Automatically Choosing the Number of Clusters

DP-GMMs, DP-means, CH index
(see also: gap statistic)

slides by
George Chen
Carnegie Mellon University
Fall 2017

GMM with *k* Clusters

<u>Cluster 1</u>

Probability of generating a point from cluster 1 = π_1

Gaussian mean = μ_1

Gaussian covariance = Σ_1

…

<u>Cluster *k*</u>

Probability of generating a point from cluster k = π_k

Gaussian mean = μ_k

Gaussian covariance = Σ_k

How to generate points from this GMM:

1. Flip biased k-sided coin (the sides have probabilities $\pi_1, \ldots, \pi_k$)
2. Let Z be the side that we got (it is some value $1, \ldots, k$)
3. Sample 1 point from Gaussian mean μ_Z, covariance Σ_Z

Learning a GMM

Demo

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

DP-GMM High-Level Idea

<u>Cluster 1</u>

Probability of generating a point from cluster 1 = π_1

Gaussian mean = μ_1

Gaussian covariance = Σ_1

DP-GMM High-Level Idea

	Cluster 1	Cluster 2
Probability of generating a point from cluster 1	$= \pi_1$	π_2
Gaussian mean	$= \mu_1$	μ_2
Gaussian covariance	$= \Sigma_1$	Σ_2

DP-GMM High-Level Idea

	Cluster 1	Cluster 2	Cluster 3
Probability of generating a point from cluster 1 = π_1		π_2	π_3
Gaussian mean = μ_1		μ_2	μ_3
Gaussian covariance = Σ_1		Σ_2	Σ_3

DP-GMM High-Level Idea

Cluster 1	Cluster 2	Cluster 3	
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	

DP-GMM High-Level Idea

	<u>Cluster 1</u>	<u>Cluster 2</u>	<u>Cluster 3</u>	
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

DP-GMM High-Level Idea

	Cluster 1	Cluster 2	Cluster 3	
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 is a parameter that controls how these π values roughly decay

There are an infinite number of parameters

DP-GMM High-Level Idea

	Cluster 1	Cluster 2	Cluster 3	
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 is a parameter that controls how these π values roughly decay

There are an infinite number of parameters

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

DP-GMM High-Level Idea

	Cluster 1	Cluster 2	Cluster 3	
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 is a parameter that controls how these π values roughly decay

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, …)

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)

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

DP-GMM High-Level Idea

	Cluster 1	Cluster 2	Cluster 3	
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 is a parameter that controls how these π values roughly decay

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

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

Learning a DP-GMM

Two main approaches:

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)

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)
  - Algorithm is somewhat similar to *k*-means/EM for GMMs

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

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)
    - 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!)

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)
  - 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)

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)

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.

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

What approximates learning DP-GMMs?

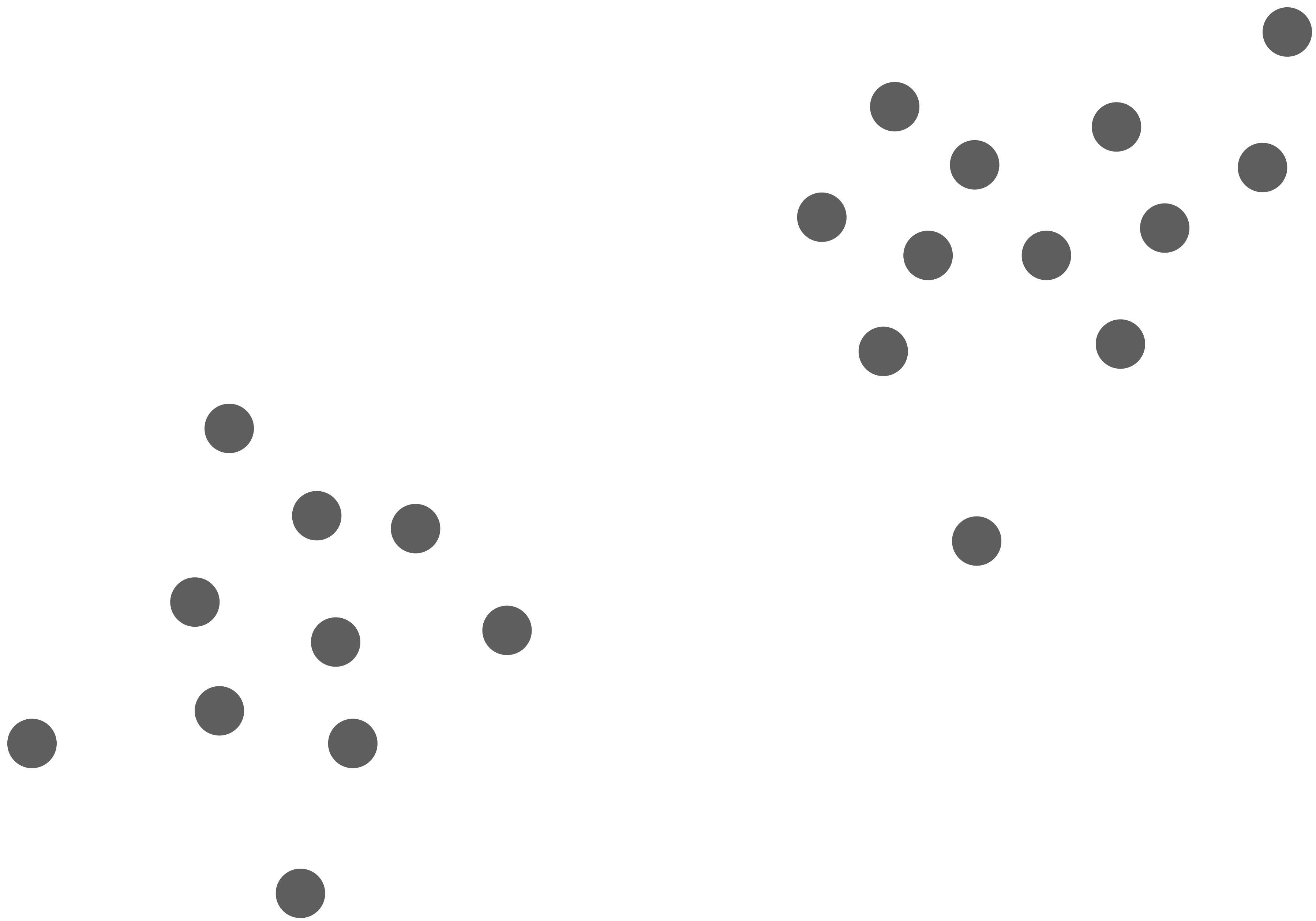
k-means approximates (a special case of) learning GMM's.

What approximates learning DP-GMMs?

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

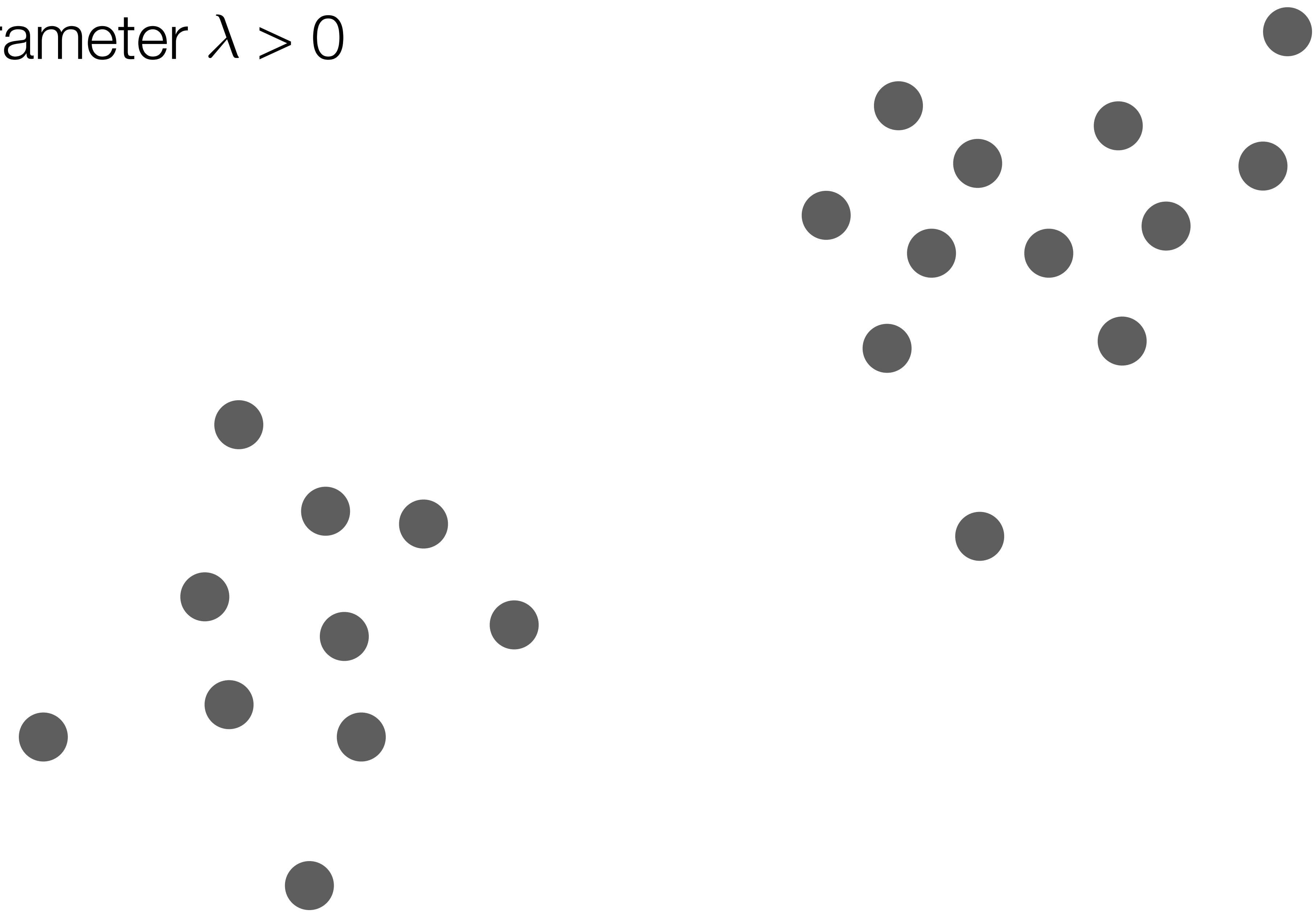
DP-means

DP-means



DP-means

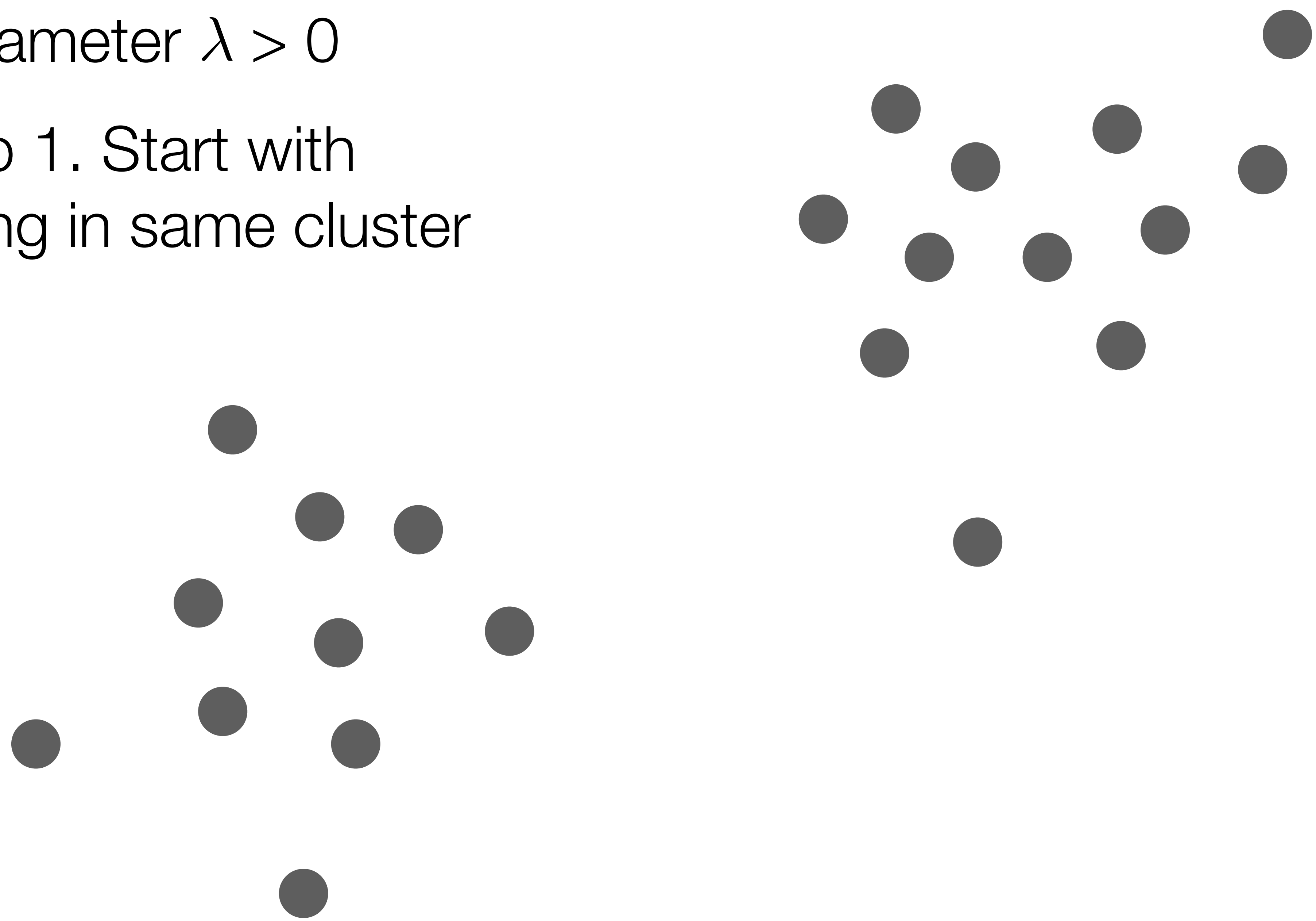
Step 0. Pick concentration parameter $\lambda > 0$



DP-means

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

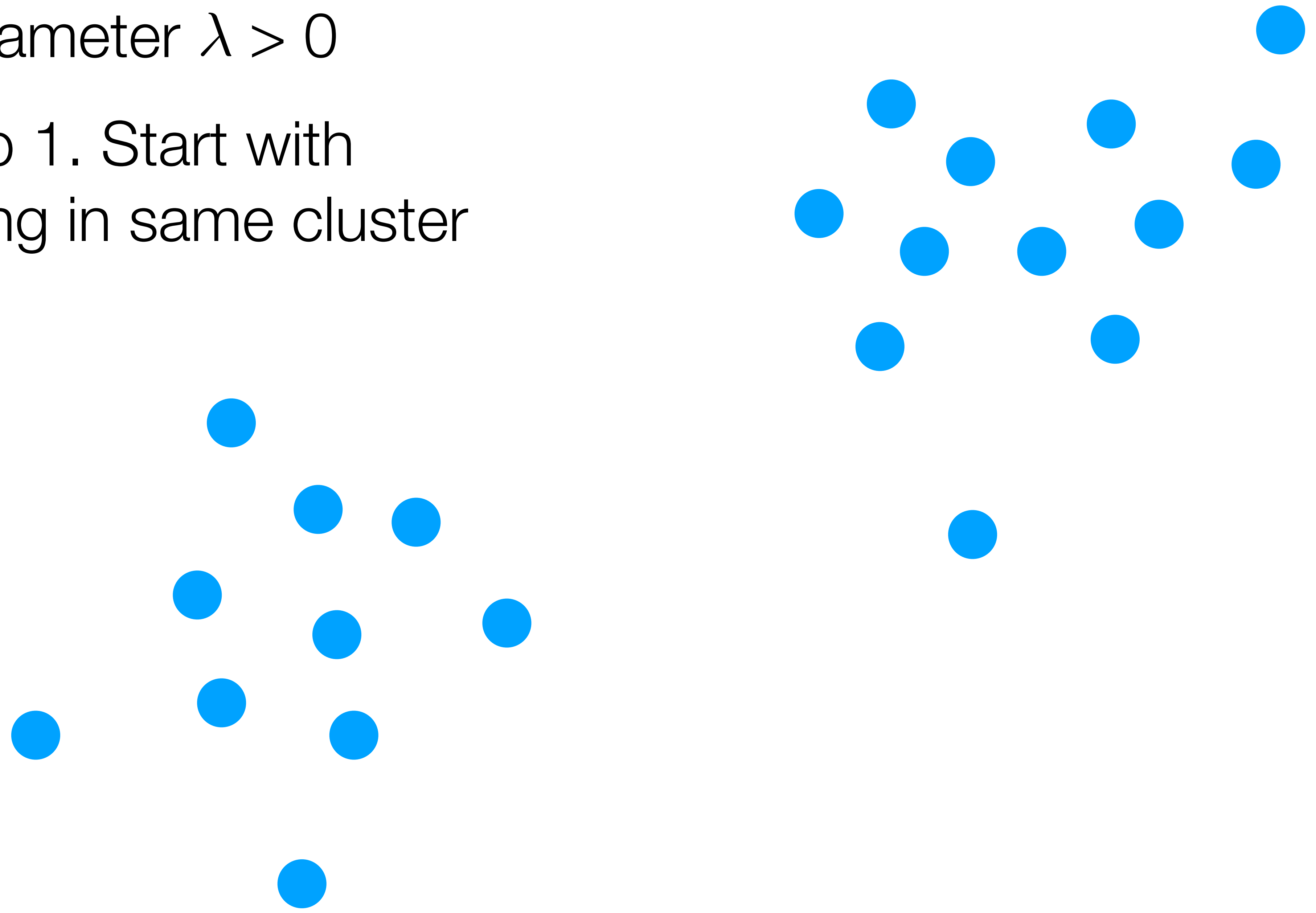
Step 1. Start with everything in same cluster



DP-means

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

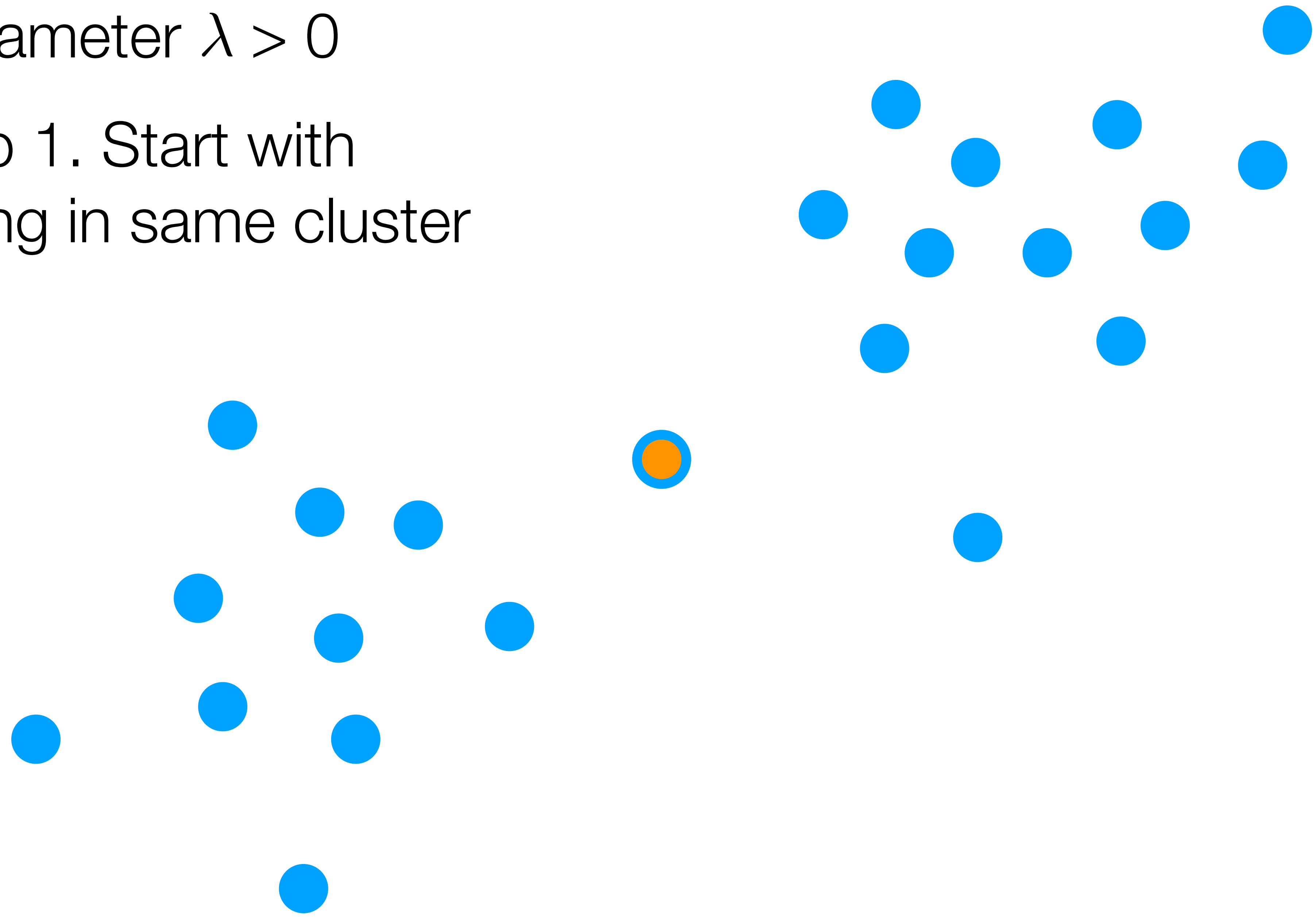
Step 1. Start with everything in same cluster



DP-means

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

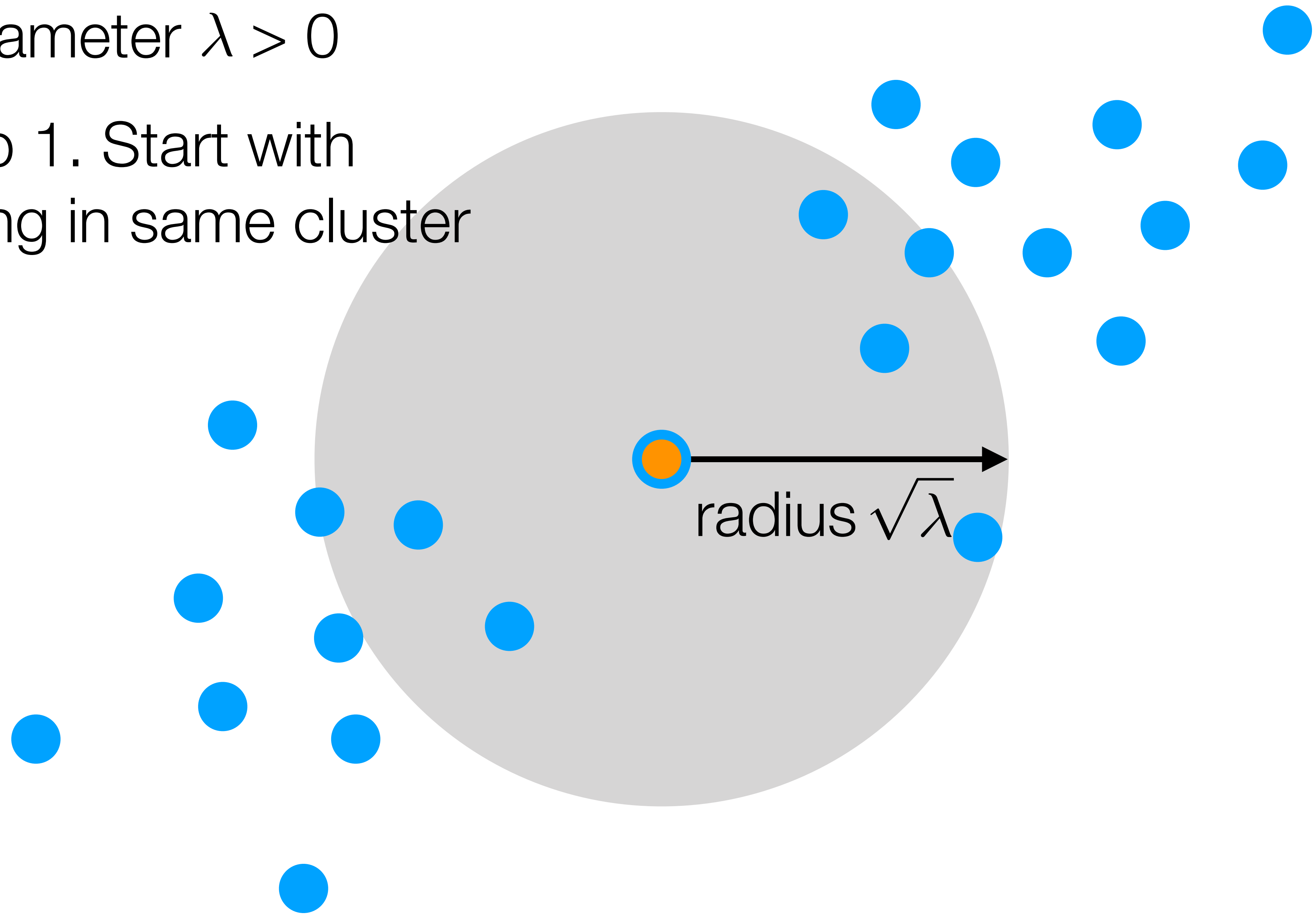
Step 1. Start with everything in same cluster



DP-means

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

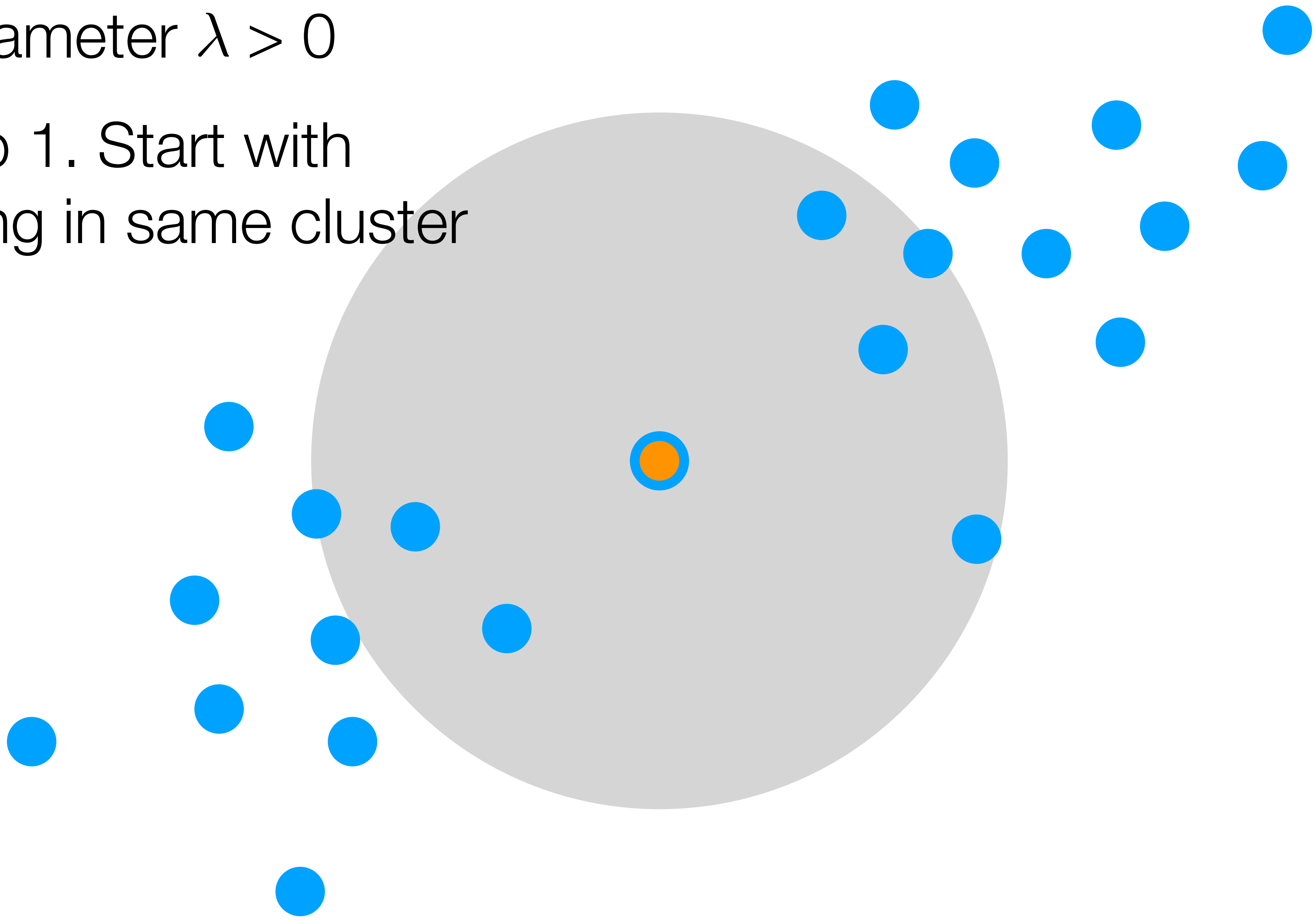
Step 1. Start with everything in same cluster



DP-means

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

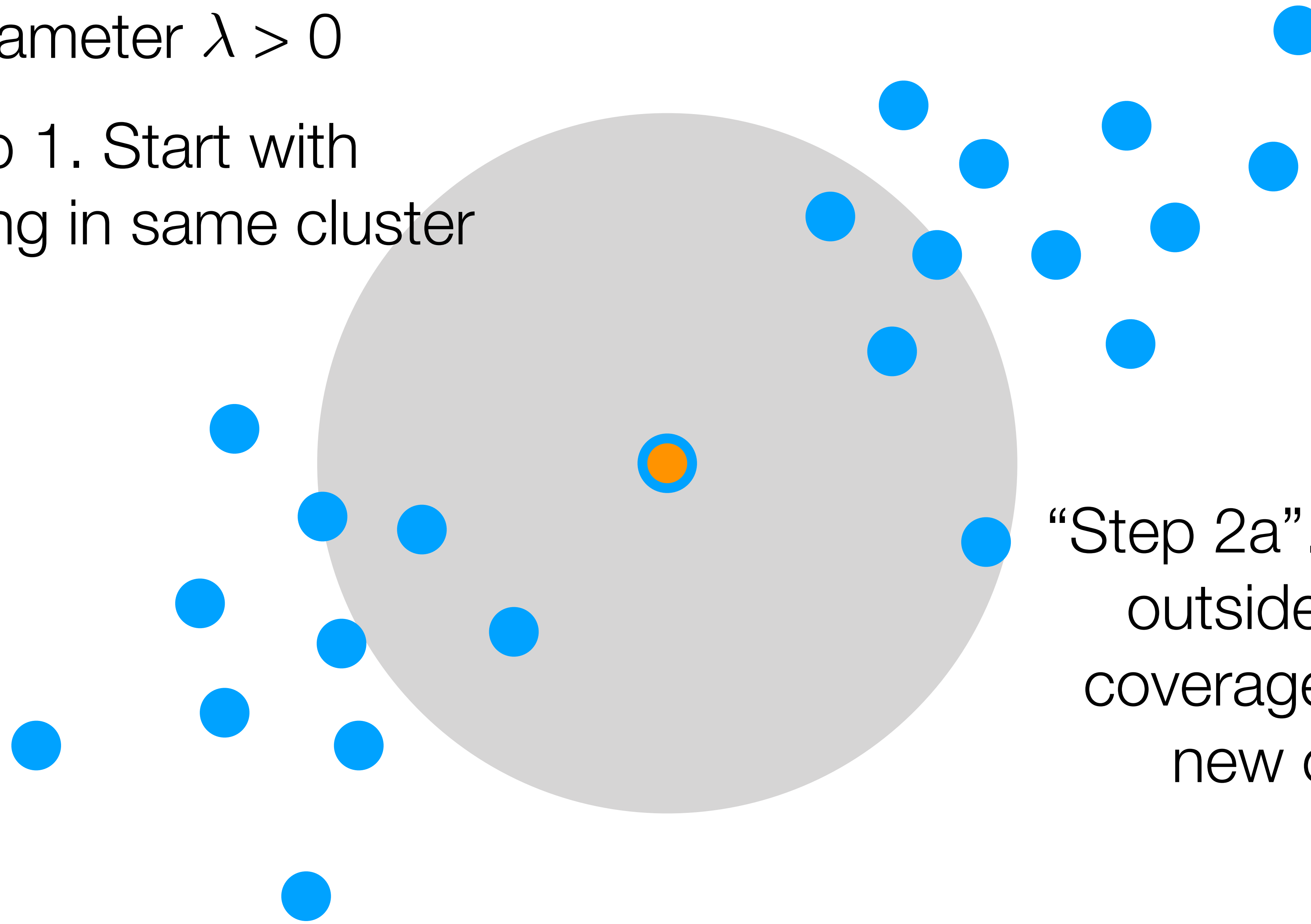
Step 1. Start with everything in same cluster



