New Developments In The Theory Of Clustering

that's all very well in practice, but does it work in theory ?

Sergei Vassilvitskii (Yahoo! Research)
Suresh Venkatasubramanian (U. Utah)

Overview

What we will cover

A few of the recent theory results on clustering:

- Practical algorithms that have strong theoretical guarantees
- Models to explain behavior observed in practice

Overview

What we will not cover

The rest:

- Recent strands of theory of clustering such as metaclustering and privacy preserving clustering
- Clustering with distributional data assumptions
- Proofs

Outline

I Euclidean Clustering and `k-means` algorithm

II Bregman Clustering and `k-means`

III Stability

Outline

I Euclidean Clustering and `k-means` algorithm
- What to do to select initial centers (and what not to do)
- How long does `k-means` take to run in theory, practice and theoretical practice
- How to run `k-means` on large datasets

II Bregman Clustering and `k-means`

III Stability

Outline

I Euclidean Clustering and `k-means` algorithm
- What to do to select initial centers (and what not to do)
- How long does `k-means` take to run in theory, practice and theoretical practice
- How to run `k-means` on large datasets

II Bregman Clustering and `k-means`
- Bregman Clustering as generalization of `k-means`
- Performance Results

III Stability

Outline

I Euclidean Clustering and `k-means` algorithm
- What to do to select initial centers (and what not to do)
- How long does `k-means` take to run in theory, practice and theoretical practice
- How to run `k-means` on large datasets

II Bregman Clustering and `k-means`
- Bregman Clustering as generalization of `k-means`
- Performance Results

III Stability
- How to relate closeness in cost function to closeness in clusters.

Euclidean Clustering and k-means

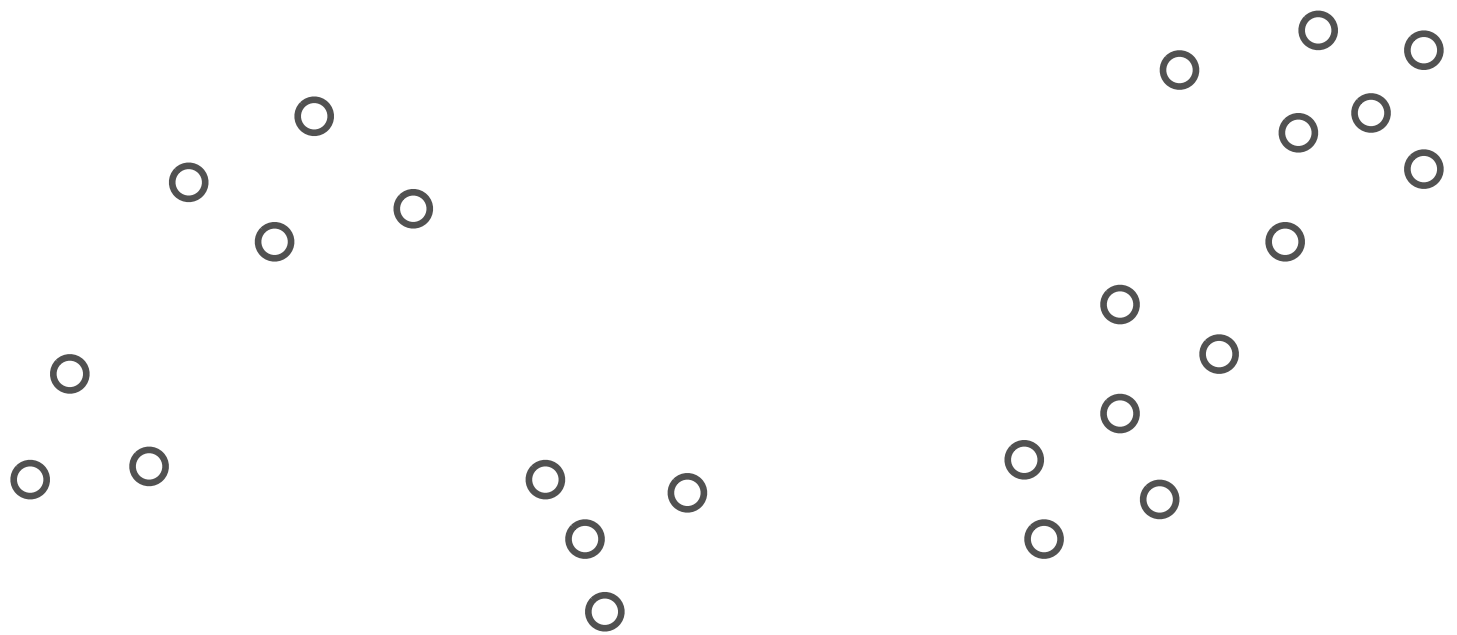
Introduction

What does it mean to cluster?

Given n points in $\mathbb{R}^d$ find the best way to split them into k groups.

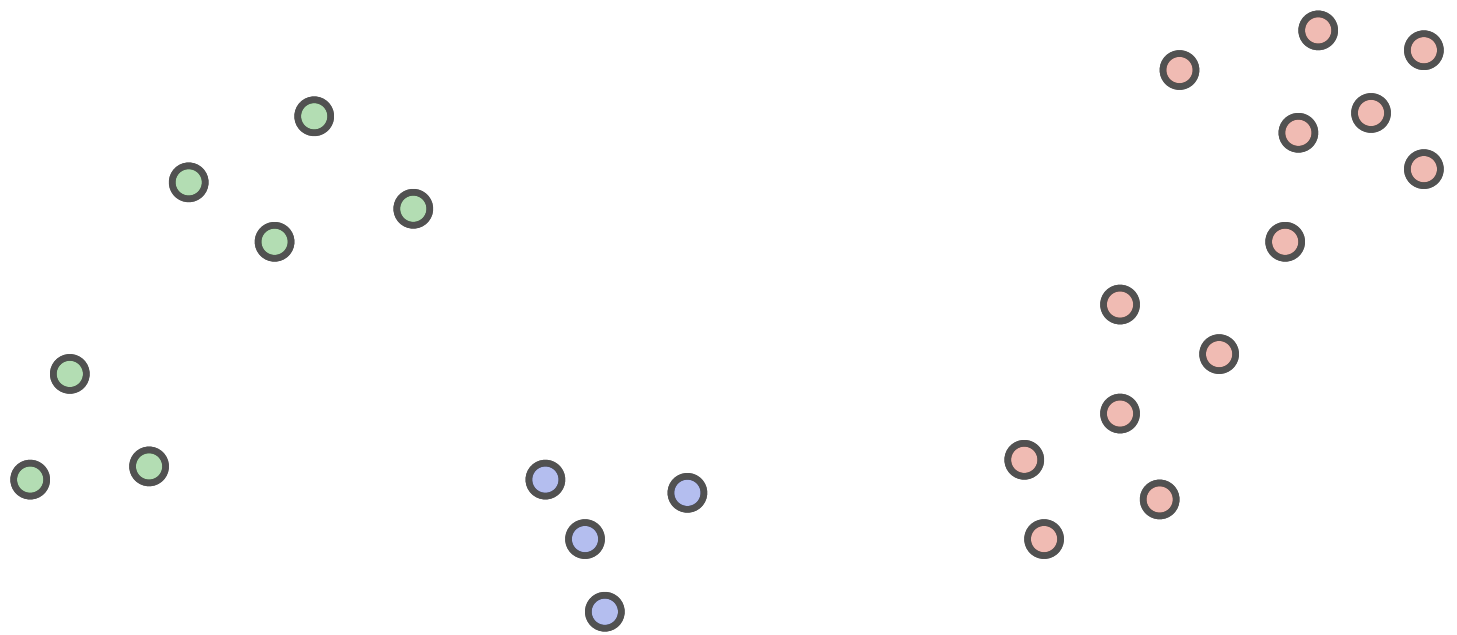
How do we define “best" ?

Example:



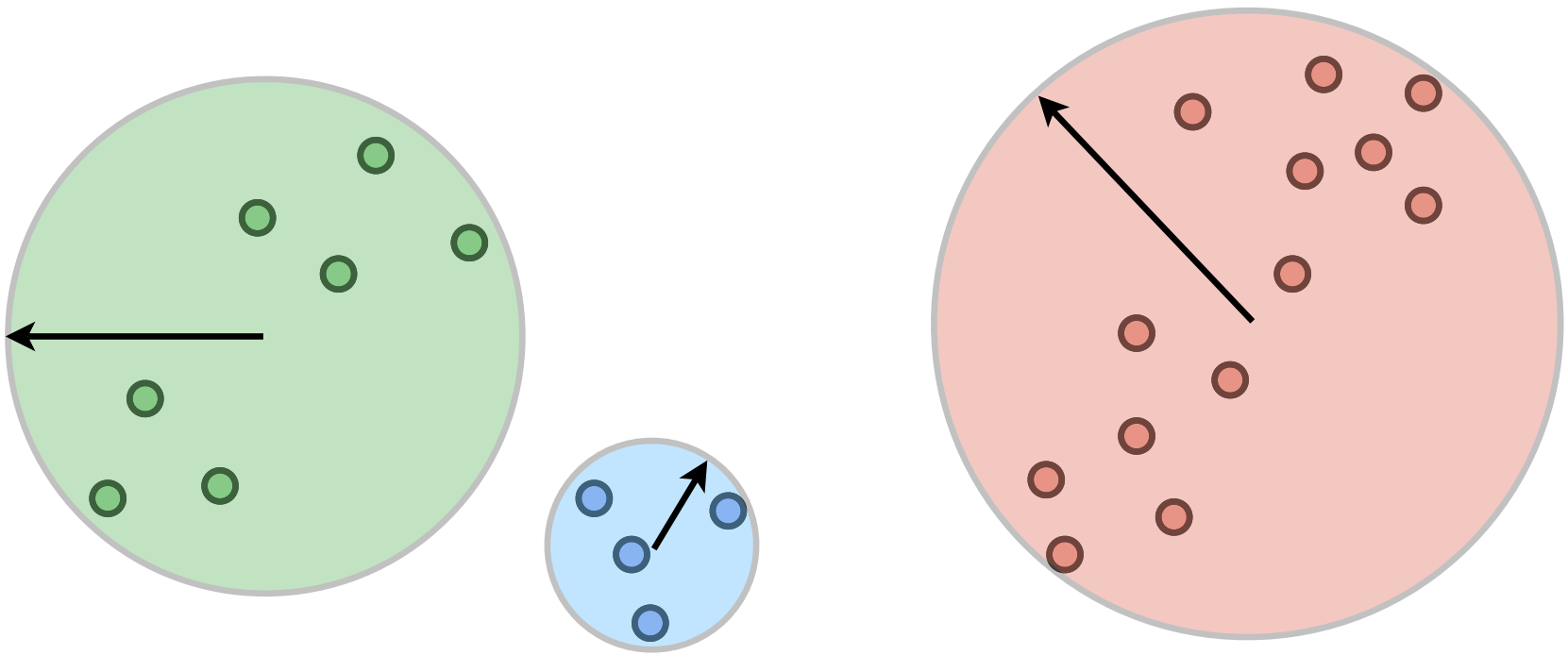
How do we define “best" ?

Example:



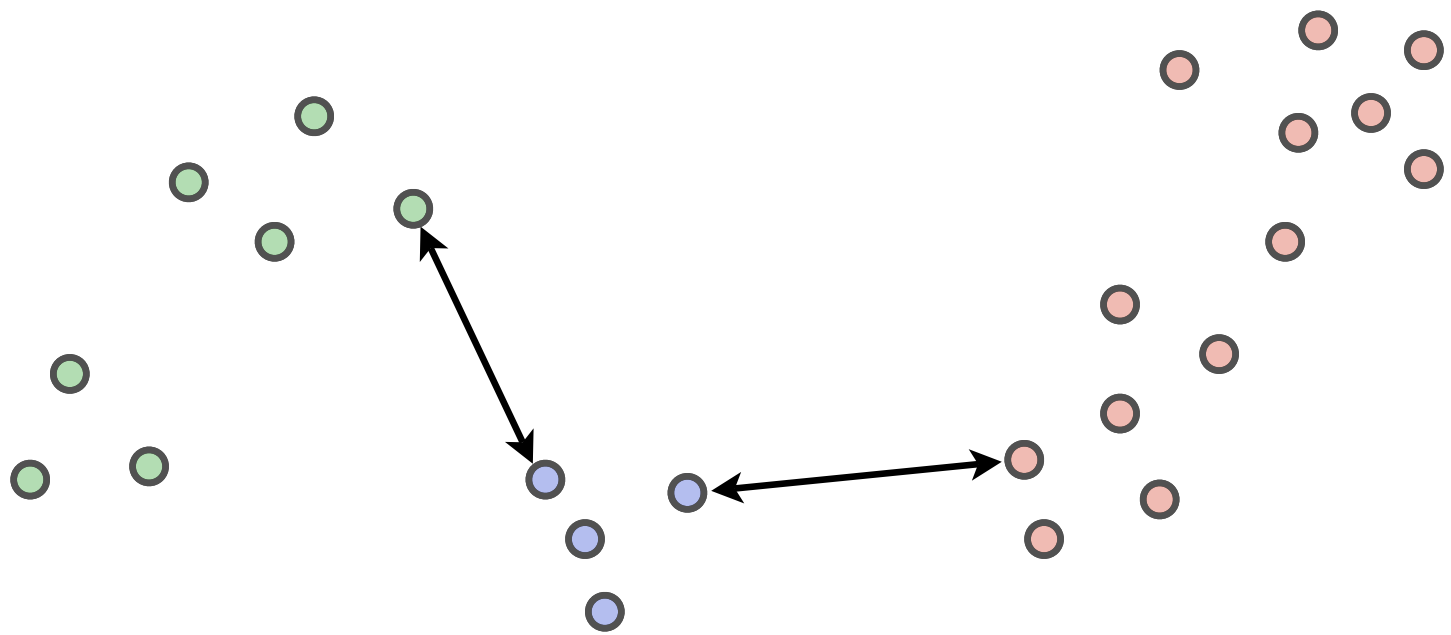
How do we define “best" ?

Minimize the maximum radius of a cluster



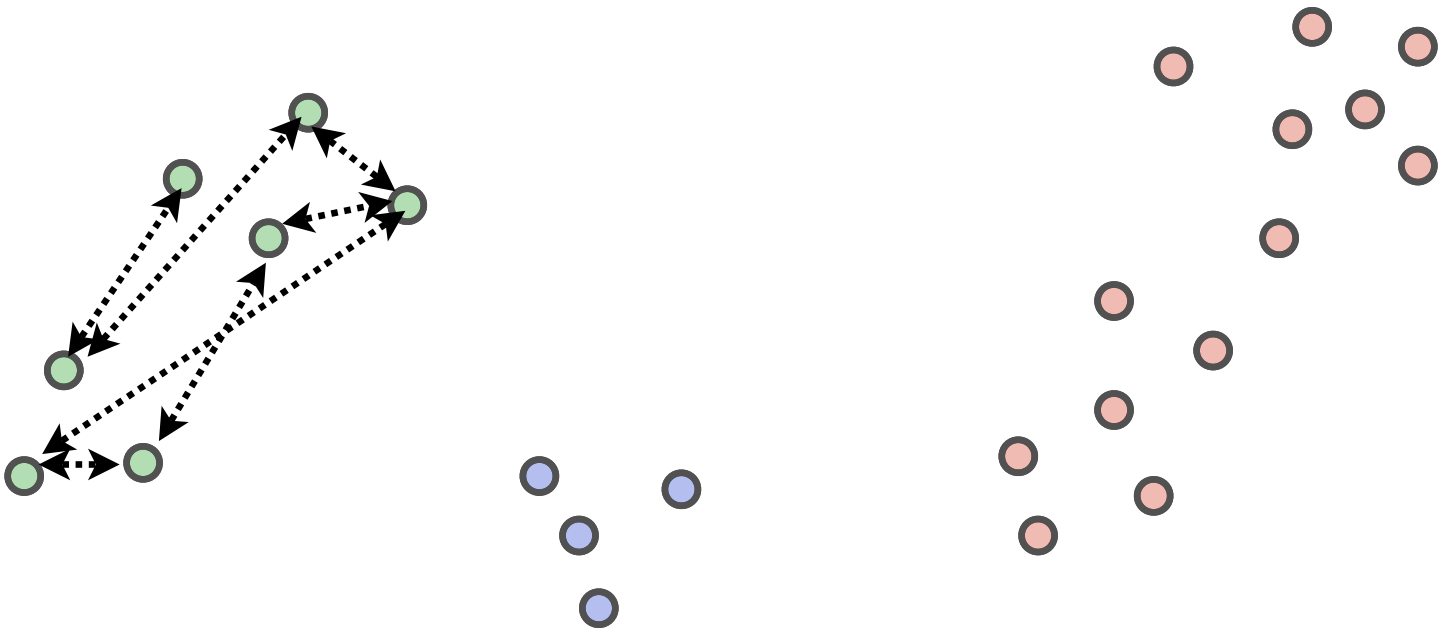
How do we define “best" ?

Maximize the average inter-cluster distance



How do we define “best" ?

Minimize the variance within each cluster.



How do we define “best" ?

Minimize the variance within each cluster.

Minimizing total variance

For each cluster $C_i \in \mathscr{C}$, $c_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$ is the expected location of a point in a cluster.
Then the variance of each cluster is:

$$\sum_{x \in C_i} = \|x - c_i\|^2$$

And the total objective is:

$$\phi = \sum_{c_i} \sum_{x \in C_i} \|x - c_i\|^2$$

Approximations

Minimizing Variance

Given X and k, find a clustering $\mathscr{C} = \{C_1, C_2, \ldots, C_k\}$ that minimizes: $\phi(X, \mathscr{C}) = \sum_{c_i} \sum_{x \in C_i} \|x - c_i\|^2$

Approximations

Minimizing Variance

Given X and k, find a clustering $\mathscr{C} = \{C_1, C_2, \ldots, C_k\}$ that minimizes: $\phi(X, \mathscr{C}) = \sum_{c_i} \sum_{x \in C_i} \|x - c_i\|^2$

Definition

Let ϕ^* denote the value of the optimum solution above. We say that a clustering $\mathscr{C}'$ is α-approximate if:

$$\phi^* \leq \phi(X, \mathscr{C}') \leq \alpha \cdot \phi^*$$

Approximations

Minimizing Variance

Given X and k, find a clustering $\mathscr{C} = \{C_1, C_2, \ldots, C_k\}$ that minimizes: $\phi(X, \mathscr{C}) = \sum_{c_i} \sum_{x \in C_i} \|x - c_i\|^2$

Solving this problem

This problem is NP-complete, even when the pointset X lies in two dimensions...

Approximations

Minimizing Variance

Given X and k, find a clustering $\mathscr{C} = \{C_1, C_2, \ldots, C_k\}$ that minimizes: $\phi(X, \mathscr{C}) = \sum_{c_i} \sum_{x \in C_i} \|x - c_i\|^2$

Solving this problem

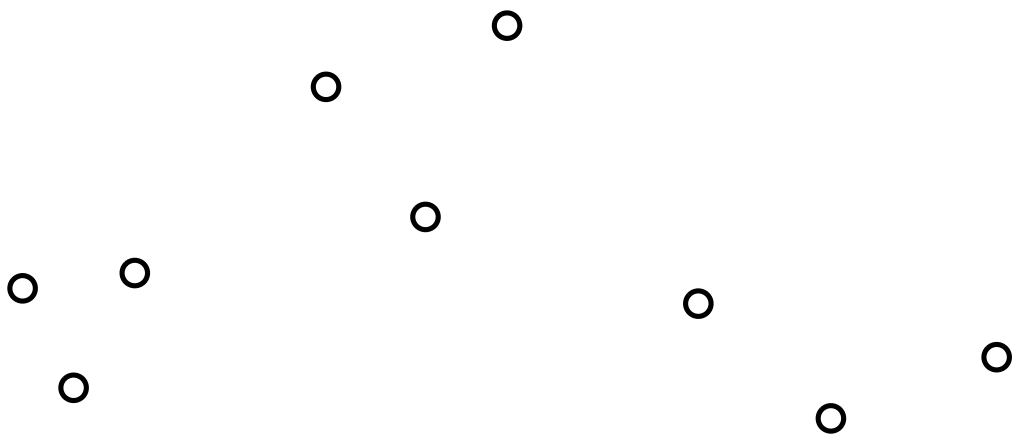
This problem is NP-complete, even when the pointset X lies in two dimensions...

...but we've been solving it for over 50 years! [S56][L57][M67]

