Carnegie Mellon University
Heinz College

Clustering: Automatic Selection of k, Hierarchical Clustering

George Chen

Automatic Selection of *k*

Dirichlet Process Gaussian Mixture Model (DP-GMM):

- Number of clusters is effectively random, and *can grow with the amount of data you have!*
- While you don't have to choose ***k***, you have to choose a different parameter which says basically how likely new points are to form new clusters vs join existing clusters

DP-GMM High-Level Idea

Cluster 1	Cluster 2	Cluster 3	
	There is a parameter that controls how these π values roughly decay		
Probability of generating a point from cluster 1 = π_1	π_2	π_3	…
Gaussian mean = μ_1	μ_2	μ_3	It goes on forever!
Gaussian covariance = Σ_1	Σ_2	Σ_3	
	There are an infinite number of parameters		

(Rough idea) How to generate points from this DP-GMM:

1. Flip biased ∞-sided coin (the sides have probabilities π_1, π_2, π_3, …)
2. Let Z be the side that we got (it is a positive integer)
3. Sample 1 point from Gaussian mean μ_Z, covariance Σ_Z

Remark: For any given dataset, when learning the DP-GMM, there aren't going to be an infinite number of clusters found

Automatic Selection of *k*

Dirichlet Process Gaussian Mixture Model (DP-GMM):

- Number of clusters is effectively random, and *can grow with the amount of data you have!*

- While you don't have to choose ***k***, you have to choose a different parameter which says basically how likely you are to form new clusters vs try to stick to already existing clusters

- An example of a *Bayesian nonparametric model*
(roughly: a generative model with an *infinite number of parameters*, where the *parameters are random*)

Learning a DP-GMM

Two main approaches:

- Finite approximation where you specify some maximum number of possible clusters (the algorithm will find up to that many clusters)

  This is what's implemented in *sklearn*

  - Algorithm is somewhat similar to k-means/EM for GMMs
  - Algorithm output: very similar to regular GMM fitting

- Random sampling approach (no finite approximation needed!)
  - Algorithm output: a bunch of samples of different cluster assignments (can pick one with highest probability)

    This is what's implemented in R (package *dpmixsim*)

Learning a DP-GMM

Demo

k-means approximates
(a special case of) learning GMM's.

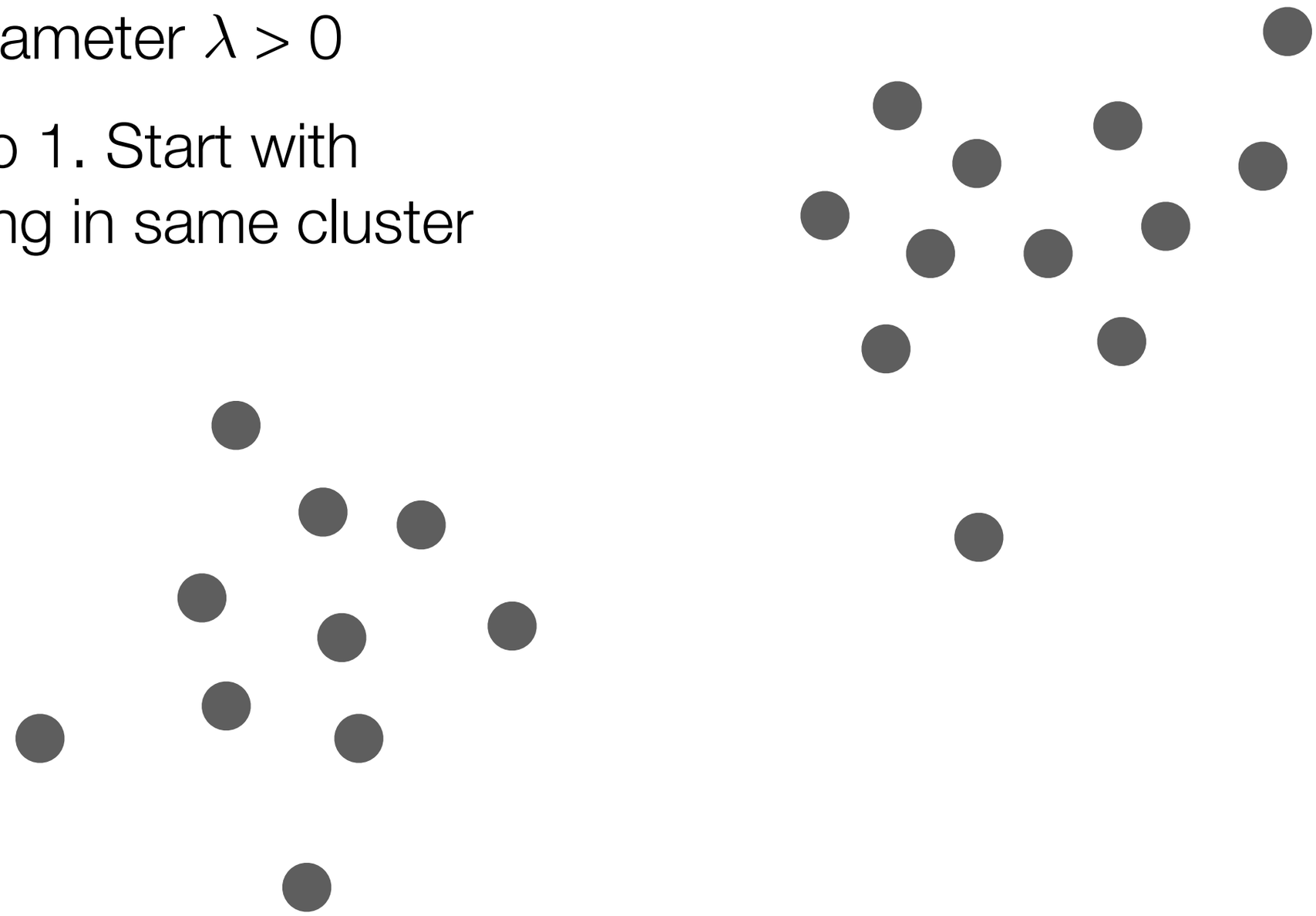
What approximates learning DP-GMMs?

This next algorithm will give you a sense of how we get around specifying the number of clusters directly

DP-means

Step 0. Pick concentration parameter $\lambda > 0$

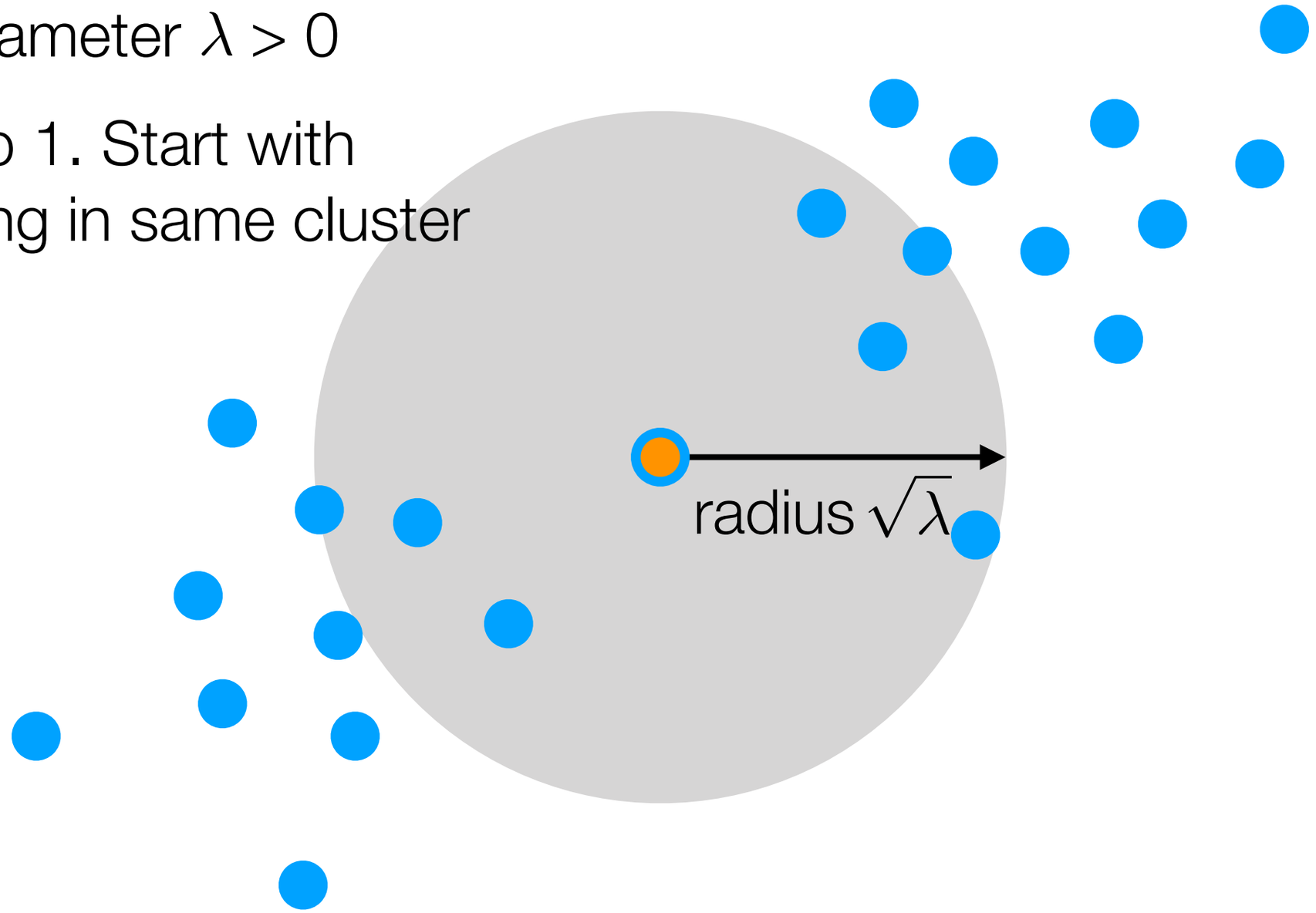
Step 1. Start with everything in same cluster



DP-means

Step 0. Pick concentration parameter $\lambda > 0$

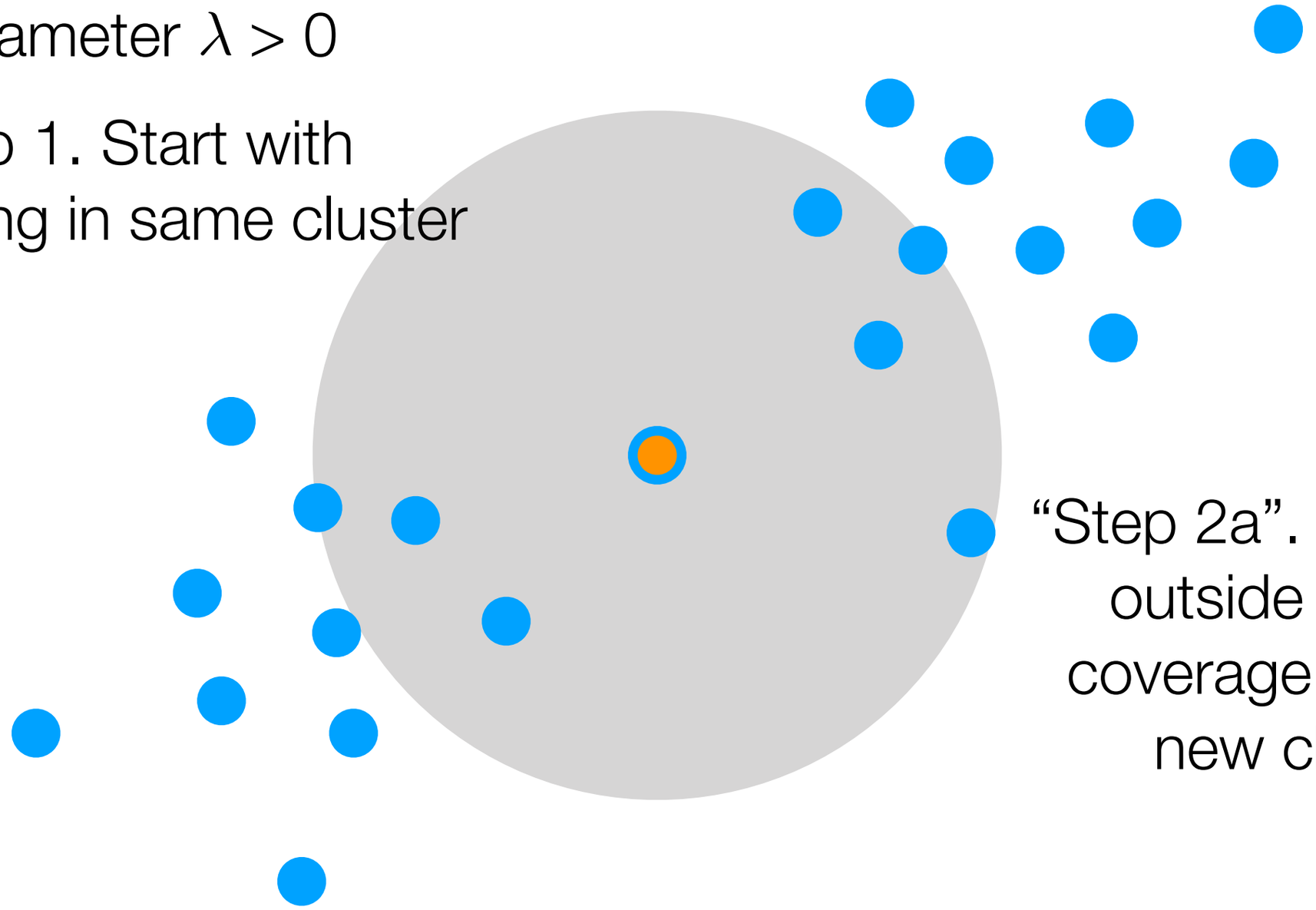
Step 1. Start with everything in same cluster



DP-means

Step 0. Pick concentration parameter $\lambda > 0$

Step 1. Start with everything in same cluster

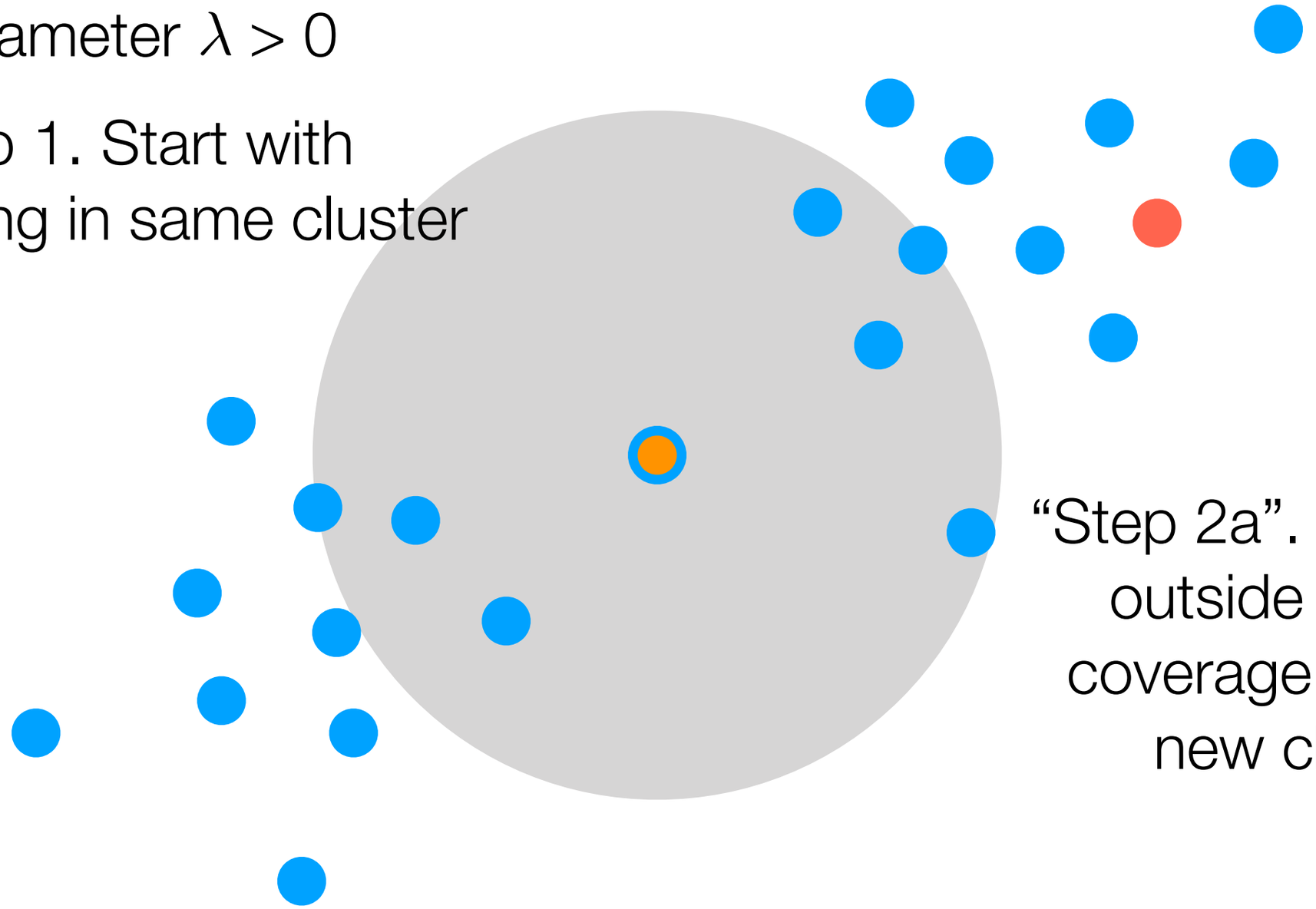


"Step 2a". Pick point outside of gray coverage to make new cluster

DP-means

Step 0. Pick concentration parameter $\lambda > 0$

Step 1. Start with everything in same cluster

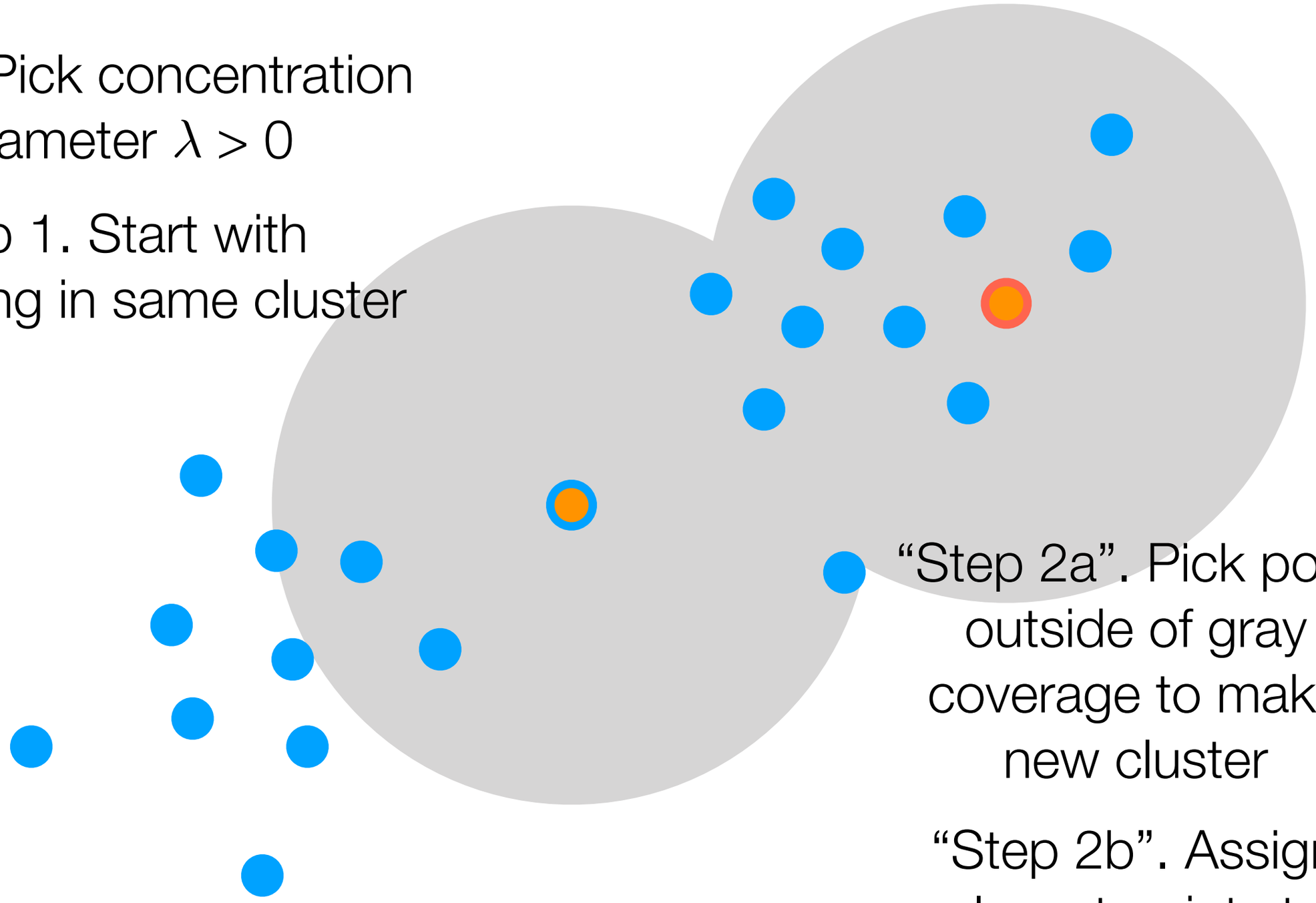


"Step 2a". Pick point outside of gray coverage to make new cluster

DP-means

Step 0. Pick concentration parameter $\lambda > 0$

Step 1. Start with everything in same cluster



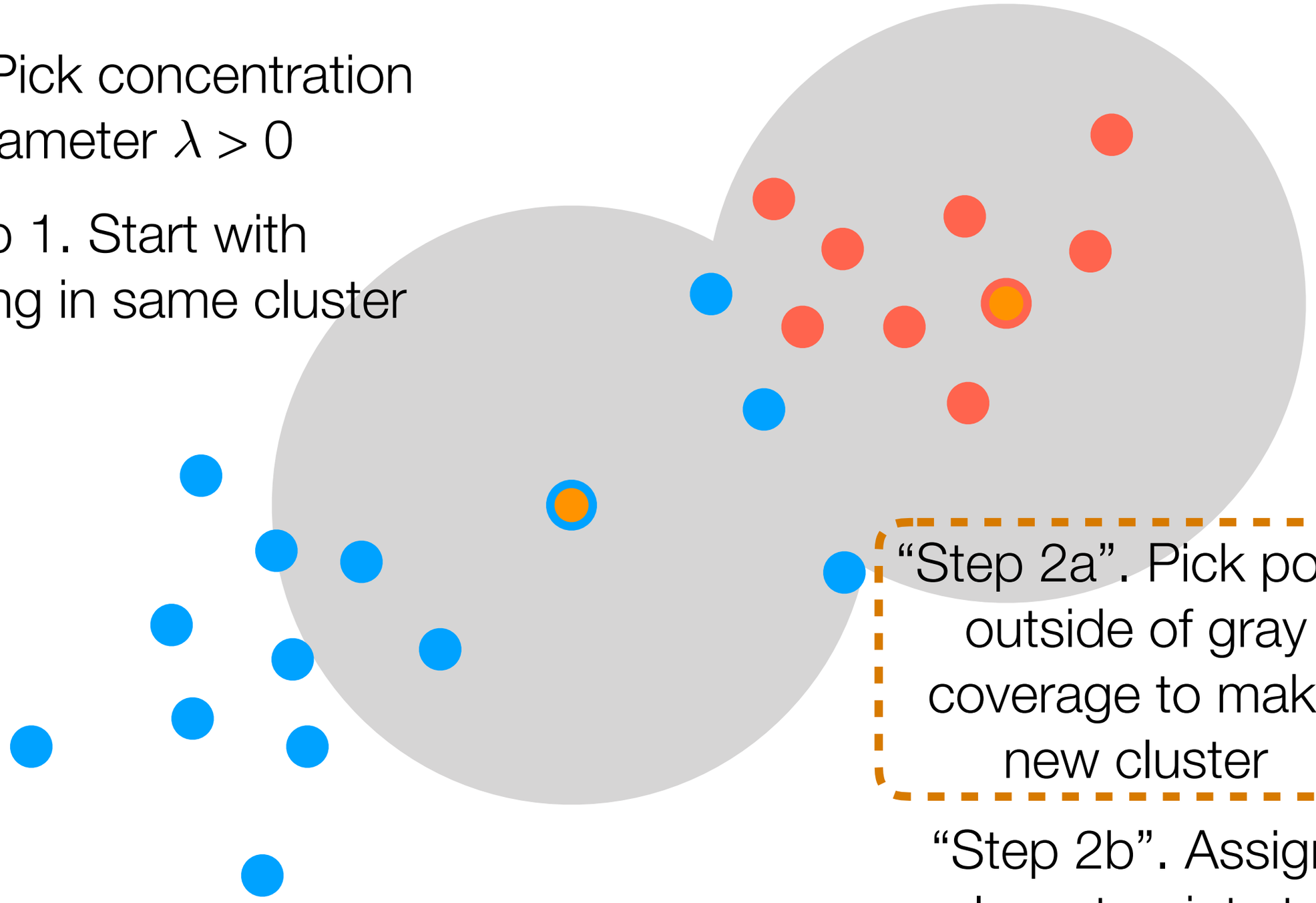
"Step 2a". Pick point outside of gray coverage to make new cluster

