Carnegie Mellon University
Heinz College

95-865 Unstructured Data Analytics

Week 4: More clustering, topic models

George Chen

Cluster Interpretation

Demo

Automatically Choosing *k*

For k = 2, 3, … up to some user-specified max value:

Fit model using k

Compute a score for the model

But what score function should we use?

Use whichever k has the best score

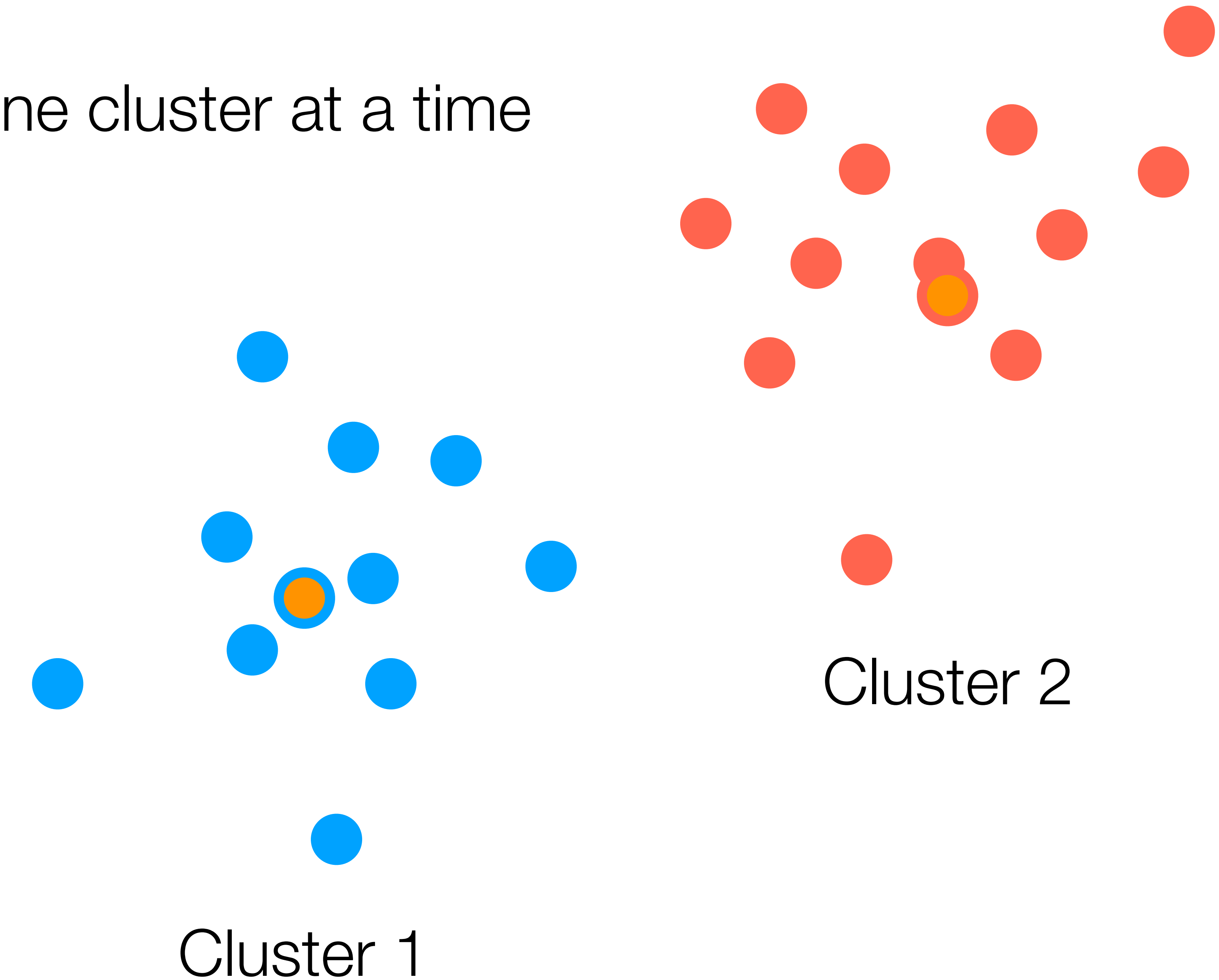
No single way of choosing k is the “best” way

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

But hey it’s worth a shot

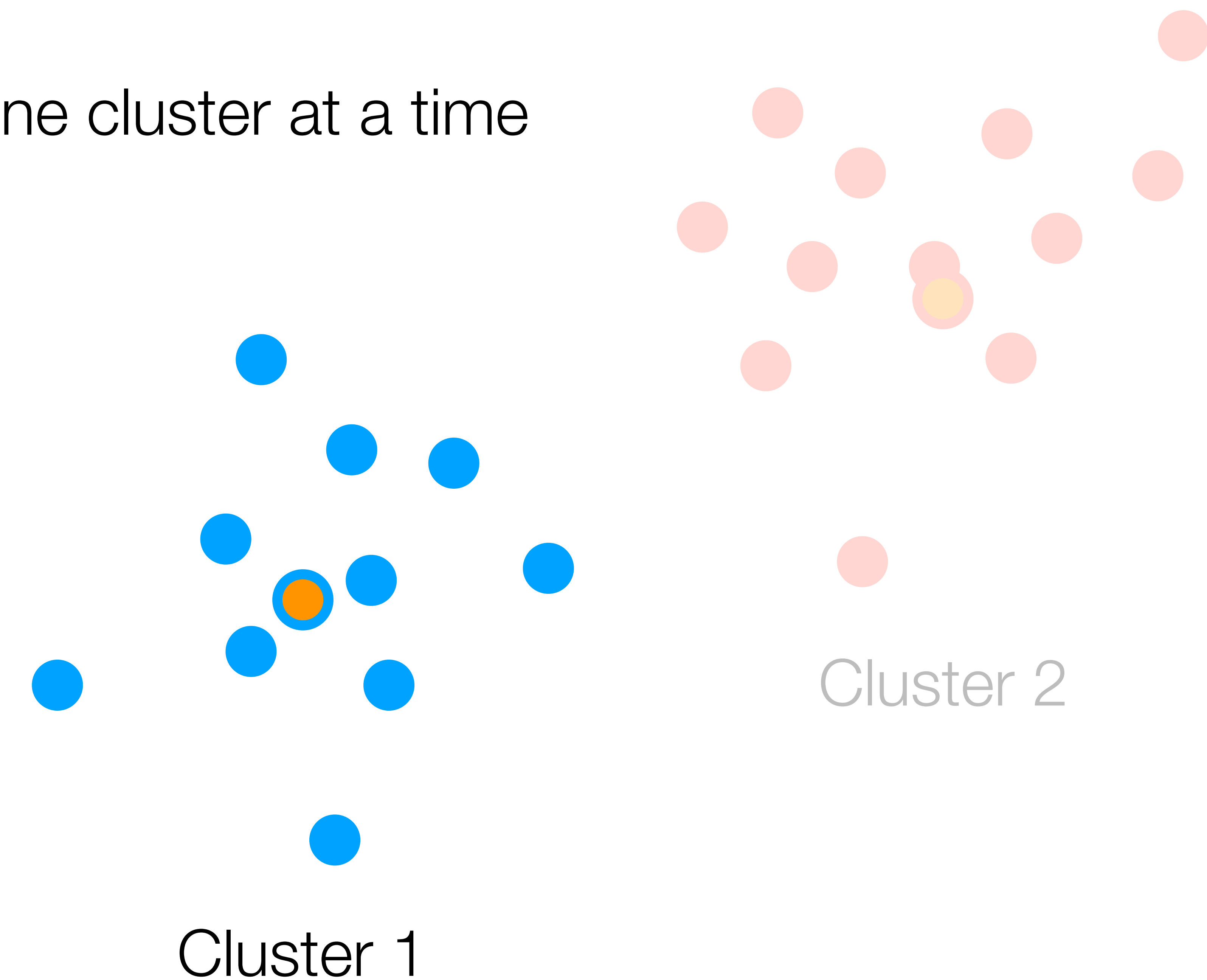
Residual Sum of Squares

Look at one cluster at a time



Residual Sum of Squares

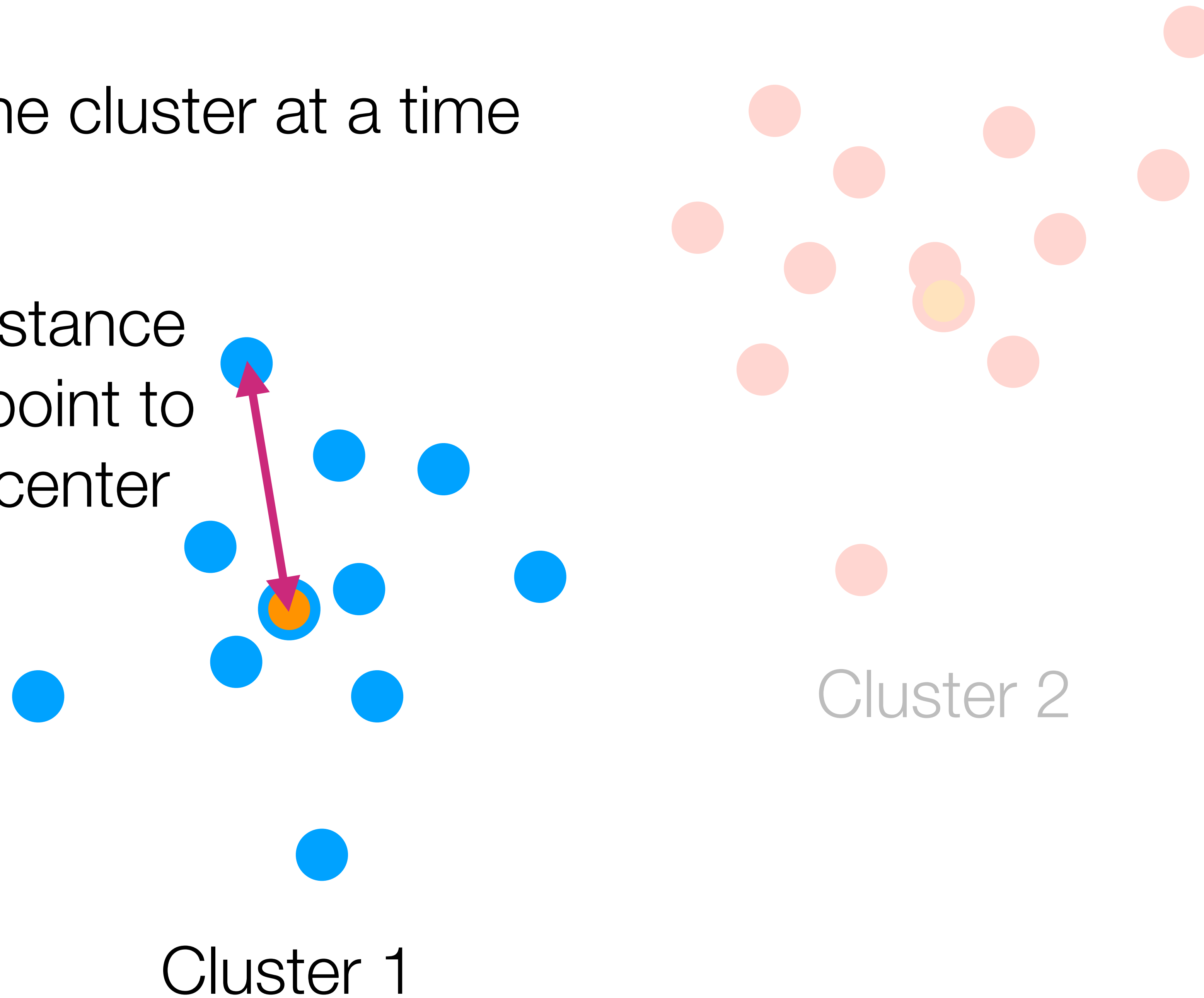
Look at one cluster at a time



Residual Sum of Squares

Look at one cluster at a time

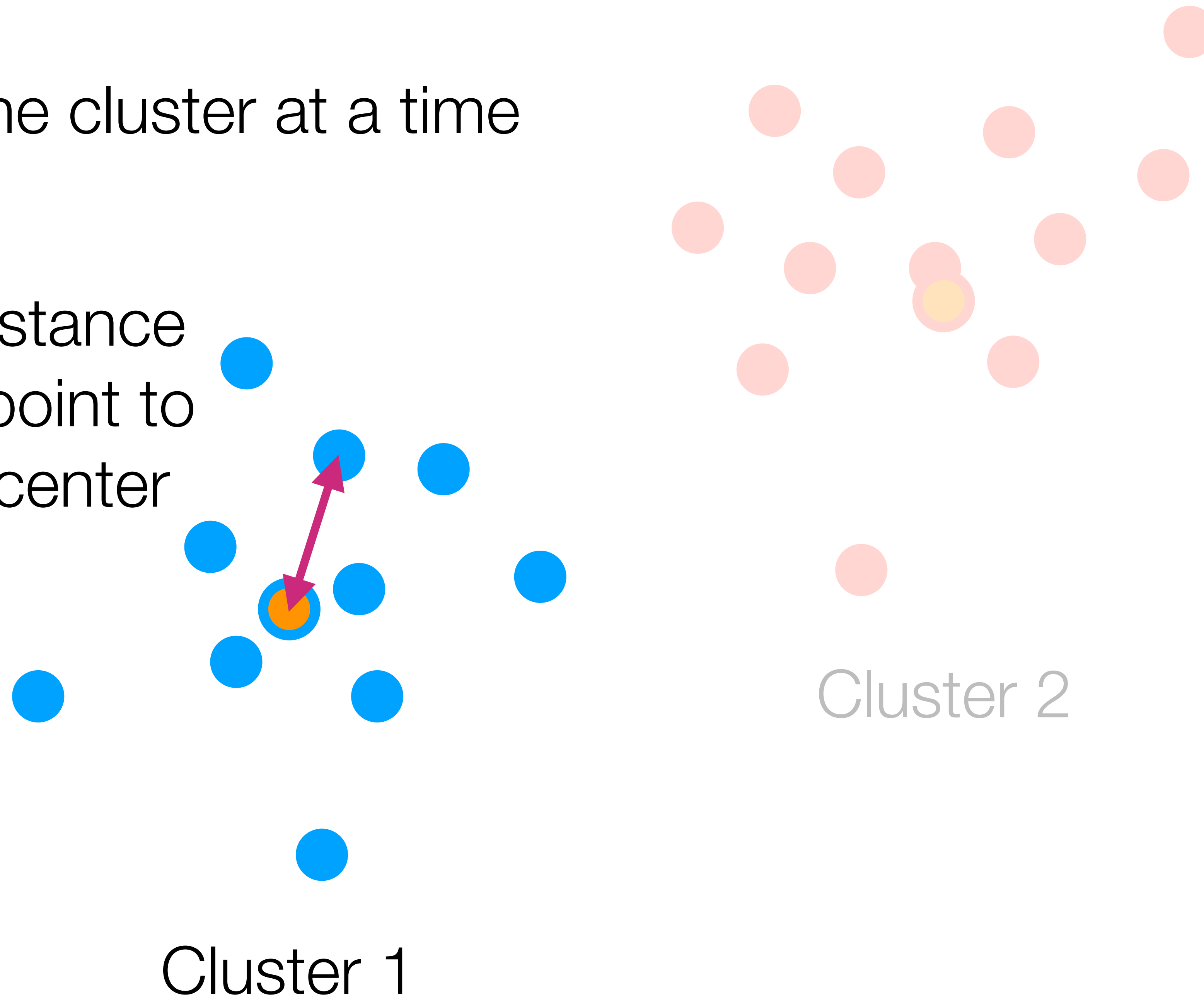
Measure distance
from each point to
its cluster center



Residual Sum of Squares

Look at one cluster at a time

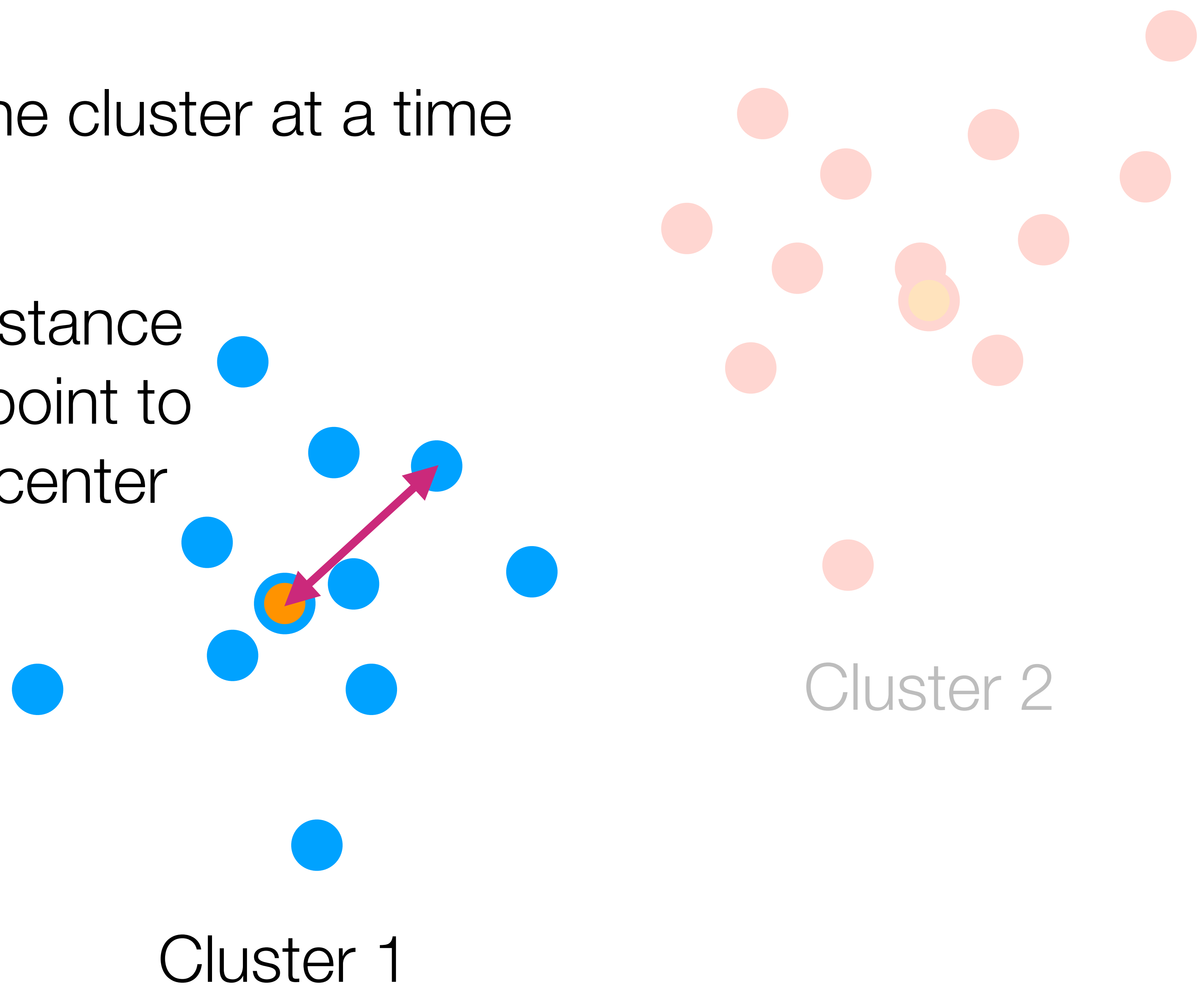
Measure distance from each point to its cluster center



