Tackling Noise in Active Semi-Supervised Clustering

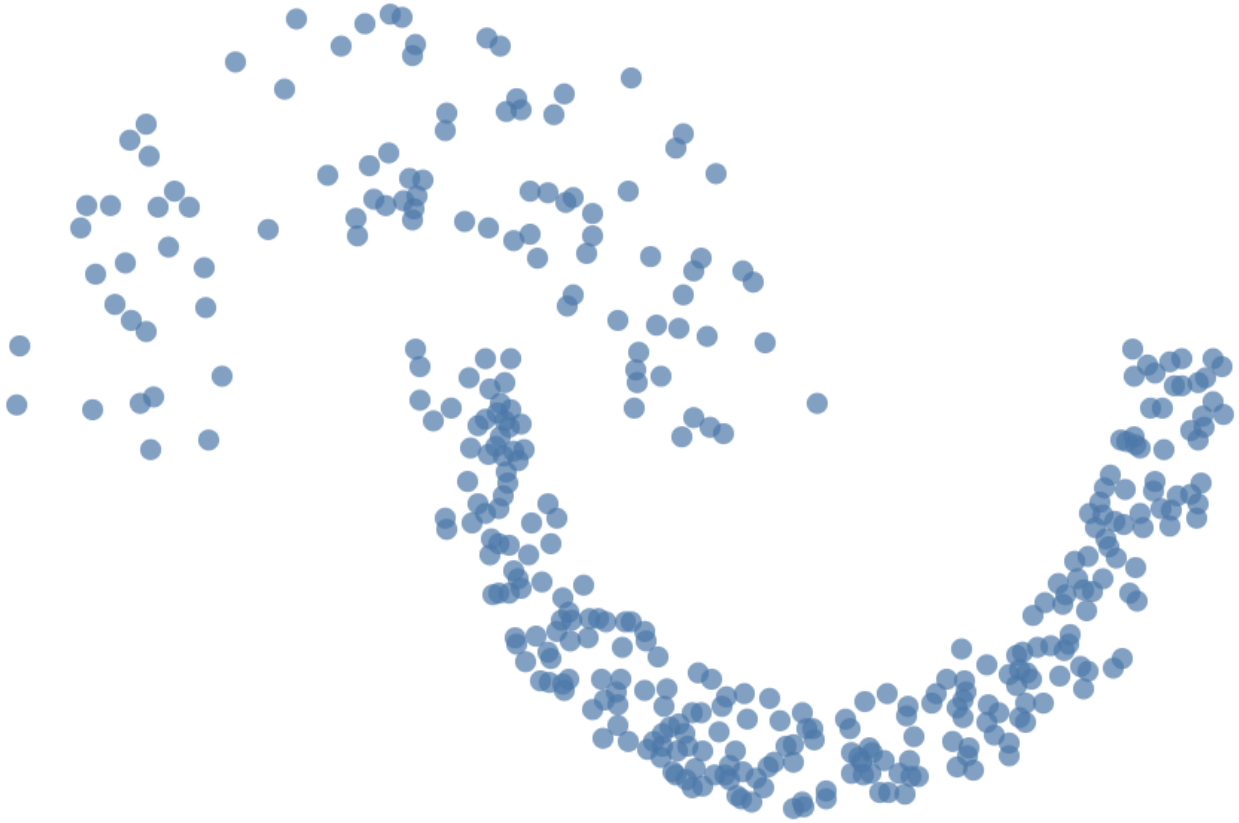
Jonas Soenen

Unsupervised clustering is not enough

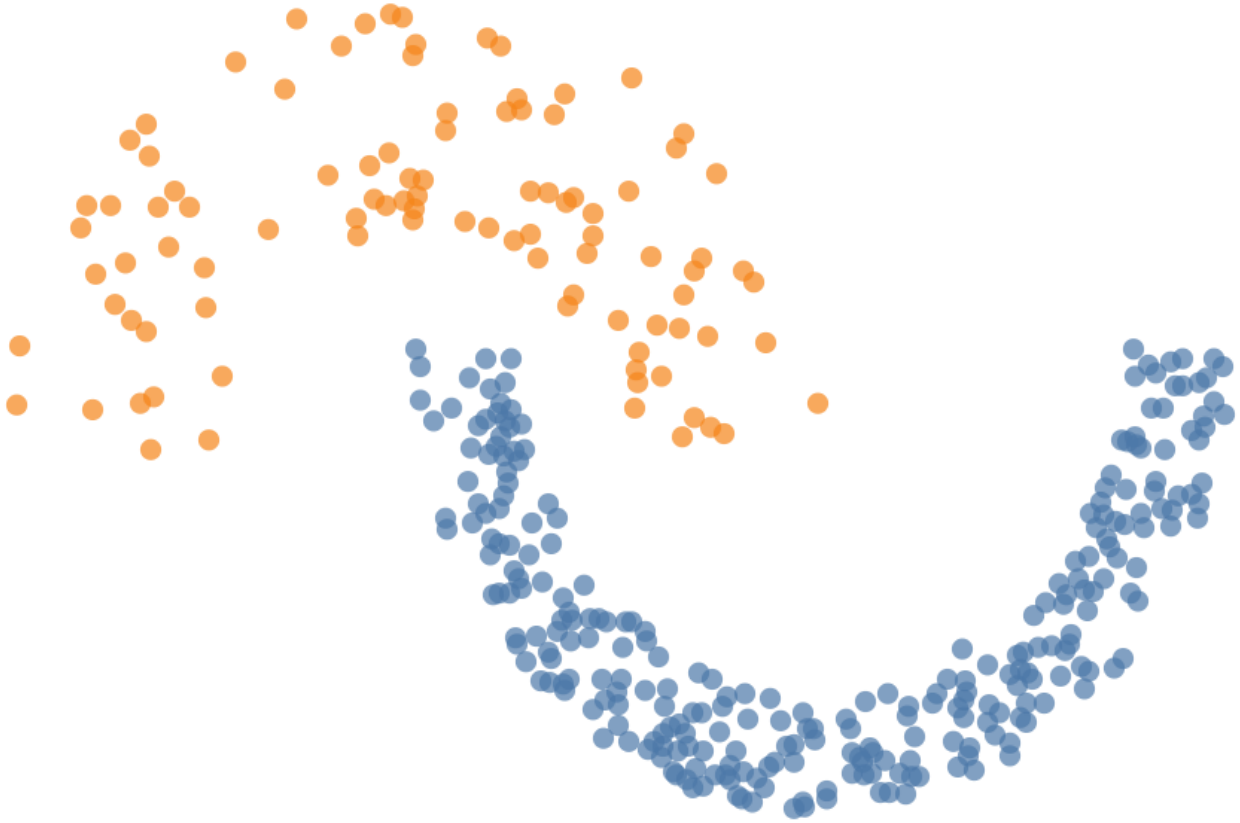
Semi-supervised, Active and Robust clustering

