

# *k*-Means Clustering

Julia Kempe & David S. Rosenberg

CDS, NYU

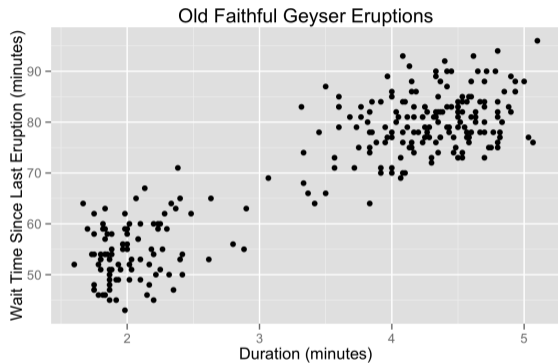
May 1, 2019

- 1  $k$ -Means Clustering
- 2  $k$ -Means: Failure Cases
- 3  $k$ -means Formalized

# *k*-Means Clustering

---

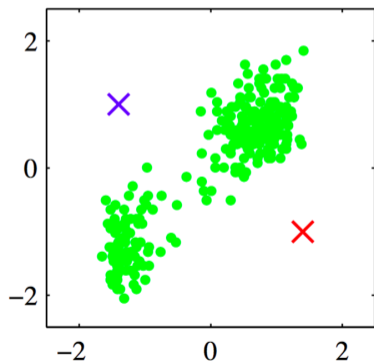
## Example: Old Faithful Geyser



- Looks like two clusters.
- How to find these clusters algorithmically?

## k-Means: By Example

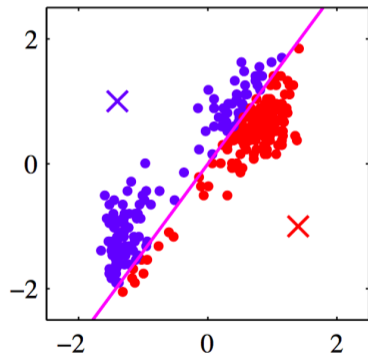
- Standardize the data.
- Choose two cluster centers.



From Bishop's *Pattern recognition and machine learning*, Figure 9.1(a).

## k-means: by example

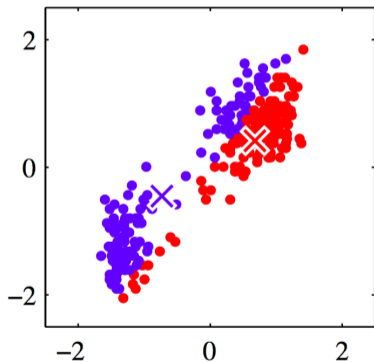
- Assign each point to closest center.



From Bishop's *Pattern recognition and machine learning*, Figure 9.1(b).

## k-means: by example

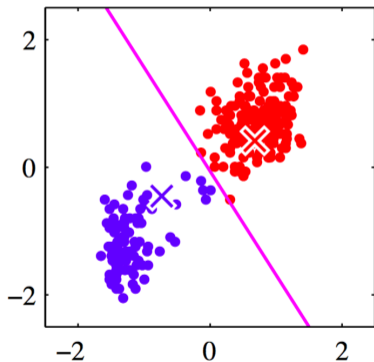
- Compute new class centers.



From Bishop's *Pattern recognition and machine learning*, Figure 9.1(c).

## k-means: by example

- Assign points to closest center.

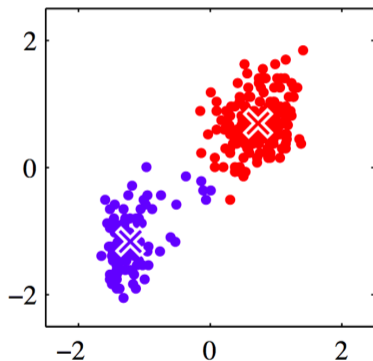


From Bishop's *Pattern recognition and machine learning*, Figure 9.1(d).



## k-means: by example

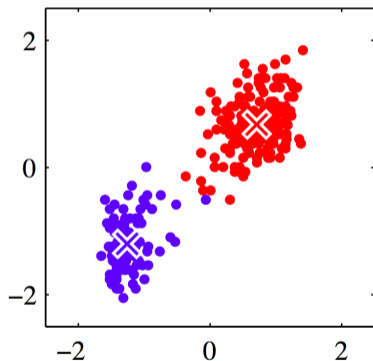
- Compute cluster centers.



From Bishop's *Pattern recognition and machine learning*, Figure 9.1(e).

## k-means: by example

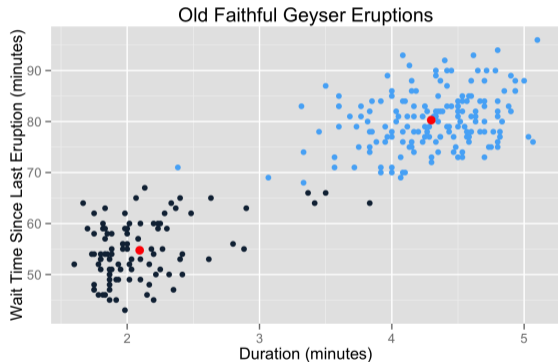
- Iterate until convergence.