Residual Sum of Squares

Look at one cluster at a time

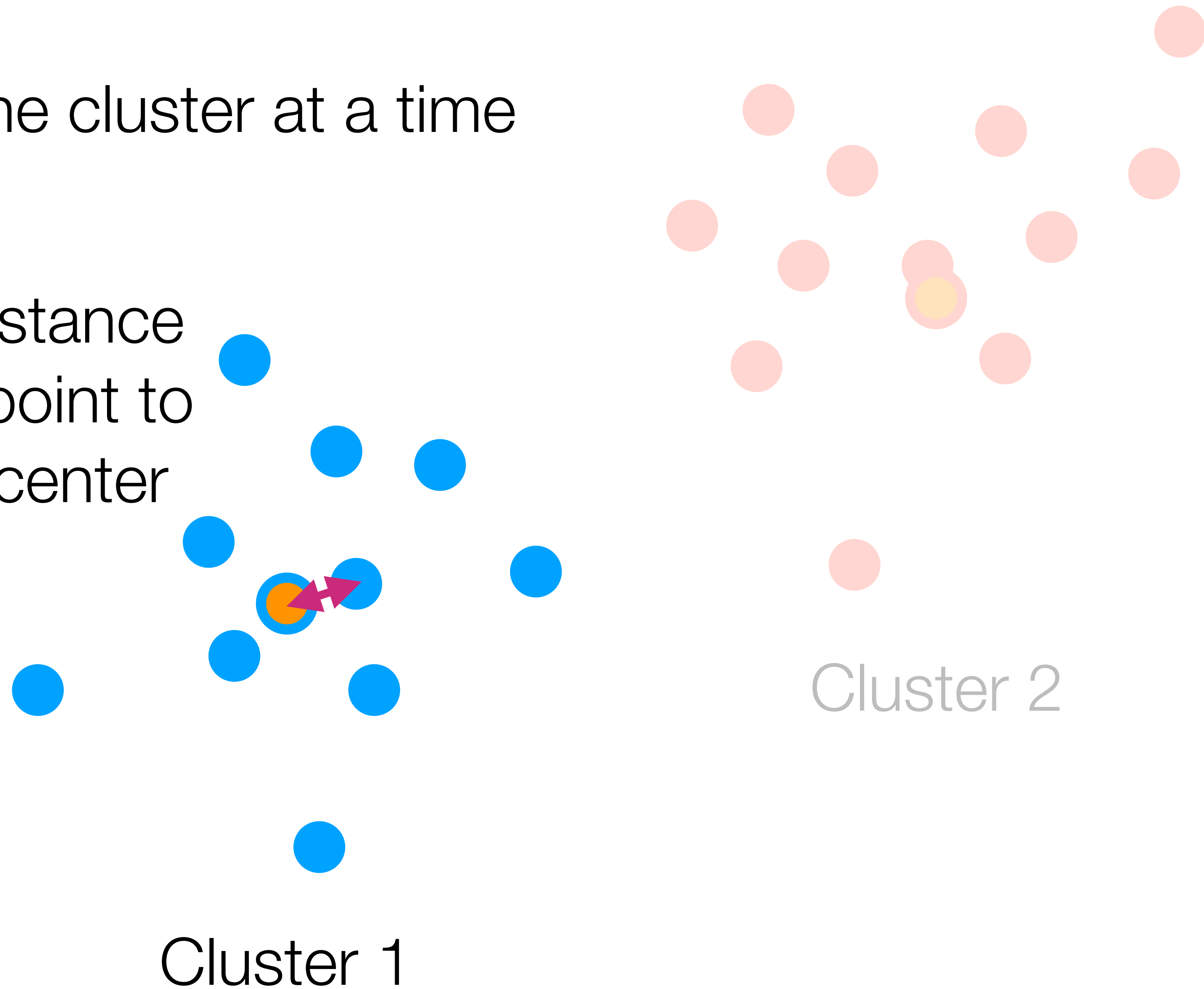
Measure distance from each point to its cluster center



Residual Sum of Squares

Look at one cluster at a time

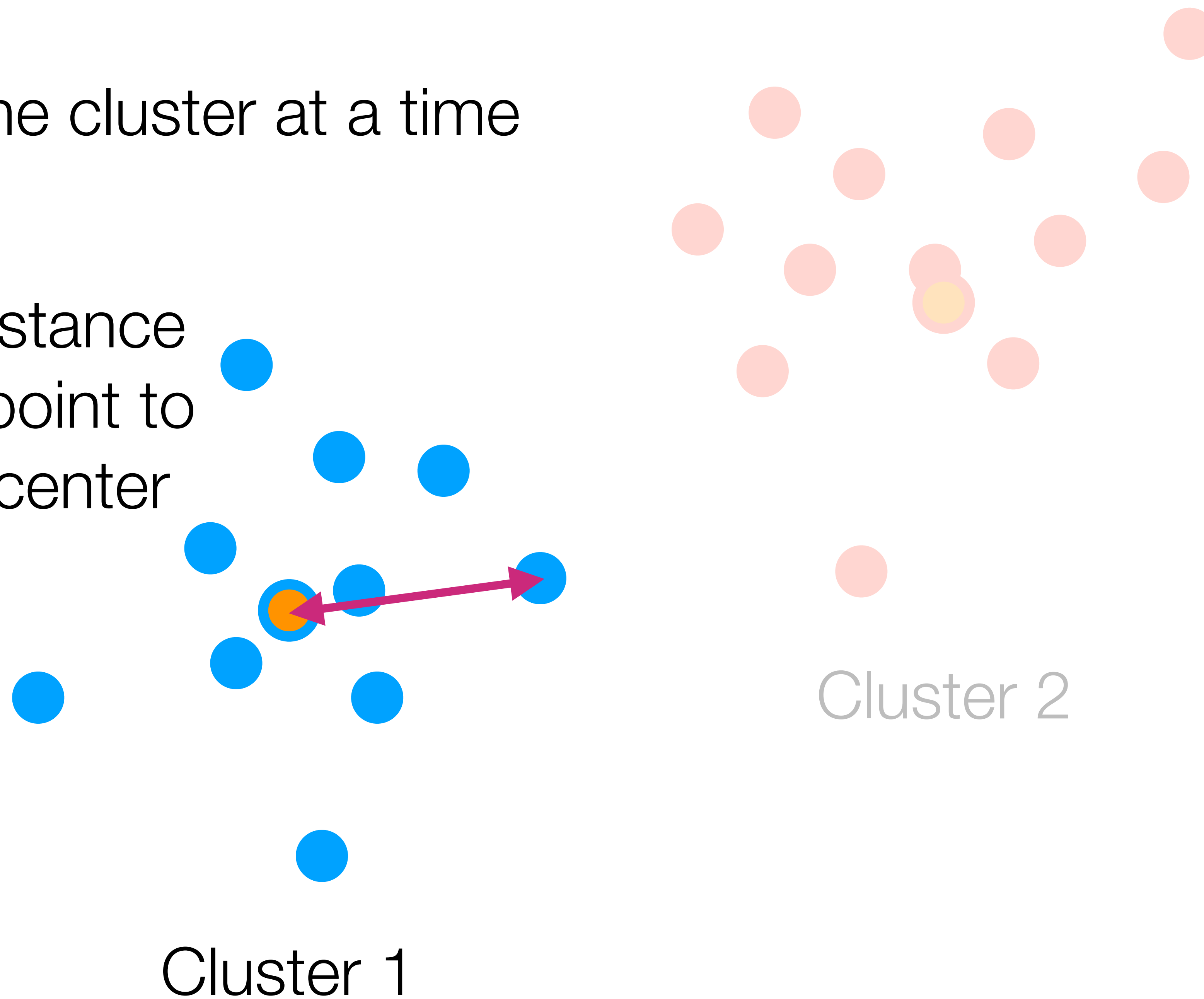
Measure distance
from each point to
its cluster center



Residual Sum of Squares

Look at one cluster at a time

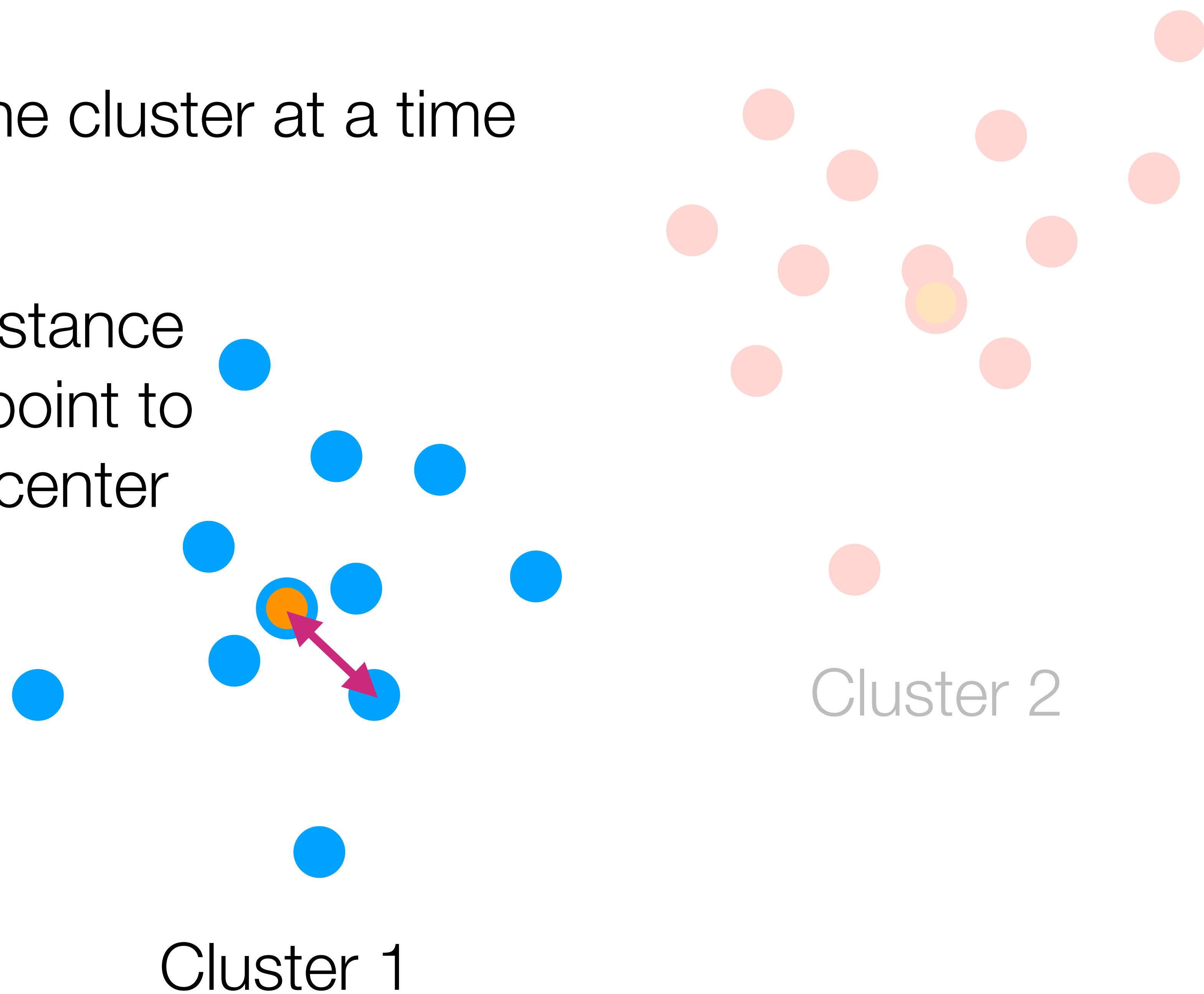
Measure distance from each point to its cluster center



Residual Sum of Squares

Look at one cluster at a time

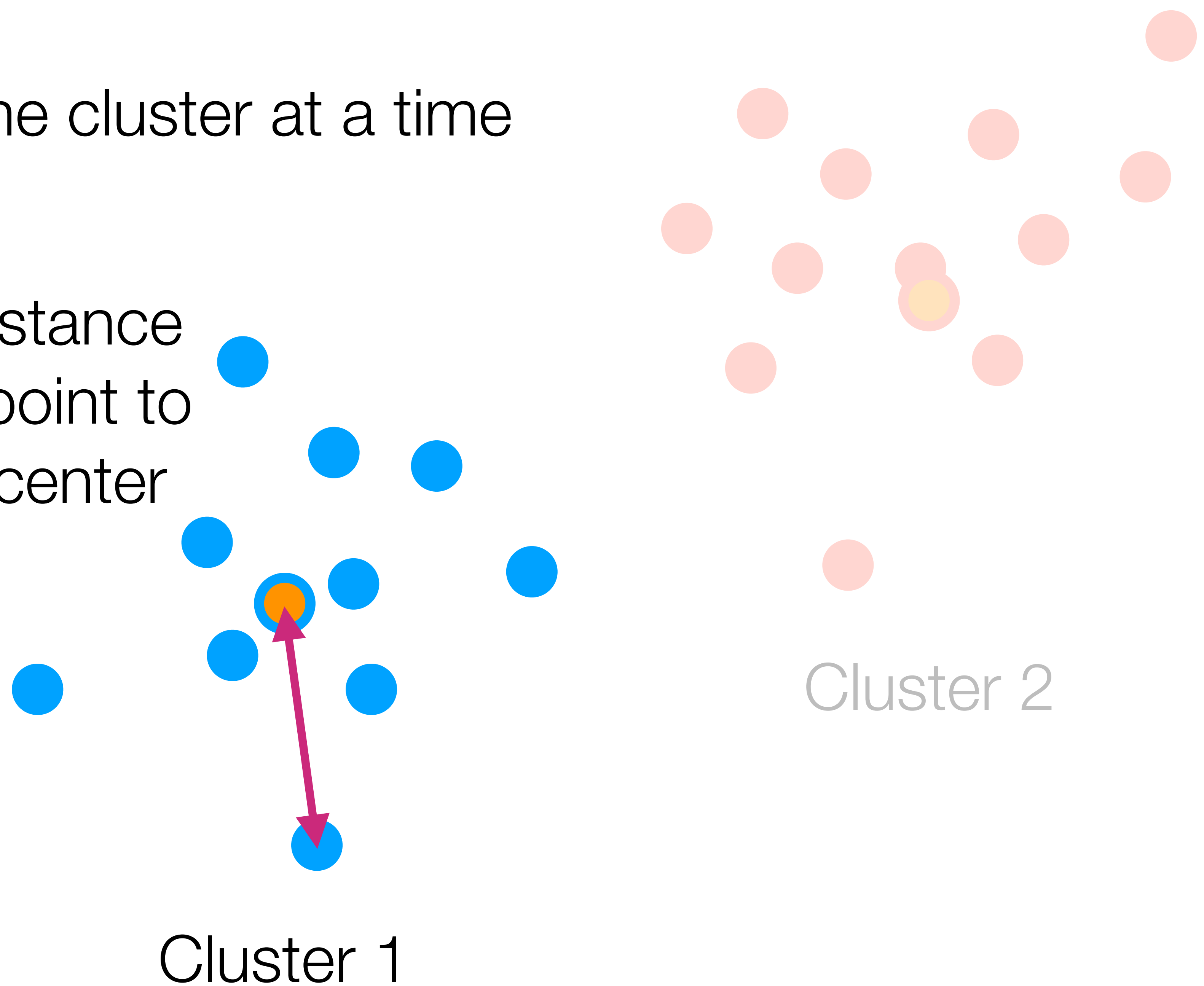
Measure distance
from each point to
its cluster center



Residual Sum of Squares

Look at one cluster at a time

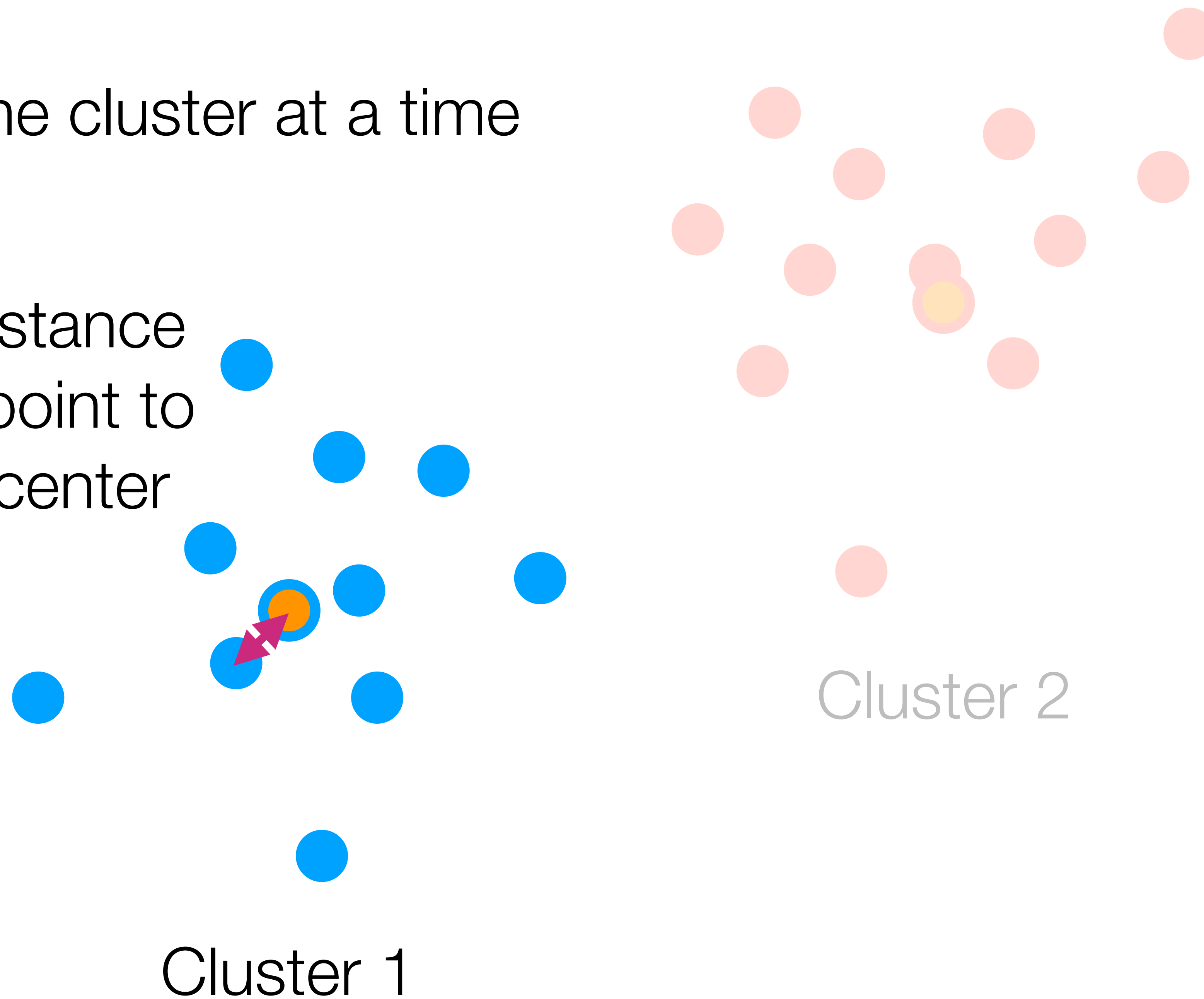
Measure distance from each point to its cluster center



