Clustering algorithms

2IMW30 - Foundations of data mining
TU Eindhoven, Quartile 3, 2016-2016

Anne Driemel

Overview of this lecture

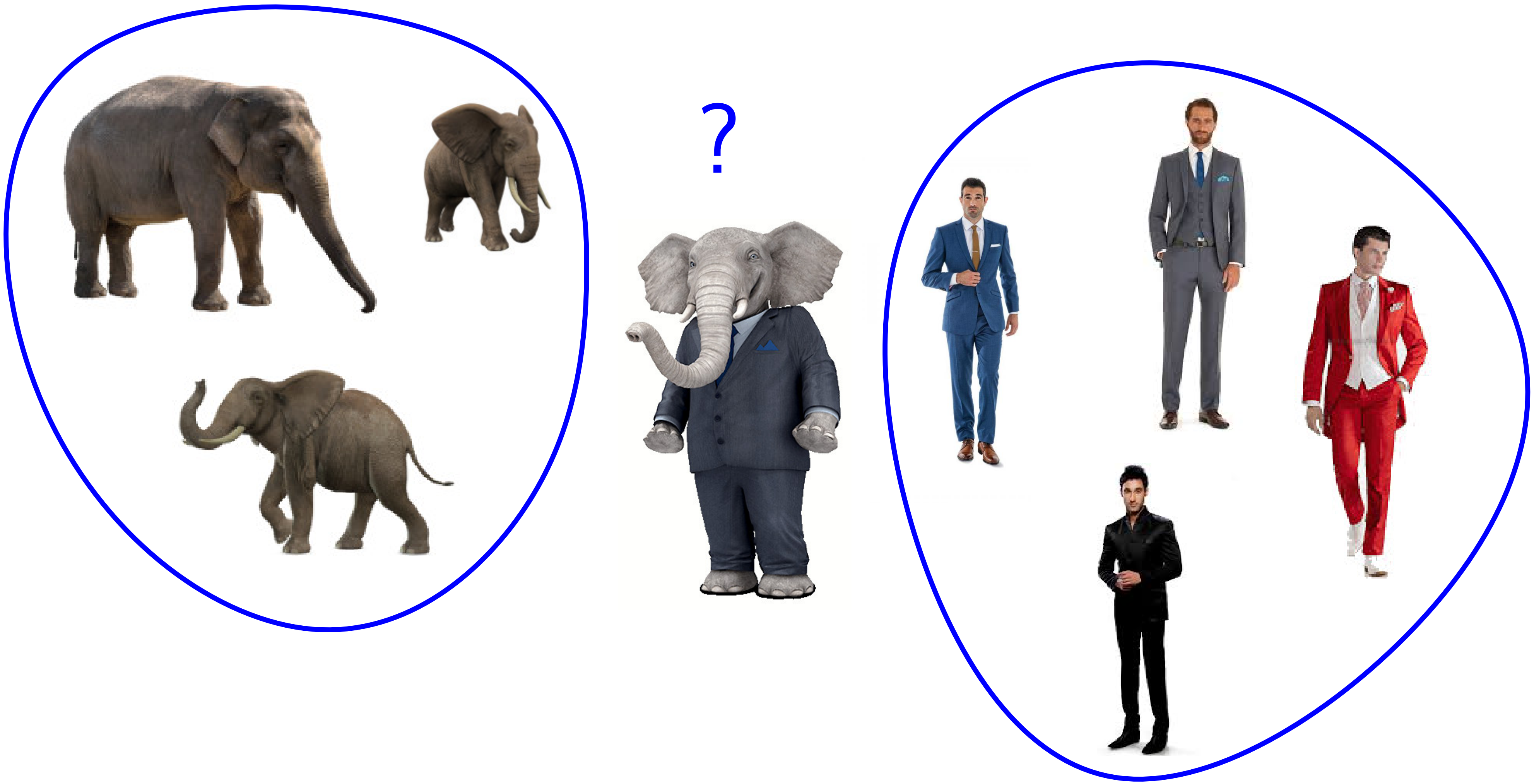
- Clustering
- Facility Location
- Gonzales' algorithm
- Loyd's algorithm (k-means)
- k-means++ algorithm
- Clustering in graphs

What is Clustering?

Clustering is the task of grouping similar objects into clusters

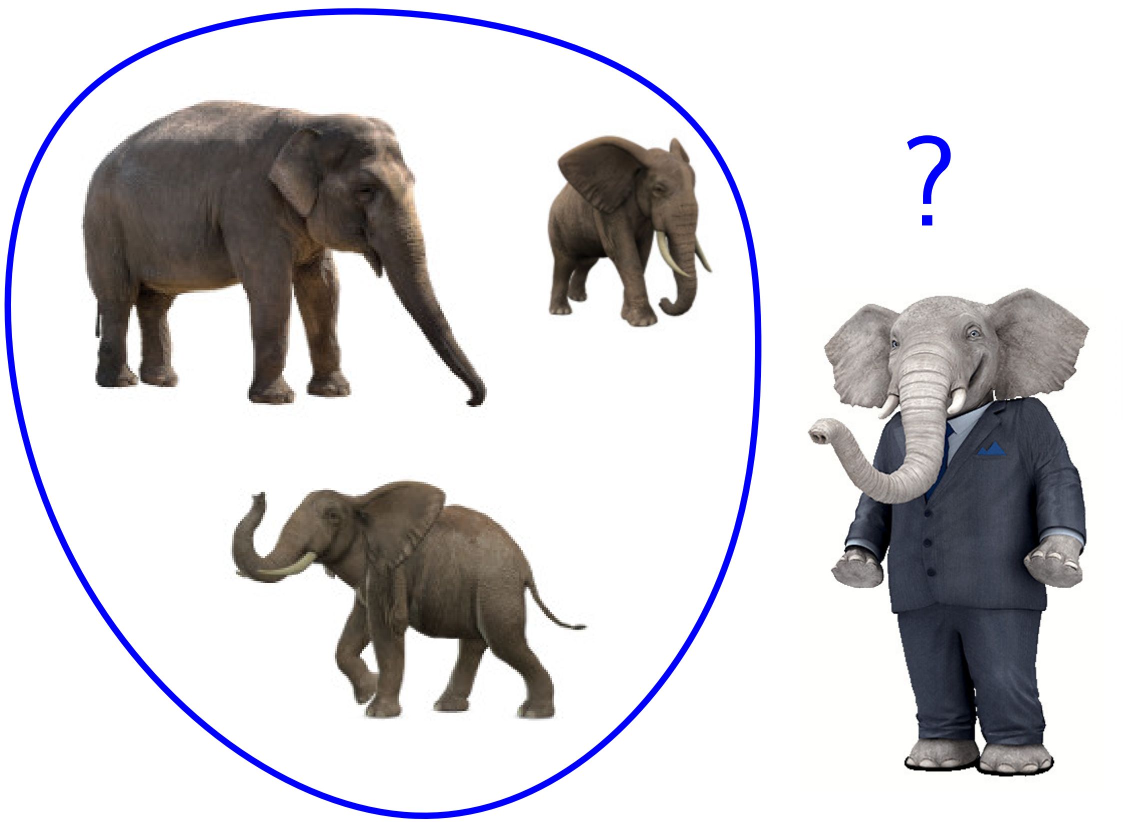
What is Clustering?

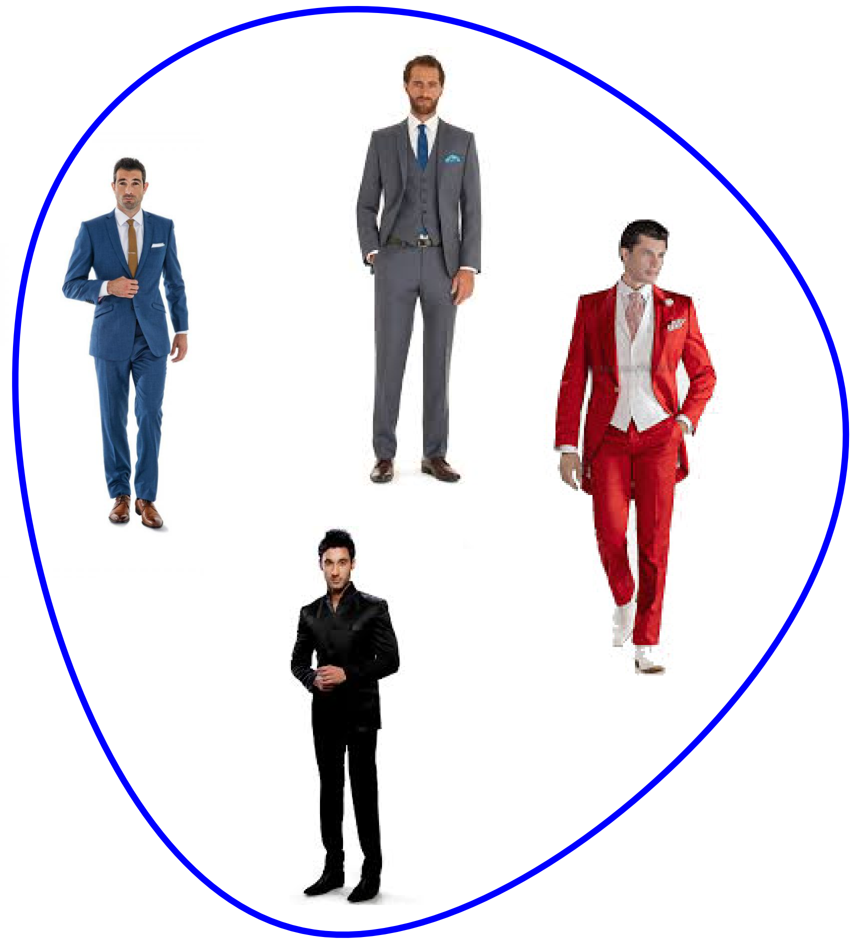
Clustering is the task of grouping similar objects into clusters



What is Clustering?

Clustering is the task of grouping similar objects into clusters



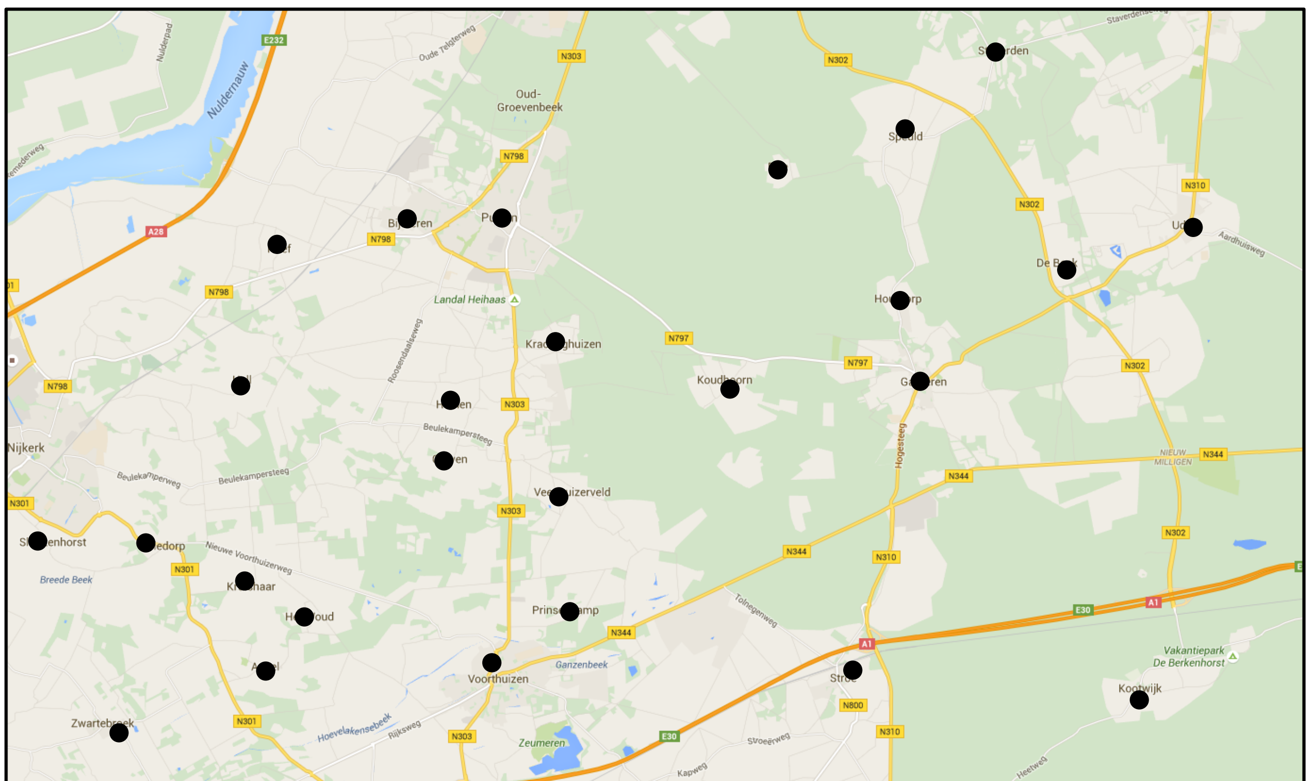


Issues:

- errors and unusual things ("noise")
- what is the "right" clustering?

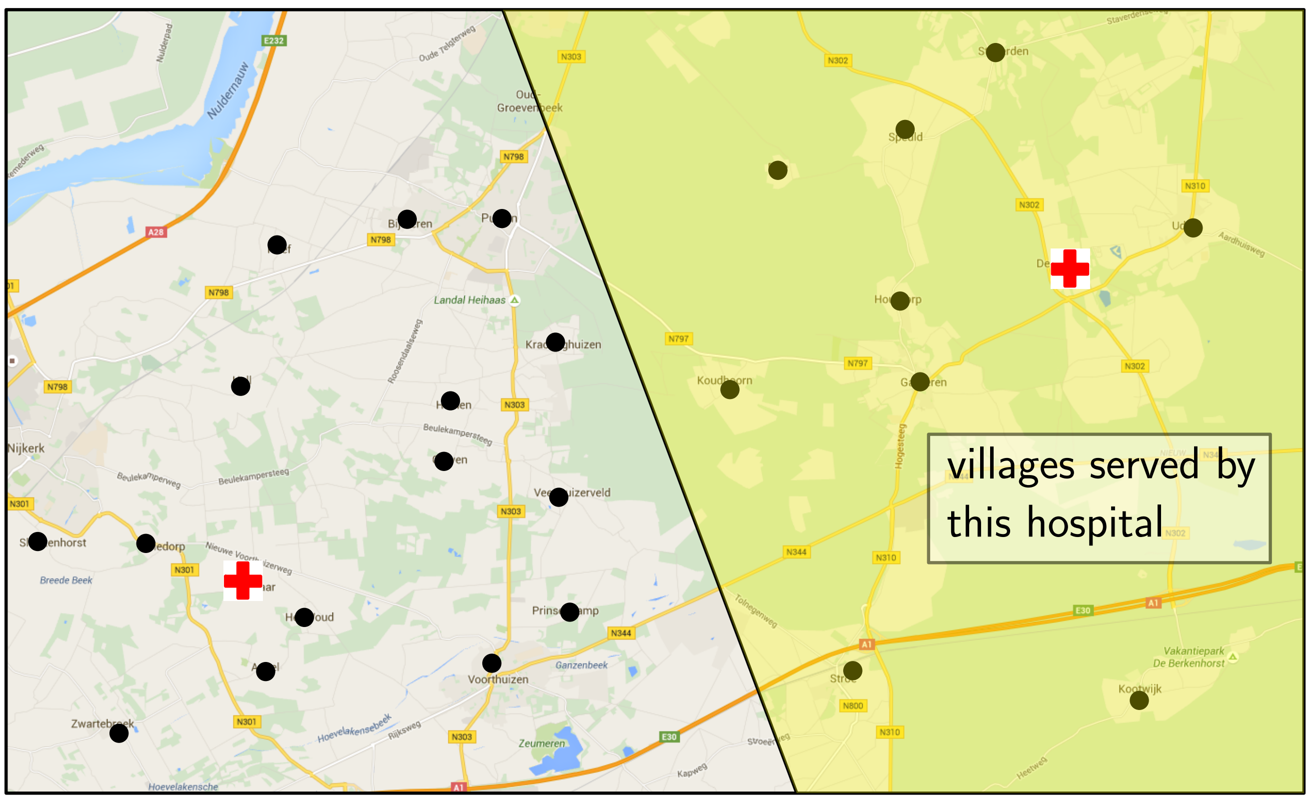
Facility Location

You may build two hospitals in two different villages serving the surrounding villages. Where do you place them to minimize the maximal distance from any village to its serving hospital?



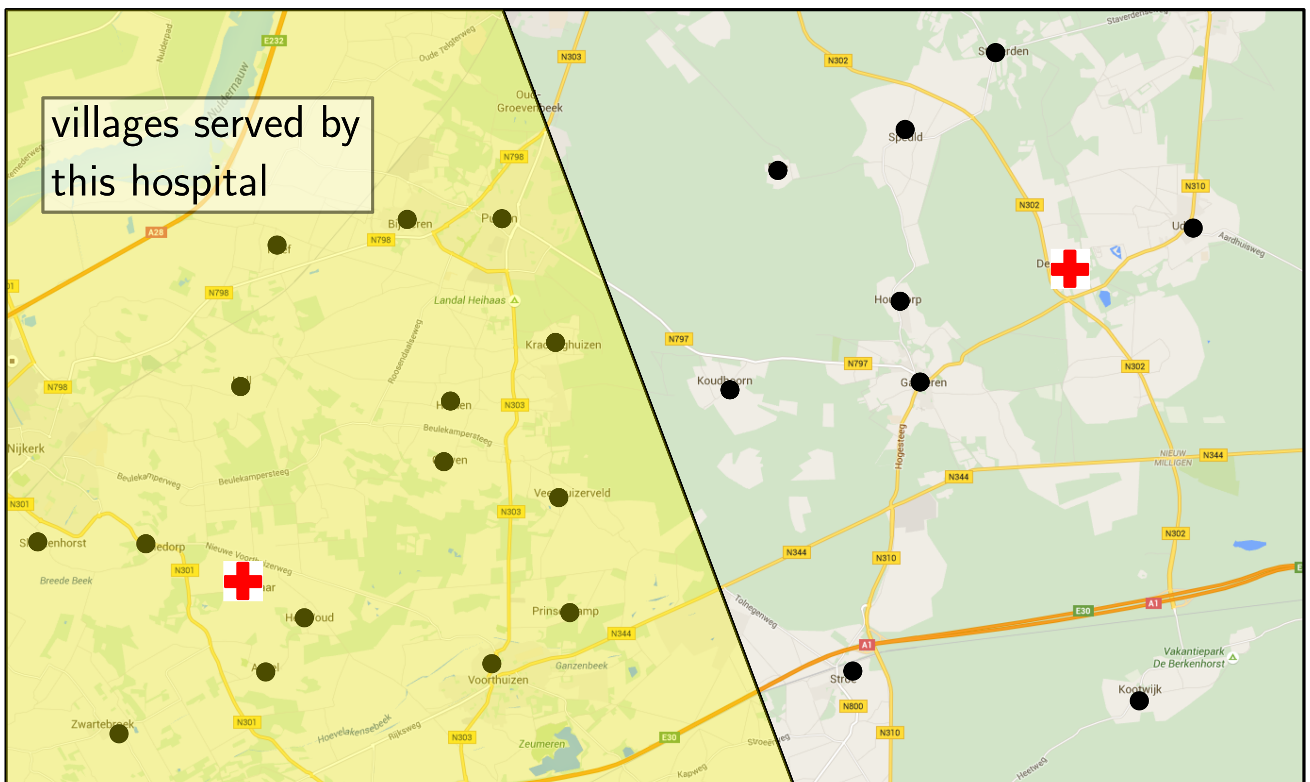
Facility Location

You may build two hospitals in two different villages serving the surrounding villages. Where do you place them to minimize the maximal distance from any village to its serving hospital?



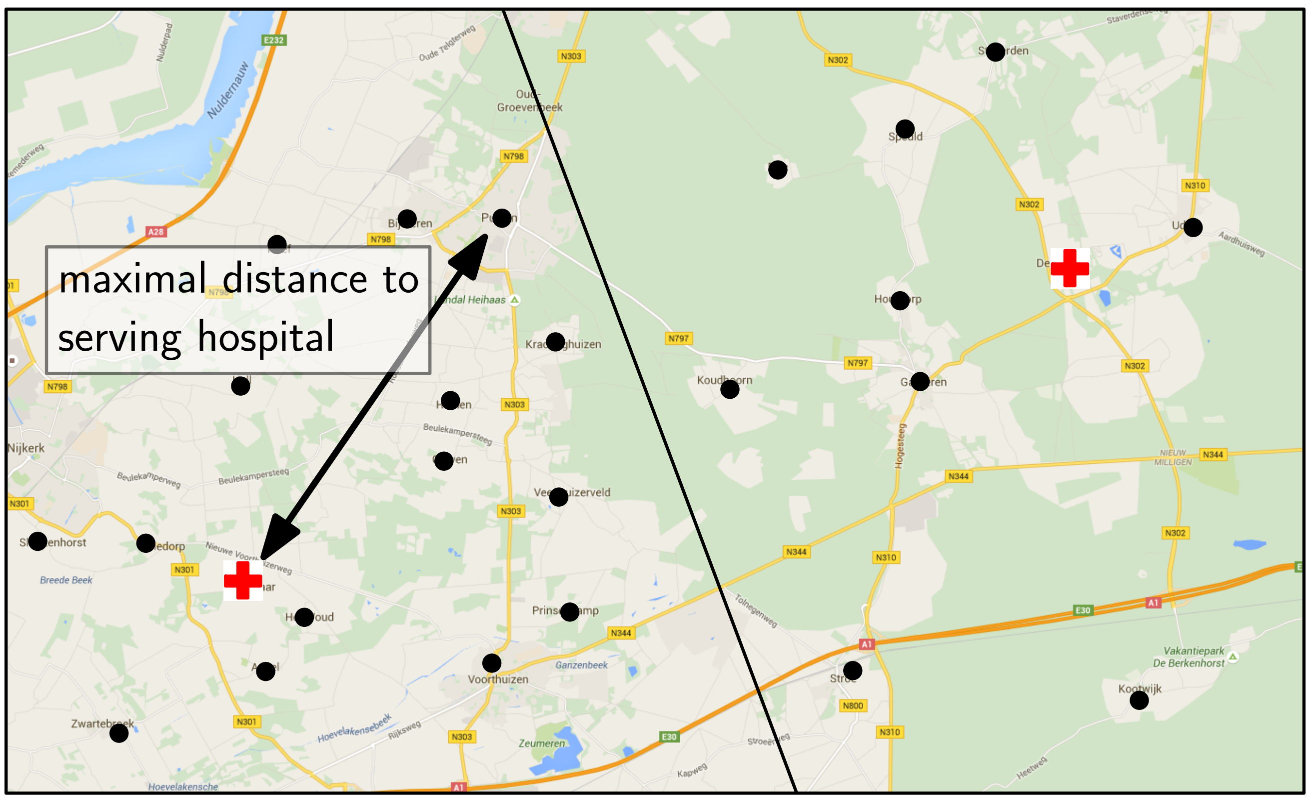
Facility Location

You may build two hospitals in two different villages serving the surrounding villages. Where do you place them to minimize the maximal distance from any village to its serving hospital?



