K-Center Clustering

Isuru Gunasekara

University of Ottawa

aguna100@uottawa.ca

November 21,2016

Overview

What is Clustering?

The process of finding interesting structure in a set of given data [1].
A clustering problem is usually defined by a set of items and a distance function defined between these items.

Metric space

Metric space

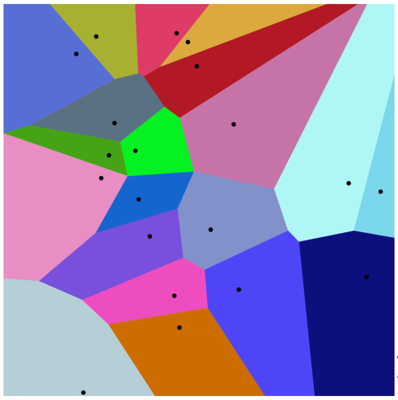
A metric space is a pair ($\mathcal{X}$,d) where $\mathcal{X}$ is a set and **d**:$\mathcal{X} \times \mathcal{X} \to [0, \infty)$ is a metric, satisfying the following axioms:

1. Reflexivity: $d(x, y) = 0 \iff x = y$
2. Symmetry: $d(x, y) = d(y, x)$
3. Triangle inequality: $d(x, z) \leq d(x, y) + d(y, z)$

A very common example of a metric space is $\mathbb{R}^2$ with regular Euclidean distance.

Voronoi Partitions

- Given a set of centers $\mathcal{C}$, every point of $\mathbf{P}$ is assigned to it's nearest neighbor in $\mathcal{C}$
- All the points of $\mathbf{P}$ that are assigned to a center $\bar{c}$ form the cluster of $\bar{c}$, denoted by: $\Pi(\mathcal{C}, \bar{c}) = \{p \in \mathbf{P} | d(p, \bar{c}) \leq d(p, \mathcal{C})\}$
- This scheme of partitioning is known as *Voronoi partitions*

[1]

[1] Source: https://en.wikipedia.org/wiki/Voronoi_diagram

Problem Statement

Clustering

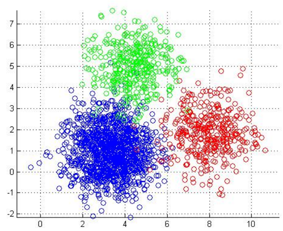
Preliminaries
k-Center
GreedyKCenter
Greedy Permutation
k-median clustering
Local Search

For a metric space $(\mathcal{X}$,d),

- Input: a set $\mathbf{P} \subseteq \mathcal{X}$,and a parameter k.
- Output: a set $\mathcal{C}$ of k points.
- Goal: Minimize the cost $r_\infty^{\mathcal{C}}(\mathbf{P}) = \max_{p \in P}$ d(p,$\mathcal{C}$)

Formally, $r_\infty^{opt}(\mathbf{P}, k) = \min_{C,|C|=k} r_\infty^{\mathcal{C}}(\mathbf{P})$

- That is, Every point in a cluster is in distance at most $r_\infty^{\mathcal{C}}(P)$ from it's respective center.
- k-center clustering is NP-HARD.



[a]

[a] Source: https://www.mathworks.com

Approximation algorithms

- It's unlikely that there can ever be efficient polynomial time exact algorithms solving NP-hard problems.
- Therefore we'll have to resort to approximation algorithms such as:
  - Greedy algorithms.
  - Local search.
  - $\cdots$

The Greedy Clustering Algorithm

- Pick an arbitrary point $\bar{c}_1$ into C_1
- For every point $p \in \mathbf{P}$ compute $d_1[p]$ from $\bar{c}_1$
- Pick the point $\bar{c}_2$ with highest distance from $\bar{c}_1$. (This is the point realizing $r_1 = \max_{p \in P} d_1[p]$)
- Add it to the set of centers and denote this expanded set of centers as C_2. Continue this till k centers are found

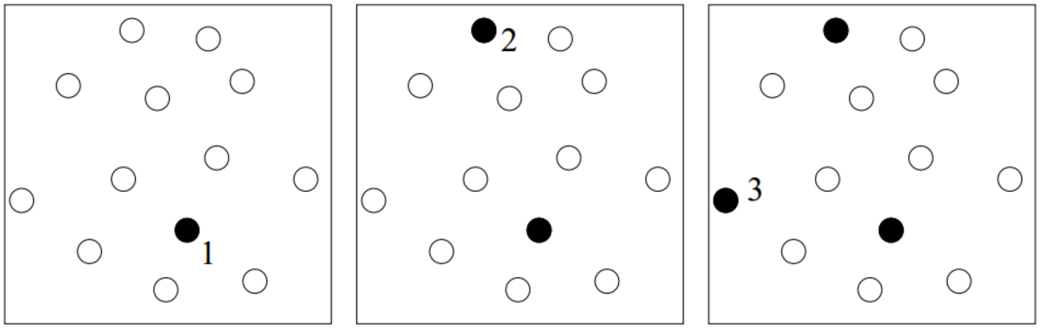


Figure: Visualizing the greedy algorithm[2]

[2] Source: Sanders/van Stee: Approximations- und Online-Algorithmen

Making things slightly faster

- In i$^{\text{th}}$ iteration the point $\bar{c}_i$ realizing, $r_{i-1} = \max_{p \in P} d_{i-1}[p] = \max_{p \in P} d(p, C_{i-1})$ is added to the set of centers C_{i-1} to form C_i
- r_{i-1} is the radius of the clustering and is calculated in every iteration.
- This process is repeated k times
- That is, in every iteration, the distance from all points in **P** to the set of current centers is calculated.

- But,

$$\begin{aligned} d_i[p] &= d(p, C_i) \\ &= min(d(p, C_{i-1}), d(p, \bar{c}_i)) \\ &= min(d_{i-1}[p], d(p, \bar{c}_i)) \end{aligned}$$

- What if for each $p \in P$ we maintain a single variable $d[p]$ with it's current distance to the closest center in the current center set.
- Then only $d(p, \bar{c}_i)$ is needed to calculate the radius.

Running time

- The i^{th} iteration of choosing the i^{th} center takes $\mathcal{O}(n)$ time.
- There are k such iterations.
- Thus, overall the algorithm takes $\mathcal{O}(nk)$ time

Example

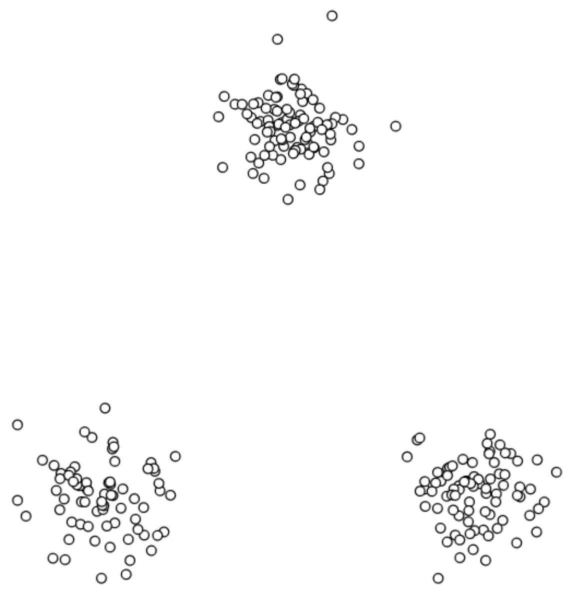
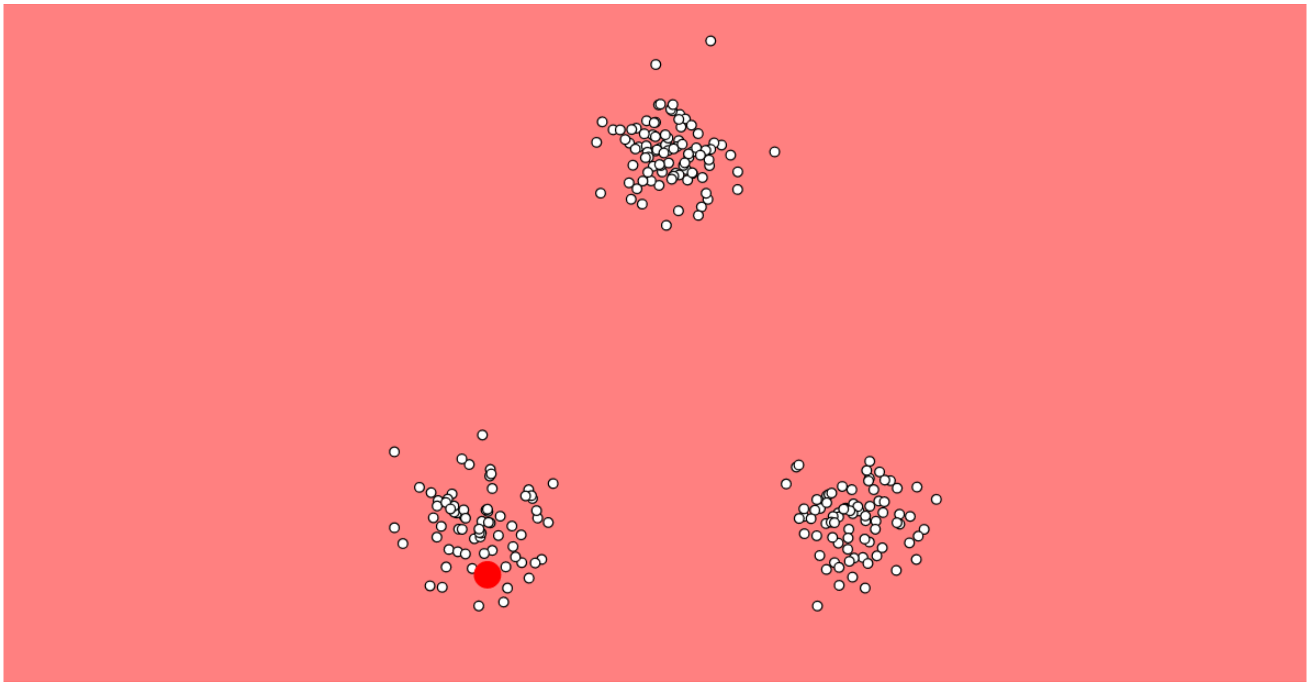


