K-MEANS CLUSTERING

Marco Bressan

Università degli Studi di Milano

April 20, 2021

Clustering: super quick intro

Clustering = "group together similar objects"

Clustering: super quick intro

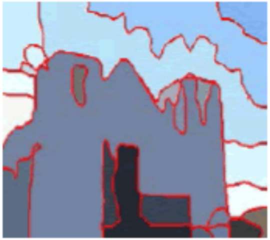
Clustering = "group together similar objects"

Unlike many basic computational problems (sorting, compression, ...), clustering is very **vague** and **underspecified**:

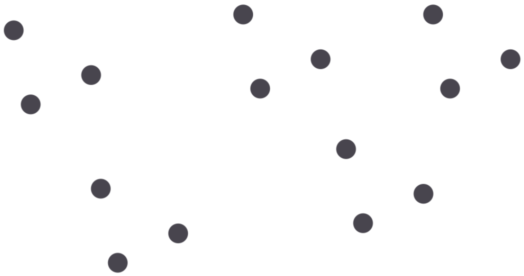
- how do we represent objects?
- what does "similar" mean?
- how many clusters should we form?
- how to measure the quality of a cluster?

Examples

Examples



Examples

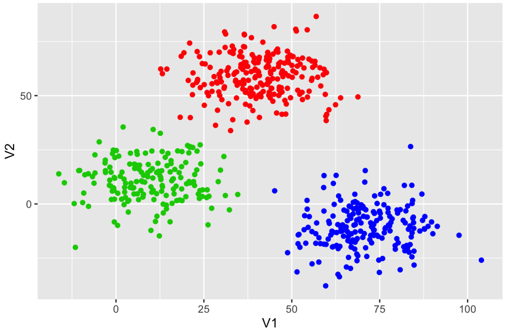


Today we will consider clustering of points in $\mathbb{R}^d$.

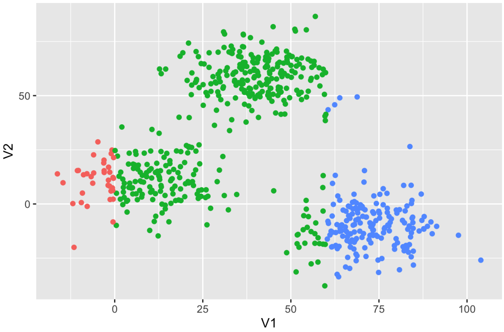
K-means

What does it mean for a clustering to be good?

A "good" clustering



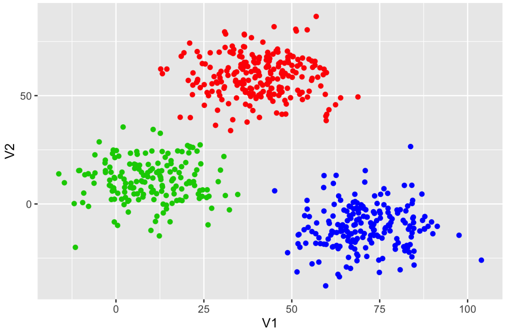
A "bad" clustering



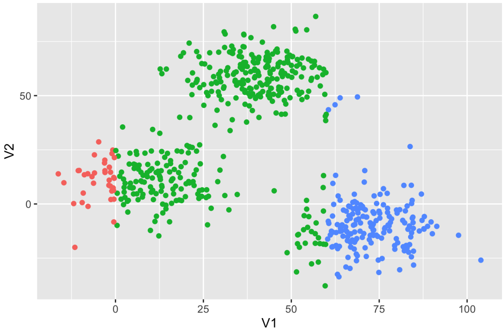
K-means

What does it mean for a clustering to be good?

A “good” clustering



A “bad” clustering



Intuition: a cluster is good if its points are all close to some “central” point.

K-means

Suppose we know k, the number of clusters to be formed.

Given an input set $X \subset \mathbb{R}^d$, we choose k **centers** $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k \in \mathbb{R}^d$.
(Note: we are free to choose the centers anywhere — they need not be in X).

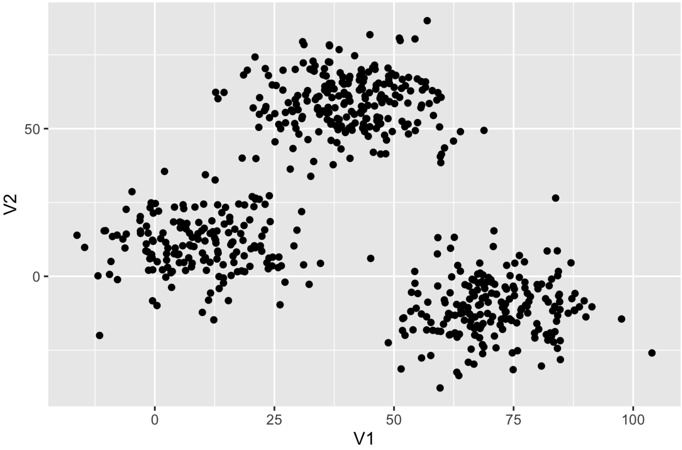
We assign every point $\boldsymbol{x} \in X$ to the closest center among $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k \in \mathbb{R}^d$.

We pay the cost:

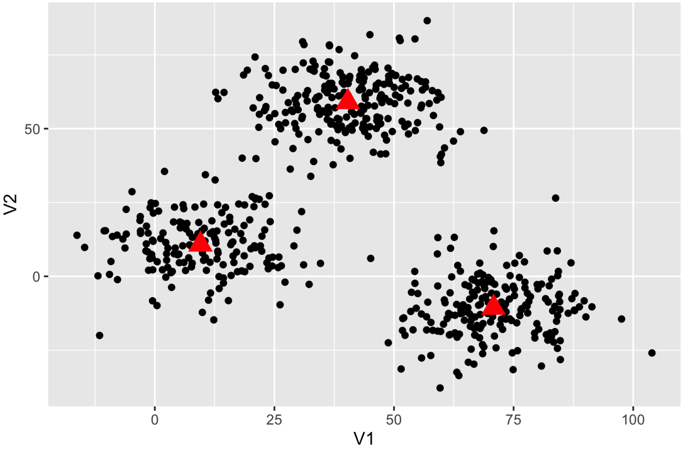
$$\phi(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k) = \sum_{\boldsymbol{x} \in X} \min_{j=1}^{k} \|\boldsymbol{x} - \boldsymbol{c}_j\|_2^2$$

That is, each point pays the squared distance to its center.