Facility Location

You may build two hospitals in two different villages serving the surrounding villages. Where do you place them to minimize the maximal distance from any village to its serving hospital?



..more formally (k-center problem)

Input: set of points $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^d$, value of k
Output: set of centers $C = \{c_1, \ldots, c_k\} \subseteq P$

Problem:

- each $p_i \in P$ is "served by" its closest center

$$\operatorname*{argmin}_{c_j \in C} \|p_i - c_j\|$$

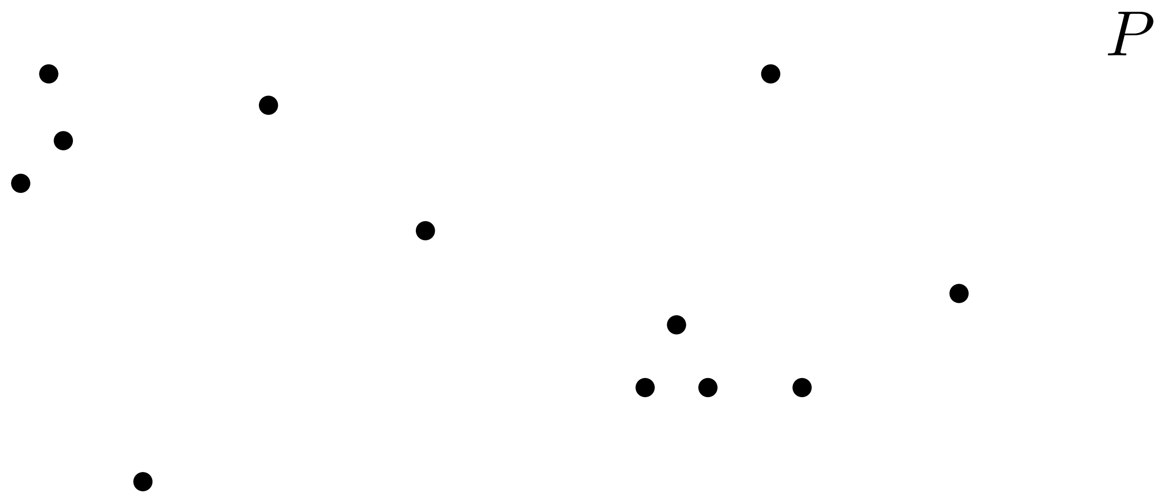
- all points served by a center c_j together form a "cluster"
- we want to choose $\{c_1, \ldots, c_k\}$ to minimize the cost function

$$\phi(P, C) = \max_{p_i \in P} \left\| p_i - \operatorname*{argmin}_{c_j \in C} \|p_i - c_j\| \right\|$$

Gonzales' algorithm

Input: set of points $P = \{p_1, \dots, p_n\} \subseteq \mathbb{R}^d$, value of k
Output: set of centers $C = \{c_1, \dots, c_k\} \subseteq P$

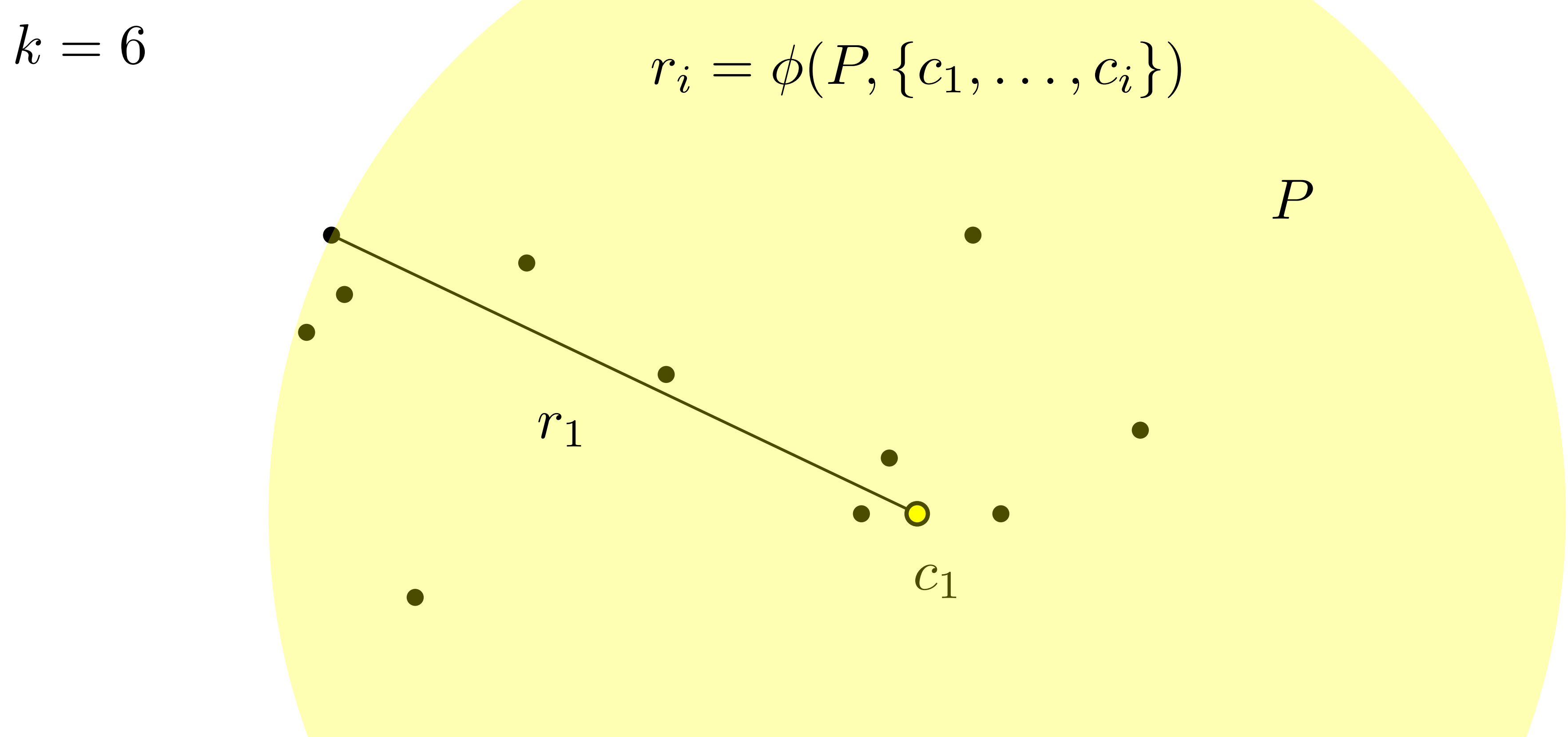
$k = 6$



"Farthest-first" greedy algorithm: always choose next center c_{i+1} as the point that maximizes the current cost r_i

Gonzales' algorithm

Input: set of points $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^d$, value of k
Output: set of centers $C = \{c_1, \ldots, c_k\} \subseteq P$



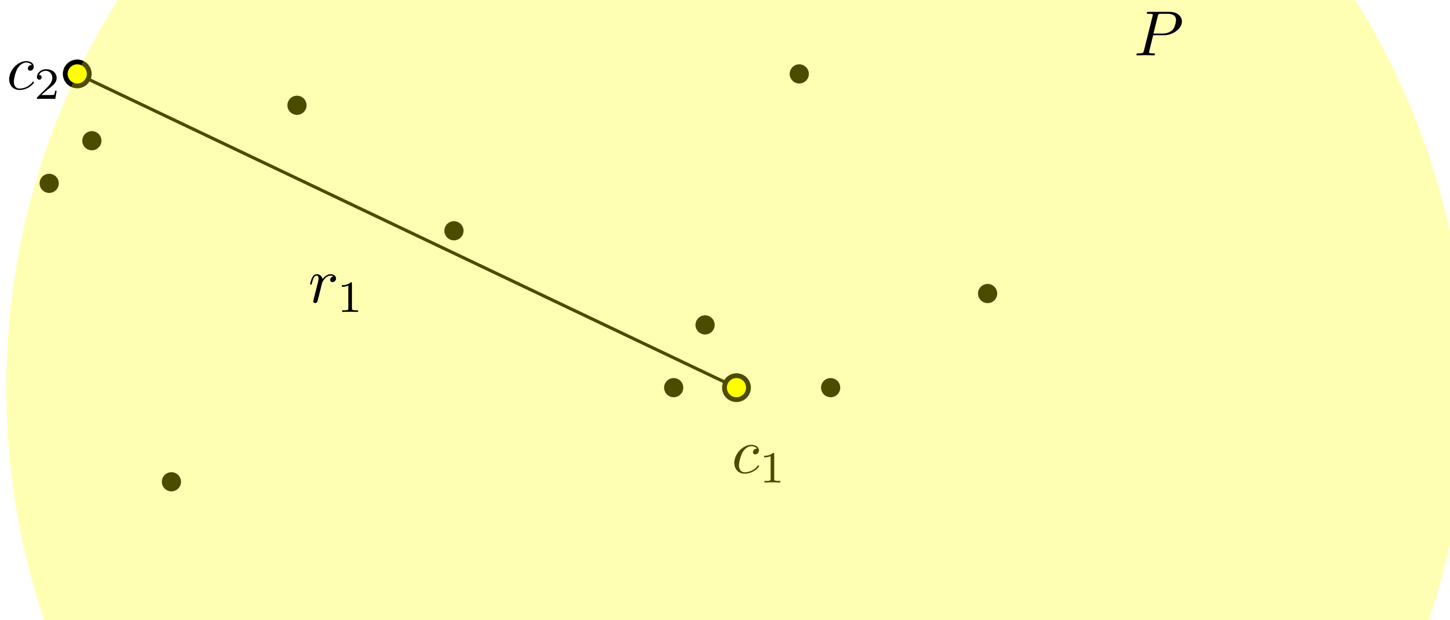
"Farthest-first" greedy algorithm: always choose next center c_{i+1} as the point that maximizes the current cost r_i

Gonzales' algorithm

Input: set of points $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^d$, value of k
Output: set of centers $C = \{c_1, \ldots, c_k\} \subseteq P$

$k = 6$

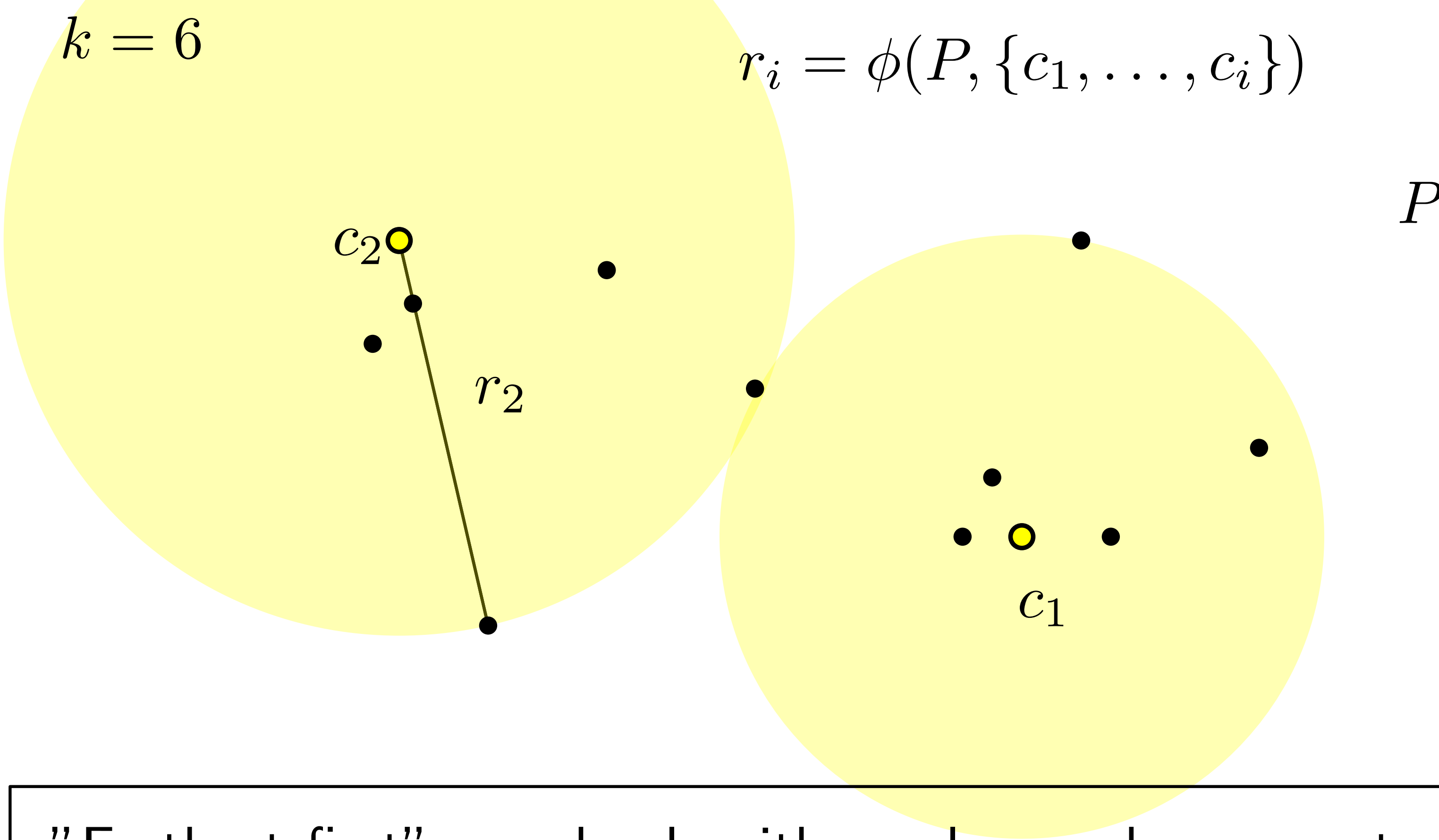
$r_i = \phi(P, \{c_1, \ldots, c_i\})$



"Farthest-first" greedy algorithm: always choose next center c_{i+1} as the point that maximizes the current cost r_i

Gonzales' algorithm

Input: set of points $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^d$, value of k
Output: set of centers $C = \{c_1, \ldots, c_k\} \subseteq P$



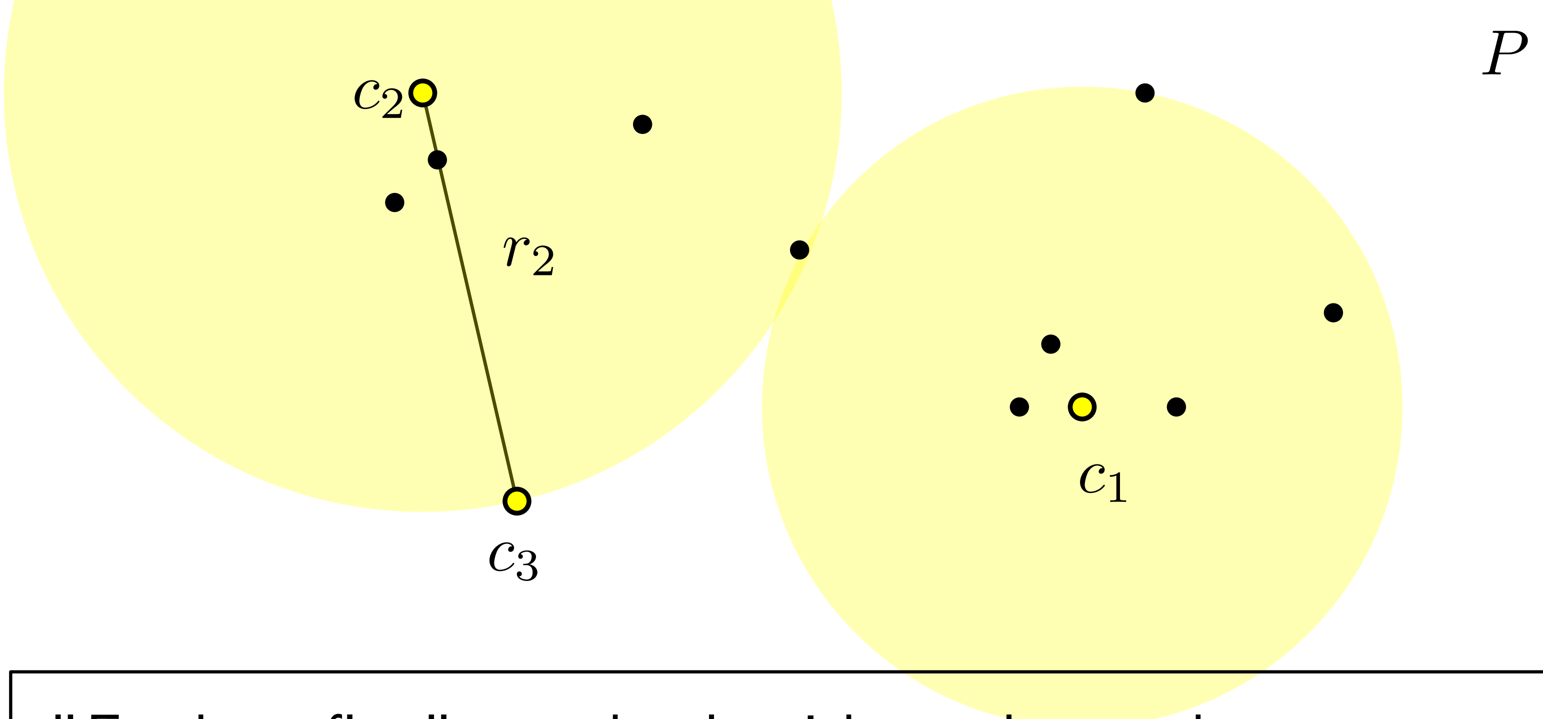
"Farthest-first" greedy algorithm: always choose next center c_{i+1} as the point that maximizes the current cost r_i

Gonzales' algorithm

Input: set of points $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^d$, value of k
Output: set of centers $C = \{c_1, \ldots, c_k\} \subseteq P$

$k = 6$

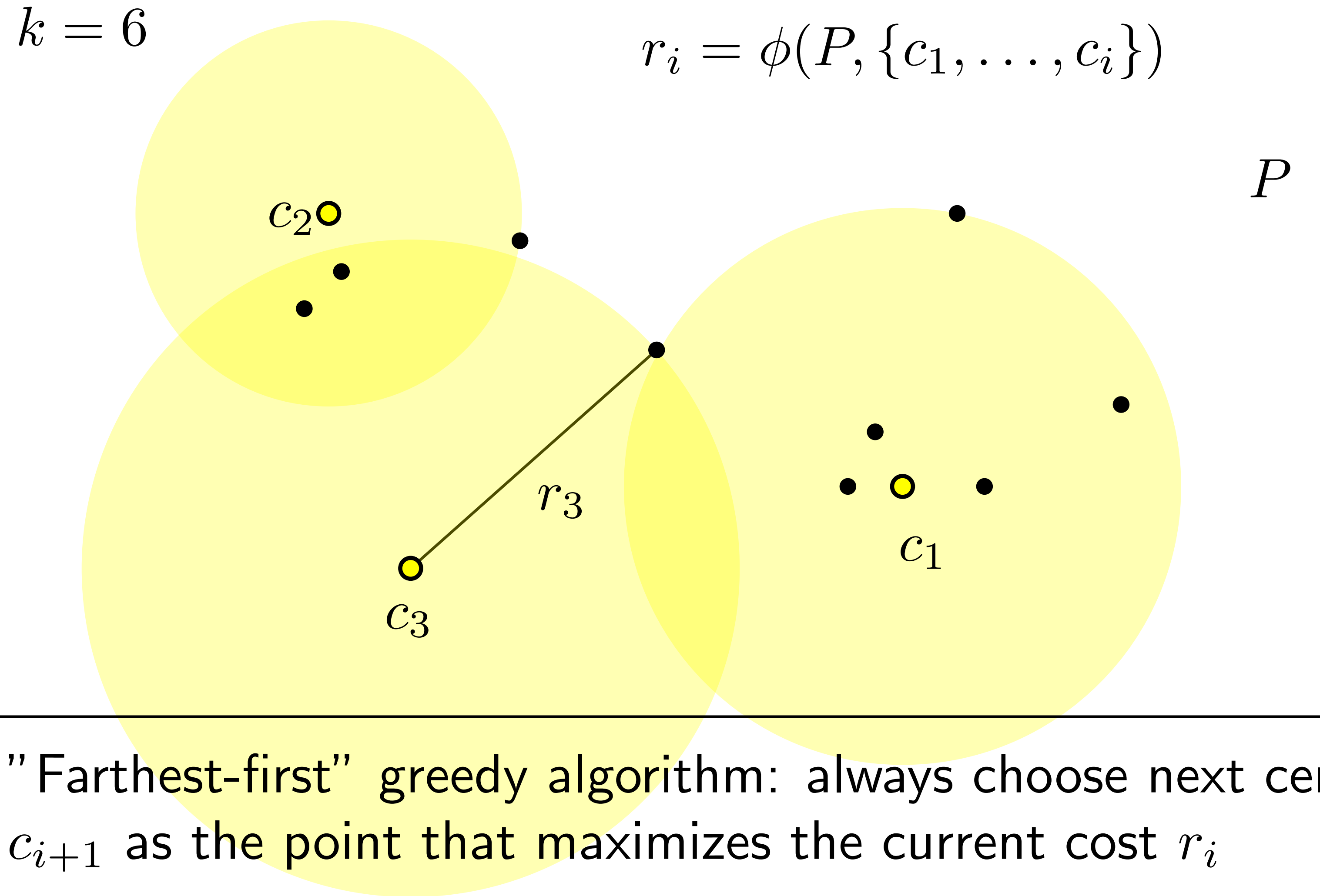
$r_i = \phi(P, \{c_1, \ldots, c_i\})$



"Farthest-first" greedy algorithm: always choose next center c_{i+1} as the point that maximizes the current cost r_i

