IIIT HYDERABAD

SMAI PROJECT

Presentation

Aryan & Pratham
(2021111018) (2021102036)

Incremental Clustering: The Case for Extra Clusters

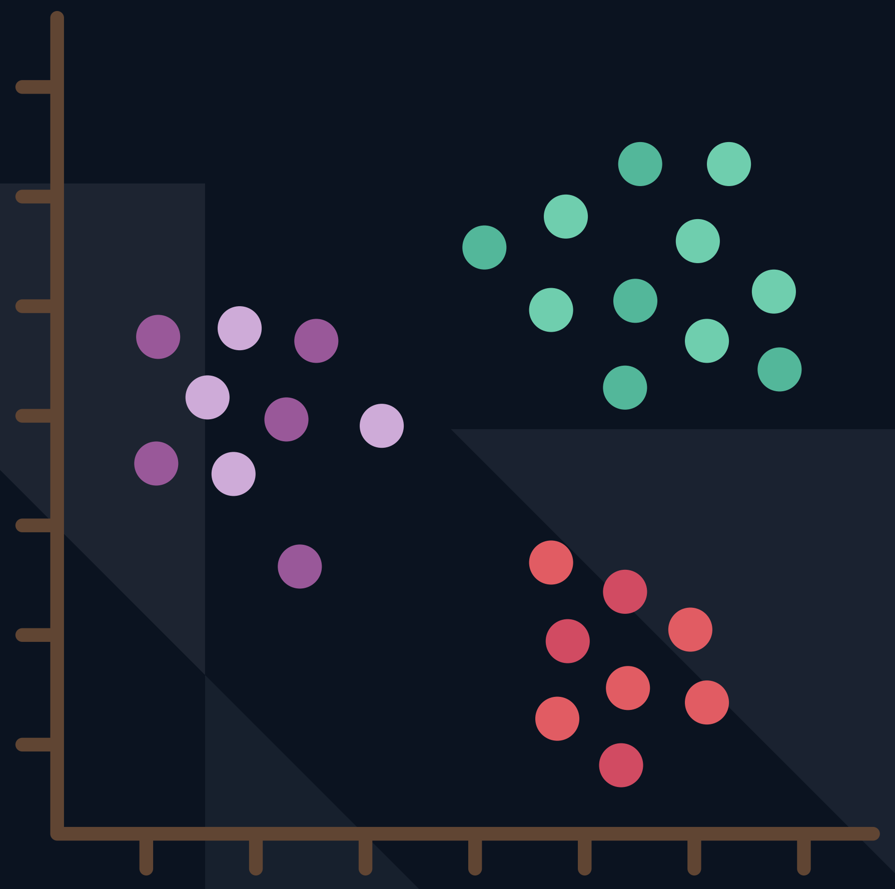
In this project, we were tasked with implementing and creating a blog about the research paper 'Incremental Clustering: The Case for Extra Clusters'. The paper dives into analyzing incremental clustering methods and focuses on the types of cluster structures they can detect. The results show that incremental methods perform worse compared to the batch model. This highlights that certain cluster patterns that are easily spotted using batch methods become hard to identify using incremental methods. Additionally, the paper suggests a solution to the limitations of incremental clustering by proposing the use of extra clusters.

Introduction

The project begins by introducing the varius incremental clustering algorithms.

- **Incremental K-means**
- **Incremental Agglomerative**
- **Incremental Nearest Neighbours**