Example with k=3

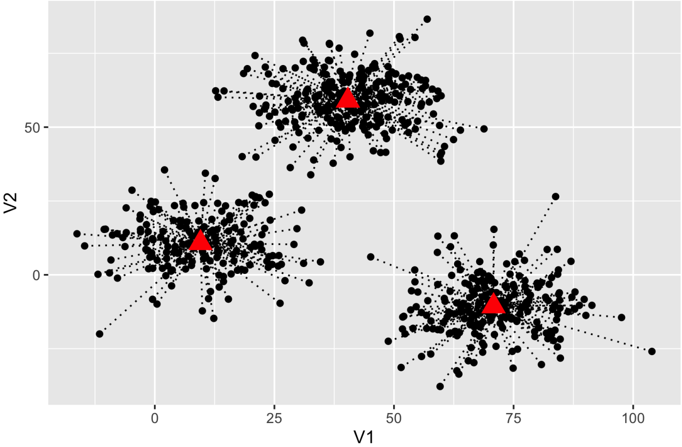


Example with k=3



$$c_1 = (40, 59)$$
$$c_2 = (10, 11)$$
$$c_3 = (71, -11)$$

Example with k=3



$$c_1 = (40, 59)$$
$$c_2 = (10, 11)$$
$$c_3 = (71, -11)$$

$$\phi(c_1, c_2, c_3) = 121368.6$$

K-means

Each choice of centers $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k$ identifies a clustering $\mathcal{C} = (C_1, \ldots, C_k)$:

$$C_i = \{x \in X : d(x, c_i) < d(x, c_j) \, \forall j \neq i\}$$

K-means

Each choice of centers $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k$ identifies a clustering $\mathcal{C} = (C_1, \ldots, C_k)$:

$$C_i = \{x \in X : d(x, c_i) < d(x, c_j) \, \forall j \neq i\}$$

The **optimal k-means clustering**, denoted by $\mathcal{C}^{OPT}$, is the one given by the centers that minimize ϕ,

$$\boldsymbol{c}_1^{OPT}, \ldots, \boldsymbol{c}_k^{OPT} = \arg \min_{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k} \phi(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k)$$

K-means

Each choice of centers $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k$ identifies a clustering $\mathcal{C} = (C_1, \ldots, C_k)$:

$$C_i = \{x \in X : d(x, c_i) < d(x, c_j) \, \forall j \neq i\}$$

The **optimal k-means clustering**, denoted by $\mathcal{C}^{OPT}$, is the one given by the centers that minimize ϕ,

$$\boldsymbol{c}_1^{OPT}, \ldots, \boldsymbol{c}_k^{OPT} = \arg \min_{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k} \phi(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k)$$

The optimal clusters are denoted $C_1^{OPT}, \ldots, C_k^{OPT}$.

Note that $\mathcal{C}^{OPT}$ may not be unique.

K-means

Each choice of centers $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k$ identifies a clustering $\mathcal{C} = (C_1, \ldots, C_k)$:

$$C_i = \{x \in X : d(x, c_i) < d(x, c_j) \, \forall j \neq i\}$$

The **optimal k-means clustering**, denoted by $\mathcal{C}^{OPT}$, is the one given by the centers that minimize ϕ,

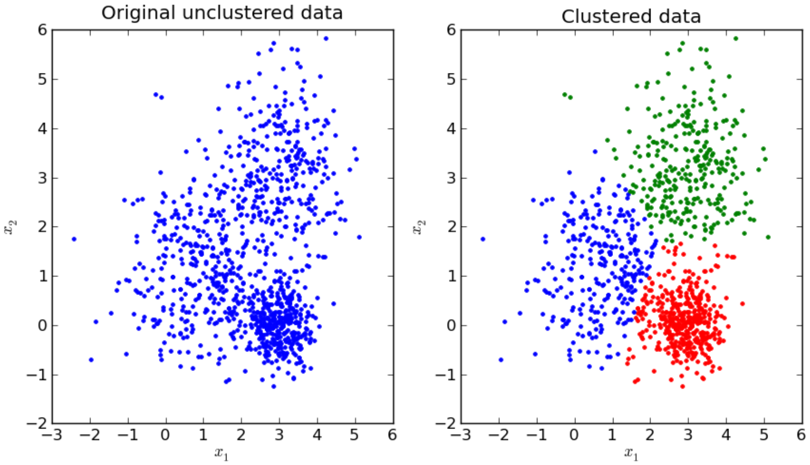
$$\boldsymbol{c}_1^{OPT}, \ldots, \boldsymbol{c}_k^{OPT} = \arg \min_{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k} \phi(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k)$$

The optimal clusters are denoted $C_1^{OPT}, \ldots, C_k^{OPT}$.

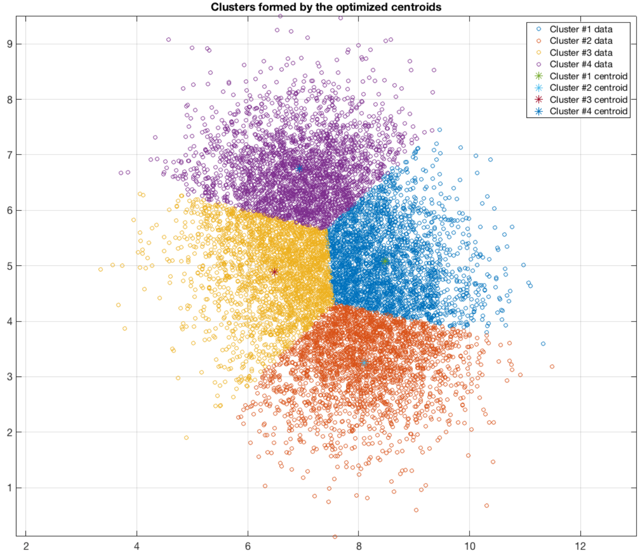
Note that $\mathcal{C}^{OPT}$ may not be unique.