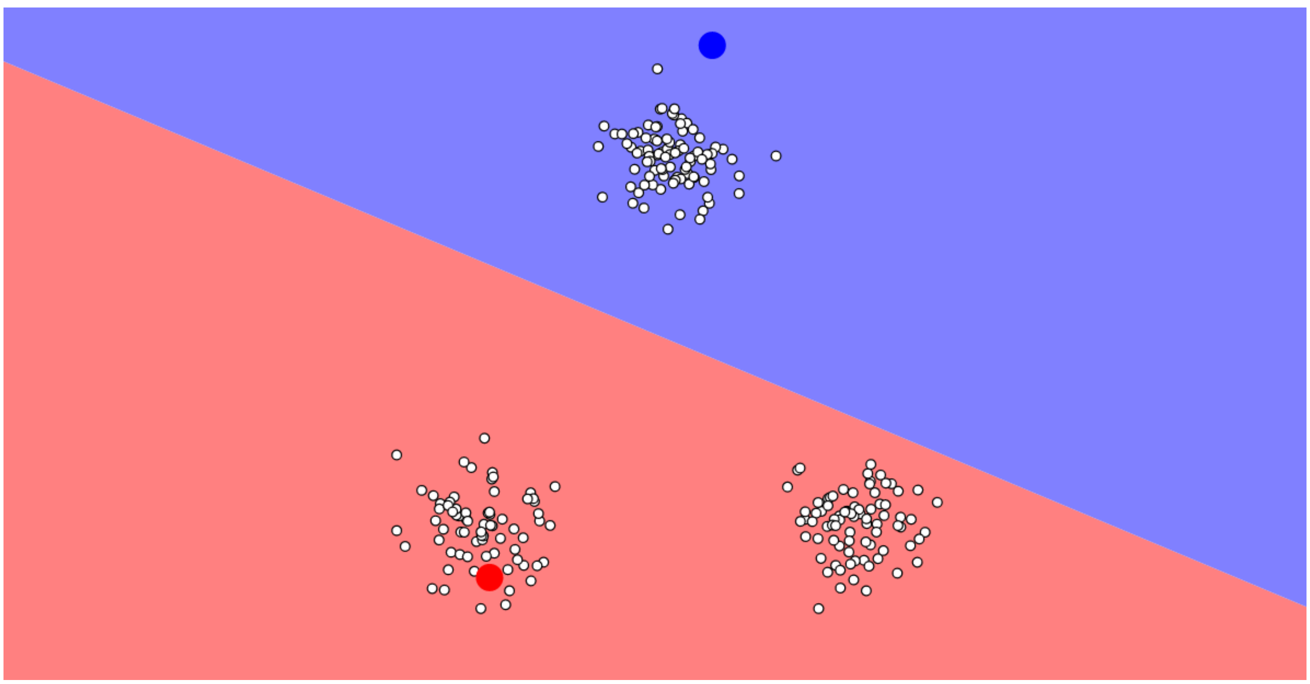
Figure: Example dataset[3]

[3] Source: https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

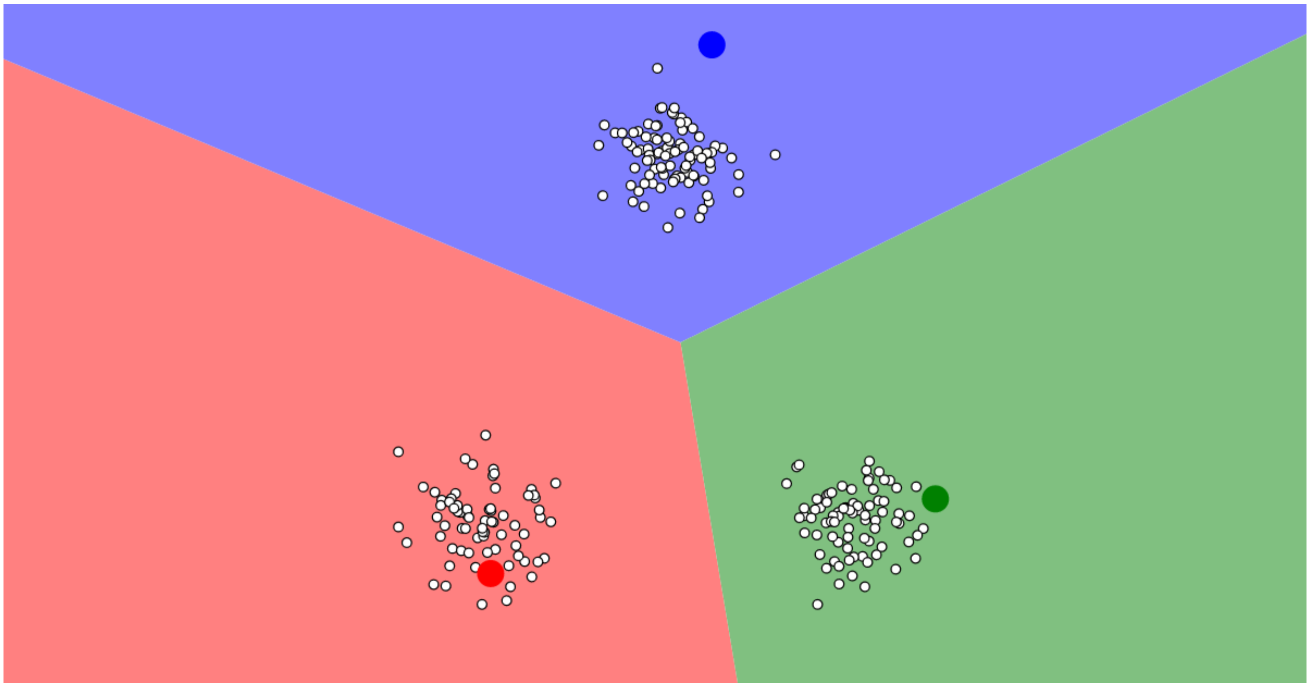
Example



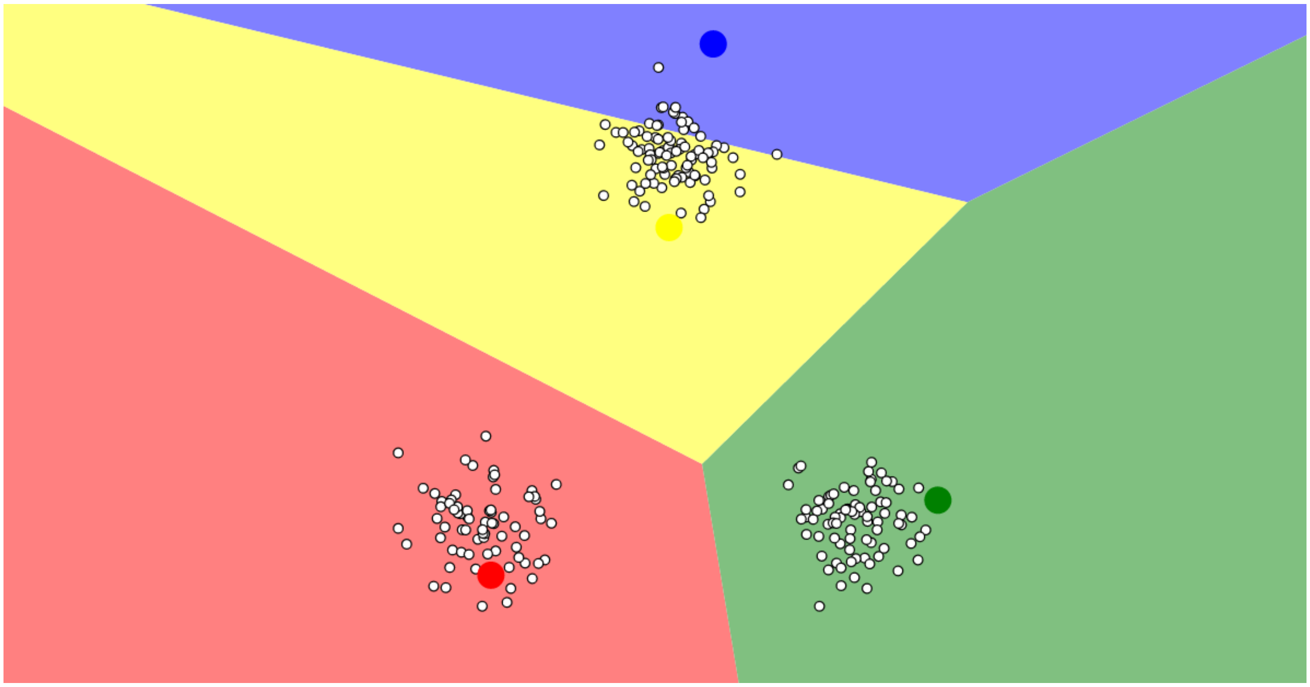
Example



Example



Importance of choosing the right k



2-approximation

Theorem

Given a set of n points $\mathbf{P} \subseteq \mathcal{X}$,belonging to a metric space ($\mathcal{X}$,d), the greedy K-center algorithm computes a set $\mathbf{K}$ of k centers, such that $\mathbf{K}$ is a 2-approximation to the optimal k-center clustering of $\mathbf{P}$.

$r_\infty^{\mathbf{K}}(\mathbf{P}) \leq 2r_\infty^{opt}(\mathbf{P}, k)$

The algorithm takes $\mathcal{O}(nk)$ time.

Proof

Case 1: Every cluster of $\mathcal{C}_{opt}$ contains exactly one point of $\mathbf{K}$

- Consider a point $p \in \mathbf{P}$
- Let $\bar{c}$ be the center it belongs to in $\mathcal{C}_{opt}$
- Let $\bar{k}$ be the center of $\mathbf{K}$ that is in $\Pi(\mathcal{C}_{opt}, \bar{c})$
- $d(p, \bar{c}) = d(p, \mathcal{C}_{opt}) \leq r_{\infty}^{opt}(\mathbf{P}, k)$
- Similarly, $d(\bar{k}, \bar{c}) = d(\bar{k}, \mathcal{C}_{opt}) \leq r_{\infty}^{opt}$
- By the triangle inequality: $d(p, \bar{k}) \leq d(p, \bar{c}) + d(\bar{c}, \bar{k}) \leq 2r_{\infty}^{opt}$

Proof, continued...