Residual Sum of Squares

Look at one cluster at a time

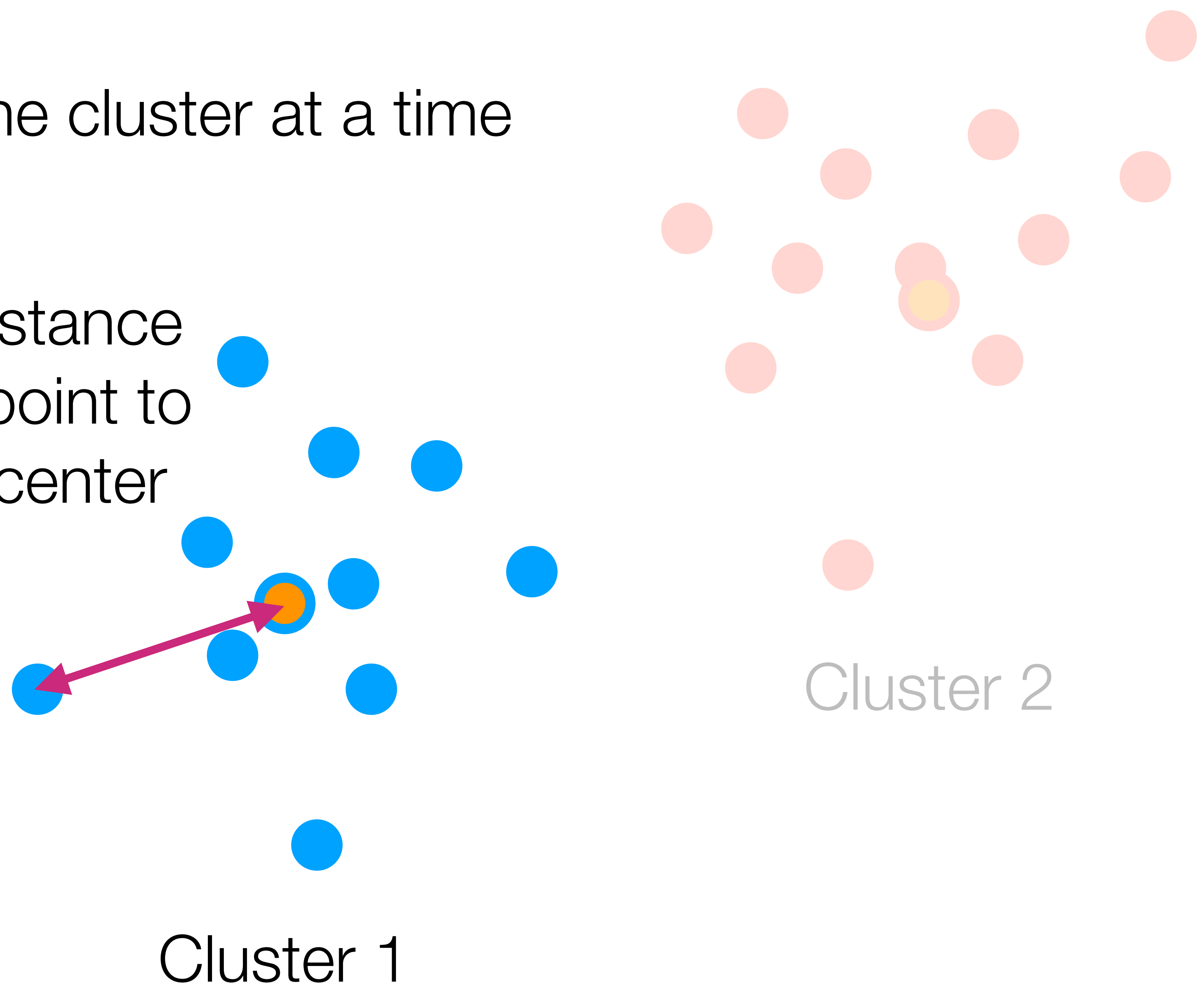
Measure distance from each point to its cluster center



Residual Sum of Squares

Look at one cluster at a time

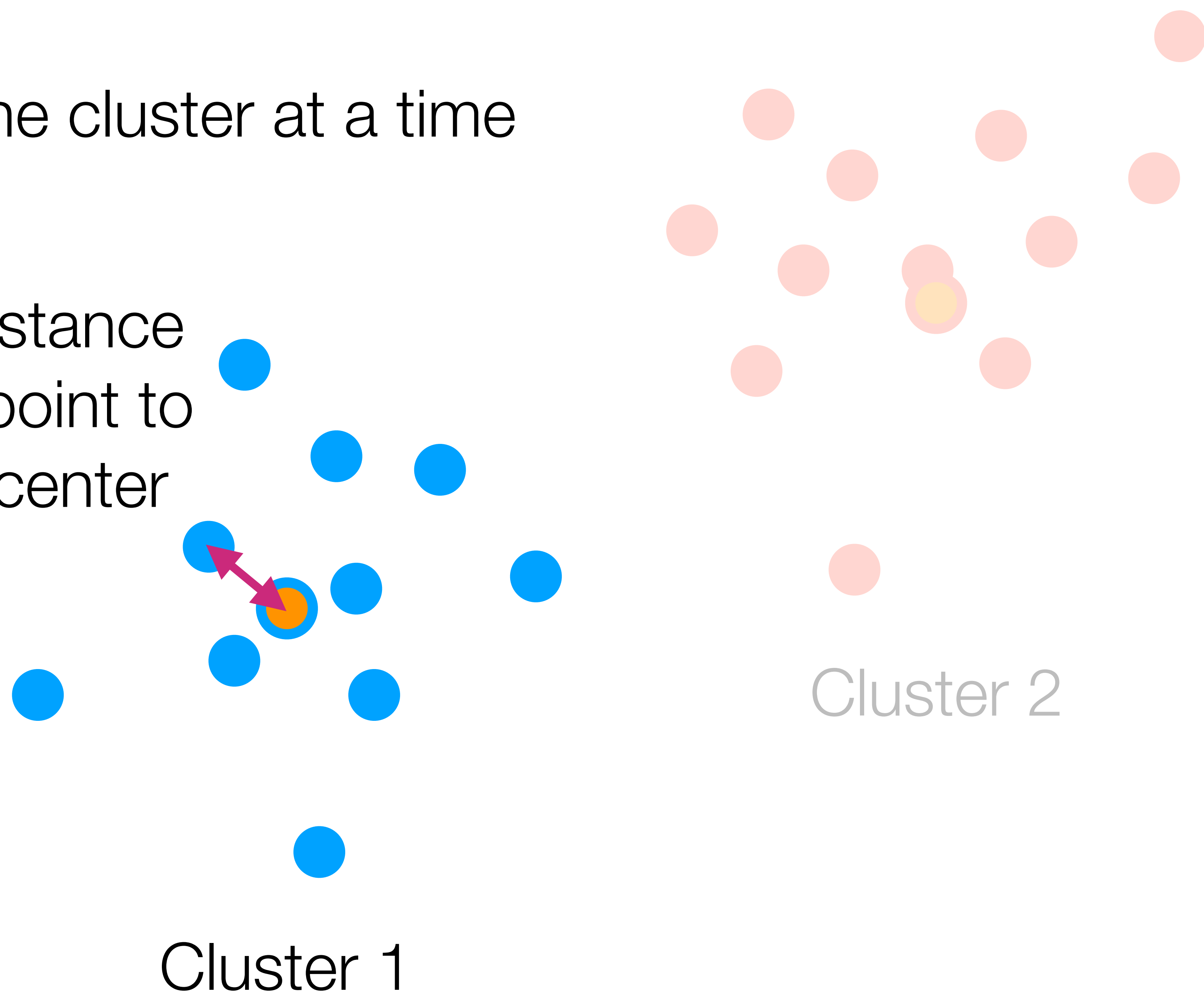
Measure distance
from each point to
its cluster center



Residual Sum of Squares

Look at one cluster at a time

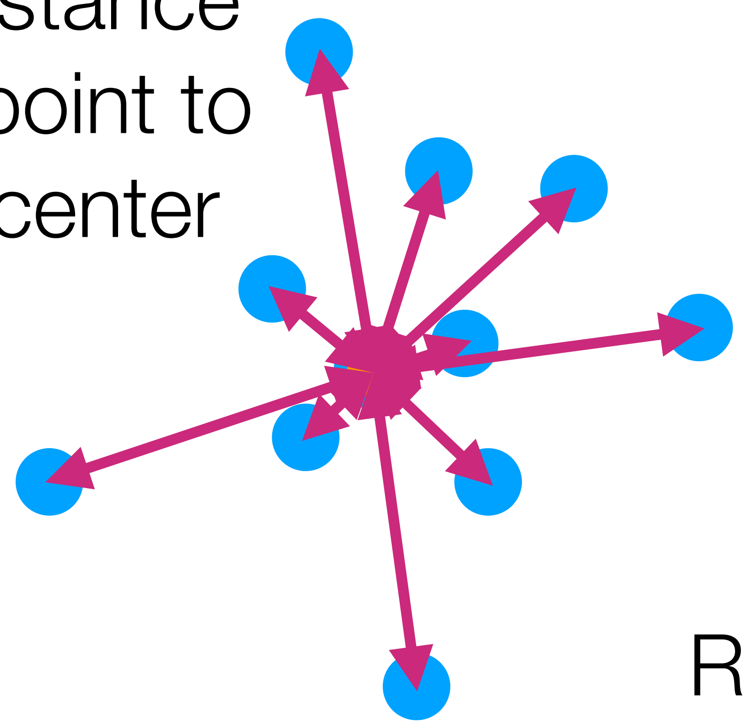
Measure distance from each point to its cluster center



Residual Sum of Squares

Look at one cluster at a time

Measure distance from each point to its cluster center

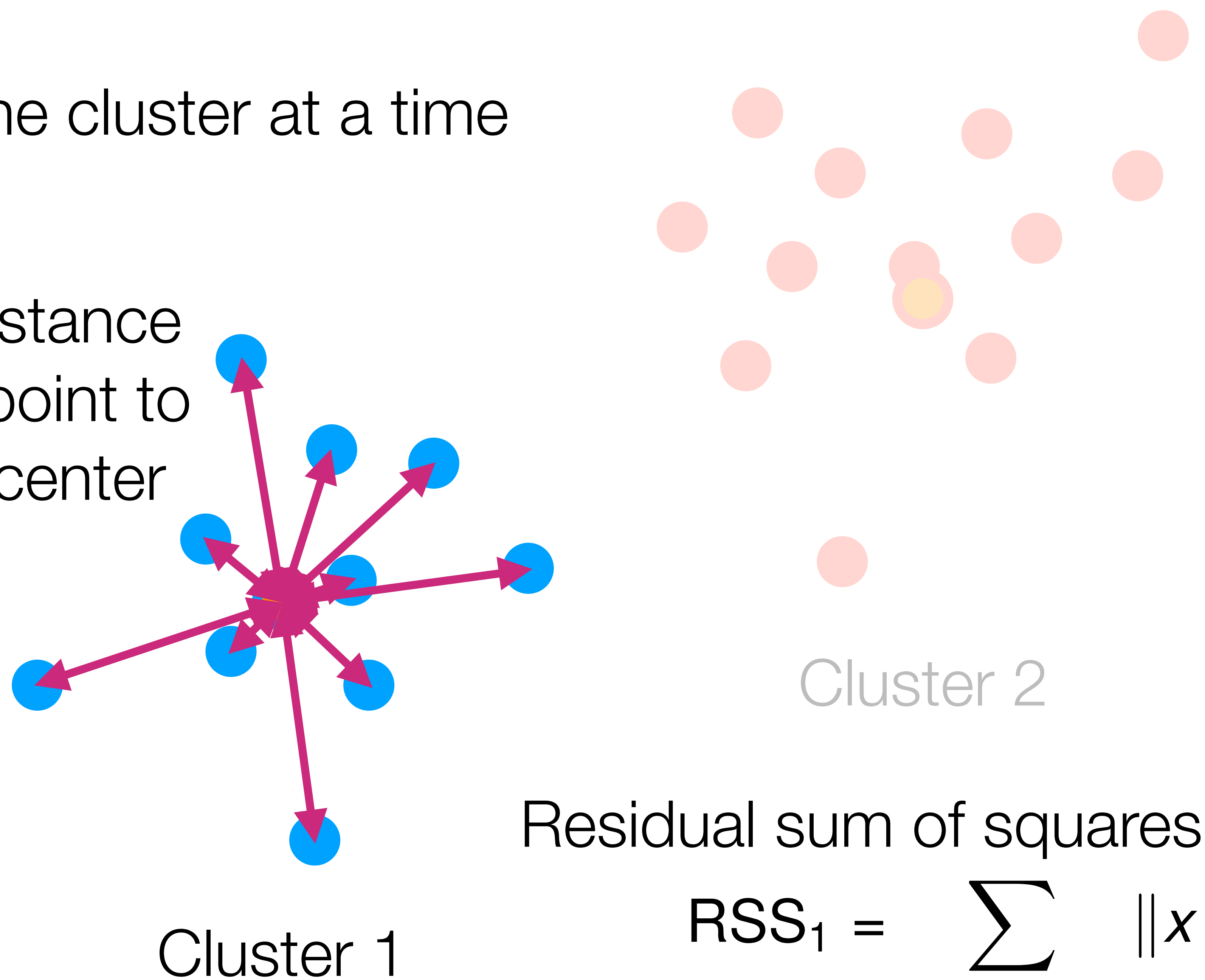


Residual sum of squares for cluster 1: sum of *squared* purple lengths

Residual Sum of Squares

Look at one cluster at a time

Measure distance from each point to its cluster center



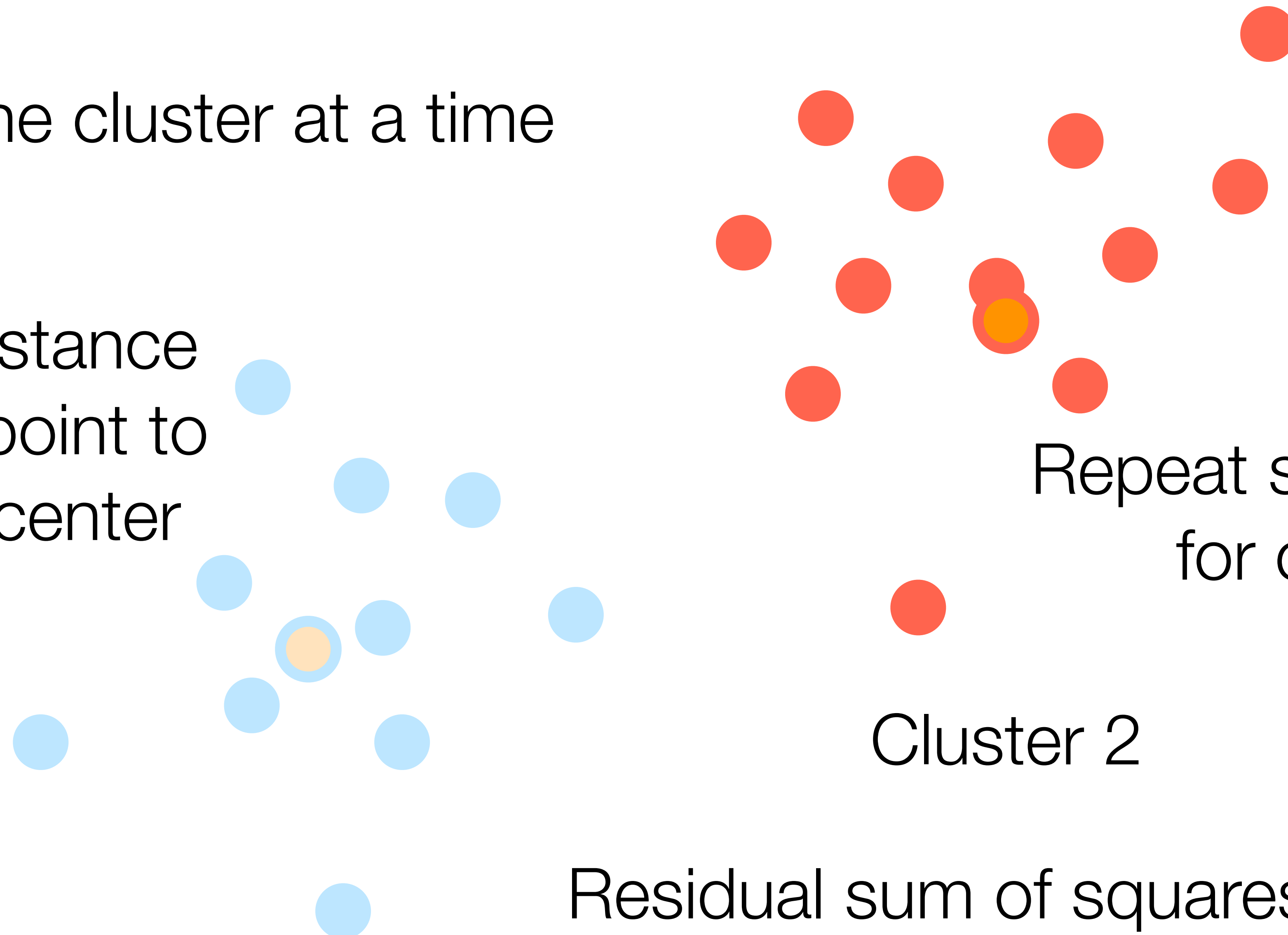
Residual sum of squares for cluster 1:

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

Residual Sum of Squares

Look at one cluster at a time

Measure distance from each point to its cluster center



Repeat similar calculation for other cluster

Cluster 2

Cluster 1

Residual sum of squares for cluster 2:

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

Residual Sum of Squares

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

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
(it does so *approximately,* with no guarantee of optimality)

RSS only really makes sense for clusters that look like circles

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

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

A Good 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*

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

mean of *all* points

Called the **CH index**
[Calinski and Harabasz 1974]

A good 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 **CH(*k*)**

(Choose *k* among 2, 3, … up to pre-specified max)

Automatically Choosing k

Demo

Going from Similarities to Clusters

There's a whole zoo of clustering methods

Two main categories we'll talk about:

Generative models

1. Pretend data generated by specific model with parameters

2. Learn the parameters ("fit model to data")

3. Use fitted model to determine cluster assignments

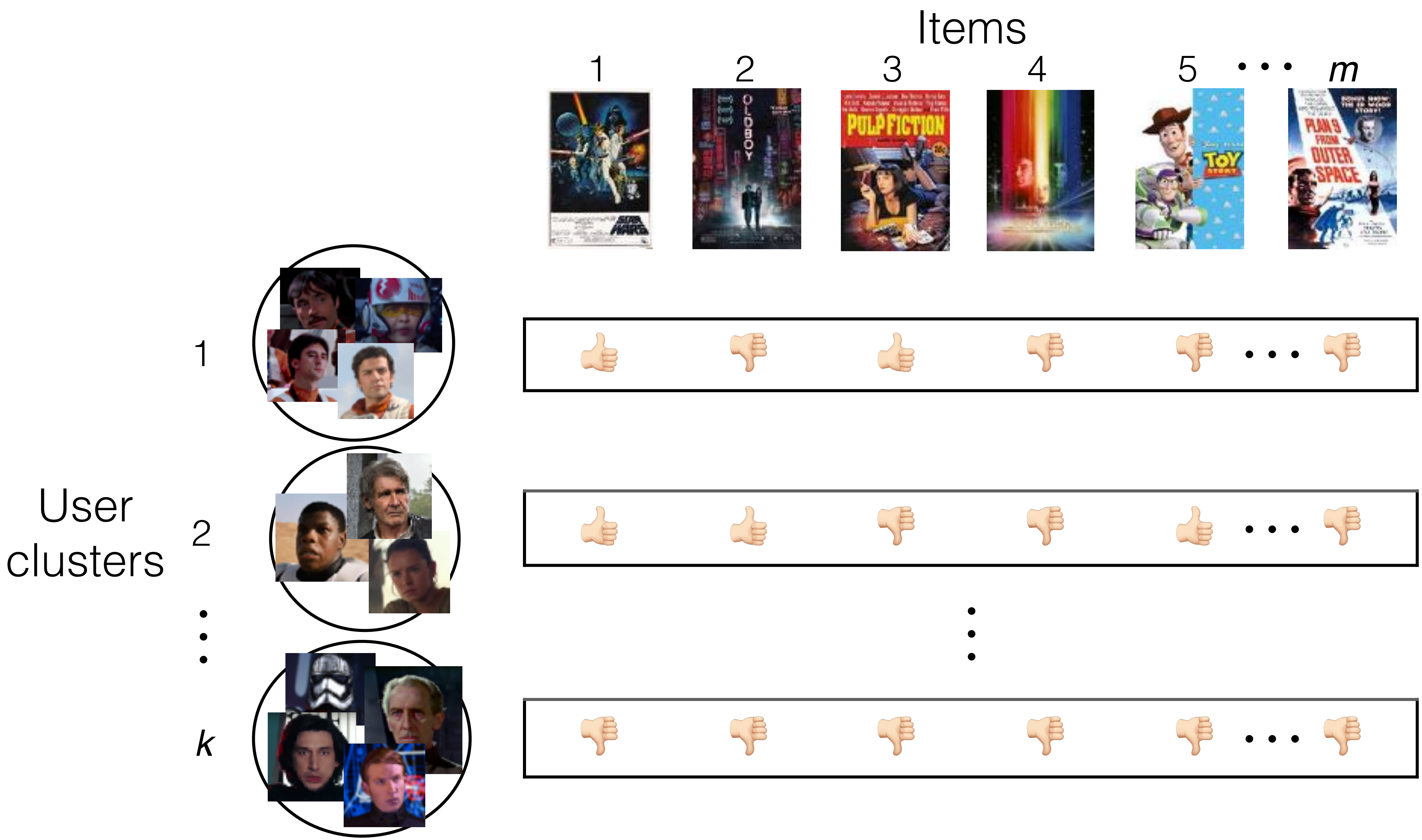
Hierarchical clustering

Top-down: Start with everything in 1 cluster and decide on how to recursively split