So, the k-means problem is: given X, compute $\mathcal{C}^{OPT}$.

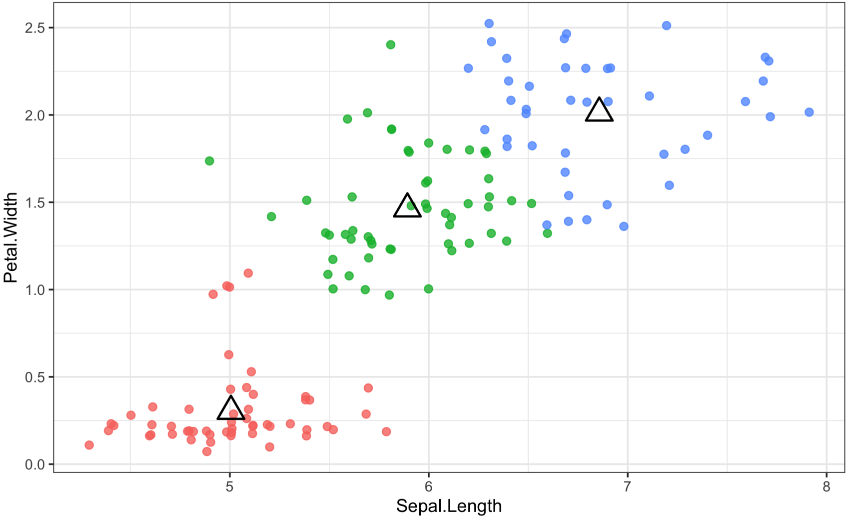
K-means examples



K-means examples



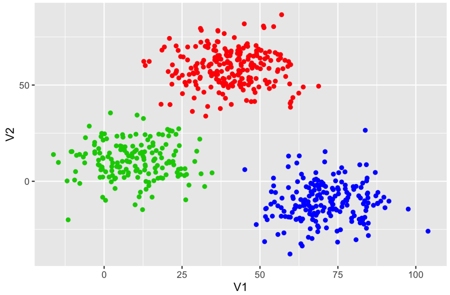
K-means examples



Beware of the dimensionality!

When $d = 2$ or $d = 3$, finding the k-means solution looks easy, because the human eye is good at it. This is not the case when $d \gg 1$.

$d = 2$



$d = 11$

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3

How to solve k-means?

Dumb approach: exhaustive enumeration.

Running time:

$$T \leq \tag{1}$$

How to solve k-means?

Dumb approach: exhaustive enumeration.

Running time:

$$T \leq \# \text{ partitions of } n \text{ points on } k \text{ clusters} \tag{1}$$

How to solve k-means?

Dumb approach: exhaustive enumeration.

Running time:

$$T \leq \#\ \text{partitions of}\ n\ \text{points on}\ k\ \text{clusters}\ \simeq k^n \tag{1}$$

How to solve k-means?

Dumb approach: exhaustive enumeration.

Running time:

$$T \leq \# \text{ partitions of } n \text{ points on } k \text{ clusters } \simeq k^n \quad (1)$$

k-means is NP-hard (Aloise et al., 2009).

Interlude: distances

Why are we using the **squared Euclidean distance** ?

Interlude: distances

Why are we using the **squared Euclidean distance** ?

Consider 10 points at distance 1 from the center:

$$\phi = 10 \times 1^2 = 10$$

versus 1 point at distance 10 from the center:

$$\phi = 1 \times 10^2 = 100$$

Interlude: distances

Why are we using the **squared Euclidean distance** ?

Consider 10 points at distance 1 from the center:

$$\phi = 10 \times 1^2 = 10$$

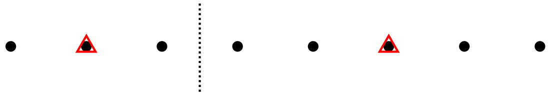
versus 1 point at distance 10 from the center:

$$\phi = 1 \times 10^2 = 100$$

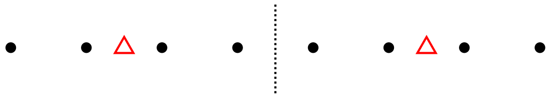
It's like progressive taxes: the farther from the center you are, the more (in proportion) you pay. This tends to give "round" clusters with points roughly equally close to the center.

Example

$X = \{1, \ldots, 8\} \subset \mathbb{R}$, $k = 2$



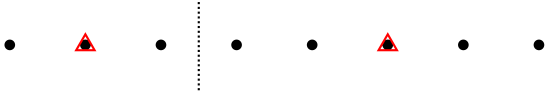
$\boldsymbol{c}_1 = (2, 0), \quad \boldsymbol{c}_2 = (6, 0)$



$\boldsymbol{c}_1 = (2.5, 0), \quad \boldsymbol{c}_2 = (6.5, 0)$

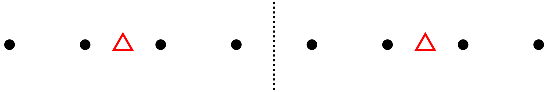
Example

$X = \{1, \ldots, 8\} \subset \mathbb{R}$, $k = 2$



$\boldsymbol{c}_1 = (2, 0)$, $\boldsymbol{c}_2 = (6, 0)$

$\sum_{\boldsymbol{x} \in X} \min_{j=1}^{k} \|\boldsymbol{x} - \boldsymbol{c}_j\|_2 = (1 + 1) + (2 + 1 + 1 + 2) = 8$