Gonzales' algorithm

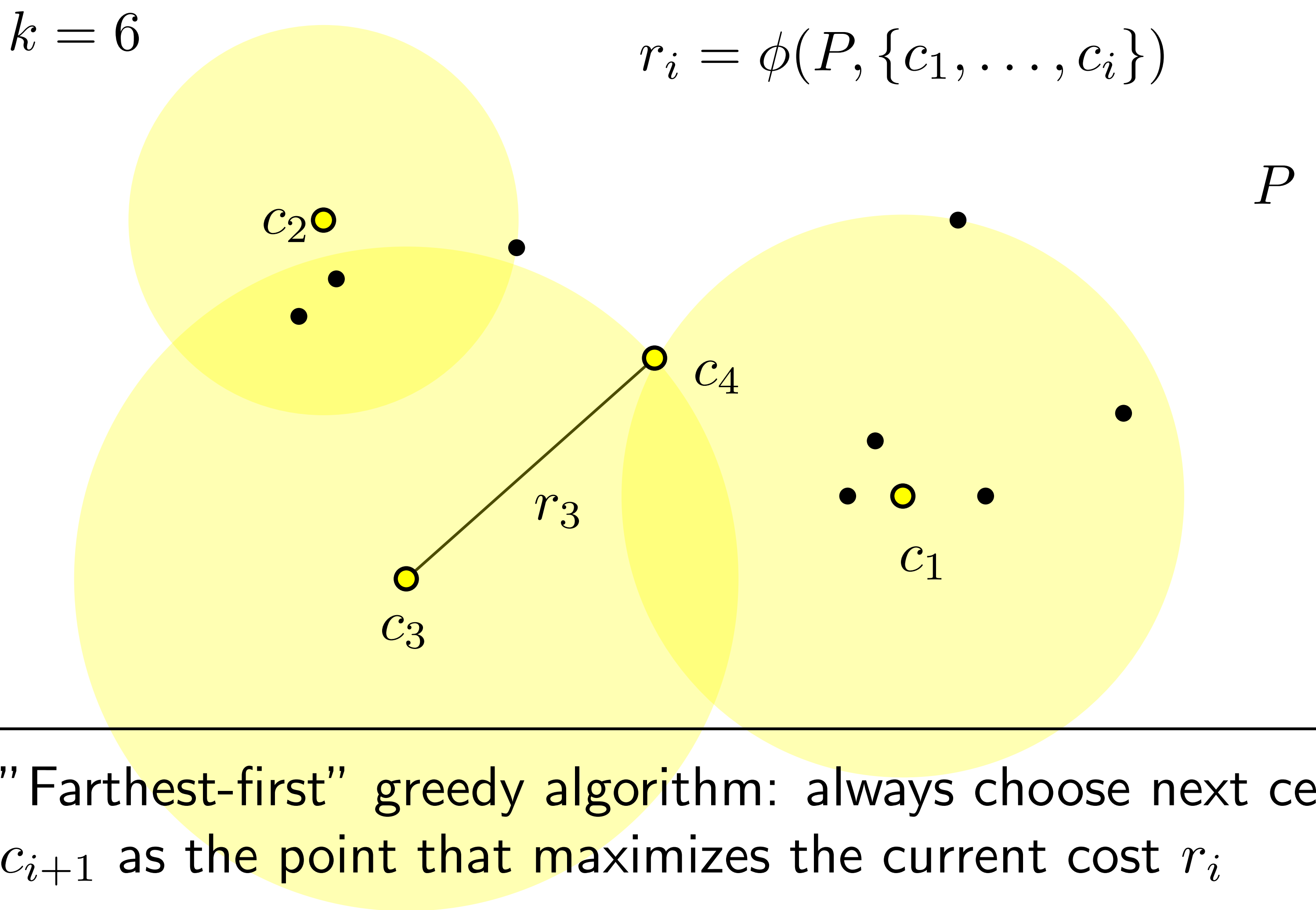
Input: set of points $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^d$, value of k
Output: set of centers $C = \{c_1, \ldots, c_k\} \subseteq P$



"Farthest-first" greedy algorithm: always choose next center c_{i+1} as the point that maximizes the current cost r_i

Gonzales' algorithm

Input: set of points $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^d$, value of k
Output: set of centers $C = \{c_1, \ldots, c_k\} \subseteq P$



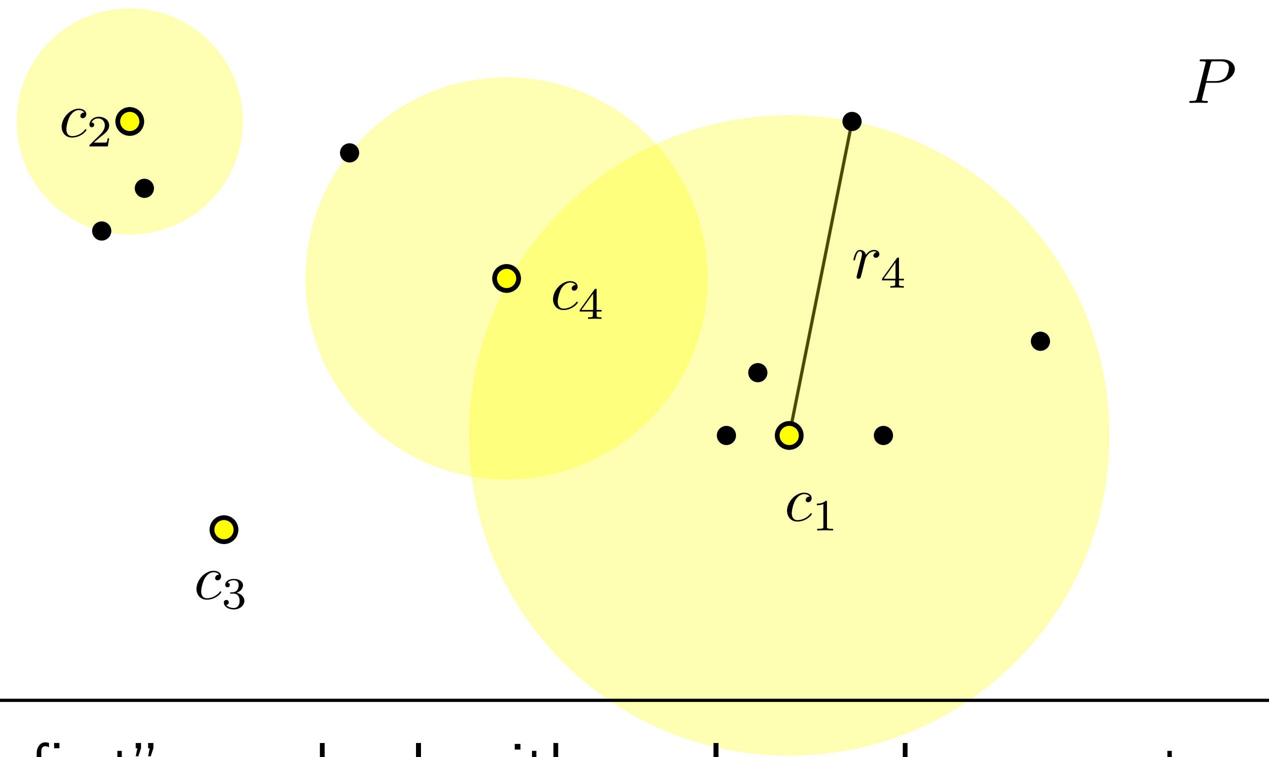
"Farthest-first" greedy algorithm: always choose next center c_{i+1} as the point that maximizes the current cost r_i

Gonzales' algorithm

Input: set of points $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^d$, value of k
Output: set of centers $C = \{c_1, \ldots, c_k\} \subseteq P$

$k = 6$

$r_i = \phi(P, \{c_1, \ldots, c_i\})$



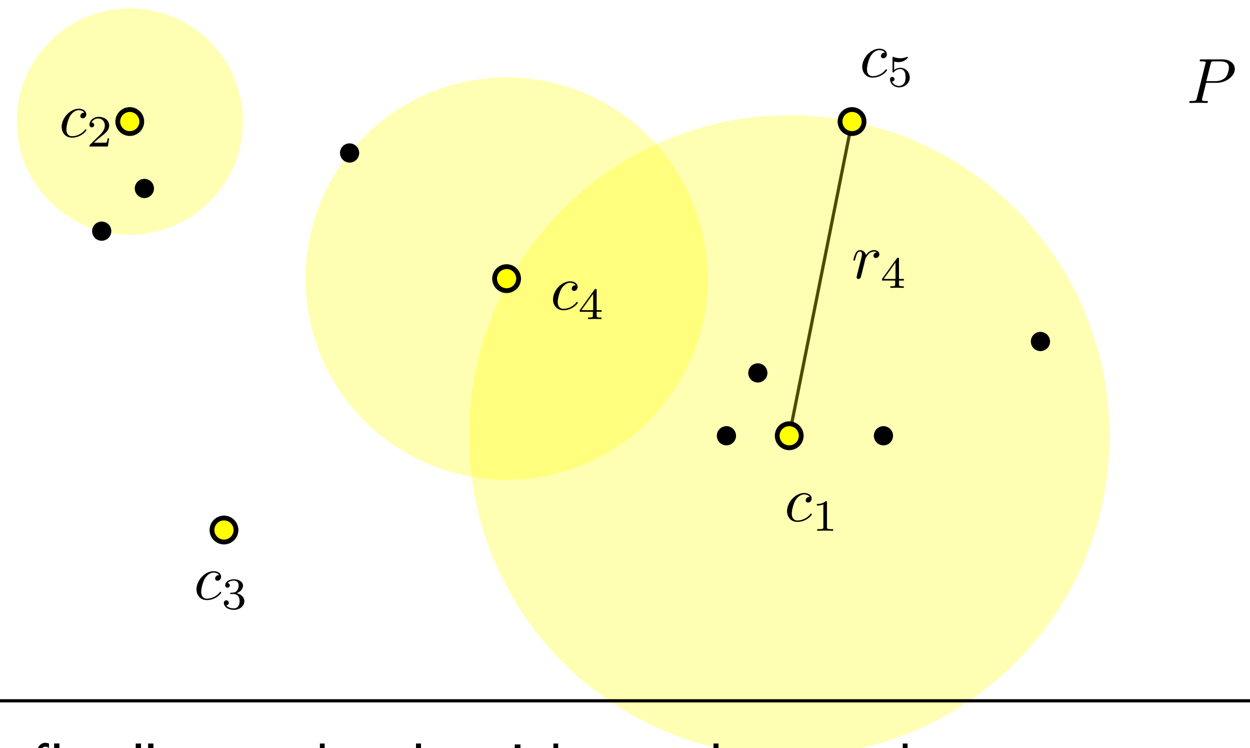
"Farthest-first" greedy algorithm: always choose next center c_{i+1} as the point that maximizes the current cost r_i

Gonzales' algorithm

Input: set of points $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^d$, value of k
Output: set of centers $C = \{c_1, \ldots, c_k\} \subseteq P$

$k = 6$

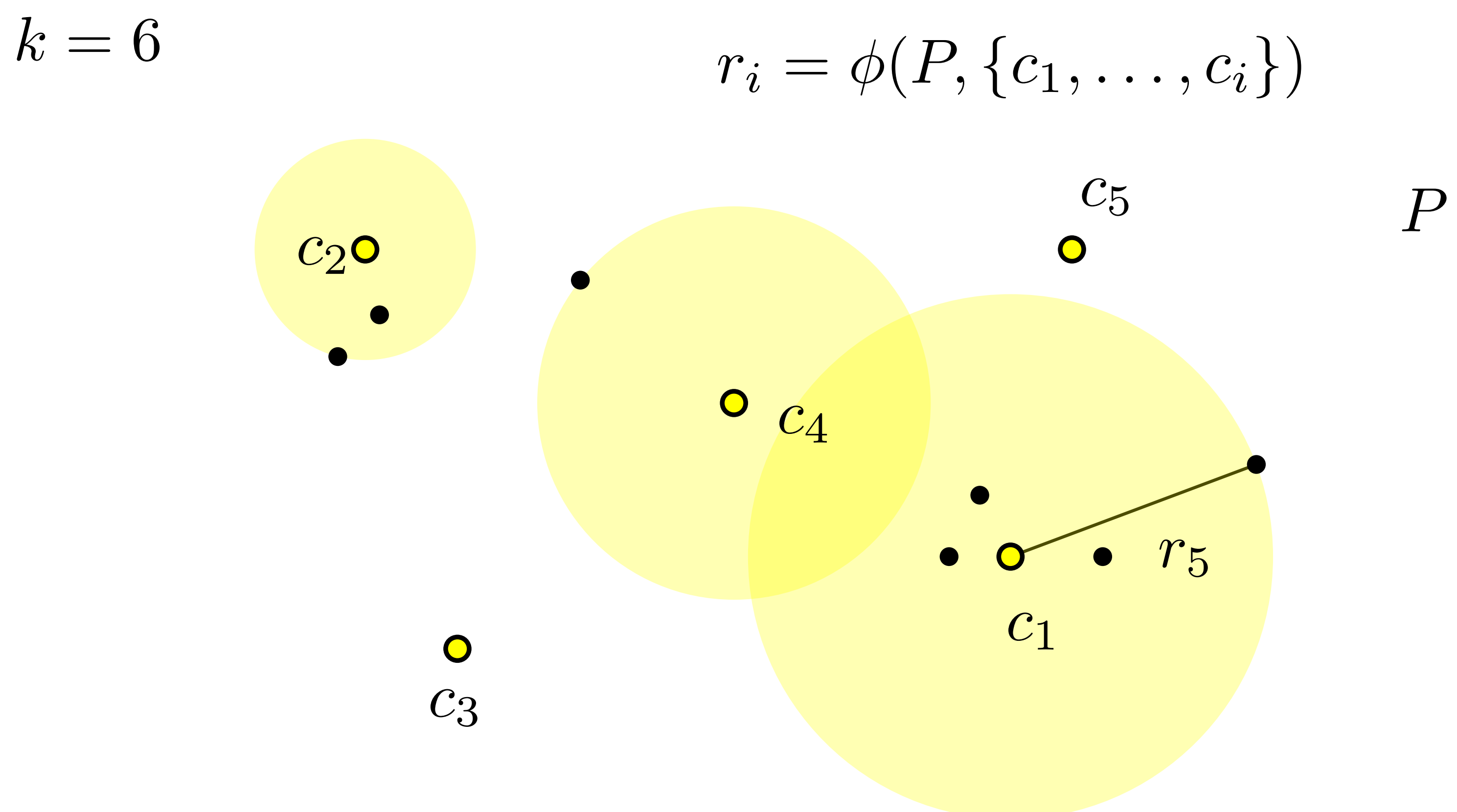
$r_i = \phi(P, \{c_1, \ldots, c_i\})$



"Farthest-first" greedy algorithm: always choose next center c_{i+1} as the point that maximizes the current cost r_i

Gonzales' algorithm

Input: set of points $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^d$, value of k
Output: set of centers $C = \{c_1, \ldots, c_k\} \subseteq P$



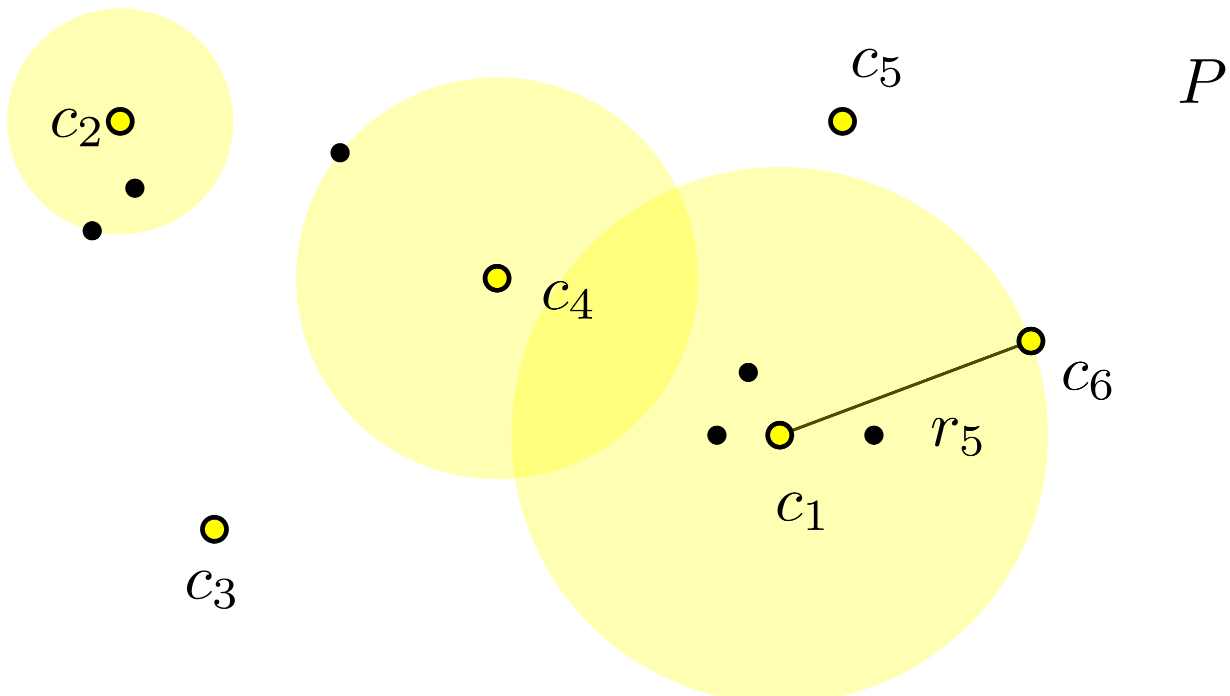
"Farthest-first" greedy algorithm: always choose next center c_{i+1} as the point that maximizes the current cost r_i

Gonzales' algorithm

Input: set of points $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^d$, value of k
Output: set of centers $C = \{c_1, \ldots, c_k\} \subseteq P$

$k = 6$

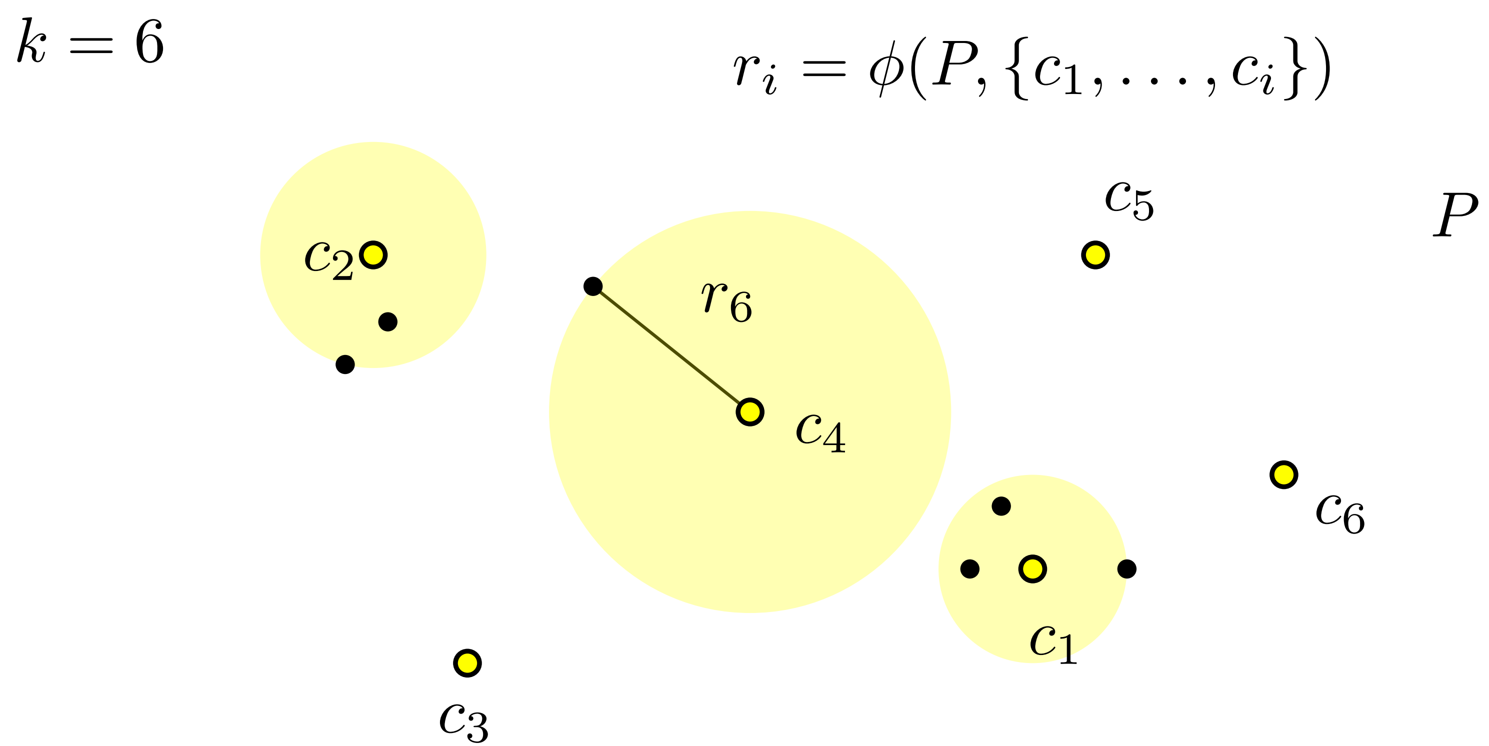
$r_i = \phi(P, \{c_1, \ldots, c_i\})$



"Farthest-first" greedy algorithm: always choose next center c_{i+1} as the point that maximizes the current cost r_i