Unsupervised clustering is not enough

Goal: group instances into clusters

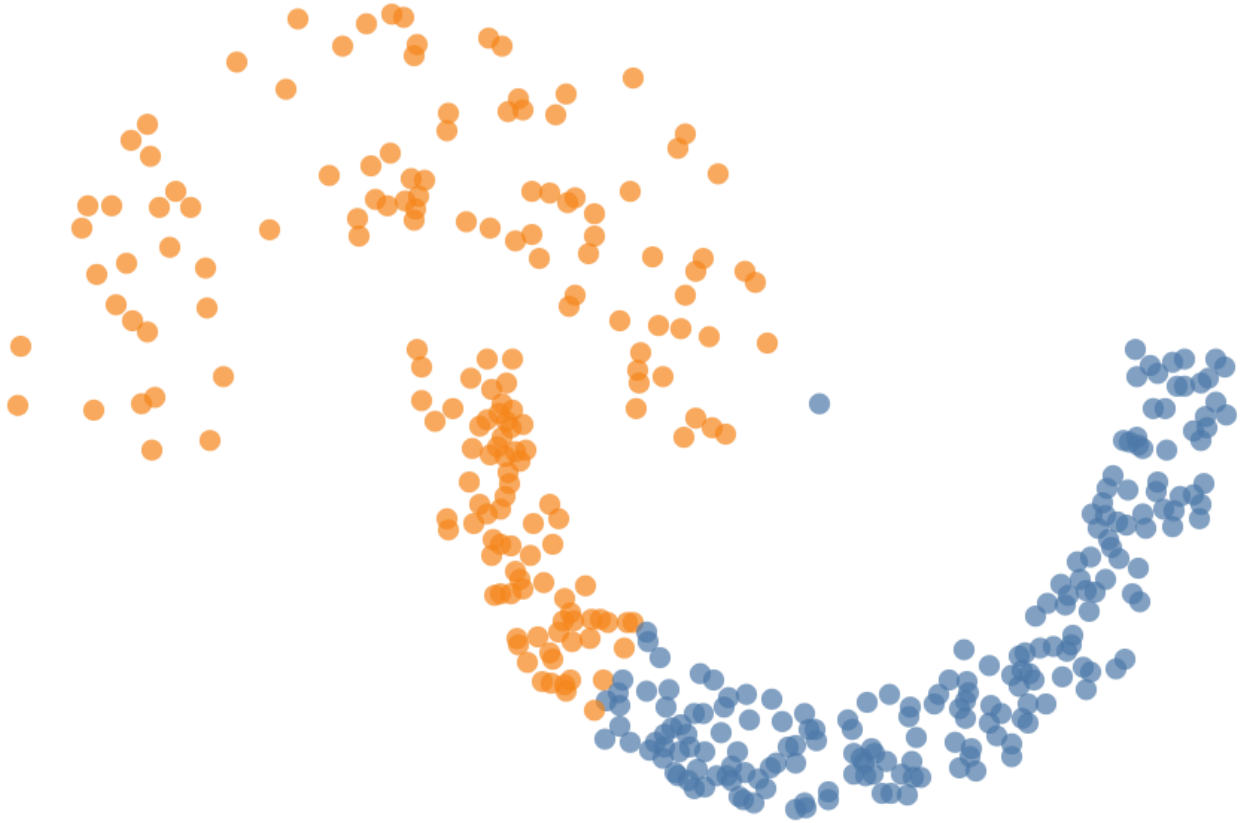


Goal: group instances into clusters



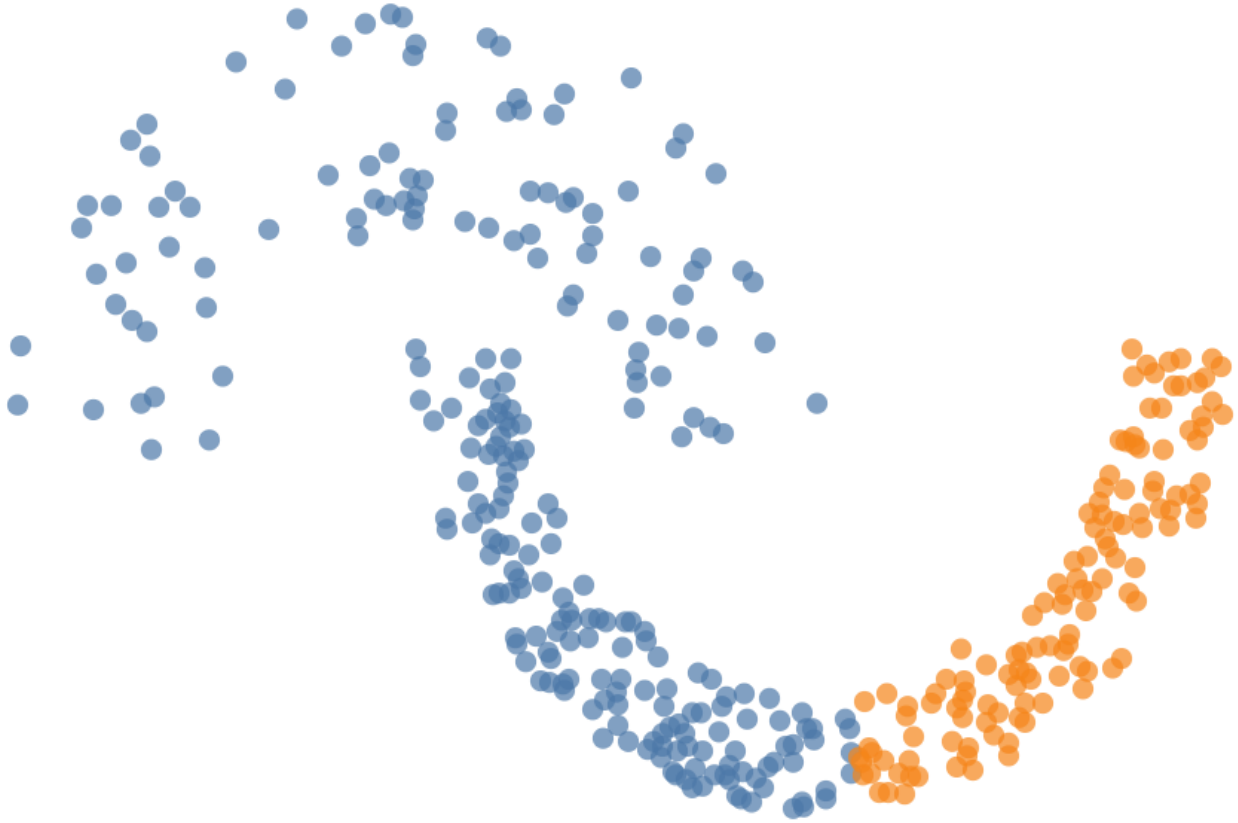
Instances from the same cluster are ‘similar’
Instances from different clusters are ‘different’

