CS630 Graduate Algorithms

September 26, 2024

by Dora Erdos and Jeffrey Considine

- Center Selection

10/10 midterm

Representative trial participants

Select a group of participants for a new vaccine trial! Each person has some health characteristics, e.g. age, weight, medications, etc. based on this we have access to the distance (dissimilarity) $d(p_i,p_j)$ between every pair of people.

problem:

Select a minimal subset of people for the trial, such that each person not in the trial is within distance T of one of the participants.

use Set Cover:

- create the universe → things to cover

  U = people

- Subsets of U:

  person p_i : S_i = people within distance T of p_i

Representative trial participants

Select a group of participants for a new vaccine trial! Each person has some health characteristics, e.g. age, weight, medications, etc. based on this we have access to the distance (dissimilarity) $d(p_i,p_j)$ between every pair of people.

problem:

Select a minimal subset of people for the trial, such that each person not in the trial is within distance T of one of the participants.

alternative problem:

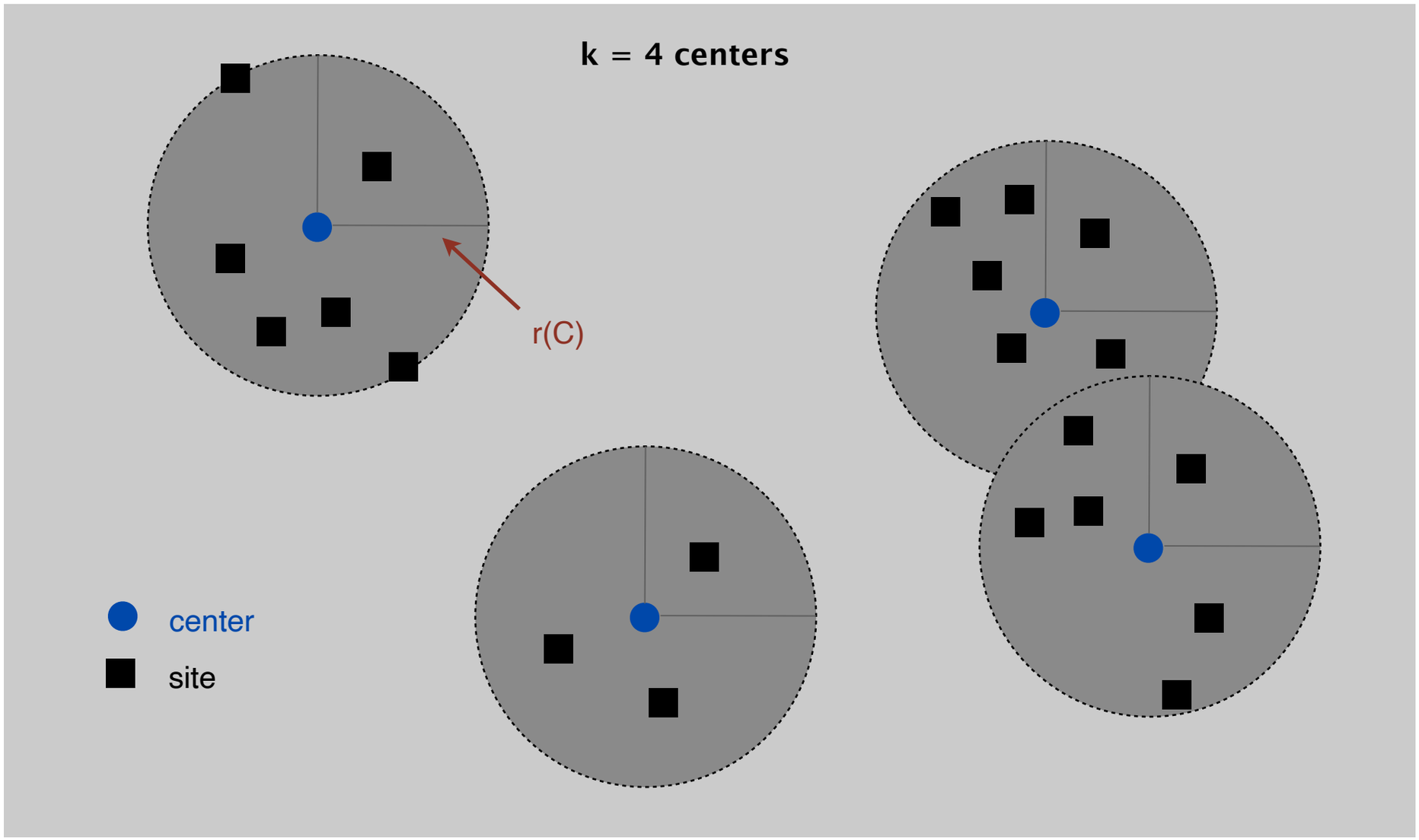
Select k participants, such that the largest distance between a person and a participant is as low as possible.

→ sensitive to outliers
→ often we prune the outliers

Center selection problem

Input. Set of n sites $s_1, \ldots, s_n$ and an integer $k > 0$.

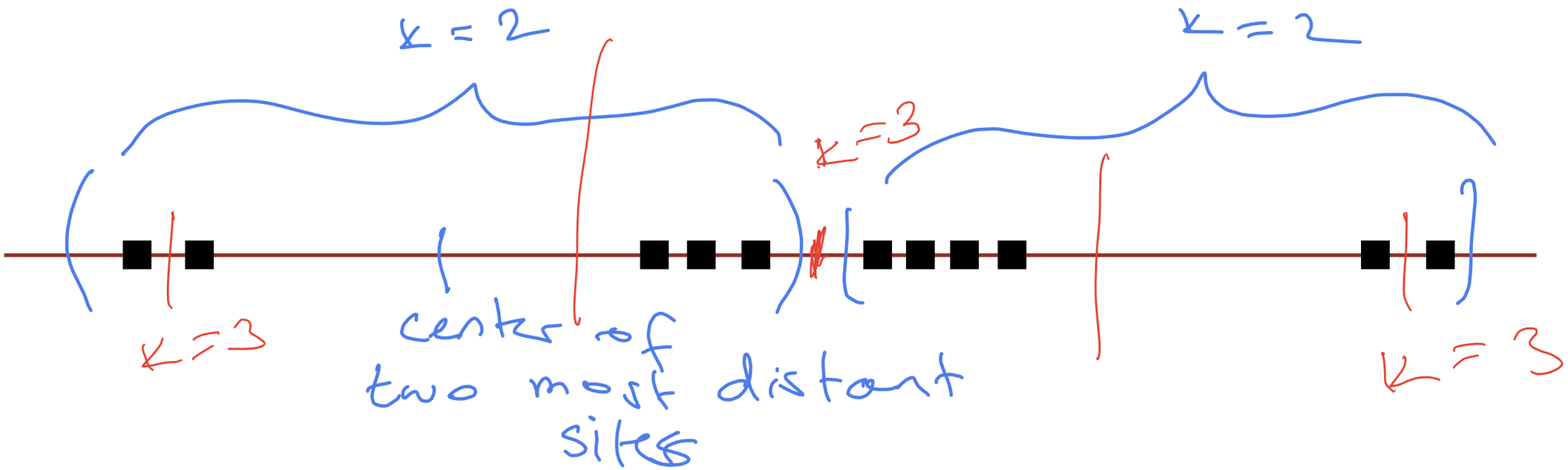
Center selection problem. Select set of k centers C so that maximum distance $r(C)$ from a site to nearest center is minimized.



1D-center selection problem

Input. Set of n sites $s_1, \ldots, s_n$ on a *line* and an integer $k > 0$.

Center selection problem. Select set of k centers C so that maximum distance $r(C)$ from a site to nearest center is minimized.



Where would you put the centers if k=2, k=3, k=4?

the solution for k=2 is not a subset for k=3

canonical DP problem O(nk)

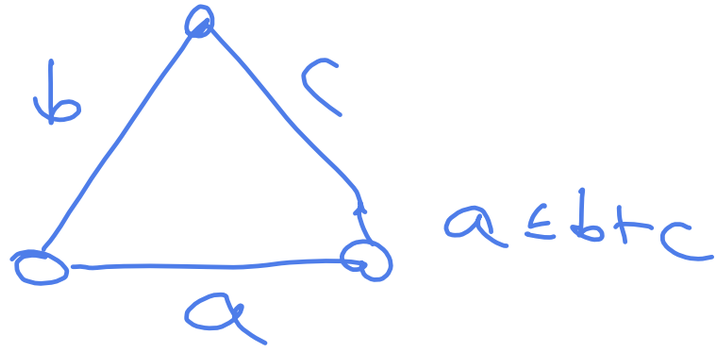
Center selection problem

Input. Set of n sites $s_1, \ldots, s_n$ and an integer $k > 0$.

Center selection problem. Select set of k centers C so that maximum distance $r(C)$ from a site to nearest center is minimized.

Distance function properties.

- $dist(x, x) = 0$ [identity]
- $dist(x, y) = dist(y, x)$ [symmetry]
- $dist(x, y) \leq dist(x, z) + dist(z, y)$ [triangle inequality]



examples: Euclidean distance

Manhattan distance : input binary vectors
dist = how many coordinates they differ

L_p space $\|x, y\|_p = \sqrt[p]{\sum (x_i - y_i)^p}$

Kendall tau distance → compare orderings

Center selection problem

Input. Set of n sites $s_1, \ldots, s_n$ and an integer $k > 0$.

Center selection problem. Select set of k centers C so that maximum distance $r(C)$ from a site to nearest center is minimized.

Notation.

- $dist(x, y)$ = distance between sites x and y.
- $dist(s_i, C) = \min_{c_j \in C} dis(s_i, c_j) =$ distance from s_i to closest center.
- $r(C) = \max_{i=1 \ldots n} dist(s_i, C) =$ smallest covering radius.

worst distance from nearest center

Goal. Find set of centers C that minimizes $r(C)$, subject to $|C| = k$.

Center selection example

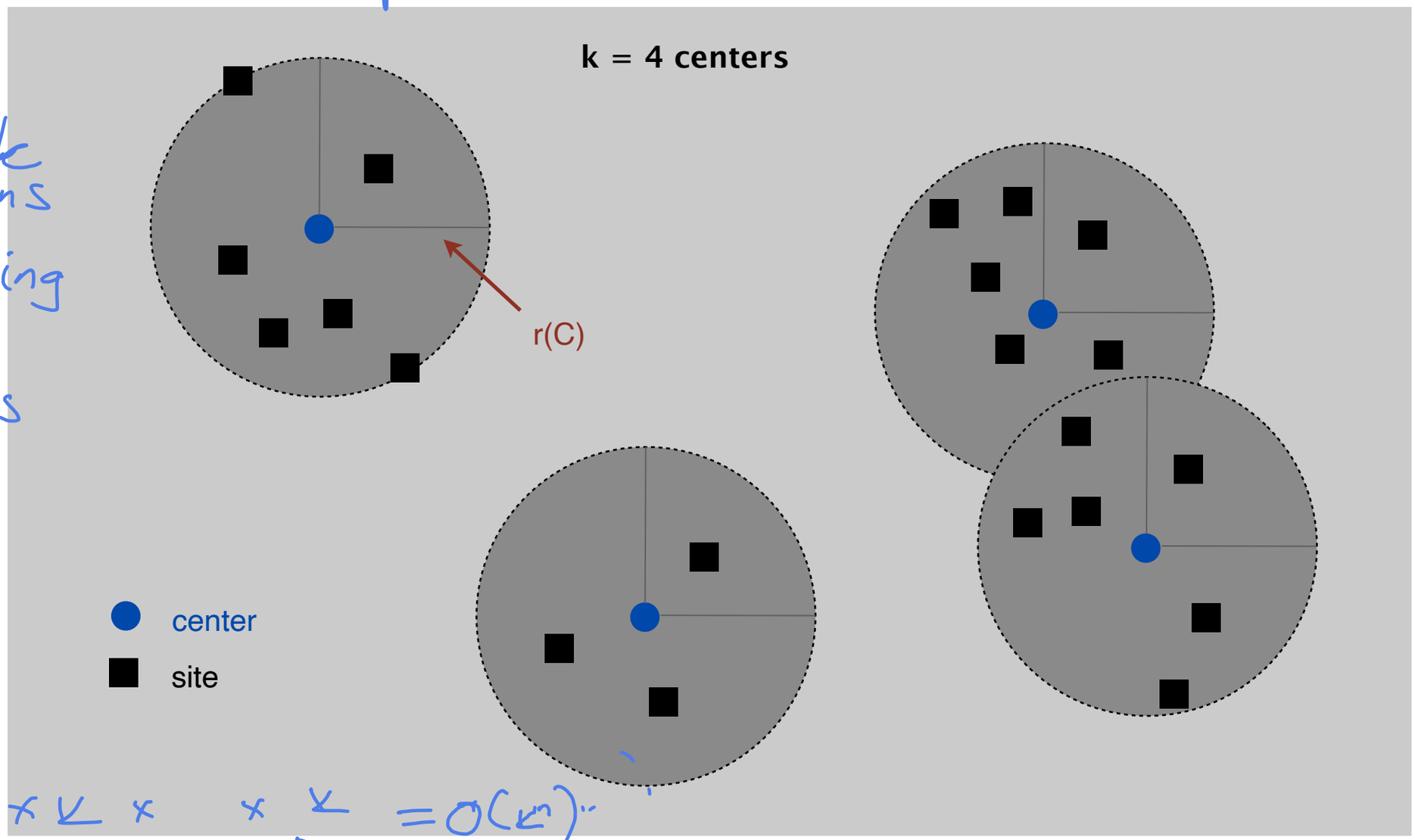
Ex: each site is a point in the plane, a center can be any point in the plane, $dist(x, y)$ = Euclidean distance.

key! → if we know which sites are in the same cluster → for each cluster we can compute the opt center

infinite many possible center locations!