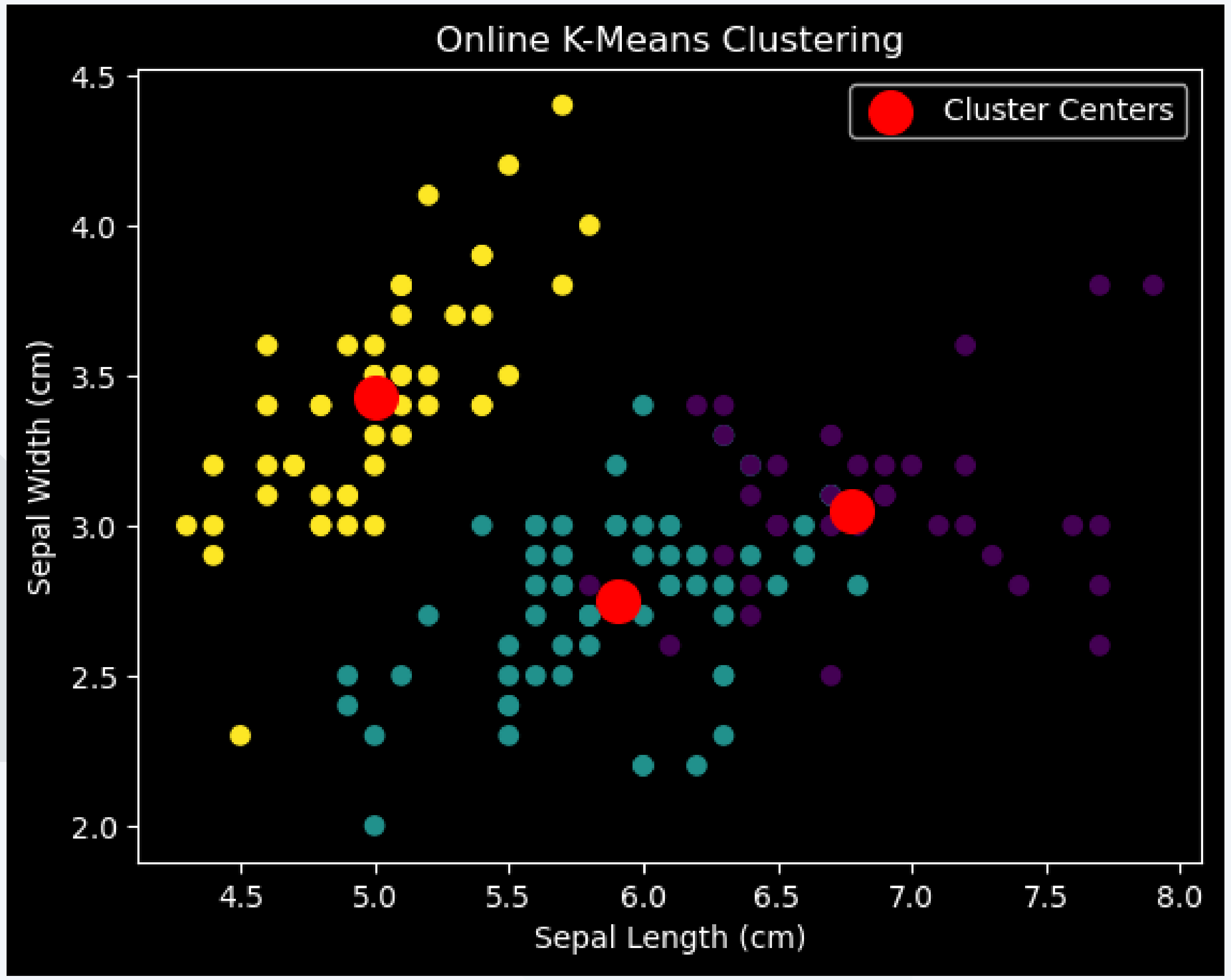
Sequential K- Means

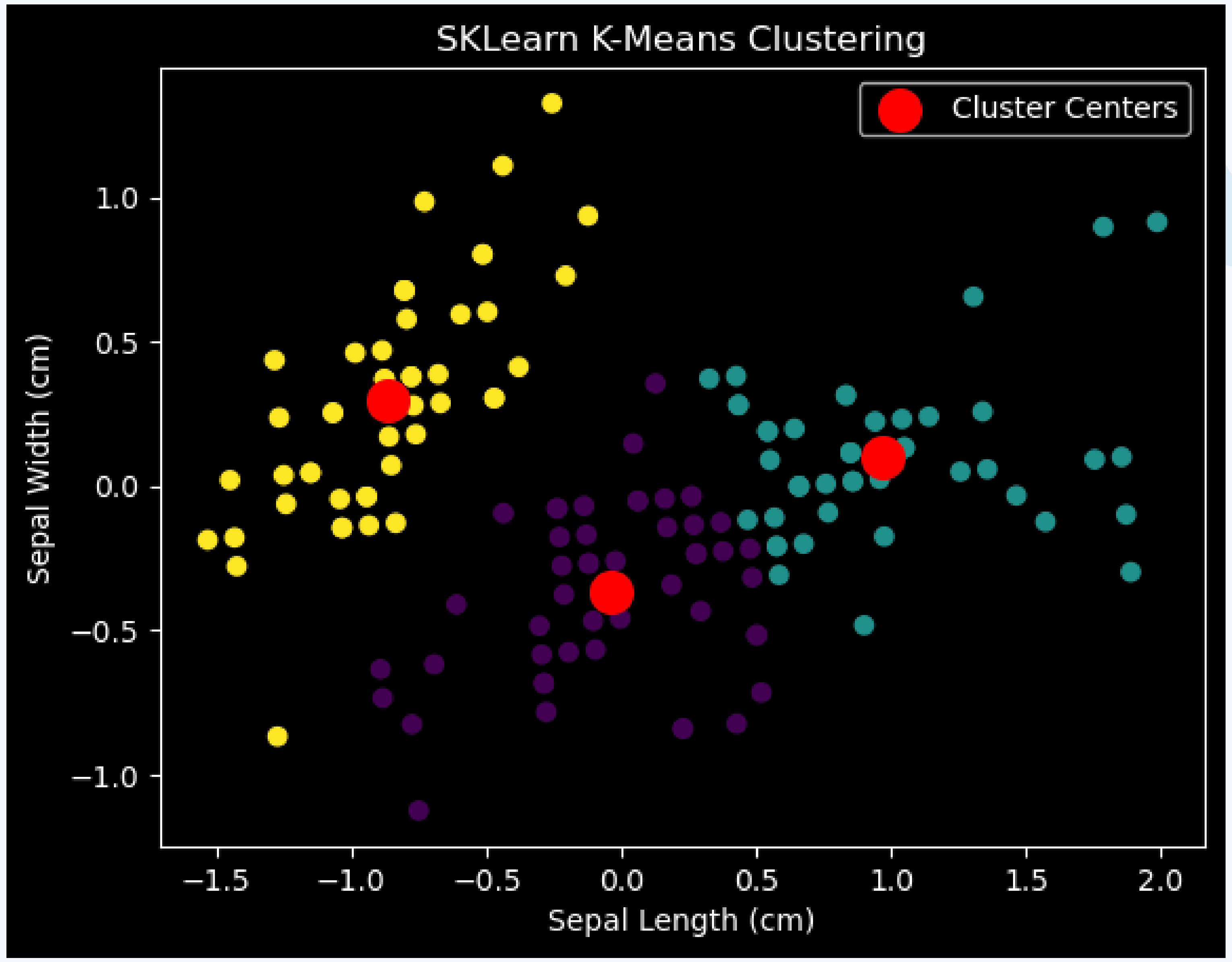


Algorithm 2.2. *Sequential k-means.*

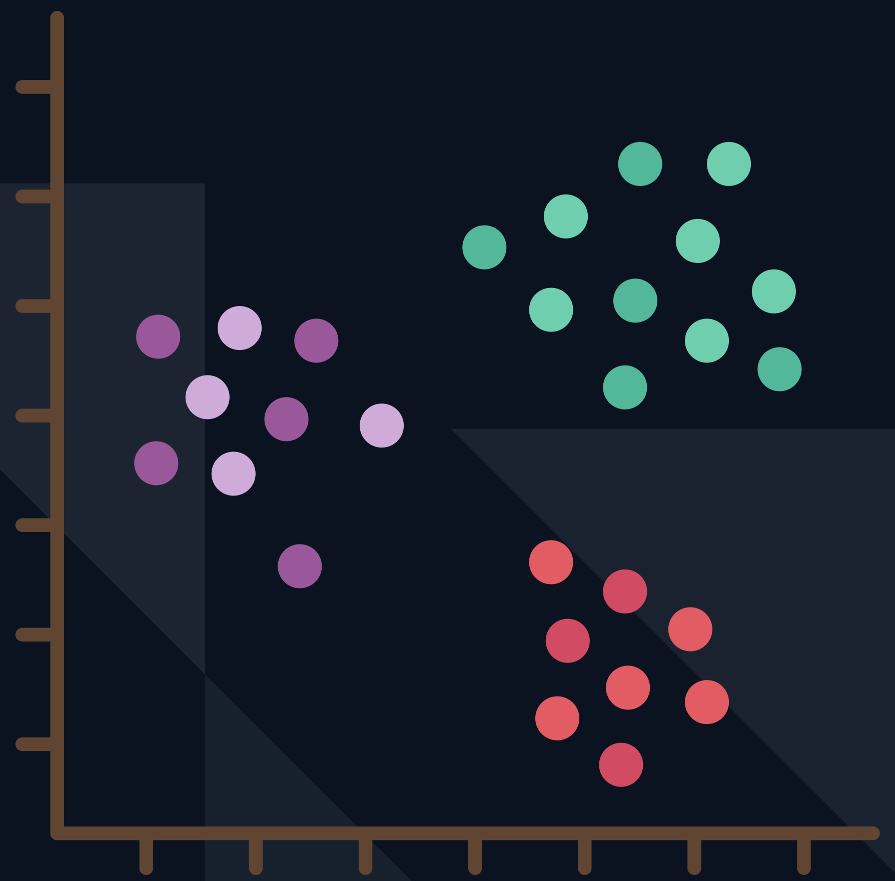
```
Set $T = (t_1, \ldots, t_k)$ to the first $k$ data points
Initialize the counts $n_1, n_2, \ldots, n_k$ to $1$
Repeat:
    Acquire the next example, $x$
    If $t_i$ is the closest center to $x$:
        Increment $n_i$
        Replace $t_i$ by $t_i + (1/n_i)(x - t_i)$
```