Getting the correct clustering is not always straightforward



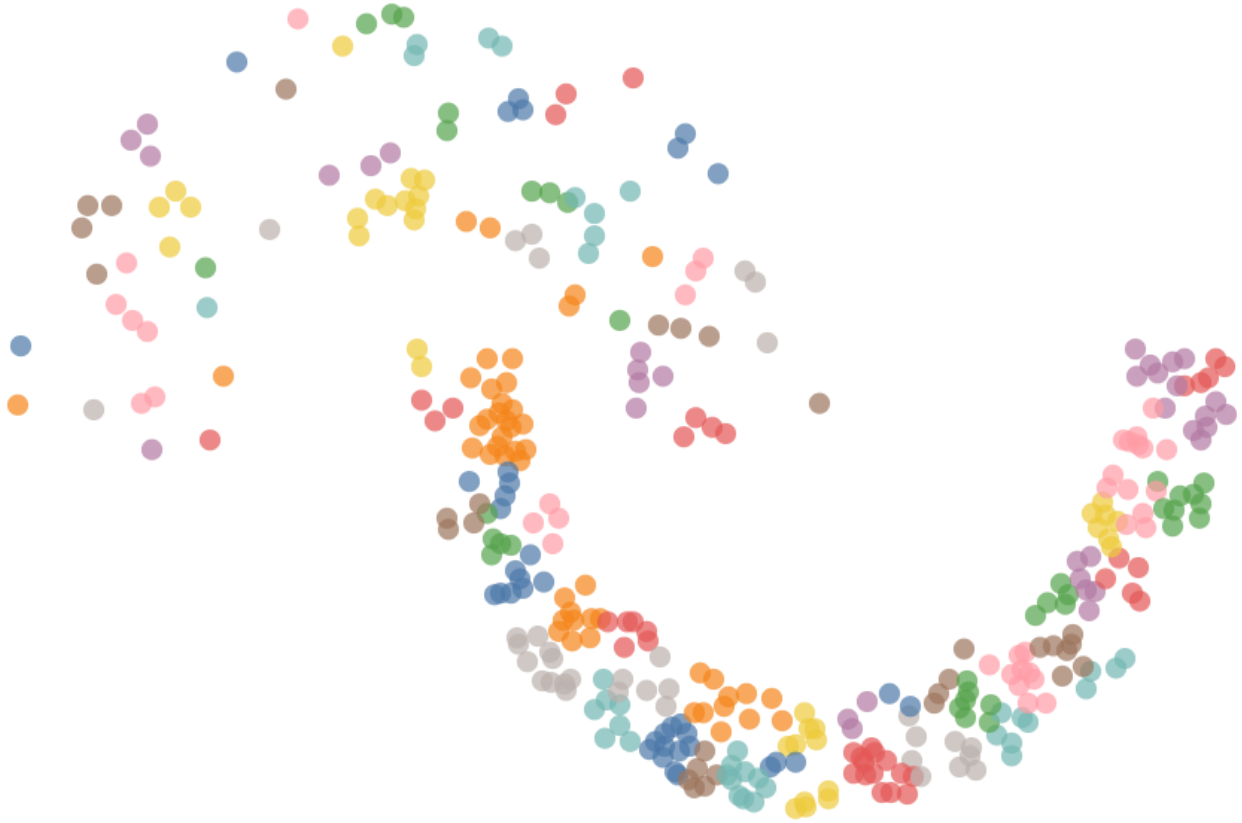
K-means(k = 2)

Getting the correct clustering is not always straightforward



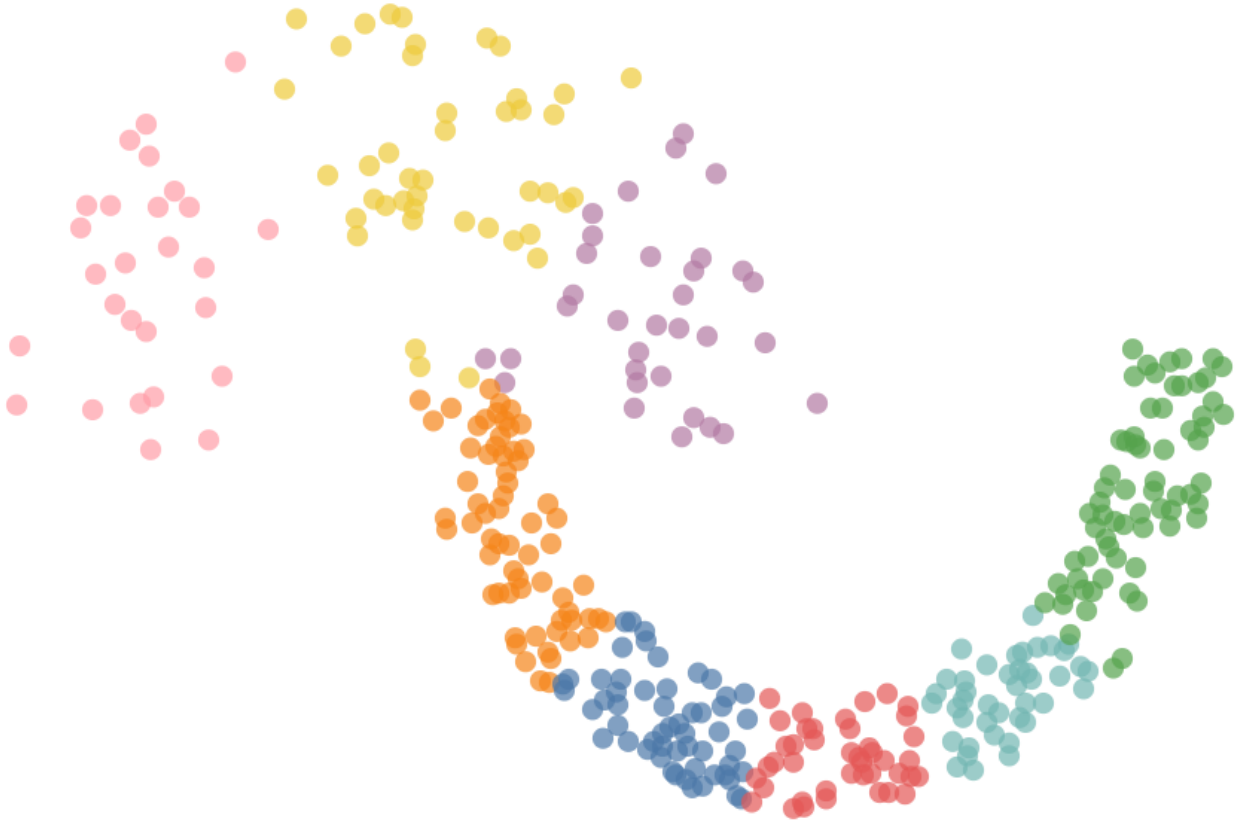
MeanShift()