DP-means

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

Step 1. Start with everything in same cluster

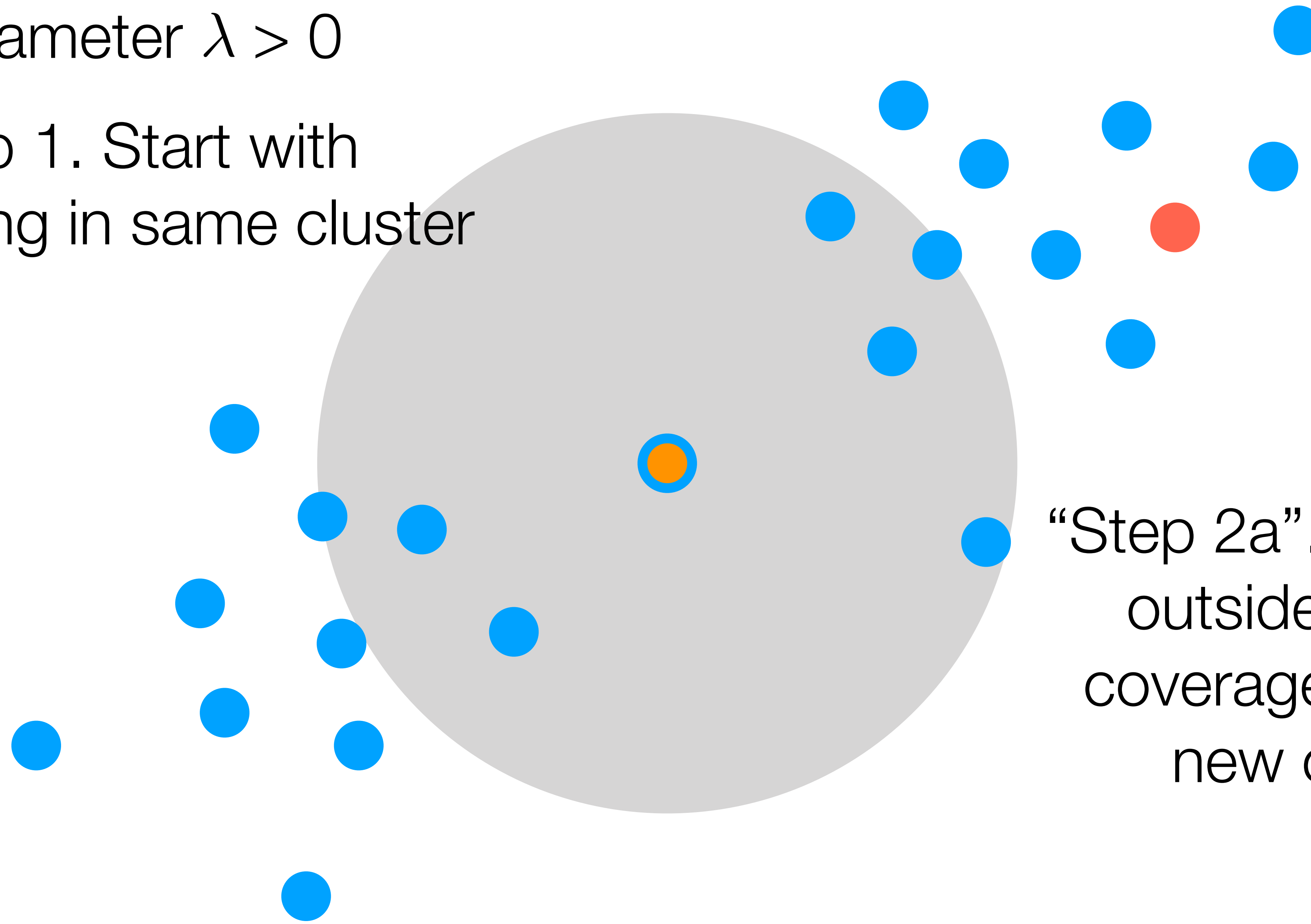


"Step 2a". Pick point outside of gray coverage to make new cluster

DP-means

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

Step 1. Start with everything in same cluster

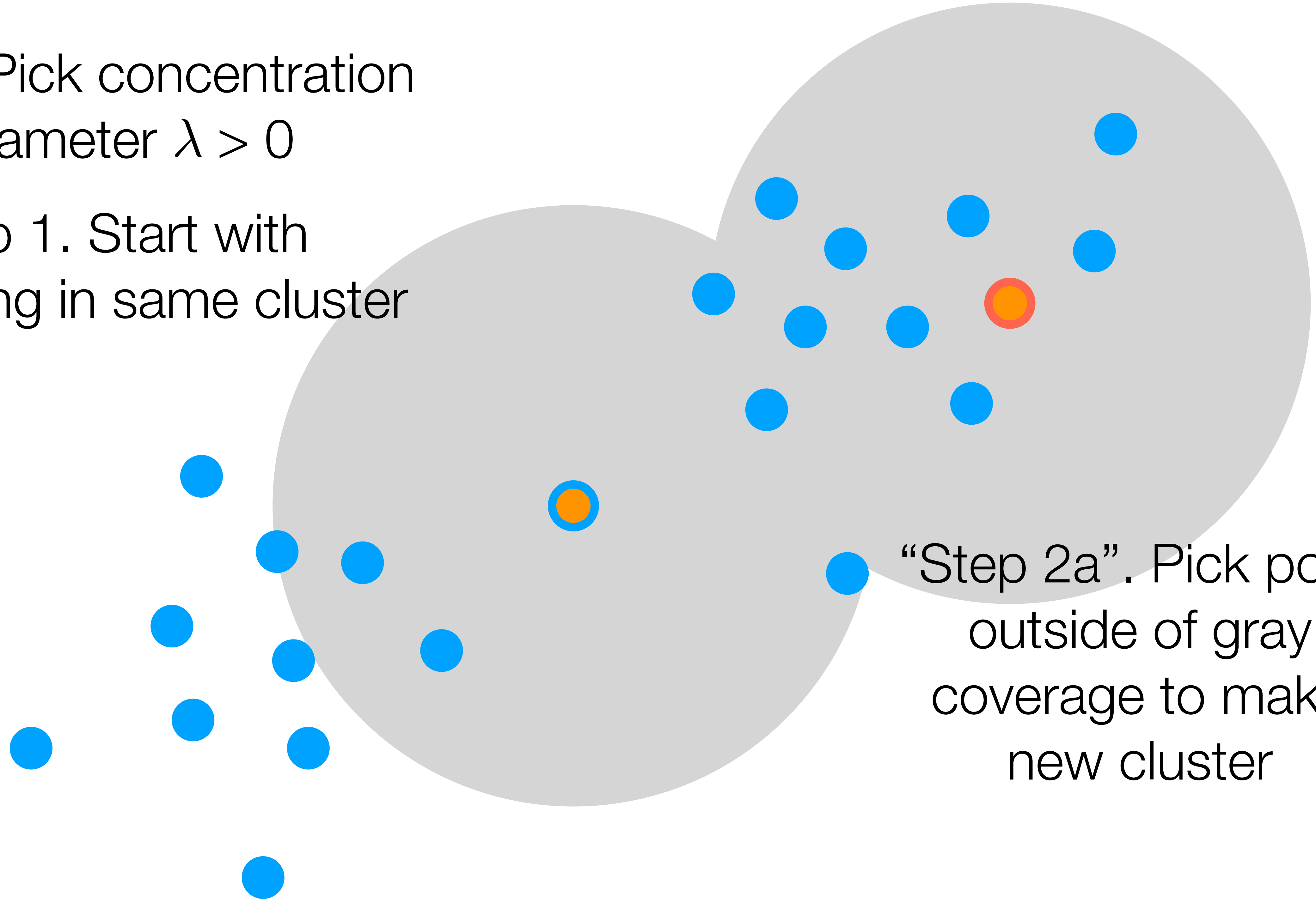


"Step 2a". Pick point outside of gray coverage to make new cluster

DP-means

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

Step 1. Start with everything in same cluster

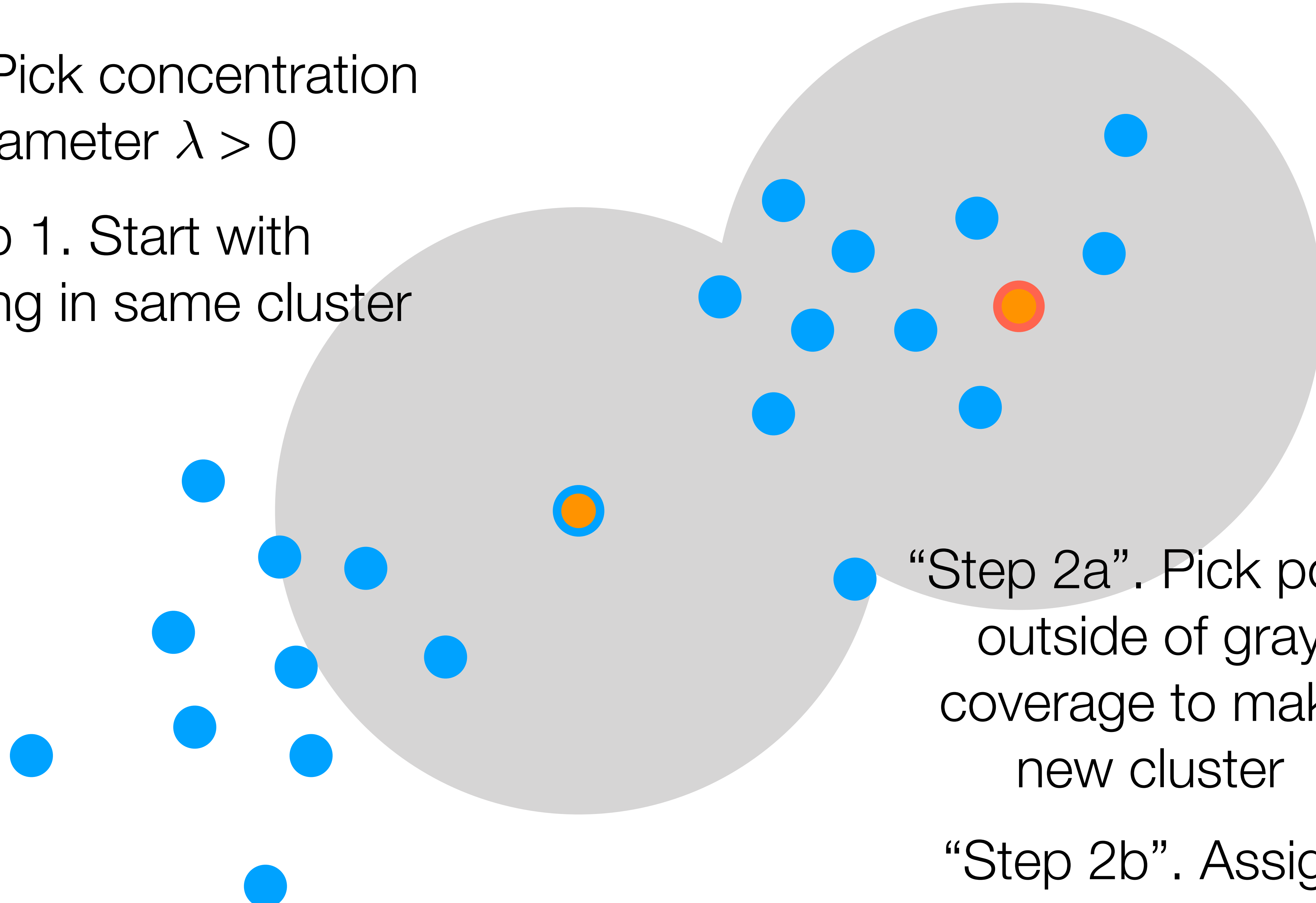


"Step 2a". Pick point outside of gray coverage to make new cluster

DP-means

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

Step 1. Start with everything in same cluster



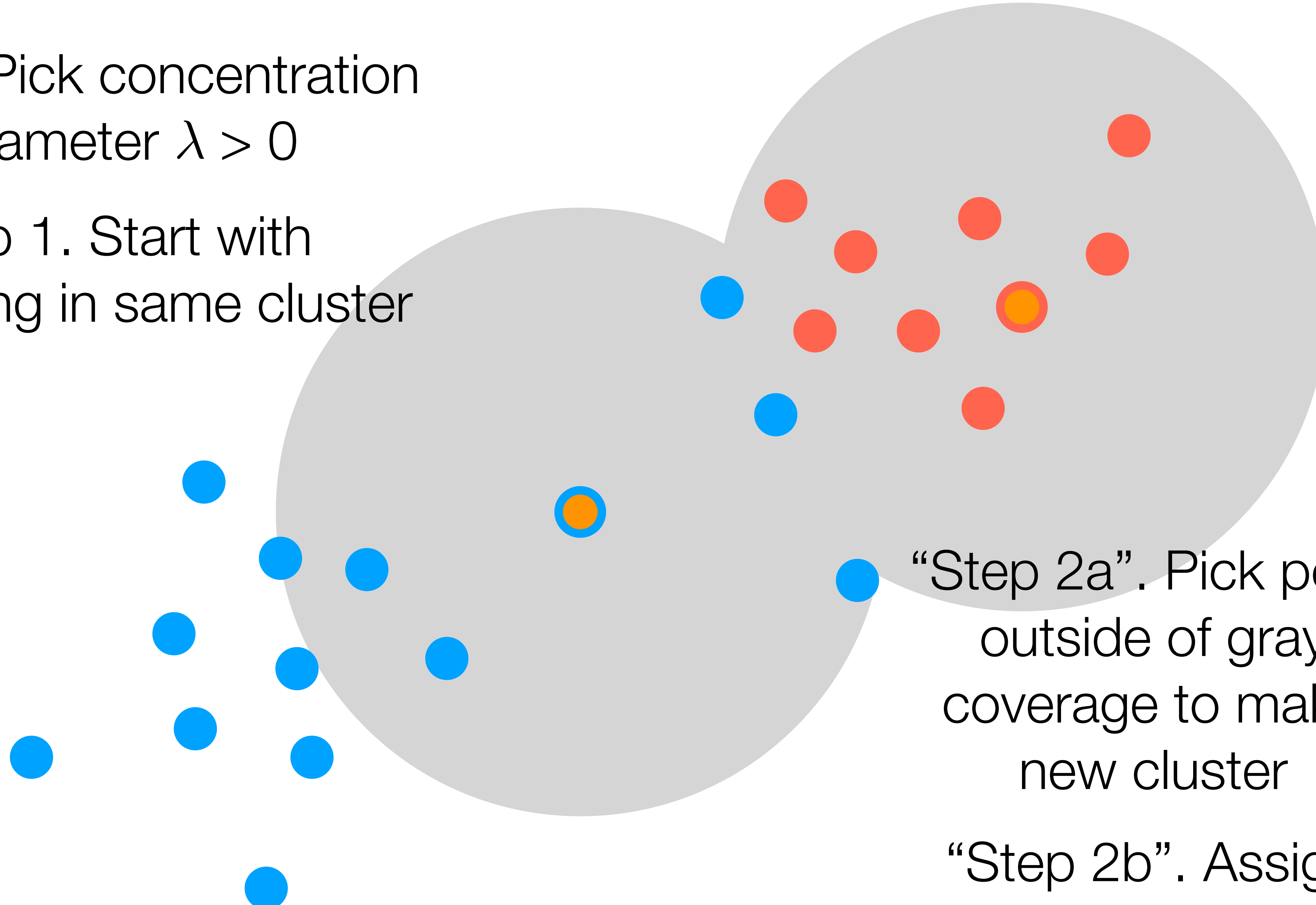
"Step 2a". Pick point outside of gray coverage to make new cluster

"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



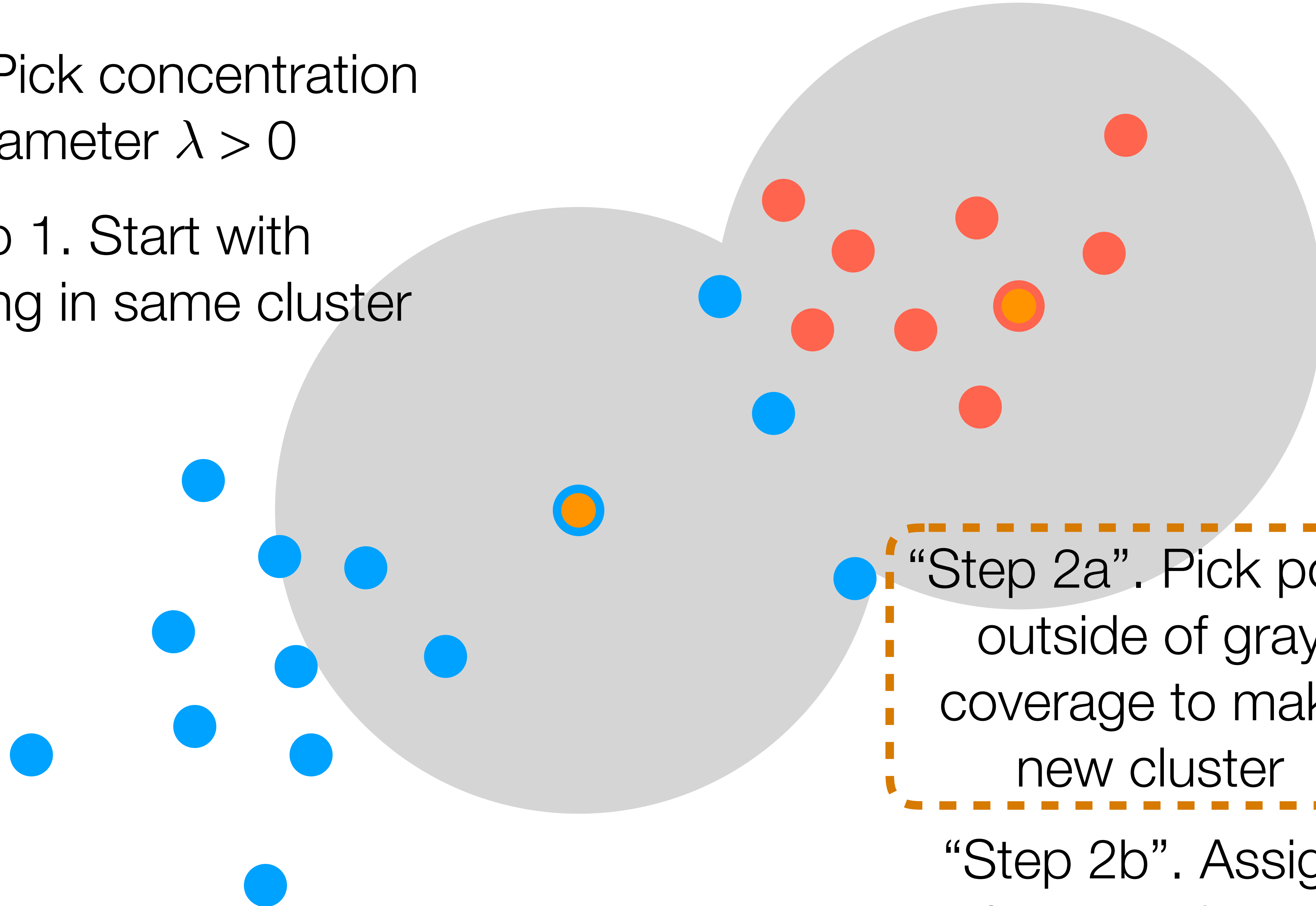
“Step 2a”. Pick point outside of gray coverage to make new cluster

“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



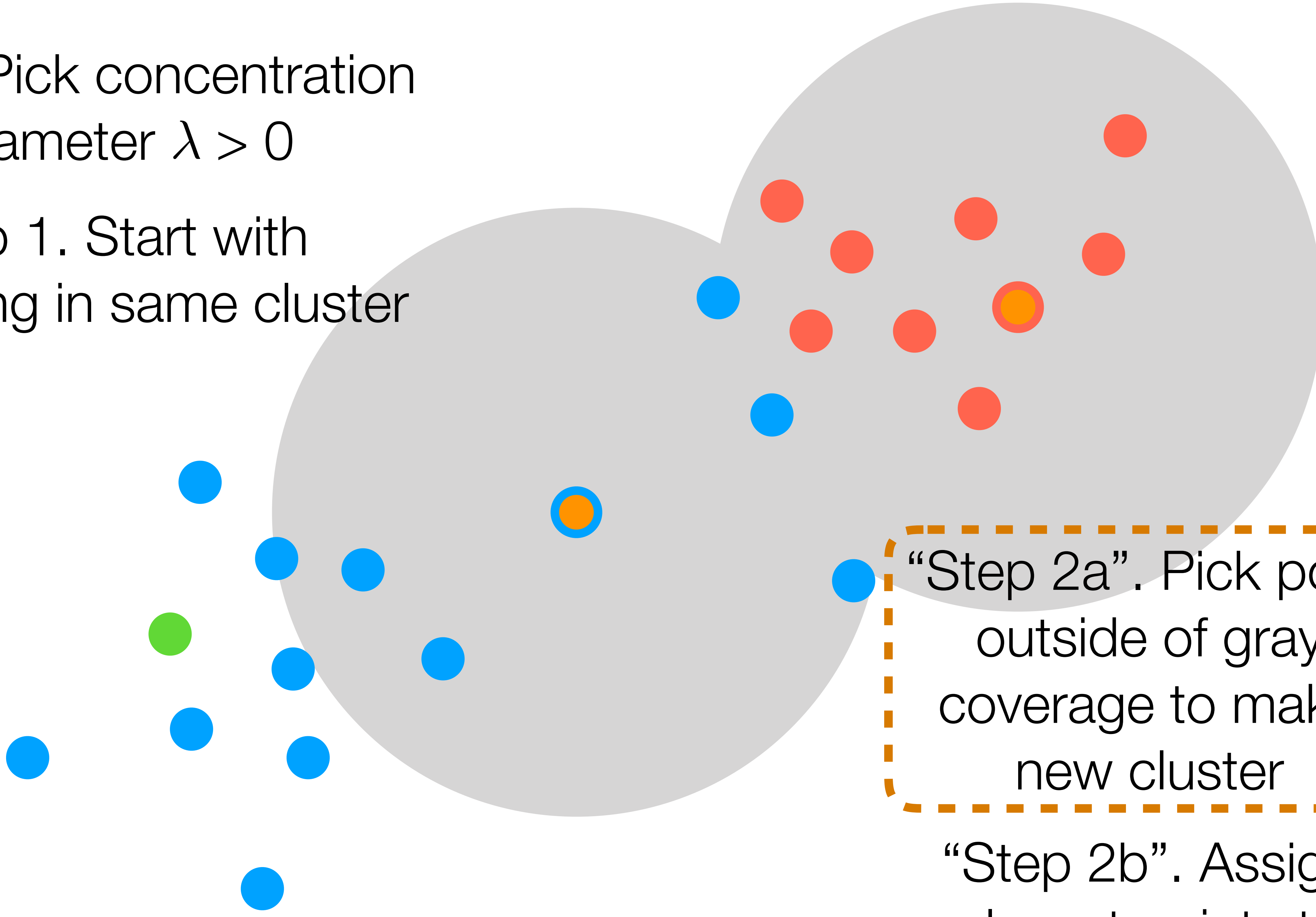
"Step 2a". Pick point outside of gray coverage to make new cluster

"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



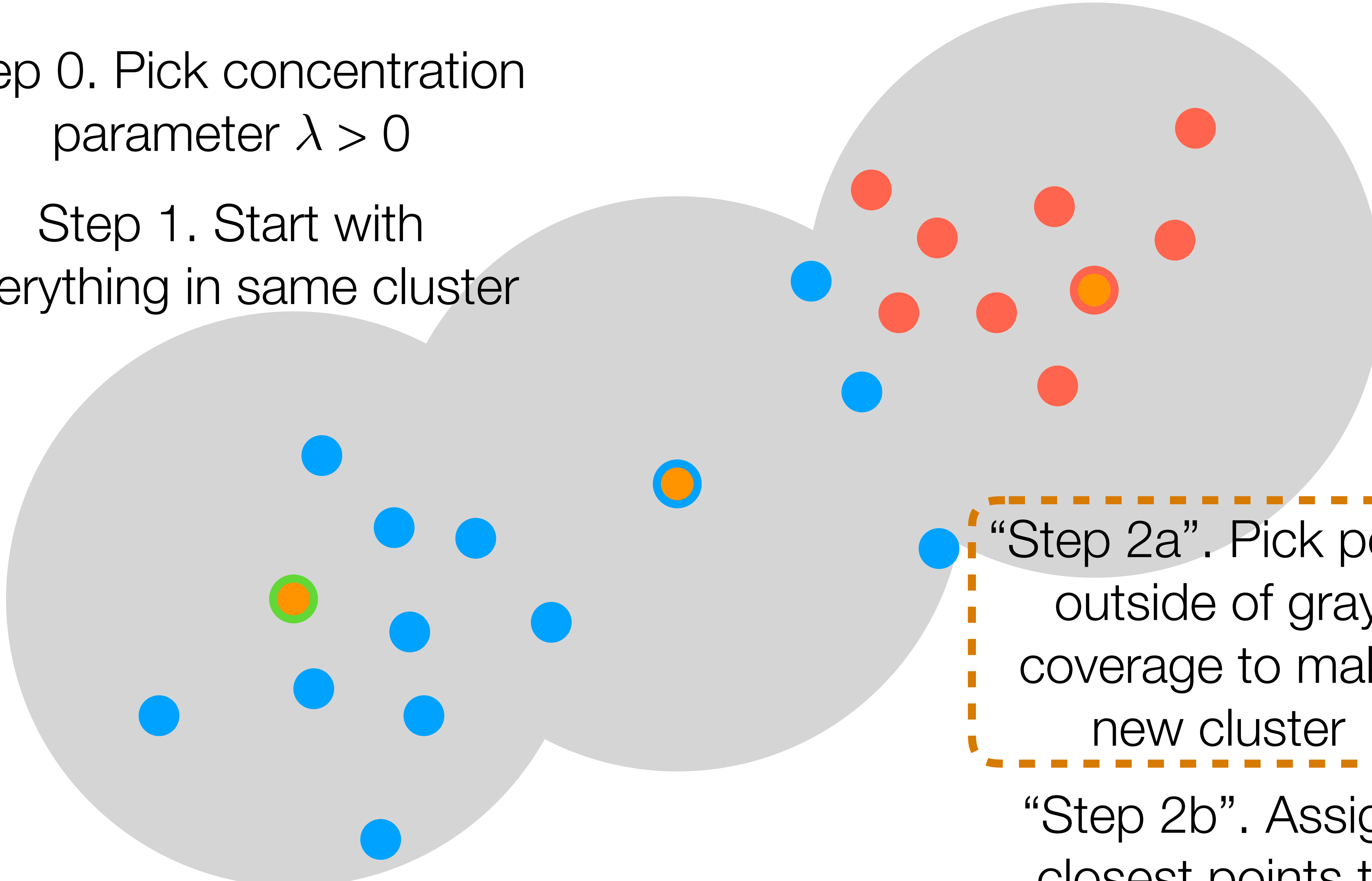
"Step 2a". Pick point outside of gray coverage to make new cluster

"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



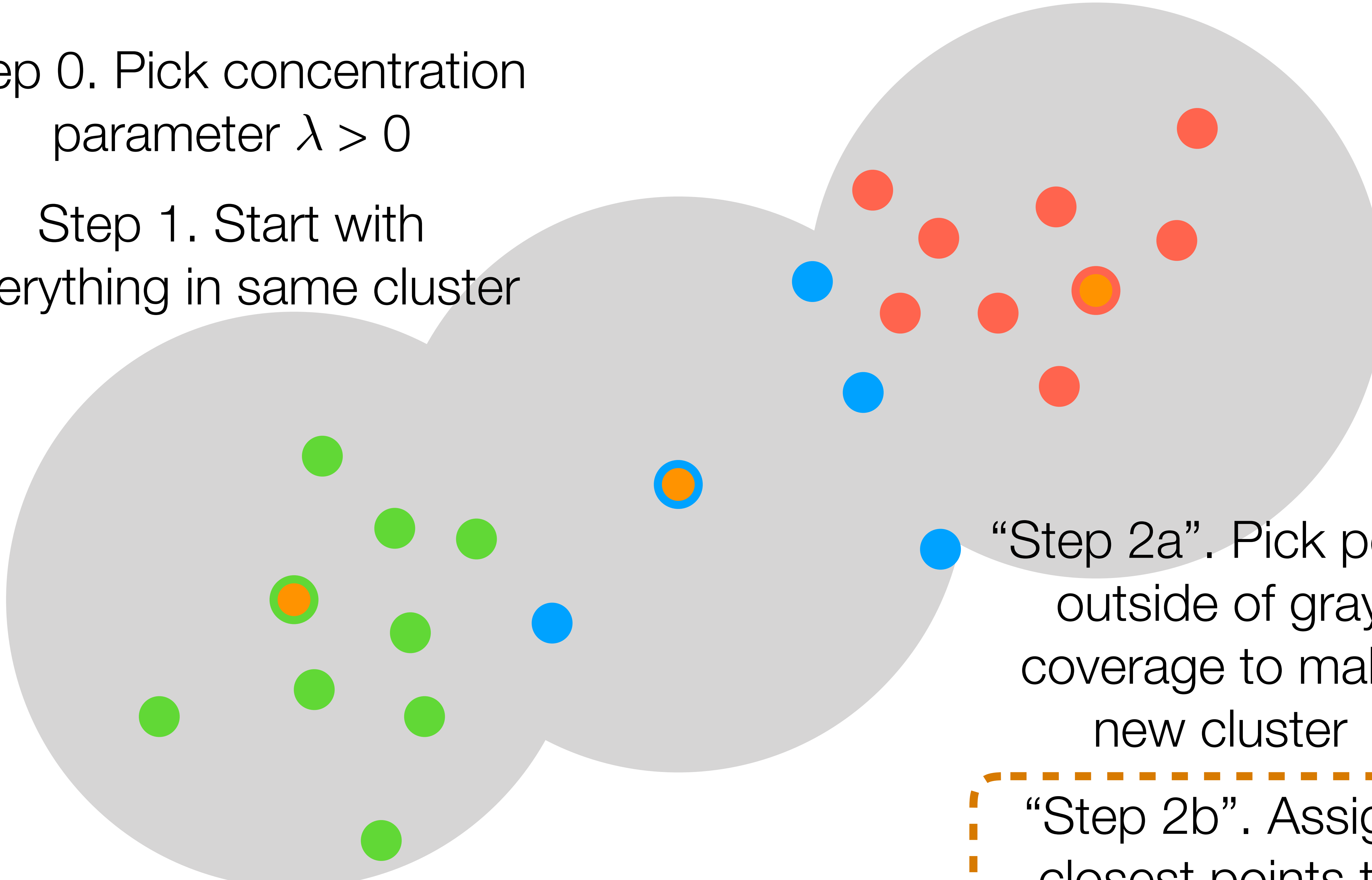
“Step 2a”. Pick point outside of gray coverage to make new cluster