"Step 2b". Assign closest points to current clusters

DP-means

Step 0. Pick concentration parameter $\lambda > 0$

Step 1. Start with everything in same cluster



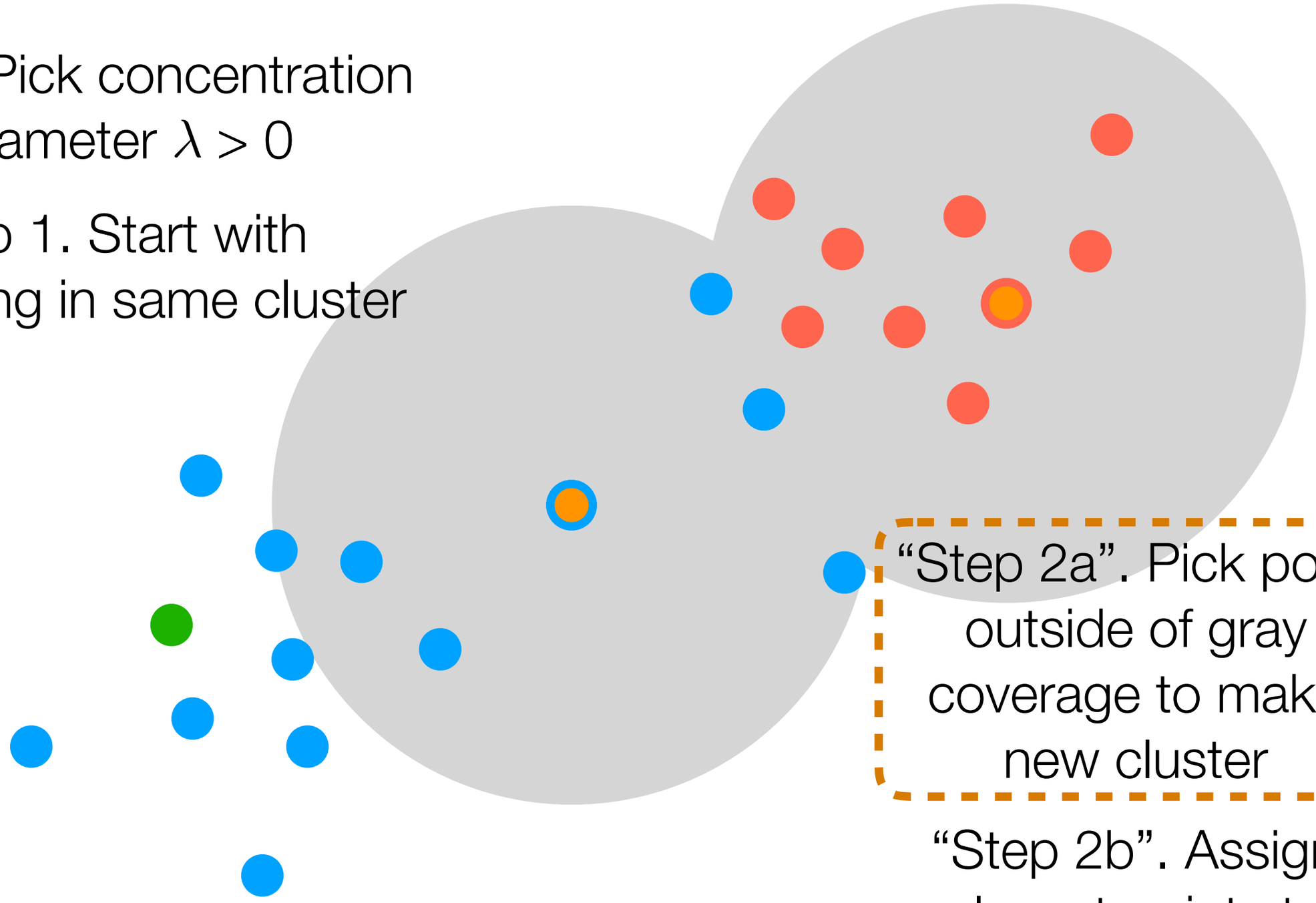
"Step 2a". Pick point outside of gray coverage to make new cluster

"Step 2b". Assign closest points to current clusters

DP-means

Step 0. Pick concentration parameter $\lambda > 0$

Step 1. Start with everything in same cluster



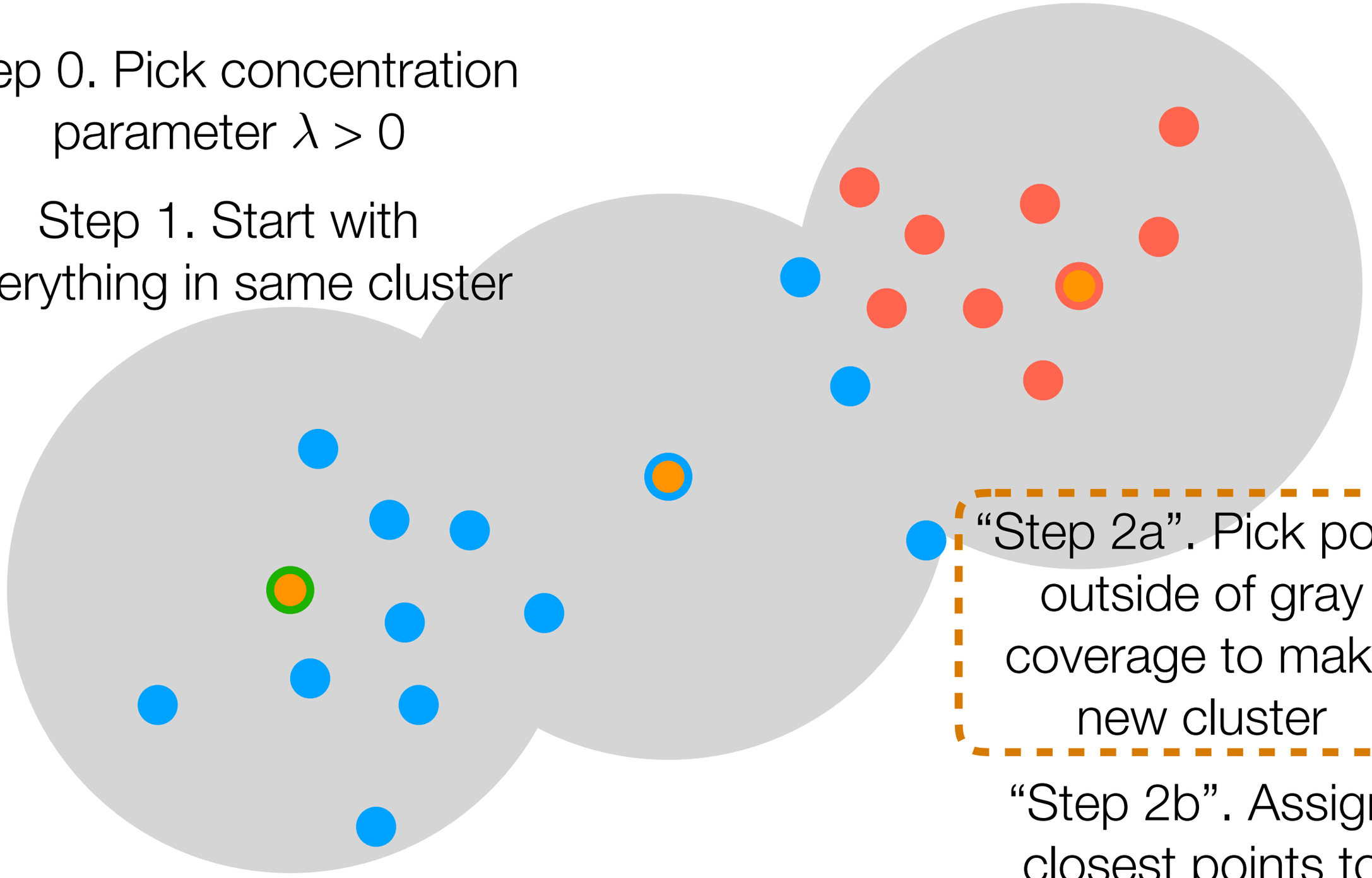
"Step 2a". Pick point outside of gray coverage to make new cluster

"Step 2b". Assign closest points to current clusters

DP-means

Step 0. Pick concentration parameter $\lambda > 0$

Step 1. Start with everything in same cluster



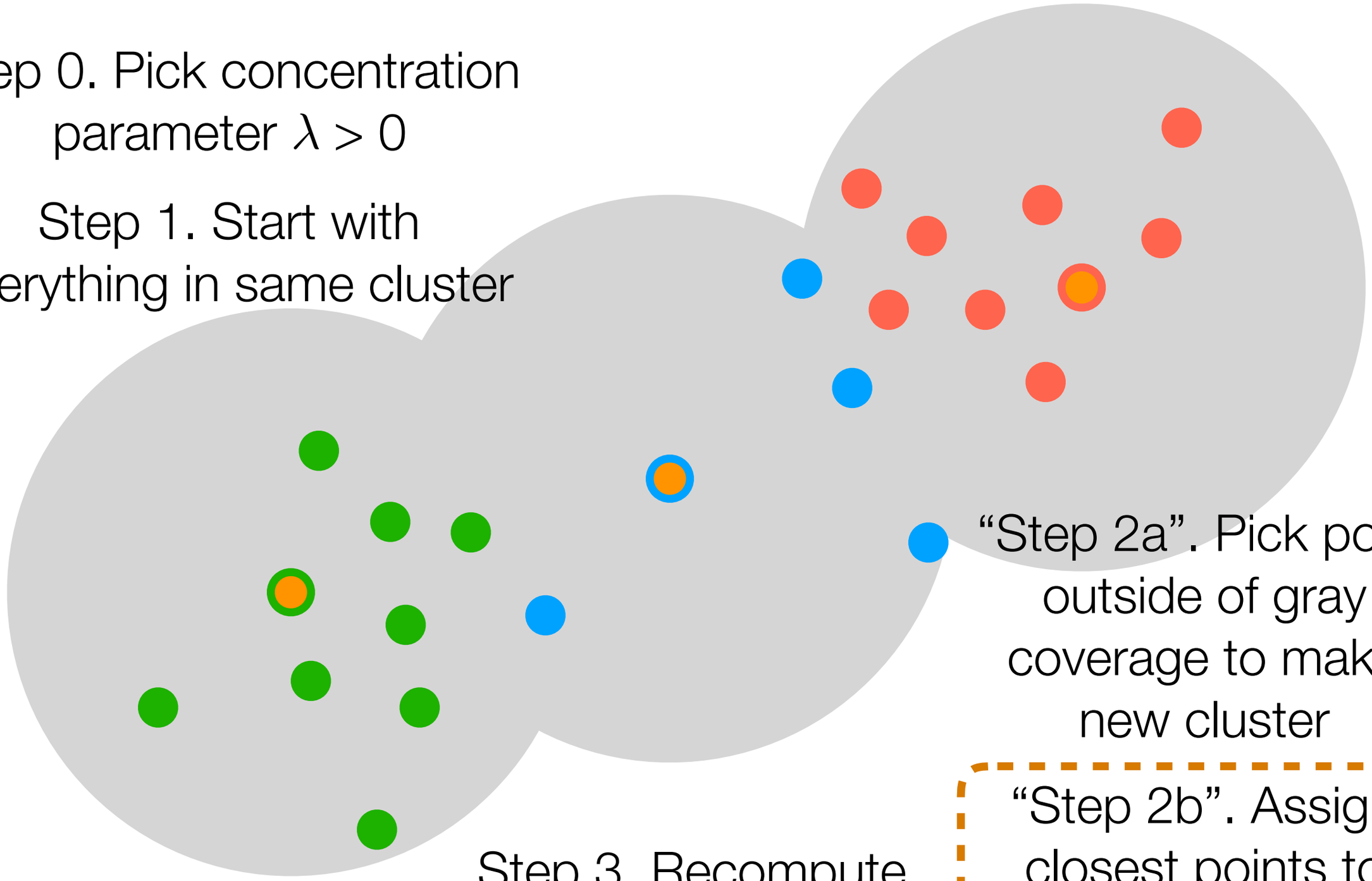
"Step 2a". Pick point outside of gray coverage to make new cluster

"Step 2b". Assign closest points to current clusters

DP-means

Step 0. Pick concentration parameter $\lambda > 0$

Step 1. Start with everything in same cluster



“Step 2a”. Pick point outside of gray coverage to make new cluster

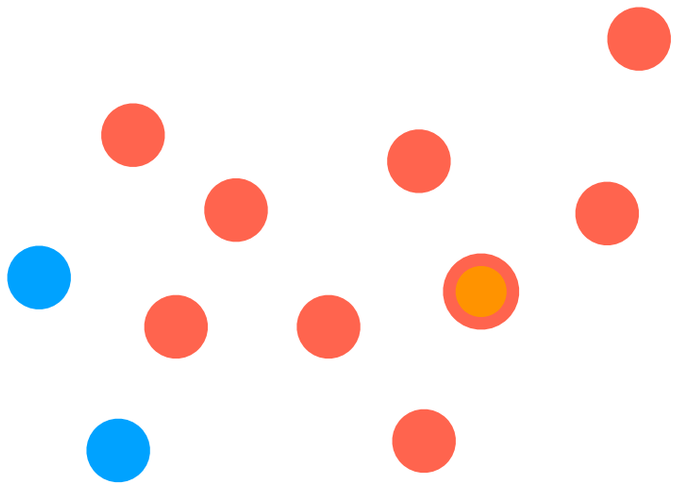
“Step 2b”. Assign closest points to current clusters

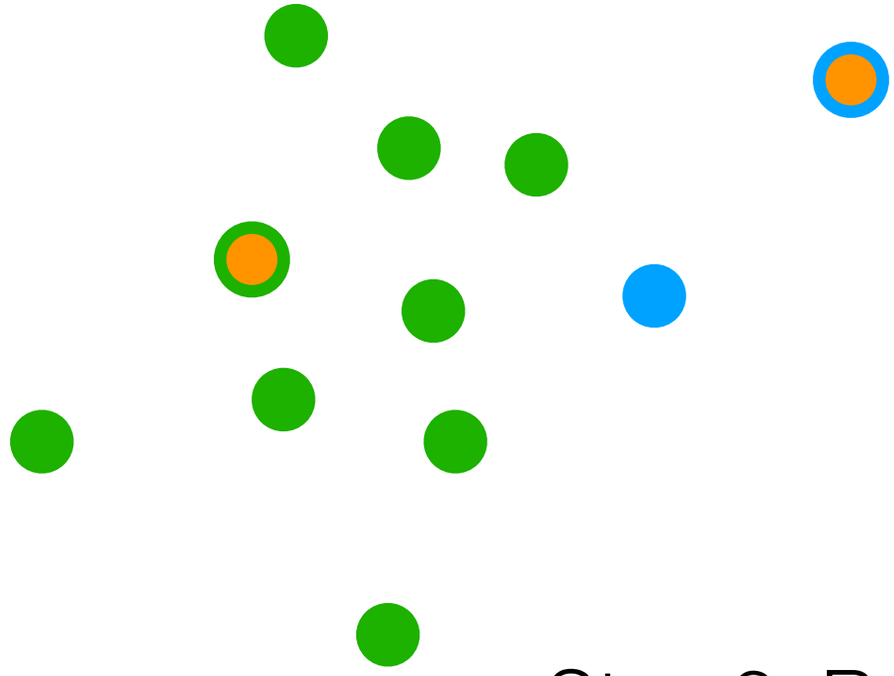
Step 3. Recompute cluster centers

DP-means

Step 0. Pick concentration parameter $\lambda > 0$

Step 1. Start with everything in same cluster





Step 3. Recompute cluster centers

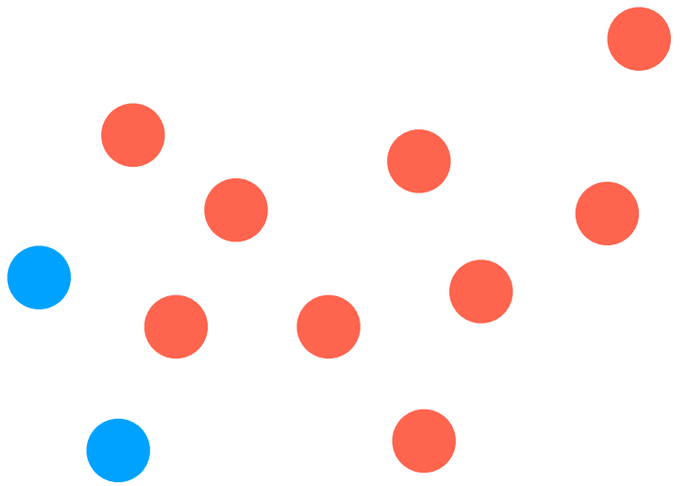
“Step 2a”. Pick point outside of gray coverage to make new cluster

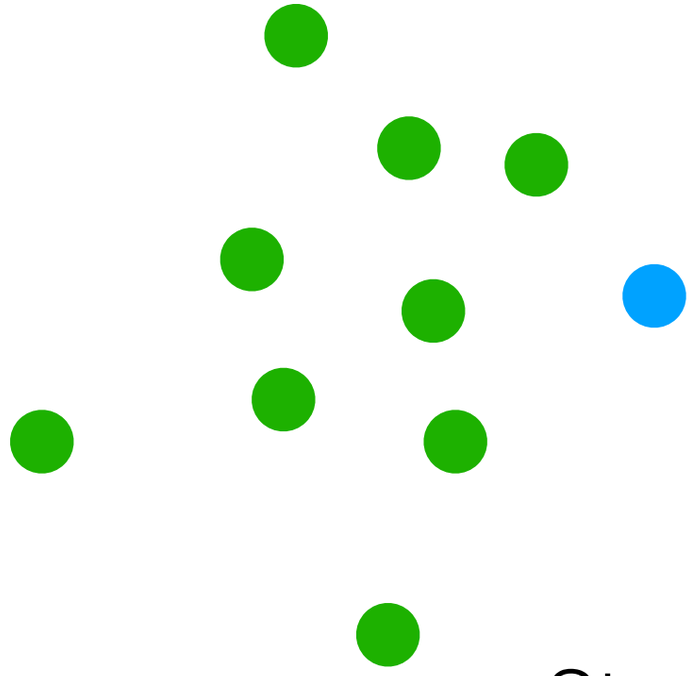
“Step 2b”. Assign closest points to current clusters

DP-means

Step 0. Pick concentration parameter $\lambda > 0$

Step 1. Start with everything in same cluster





Step 3. Recompute cluster centers

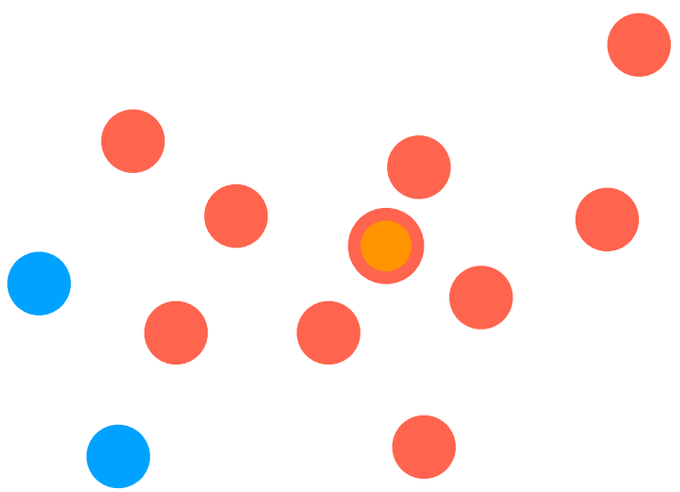
“Step 2a”. Pick point outside of gray coverage to make new cluster

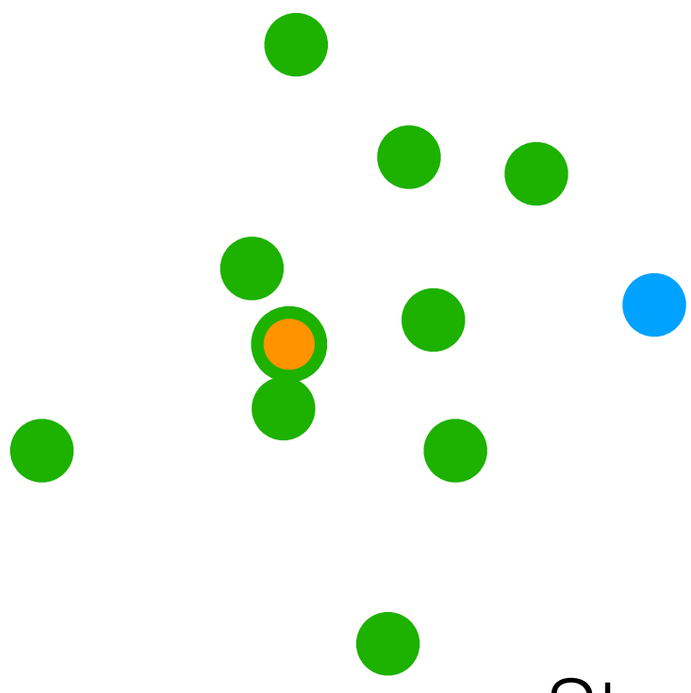
“Step 2b”. Assign closest points to current clusters

DP-means

Step 0. Pick concentration parameter $\lambda > 0$

Step 1. Start with everything in same cluster





Step 3. Recompute cluster centers

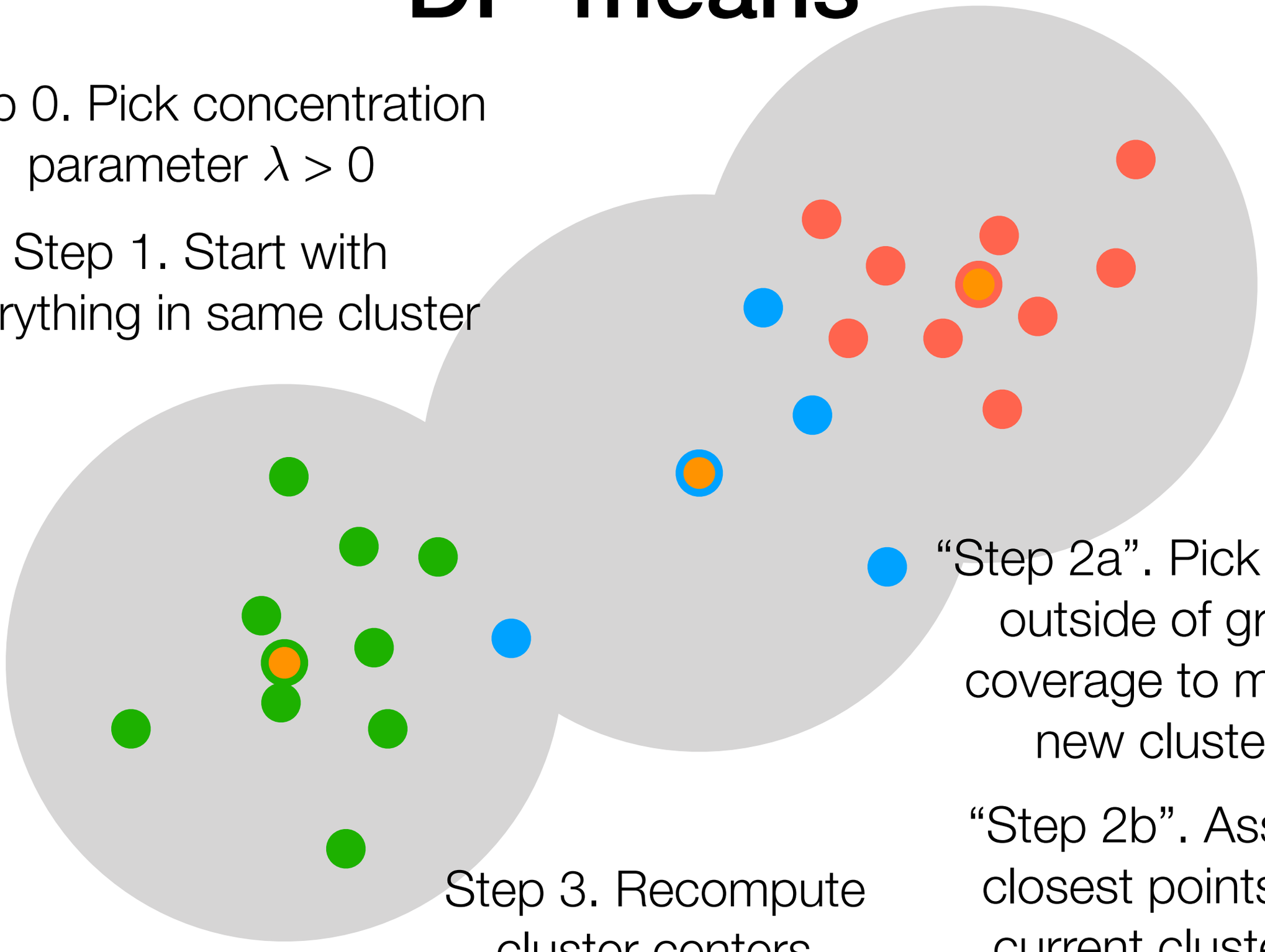
“Step 2a”. Pick point outside of gray coverage to make new cluster

“Step 2b”. Assign closest points to current clusters

DP-means

Step 0. Pick concentration parameter $\lambda > 0$

Step 1. Start with everything in same cluster



Step 3. Recompute cluster centers

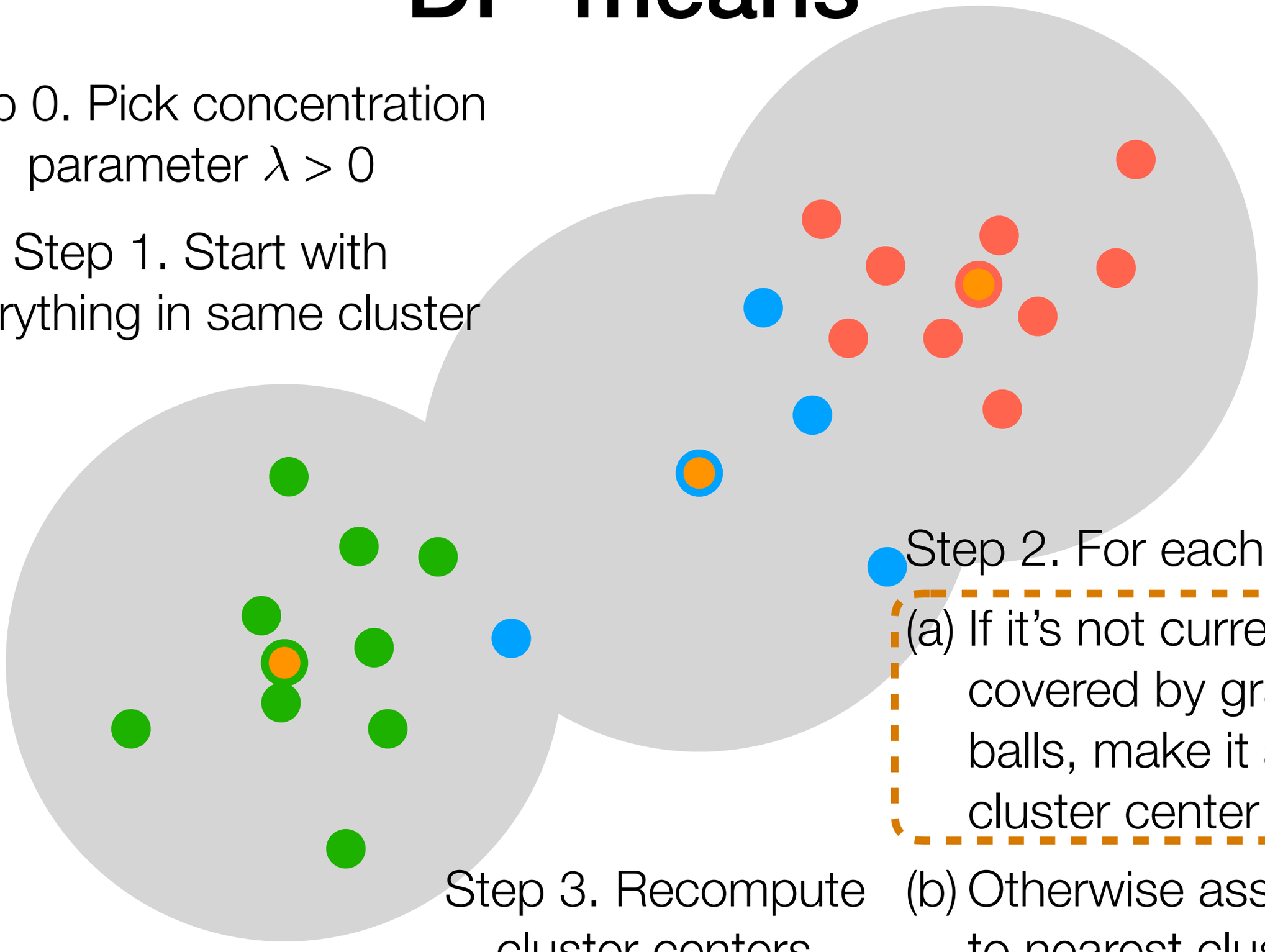
“Step 2a”. Pick point outside of gray coverage to make new cluster