Getting the correct clustering is not always straightforward



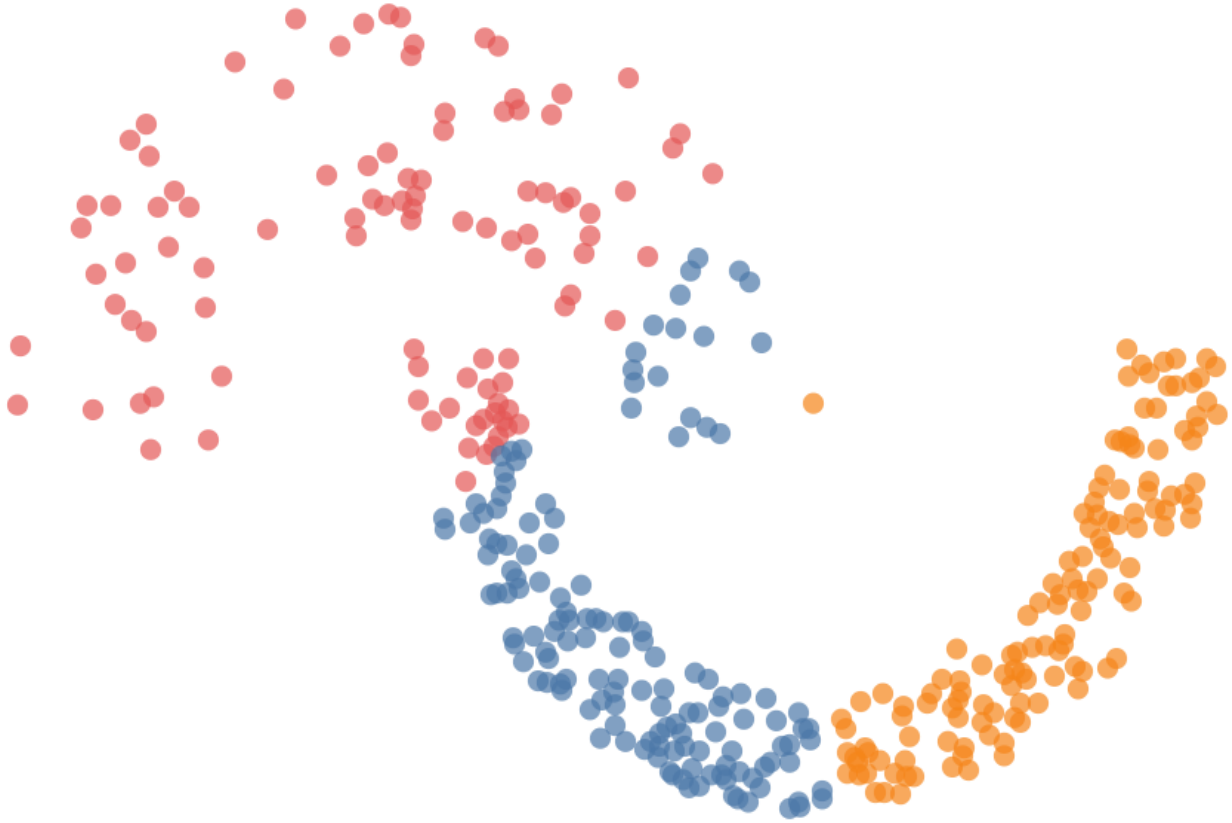
MeanShift(bandwidth = 1)

Getting the correct clustering is not always straightforward



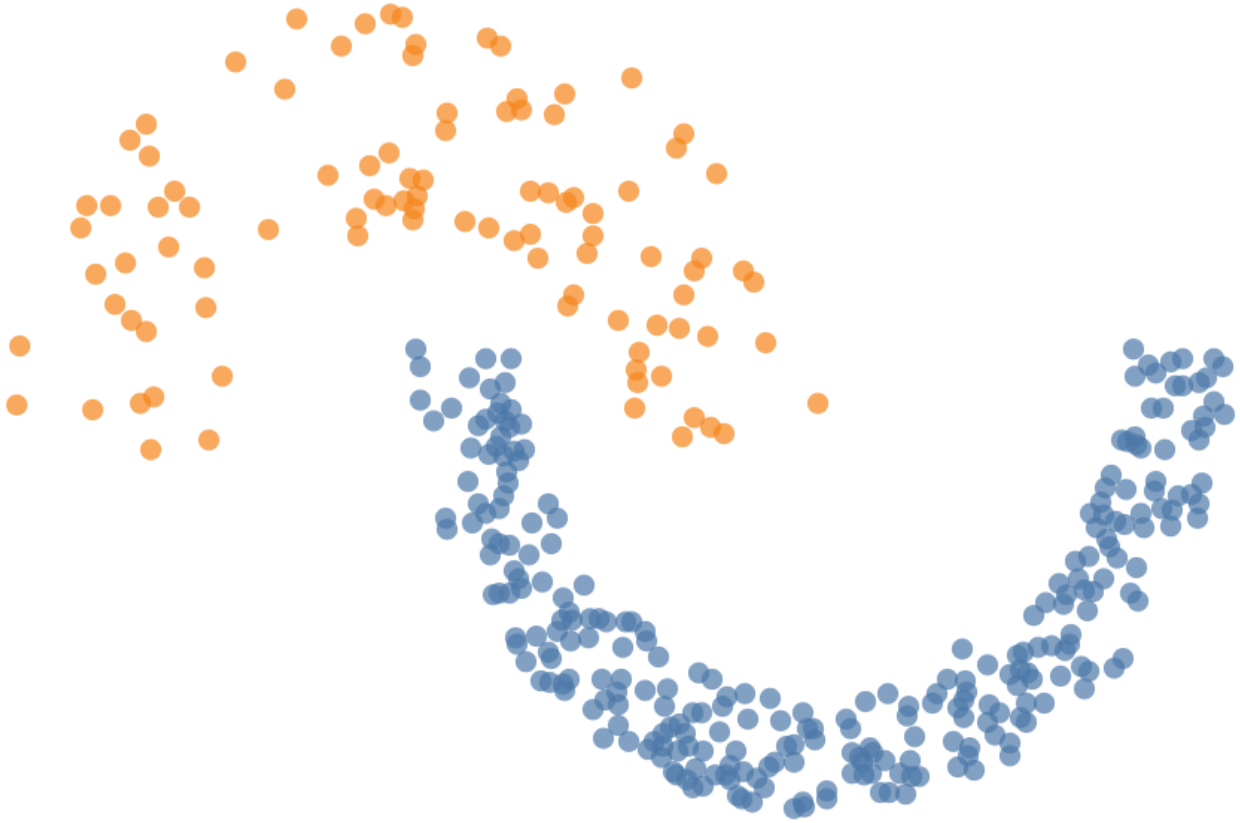
MeanShift(bandwidth = 5)

Getting the correct clustering is not always straightforward



MeanShift(bandwidth = 7)

Getting the correct clustering is not always straightforward



SpectralClustering(k=2)

Images: CMU faces dataset

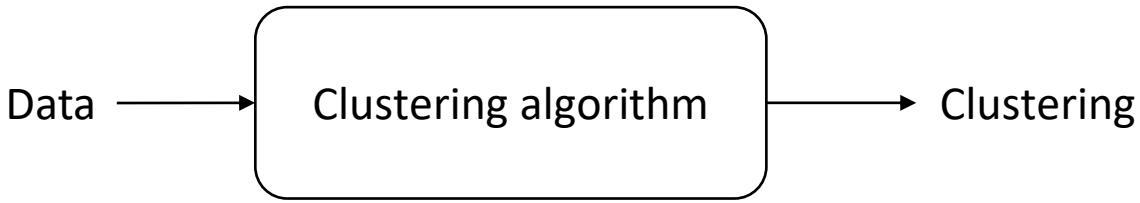
Multiple clusterings might exist for the same dataset

Expression

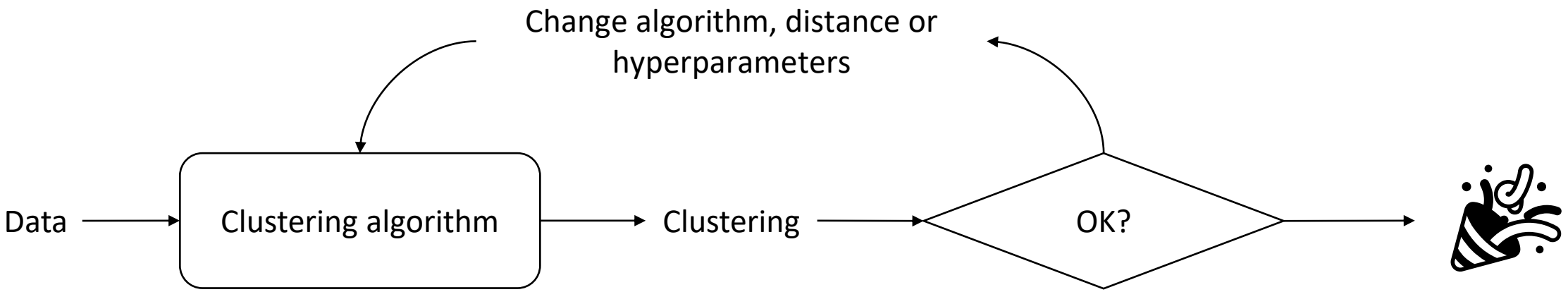
Identity

How will an unsupervised algorithm know which one you want?