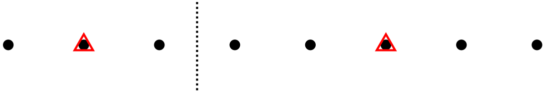


$\boldsymbol{c}_1 = (2.5, 0)$, $\boldsymbol{c}_2 = (6.5, 0)$

$\sum_{\boldsymbol{x} \in X} \min_{j=1}^{k} \|\boldsymbol{x} - \boldsymbol{c}_j\|_2 = (1.5 + 0.5 + 0.5 + 1.5) \cdot 2 = 8$

Example

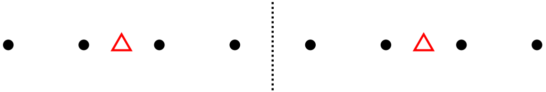
$X = \{1, \ldots, 8\} \subset \mathbb{R},\ k = 2$



$\boldsymbol{c}_1 = (2, 0), \quad \boldsymbol{c}_2 = (6, 0)$

$\sum_{\boldsymbol{x} \in X} \min_{j=1}^{k} \|\boldsymbol{x} - \boldsymbol{c}_j\|_2 = (1 + 1) + (2 + 1 + 1 + 2) = 8$

$\sum_{\boldsymbol{x} \in X} \min_{j=1}^{k} \|\boldsymbol{x} - \boldsymbol{c}_j\|_2^2 = (1 + 1) + (4 + 1 + 1 + 4) = 12$



$\boldsymbol{c}_1 = (2.5, 0), \quad \boldsymbol{c}_2 = (6.5, 0)$

$\sum_{\boldsymbol{x} \in X} \min_{j=1}^{k} \|\boldsymbol{x} - \boldsymbol{c}_j\|_2 = (1.5 + 0.5 + 0.5 + 1.5) \cdot 2 = 8$

$\sum_{\boldsymbol{x} \in X} \min_{j=1}^{k} \|\boldsymbol{x} - \boldsymbol{c}_j\|_2^2 = (1.5^2 + 0.5^2 + 0.5^2 + 1.5^2) \cdot 2 = 10 \ \leftarrow$ **k-means**

Lloyd's algorithm

The main and most used k-means algorithm: Lloyd's algorithm.

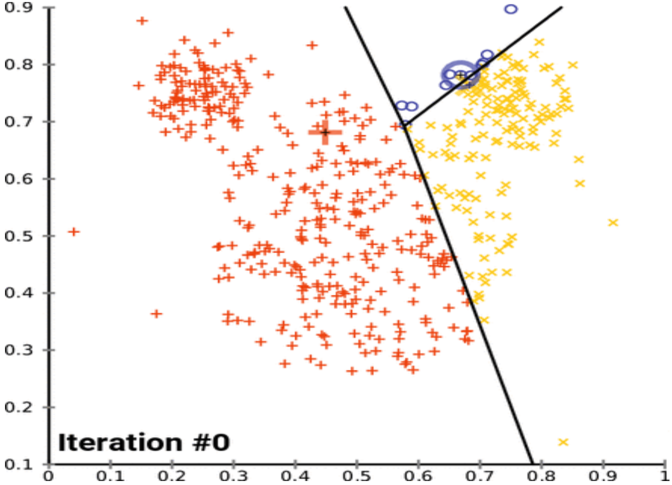
(Very often, by "k-means" people actually mean Lloyd's algorithm.)

Algorithm 1: Lloyd(X, k)

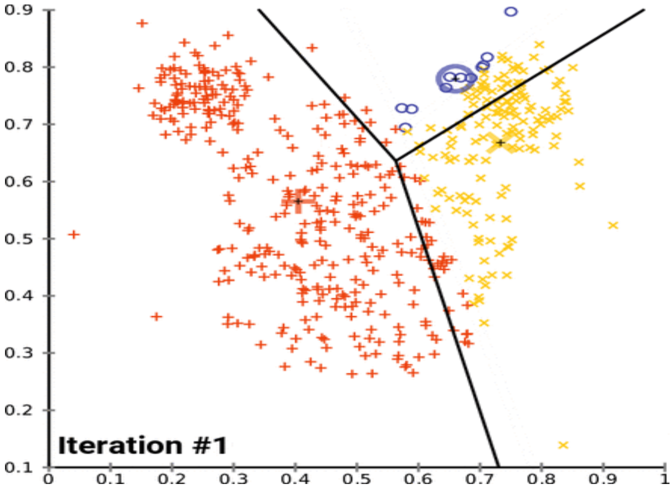
---

choose k distinct points $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k$ u.a.r. from X;
do
  for $i = 1, \ldots, k$ let $C_i =$ the set of points closest to $\boldsymbol{c}_i$;
  for $i = 1, \ldots, k$ let $\boldsymbol{c}_i =$ the center of mass of C_i;
until $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k$ *do not change*;
return $C_1, \ldots, C_k$;