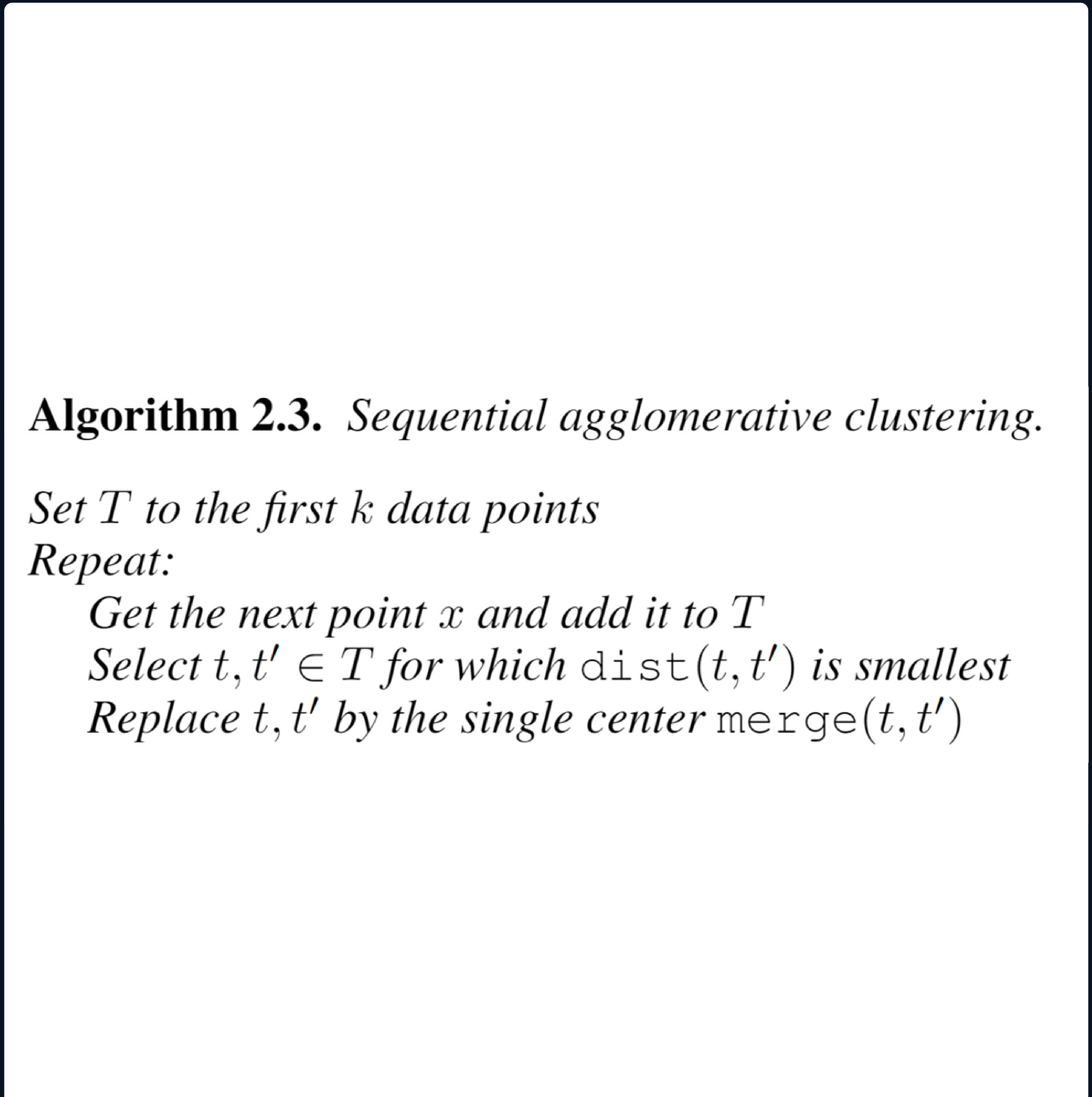
Comparing with batch clustering





Sequential Agglomerative

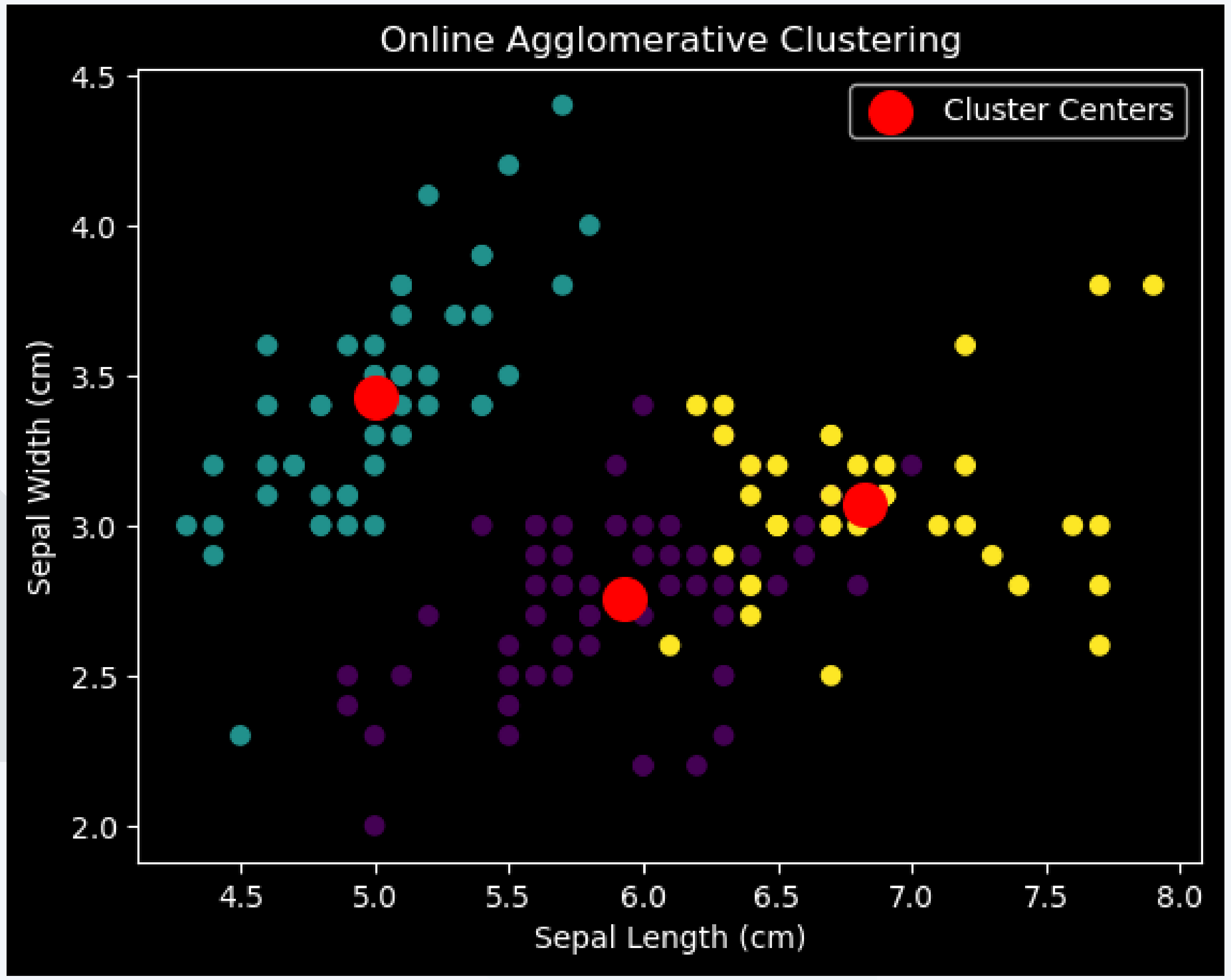


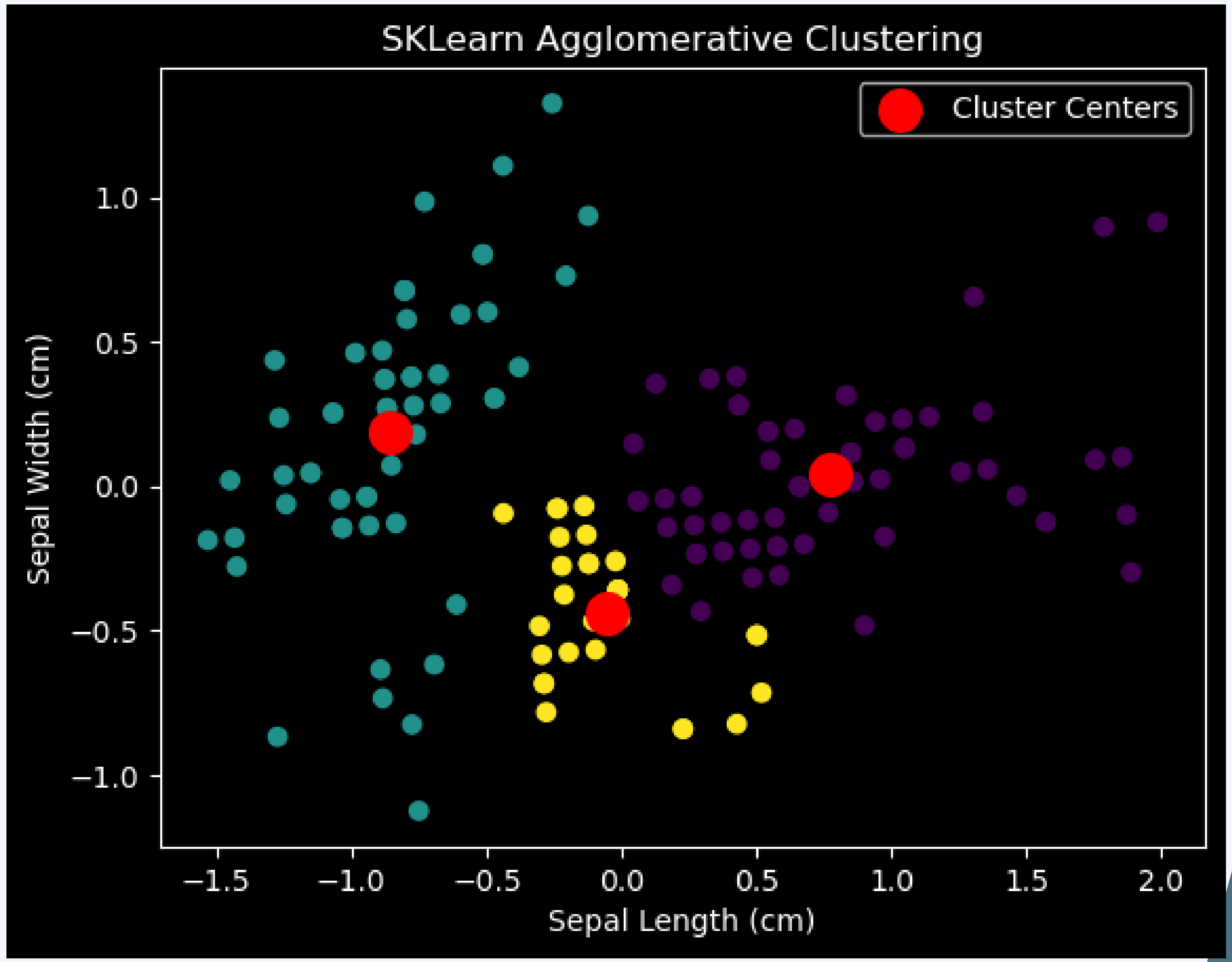


Algorithm 2.3. *Sequential agglomerative clustering.*

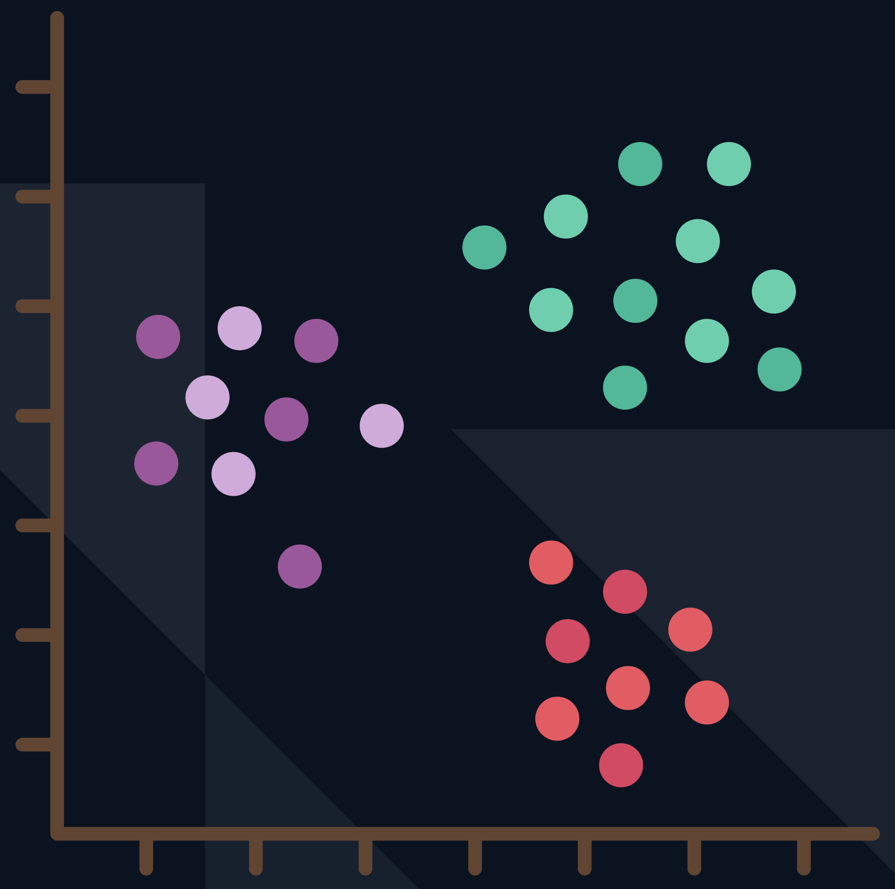
```
Set $T$ to the first $k$ data points
Repeat:
    Get the next point $x$ and add it to $T$
    Select $t, t' \in T$ for which $\texttt{dist}(t, t')$ is smallest
    Replace $t, t'$ by the single center $\texttt{merge}(t, t')$
```