“Step 2b”. Assign closest points to current clusters

DP-means

Step 0. Pick concentration parameter $\lambda > 0$

Step 1. Start with everything in same cluster



Step 2. For each point:

(a) If it's not currently covered by gray balls, make it a new cluster center

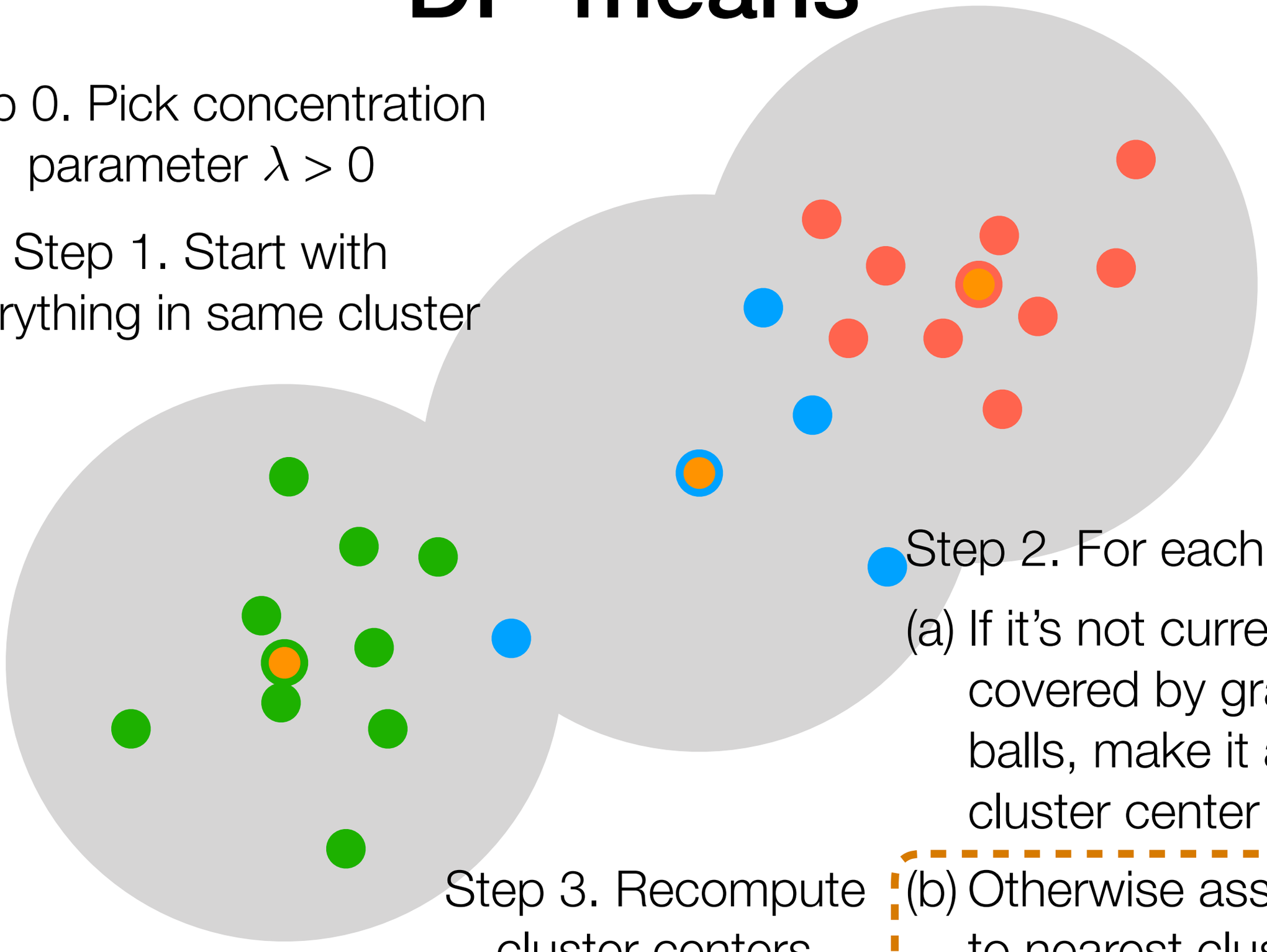
(b) Otherwise assign it to nearest cluster

Step 3. Recompute cluster centers

DP-means

Step 0. Pick concentration parameter $\lambda > 0$

Step 1. Start with everything in same cluster



Step 2. For each point:

(a) If it's not currently covered by gray balls, make it a new cluster center

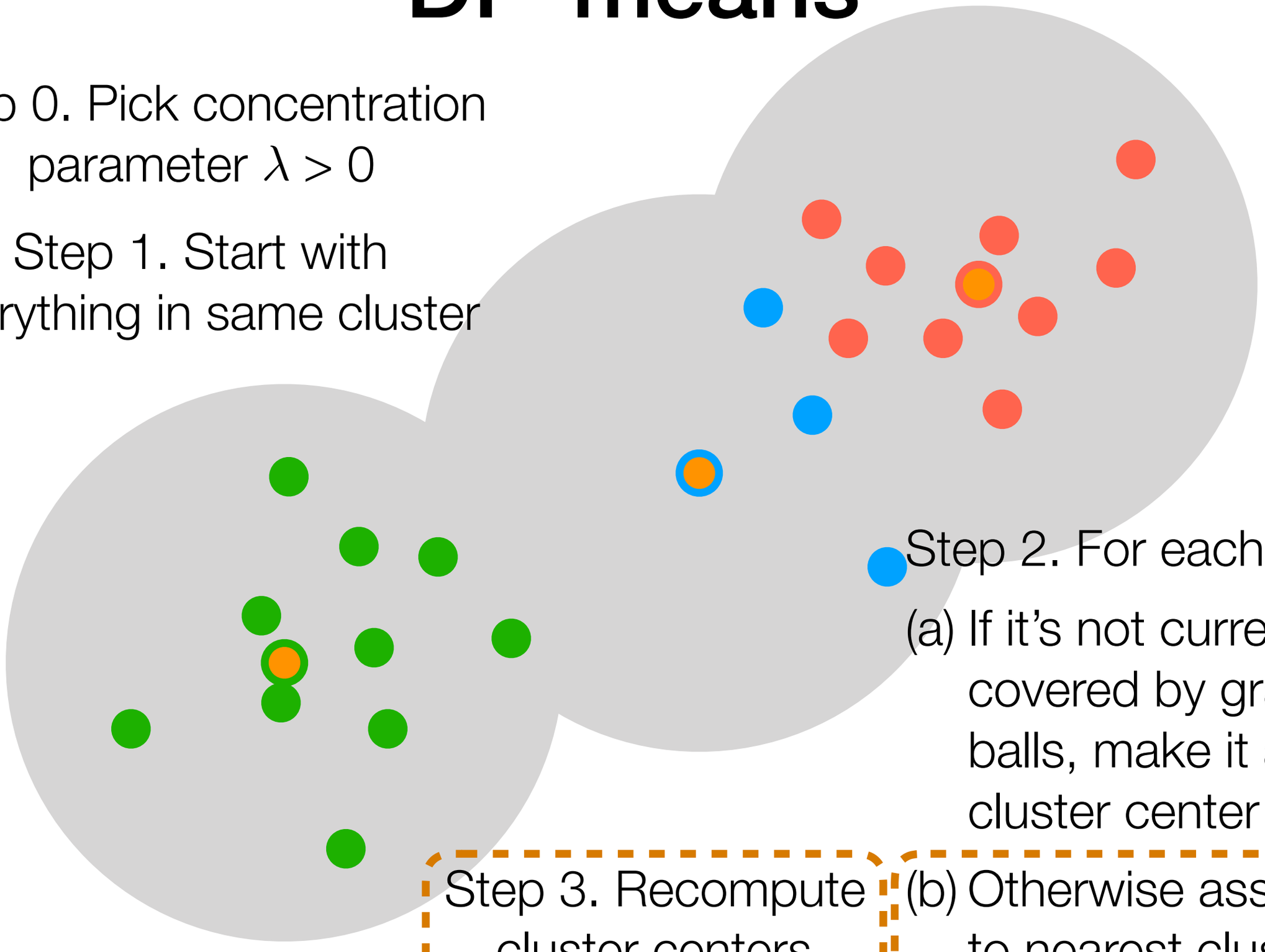
(b) Otherwise assign it to nearest cluster

Step 3. Recompute cluster centers

DP-means

Step 0. Pick concentration parameter $\lambda > 0$

Step 1. Start with everything in same cluster



Step 2. For each point:

(a) If it's not currently covered by gray balls, make it a new cluster center

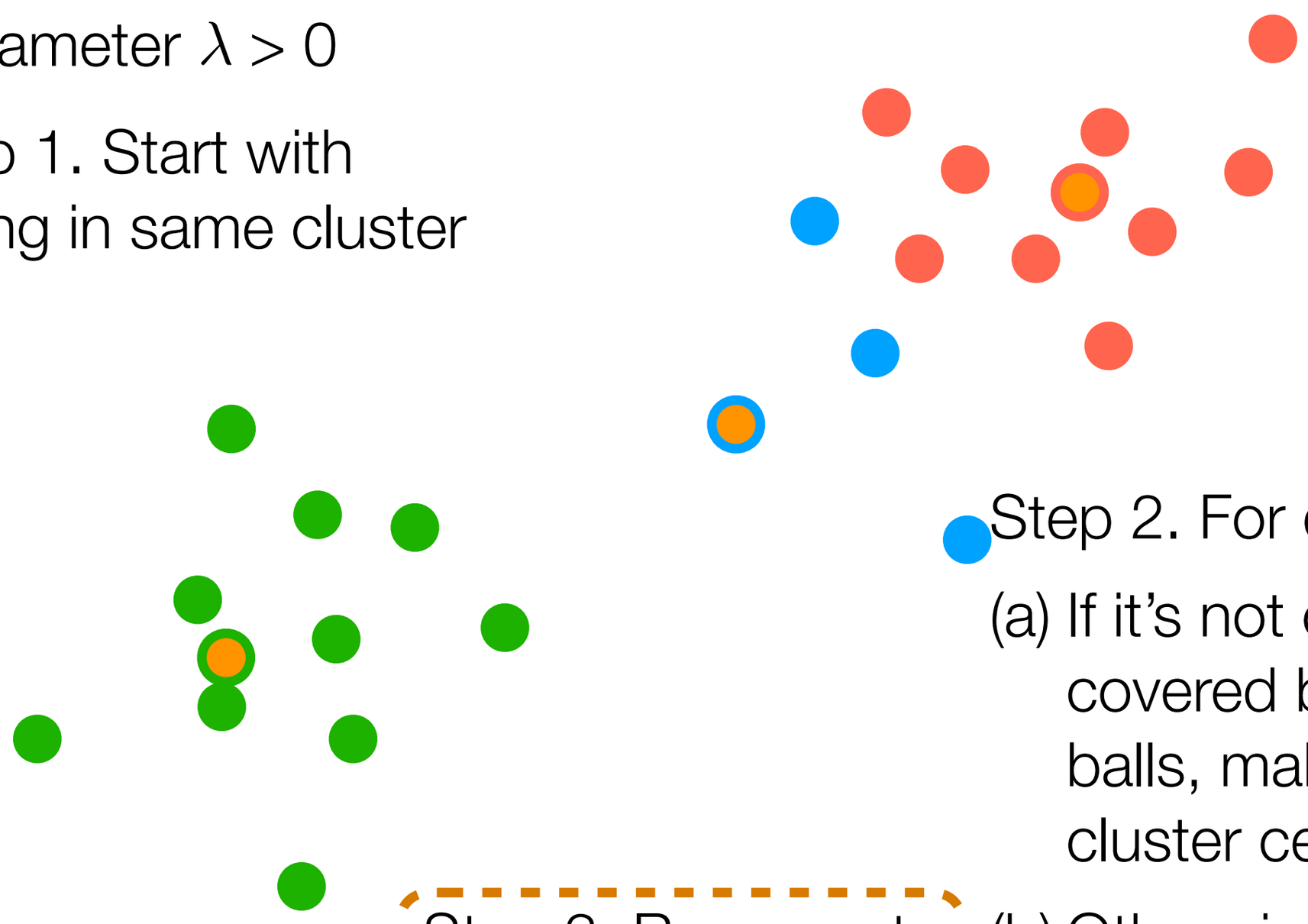
(b) Otherwise assign it to nearest cluster

Step 3. Recompute cluster centers

DP-means

Step 0. Pick concentration parameter $\lambda > 0$

Step 1. Start with everything in same cluster



Step 2. For each point:

(a) If it's not currently covered by gray balls, make it a new cluster center

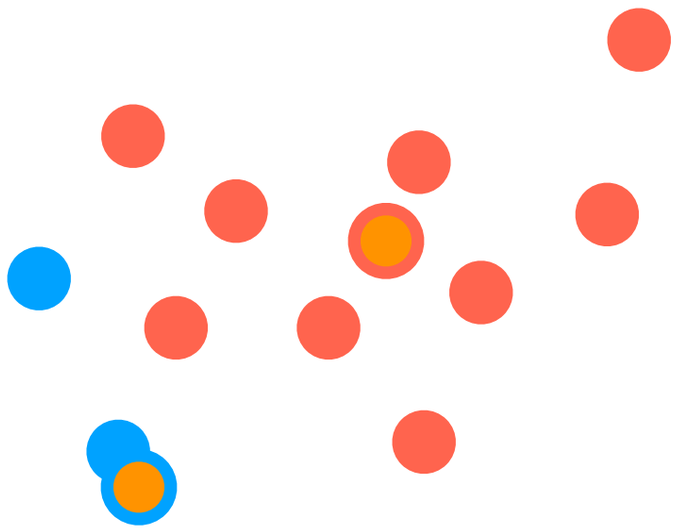
(b) Otherwise assign it to nearest cluster

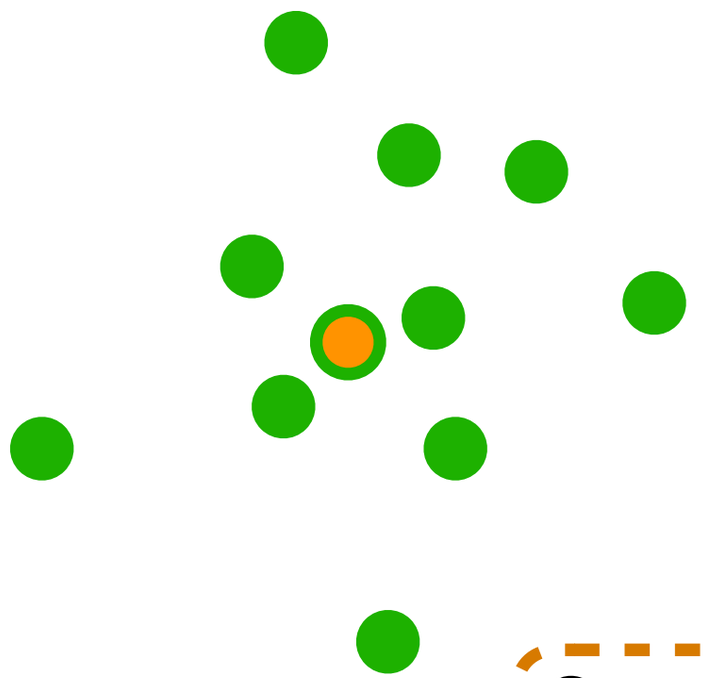
Step 3. Recompute cluster centers

DP-means

Step 0. Pick concentration parameter $\lambda > 0$

Step 1. Start with everything in same cluster





Step 2. For each point:

(a) If it's not currently covered by gray balls, make it a new cluster center

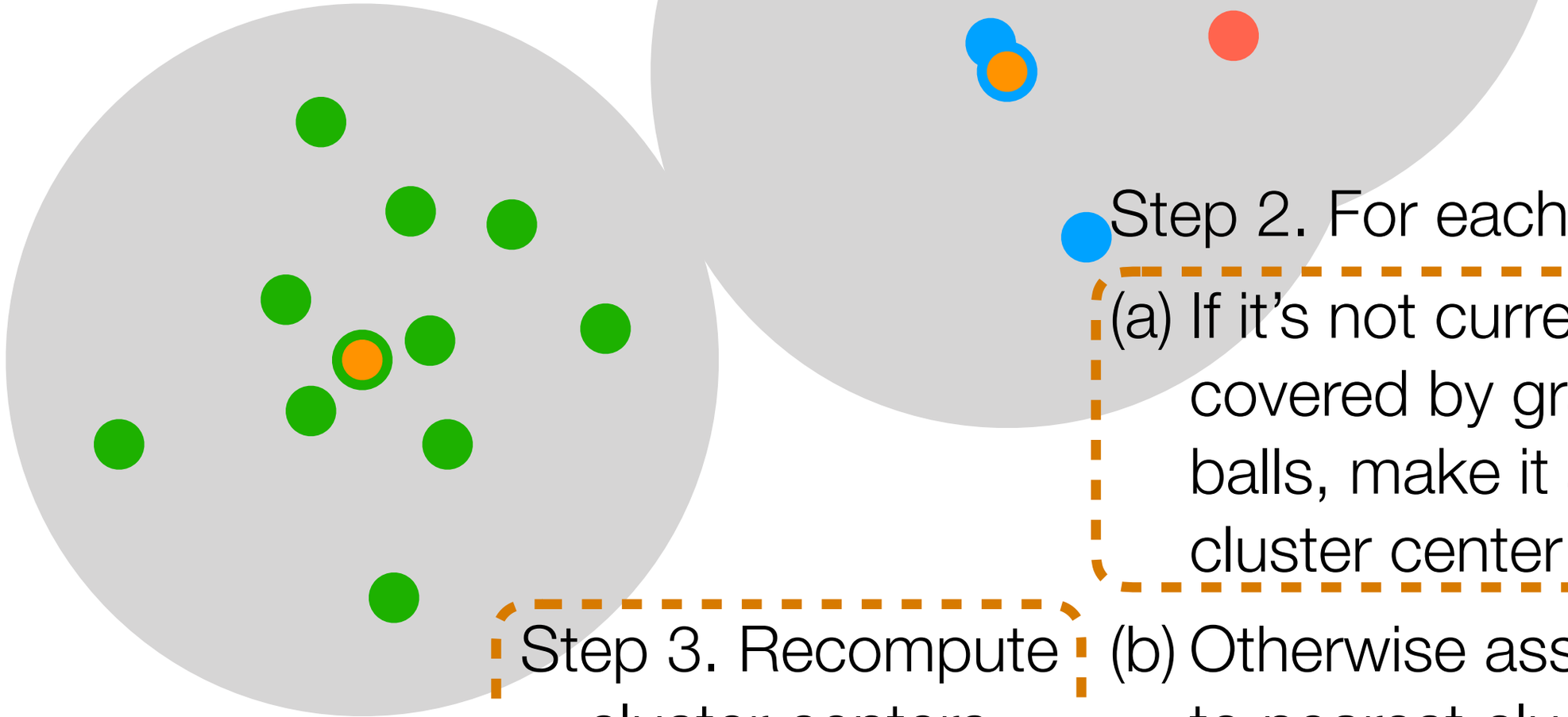
(b) Otherwise assign it to nearest cluster

Step 3. Recompute cluster centers

DP-means

Step 0: Pick concentration parameter $\lambda > 0$

Step 1: Start with everything in same cluster



Step 2. For each point:

(a) If it's not currently covered by gray balls, make it a new cluster center

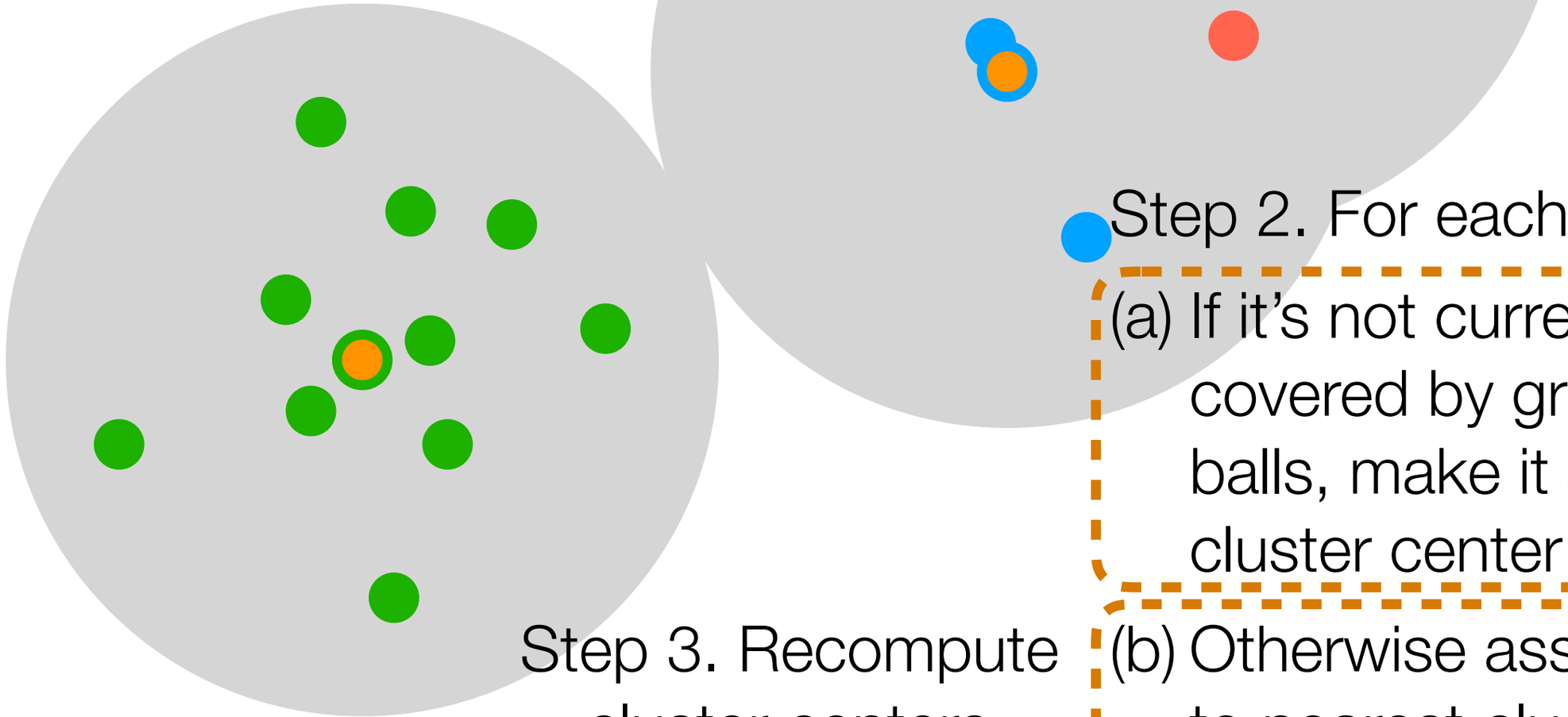
(b) Otherwise assign it to nearest cluster

Step 3. Recompute cluster centers

DP-means

Step 0: Pick concentration parameter $\lambda > 0$

Step 1: Start with everything in same cluster



Step 2. For each point:

(a) If it's not currently covered by gray balls, make it a new cluster center

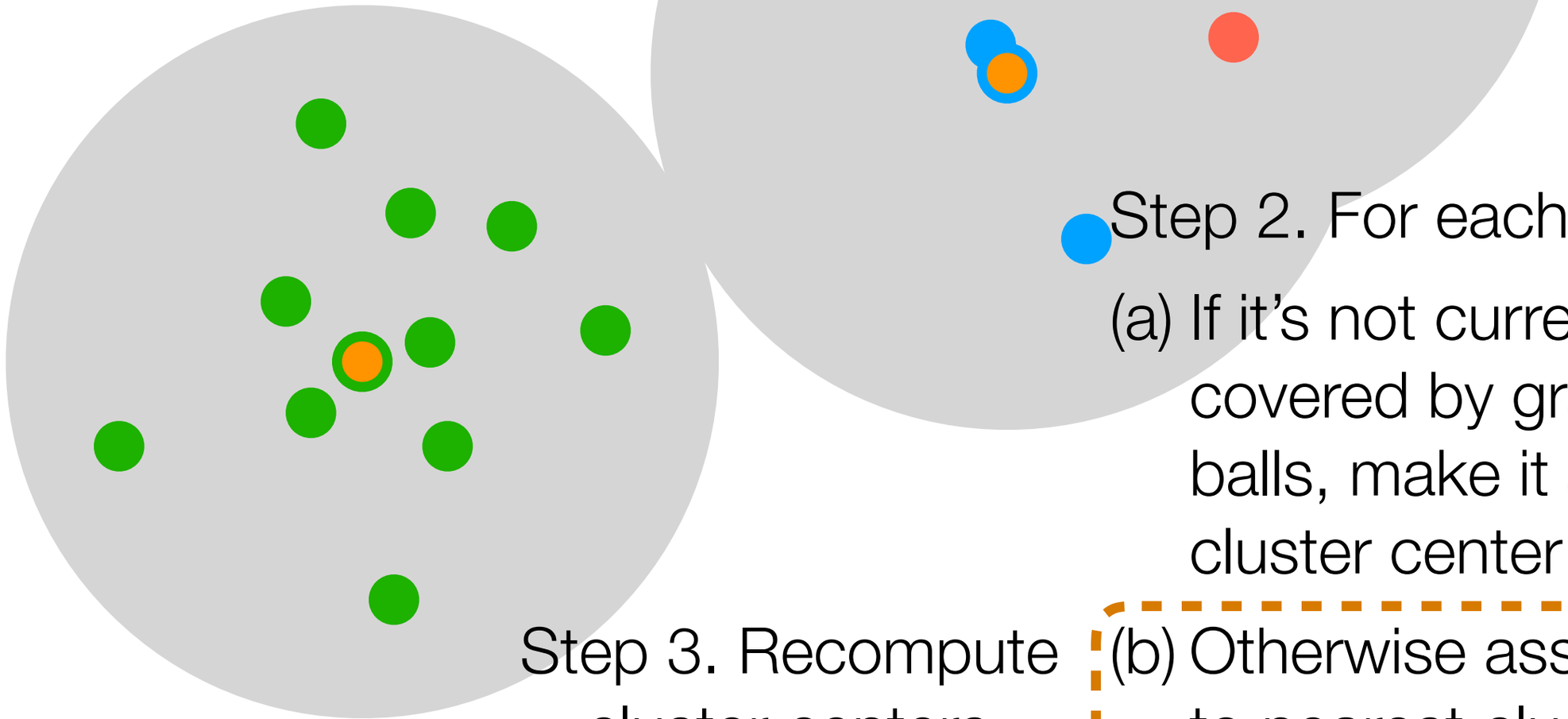
(b) Otherwise assign it to nearest cluster

