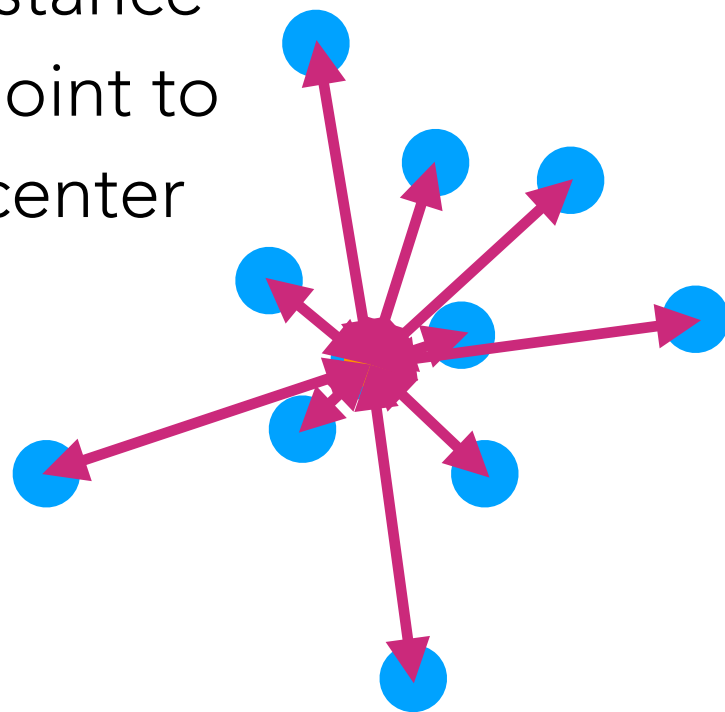


Cluster 2

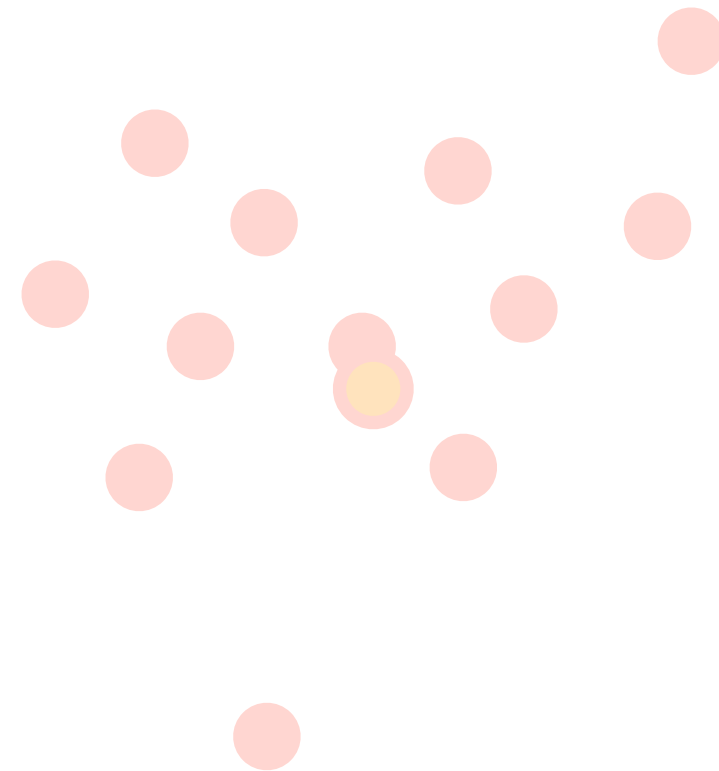
# Residual Sum of Squares

Look at one cluster at a time

Measure distance from each point to its cluster center



Cluster 1



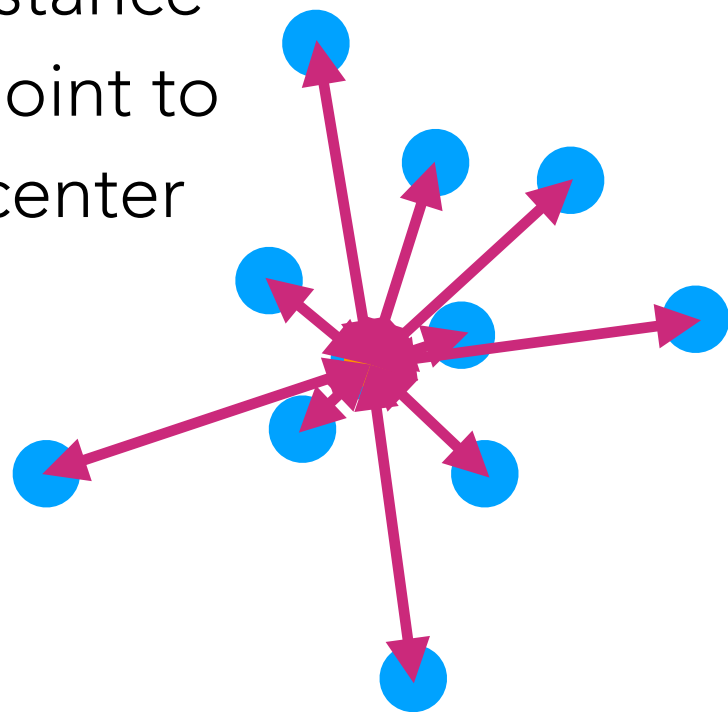
Cluster 2

Residual sum of squares for cluster 1:  