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

DP-means

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

Step 1. Start with everything in same cluster



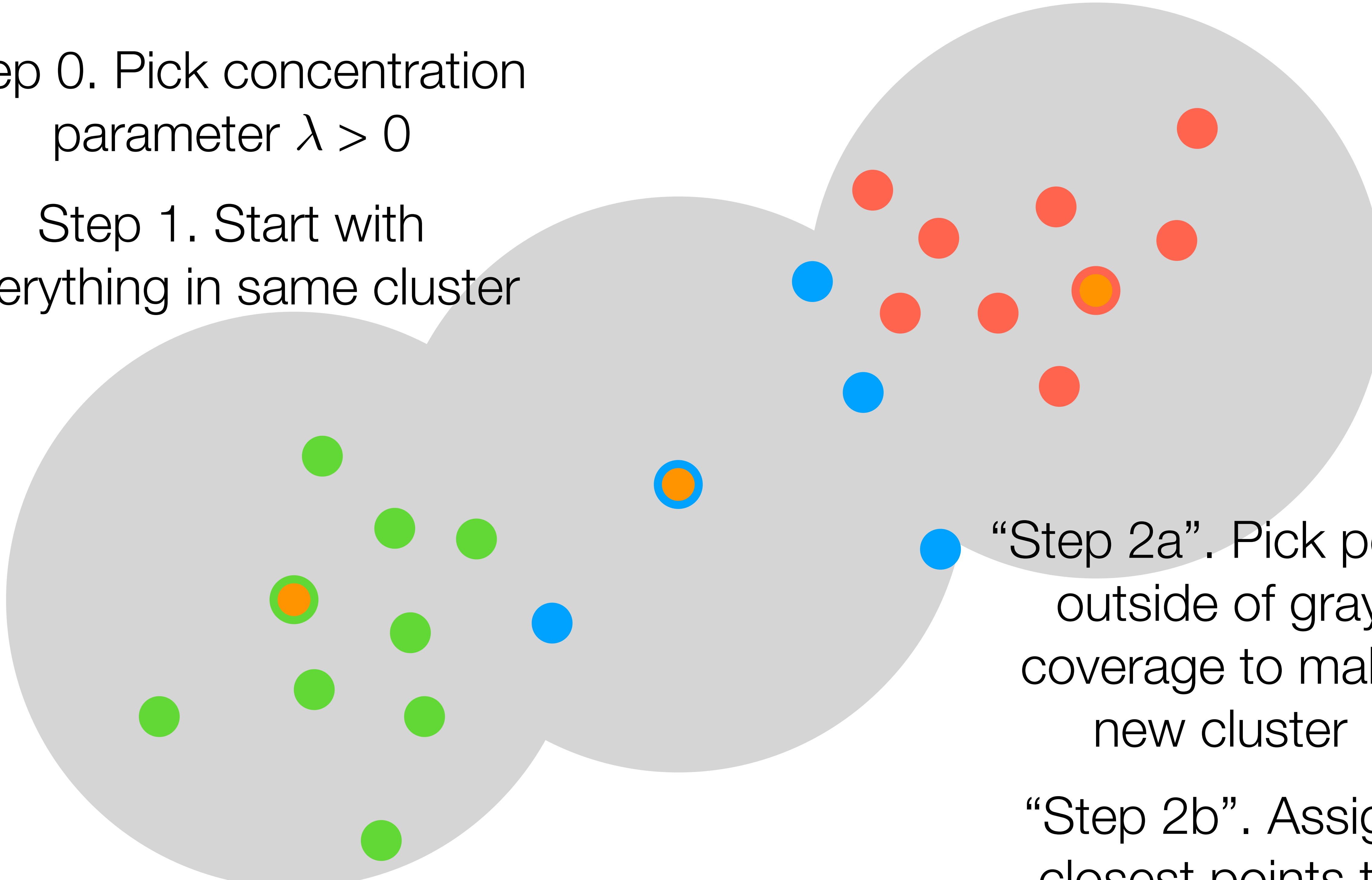
“Step 2a”. Pick point outside of gray coverage to make new cluster

“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



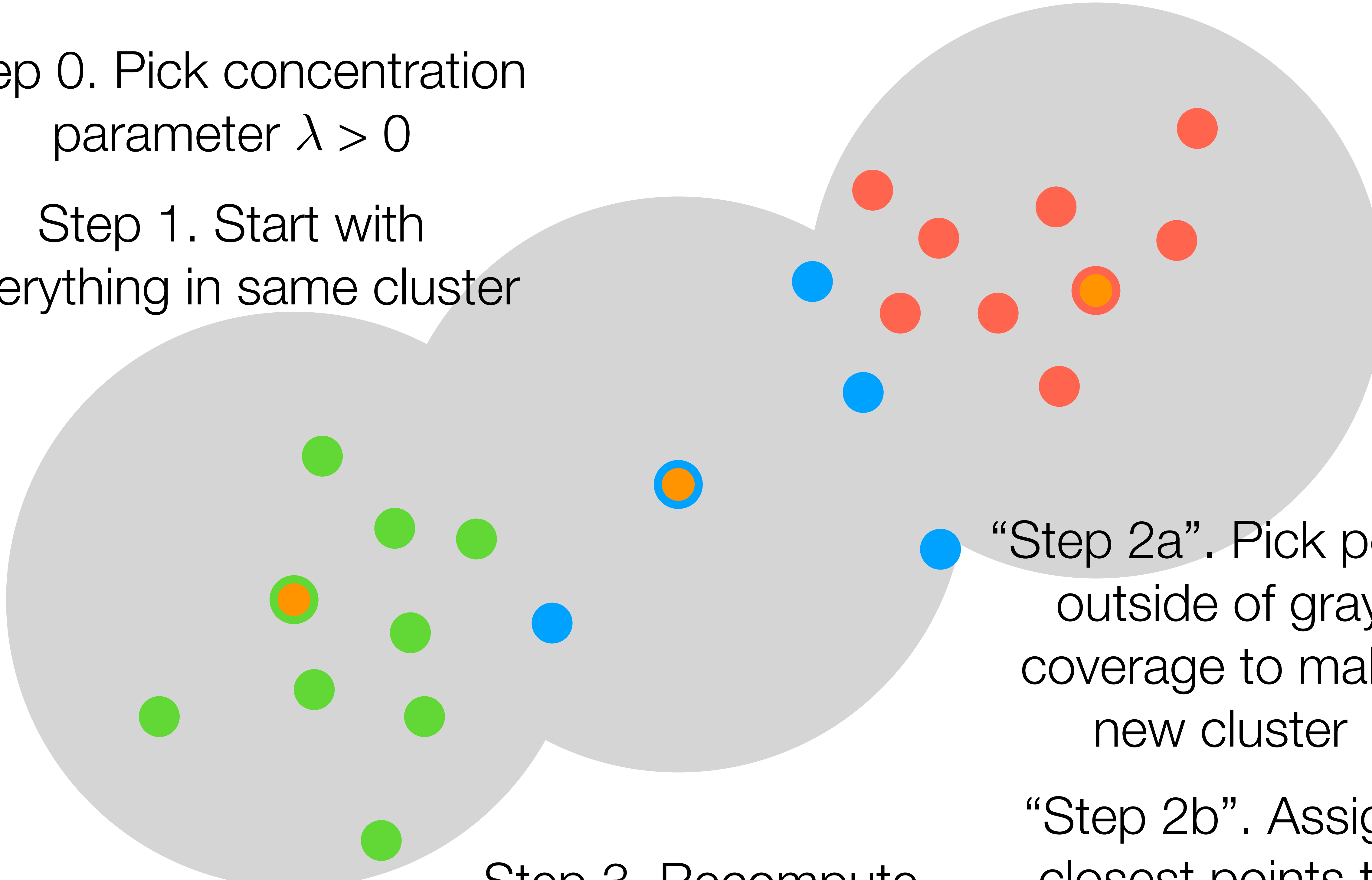
"Step 2a". Pick point outside of gray coverage to make new cluster

"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

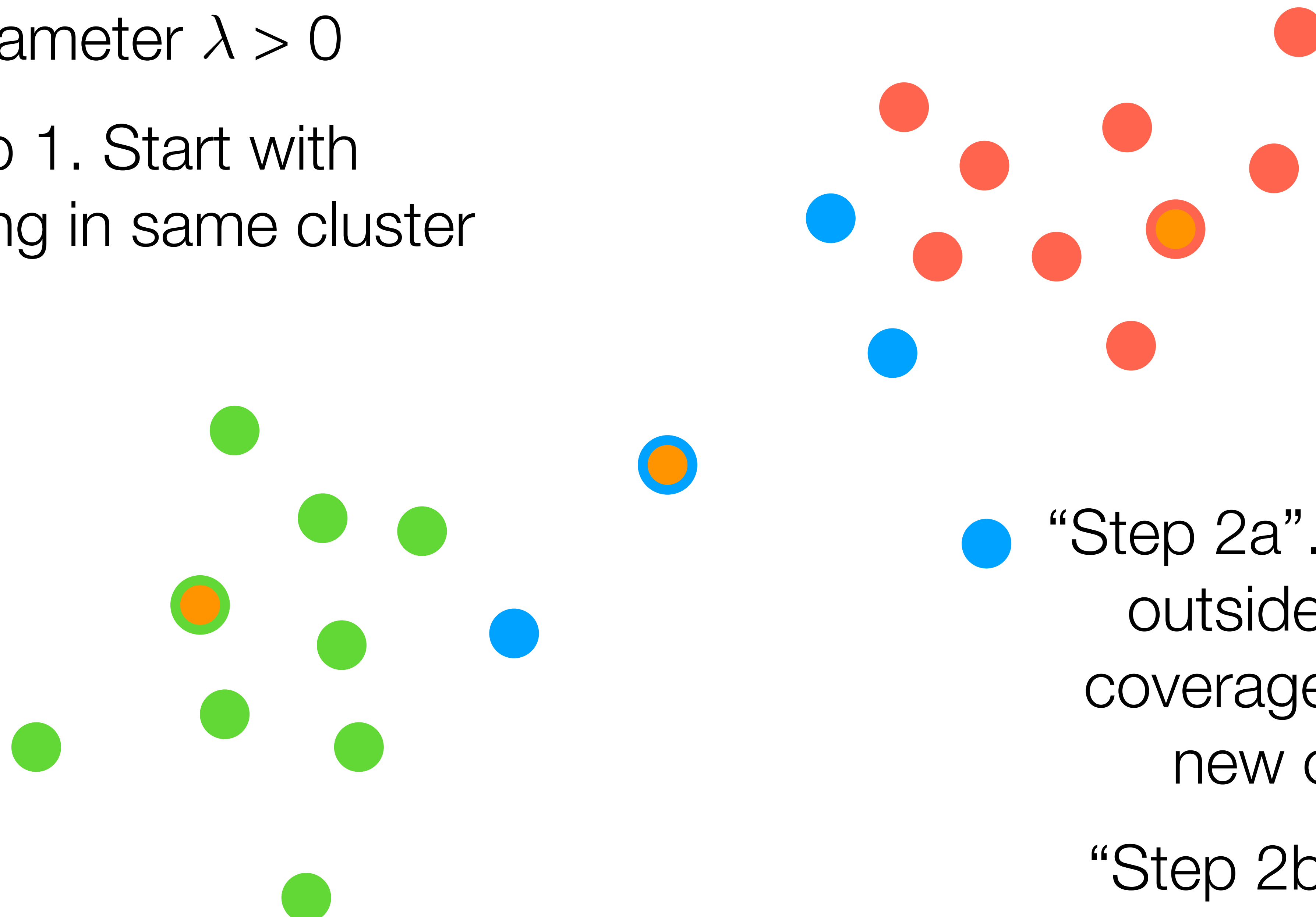
“Step 2b”. Assign closest points to current clusters

Step 3. Recompute cluster centers

DP-means

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

Step 1. Start with everything in same cluster



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

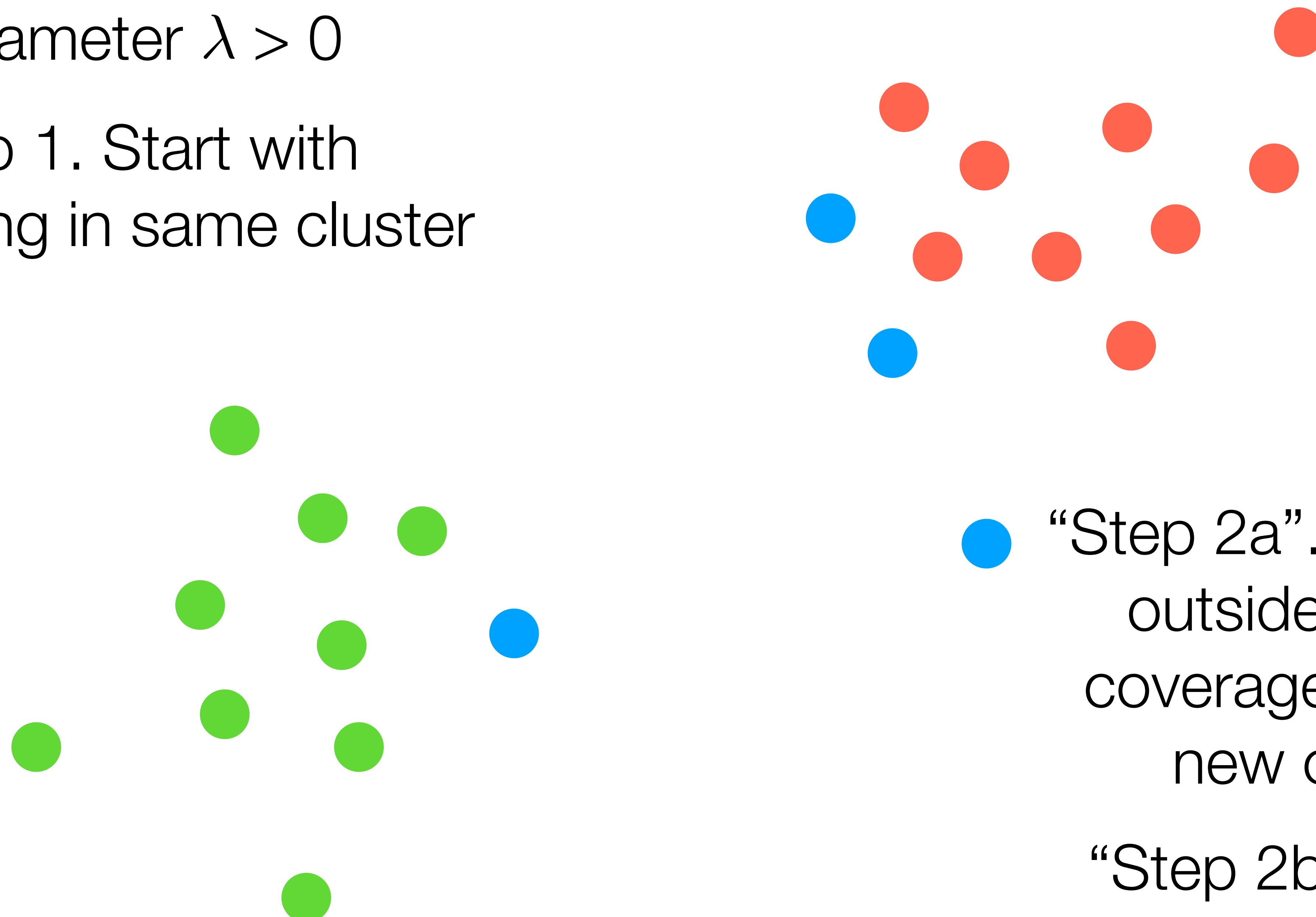
"Step 2b". Assign closest points to current clusters

Step 3. Recompute cluster centers

DP-means

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

Step 1. Start with everything in same cluster



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

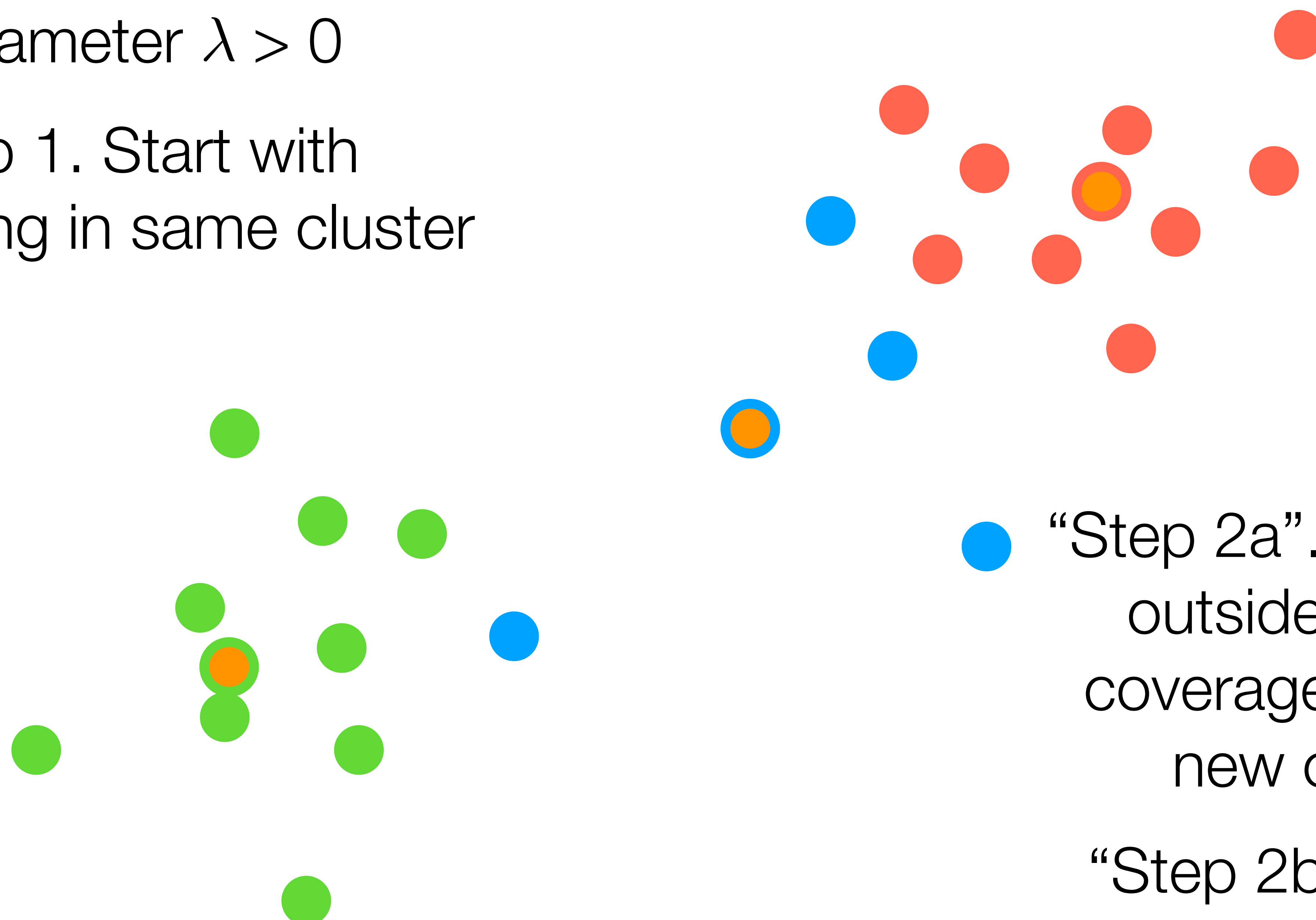
"Step 2b". Assign closest points to current clusters

Step 3. Recompute cluster centers

DP-means

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

Step 1. Start with everything in same cluster



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

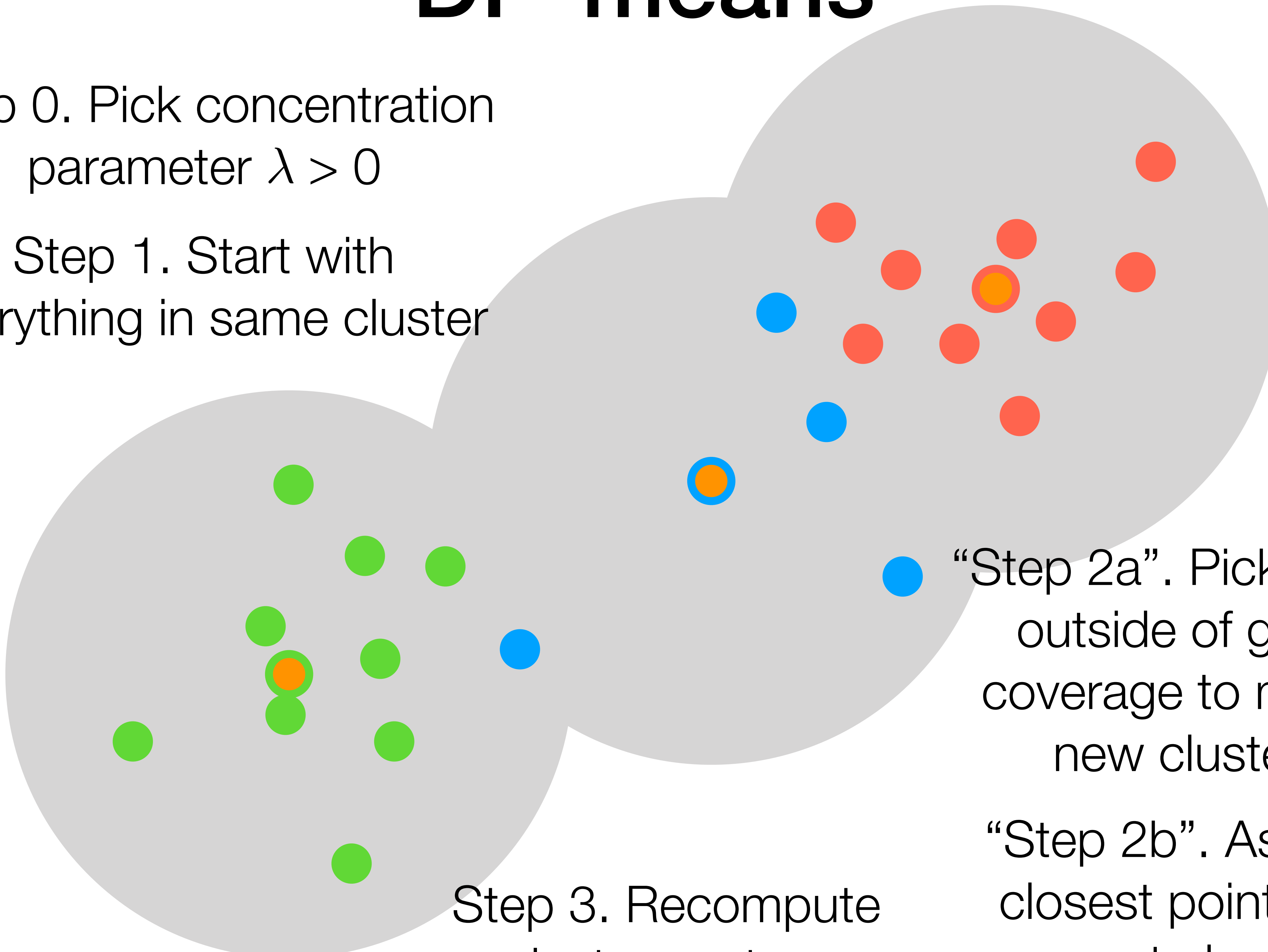
“Step 2b”. Assign closest points to current clusters

Step 3. Recompute cluster centers

DP-means

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

Step 1. Start with everything in same cluster



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

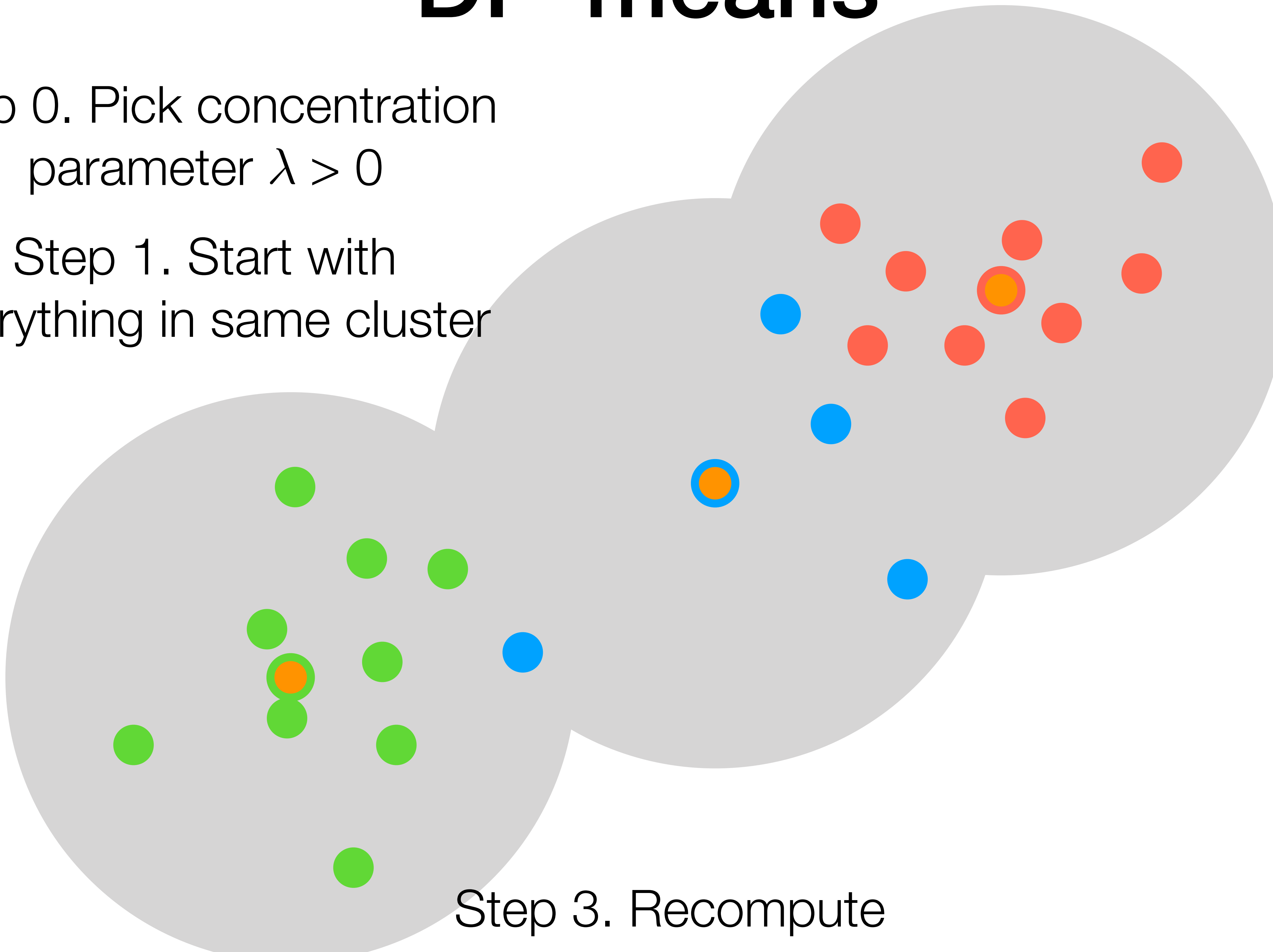
"Step 2b". Assign closest points to current clusters