Brute force = try all possible combinations

↓ all possible combinations of assigning sites to k clusters



$k \times k \times k \times \cdots \times k = O(k^n)$

s_1 s_2 s_3 ... s_n

Greedy algorithm

Select centers one at a time:

c_1 : pick the best location considering all sites

c_2 : pick location to reduce radius (the current) as much as possible

c_3 : —"—

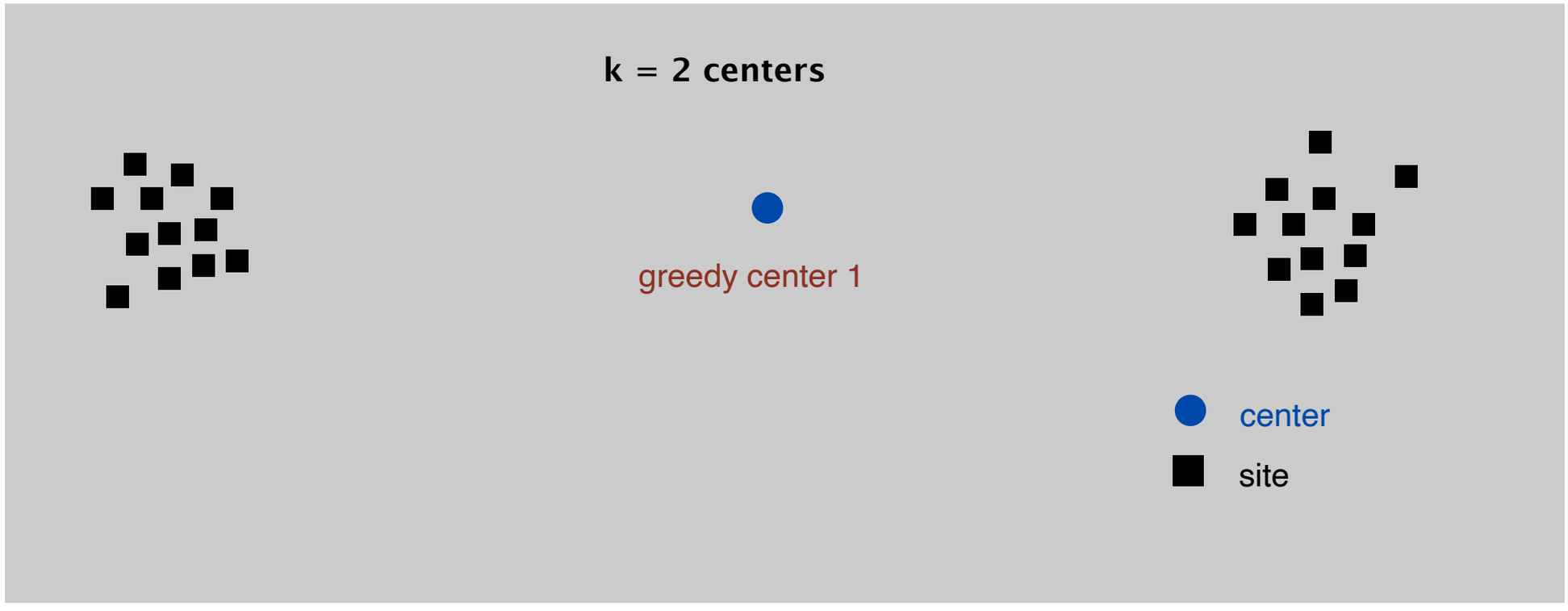
⋮

c_k : —"—

Greedy algorithm – one that doesn't work

Greedy algorithm. Put the first center at the best possible location for a single center, and then keep adding centers so as to reduce the covering radius each time by as much as possible.

Remark: r(C) might get very large



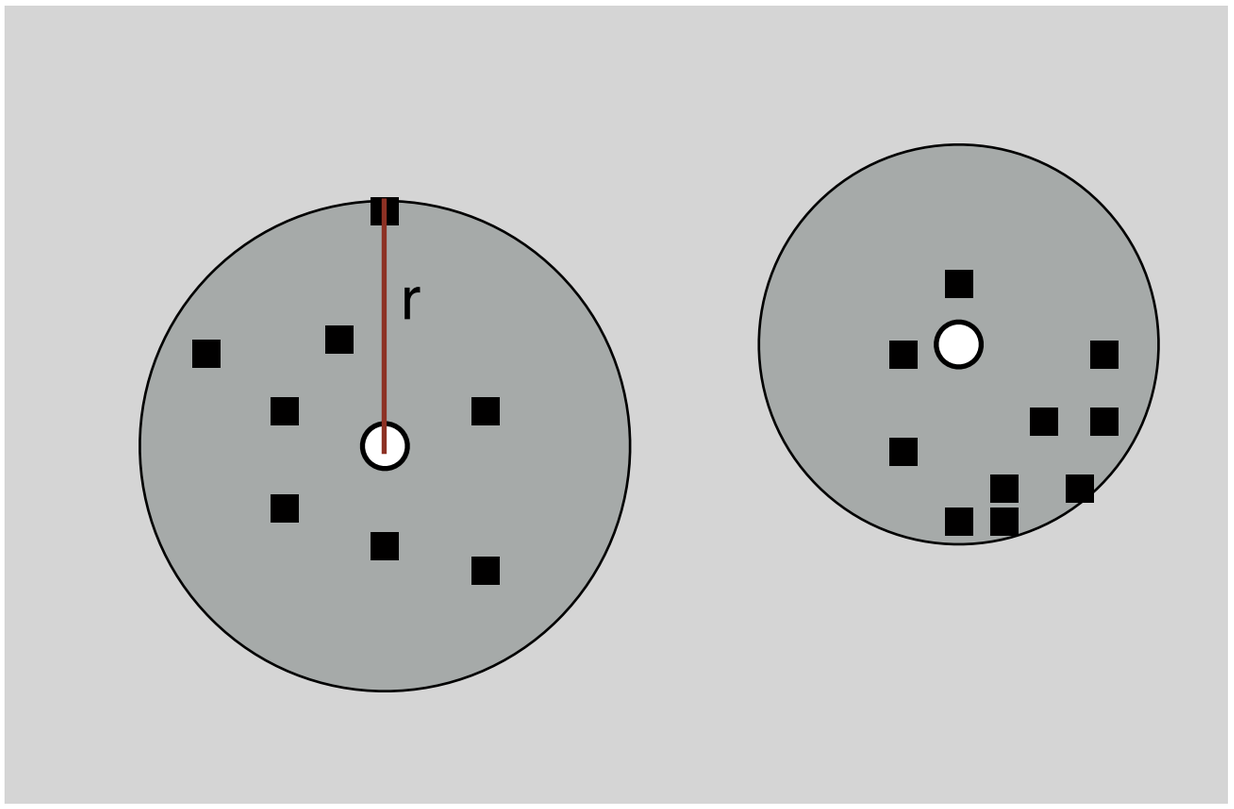
sites as centers

idea: select the k centers among the sites $s_1,\ldots,s_n$

pros, cons?

pro: - finite # of locations to check
- every center is dist = 0 from at least one site

con:
- previous greedy algo would still pick a point in the center
- not favorable to outliers
- not an optimal solution



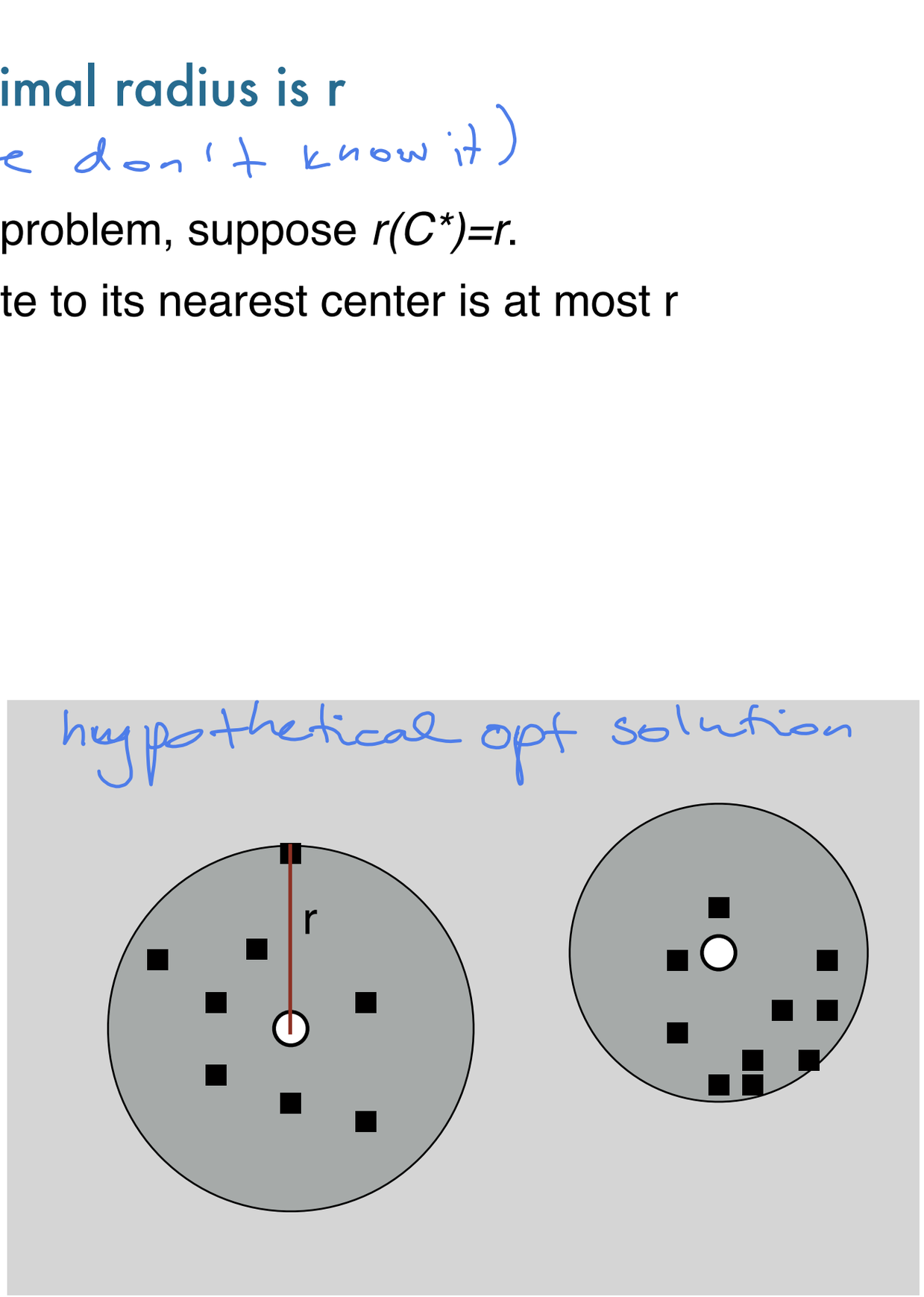
⟹ hopefully it yields a good approx

distance from center when optimal radius is r

actual optimum (we don't know it)

Let C^* be the solution to the k-enter problem, suppose $r(C^*)=r$.

- the distance $dist(s_i, C^*)$ of any site to its nearest center is at most r



TopHat - sites as centers

Let C^* be the solution to the k-enter problem, suppose $r(C^*)=r$.

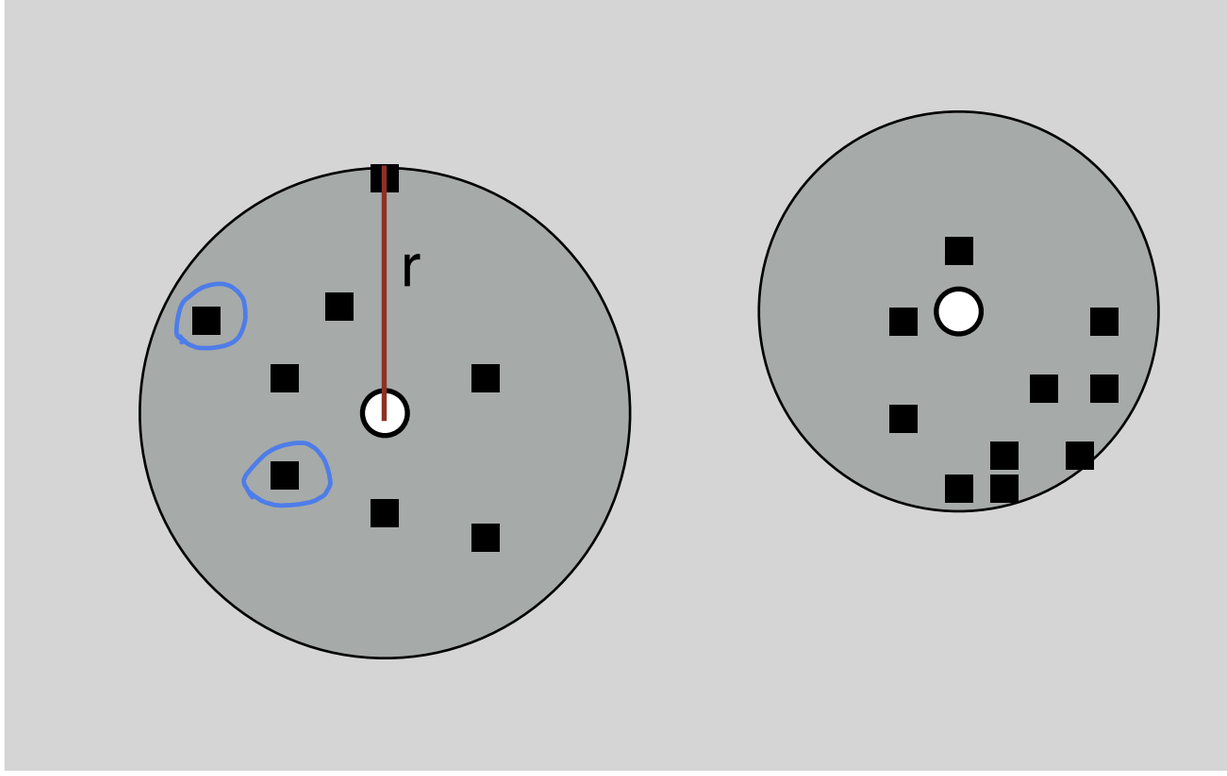
- the distance $dist(s_i, C^*)$ of any site to its nearest center is at most r

Question:

c is one of the optimal centers. What is the best upper bound on the distance between any two sites in c's cluster?

same cluster

A. max distance between any two sites in the dataset
B. r
C. 2r
D. r^2
E. we don't know



next slide →

optimal radius r – max distance within clusters

Let *C** be the solution to the k-enter problem, suppose *r(C*)=r*.

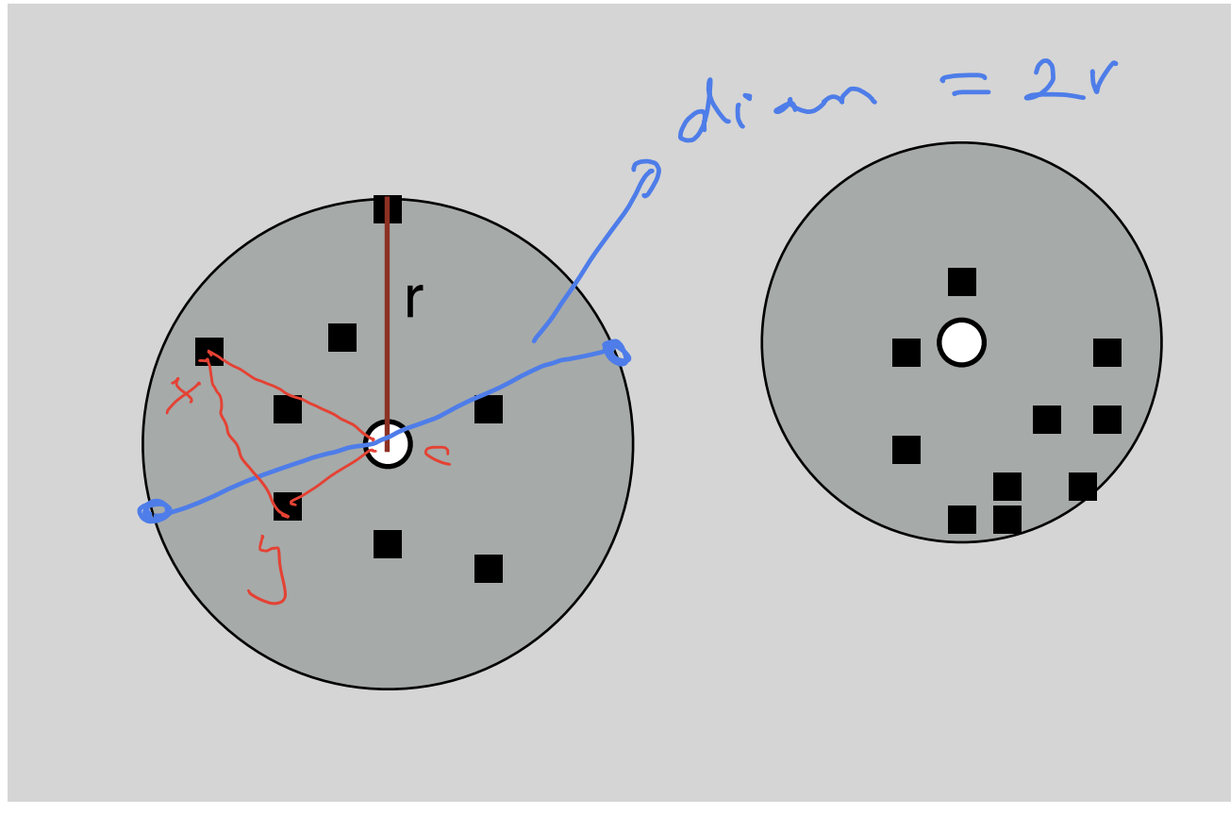
claim: for any two sites s and z assigned to center c $dist(s, z) \leq 2r$