k-means

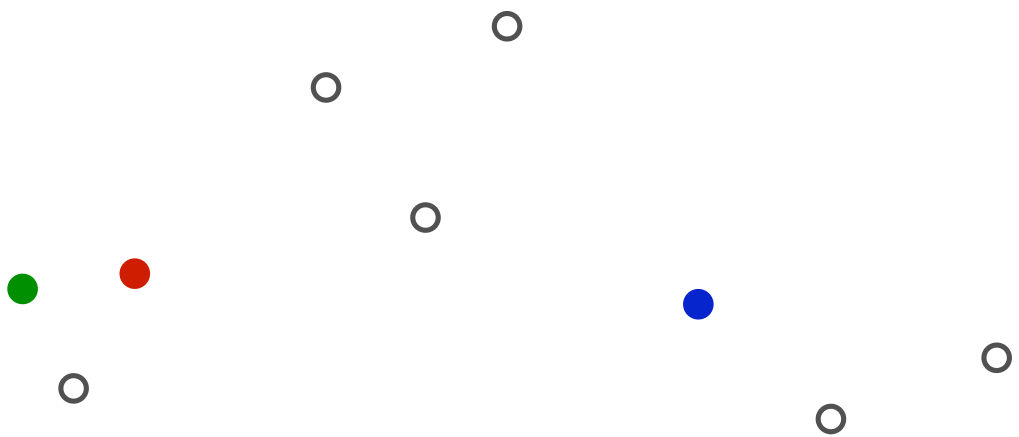
Example

Given a set of data points



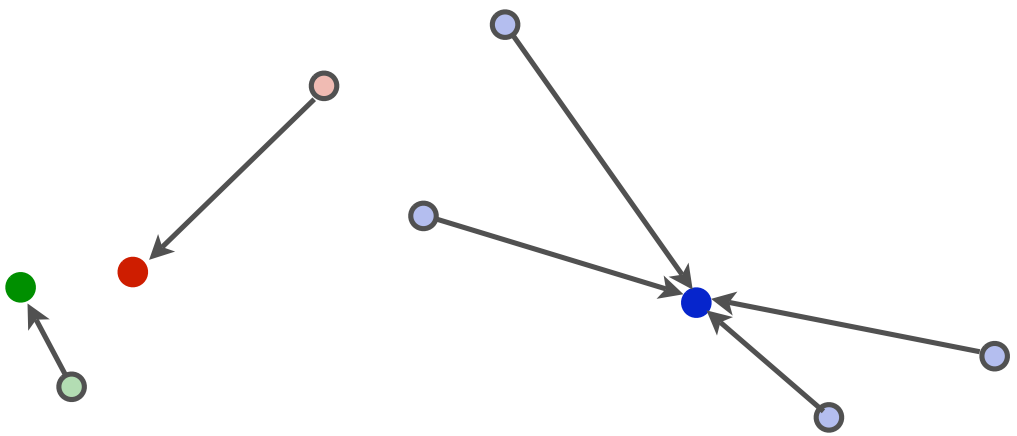
Example

Select initial centers at random



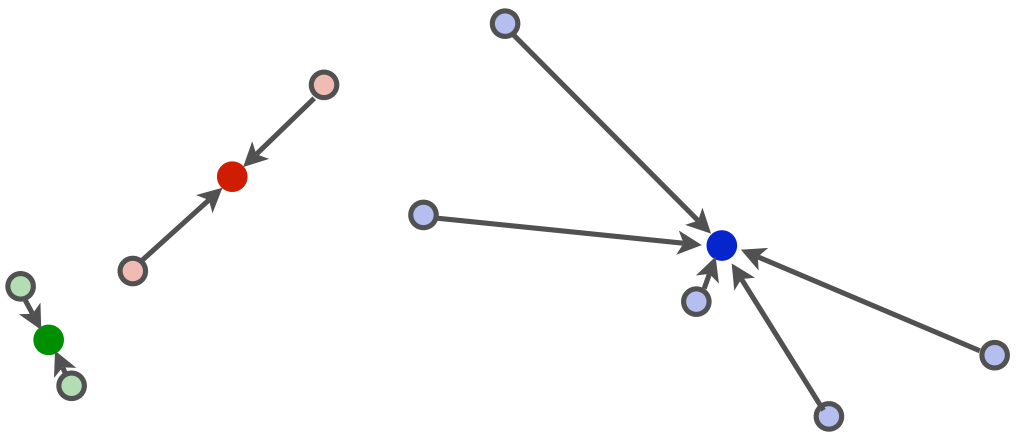
Example

Assign each point to nearest center



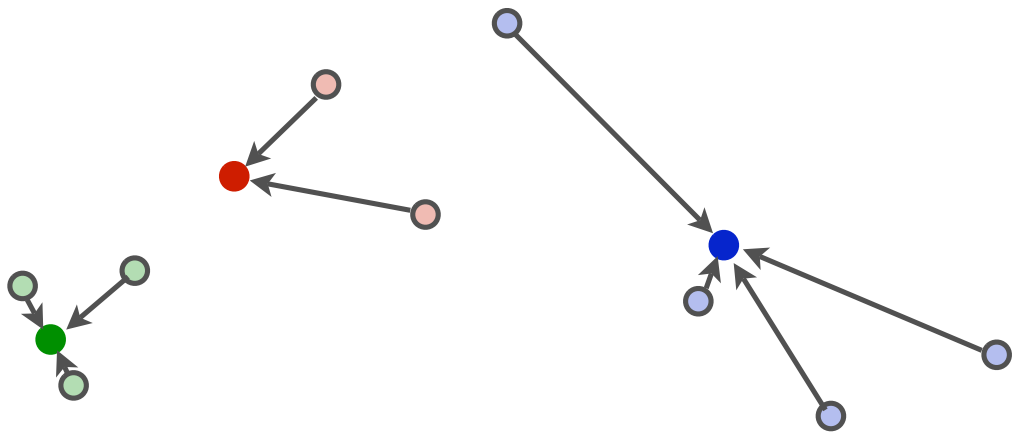
Example

Recompute optimum centers given a fixed clustering



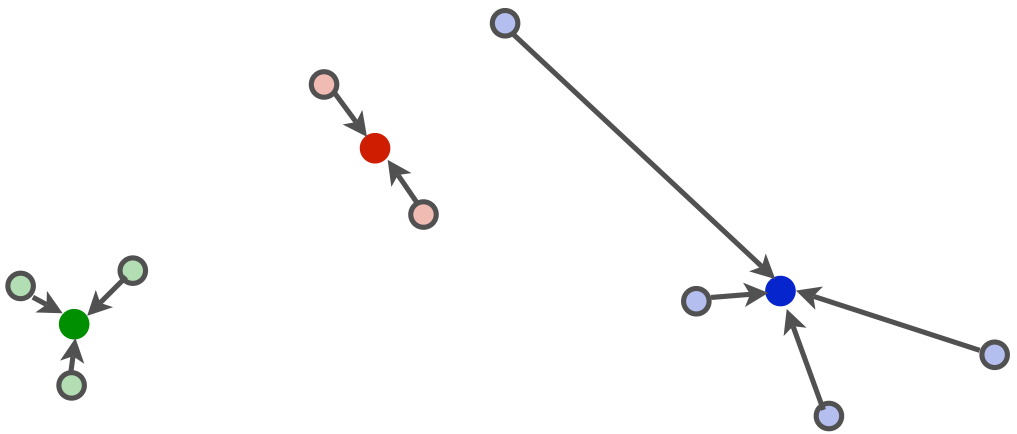
Example

Repeat



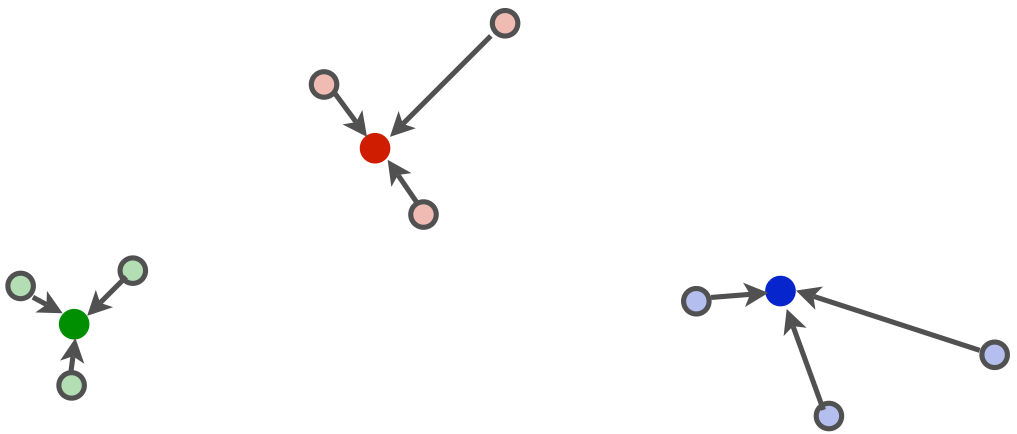
Example

Repeat



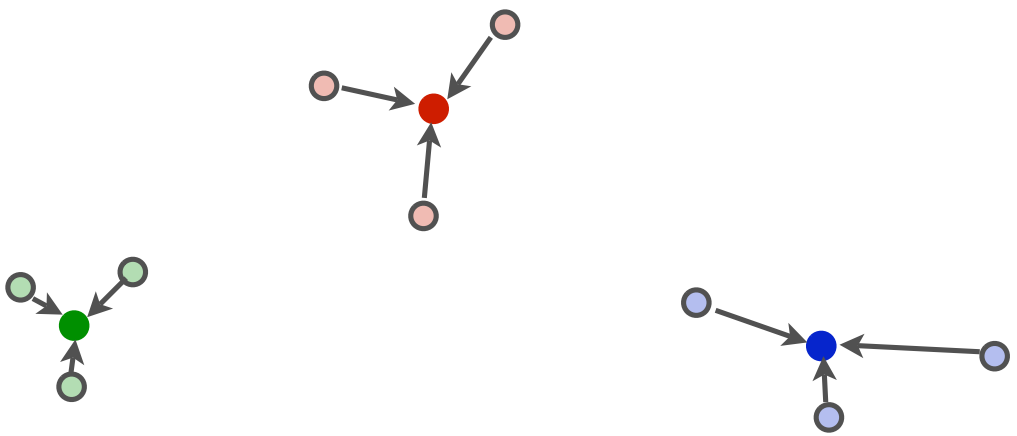
Example

Repeat



Example

Until the clustering doesn't change



Performance

This algorithm terminates!

Recall the total error:

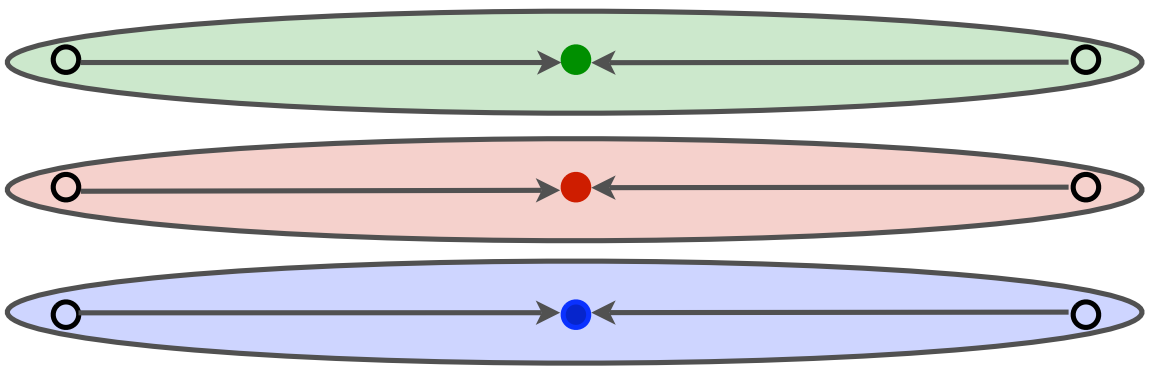
$$\phi(X, \mathscr{C}) = \sum_{c_i} \sum_{x \in C_i} \|x - c_i\|^2$$

In every iteration ϕ is reduced:

- Assigning each point to the nearest center reduces ϕ
- Given a fixed cluster, the mean is the optimal location for the center (requires proof)

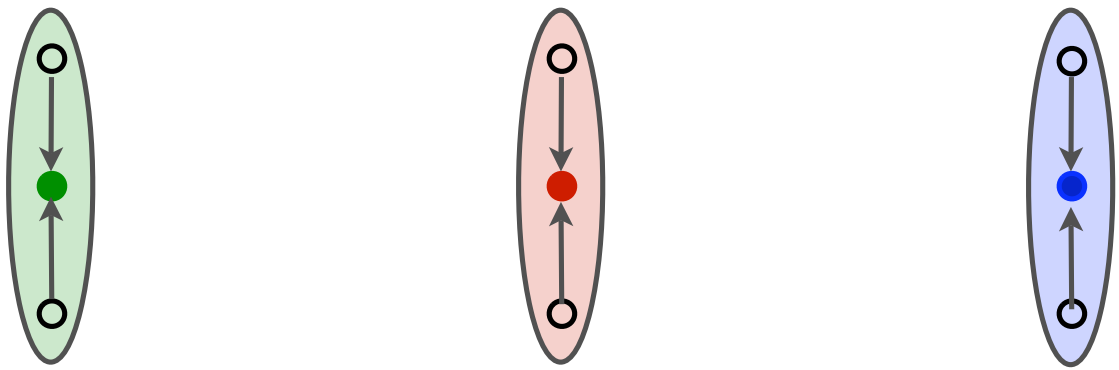
Performance

The algorithm finds a local minimum . . .



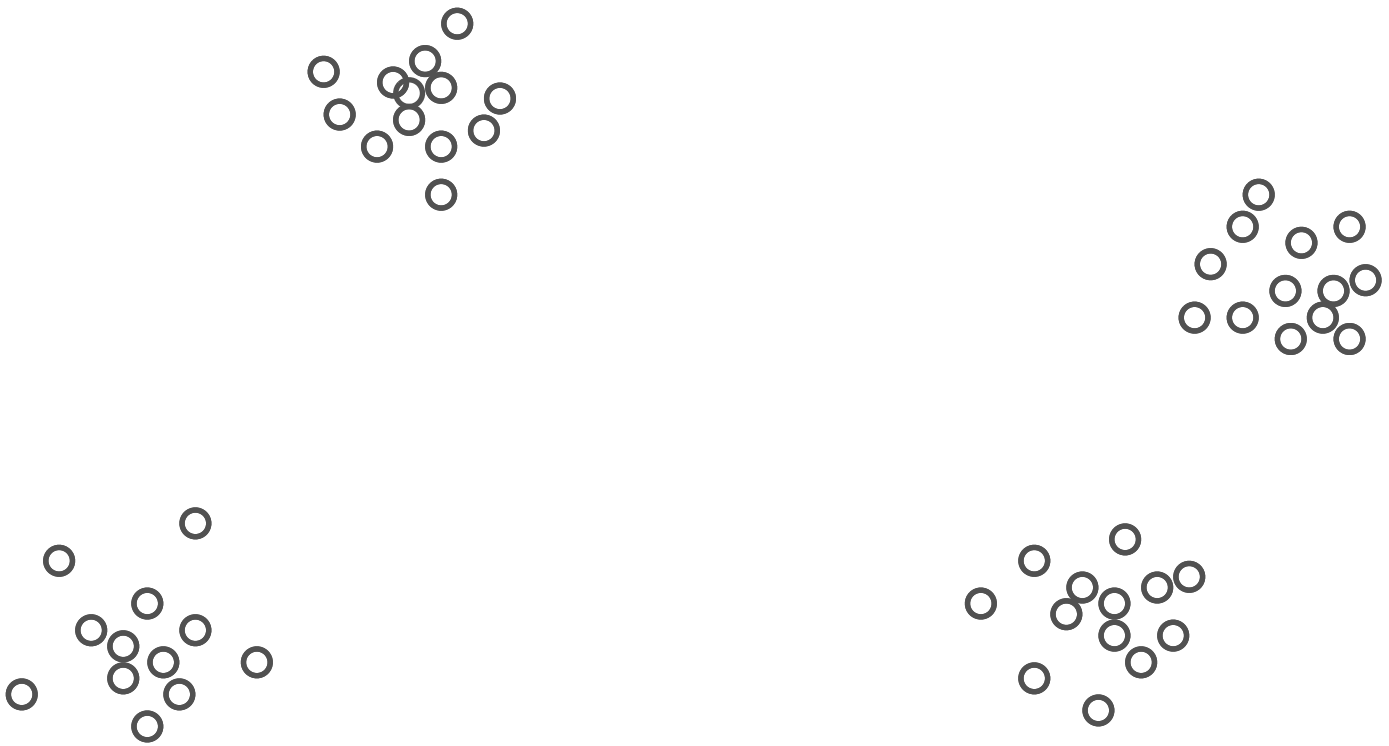
Performance

. . . that's potentially arbitrarily worse than optimum solution



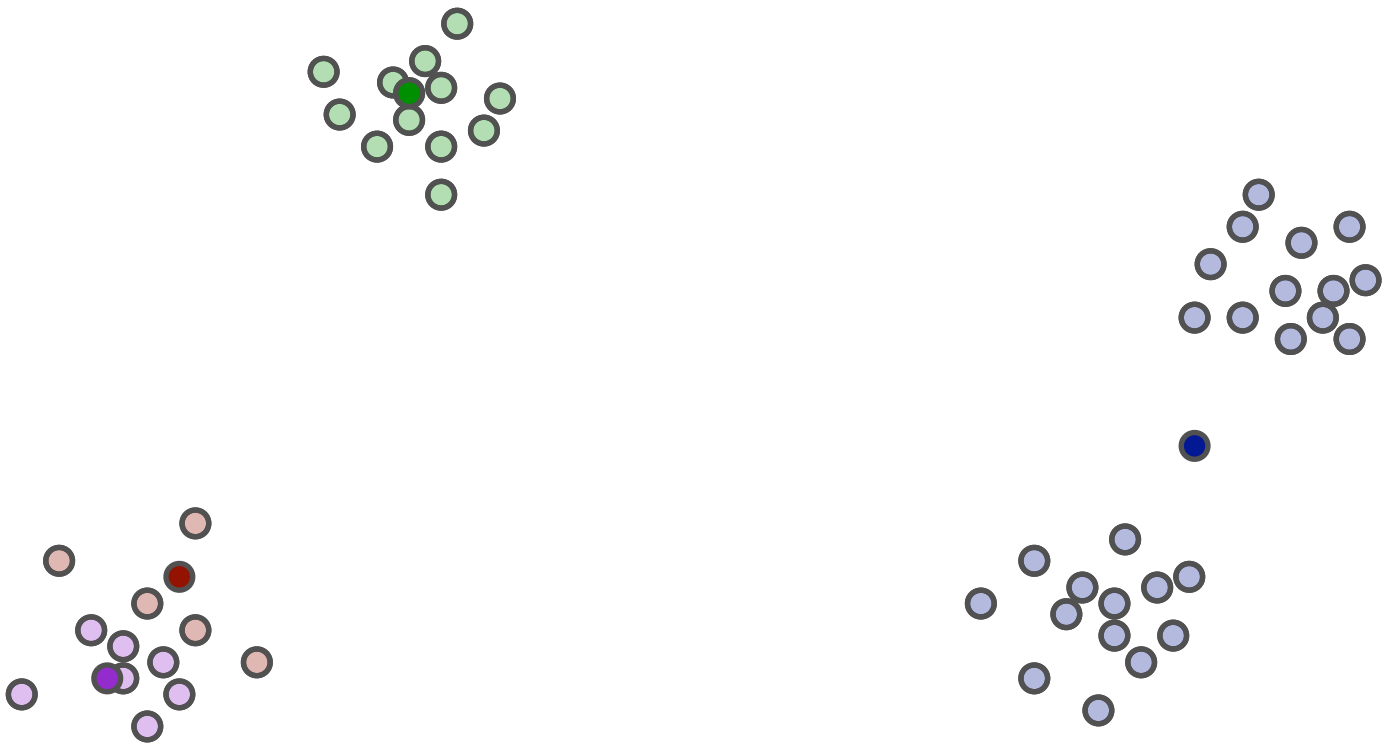
Performance

But does this really happen?



Performance

But does this really happen? YES!



Performance

Finding a good set of initial points is a black art

Performance

Finding a good set of initial points is a black art

- Try many times with different random seeds
  - Most common method
  - Has limited benefit even in case of Gaussians

Performance

Finding a good set of initial points is a black art

- Try many times with different random seeds
  - Most common method
  - Has limited benefit even in case of Gaussians
- Find a different way to initialize centers
  - Hundreds of heuristics
  - Including pre & post processing ideas

Performance

Finding a good set of initial points is a black art

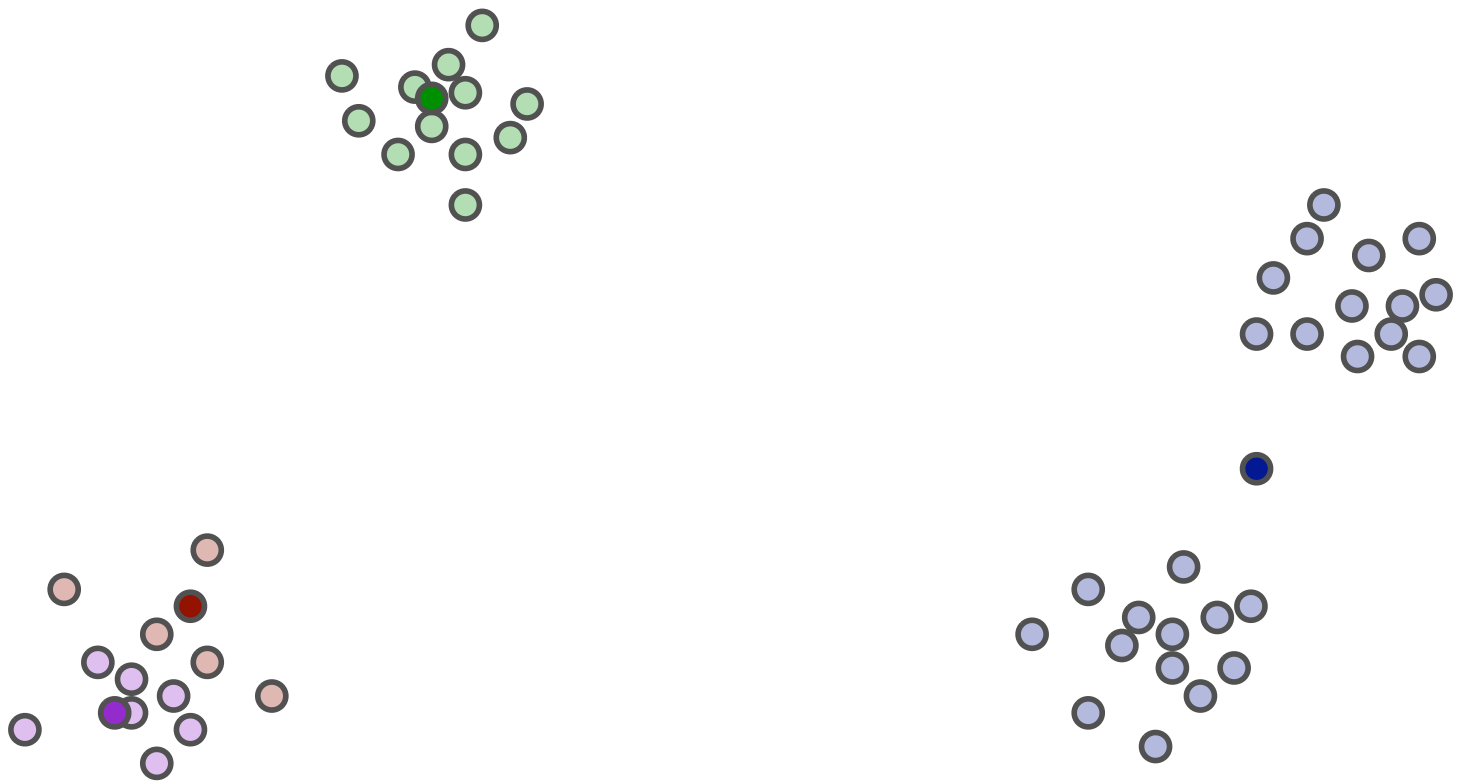
- Try many times with different random seeds
  - Most common method
  - Has limited benefit even in case of Gaussians
- Find a different way to initialize centers
  - Hundreds of heuristics
  - Including pre & post processing ideas

There exists a fast and simple initialization scheme with provable performance guarantees