Case 2: There are two centers $\bar{k}$ and $\bar{u}$ of $\mathbf{K}$ that are both in $\Pi(\mathcal{C}_{opt}, \bar{c})$, for some $\bar{c} \in \mathcal{C}_{opt}$ (By pigeon hole principle, this is the only other possibility)

- Assume, without loss of genarality, that $\bar{u}$ was added later to the center set $\mathbf{K}$ by the greedy algorithm, say in i^{th} iteration.
- But since the greedy algorithm always chooses the point furthest away from the current set of centers, we have that $\bar{c} \in \mathcal{C}_{i-1}$ and,

$$
\begin{aligned}
r_\infty^{\mathbf{K}}(\mathbf{P}) \leq r_\infty^{\mathcal{C}_{i-1}}(\mathbf{P}) &= d(\bar{u}, \mathcal{C}_{i-1}) \\
&\leq d(\bar{u}, \bar{k}) \\
&\leq d(\bar{u}, \bar{c}) + d(\bar{c}, \bar{k}) \\
&\leq 2 r_\infty^{opt}
\end{aligned}
$$

The greedy permutation

- What if $k = n$
- Then the algorithm generates a permutation of $\mathbf{P}$
- That is, $\mathbf{P} = \mathcal{C} = \langle \bar{c}_1, \bar{c}_2, ..., \bar{c}_n \rangle$
- $\mathcal{C}$ is the *greedy permutation* of $\mathbf{P}$.
- The associated sequence of radiuses $= \langle r_1, r_2, ..., r_n \rangle$
- All the points of $\mathbf{P}$ are in distance at most r_i from the points of $\mathcal{C}_i = \langle \bar{c}_1, \bar{c}_2, ..., \bar{c}_i \rangle$

r-net

Definition

A set $S \subseteq \mathbf{P}$ is a *r-net* for $\mathbf{P}$ if the following two properties hold

- **Covering property**: All the points of $\mathbf{P}$ are in distance at most r from the points of S
- **Separation property**: For any pair of points $p, q \in S$, $d(p, q) \geq r$

Clustering and *r-nets*

Theorem

Let **P** *be a set of* n *points in a finite metric space, and let its greedy permutation be* $\langle \bar{c}_1, \bar{c}_2, ..., \bar{c}_n \rangle$ *with the associated sequence of radiuses* $\langle \bar{r}_1, \bar{r}_2, ..., \bar{r}_n \rangle$*. For any* i*,* $\mathcal{C}_i = \langle \bar{c}_1, \bar{c}_2, ..., \bar{c}_i \rangle$ *is a* r_i*-net of* **P**

Proof.

Separation property

- $r_k = d(\bar{c}_k, \mathcal{C}_{k-1}) \forall k = 1, .., n$
- For $j < k \leq n, d(\bar{c}_j, \bar{c}_k) \geq r_k$

Covering property follows by the definition of clustering. $\square$

k-median clustering

- Input: A set $\mathbf{P} \subseteq \mathcal{X}$ and a parameter k.
- Output: Find k points $\mathcal{C}$ s.t. the sum of distances of points of $\mathbf{P}$ to their closest point in $\mathcal{C}$ is minimized.

Notations

- Consider the set **U** of all k-tuples of points of **P**
- Let p_i denote the i^{th} point of **P**, for $i = 1, 2, ..., n$, where $n = |\mathbf{P}|$
- For $\mathcal{C} \in \mathbf{U}$, consider the n dimensional point

$$\phi(\mathcal{C}) = (d(p_1, \mathcal{C}), d(p_2, \mathcal{C}), ..., d(p_n, \mathcal{C}))$$

- $r_\infty^{\mathcal{C}}(\mathbf{P}) = ||\phi(\mathcal{C})||_\infty = max_i d(p_i, \mathcal{C})$ (by [4]) and $r_\infty^{opt}(\mathbf{P}, k) = \min_{\mathcal{C} \in \mathbf{U}} ||\phi(\mathcal{C})||_\infty$
- Similarly, $r_1^{\mathcal{C}}(\mathbf{P}) = ||\phi(\mathcal{C})||_1 = \sum_i d(p_i, \mathcal{C})$ and $r_1^{opt}(\mathbf{P}, k) = \min_{\mathcal{C} \in \mathbf{U}} ||\phi(\mathcal{C})||_1$

[4] Weierstrass extreme value theorem

- k-center clustering under this interpretation is just finding the point minimizing the l_∞ norm in a set of points in n dimensions.
- k-median clustering is to find the point minimizing the norm under the l_1 norm.

Relations between p-norms

- The p-norm is given by, $||x||_p = (\sum_{i=1}^{n} |x_i|^p)^{1/p}$
- For $0 < p < q$, $||x||_p \geq ||x||_q$
- $||x||_1 \leq \sqrt{n}||x||_2$ and $||x||_2 \leq \sqrt{n}||x||_\infty$

$2n$-approximation

- For any point set $\mathbf{P}$ of n points and a parameter k,

$$r_\infty^{opt}(\mathbf{P}, k) \leq r_1^{opt}(\mathbf{P}, k) \leq n \cdot r_\infty^{opt}(\mathbf{P}, k)$$

- From above, if we compute a set of centers $\mathcal{C}$,

$$\begin{aligned} r_1^{\mathcal{C}}(\mathbf{P})/2n \leq r_\infty^{\mathcal{C}}(\mathbf{P})/2 \leq r_\infty^{opt}(\mathbf{P}, k) \\ \leq r_1^{opt}(\mathbf{P}, k) \\ (\leq r_1^{\mathcal{C}}(\mathbf{P})) \end{aligned}$$

- This gives, $r_1^{\mathcal{C}}(\mathbf{P}) \leq 2n \, r_1^{opt}(\mathbf{P}, k)$
- Namely, $\mathcal{C}$ is a $2n$-approximation to the optimal solution.

Local Search

- Let $0 < \tau < 1$
- Initially set the current set of centers $\mathcal{C}_{curr}$ to be $\mathcal{C}$
- At each iteration, check if $\mathcal{C}_{curr}$ can be improved by replacing one of the centers.
- There are at most $|\mathbf{P}| \cdot |\mathcal{C}_{curr}| = nk$ choices to consider.
- Pick $\bar{c} \in \mathcal{C}_{curr}$ to throw away and replace it by $\bar{e} \in (\mathbf{P} \setminus \mathcal{C}_{curr})$

- New candidate set of centers $\mathbf{K} \leftarrow (\mathcal{C}_{curr} \setminus \{\bar{c}\}) \cup \{\bar{e}\}$
- If $r_1^{\mathbf{K}}(\mathbf{P}) \leq (1-\tau) r_1^{\mathcal{C}_{curr}}(\mathbf{P})$ then set $\mathcal{C}_{curr} \leftarrow \mathbf{K}$ and repeat.
- Stop when there is no exchange that would improve the current solution by a factor of at least $(1-\tau)$
- The final content of $\mathcal{C}_{curr}$ is the required **constant factor approximation**

Example (k-means)

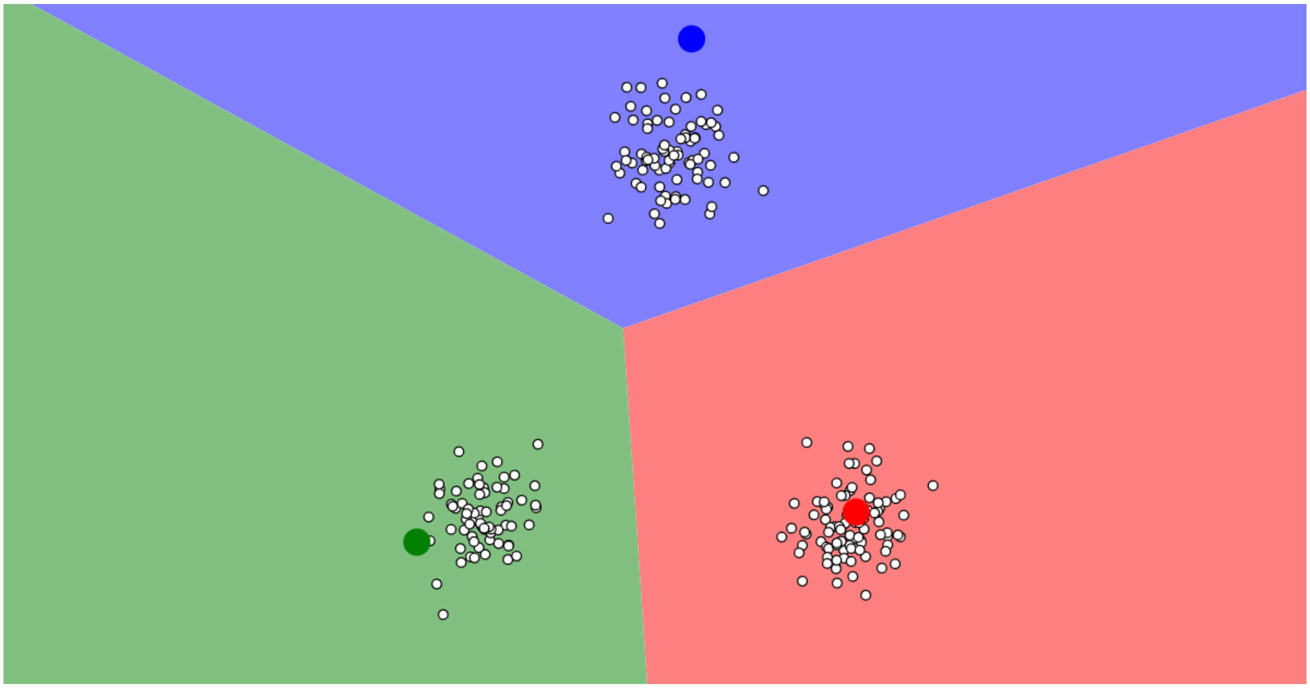
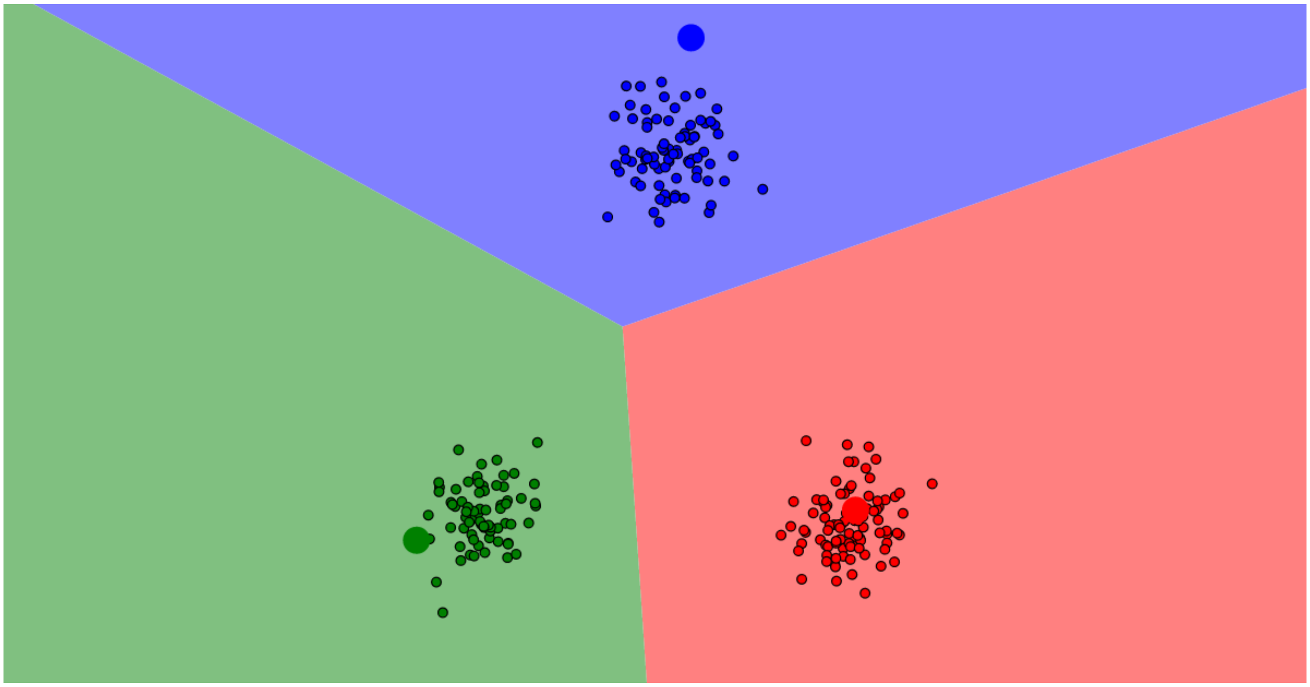