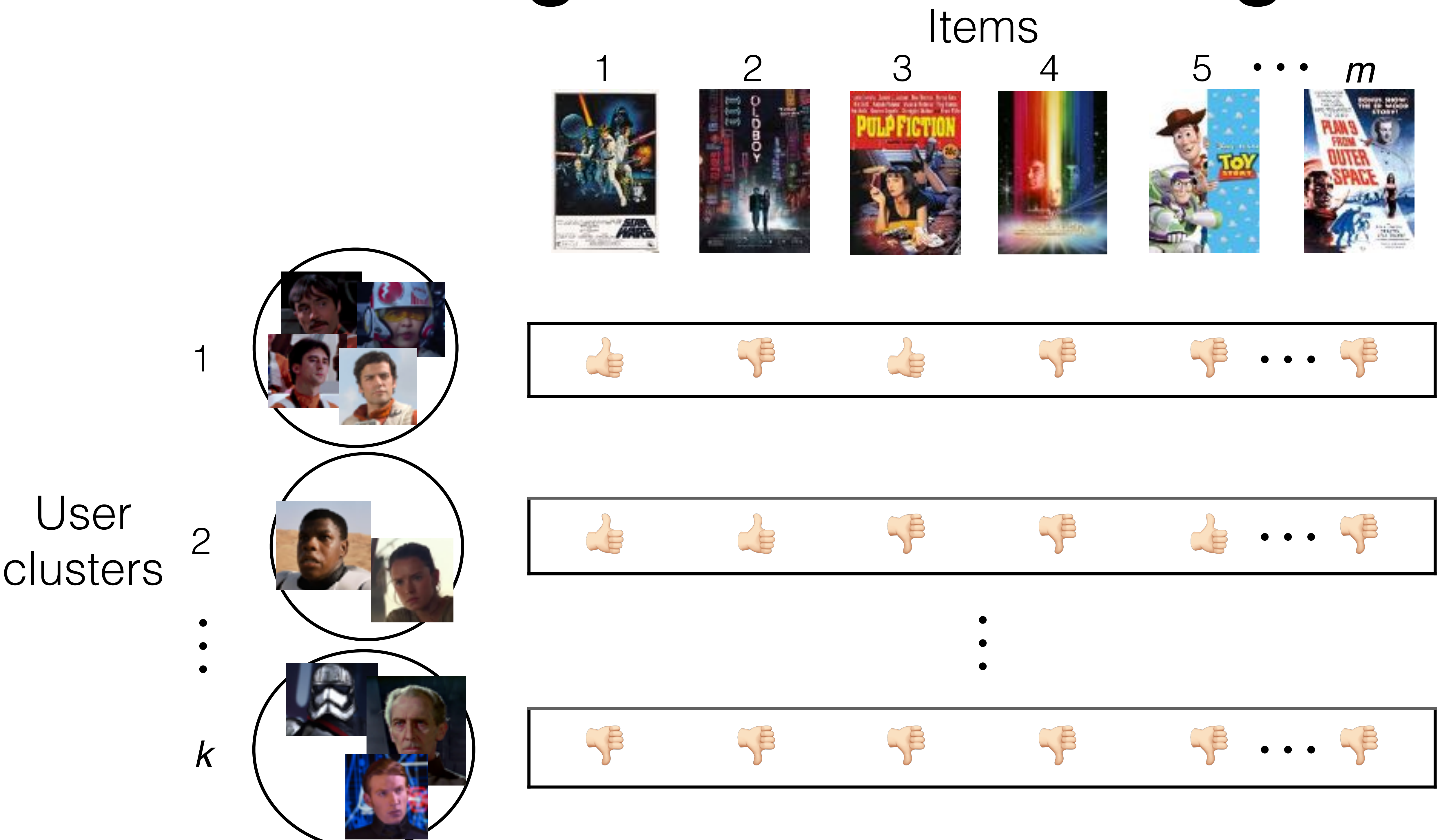
Bottom-up: Start with everything in its own cluster and decide on how to iteratively merge clusters

Is Clustering Structure Enough?

Is Clustering Structure Enough?

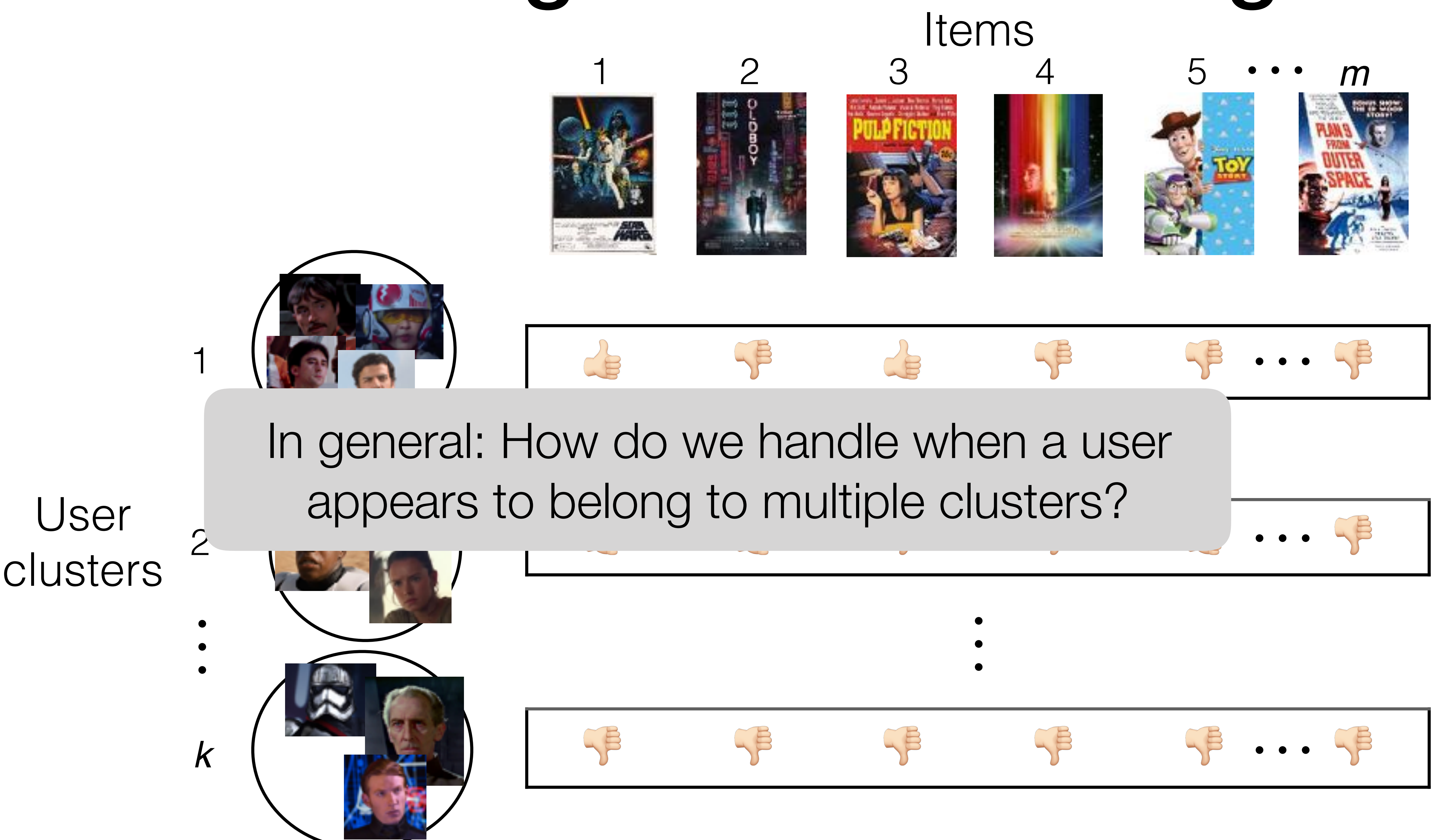


Is Clustering Structure Enough?

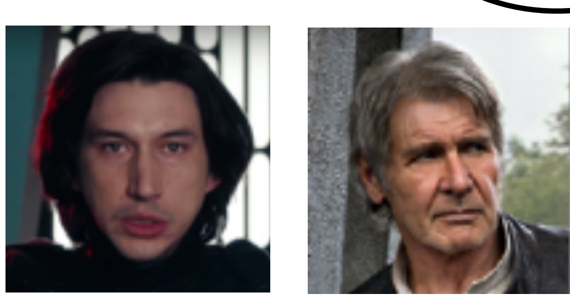


What if these two users shared a Netflix account (and used the same user profile)?

Is Clustering Structure Enough?



In general: How do we handle when a user appears to belong to multiple clusters?



What if these two users shared a Netflix account (and used the same user profile)?

Topic Modeling

Movie recommendation

Each user is part of multiple “clusters”/topics

Each cluster/topic consists of a bunch of movies
(example clusters: “sci-fi epics”, “cheesy rom-coms”)

Text

Each document is part of multiple topics

Each topic consists of a bunch of regularly co-occurring words
(example topics: “sports”, “medicine”, “movies”, “finance”)

Health care

Each patient’s health records explained by multiple “topics”

Each topic consists of co-occurring “events”
(example topics: “heart condition”, “severe pancreatitis”)

Topic Modeling

Movie recommendation

Each user is part of multiple “clusters”/topics

Each cluster/topic consists of a bunch of movies
(ex s”)

In all of these examples:

- Each data point (a feature vector) is part of multiple topics
- Each topic corresponds to specific feature values in the feature vector likely appearing

Each to words
(exar nce”)

Health care

Each patient’s health records explained by multiple “topics”

Each topic consists of co-occurring “events”
(example topics: “heart condition”, “severe pancreatitis”)

Latent Dirichlet Allocation (LDA)

- Easy to describe in terms of text (but works for not just text)
- A generative model
- Input: “document-word” matrix, and pre-specified # topics k

i-th row, j-th column: # times word j appears in doc i

- Output: what the k topics are (details on this shortly)

LDA Example

		Document	
		Alice's text	Bob's text
Topic	weather	0.1	0.5
	food	0.9	0.5

		Topic	
		weather	food
Word	cold	0.3	0.1
	hot	0.7	0.3
	apple	0.0	0.5
	pie	0.0	0.1

Each word in Alice's text is generated by:

1. Flip 2-sided coin for Alice
2. If weather: flip 4-sided coin for weather
   If food: flip 4-sided coin for food

LDA Example

		Document	
		Alice's text	Bob's text
Topic	weather	0.1	0.5
	food	0.9	0.5

		Topic	
		weather	food
Word	cold	0.3	0.1
	hot	0.7	0.3
	apple	0.0	0.5
	pie	0.0	0.1

Each word in Bob's text is generated by:

1. Flip 2-sided coin for Bob
2. If weather: flip 4-sided coin for weather
   If food: flip 4-sided coin for food

LDA Example

		Document	
		Alice's text	Bob's text
Topic	weather	0.1	0.5
	food	0.9	0.5

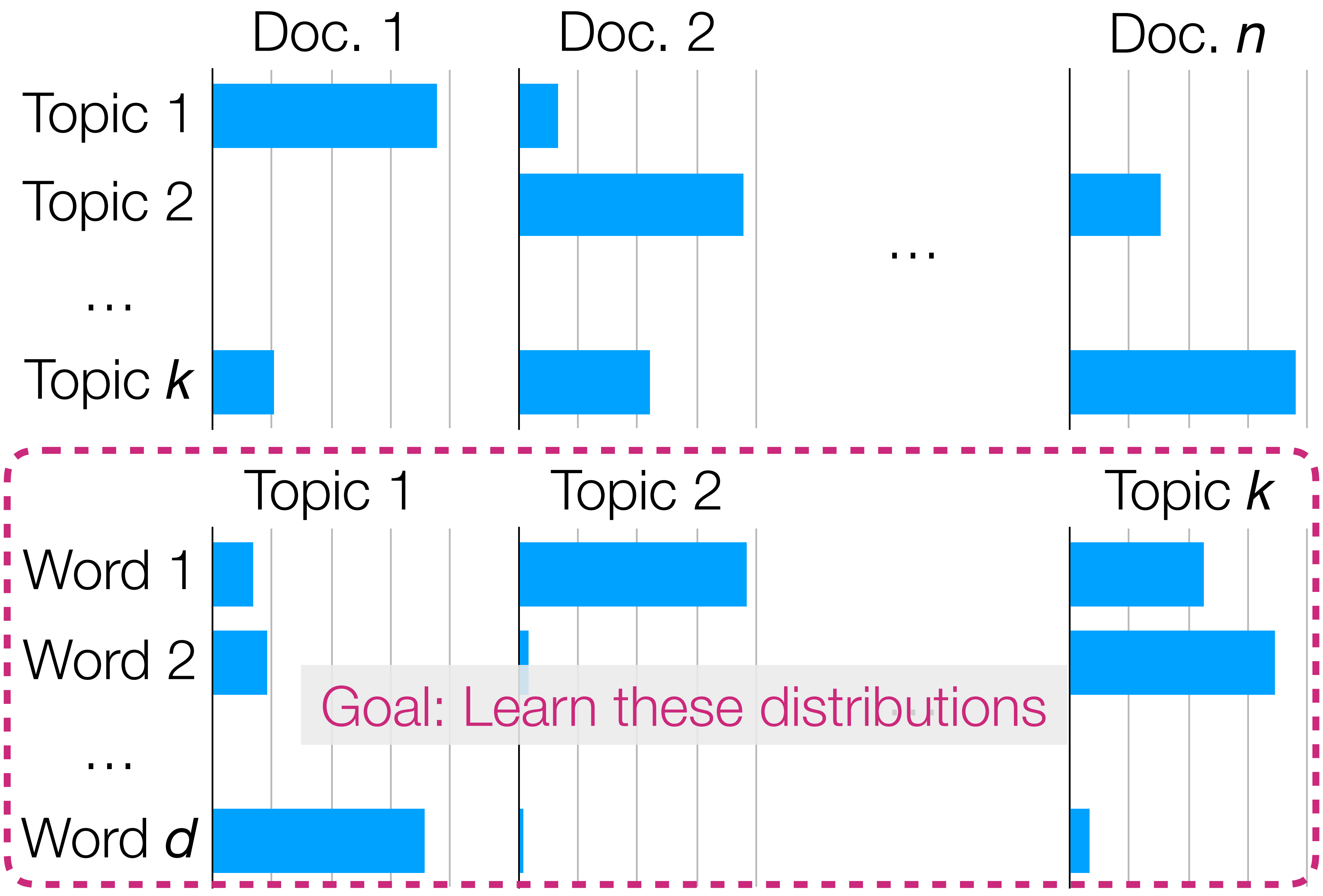
		Topic	
		weather	food
Word	cold	0.3	0.1
	hot	0.7	0.3
	apple	0.0	0.5
	pie	0.0	0.1

Each word in doc. *i* is generated by:

1. Flip 2-sided coin for doc. *i*
2. If weather: flip 4-sided coin for weather
   If food: flip 4-sided coin for food

"Learning the topics" means figuring out these 4-sided coin probabilities

LDA



LDA models each word in document *i* to be generated as:

- Randomly choose a topic *Z* (use topic distribution for doc *i*)
- Randomly choose a word (use word distribution for topic *Z*)

LDA

- Easy to describe in terms of text (but works for not just text)
- A generative model
- Input: “document-word” matrix, and pre-specified # topics *k*

i-th row, *j*-th column: # times word *j* appears in doc *i*

- Output: the *k* topics’ distribution of words

LDA

Demo

How to Choose Number of Topics *k*?

Something like CH index is also possible:

avoid numerical issues

For a specific topic, look at the *m* most probable words ("top words")

Coherence (within cluster/topic variability):

$$\sum_{\substack{\text{top words } v,w \\ \text{that are not the same}}} \log \frac{\#\text{ documents that contain both } v \text{ and } w + 0.1}{\#\text{ documents that contain } w}$$

log of P(see word v | see word w)

Inter-topic similarity (between cluster/topic variability):

Can average each of these across the topics

Count # top words that do not appear in any of the other topics' *m* top words

(number of "unique words")

Topic Modeling: Last Remarks

- There are actually *many* topic models, not just LDA
    - Correlated topic models, Pachinko allocation, biterm topic models, anchor word topic models, …
- Dynamic topic models: tracks how topics change *over time*
    - Example: for text over time, figure out how topics change
    - Example: for recommendation system, figure out how user tastes change over time

Now…back to clustering

Going from Similarities to Clusters

There's a whole zoo of clustering methods

Two main categories we'll talk about:

Generative models

1. Pretend data generated by specific model with parameters

2. Learn the parameters ("fit model to data")

3. Use fitted model to determine cluster assignments

Hierarchical clustering

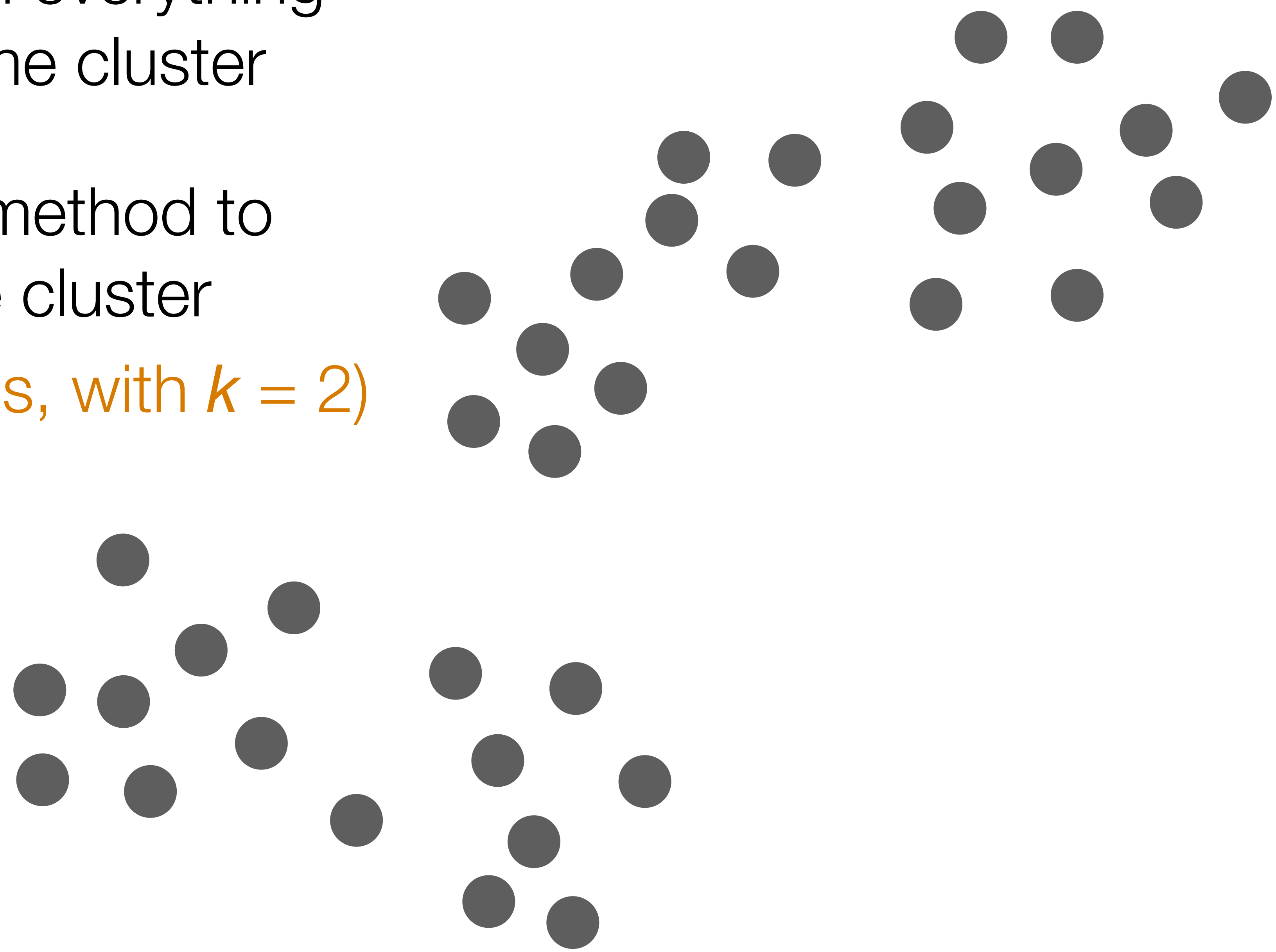
Top-down: Start with everything in 1 cluster and decide on how to recursively split