Gonzales' algorithm

Input: set of points $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^d$, value of k
Output: set of centers $C = \{c_1, \ldots, c_k\} \subseteq P$



"Farthest-first" greedy algorithm: always choose next center c_{i+1} as the point that maximizes the current cost r_i

Gonzales' algorithm

Input: set of points $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^d$, value of k
Output: set of centers $C = \{c_1, \ldots, c_k\} \subseteq P$

Objective:

minimize $\quad \phi(P, C) = \max_{p_i \in P} \|p_i - \operatorname{argmin}_{c_j \in C} \|p_i - c_j\|\|$

Algorithm:

- choose an arbitrary point $p_i \in P$ and set $c_1 = p_i$
- for $t = 1, \ldots, k-1$ set

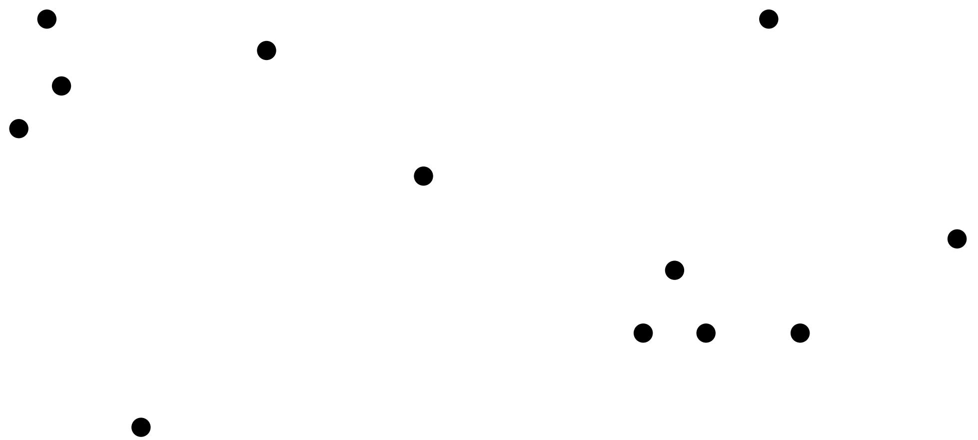
$$c_{t+1} = \operatorname*{argmax}_{p_i \in P} \left\| p_i - \operatorname*{argmin}_{c_j \in \{c_1, \ldots, c_t\}} \|p_i - c_j\| \right\|$$

"Farthest-first" greedy algorithm: always choose next center c_{i+1} as the point that maximizes the current cost r_i

Gonzales' algorithm (Analysis)

Claim: $r_k \leq 2\phi(P, C^*)$ (where C^* is an optimal solution)

Proof:

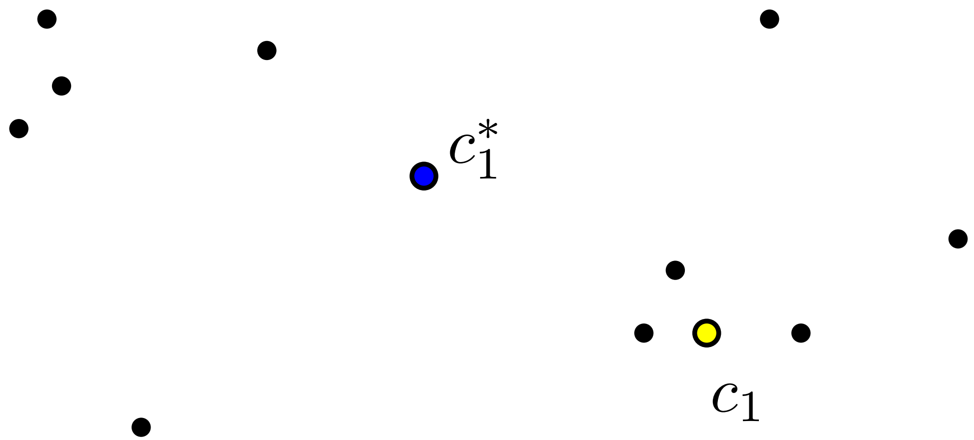


Gonzales' algorithm (Analysis)

Claim: $r_k \leq 2\phi(P, C^*)$ (where C^* is an optimal solution)

Proof:

$(k = 1)$

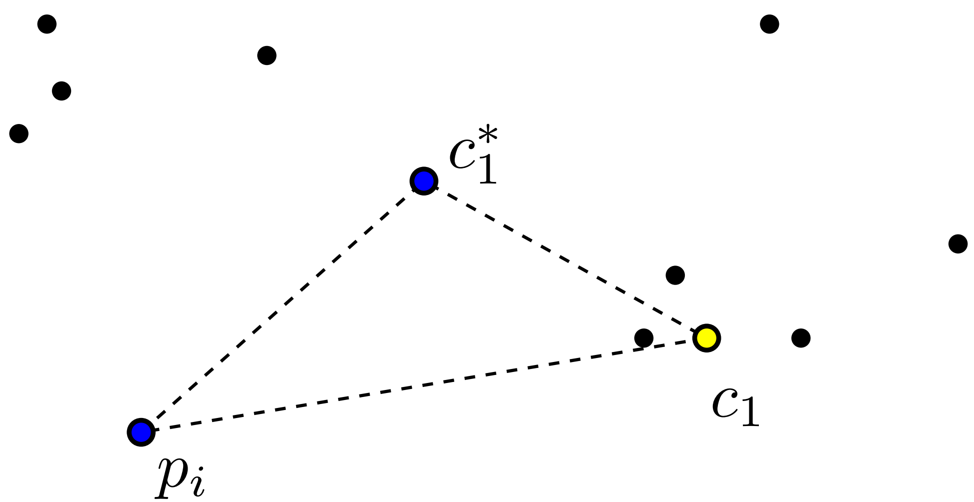


Gonzales' algorithm (Analysis)

Claim: $r_k \leq 2\phi(P, C^*)$ (where C^* is an optimal solution)

Proof:

$(k = 1)$



(Triangle inequality)

$\forall p_i \in P :$ $\|p_i - c_1\| \leq \|p_i - c_1^*\| + \|c_1^* - c_1\|$

Gonzales' algorithm (Analysis)

Claim: $r_k \leq 2\phi(P, C^*)$ (where C^* is an optimal solution)

Pro

$(k$

$\forall p_i$

The triangle inequality (Euclidean distance):

$$\forall x, y, z \in \mathbb{R}^d : \|x - y\| \leq \|x - z\| + \|z - y\|$$

Proof:

Let $x = (x_1, \ldots, x_d), y = (y_1, \ldots, y_d), z = (z_1, \ldots, z_d)$

$\forall\ i \in \{1, \ldots, d\}$: $(x_i - y_i)^2 \leq (x_i - z_i)^2 + (z_i - y_i)^2$

$\Rightarrow \sum_{i=1}^{d}(x_i - y_i)^2 \leq \sum_{i=1}^{d}(x_i - z_i)^2 + \sum_{i=1}^{d}(z_i - y_i)^2$

$\Rightarrow \|x - y\|^2 \leq \|x - z\|^2 + \|z - y\|^2$

$\Rightarrow \|x - y\|^2 \leq \|x - z\|^2 + \|z - y\|^2 + 2\|x - z\|\|z - y\|$

$\Rightarrow \|x - y\|^2 \leq (\|x - z\| + \|z - y\|)^2$

$\Rightarrow \|x - y\| \leq \|x - z\| + \|z - y\|$

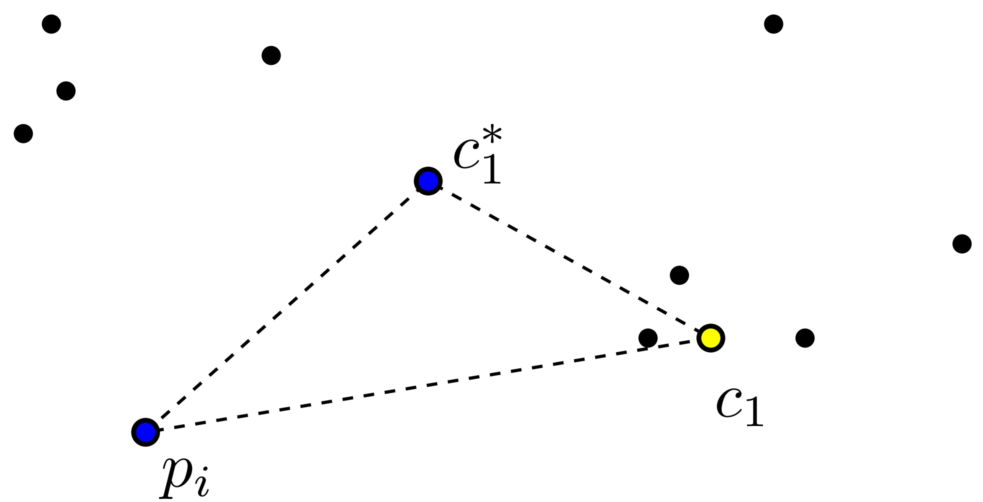
$\square$

Gonzales' algorithm (Analysis)

Claim: $r_k \leq 2\phi(P, C^*)$ (where C^* is an optimal solution)

Proof:

$(k = 1)$



(Triangle inequality)

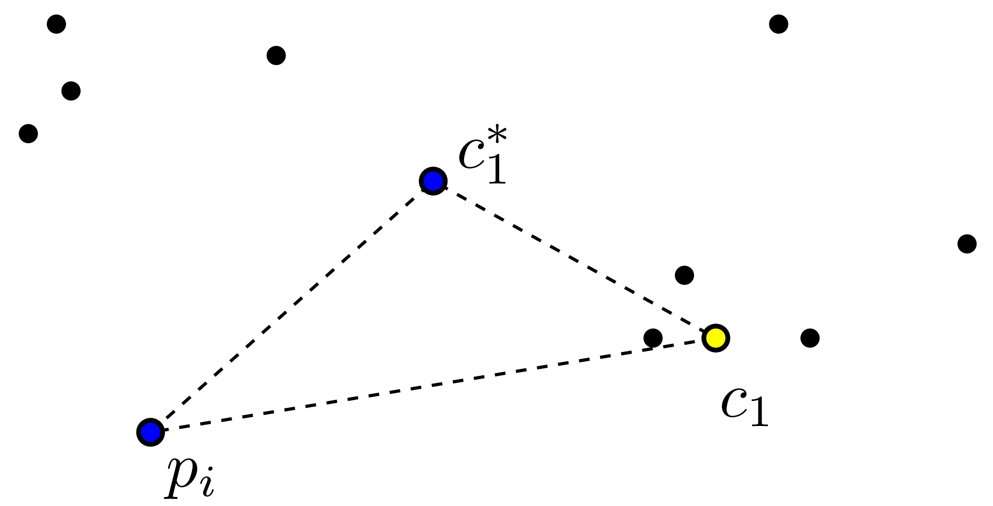
$$\forall p_i \in P : \quad \|p_i - c_1\| \leq \underbrace{\|p_i - c_1^*\|}_{\leq \phi(P, C^*)} + \underbrace{\|c_1^* - c_1\|}_{\leq \phi(P, C^*)}$$

Gonzales' algorithm (Analysis)

Claim: $r_k \leq 2\phi(P, C^*)$ (where C^* is an optimal solution)

Proof:

$(k = 1)$



(Triangle inequality)

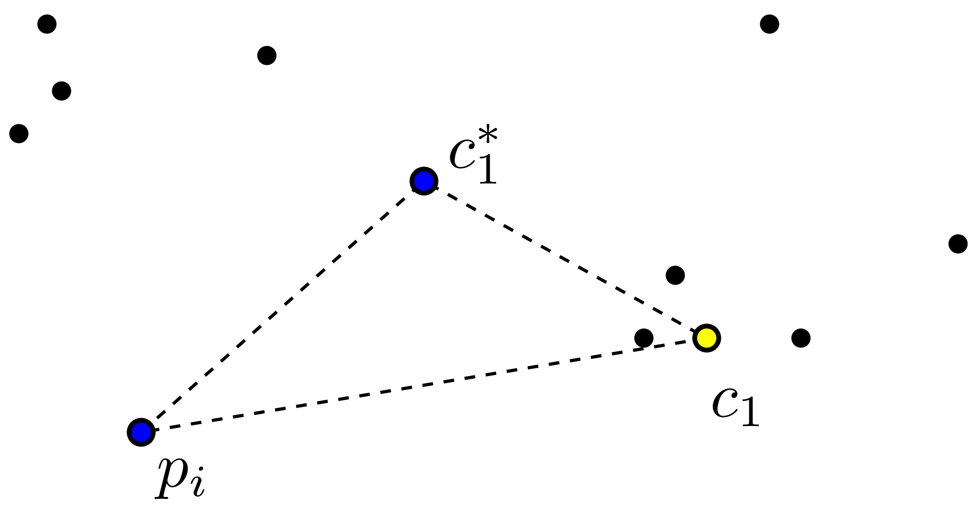
$$\forall p_i \in P : \quad \|p_i - c_1\| \leq \underbrace{\|p_i - c_1^*\|}_{\leq \phi(P, C^*)} + \underbrace{\|c_1^* - c_1\|}_{\leq \phi(P, C^*)} \leq 2\phi(P, C^*)$$

Gonzales' algorithm (Analysis)

Claim: $r_k \leq 2\phi(P, C^*)$ (where C^* is an optimal solution)

Proof:

$(k = 1)$



(Triangle inequality)

$$\forall p_i \in P: \quad \|p_i - c_1\| \leq \underbrace{\|p_i - c_1^*\|}_{\leq \phi(P, C^*)} + \underbrace{\|c_1^* - c_1\|}_{\leq \phi(P, C^*)} \leq 2\phi(P, C^*)$$

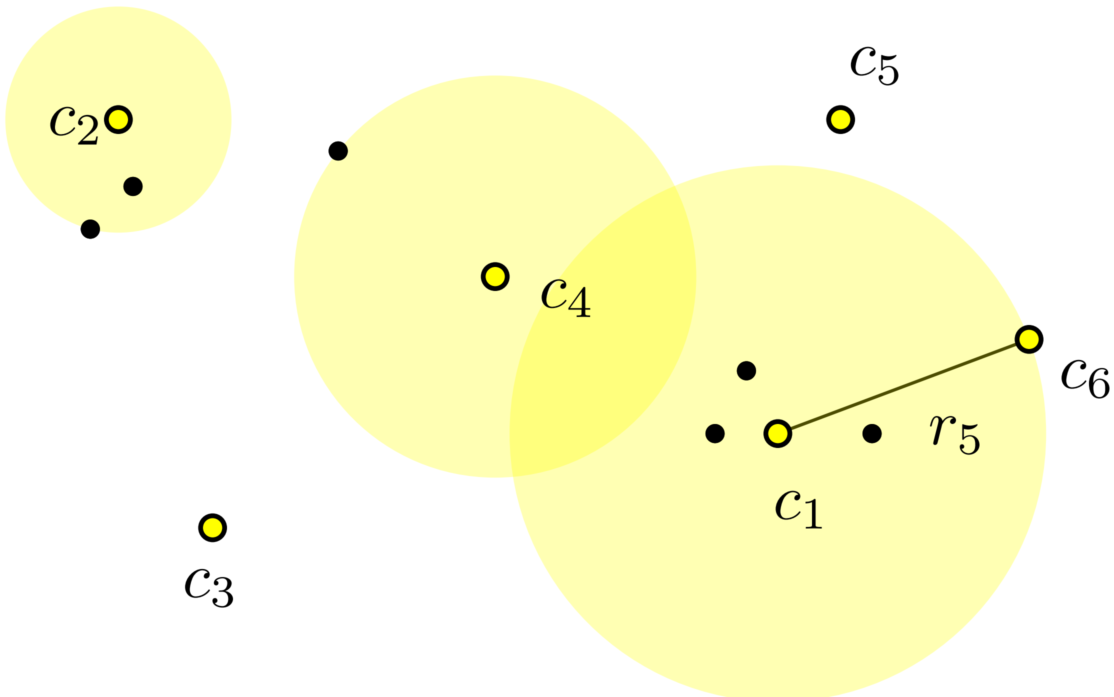
$$\Rightarrow r_1 \leq 2\phi(P, C^*)$$

Gonzales' algorithm (Analysis)

Claim: $r_k \leq 2\phi(P, C^*)$ (where C^* is an optimal solution)

Proof:

$(k \geq 1)$

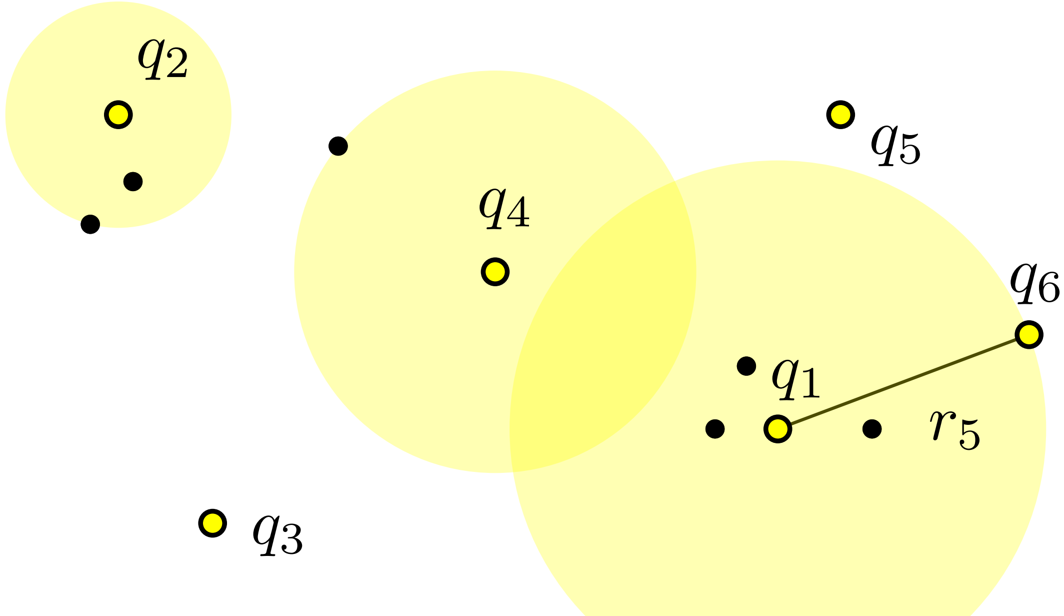


Gonzales' algorithm (Analysis)

Claim: $r_k \leq 2\phi(P, C^*)$ (where C^* is an optimal solution)

Proof:

$(k \geq 1)$



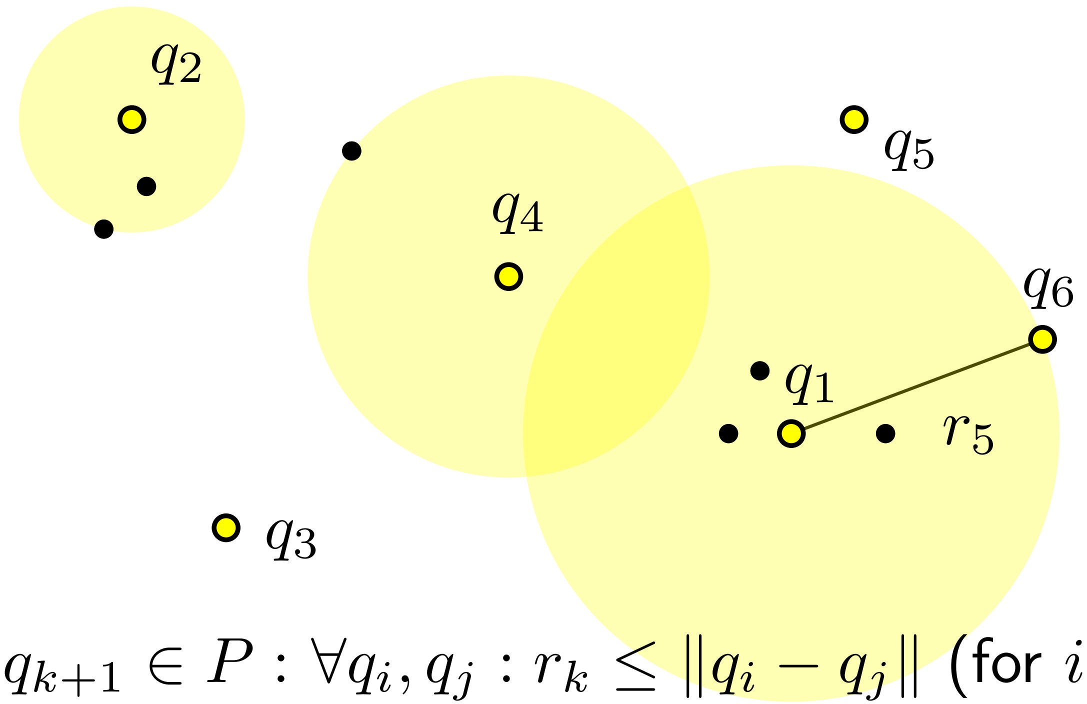
$\exists q_1, \ldots, q_k, q_{k+1} \in P : \forall q_i, q_j : r_k \leq \|q_i - q_j\|$ (for $i \neq j$)