Images: CMU faces dataset

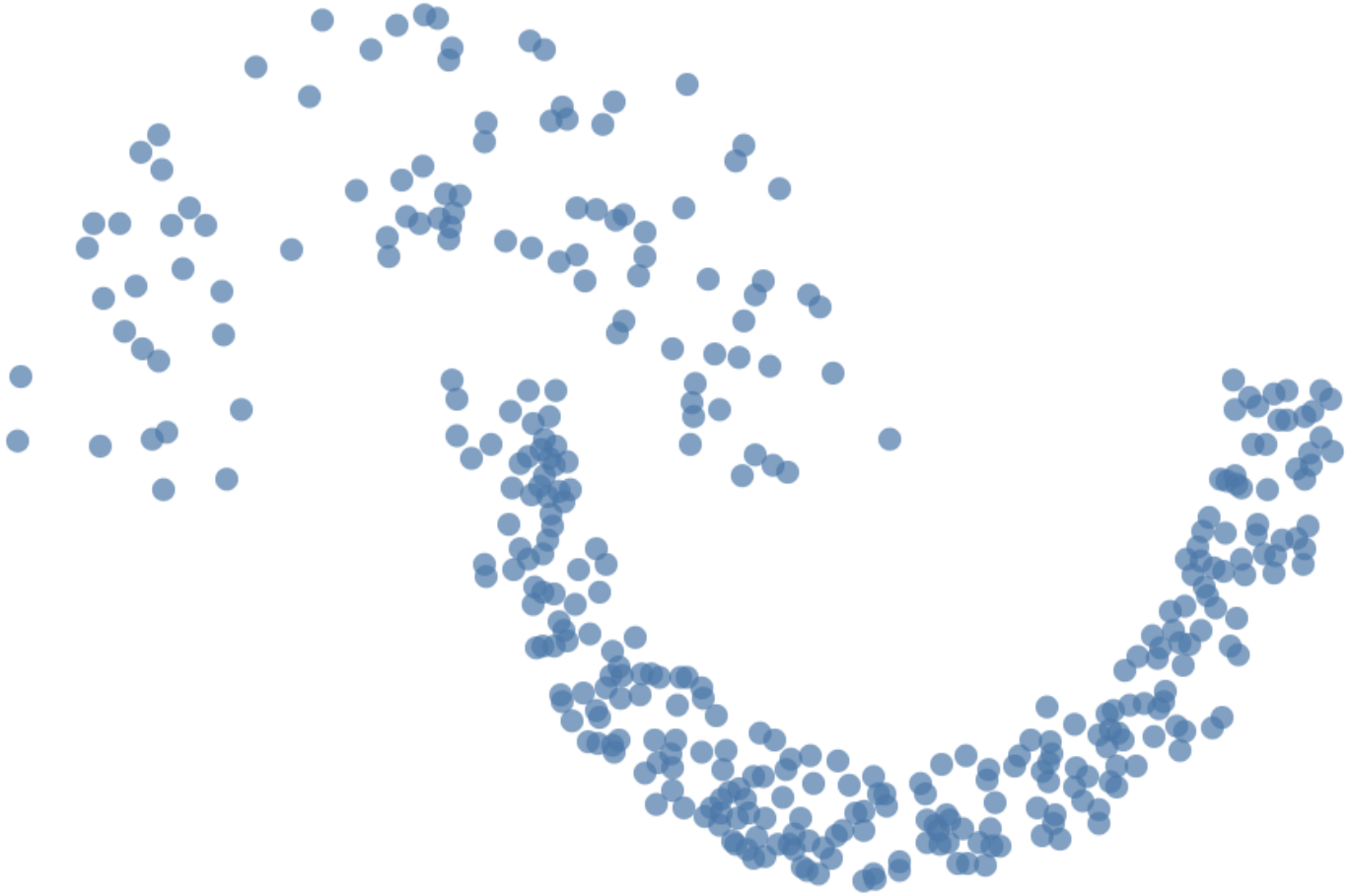

Data
Clustering algorithm
Clustering

Unsupervised clustering requires **trial and error** and **expertise**

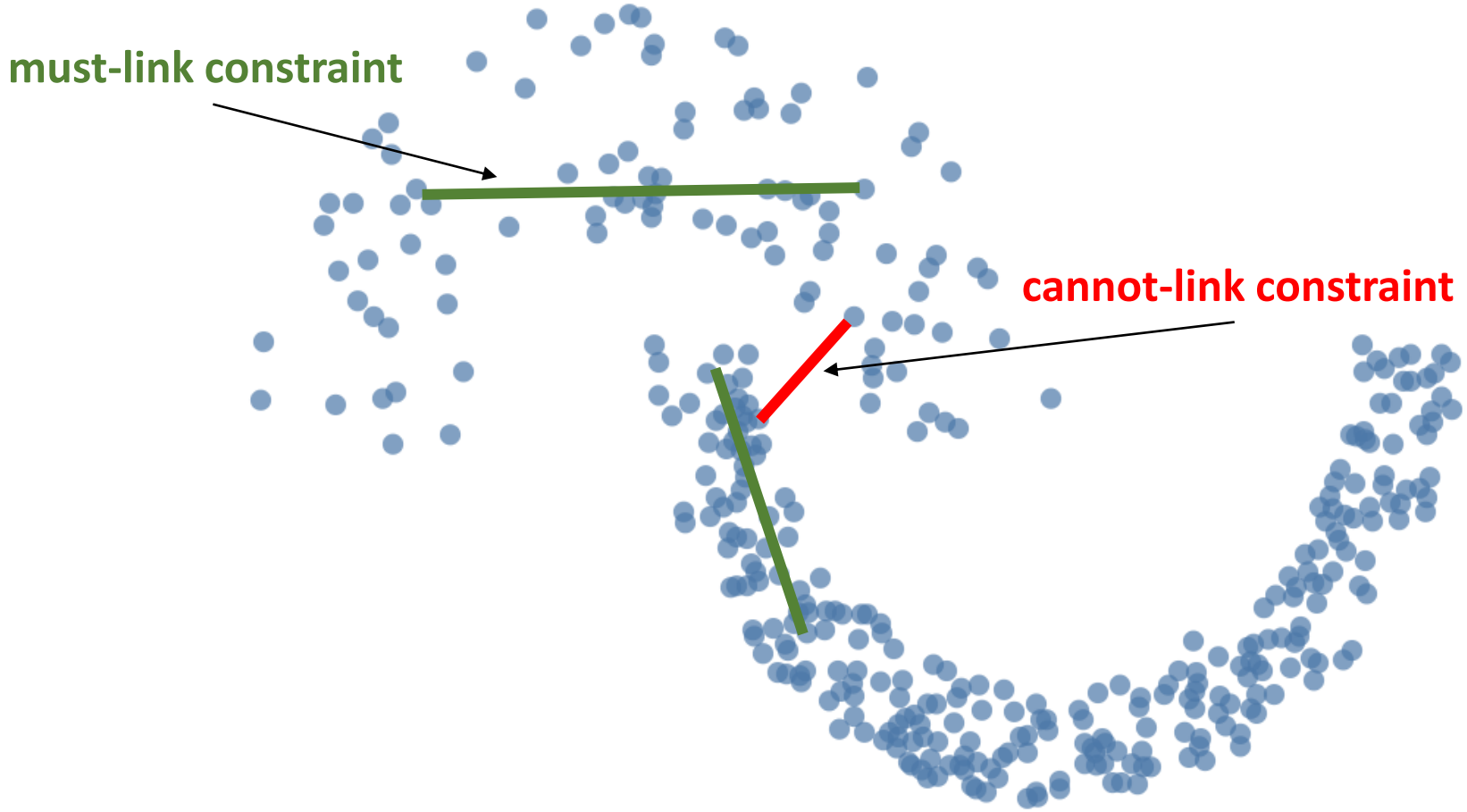


Semi-Supervised, Active and Robust clustering

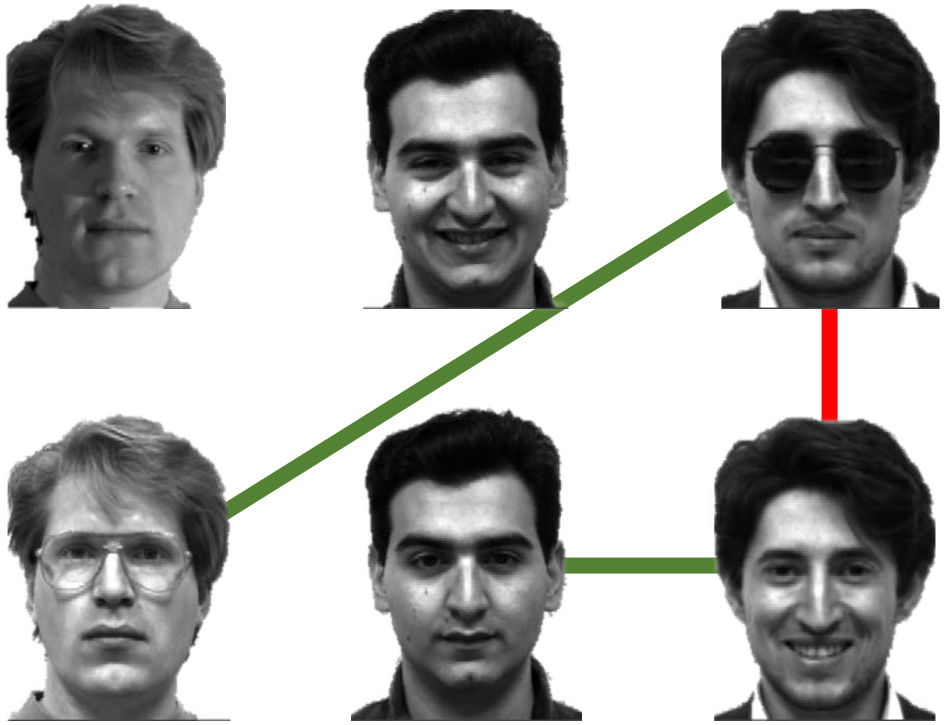