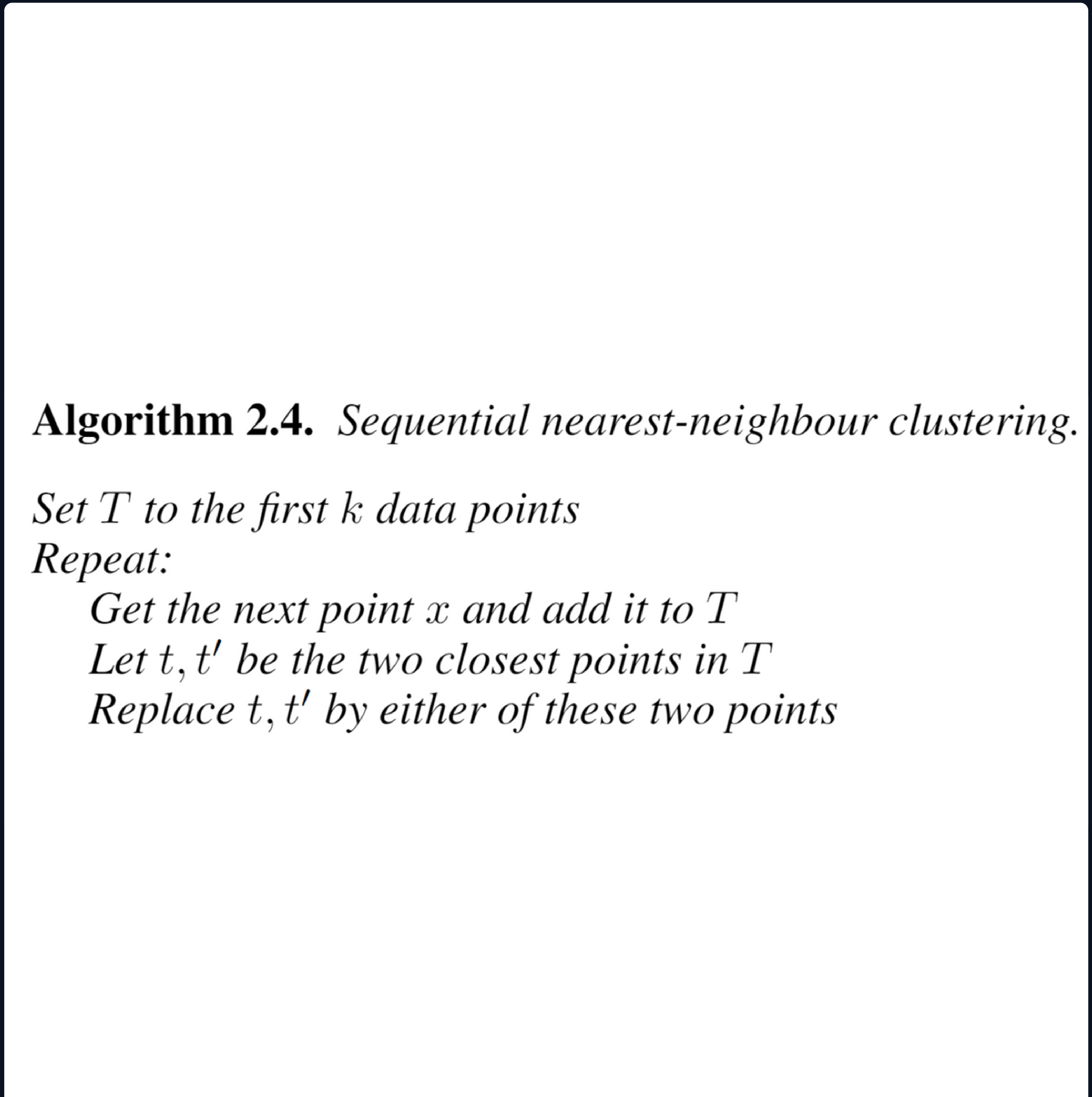
Comparing with batch clustering





Sequential Nearest Neighbours

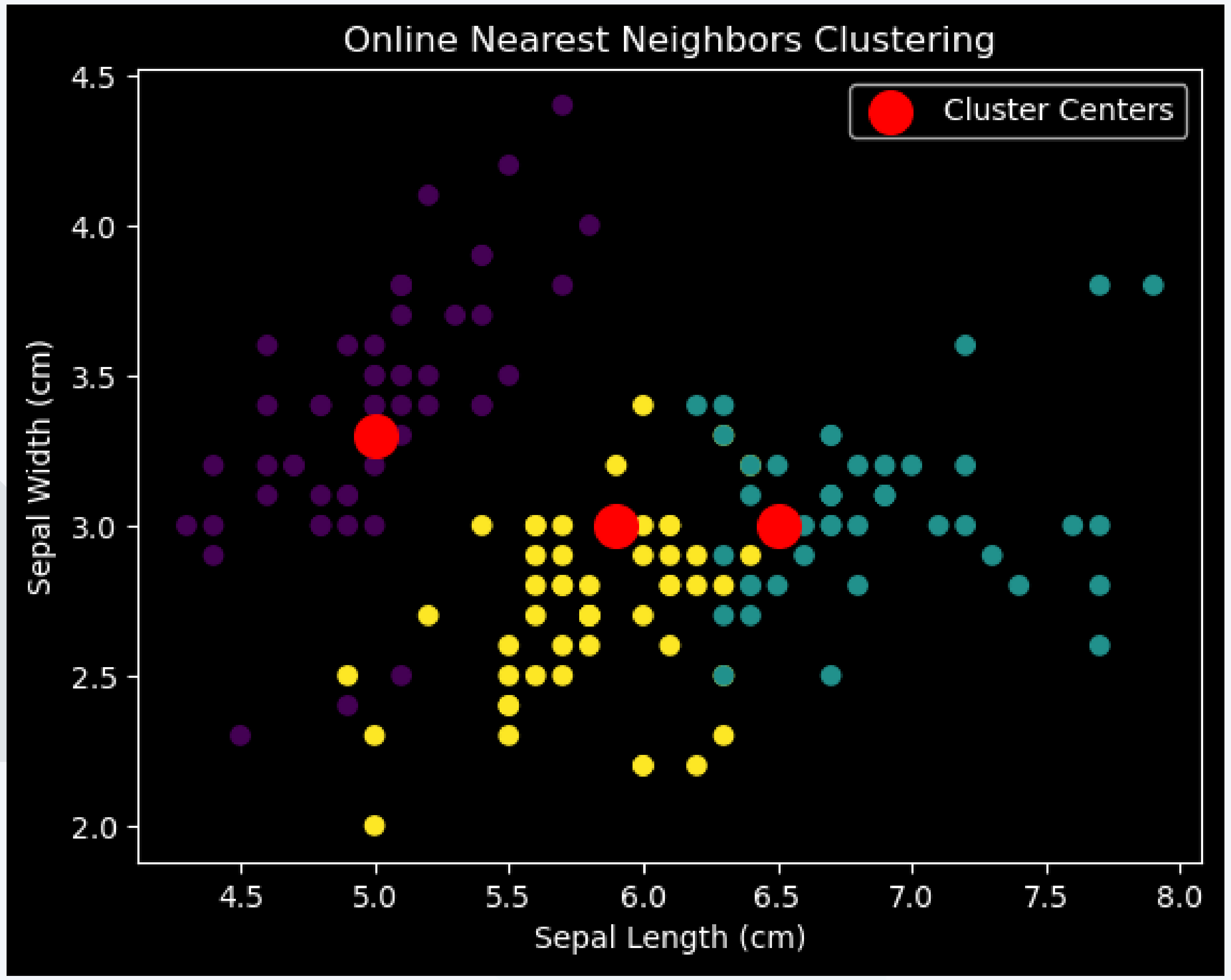


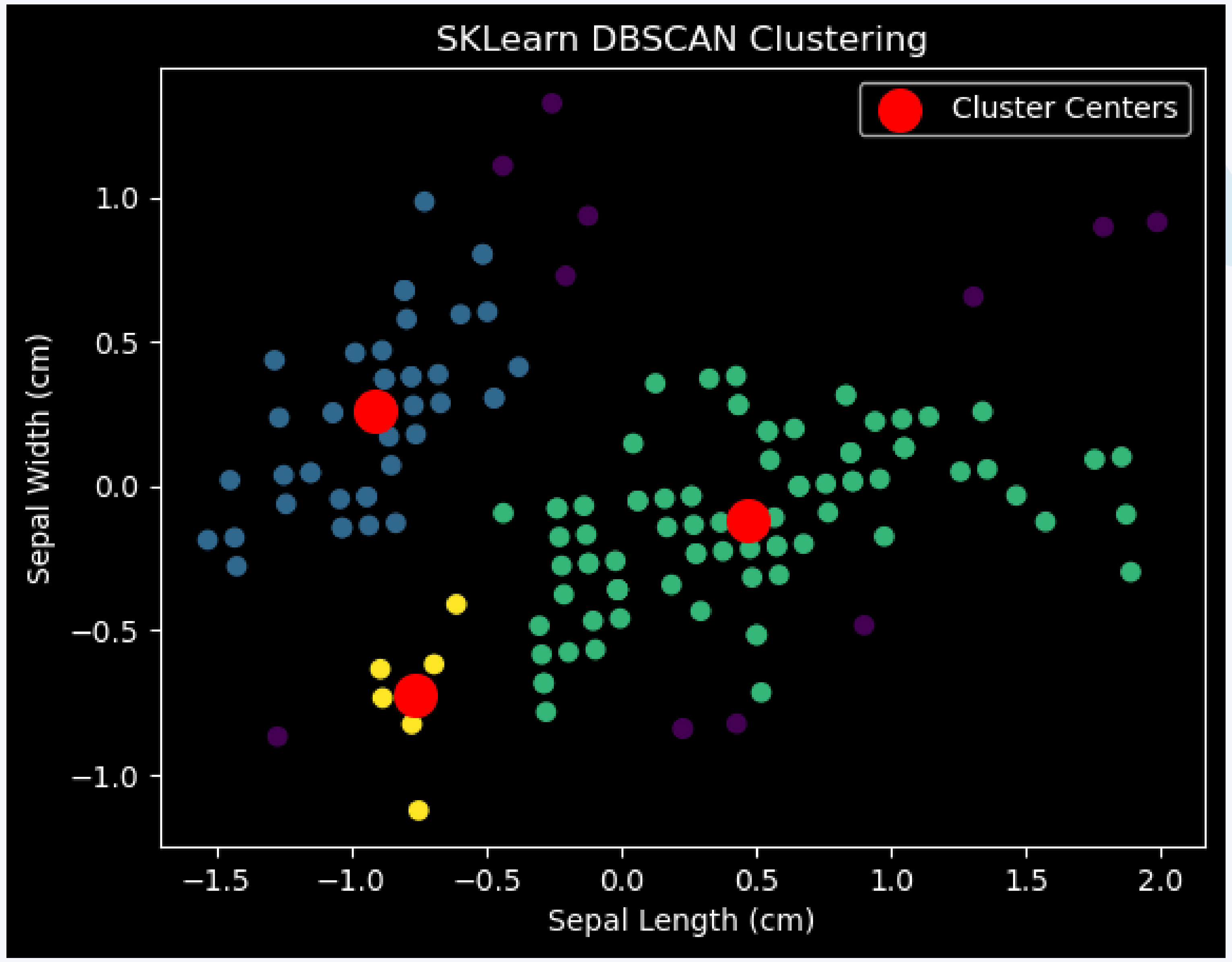


Algorithm 2.4. *Sequential nearest-neighbour clustering.*

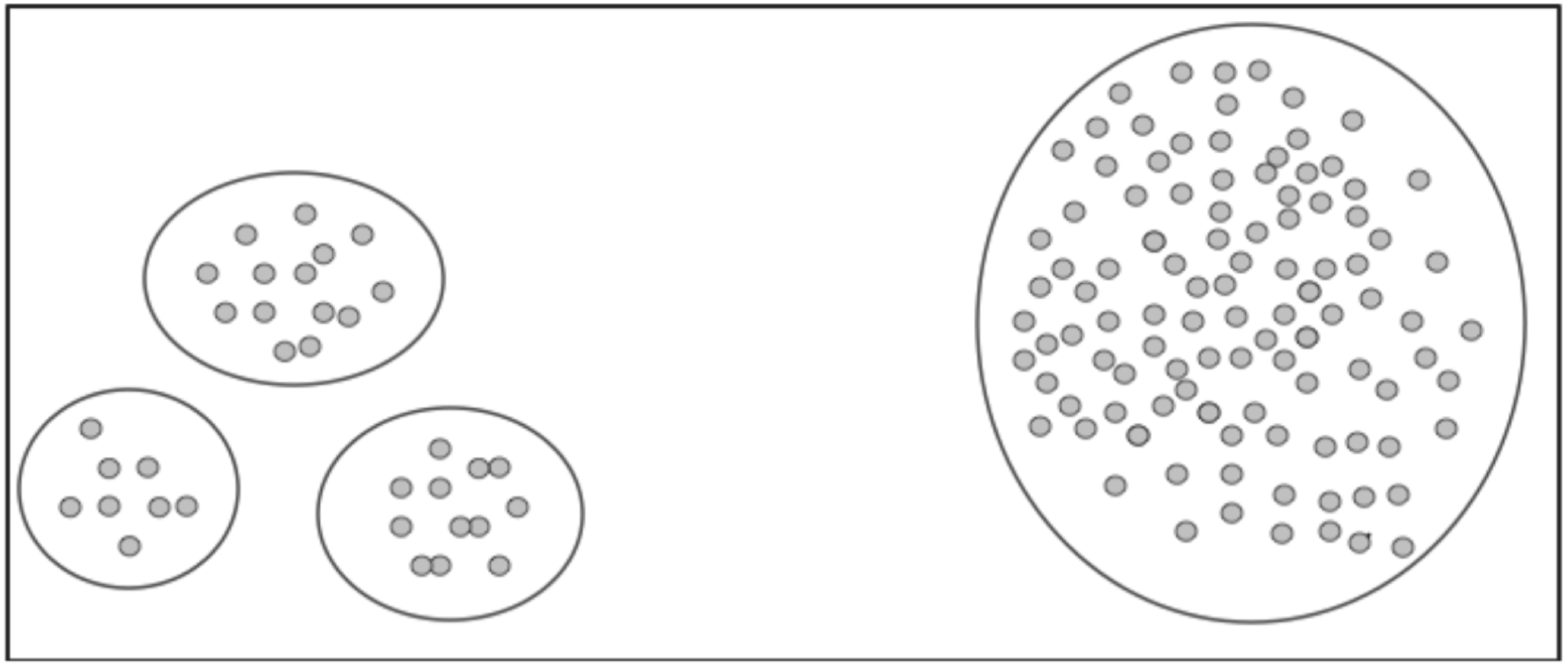
```
Set $T$ to the first $k$ data points
Repeat:
    Get the next point $x$ and add it to $T$