$dist(x,y) \leq dist(x,c) + dist(c,y)$

$\leq r \qquad \leq r$

$= 2r$

use the fact that sites in the same cluster are close to each other



optimal radius r – max distance within clusters

Let C^* be the solution to the k-enter problem, suppose $r(C^*)=r$.

claim: for any two sites s and z assigned to center c $dist(s,z) \leq 2r$

proof:

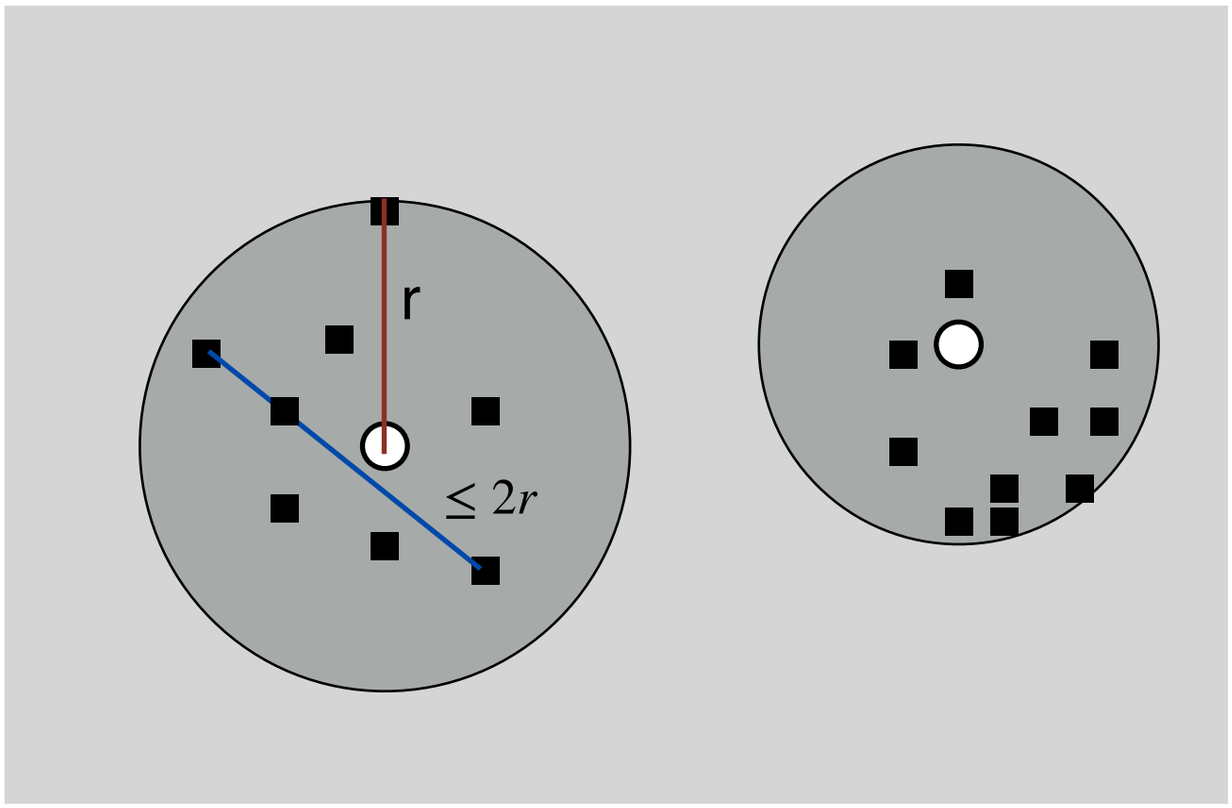
by triangle inequality we know

$$dist(s,z) \leq dist(s,c) + dist(c,z)$$

since s and z are both in the cluster of c we also know

$$dist(s,c) \leq r$$

$$dist(z,c) \leq r$$



2-approximation greedy algorithm

Let C^* be the solution to the k-center problem, suppose $r(C^*)=r$.

select the k centers among the sites $s_1,\ldots,s_n$

Algorithm:

pick sites as centers one at a time.

pick first at random

→ take furthest site next

→ furthest from both. next, etc.

2-approximation greedy algorithm

Let C^* be the solution to the k-center problem, suppose $r(C^*)=r$.

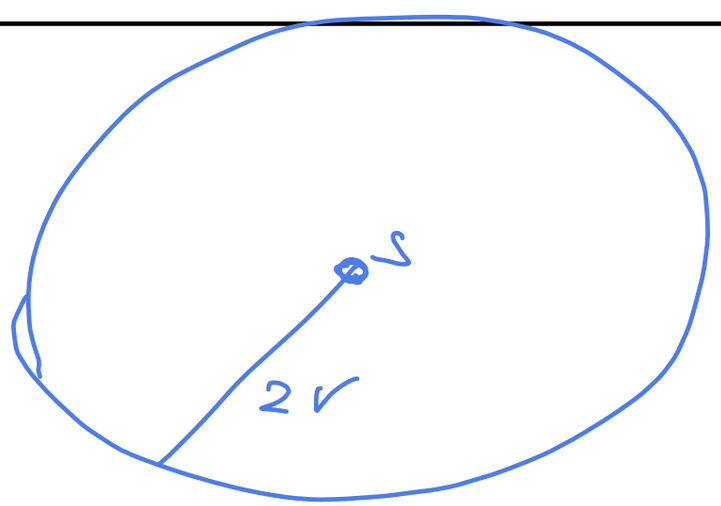
select the k centers among the sites $s_1,\ldots,s_n$

r is an input. But we don't know it! ↓ will figure out later

Algorithm 1: kCenterWithR$(k, s_1, s_2, \ldots, s_n, r)$

```
   /* s_i are the sites, r is the optimal radius                    */
1  S ← s_1, ... s_n /* uncovered sites                               */
2  C ← ∅ /* centers                                                 */
3  for j = 1 to k do
4  |   s ← random element from S;
5  |   C ← C ∪ {s};
6  |   remove sites from S with distance ≤ 2r from s;
7  return C
```

sites not covered → not assigned to cluster

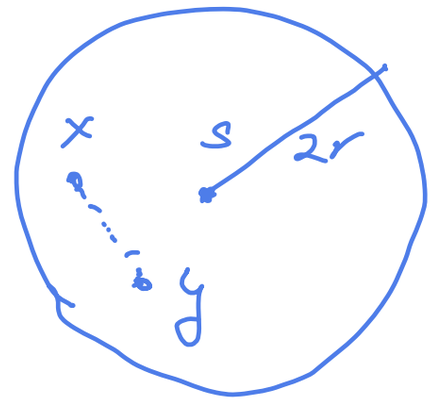


2-approximation greedy algorithm

Toptal

Algorithm 1: kCenterWithR$(k, s_1, s_2, \ldots, s_n, r)$

```
   /* s_i are the sites, r is the optimal radius                    */
1  S ← s_1, ... s_n /* uncovered sites                               */
2  C ← ∅ /* centers                                                 */
3  for j = 1 to k do
4  |   s ← random element from S;
5  |   C ← C ∪ {s};
6  |   remove sites from S with distance ≤ 2r from s;
7  return C
```



Select the true statements about this algorithm.

s is one of the sites selected as a center (line 4)

should be ≤ twice the radius
2(2r)=4r

A. if a pair of sites x and y are assigned to site s (line 6), then $dist(x, y) \leq 2r$

B. after k iterations some sites may remain unassigned → we know the opt rad = r (next slide)

C. sites in S that are in the same cluster as s in the *optimal solution* are assigned to s (line 6) → we know that every site originally in the same cluster as s is at dist ≤ 2r

D. the radius of the clusters yielded by this algorithm is 2r
by design of algo

2-approximation greedy algorithm

Algorithm 1: kCenterWithR$(k, s_1, s_2, \ldots, s_n, r)$

```
   /* s_i are the sites, r is the optimal radius                    */
1  S ← s_1, ... s_n /* uncovered sites                               */
2  C ← ∅ /* centers                                                 */
3  for j = 1 to k do
4  |   s ← random element from S;
5  |   C ← C ∪ {s};
6  |   remove sites from S with distance ≤ 2r from s;
7  return C
```

Is this indeed a 2-approximation?

meaning: greedy solution is at most 2x worst than the optimal

opt returns centers with radius r

greedy — " — radius 2r

2-approximation greedy algorithm

Algorithm 1: kCenterWithR$(k, s_1, s_2, \ldots, s_n, r)$

```
   /* s_i are the sites, r is the optimal radius                    */
1  S ← s_1, ... s_n /* uncovered sites                               */
2  C ← ∅ /* centers                                                 */
3  for j = 1 to k do
4  |   s ← random element from S;
5  |   C ← C ∪ {s};
6  |   remove sites from S with distance ≤ 2r from s;
7  return C
```

Do the resulting 2r radius clusters cover all the sites?

In the optimal clustering: every site is within $2r$ dist of sites in the same cluster

$\Rightarrow$ when we pick s specifically as the next center
$\rightarrow$ we assign every site within $2r$
$\rightarrow\rightarrow$ these sites are assigned.

conclusion: s covers its entire original cluster (maybe more)

2-approximation greedy algorithm

Algorithm 1: kCenterWithR$(k, s_1, s_2, \ldots, s_n, r)$

```
   /* s_i are the sites, r is the optimal radius                    */
1  S ← s_1, ... s_n /* uncovered sites                               */
2  C ← ∅ /* centers                                                 */
3  for j = 1 to k do
4  |   s ← random element from S;
5  |   C ← C ∪ {s};
6  |   remove sites from S with distance ≤ 2r from s;
7  return C
```

Is this indeed a 2-approximation? to be answered:

- does this cover all the sites? ← does it actually solve the problem
- is there indeed an approx factor of 2?

- Sites that are removed are within 2r of their center by design of the algorithm.
- We know that there exist an optimal clustering with k centers (and radius r)
- When site s is selected as the next center, all sites that were in s' optimal cluster are within radius 2r of s, hence are removed
- in each iteration we remove every site from an entire optimal cluster (possible more)
- after k iterations we can't have any sites remaining

2-approximation greedy algorithm – binary search for r

The previous algorithm assumed that the optimal r is known. What can we do if we don't know r?

first try $r = \frac{\text{max dist among sites}}{2}$

↓

if less than k clusters

$\frac{r}{2}$

↓

$\frac{\frac{r}{2}}{2}$

⋮

2-approximation farthest-first greedy algorithm

We don't need r!

idea: instead of picking random sites as centers, always pick the one furthest from the previous

Algorithm 1: kCenter$(k, s_1, s_2, \ldots, s_n)$

```
/* s_i are the sites, r is the optimal radius                    */
C ← initialize as a random site s;
for j = 2 to k do
    s ← the site with maximum dist(s, C);
    C ← C ∪ {s};
return C
```

goal: in each iter pick a center that is more than dist 2r from previous centers

⟶ if such a site exist, then the furthest center is like that

(notice that we didn't remove anything from S)

2-approximation farthest-first greedy algorithm

Algorithm 1: kCenter$(k, s_1, s_2, \ldots, s_n)$

```
   /* s_i are the sites, r is the optimal radius                    */
1  C ← initialize as a random site s;
2  for j = 2 to k do
3  |   s ← the site with maximum dist(s, C);
4  |   C ← C ∪ {s};
5  return C
```

Theorem: kCenter is a 2-approximation algorithm.

proof:

2-approximation farthest-first greedy algorithm

Algorithm 1: kCenter$(k, s_1, s_2, \ldots, s_n)$

```
   /* s_i are the sites, r is the optimal radius                    */
1  C ← initialize as a random site s;
2  for j = 2 to k do
3  |   s ← the site with maximum dist(s, C);
4  |   C ← C ∪ {s};
5  return C
```

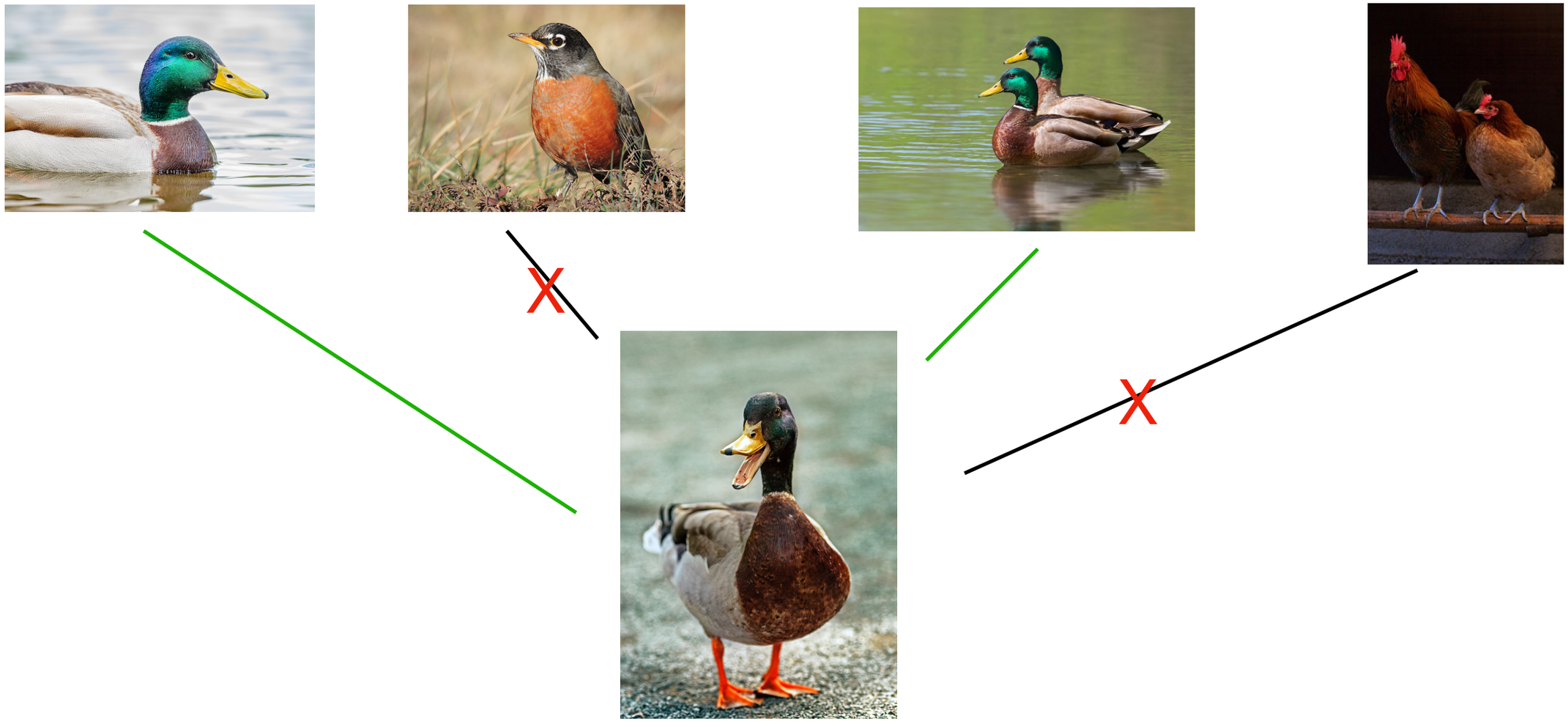
Theorem: kCenter is a 2-approximation algorithm.