Step 3. Recompute cluster centers

DP-means

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

Step 1. Start with everything in same cluster

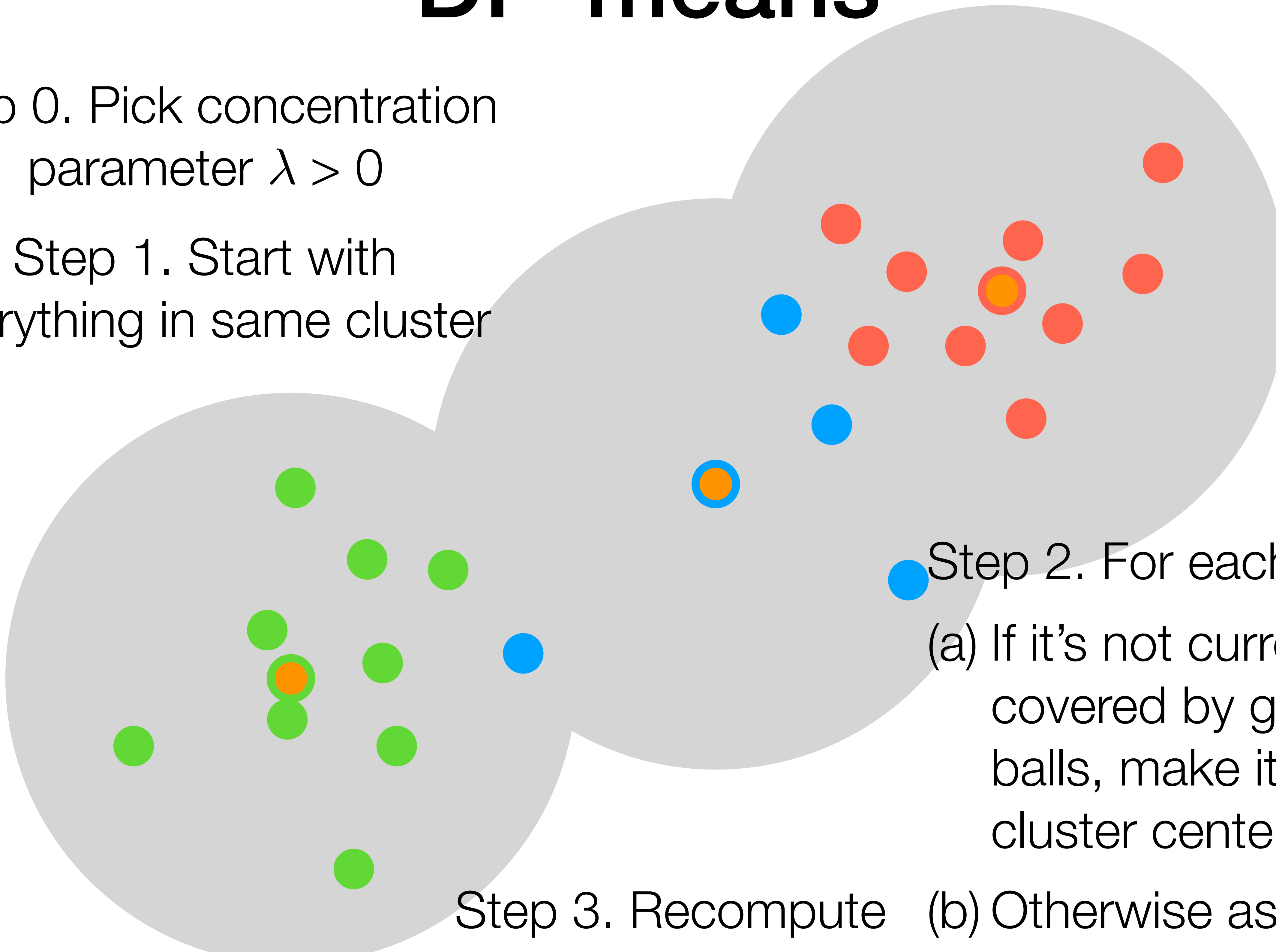


Step 3. Recompute cluster centers

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

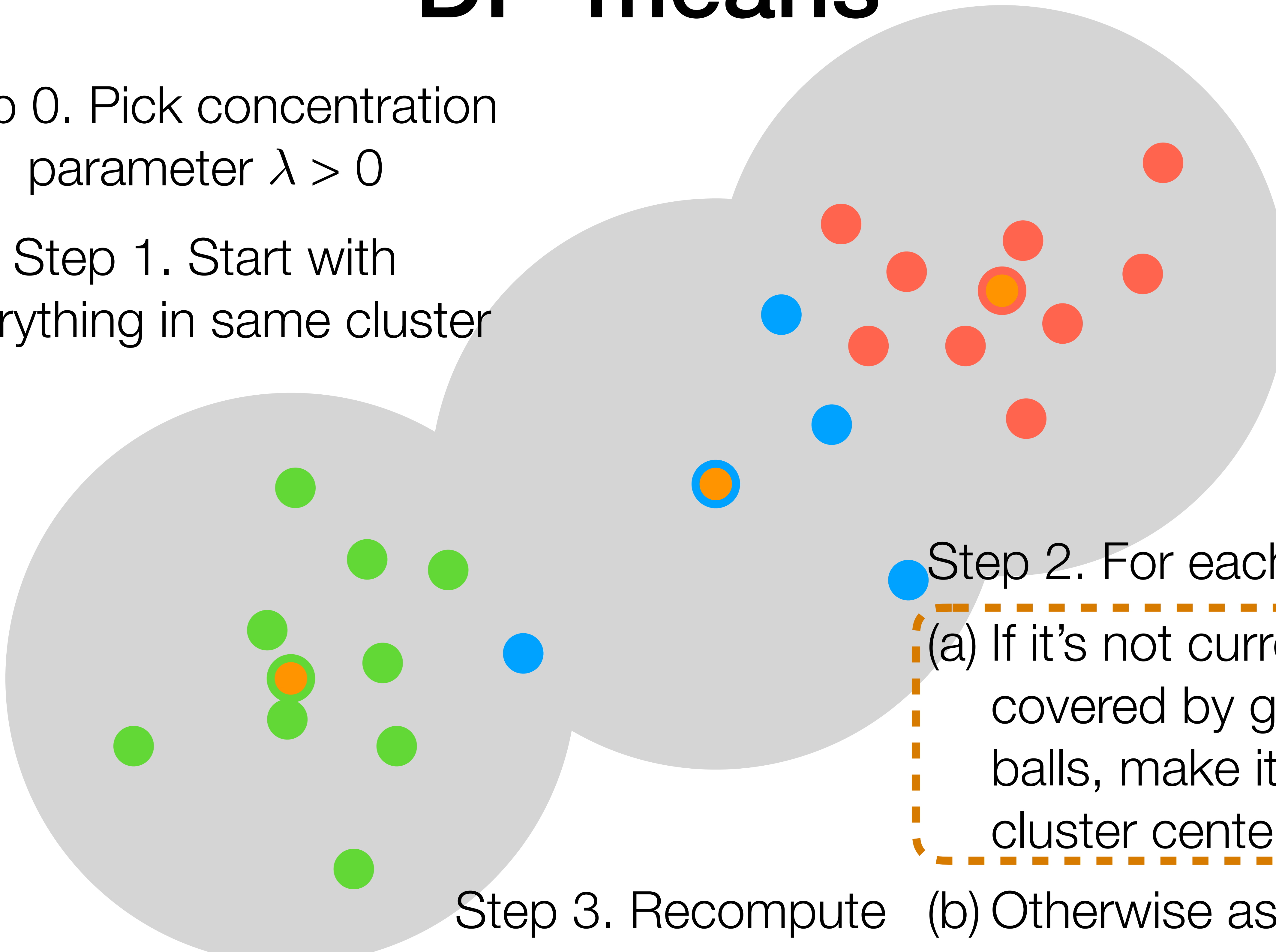
(b) Otherwise assign it to nearest cluster

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

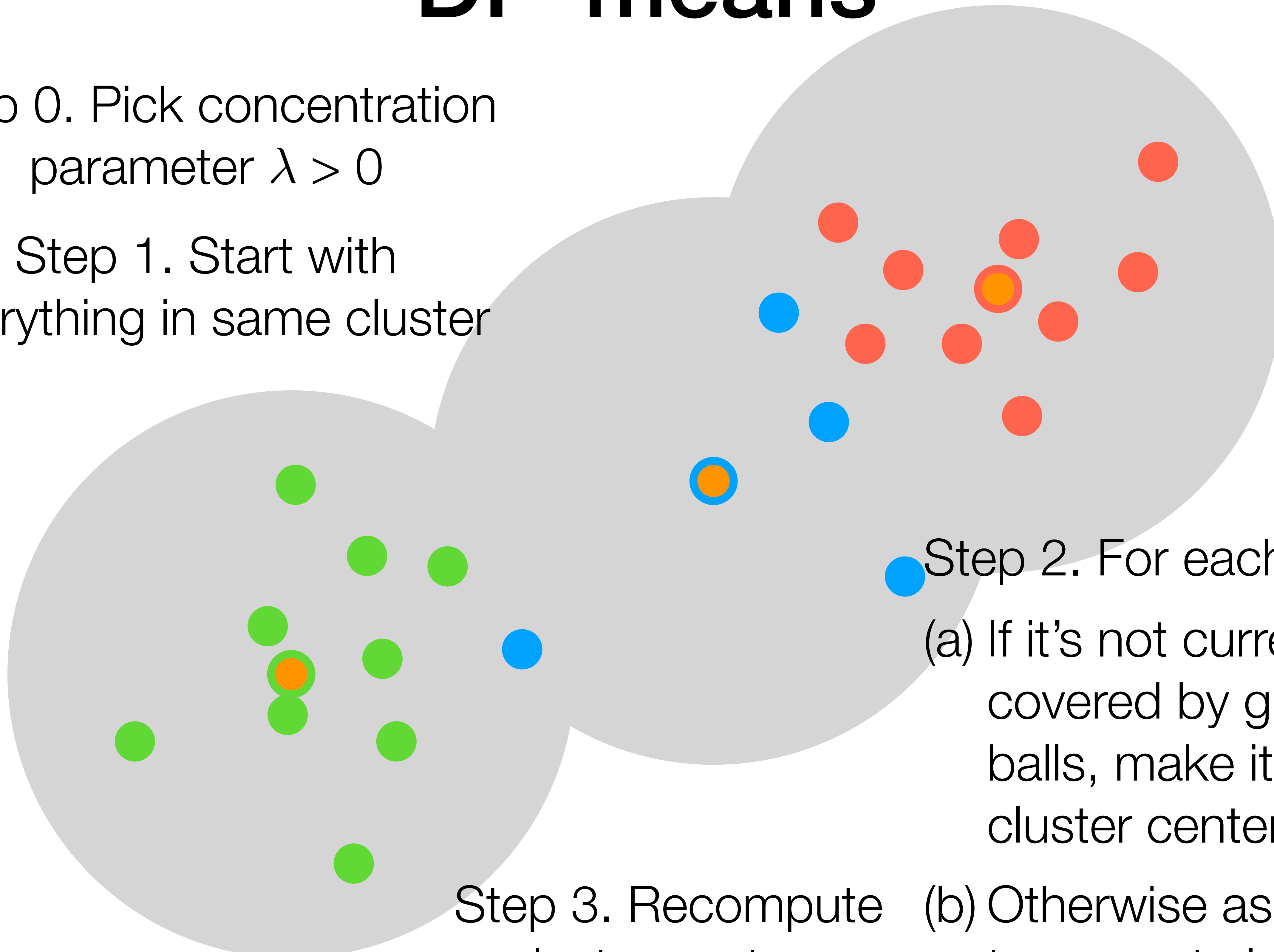
(b) Otherwise assign it to nearest cluster

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

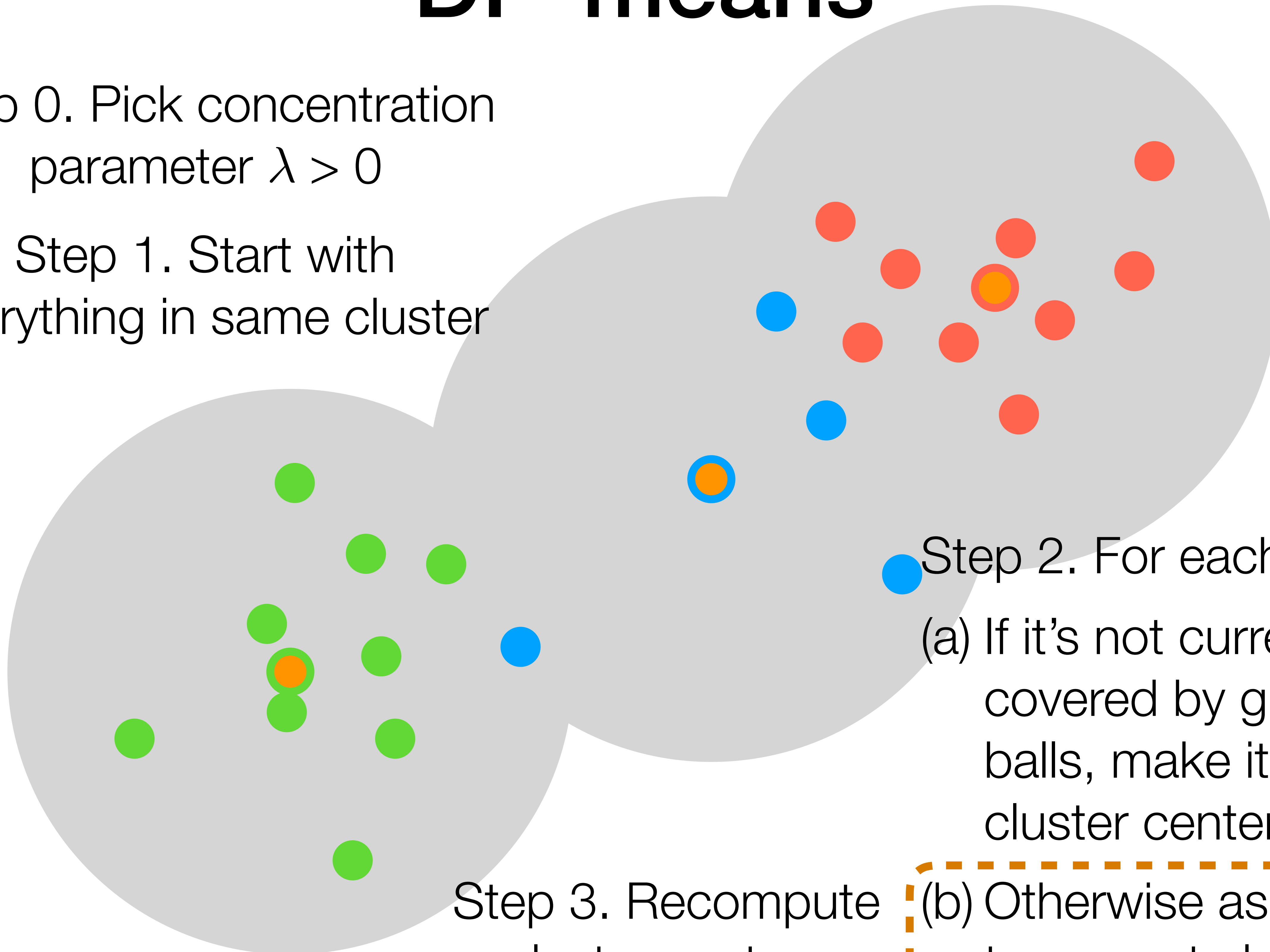
(b) Otherwise assign it to nearest cluster

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

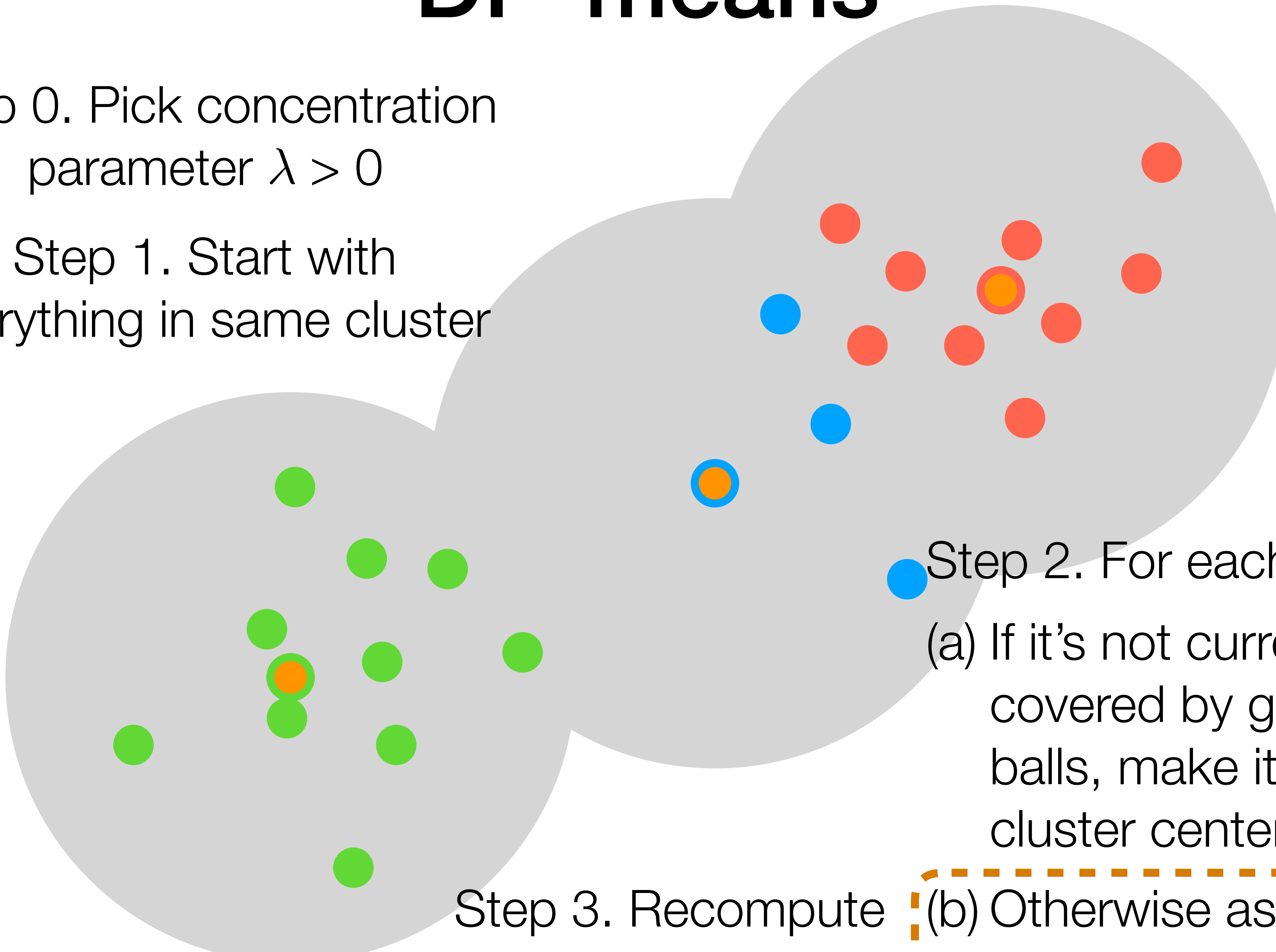
(b) Otherwise assign it to nearest cluster

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

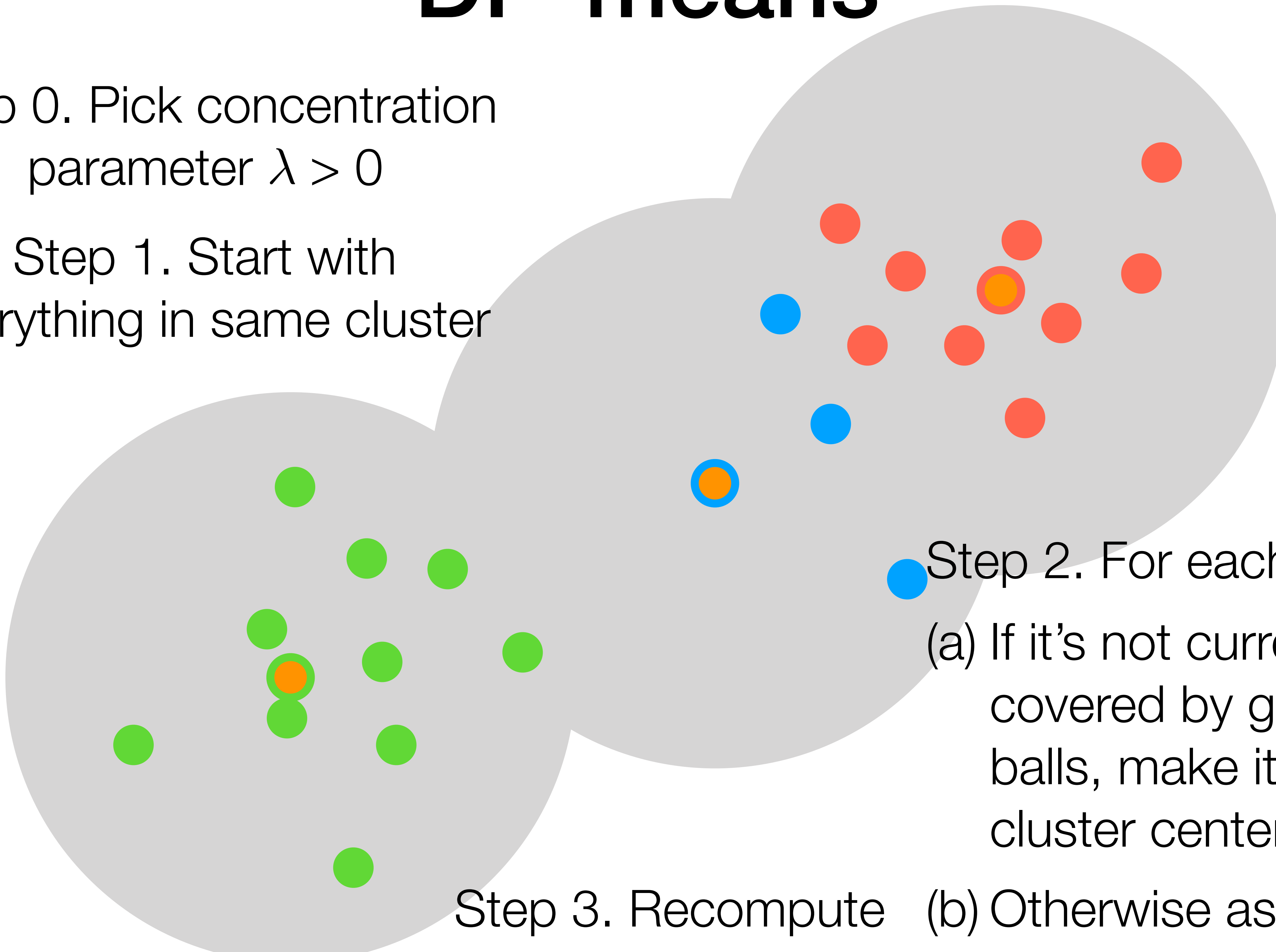
(b) Otherwise assign it to nearest cluster

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

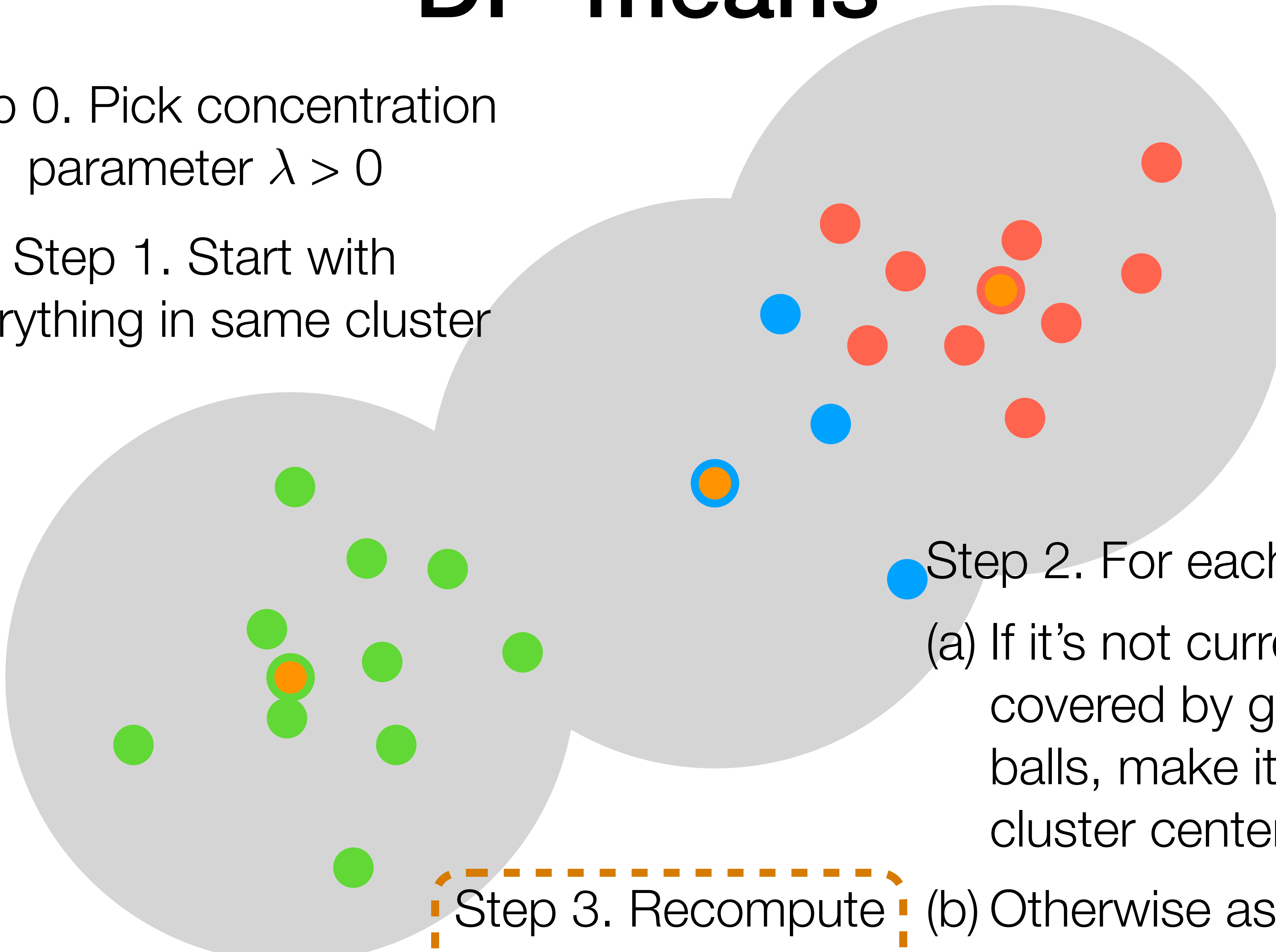
(b) Otherwise assign it to nearest cluster

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

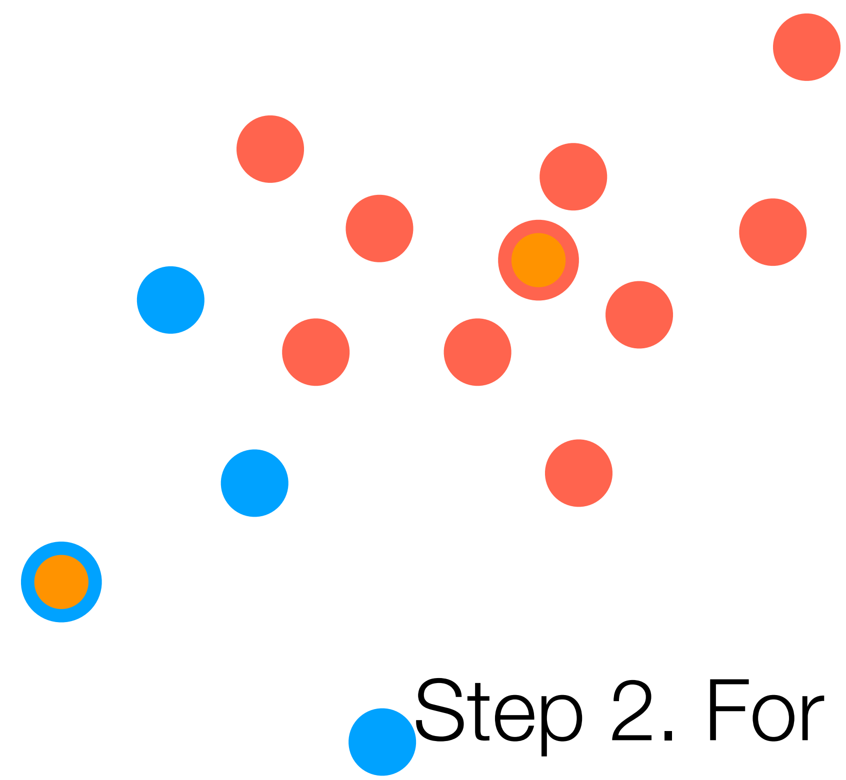
(b) Otherwise assign it to nearest cluster

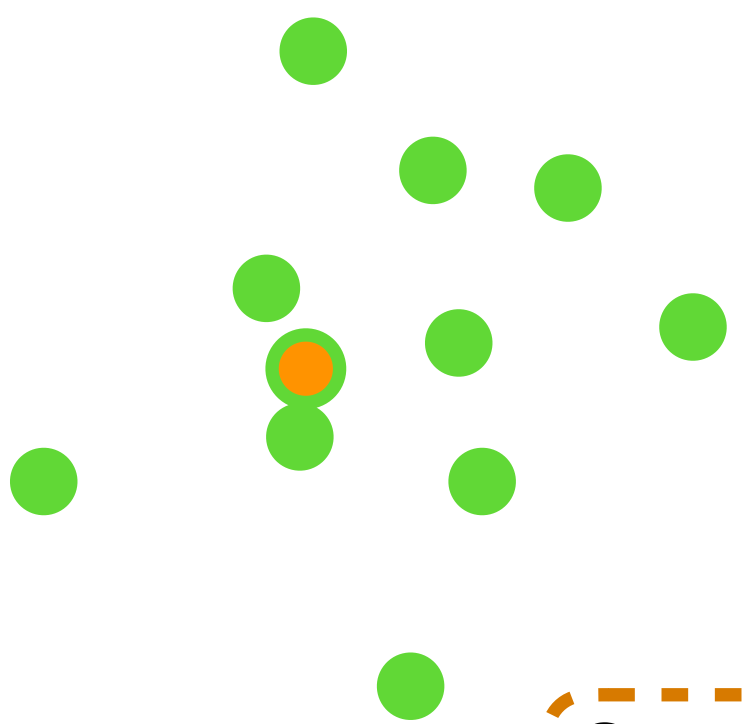
Step 3. Recompute cluster centers

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

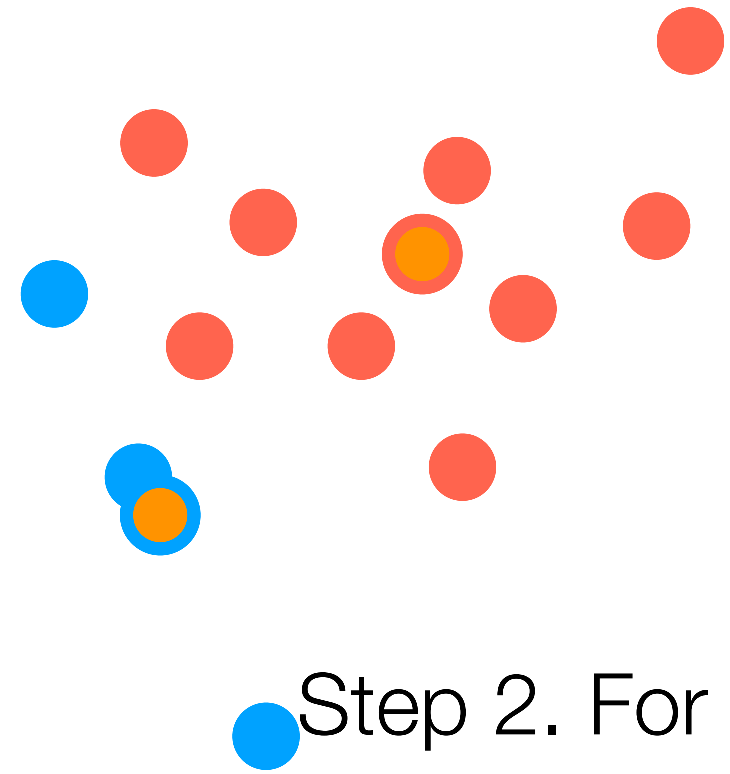
(b) Otherwise assign it to nearest cluster

Step 3. Recompute cluster centers

DP-means

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

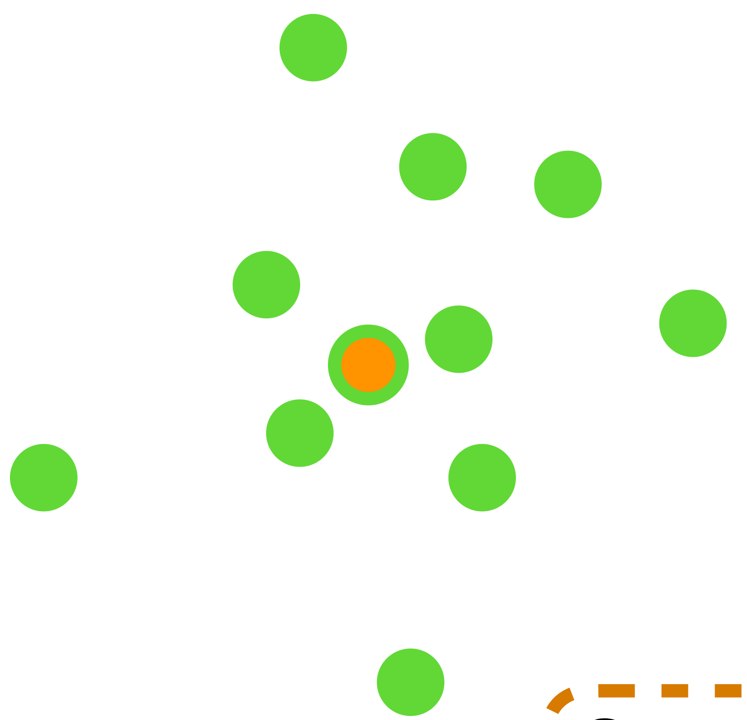
Step 1. Start with everything in same cluster