    Let $t, t'$ be the two closest points in $T$
    Replace $t, t'$ by either of these two points
```

Comparing with batch clustering





Nice Clustering:



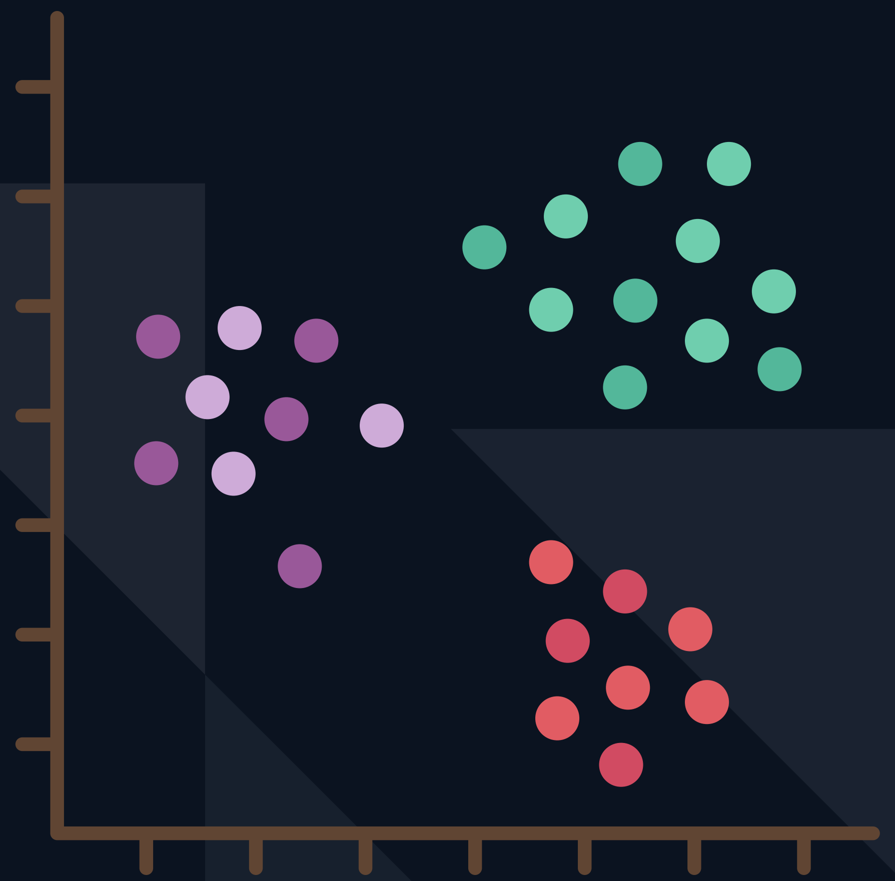
Definition 3.1 (Nice clustering). *A clustering $\mathcal{C}$ of $(\mathcal{X}, d)$ is* **nice** *if for all $x, y, z \in \mathcal{X}$, $d(y,x) < d(z,x)$ whenever $x \sim_{\mathcal{C}} y$ and $x \not\sim_{\mathcal{C}} z$.*

