MATLAB EXPO

November 13–14, 2024 | Online

The CLASSIX Story

Developing the same algorithm in MATLAB and Python simultaneously

Stefan Güttel

University of Manchester

Mike Croucher

MathWorks

MathWorks

CLASSIX

Fast and Explainable Clustering

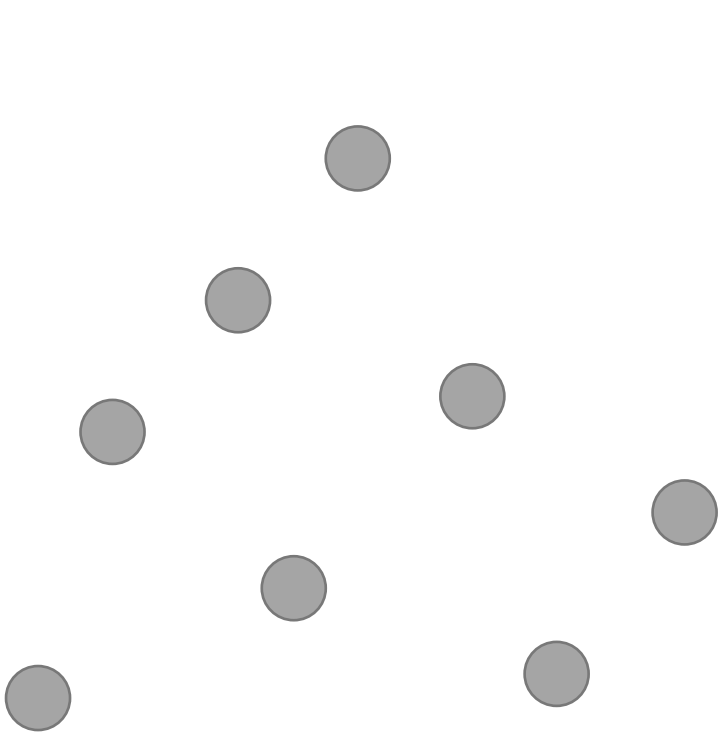
A data clustering method that is

- fast
- memory efficient
- conceptually very simple
- non-iterative and 100% deterministic
- easy to tune with two hyperparameters
- computing fully explainable clustering results

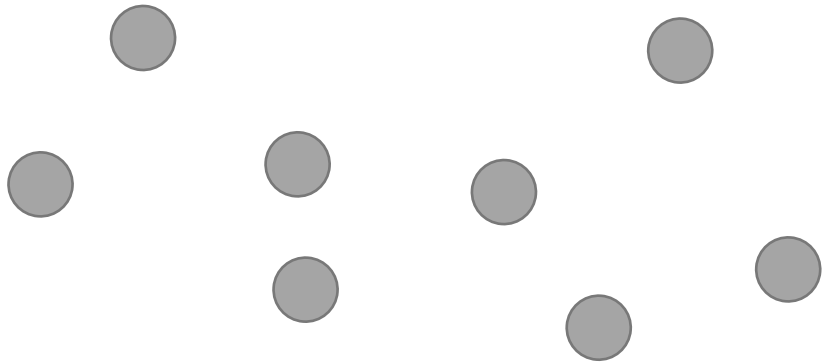
Currently two implementations

- Python https://github.com/nla-group/classix
- MATLAB https://github.com/nla-group/classix-matlab

CLASSIX
Fast and Explainable Clustering

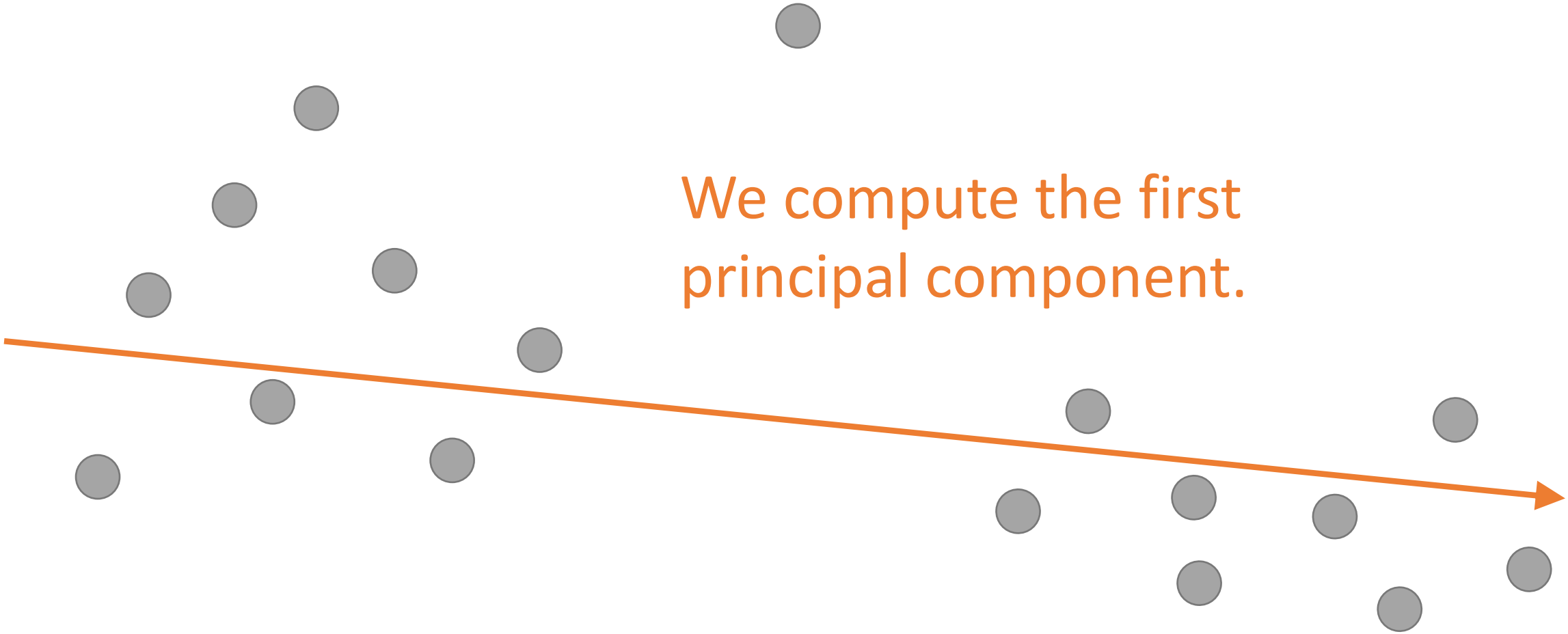


Here are some data points we want to cluster.

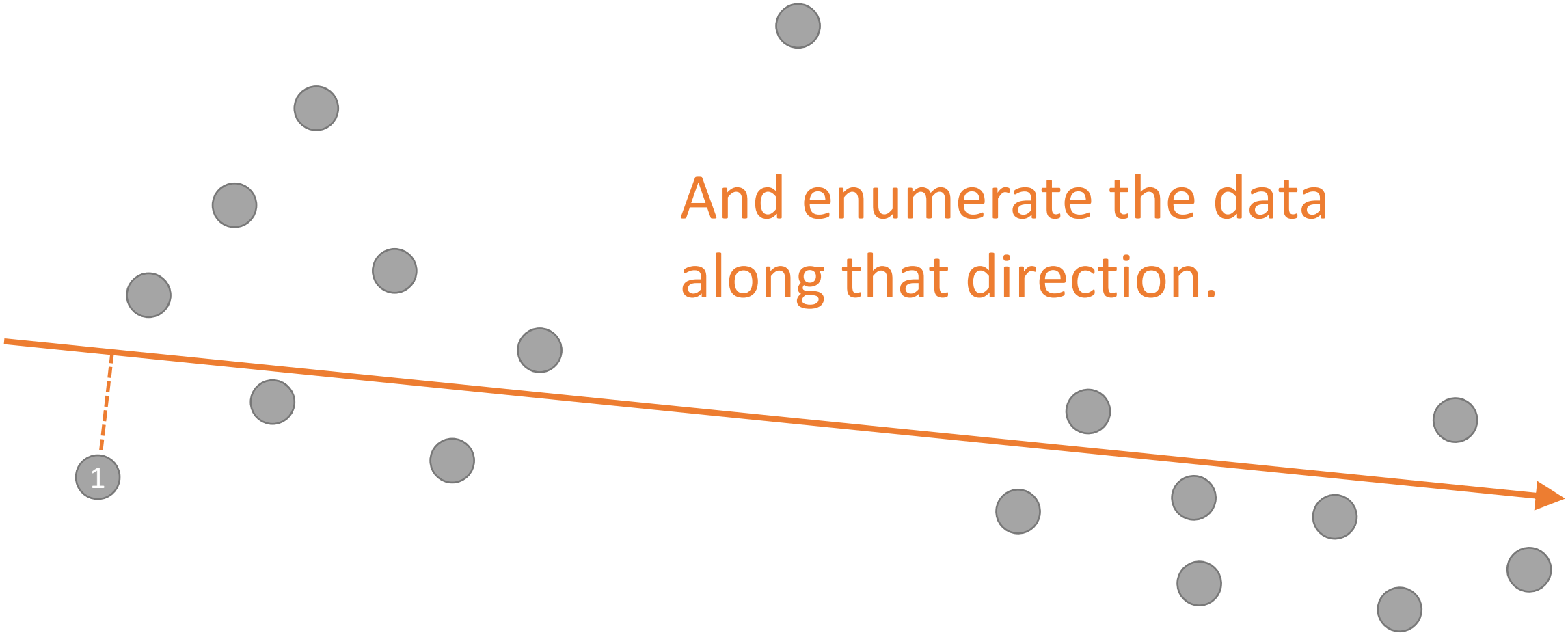


CLASSIX

Fast and Explainable Clustering

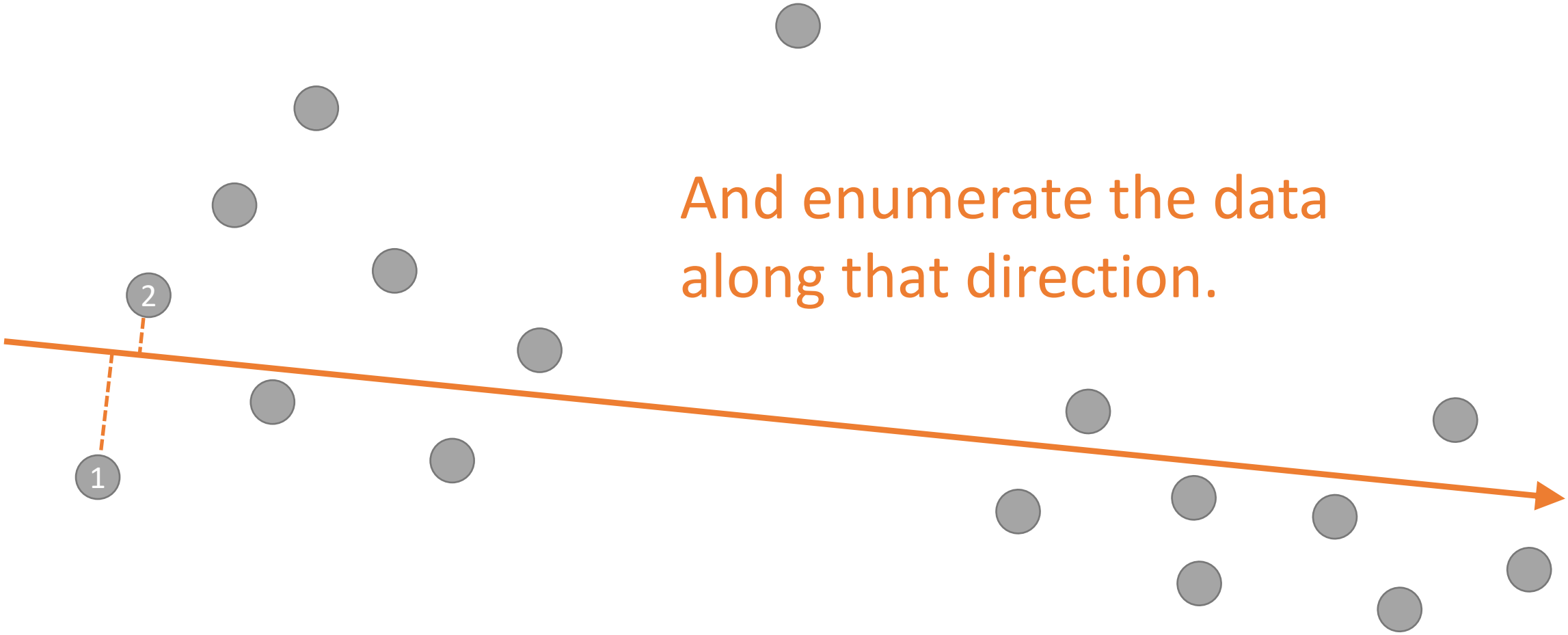


CLASSIX
Fast and Explainable Clustering



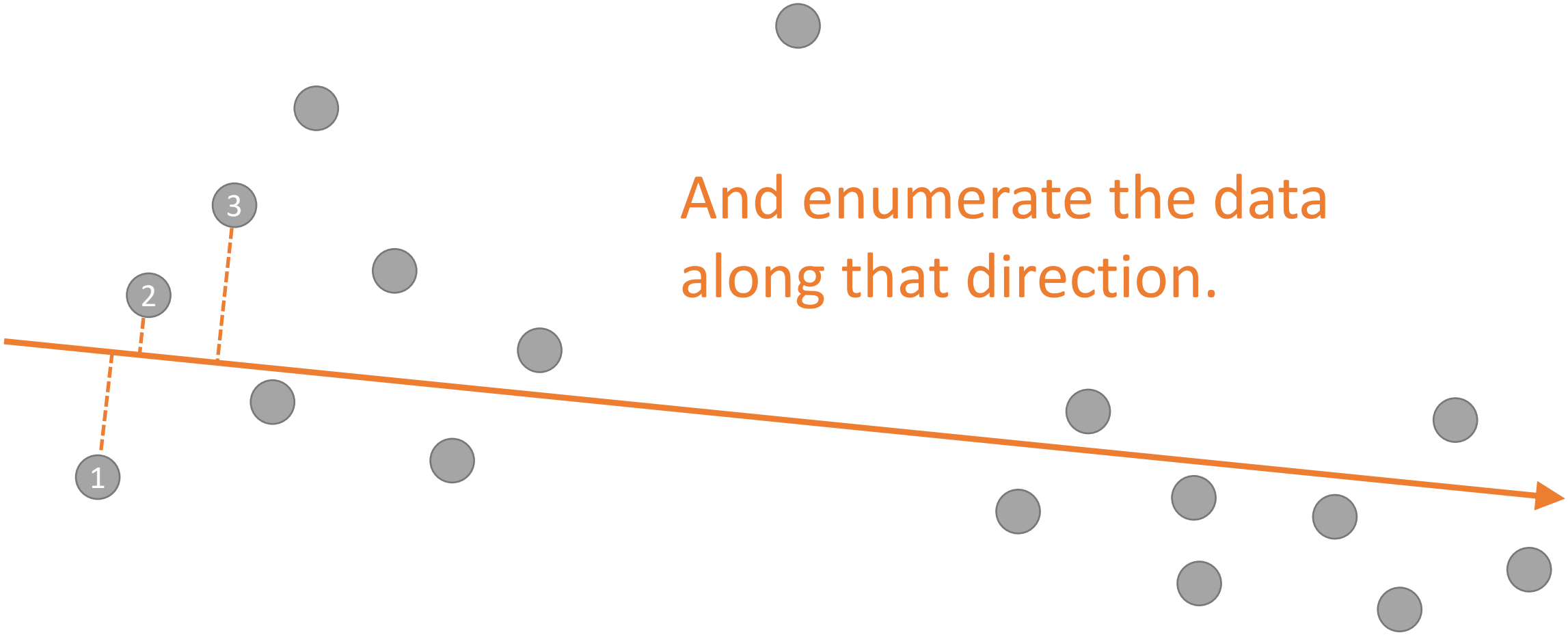
And enumerate the data along that direction.

CLASSIX
Fast and Explainable Clustering



And enumerate the data along that direction.

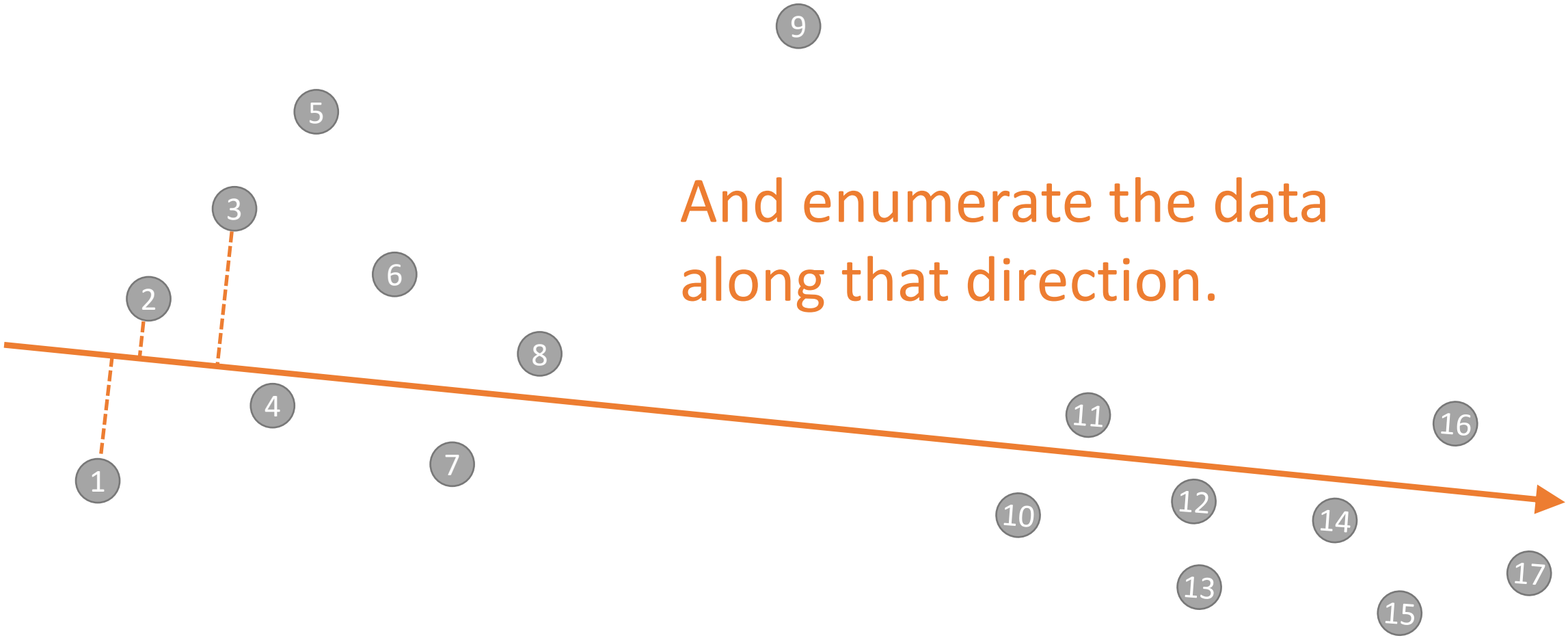
CLASSIX
Fast and Explainable Clustering



And enumerate the data along that direction.

CLASSIX

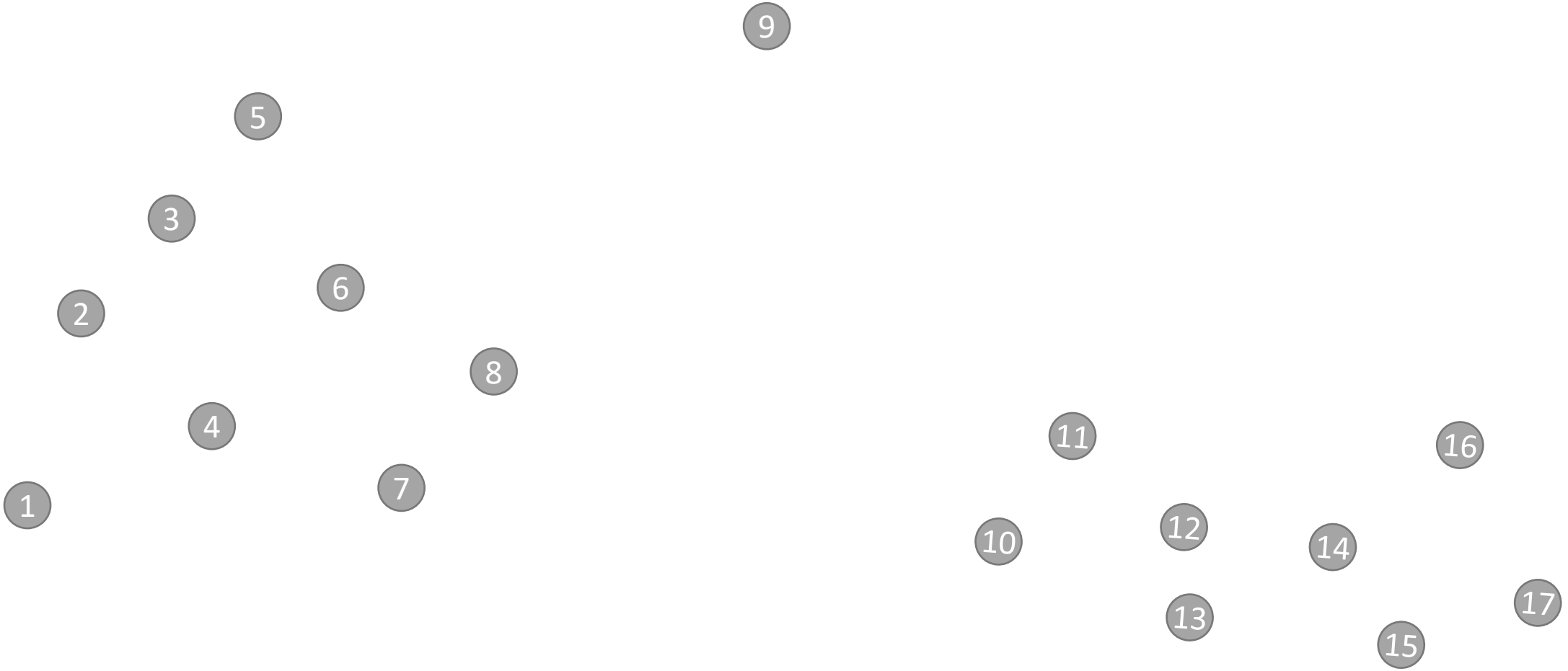
Fast and Explainable Clustering



And enumerate the data along that direction.