Step 3. Recompute cluster centers

DP-means

Step 0: Pick concentration parameter $\lambda > 0$

Step 1: Start with everything in same cluster



Step 2. For each point:

(a) If it's not currently covered by gray balls, make it a new cluster center

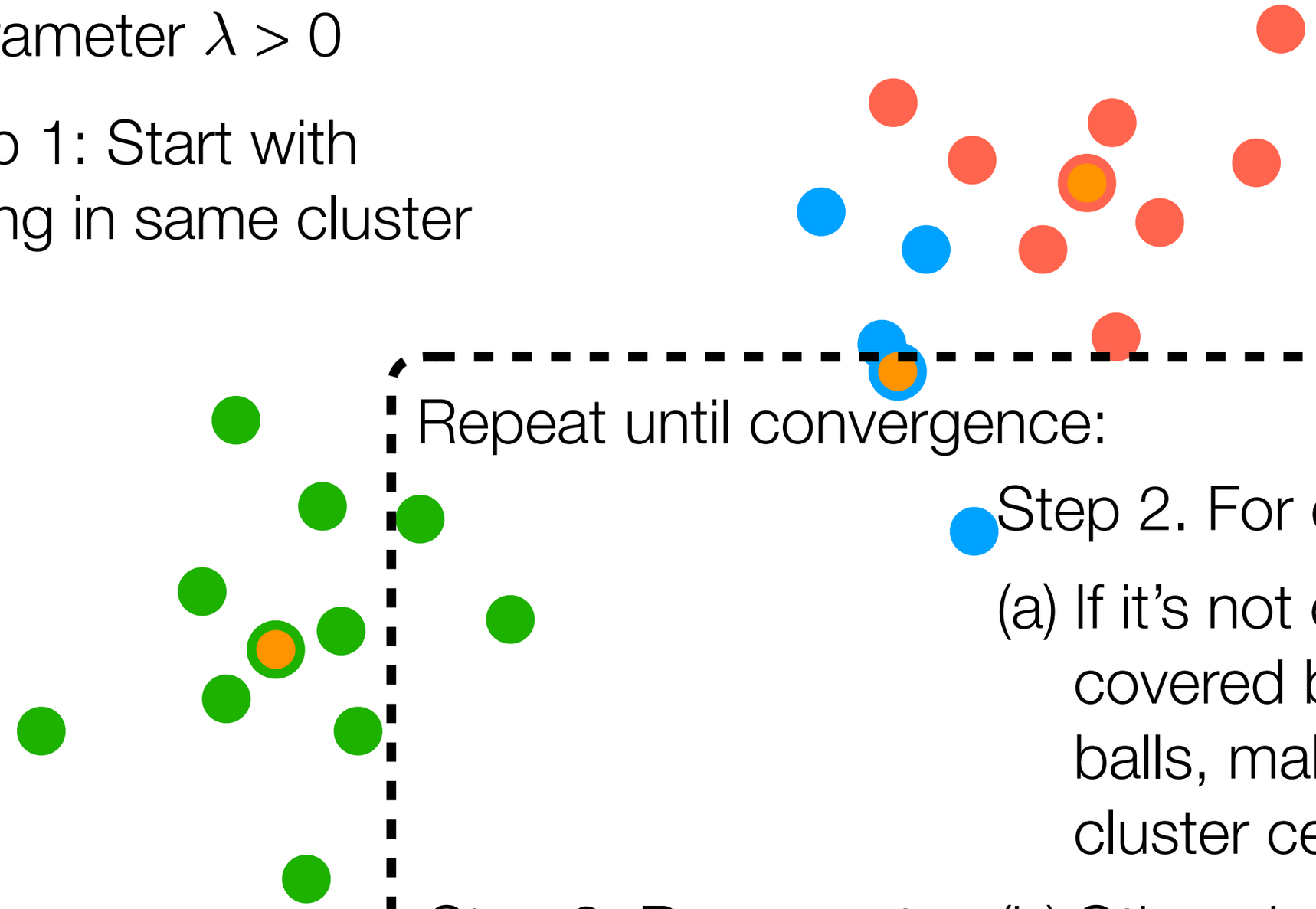
(b) Otherwise assign it to nearest cluster

Step 3. Recompute cluster centers

DP-means

Step 0: Pick concentration parameter $\lambda > 0$

Step 1: Start with everything in same cluster



Repeat until convergence:

Step 2. For each point:

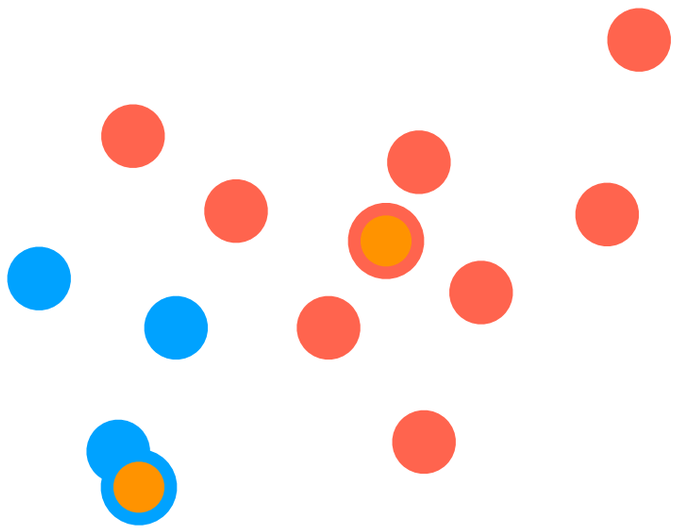
(a) If it's not currently covered by gray balls, make it a new cluster center

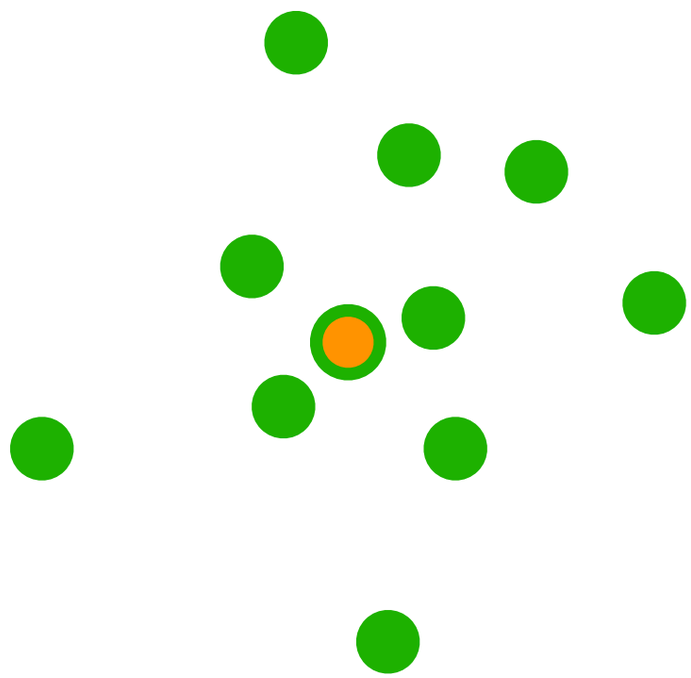
(b) Otherwise assign it to nearest cluster

Step 3. Recompute cluster centers

DP-means

As you saw in the DP-GMM demo (and is similar with DP-means), DP-means can produce a few extra small clusters

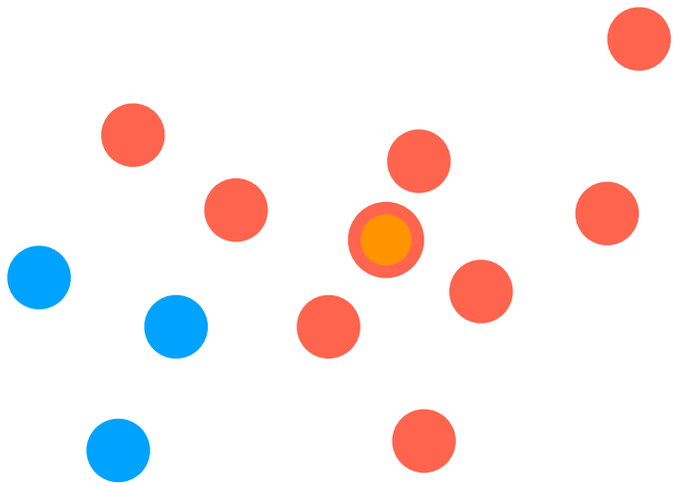


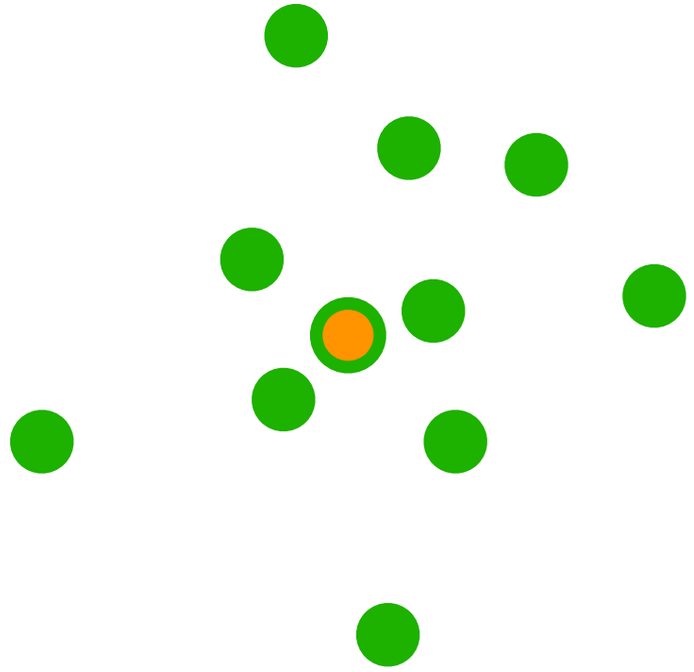


In practice: reassign points in small clusters to bigger clusters

DP-means

As you saw in the DP-GMM demo (and is similar with DP-means), DP-means can produce a few extra small clusters

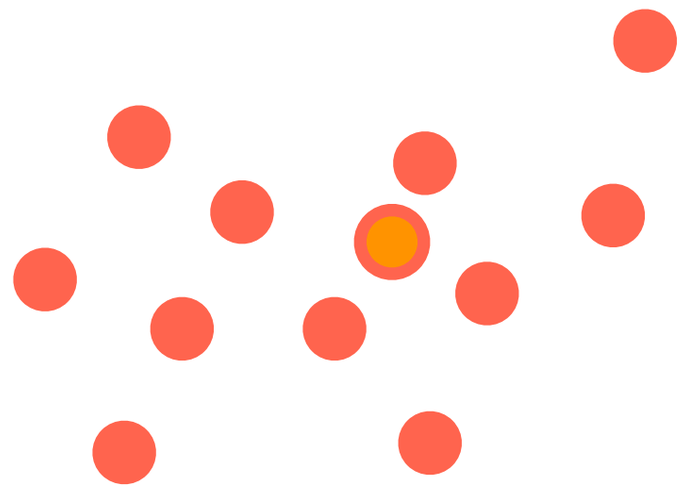


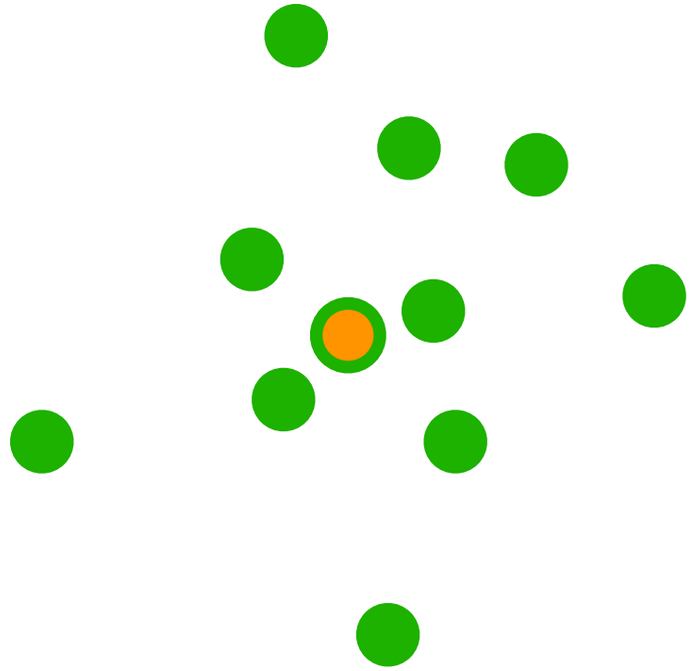


In practice: reassign points in small clusters to bigger clusters

DP-means

As you saw in the DP-GMM demo (and is similar with DP-means), DP-means can produce a few extra small clusters



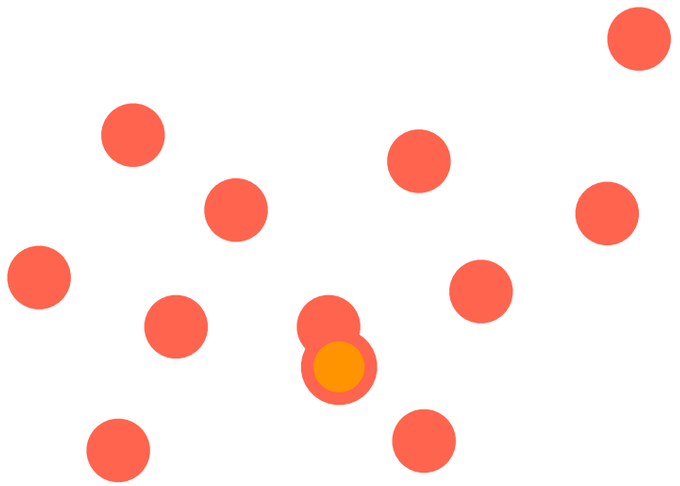


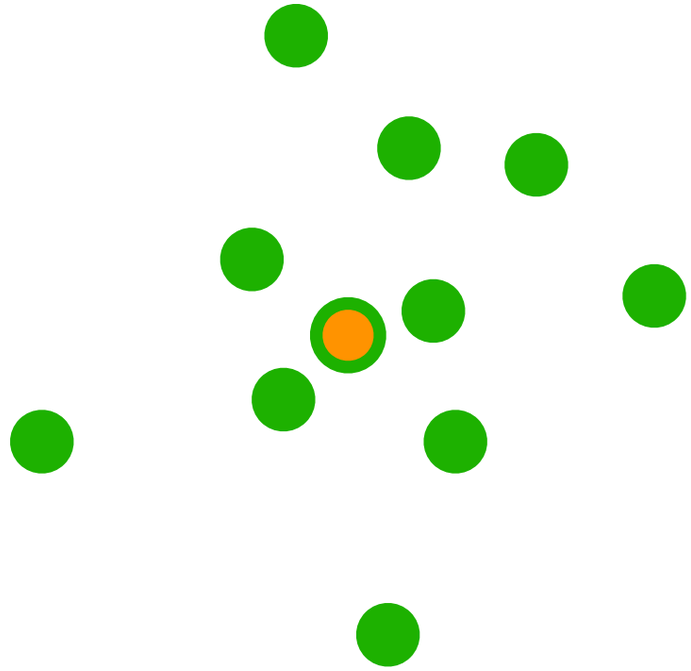
In practice: reassign points in small clusters to bigger clusters

Can recompute cluster centers

DP-means

As you saw in the DP-GMM demo (and is similar with DP-means), DP-means can produce a few extra small clusters





In practice: reassign points in small clusters to bigger clusters

Can recompute cluster centers

Big picture: DP-means & DP-GMM have a “concentration” parameter roughly controlling *size* of clusters rather than *number* of clusters

If your problem can more naturally be thought of as having cluster sizes that should not be too large, can use DP-means/DP-GMM instead of k-means/GMM

Real example. *Satellite image analysis of rural India to find villages*

Each cluster is a village: don’t know how many villages there are total but rough upper bound on radius of village can be specified

➔ DP-means provides a decent solution!

Other Ways for Choosing *k*

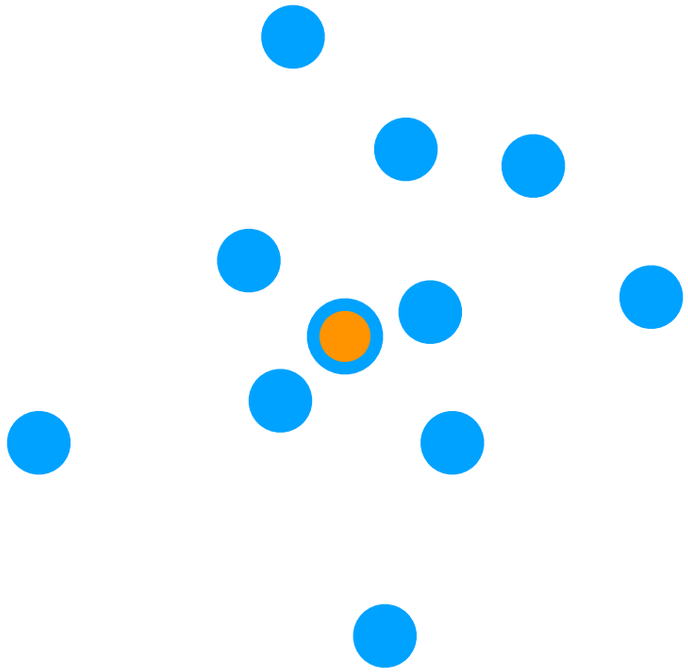
- Choose a cost function to compute for different *k*
  - In general, not easy! Need some intuition for what "good" clusters are
    - Ideally: cost function should relate to your application of interest
- Pick *k* achieving lowest cost