Bottom-up: Start with everything in its own cluster and decide on how to iteratively merge clusters

Top-down: Divisive Clustering

0. Start with everything in the same cluster

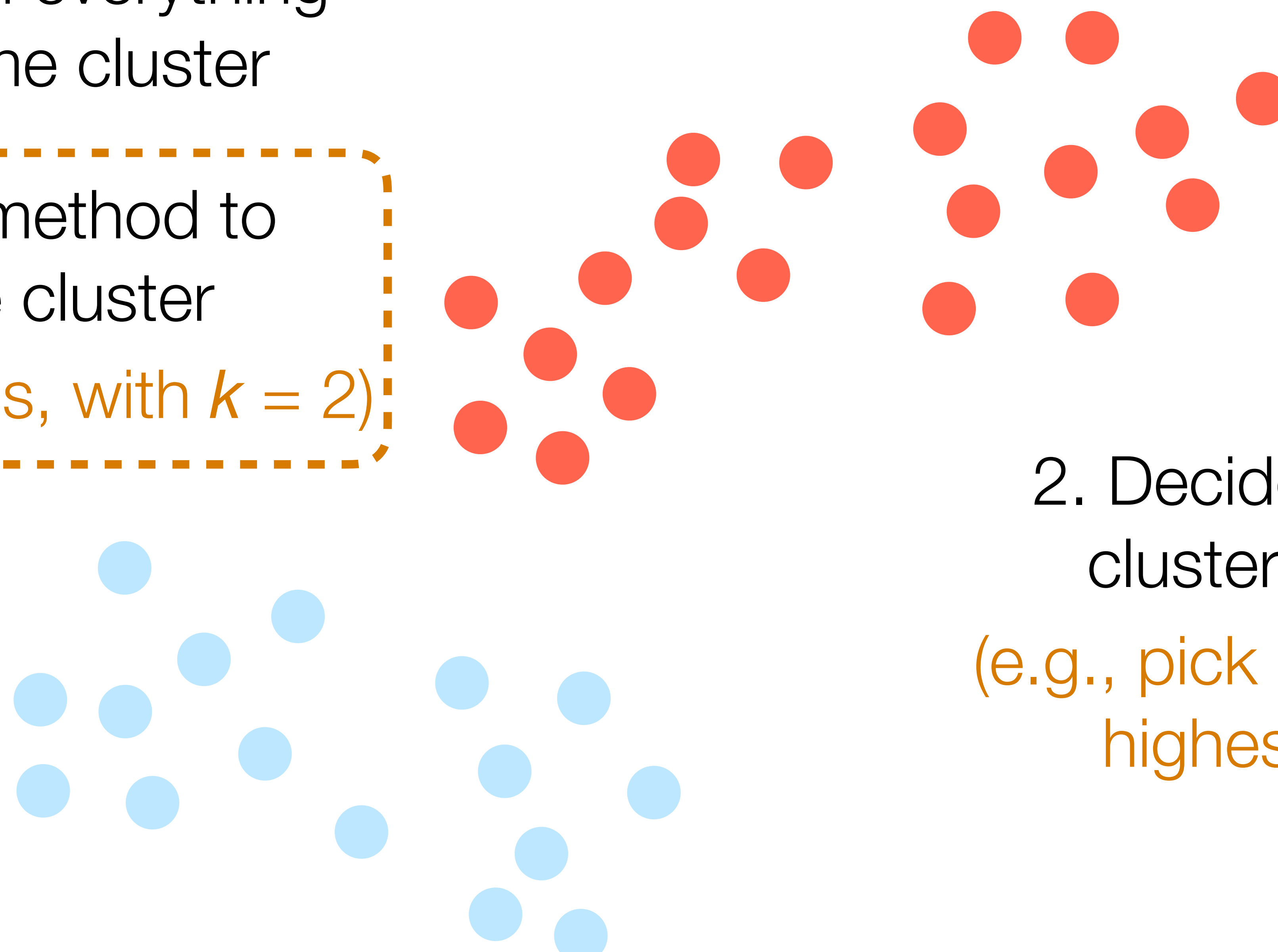
1. Use a method to split the cluster (e.g., k-means, with $k = 2$)



Top-down: Divisive Clustering

0. Start with everything in the same cluster

1. Use a method to split the cluster (e.g., *k*-means, with $k = 2$)

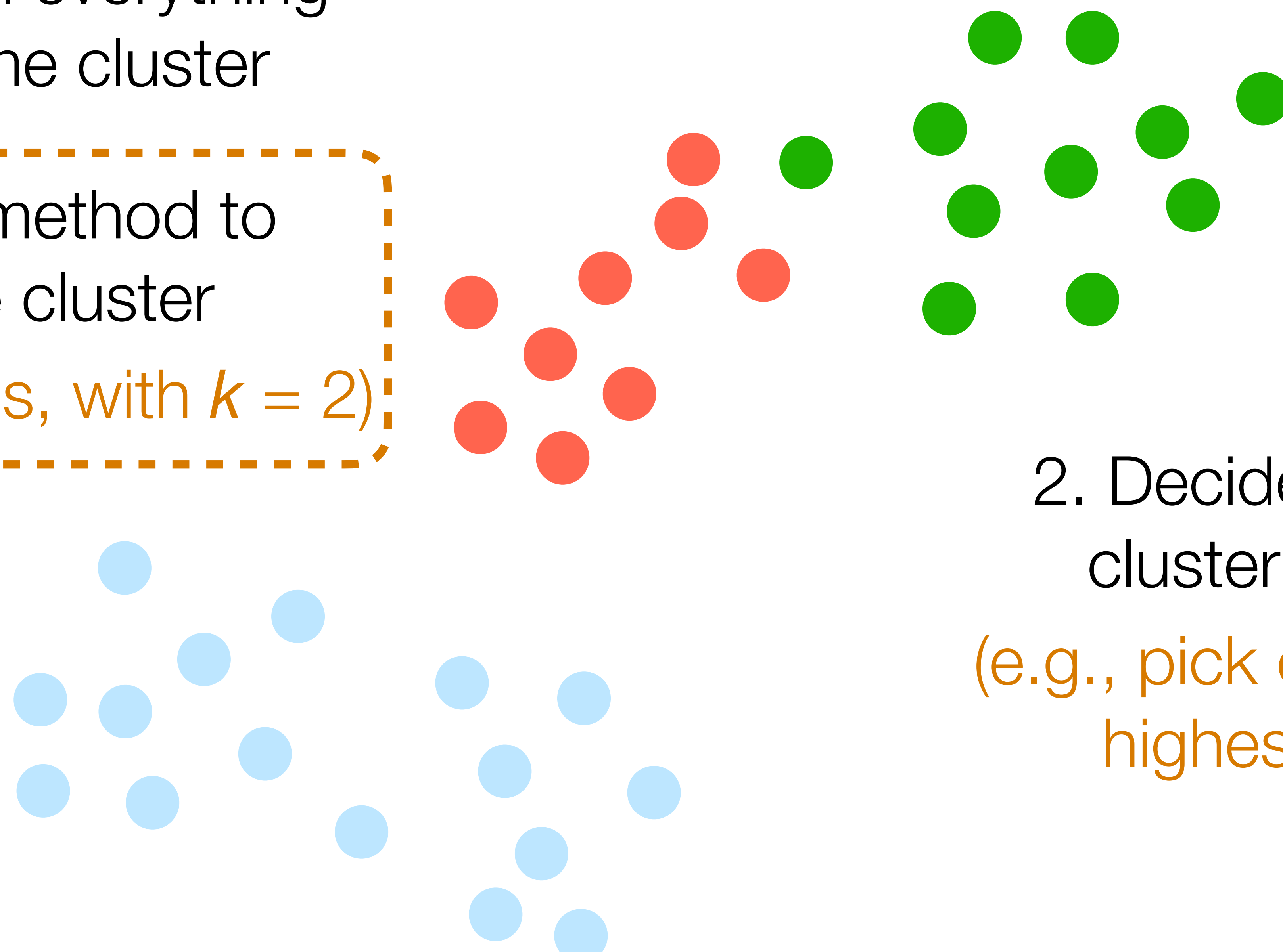


2. Decide on next cluster to split (e.g., pick cluster with highest RSS)

Top-down: Divisive Clustering

0. Start with everything in the same cluster

1. Use a method to split the cluster
(e.g., *k*-means, with k = 2)



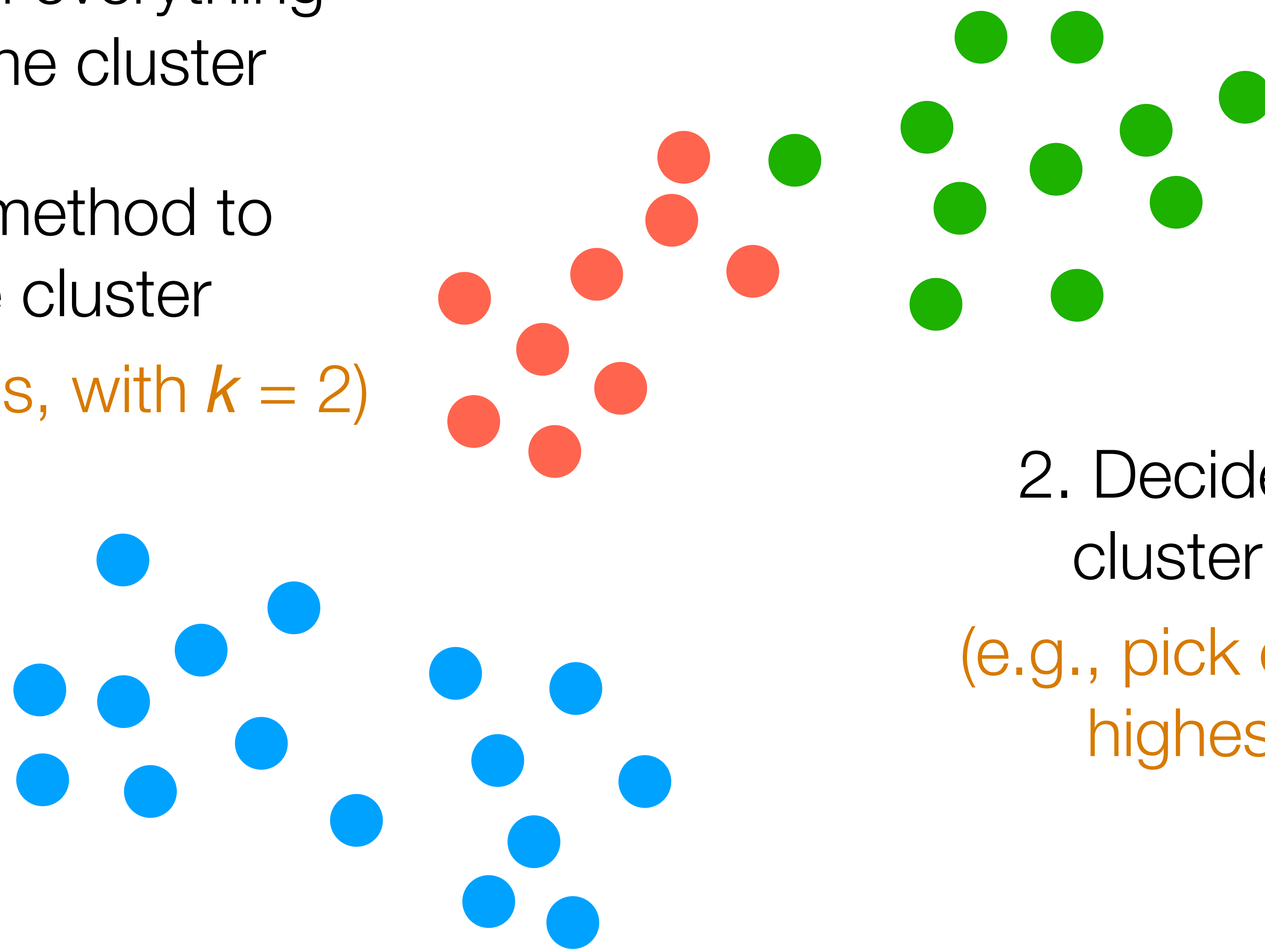
2. Decide on next cluster to split
(e.g., pick cluster with highest RSS)

Top-down: Divisive Clustering

0. Start with everything in the same cluster

1. Use a method to split the cluster

(e.g., *k*-means, with *k* = 2)



2. Decide on next cluster to split

(e.g., pick cluster with highest RSS)

Top-down: Divisive Clustering

0. Start with everything in the same cluster

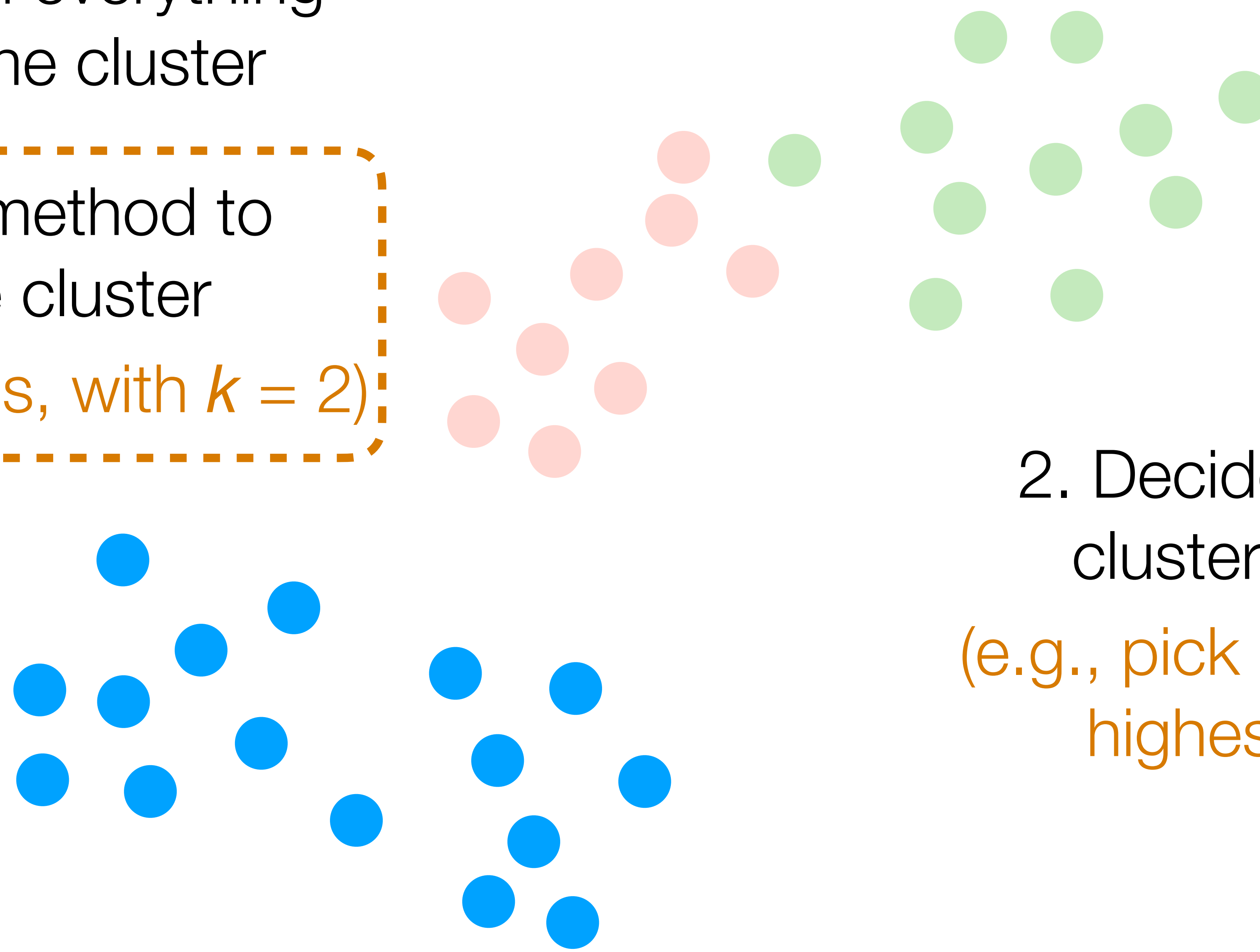
1. Use a method to split the cluster (e.g., *k*-means, with *k* = 2)