Gonzales' algorithm (Analysis)

Claim: $r_k \leq 2\phi(P, C^*)$ (where C^* is an optimal solution)

Proof:

$(k \geq 1)$



$\exists q_1, \ldots, q_k, q_{k+1} \in P : \forall q_i, q_j : r_k \leq \|q_i - q_j\|$ (for $i \neq j$)

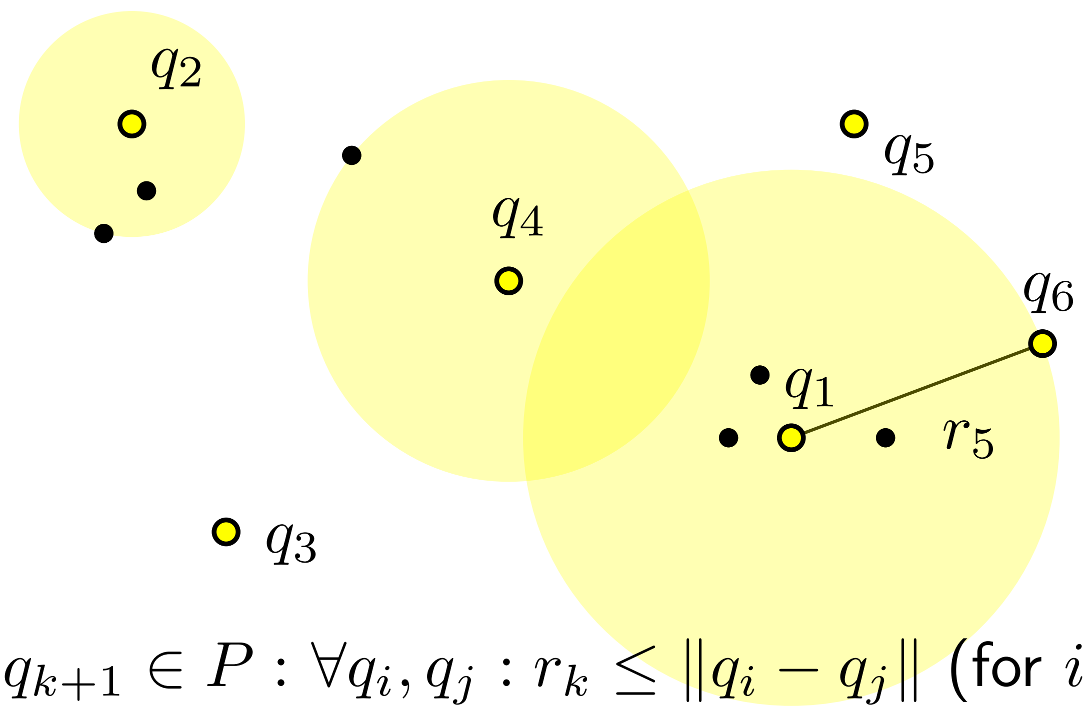
$\Rightarrow$ by the pidgeon hole principle: $\exists q_i, q_j$ served by the same c_s^*

Gonzales' algorithm (Analysis)

Claim: $r_k \leq 2\phi(P, C^*)$ (where C^* is an optimal solution)

Proof:

$(k \geq 1)$



$\exists q_1, \ldots, q_k, q_{k+1} \in P : \forall q_i, q_j : r_k \leq \|q_i - q_j\|$ (for $i \neq j$)

$\Rightarrow$ by the pidgeon hole principle: $\exists q_i, q_j$ served by the same c_s^*

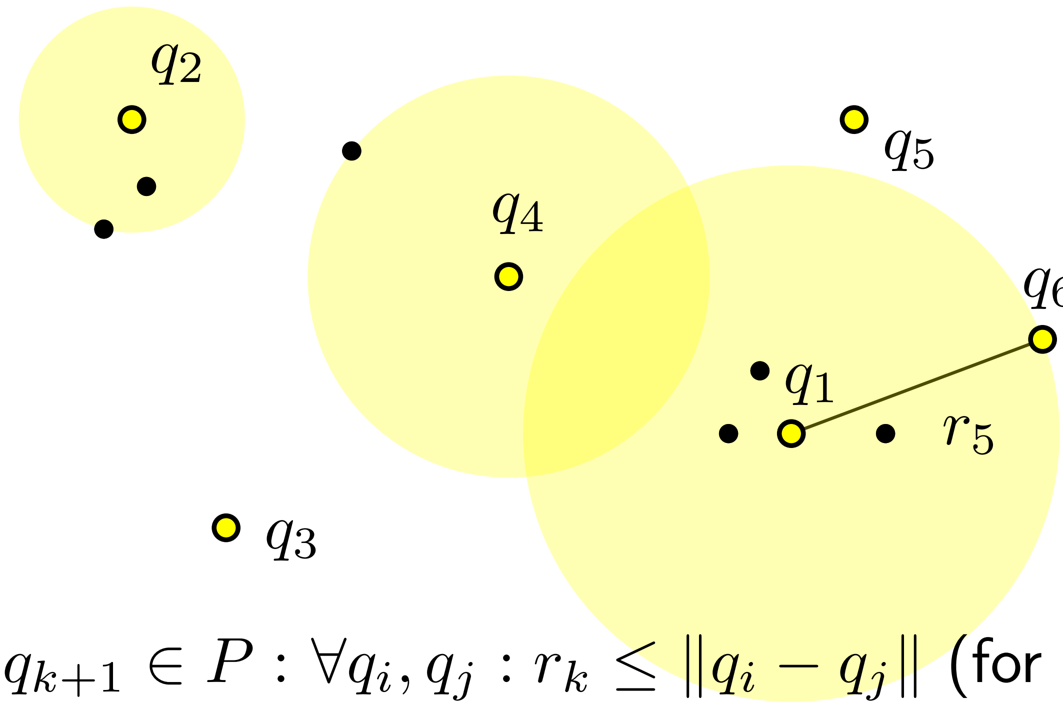
$\Rightarrow$ by the triangle inequality for q_i, q_j, c_s^*: $\|q_i - q_j\| \leq 2\phi(P, C^*)$ (see also case $k = 1$)

Gonzales' algorithm (Analysis)

Claim: $r_k \leq 2\phi(P, C^*)$ (where C^* is an optimal solution)

Proof:

$(k \geq 1)$



$\exists q_1, \ldots, q_k, q_{k+1} \in P : \forall q_i, q_j : r_k \leq \|q_i - q_j\|$ (for $i \neq j$)

$\Rightarrow$ by the pidgeon hole principle: $\exists q_i, q_j$ served by the same c_s^*

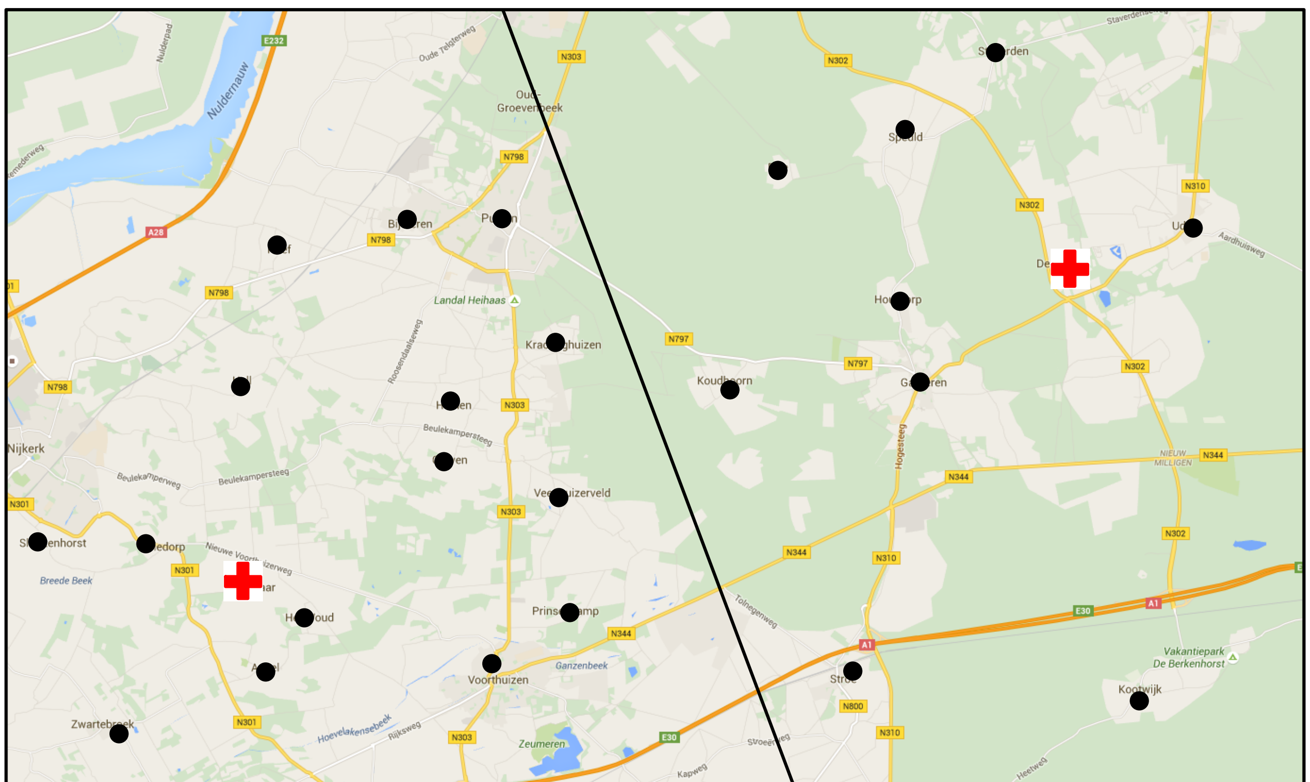
$\Rightarrow$ by the triangle inequality for q_i, q_j, c_s^*: $\|q_i - q_j\| \leq 2\phi(P, C^*)$ (see also case $k = 1$)

So we have $r_k \leq \|p_i - p_j\| \leq 2\phi(P, C^*)$

Facility Location (Variant)

You may build two hospitals in two different villages serving the surrounding villages. Where do you place them to minimize the maximal distance from any village to its serving hospital?

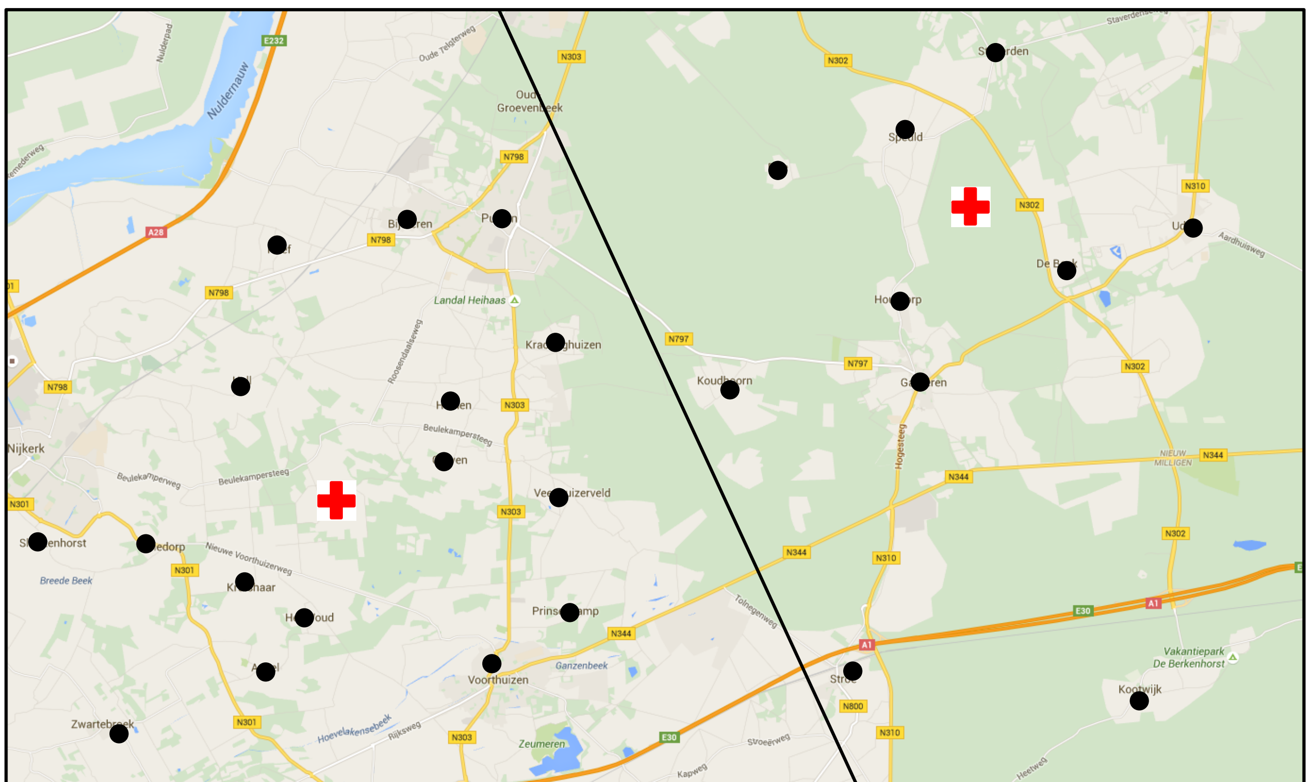
Variant: • minimize the (squared) average distance



Facility Location (Variant)

You may build two hospitals in two different villages serving the surrounding villages. Where do you place them to minimize the maximal distance from any village to its serving hospital?

Variant:
- minimize the (squared) average distance
- hospitals may be built "in the middle of nowhere"



. . . more formally (k-means problem)

Input: set of points $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^d$, value of k
Output: set of centers $C = \{c_1, \ldots, c_k\} \subseteq \mathbb{R}^d$

centers may be in the middle of nowhere

Problem:

- each $p_i \in P$ is associated with its closest center

$$\operatorname*{argmin}_{c_j \in C} \|p_i - c_j\|$$

- points associated with a center c_j together form a "cluster".

- we want to choose $\{c_1, \ldots, c_k\}$ to minimize the cost function

average instead of maximum

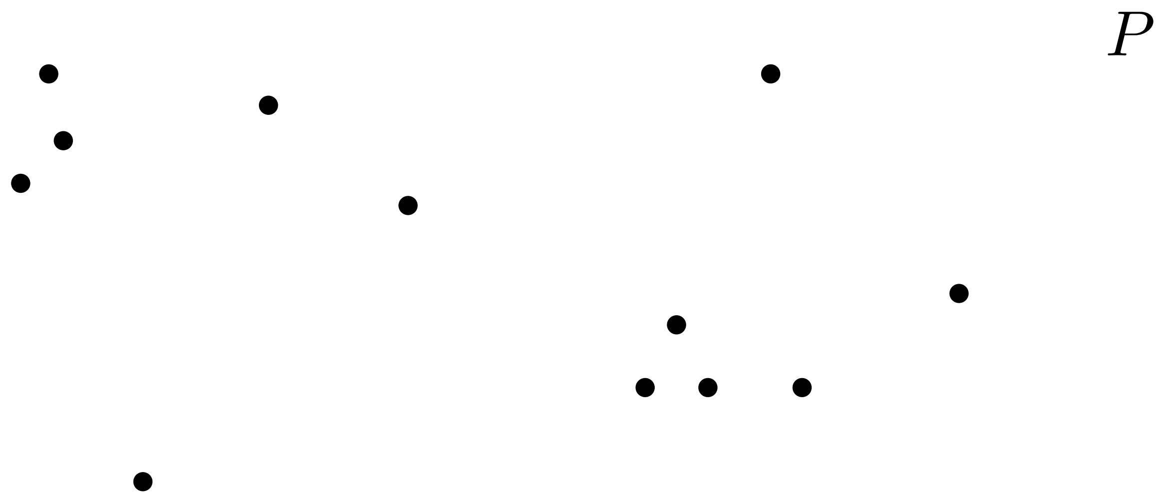
$$\phi(P, C) = \sum_{p_i \in P} \left\| p_i - \operatorname*{argmin}_{c_j \in C} \|p_i - c_j\| \right\|^2$$

(squared distance)

Lloyd's algorithm (k-means problem)

Input: set of points $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^d$, value of k
Output: set of centers $C = \{c_1, \ldots, c_k\} \subseteq \mathbb{R}^d$

$k = 3$

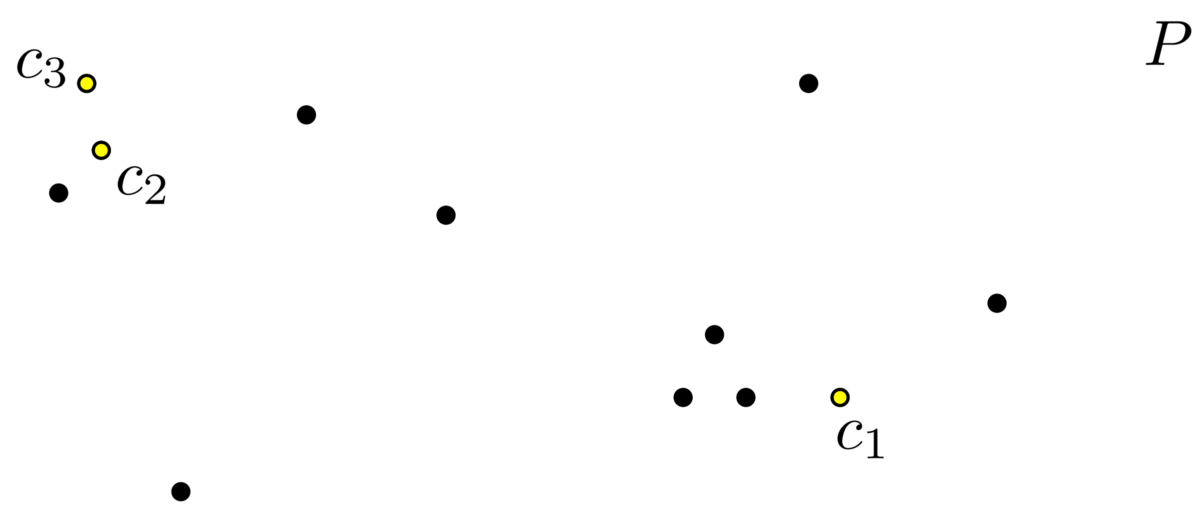


Iterative refinement with two steps:
(step 1) update assignment (step 2) update centers

Lloyd's algorithm (k-means problem)

Input: set of points $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^d$, value of k
Output: set of centers $C = \{c_1, \ldots, c_k\} \subseteq \mathbb{R}^d$

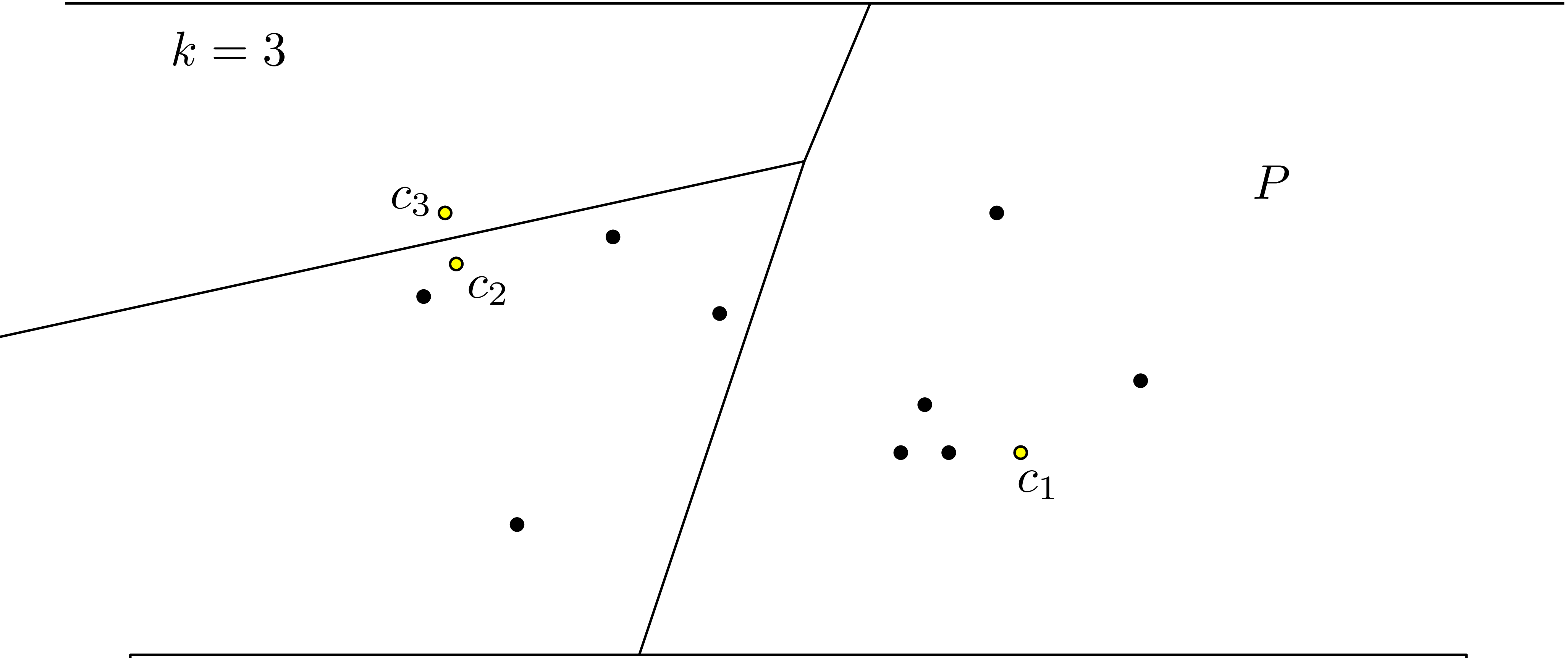
$k = 3$



Iterative refinement with two steps:
(step 1) update assignment (step 2) update centers