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

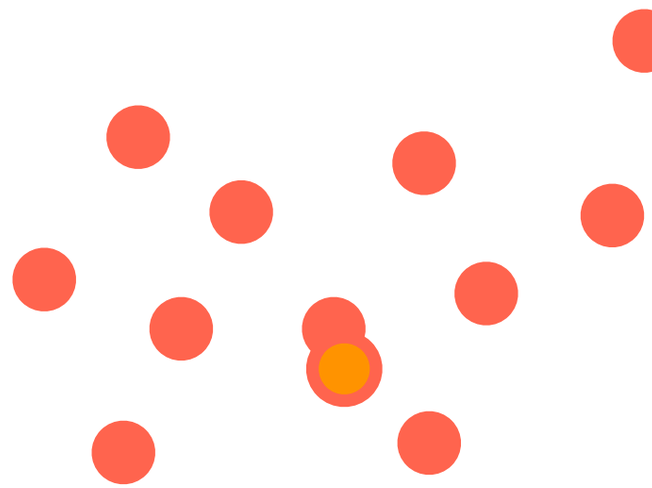
But hey it’s worth a shot

Residual Sum of Squares

Look at one cluster at a time



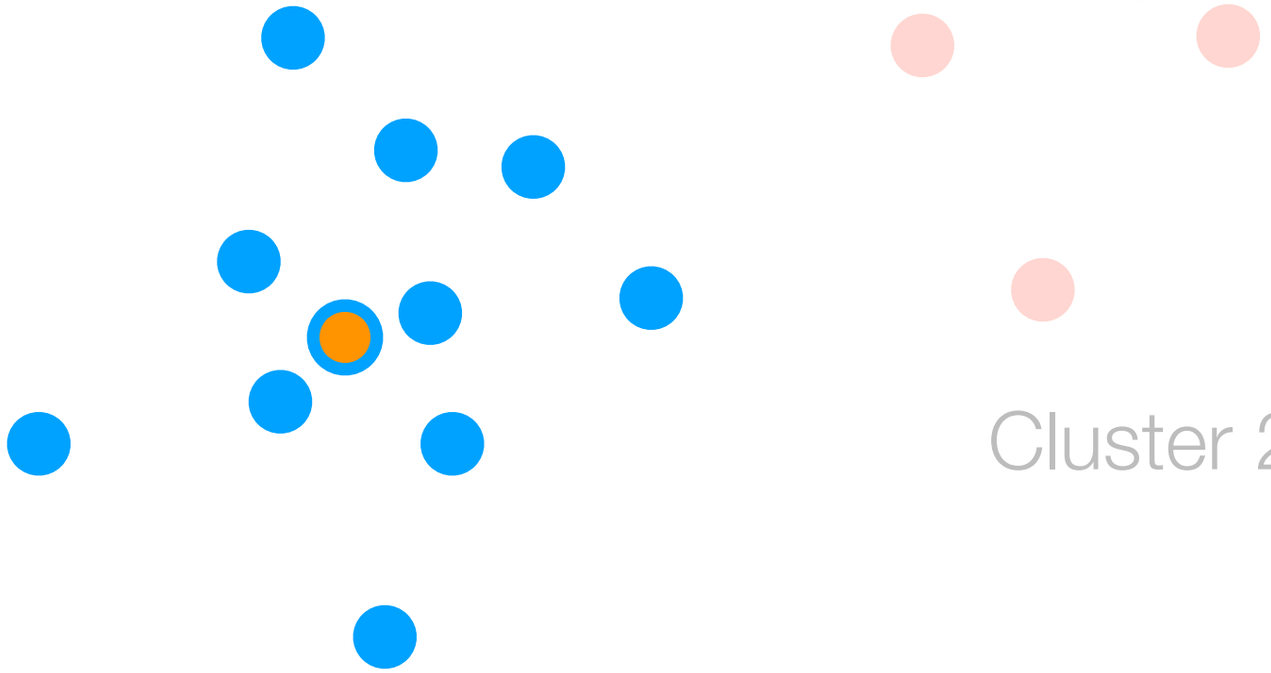
Cluster 1



Cluster 2

Residual Sum of Squares

Look at one cluster at a time



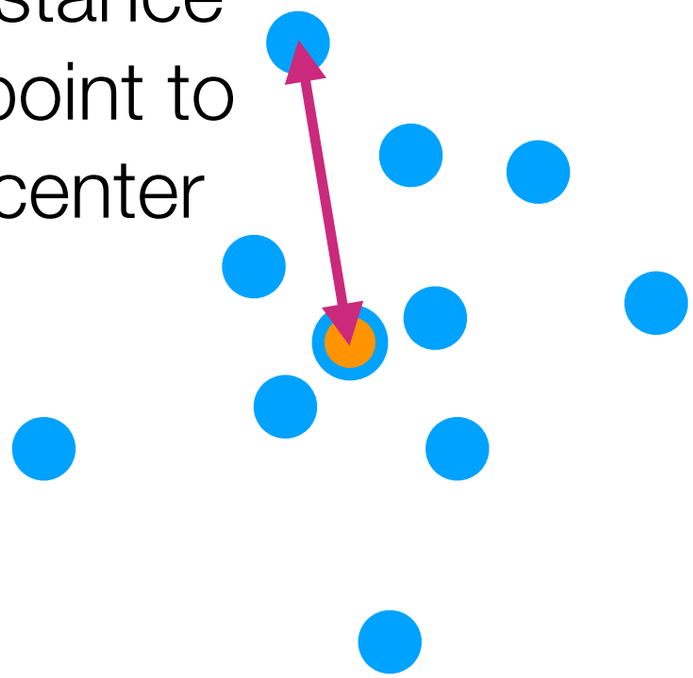
Cluster 1

Cluster 2

Residual Sum of Squares

Look at one cluster at a time

Measure distance from each point to its cluster center



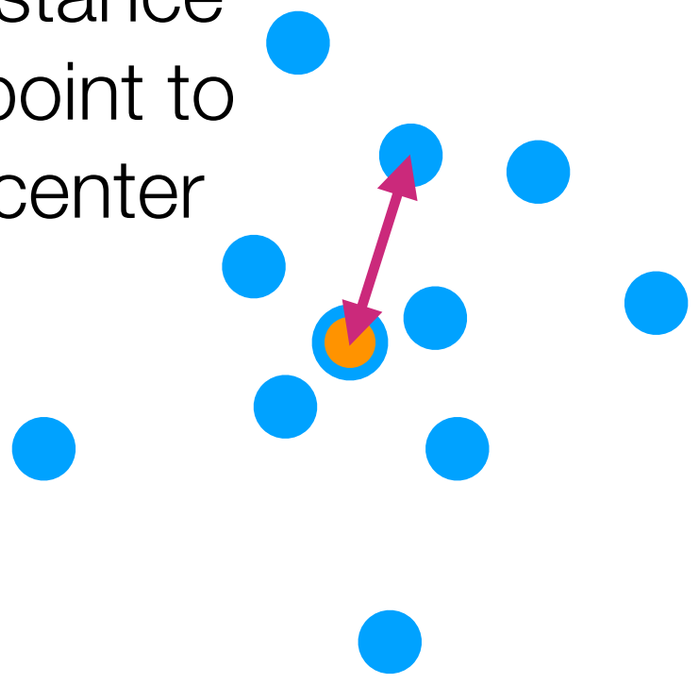
Cluster 1

Cluster 2

Residual Sum of Squares

Look at one cluster at a time

Measure distance from each point to its cluster center



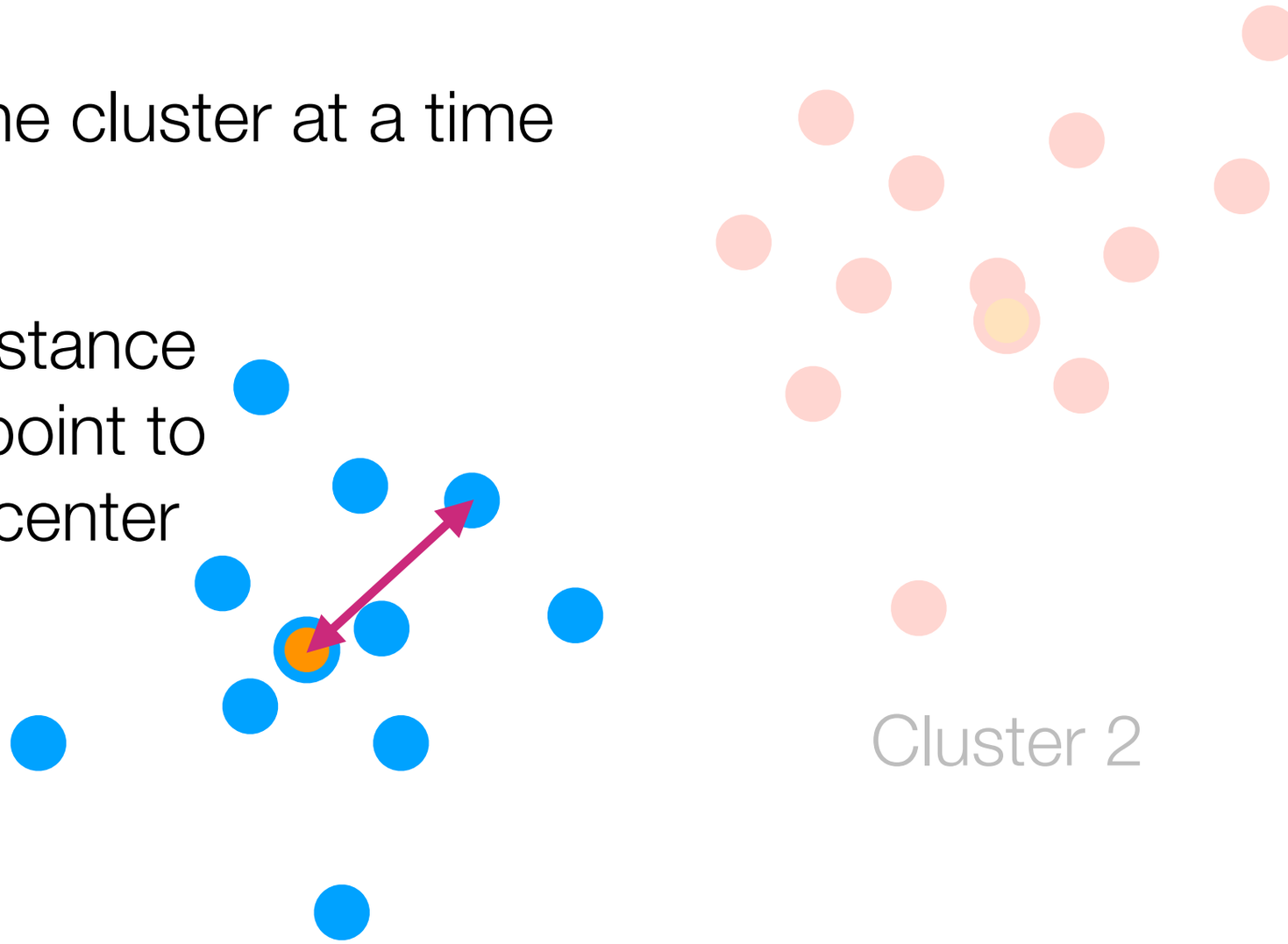
Cluster 1

Cluster 2

Residual Sum of Squares

Look at one cluster at a time

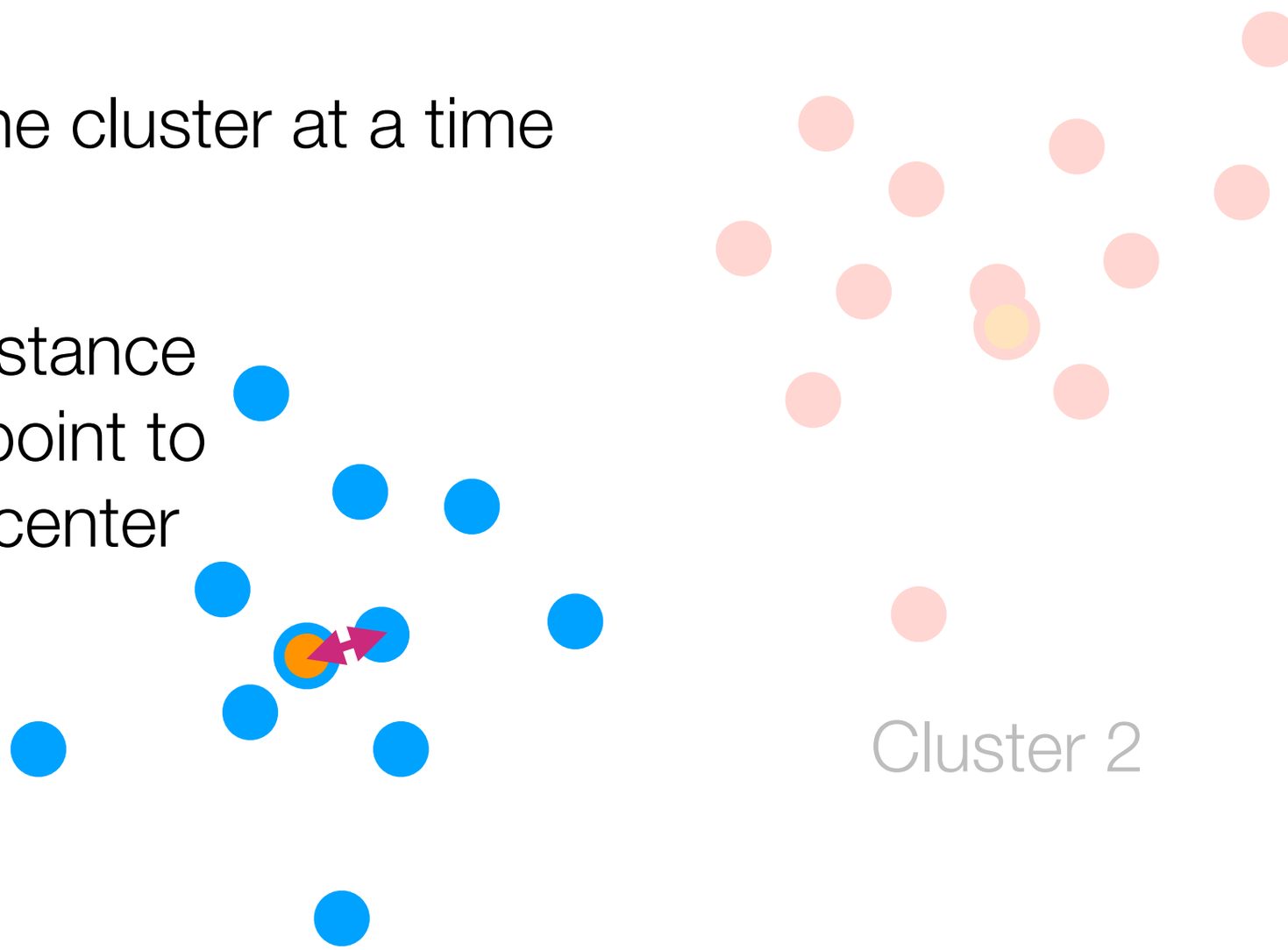
Measure distance from each point to its cluster center



Residual Sum of Squares

Look at one cluster at a time

Measure distance from each point to its cluster center



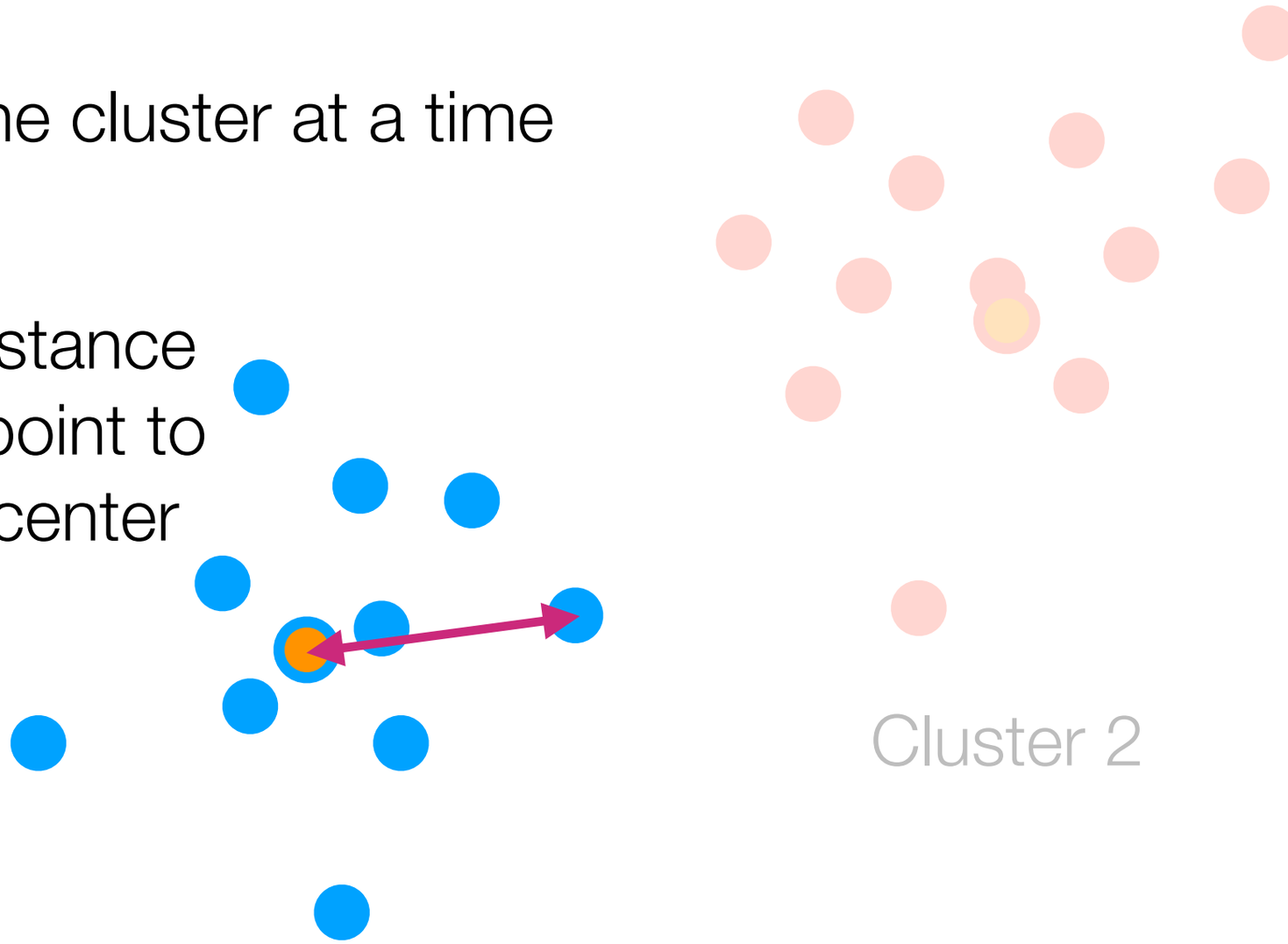
Cluster 1

Cluster 2

Residual Sum of Squares

Look at one cluster at a time

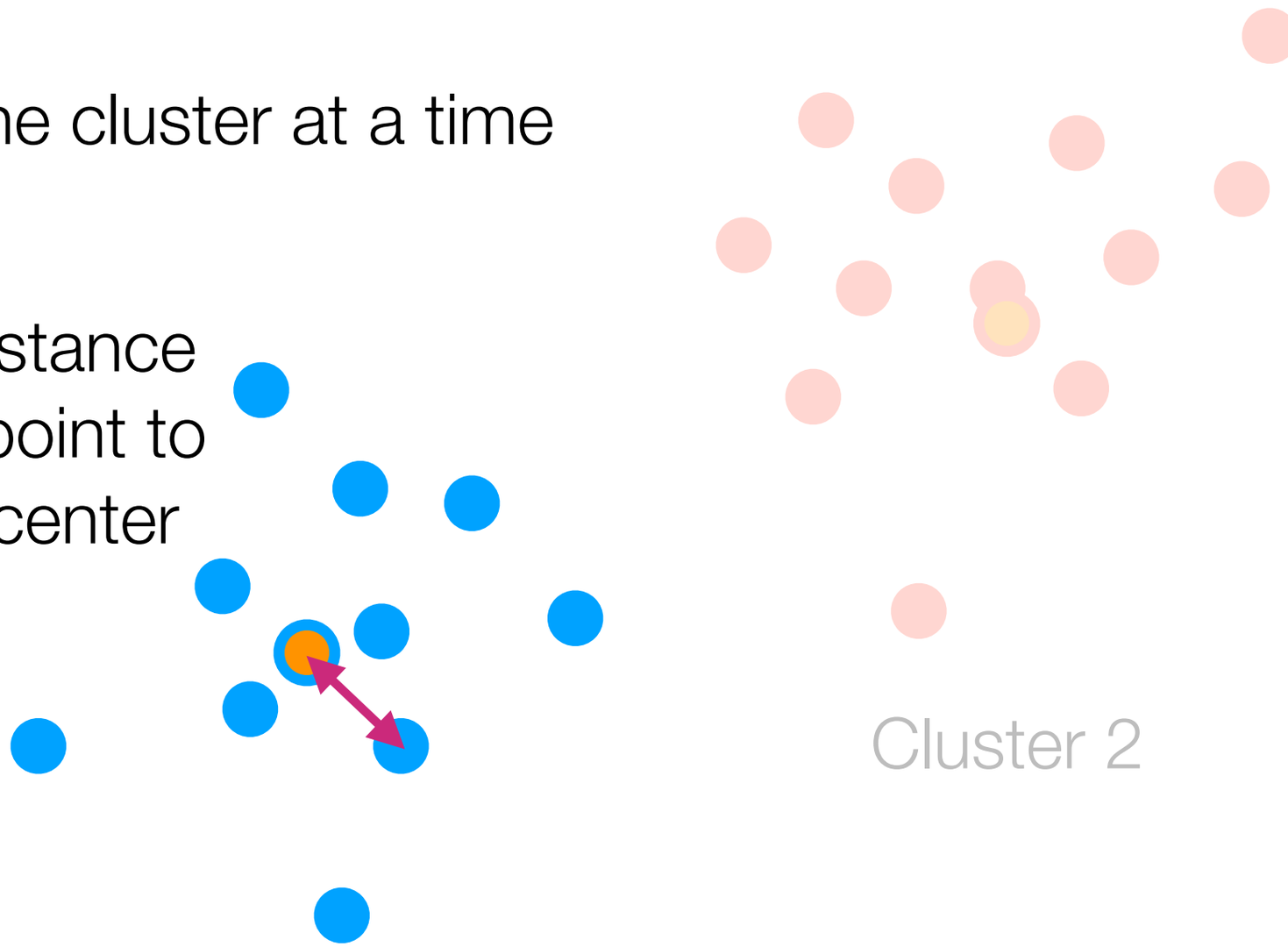
Measure distance from each point to its cluster center



Residual Sum of Squares

Look at one cluster at a time

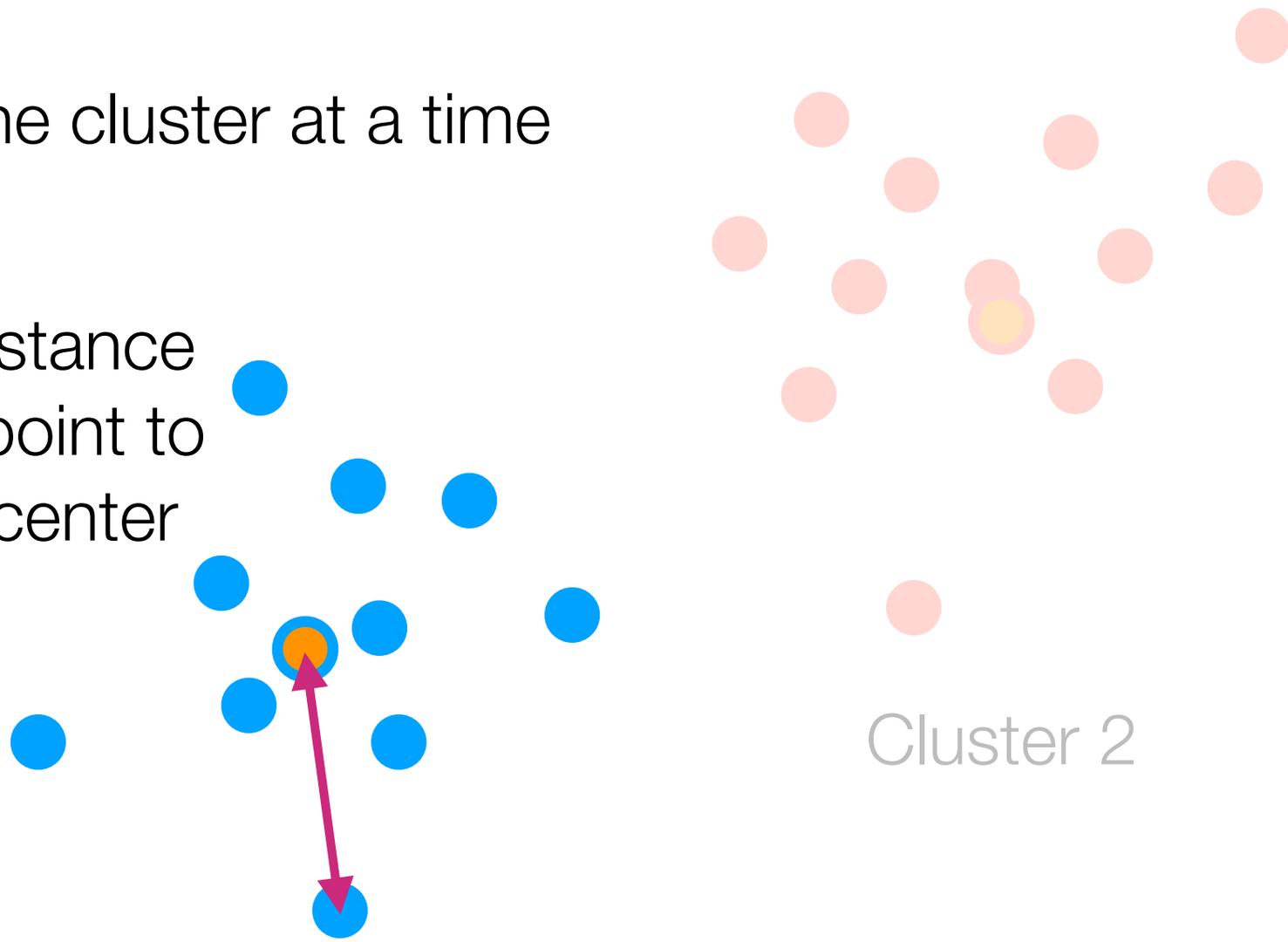
Measure distance from each point to its cluster center



Residual Sum of Squares

Look at one cluster at a time

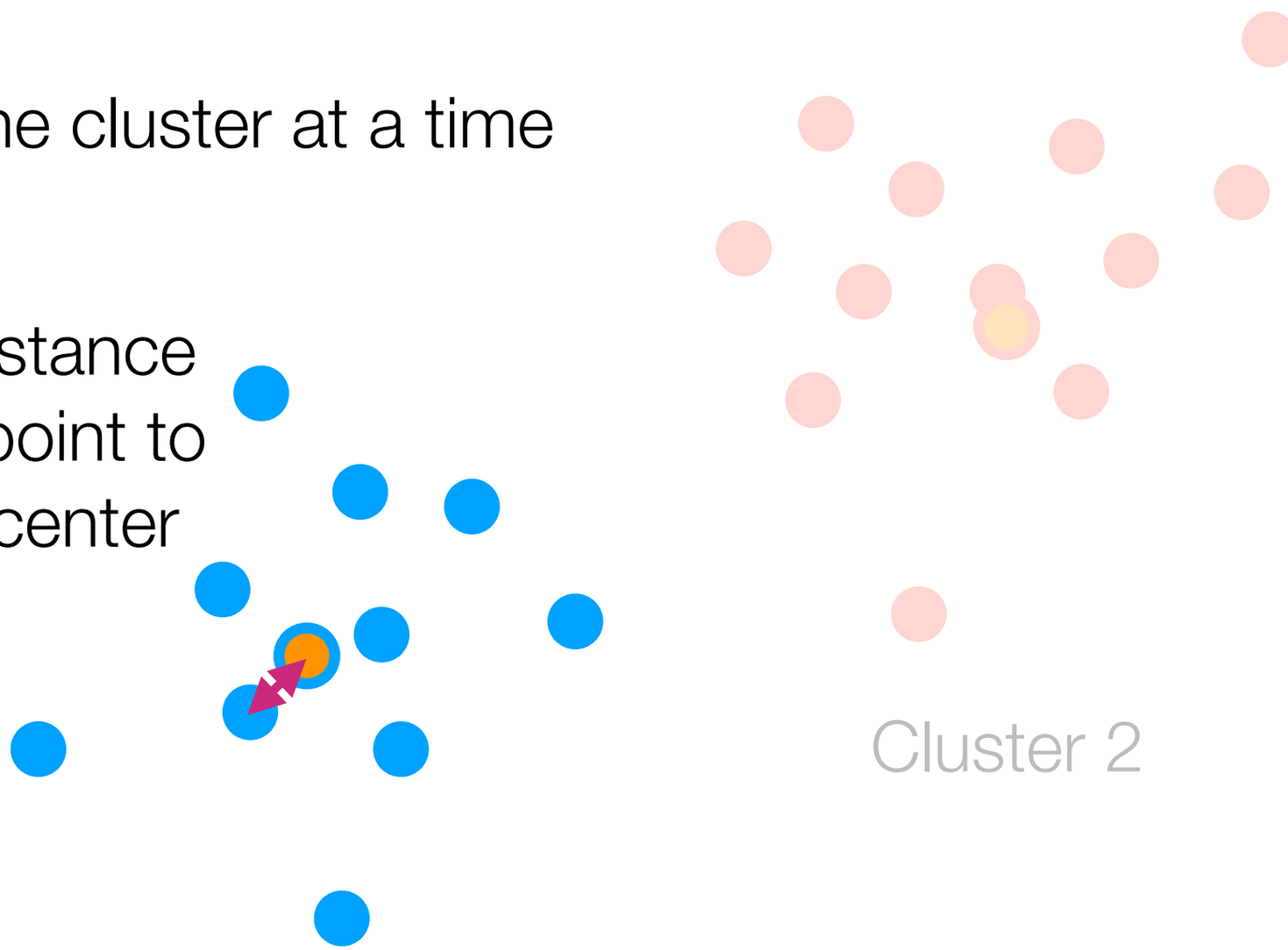
Measure distance from each point to its cluster center



Residual Sum of Squares

Look at one cluster at a time

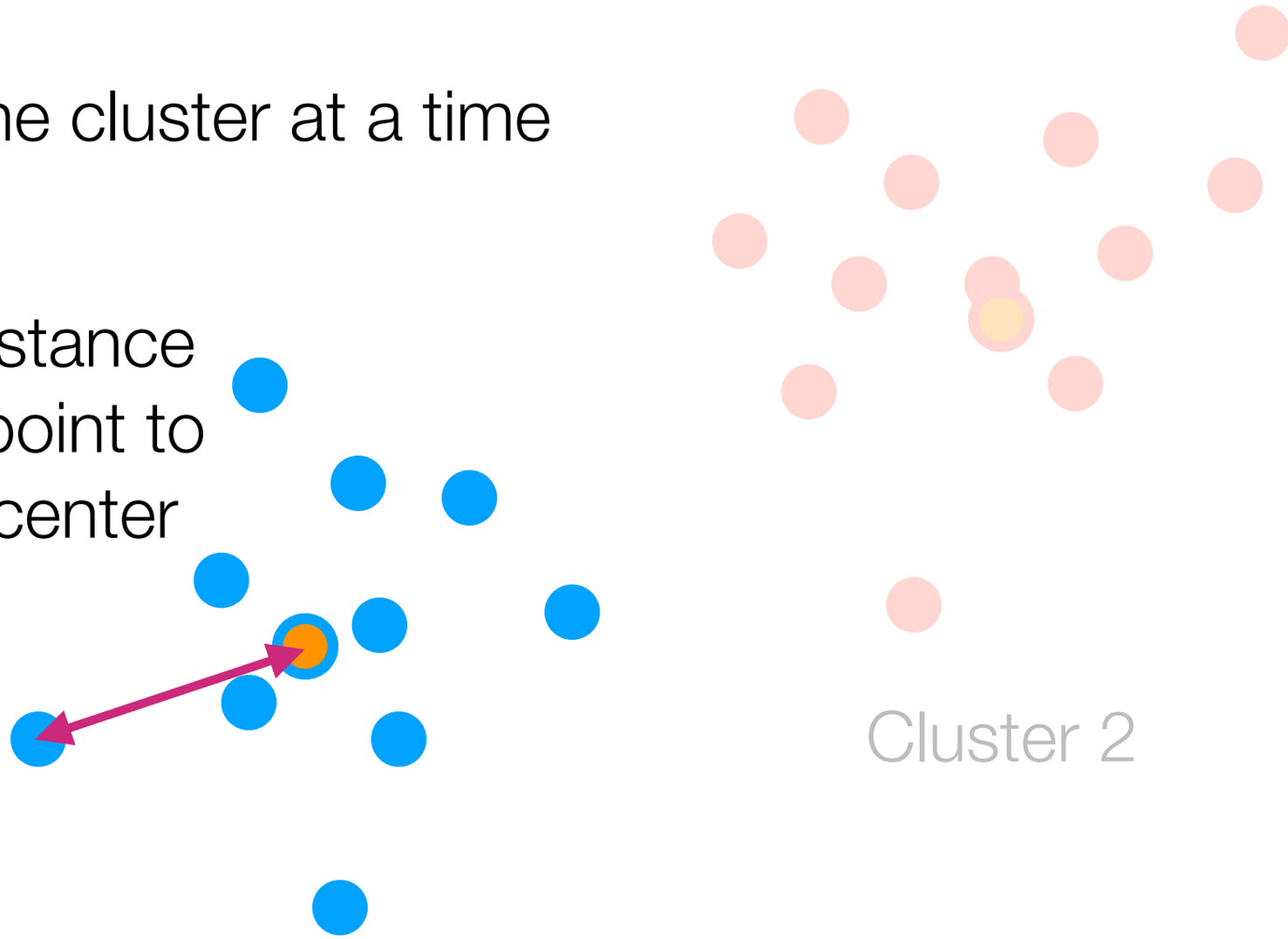
Measure distance from each point to its cluster center