The Uniqueness of Nice k-Clustering

Observation 3.2. *If we select one point from every cluster of a nice clustering $\mathcal{C}$, the resulting set induces $\mathcal{C}$. (Moreover, niceness is the minimal property under which this holds.)*

A nice k-clustering is not, in general, unique. For example, consider $\mathcal{X} = \{1, 2, 4, 5\}$ on the real line under the usual distance metric; then both $\{\{1\}, \{2\}, \{4, 5\}\}$ and $\{\{1, 2\}, \{4\}, \{5\}\}$ are nice 3-clusterings of $\mathcal{X}$. Thus we start by considering data with a *unique* nice k-clustering.

Nice-Detecting Incremental Clustering Algorithm



Definition

An incremental clustering algorithm **A** is nice-detecting if, given a positive integer k and (**X** , **d**) that has a unique nice k-clustering **C**, the procedure **A**(**O**[**X**], **d**, **k**) outputs C for any ordering function **O**.

Definition : M -configuration

Definition 3.4. *In any metric space* $(\mathcal{X}, d)$*, for any integer* $M > 0$*, define an* M-configuration *to be a collection of* $2M + 1$ *points* $x_o, x_1, \ldots, x_M, x'_1, \ldots, x'_M \in \mathcal{X}$ *such that*

- *All interpoint distances are in the range* $[1, 2]$.
- $d(x_o, x_i), d(x_o, x'_i) \in (3/2, 2]$ *for all* $i \geq 1$.
- $d(x_i, x_j), d(x'_i, x'_j), d(x_i, x'_j) \in [1, 3/2]$ *for all* $i \neq j \geq 1$.
- $d(x_i, x'_i) > d(x_o, x_i)$.

Lemma 3.5. *Let $x_o, x_1, \ldots, x_M, x'_1, \ldots, x'_M$ be any M-configuration in $(\mathcal{X}, d)$. Pick any index $1 \leq j \leq M$ and any subset $S \subset [M]$ with $|S| > 1$. Then the set $A = \{x_o, x'_j\} \cup \{x_i : i \in S\}$ has a nice 2-clustering if and only if $j \notin S$.*

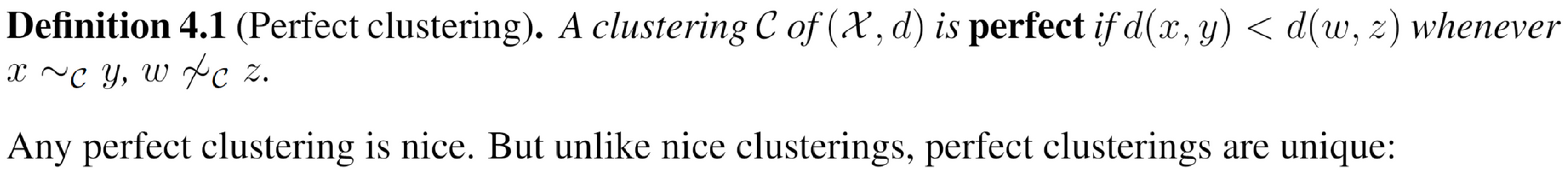
Theorem 3.6. *Let $(\mathcal{X}, d)$ be any metric space that contains two M-configurations separated by a distance of at least 4. Then, there is no deterministic incremental algorithm with $\leq M/2$ bits of storage that is guaranteed to recover nice 3-clusterings of data sets drawn from $\mathcal{X}$, even when limited to instances in which such clusterings are unique.*

Lemma 3.7. *There is an absolute constant c_o such that for any dimension p, the Euclidean space $\mathbb{R}^p$, with L_2 norm, contains M-configurations for all $M < 2^{c_o p}$.*

The overall conclusions are the following.

Theorem 3.8. *There is no memory-bounded deterministic nice-detecting incremental clustering algorithm that works in arbitrary metric spaces. For data in $\mathbb{R}^p$ under the ℓ_2 metric, there is no deterministic nice-detecting incremental clustering algorithm using less than $2^{c_o p - 1}$ bits of memory.*

Perfect Clustering:



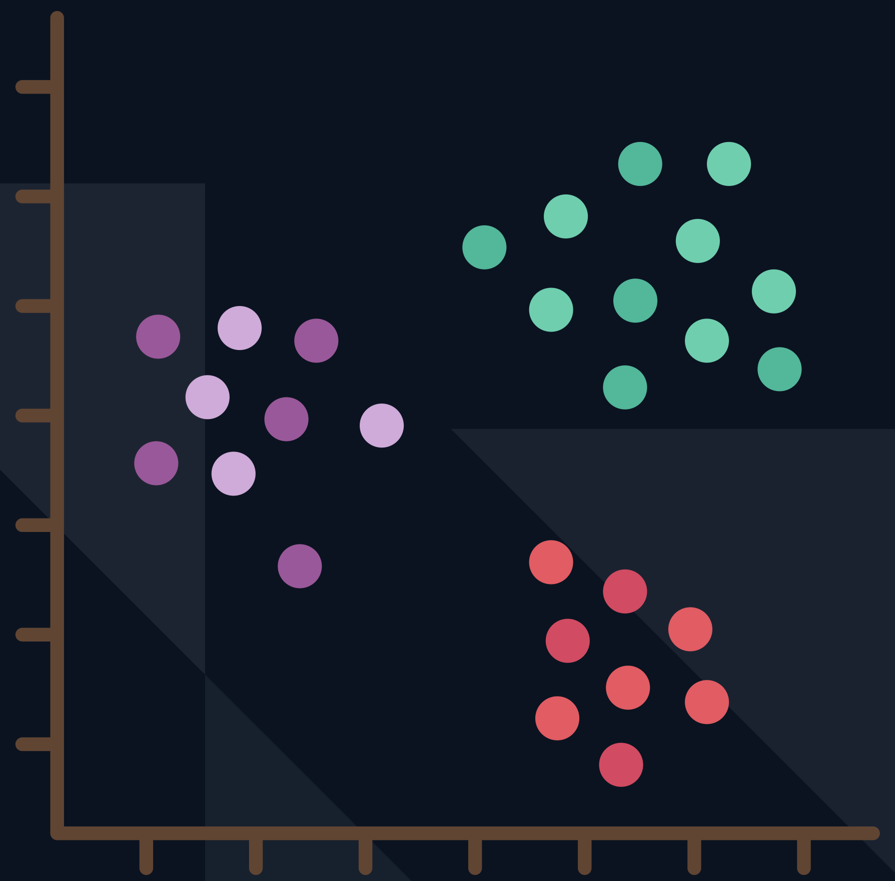
Definition 4.1 (Perfect clustering)**.** *A clustering* $\mathcal{C}$ *of* $(\mathcal{X}, d)$ *is* **perfect** *if* $d(x, y) < d(w, z)$ *whenever* $x \sim_{\mathcal{C}} y$, $w \not\sim_{\mathcal{C}} z$.

Any perfect clustering is nice. But unlike nice clusterings, perfect clusterings are unique:

To define a perfect clustering, we consider a clustering C of a given set (X, d). This clustering is deemed "perfect" if, for any pair of points x and y within the same cluster and another pair w and z in different clusters, the distance between x and y is strictly less than the distance between w and z.

A remarkable feature of perfect clusterings is their uniqueness. For any given set (X, d) and positive integer k, there exists at most one perfect k-clustering of (X, d).