Lloyd's algorithm (k-means problem)

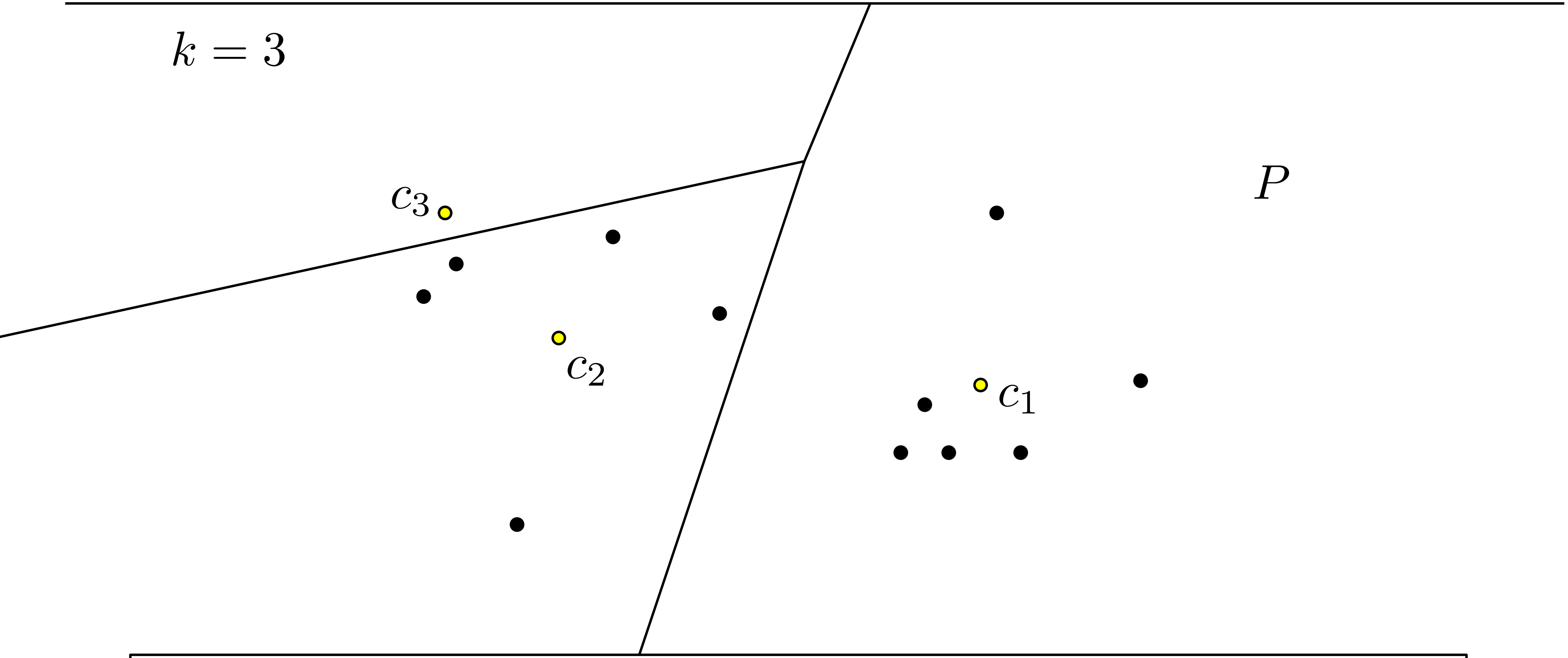
Input: set of points $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^d$, value of k
Output: set of centers $C = \{c_1, \ldots, c_k\} \subseteq \mathbb{R}^d$



Iterative refinement with two steps:
(step 1) update assignment (step 2) update centers

Lloyd's algorithm (k-means problem)

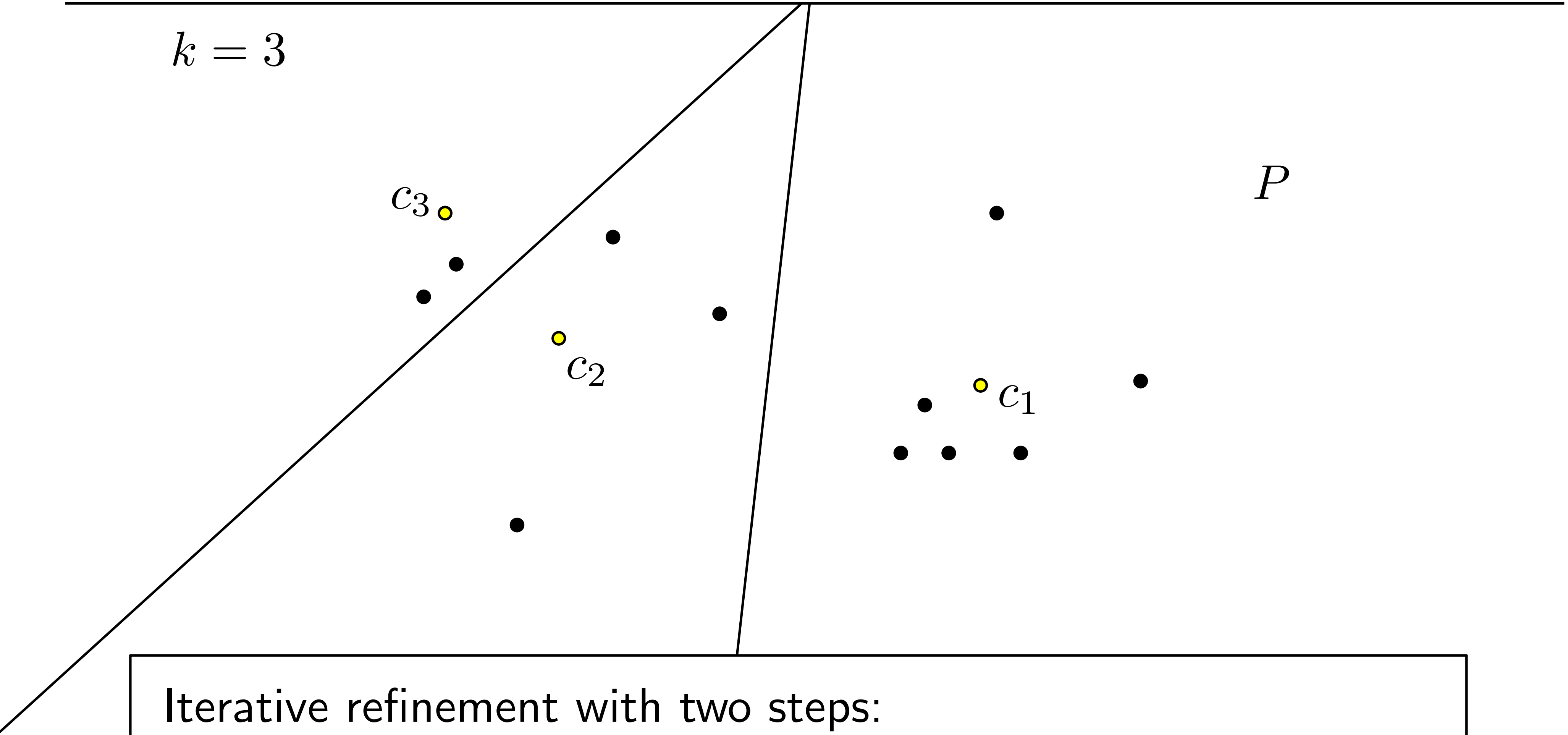
Input: set of points $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^d$, value of k
Output: set of centers $C = \{c_1, \ldots, c_k\} \subseteq \mathbb{R}^d$



Iterative refinement with two steps:
(step 1) update assignment (step 2) update centers

Lloyd's algorithm (k-means problem)

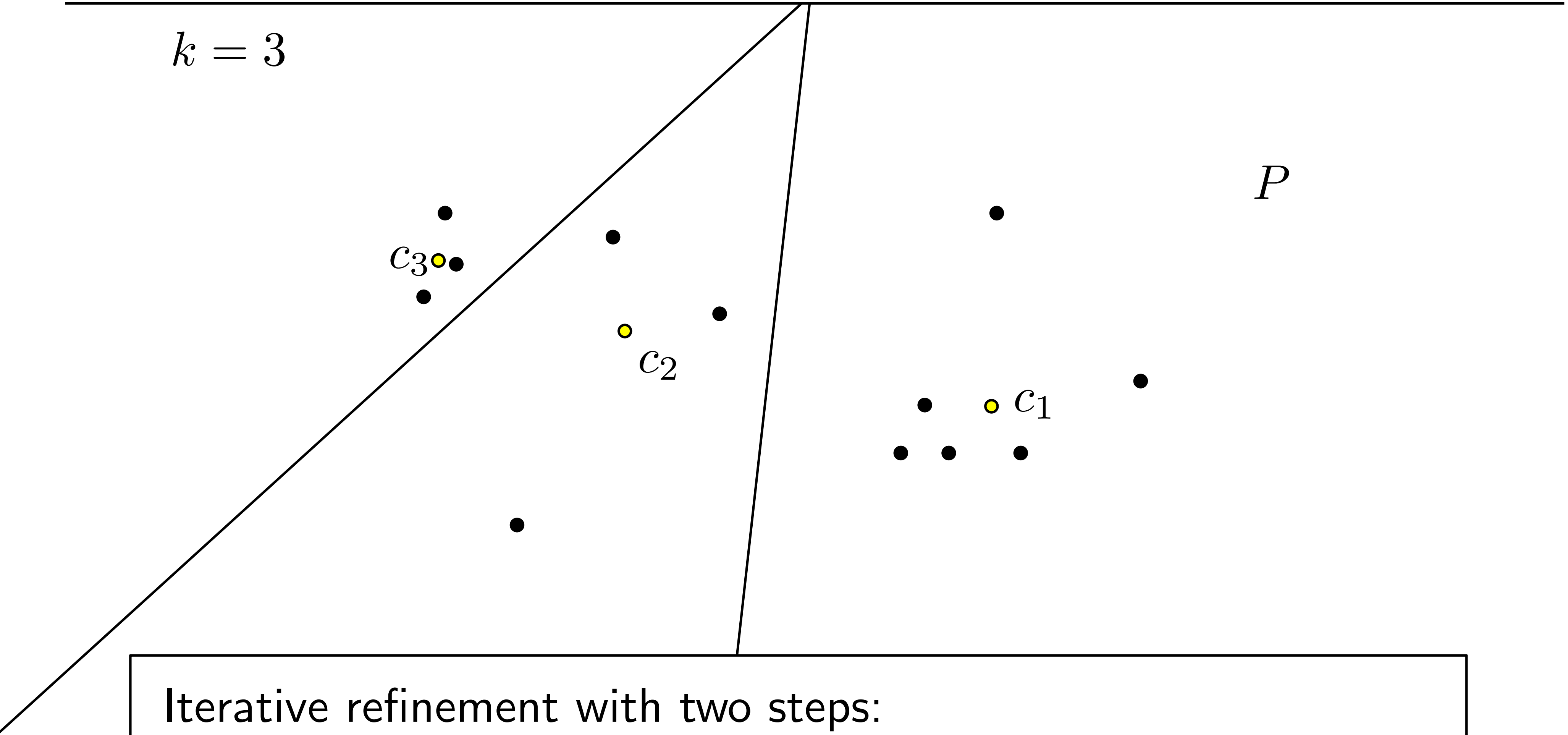
Input: set of points $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^d$, value of k
Output: set of centers $C = \{c_1, \ldots, c_k\} \subseteq \mathbb{R}^d$



Iterative refinement with two steps:
(step 1) update assignment (step 2) update centers

Lloyd's algorithm (k-means problem)

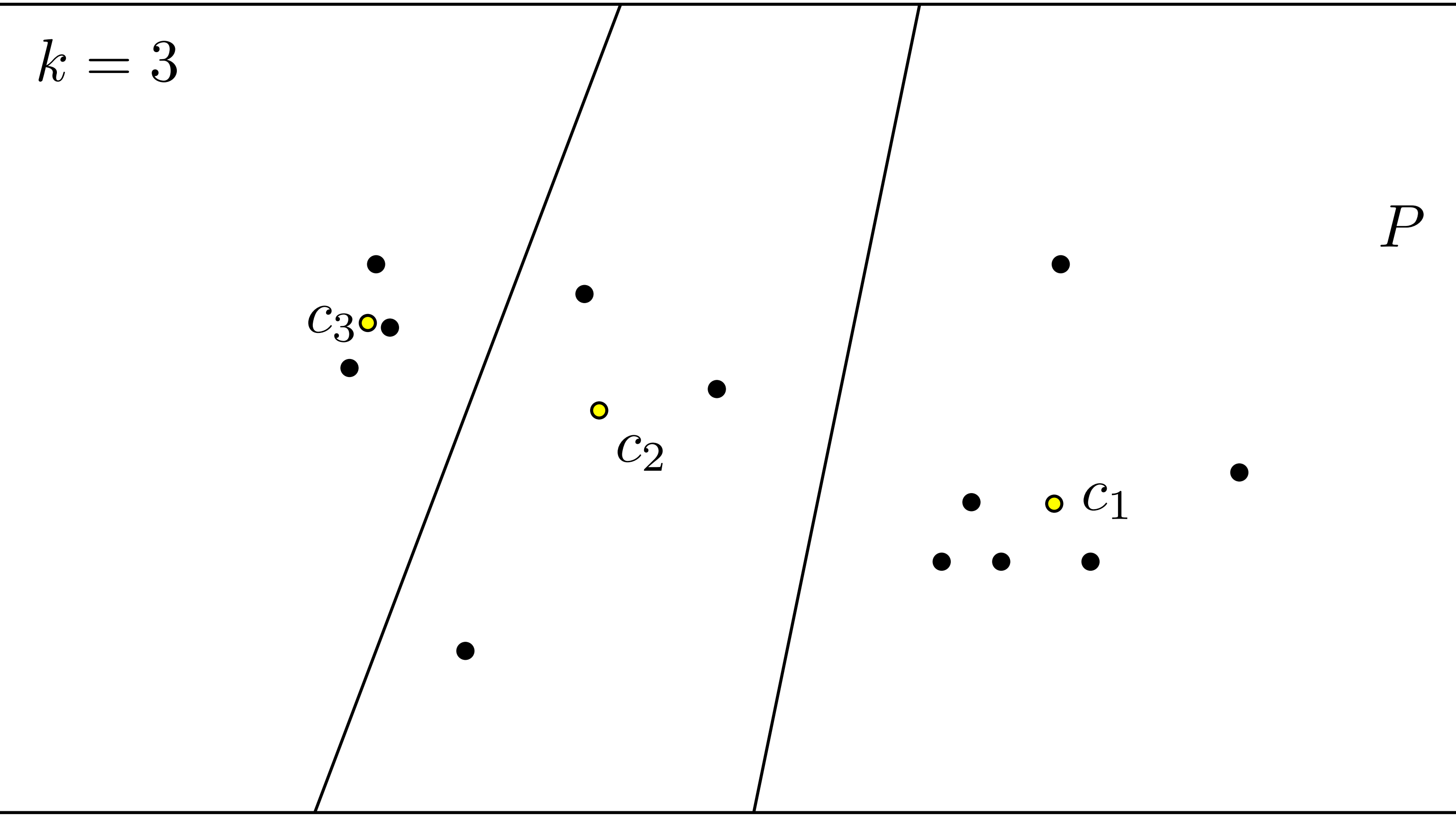
Input: set of points $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^d$, value of k
Output: set of centers $C = \{c_1, \ldots, c_k\} \subseteq \mathbb{R}^d$



Iterative refinement with two steps:
(step 1) update assignment (step 2) update centers

Lloyd's algorithm (k-means problem)

Input: set of points $P = \{p_1, \dots, p_n\} \subseteq \mathbb{R}^d$, value of k
Output: set of centers $C = \{c_1, \dots, c_k\} \subseteq \mathbb{R}^d$



Iterative refinement with two steps:
(step 1) update assignment (step 2) update centers

Lloyd's algorithm (k-means problem)

Input: set of points $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^d$, value of k
Output: set of centers $C = \{c_1, \ldots, c_k\} \subseteq \mathbb{R}^d$

Objective:

$$\text{minimize} \quad \phi(P, C) = \sum_{p_i \in P} \|p_i - \operatorname*{argmin}_{c_j \in C} \|p_i - c_j\|\|^2$$

Algorithm:

- choose initial centers arbitrarily $\{c_1, \ldots, c_k\}$ from P
- until $\{c_1, \ldots, c_k\}$ does not change anymore:

  (1) assign each $p_i \in P$ to its closest center

  (2) update center for each cluster Θ_j:

$$c_j := \frac{1}{m} \sum_{p_i \in \Theta_j} p_i$$

Lloyd's algorithm (k-means problem)

Does this always terminate?

Yes, because:

- each step decreases the cost $\phi(P, C)$
- there exists only a finite number of cluster assignments
- the computed centers are optimal for the current assignment.

Lloyd's algorithm (k-means problem)

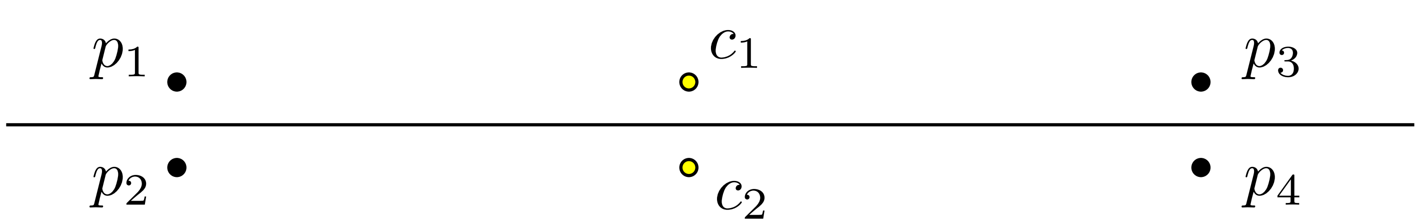
Does this always terminate?

Yes, because:

- each step decreases the cost $\phi(P, C)$
- there exists only a finite number of cluster assignments
- the computed centers are optimal for the current assignment.

Does it converge to an optimal solution?