Instead of finetuning distance, algorithm and parameters…



Instead of finetuning distance, algorithm and parameters… provide some example pairs

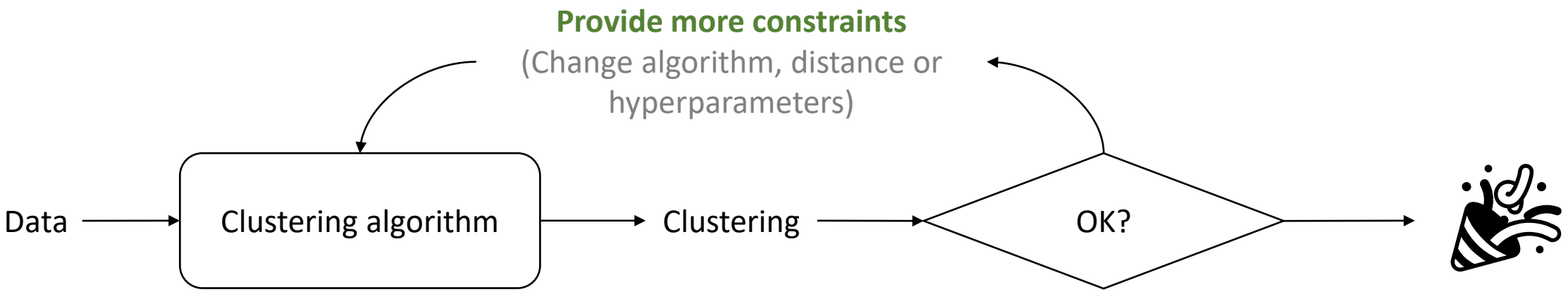


Instead of finetuning algorithm, distance and parameters… provide some example pairs