CLASSIX

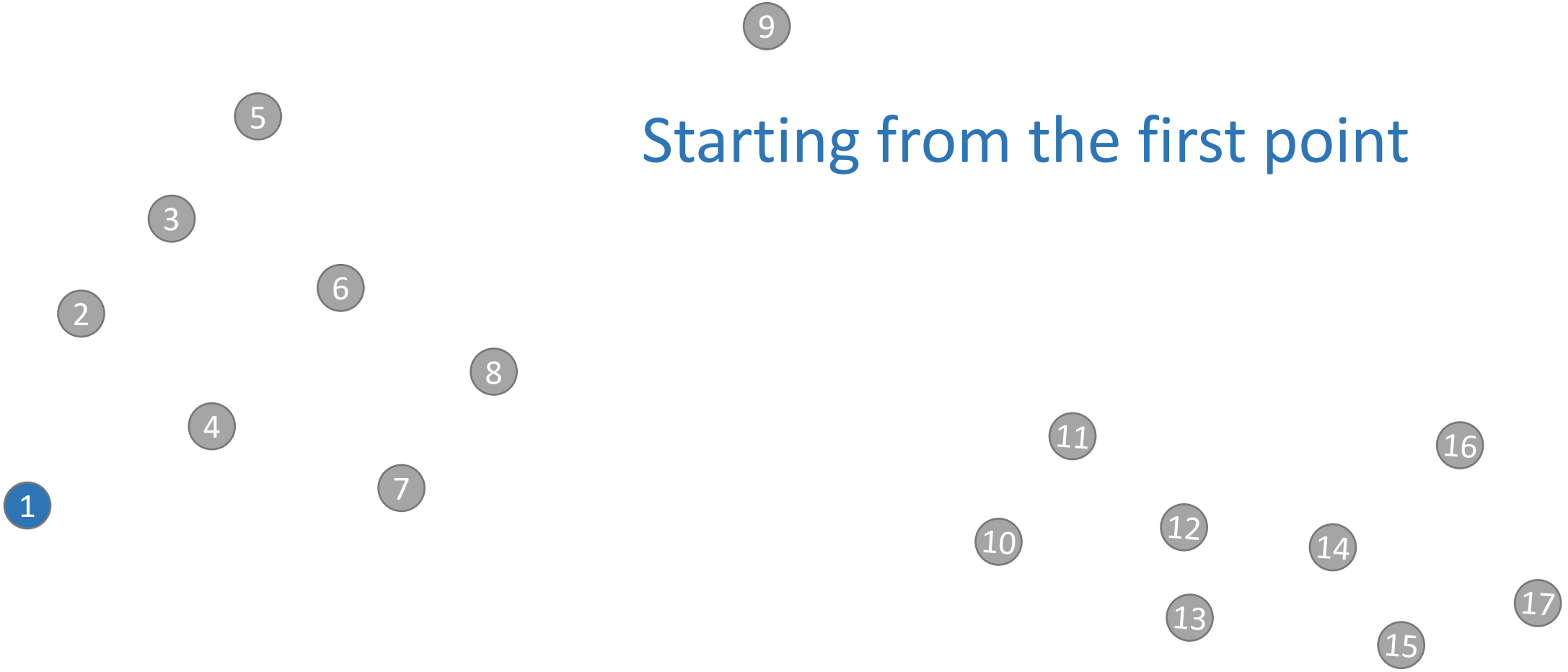
Fast and Explainable Clustering



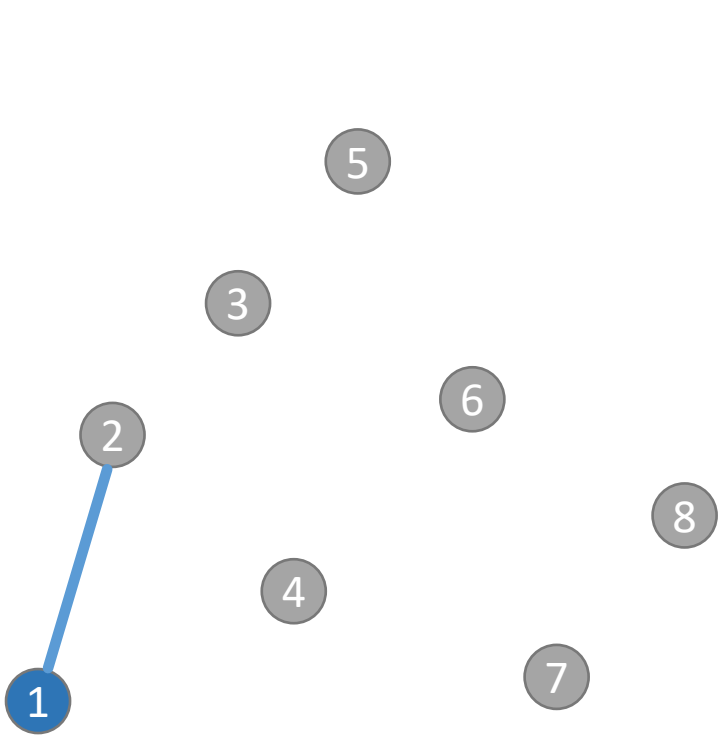
CLASSIX

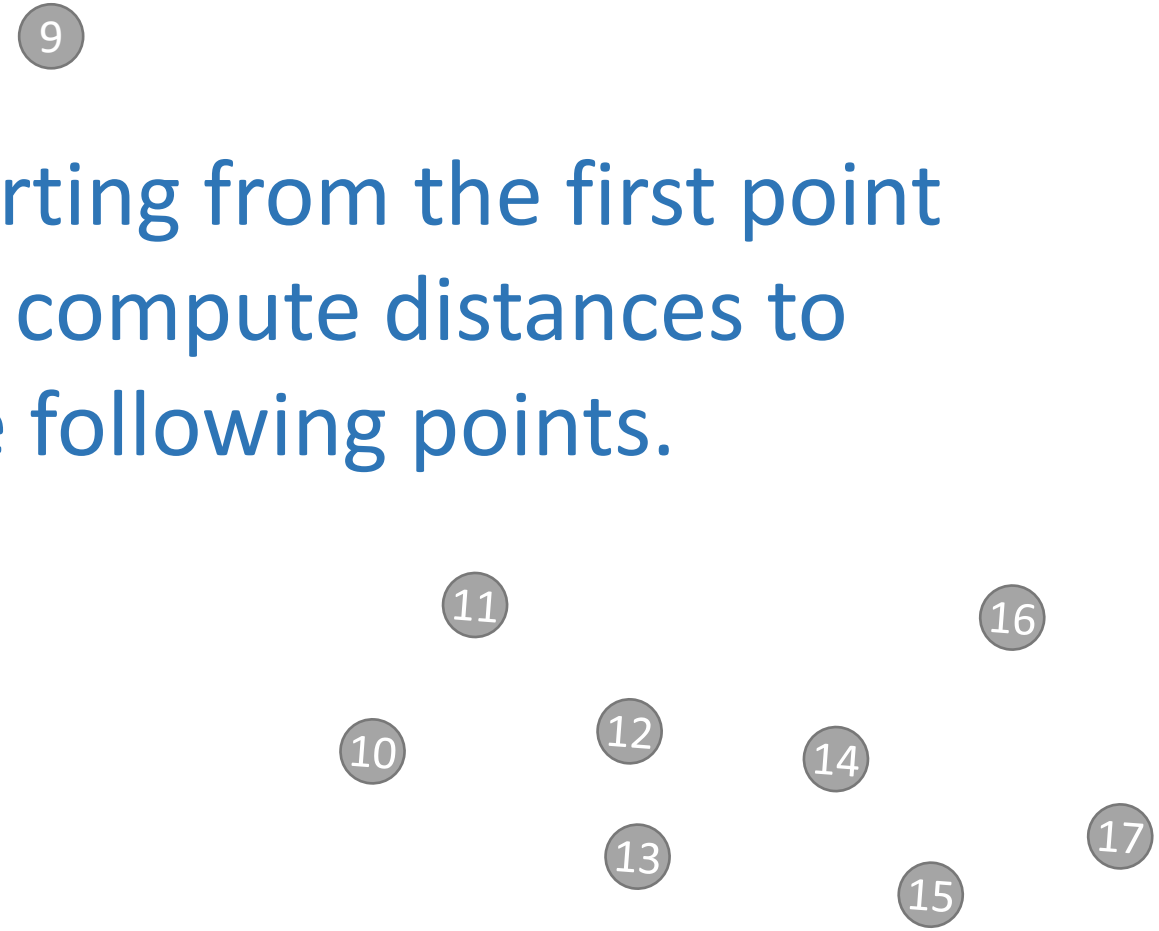
Fast and Explainable Clustering

Starting from the first point



CLASSIX
Fast and Explainable Clustering

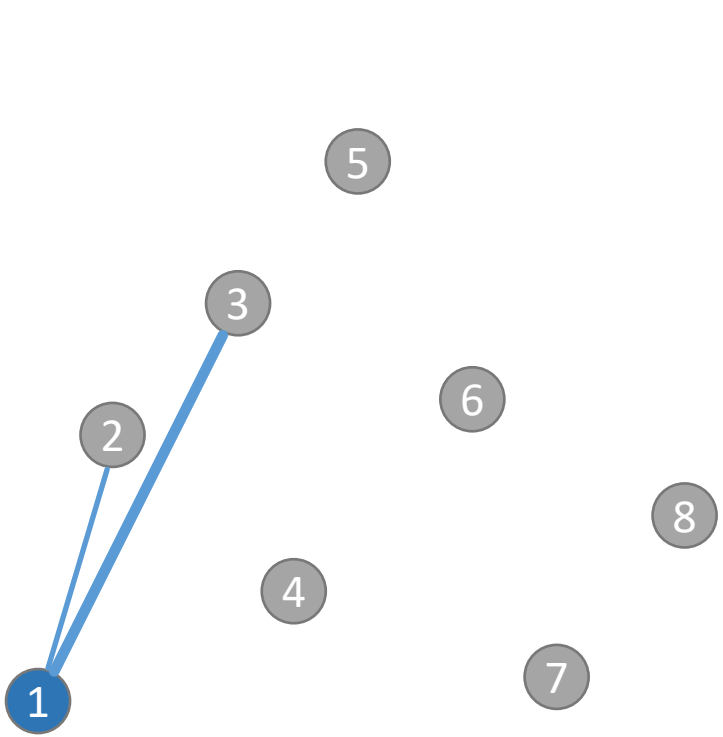




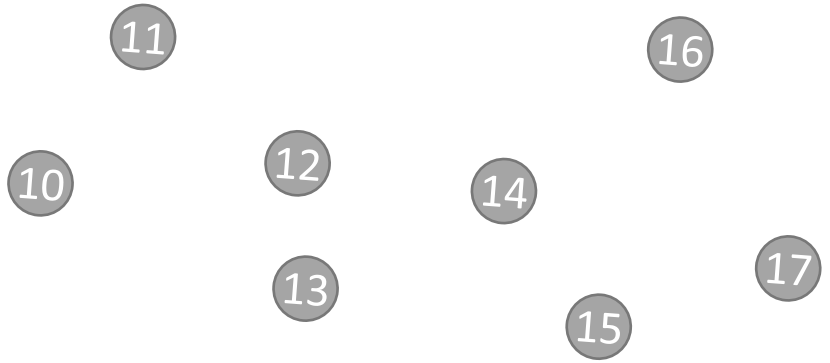
Starting from the first point we compute distances to the following points.

CLASSIX

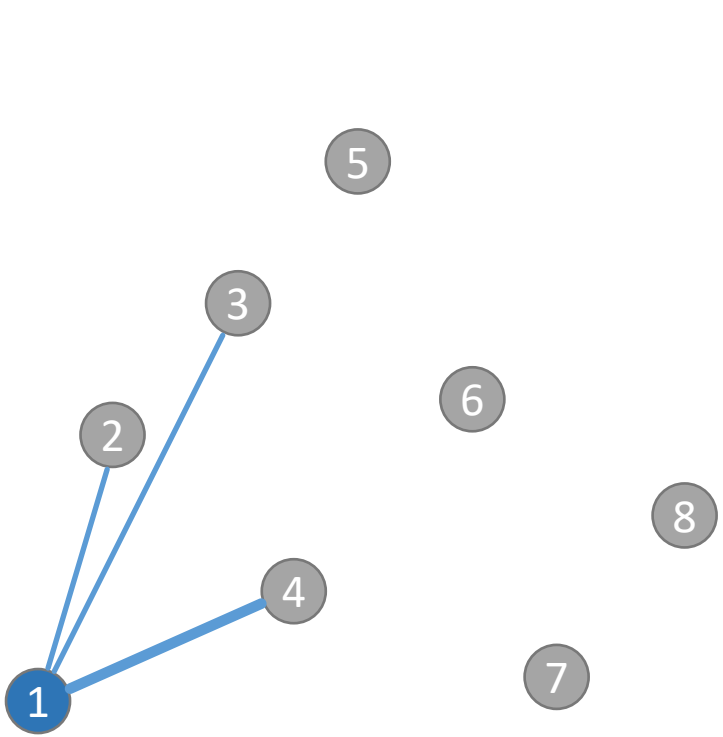
Fast and Explainable Clustering



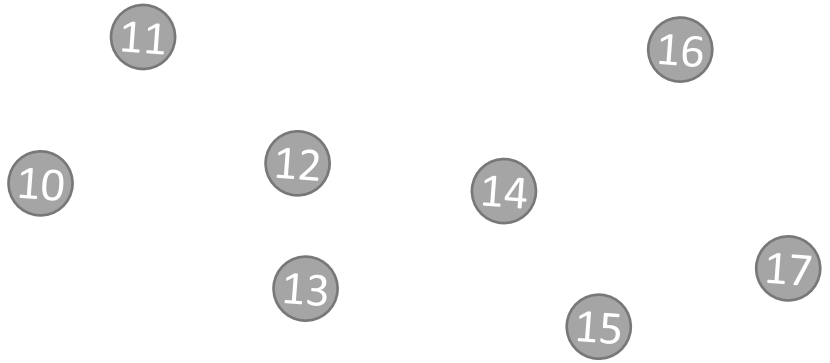
Starting from the first point we compute distances to the following points.



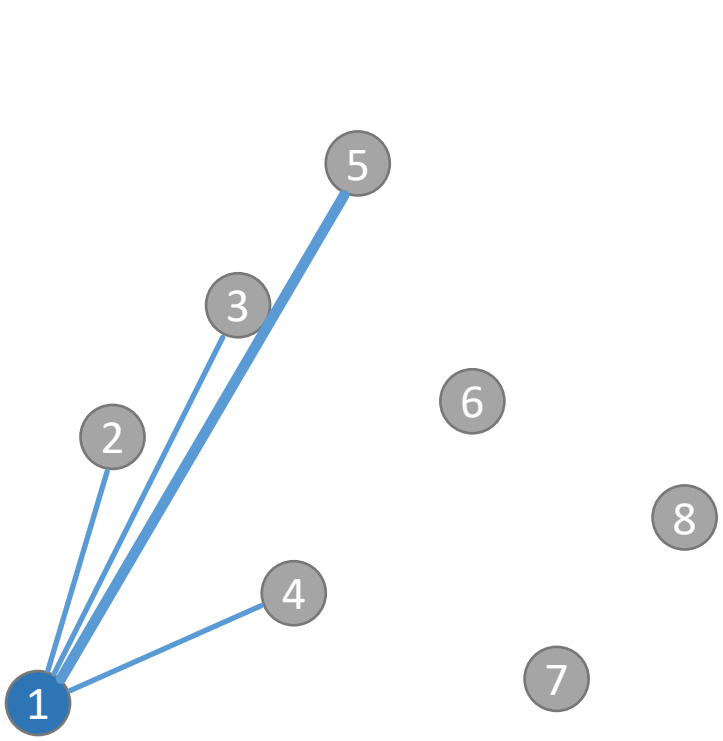
CLASSIX
Fast and Explainable Clustering



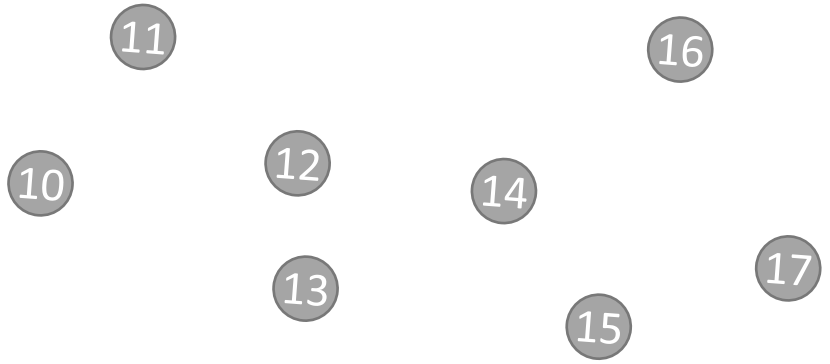
Starting from the first point we compute distances to the following points.



CLASSIX
Fast and Explainable Clustering

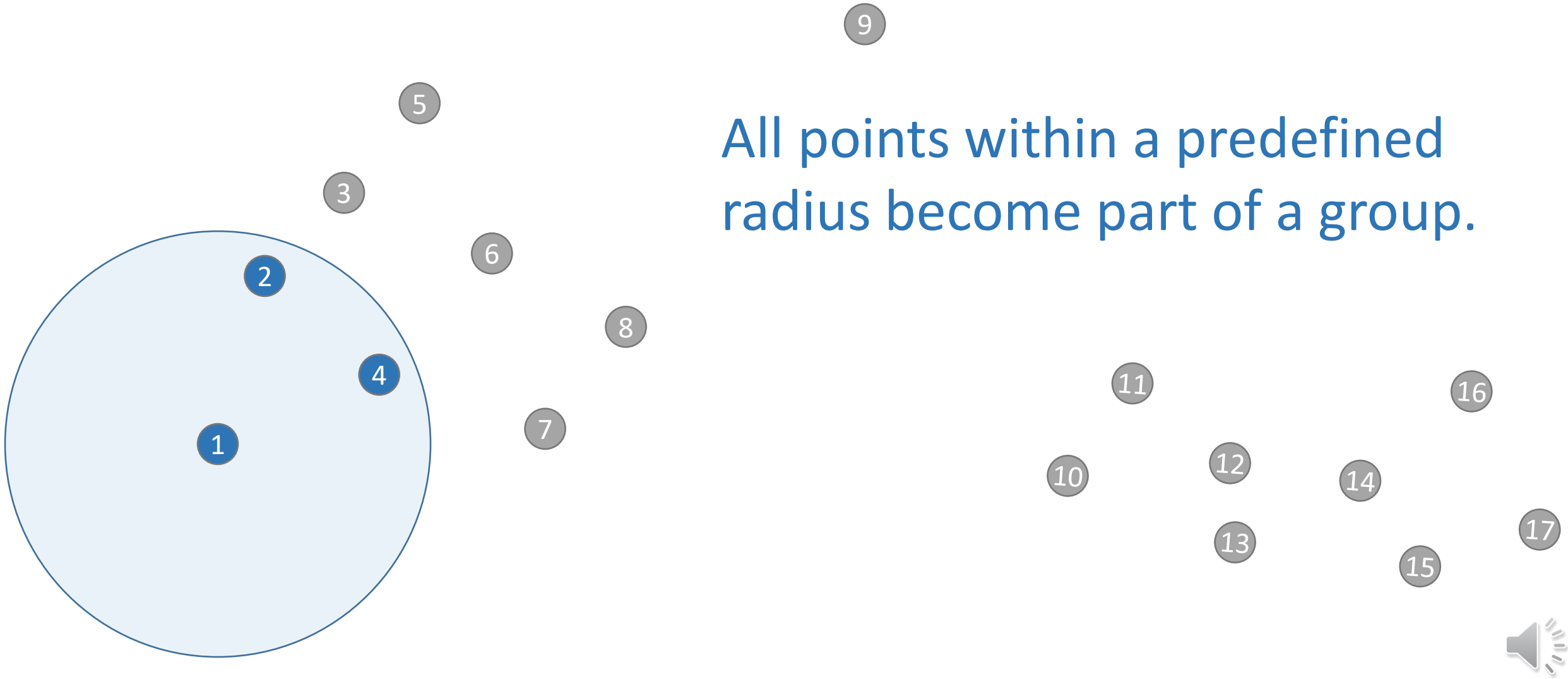


Starting from the first point we compute distances to the following points.



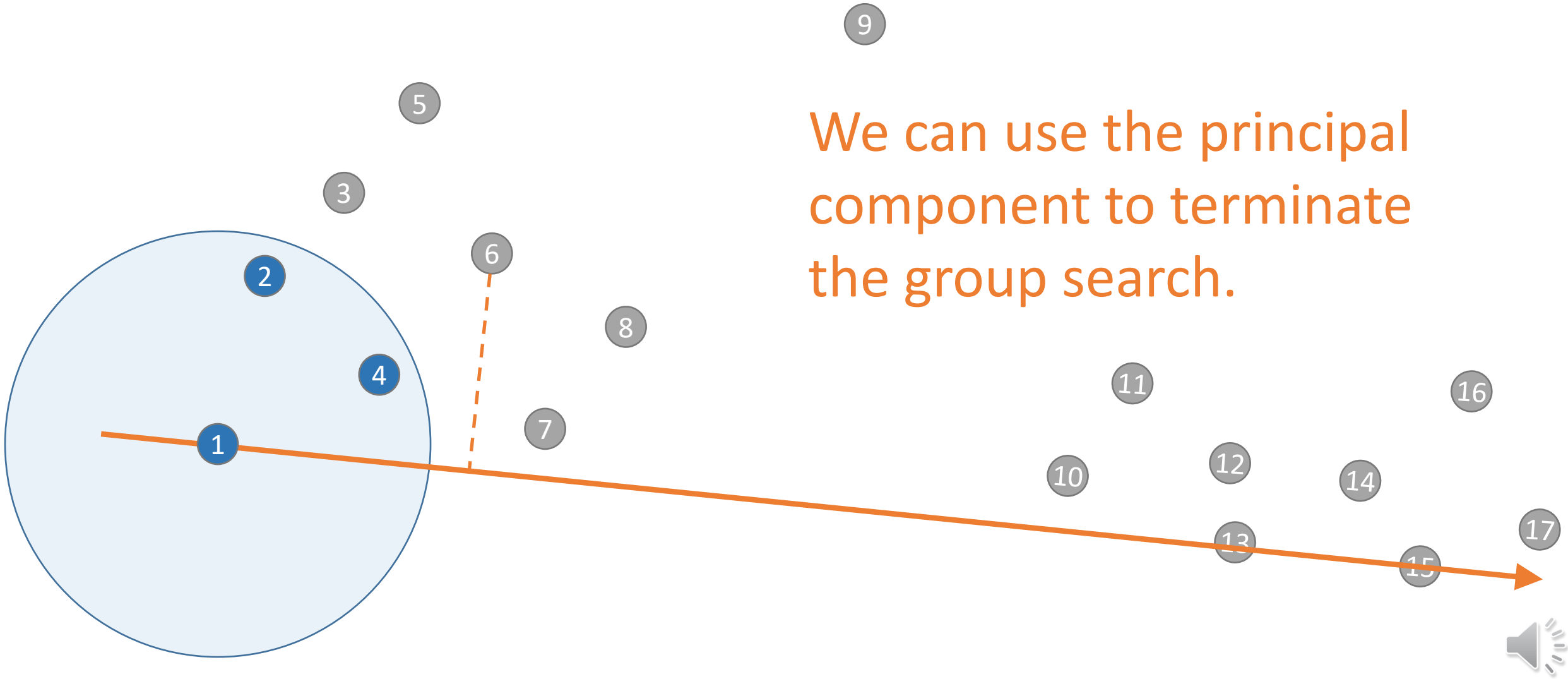
CLASSIX

Fast and Explainable Clustering



All points within a predefined radius become part of a group.