Yes, if each optimal cluster contains one of the inital centers.
Otherwise the solution it converges to may be arbitrarily bad:



k-means++ Algorithm

Input: set of points $P = \{p_1, \dots, p_n\} \subseteq \mathbb{R}^d$, value of k
Output: set of centers $C = \{c_1, \dots, c_k\} \subseteq \mathbb{R}^2$

Algorithm:

new

- choose c_1 uniformly at random from P
- for $r = 2, \dots, k$: choose $c_r = p_i$ with probability α_i

$$\alpha_i := \frac{\psi_i}{\sum_{s=1,\dots,n} \psi_s} \qquad \psi_i := \left\| p_i - \operatorname*{argmin}_{c_j \in \{c_1,\dots,c_{r-1}\}} \|p_i - c_j\| \right\|^2$$

k-means++ Algorithm

Input: set of points $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^d$, value of k
Output: set of centers $C = \{c_1, \ldots, c_k\} \subseteq \mathbb{R}^2$

Algorithm:

new

- choose c_1 uniformly at random from P
- for $r = 2, \ldots, k$: choose $c_r = p_i$ with probability α_i

$$\alpha_i := \frac{\psi_i}{\sum_{s=1,\ldots,n} \psi_s} \qquad \psi_i := \left\| p_i - \operatorname*{argmin}_{c_j \in \{c_1,\ldots,c_{r-1}\}} \|p_i - c_j\| \right\|^2$$

as before

- until $\{c_1, \ldots, c_k\}$ does not change anymore:

  (1) assign each $p_i \in P$ to its closest center

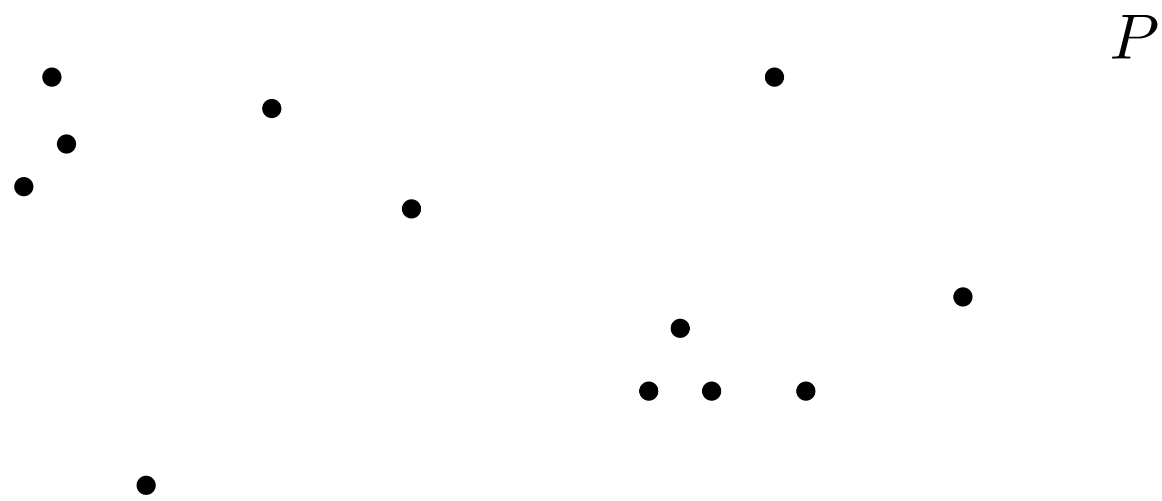
  (2) update center for each cluster Θ_j:

$$c_j := \frac{1}{m} \sum_{p_i \in \Theta_j} p_i$$

k-means++ Algorithm

Input: set of points $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^d$, value of k
Output: set of centers $C = \{c_1, \ldots, c_k\} \subseteq \mathbb{R}^2$

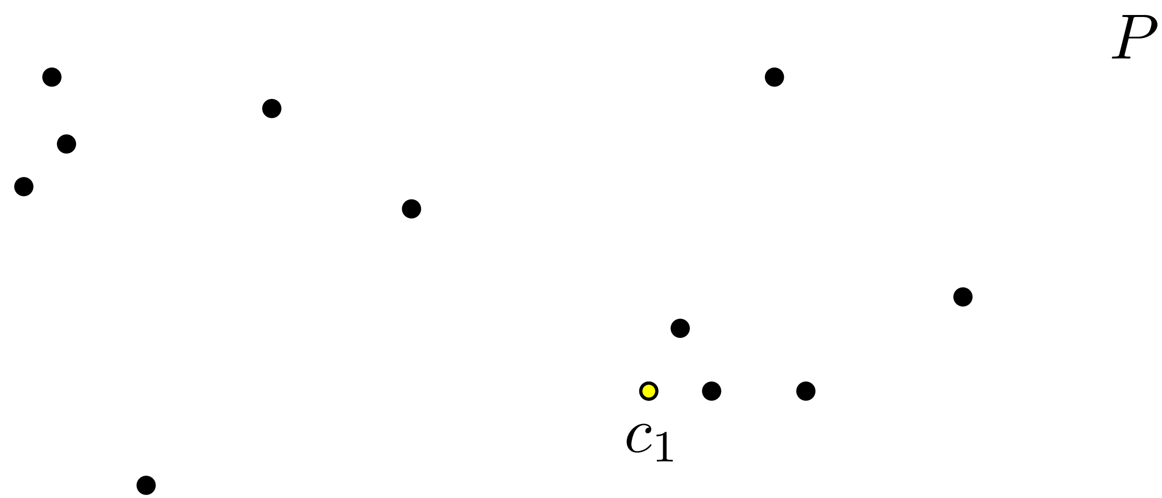
$k = 3$



k-means++ Algorithm

Input: set of points $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^d$, value of k
Output: set of centers $C = \{c_1, \ldots, c_k\} \subseteq \mathbb{R}^2$

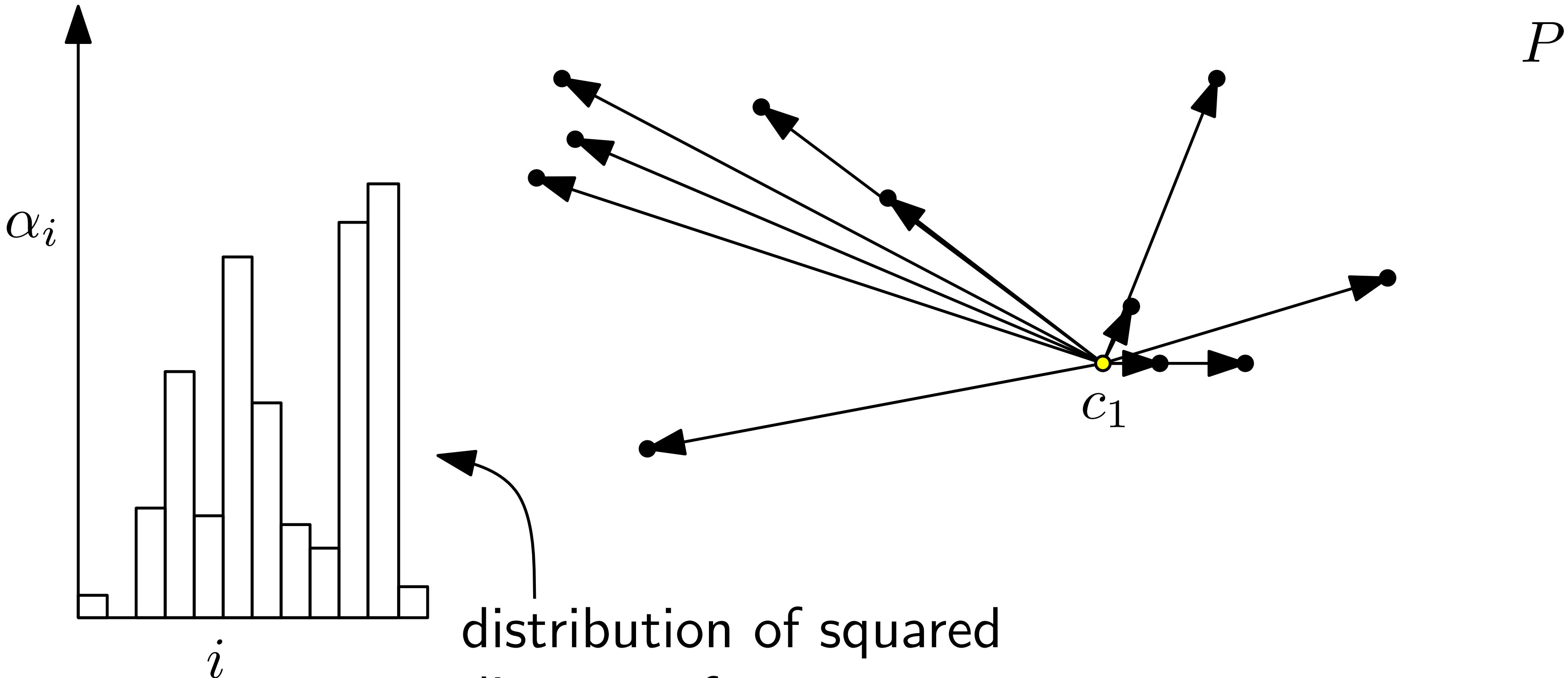
$k = 3$



k-means++ Algorithm

Input: set of points $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^d$, value of k
Output: set of centers $C = \{c_1, \ldots, c_k\} \subseteq \mathbb{R}^2$

$k = 3$



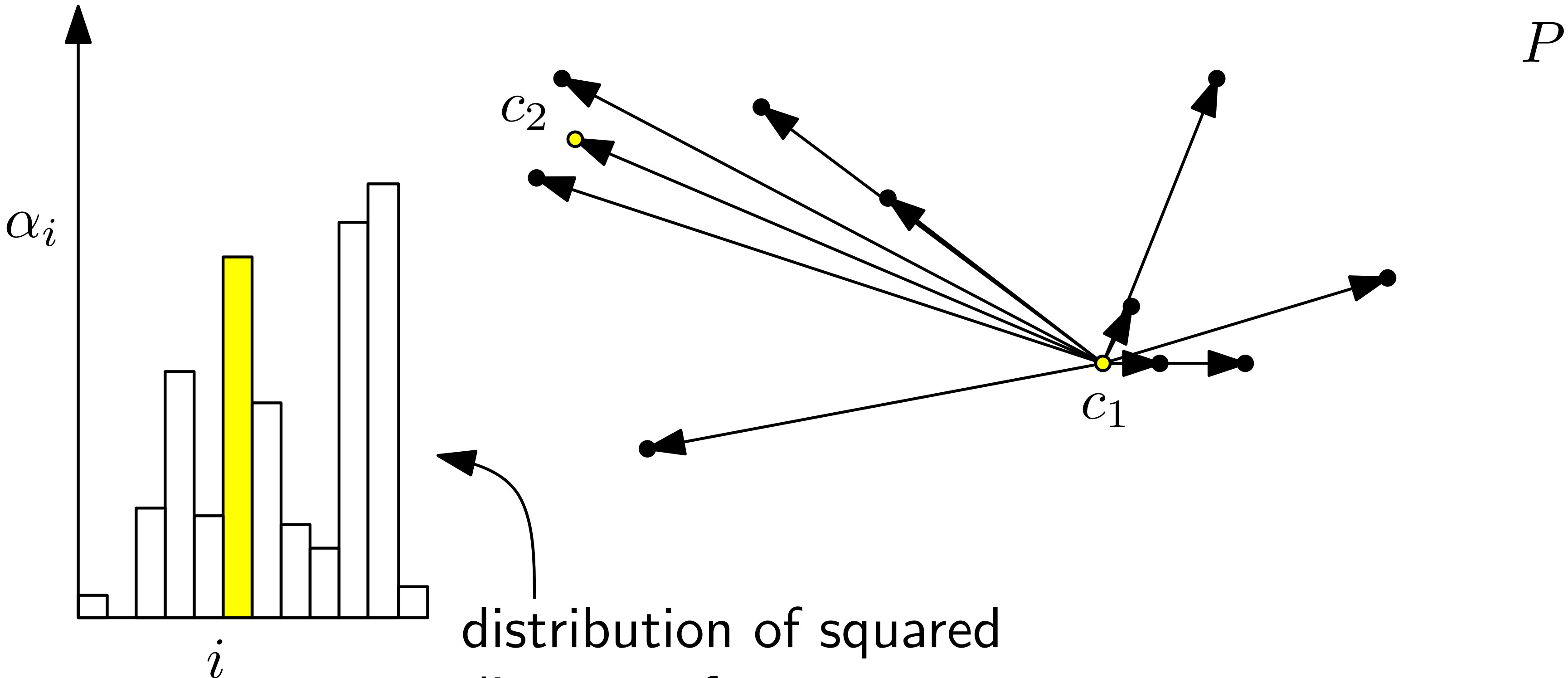
distribution of squared distances from c_1

k-means++ Algorithm

Input: set of points $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^d$, value of k
Output: set of centers $C = \{c_1, \ldots, c_k\} \subseteq \mathbb{R}^2$

$k = 3$

random sample

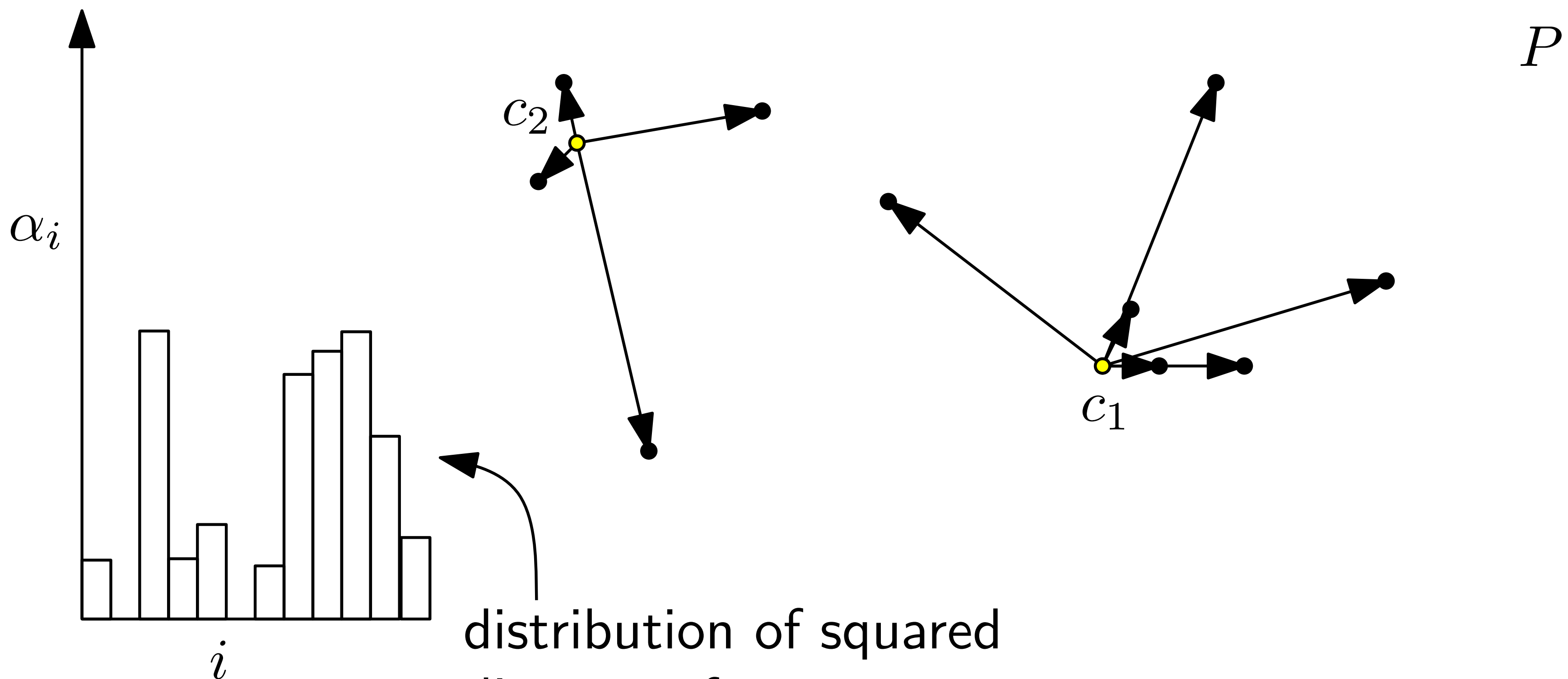


distribution of squared distances from c_1

k-means++ Algorithm

Input: set of points $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^d$, value of k
Output: set of centers $C = \{c_1, \ldots, c_k\} \subseteq \mathbb{R}^2$

$k = 3$

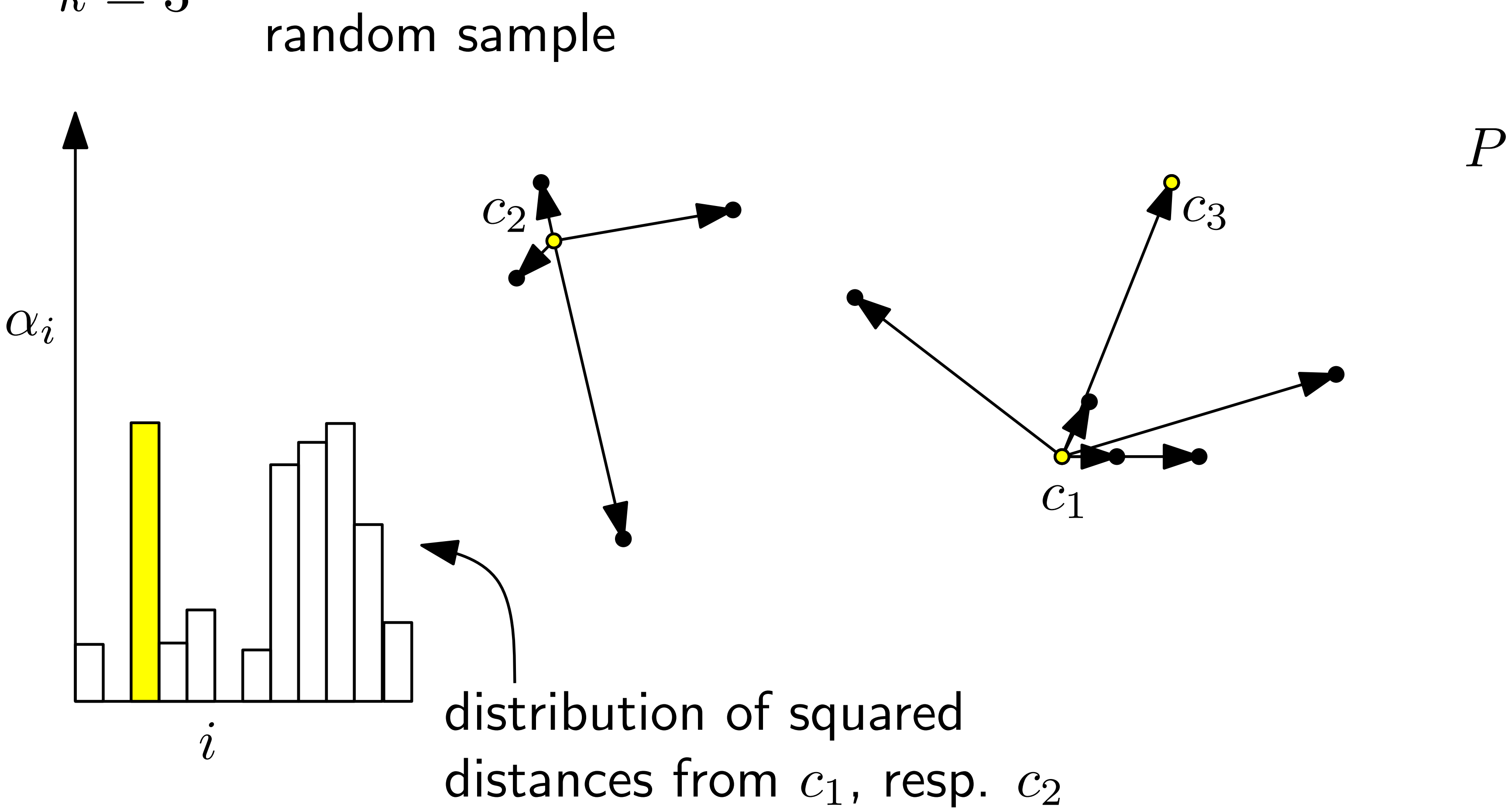


distribution of squared distances from c_1, resp. c_2

k-means++ Algorithm

Input: set of points $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^d$, value of k
Output: set of centers $C = \{c_1, \ldots, c_k\} \subseteq \mathbb{R}^2$

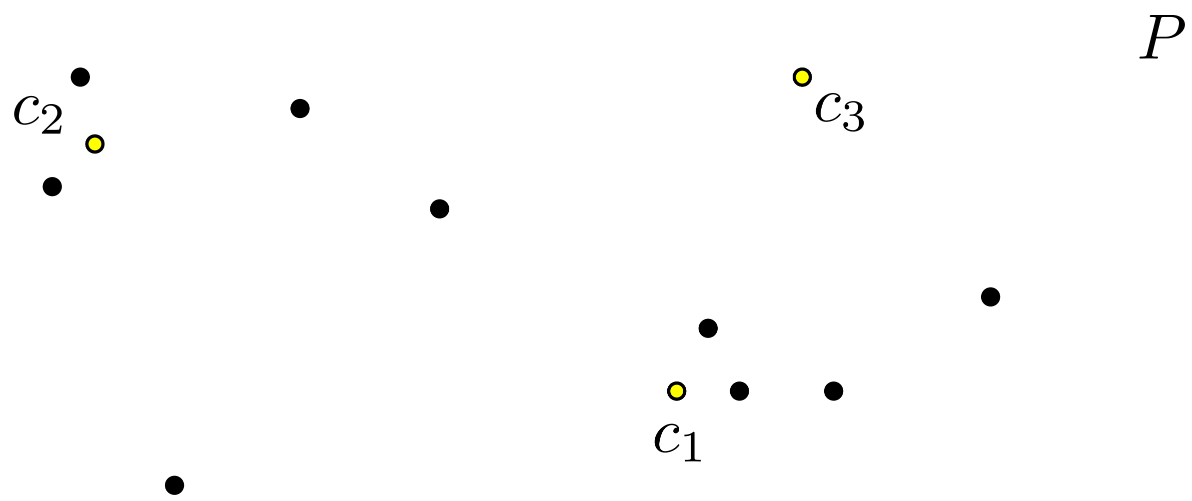
$k = 3$



k-means++ Algorithm

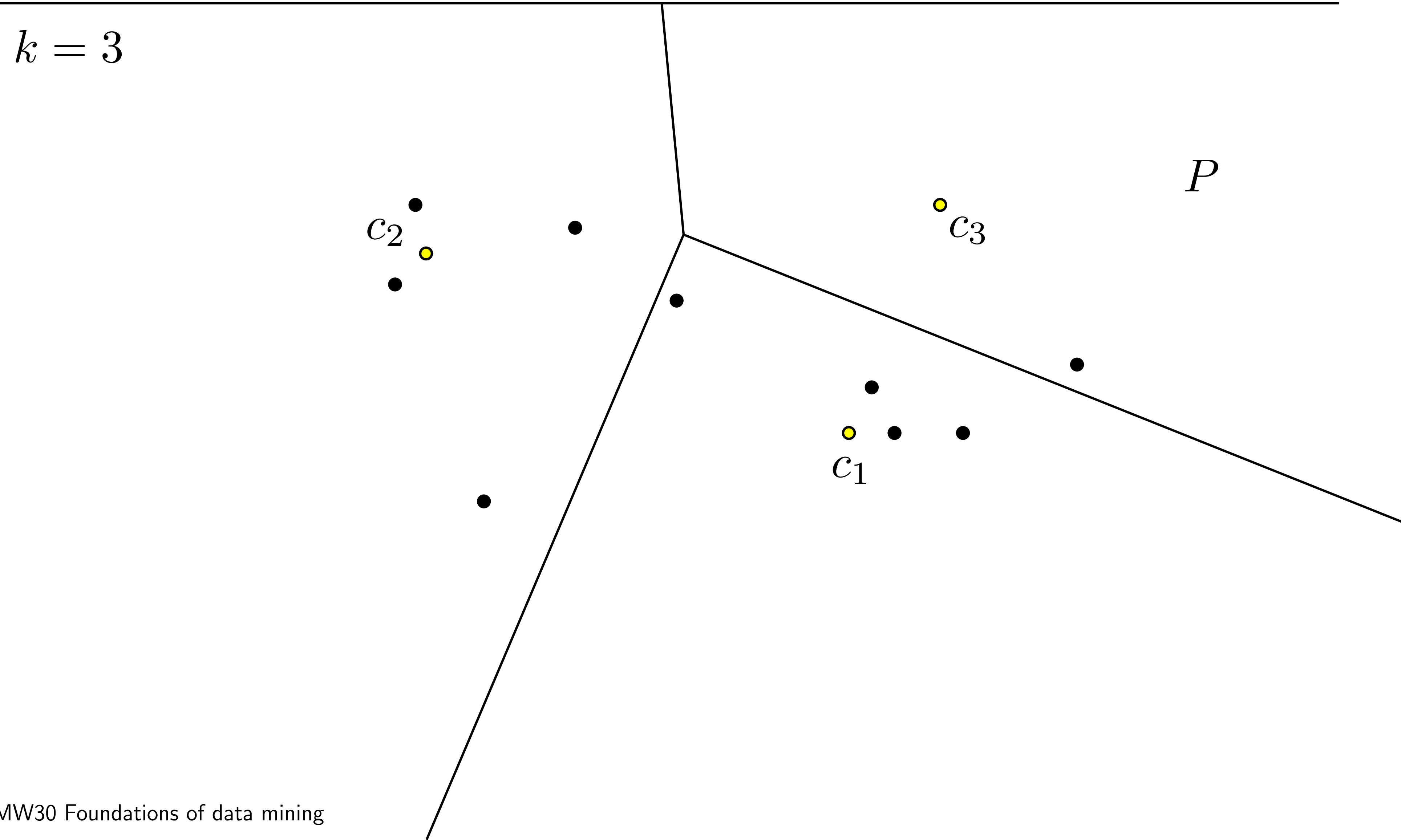
Input: set of points $P = \{p_1, \dots, p_n\} \subseteq \mathbb{R}^d$, value of k
Output: set of centers $C = \{c_1, \dots, c_k\} \subseteq \mathbb{R}^2$