Random Initializations on Gaussians

Random Initializations on Gaussians

Some Gaussians are combined

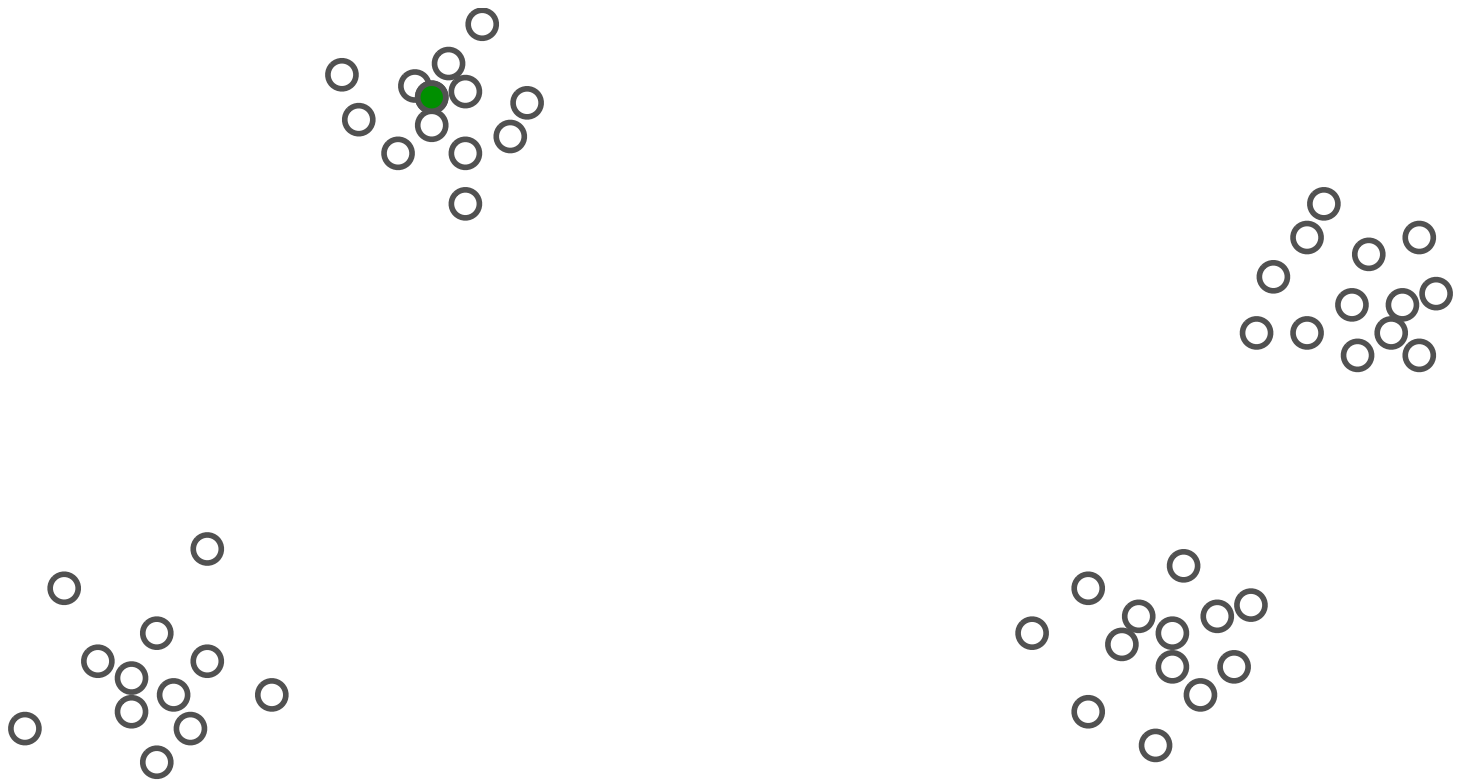


Seeding on Gaussians

But the Gaussian case has an easy fix: use a furthest point heuristic

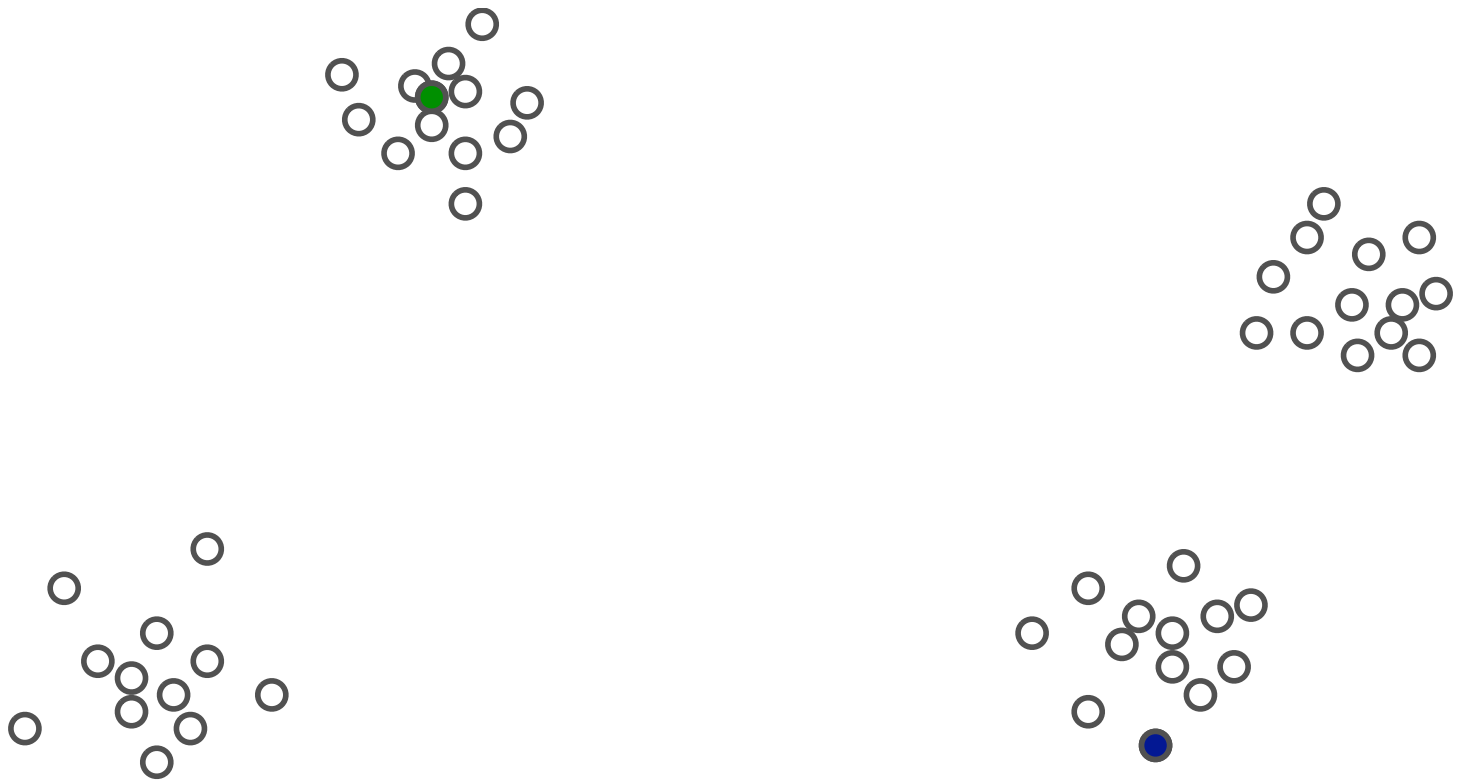
Seeding on Gaussians

But the Gaussian case has an easy fix: use a furthest point heuristic



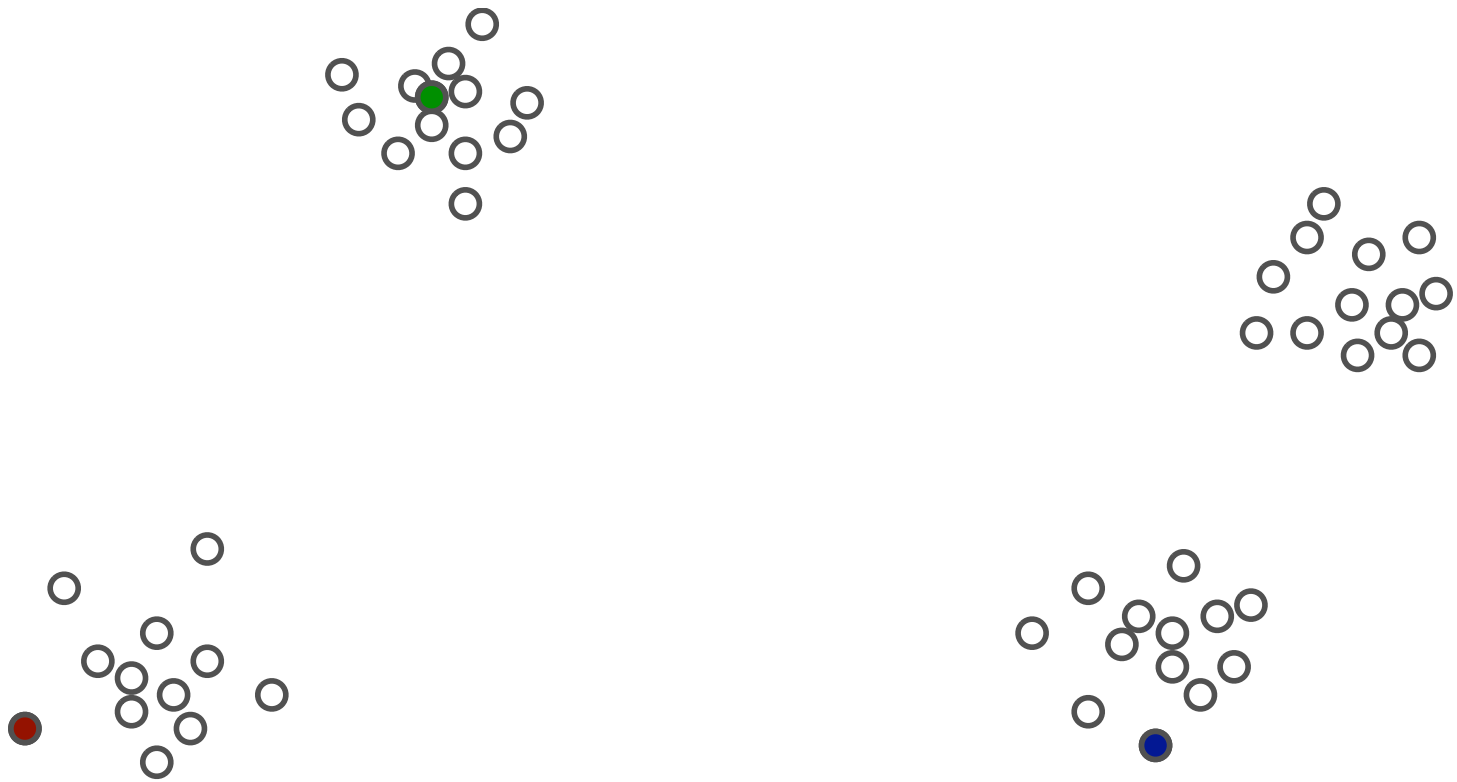
Seeding on Gaussians

But the Gaussian case has an easy fix: use a furthest point heuristic



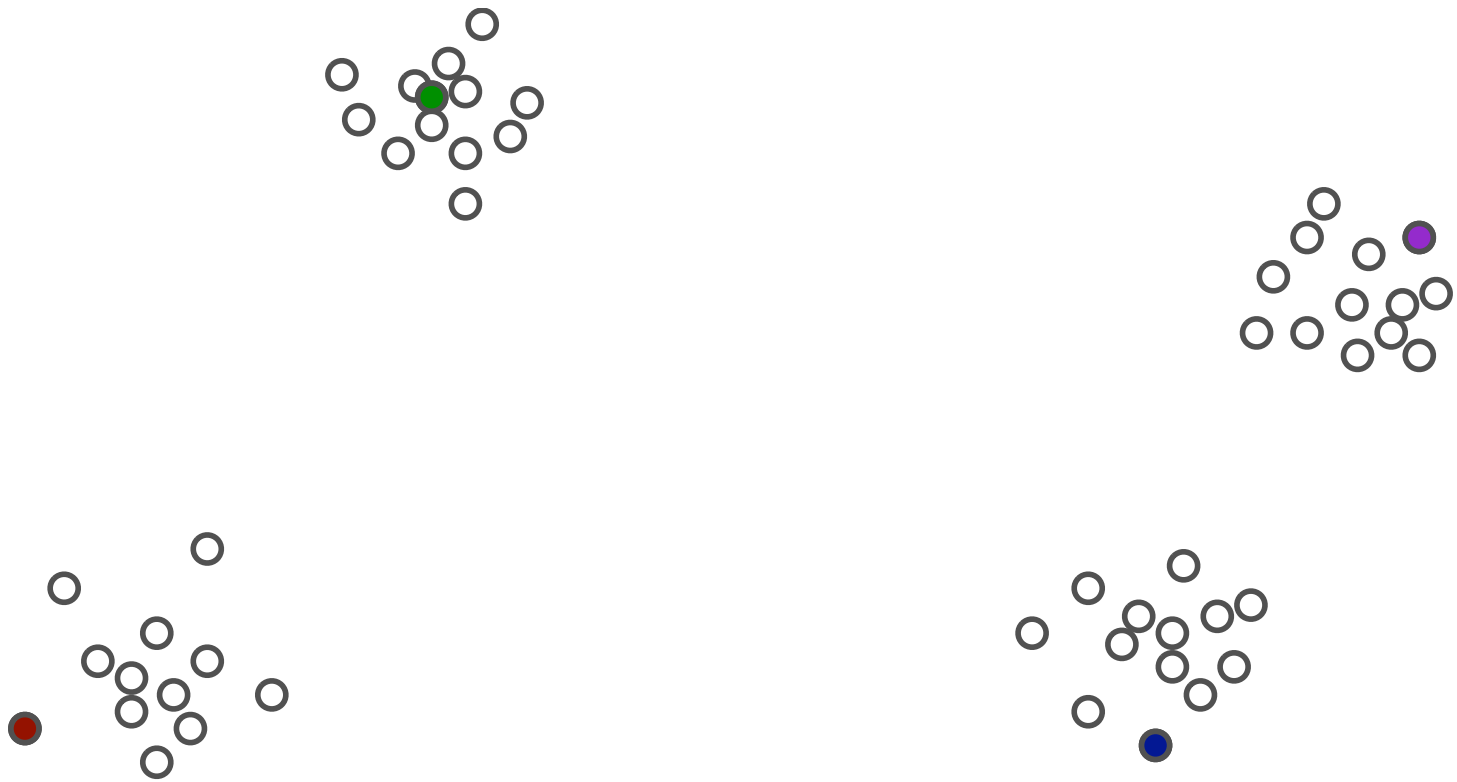
Seeding on Gaussians

But the Gaussian case has an easy fix: use a furthest point heuristic



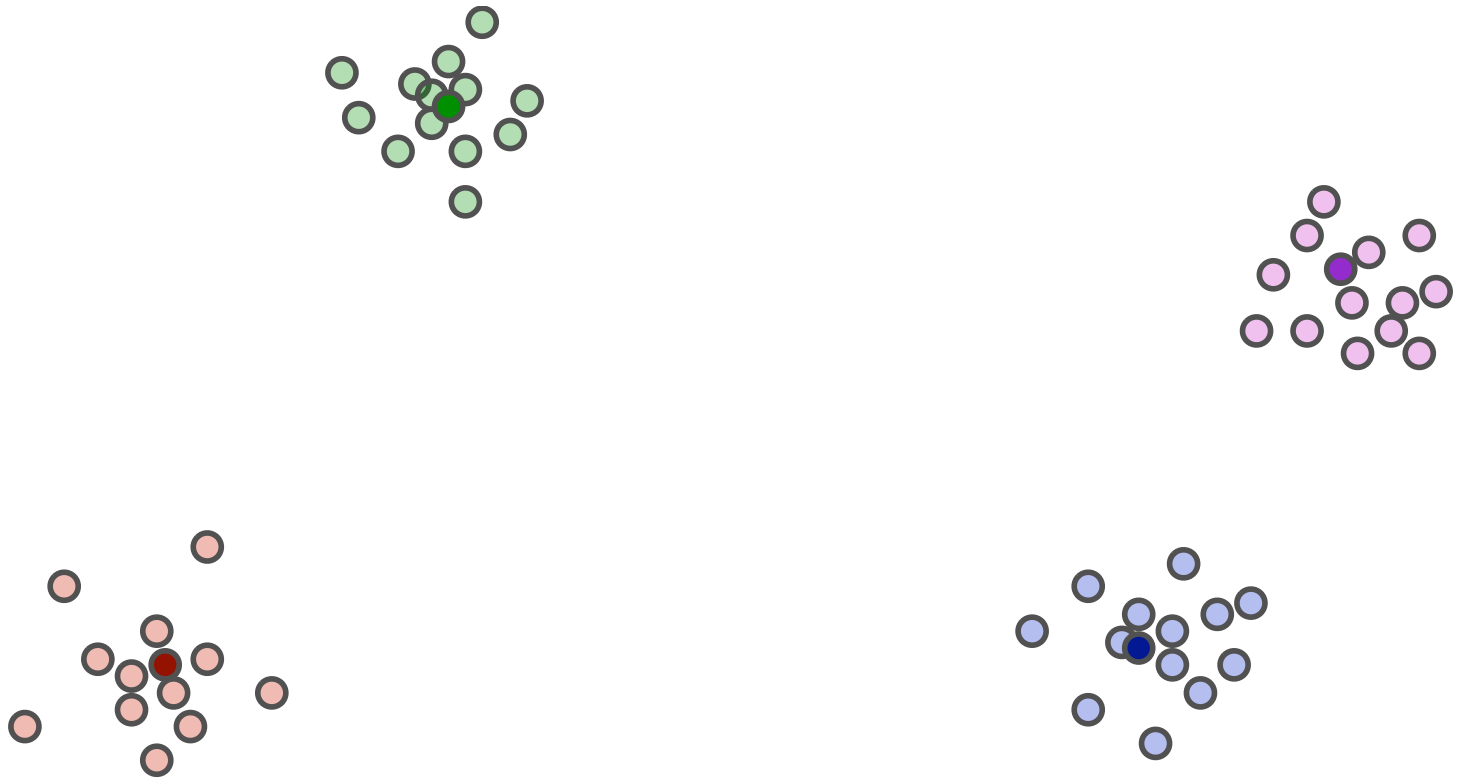
Seeding on Gaussians

But the Gaussian case has an easy fix: use a furthest point heuristic



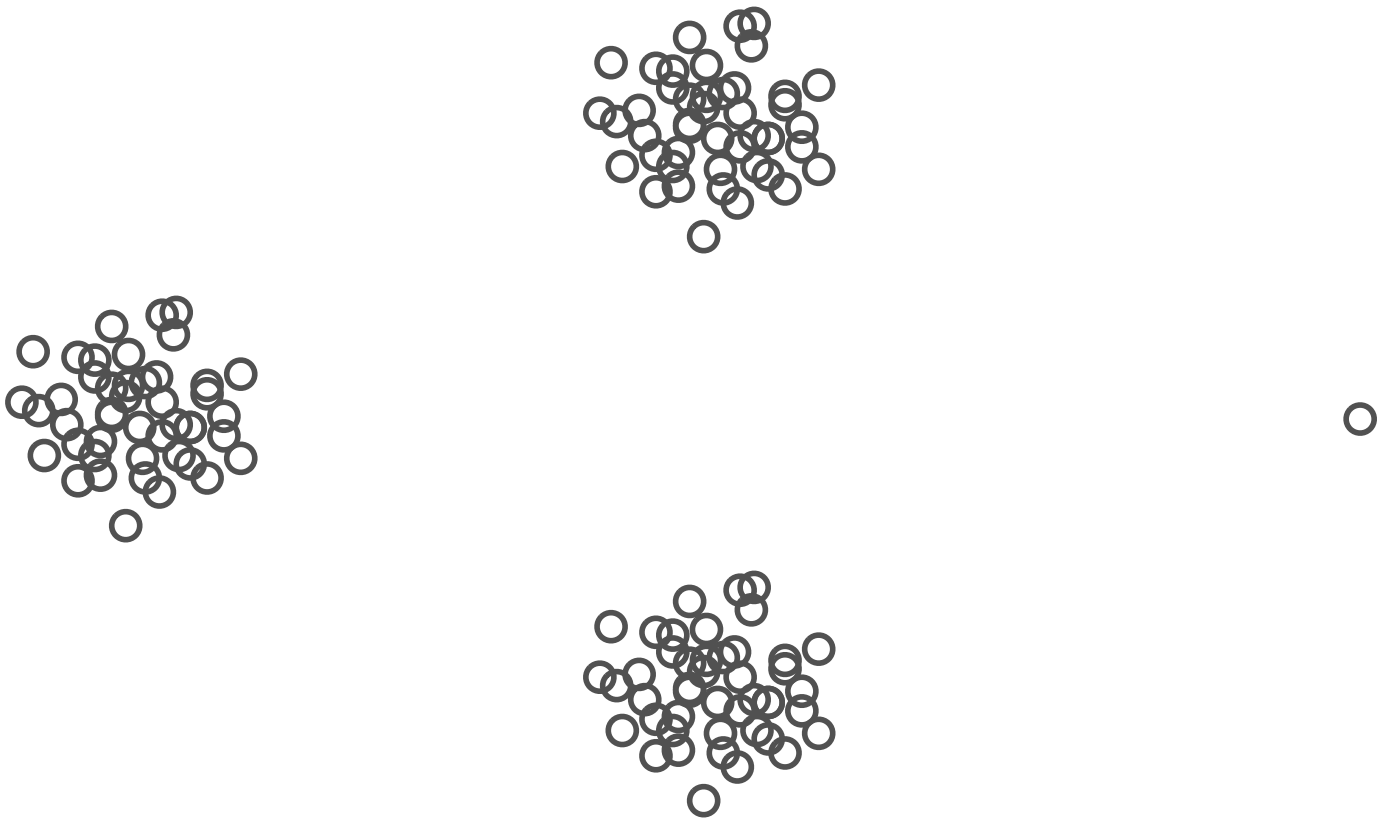
Seeding on Gaussians

But the Gaussian case has an easy fix: use a furthest point heuristic



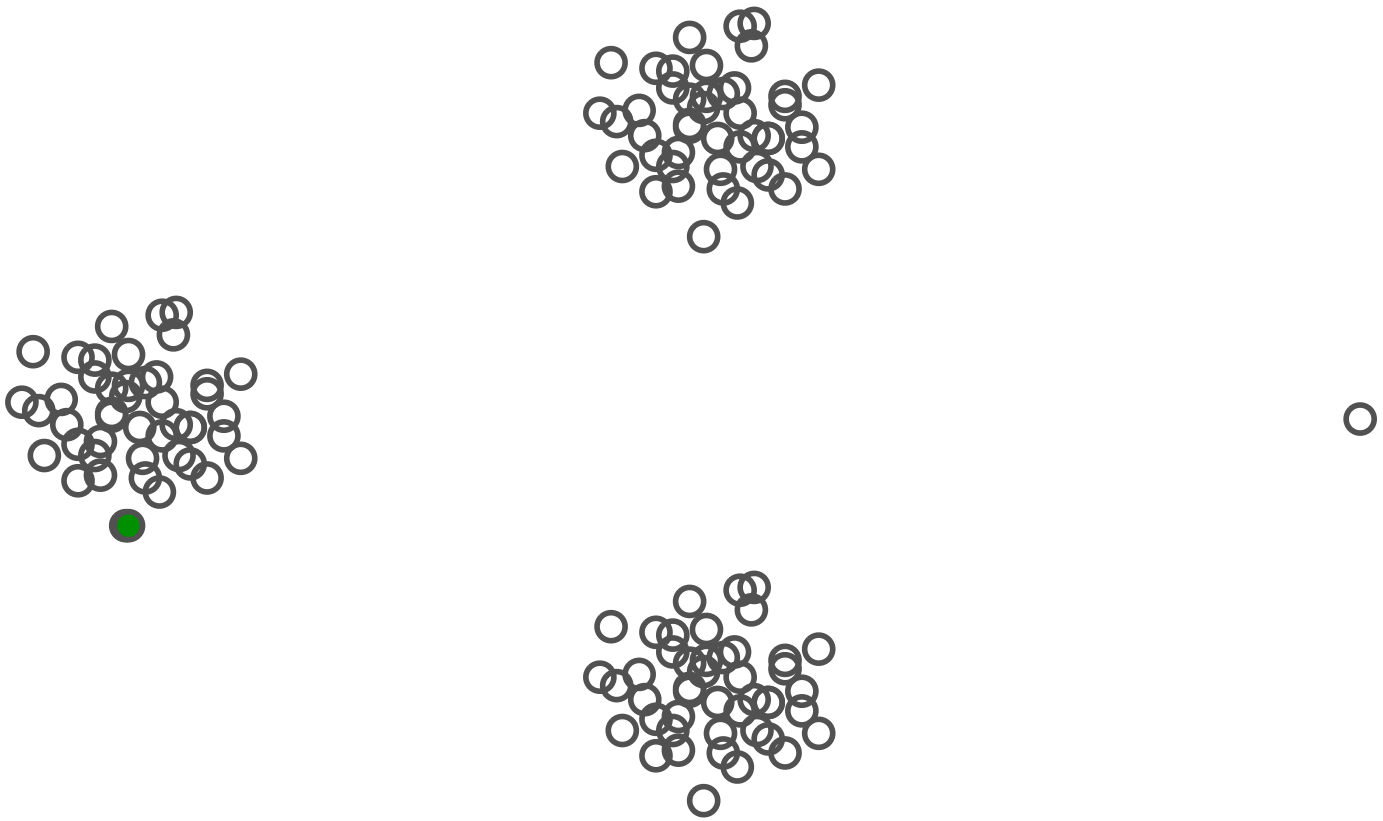
Seeding on Gaussians

But this fix is overly sensitive to outliers



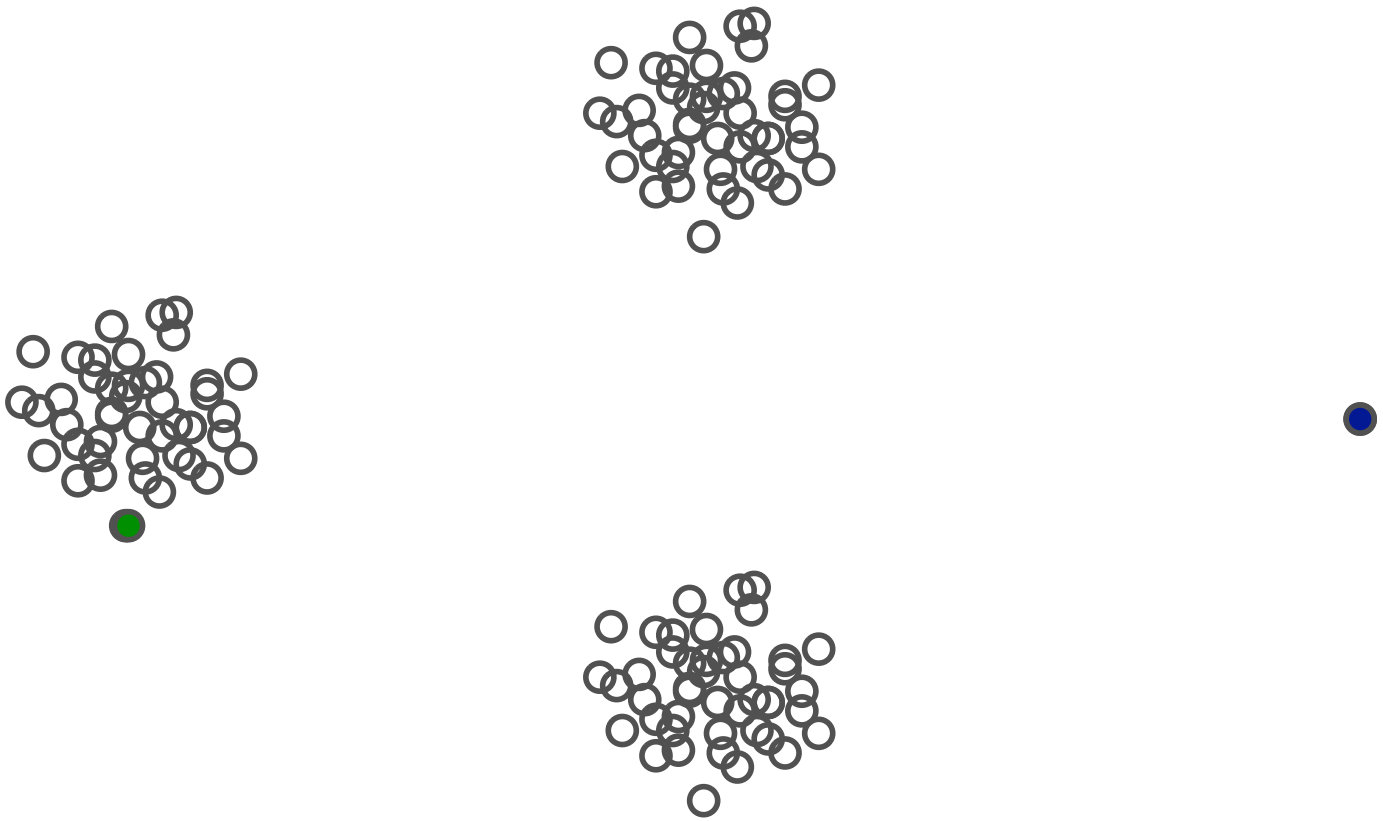
Seeding on Gaussians

But this fix is overly sensitive to outliers



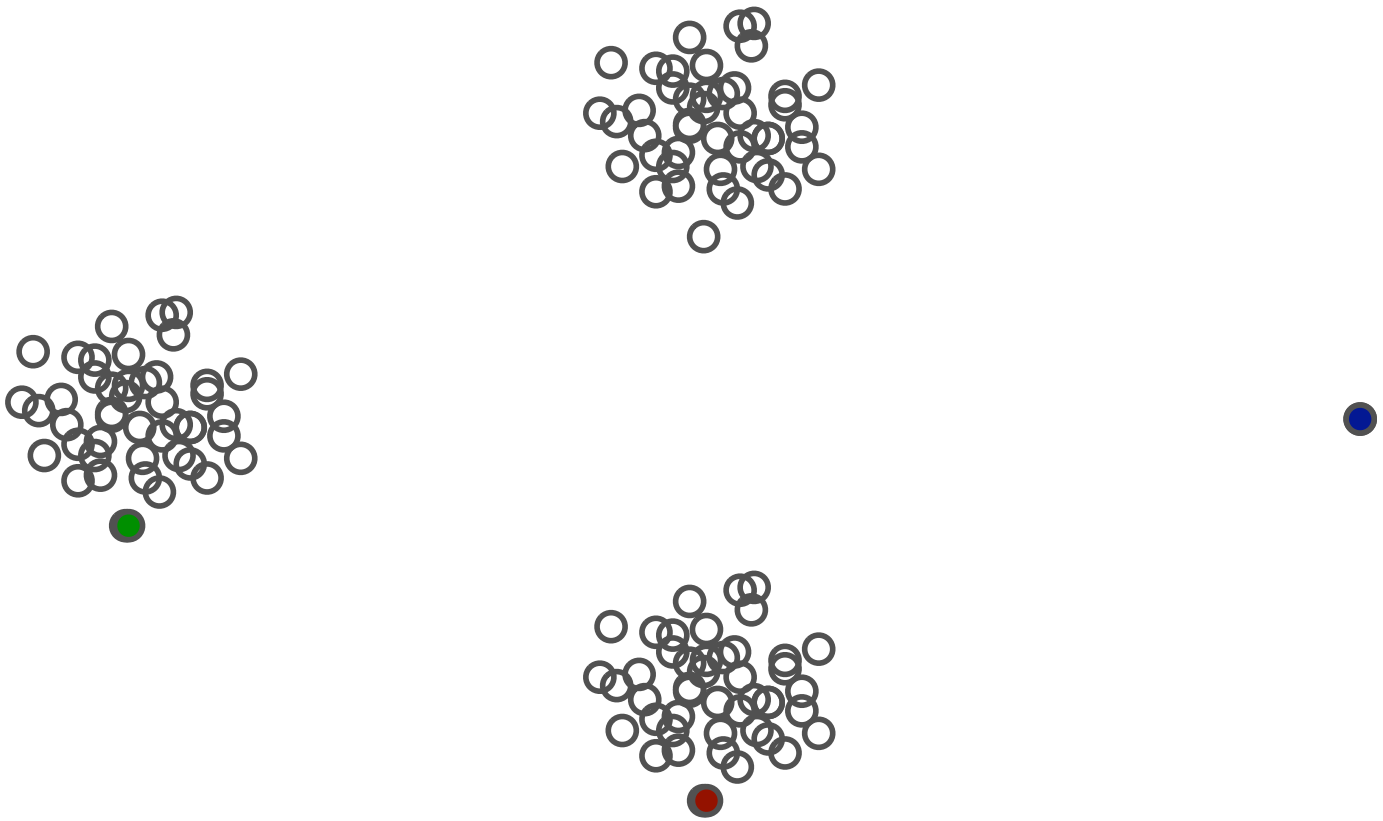
Seeding on Gaussians

But this fix is overly sensitive to outliers



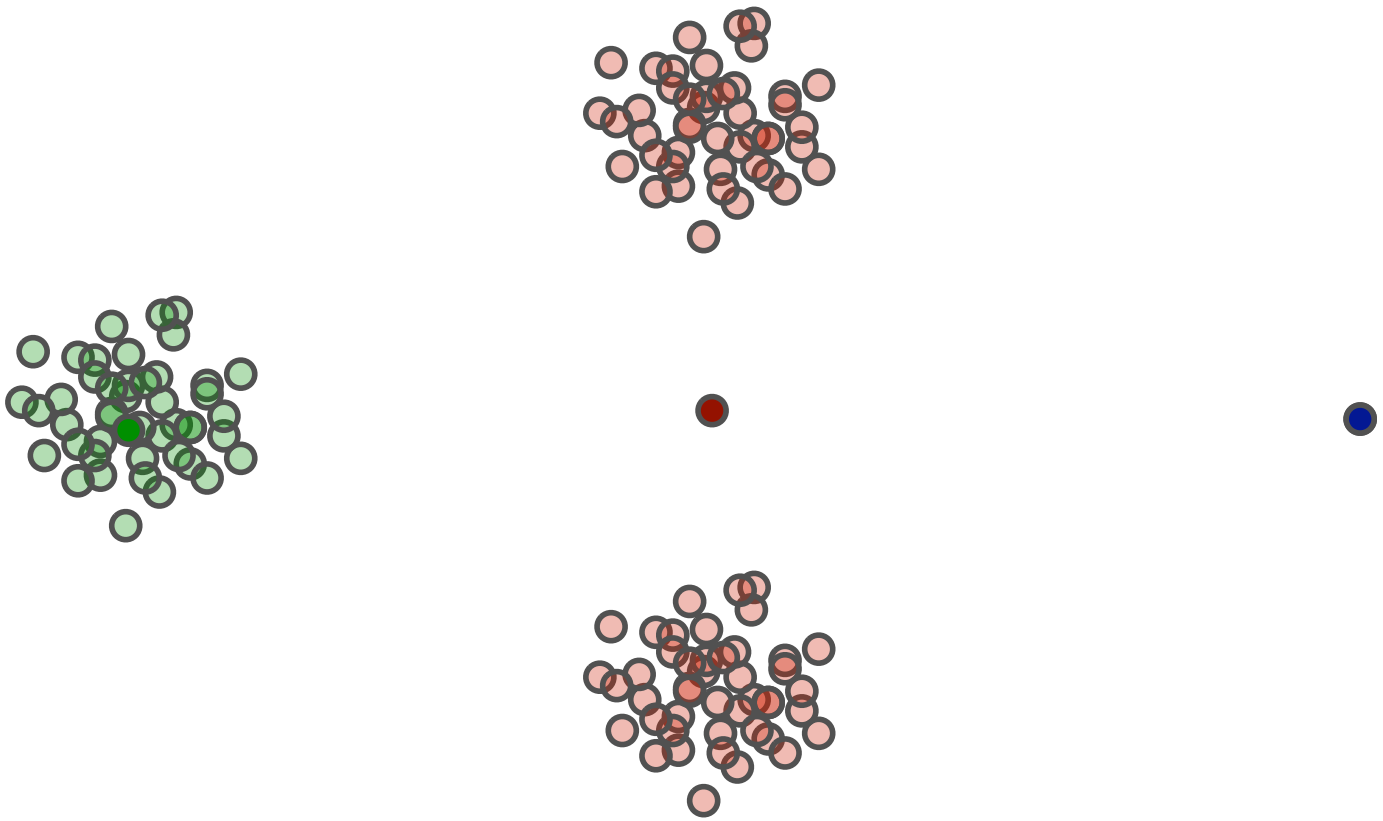
Seeding on Gaussians

But this fix is overly sensitive to outliers



Seeding on Gaussians

But this fix is overly sensitive to outliers



What if we interpolate between the two methods?

What if we interpolate between the two methods?

Let $D(x)$ be the distance between a point x and its nearest cluster center. Chose the next point proportionally to $D^{\alpha}(x)$.

What if we interpolate between the two methods?

Let $D(x)$ be the distance between a point x and its nearest cluster center. Chose the next point proportionally to $D^{\alpha}(x)$.

- $\alpha = 0$ $\longrightarrow$ Random initialization

What if we interpolate between the two methods?

Let $D(x)$ be the distance between a point x and its nearest cluster center. Chose the next point proportionally to $D^{\alpha}(x)$.

- $\alpha = 0 \longrightarrow$ Random initialization
- $\alpha = \infty \longrightarrow$ Furthest point heuristic

What if we interpolate between the two methods?

Let $D(x)$ be the distance between a point x and its nearest cluster center. Chose the next point proportionally to $D^{\alpha}(x)$.

- $\alpha = 0 \longrightarrow$ Random initialization
- $\alpha = \infty \longrightarrow$ Furthest point heuristic
- $\alpha = 2 \longrightarrow$ k-means++

What if we interpolate between the two methods?

Let $D(x)$ be the distance between a point x and its nearest cluster center. Chose the next point proportionally to $D^{\alpha}(x)$.