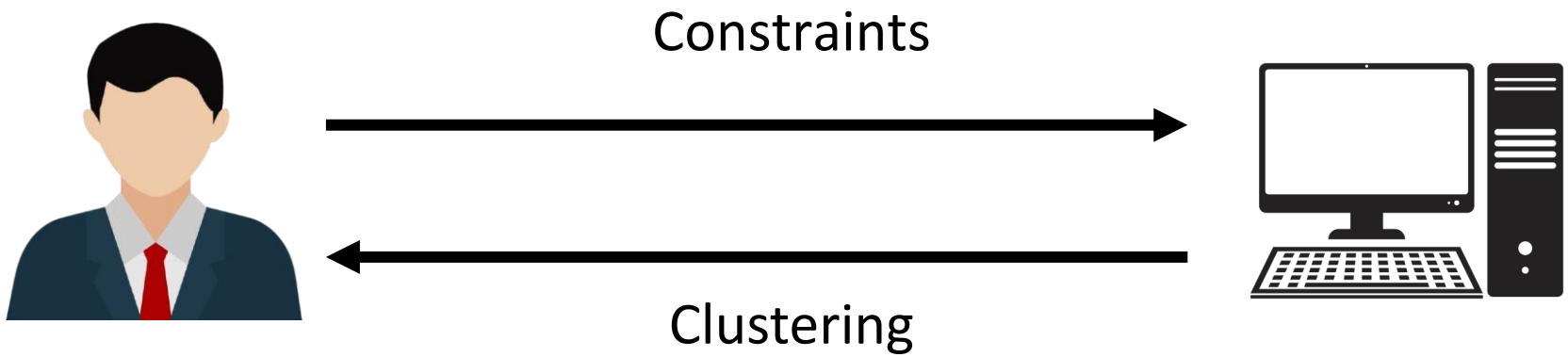
Images: CMU faces dataset

The constraints guide the algorithm towards the desired clustering



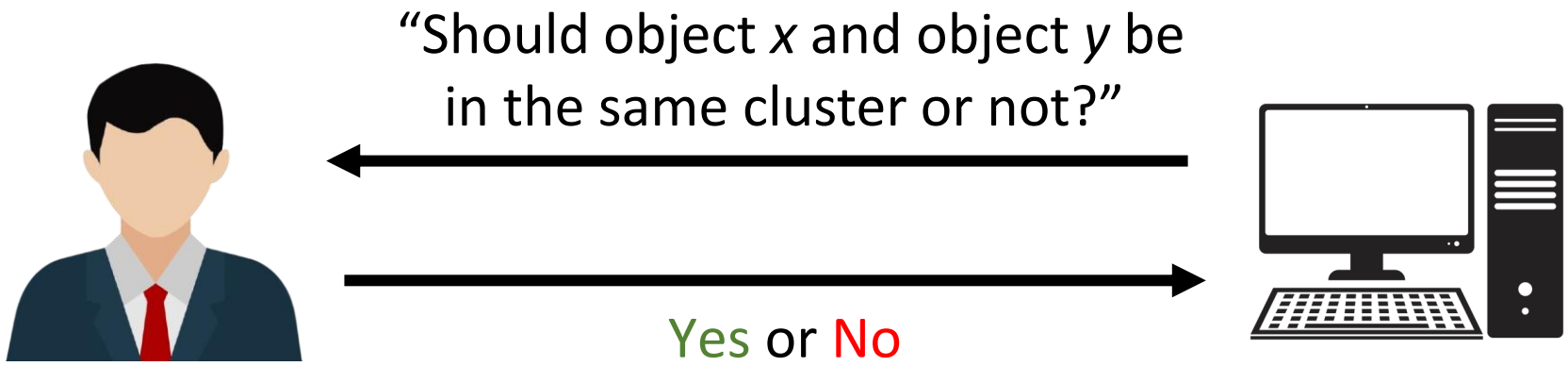
Semi-supervised, Active and Robust clustering

In semi-supervised clustering, the user leads the clustering process



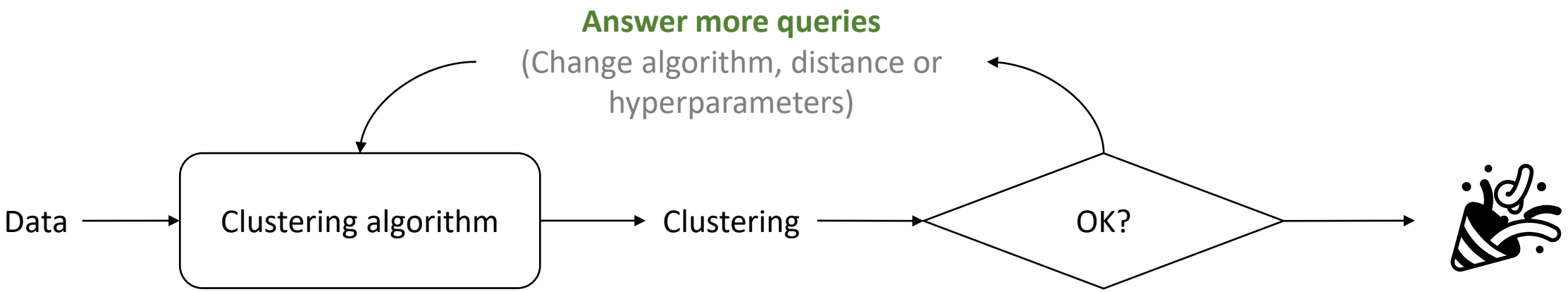
Which constraints are most useful for the algorithm?

In active semi-supervised clustering, the algorithm takes the lead



The algorithm selects the most informative questions first

In active semi-supervised clustering, the algorithm takes the lead



cobras

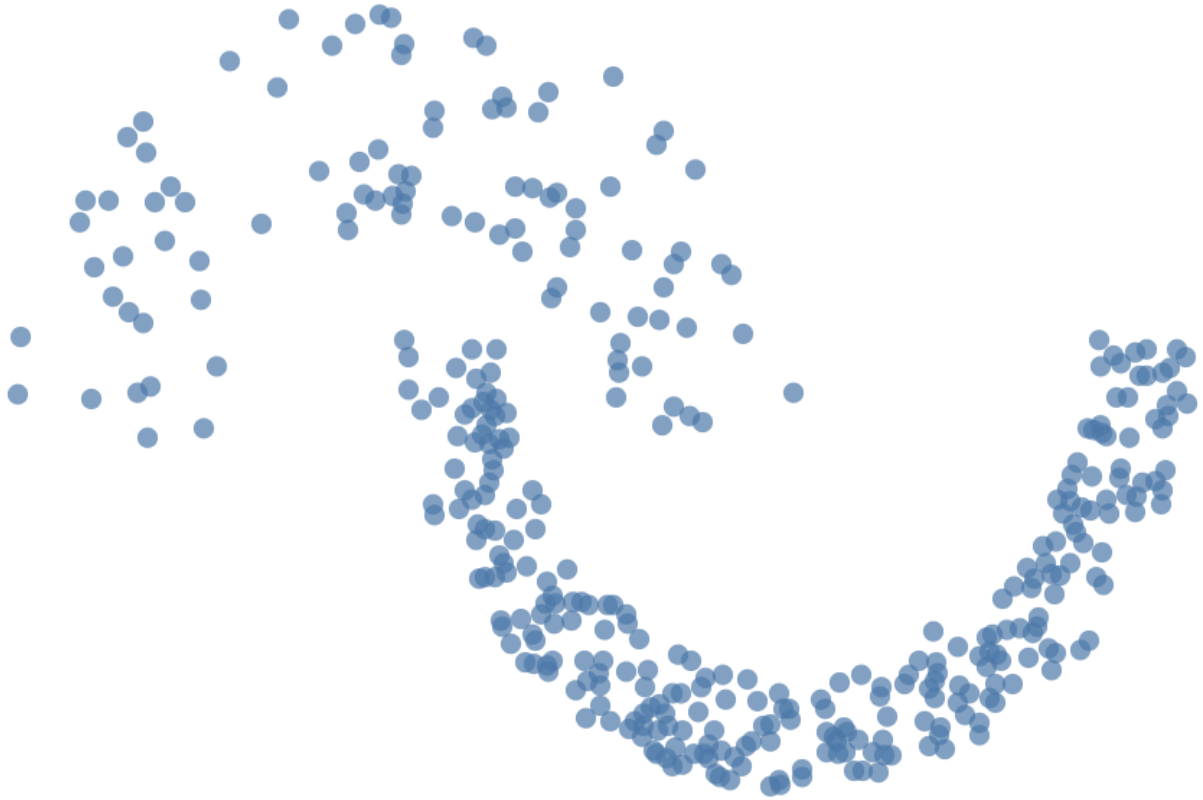
A state-of-the-art active semi-supervised clustering algorithm

Query efficient	Small amount of queries
Time efficient	Limited waiting between queries
Anytime	Intermediate results