$k = 3$



k-means++ Algorithm

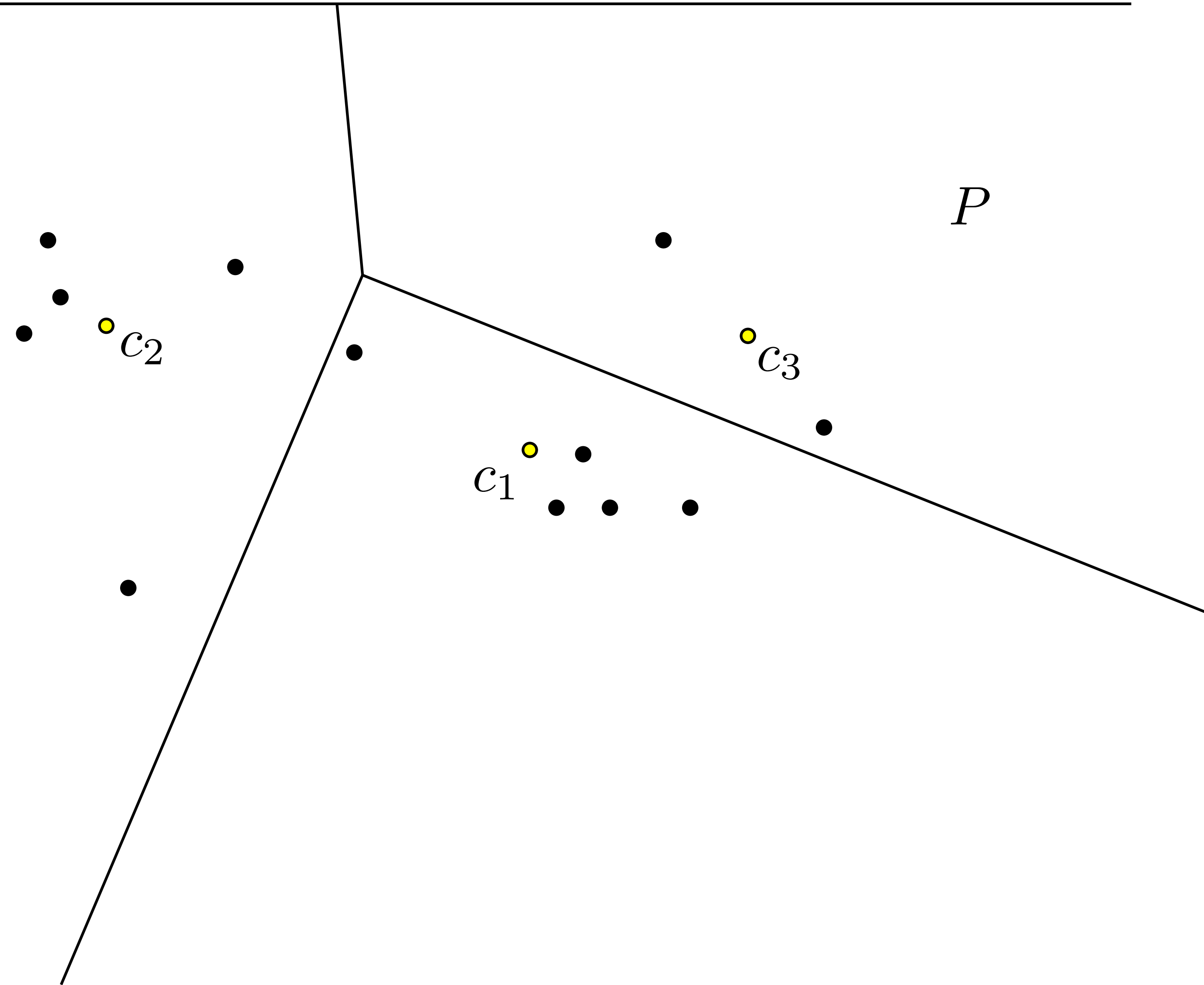
Input: set of points $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^d$, value of k
Output: set of centers $C = \{c_1, \ldots, c_k\} \subseteq \mathbb{R}^2$



k-means++ Algorithm

Input: set of points $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^d$, value of k
Output: set of centers $C = \{c_1, \ldots, c_k\} \subseteq \mathbb{R}^2$

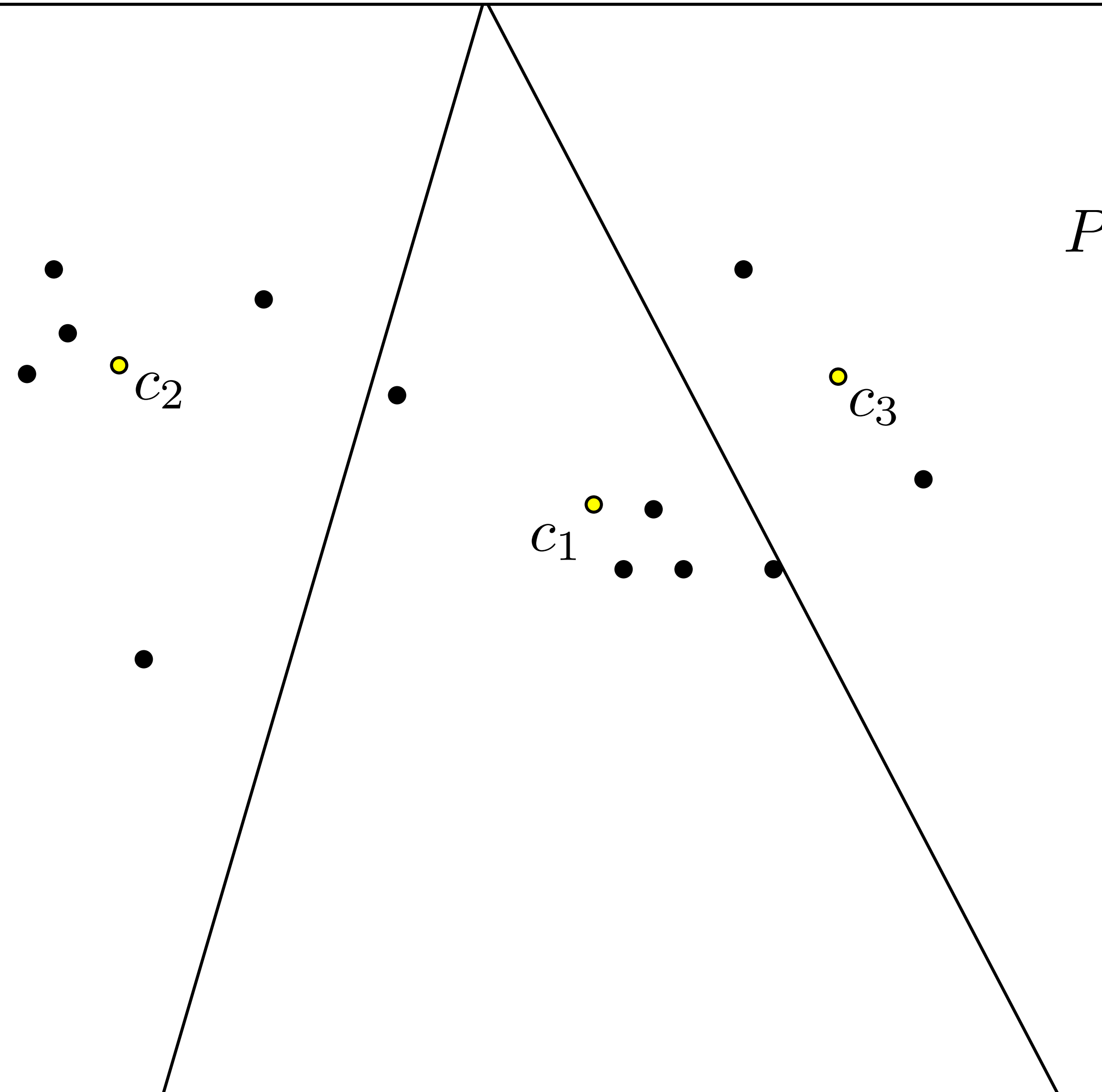
$k = 3$



k-means++ Algorithm

Input: set of points $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^d$, value of k
Output: set of centers $C = \{c_1, \ldots, c_k\} \subseteq \mathbb{R}^2$

$k = 3$



k-means++ Algorithm

Note the similarity to Gonzales algorithm:

- **Gonzales**: choose the next center from P as the point that maximizes the current cost
- **k-means++**: choose the next center from P with probability relative to the contribution to the current cost

k-means++ Algorithm

Note the similarity to Gonzales algorithm:

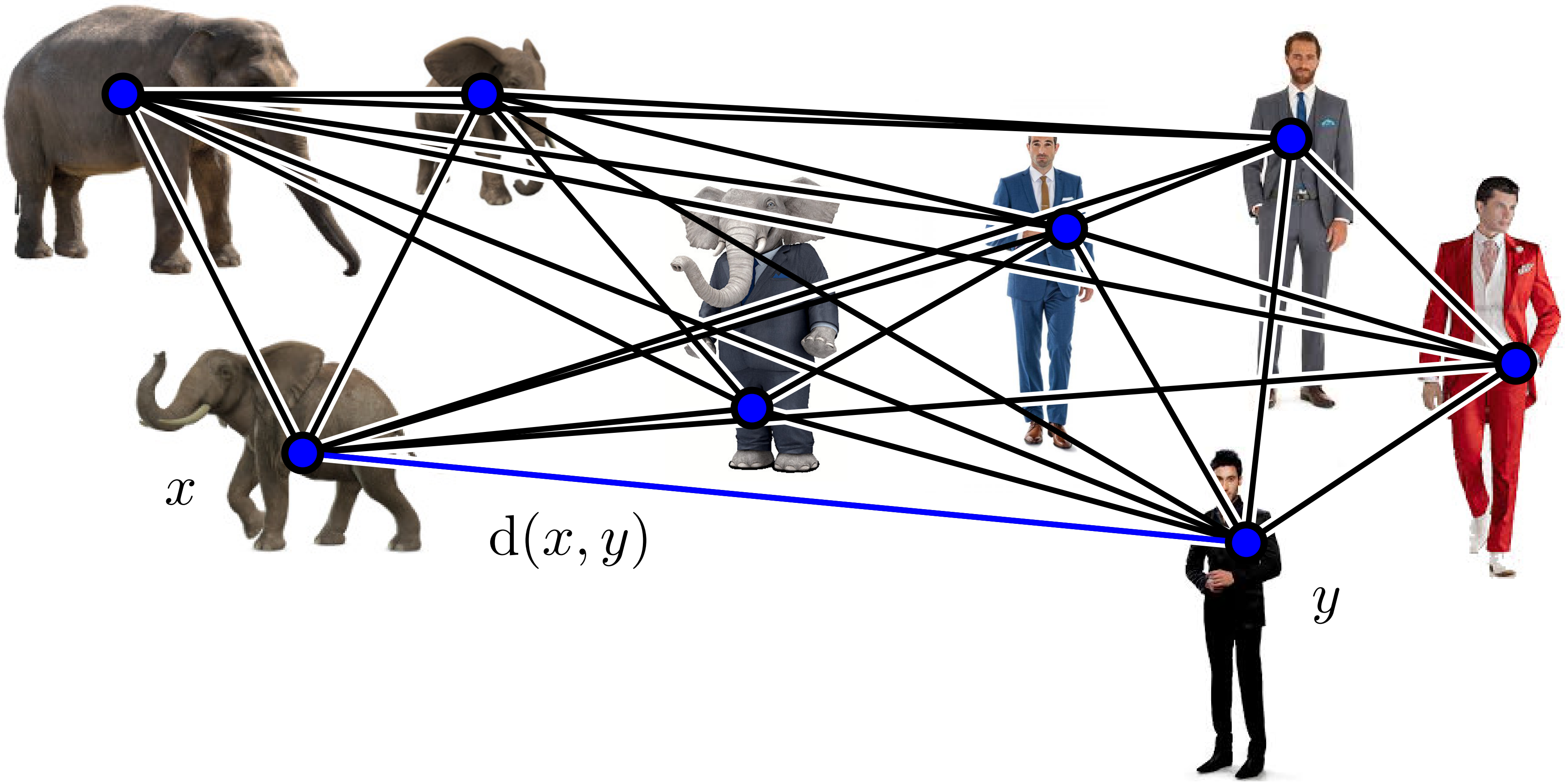
- **Gonzales**: choose the next center from P as the point that maximizes the current cost
- **k-means++**: choose the next center from P with probability relative to the contribution to the current cost

Does it converge to an optimal solution?

In expectation, the solution will be at most a factor $O(\log k)$ worse than the optimal solution (already after random initialization of centers).

Clustering in Graphs

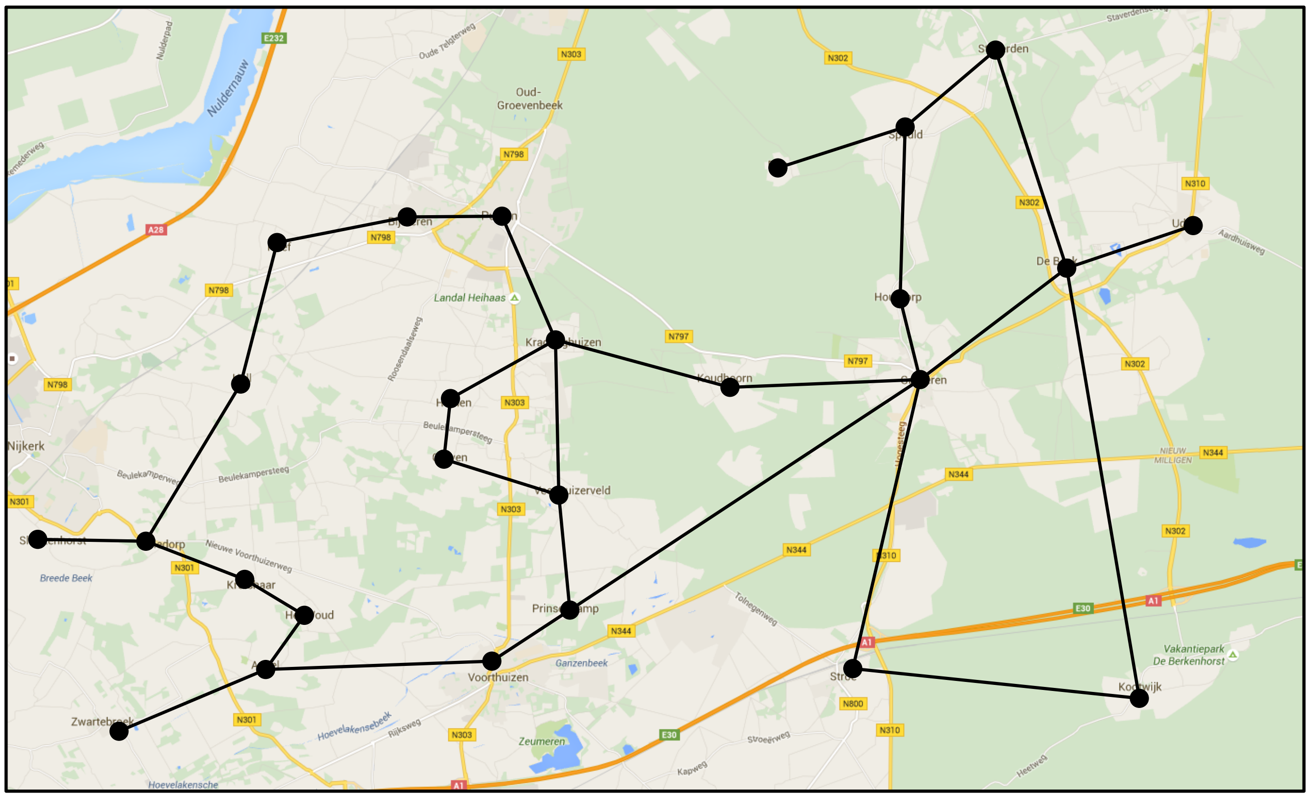
- Vertices of the graph represent the objects to be clustered
- Distance is measured by shortest path



Facility Location with Road Network

You may build two hospitals in two different villages serving the surrounding villages. Where do you place them to minimize the maximal distance from any village to its serving hospital?

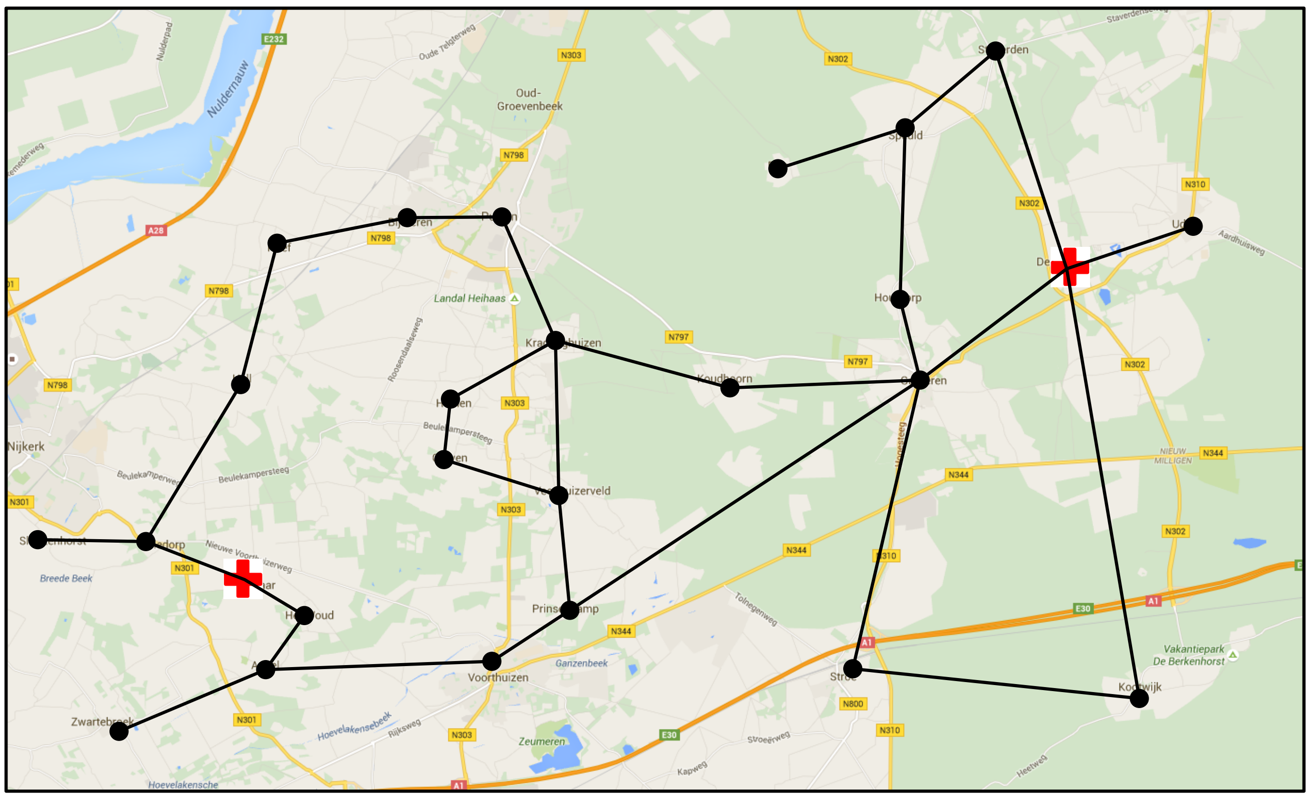
Variant: Measure distance along the road network (travel distance).



Facility Location with Road Network

You may build two hospitals in two different villages serving the surrounding villages. Where do you place them to minimize the maximal distance from any village to its serving hospital?

Variant: Measure distance along the road network (travel distance).



Summary

- Clustering
- Facility Location
- Gonzales' algorithm
- Loyd's algorithm (k-means)
- k-means++ algorithm
- Clustering in graphs

References

- Avrim Blum, John Hopcroft, Ravindran Khannan: *Foundations of Data Science*
- Sariel Har-Peled: *Geometric Approximation Algorithms*
- Arthur, D. and Vassilvitskii, S. (2007). "k-means++: the advantages of careful seeding" (PDF). Proc. 18th ACM-SIAM Symposium on Discrete Algorithms. pp. 1027-1035.