- $\alpha = 0 \longrightarrow$ Random initialization
- $\alpha = \infty \longrightarrow$ Furthest point heuristic
- $\alpha = 2 \longrightarrow$ k-means++

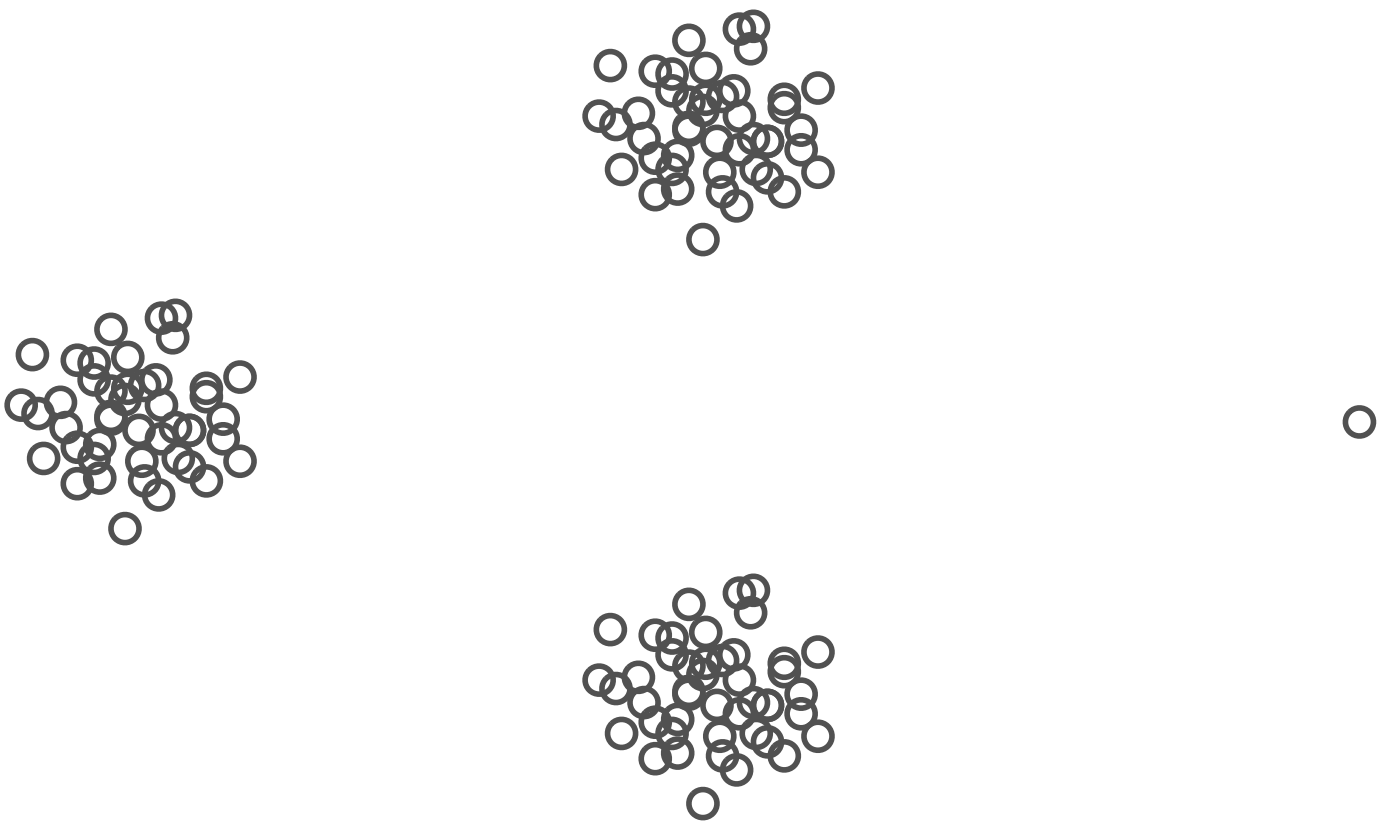
More generally

Set the probability of selecting a point proportional to its contribution to the overall error.

- If minimizing $\sum_{c_i} \sum_{x \in C_i} \|x - c_i\|$, sample according to D.
- If minimizing $\sum_{c_i} \sum_{c \in C_i} \|x - c_i\|_{\infty}$, sample according to D^{∞} (take the furthest point).

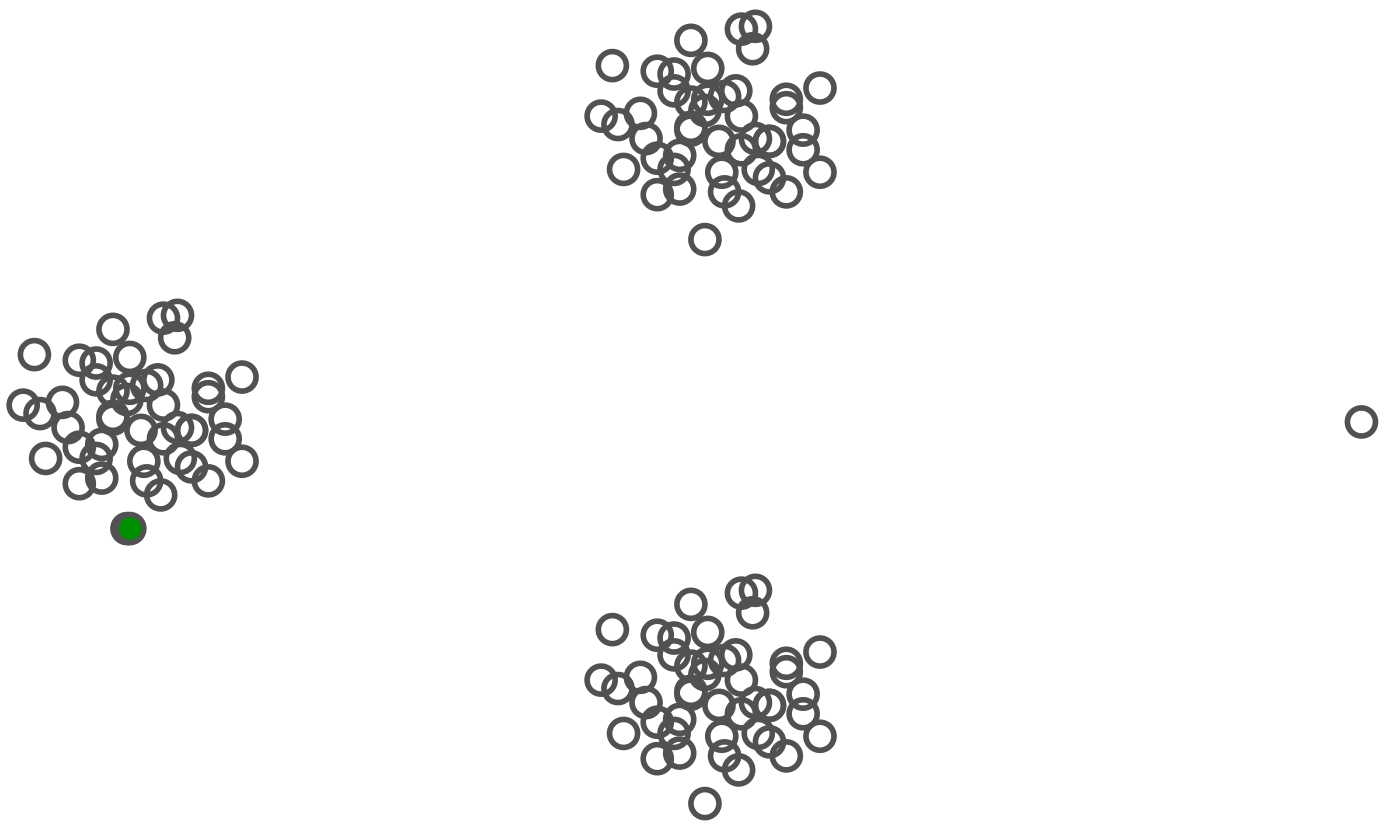
Example of k-means++

If the data set looks Gaussian...



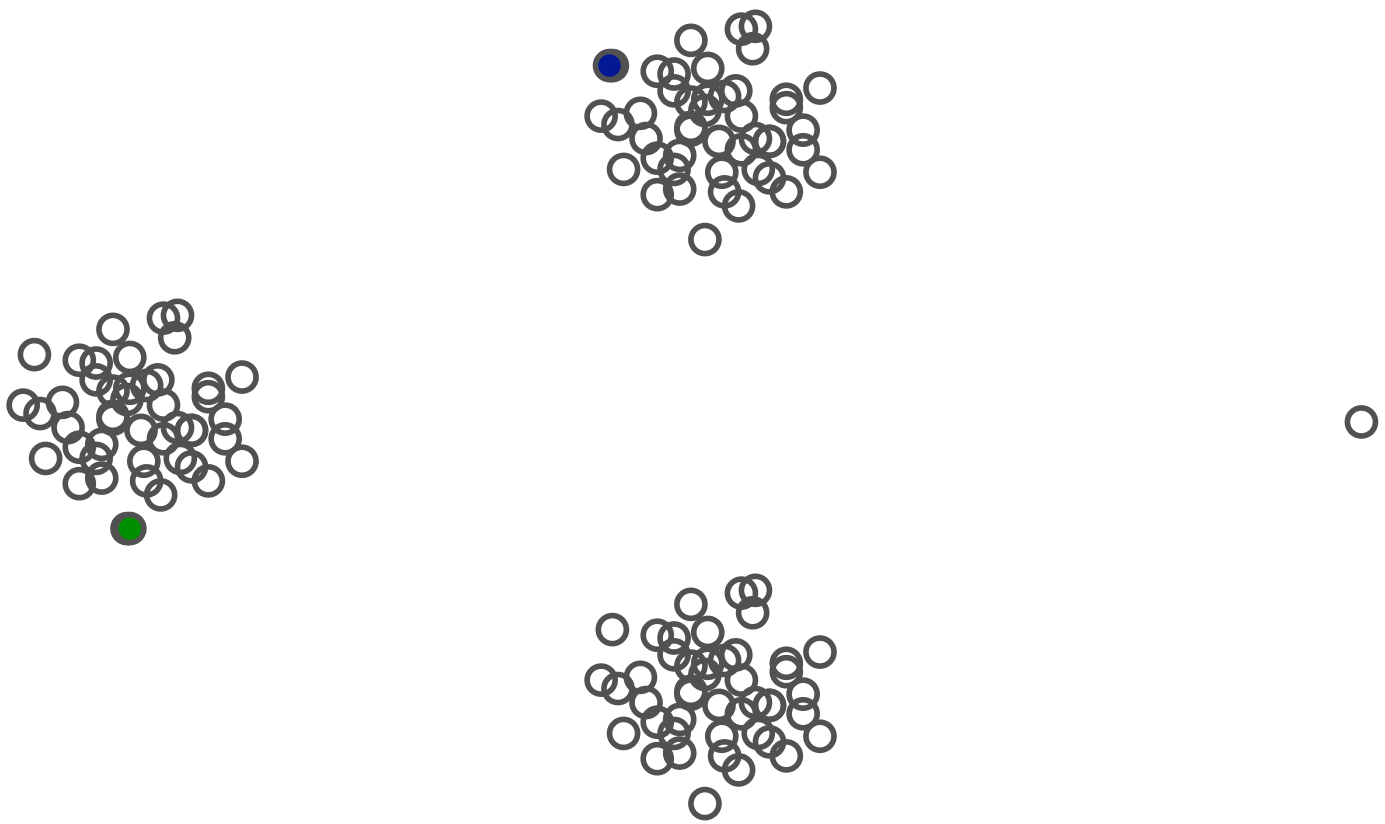
Example of k-means++

If the data set looks Gaussian...



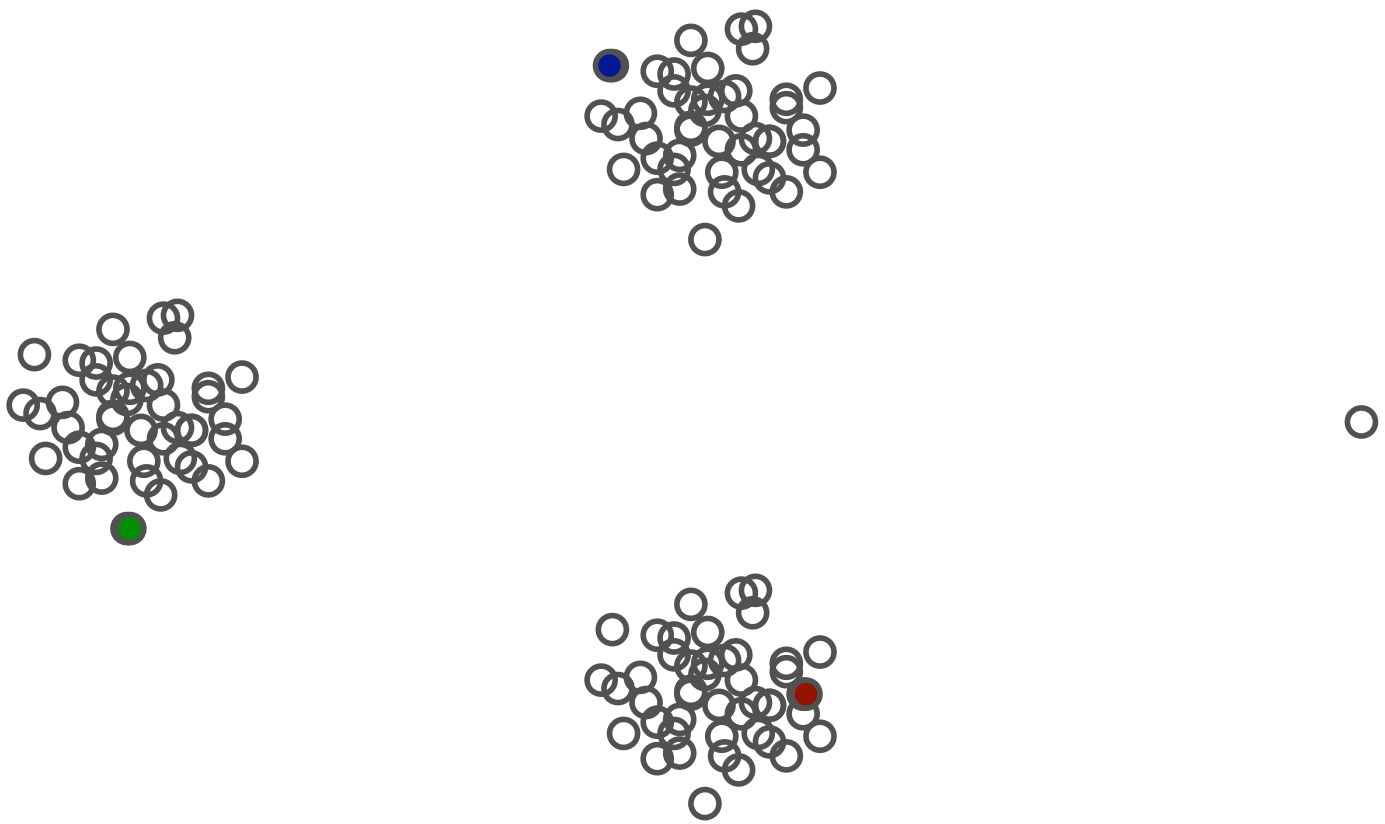
Example of k-means++

If the data set looks Gaussian...



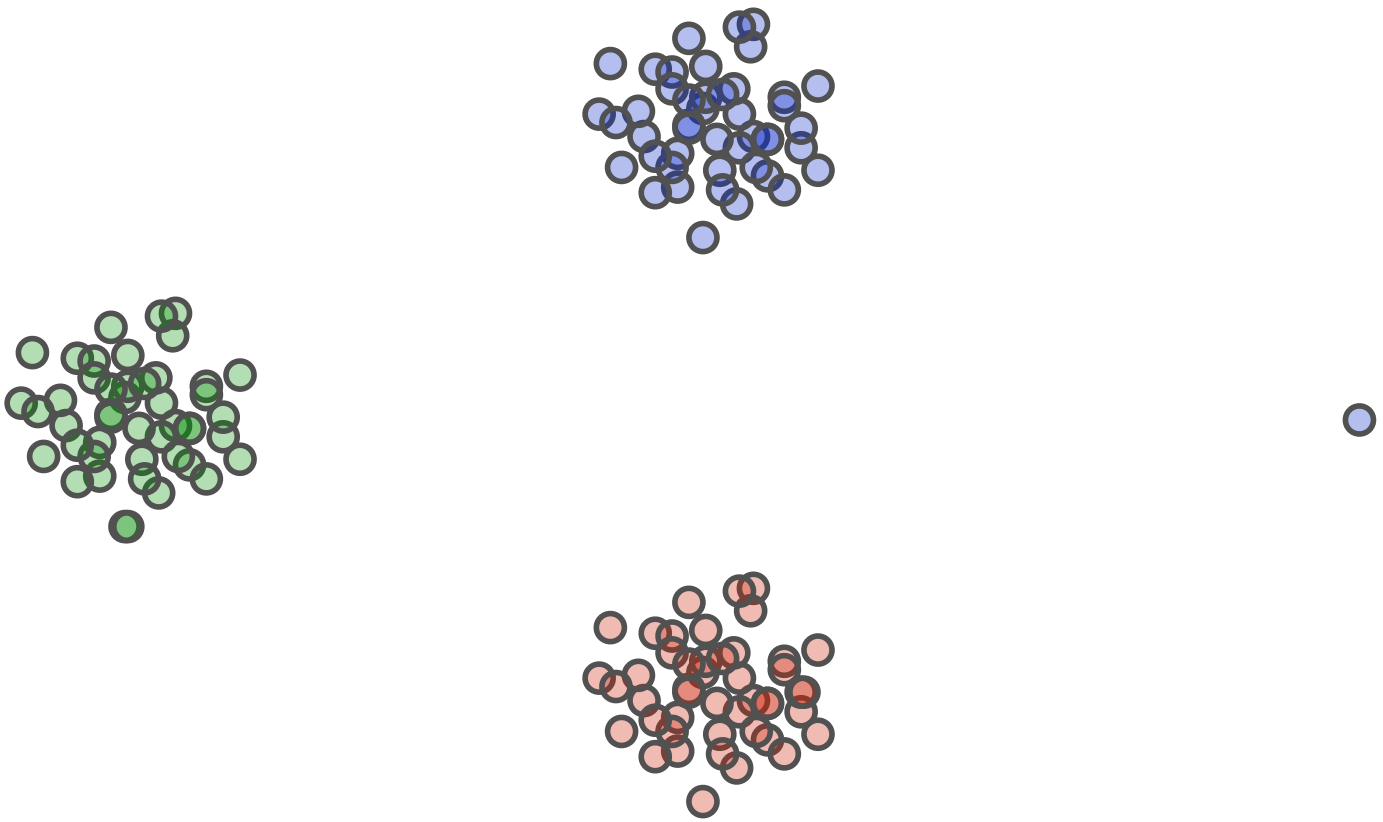
Example of k-means++

If the data set looks Gaussian...



Example of k-means++

If the data set looks Gaussian...



Example of k-means++

If the outlier should be its own cluster . . .

Example of k-means++

If the outlier should be its own cluster ...

Example of k-means++

If the outlier should be its own cluster . . .

Example of k-means++

If the outlier should be its own cluster ...

Example of k-means++

If the outlier should be its own cluster ...

Analyzing k-means++

What can we say about performance of k-means++?

Analyzing k-means++

What can we say about performance of k-means++?

Theorem (AV07)

This algorithm always attains an $O(\log k)$ approximation in expectation

Analyzing k-means++

What can we say about performance of k-means++?

Theorem (AV07)

This algorithm always attains an $O(\log k)$ approximation in expectation

Theorem (ORSS06)

A slightly modified version of this algorithm attains an $O(1)$ approximation if the data is 'nicely clusterable' with k clusters.

Nice Clusterings

What do we mean by 'nicely clusterable'?

Intuitively, X is nicely clusterable if going from $k-1$ to k clusters drops the total error by a constant factor.

Nice Clusterings

What do we mean by 'nicely clusterable'?

Intuitively, X is nicely clusterable if going from $k-1$ to k clusters drops the total error by a constant factor.

Definition

A pointset X is (k, ϵ)-separated if $\phi_k^*(X) \leq \epsilon^2 \phi_{k-1}^*(X)$.

Why does this work?

Intuition

Look at the optimum clustering. In expectation:

1. If the algorithm selects a point from a new OPT cluster, that cluster is covered pretty well
2. If the algorithm picks two points from the same OPT cluster, then other clusters must contribute little to the overall error

Why does this work?