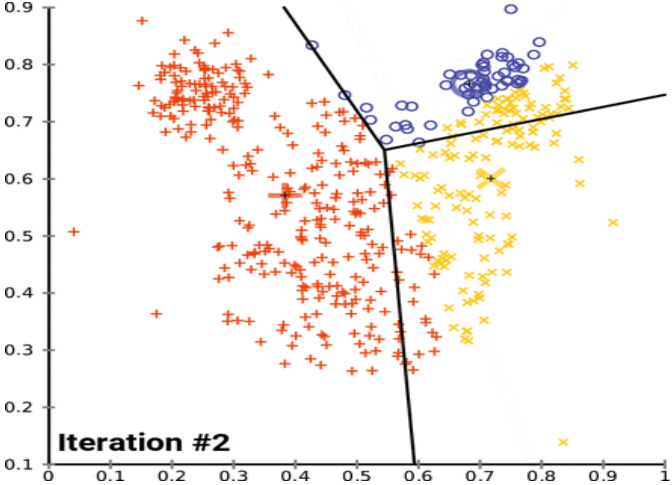
Lloyd's algorithm



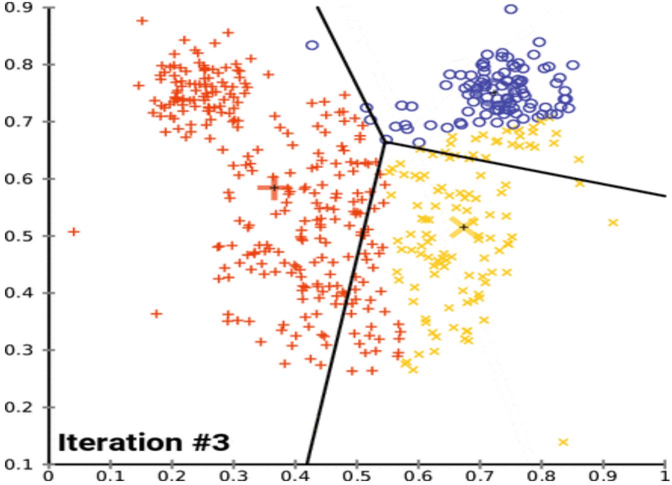
Lloyd's algorithm



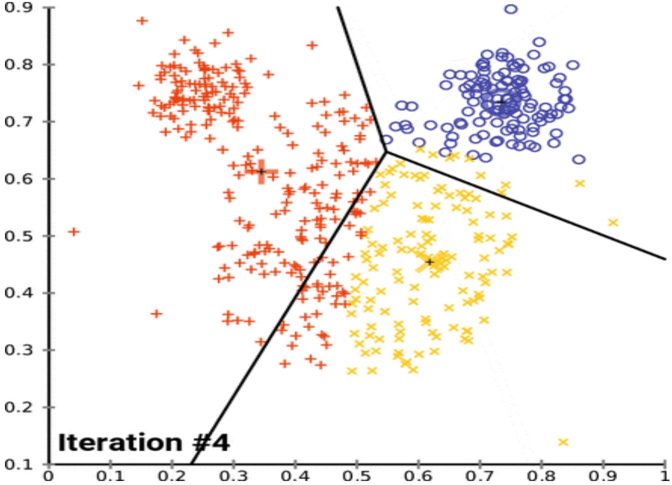
Lloyd's algorithm



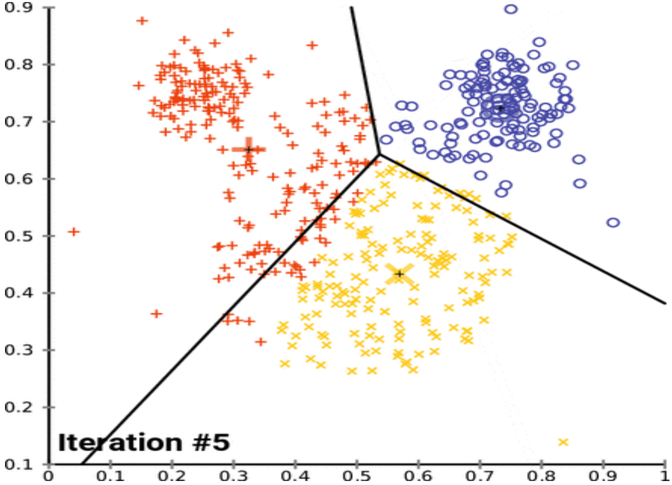
Lloyd's algorithm



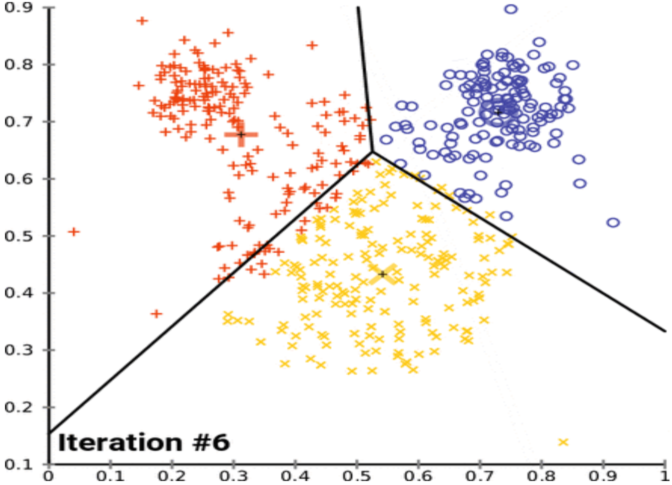
Lloyd's algorithm



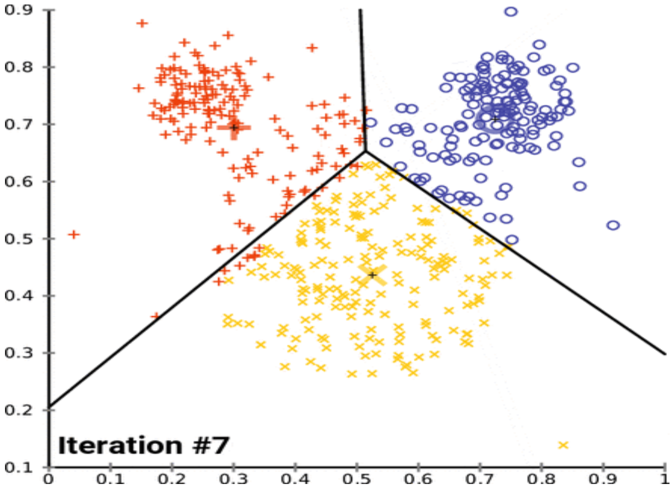
Lloyd's algorithm



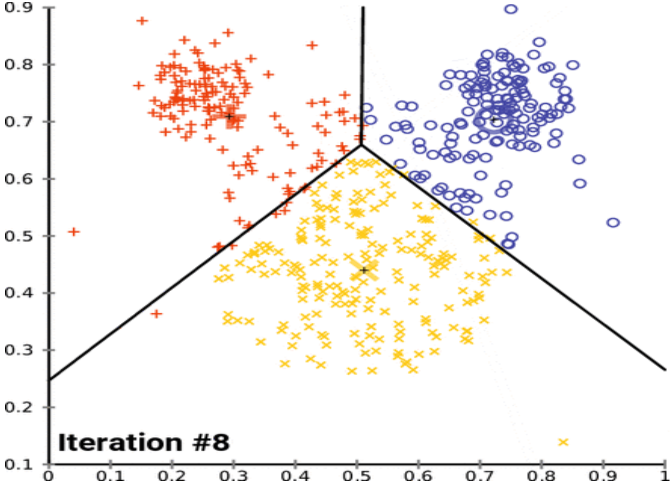
Lloyd's algorithm



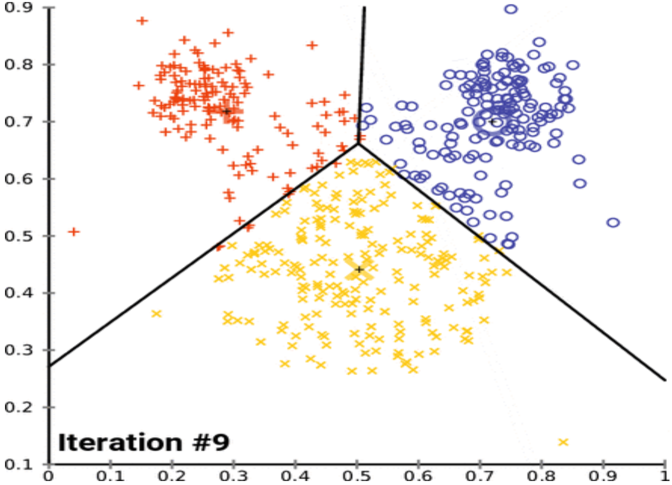
Lloyd's algorithm



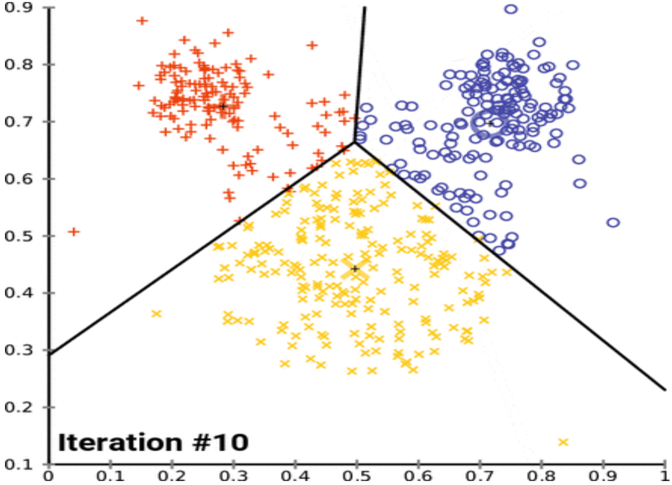
Lloyd's algorithm



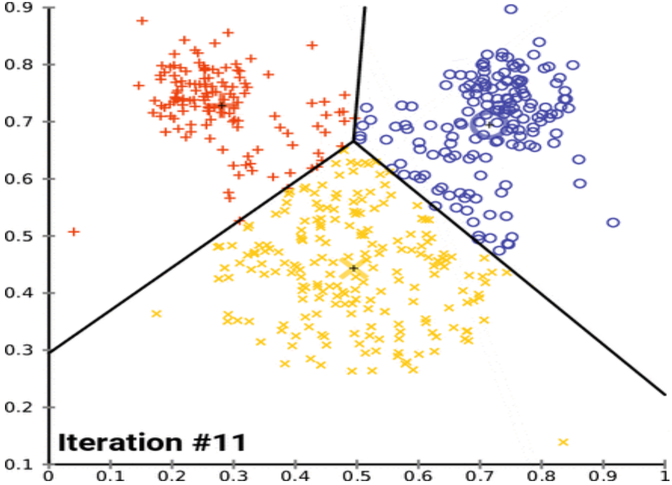
Lloyd's algorithm



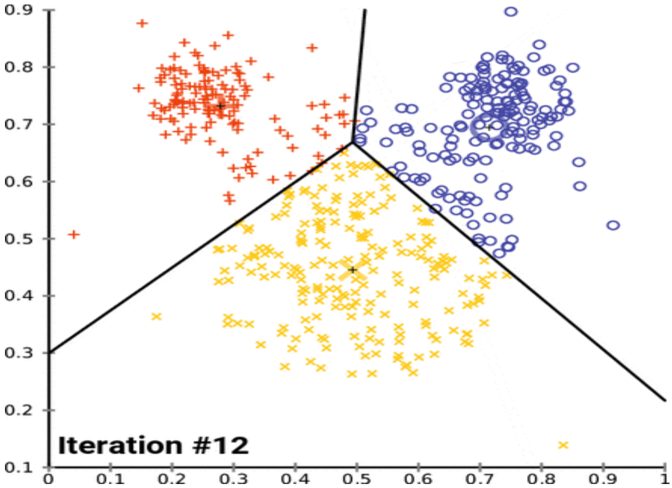
Lloyd's algorithm



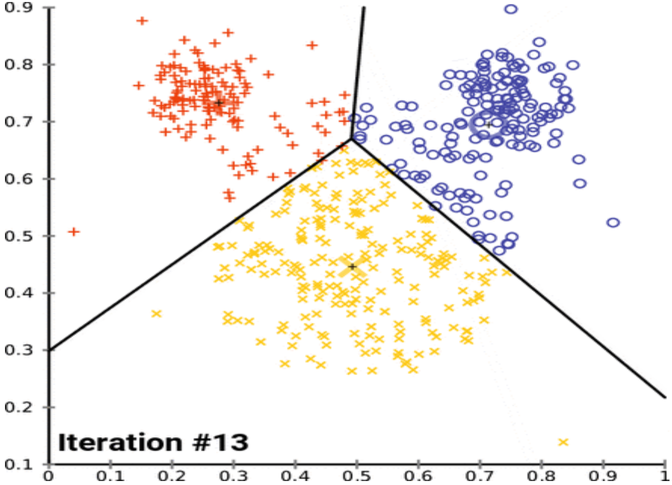
Lloyd's algorithm



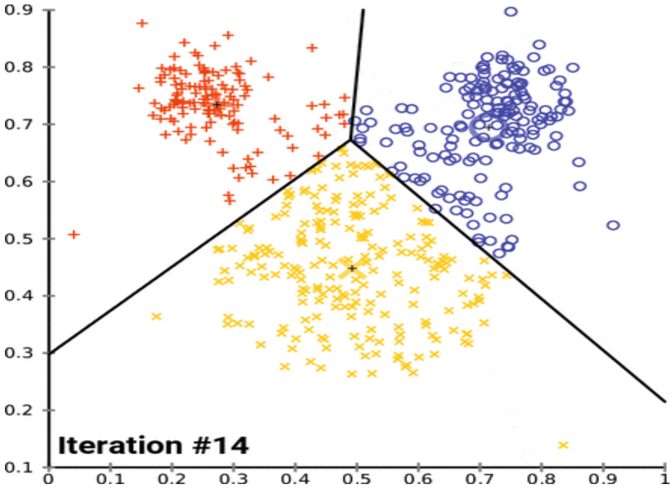
Lloyd's algorithm



Lloyd's algorithm



Lloyd's algorithm



Lloyd's algorithm

- Does it find the optimal clustering? **NO**

Lloyd's algorithm

- Does it find the optimal clustering? **NO**
- Does it find an almost-optimal clustering? **NO**

Lloyd's algorithm

- Does it find the optimal clustering? **NO**
- Does it find an almost-optimal clustering? **NO**
- Does it have good running time? **NO**

Lloyd's algorithm

- Does it find the optimal clustering? **NO**