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

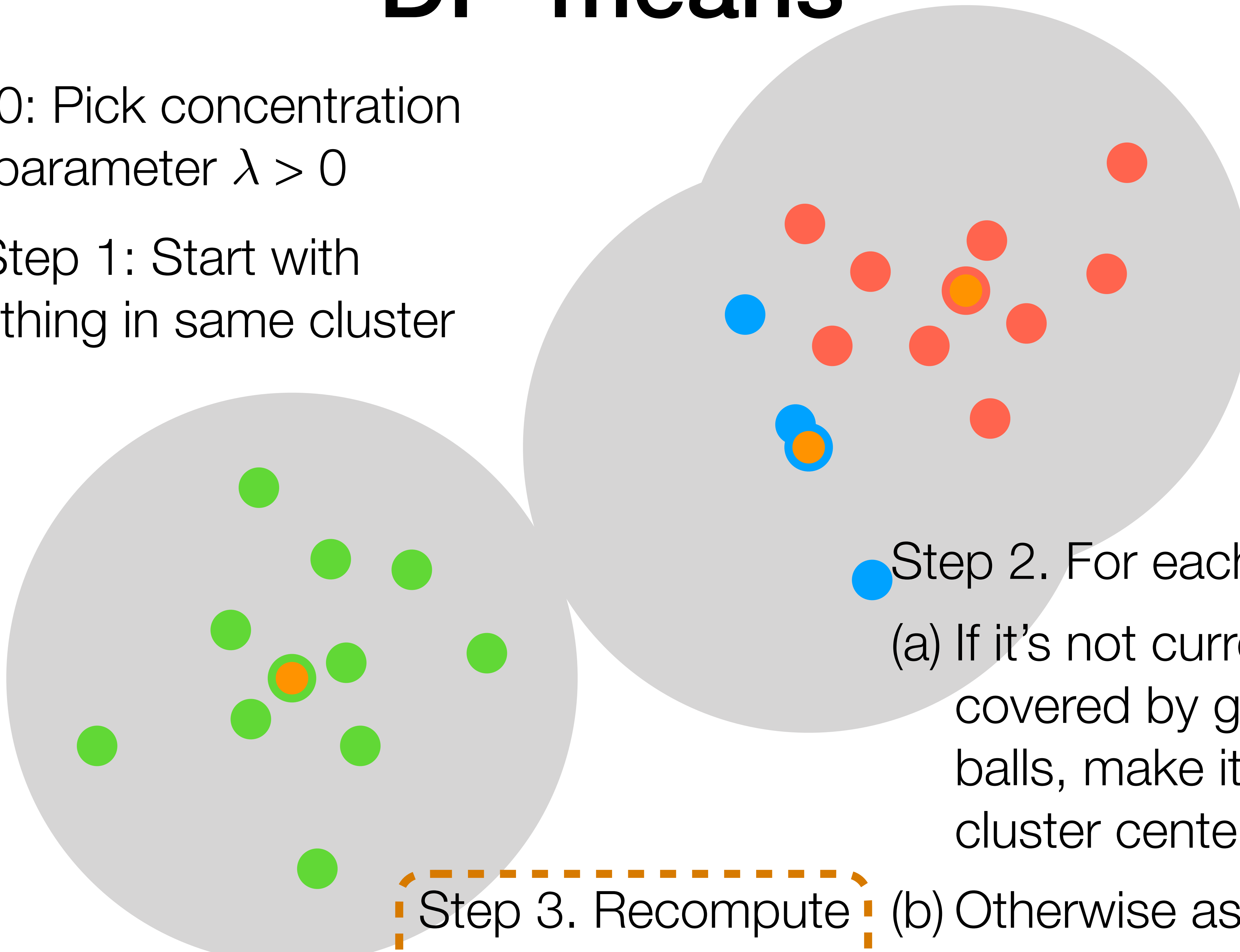


Step 3. Recompute cluster centers

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

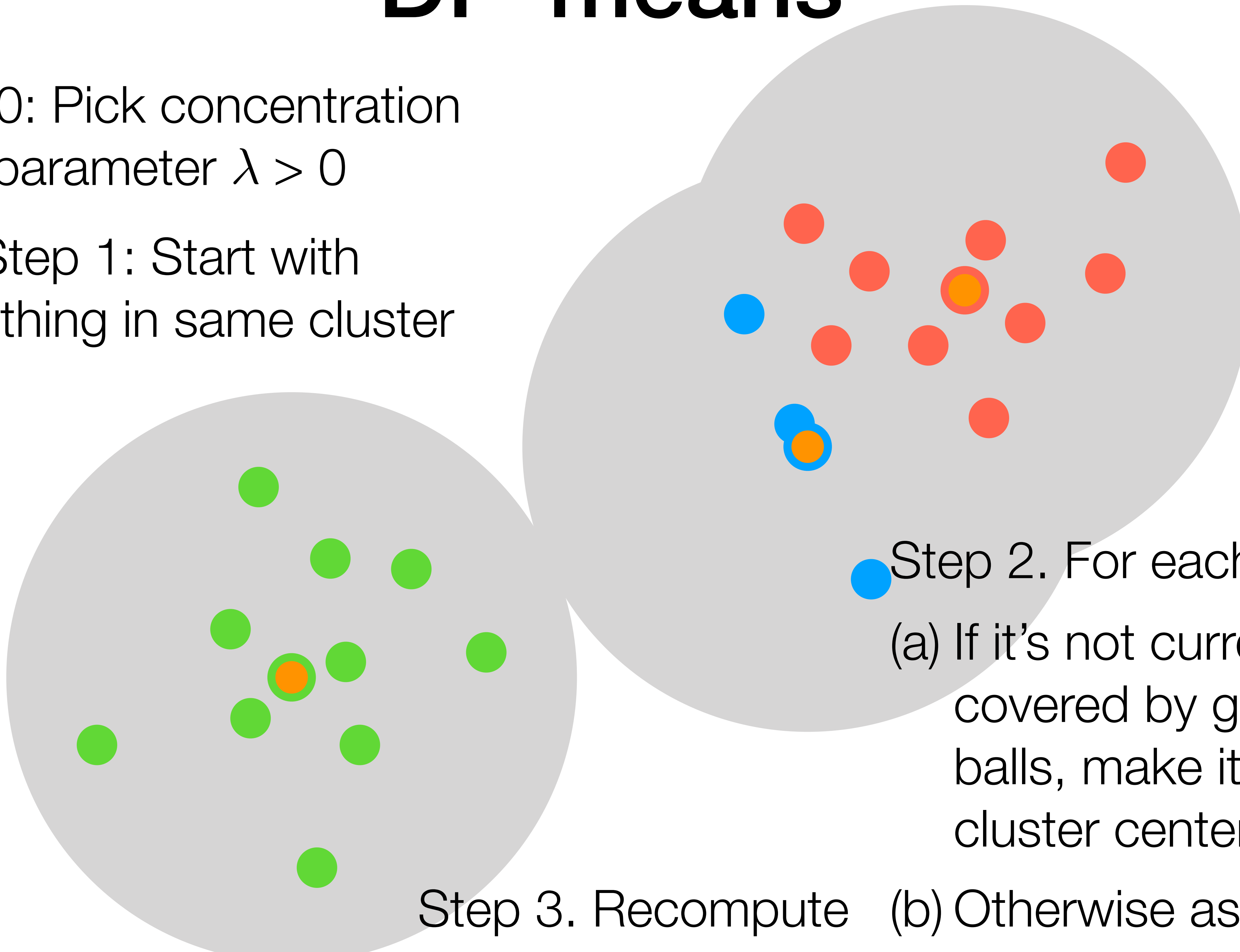
(b) Otherwise assign it to nearest cluster

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

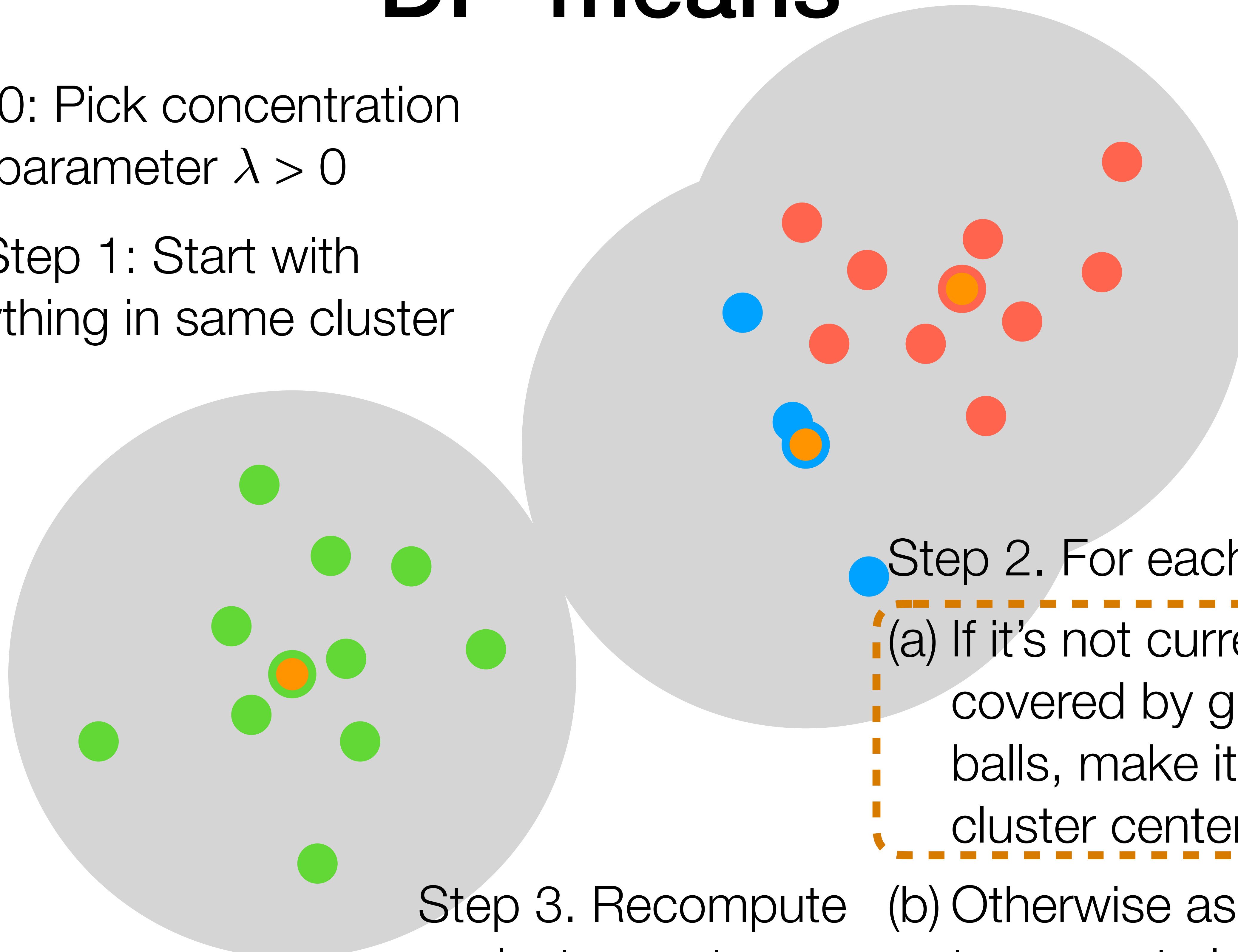
(b) Otherwise assign it to nearest cluster

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

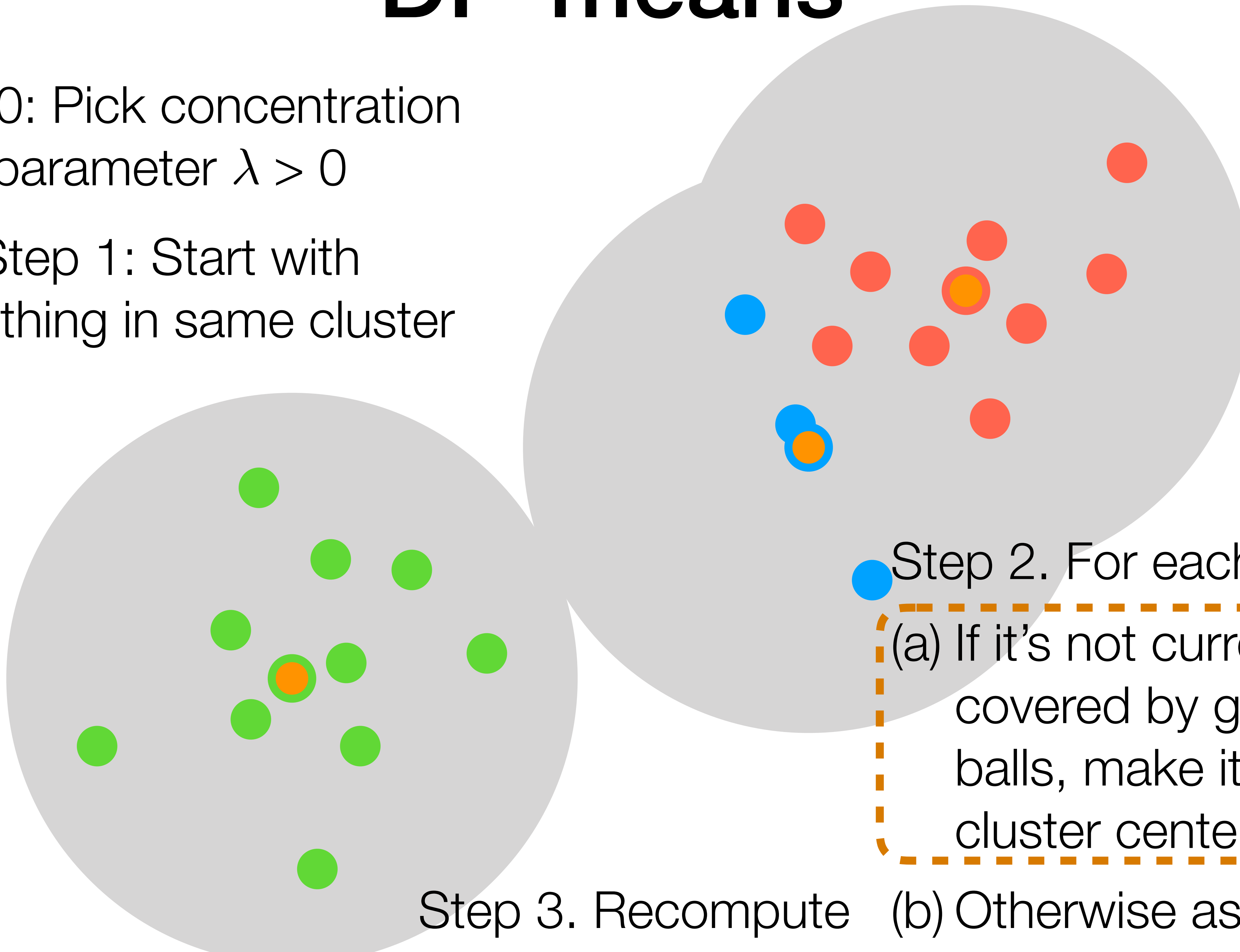
(b) Otherwise assign it to nearest cluster

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

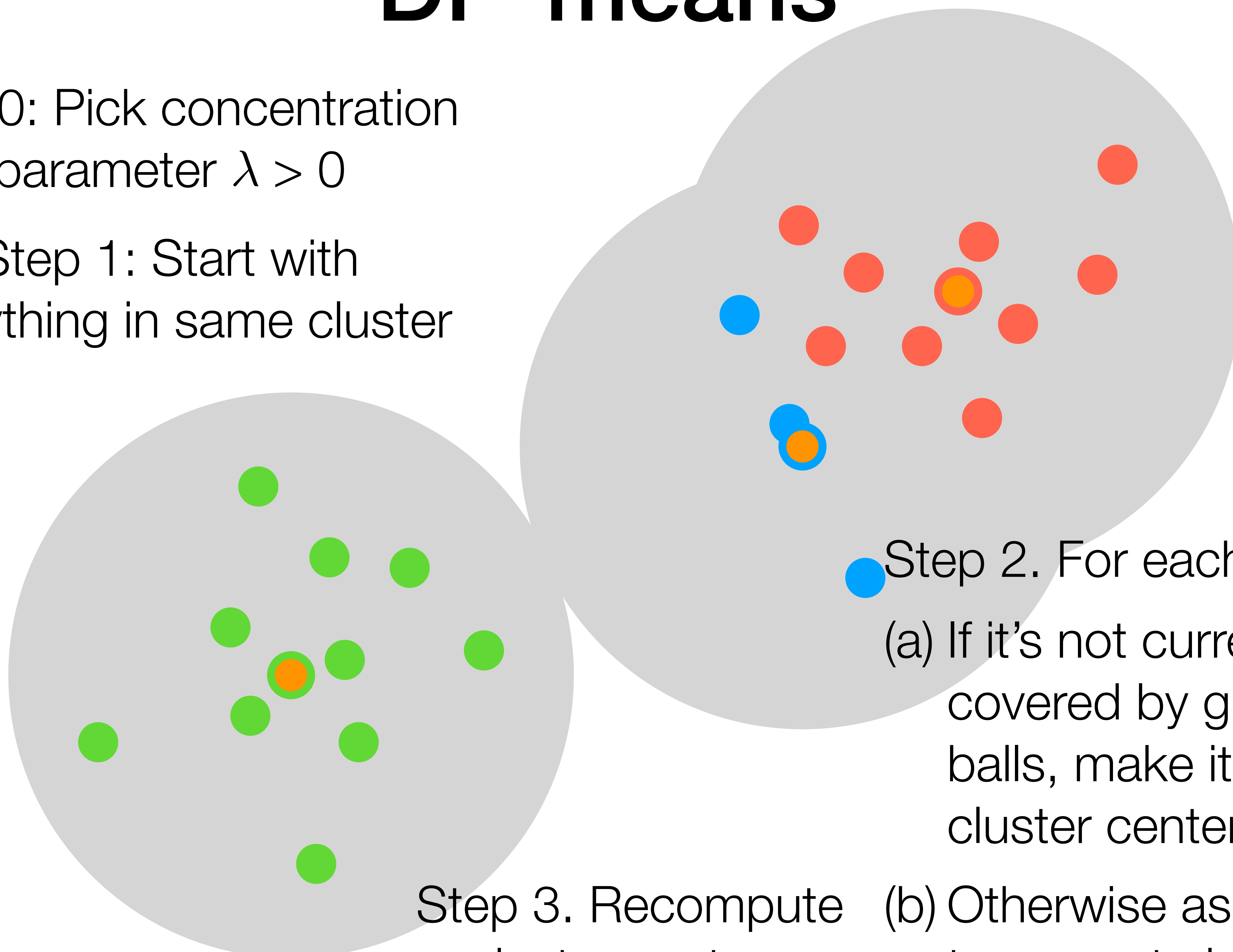
(b) Otherwise assign it to nearest cluster

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

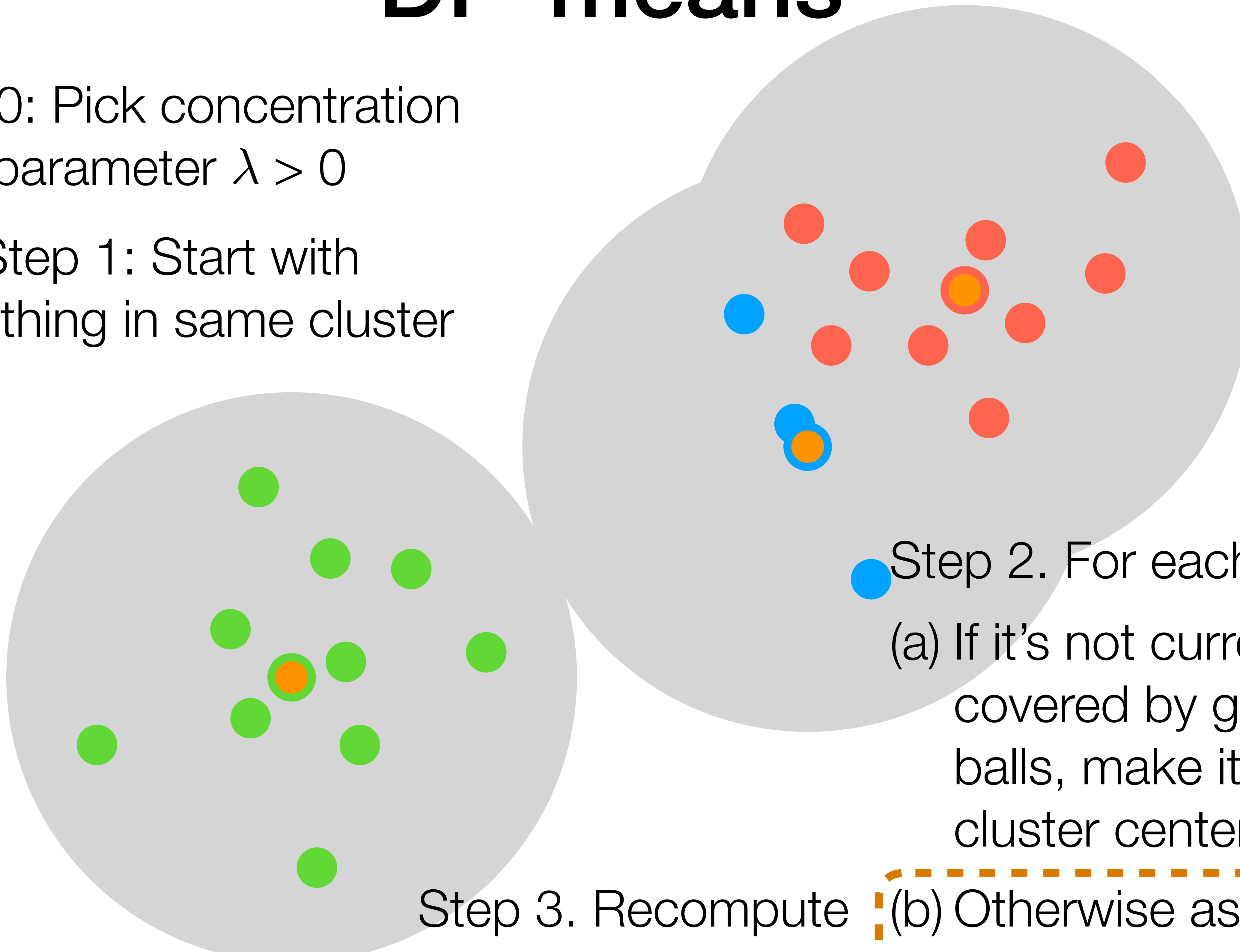
(b) Otherwise assign it to nearest cluster

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

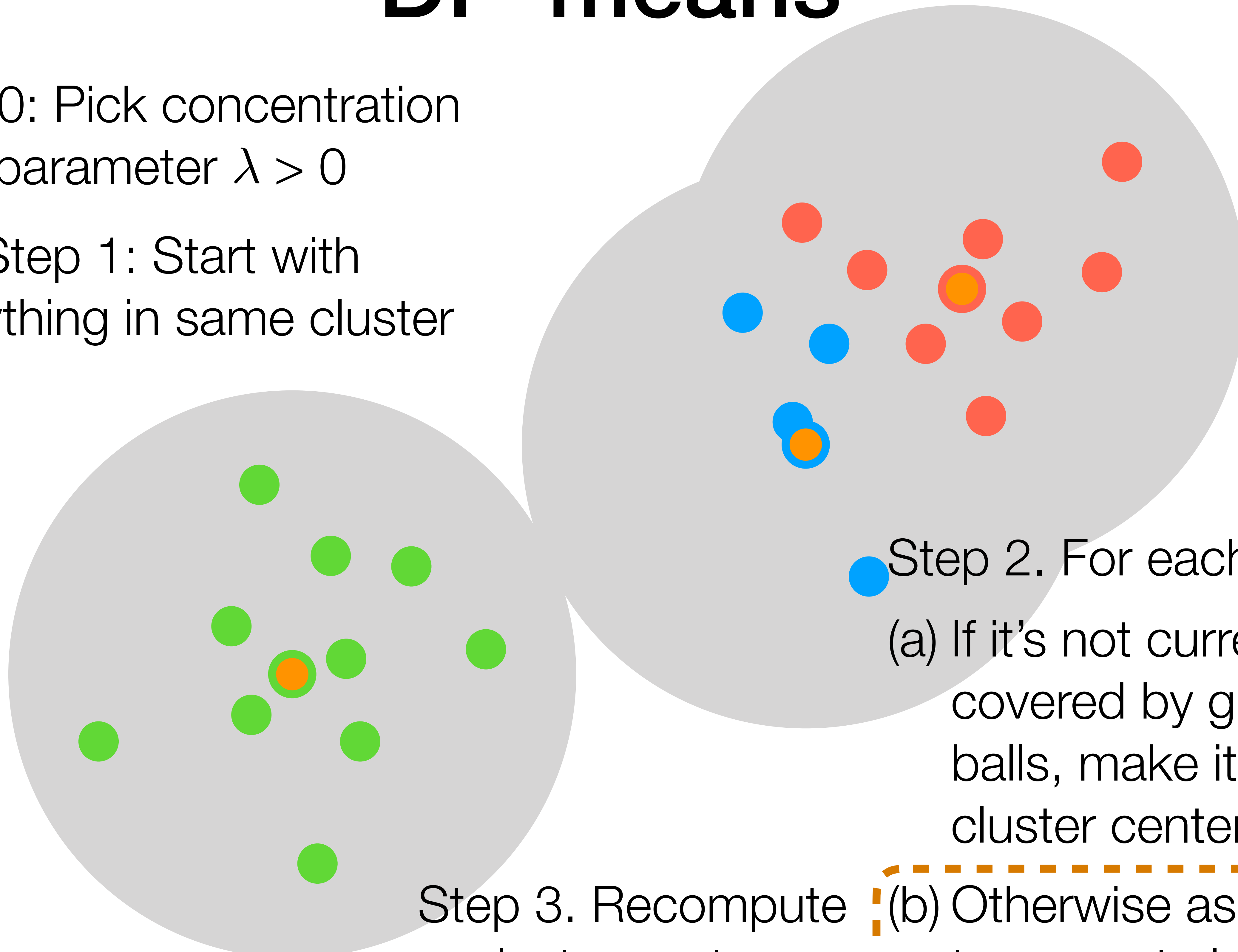
(b) Otherwise assign it to nearest cluster

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

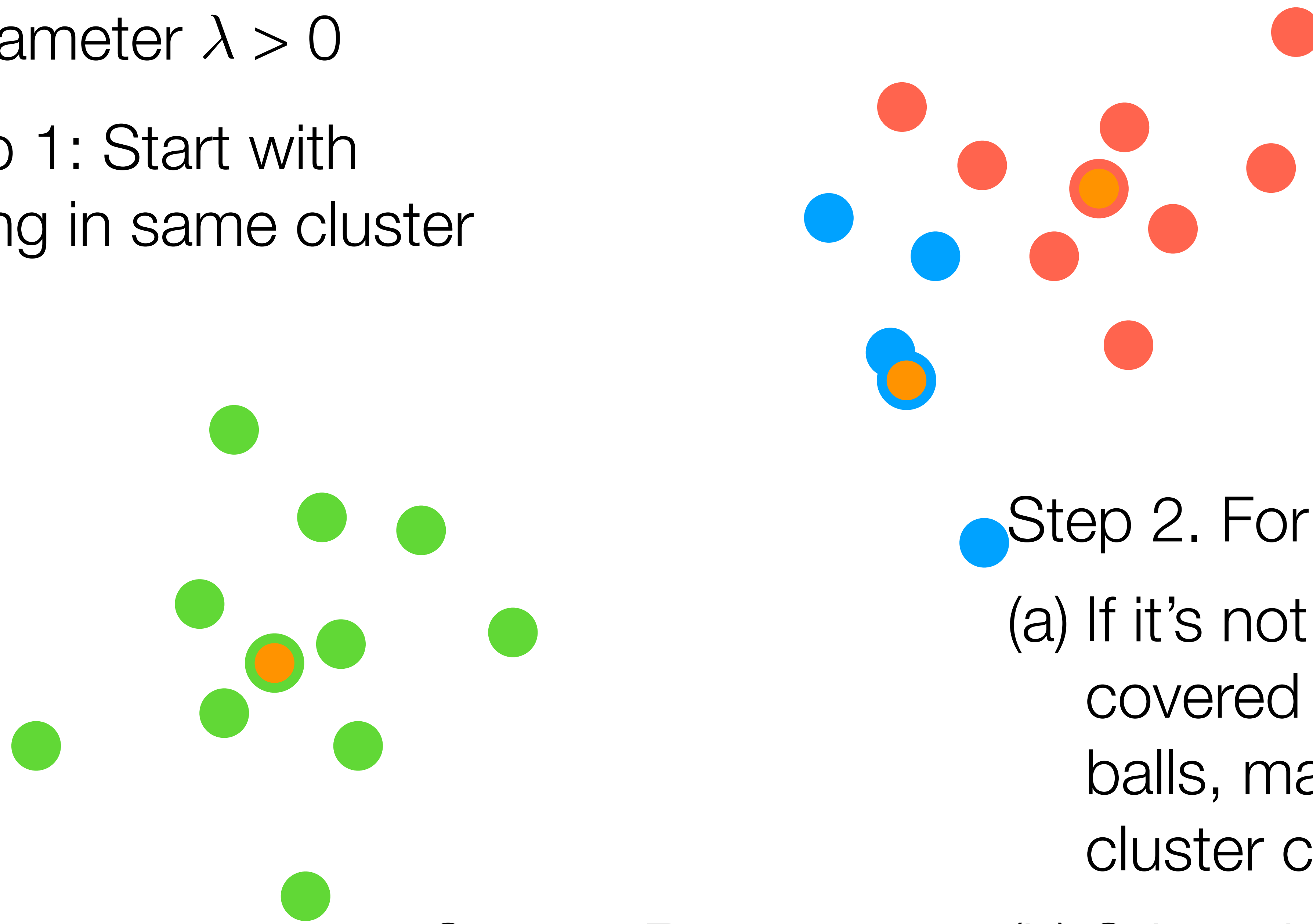
(b) Otherwise assign it to nearest cluster

Step 3. Recompute cluster centers

DP-means

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

Step 1: Start with everything in same cluster



Step 2. For each point:

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

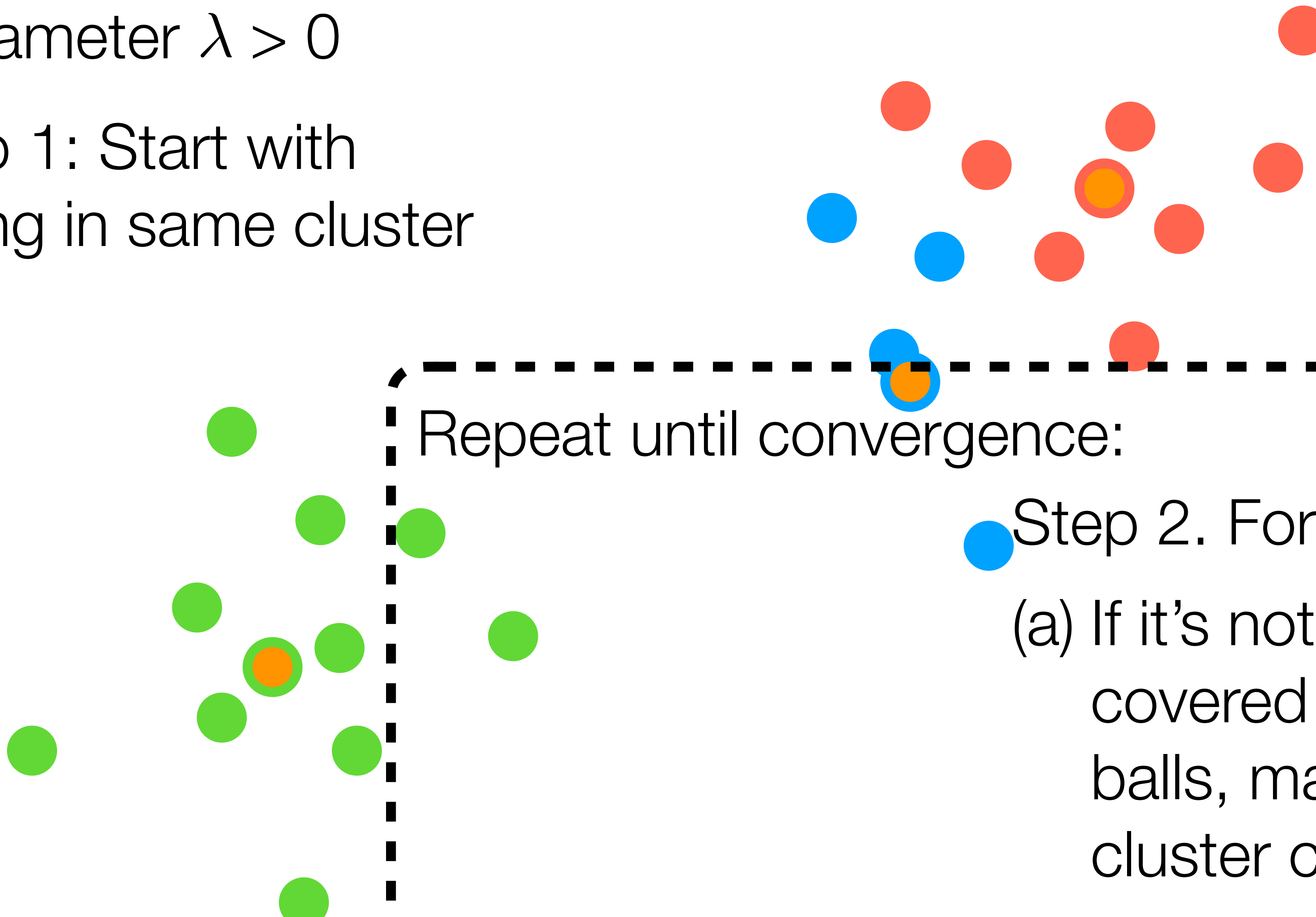
(b) Otherwise assign it to nearest cluster

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:

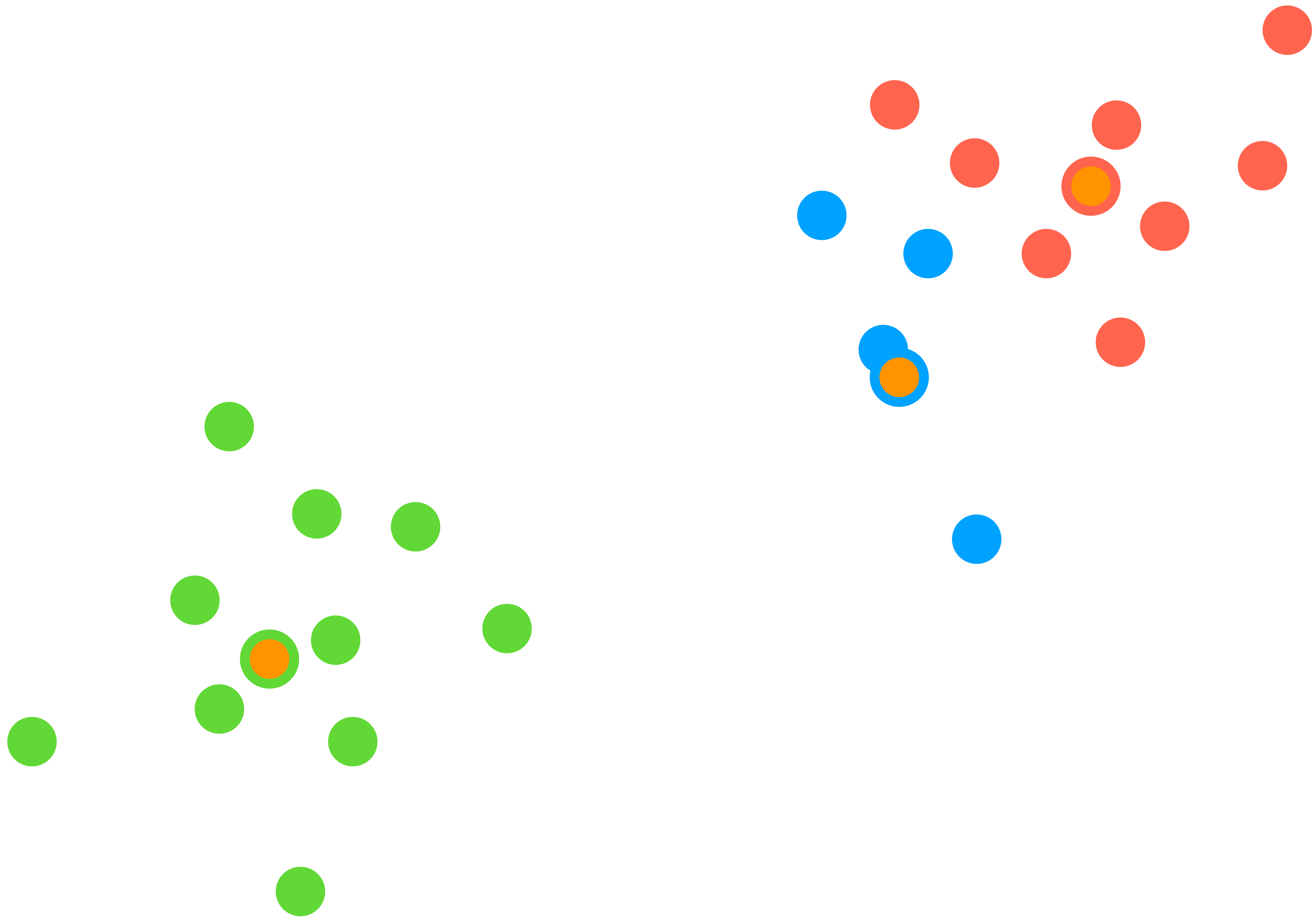
Step 2. For each point:

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

(b) Otherwise assign it to nearest cluster

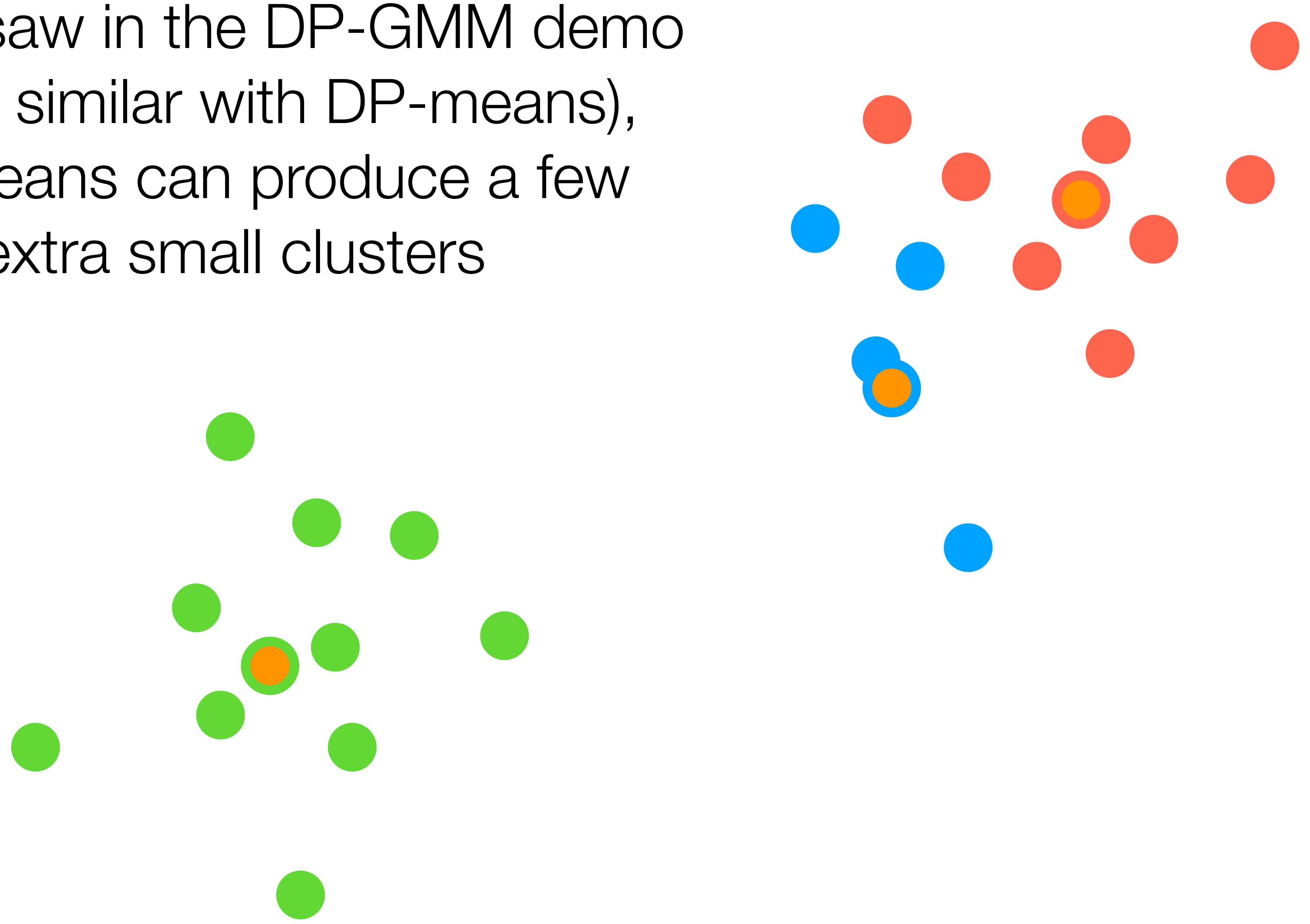
Step 3. Recompute cluster centers

DP-means



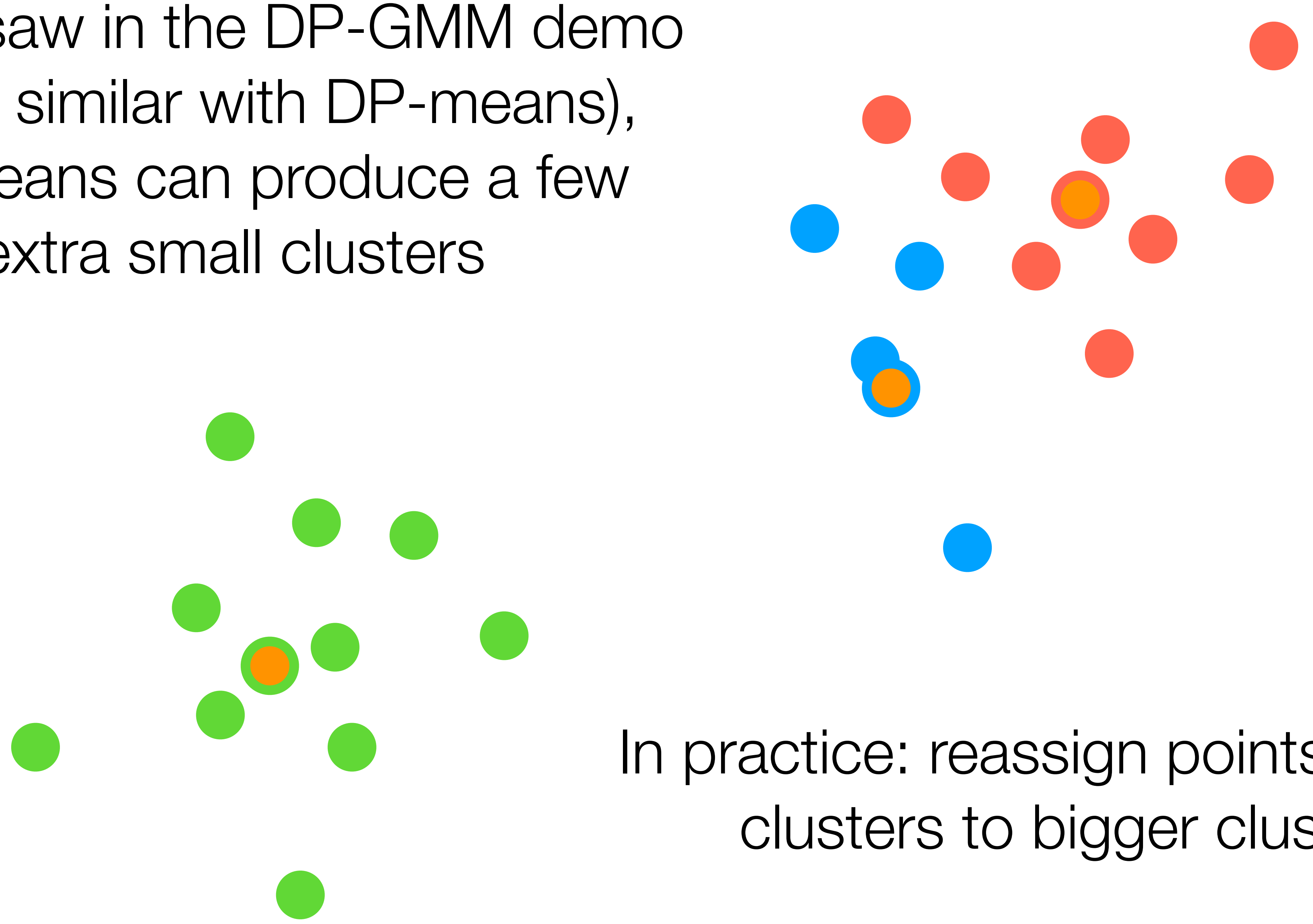
DP-means

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



DP-means

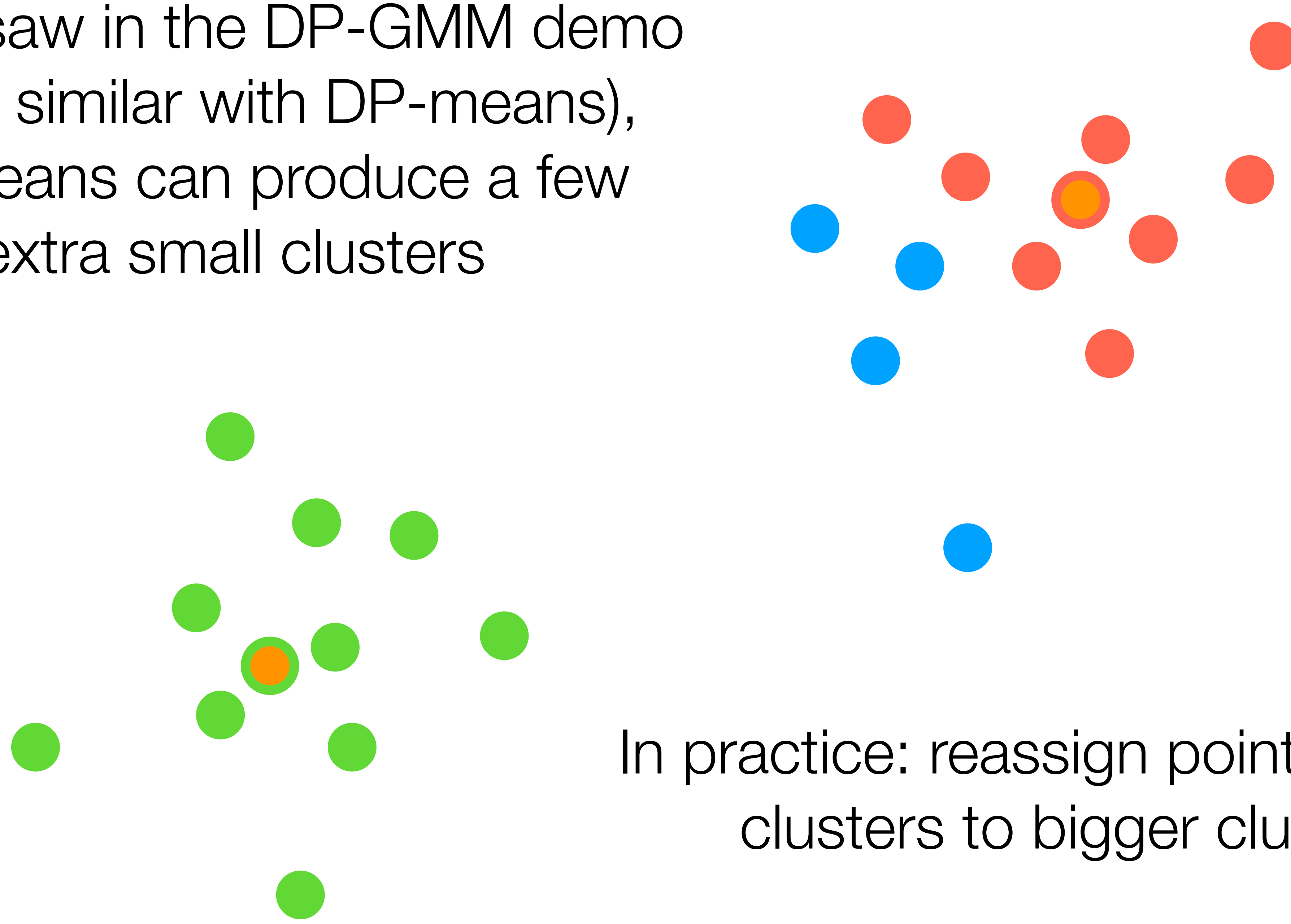
As you saw in the DP-GMM demo (and is similar with DP-means), DP-means can produce a few extra small clusters



In practice: reassign points in small clusters to bigger clusters

DP-means

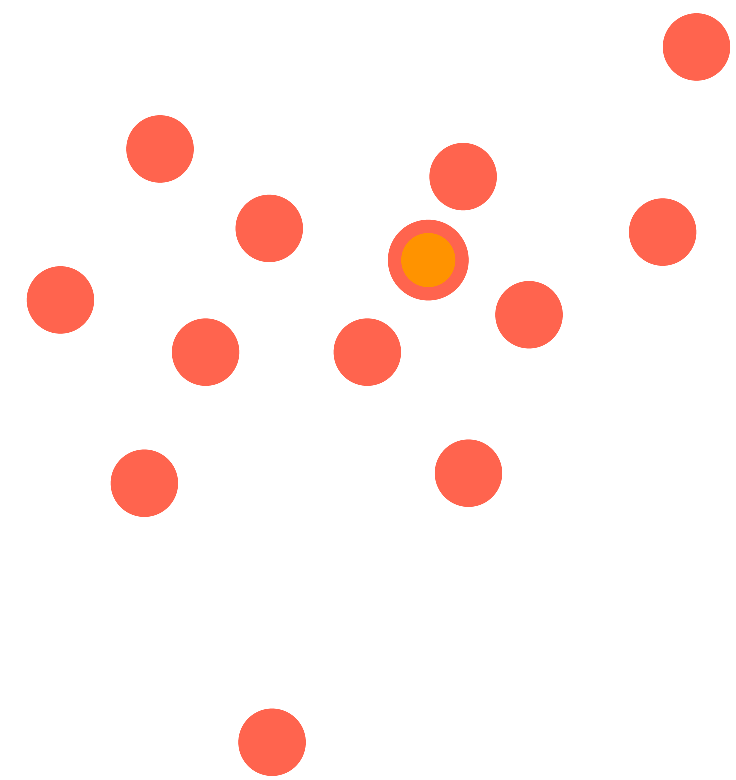
As you saw in the DP-GMM demo (and is similar with DP-means), DP-means can produce a few extra small clusters

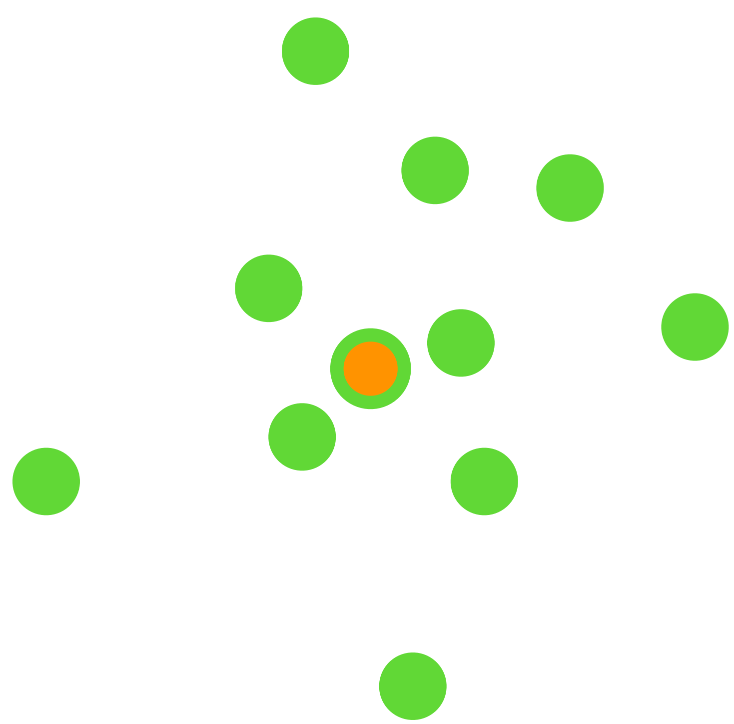


In practice: reassign points in small clusters to bigger clusters

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

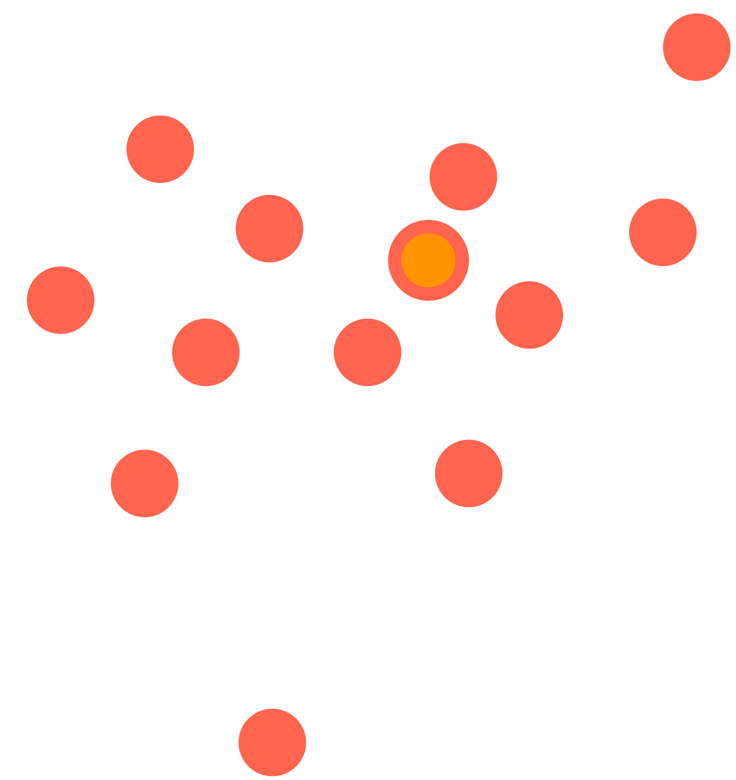


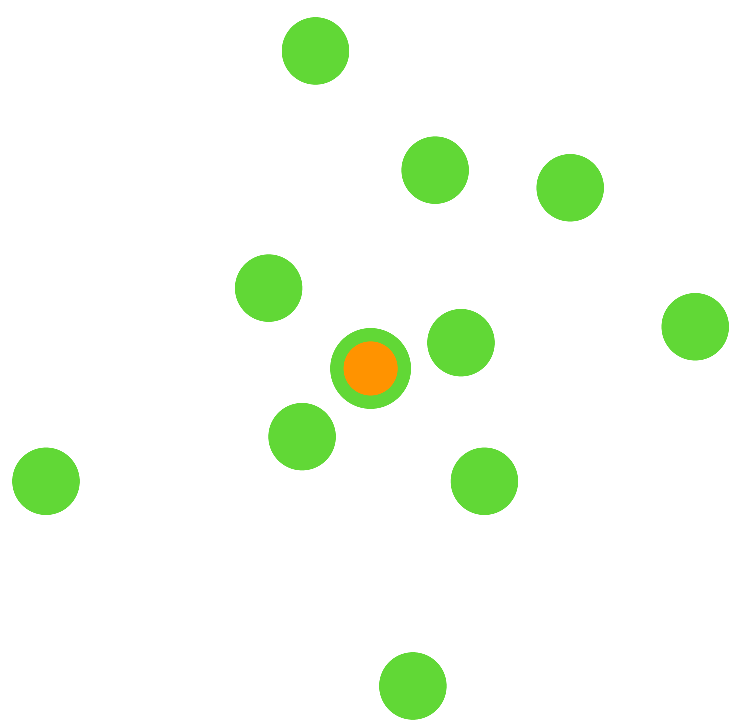


In practice: reassign points in small clusters to bigger clusters

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



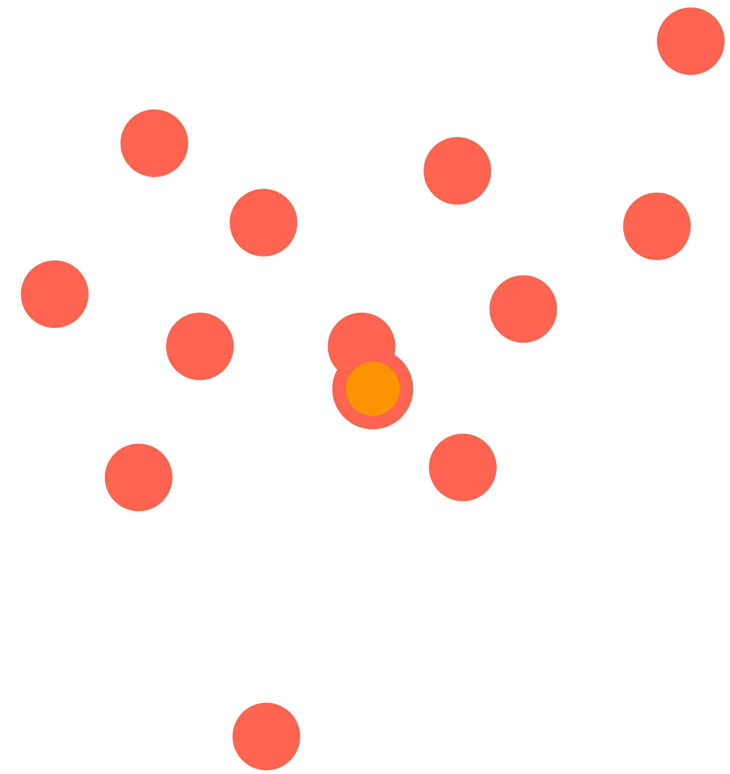


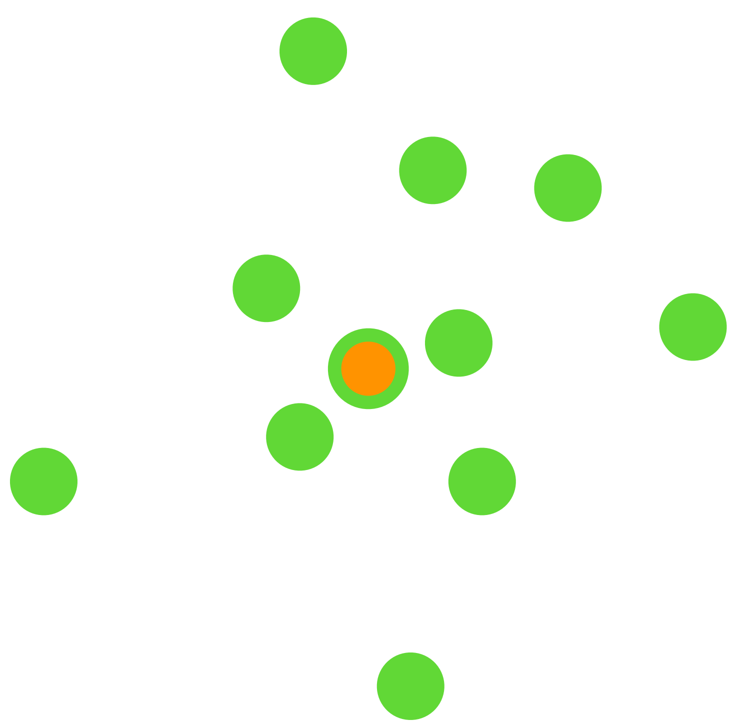
In practice: reassign points in small clusters to bigger clusters

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

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

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*

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

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*

Other Ways for Choosing *k*

- Choose a cost function to compute for different *k*

Other Ways for Choosing *k*

- Choose a cost function to compute for different *k*

- Pick *k* achieving lowest cost

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
- Pick *k* achieving lowest cost