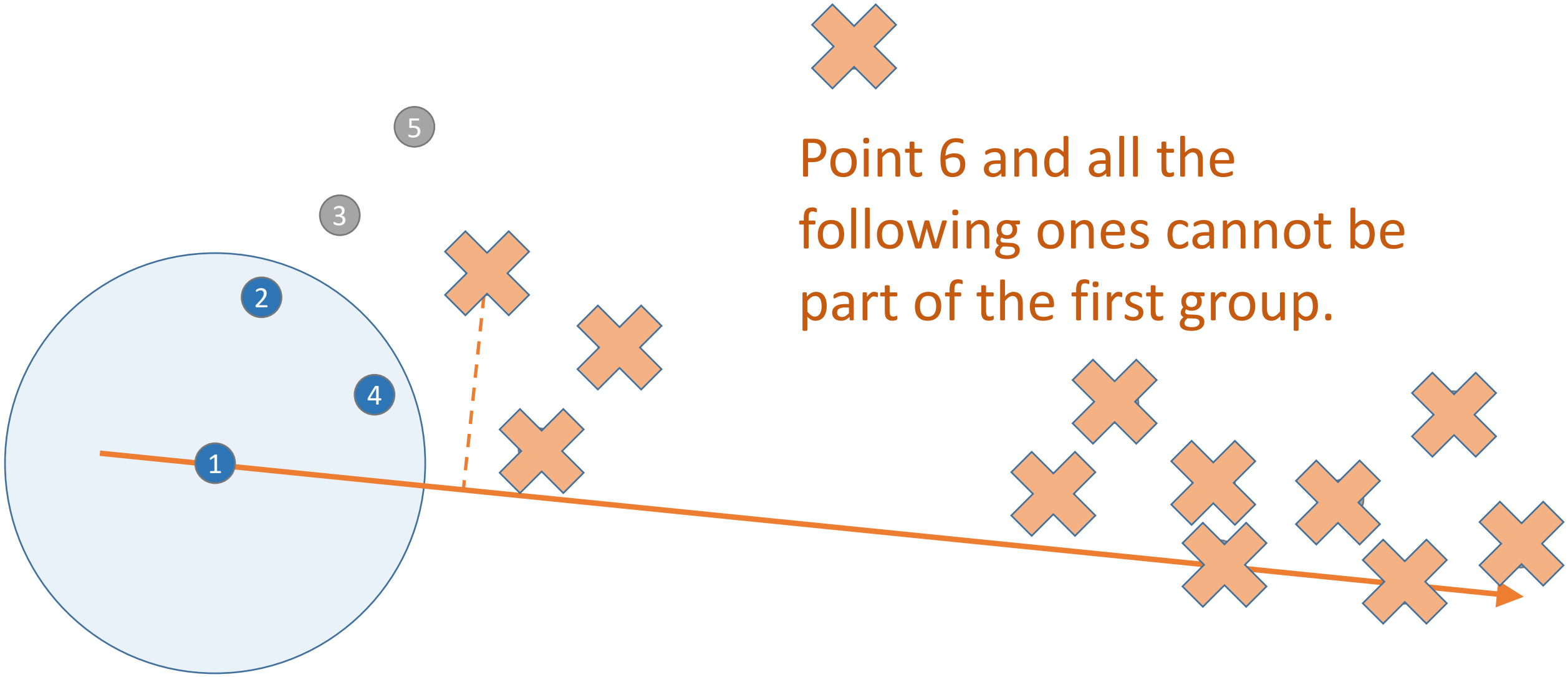
CLASSIX
Fast and Explainable Clustering



We can use the principal component to terminate the group search.

CLASSIX

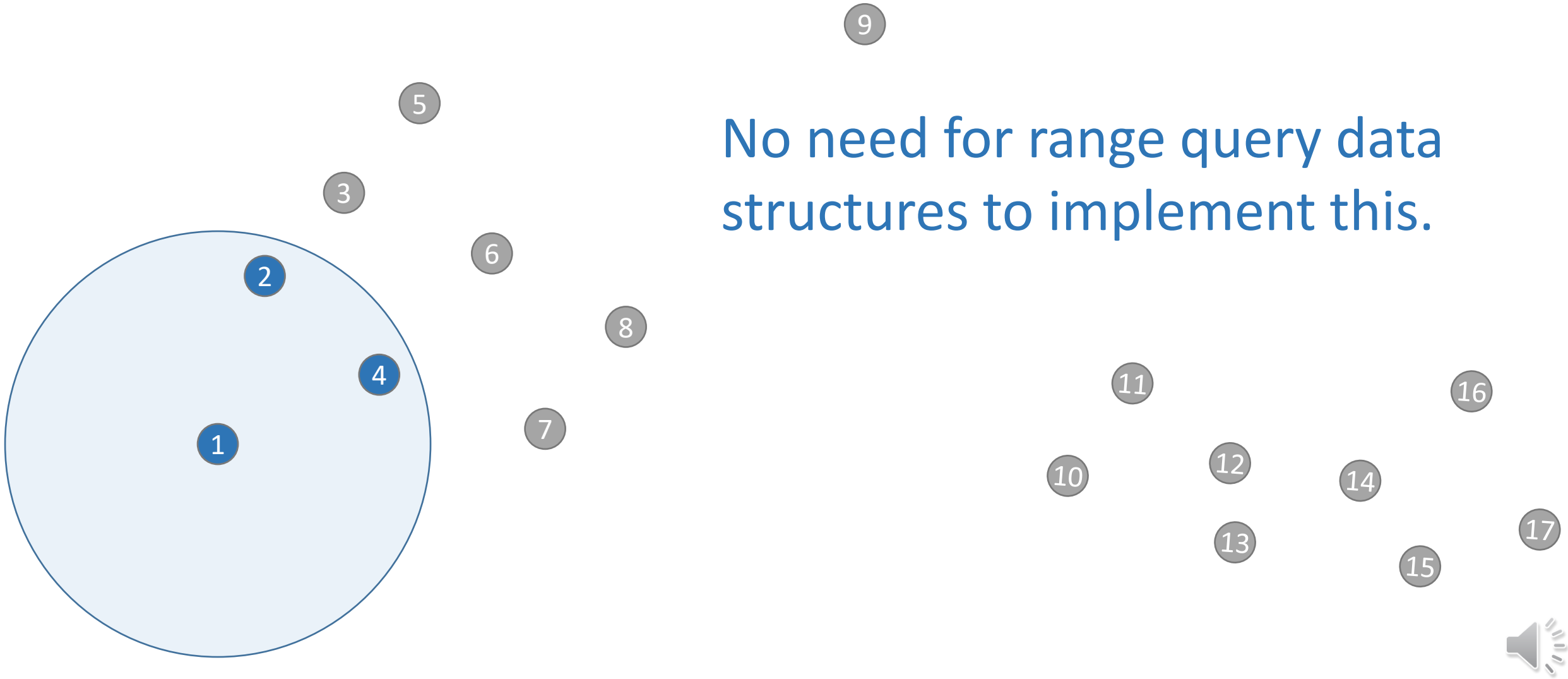
Fast and Explainable Clustering



Point 6 and all the following ones cannot be part of the first group.

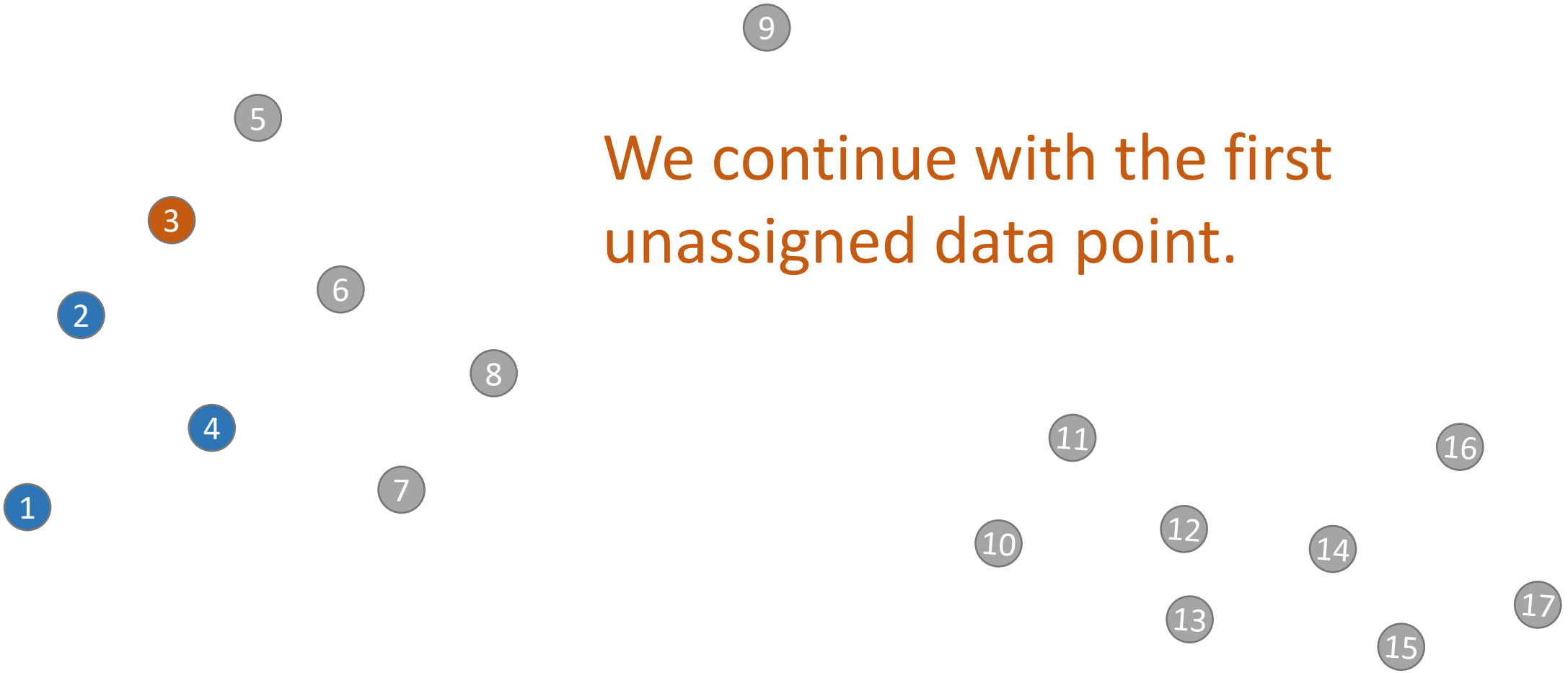
CLASSIX

Fast and Explainable Clustering



No need for range query data structures to implement this.

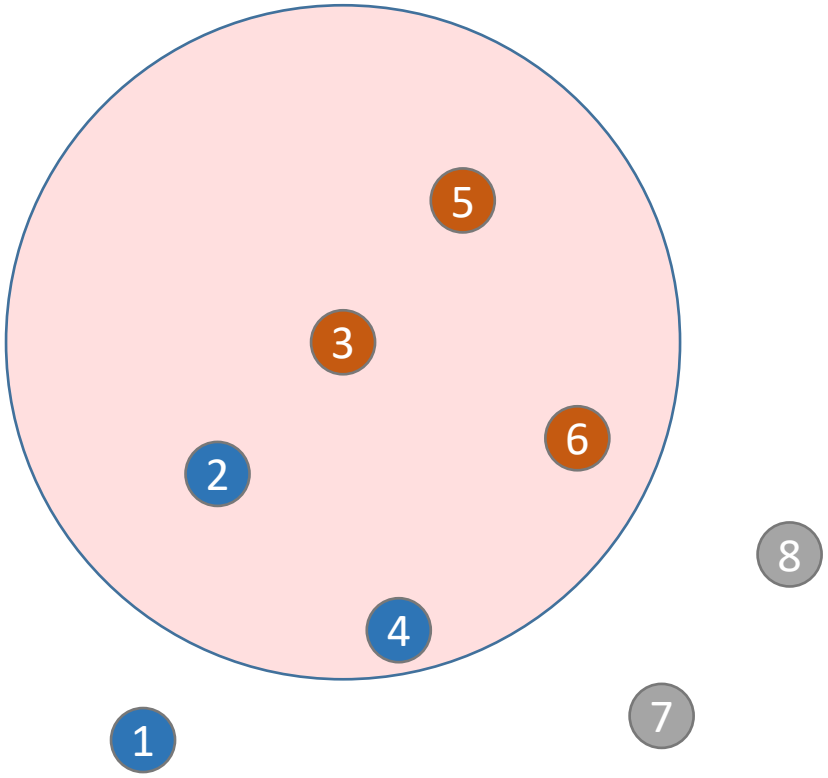
CLASSIX
Fast and Explainable Clustering



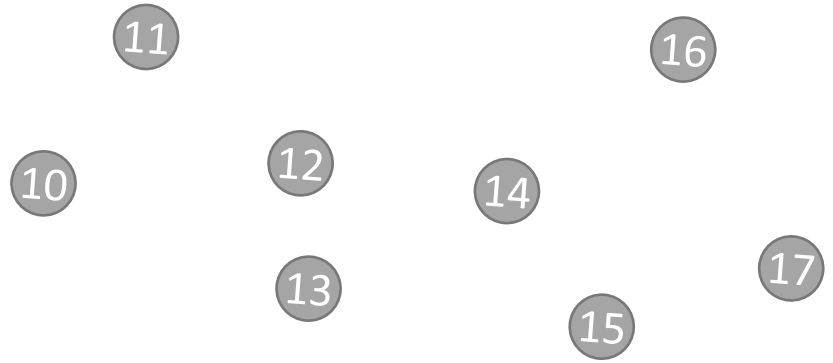
We continue with the first unassigned data point.

CLASSIX

Fast and Explainable Clustering

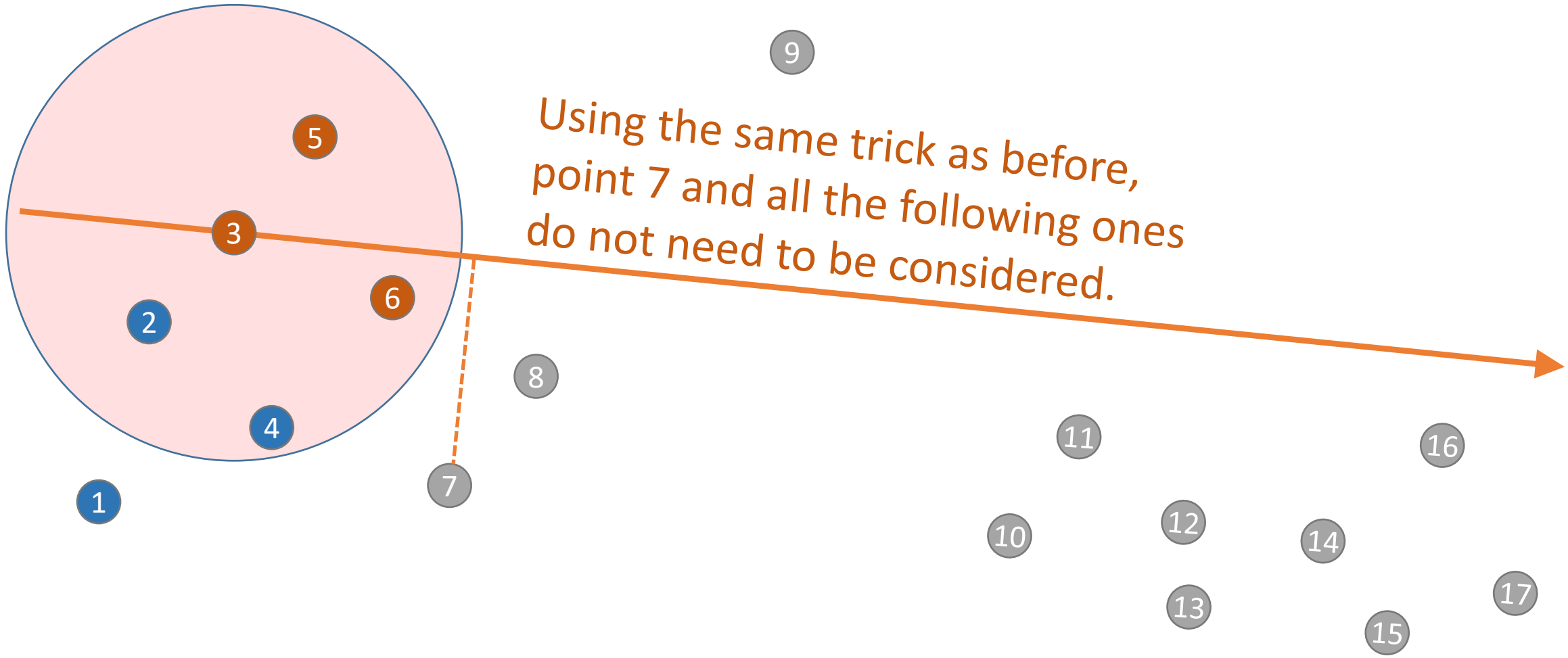


And group all data points within the predefined radius.



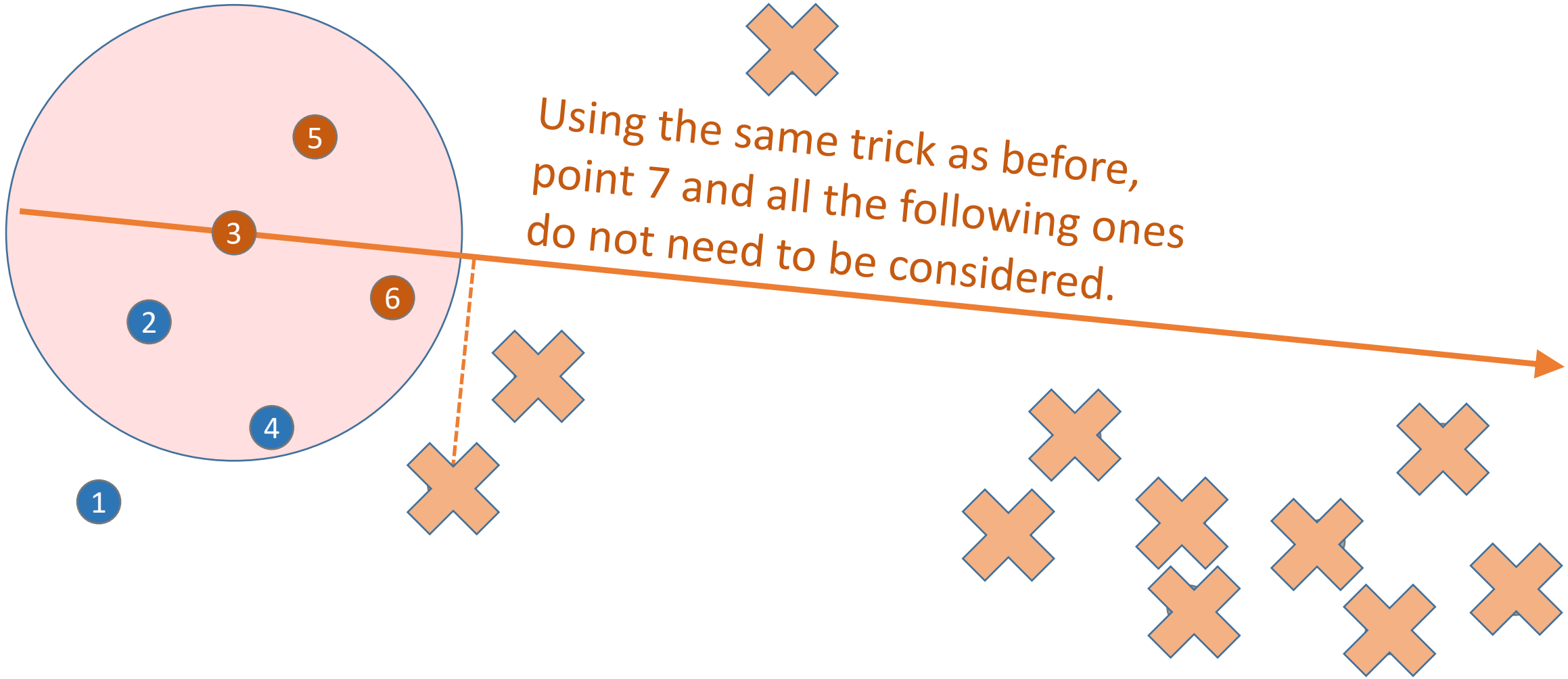
CLASSIX

Fast and Explainable Clustering



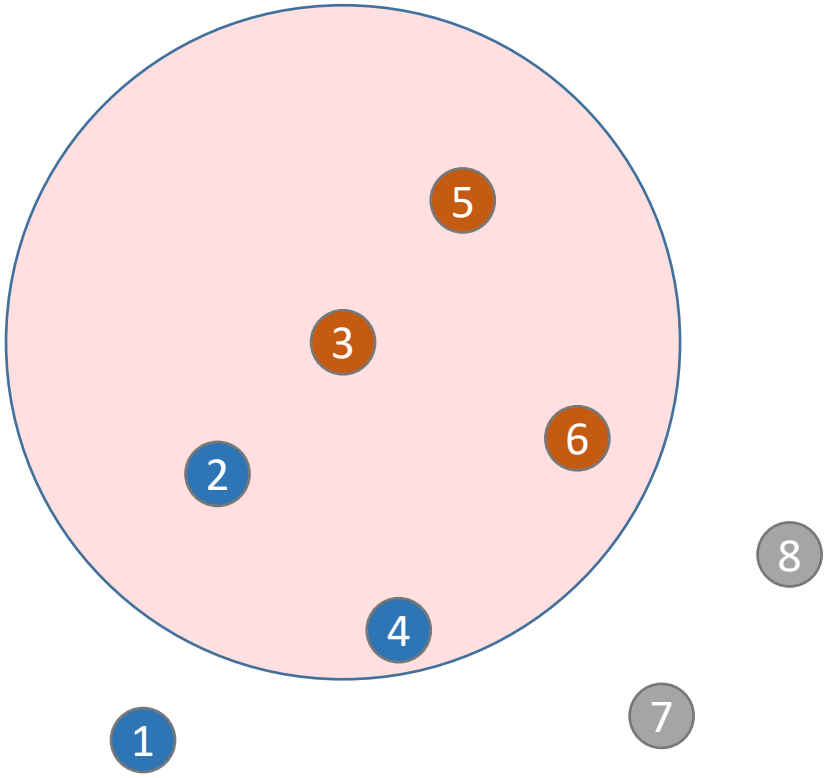
CLASSIX

Fast and Explainable Clustering

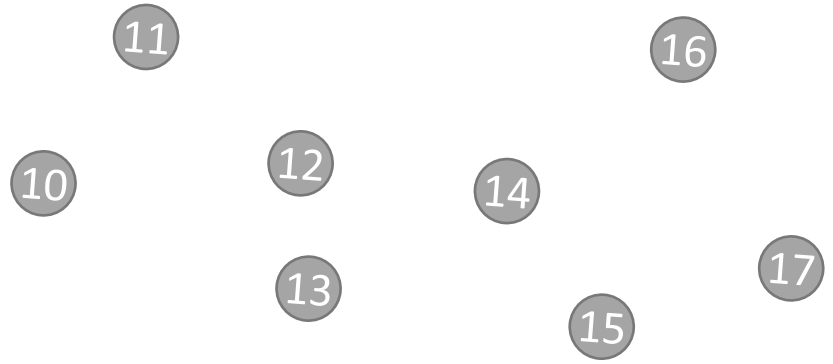


CLASSIX

Fast and Explainable Clustering

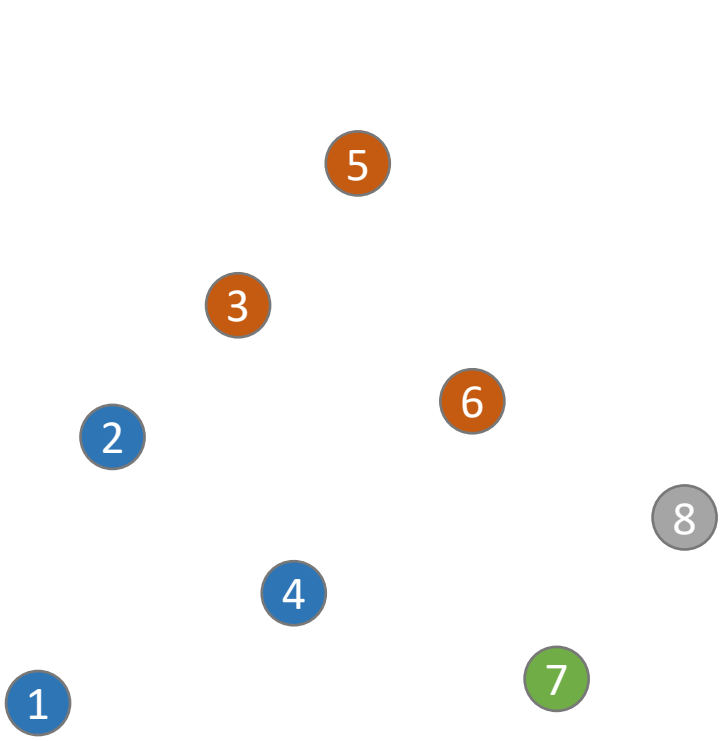


This early search termination keeps the number of distance calculations low.