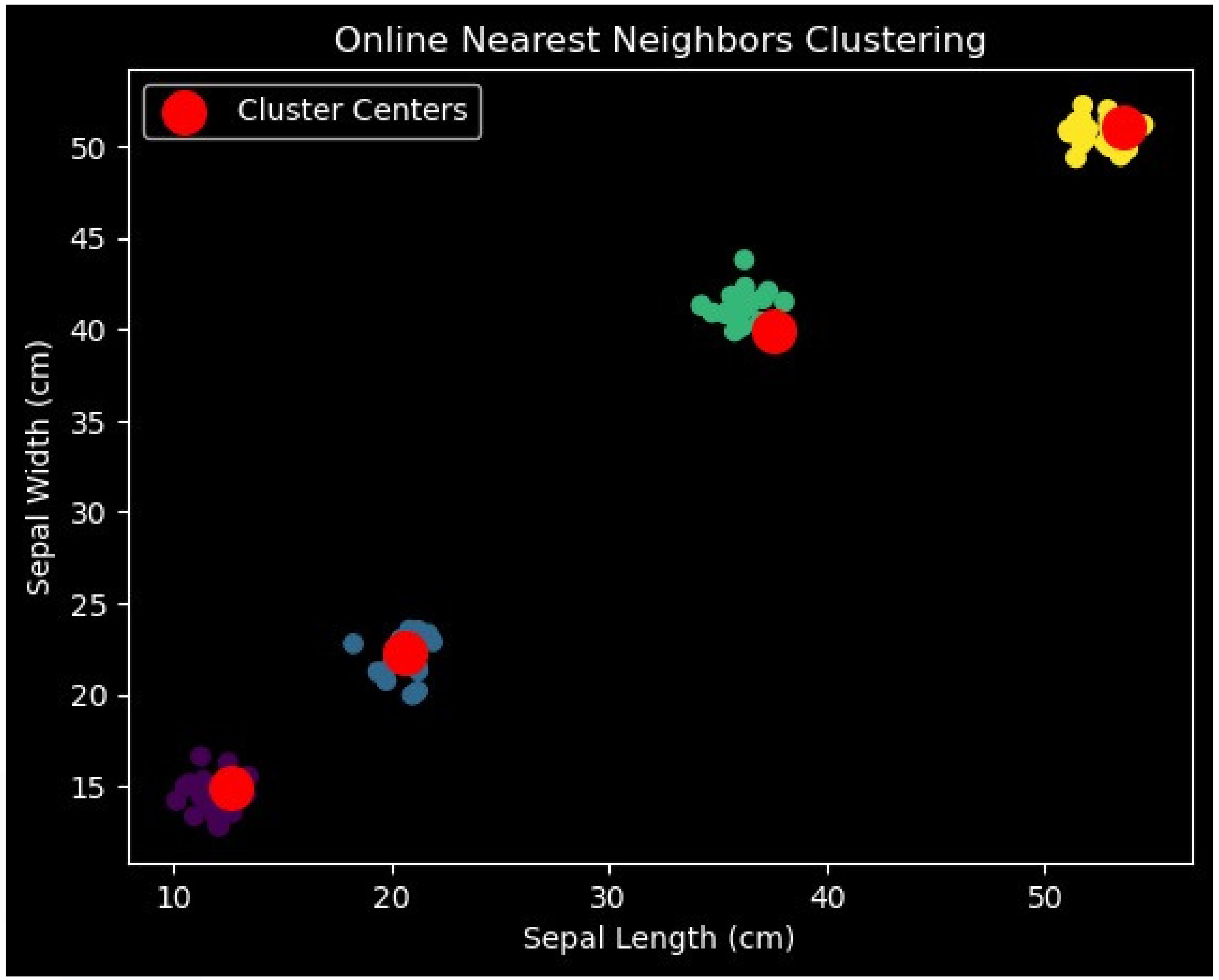
Perfect-Detecting Incremental Algorithms



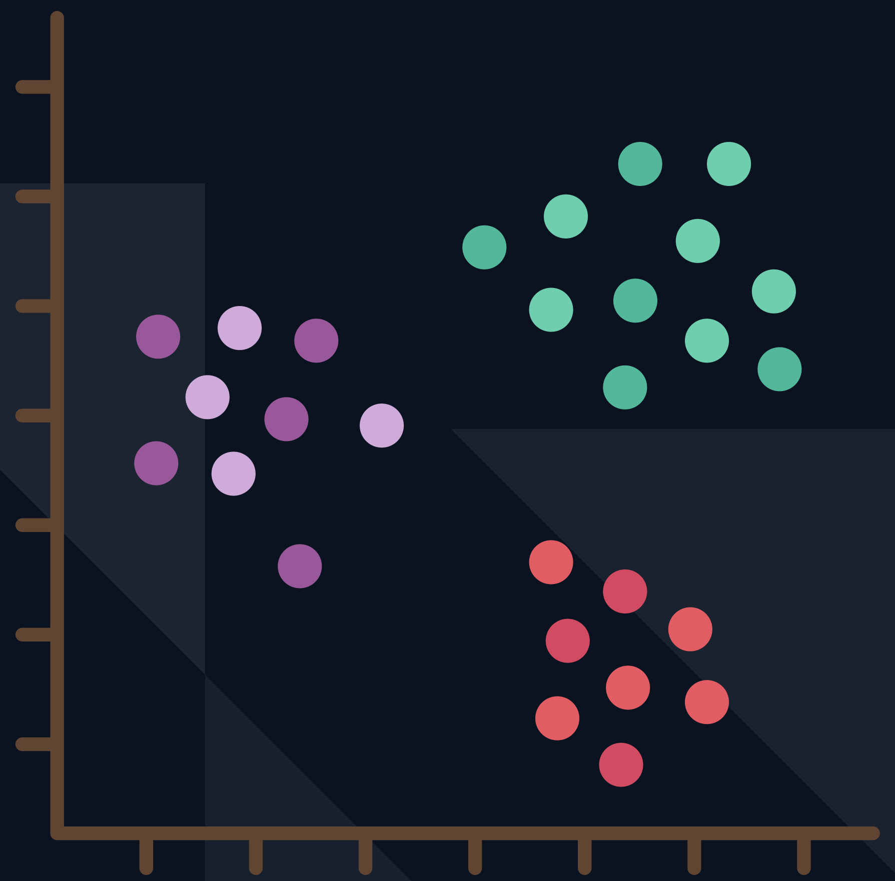
Definition

An incremental clustering algorithm **A** is perfect-detecting if, given a positive integer k and (**X** , **d**) that has a unique perfect k-clustering **C**, the procedure **A**(**O**[**X**], **d**, **k**) outputs **C** for any ordering function **O**.

Theorem 4.3. *Sequential nearest-neighbour clustering (Algorithm 2.4) is perfect-detecting.*



Refinement



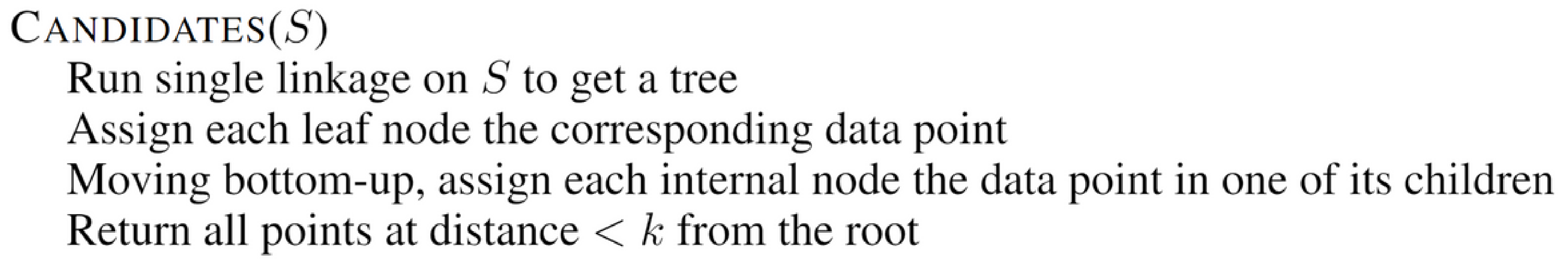
Definition

Instead of aiming to exactly discover the target partition, it is sufficient in some applications to merely uncover a refinement of it. Formally, a

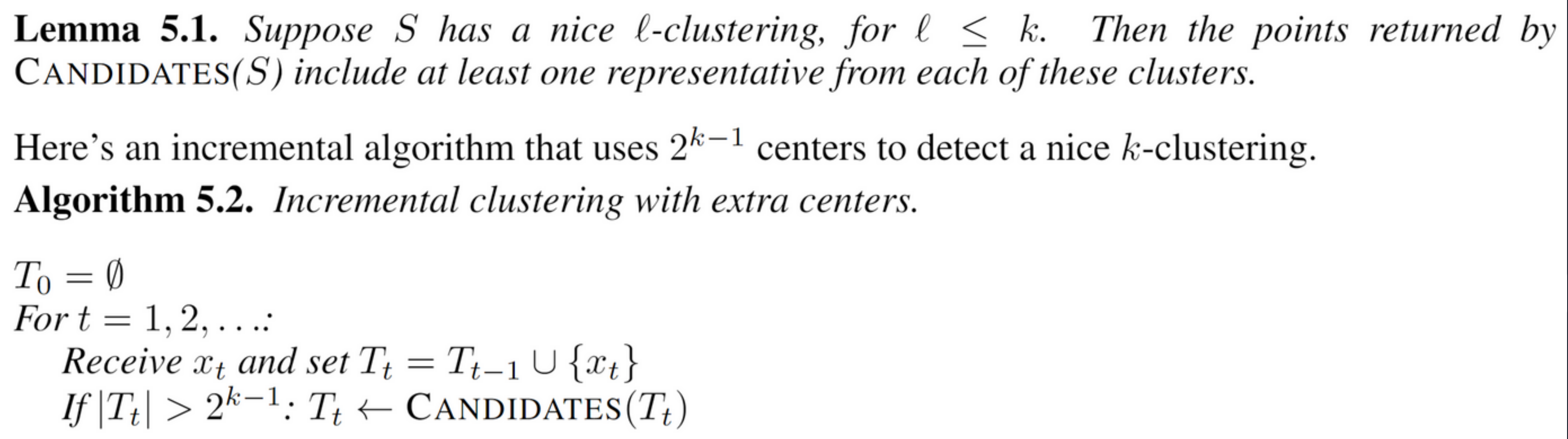
clustering C of X is a refinement of clustering C 0 of X , if x C y implies x C 0 y for all x, y ∈ X .