2. Decide on next cluster to split (e.g., pick cluster with highest RSS)

Top-down: Divisive Clustering

0. Start with everything in the same cluster

1. Use a method to split the cluster
(e.g., k-means, with k = 2)

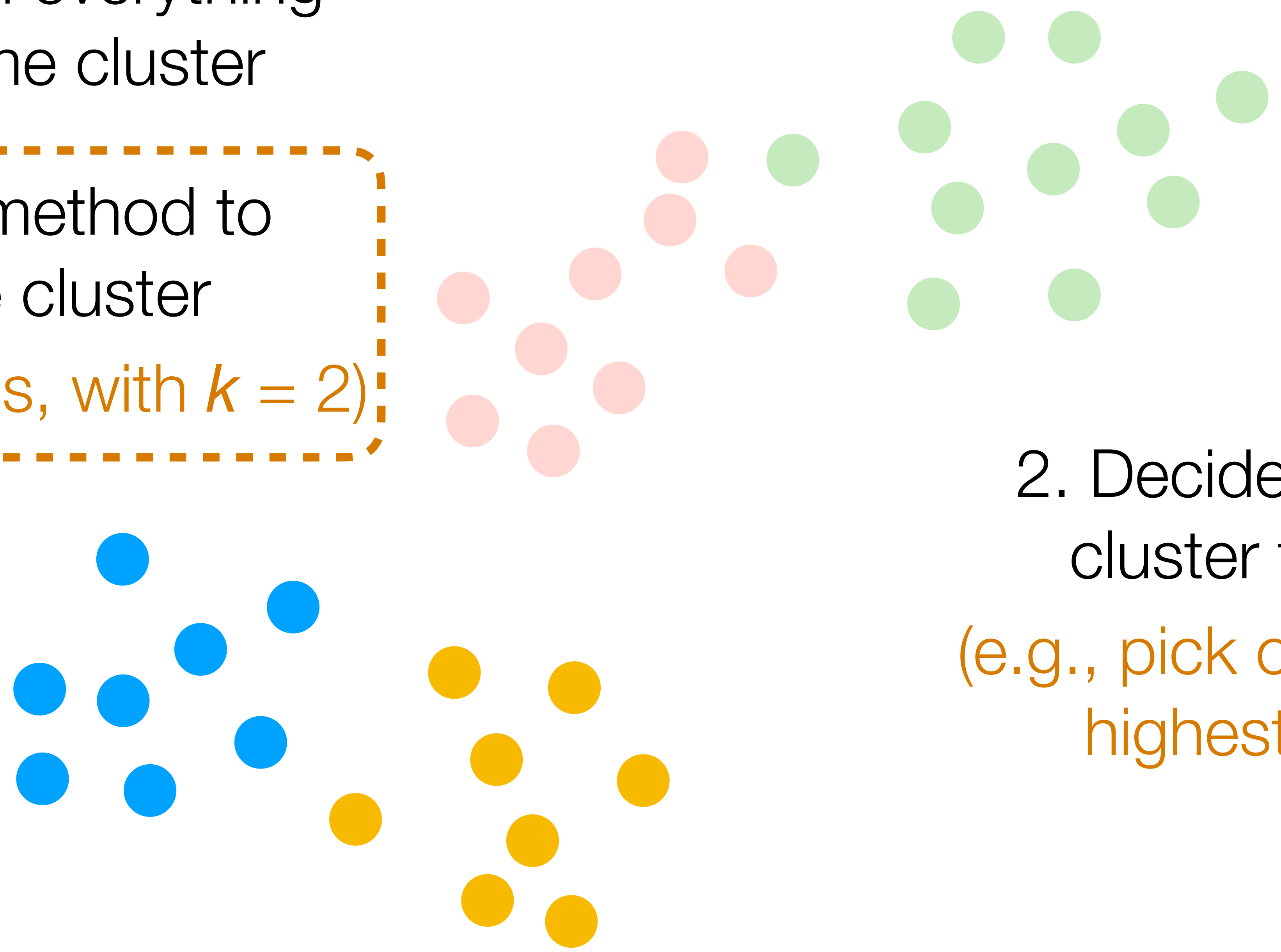


2. Decide on next cluster to split
(e.g., pick cluster with highest RSS)

Top-down: Divisive Clustering

0. Start with everything in the same cluster

1. Use a method to split the cluster
(e.g., *k*-means, with k = 2)



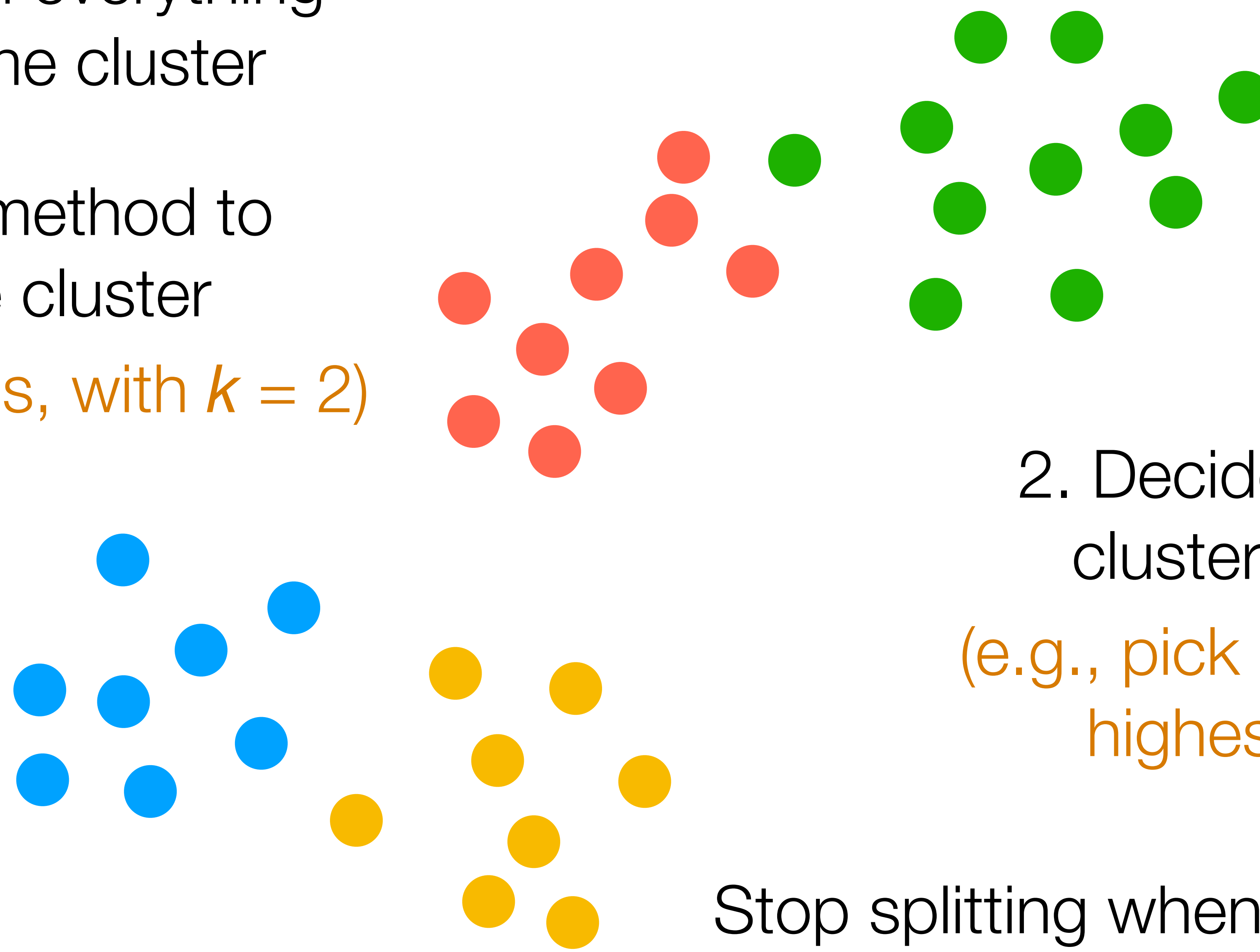
2. Decide on next cluster to split
(e.g., pick cluster with highest RSS)

Top-down: Divisive Clustering

0. Start with everything in the same cluster

1. Use a method to split the cluster

(e.g., *k*-means, with *k* = 2)



2. Decide on next cluster to split

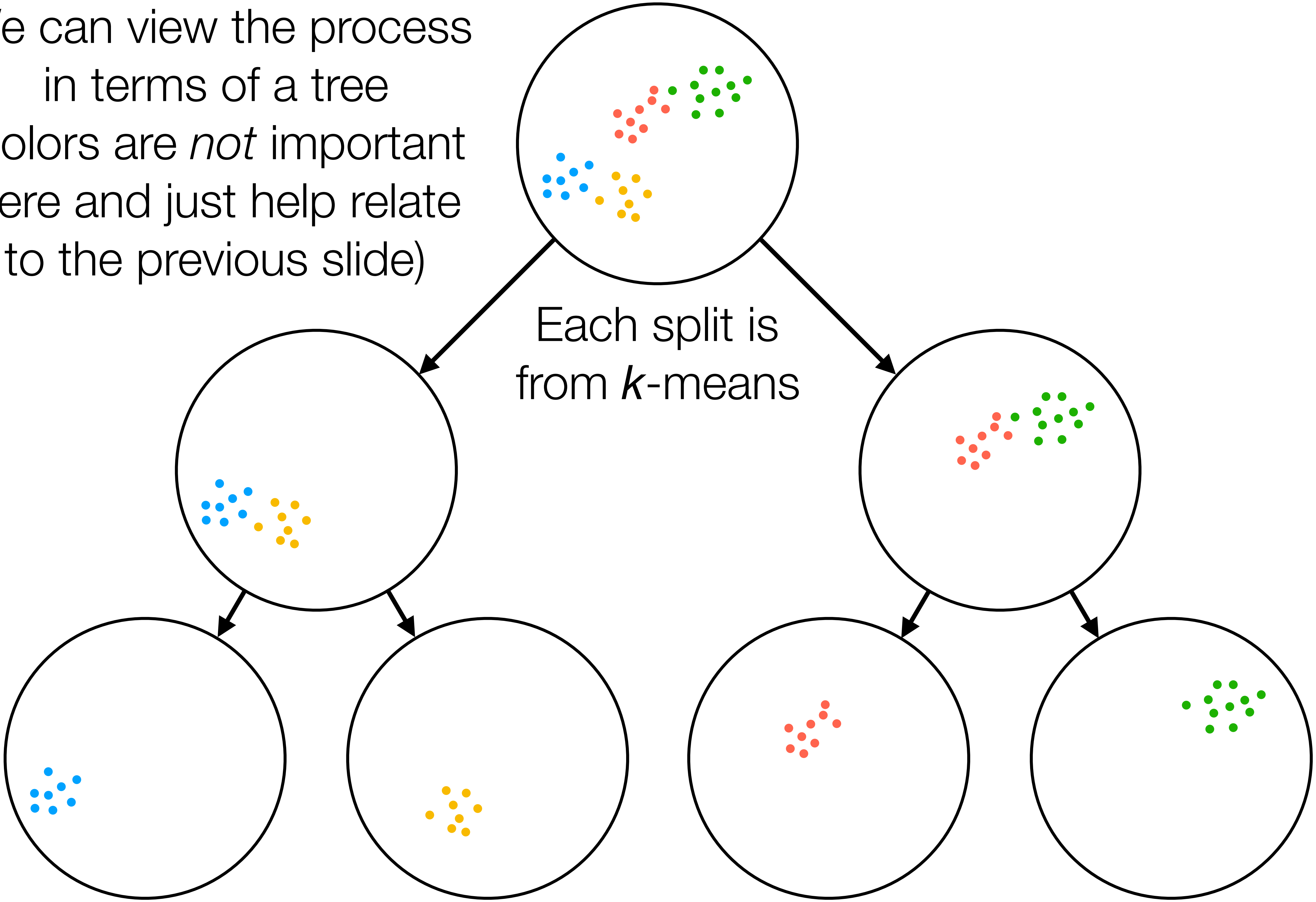
(e.g., pick cluster with highest RSS)

Stop splitting when some termination condition is reached

(e.g., highest cluster RSS is small enough)

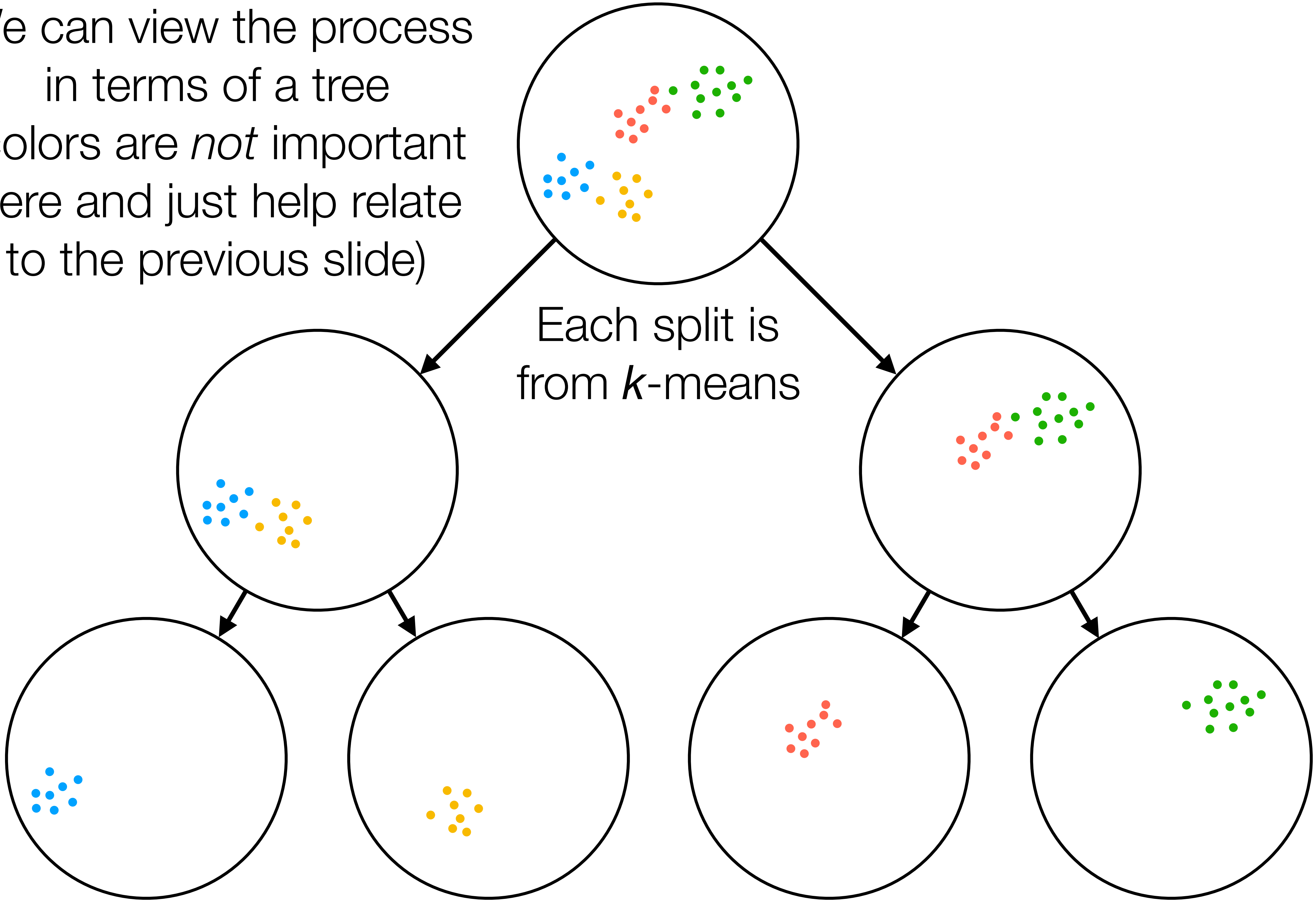
Top-down: Divisive Clustering

We can view the process in terms of a tree (colors are *not* important here and just help relate to the previous slide)



Top-down: Divisive Clustering

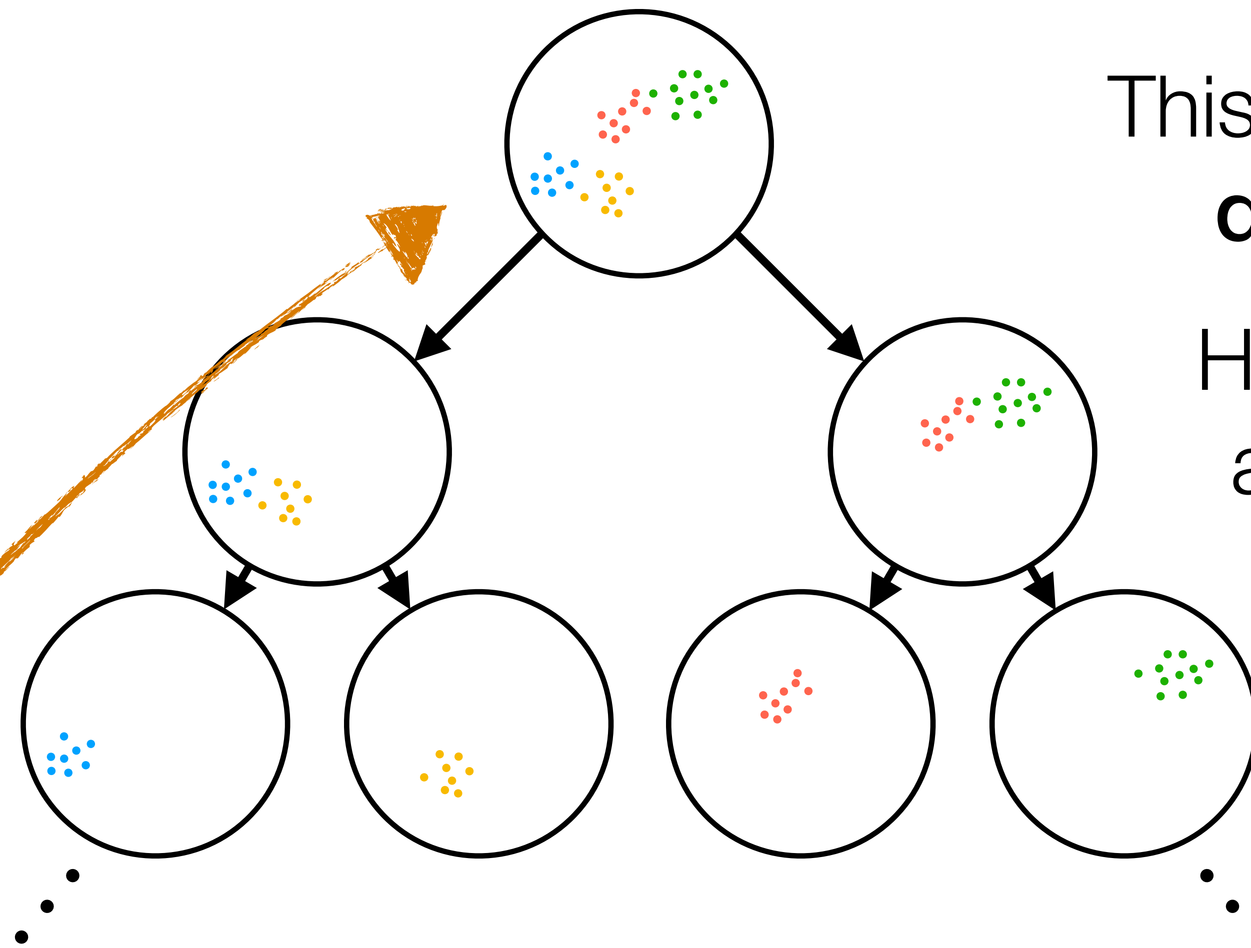
We can view the process in terms of a tree (colors are *not* important here and just help relate to the previous slide)



We could keep splitting until the leaves each have 1 point

Top-down: Divisive Clustering

Agglomerative clustering (bottom up) goes the other way



This tree is called a **dendrogram**

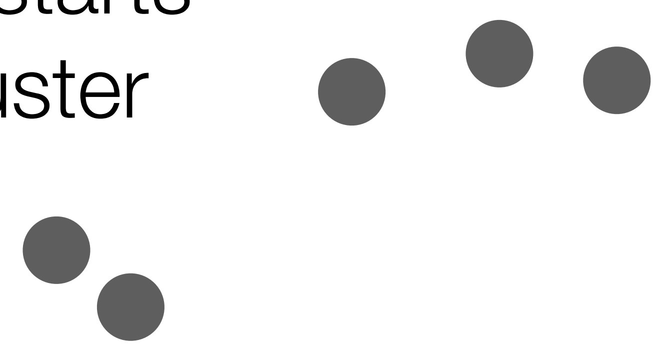
Helpful for visualizing all the intermediate clustering stages

Divisive clustering uses *global* information and keeps splitting

We could keep splitting until the leaves each have 1 point

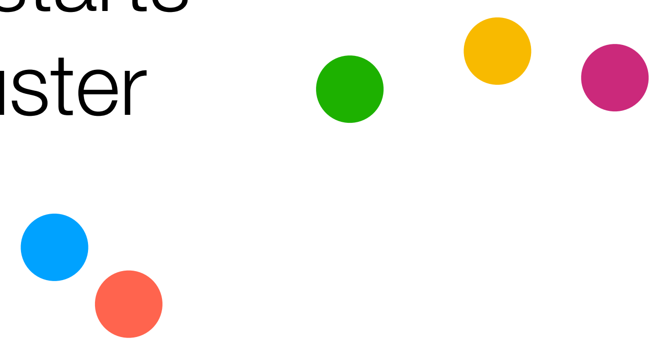
Bottom-up: Agglomerative Clustering

0. Every point starts
as its own cluster



Bottom-up: Agglomerative Clustering

0. Every point starts as its own cluster

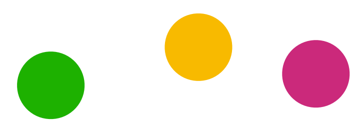


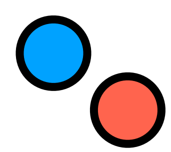
1. Find the “most similar” two clusters

(e.g., pick pair of clusters with closest cluster centers)

Bottom-up: Agglomerative Clustering

0. Every point starts
as its own cluster





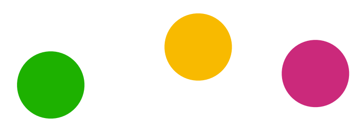
1. Find the “most similar” two clusters

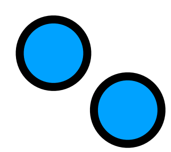
(e.g., pick pair of clusters with
closest cluster centers)