- Does it find an almost-optimal clustering? **NO**
- Does it have good running time? **NO**
- Why is it so used? It works well in practice

Lloyd's algorithm

Claim: for any constant $a \geq 1$ (say, $a = 1000000$) there are instances on which Lloyd's algorithm, with probability $1 - O(n^{-1})$, returns a clustering $\mathcal{C}$ such that

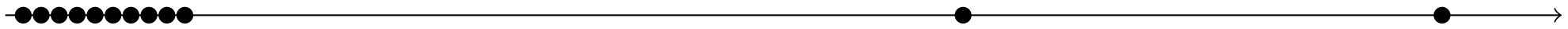
$$\phi(\mathcal{C}) \geq a \cdot \phi(\mathcal{C}_{OPT}) \tag{2}$$

Lloyd's algorithm

Claim: for any constant $a \geq 1$ (say, $a = 1000000$) there are instances on which Lloyd's algorithm, with probability $1 - O(n^{-1})$, returns a clustering $\mathcal{C}$ such that

$$\phi(\mathcal{C}) \geq a \cdot \phi(\mathcal{C}_{OPT}) \qquad (2)$$

Proof: consider this instance on the real line, with $k = 3$, and X formed by $n - 2$ points in $[0, 1]$ and two "outliers" at $x = 2\sqrt{an}$ and $x = 3\sqrt{an}$:

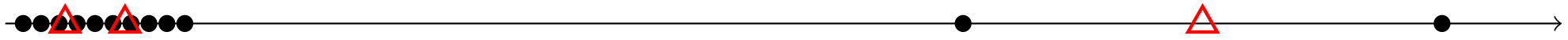


Lloyd's algorithm

Claim: for any constant $a \geq 1$ (say, $a = 1000000$) there are instances on which Lloyd's algorithm, with probability $1 - O(n^{-1})$, returns a clustering $\mathcal{C}$ such that

$$\phi(\mathcal{C}) \geq a \cdot \phi(\mathcal{C}_{OPT}) \tag{2}$$