Detecting nice k-clusters

Algorithm



```
CANDIDATES($S$)
    Run single linkage on $S$ to get a tree
    Assign each leaf node the corresponding data point
    Moving bottom-up, assign each internal node the data point in one of its children
    Return all points at distance $< k$ from the root
```



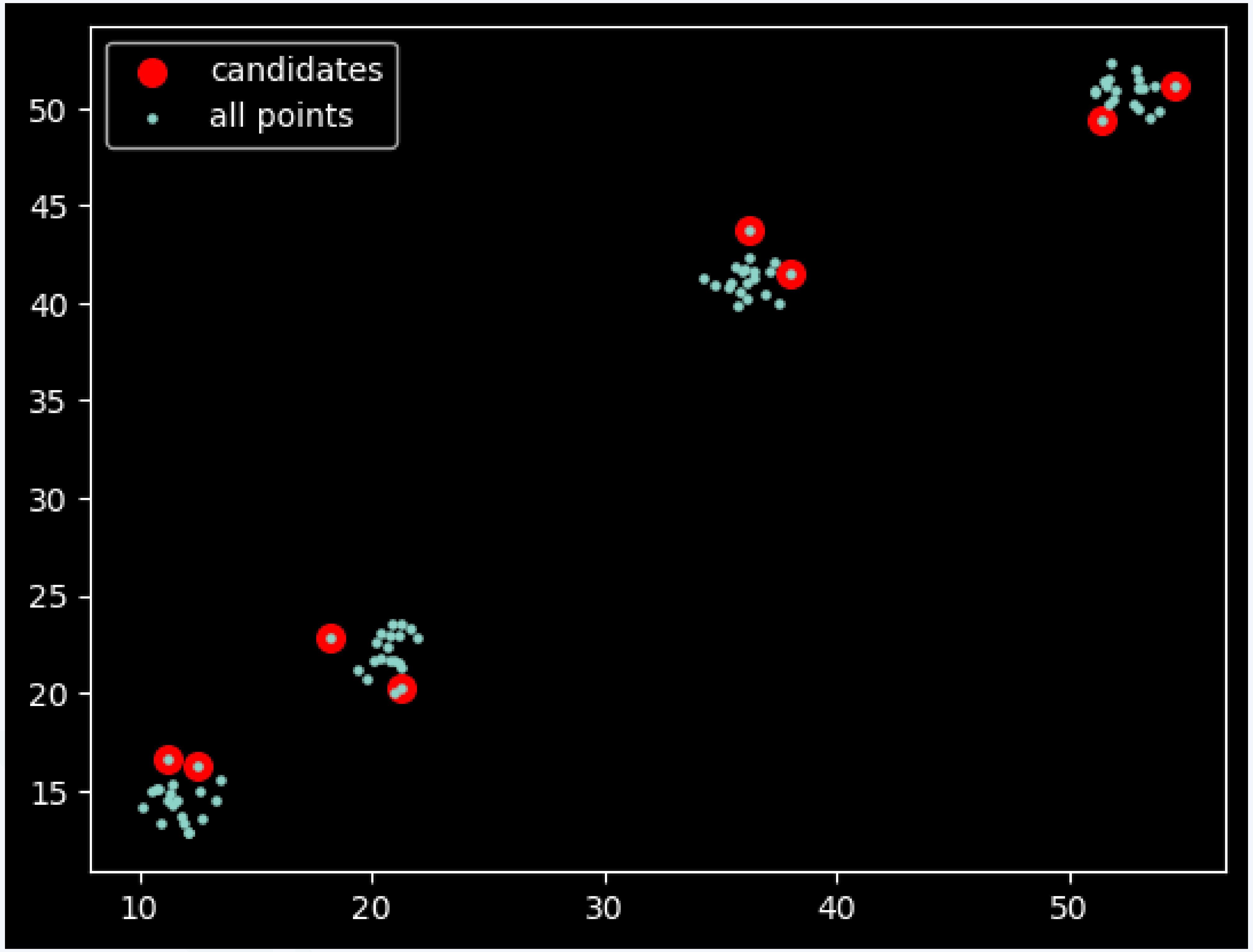
Lemma 5.1. *Suppose S has a nice ℓ-clustering, for $\ell \leq k$. Then the points returned by* CANDIDATES*(S) include at least one representative from each of these clusters.*

Here's an incremental algorithm that uses 2^{k-1} centers to detect a nice k-clustering.

Algorithm 5.2. *Incremental clustering with extra centers.*

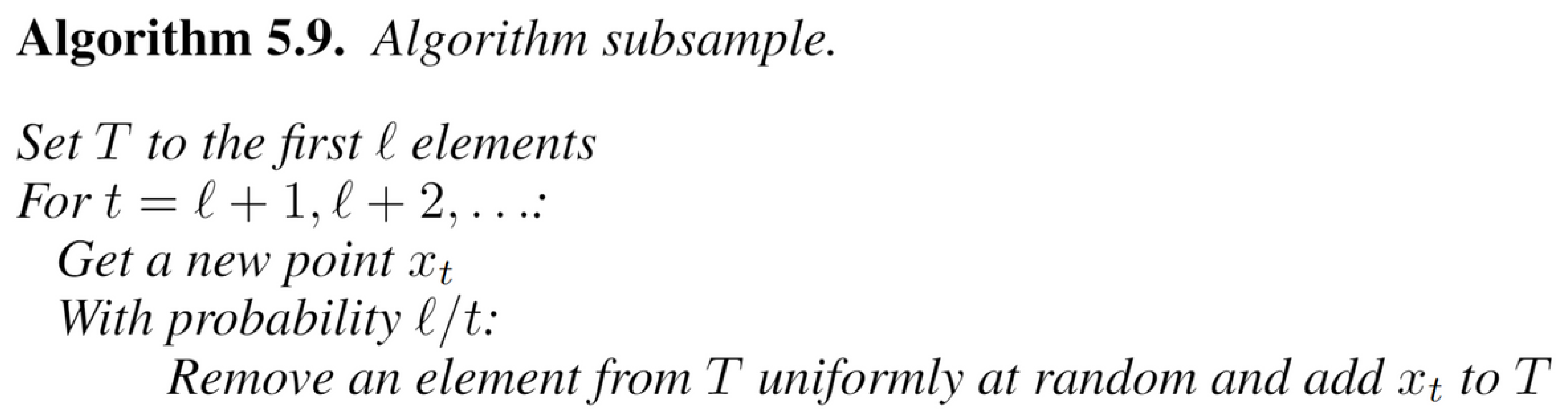
```
$T_0 = \emptyset$
For $t = 1, 2, \ldots$:
    Receive $x_t$ and set $T_t = T_{t-1} \cup \{x_t\}$
    If $|T_t| > 2^{k-1}$: $T_t \leftarrow$ CANDIDATES$(T_t)$
```

Candidates vs Points



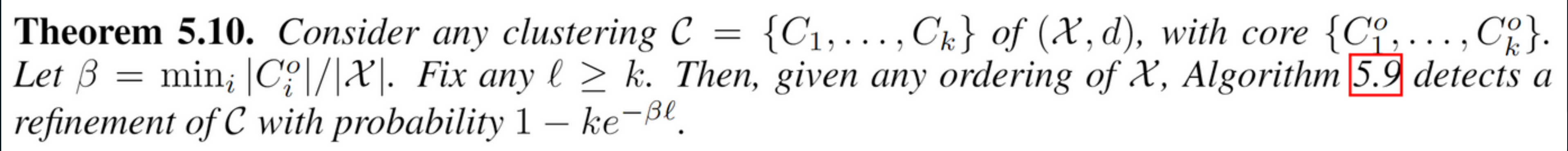
Detecting Refined Clusters

Algorithm



Algorithm 5.9. *Algorithm subsample.*

```
Set $T$ to the first $\ell$ elements
For $t = \ell + 1, \ell + 2, \ldots$:
    Get a new point $x_t$
    With probability $\ell / t$:
            Remove an element from $T$ uniformly at random and add $x_t$ to $T$
```



Theorem 5.10. *Consider any clustering $\mathcal{C} = \{C_1, \ldots, C_k\}$ of $(\mathcal{X}, d)$, with core $\{C_1^o, \ldots, C_k^o\}$. Let $\beta = \min_i |C_i^o| / |\mathcal{X}|$. Fix any $\ell \geq k$. Then, given any ordering of $\mathcal{X}$, Algorithm $\boxed{5.9}$ detects a refinement of $\mathcal{C}$ with probability $1 - ke^{-\beta \ell}$.*

Thank You

Blog Link