Residual Sum of Squares

Look at one cluster at a time

Measure distance from each point to its cluster center



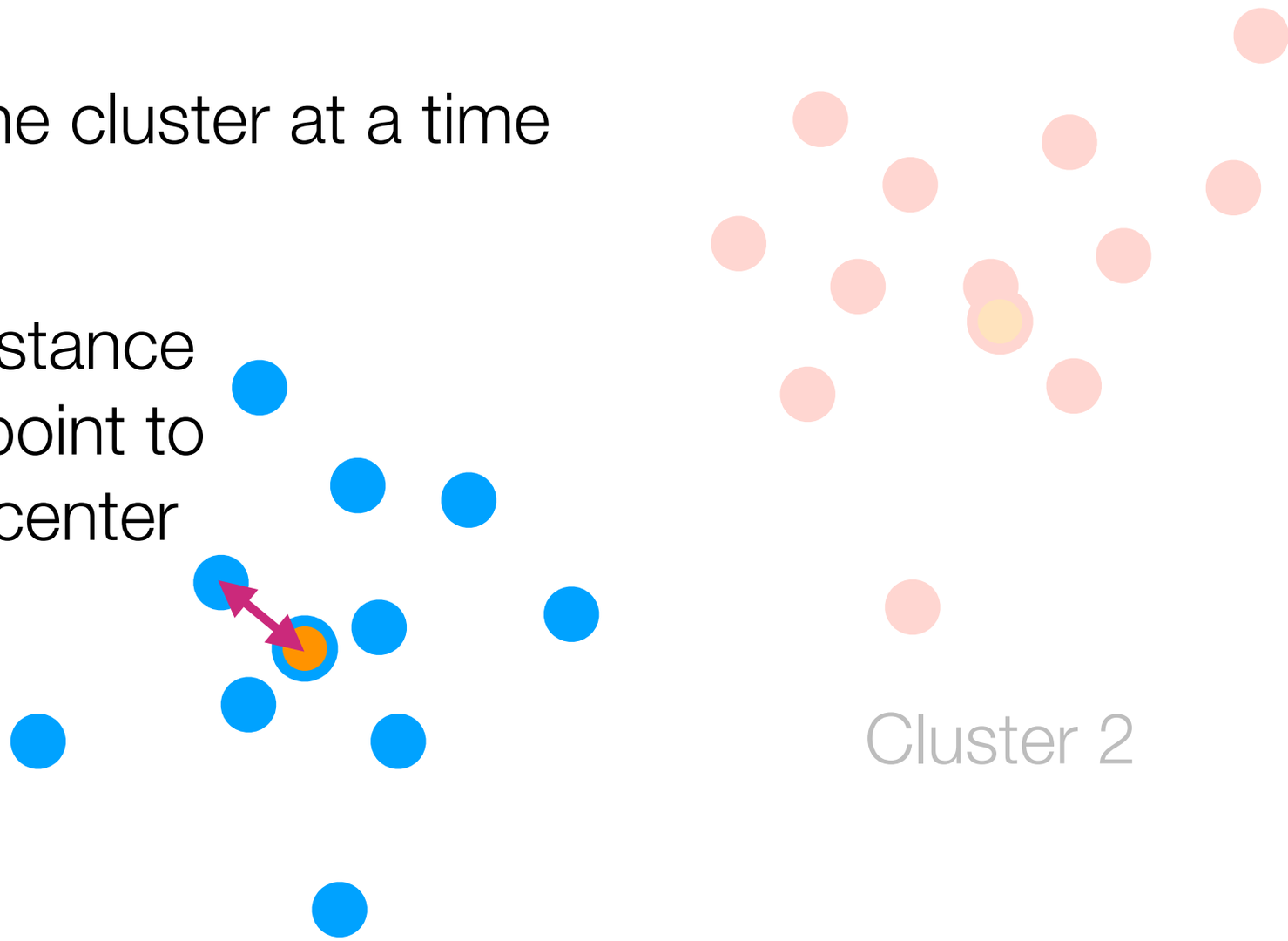
Cluster 1

Cluster 2

Residual Sum of Squares

Look at one cluster at a time

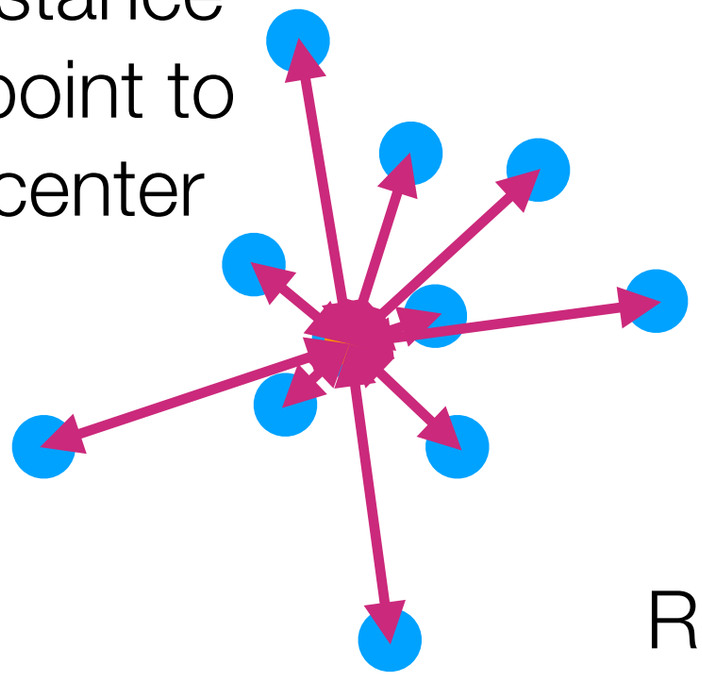
Measure distance from each point to its cluster center



Residual Sum of Squares

Look at one cluster at a time

Measure distance from each point to its cluster center



Cluster 1

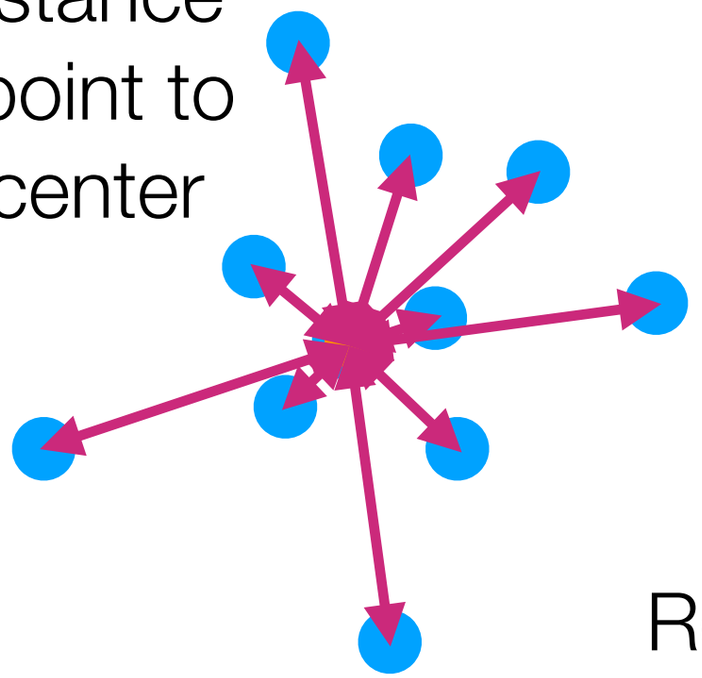
Cluster 2

Residual sum of squares 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

Cluster 2

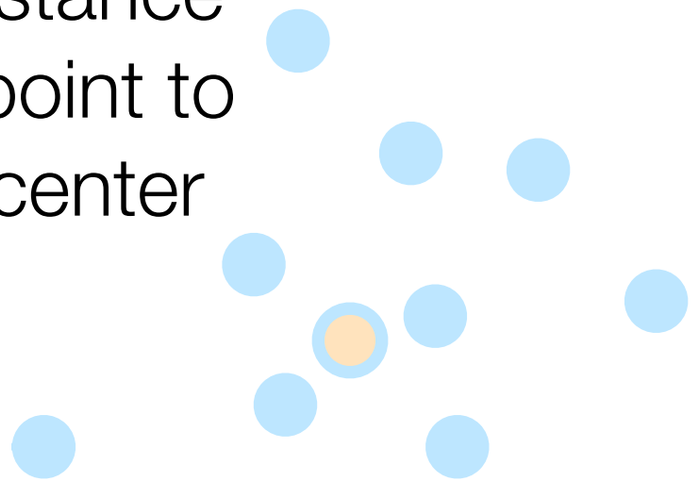
Residual sum of squares for cluster 1:

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

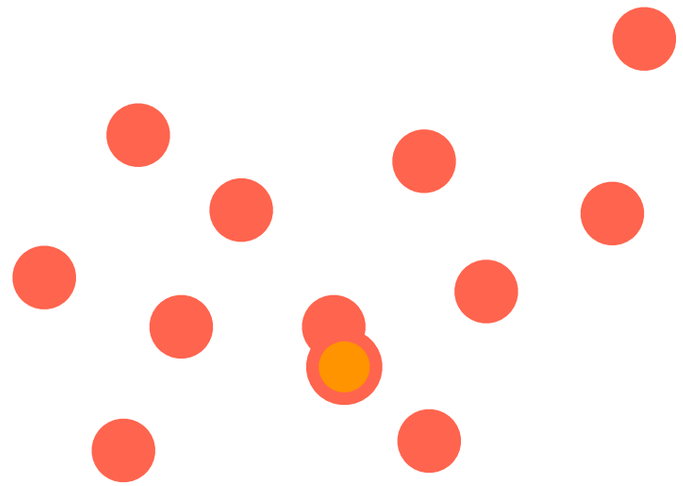
Residual Sum of Squares

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

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

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 RSS not a good way to choose *k*?

What is RSS 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*

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

μ: 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)

Hierarchical 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

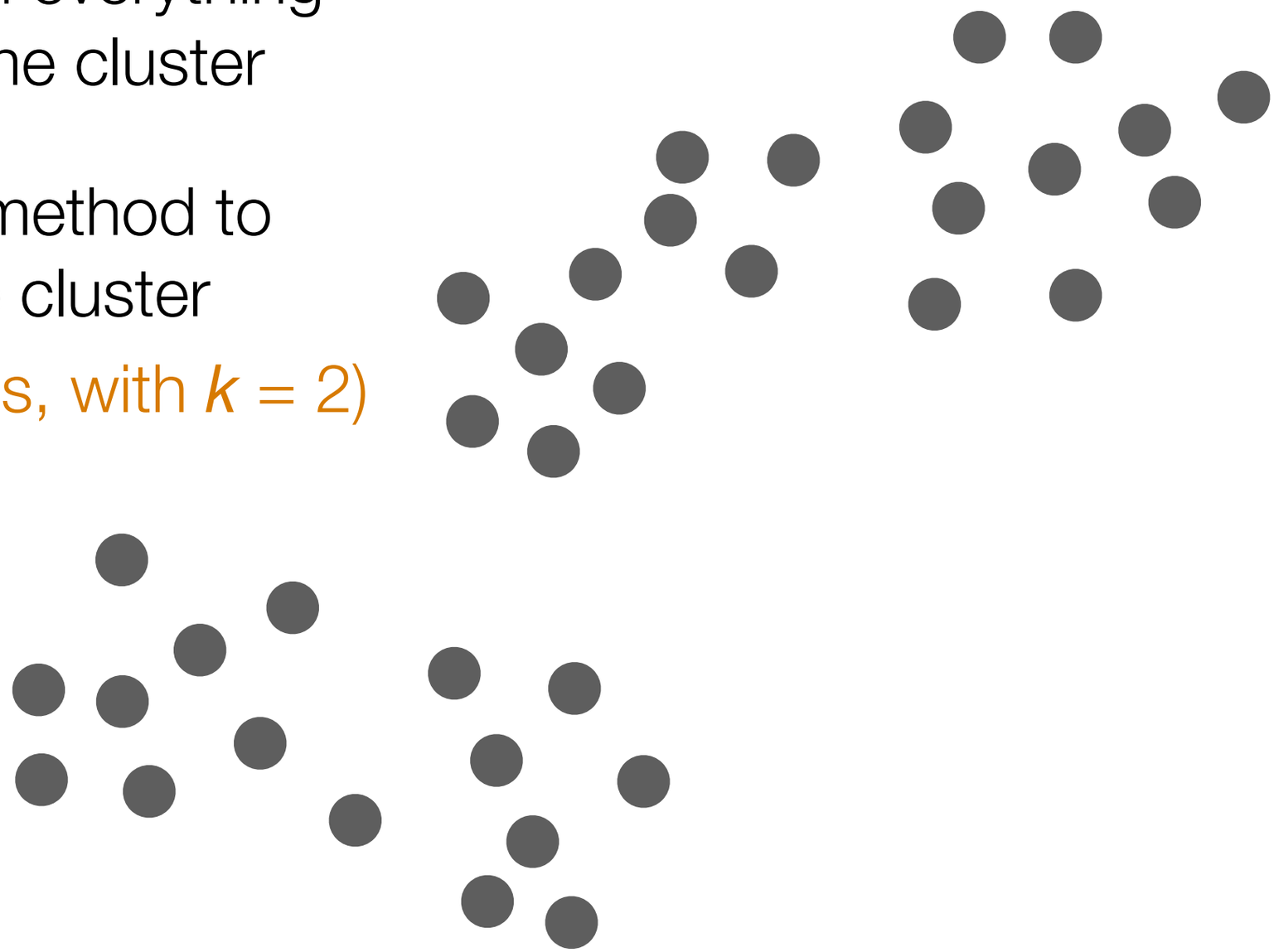
Bottom-up: Start with everything in its own cluster and decide on how to iteratively merge clusters

Divisive Clustering

0. Start with everything in the same cluster

1. Use a method to split the cluster

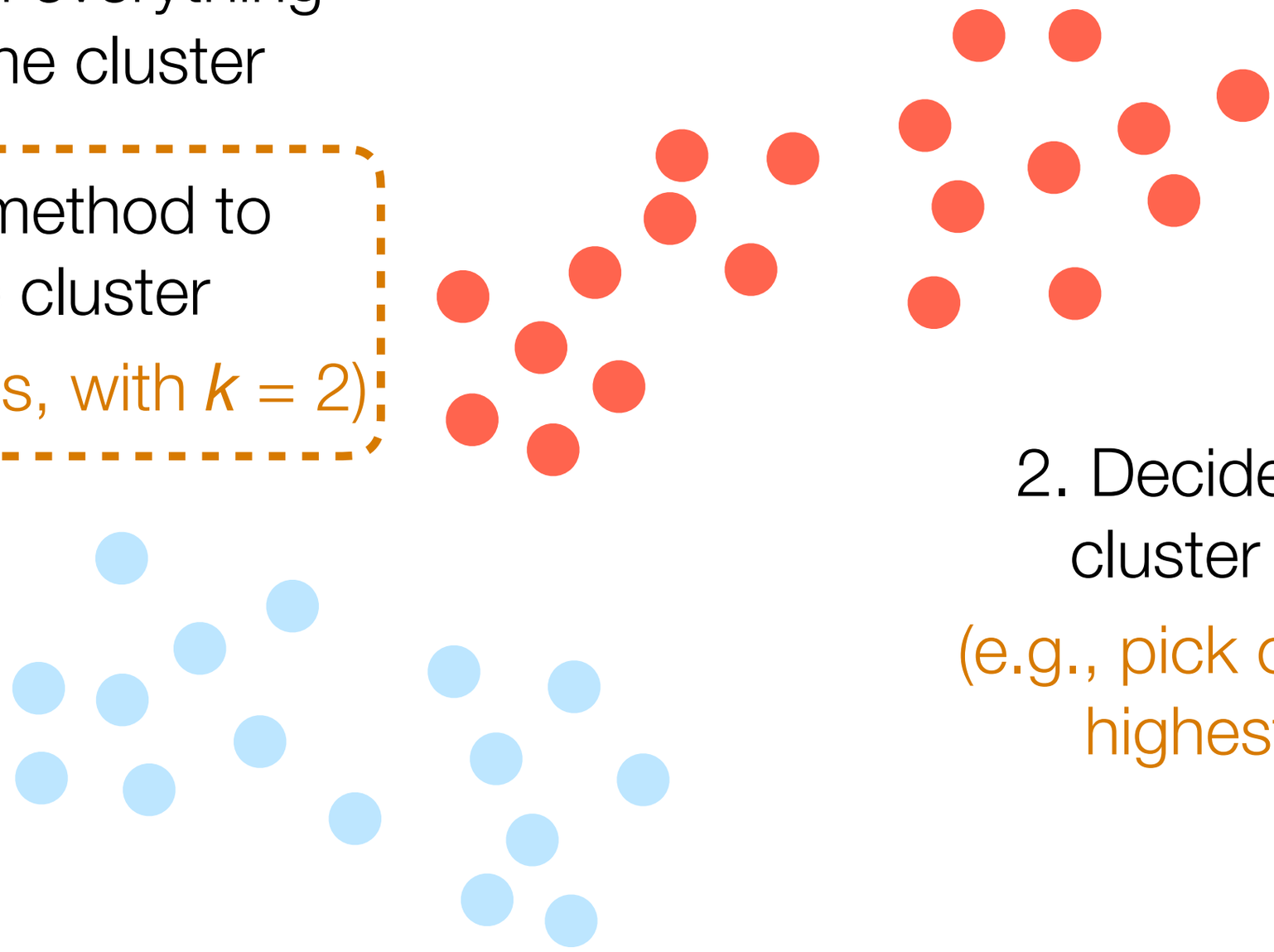
(e.g., *k*-means, with *k* = 2)



Divisive Clustering

0. Start with everything in the same cluster

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

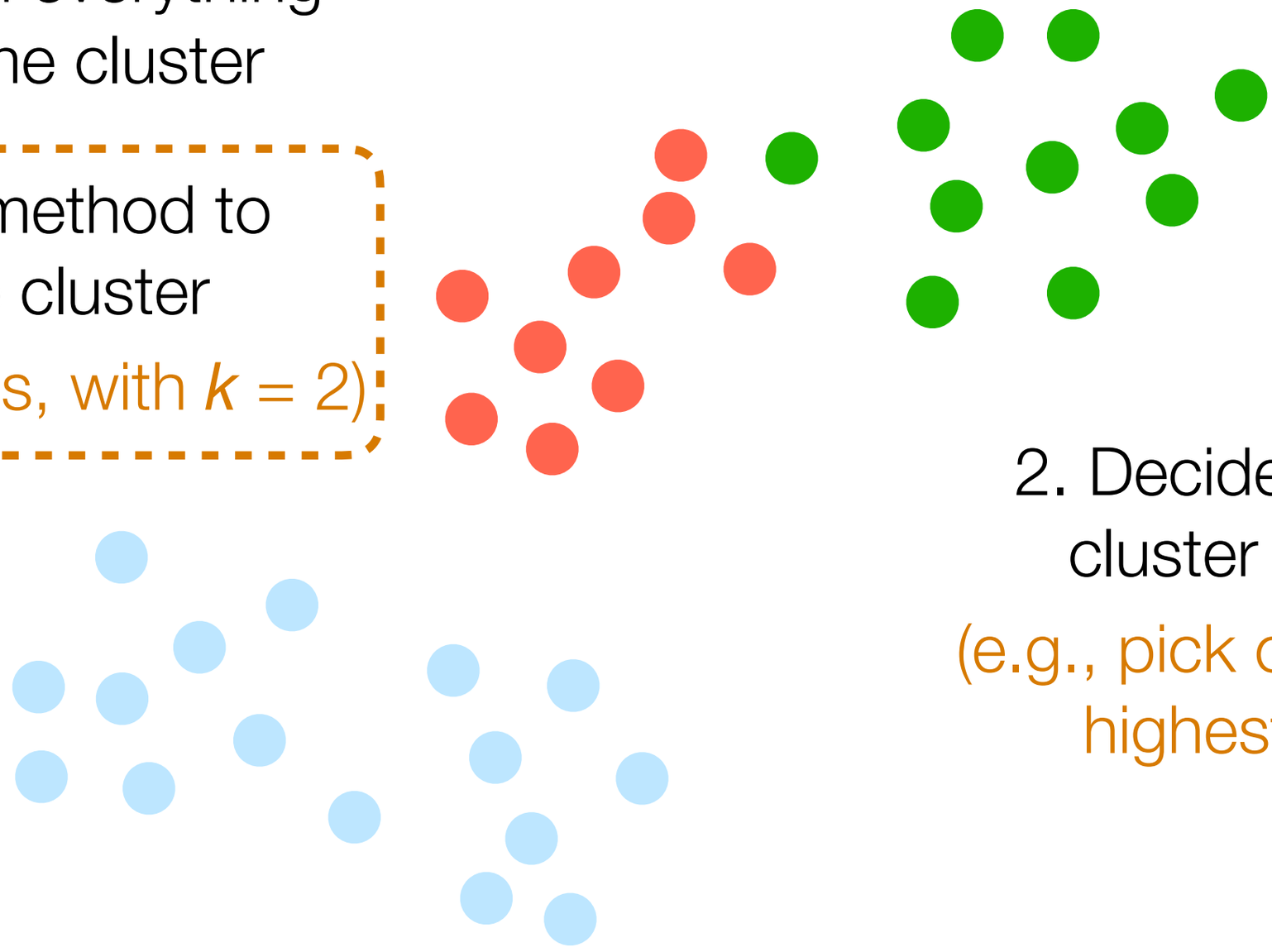


2. Decide on next cluster to split
(e.g., pick cluster with highest RSS)

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)



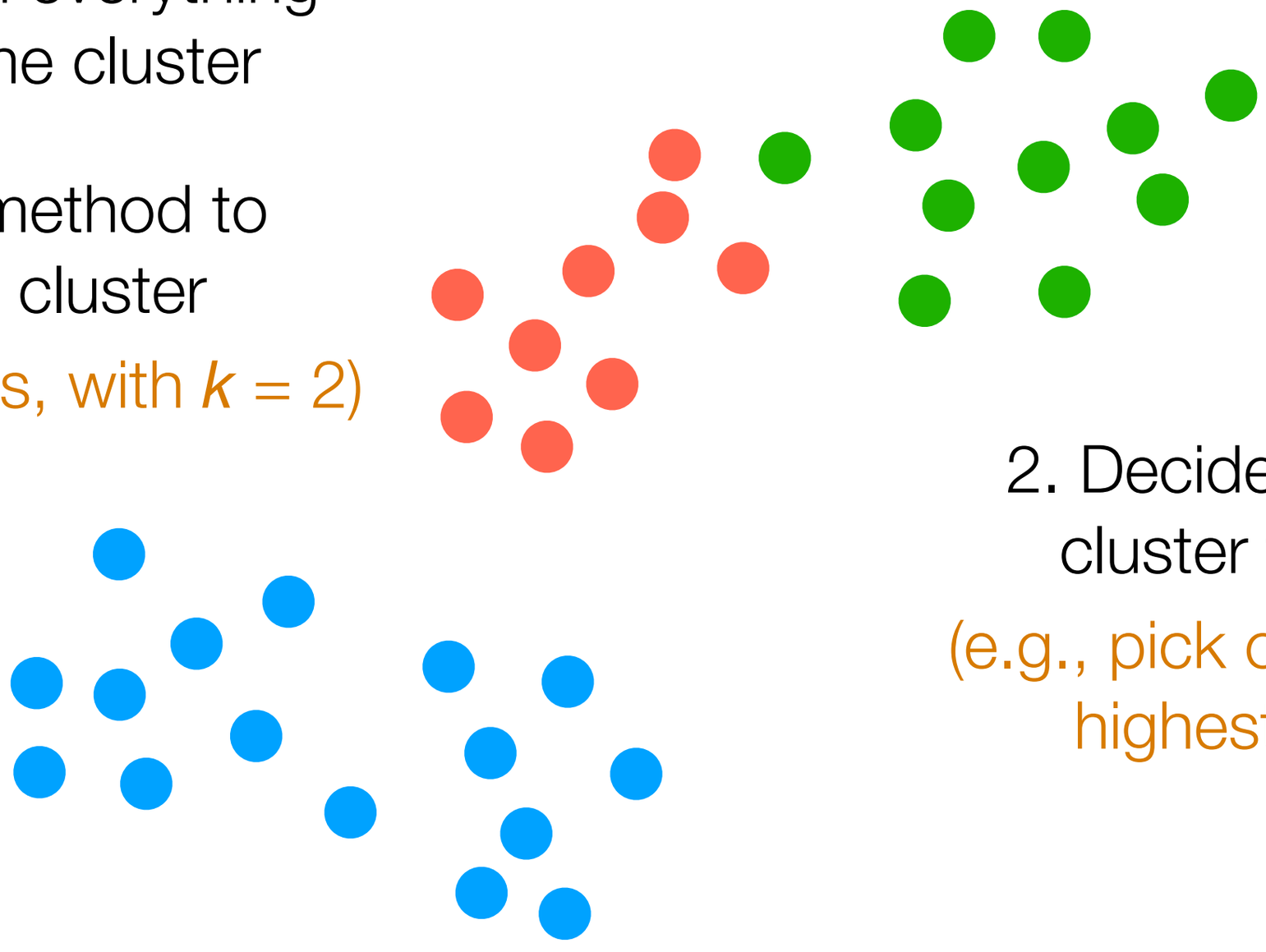
2. Decide on next cluster to split
(e.g., pick cluster with highest RSS)