Intuition

Look at the optimum clustering. In expectation:

1. If the algorithm selects a point from a new OPT cluster, that cluster is covered pretty well
2. If the algorithm picks two points from the same OPT cluster, then other clusters must contribute little to the overall error

As long as the points are reasonably well separated, the first condition holds.

Why does this work?

Intuition

Look at the optimum clustering. In expectation:

1. If the algorithm selects a point from a new OPT cluster, that cluster is covered pretty well
2. If the algorithm picks two points from the same OPT cluster, then other clusters must contribute little to the overall error

As long as the points are reasonably well separated, the first condition holds.

Two theorems

- Assume the points are (k, ϵ)-separated and get an $O(1)$ approximation.
- Make no assumptions about separability and get an $O(\log k)$ approximation.

Summary

`k-means++` Summary:

- To select the next cluster, sample a point in proportion to its current contribution to the error
- Works for k-means, k-median, other objective functions
- Universal $O(\log k)$ approximation, $O(1)$ approximation under some assumptions
- Can be implemented to run in $O(nkd)$ time (same as a single `k-means` step)

Summary

`k-means++` Summary:

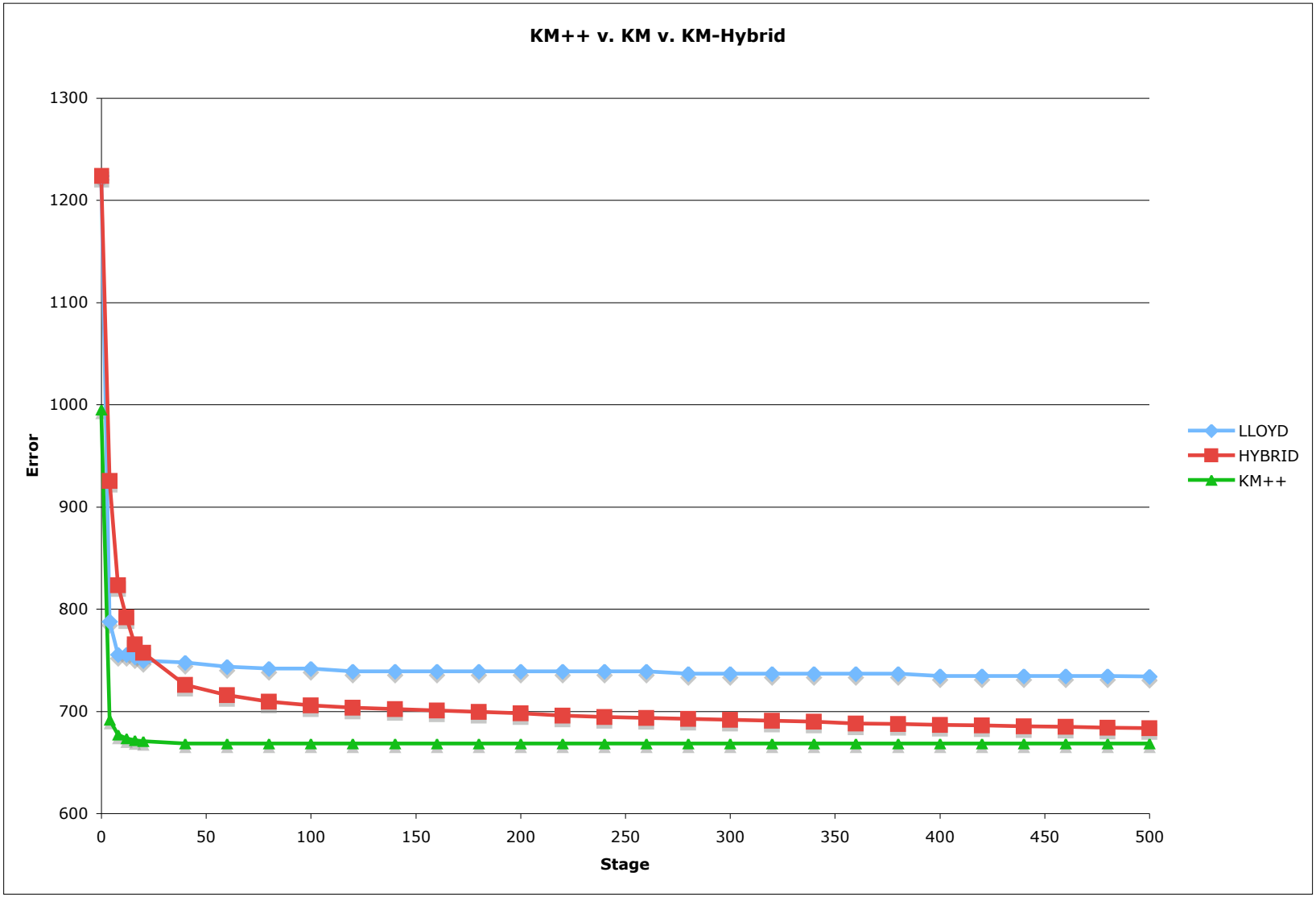
- To select the next cluster, sample a point in proportion to its current contribution to the error
- Works for k-means, k-median, other objective functions
- Universal $O(\log k)$ approximation, $O(1)$ approximation under some assumptions
- Can be implemented to run in $O(nkd)$ time (same as a single `k-means` step)

But does it actually work?

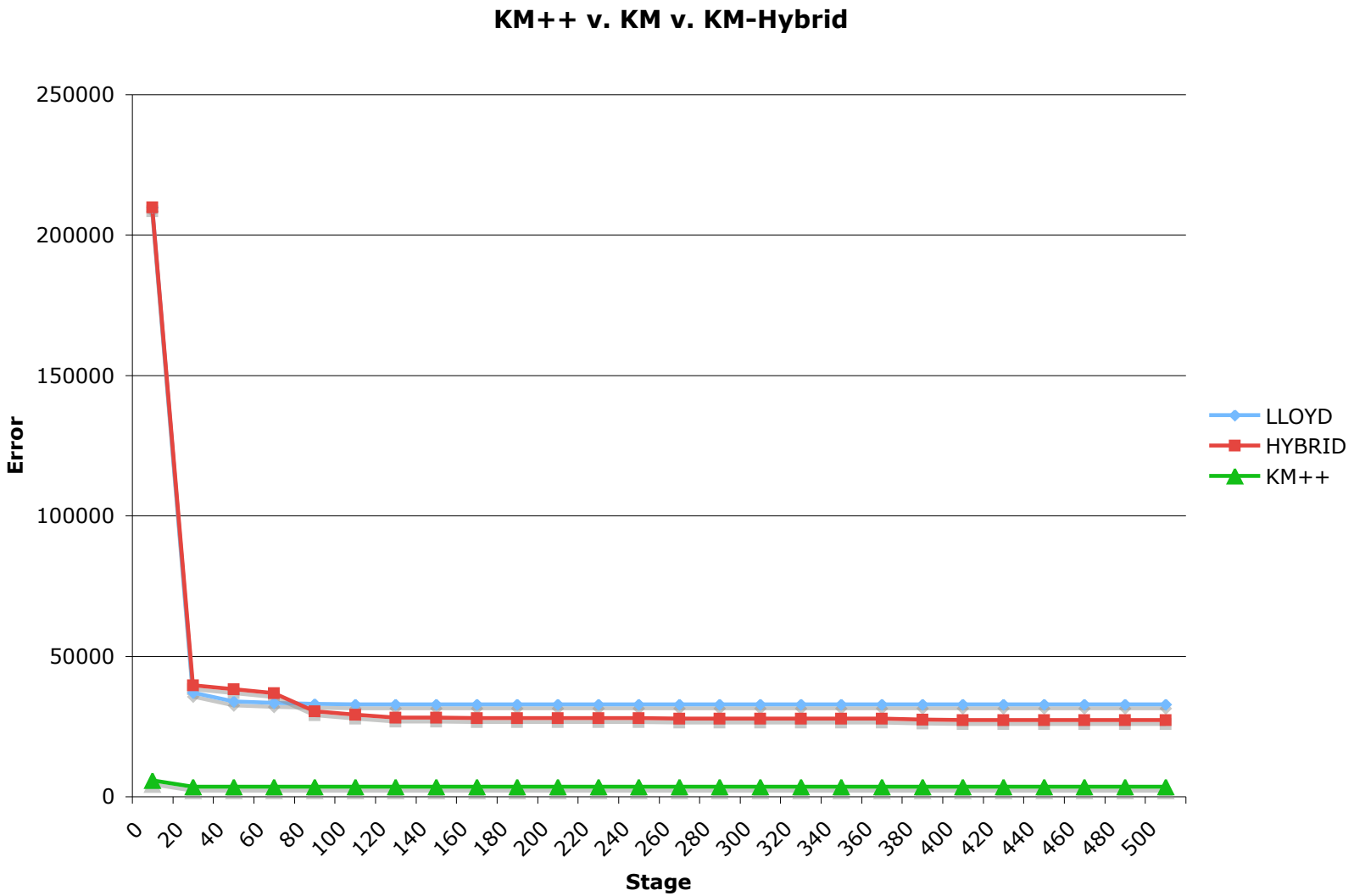
Large Evaluation

Data set	ϕ: k-means++ vs k-means ($k = 10, 25, 50, 100$)			
1	1.011	1.036	1.115	1.544
2	1.052	1.344	1.523	1.421
3	1.122	2.083	2.987	3.481
4	1.007	1.206	1.207	1.421
5	1.038	1.303	1.367	1.861
6	1.010	1.013	1.018	1.043
7	1.027	1.160	1.560	2.056
8	0.997	1.009	1.044	1.186
9	1.090	1.181	1.314	1.297
10	13.87	132.7	684.9	3728
11	1.002	1.003	1.011	1.046
12	2.420	5.027	11.08	22.76
13	2.202	9.532	11.96	30.17
14	1.020	1.039	7.942	1.008
15	1.013	554.2	1.003	1.004
Range	[0.997, 3728]			
Median	1.207			
Mean	87.93			

Typical Run


KM++ v. KM v. KM-Hybrid
Error
Stage
1300
1200
1100
1000
900
800
700
600
0
50
100
150
200
250
300
350
400
450
500
LLOYD
HYBRID
KM++

Other Runs


KM++ v. KM v. KM-Hybrid
250000
200000
150000
100000
50000
0
Error
0
20
40
60
80
100
120
140
160
180
200
220
240
260
280
300
320
340
360
380
400
420
440
460
480
500
Stage
LLOYD
HYBRID
KM++

Convergence

How fast does `k-means` converge?

It appears the algorithm converges in under 100 iterations (even faster with smart initialization).

Convergence

How fast does `k-means` converge?

It appears the algorithm converges in under 100 iterations (even faster with smart initialization).

Theorem (V09)

There exists a pointset X in $\mathbb{R}^2$ and a set of initial centers $\mathscr{C}$ so that `k-means` *takes $2^{\Omega(k)}$ iterations to converge when initialized with $\mathscr{C}$.*

Theory vs. Practice

Finding the disconnect

In theory:

- `k-means` might run in exponential time

In practice:

- `k-means` converges after a handful of iterations

It works in practice but it does not work in theory!

Finding the disconnect

Robustness of worst case examples

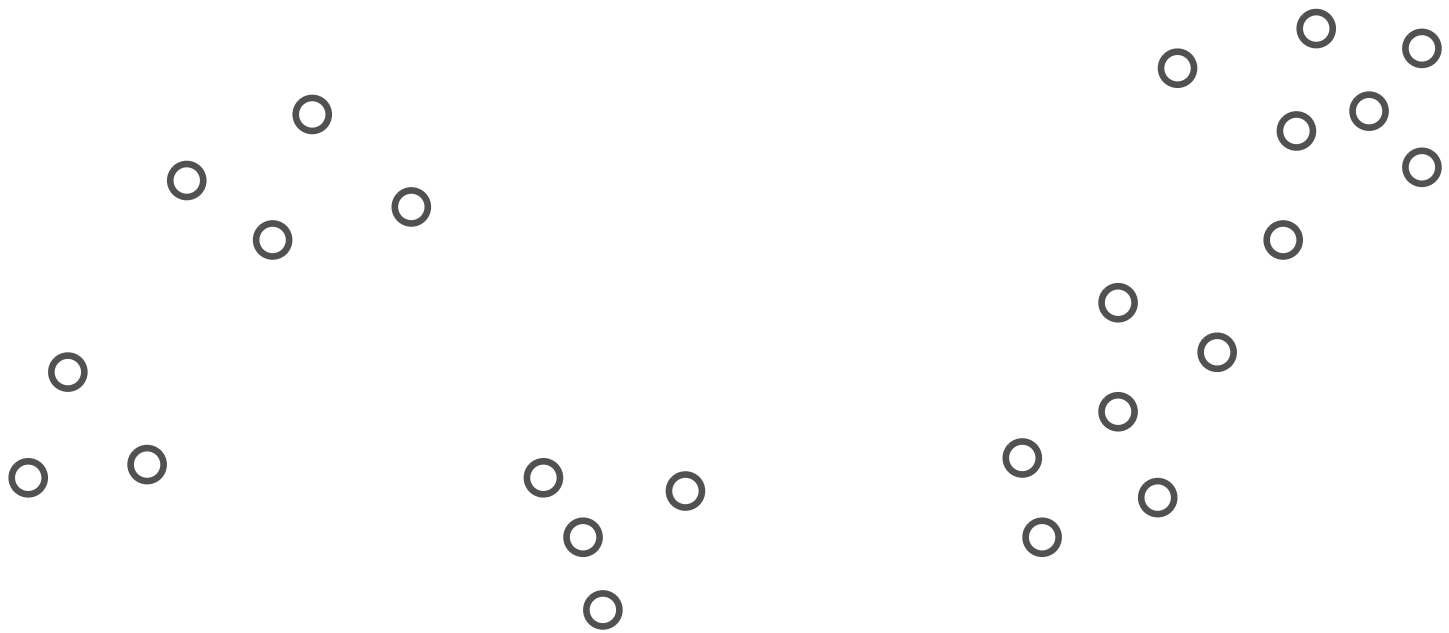
Perhaps the worst case examples are too precise, and can never arise out of natural data

Quantifying the robustness

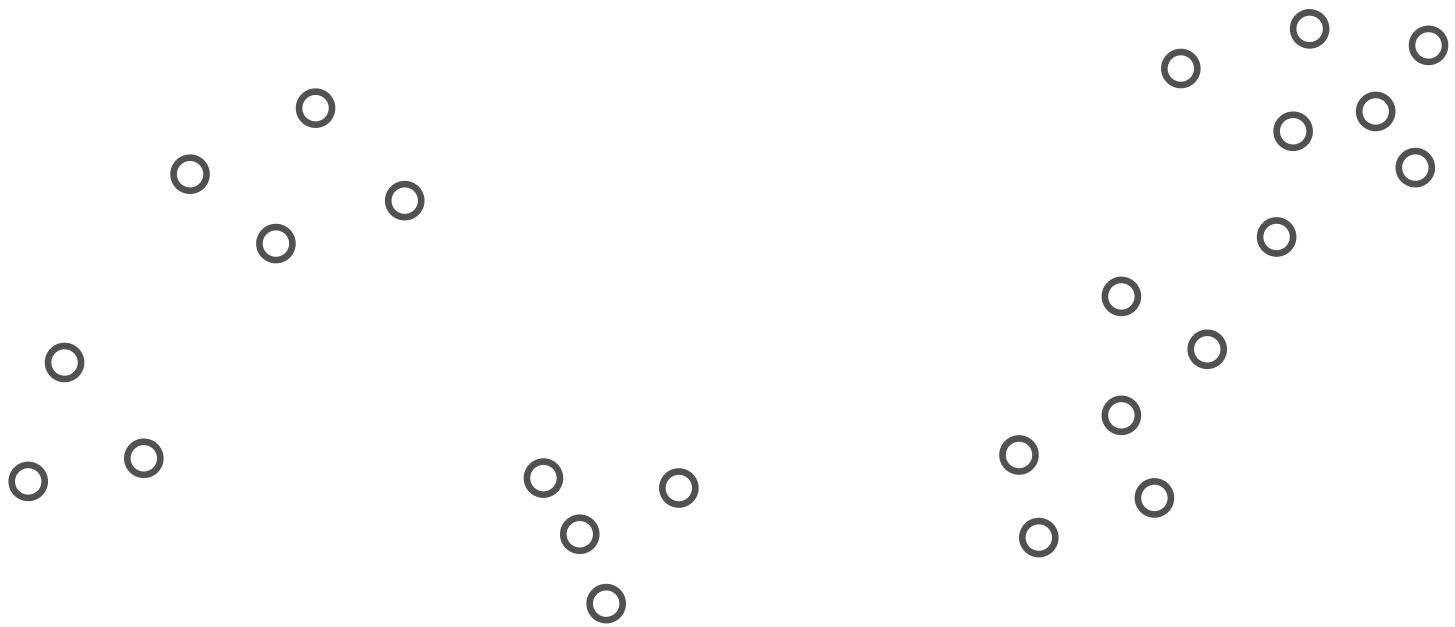
If we slightly perturb the points of the example:

- The optimum solution shouldn't change too much
- Will the running time stay exponential?

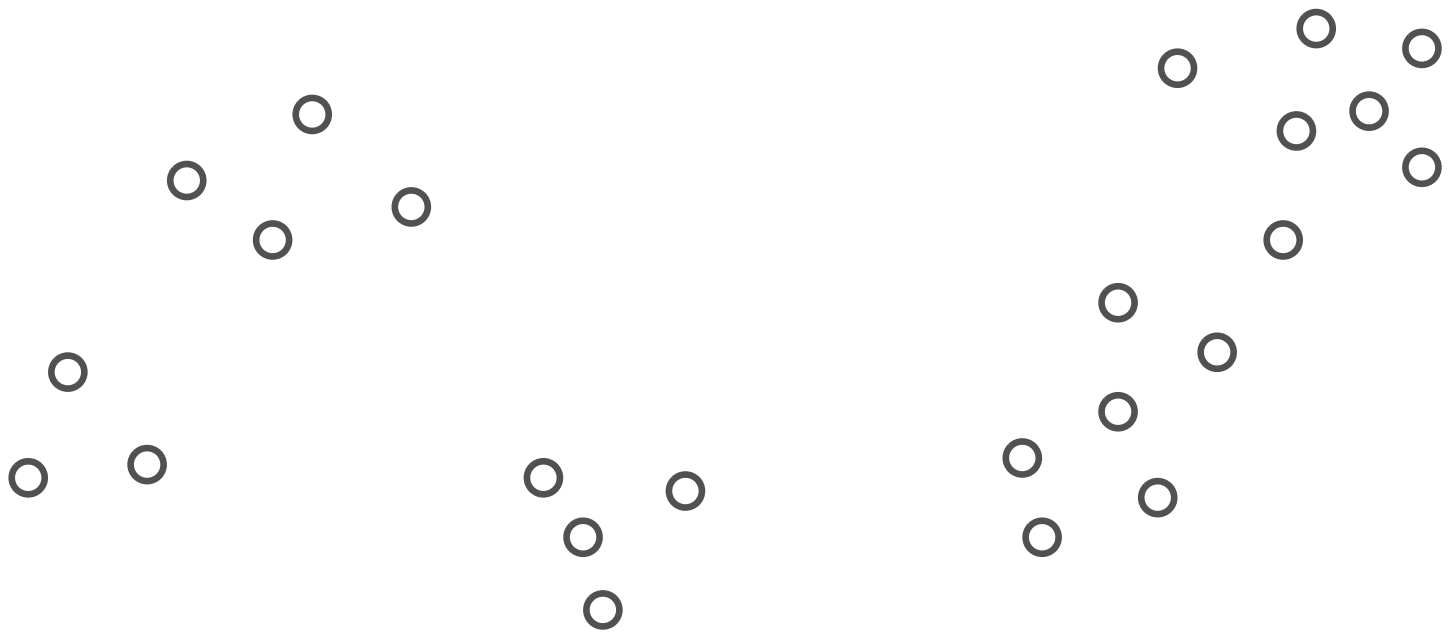
Small Perturbations



Small Perturbations



Small Perturbations



Smoothed Analysis

Perturbation

To each point $x \in X$ add independent noise drawn from $N(0, \sigma^2)$.

Definition

The smoothed complexity of an algorithm is the maximum expected running time after adding the noise:

$$\max_X \mathbb{E}_\sigma [Time(X + \sigma)]$$

Smoothed Analysis

Theorem (AMR09)

The smoothed complexity of `k-means` *is bounded by*

$$O\left(\frac{n^{34}k^{34}d^8D^6\log^4 n}{\sigma^6}\right)$$

Notes

- While the bound is large, it is not exponential ($2^k \gg k^{34}$ for large enough k)
- The $(D/\sigma)^6$ factor shows the bound is scale invariant

Smoothed Analysis

Comparing bounds

The smoothed complexity of `k-means` is polynomial in n, k and D/σ where D is the diameter of X, whereas the worst case complexity of `k-means` is exponential in k

Implications

The pathological examples:

- Are very brittle
- Can be avoided with a little bit of random noise

k-means Summary

Running Time

- Exponential worst case running time
- Polynomial typical case running time

k-means Summary

Running Time

- Exponential worst case running time
- Polynomial typical case running time

Solution Quality

- Arbitrary local optimum, even with many random restarts
- Simple initialization leads to a good solution

Implementing `k-means++`

Initialization:

- Takes $O(nd)$ time and one pass over the data to select the next center
- Takes $O(nkd)$ time total

Overall running time:

- Each round of `k-means` takes $O(nkd)$ running time
- Typically finish after a constant number of rounds

Large Datasets

Implementing `k-means++`

Initialization:

- Takes $O(nd)$ time and one pass over the data to select the next center
- Takes $O(nkd)$ time total

Overall running time:

- Each round of `k-means` takes $O(nkd)$ running time
- Typically finish after a constant number of rounds

Large Data

What if $O(nkd)$ is too much, can we parallelize this algorithm?

Parallelizing k-means

Approach

Partition the data:

- Split X into $X_1, X_2, \ldots, X_m$ of roughly equal size.

Parallelizing k-means

Approach

Partition the data:

- Split X into $X_1, X_2, \ldots, X_m$ of roughly equal size.

In parallel compute a clustering on each partition:

- Find $\mathscr{C}^j = \{C_1^j, \ldots, C_k^j\}$: a good clustering on each partition, and denote by w_i^j the number of points in cluster C_i^j.

Parallelizing k-means

Approach

Partition the data:

- Split X into $X_1, X_2, \ldots, X_m$ of roughly equal size.

In parallel compute a clustering on each partition:

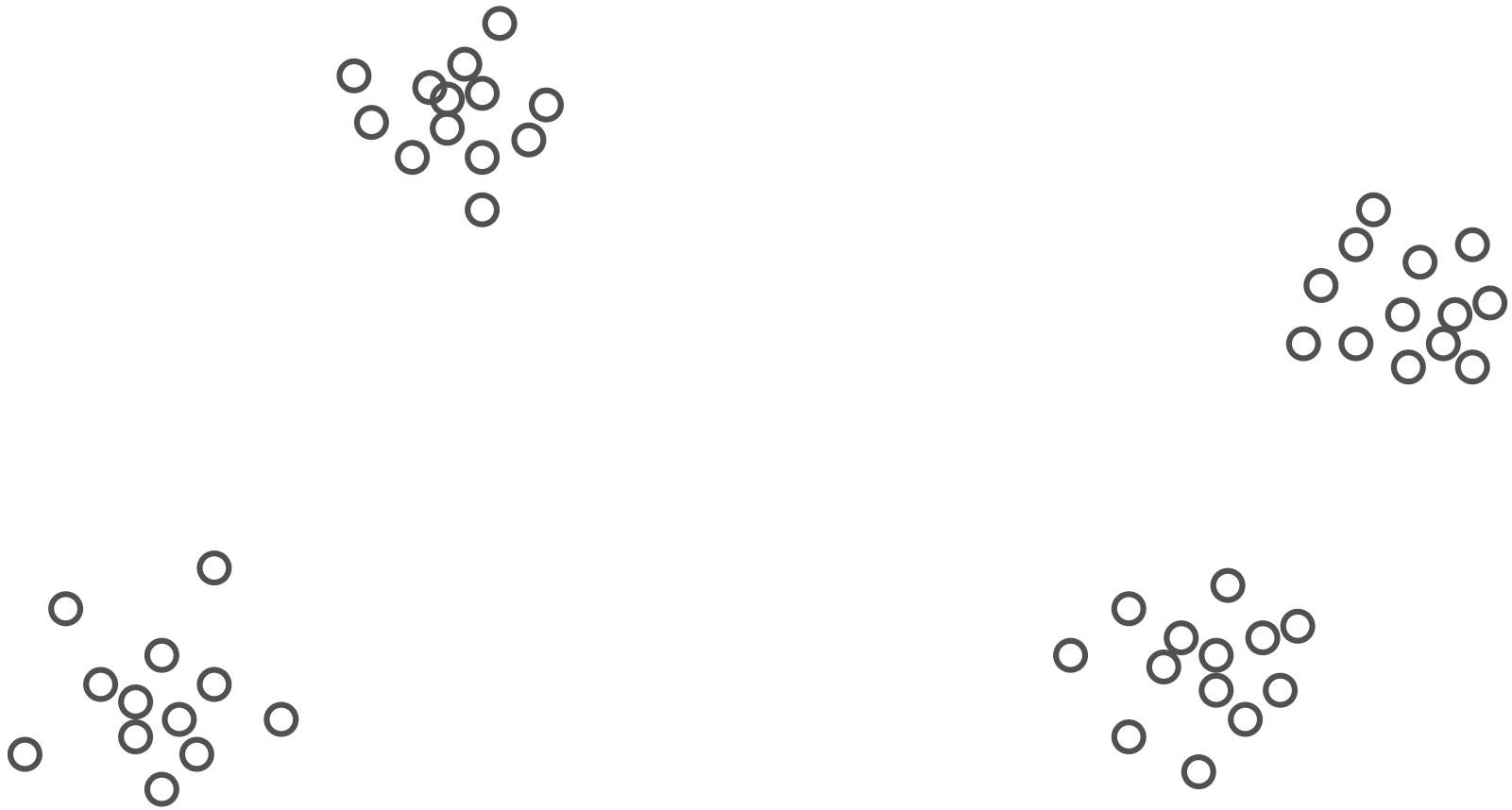
- Find $\mathscr{C}^j = \{C_1^j, \ldots, C_k^j\}$: a good clustering on each partition, and denote by w_i^j the number of points in cluster C_i^j.