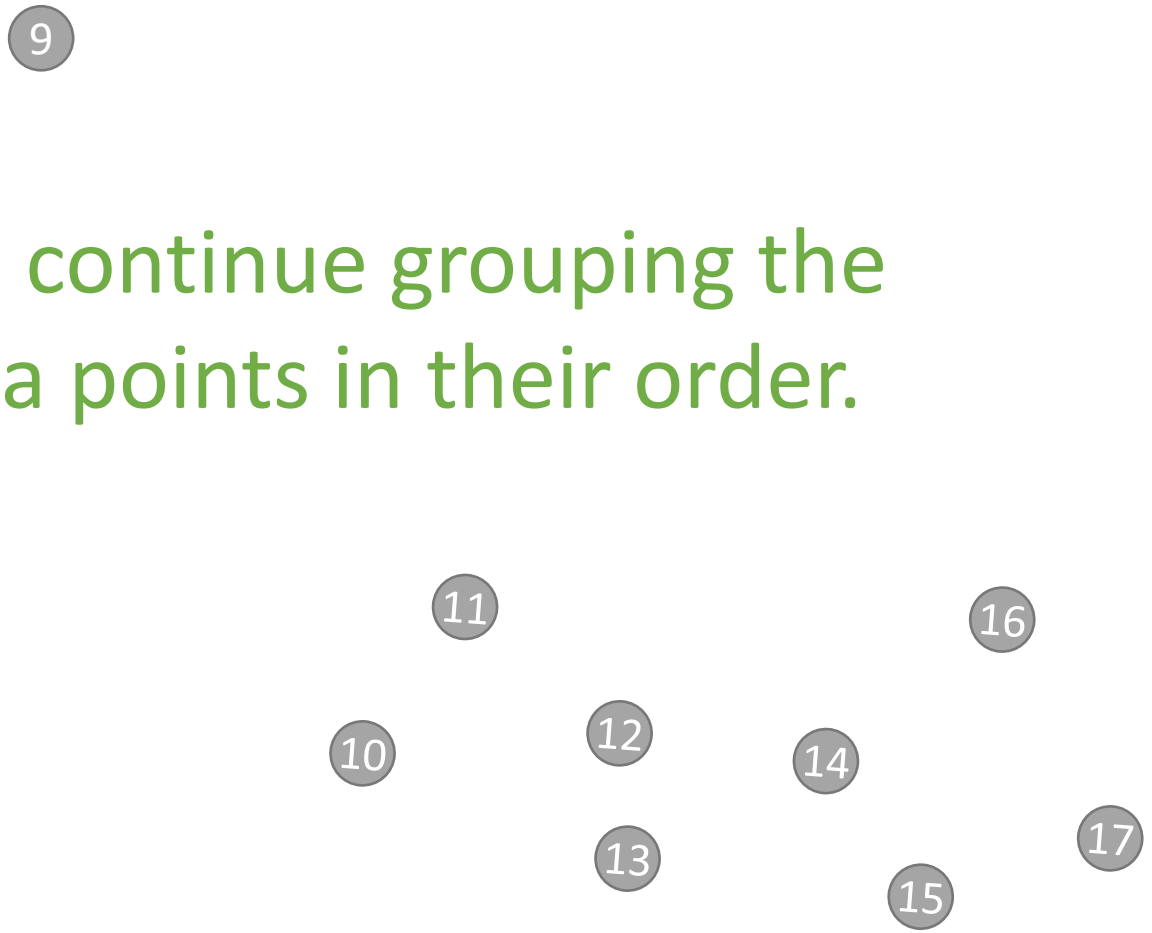


CLASSIX

Fast and Explainable Clustering

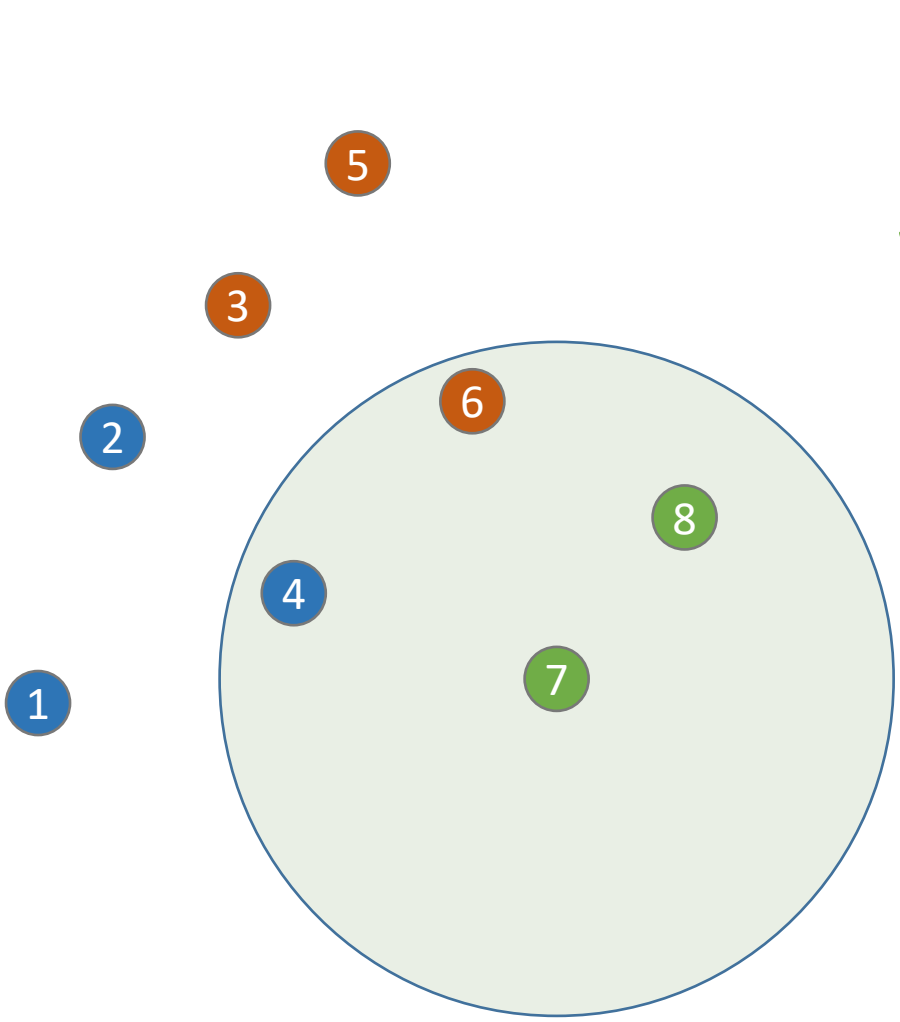


We continue grouping the data points in their order.

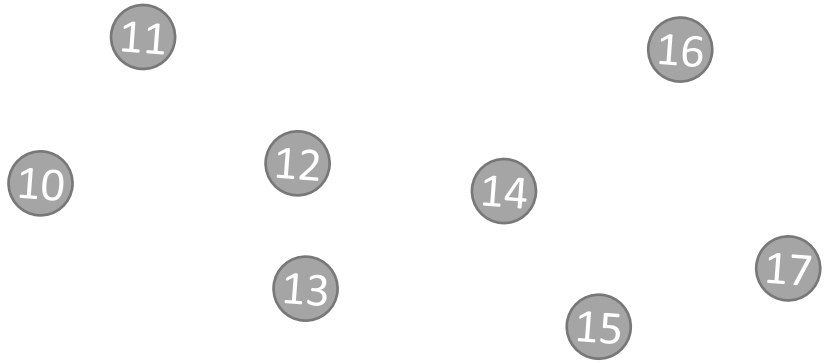


CLASSIX

Fast and Explainable Clustering

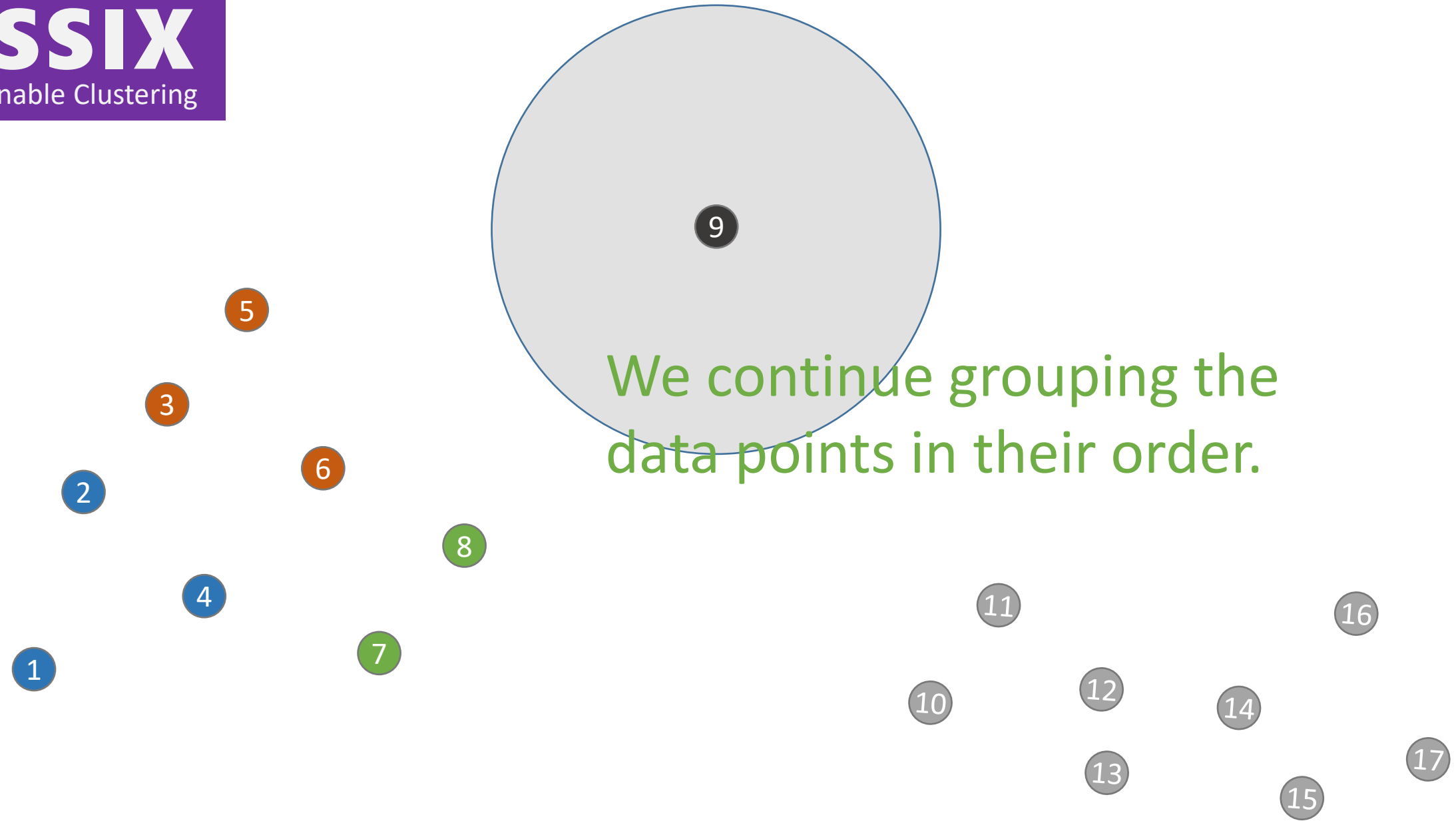


We continue grouping the data points in their order.



CLASSIX

Fast and Explainable Clustering

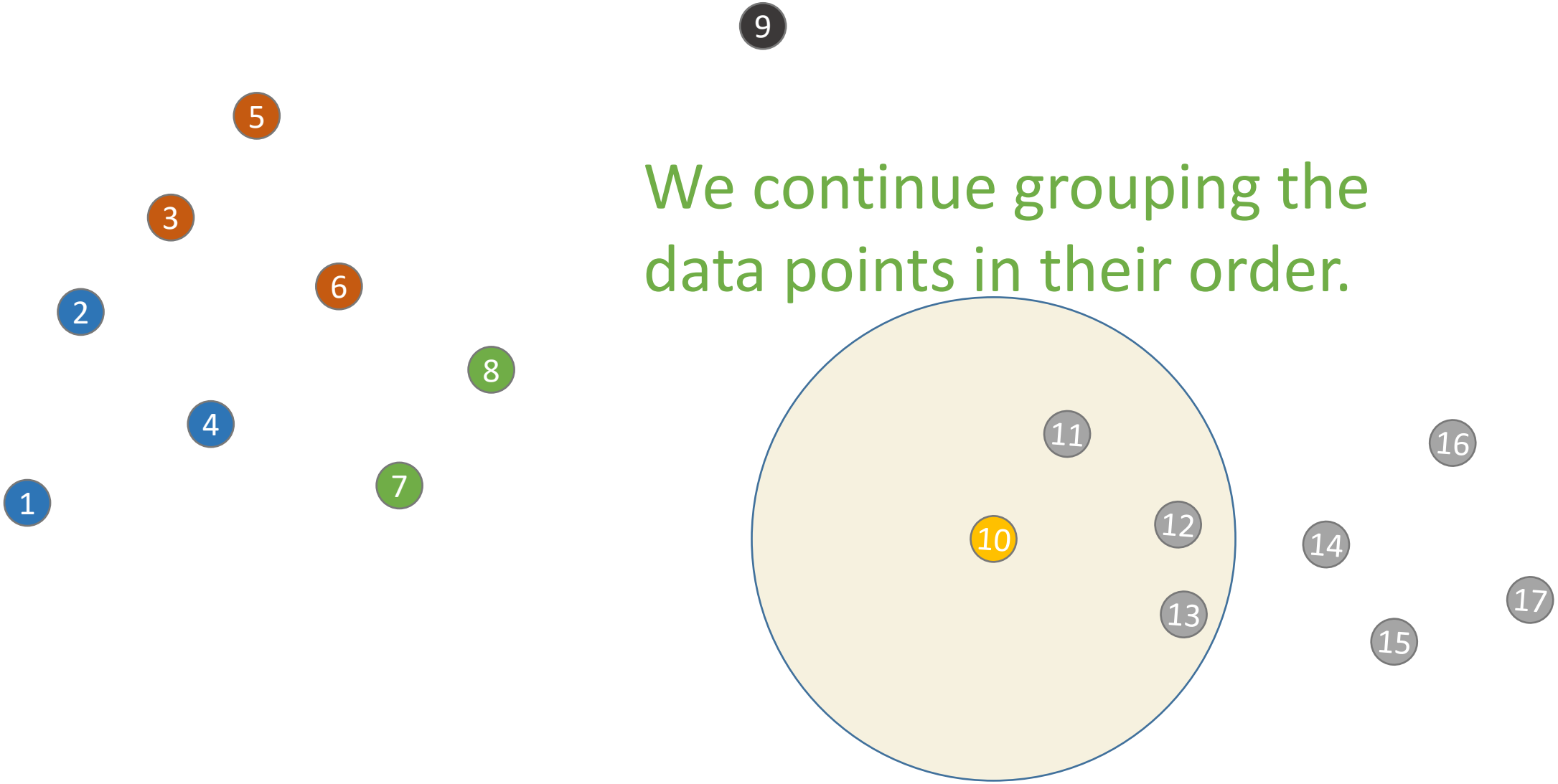


We continue grouping the data points in their order.

CLASSIX

Fast and Explainable Clustering

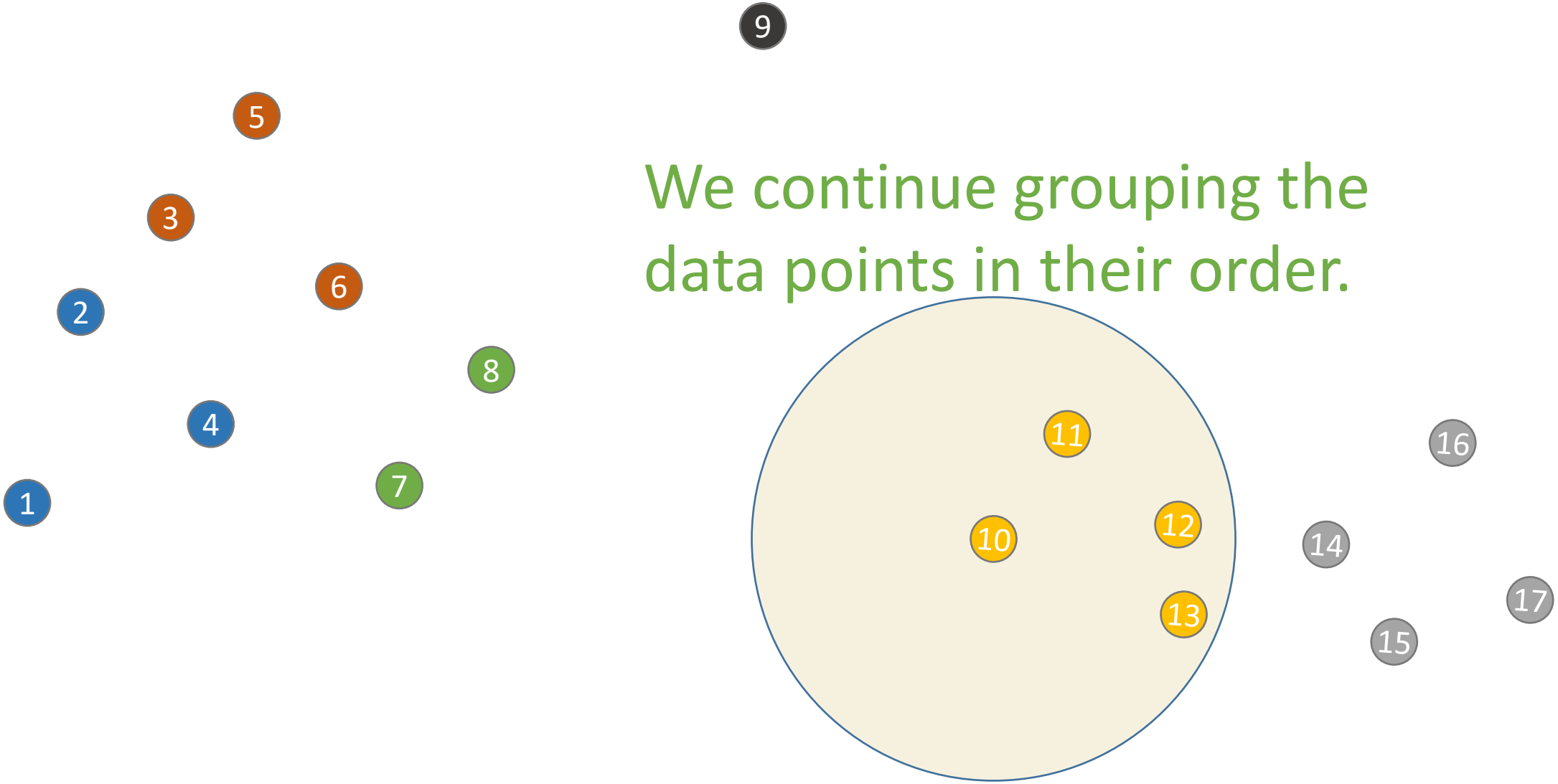
We continue grouping the data points in their order.