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

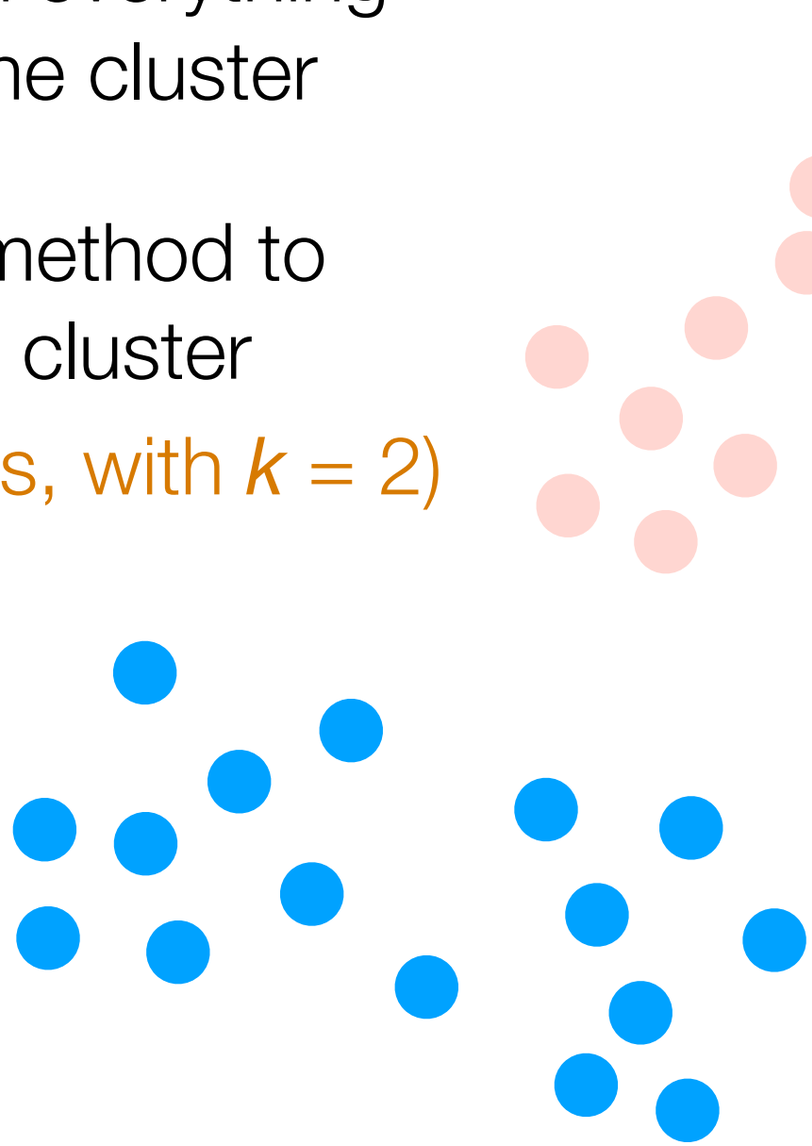
(e.g., pick cluster with highest RSS)

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

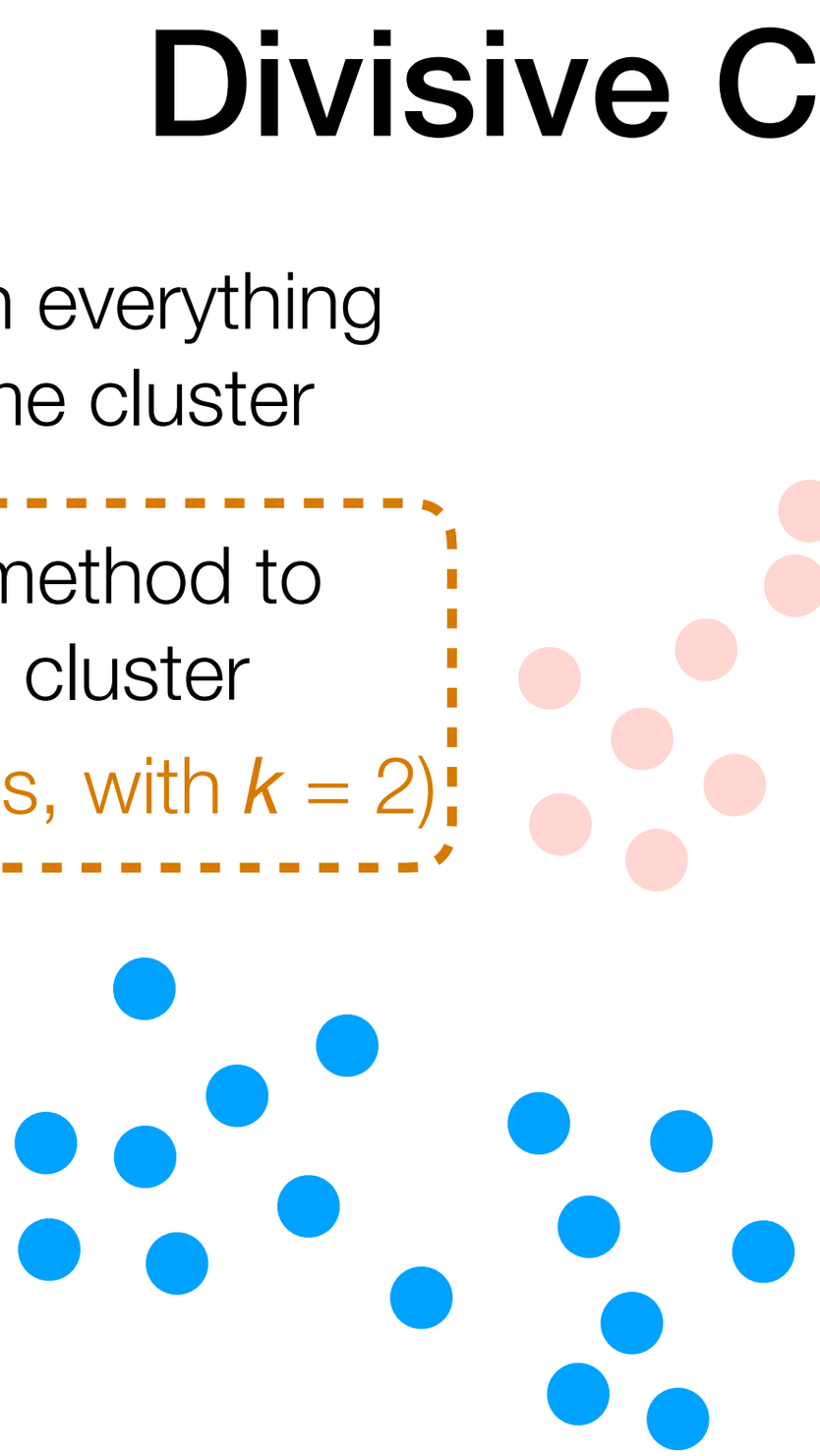
(e.g., pick cluster with highest RSS)

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

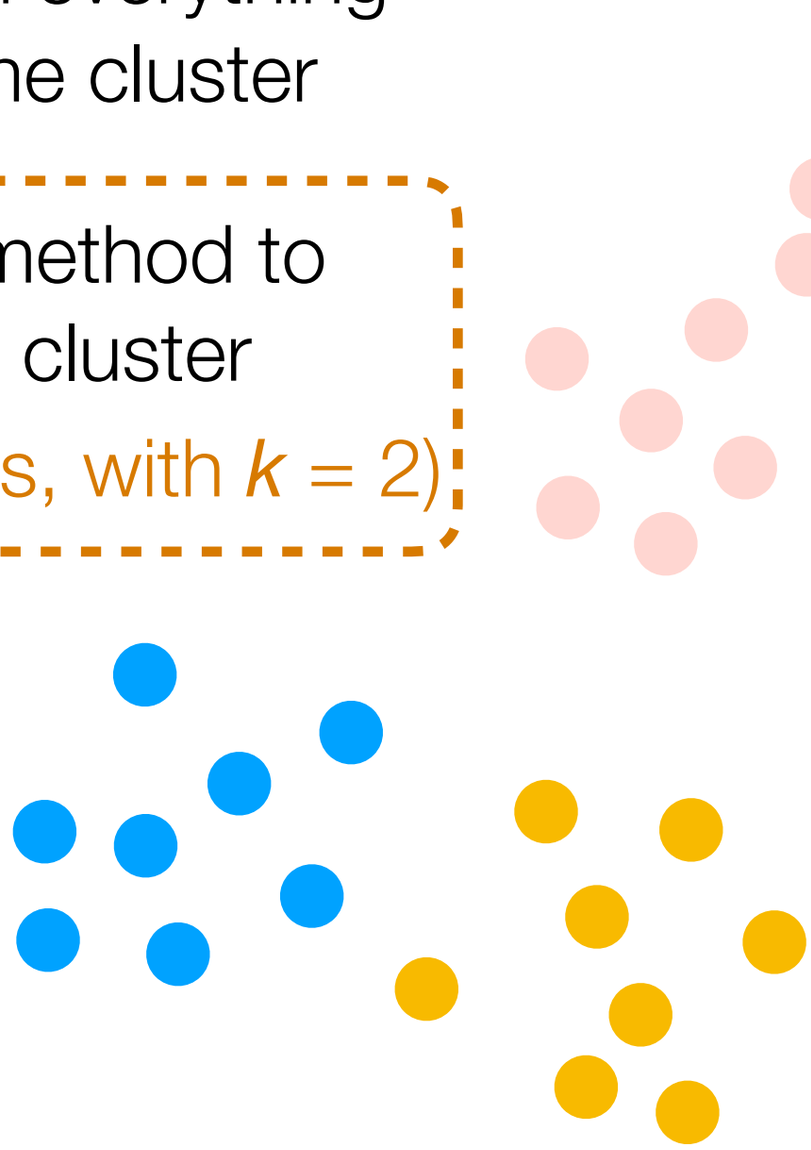
(e.g., pick cluster with highest RSS)

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

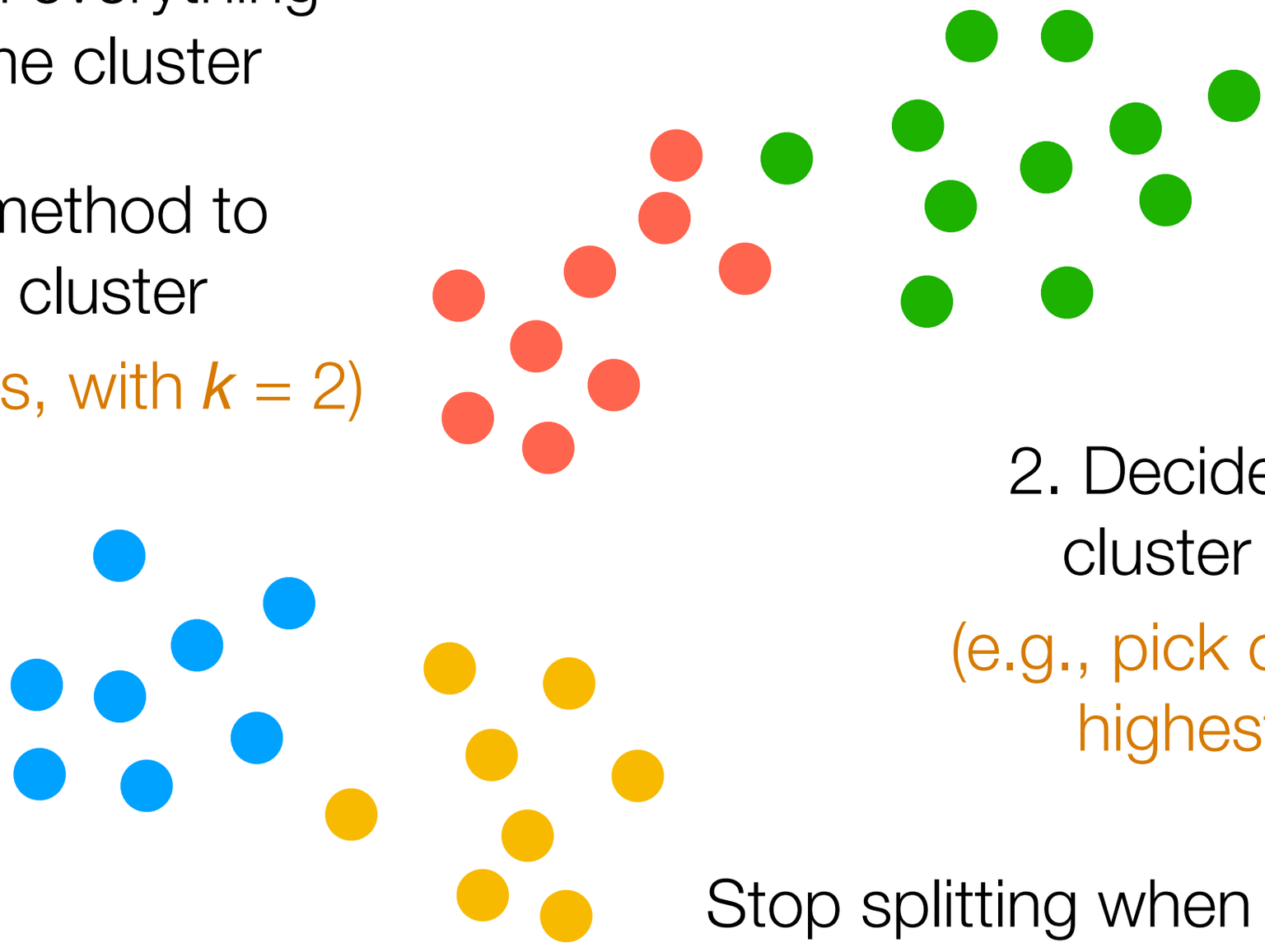
(e.g., pick cluster with highest RSS)

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

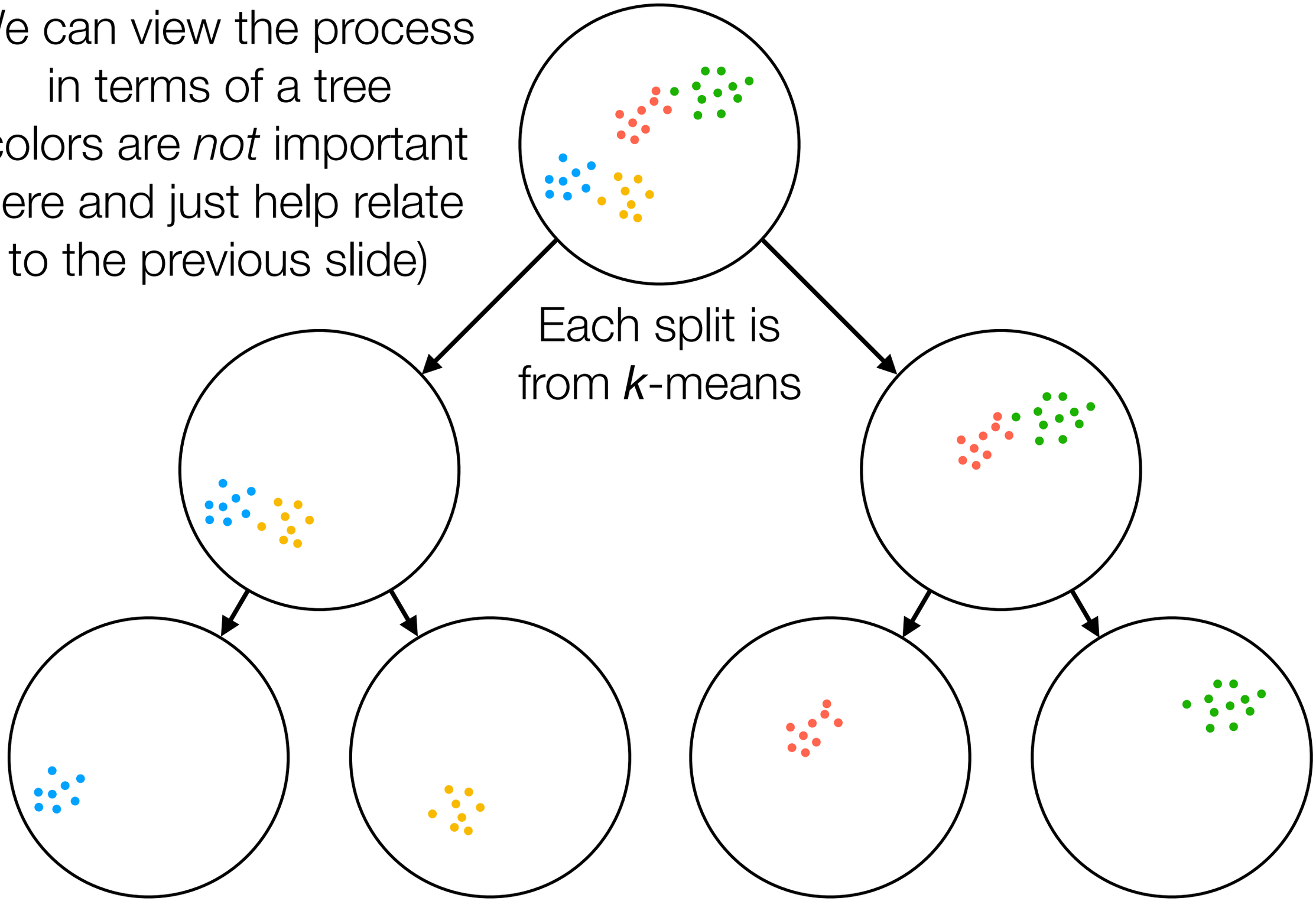
(e.g., pick cluster with highest RSS)

Stop splitting when some termination condition is reached

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

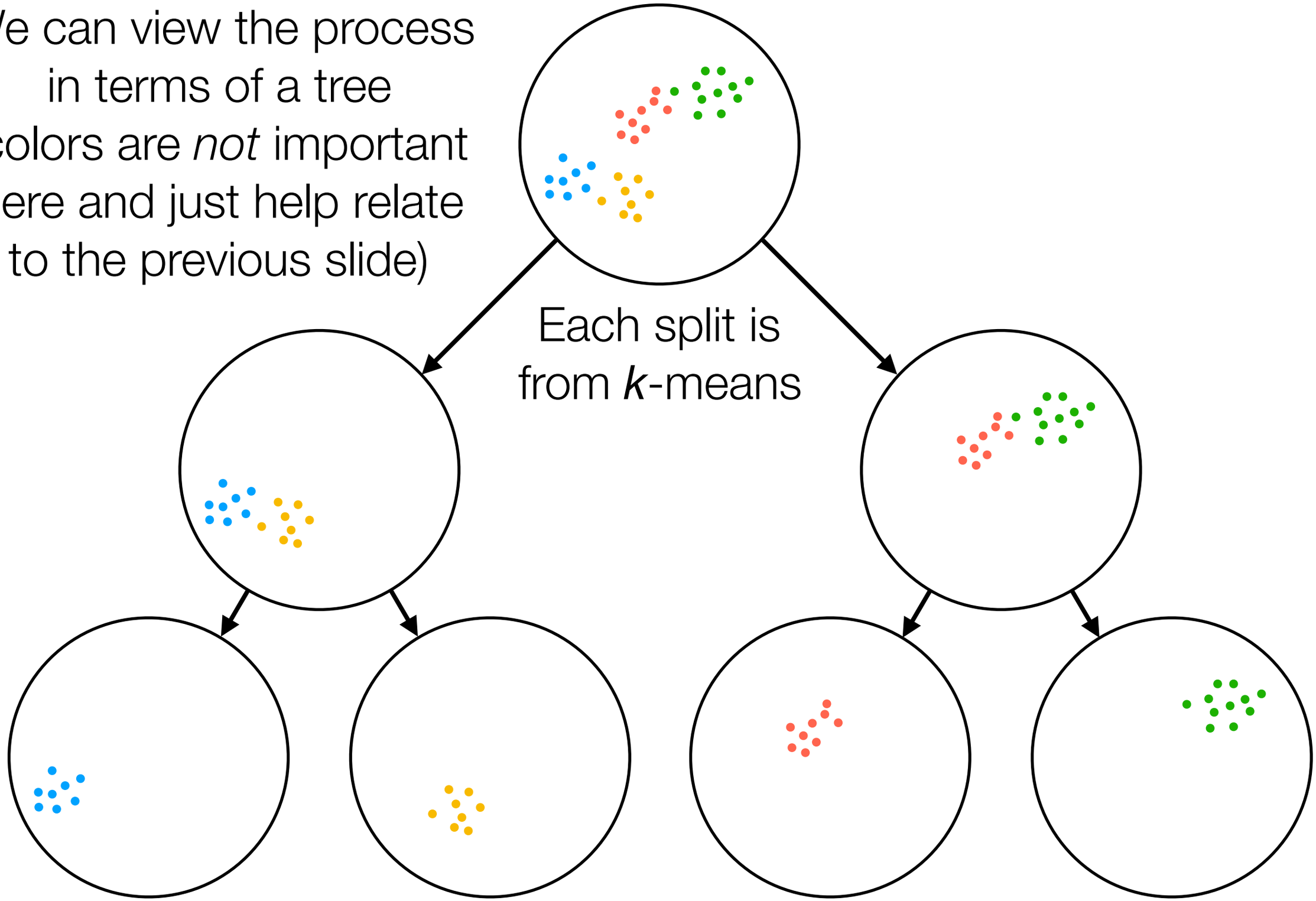
Divisive Clustering

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



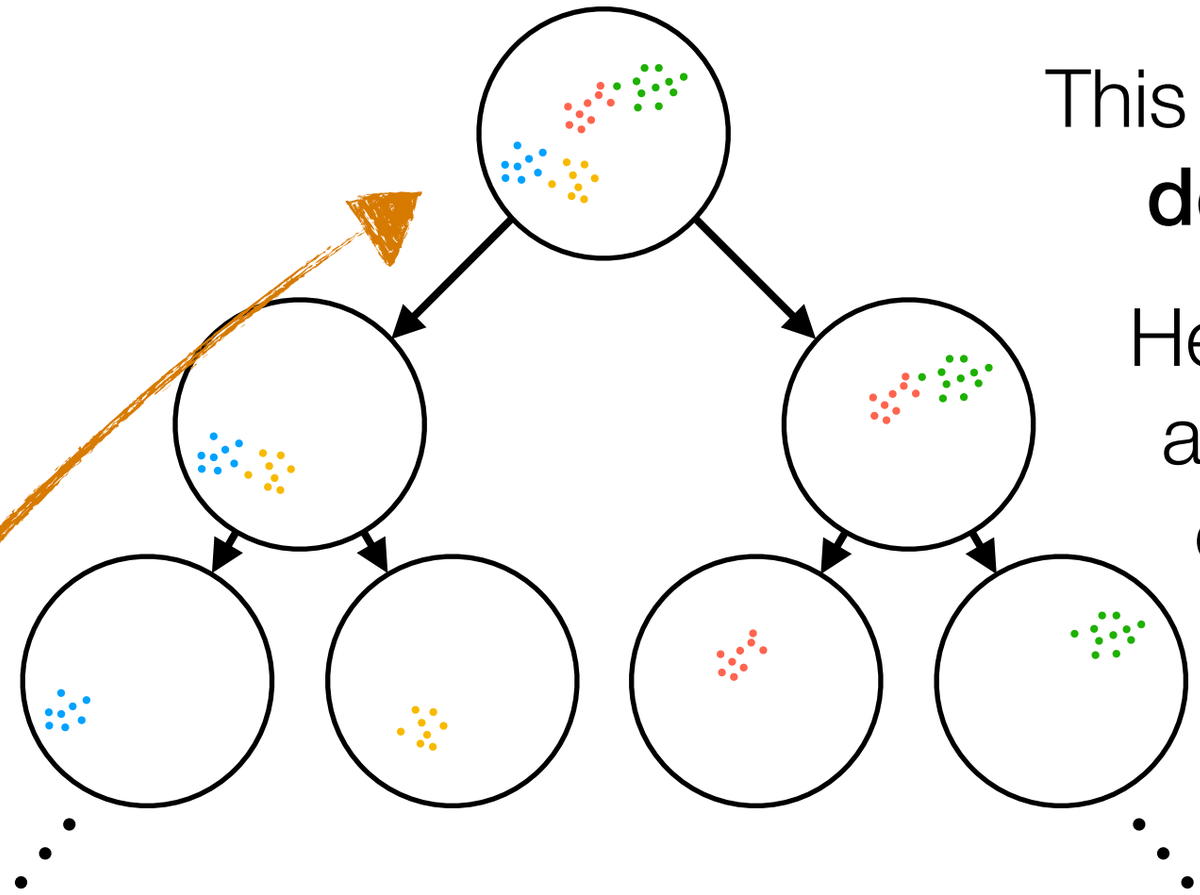
Divisive Clustering

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

Divisive Clustering



This tree is called a **dendrogram**

Helpful for visualizing all the intermediate clustering stages

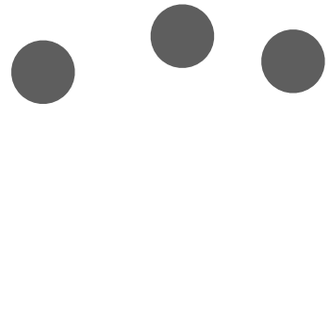
Agglomerative clustering goes the other way

Divisive clustering uses *global* information and keeps splitting

We could keep splitting until the leaves each have 1 point

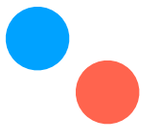
Agglomerative Clustering

0. Every point starts as its own cluster



Agglomerative Clustering

0. Every point starts as its own cluster

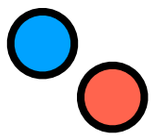


1. Find the “most similar” two clusters

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

Agglomerative Clustering

0. Every point starts as its own cluster



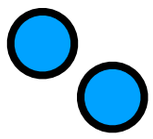
1. Find the “most similar” two clusters

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

2. Merge them

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)

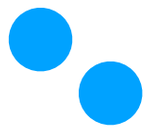
2. Merge them

Agglomerative Clustering

0. Every point starts as its own cluster

1. Find the “most similar” two clusters

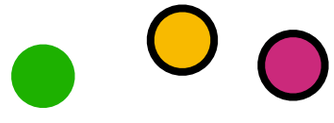
(e.g., pick pair of clusters with closest cluster centers)



2. Merge them

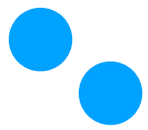
Agglomerative Clustering

0. Every point starts as its own cluster



1. Find the “most similar” two clusters

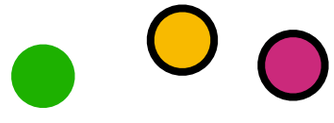
(e.g., pick pair of clusters with closest cluster centers)



2. Merge them

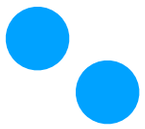
Agglomerative Clustering

0. Every point starts as its own cluster



1. Find the “most similar” two clusters

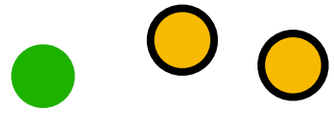
(e.g., pick pair of clusters with closest cluster centers)



2. Merge them

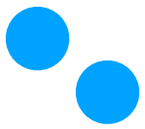
Agglomerative Clustering

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)