proof:

- we know from the previous proof that any two sites in the same cluster have distance at most 2r
- at iteration j, if there is a site not within 2r of the current C, then the furthest site is definitely such a site
- after k iterations we can't have any site uncovered because we know there is an optimal solution with k sites. → previous proof

Classification

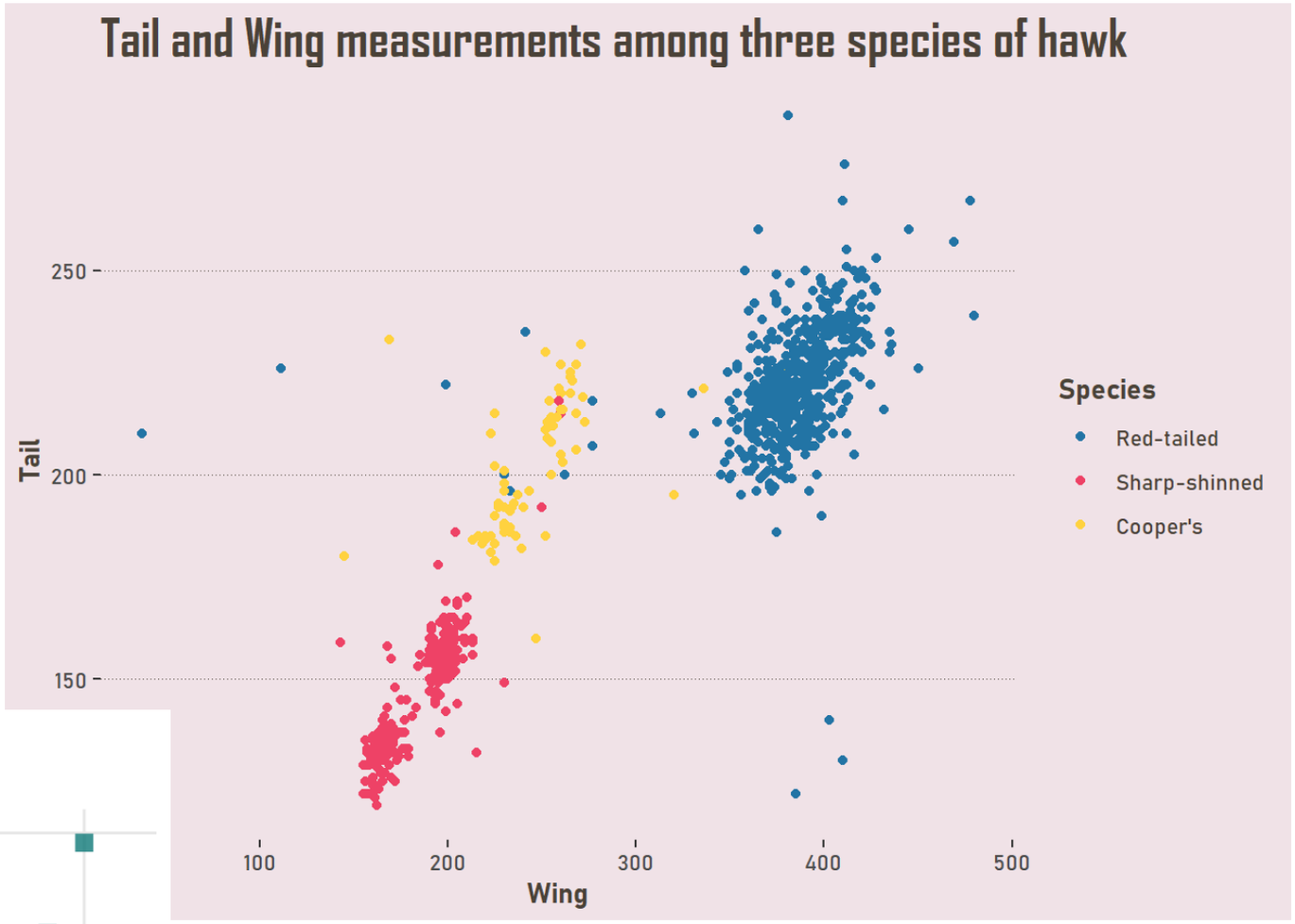
- Want to assign a label to some data item — picture of animal
- Compare it to items that you already know the label of — other animals
- Choose the label of the most similar item(s)



If it walks like a duck, it quacks like a duck, then most likely it is a duck.