sum of *squared* purple lengths

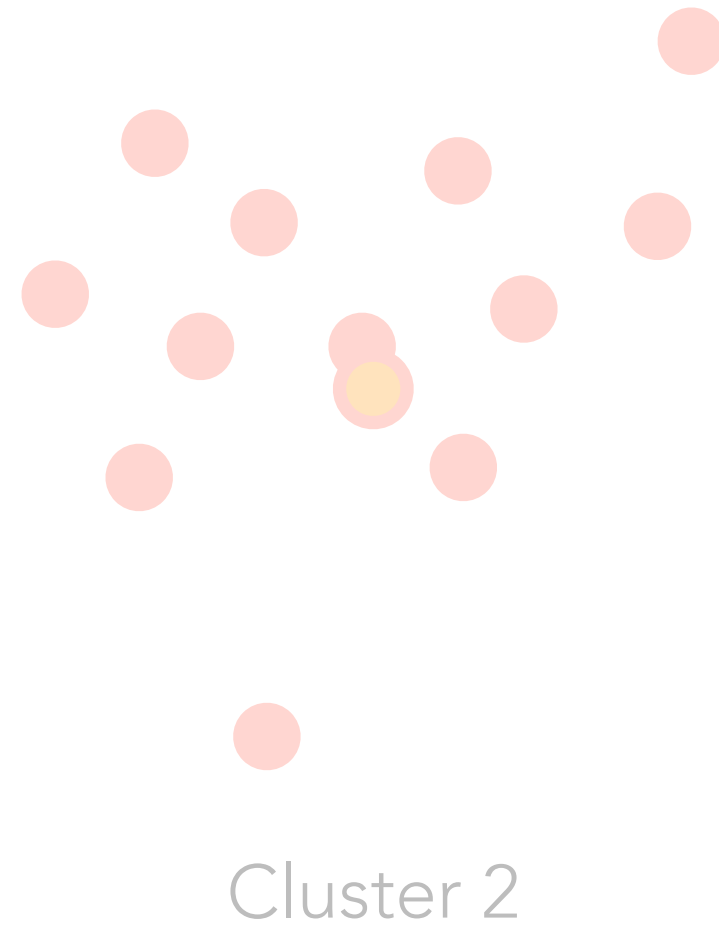
# Residual Sum of Squares

Look at one cluster at a time

Measure distance from each point to its cluster center



Cluster 1



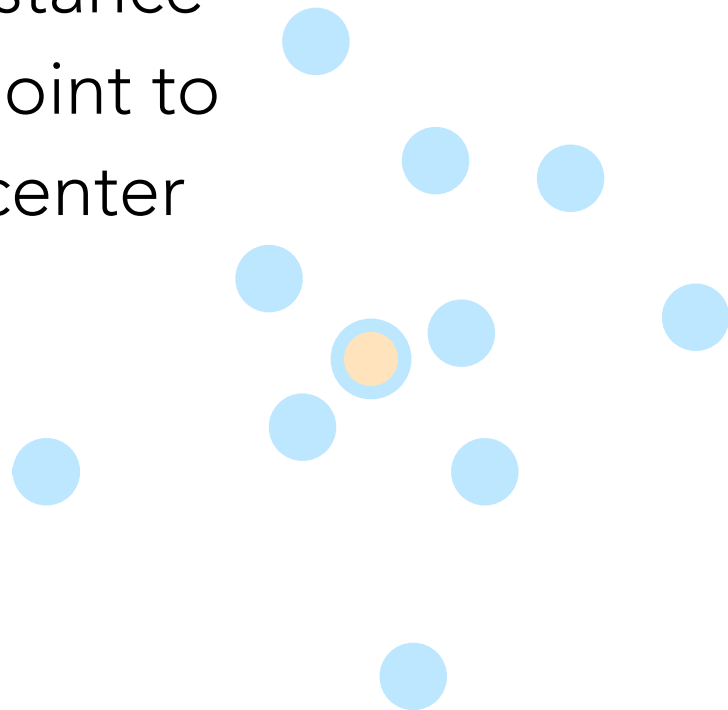
Residual sum of squares for cluster 1:

$$\text{RSS}_1 = \sum_{x \in \text{cluster 1}} \|x - \mu_1\|^2$$

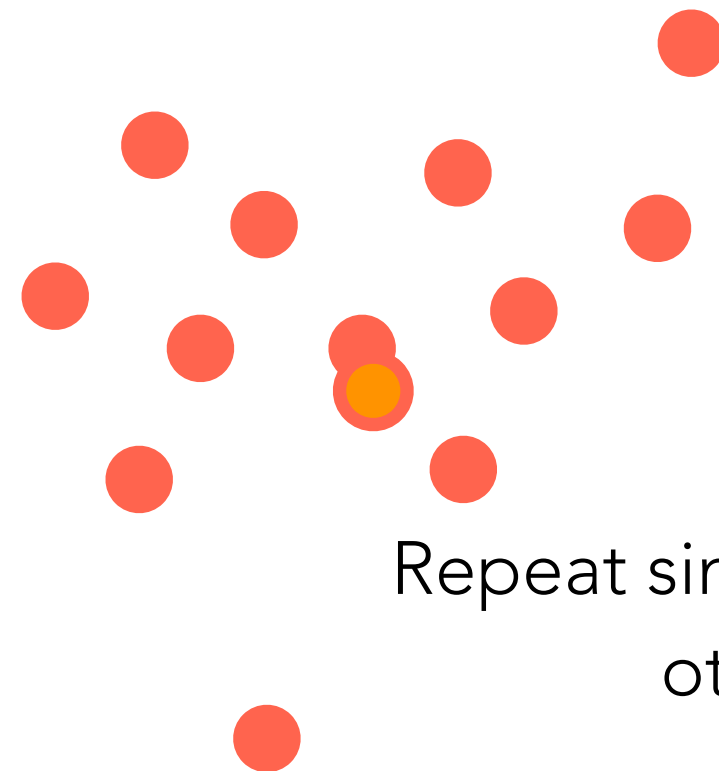
# Residual Sum of Squares

Look at one cluster at a time

Measure distance  
from each point to  
its cluster center



Cluster 1



Repeat similar calculation for  
other cluster

Cluster 2

Residual sum of squares for cluster 2:

$$\text{RSS}_2 = \sum_{x \in \text{cluster 2}} \|x - \mu_2\|^2$$

# Residual Sum of Squares

Look at one cluster at a time

$$\text{RSS} = \text{RSS}_1 + \text{RSS}_2 = \sum_{x \in \text{cluster 1}} \|x - \mu_1\|^2 + \sum_{x \in \text{cluster 2}} \|x - \mu_2\|^2$$

Measure distance  
from each point to  
its cluster center

In general if there are  $k$  clusters:

$$\text{RSS} = \sum_{g=1}^k \text{RSS}_g = \sum_{g=1}^k \sum_{x \in \text{cluster } g} \|x - \mu_g\|^2$$

Remark:  $k$ -means *tries* to minimize RSS for a fixed value of  $k$   
(it does so *approximately*, with no guarantee of optimality)

RSS does not account for clusters having, for instance, ellipse shapes

Why is minimizing RSS a bad way to choose  $k$ ?

What happens when  $k$  is equal to the number of data points?



# A Different Way to Choose $k$

RSS measures *within-cluster variation*

$$W = \text{RSS} = \sum_{g=1}^k \text{RSS}_g = \sum_{g=1}^k \sum_{x \in \text{cluster } g} \|x - \mu_g\|^2$$

Want to also measure *between-cluster variation*

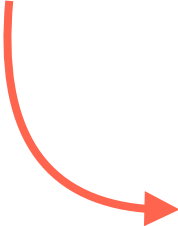
$$B = \sum_{g=1}^k (\# \text{ points in cluster } g) \|\mu_g - \mu\|^2$$

Called the CH index

[Calinski and Harabasz 1974]

mean of *all* points

A score function to use for choosing  $k$ :


$$\text{CH}(k) = \frac{B \cdot (n - k)}{W \cdot (k - 1)}$$

$n$  = total # points

Pick  $k$  with highest  $\text{CH}(k)$

(Choose  $k$  among 2, 3, ... up to pre-specified max)

# Automatically Choosing the Number of Clusters $k$

Demo