CLASSIX

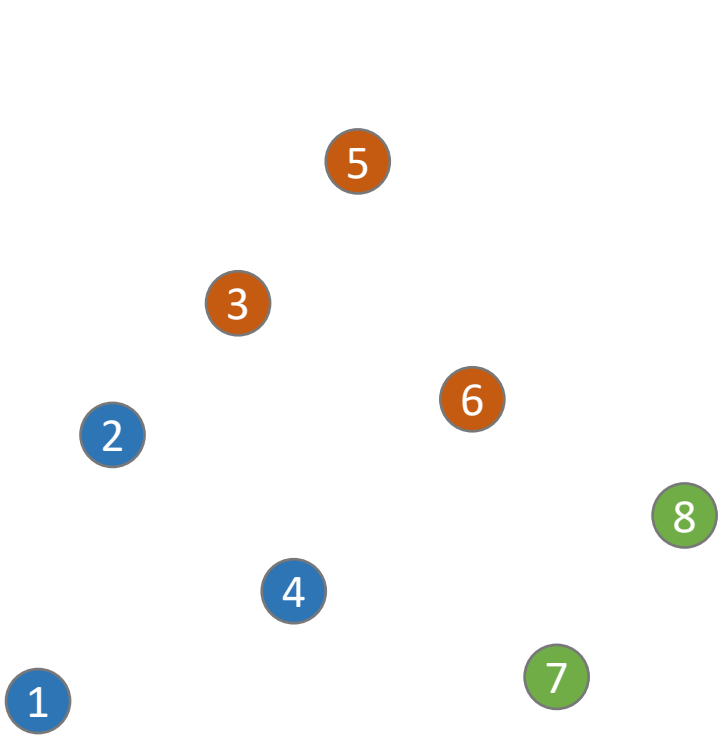
Fast and Explainable Clustering

We continue grouping the data points in their order.



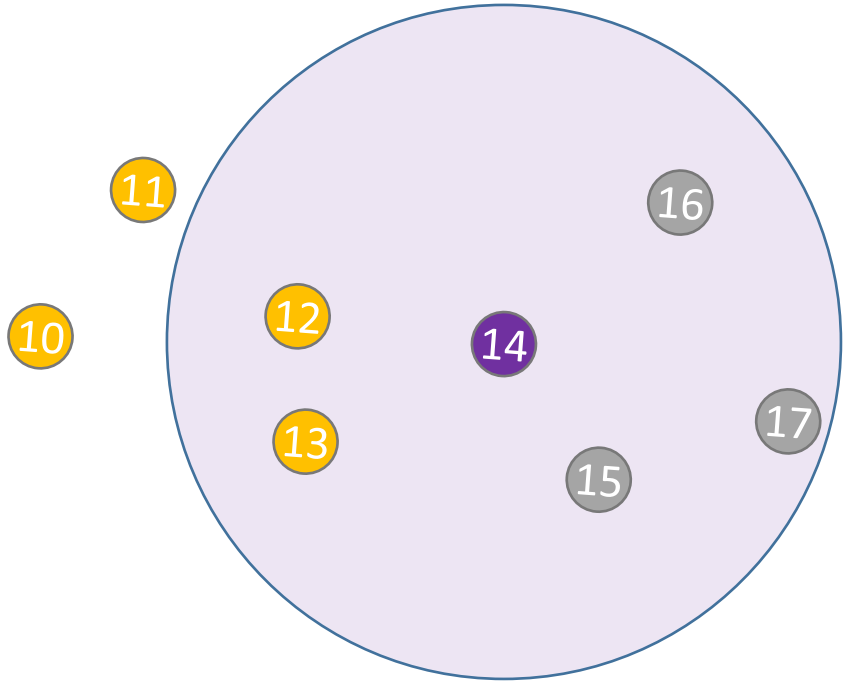
CLASSIX

Fast and Explainable Clustering



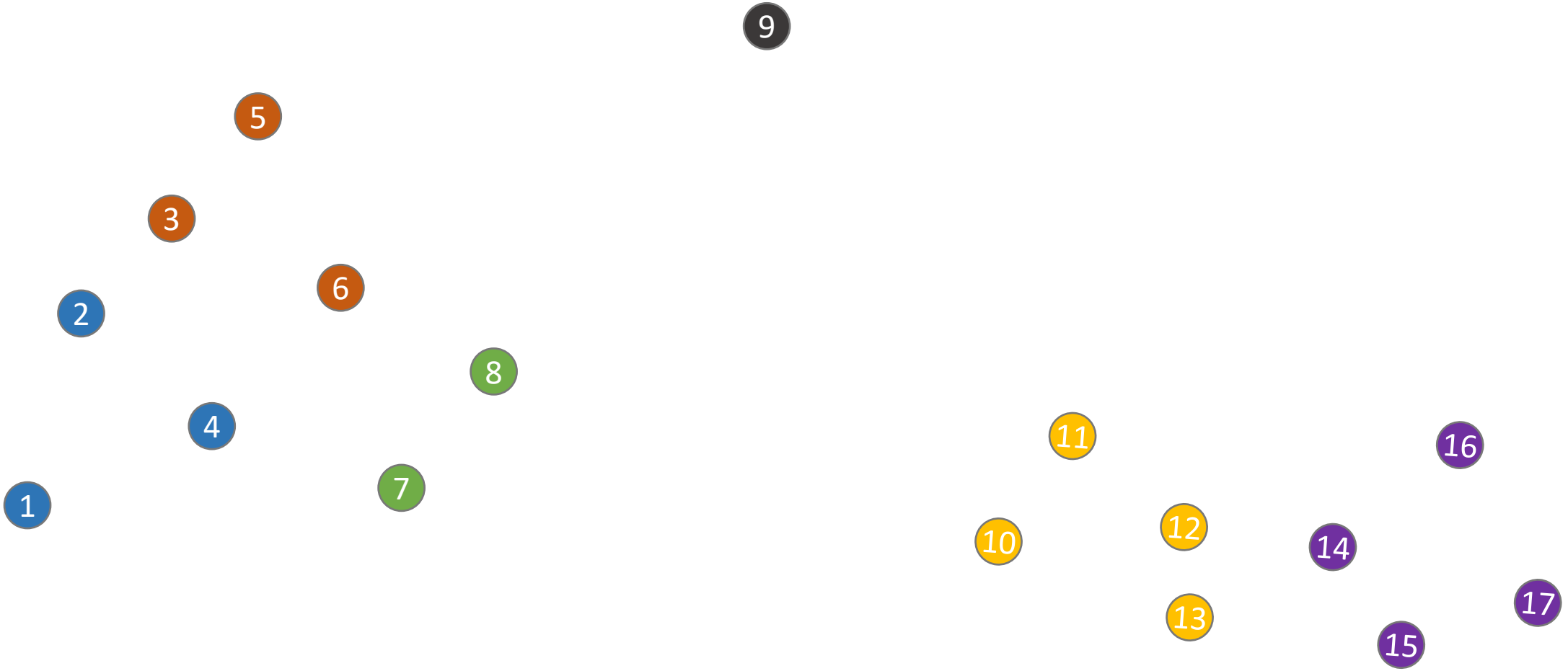
9

We continue grouping the data points in their order.



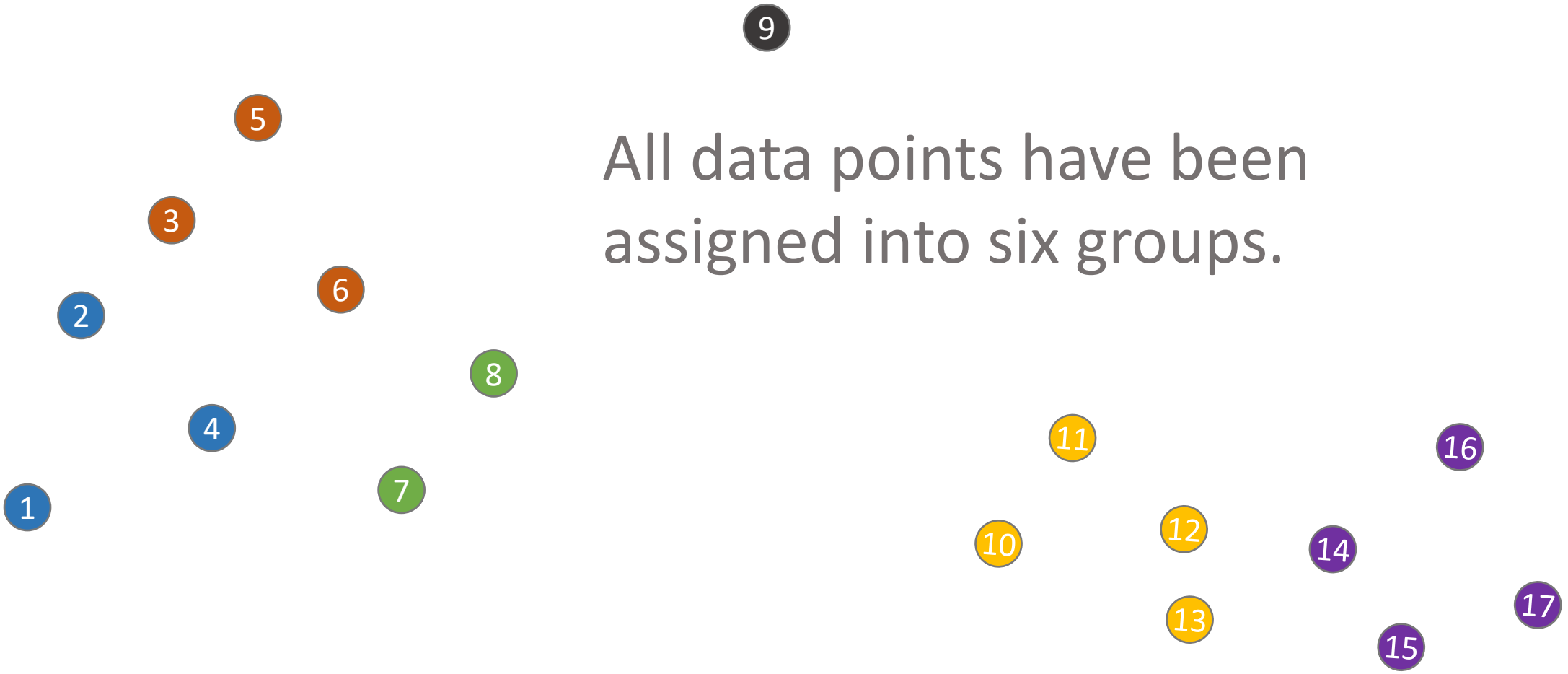
CLASSIX

Fast and Explainable Clustering



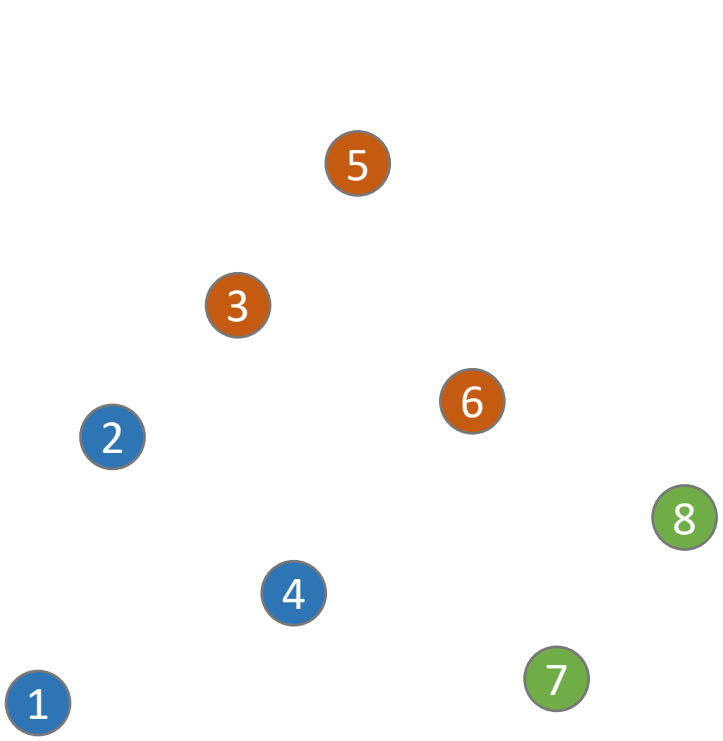
CLASSIX

Fast and Explainable Clustering



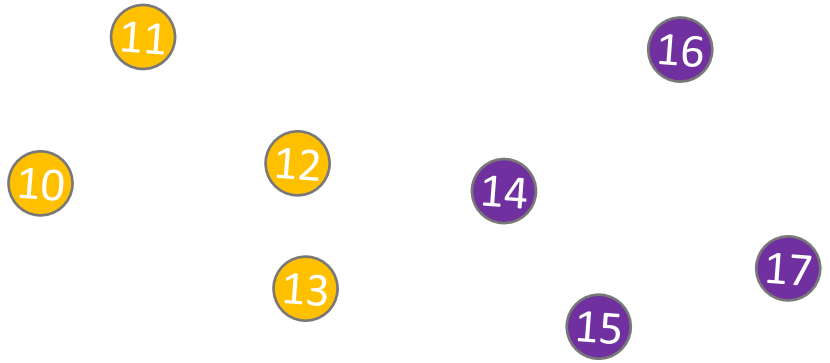
All data points have been assigned into six groups.

CLASSIX
Fast and Explainable Clustering

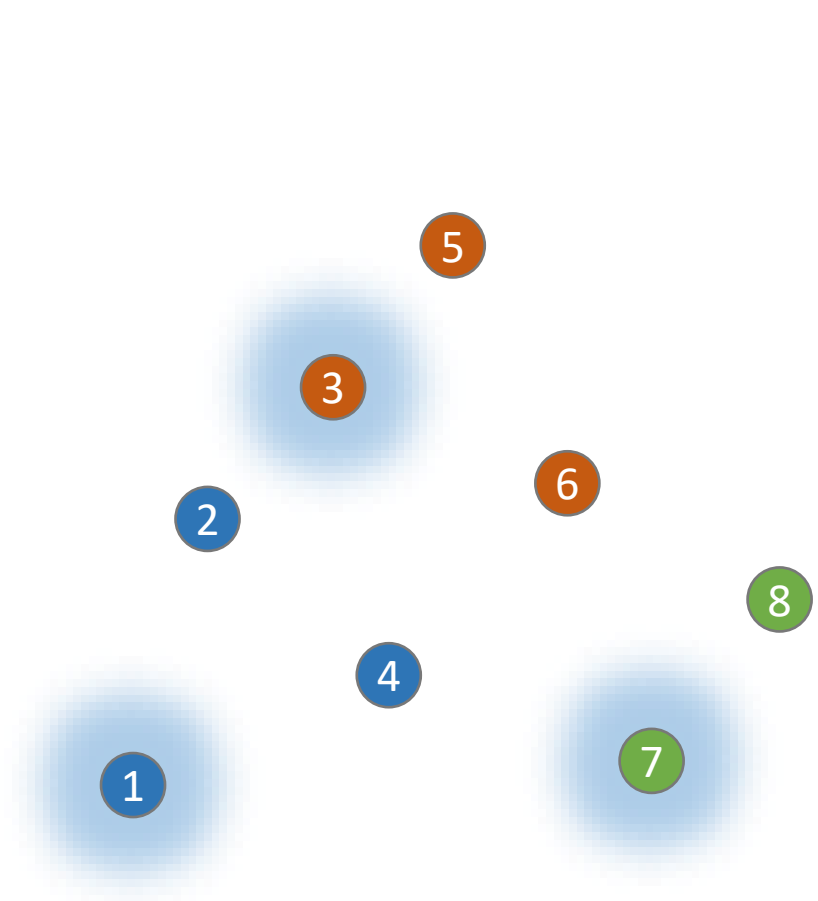


9

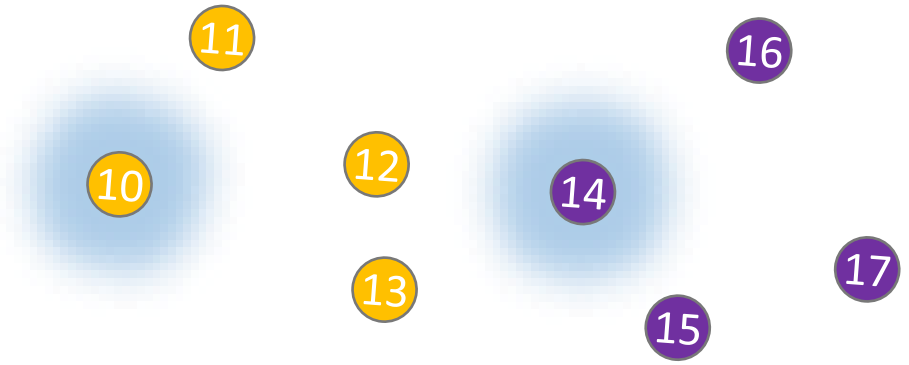
Now we just have to merge these groups into clusters.



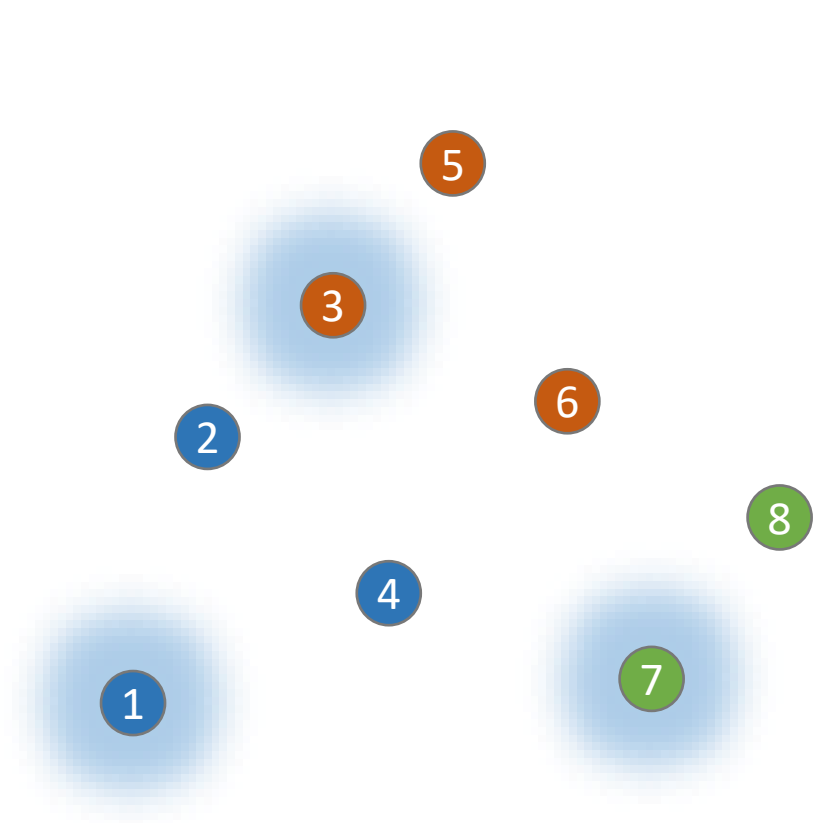
CLASSIX
Fast and Explainable Clustering



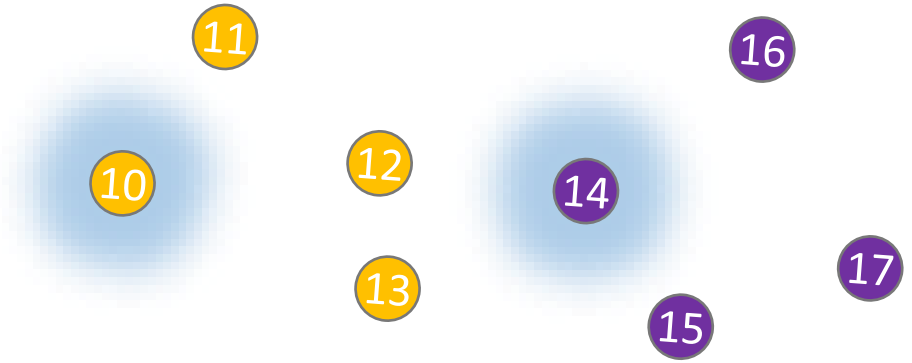
This can be done efficiently by using the starting points of each group.