From Bishop's *Pattern recognition and machine learning*, Figure 9.1(i).

# k-Means Algorithm: Standardizing the data

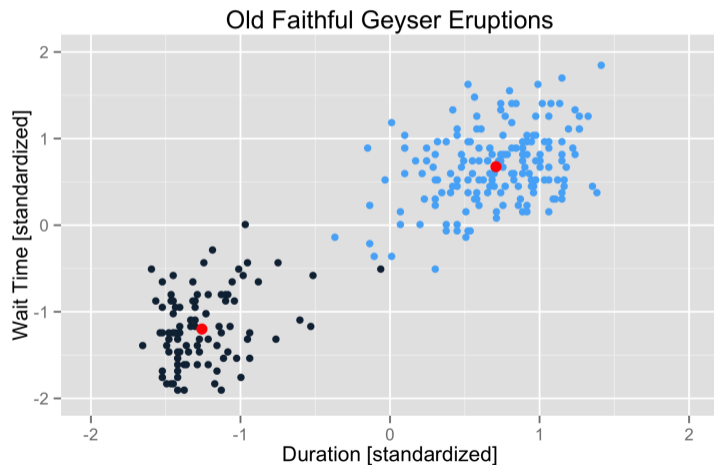
- Without standardizing:



- Blue and black show results of k-means clustering
- Wait time dominates the distance metric

# k-Means Algorithm: Standardizing the data

- With standardizing:



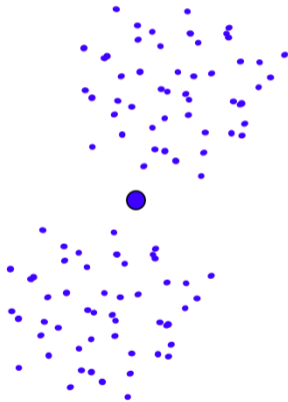
## *k*-Means: Failure Cases

## $k$ -Means: Suboptimal Local Minimum

- The clustering for  $k = 3$  below is a local minimum, but suboptimal:



Would be better to have  
one cluster here



... and two clusters here

## *k*-means Formalized

## k-Means: Setting

- Let  $\mathcal{X}$  be a space with some distance metric  $d$ .
  - Most commonly,  $\mathcal{X} = \mathbf{R}^d$  and  $d(x, x') = \|x - x'\|$ .
- Dataset  $\mathcal{D} = \{x_1, \dots, x_n\} \subset \mathcal{X}$ .
- Goal: Partition data  $\mathcal{D}$  into  $k$  disjoint sets  $C_1, \dots, C_k$ .
- The **centroid** of  $C_i$  is defined to be

$$\mu_i = \mu(C_i) = \arg \min_{\mu \in \mathcal{X}} \sum_{x \in C_i} d(x, \mu)^2.$$

- Note: For Euclidean distance on  $\mathbf{R}^d$ ,  $\mu(C_i)$  is the mean of  $C_i$ .



## $k$ -Means: Objective function

- The  $k$ -means objective is

$$\begin{aligned} J_{k\text{-means}}(C_1, \dots, C_k) &= \sum_{i=1}^k \sum_{x \in C_i} d(x, \mu(C_i))^2 \\ &= \min_{\mu_1, \dots, \mu_k \in \mathcal{X}} \sum_{i=1}^k \sum_{x \in C_i} d(x, \mu_i)^2 \end{aligned}$$

- In **vector quantization**, we represent each  $x \in C_i$  by the centroid  $\mu_i$ .
- We can think of this as lossy data compression,
  - the  $k$ -means objective can be viewed as the reconstruction error.
- How many bits does it take to represent each point with vector quantization?
  - If  $k = 2^d$ , then  $d$  bits. (Fewer on average if the clusters have unequal sizes.)

# k-Means: Algorithm

- input:  $\mathcal{D} = \{x_1, \dots, x_d\} \subset \mathcal{X}$
- initialize: Randomly choose initial centroids  $\mu_1, \dots, \mu_k \subset \mathcal{D}$ .
- repeat until convergence (i.e. until the centroids or clusters repeat):
  - $\forall i$ , let  $C_i = \{x \in \mathcal{D} : i = \arg \min_j d(x, \mu_j)\}$ . (break ties in some arbitrary manner)
  - $\forall i$ , let  $\mu_i = \arg \min_{\mu \in \mathcal{X}} \sum_{x \in C_i} d(x, \mu)^2$ . (For Euclidean distance,  $\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$ )

- In  $k$ -means, objective never increases, but no guarantee to find minimizer.
- General recommendation is to re-run with several random starting initial centroids.
- $k$ -means++ is a way to randomly initialize the centroids with some guarantees:
  - Randomly choose first centroid from the data points  $\mathcal{D}$ .
  - For each of the remaining  $k - 1$  centroids:
    - Compute distance from each  $x_i$  to the closest already chosen centroid.
    - Randomly choose next centroid with probability proportional to the computed distance squared.
- If we let  $J_{k\text{-means}}^*$  be the minimizer of the  $k$ -means objective, then using  $k$ -means++ for initialization guarantees that

$$\mathbb{E}[J_{k\text{-means}}(C_1, \dots, C_k)] \leq 8(\log k + 2) J_{k\text{-means}}^*.$$