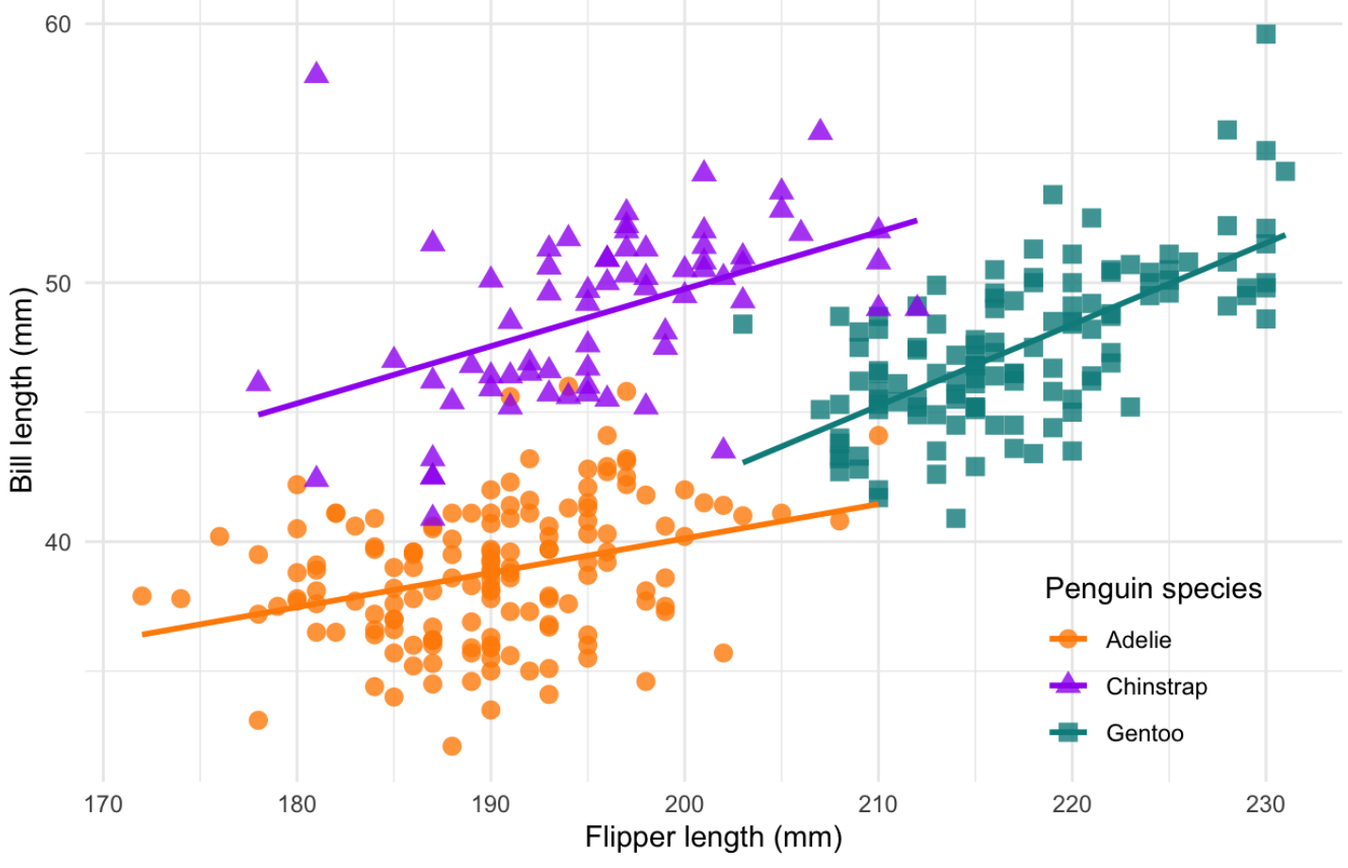
Penguins and hawks

Cornel College at Mount Vernon, Iowa: McBride Hawks dataset



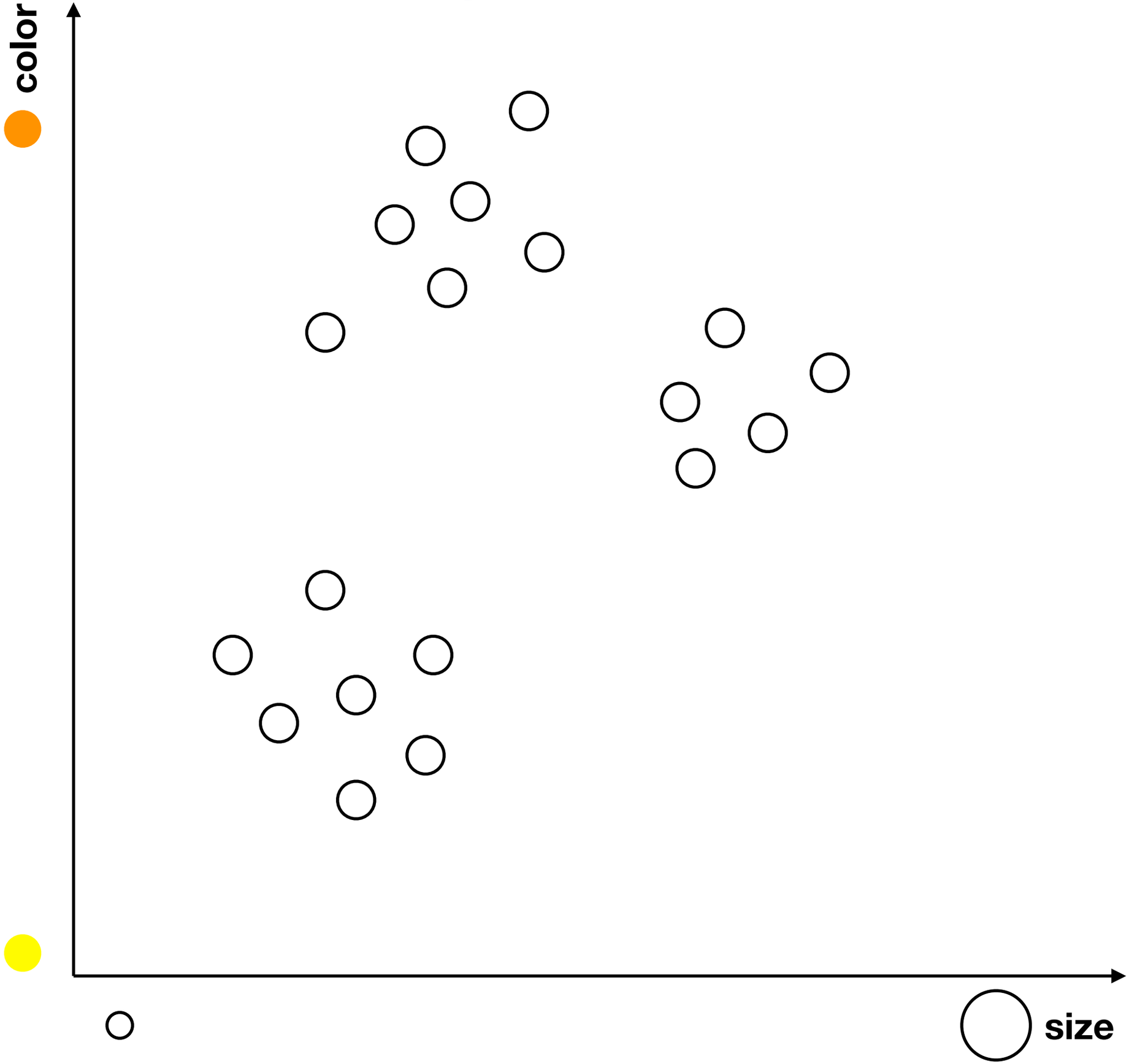
Flipper and bill length

Dimensions for Adelie, Chinstrap and Gentoo Penguins at Palmer Station LTER

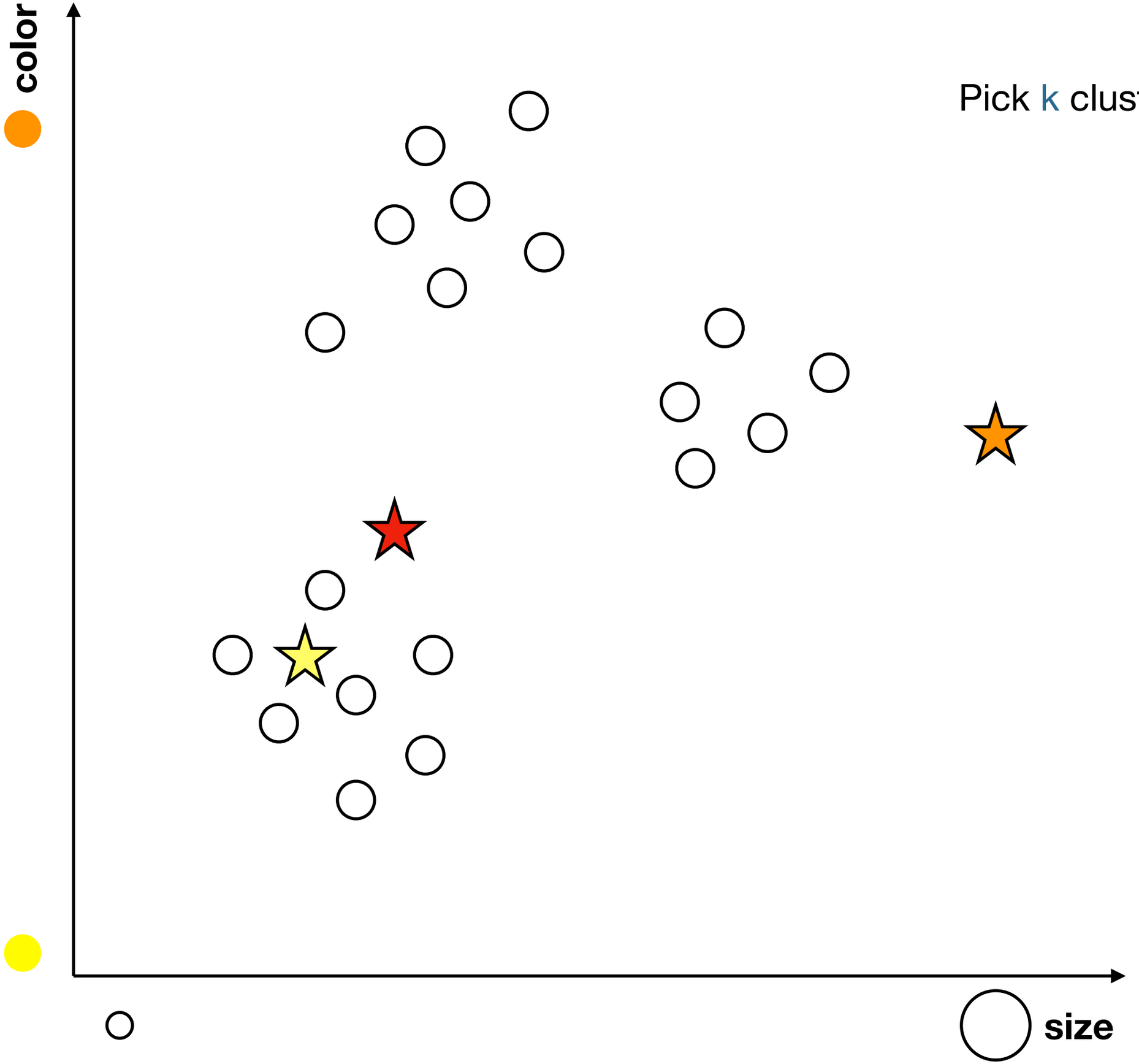


Kristen Gorman: Palmer Archipelago (Antarctica) dataset

Iterative cluster assignment

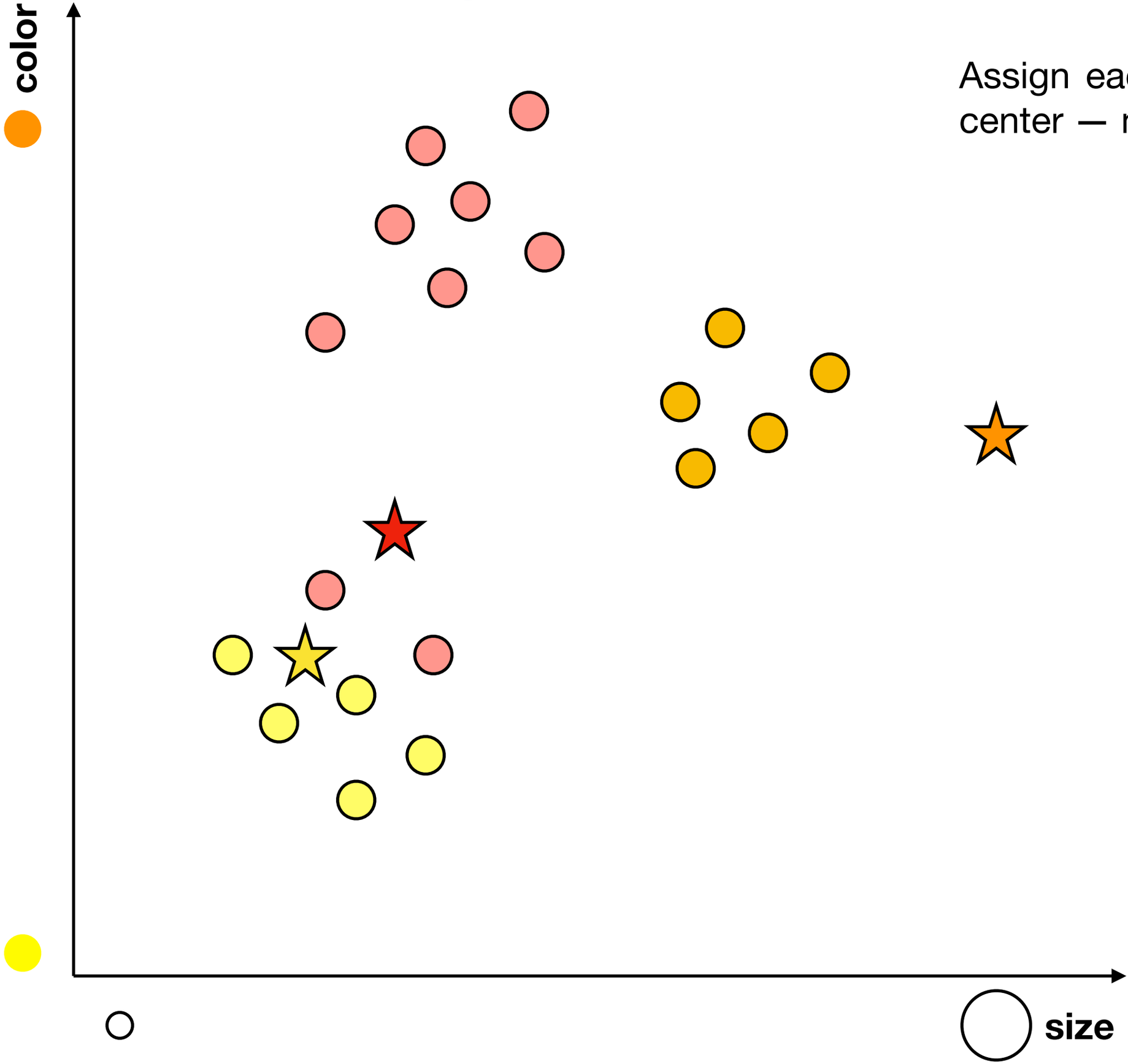


Iterative cluster assignment



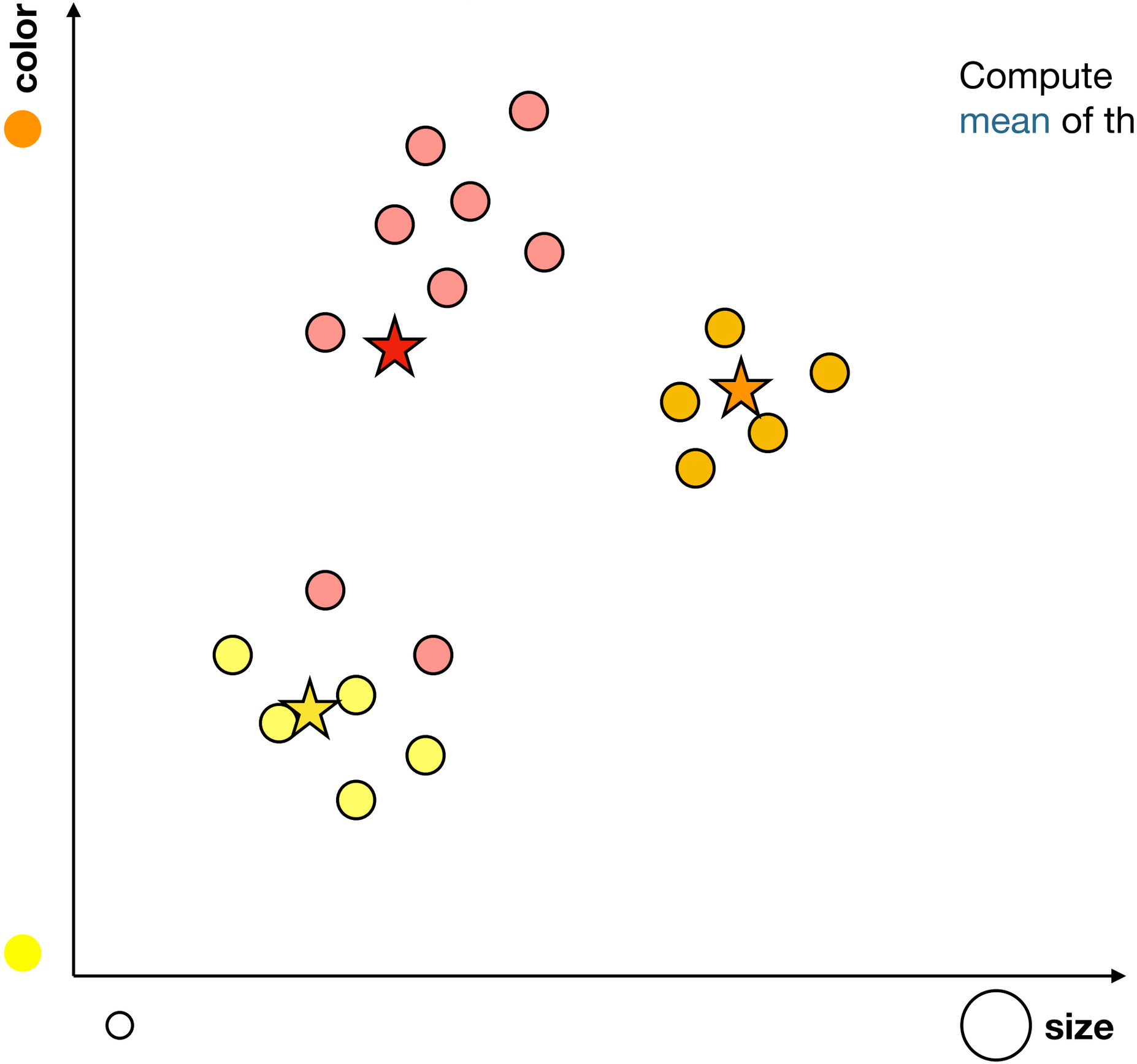
Pick k cluster centers — mean

Iterative cluster assignment



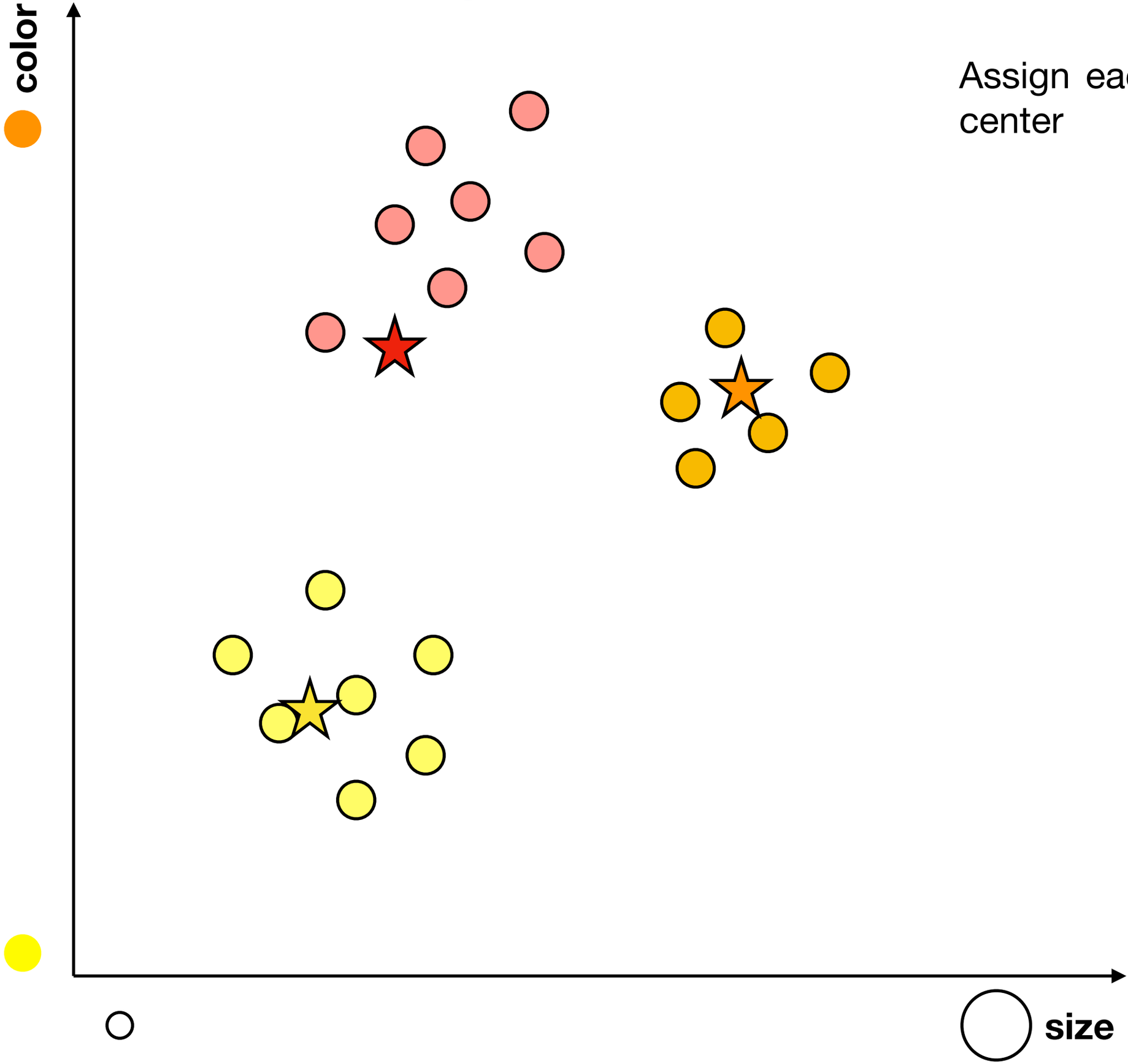
Assign each point to its nearest center — most likely cluster

Iterative cluster assignment



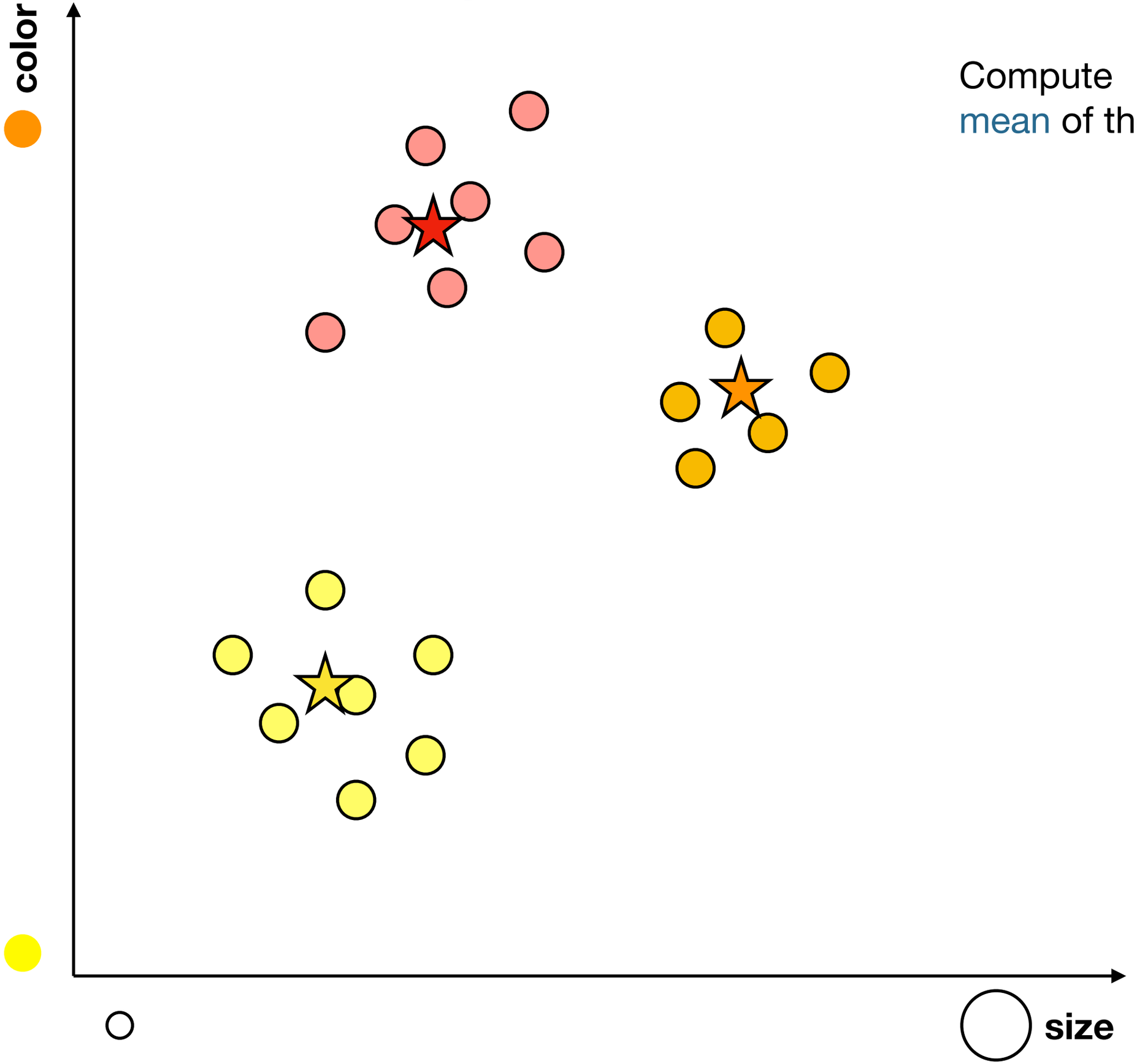
Compute new centers as the mean of the assigned vectors