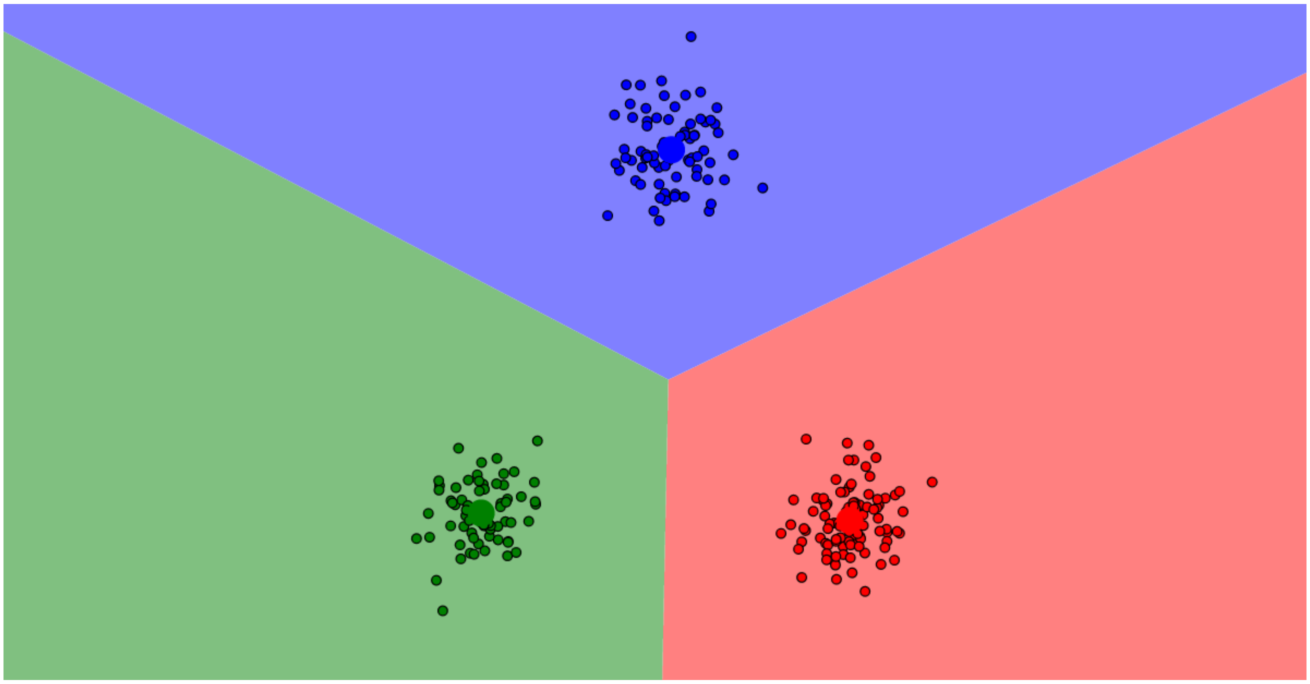
Figure: Dataset initialized with k-center[5]

[5] Source: https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

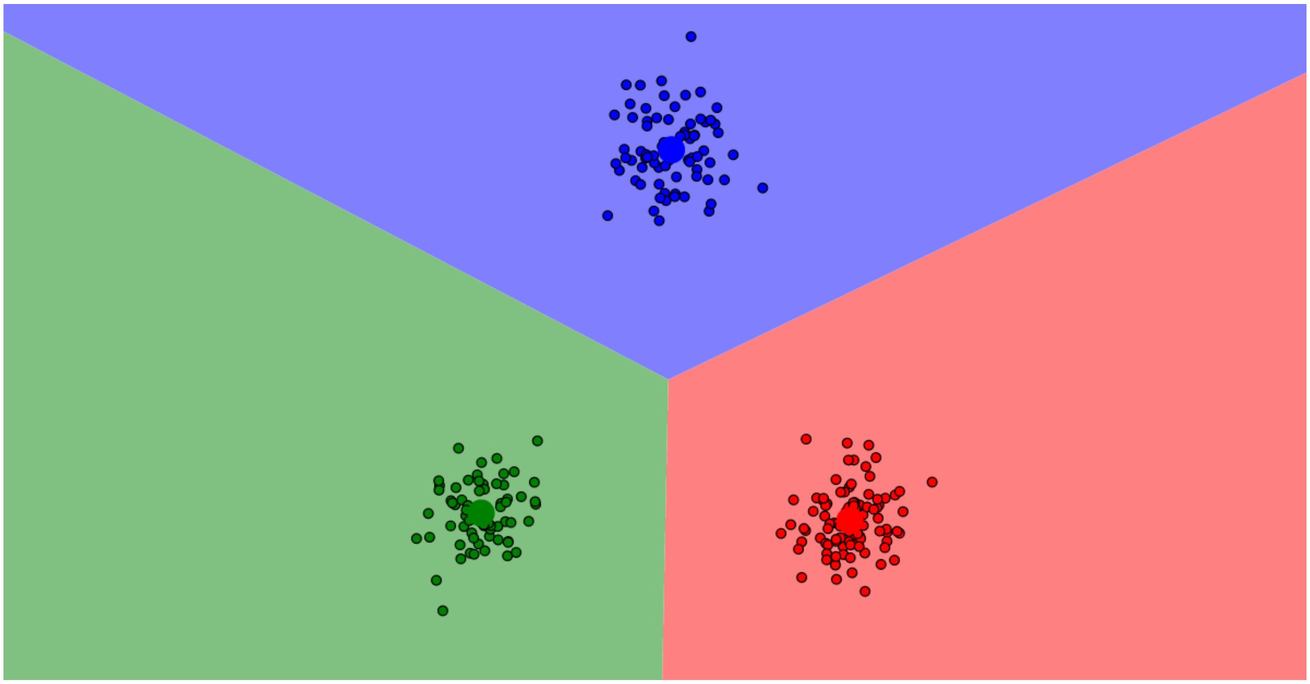
Assign Points



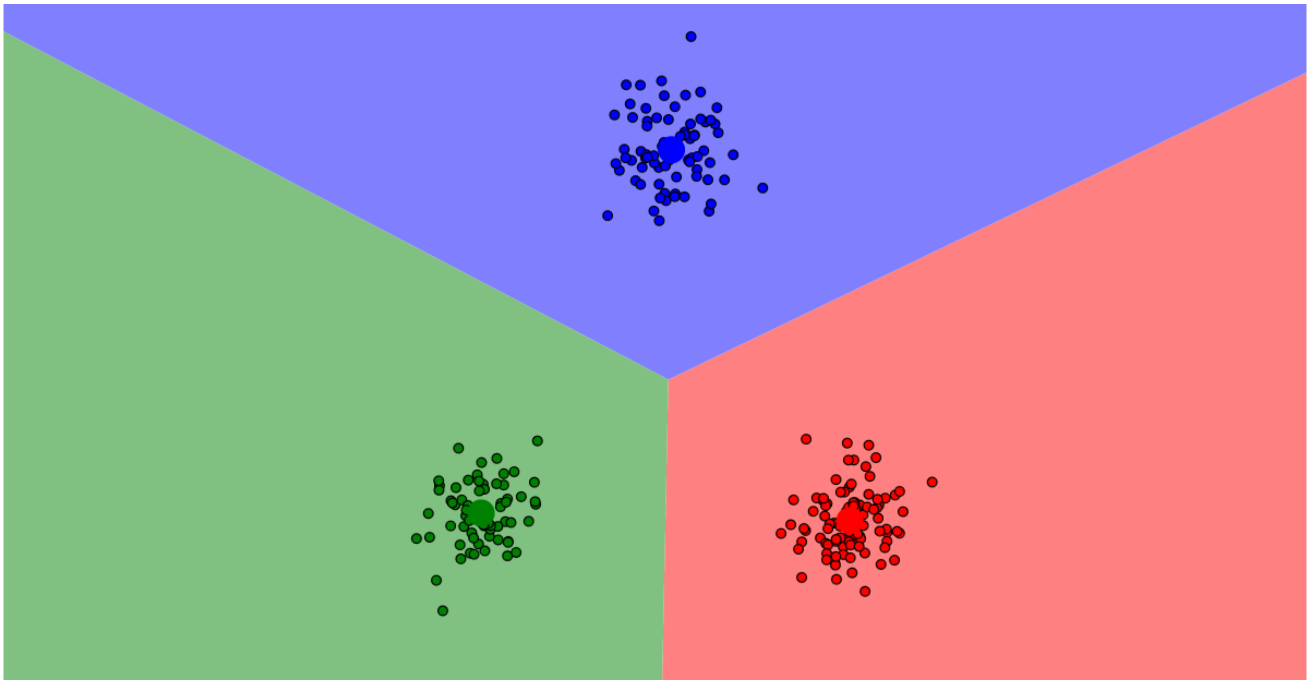
Update Centroids



Re-assign Points



Update Centroids (Nothing changes!)