Cluster the clusters:

- Let $Y = \cup_{1 \leq j \leq m} \mathscr{C}^j$. Find a clustering of Y, weighted by the weights $W = \{w_i^j\}$.

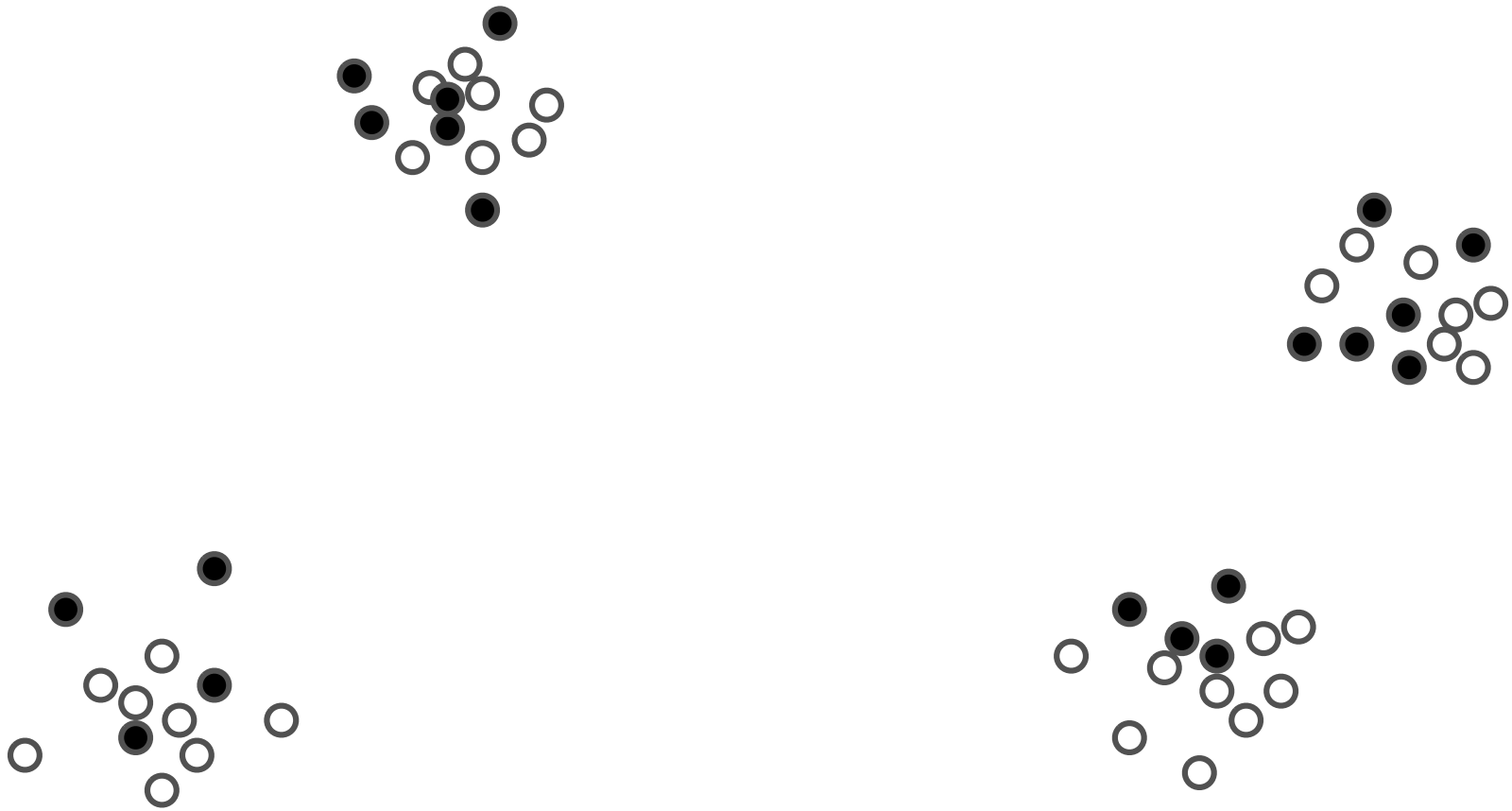
Parallelization Example

Given X



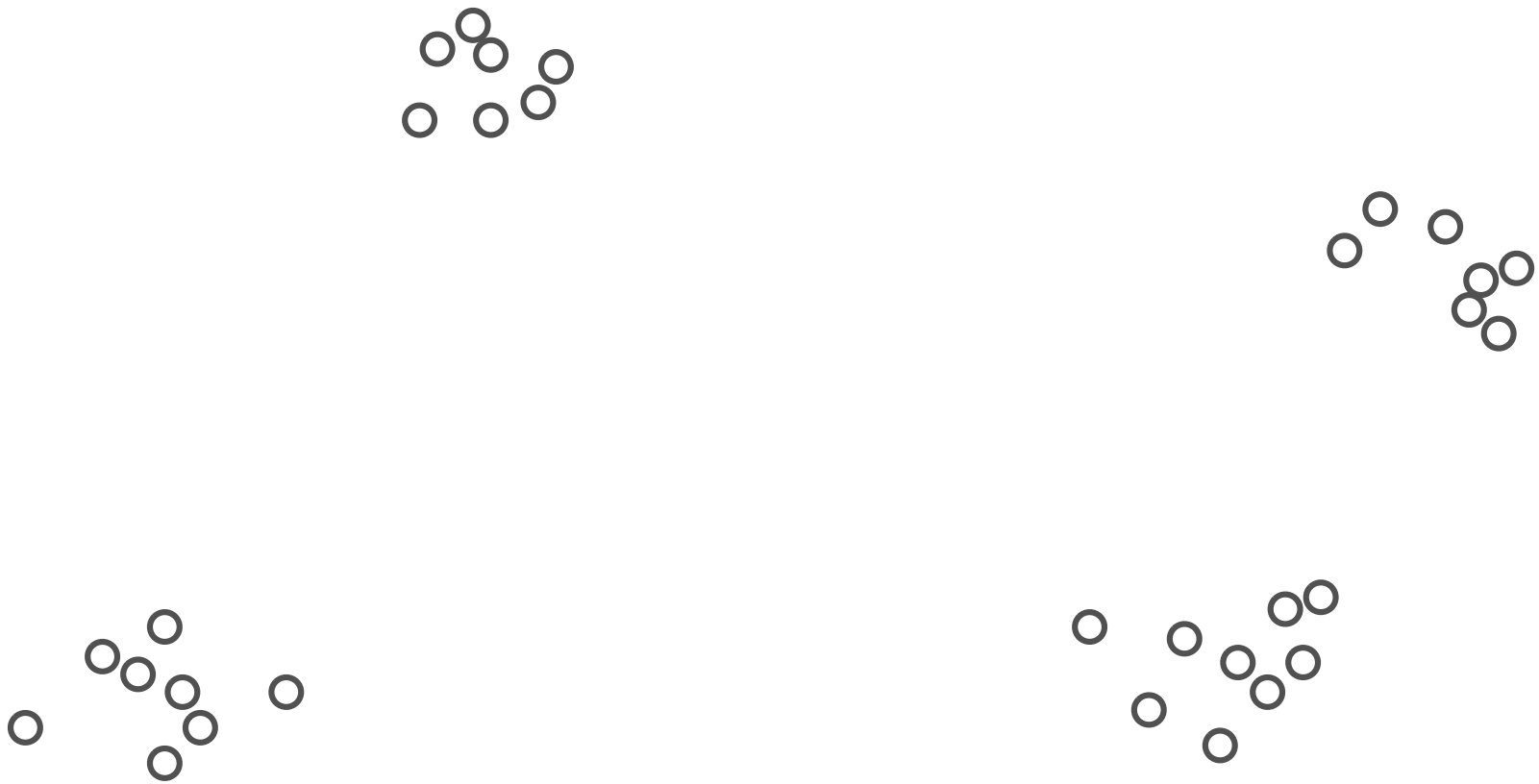
Parallelization Example

Partition the dataset



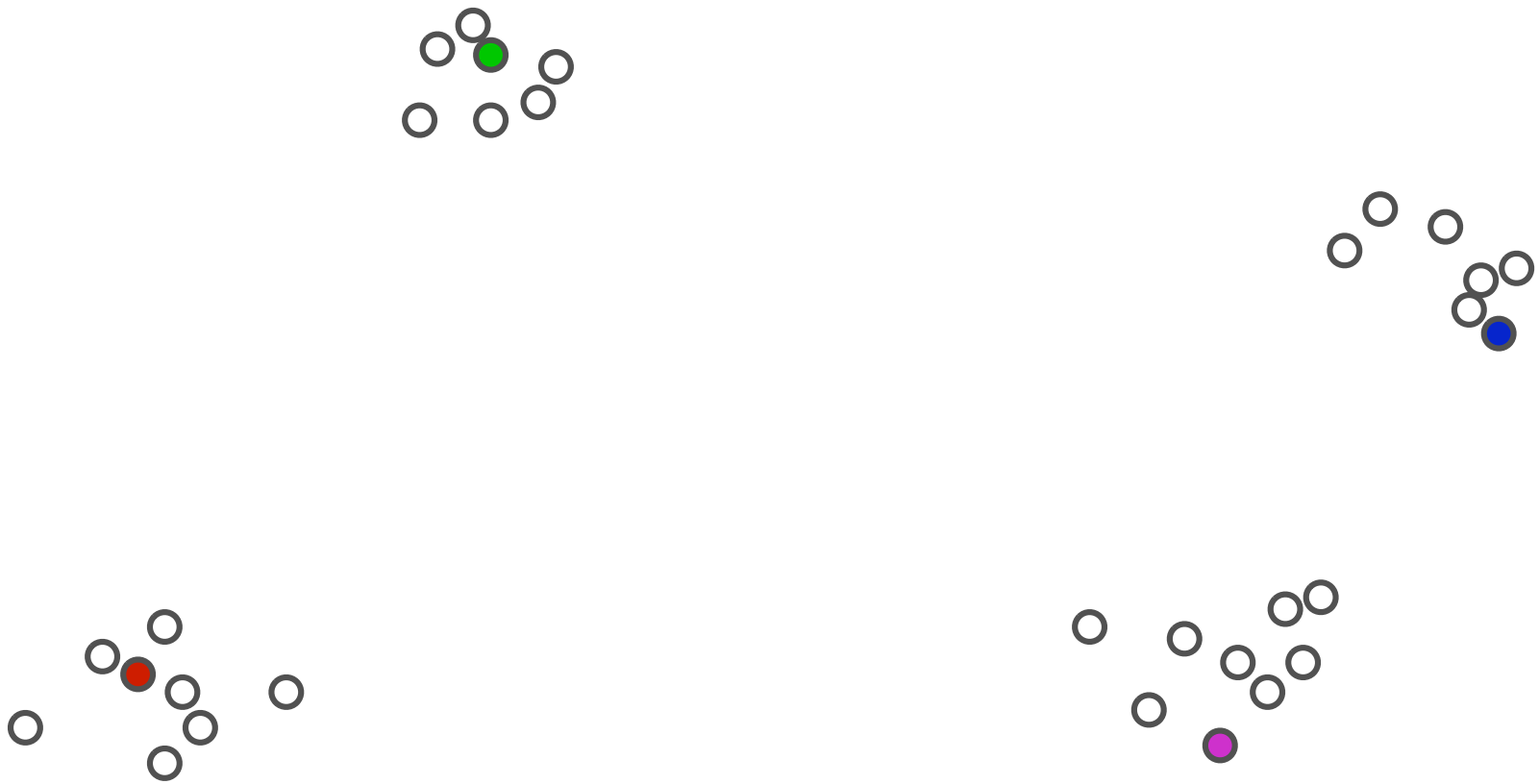
Parallelization Example

Cluster each partition separately



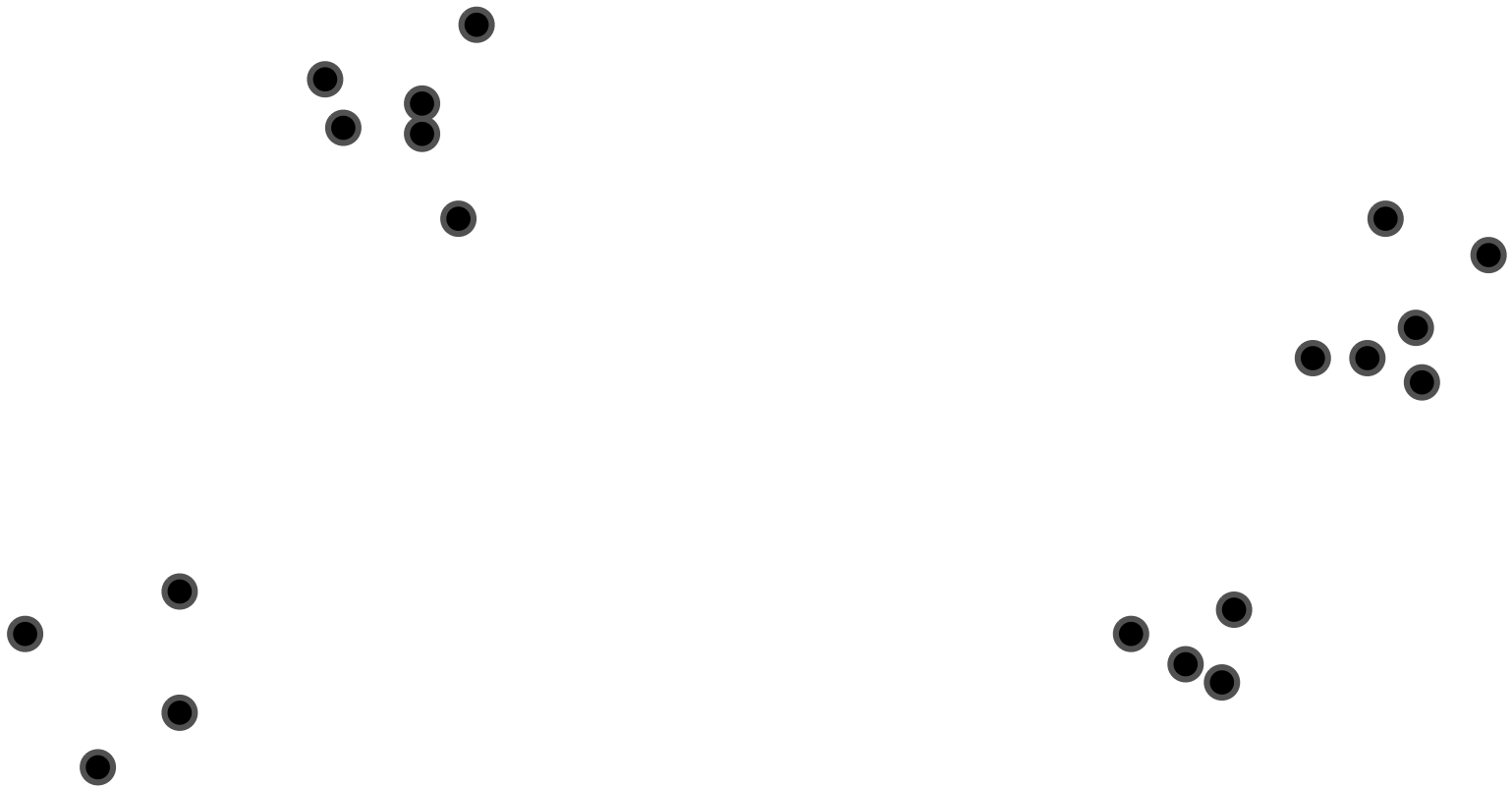
Parallelization Example

Cluster each partition separately



Parallelization Example

Cluster each partition separately



Parallelization Example

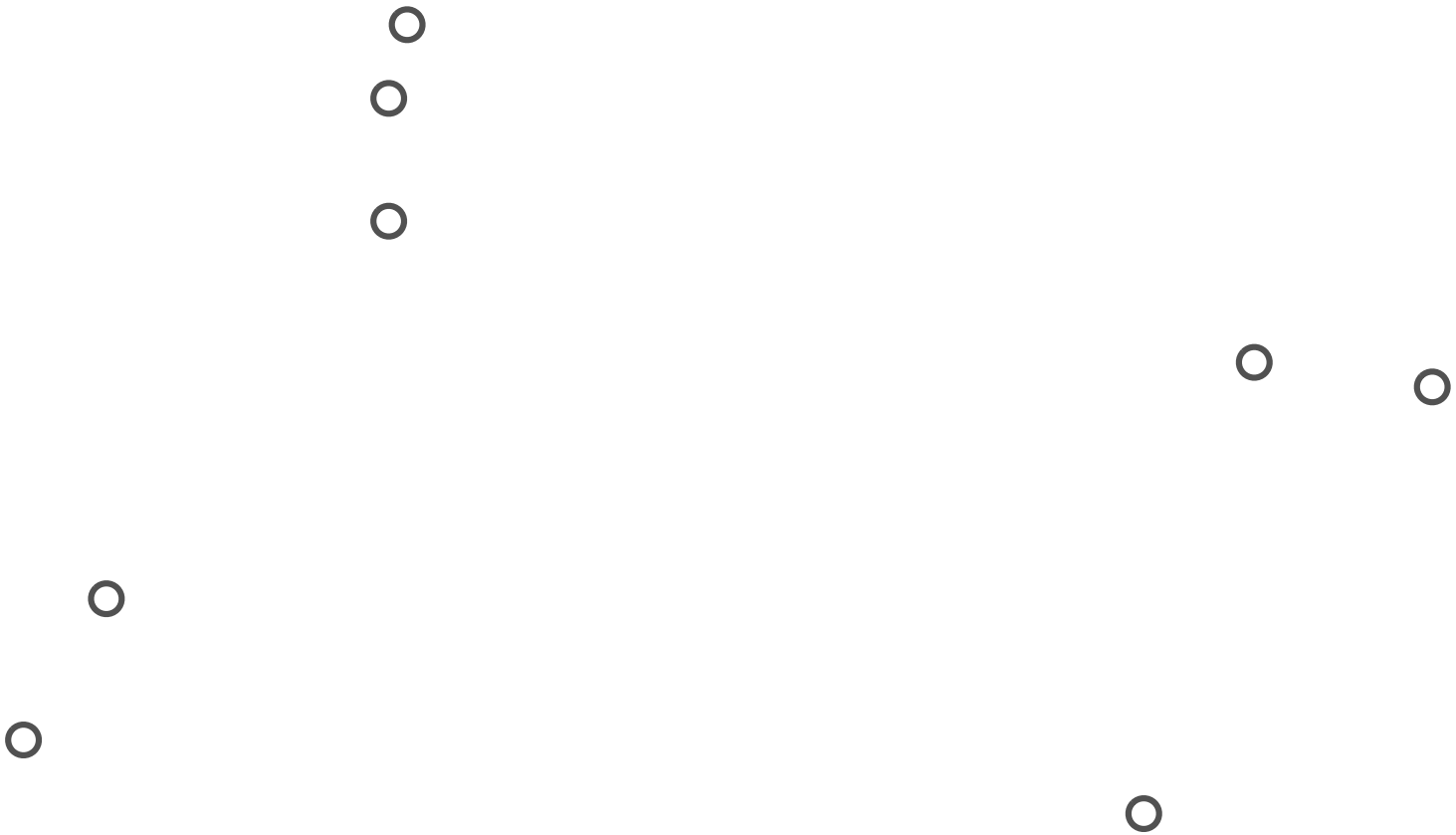
Cluster each partition separately

Parallelization Example

Cluster the clusters

Parallelization Example

Cluster the clusters



Parallelization Example

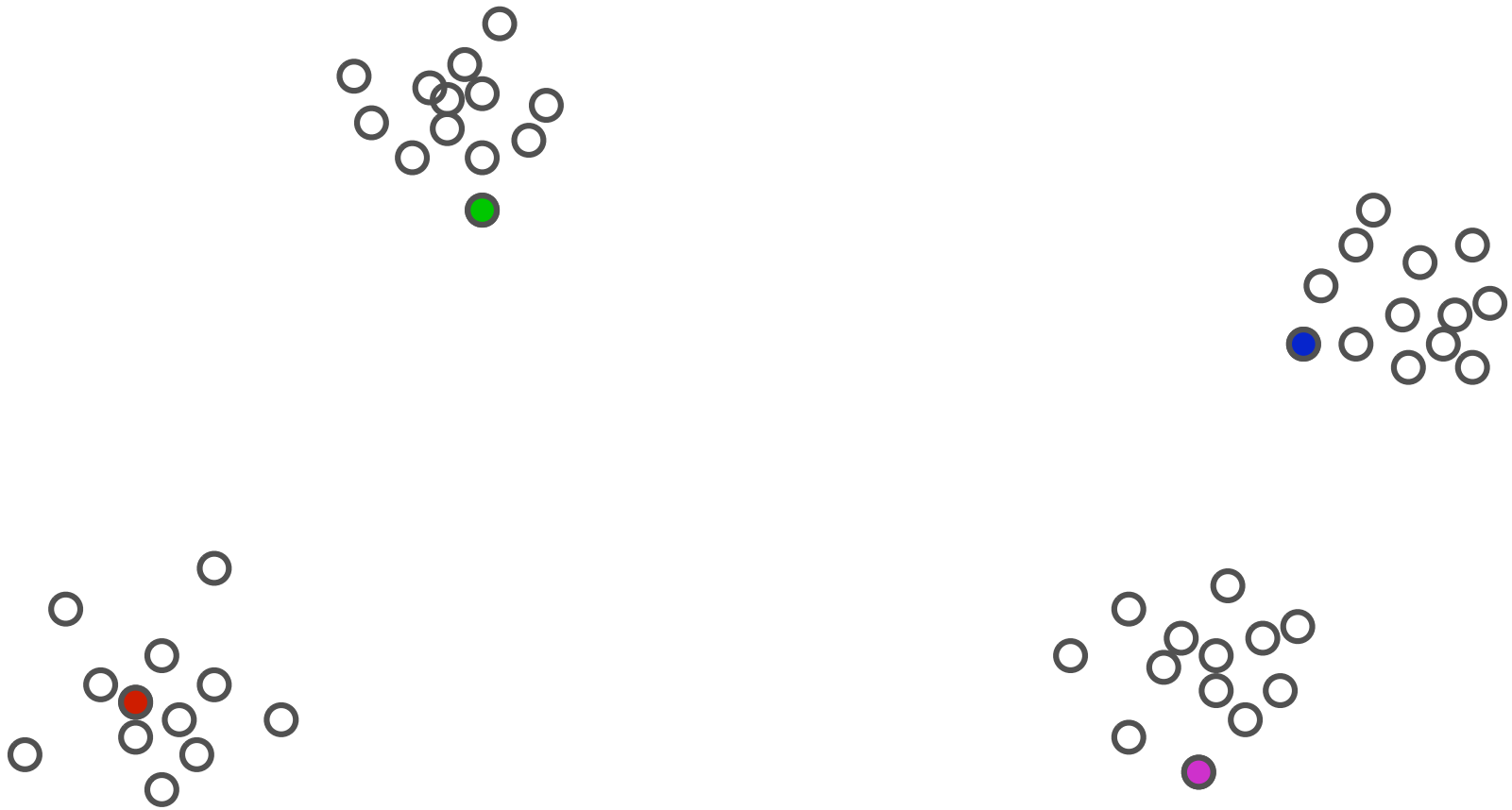
Cluster the clusters

Parallelization Example

Final clustering:

Parallelization Example

Final clustering:



Quality of the solution

What happens when we approximate the approximation?

- Suppose the algorithm in phase 1 gave a β-approximate solution to its input
- Algorithm in phase 2 gave a γ-approximate solution to its (smaller) input

Quality of the solution

What happens when we approximate the approximation?

- Suppose the algorithm in phase 1 gave a β-approximate solution to its input
- Algorithm in phase 2 gave a γ-approximate solution to its (smaller) input

Theorem (GNMO00, AJM09)

The two phase algorithm gives a $4\gamma(1+\beta)+2\beta$ approximate solution.

Analysis

Running time

Suppose we partition the input across m different machines.

- First phase running time: $O(\frac{nkd}{m})$.
- Second phase running time $O(mk^2d)$.

Improving the algorithm

Approximation Guarantees

Using `k-means++` sets $\beta = \gamma = O(\log k)$ and leads to a $O(\log^2 k)$ approximation.

Improving the algorithm

Approximation Guarantees

Using `k-means++` sets $\beta = \gamma = O(\log k)$ and leads to a $O(\log^2 k)$ approximation.

Improving the Approximation

Must improve the approximation guarantee of the first round, but can use a larger k to ensure every cluster is well summarized.

Improving the algorithm

Approximation Guarantees

Using `k-means++` sets $\beta = \gamma = O(\log k)$ and leads to a $O(\log^2 k)$ approximation.

Improving the Approximation

Must improve the approximation guarantee of the first round, but can use a larger k to ensure every cluster is well summarized.

Theorem (ADK09)

Running `k-means++` *initialization for* $O(k)$ *rounds leads to a* $O(1)$ *approximation to the optimal solution (but uses more centers than OPT).*

Two round k-means++

Final Algorithm

Partition the data:

- Split X into $X_1, X_2, \ldots, X_m$ of roughly equal size.

Two round k-means++

Final Algorithm

Partition the data:

- Split X into $X_1, X_2, \ldots, X_m$ of roughly equal size.

Compute a clustering using $\ell = O(k)$ centers each partition:

- Find $\mathscr{C}^j = \{C_1^j, \ldots, C_\ell^j\}$ using k-means++ on each partition, and denote by w_i^j the number of points in cluster C_i^j.

Two round k-means++

Final Algorithm

Partition the data:

- Split X into $X_1, X_2, \ldots, X_m$ of roughly equal size.