Proof: consider this instance on the real line, with $k = 3$, and X formed by $n - 2$ points in $[0, 1]$ and two "outliers" at $x = 2\sqrt{an}$ and $x = 3\sqrt{an}$:



With probability $1 - O(\frac{1}{n})$, Lloyd's draws at most 1 center among the outliers, in which case it ends up $\phi(\mathcal{C}) \geq 2 \cdot (\frac{1}{2}\sqrt{an})^2 > a \cdot \frac{n}{4}$.

However, $\phi(\mathcal{C}_{OPT}) \leq (\frac{1}{2})^2 \cdot n = \frac{n}{4}$ (check it!).

Lloyd's algorithm

Claim: the worst-case running time of Lloyd's algorithm is $2^{\Omega(\sqrt{n})}$.

Proof: see Arthur and Vassilvitskii, *How slow is the k-means method?* , Symposium on Computational Geometry, 2006. `https://doi.org/10.1145/1137856.1137880`.

Lloyd's algorithm

Does it even converge?

Back to k-means

Fundamental fact.

Let $C \subset \mathbb{R}^d$ be a finite set of points, and let $\boldsymbol{\mu}$ be its center of mass:

$$\boldsymbol{\mu} = \frac{1}{|C|} \sum_{\boldsymbol{x} \in C} \boldsymbol{x}$$

Then, for any point $\boldsymbol{c} \in \mathbb{R}^d$, we have the identity:

$$\sum_{\boldsymbol{x} \in C} \|\boldsymbol{x} - \boldsymbol{c}\|_2^2 = \sum_{\boldsymbol{x} \in C} \|\boldsymbol{x} - \boldsymbol{\mu}\|_2^2 \ + \ |C| \cdot \|\boldsymbol{c} - \boldsymbol{\mu}\|_2^2$$

Lloyd's algorithm always terminates

Theorem. Lloyd's algorithm always terminates.