2. Merge them

Bottom-up: Agglomerative Clustering

0. Every point starts as its own cluster





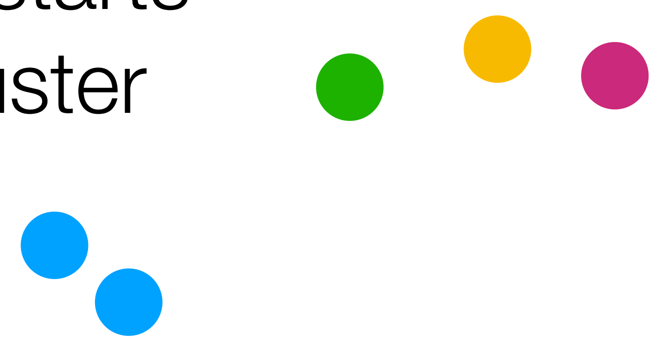
1. Find the “most similar” two clusters

(e.g., pick pair of clusters with closest cluster centers)

2. Merge them

Bottom-up: Agglomerative Clustering

0. Every point starts
as its own cluster



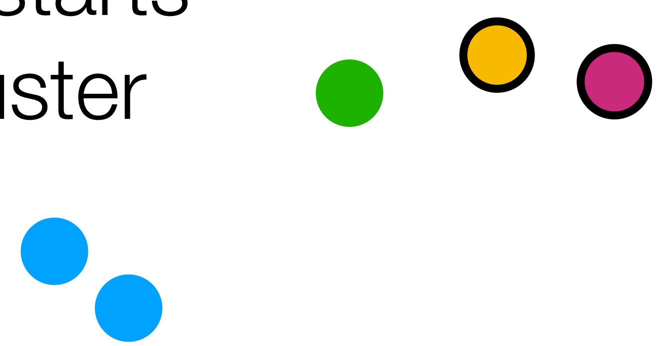
1. Find the “most similar” two clusters

(e.g., pick pair of clusters with
closest cluster centers)

2. Merge them

Bottom-up: Agglomerative Clustering

0. Every point starts
as its own cluster



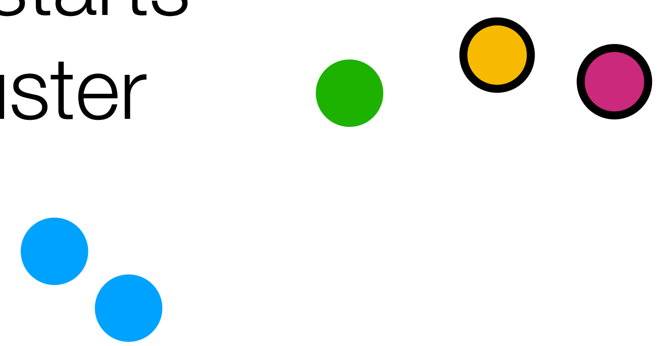
1. Find the “most similar” two clusters

(e.g., pick pair of clusters with
closest cluster centers)

2. Merge them

Bottom-up: Agglomerative Clustering

0. Every point starts
as its own cluster



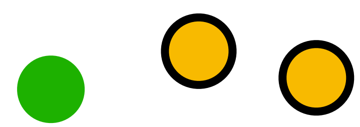
1. Find the "most similar" two clusters

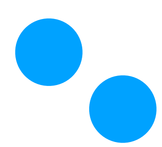
(e.g., pick pair of clusters with
closest cluster centers)

2. Merge them

Bottom-up: Agglomerative Clustering

0. Every point starts
as its own cluster





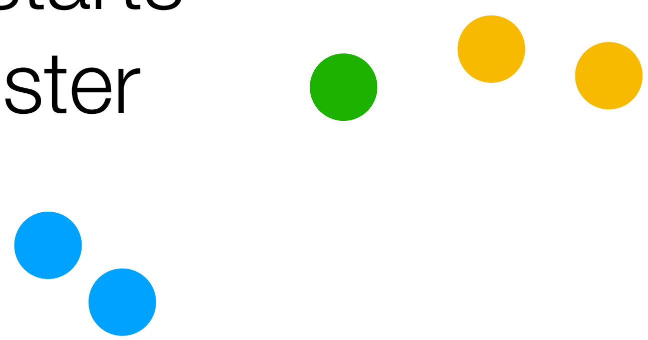
1. Find the “most similar” two clusters

(e.g., pick pair of clusters with
closest cluster centers)

2. Merge them

Bottom-up: Agglomerative Clustering

0. Every point starts as its own cluster



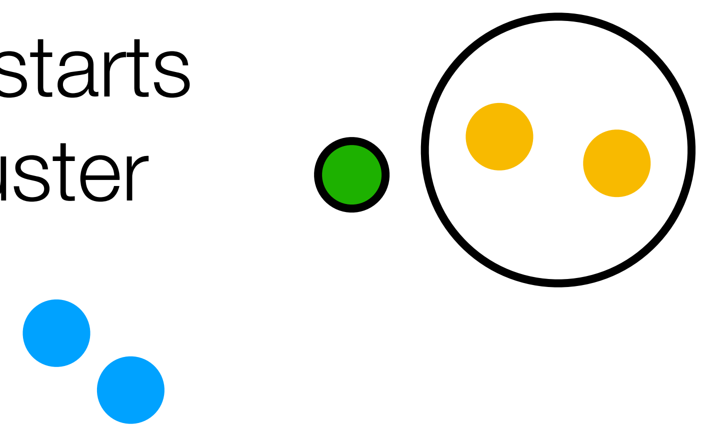
1. Find the “most similar” two clusters

(e.g., pick pair of clusters with closest cluster centers)

2. Merge them

Bottom-up: Agglomerative Clustering

0. Every point starts
as its own cluster



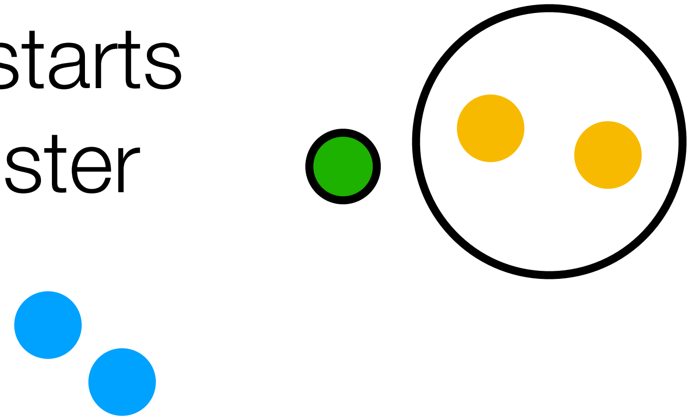
1. Find the “most similar” two clusters

(e.g., pick pair of clusters with
closest cluster centers)

2. Merge them

Bottom-up: Agglomerative Clustering

0. Every point starts
as its own cluster



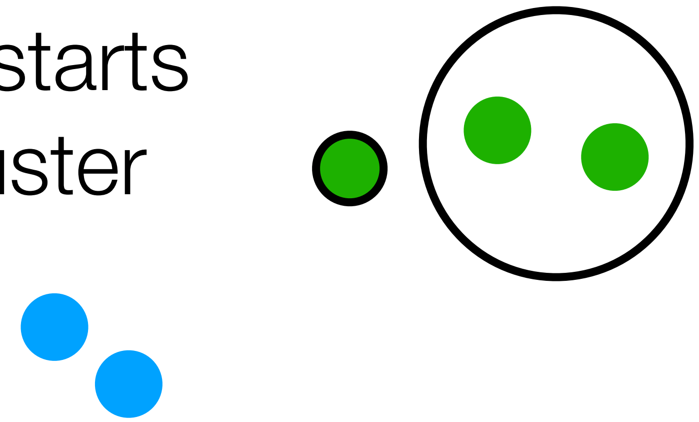
1. Find the “most similar” two clusters

(e.g., pick pair of clusters with
closest cluster centers)

2. Merge them

Bottom-up: Agglomerative Clustering

0. Every point starts as its own cluster



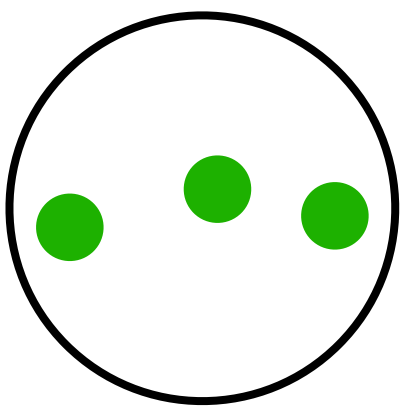
1. Find the “most similar” two clusters

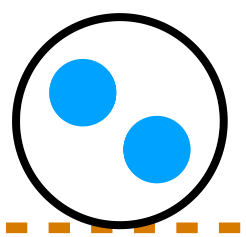
(e.g., pick pair of clusters with closest cluster centers)

2. Merge them

Bottom-up: Agglomerative Clustering

0. Every point starts as its own cluster





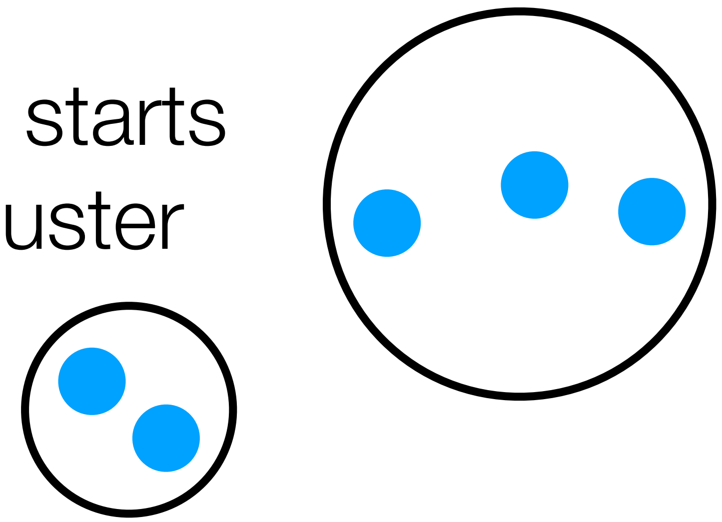
1. Find the “most similar” two clusters

(e.g., pick pair of clusters with closest cluster centers)

2. Merge them

Bottom-up: Agglomerative Clustering

0. Every point starts
as its own cluster



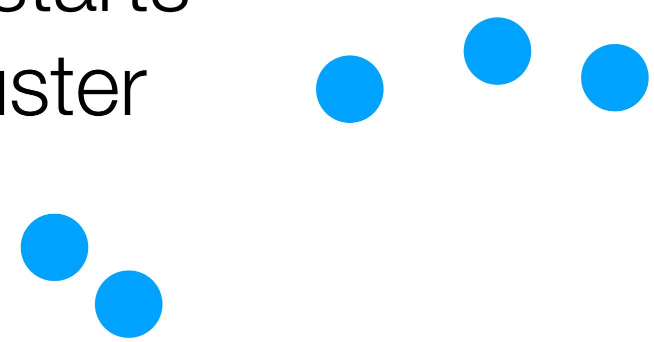
1. Find the “most similar” two clusters

(e.g., pick pair of clusters with
closest cluster centers)

2. Merge them

Bottom-up: Agglomerative Clustering

0. Every point starts as its own cluster



1. Find the "most similar" two clusters

(e.g., pick pair of clusters with closest cluster centers)

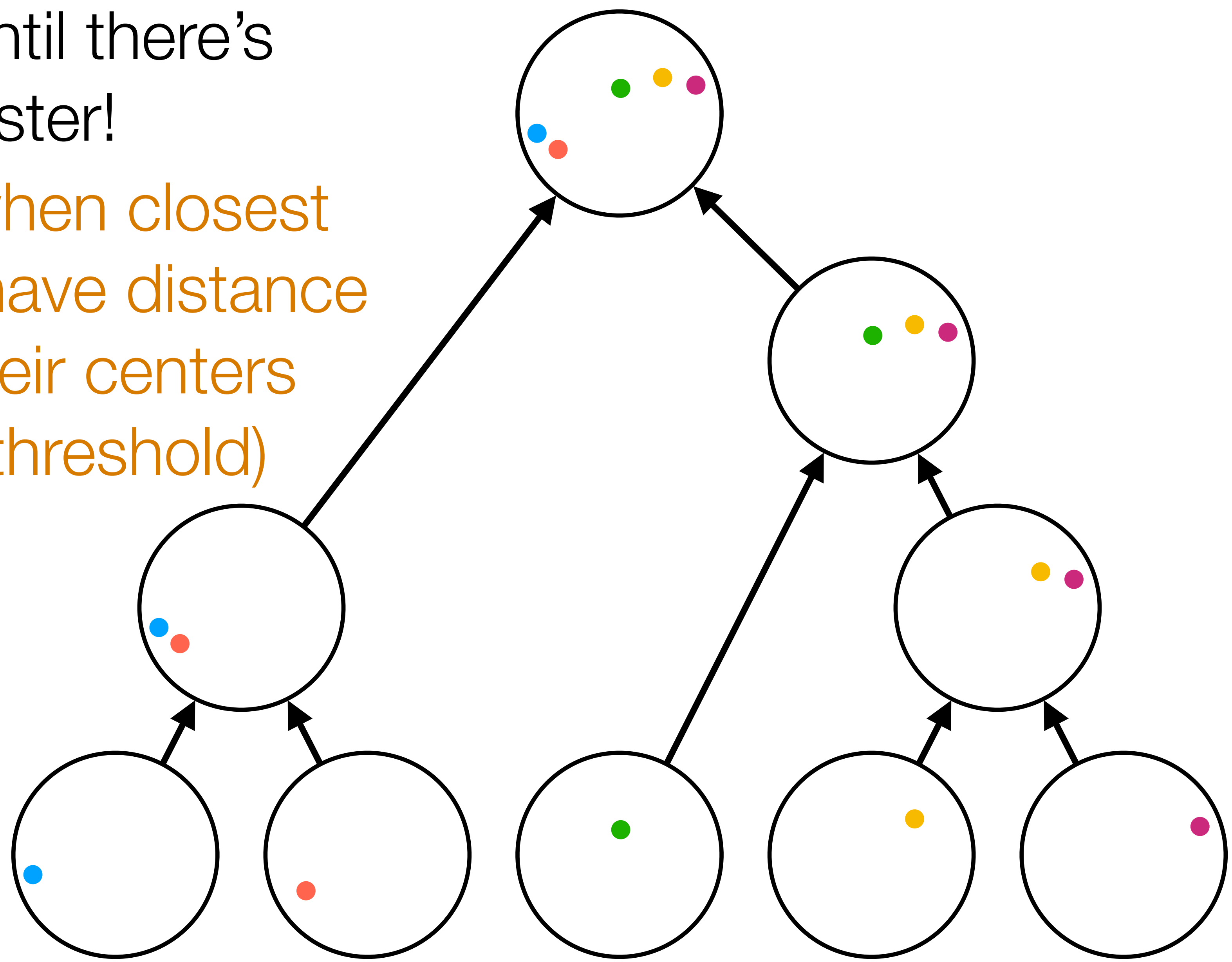
2. Merge them

Bottom-up: Agglomerative Clustering

Don't have to keep merging until there's 1 cluster!

(e.g., stop when closest two clusters have distance between their centers exceed a threshold)

Dendrogram

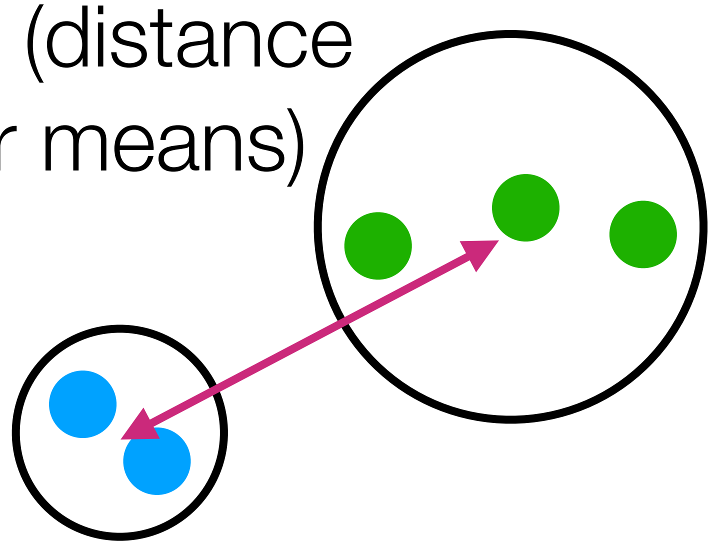


Agglomerative clustering uses *local* information and keeps merging

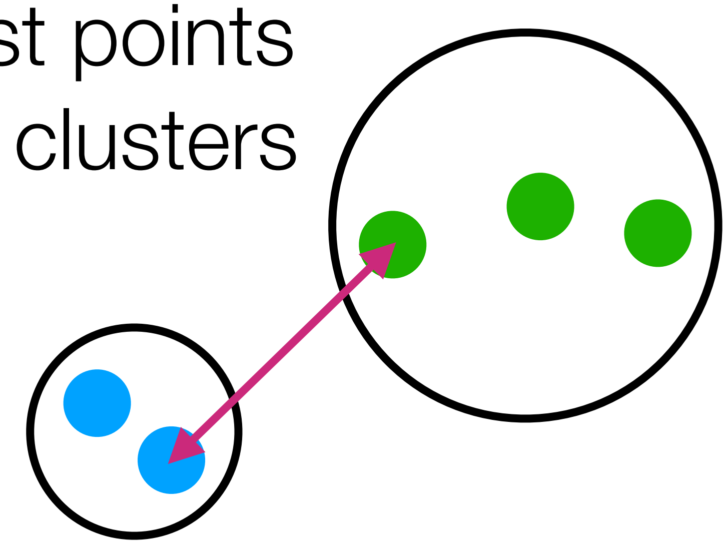
Bottom-up: Agglomerative Clustering

Some ways to define what it means for two clusters to be “close” (needed to find most similar clusters):

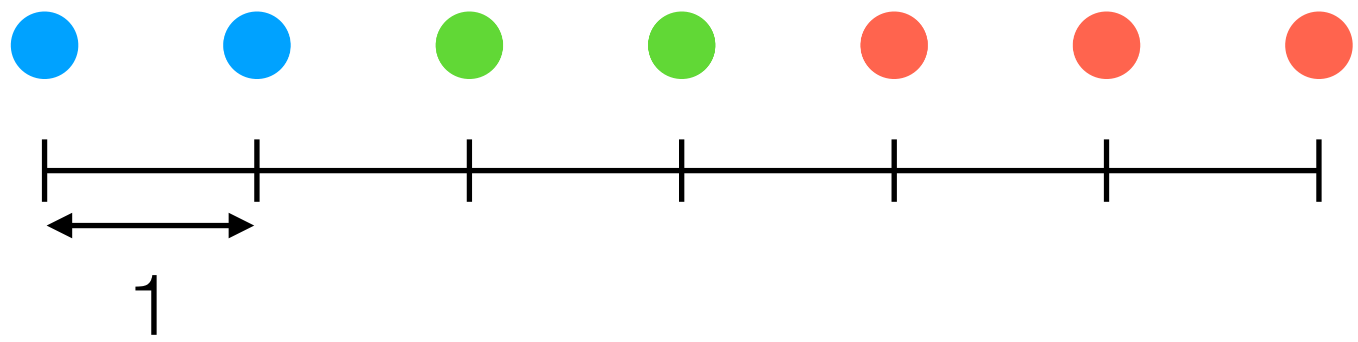
Centroid linkage: what we saw already (distance between cluster means)



Single linkage: use distance between closest points across the two clusters



Example: Single Linkage



What would single linkage merge next?