Iterative cluster assignment



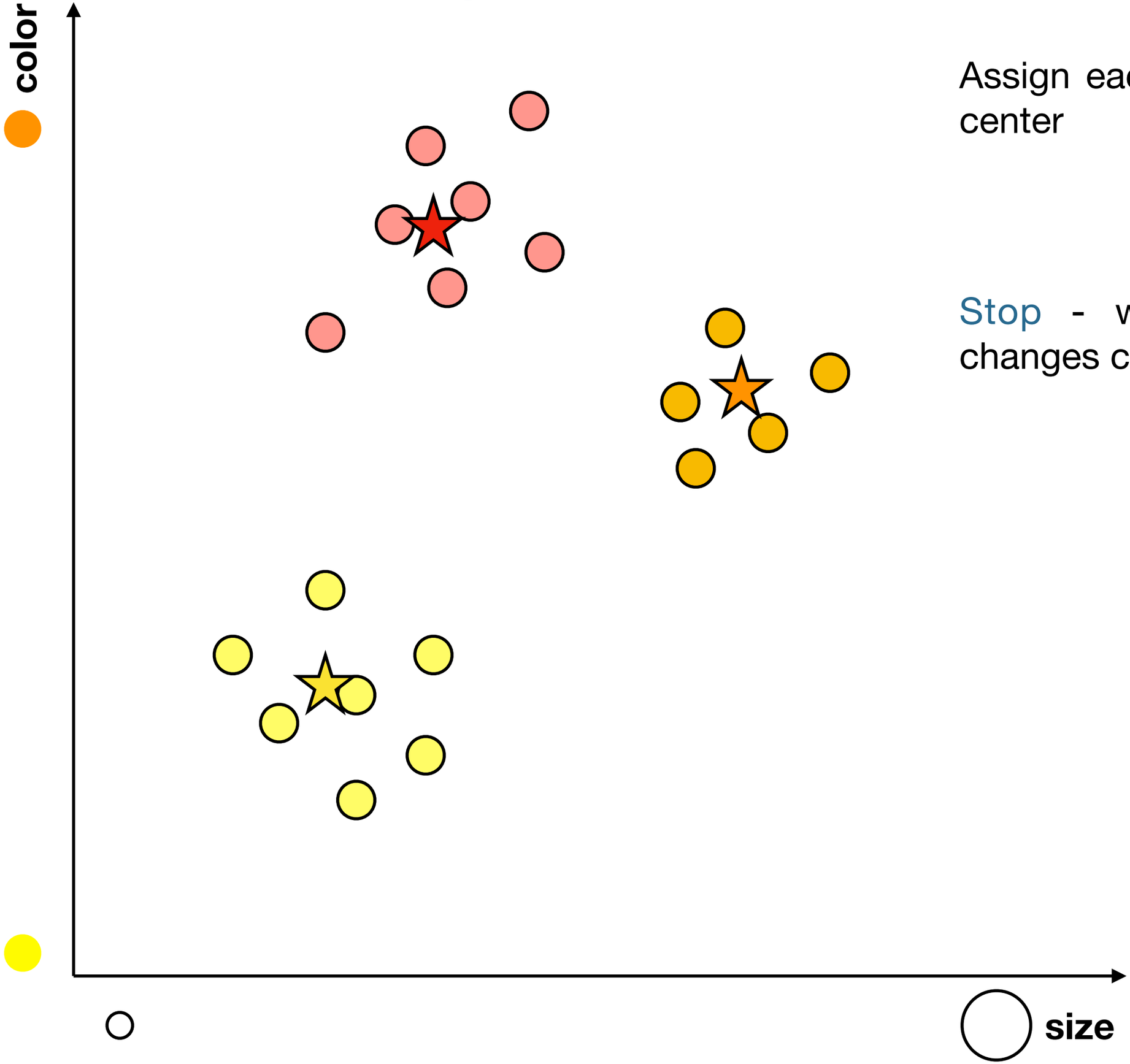
Assign each point to its nearest center

Iterative cluster assignment



Compute new centers as the mean of the assigned vectors

Iterative cluster assignment



Assign each point to its nearest center

Stop - when no data point changes cluster.

S

k-means: Given n data points s_1, s_2, …, s_n and an integer k, find k points c_1, c_2, …, c_k, called centers or means, such that when each x_i is assigned to the center nearest to it, then the sum of squared distances is minimized.

$$\sum_{i=1}^{n} dist(s_i, C)^2 \rightarrow \min$$

k-median: similar to k means, but with a different objective function.

$$\sum_{i=1}^{n} \min_j |s_i - c_j| \rightarrow \min$$

k-medoid: cluster centers are data items.

k-center: minimize the max distance

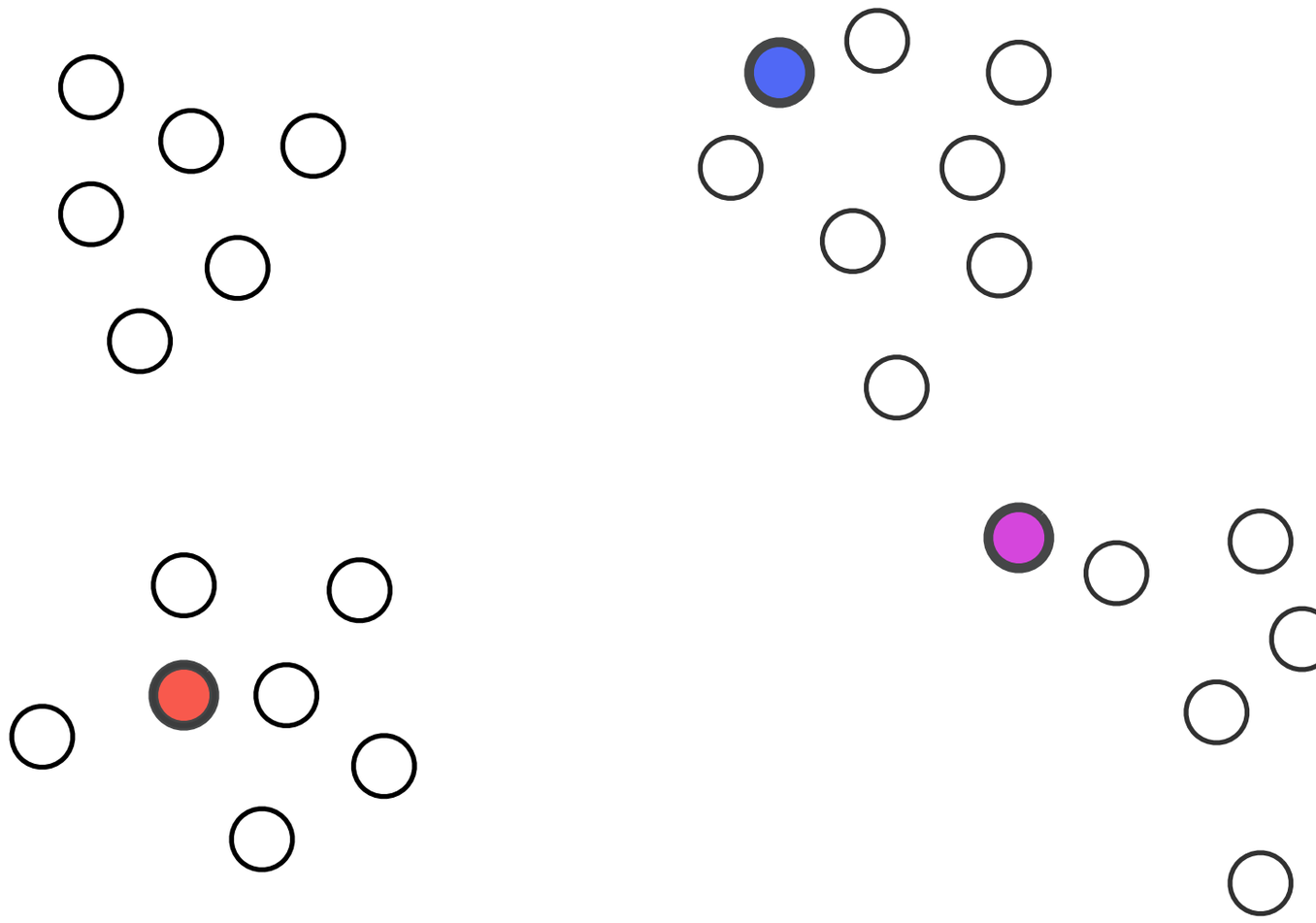
$$\max_{i=1}^{n} dist(s_i, C) = r(C) \rightarrow \min$$

algorithm:

1. pick k cluster centers among x_1, x_2, …, x_n at random
2. assign each data point to it's nearest center (Euclidean distance)
3. within each cluster find which of the points minimizes the total distance and assign that as the new center
4. repeat steps 2 and 3. until convergence