Compute a clustering using $\ell = O(k)$ centers each partition:

- Find $\mathscr{C}^j = \{C_1^j, \ldots, C_\ell^j\}$ using k-means++ on each partition, and denote by w_i^j the number of points in cluster C_i^j.

Cluster the clusters.

- Let $Y = \cup_{1 \le j \le m} \mathscr{C}^j$ be a set of $O(\ell m)$ points. Use k-means++ to cluster Y, weighted by the weights $W = \{w_i^j\}$.

Theorem

The algorithm achieves an $O(1)$ approximation in time $O(\frac{nkd}{m} + mk^2 d)$

Summary

Before...

`k-means` used to be a prime example of the disconnect between theory and practice – it works well, but has horrible worst case analysis

...and after

Smoothed analysis explains the running time and rigorously analyzed initializations routines help improve clustering quality.

Outline

I Euclidean Clustering and `k-means` algorithm

- What to do to select initial centers (and what not to do)
- How long does `k-means` take to run in theory, practice and theoretical practice
- How to run `k-means` on large datasets

Outline

I Euclidean Clustering and `k-means` algorithm
- What to do to select initial centers (and what not to do)
- How long does `k-means` take to run in theory, practice and theoretical practice
- How to run `k-means` on large datasets

II Bregman Clustering and `k-means`
- Bregman Clustering as generalization of `k-means`
- Performance Results

Outline

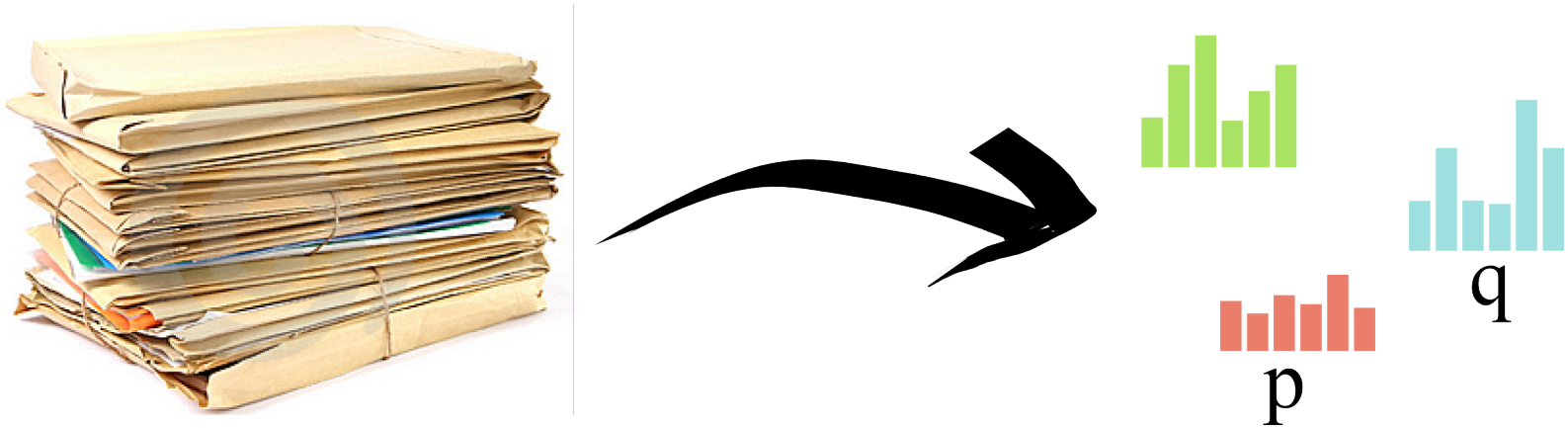
I Euclidean Clustering and `k-means` algorithm
- What to do to select initial centers (and what not to do)
- How long does `k-means` take to run in theory, practice and theoretical practice
- How to run `k-means` on large datasets

II Bregman Clustering and `k-means`
- Bregman Clustering as generalization of `k-means`
- Performance Results

III Stability
- How to relate closeness in cost function to closeness in clusters.

Clustering With Non-Euclidean Metrics

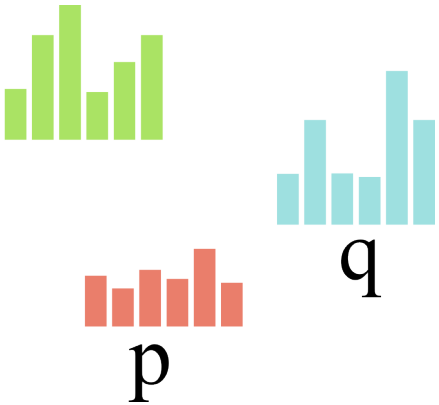
Application I: Clustering Documents



Kullback-Leibler distance:

$$D(p,q) = \sum_i p_i \log \frac{p_i}{q_i}$$

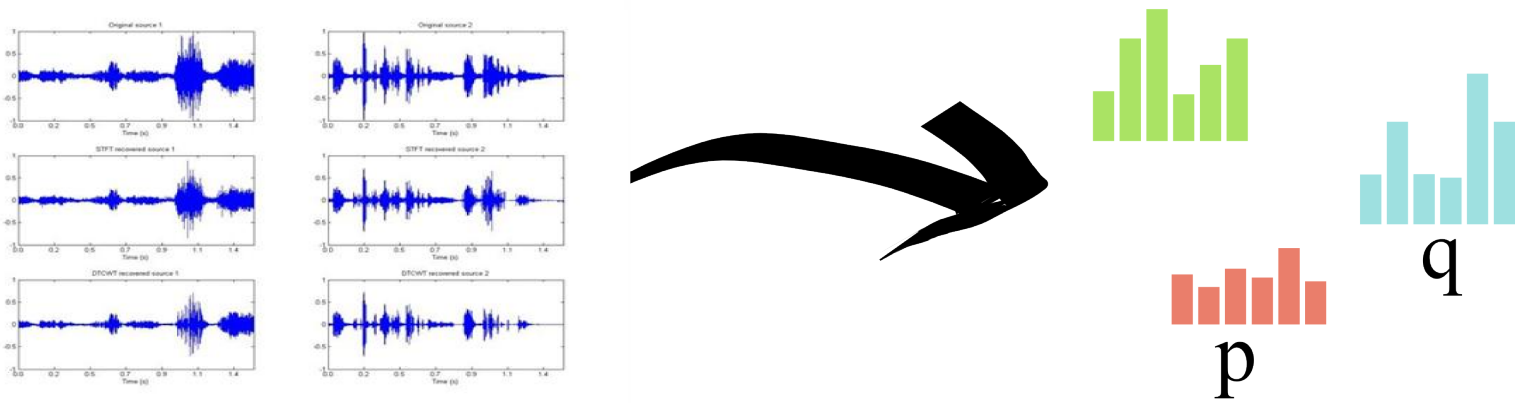
Application II: Image Analysis



Kullback-Leibler distance:

$$D(p,q) = \sum_i p_i \log \frac{p_i}{q_i}$$

Application III: Speech Analysis



Itakuro-Saito distance:

$$D(p,q) = \sum_i \frac{p_i}{q_i} - \log \frac{p_i}{q_i} - 1$$

Bregman Divergences

Definition

Let $\phi : \mathbb{R}^d \to \mathbb{R}$ be a strictly convex function. The Bregman divergence d_ϕ is defined as

$$D_\phi(x \,\|\, y) = \phi(x) - \phi(y) - \langle \nabla\phi(y), x - y \rangle$$

Examples:

Kullback-Leibler: $\phi(x) = \sum x_i \ln x_i - x_i$, $D_\phi(x \,\|\, y) = \sum x_i \ln \frac{x_i}{y_i}$

Itakura-Saito: $\phi(x) = -\sum \ln x_i$, $D_\phi(x \,\|\, y) = \sum_i \frac{x_i}{y_i} - \log \frac{x_i}{y_i} - 1$

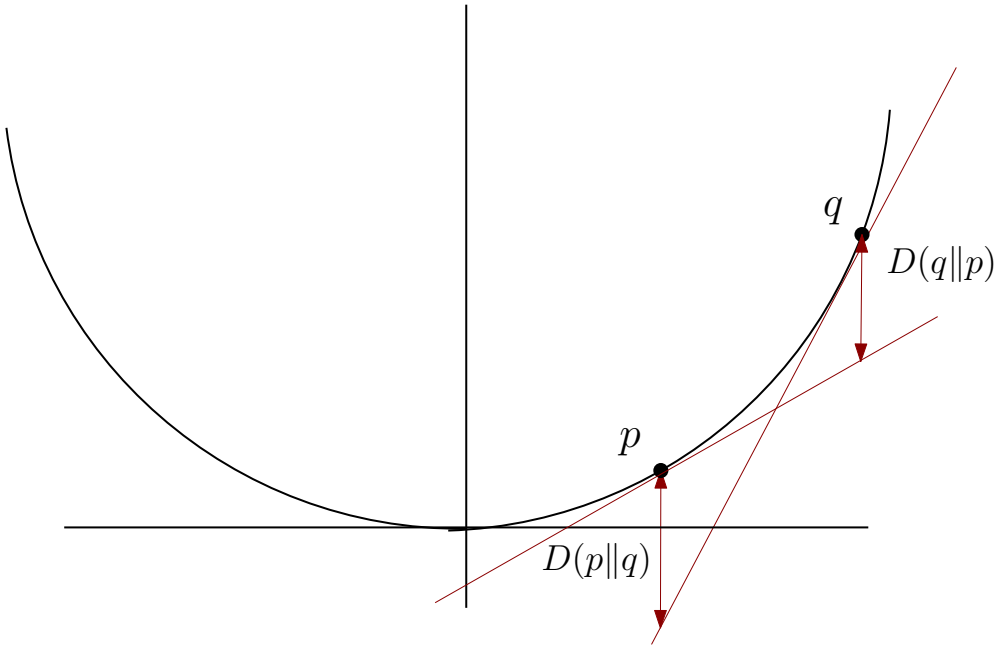
ℓ_2^2: $\phi(x) = \frac{1}{2}\|x\|^2, D_\phi(x \,\|\, y) = \|x - y\|^2$

Overview

k-means clustering $\equiv$ Bregman clustering

- The algorithm works the same way.
- Same (bad) worst-case behavior
- Same (good) *smoothed* behavior
- Same (good) quality guarantees, with correct initialization

Properties



$$D_\phi(x \parallel y) = \phi(x) - \phi(y) - \langle \nabla\phi(y), x - y \rangle$$

- Asymmetry: In general, $D_\phi(p \parallel q) \neq D_\phi(q \parallel p)$
- No triangle inequality: $D_\phi(p \parallel q) + D_\phi(q \parallel r)$ can be less than $D_\phi(p \parallel r)$!

How can we now do clustering ?

Breaking down k-means

Initialize cluster centers
while not converged **do**
    Assign points to nearest cluster center
    Find new cluster center by averaging points assigned together
end while

Key Point

Setting cluster center as centroid minimizes the average **squared distance** to center

Breaking down k-means

Initialize cluster centers
while not converged **do**
  Assign points to nearest cluster center

  Find new cluster center by averaging points assigned together

end while

Key Point

Setting cluster center as centroid minimizes the average **squared distance** to center

Bregman Centroids

Problem

Given points $x_1, \ldots x_n \in \mathbb{R}^d$, find c such that

$$\sum_i D_\phi(x_i \parallel c)$$

is minimized.

Answer

$$c = \frac{1}{n} \sum x_i$$

Independent of ϕ [BMDG05] !

Bregman k-means

Initialize cluster centers
while not converged **do**
  Assign points to nearest cluster center (by measuring $D_\phi(x \| c)$)
  Find new cluster center by averaging points assigned together
end while

Key Point

Setting cluster center as centroid minimizes average **Bregman divergence** to center

Convergence

Lemma ([BMDG05])

The (Bregman) k-means algorithm converges in cost.

Euclidean distance:
The quantity

$$\sum_{C}\sum_{x\in C}\|x-\text{center}(C)\|^2$$

decreases with each iteration of k-means

Bregman divergence: Bregman Information:

$$\sum_{C}\sum_{x\in C}D_\phi(x\parallel \text{center}(C))$$

decreases with each iteration of the Bregman k-means algorithm.

EM and Soft Clustering

Expectation maximization:

Initialize density parameters and means for k distributions
while not converged **do**
For distribution i and point x, compute *conditional probability* $p(i|x)$ that x was drawn from i (by Bayes rule)
For each distribution i, recompute new density parameters and means (via maximum likelihood)
end while

- This yields a *soft clustering* of points to "clusters"
- Originally used for mixtures of Gaussians.

Exponential Families And Bregman Divergences

Definition (Exponential Family)

Parametric family of distributions $p_{\Psi,\theta}$ is an exponential family if each density is of the form

$$p_{\Psi,\theta} = \exp(\langle x, \theta \rangle - \Psi(\theta)) p_0(x)$$

with Ψ convex.

Let $\phi(t) = \Psi^*(t)$ be the *Legendre-Fenchel dual* of $\Psi(x)$:

$$\phi(t) = \sup_x \big(\langle x, t \rangle - \Psi(x) \big)$$

Theorem ([BMDG05])

$$p_{\Psi,\theta} = \exp(-D_\phi(x \parallel \mu)) b_\phi(x)$$

where μ is the expectation parameter $\nabla\Psi(\theta)$

EM: Euclidean and Bregman

Expectation maximization:

Initialize density parameters and means for k distributions
while not converged **do**
For distribution i and point x, compute *conditional probability* $p(i|x)$ that x was drawn from i (by Bayes rule)
For each distribution i, recompute new density parameters and means (via maximum likelihood)
end while

Choosing the corresponding Bregman divergence $D_\phi(\cdot \,\|\, \cdot), \phi = \Psi^*$ gives mixture density estimation for any exponential family $p_{\Psi,\theta}$.

Performance Analysis

Performance Analysis

Two questions:

Problem (Rate of convergence)

Given an arbitrary set of n points in d dimensions, how long does it take for (Bregman) k-means to converge ?

Performance Analysis

Two questions:

Problem (Rate of convergence)

Given an arbitrary set of n points in d dimensions, how long does it take for (Bregman) k-means to converge ?

Problem (Quality of Solution)

Let OPT denote the optimal clustering that minimizes the average sum of (Bregman) distances to cluster centers. How close to OPT is the solution returned by (Bregman) k-means ?

Performance Analysis

Two questions:

Problem (Rate of convergence)

Given an arbitrary set of n points in d dimensions, how long does it take for (Bregman) k-means to converge ?

Convergence of k-means

Parameters: n, k, d.

☺ Good news

k-means always converges in $O(n^{kd})$ time.

☹ Bad news

k-means can take time $2^{\Omega(k)}$ to converge:

Convergence of k-means

Parameters: n, k, d.

☺ Good news

k-means always converges in $O(n^{kd})$ time.

☹ Bad news

k-means can take time $2^{\Omega(k)}$ to converge:

- Even if $d = 2$, i.e in the plane

Convergence of k-means

Parameters: n, k, d.

☺ Good news

k-means always converges in $O(n^{kd})$ time.

☹ Bad news

k-means can take time $2^{\Omega(k)}$ to converge:

- Even if $d = 2$, i.e in the plane
- Even if centers are chosen from the initial data

Convergence of Bregman k-means

Euclidean distance:
k-means can take time $2^{\Omega(k)}$ to converge:

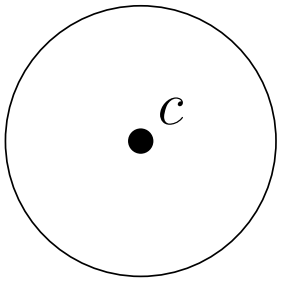
- Even if $d = 2$, i.e in the plane
- Even if centers are chosen from the initial data

Bregman divergence:
For some Bregman divergences, k-means can take time $2^{\Omega(k)}$ to converge[MR09]:

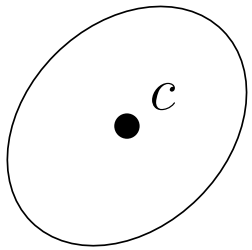
- Even if $d = 2$, i.e in the plane
- Even if centers are chosen from the initial data

Proof Idea

"Well behaved" Bregman divergences look "locally Euclidean":

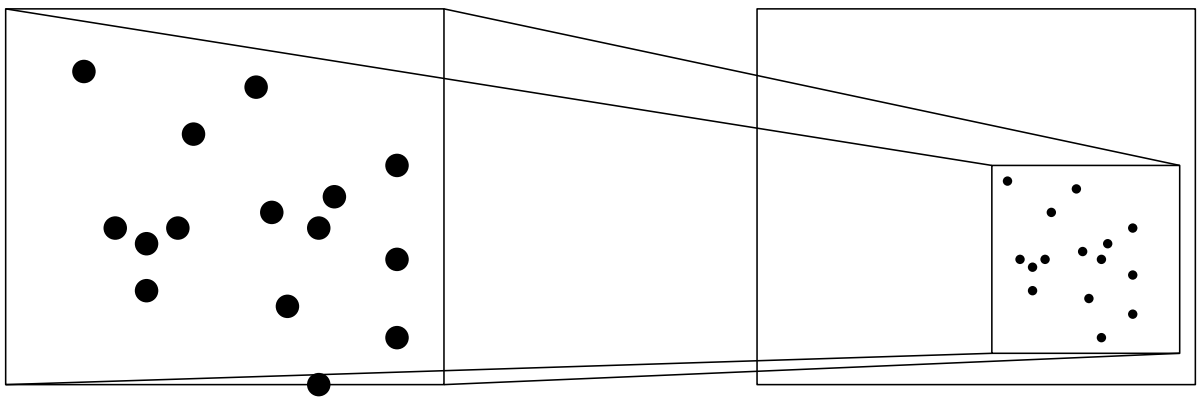


$\{x | \|x - c\|^2 \leq 1\}$



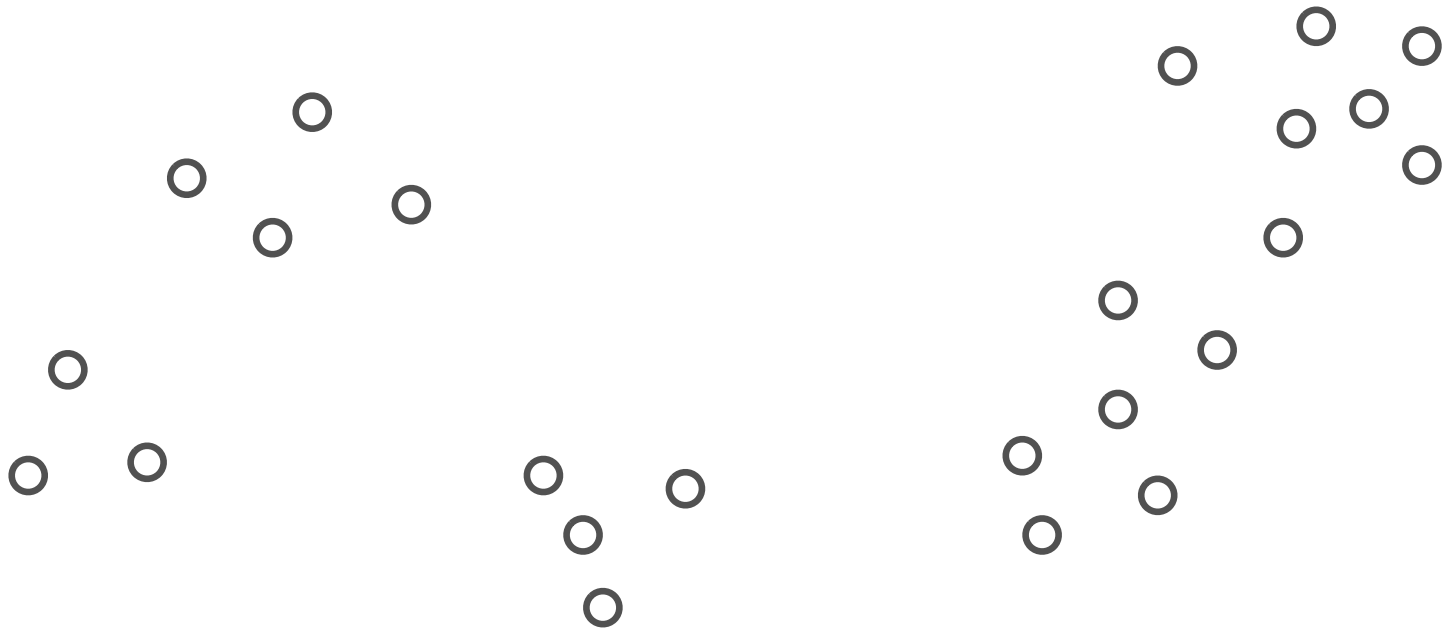
$\{x \mid D_\phi(x, c) \leq 1\}$

Take a bad Euclidean instance and shrink it to make it local.



Smoothed Analysis

Real inputs aren't worst-case!



Analyze **expected** run-time over perturbations.

Smoothed Analysis

Real inputs aren't worst-case!

Analyze **expected** run-time over perturbations.

k-means: Worst-case vs Smoothed

Theorem

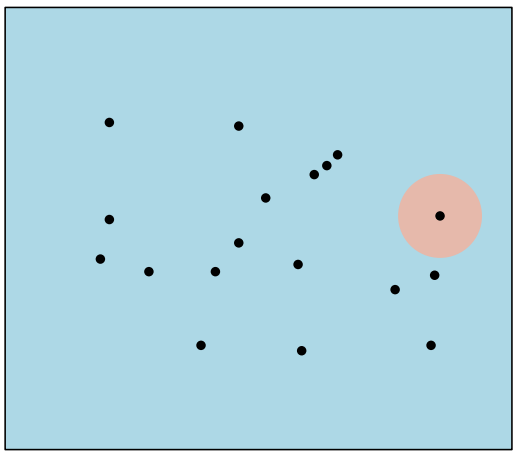
*Smoothed complexity of k-means using Gaussian noise with variance σ is **polynomial** in n and $1/\sigma$.*

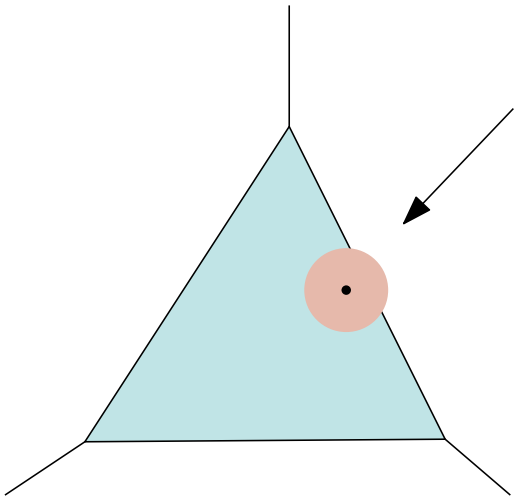
Compare this to worst-case lower bound of $2^{\Theta(n)}$

Bregman Smoothing

Normal smoothing doesn't work !

$$\Delta_n = \{(x_1, \ldots x_n) \mid \Sigma\, x_i = 1\}$$



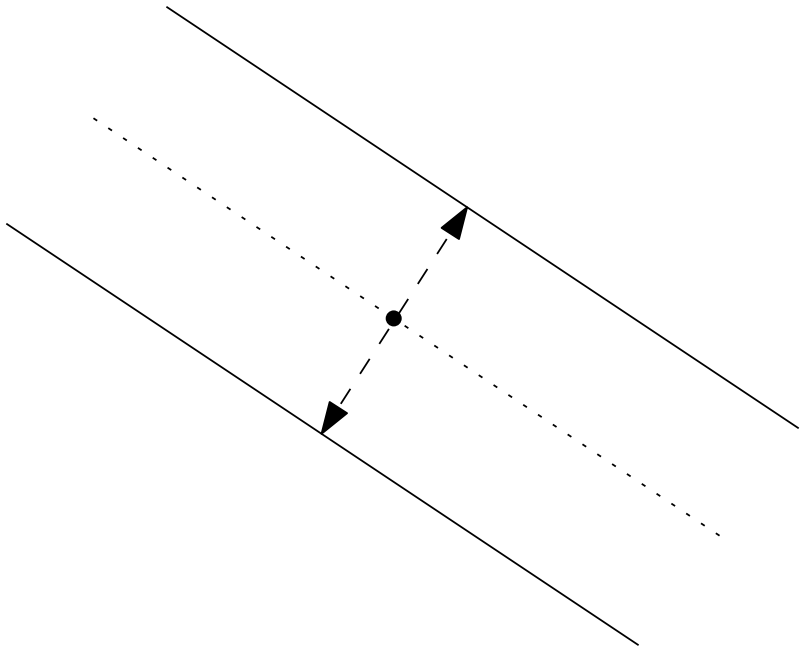


Bregman smoothing

More general notion of smoothing:

Bregman smoothing

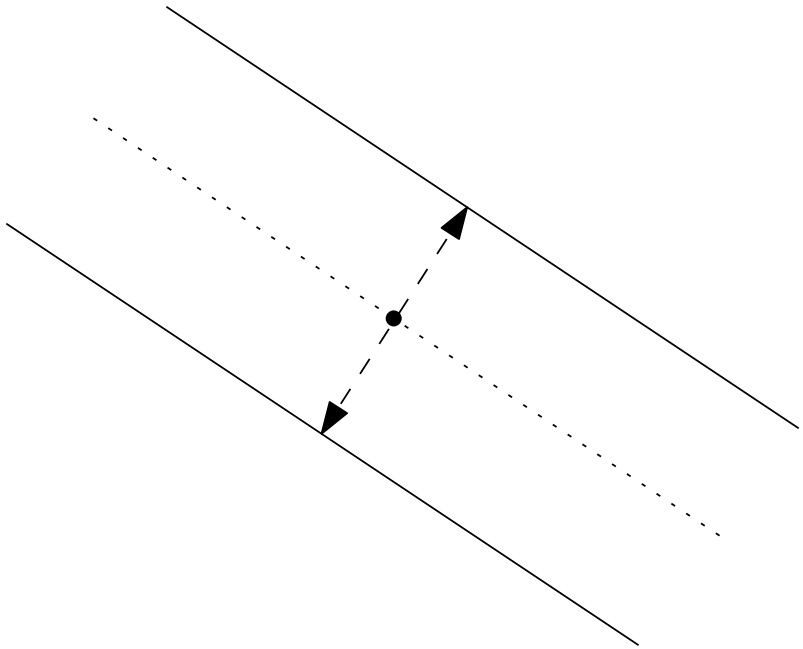
More general notion of smoothing:



- perturbation should stay close to a hyperplane

Bregman smoothing

More general notion of smoothing:



- perturbation should stay close to a hyperplane
- density of perturbation is proportional to $1/\sigma^d$

Bregman smoothing: Results

Theorem ([MR09])

For "well-behaved" Bregman divergences, smoothed complexity is bounded by $poly(n^{\sqrt{k}}, 1/\sigma)$ *and* $k^{kd}poly(n, 1/\sigma)$.