Distance between blue and green: 1

Distance between blue and red: 3

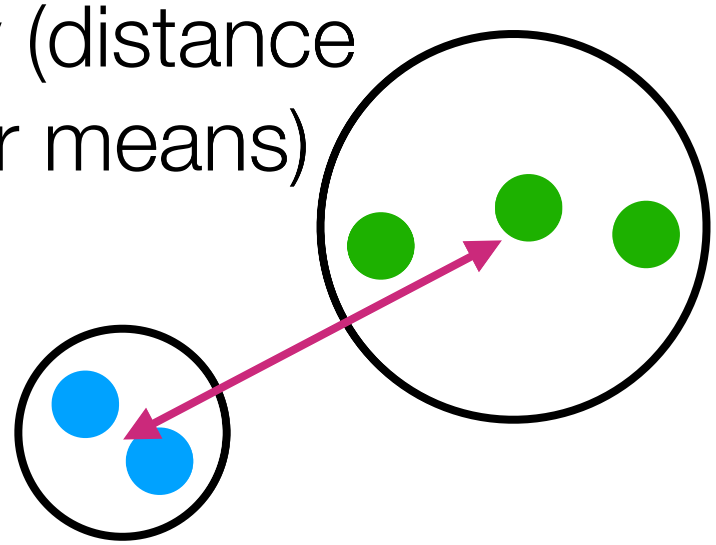
Distance between green and red: 1

Single linkage would merge either blue with green, or green with red

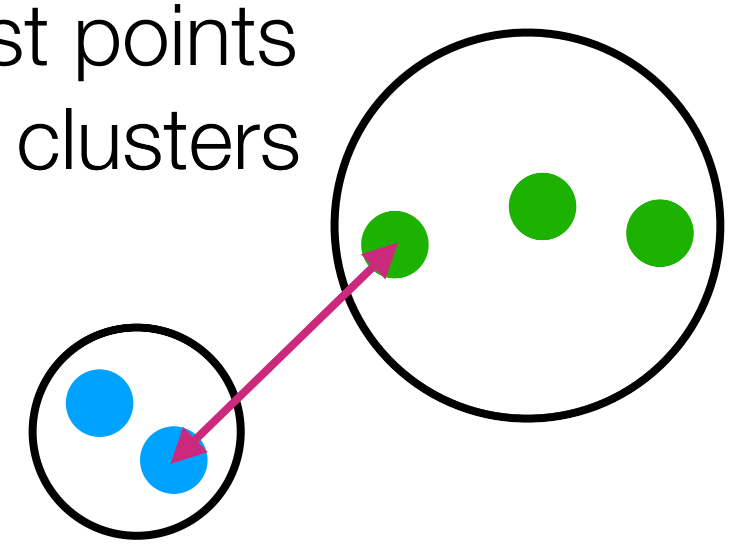
Bottom-up: Agglomerative Clustering

Some ways to define what it means for two clusters to be "close" (needed to find most similar clusters):

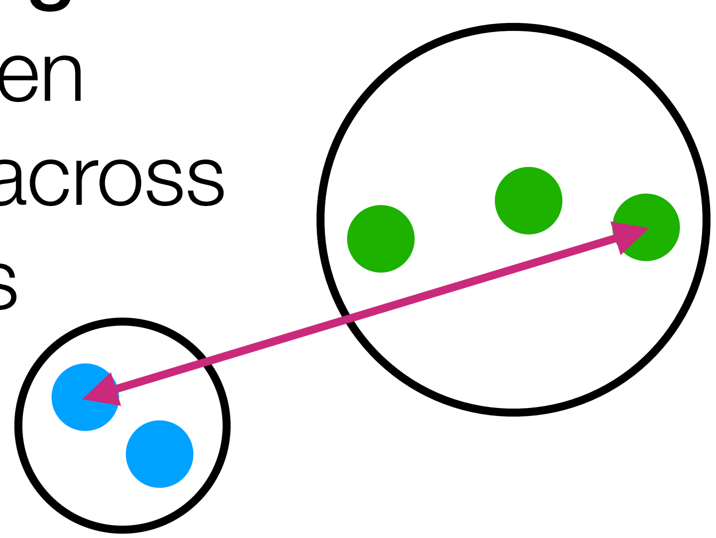
Centroid linkage: what we saw already (distance between cluster means)



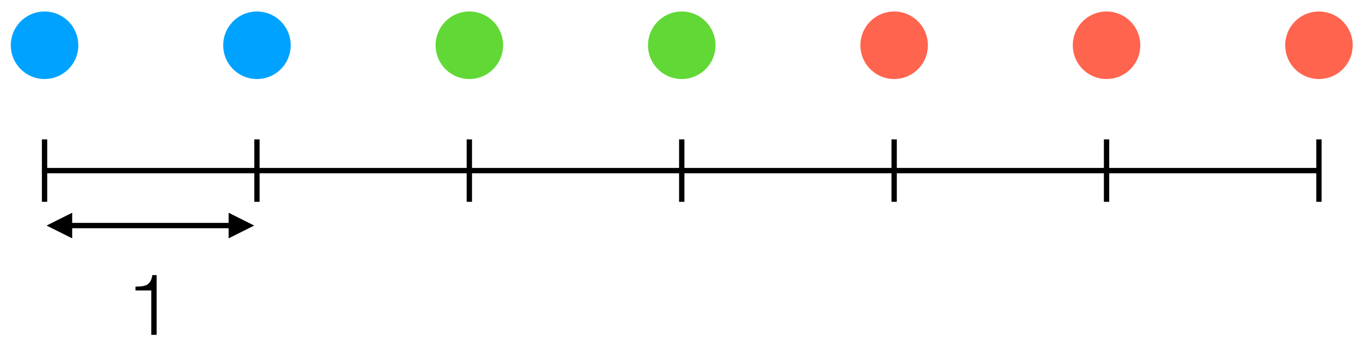
Single linkage: use distance between closest points across the two clusters



Complete linkage: use distance between farthest points across the two clusters



Example: Complete Linkage



What would complete linkage merge next?

Distance between blue and green: 3

Distance between blue and red: 6

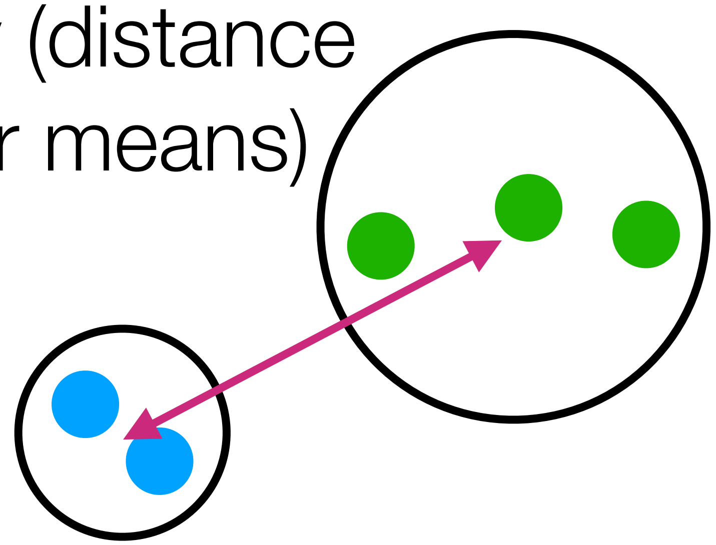
Distance between green and red: 4

Complete linkage would merge blue and green

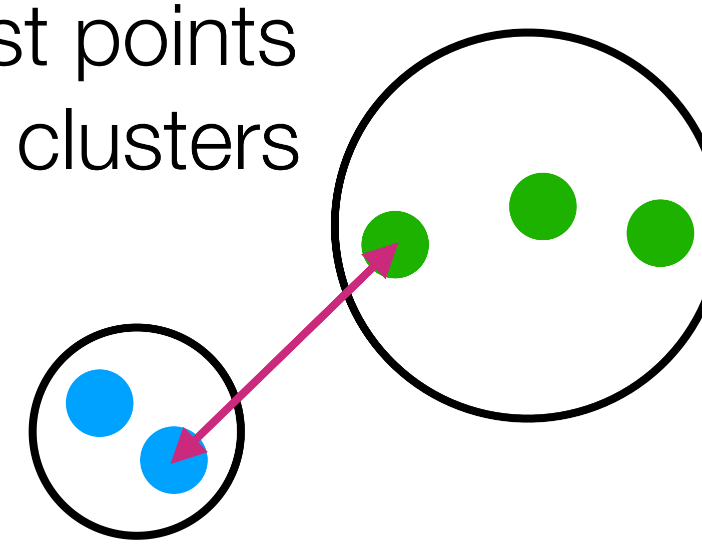
Bottom-up: Agglomerative Clustering

Some ways to define what it means for two clusters to be “close” (needed to find most similar clusters):

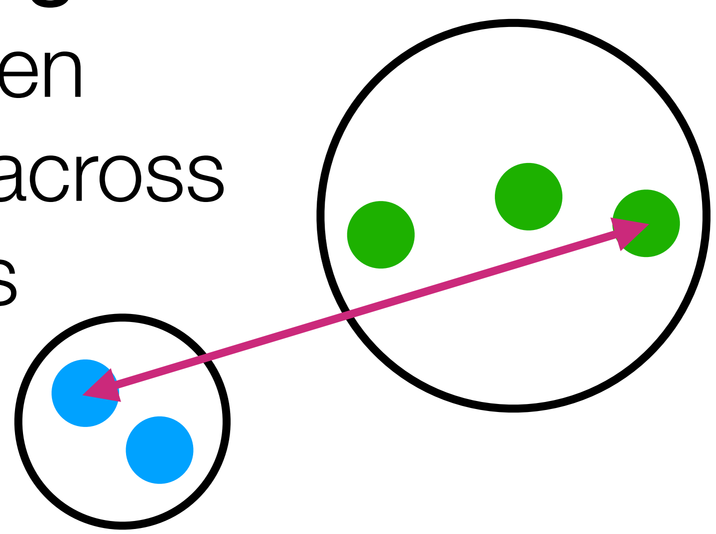
Centroid linkage: what we saw already (distance between cluster means)



Single linkage: use distance between closest points across the two clusters



Complete linkage: use distance between farthest points across the two clusters



There are other ways as well: none are perfect

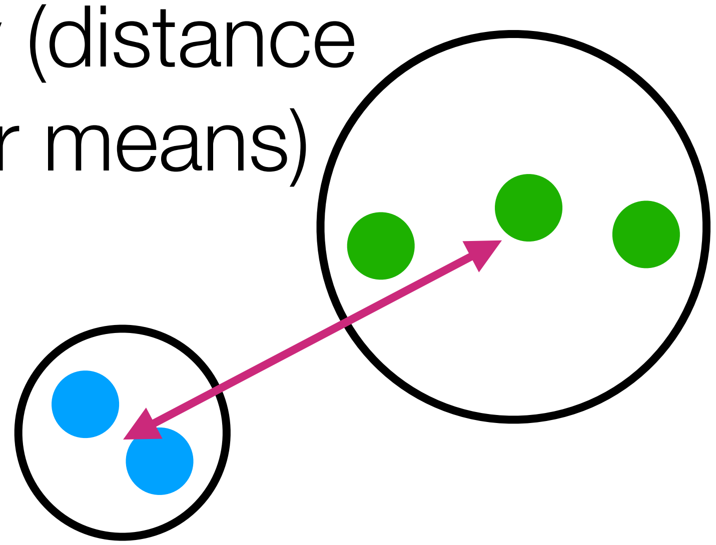
Hierarchical Clustering

Demo

Bottom-up: Agglomerative Clustering

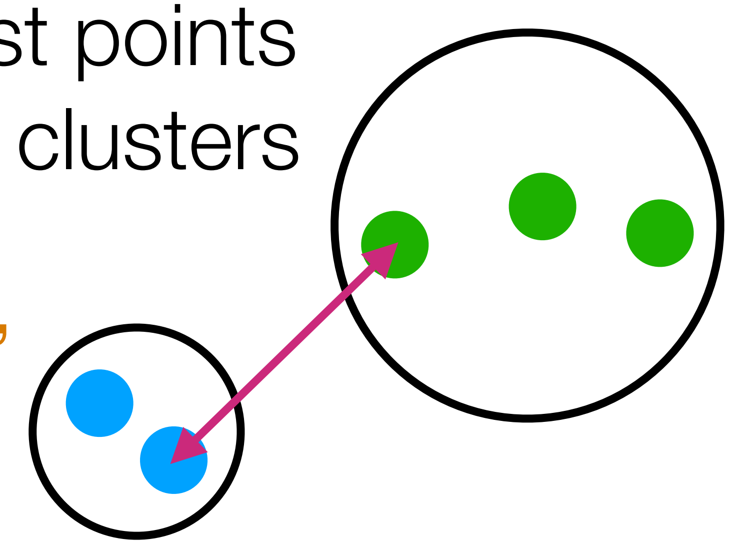
Some ways to define what it means for two clusters to be “close” (needed to find most similar clusters):

Centroid linkage: what we saw already (distance between cluster means)



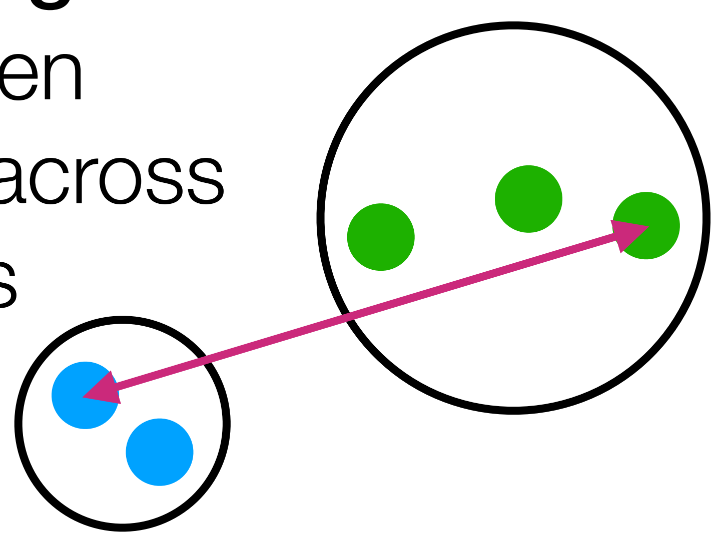
Ignores
items in
each cluster

Single linkage: use distance between closest points across the two clusters



Has “chaining” behavior

Complete linkage: use distance between farthest points across the two clusters



Has “crowding” behavior

There are other ways as well: none are perfect

Going from Similarities to Clusters

There's a whole zoo of clustering methods

Two main categories we'll talk about:

Generative models

1. Pretend data generated by specific model with parameters

2. Learn the parameters ("fit model to data")

3. Use fitted model to determine cluster assignments

Hierarchical clustering

Top-down: Start with everything in 1 cluster and decide on how to recursively split

Bottom-up: Start with everything in its own cluster and decide on how to iteratively merge clusters

Going from Similarities to Clusters

Generative models

1. Pretend data generated by specific model with parameters

2. Learn the parameters ("fit model to data")

3. Use fitted model to determine cluster assignments

The most popular models effectively assume Euclidean distance…

You learn a model
➔ can predict cluster assignments for points not seen in training

Hierarchical clustering

Top-down: Start with everything in 1 cluster and decide on how to recursively split

Bottom-up: Start with everything in its own cluster and decide on how to iteratively merge clusters

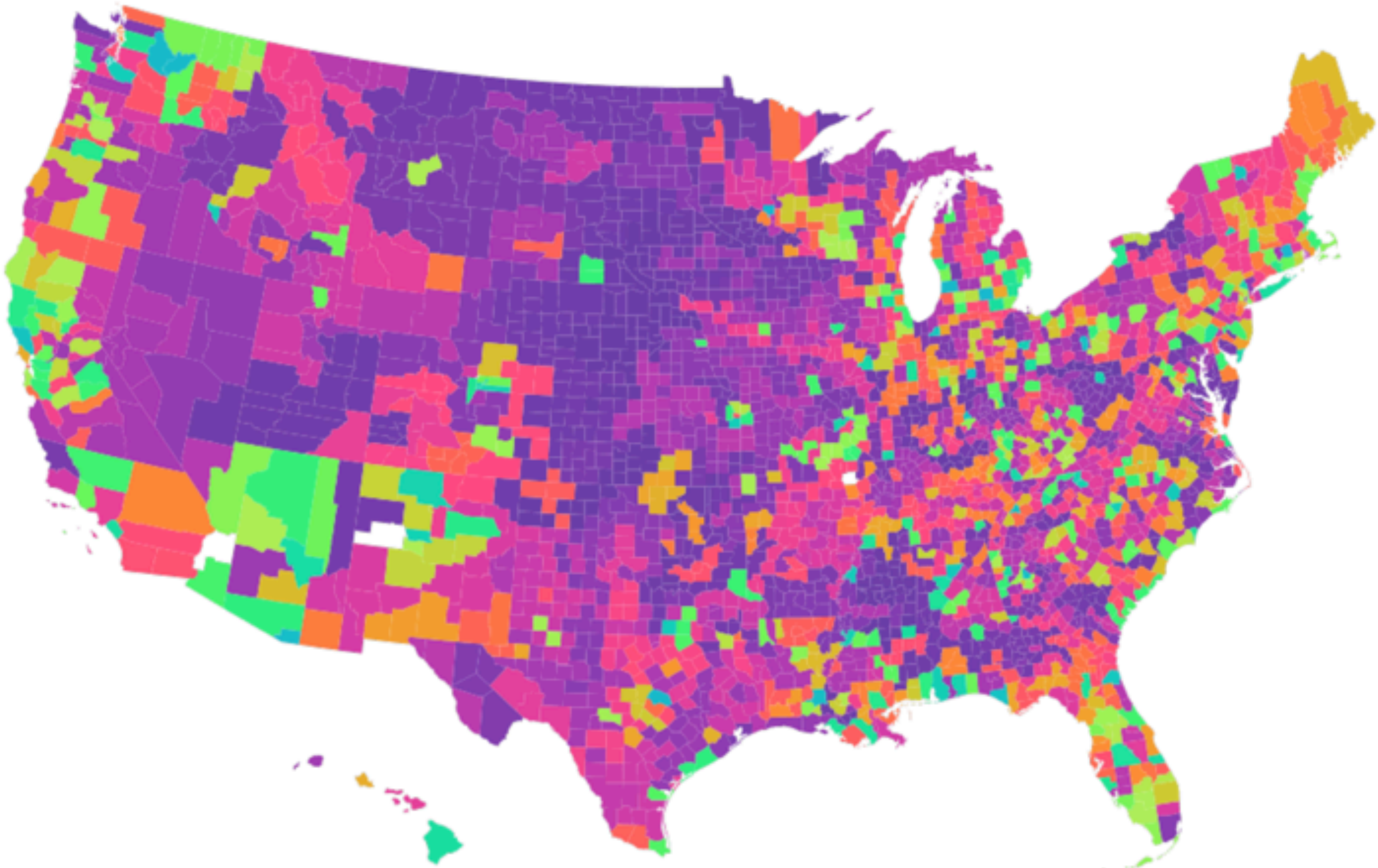
Easily works with different distances (not just Euclidean)

Great for problems that don't need to predict clusters for future points

Different split/merge criteria lead to clusters that look specific ways (e.g., chaining, crowding)

Example: Clustering on U.S. Counties

(using opioid death rate data across 37 years)



No need to predict which cluster new counties should belong to, since we're already looking at all U.S. counties!

Image source: Amanda Coston

How to Choose a Clustering Method?

In general: not easy!

Some questions to think about:

- What features to even cluster on?
- For your application, what distance/similarity makes sense?
- Do you care about cluster assignments for new points?

It's possible that several clustering methods give similar results (*which is great!* — it means that there are some reasonably "stable" clusters in your data)

- Example: *tons* of clustering methods can figure out from senate voting data who Democrats and Republicans are (of course, *without* knowing each senator's political party)

Clustering Last Remarks

Ultimately, *you* have to decide on which clustering method and number of clusters make sense for your data

- After you run a clustering algorithm, make visualizations to interpret the clusters *in the context of your application*!
- Do not just blindly rely on numerical metrics (e.g., CH index)
- Some times it makes more sense to define your own score function for how good a clustering assignment is

If you can set up a prediction task, then you can use the prediction task to guide the clustering