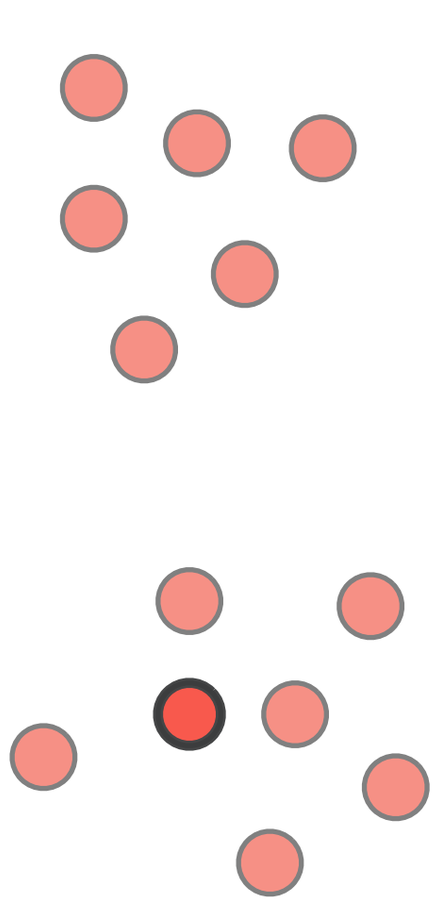
k-means: random initialization

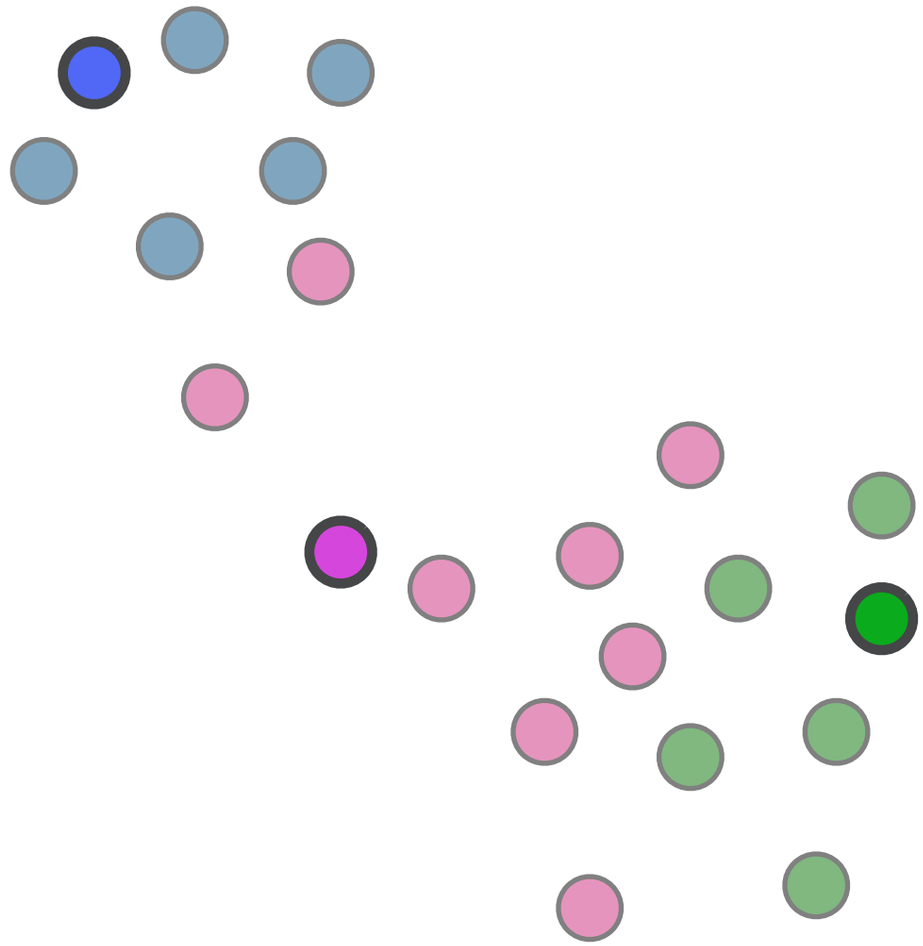
Initialize



k-means: random initialization

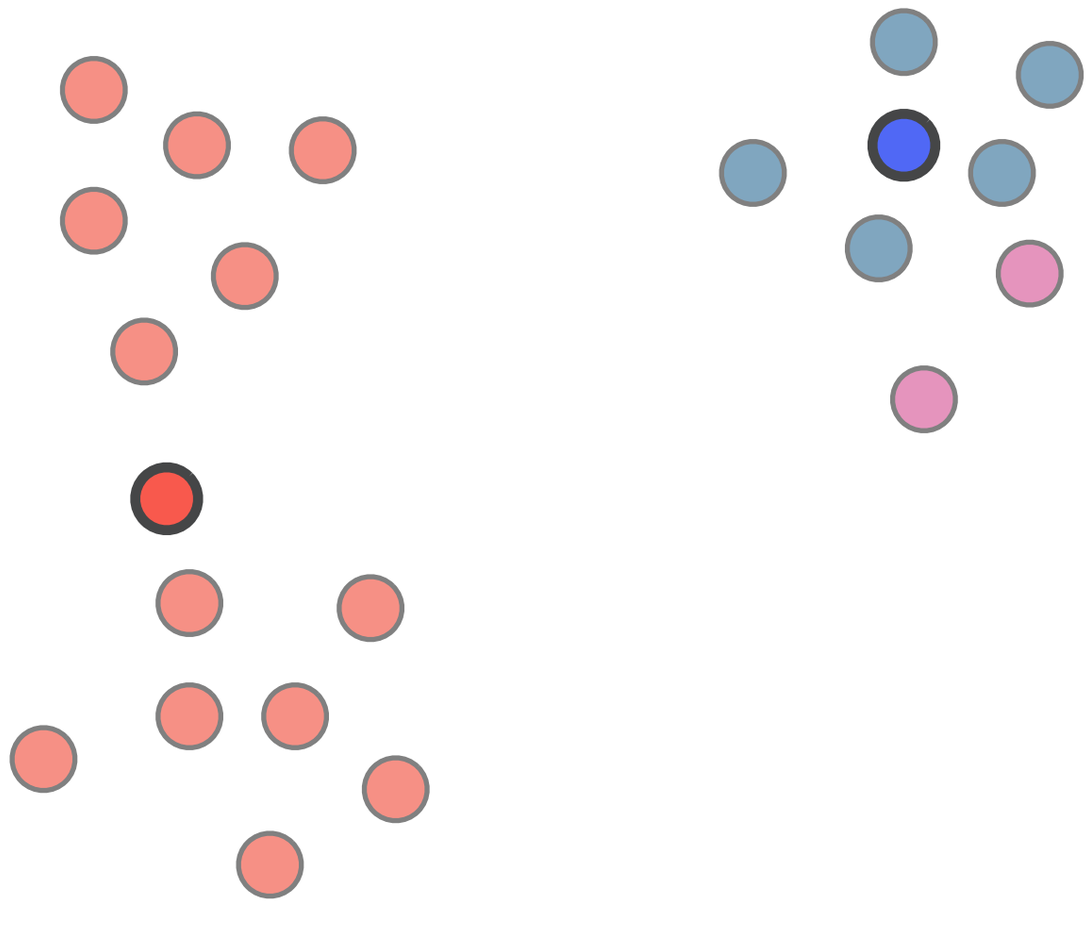
assign points to nearest cluster





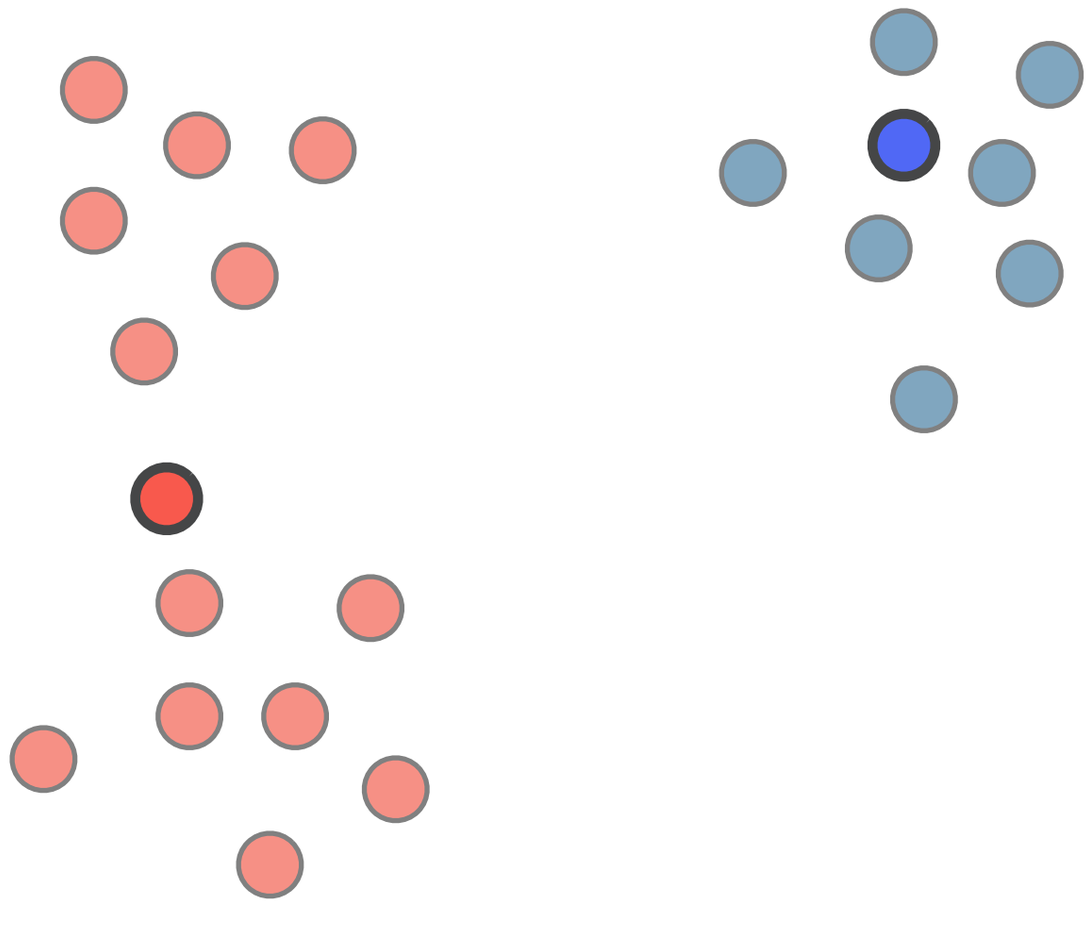
k-means: random initialization

recompute cluster centers



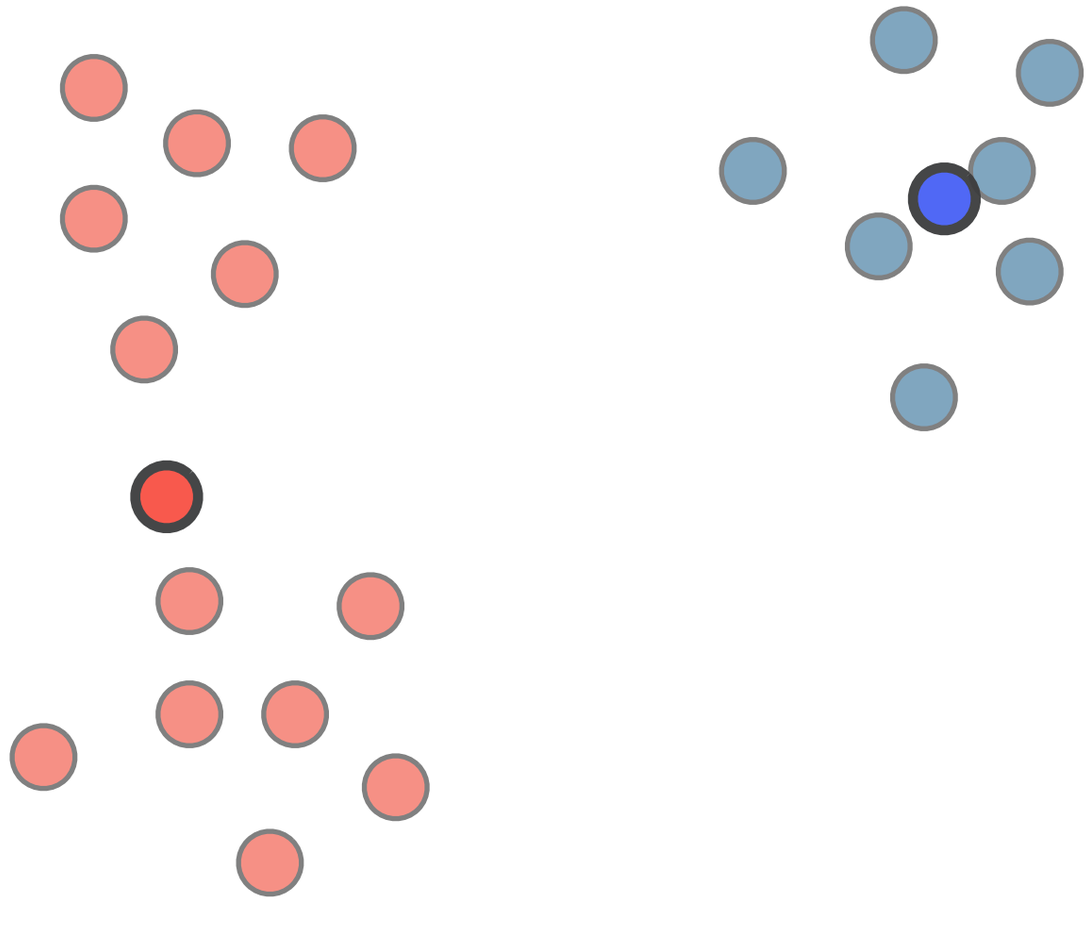
k-means: random initialization

re-assign points



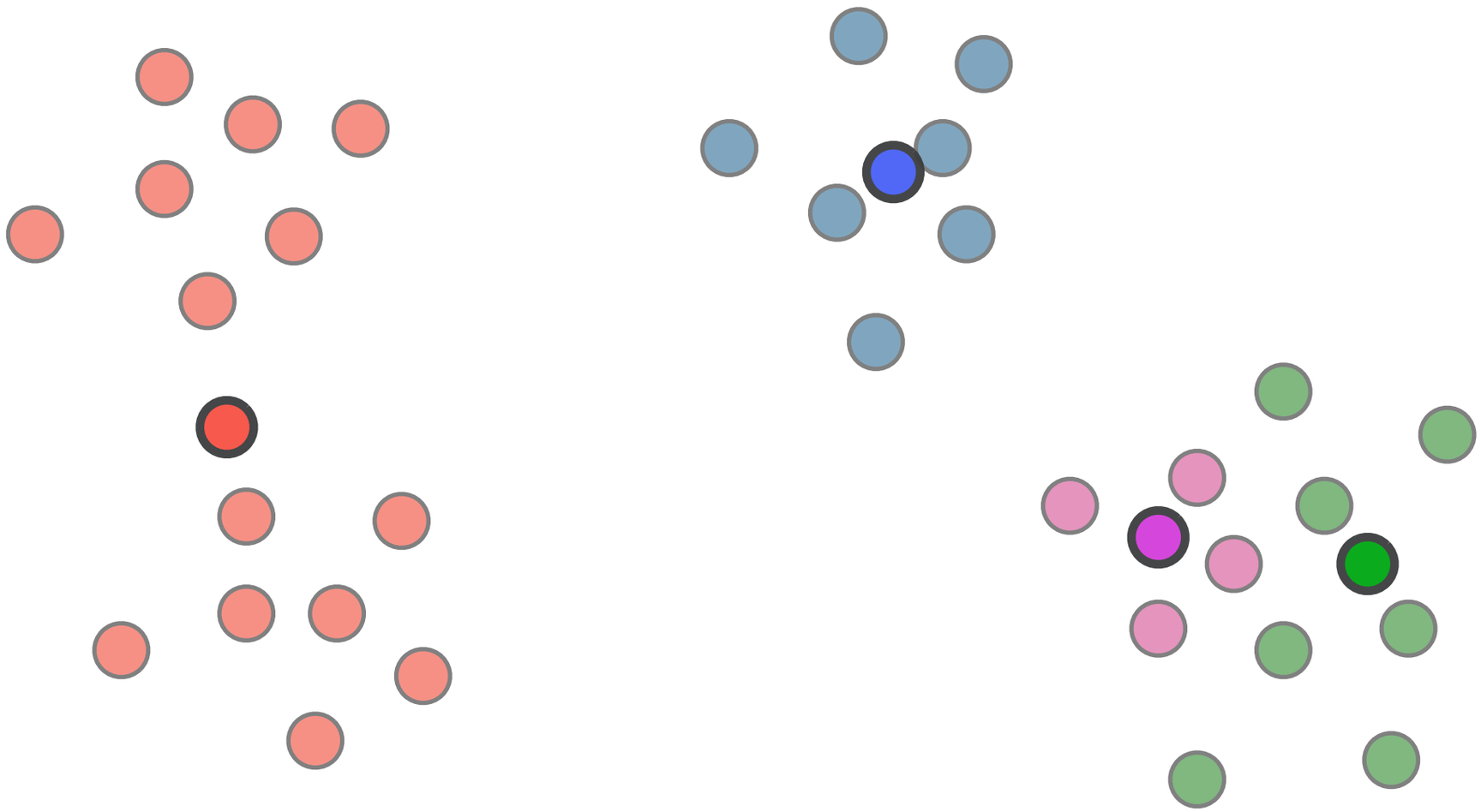
k-means: random initialization

re-compute centers



k-means: random initialization

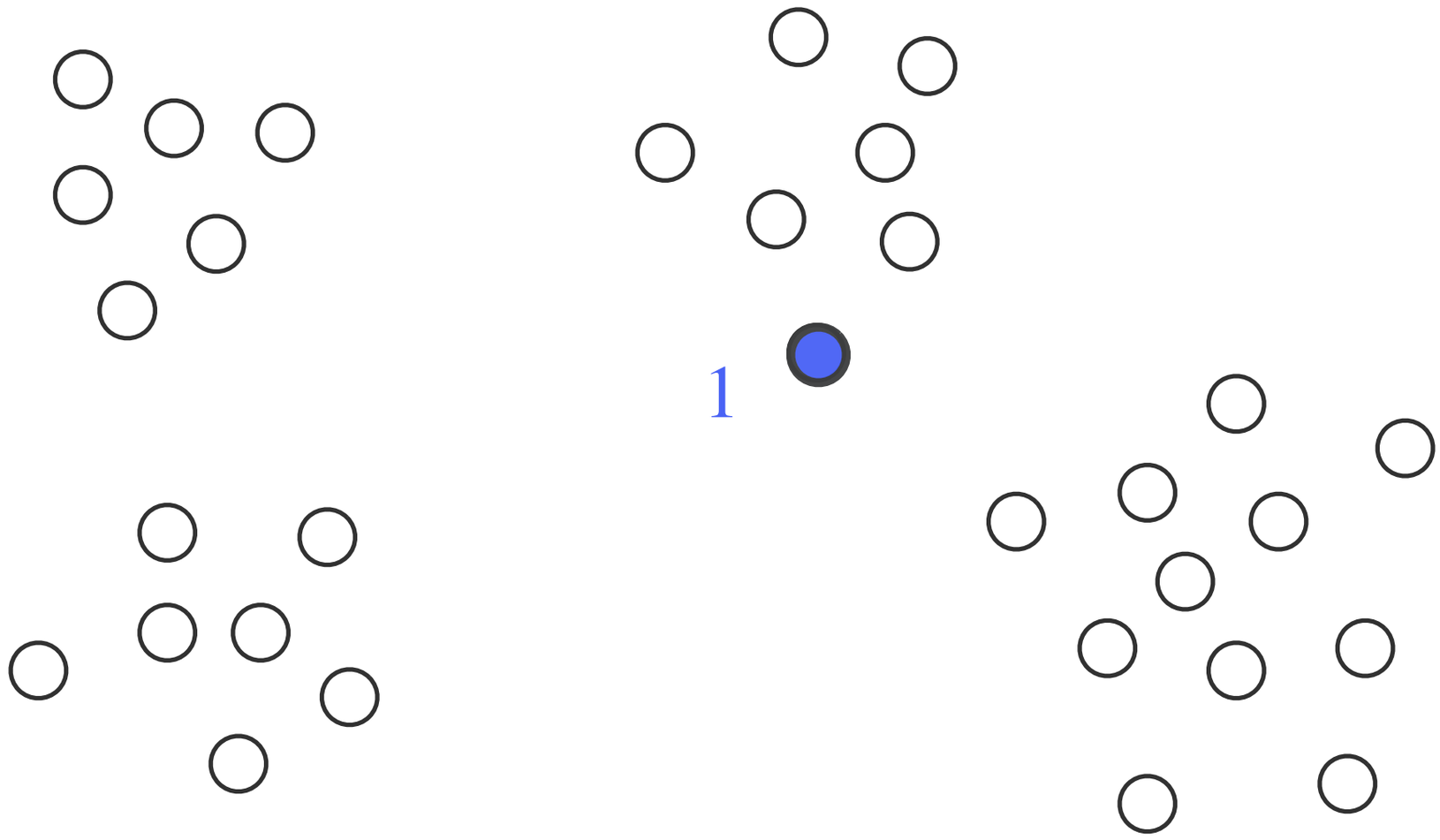
re-compute centers



sensitive to the choice of the initial centers; clusters may turn out suboptimal

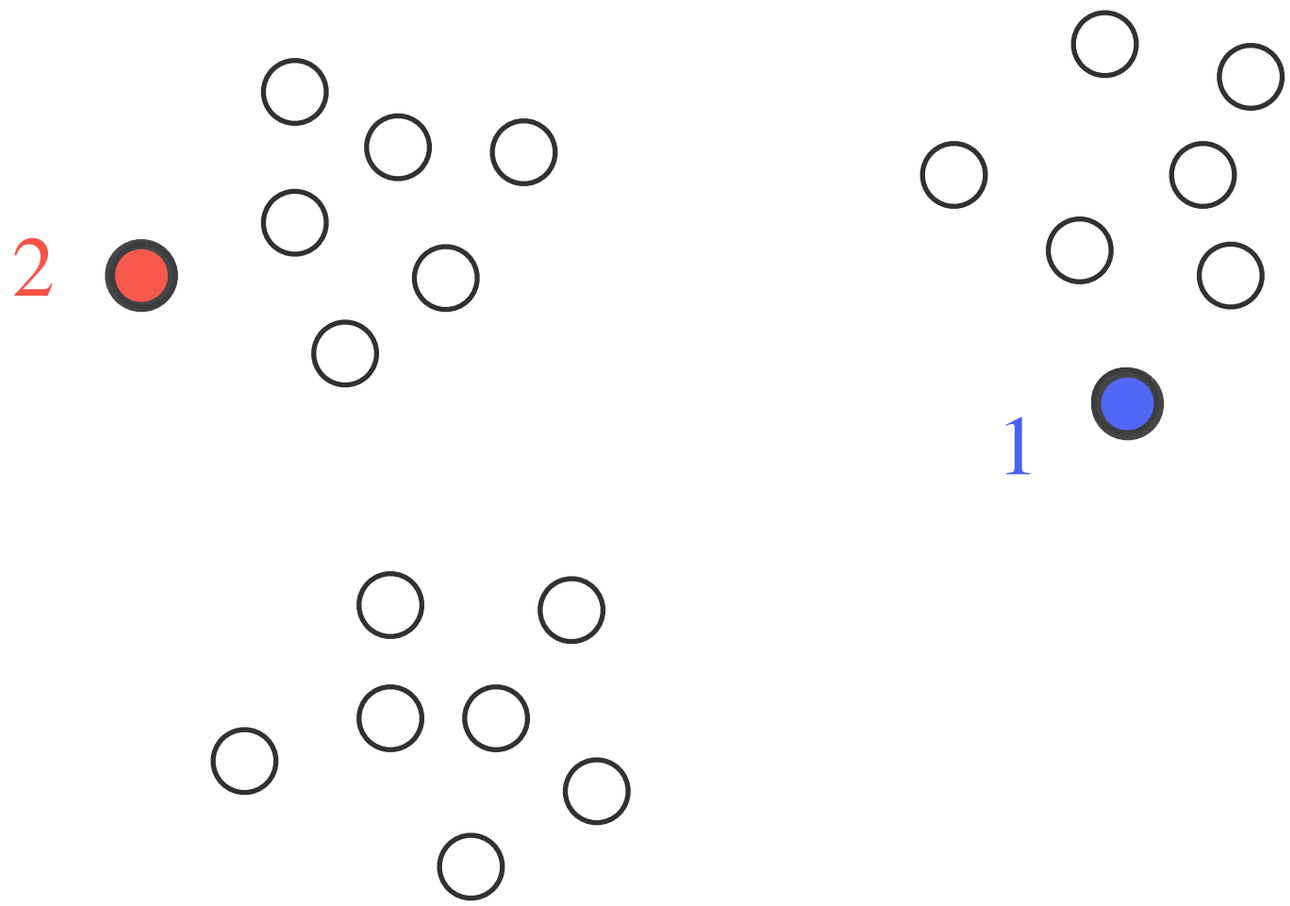
k-means: initialize furthest first - better

pick first data point at random



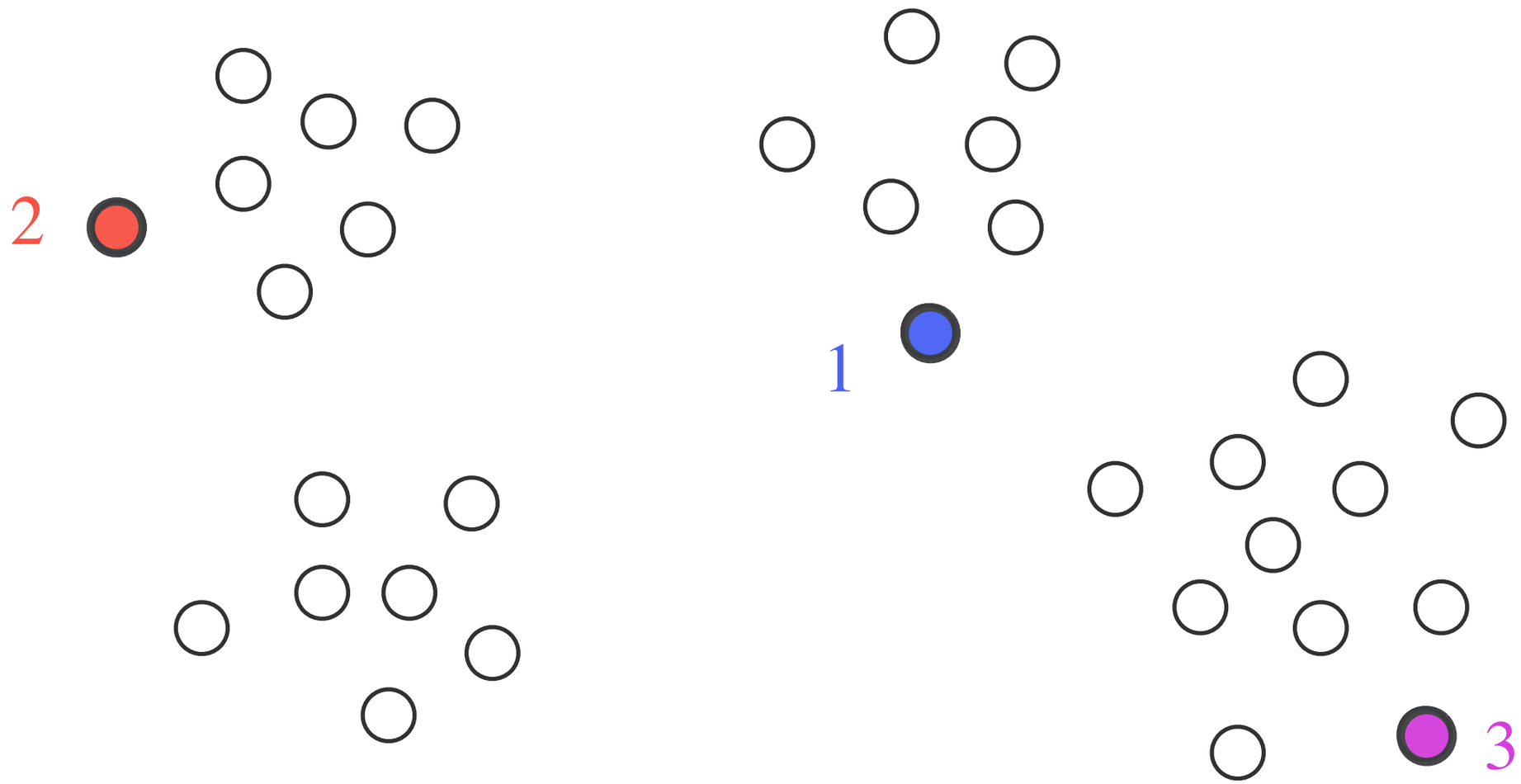
k-means: initialize furthest first