CLASSIX
Fast and Explainable Clustering

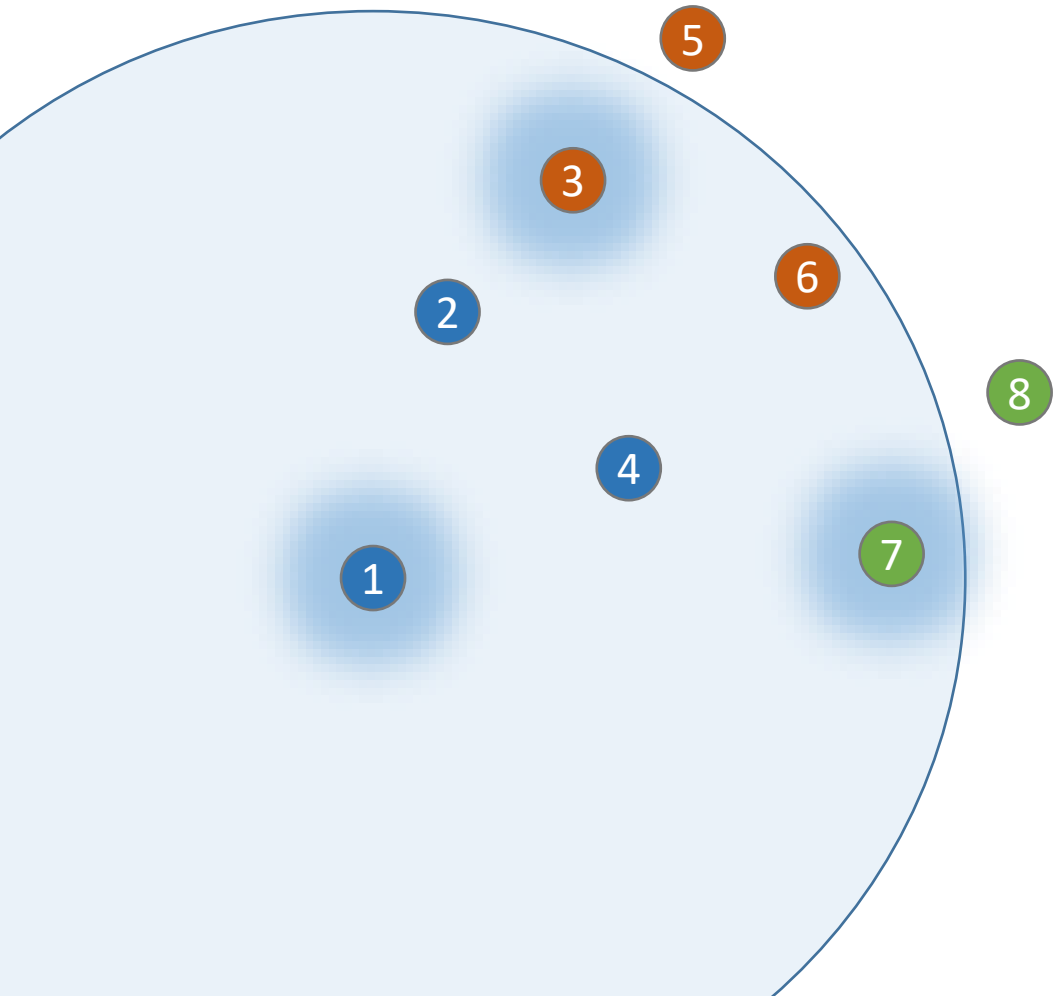


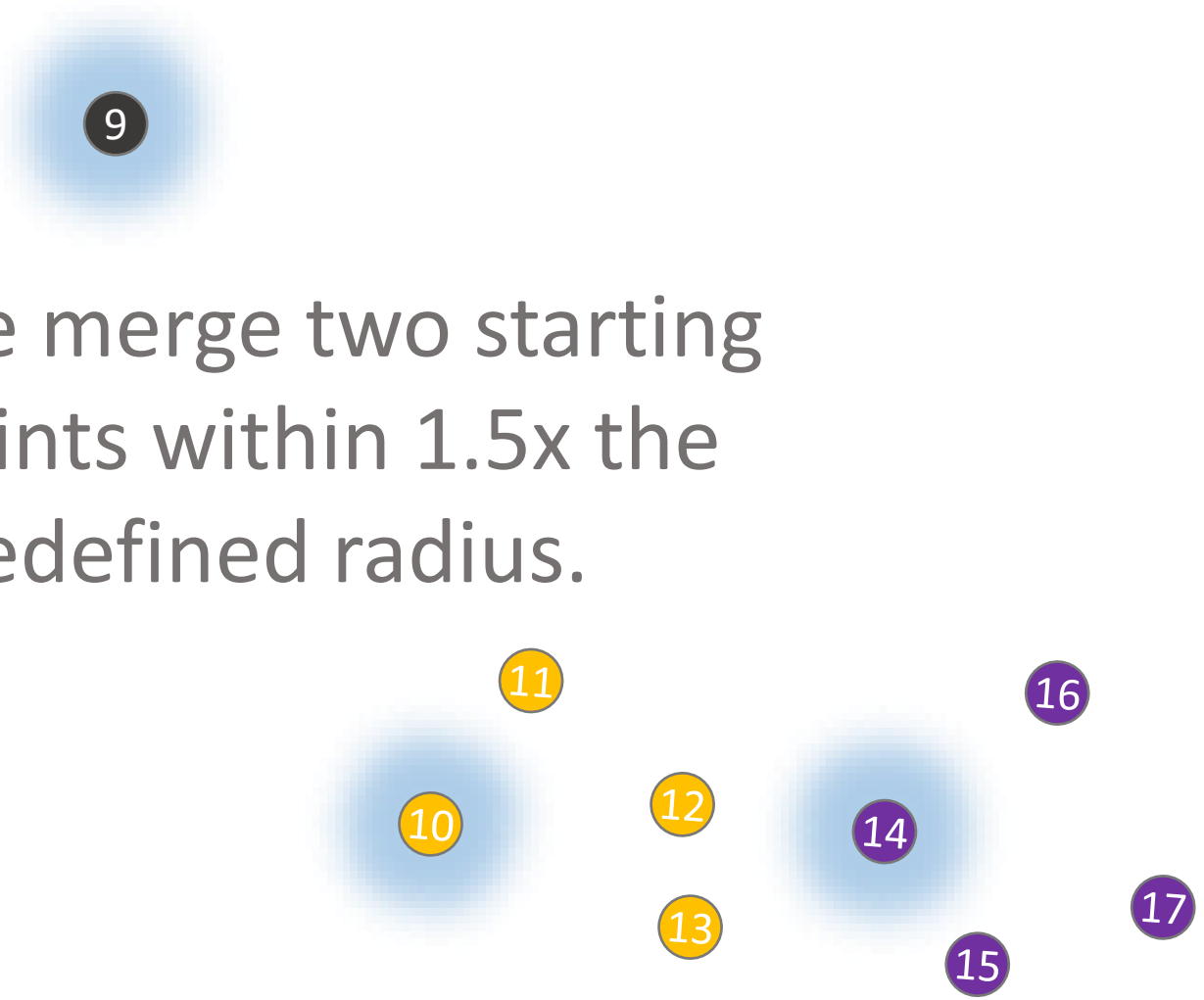
The starting points are already sorted and we can again use early search termination.



CLASSIX

Fast and Explainable Clustering

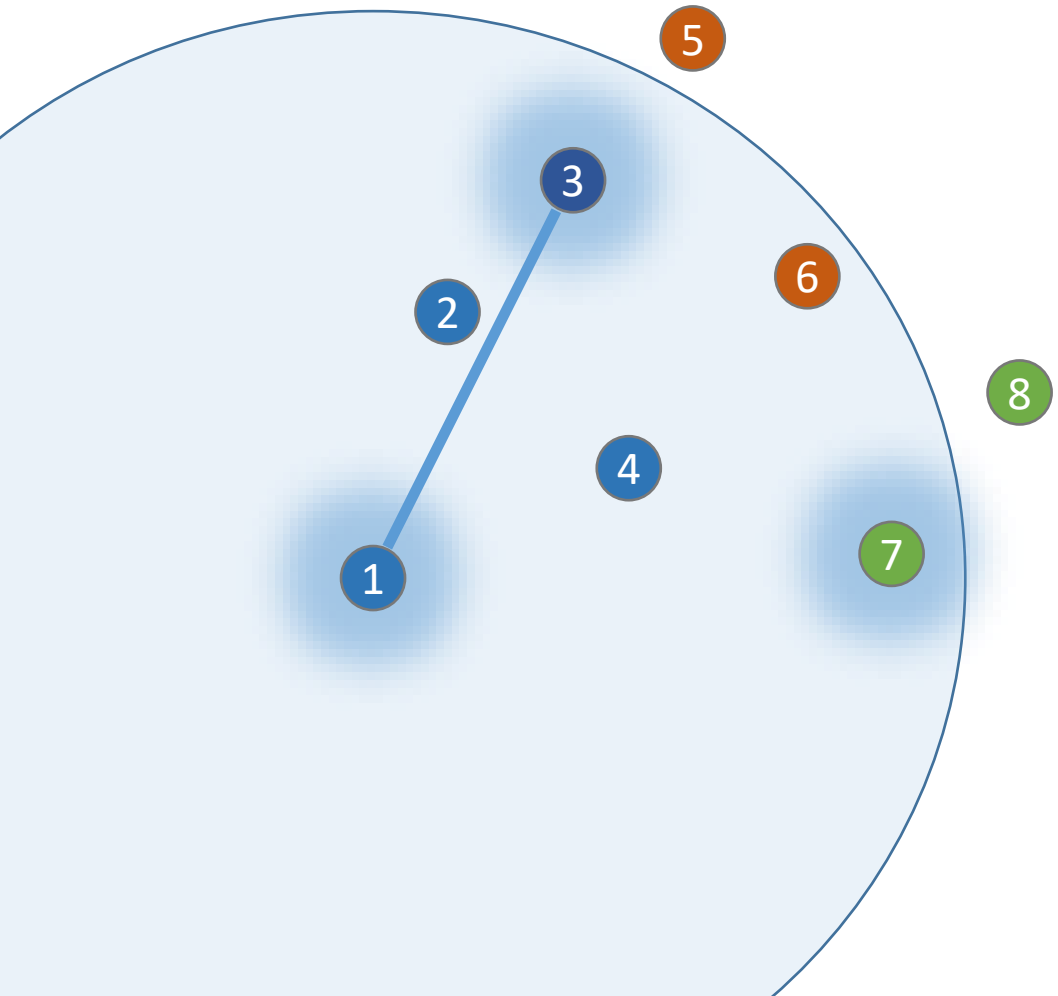




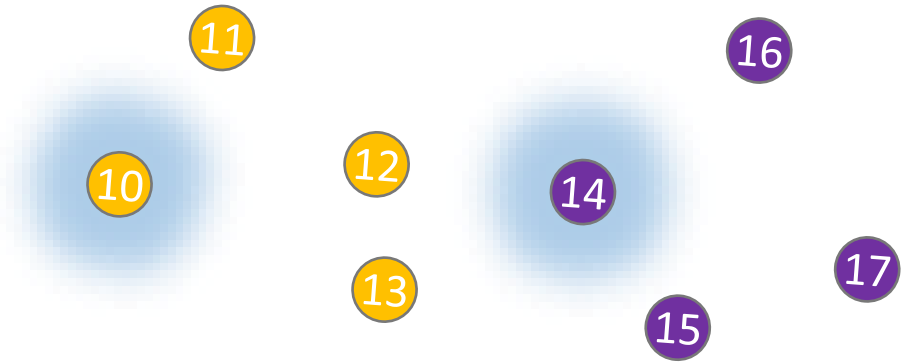
We merge two starting points within 1.5x the predefined radius.

CLASSIX

Fast and Explainable Clustering

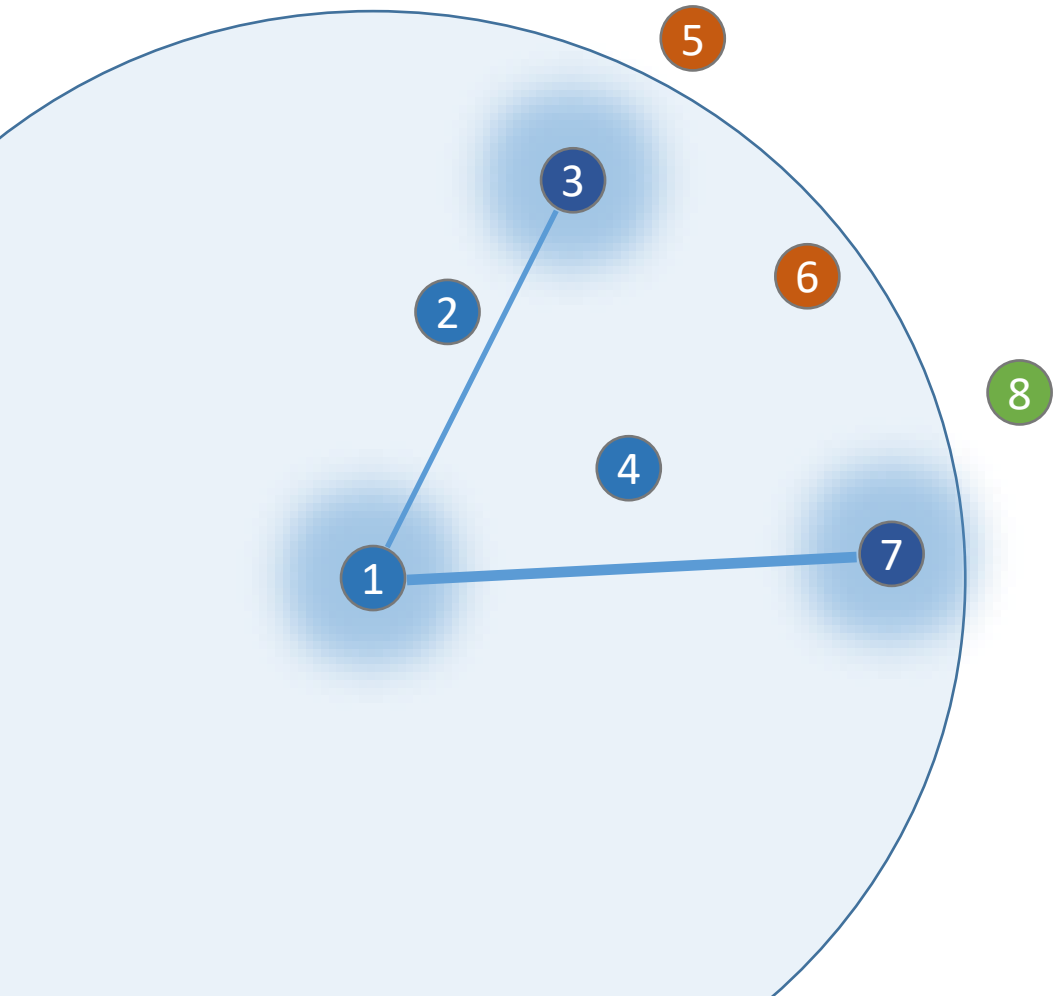


We merge two starting points within 1.5x the predefined radius.

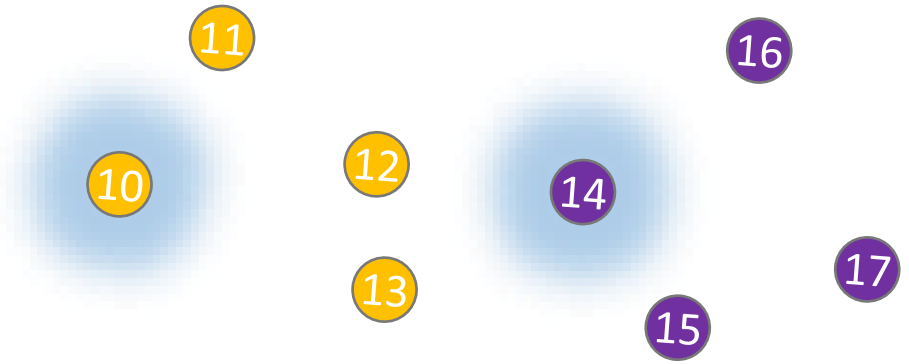


CLASSIX

Fast and Explainable Clustering

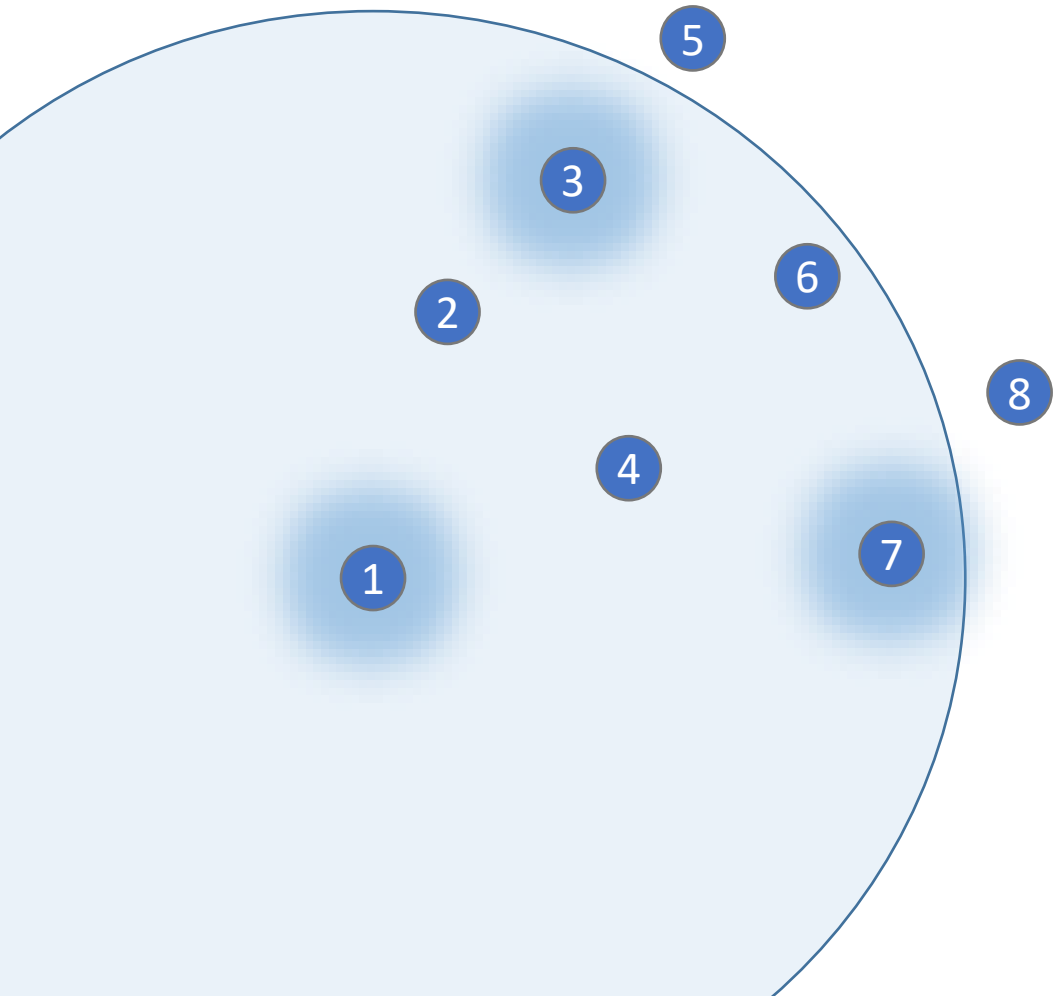


We merge two starting points within 1.5x the predefined radius.

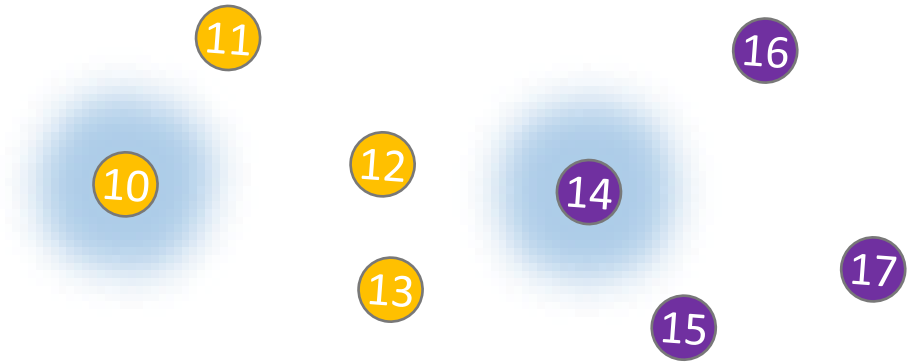


CLASSIX

Fast and Explainable Clustering

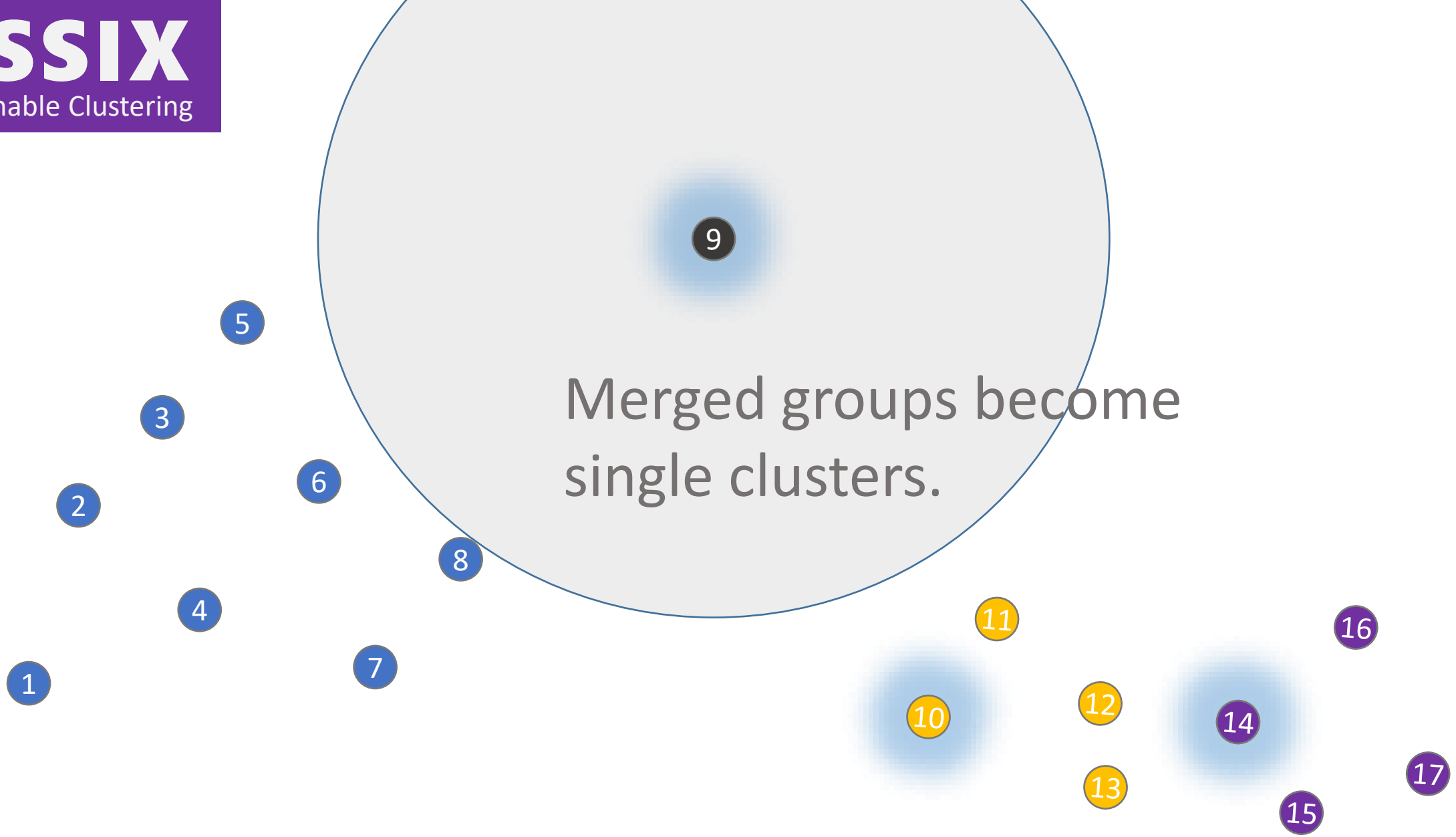


Merged groups become single clusters.