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

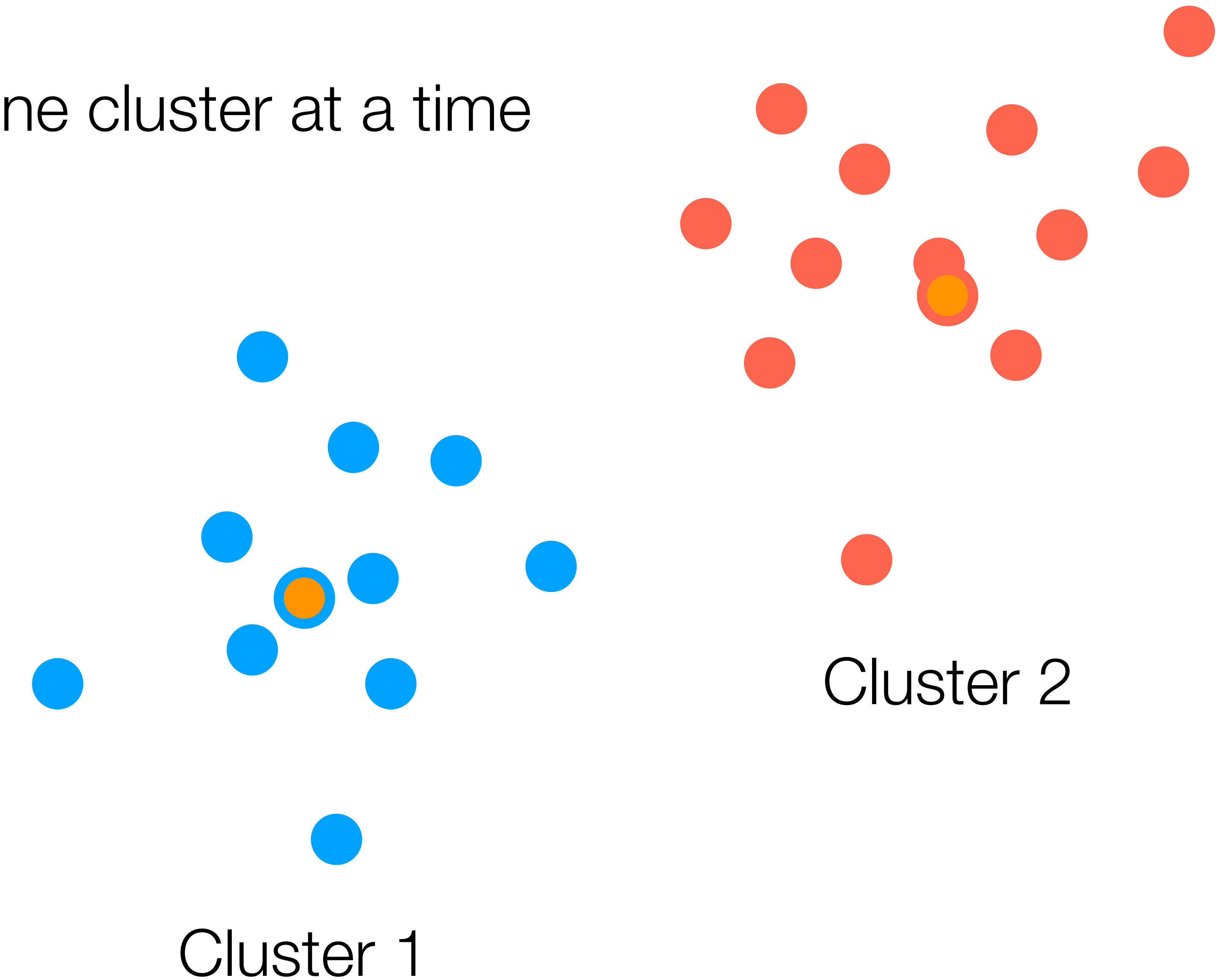
Here’s an example of a cost function you don’t want to use

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

But hey it’s worth a shot

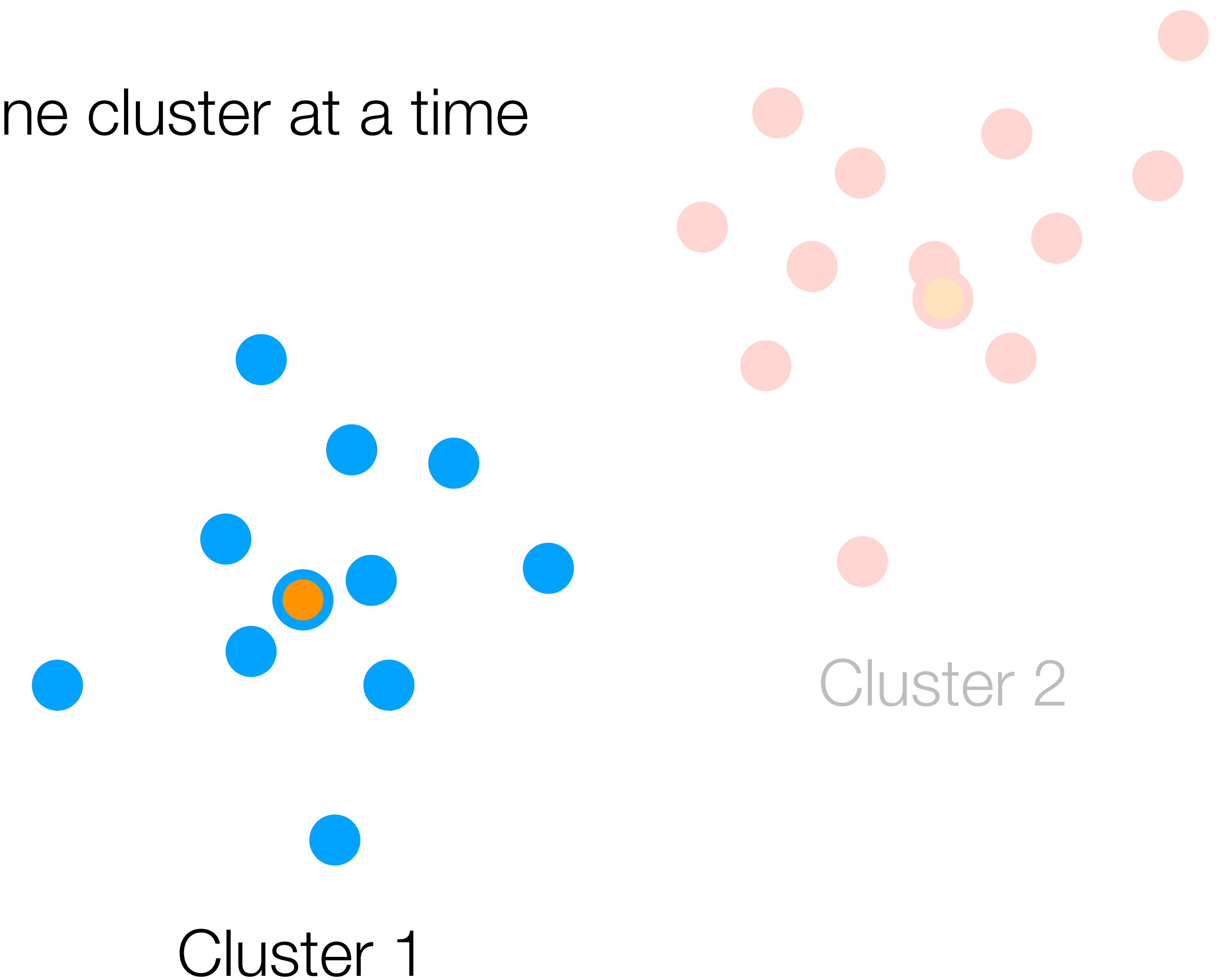
Residual Sum of Squares

Look at one cluster at a time



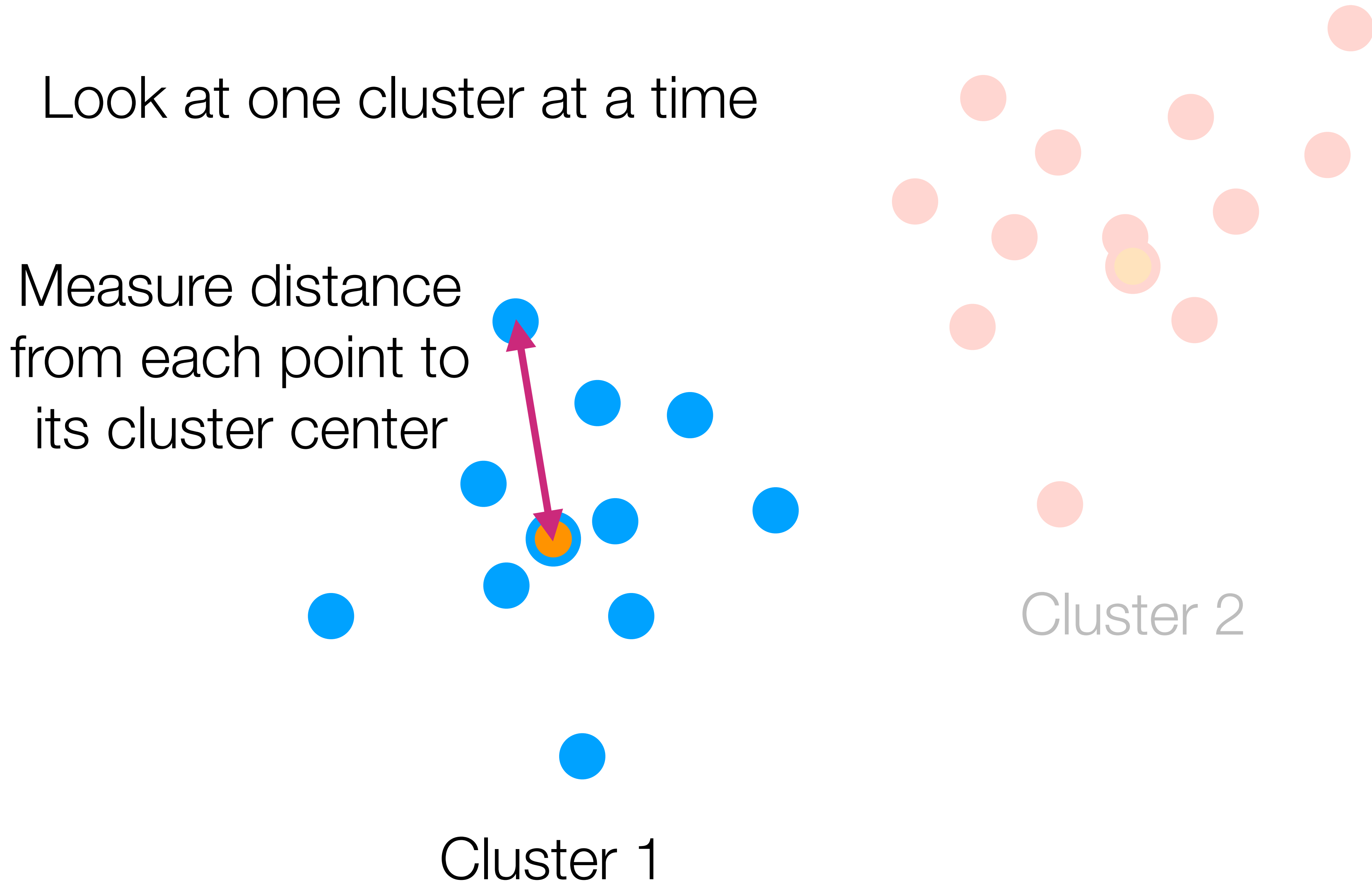
Residual Sum of Squares

Look at one cluster at a time



Residual Sum of Squares

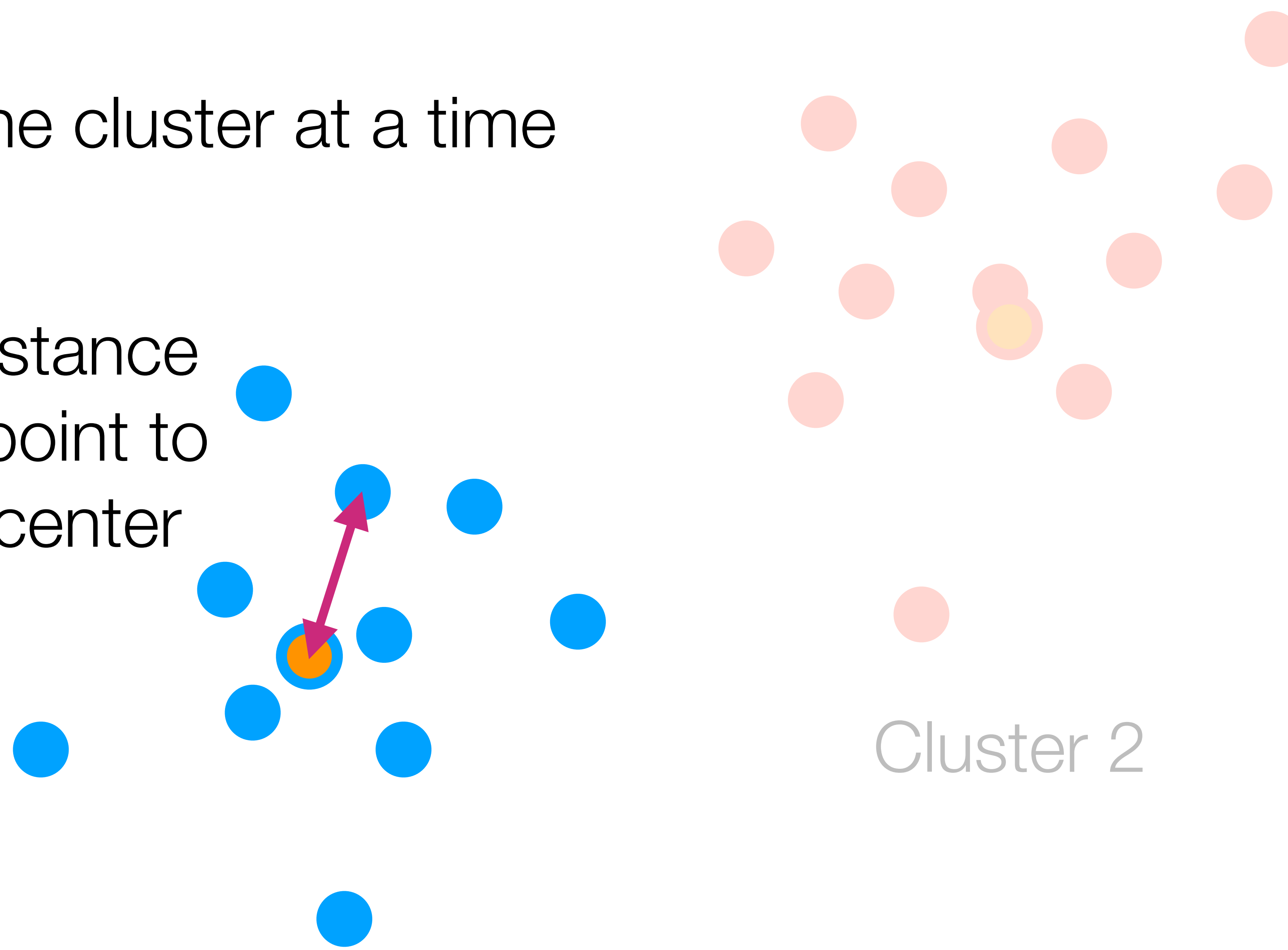
Look at one cluster at a time



Residual Sum of Squares

Look at one cluster at a time

Measure distance from each point to its cluster center



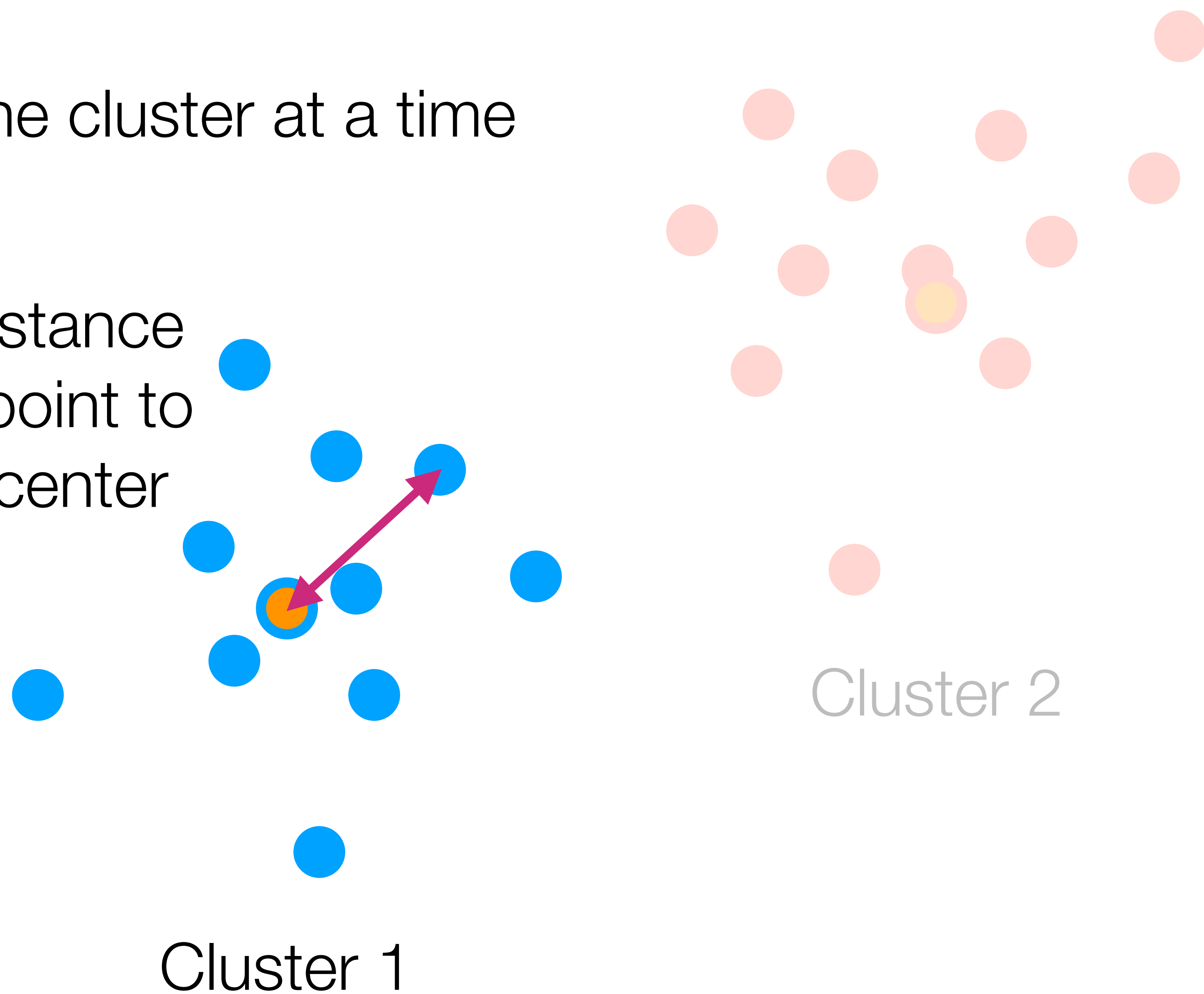
Cluster 2

Cluster 1

Residual Sum of Squares

Look at one cluster at a time

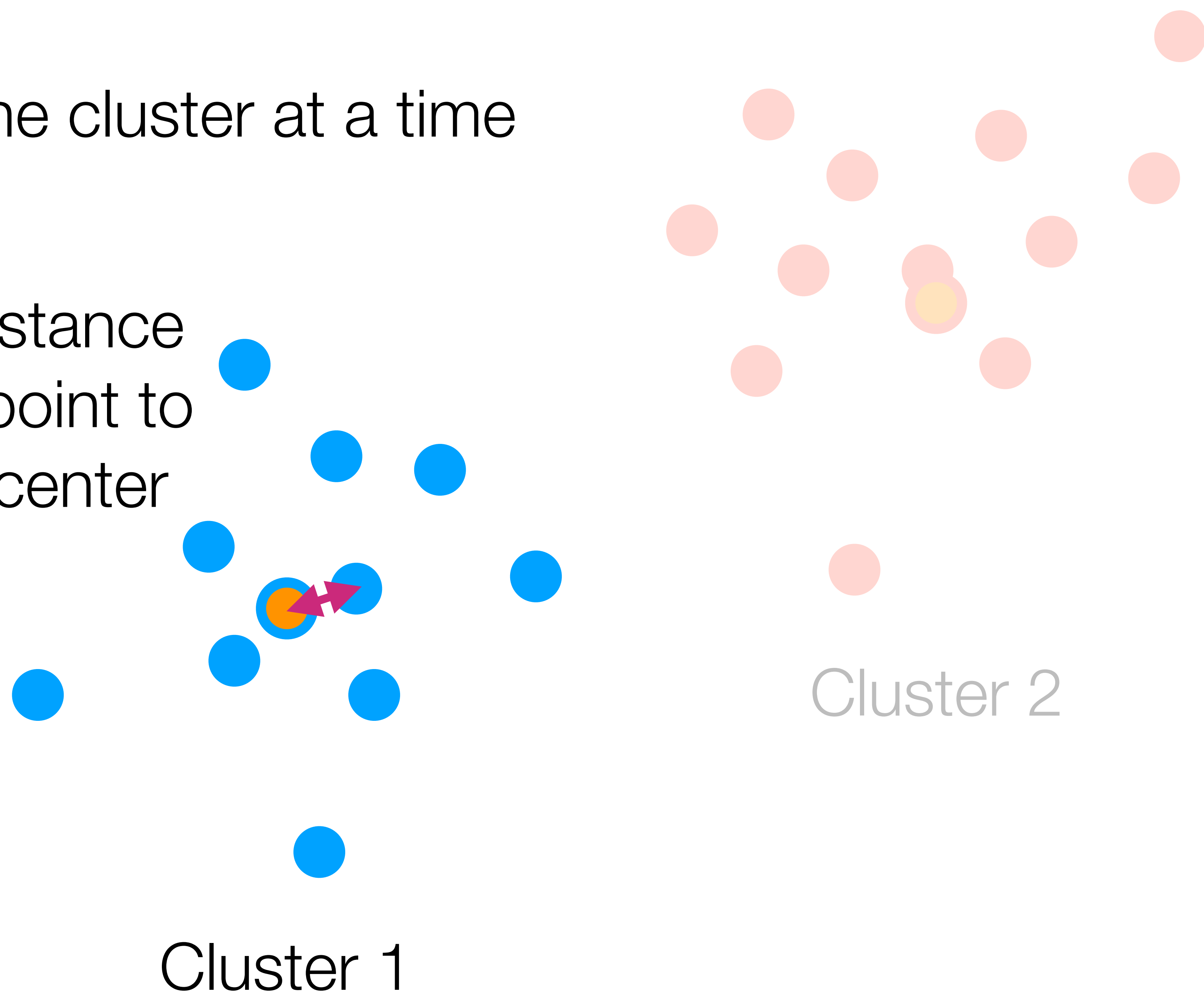
Measure distance from each point to its cluster center



Residual Sum of Squares

Look at one cluster at a time

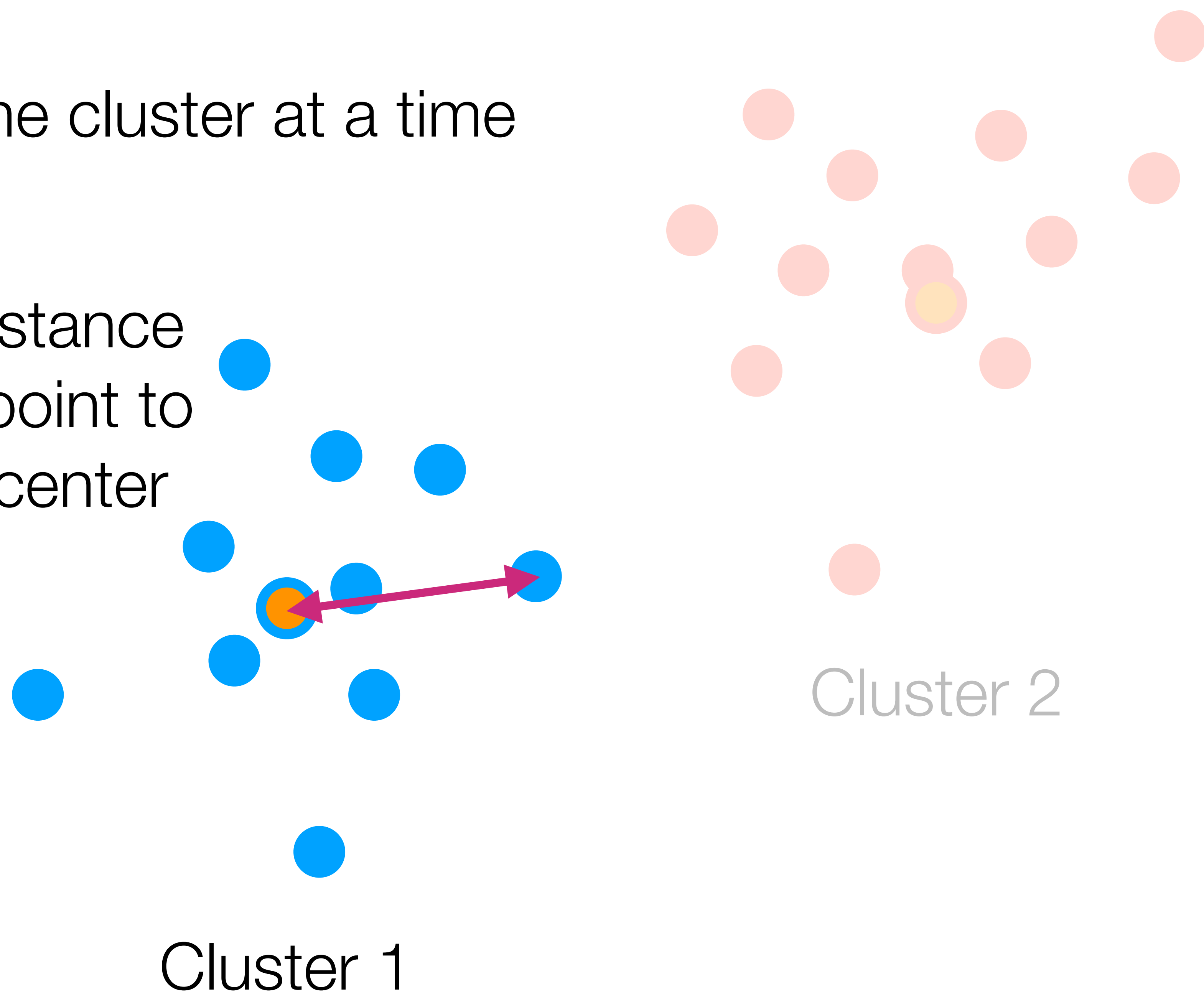
Measure distance from each point to its cluster center



Residual Sum of Squares

Look at one cluster at a time

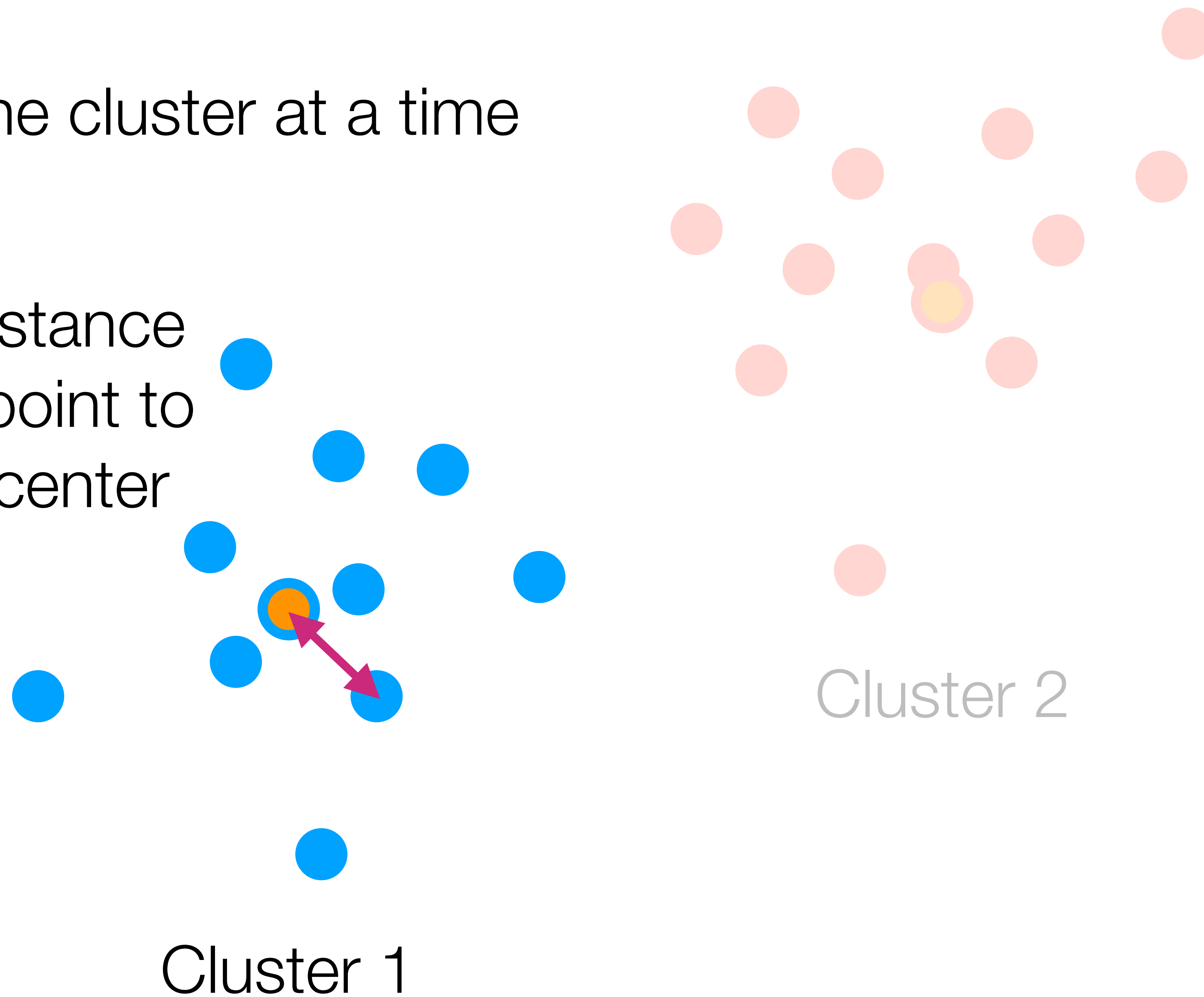
Measure distance from each point to its cluster center



Residual Sum of Squares

Look at one cluster at a time

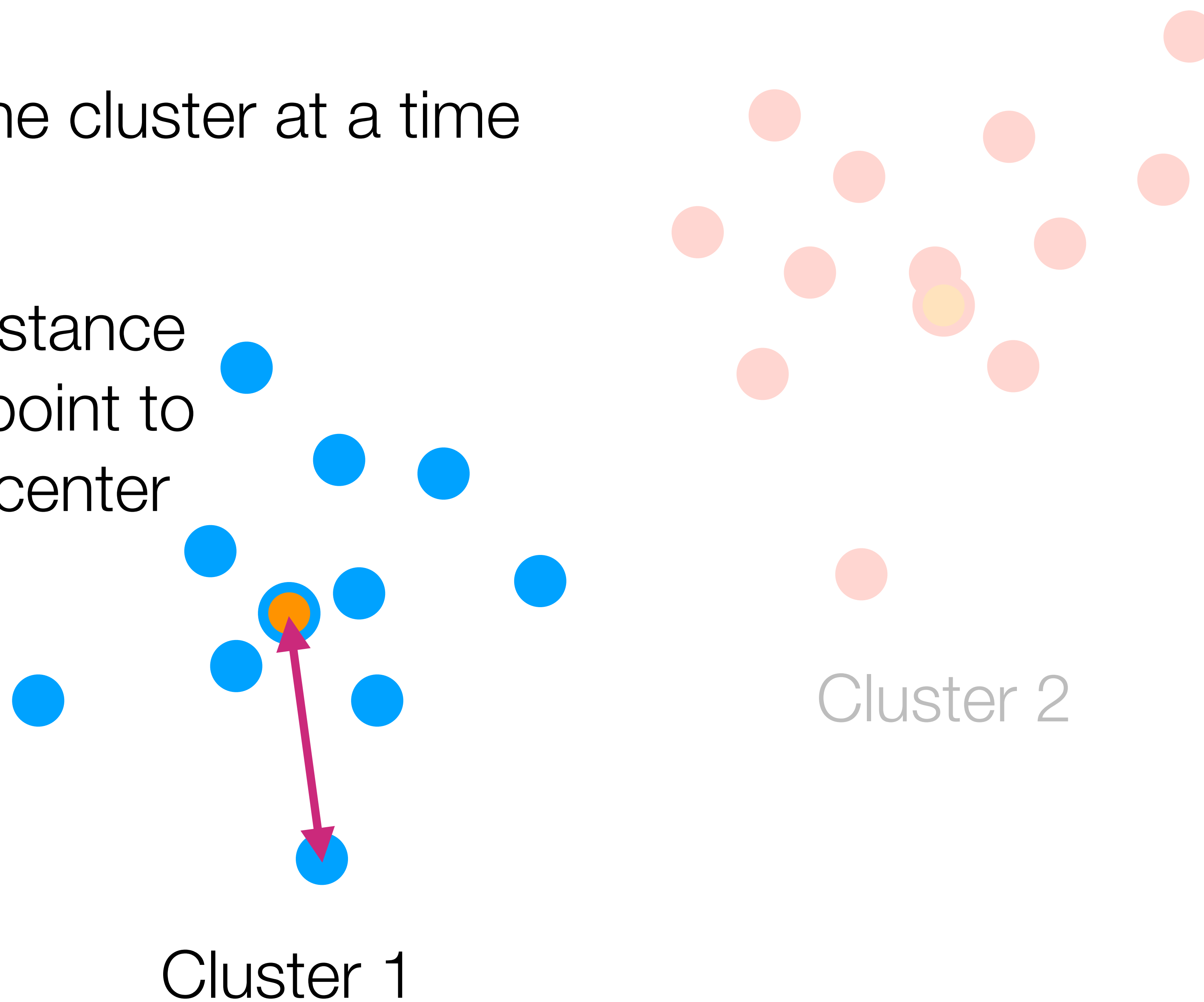
Measure distance from each point to its cluster center