Example 2 (k-means)

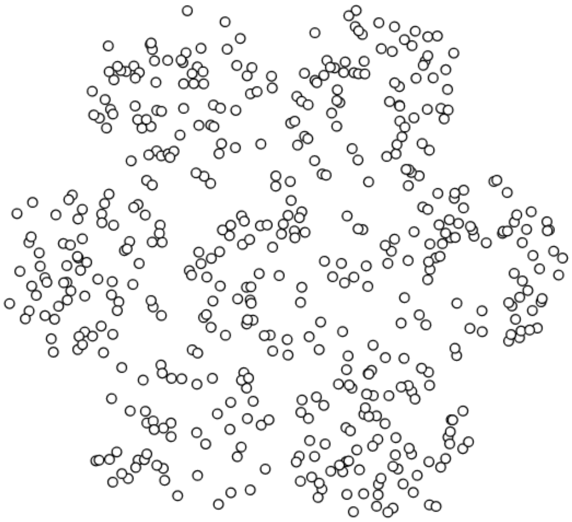
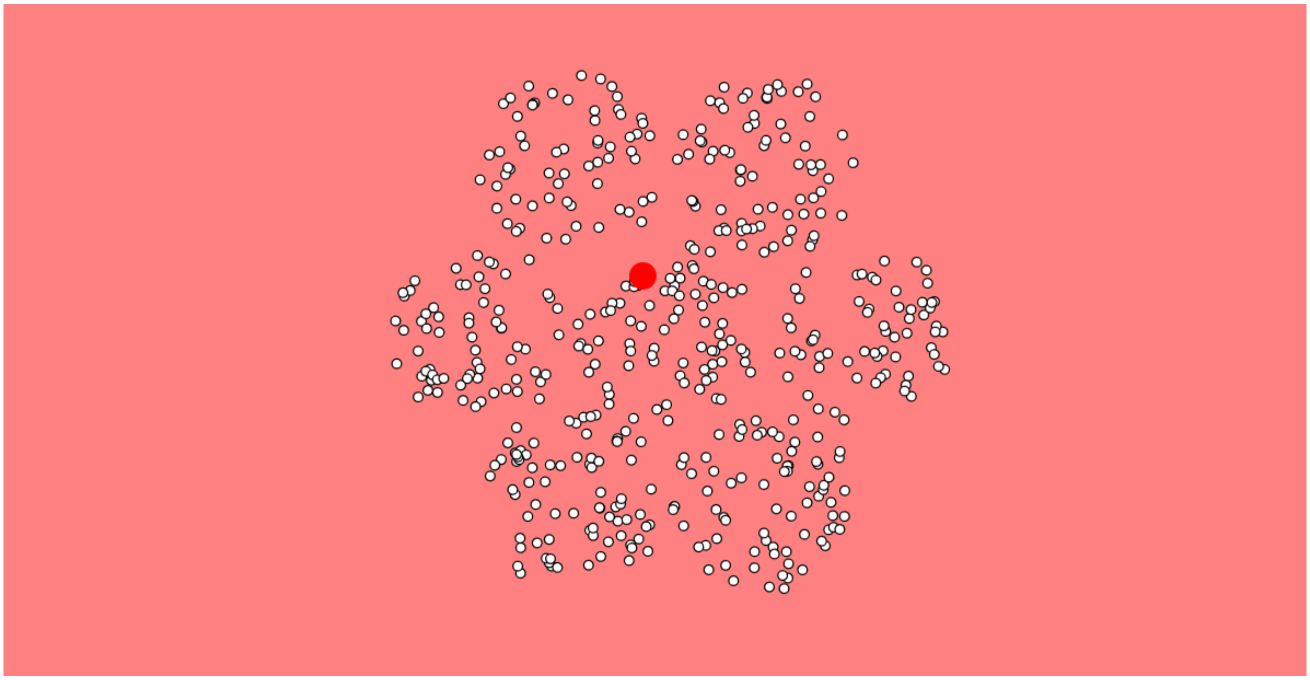


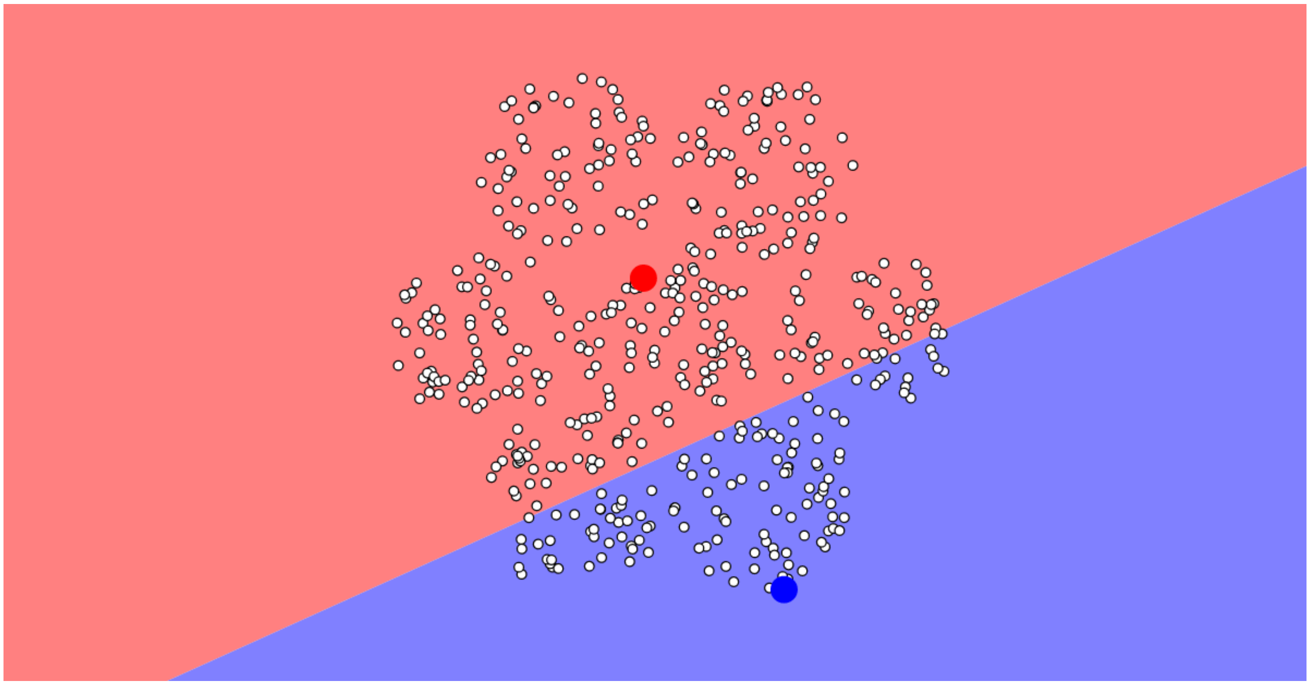
Figure: Example dataset[6]

[6] Source: https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

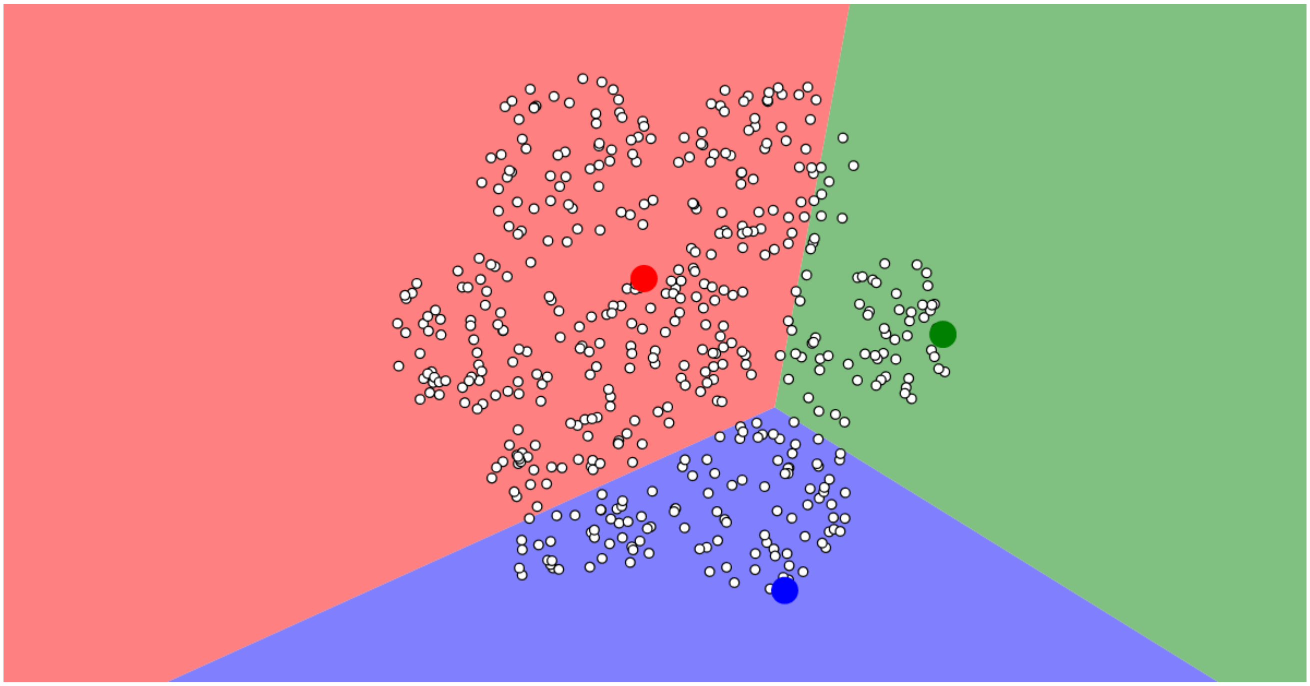
Initialize with k-centers...



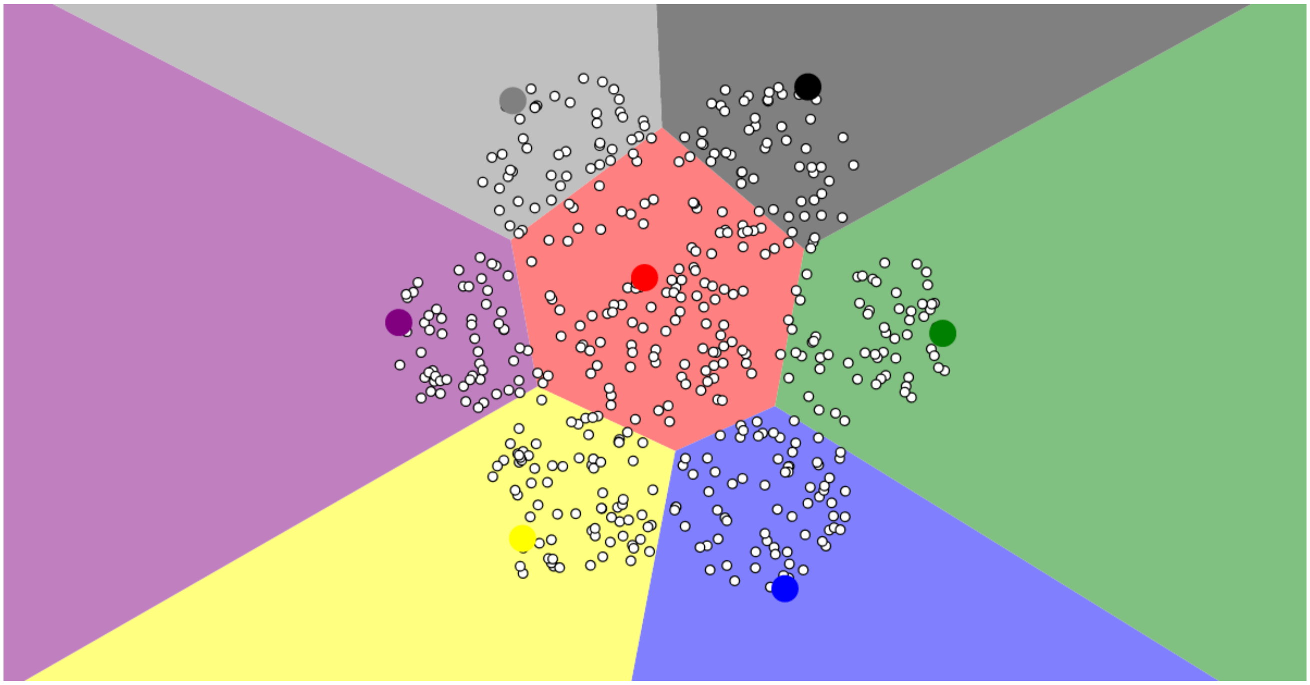
Initialize with k-centers...