Proof stategy: We show that, if the centers are moved, then ϕ decreases *strictly*, and it can do so at most k^n times (the number of possible clusterings).

Lloyd's algorithm always terminates

Claim. In any given iteration, if some center is moved, then ϕ decreases strictly.

Proof. Recall the two steps of the iteration:

for $i = 1, \ldots, k$ *let* $C_i =$ *the set of points closest to* $\boldsymbol{c}_i$
for $i = 1, \ldots, k$ *let* $\boldsymbol{c}_i =$ *the center of mass of* C_i

Denote the state of the algorithm by:

$\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k$	$C_1, \ldots, C_k$	at the beginning of the iteration
$\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k$	$C'_1, \ldots, C'_k$	after the first step
$\boldsymbol{c}'_1, \ldots, \boldsymbol{c}'_k$	$C'_1, \ldots, C'_k$	after the second step

We denote

$$\phi(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k, C_1, \ldots, C_k) = \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in C_i} \|\boldsymbol{x} - \boldsymbol{c}_i\|_2^2 \tag{3}$$

Lloyd's algorithm always terminates

First, we prove:

$$\phi(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k, C_1, \ldots, C_k) \geq \phi(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k, C'_1, \ldots, C'_k)$$

Lloyd's algorithm always terminates

First, we prove:

$$\phi(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k, C_1, \ldots, C_k) \geq \phi(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k, C'_1, \ldots, C'_k)$$

This holds since, for a fixed choice of $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k$, the partition $C'_1, \ldots, C_k$ minimizes ϕ, since it assigns every point to the nearest center.

Lloyd's algorithm always terminates

Second, we prove that, if $\boldsymbol{c}'_i \neq \boldsymbol{c}_i$ for some i, then:

$$\phi(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k, C'_1, \ldots, C'_k) > \phi(\boldsymbol{c}'_1, \ldots, \boldsymbol{c}'_k, C'_1, \ldots, C'_k)$$

Lloyd's algorithm always terminates

Second, we prove that, if $\boldsymbol{c}'_i \neq \boldsymbol{c}_i$ for some i, then:

$$\phi(\boldsymbol{c}_1, \dots, \boldsymbol{c}_k, C'_1, \dots, C'_k) > \phi(\boldsymbol{c}'_1, \dots, \boldsymbol{c}'_k, C'_1, \dots, C'_k)$$

Consider indeed any such i and recall that $\boldsymbol{c}'_i$ is the *center of mass* of C'_i. By the fundamental observation above,

$$\begin{aligned}\sum_{\boldsymbol{x} \in C'_i} \|\boldsymbol{x} - \boldsymbol{c}_i\|_2^2 &= \sum_{\boldsymbol{x} \in C'_i} \|\boldsymbol{x} - \boldsymbol{c}'_i\|_2^2 + |C'_i| \cdot \|\boldsymbol{c}_i - \boldsymbol{c}'_i\|_2^2 \\ &> \sum_{\boldsymbol{x} \in C'_i} \|\boldsymbol{x} - \boldsymbol{c}'_i\|_2^2\end{aligned}$$

Summing over all clusters yields the claim.

Lloyd's algorithm always terminates

So, if some center is moved, we have:

$$\phi(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k, C_1, \ldots, C_k) \geq \phi(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k, C'_1, \ldots, C'_k) > \phi(\boldsymbol{c}'_1, \ldots, \boldsymbol{c}'_k, C'_1, \ldots, C'_k)$$

which means that ϕ decreases strictly.

Lloyd's algorithm always terminates

Finally, we note that ϕ can decrease only a finite number of times.

Lloyd's algorithm always terminates

Finally, we note that ϕ can decrease only a finite number of times.

First, ϕ is a function of the current clustering $C_1, \ldots, C_k$, which can take on at most k^n distinct values. Thus, if the algorithm did more than k^n iterations, it would go twice over the same clustering. This implies that ϕ takes the same value in two distinct iterations (ignoring the last one). This is absurd since ϕ always decreases.

Lloyd's algorithm always terminates

Finally, we note that ϕ can decrease only a finite number of times.

First, ϕ is a function of the current clustering $C_1, \ldots, C_k$, which can take on at most k^n distinct values. Thus, if the algorithm did more than k^n iterations, it would go twice over the same clustering. This implies that ϕ takes the same value in two distinct iterations (ignoring the last one). This is absurd since ϕ always decreases.

This completes the proof of the theorem.

Lloyd's algorithm

How much does one iteration of Lloyd's algorithm take?

Algorithm 2: Lloyd(X, k)

---

choose k distinct points $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k$ u.a.r. from X;

do

  for $i = 1, \ldots, k$ let C_i = the set of points closest to $\boldsymbol{c}_i$;

  for $i = 1, \ldots, k$ let $\boldsymbol{c}_i$ = the center of mass of C_i;

until $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k$ *do not change*;

return $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k$;

Lloyd's algorithm

How much does one iteration of Lloyd's algorithm take?

Algorithm 3: Lloyd(X, k)

---

choose k distinct points $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k$ u.a.r. from X;
do

    ~~for $i = 1, \ldots, k$ let C_i = the set of points closest to $\boldsymbol{c}_i$~~;
    for $\boldsymbol{x} \in X$ let $i_x = \arg\min_{i \in [k]} \|\boldsymbol{x} - \boldsymbol{c}_i\|_2^2$;
    for $i = 1, \ldots, k$ let $\boldsymbol{c}_i$ = the center of mass of C_i;

until $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k$ *do not change*;
return $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k$;

$\Rightarrow O(n \cdot k \cdot d)$ per iteration

Recap

k-means

- probably the most popular idea of “clustering”
- formalizes clustering as an optimization problem
- NP-hardness

Lloyd’s algorithm

- unbounded approximation ratio
- worst-case running time $2^{\Omega(n)}$
- but, in practice, it works well