CLASSIX

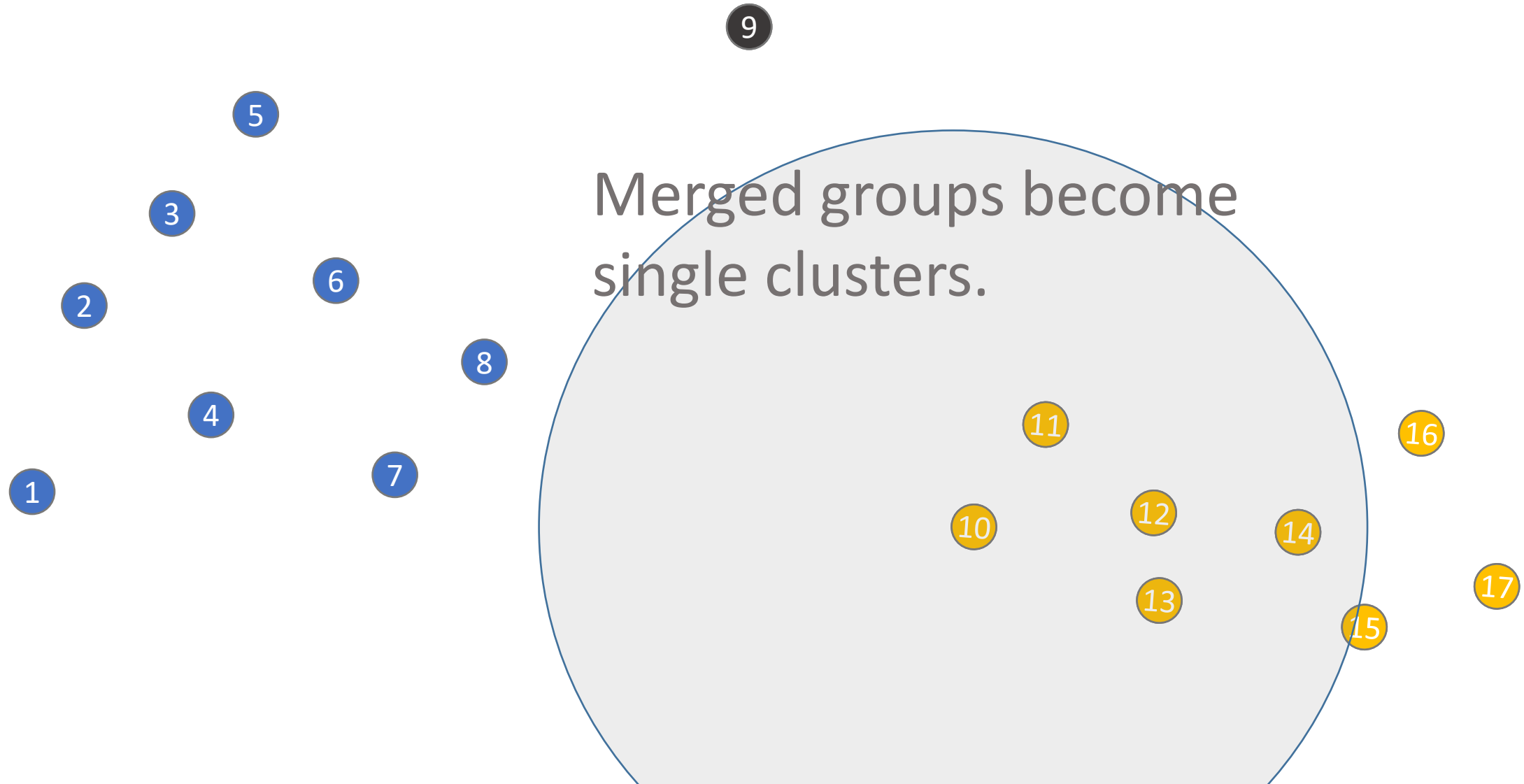
Fast and Explainable Clustering



Merged groups become single clusters.

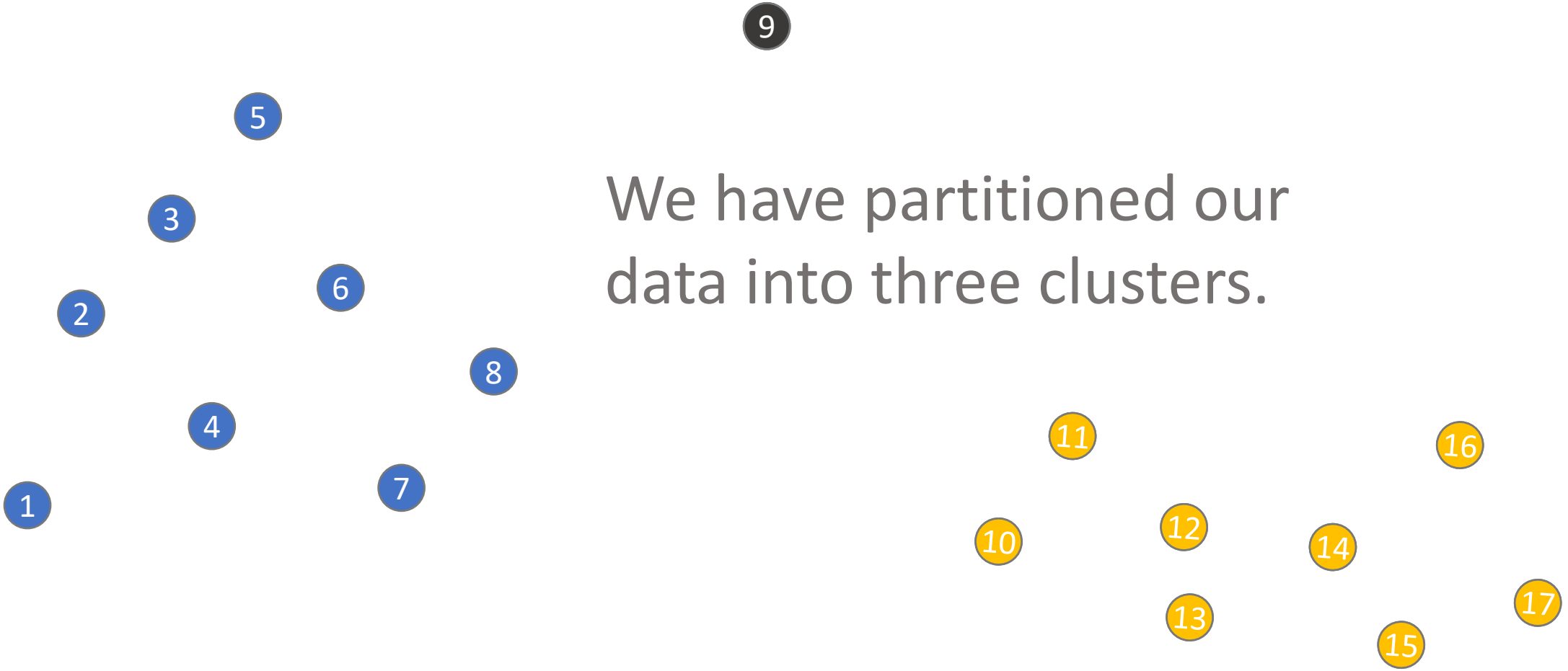
CLASSIX

Fast and Explainable Clustering



CLASSIX

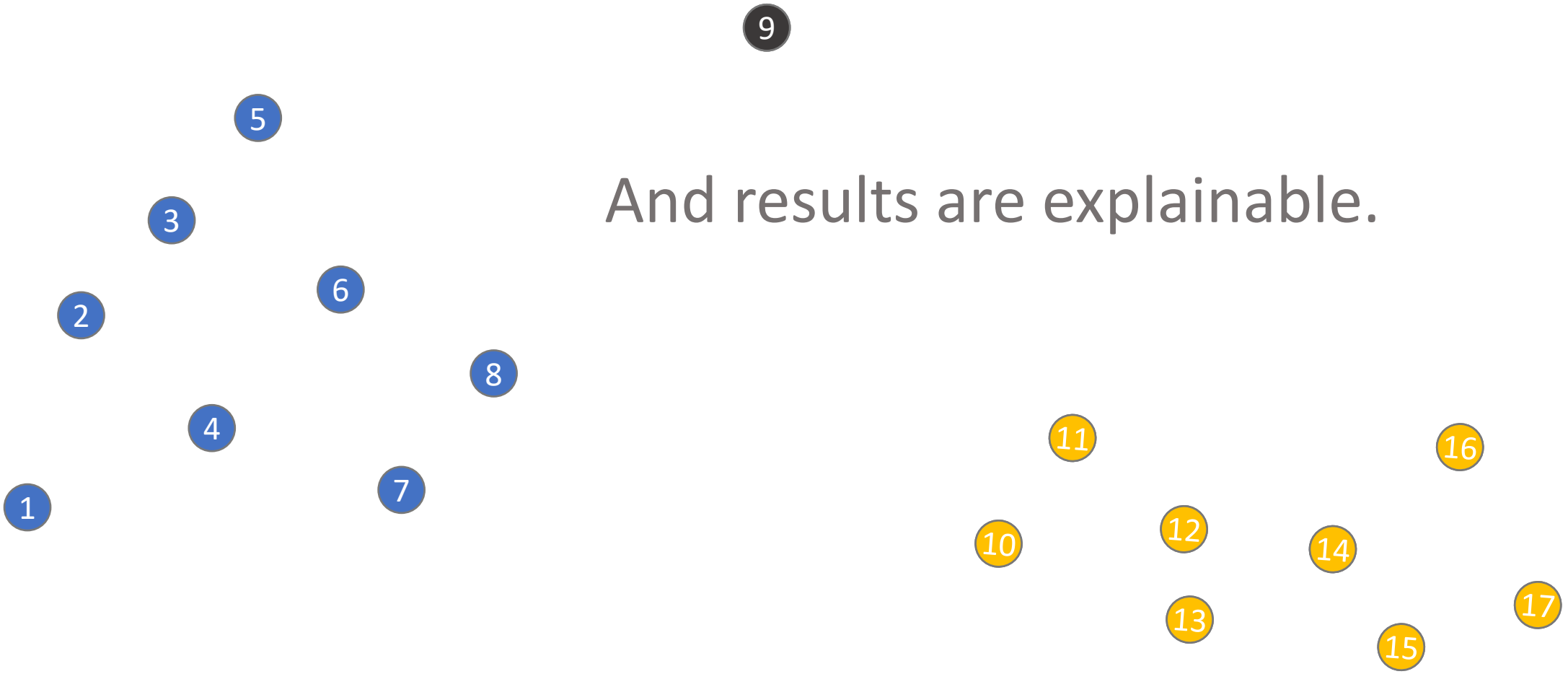
Fast and Explainable Clustering



We have partitioned our data into three clusters.

CLASSIX

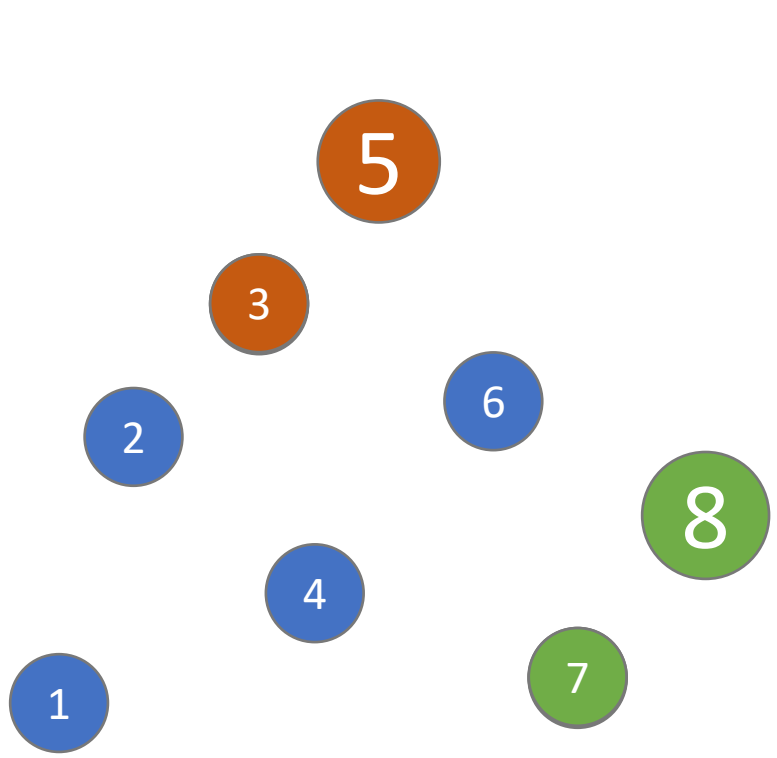
Fast and Explainable Clustering



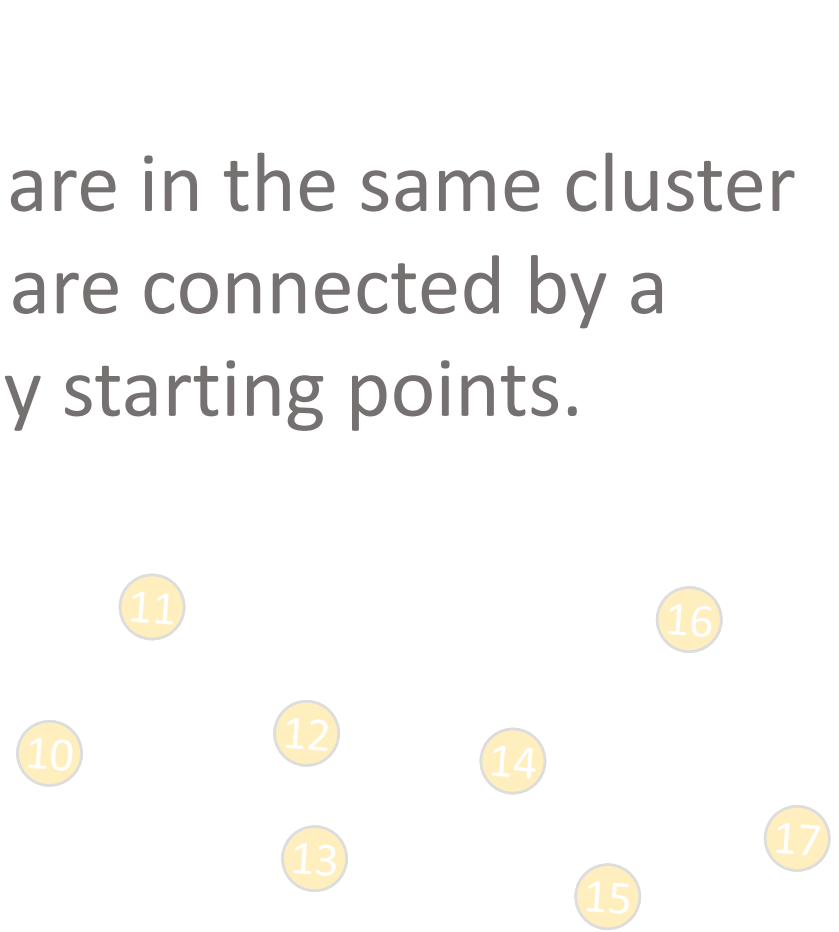
And results are explainable.

CLASSIX

Fast and Explainable Clustering

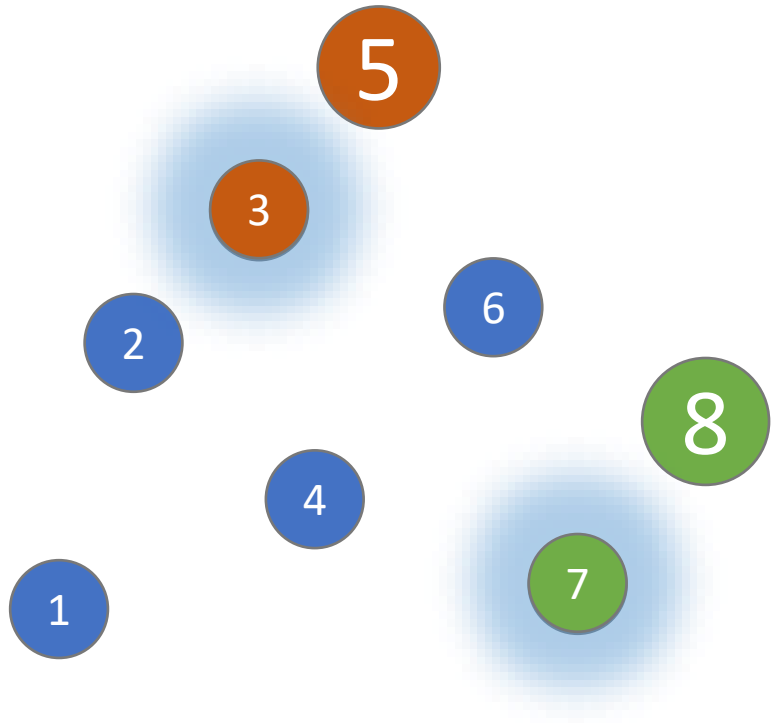


Point 5 and 8 are in the same cluster because they are connected by a path of nearby starting points.



CLASSIX

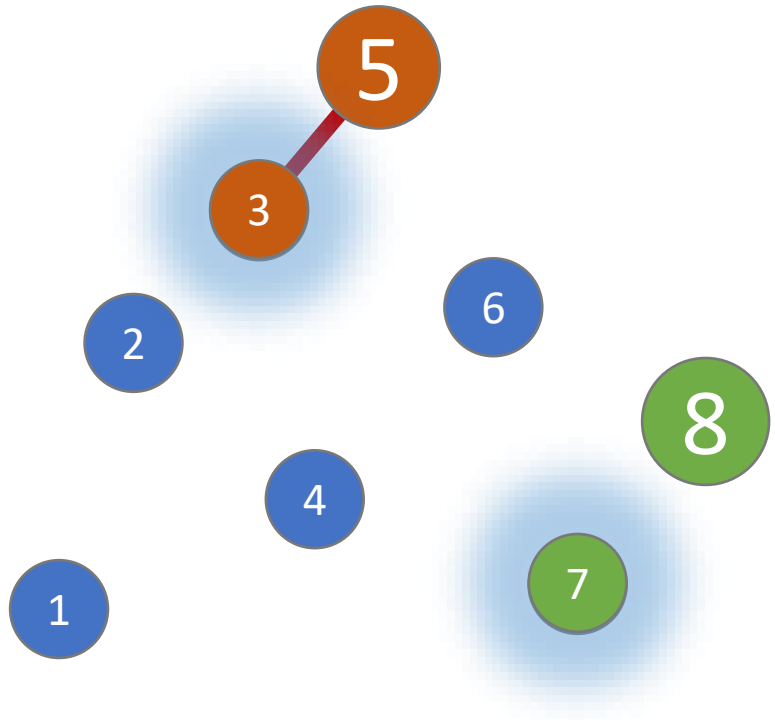
Fast and Explainable Clustering



Point 5 and 8 are in the same cluster because they are connected by a path of nearby starting points.

CLASSIX

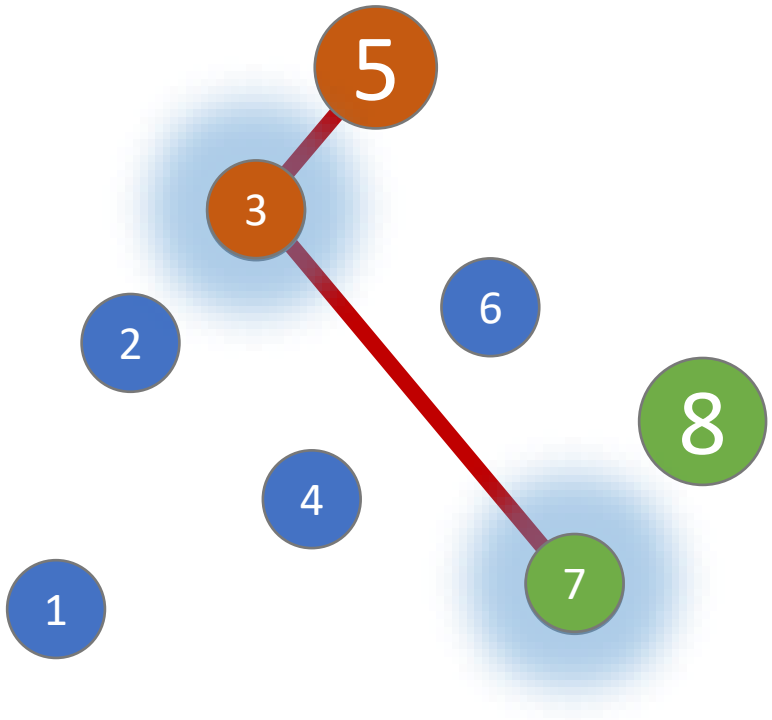
Fast and Explainable Clustering



Point 5 and 8 are in the same cluster because they are connected by a path of nearby starting points.

CLASSIX

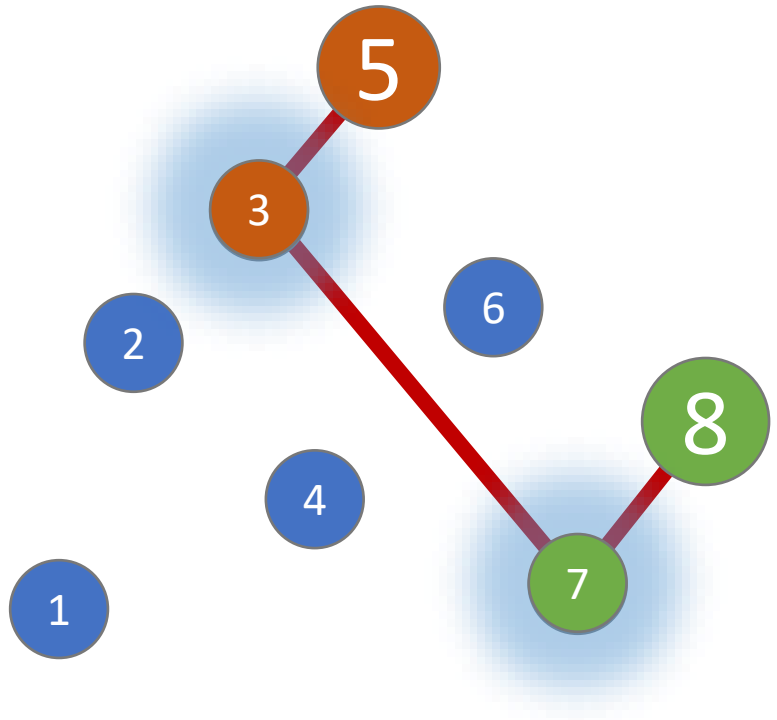
Fast and Explainable Clustering



Point 5 and 8 are in the same cluster because they are connected by a path of nearby starting points.

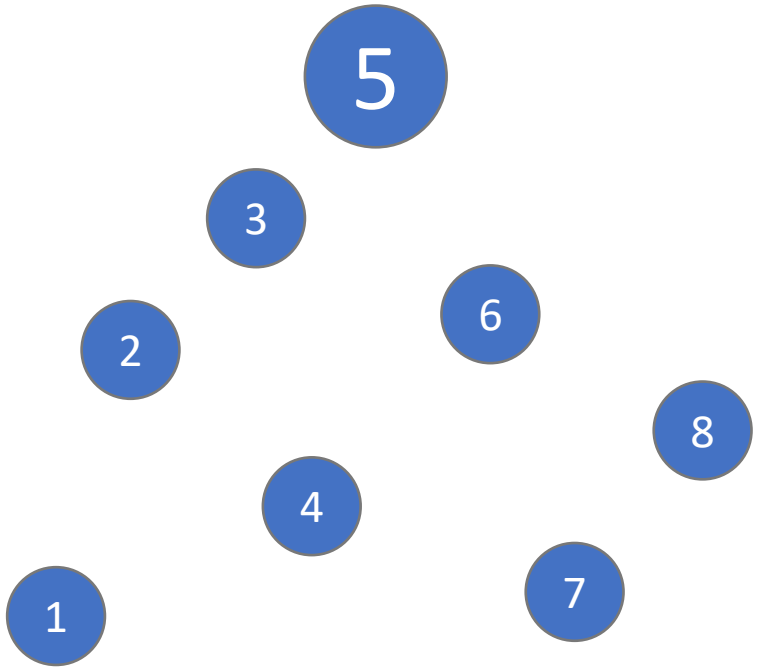
CLASSIX

Fast and Explainable Clustering



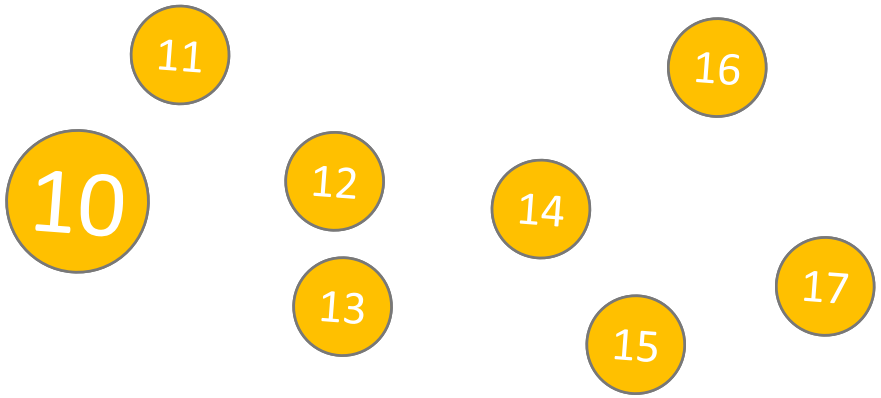
Point 5 and 8 are in the same cluster because they are connected by a path of nearby starting points.