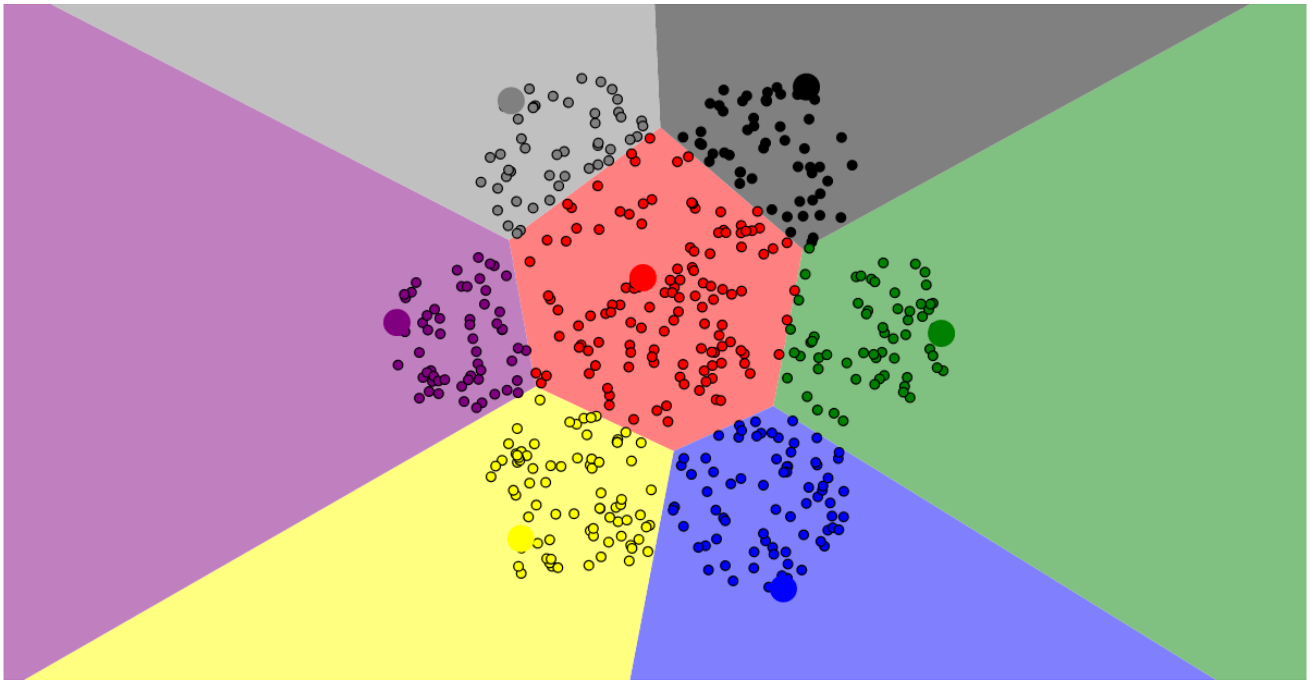
Initialize with k-centers...



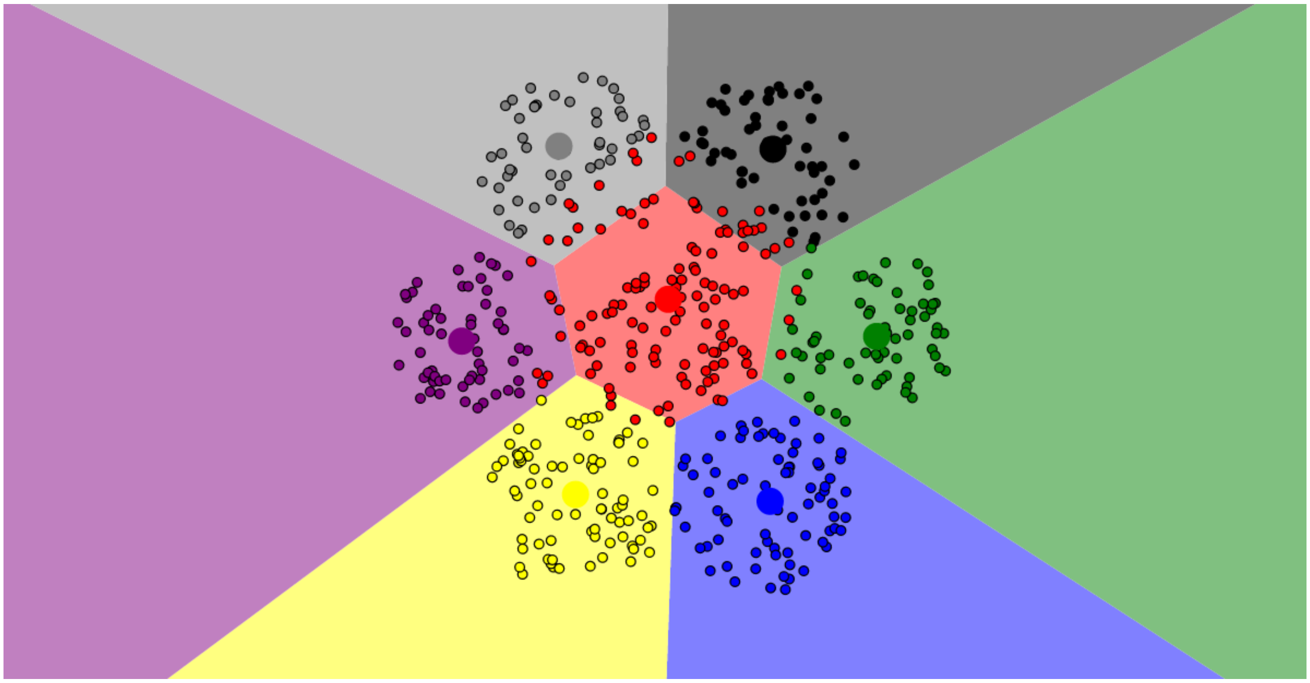
Initialize with k-centers... (Cheating when choosing k!!)