pick first data point at random
pick the point that's furthest from the first as the second center

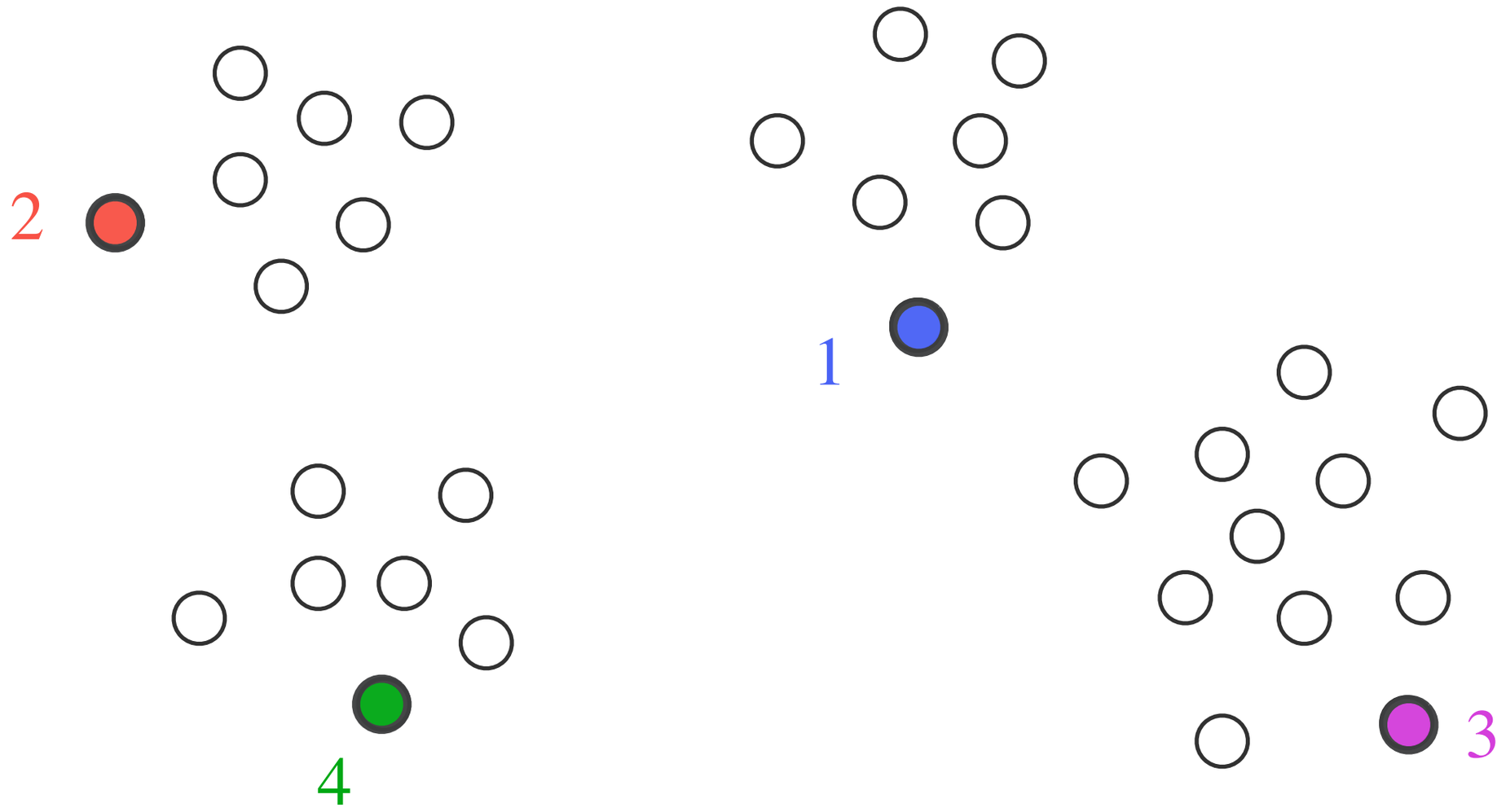


k-means: initialize furthest first

pick first data point at random
pick the point that's furthest from the first as the second center
pick the point that's furthest from the two as the third center

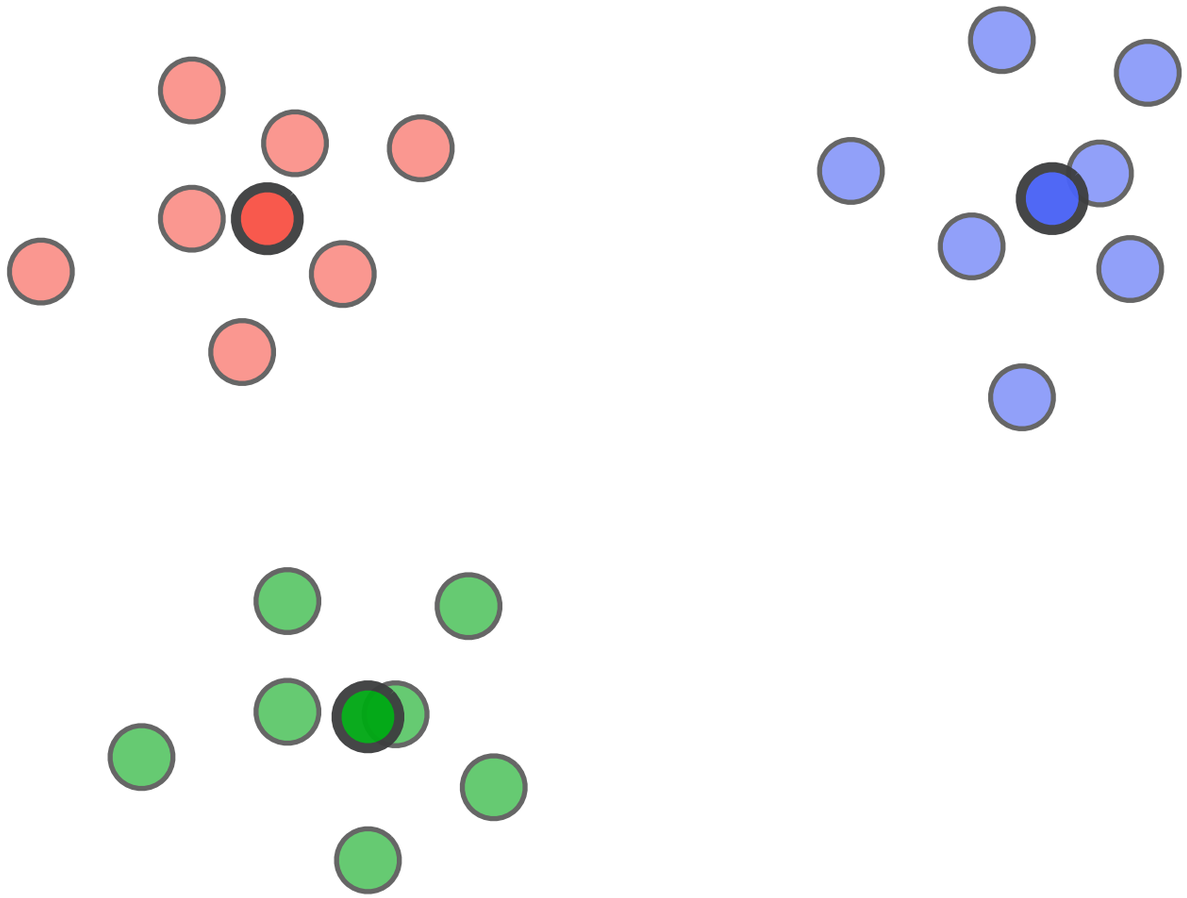


k-means: initialize furthest first



k-means: initialize furthest first

Use the furthest-first centers as starting point and run k-means.



k-means++ — provable performance guarantee

Theorem: The value of the objective function (i.e. the minimum of sum of square distances from the centers) is within an O(log k)-factor of the optimal in expectation

We say that kmeans++ is an log(k) - approximation algorithm

k is part of input

⇒ this is worst than our 2-approx

difference: two problems have different objective functions