Residual Sum of Squares

Look at one cluster at a time

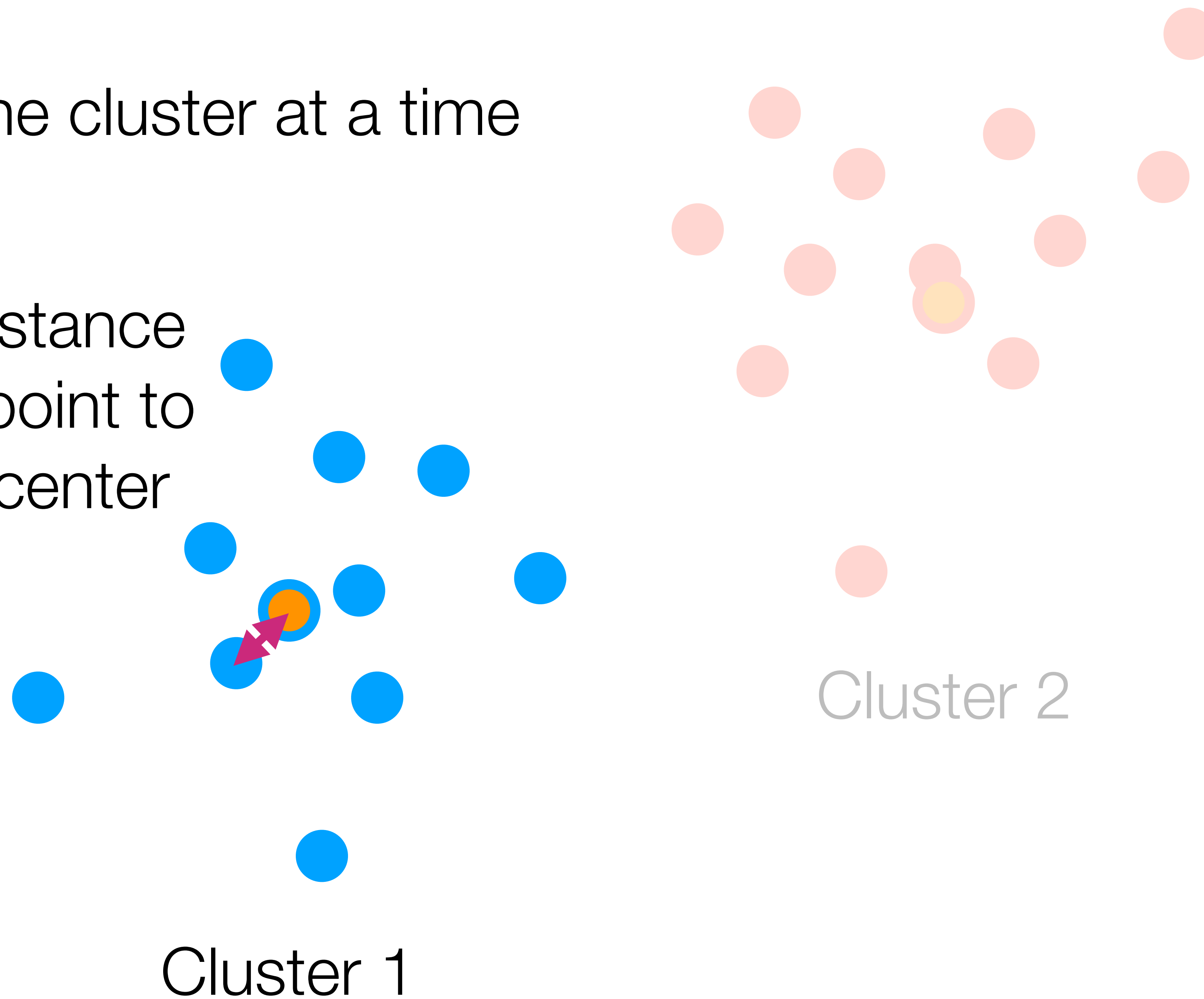
Measure distance
from each point to
its cluster center



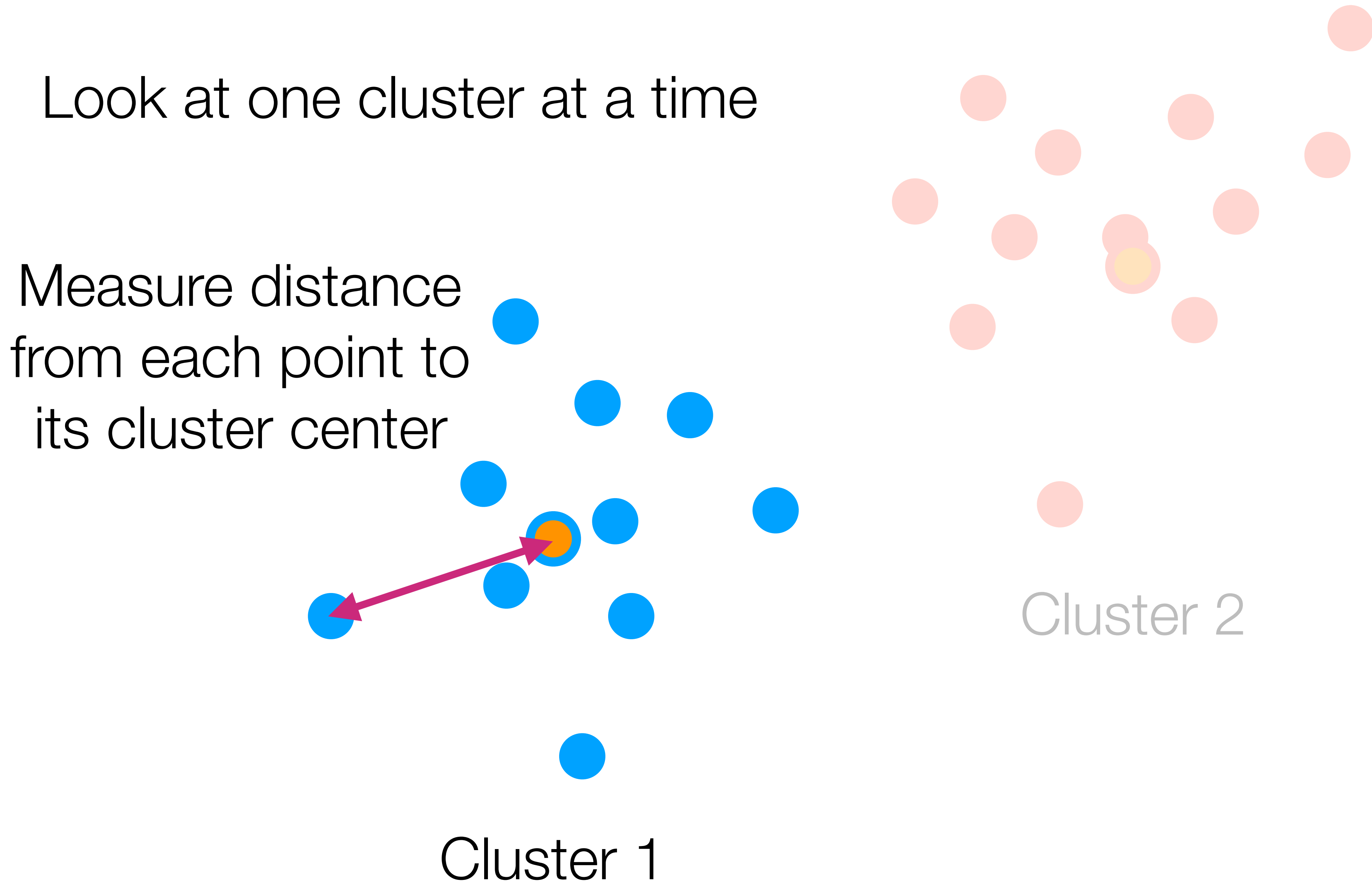
Residual Sum of Squares

Look at one cluster at a time

Measure distance from each point to its cluster center



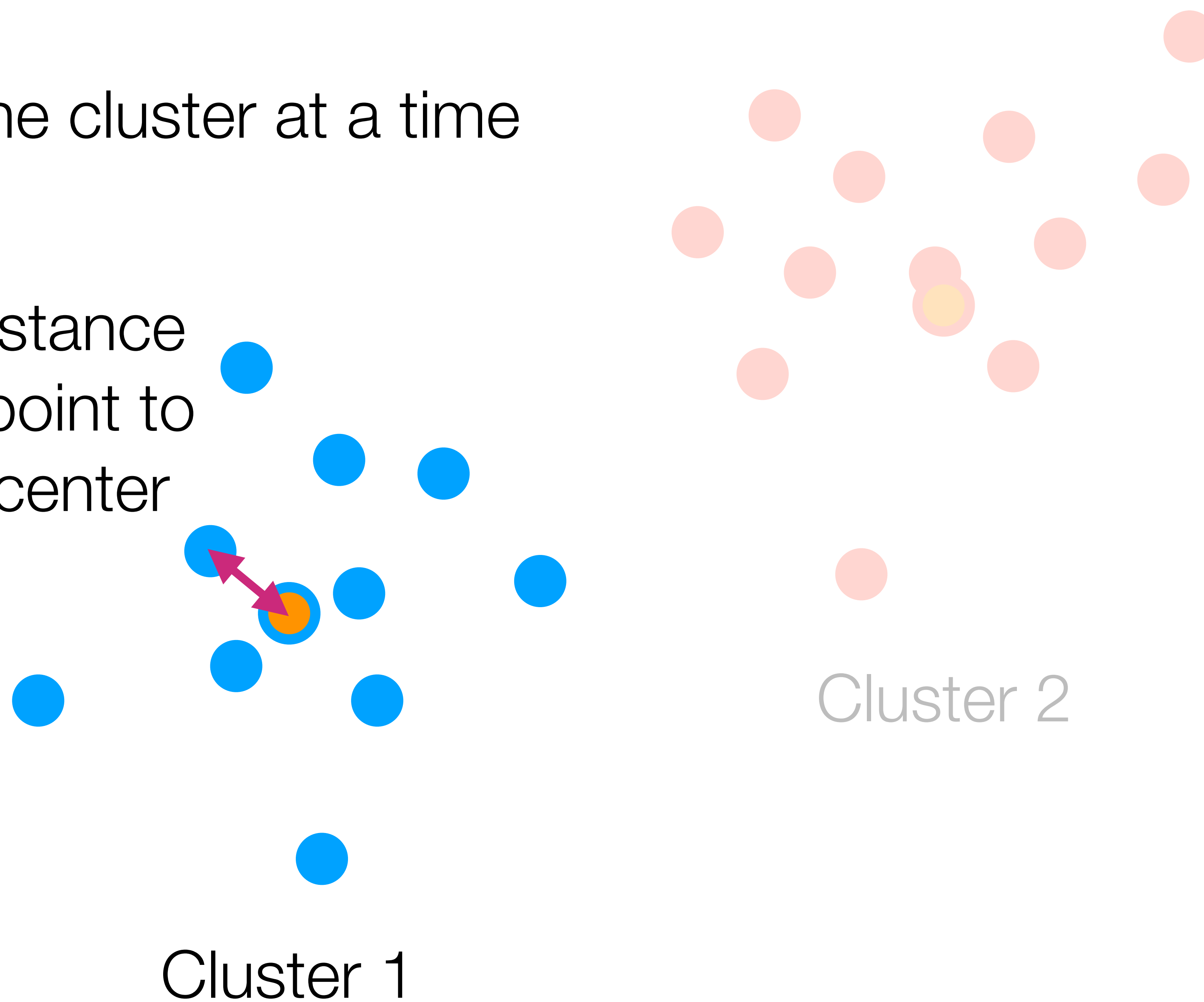
Residual Sum of Squares



Residual Sum of Squares

Look at one cluster at a time

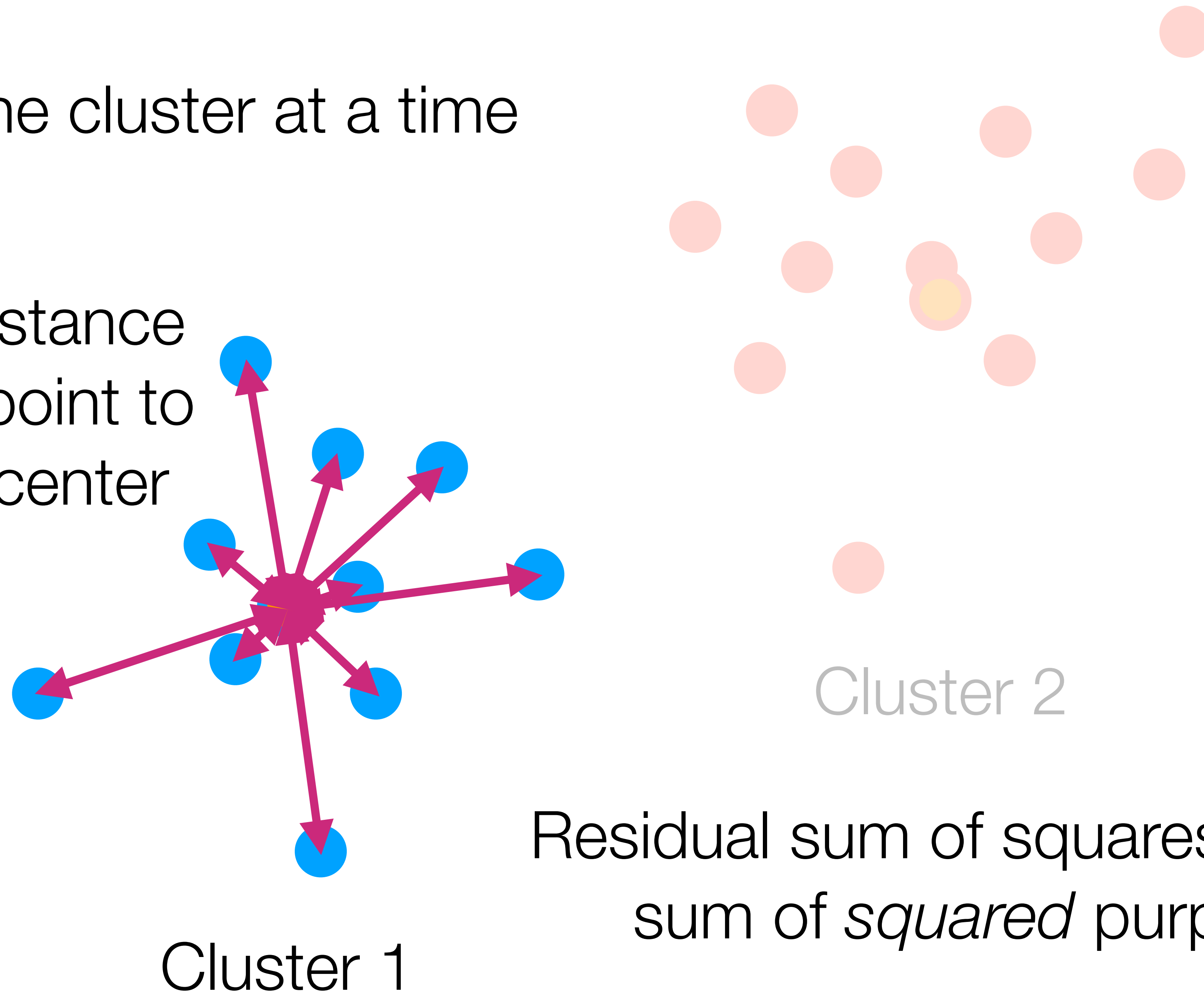
Measure distance
from each point to
its cluster center



Residual Sum of Squares

Look at one cluster at a time

Measure distance from each point to its cluster center

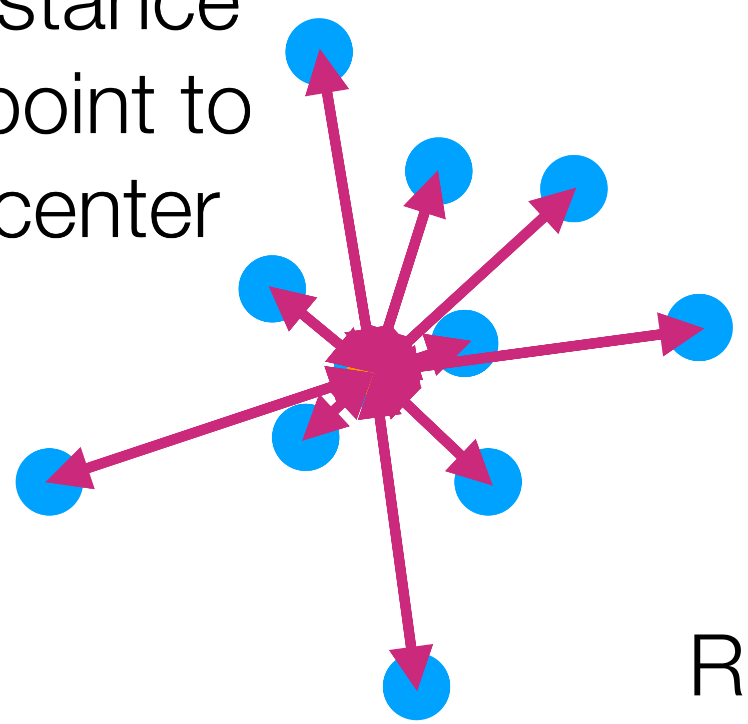


Residual sum of squares for cluster 1: sum of *squared* purple lengths

Residual Sum of Squares

Look at one cluster at a time

Measure distance from each point to its cluster center



Cluster 1

Cluster 2

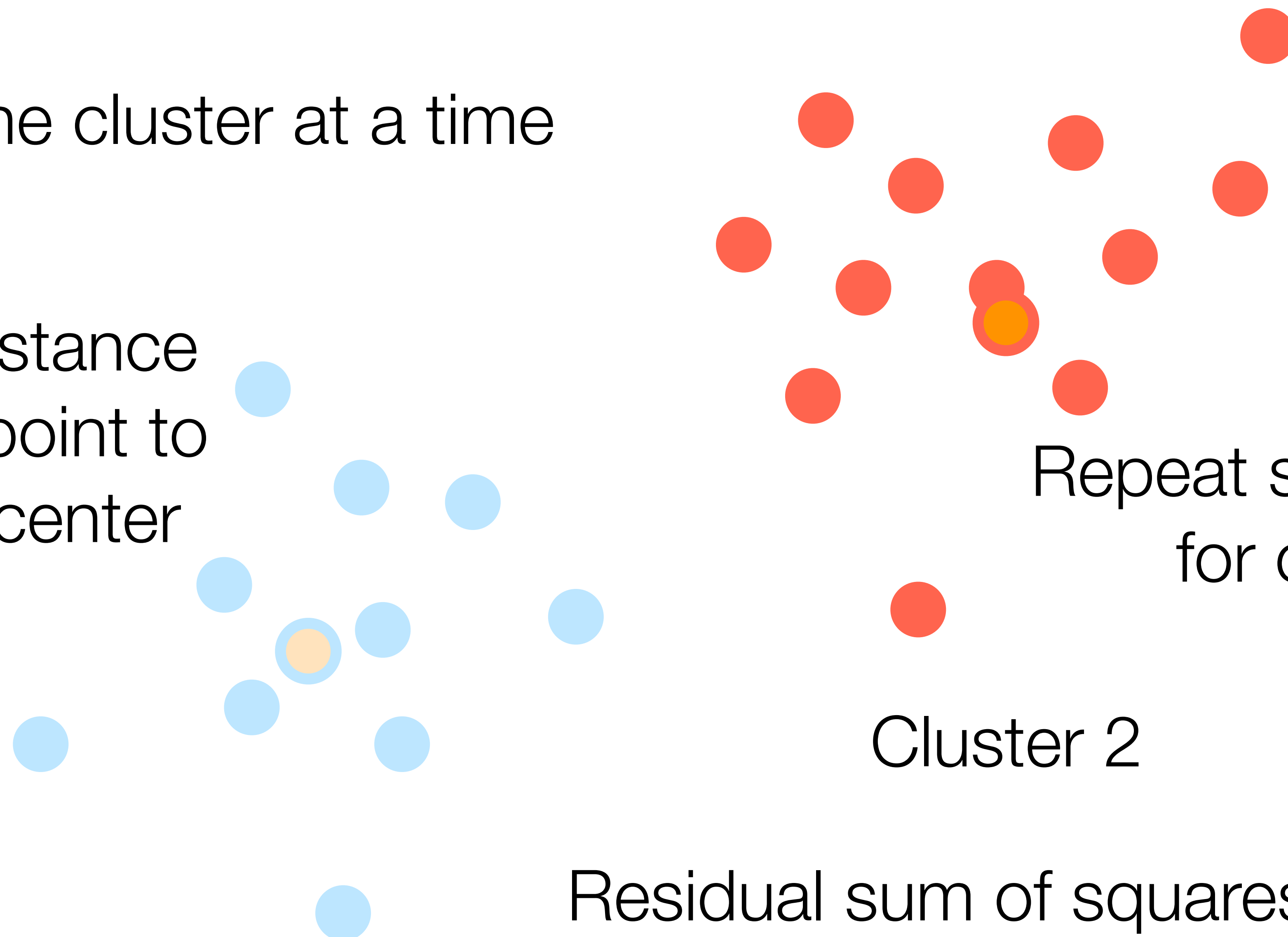
Residual sum of squares for cluster 1:

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

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

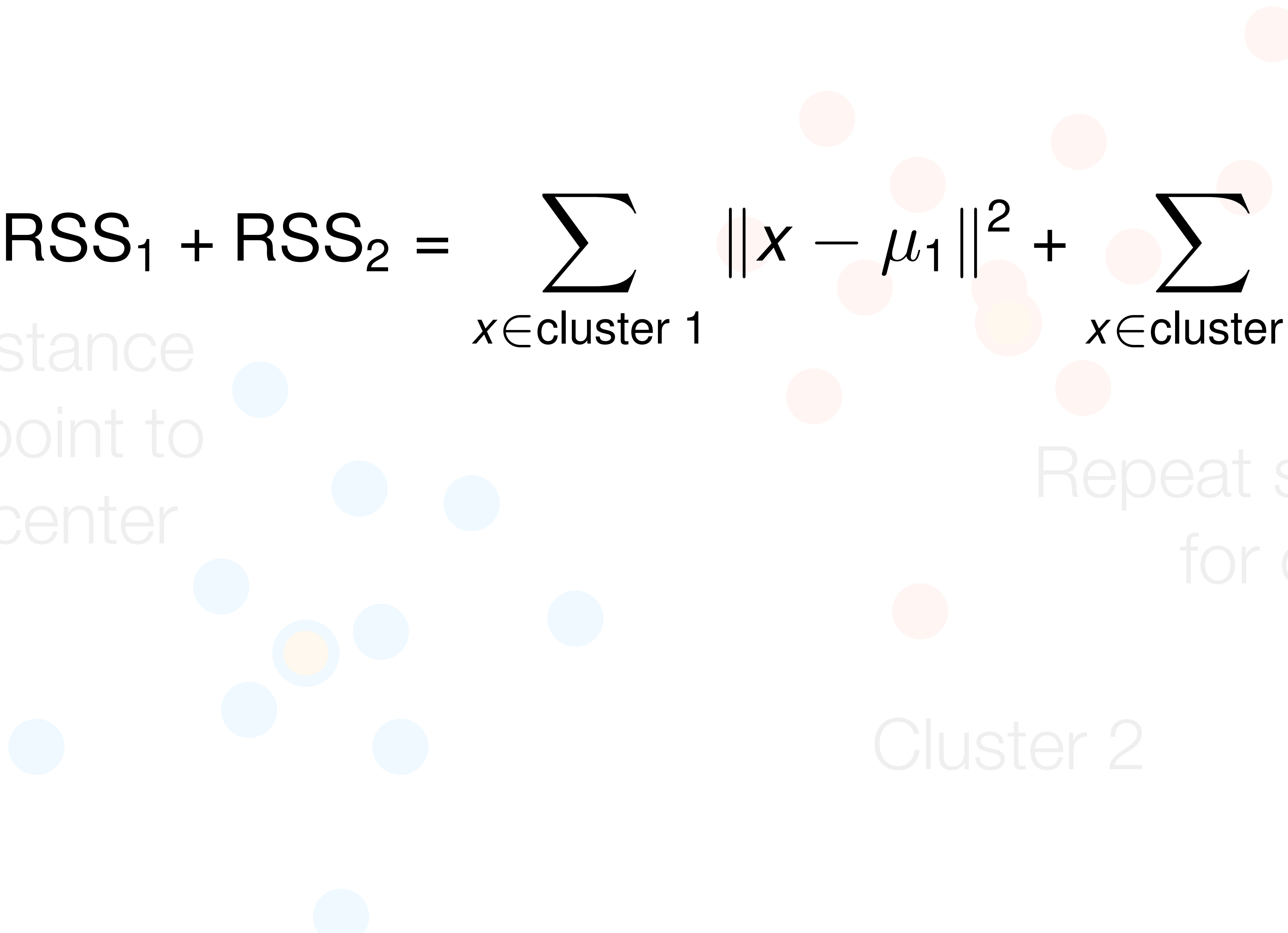
Residual sum of squares for cluster 2:

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

Residual Sum of Squares

Residual Sum of Squares

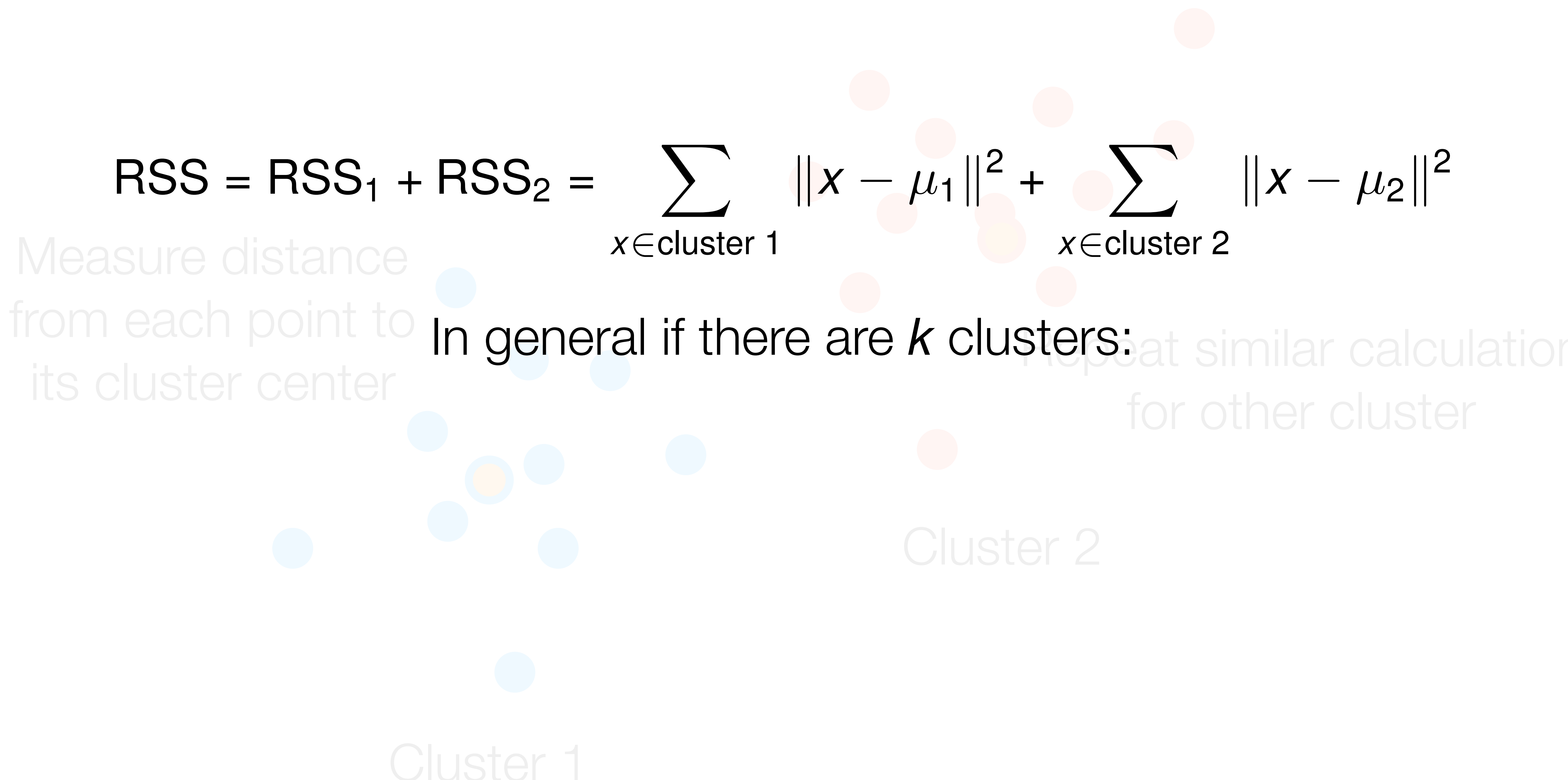
$$\text{RSS} = \text{RSS}_1 + \text{RSS}_2 = \sum_{x \in \text{cluster 1}} \|x - \mu_1\|^2 + \sum_{x \in \text{cluster 2}} \|x - \mu_2\|^2$$



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:



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:

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

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)

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?

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*

A Good Way to Choose *k*

RSS measures *within-cluster variation*

A Good Way to Choose *k*

RSS measures *within-cluster variation*

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

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*

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*

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

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

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

A good score function to use for choosing *k*:

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

A good score function to use for choosing *k*:

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

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

A good score function to use for choosing *k*:

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

n = total # 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

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*)

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

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)

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)

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)

Another good way is called the **gap statistic** [Tibshirani et al 2001]