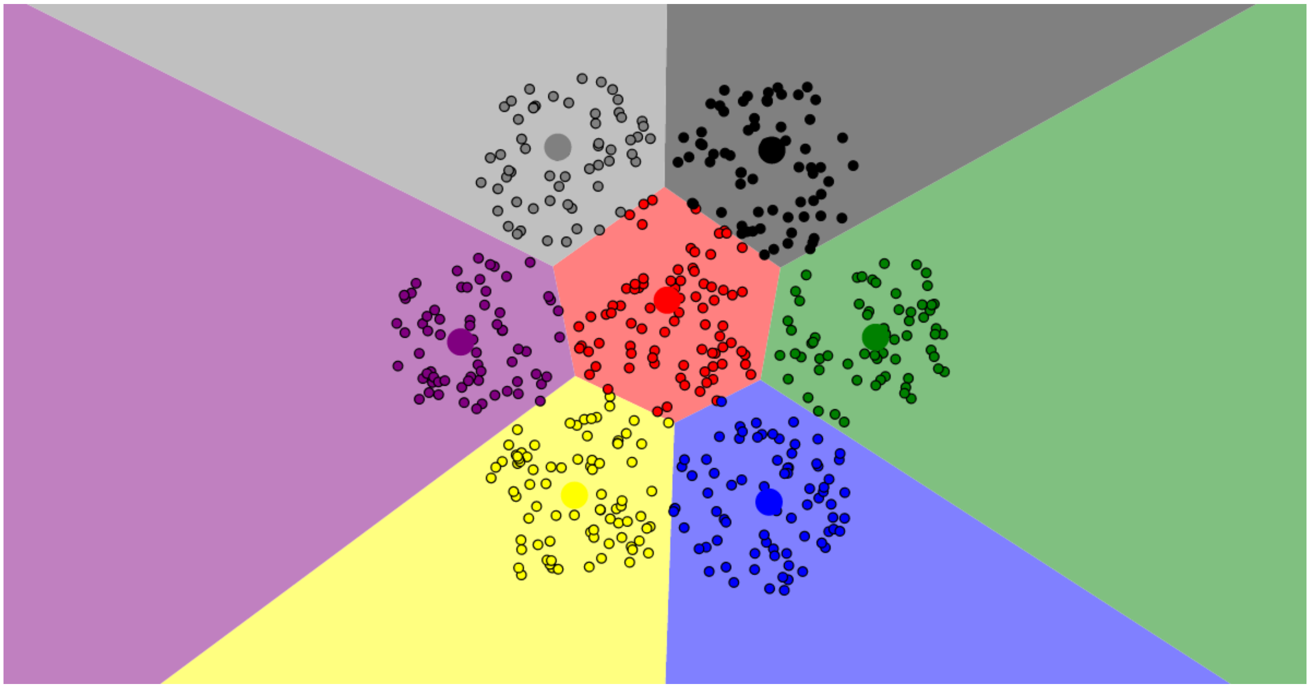
Assign points



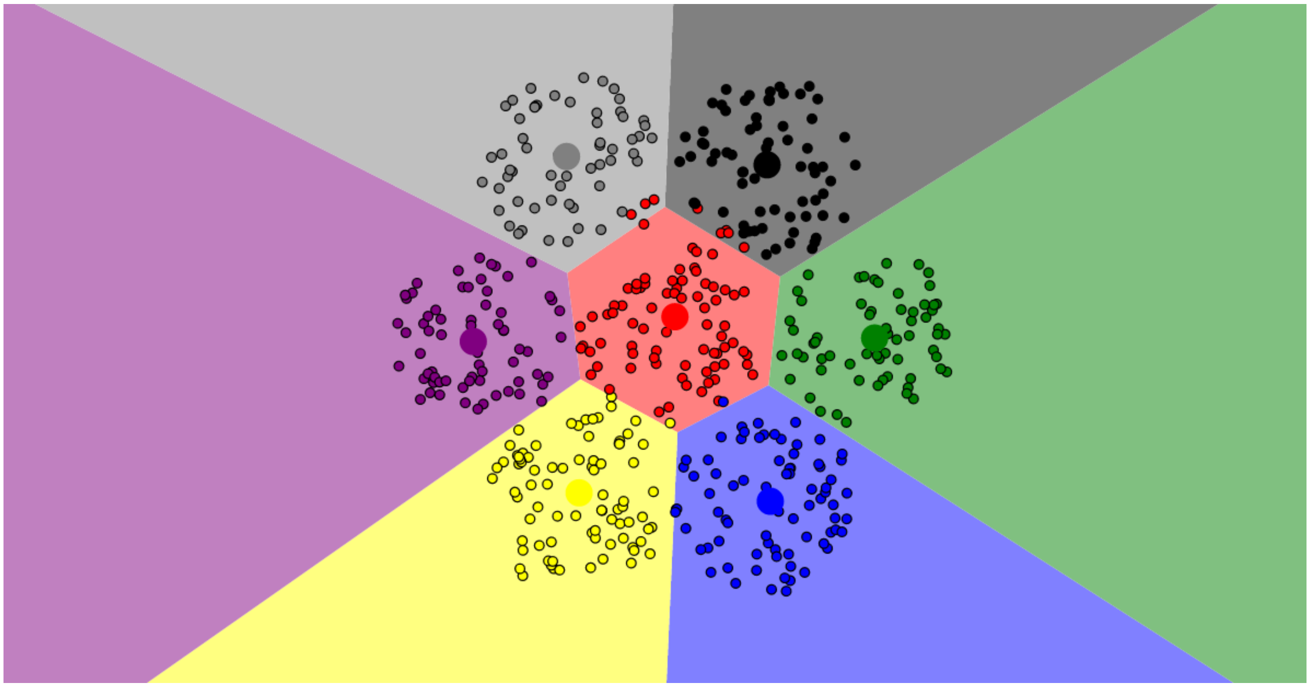
Update Centroids



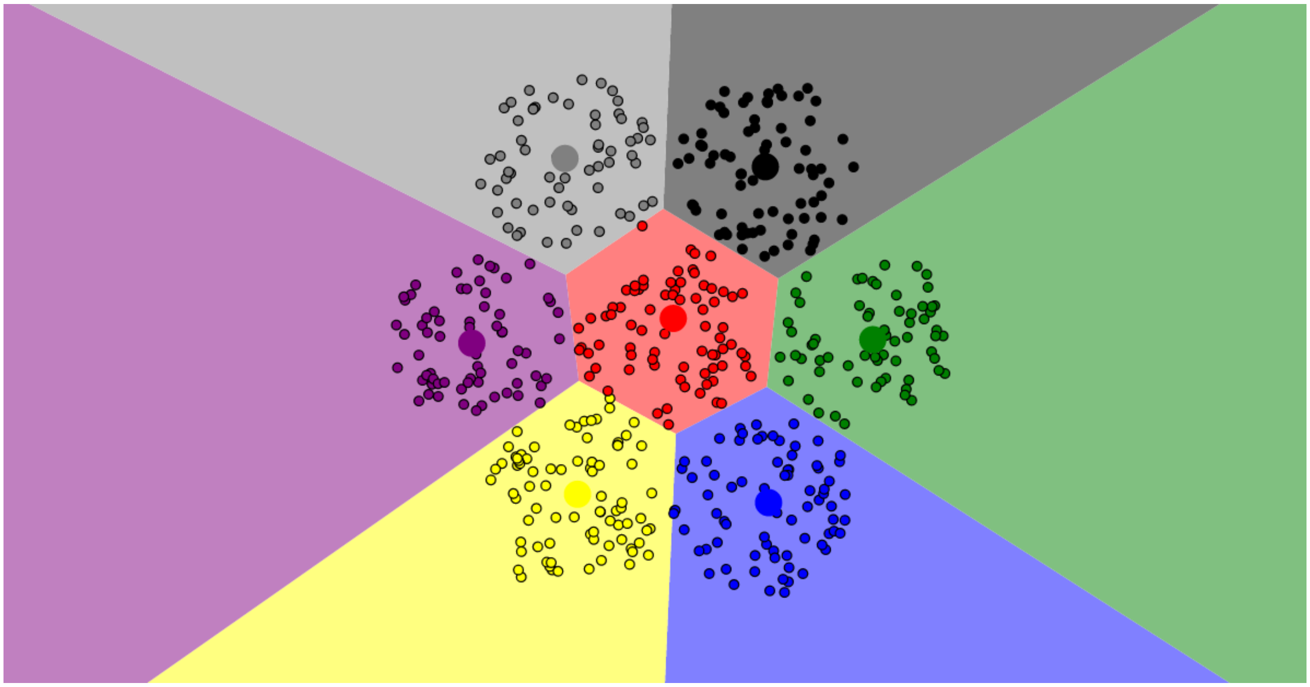
Assign points



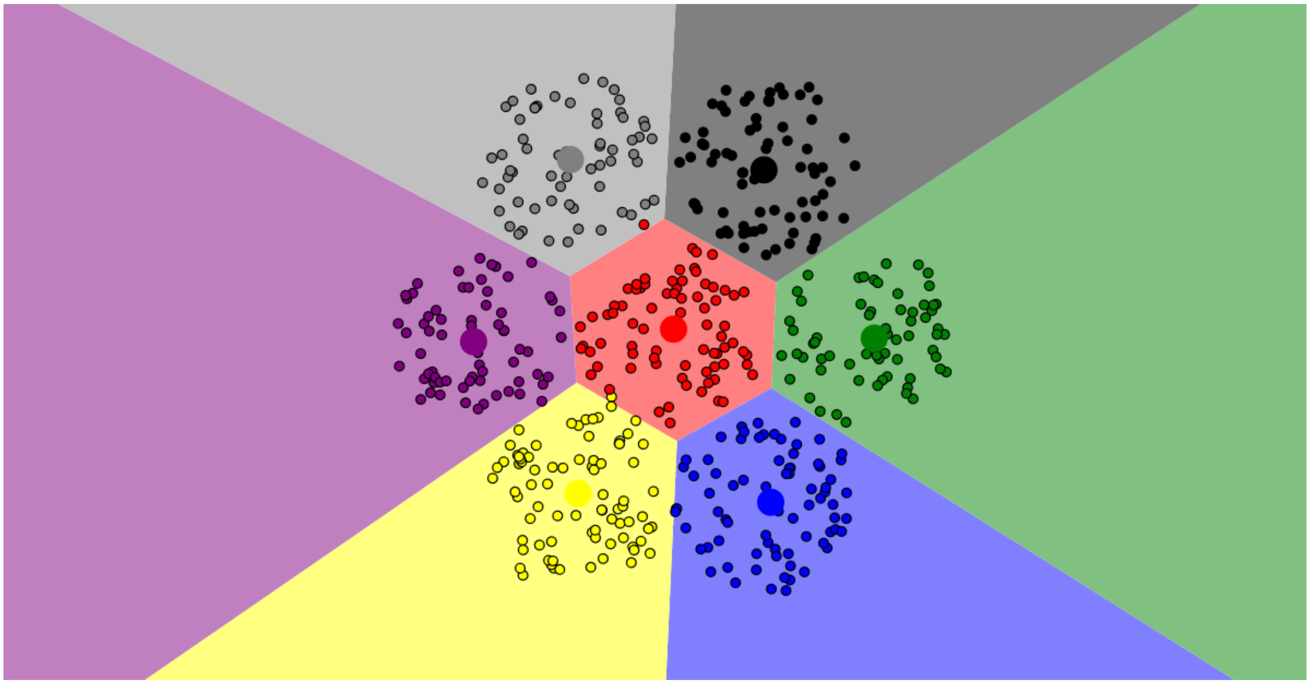
Update Centroids



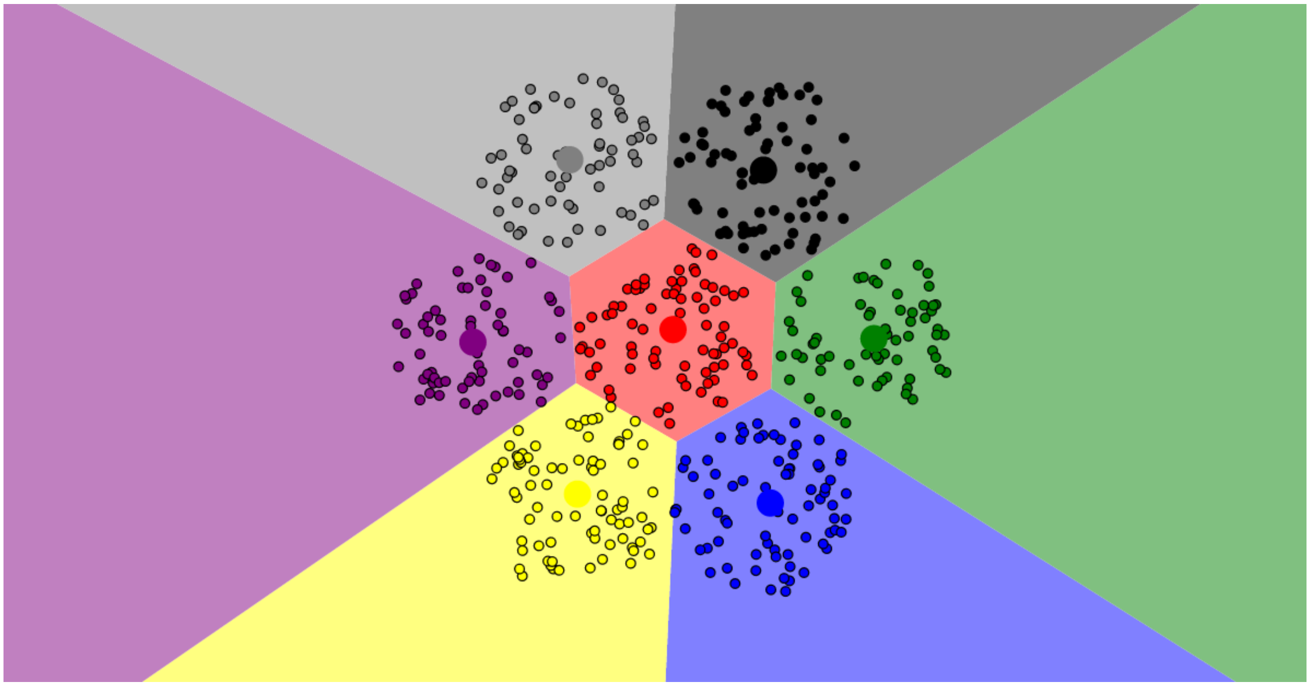
Assign points



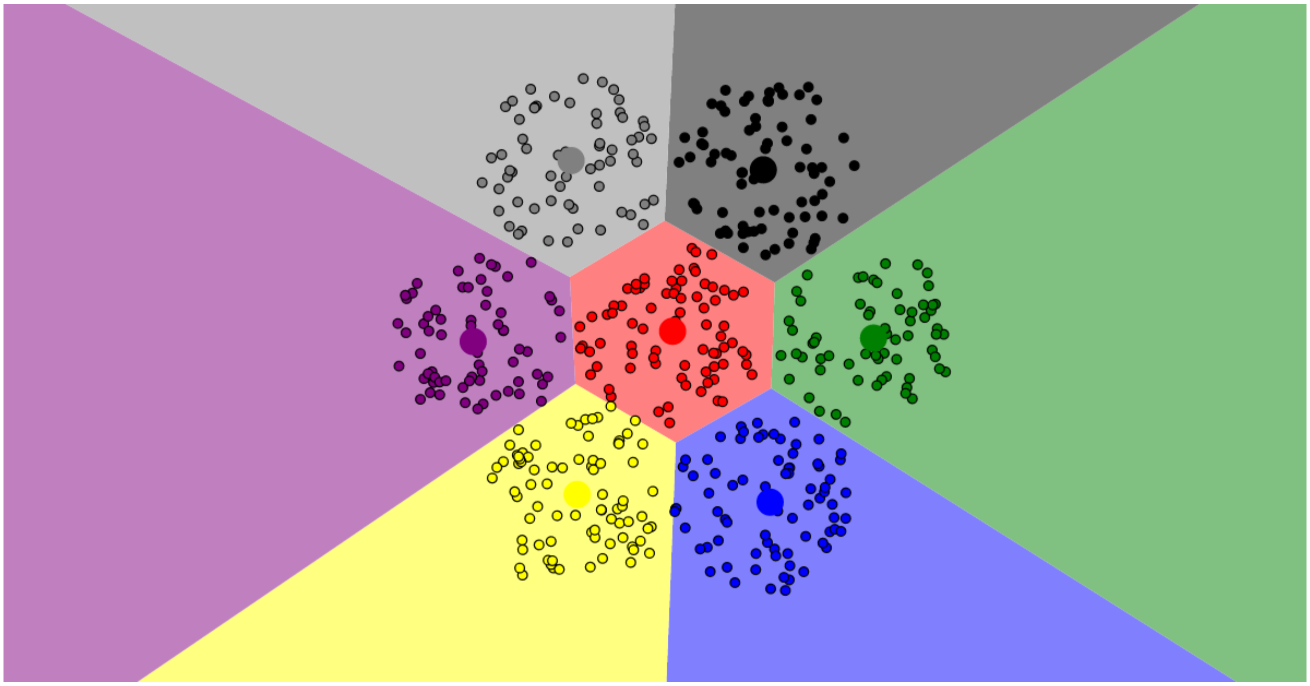
Update Centroids



Assign points



Update Centroids



Assign points (No change)

Clustering

Preliminaries

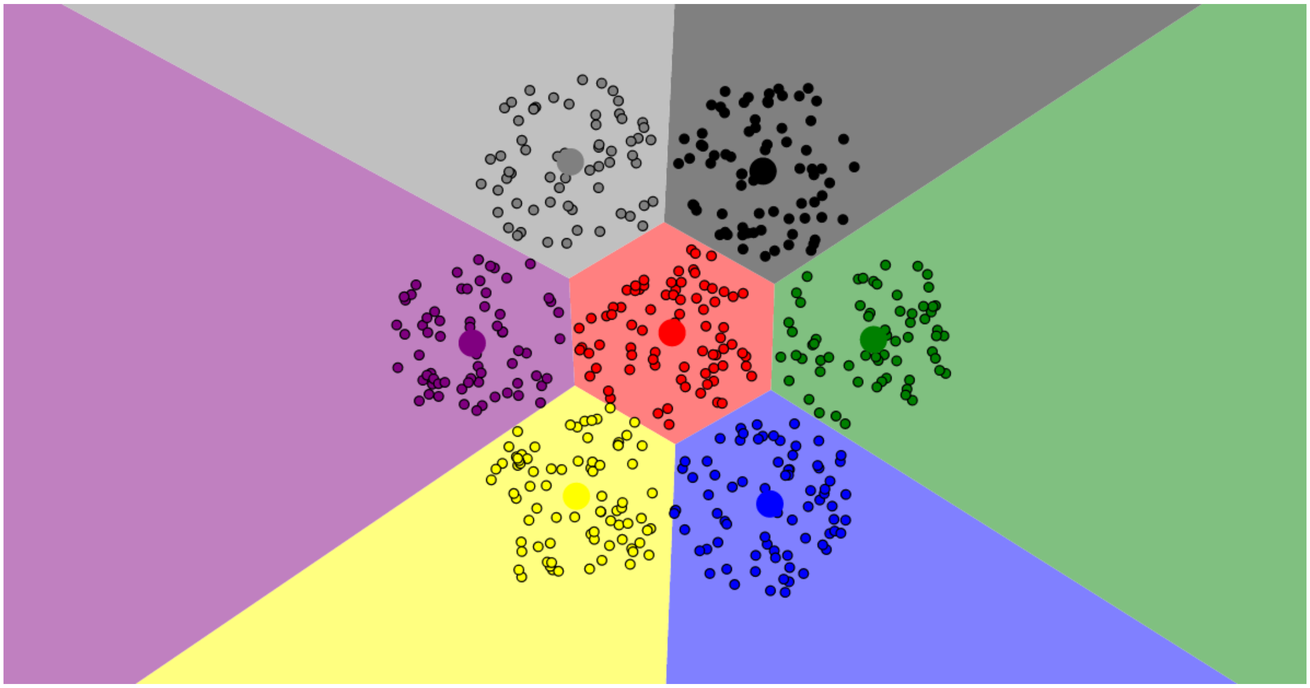
k-Center

GreedyKCenter

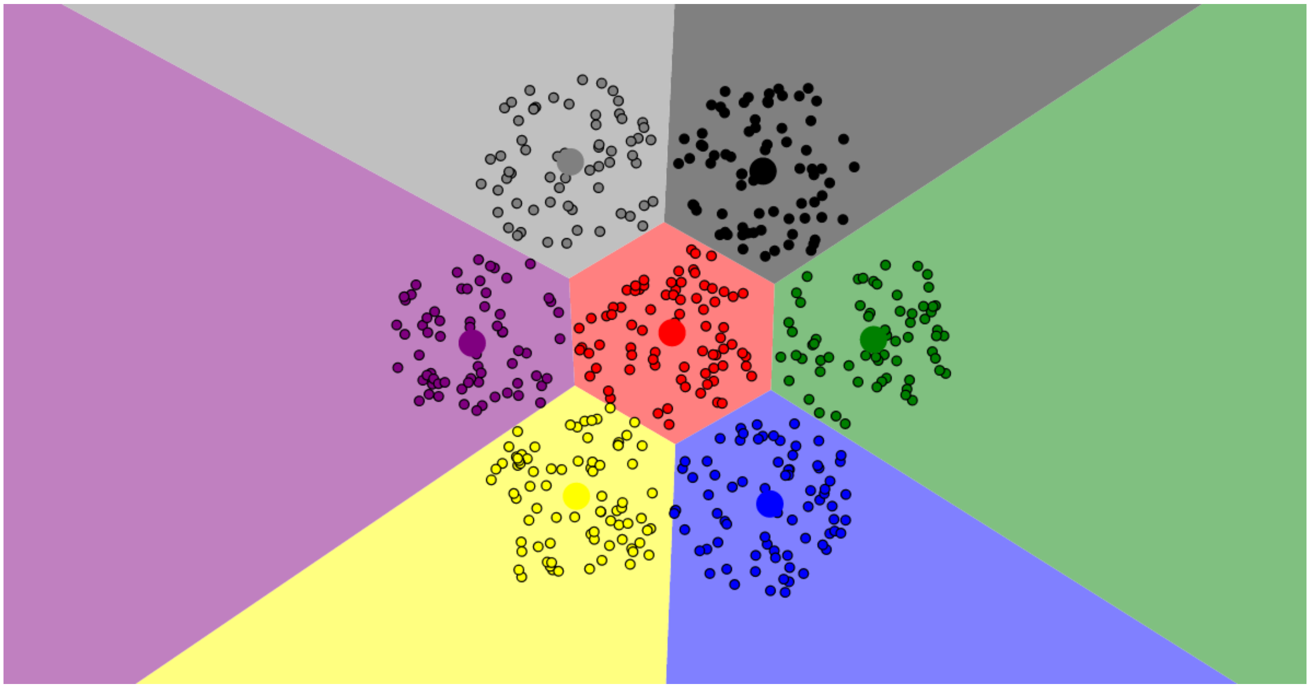
Greedy Permutation

k-median clustering

Local Search



Update Centroids (No change)



Running time

The running time of the algorithm:

$$\mathcal{O}\left((nk)^2 \log_{1/(1-\tau)} \frac{r_1^{\mathcal{C}}(\mathbf{P})}{r_1^{opt}(\mathbf{P}, k)} \right) = \mathcal{O}\big((nk)^2 \log_{1+\tau}(2n) \big)$$

$$= \mathcal{O}\left((nk)^2 \frac{\log n}{\ln(1+\tau)} \right)$$

$$= \mathcal{O}\left((nk)^2 \frac{\log n}{\tau} \right)$$

References

[1] S. Har Paled, *Geometric Approximation Algorithms.* University of Illinois, 2006

[2] Jia Li: K-Center and Dendogram Clustering
`http://sites.stat.psu.edu/ jiali/course/stat597e/notes2/kcen`

[3] Nafthali Harris: Visualizing K-Means Clustering
`https://www.naftaliharris.com/blog/visualizing-k-means-cluste`

[4] Pankaj Agarwal: Clustering
`https://users.cs.duke.edu/ kell/pdf/scribeNotes2.pdf`

[5] Sanjoy Dasgupta: Clustering in Metric Spaces
`https://cseweb.ucsd.edu/ dasgupta/291-geom/kcenter.pdf`

The End

Max diameter clustering

This is very similar to k-center clustering [5].

- Input: A finite set of points $\mathbf{P}$ in some metric space, $(\mathcal{X}, d)$ and an integer k
- Output: A partition of $\mathbf{P}$ into k clusters.
- Goal: Minimize the maximum diameter of the clusters. That is, the cost of a partition $\mathbf{P} = \mathcal{C}_1 \cup \mathcal{C}_2 \cup .. \cup \mathcal{C}_k$ is,

$$max_j \, max_{x,x' \in \mathcal{C}_j} d(x, x')$$