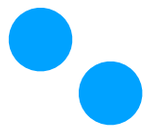
2. Merge them

Agglomerative Clustering

0. Every point starts as its own cluster

1. Find the “most similar” two clusters

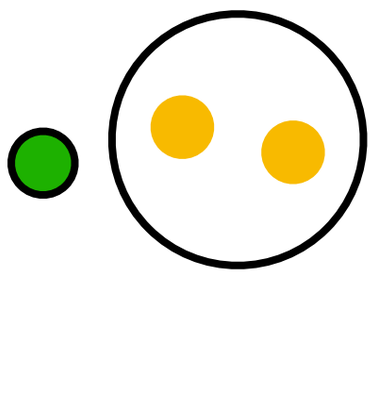
(e.g., pick pair of clusters with closest cluster centers)



2. Merge them

Agglomerative Clustering

0. Every point starts as its own cluster



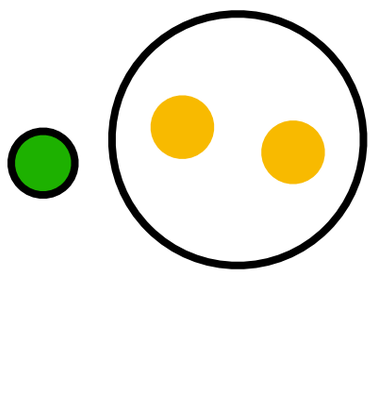
1. Find the “most similar” two clusters

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

2. Merge them

Agglomerative Clustering

0. Every point starts as its own cluster



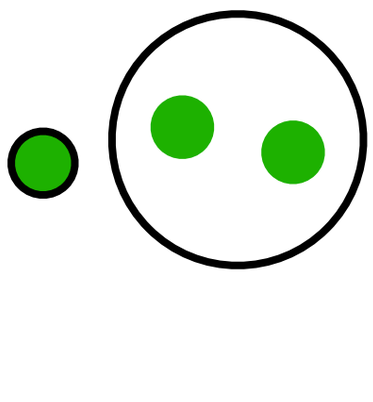
1. Find the “most similar” two clusters

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

2. Merge them

Agglomerative Clustering

0. Every point starts as its own cluster



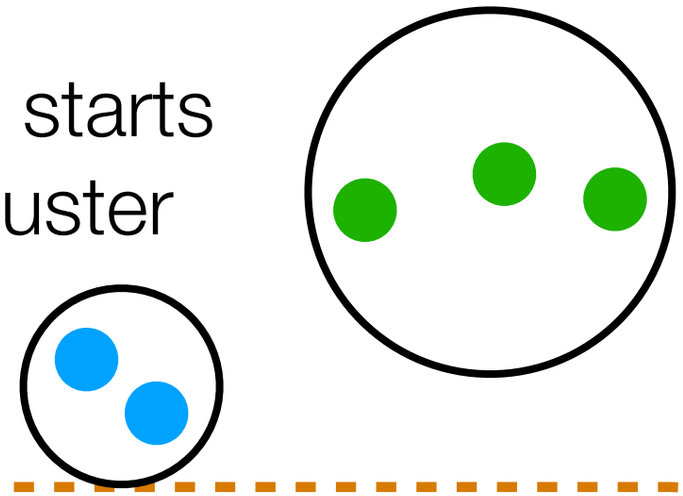
1. Find the “most similar” two clusters

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

2. Merge them

Agglomerative Clustering

0. Every point starts as its own cluster



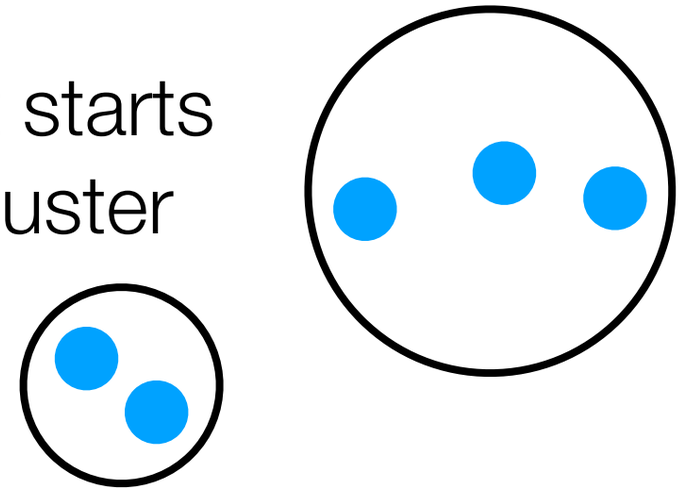
1. Find the “most similar” two clusters

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

2. Merge them

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)

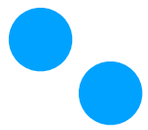
2. Merge them

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)



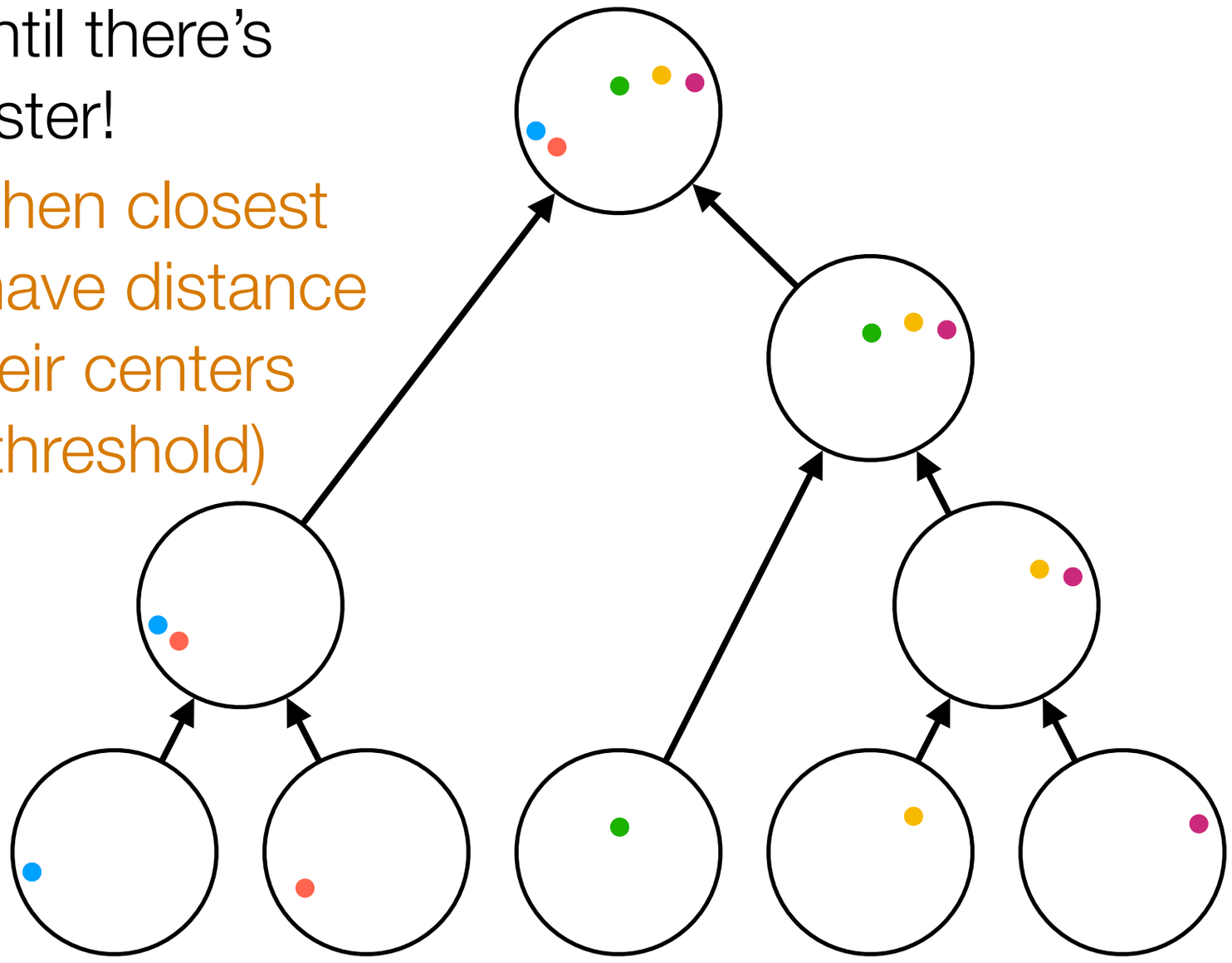
2. Merge them

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



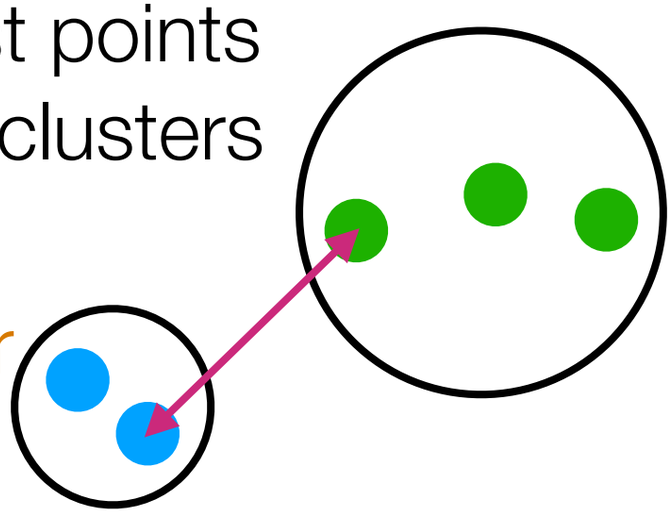
Agglomerative clustering uses *local* information and keeps merging

Agglomerative Clustering

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

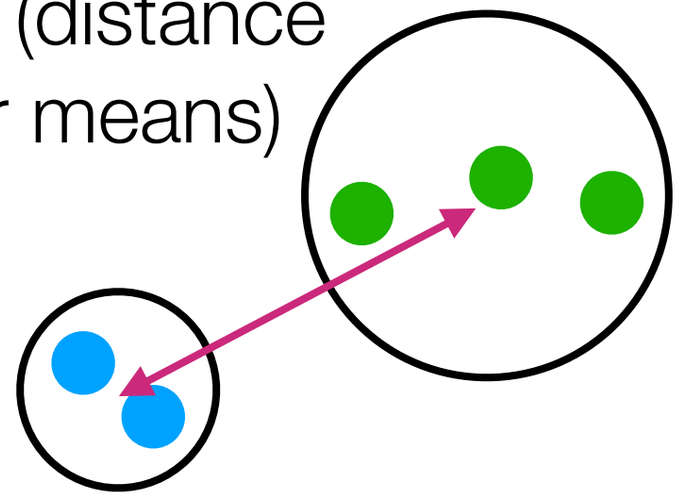
Single linkage: use distance between closest points across the two clusters

Can end up chaining together too many things



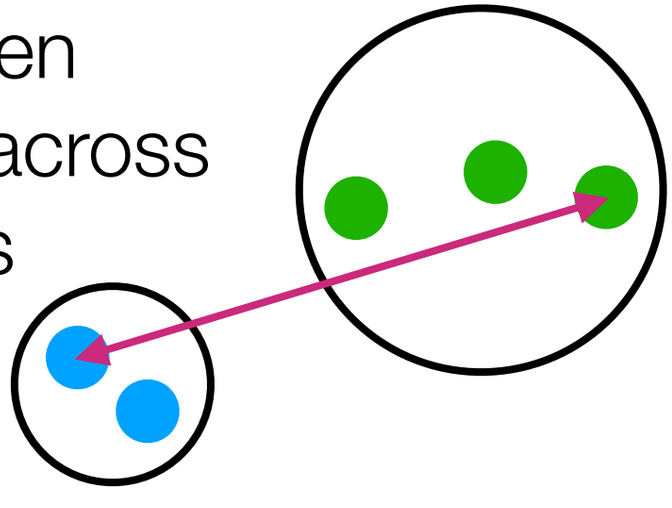
Centroid linkage: what we saw already (distance between cluster means)

Ignores # items in each cluster

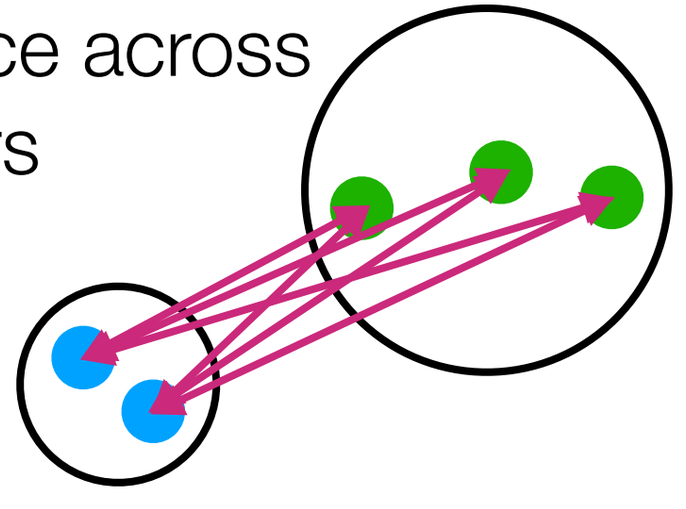


Complete linkage: use distance between farthest points across the two clusters

Get “crowding” behavior



Average linkage: use average distance across all possible pairs

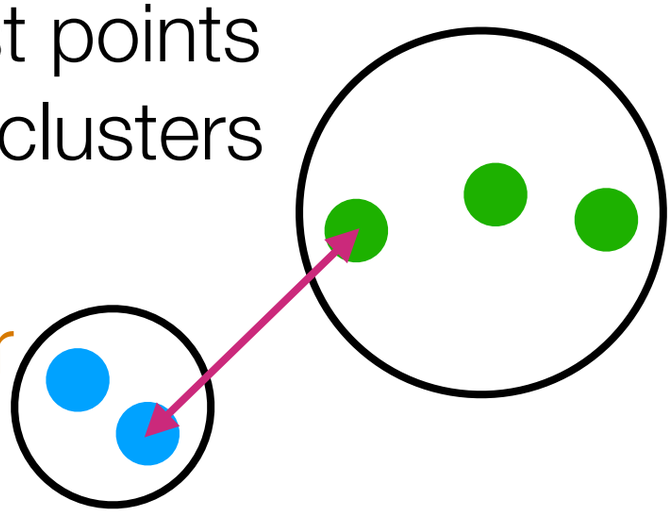


Agglomerative Clustering

Some ways to define what it means
(needed to find most similar clusters

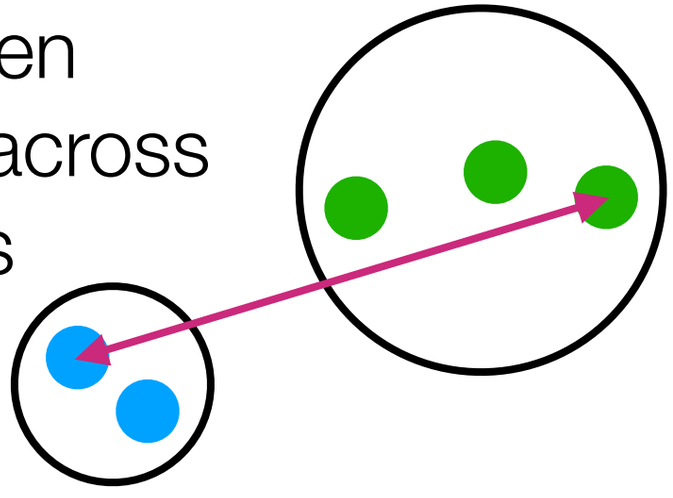
Single linkage: use distance between closest points across the two clusters

Can end up chaining together too many things



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

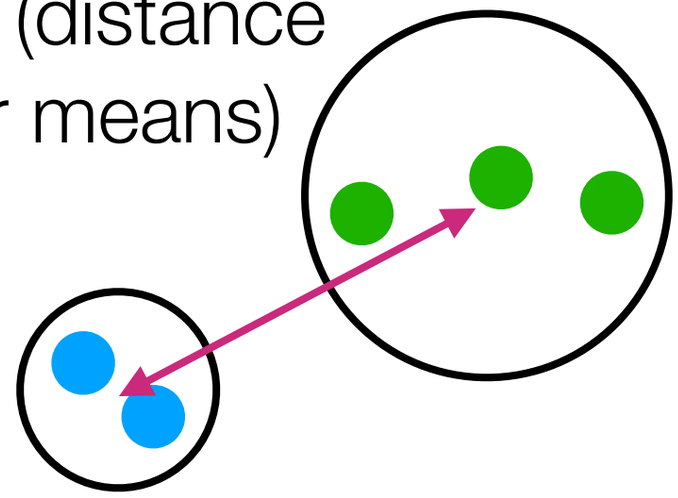
Get "crowding" behavior



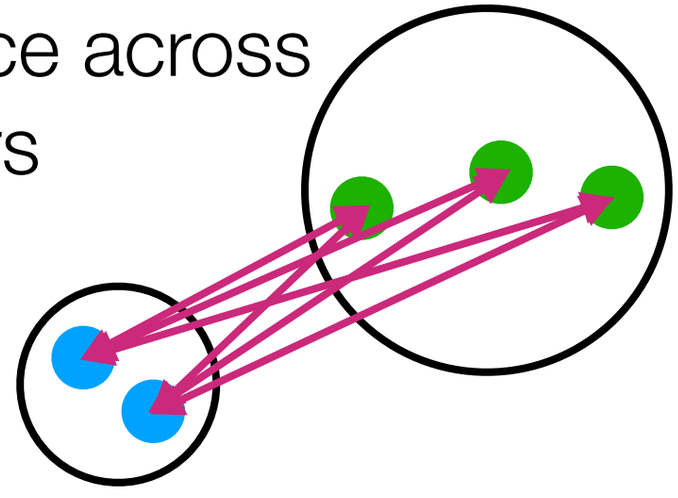
Clustering can change with monotonic transform of distance

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

Ignores # items in each cluster



Average linkage: use average distance across all possible pairs

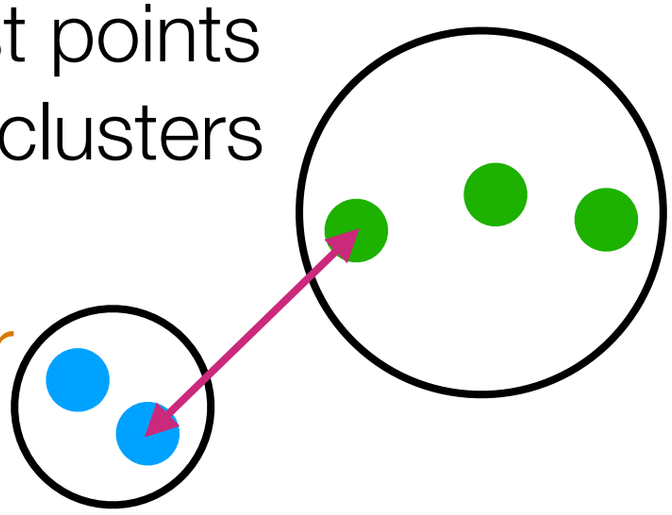


Agglomerative Clustering

Clustering stays the same with monotonic transform of distance

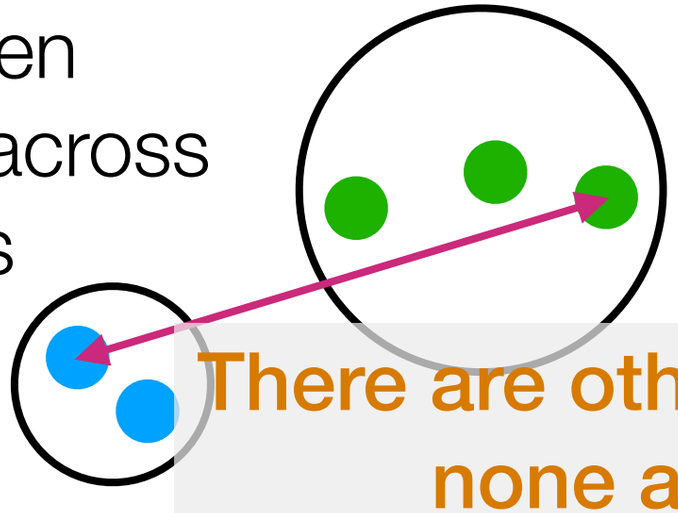
Single linkage: use distance between closest points across the two clusters

Can end up chaining together too many things



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

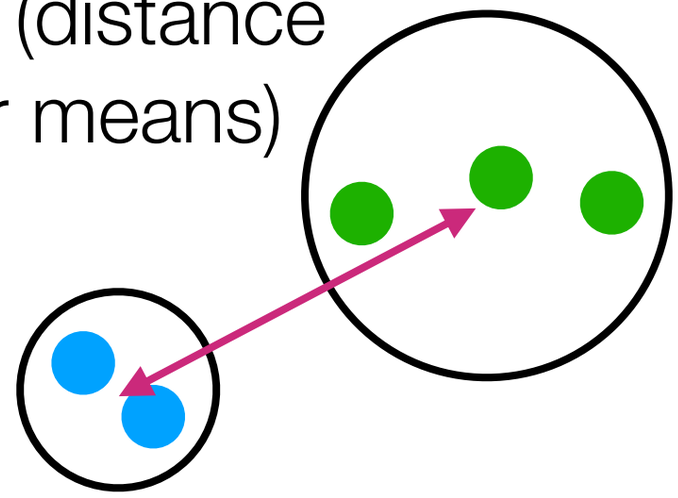
Get "crowding" behavior



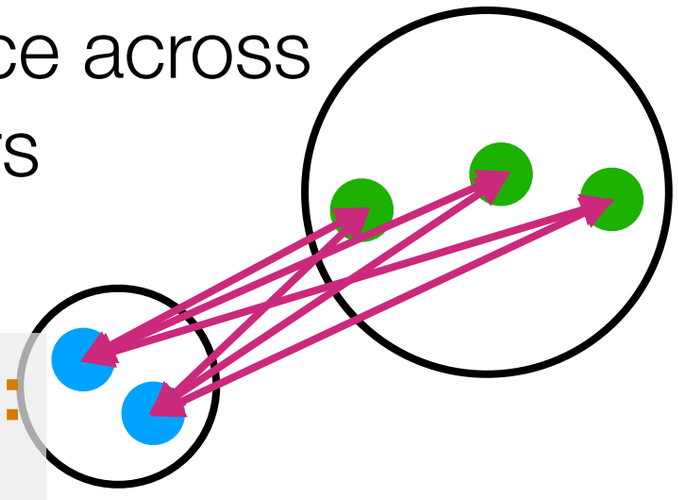
Clustering can change with monotonic transform of distance

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

Ignores # items in each cluster



Average linkage: use average distance across all possible pairs



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

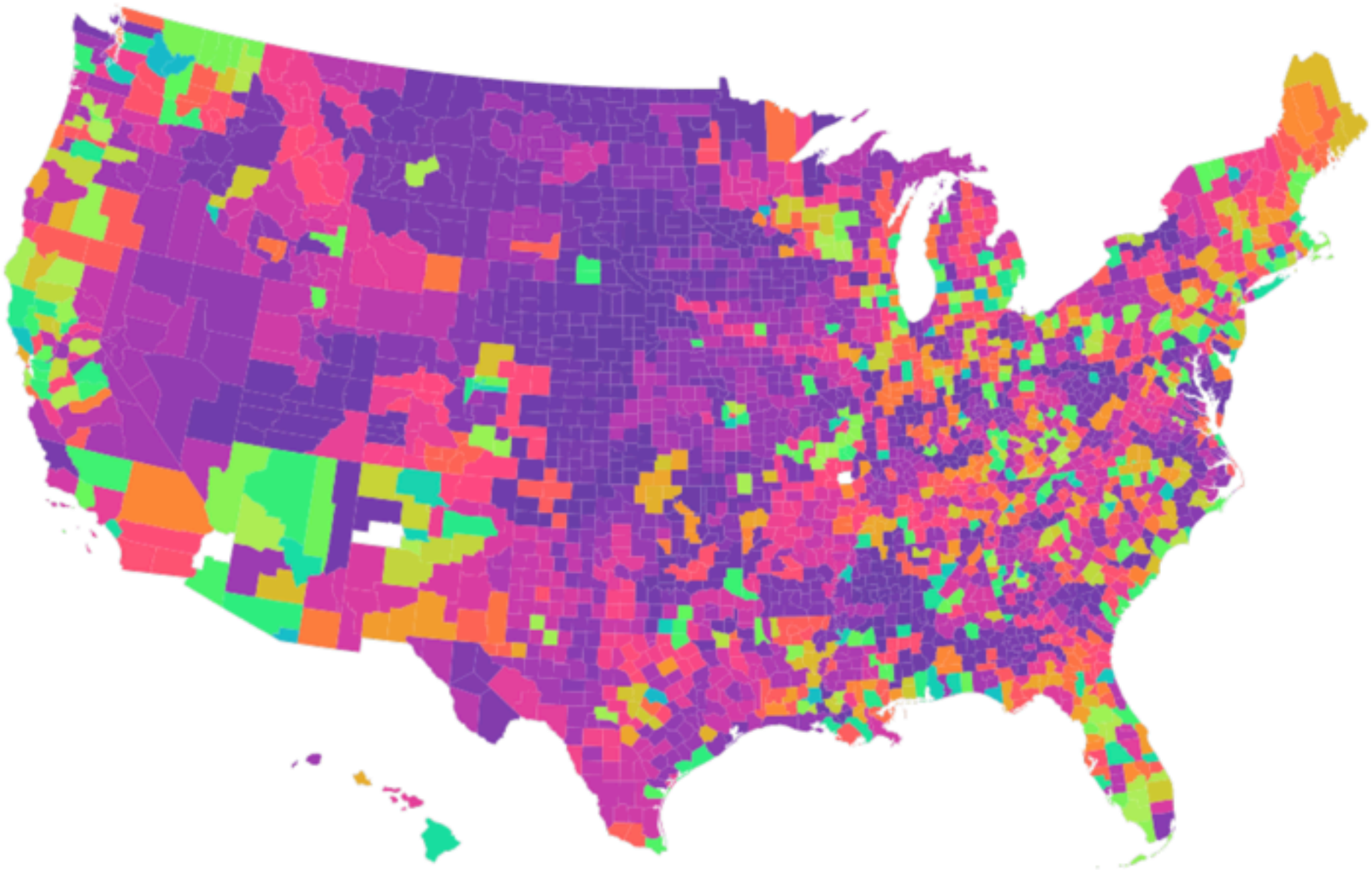
Easily works with different distances (not just Euclidean)

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

Clustering

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

***Many* more methods we didn't cover**

- `sklearn` has a whole bunch more (*not* close to exhaustive)
- Also: remember the recommendation system setup?
  - **Co-clustering** is the problem of clustering both users and items at the same time (`sklearn` has a few methods)