This is in comparison to worst-case bound of $2^{\Omega(n)}$.

Performance Analysis

Two questions:

Problem (Rate of convergence)

Given an arbitrary set of n points in d dimensions, how long does it take for (Bregman) k-means to converge ?

Problem (Quality of Solution)

Let OPT denote the optimal clustering that minimizes the average sum of (Bregman) distances to cluster centers. How close to OPT is the solution returned by (Bregman) k-means ?

Performance Analysis

Two questions:

Problem (Quality of Solution)

Let OPT denote the optimal clustering that minimizes the average sum of (Bregman) distances to cluster centers. How close to OPT is the solution returned by (Bregman) k-means ?

Optimality and Approximations

Problem

Given $x_1, \ldots, x_n$, and parameter k, find k centers $c_1, \ldots, c_k$ such that

$$\sum_{x=1}^{n} \min_{j=1}^{k} d(x_i, c_j)$$

is minimized.

Optimality and Approximations

Problem

Given $x_1, \ldots, x_n$, and parameter k, find k centers $c_1, \ldots, c_k$ such that

$$\sum_{x=1}^{n} \min_{j=1}^{k} d(x_i, c_j)$$

is minimized.

Problem (c-approximation)

Let OPT be the optimal solution above. Fix $c > 0$. Find centers $c'_1, \ldots c'_k$ such that if $A = \sum_{x=1}^{n} \min_{j=1}^{k} d(x_i, c'_j)$, then

$$OPT \leq A \leq c \cdot OPT$$

k-means++: Initialize carefully!

Initialization

- Let distance from x to nearest cluster center be $D(x)$
- Pick x as new center with probability

$$p(x) \propto D^2(x)$$

Properties of solution:

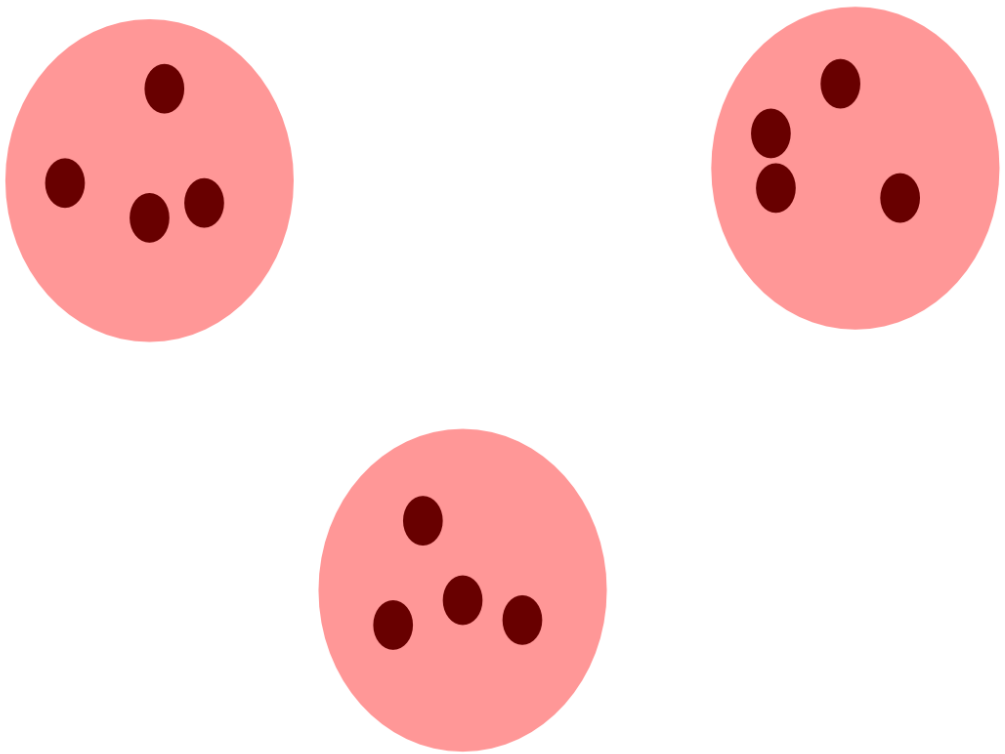
- For arbitrary data, this gives $O(\log n)$-approximation
- For "well-separated data", this gives constant ($O(1)$)-approximation.

What is 'well-separated'

Informally, data is (k, α)-well separated if the best clustering that uses $k-1$ clusters has cost that is $\geq 1/\alpha \cdot \text{OPT}$.

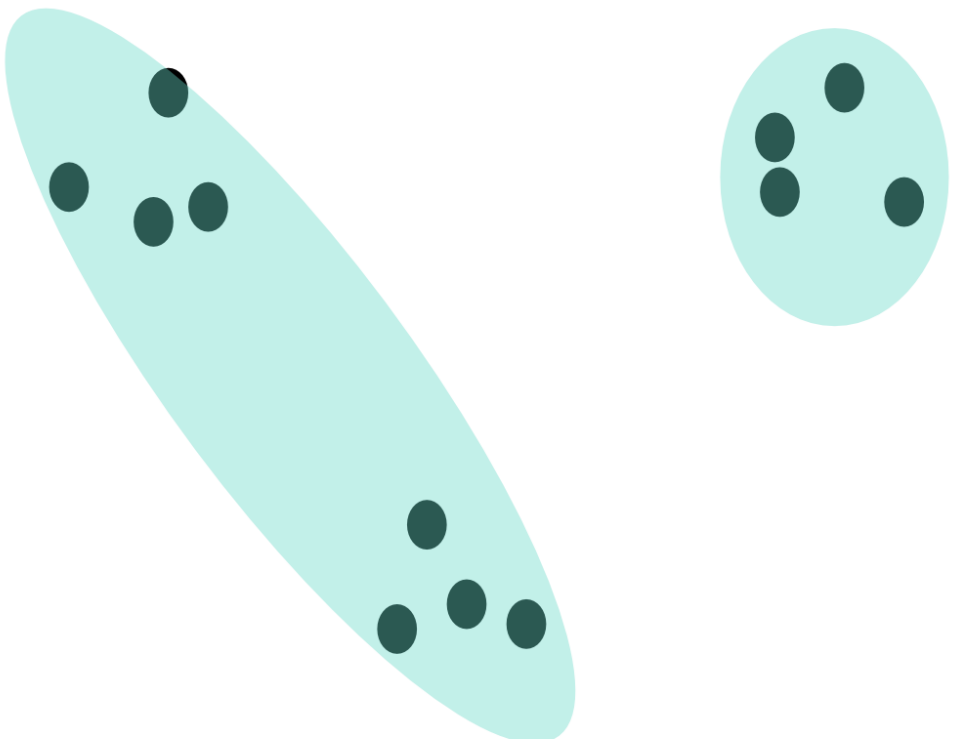
What is 'well-separated'

Informally, data is (k, α)-well separated if the best clustering that uses $k-1$ clusters has cost that is $\geq 1/\alpha \cdot \text{OPT}$.



What is 'well-separated'

Informally, data is (k, α)-well separated if the best clustering that uses $k-1$ clusters has cost that is $\geq 1/\alpha \cdot \text{OPT}$.



Bregman k-means++

Initialization

- Let Bregman divergence from x to nearest cluster center be $D(x)$
- Pick x as new center with probability

$$p(x) \propto D(x)$$

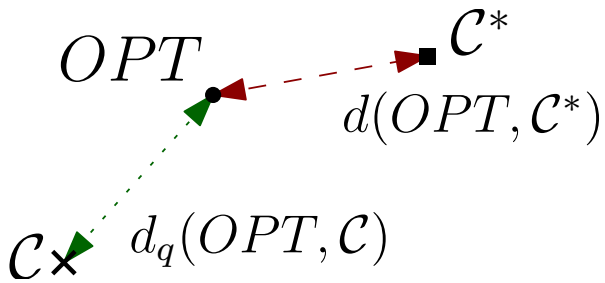
Run algorithm as before.

Theorem ([AB09, AB10])

- $O(1)$-*approximation for* (k, α)-*separated sets.*
- $O(\log n)$ *approximation in general.*

Stability in clustering

Target and Optimal clustering



Two measures of cost:

- Distance between clusterings $\mathscr{C}, \mathscr{C}^*$:

$$d(\mathscr{C}, \mathscr{C}^*) = \text{fraction of points on which they disagree}$$

- (Quality) distance from $\mathscr{C}$ to OPT:

$$d_q(\mathscr{C}, OPT) = \frac{\text{cost}(\mathscr{C})}{\text{cost}(OPT)}$$

Can closeness in d_q imply closeness in d ?

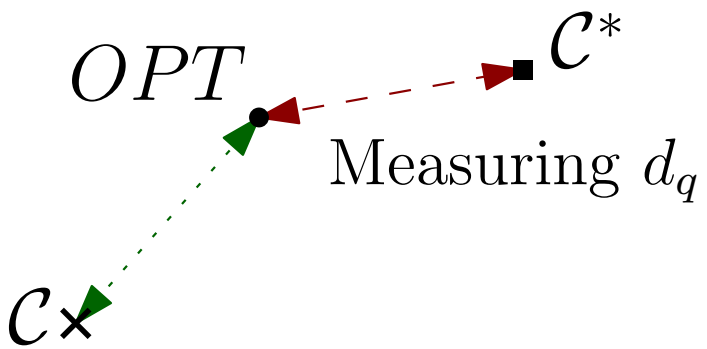
NP-hardness

NP-hardness is an obstacle to finding good clusterings.

- k-means and k-median are NP-hard, and hard to approximate in general graphs
- k-means, k-median can be approximated in $\mathbb{R}^d$ but seem to need time exponential in d
- Same is true for Bregman clustering[CM08]

Target And Optimal Clusterings

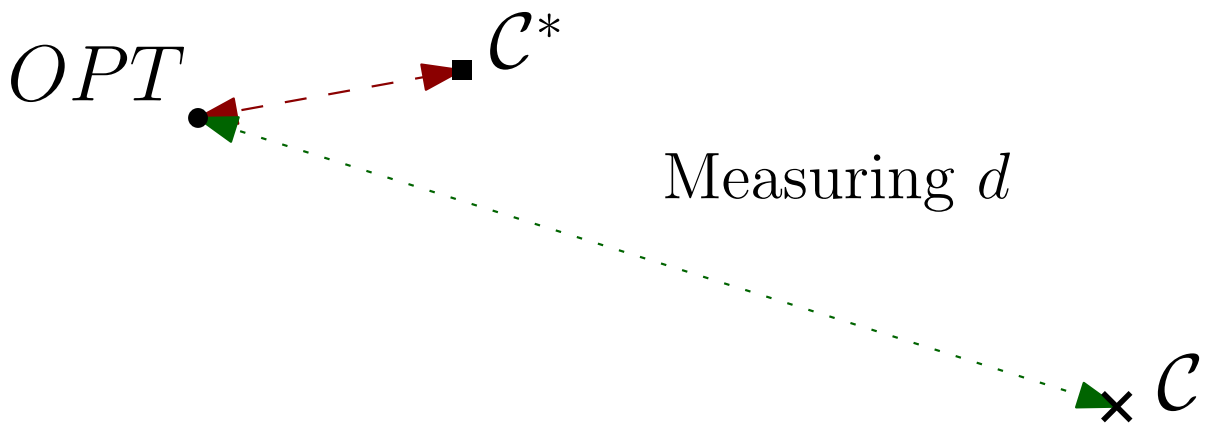
What happens if target clustering and optimal clustering are not the same ?



The two distance functions might be incompatible.

Target And Optimal Clusterings

What happens if target clustering and optimal clustering are not the same ?



The two distance functions might be incompatible.

Stability Of Clusterings

An instance is *stable* if approximating the cost function gives us a solution close to the target clustering.

View 1: If we perturb inputs, the output should not change.

Stability Of Clusterings

An instance is *stable* if approximating the cost function gives us a solution close to the target clustering.

View 1: If we perturb inputs, the output should not change.

View 2: If we change the distance function, output should not change.

Stability Of Clusterings

An instance is *stable* if approximating the cost function gives us a solution close to the target clustering.

View 1: If we perturb inputs, the output should not change.

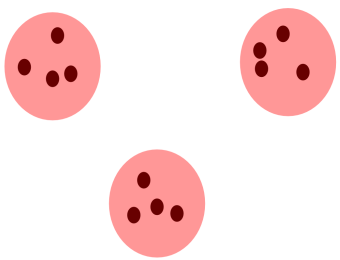
View 2: If we change the distance function, output should not change.

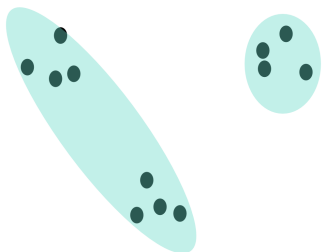
View 3: If we change the cost quality of solution, then output should not change.

Stability I: Perturbing Inputs

Well separated sets:

Data is (k, α)-well separated if the best clustering that uses $k-1$ clusters has cost that is $\geq 1/\alpha \cdot \mathrm{OPT}$.





Two interesting properties[ORSS06]:

- All optimal clusterings mostly look the same: d_q small $\Rightarrow d$ small.
- Small perturbations of the data don't change this property.

Computationally, well-separatedness makes k-means work well

Stability II: Perturbing Distance Function

Definition (α-perturbations[BL09])

A clustering instance (P,d) is α-perturbation-resilient if the optimal clustering is identical to the optimal clustering for any (P,d'), where

$$d(x,y)/\alpha \leq d'(x,y) \leq d(x,y) \cdot \alpha$$

- The smaller the α, the more resilient the instance (and the more "stable")
- Center-based clustering problems (k-median, k-means, k-center) can be solved optimally for $\sqrt{3}$-perturbation-resilient inputs[ABS10]

Stability III: Perturbing Quality of Solution

Definition ((c, ϵ)-property[BBG09])

Given an input, all clusterings that are c-approximate are also ϵ-close.

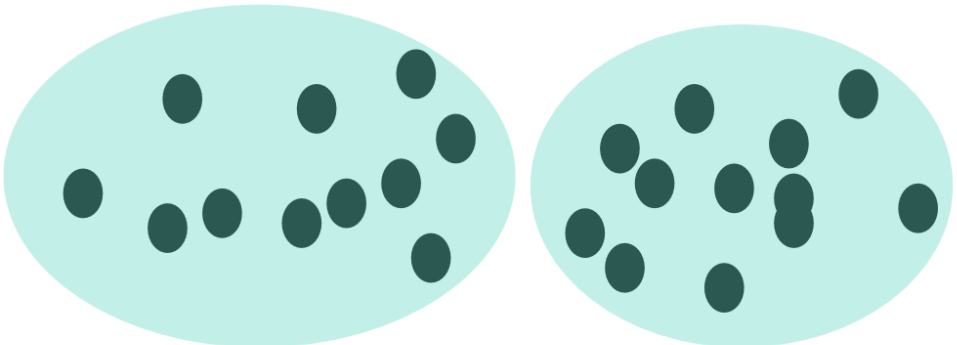
Surprising facts:

- Finding a c-approximation in general might be NP-hard.
- Finding a c-approximation here is easy !

Proof Idea

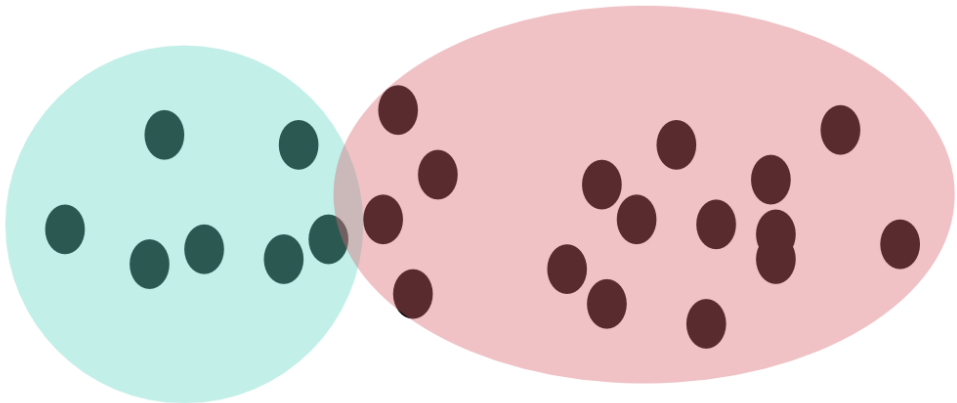
- If near-optimal clusters are close to true answer, then clusters must be well-separated.
- If clusters are well-separated, then choosing the right threshold separates them cleanly.
- Important that **ALL** near-optimal clusterings are close to true answer.

Proof Idea



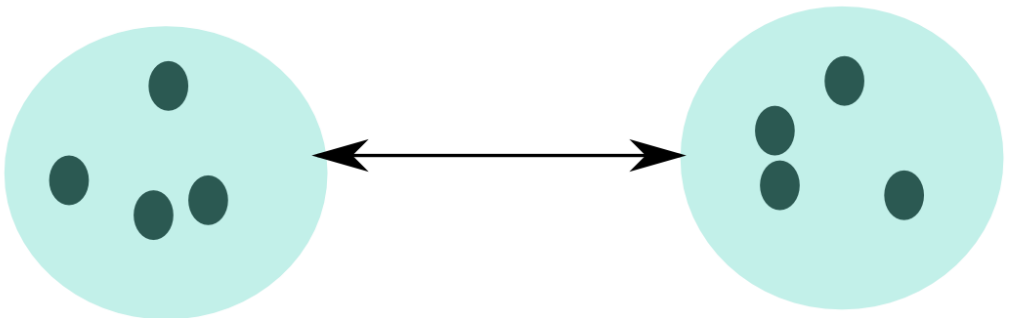
- If near-optimal clusters are close to true answer, then clusters must be well-separated.
- If clusters are well-separated, then choosing the right threshold separates them cleanly.
- Important that **ALL** near-optimal clusterings are close to true answer.

Proof Idea



- If near-optimal clusters are close to true answer, then clusters must be well-separated.
- If clusters are well-separated, then choosing the right threshold separates them cleanly.
- Important that **ALL** near-optimal clusterings are close to true answer.

Proof Idea



- If near-optimal clusters are close to true answer, then clusters must be well-separated.
- If clusters are well-separated, then choosing the right threshold separates them cleanly.
- Important that **ALL** near-optimal clusterings are close to true answer.

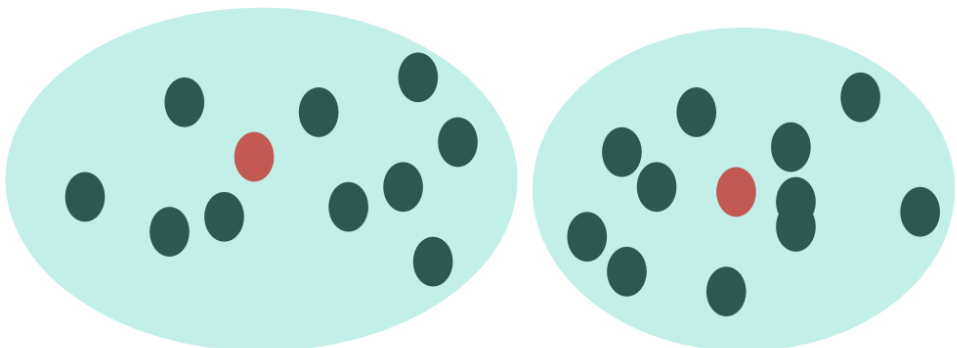
Main Result

Theorem

In polynomial time, we can find a clustering that is $O(\epsilon)$-close to the target clustering, even if finding a c-approximation is NP-hard.

Generalization

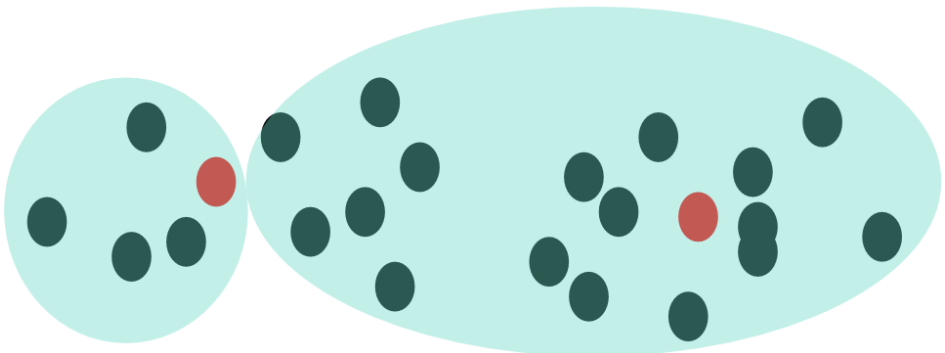
Strong assumption: **ALL** near-optimal clusterings are close to true answer.
Variant[ABS10]: Only consider *Voronoi*-based clusterings, where each point is assigned to *nearest* cluster center.



Same results hold as for previous case.

Generalization

Strong assumption: **ALL** near-optimal clusterings are close to true answer.
Variant[ABS10]: Only consider *Voronoi*-based clusterings, where each point is assigned to *nearest* cluster center.



Same results hold as for previous case.

Wrap Up

- We understand much more about the behavior of k-means, and why it does well in practice.
- A simple initialization procedure for k-means is both effective and gives provable guarantees
- Much of the theoretical machinery around k-means works for the generalization to Bregman divergences.
- New and interesting questions on the relationship between the target clustering and cost measures used to get near it: ways of subverting NP-hardness.

Thank You

Slides for this tutorial can be found at

`http://www.cs.utah.edu/~suresh/web/2010/05/08/new-developments-in-the-theory-of-clustering-tutorial/`

Research on this tutorial was partially supported by NSF CCF-0953066

References I

Marcel R. Ackermann and Johannes Blömer.

Coresets and approximate clustering for bregman divergences.
In Mathieu [Mat09], pages 1088–1097.

Marcel R. Ackermann and Johannes Blömer.

Bregman clustering for separable instances.
In Kaplan [Kap10], pages 212–223.

P. Awasthi, A. Blum, and O. Sheffet.

Clustering Under Natural Stability Assumptions.
Computer Science Department, page 123, 2010.

Ankit Aggarwal, Amit Deshpande, and Ravi Kannan.

Adaptive sampling for k-means clustering.
In *APPROX '09 / RANDOM '09: Proceedings of the 12th International Workshop and 13th International Workshop on Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 15–28, Berlin, Heidelberg, 2009. Springer-Verlag.

Nir Ailon, Ragesh Jaiswal, and Claire Monteleoni.

Streaming k-means approximation.
In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 10–18. 2009.

David Arthur, Bodo Manthey, and Heiko Röglin.

k-means has polynomial smoothed complexity.
In *FOCS '09: Proceedings of the 2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 405–414, Washington, DC, USA, 2009. IEEE Computer Society.

References II

David Arthur and Sergei Vassilvitskii.

k-means++: the advantages of careful seeding.
In *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

Maria-Florina Balcan, Avrim Blum, and Anupam Gupta.

Approximate clustering without the approximation.
In Mathieu [Mat09], pages 1068–1077.

Yonatan Bilu and Nathan Linial.

Are stable instances easy?
CoRR, abs/0906.3162, 2009.

Arindam Banerjee, Srujana Merugu, Inderjit S. Dhillon, and Joydeep Ghosh.

Clustering with bregman divergences.
Journal of Machine Learning Research, 6:1705–1749, 2005.

Kamalika Chaudhuri and Andrew McGregor.

Finding metric structure in information theoretic clustering.
In Servedio and Zhang [SZ08], pages 391–402.

Yingfei Dong, Ding-Zhu Du, and Oscar H. Ibarra, editors.

Algorithms and Computation, 20th International Symposium, ISAAC 2009, Honolulu, Hawaii, USA, December 16-18, 2009. Proceedings, volume 5878 of *Lecture Notes in Computer Science*. Springer, 2009.

S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan.

Clustering data streams.
In *FOCS '00: Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, page 359, Washington, DC, USA, 2000. IEEE Computer Society.

References III

Haim Kaplan, editor.

Algorithm Theory - SWAT 2010, 12th Scandinavian Symposium and Workshops on Algorithm Theory, Bergen, Norway, June 21-23, 2010. Proceedings, volume 6139 of *Lecture Notes in Computer Science*. Springer, 2010.

Claire Mathieu, editor.

Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009. SIAM, 2009.

Bodo Manthey and Heiko Röglin.

Worst-case and smoothed analysis of -means clustering with bregman divergences.
In Dong et al. [DDI09], pages 1024–1033.

Rafail Ostrovsky, Yuval Rabani, Leonard J. Schulman, and Chaitanya Swamy.

The effectiveness of lloyd-type methods for the k-means problem.
In *FOCS '06: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 165–176, Washington, DC, USA, 2006. IEEE Computer Society.

Rocco A. Servedio and Tong Zhang, editors.

21st Annual Conference on Learning Theory - COLT 2008, Helsinki , Finland, July 9-12, 2008. Omnipress, 2008.

Andrea Vattani.

k-means requires exponentially many iterations even in the plane.
In *SCG '09: Proceedings of the 25th annual symposium on Computational geometry*, pages 324–332, New York, NY, USA, 2009. ACM.