CLASSIX
Fast and Explainable Clustering



9

And there is no such path between points 5 and 10, hence different clusters.

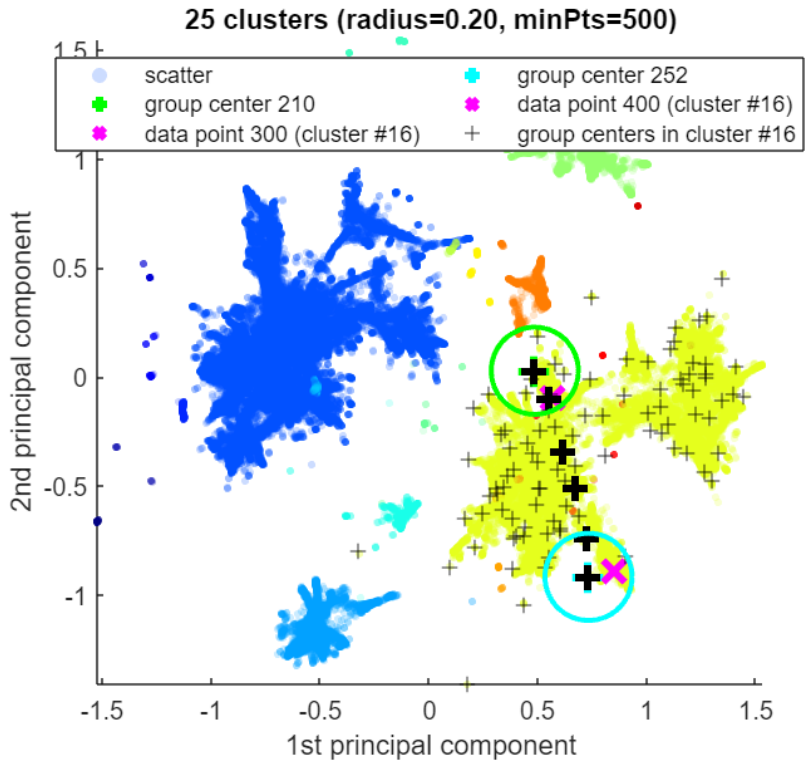


CLASSIX

Fast and Explainable Clustering

MATLAB demo

Clustering 5.7M RNA sequences of coronavirus



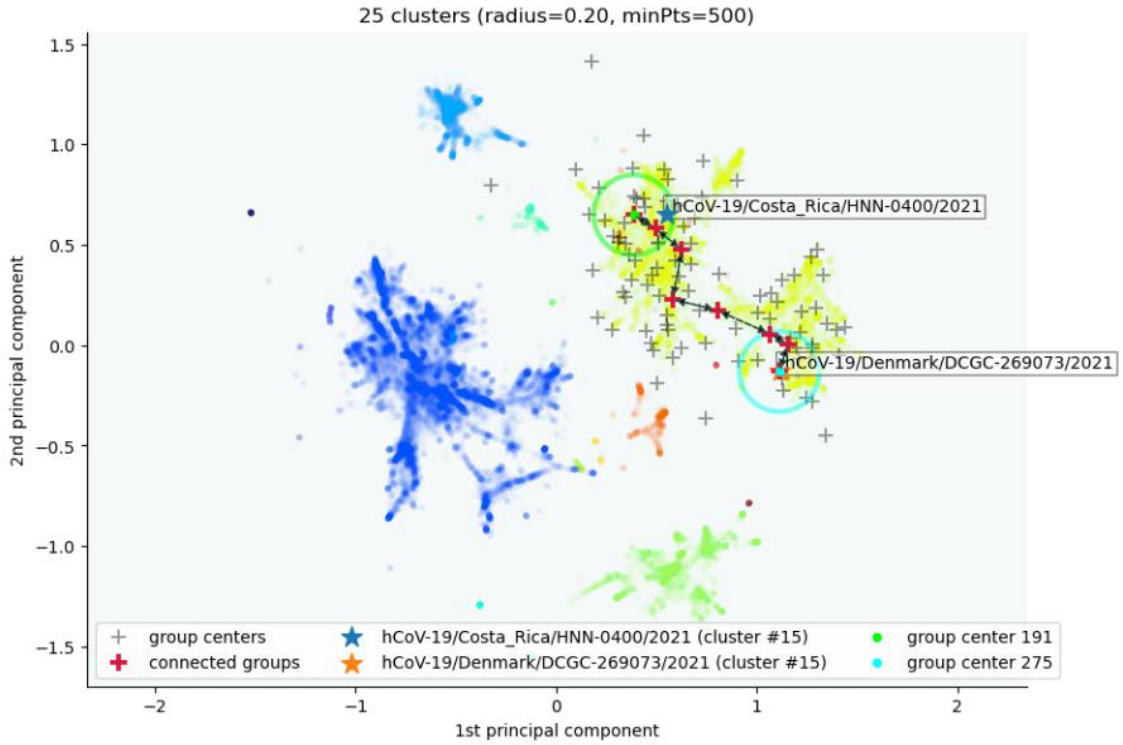
	CLASSIX	DBSCAN (5%)
Parameters	(0.2, 500)	(0.1, 1)
Runtime (s)	6.7	493
Clusters	25	39
AMI	0.61	0.60

https://github.com/nla-group/classix-matlab/blob/main/demos/Clustering_RNA_of_coronavirus.ipynb

CLASSIX
Fast and Explainable Clustering

Same in Python

Clustering 5.7M RNA sequences of coronavirus




```
Data point hCoV-19/Costa_Rica/HNN-0400/2021 is in group 191.
Data point hCoV-19/Denmark/DCGC-269073/2021 is in group 275.
Both groups were merged into cluster #15.

The two groups are connected via groups
 191 <-> 210 <-> 239 <-> 230 <-> 258 <-> 272 <-> 279 <-> 275.
```

	CLASSIX.py	HDBSCAN
Parameters	(0.2, 500)	(180000, 5)
Runtime (s)	7.7	4080
Clusters	25	4
AMI	0.61	0.59

https://github.com/nla-group/classix/blob/master/demos/Clustering_RNA_of_coronavirus.ipynb

CLASSIX
Fast and Explainable Clustering

MATLAB vs Python

Timing comparison on URI machine learning datasets

	Dim	Size	#Classes	CLASSIX.m	CLASSIX.mex	CLASSIX.py
Banknote	4	1372	2	0.044	0.031	0.078
Dermatology	34	366	6	0.019	0.017	0.047
Ecoli	7	336	7	0.015	0.012	0.028
Glass	9	214	26	0.009	0.009	0.013
Iris	4	150	4	0.008	0.007	0.010
Seeds	7	210	3	0.013	0.010	0.029
Wine	13	178	2	0.011	0.010	0.020
Phoneme	256	4509	4	20.861	6.195	5.369
VDU Signals	2	2028780	11	1.028	1.034	2.860

https://github.com/nla-group/classix-matlab/blob/main/demos/Comparing_MATLAB_and_Python_runtimes.ipynb

How did we get there?

A tale of developing the same algorithm in two languages simultaneously

Step 0: Original version of CLASSIX

Developed in Python with PhD student Xinye Chen between 2021–2022

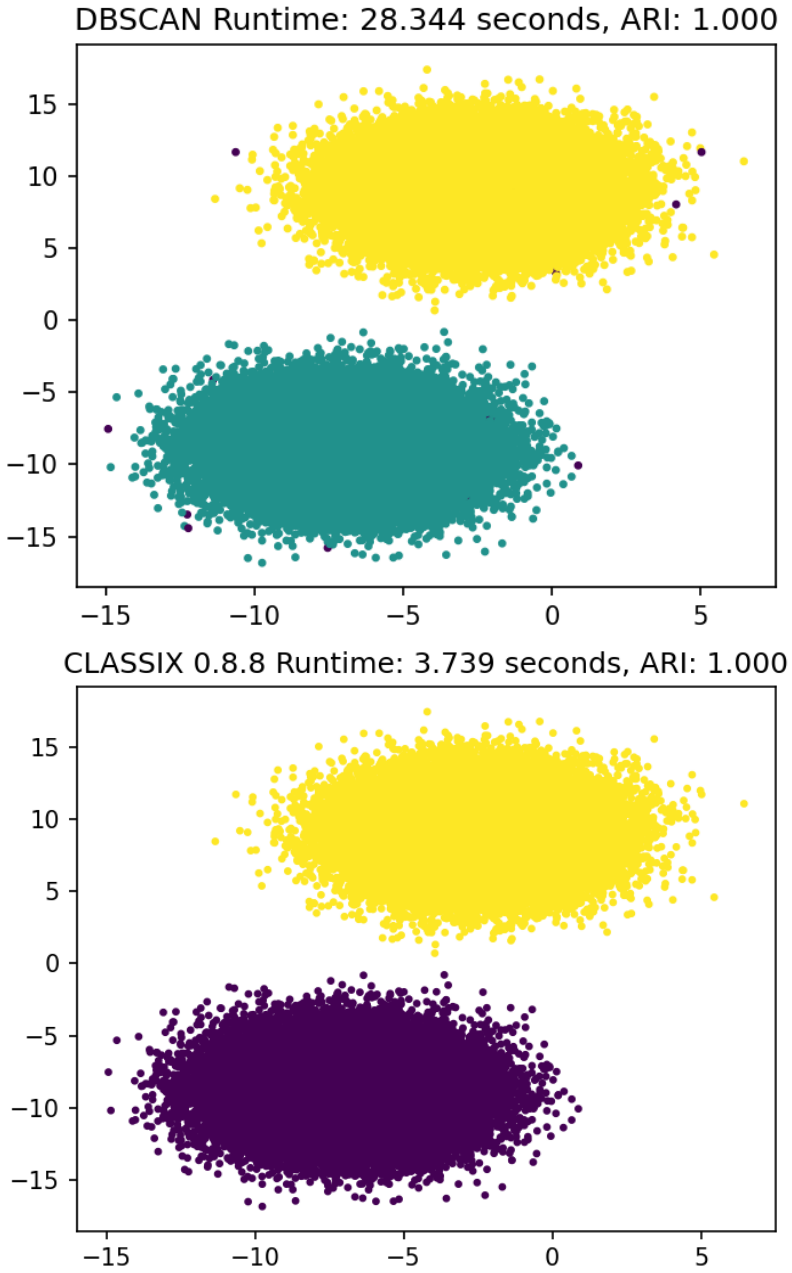
First arXiv preprint in February 2022 and GitHub release

Used inefficient disjoint set data structure for keeping track of clusters

Still significantly faster than e.g. DBSCAN

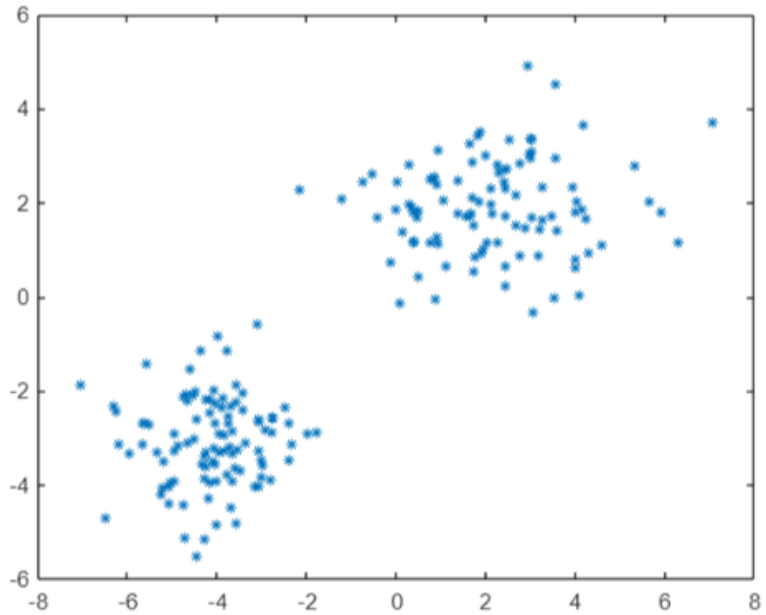
Step 0: Original version of CLASSIX

```
print('DBSCAN (sklearn 1.5.2)')
st = time()
clustering = DBSCAN(eps=3, min_samples=5).fit(X)
print('  Runtime:', time()-st, 'seconds')
print('  ARI:    ', ari(clustering.labels_, y))



print('CLASSIX.py version 0.8.8')
st = time()
clx = CLASSIX(radius=0.2, minPts=5)
clx.fit(X)
print('  Runtime:', time()-st, 'seconds')
print('  ARI:    ', ari(clx.labels_, y))
```



Step 1: Using interoperability to run CLASSIX.py in MATLAB

```
rng('default')        % For reproducibility
mu1 = [2 2];          % Mean of the 1st cluster
sigma1 = [2 0; 0 1];  % Covariance of the 1st cluster
mu2 = [-4 -3];        % Mean of the 2nd cluster
sigma2 = [1 0; 0 1];  % Covariance of the 2nd cluster
r1 = mvnrnd(mu1,sigma1,100);
r2 = mvnrnd(mu2,sigma2,100);
X = [r1; r2];
plot(X(:,1),X(:,2),"*",MarkerSize=5);
```



Calling CLASSIX is straightforward. We don't even need to convert the MATLAB array X to a Numpy array as it's all done automatically.

```
clx = py.classix.CLASSIX(radius=0.3, verbose=0);
clx = clx.fit(X);
clx.explain(plot=false);
```

```
CLASSIX clustered 200 data points with 2 features.
The radius parameter was set to 0.30 and minPts was set to 0.
```

[MATLAB Meets Python: Amplifying Research Impact with Cross-Platform Integration - MATLAB (mathworks.com)]

Step 2: Write native MATLAB version based on Python original

Profiler

PROFILER

Print | Profile Summary | Back | Forward | Find... | Highlight | Enter code to run and time | Start Profiling

FILE | NAVIGATE | SEARCH | VIEW | PROFILE

Parents (calling functions)

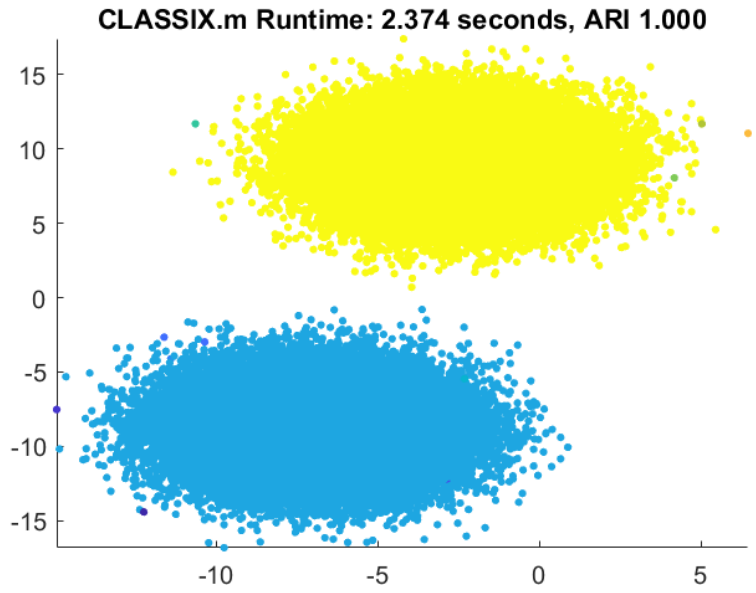
Function Name	Function Type	Calls
classix_mike>classix_mike.fast_agglomerate	Class method	30566

Lines that take the most time

Line Number	Code	Calls	Total Time (s)	% Time	Time Plot
25	`parent_of_s2 = findParent(s2);`	30566	0.205	49.6%	
24	`parent_of_s1 = findParent(obj);`	30566	0.146	35.4%	
30	`end`	30566	0.038	9.2%	
27	`if (parent_of_s1.data ~= parent_of_s2.data)`	30566	0.002	0.5%	
28	`parent_of_s1.parent = parent_of_s2;`	3325	0.002	0.5%	
All other lines			0.020	4.9%	
Totals			0.414	100%	

Children (called functions)

Function Name	Function Type	Calls	Total Time (s)	% Time	Time Plot
SET>SET.findParent	Class method	61132	0.320	77.3%	
Self time (built-ins, overhead, etc.)			0.094	22.7%	



Large amount of time spent on disjoint set structure operations!

Step 2: Write native MATLAB version based on Python original

Profiler

PROFILER

Print | Profile Summary | Back | Forward | Find... | Highlight | Enter code to run and time | Start Profiling

FILE | NAVIGATE | SEARCH | VIEW | PROFILE

Parents (calling functions)

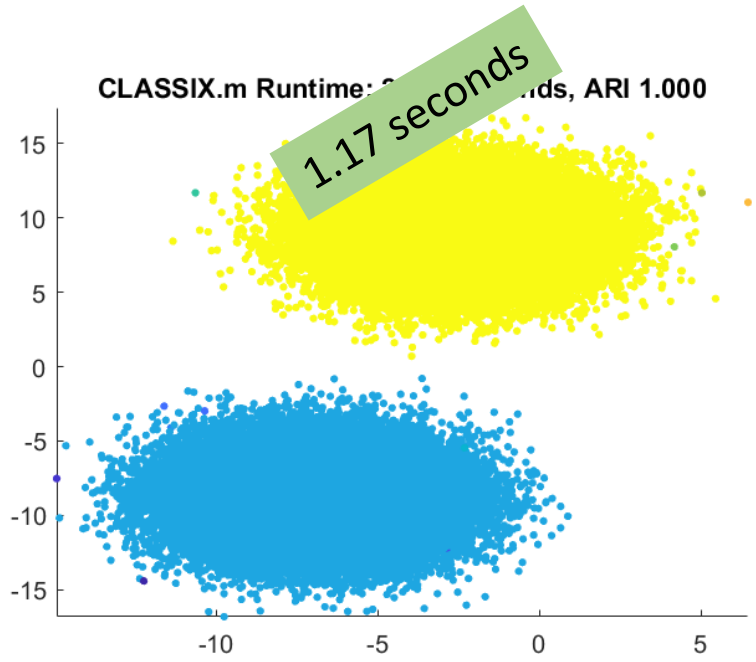
Function Name	Function Type	Calls
classix_mike>classix_mike.fast_agglomerate	Class method	30566

Lines that take the most time

Line Number	Code	Calls	Total Time (s)	% Time	Time Plot
25	`parent_of_s2 = findParent(s2);`	30566	0.205	49.6%	
24	`parent_of_s1 = findParent(obj);`	30566	0.146	35.4%	
30	`end`	30566	0.038	9.2%	
27	`if (parent_of_s1.data ~= parent_of_s2.data)`	30566	0.002	0.5%	
28	`parent_of_s1.parent = parent_of_s2;`	3325	0.002	0.5%	
All other lines			0.020	4.9%	
Totals			0.414	100%	

Children (called functions)

Function Name	Function Type	Calls	Total Time (s)	% Time	Time Plot
SET>SET.findParent	Class method	61132	0.320	77.3%	
Self time (built-ins, overhead, etc.)			0.094	22.7%	



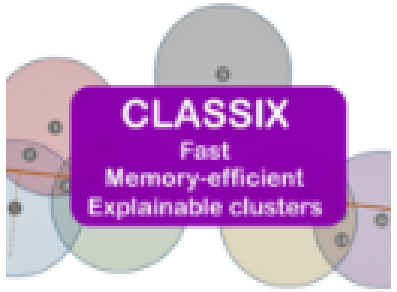
Large amount of time spent on disjoint set structure operations!

MATLAB profiler helped us to improve both MATLAB and Python versions

Step 3: Use insights to rewrite both Python and MATLAB versions

We used what we learned in Python to improve Mike's MATLAB code and got another significant speed-up.

The MATLAB version is currently faster than the Python package



Fast and explainable clustering with CLASSIX

Version 1.3 (23.8 MB) by Stefan Güttel

A fast and easy-to-use clustering method that provides explanations for the computed clusters.
https://github.com/nla-group/classix-matlab

⊕ Follow

CLASSIX.py v0.8.8 3.74 seconds	CLASSIX.m v0.1 2.37 seconds	CLASSIX.m v1.0 1.17 seconds	CLASSIX.py v1.0.0 3.14 seconds	CLASSIX.m v1.3 0.85 seconds	CLASSIX.py v1.2.5 0.94 seconds
	not using minPts			using mex for submatrix-vector product	

Step 4: Use this work to help make a faster MATLAB (R2024b)

```
Bcols = 100000;
A = rand(1,1000);
B = randn(1000,Bcols);
tic;C0 = A*B; t0=toc;
fprintf("Timing full A*B: %.3e seconds\n",t0);
```

```
Timing full A*B: 3.054e-02 seconds
```

```
tic;C1=A*B(:,1:Bcols/2);t1=toc;
```

```
fprintf("Timing A*B using only half of the columns of B: %.3e seconds\n",t1);fprintf("It is %.3fx faster to compute the full product",t1/t0)
```

```
Timing A*B using only half of the columns of B: 2.176e-01 seconds
It is 7.126x faster to compute the full product
```

```
tic;C2 = [redacted](A,'none',B,'none',1,Bcols/2);t3=toc;
fprintf("Using mtimesColumns on the first half of the matrix B is %.3fx faster than doing the full product\n",t0/t3);
```

```
Using mtimesColumns on the first half of the matrix B is 1.982x faster than doing the full product
```

This is an undocumented function and is a proof of concept that will probably change in the future!

CLASSIX
Fast and Explainable Clustering

- Make a Python package available to MATLAB users using interoperability
- Rewrite the Python code as MATLAB code
- Iterate between Python package and MATLAB toolbox, using insights from one to drive improvements in the other. New algorithm is faster in both languages than it would have been otherwise
- BONUS: MATLAB itself is improved a little

https://github.com/nla-group/classix-matlab

Paper: X. Chen & S. Güttel. Fast and explainable clustering based on sorting. Pattern Recognition, 150: 110298, 2024.