Van Craenendonck, Toon, et al. 2018

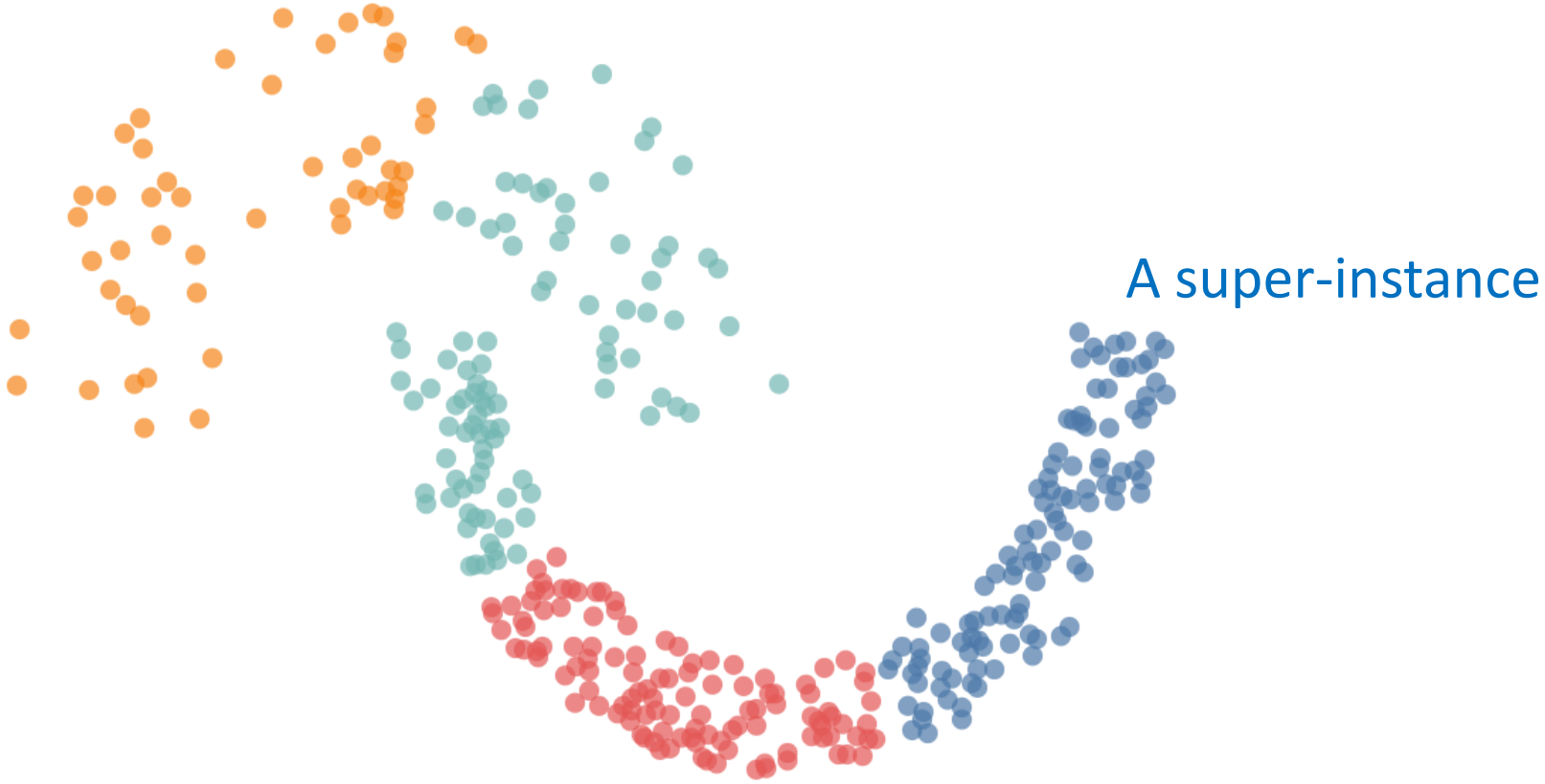
cobras

A state-of-the-art active semi-supervised clustering algorithm



cobras

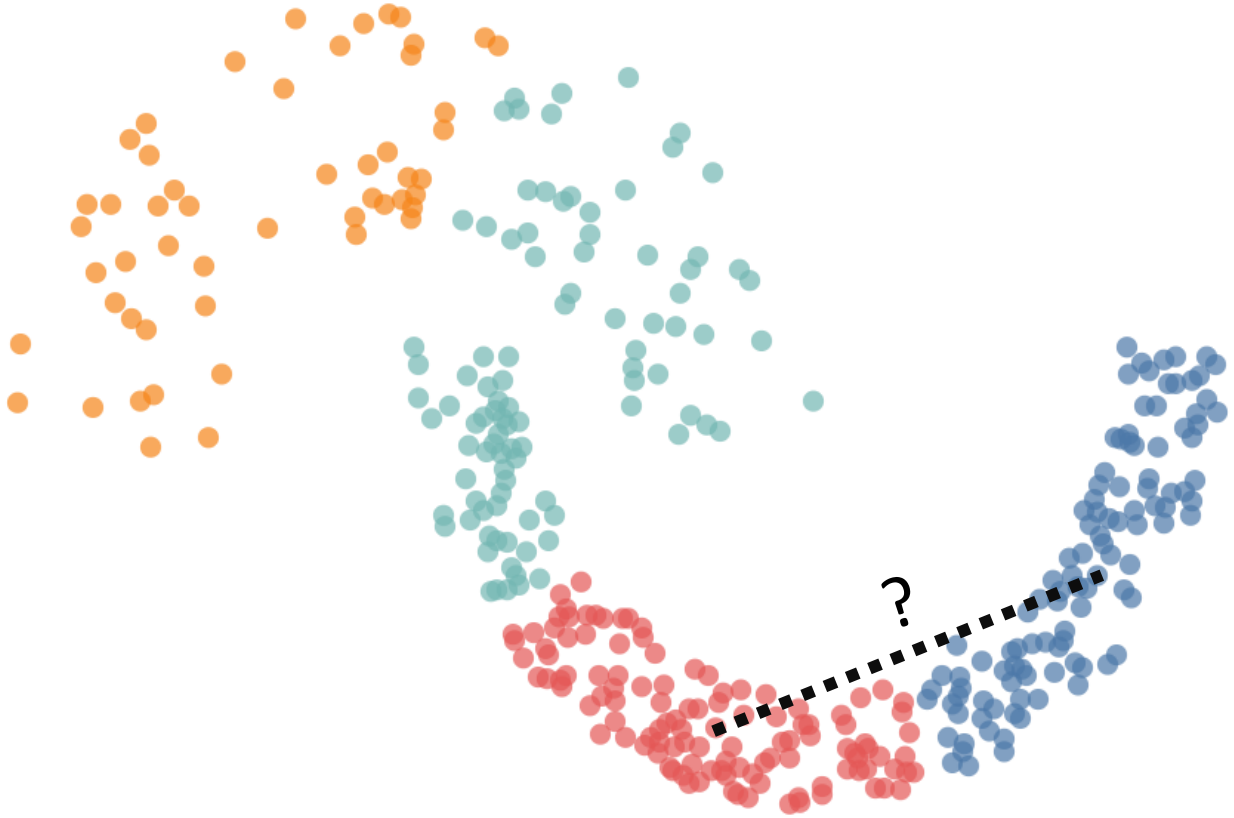
A state-of-the-art active semi-supervised clustering algorithm



Key idea 1: very similar instances are part of the same cluster

cobras

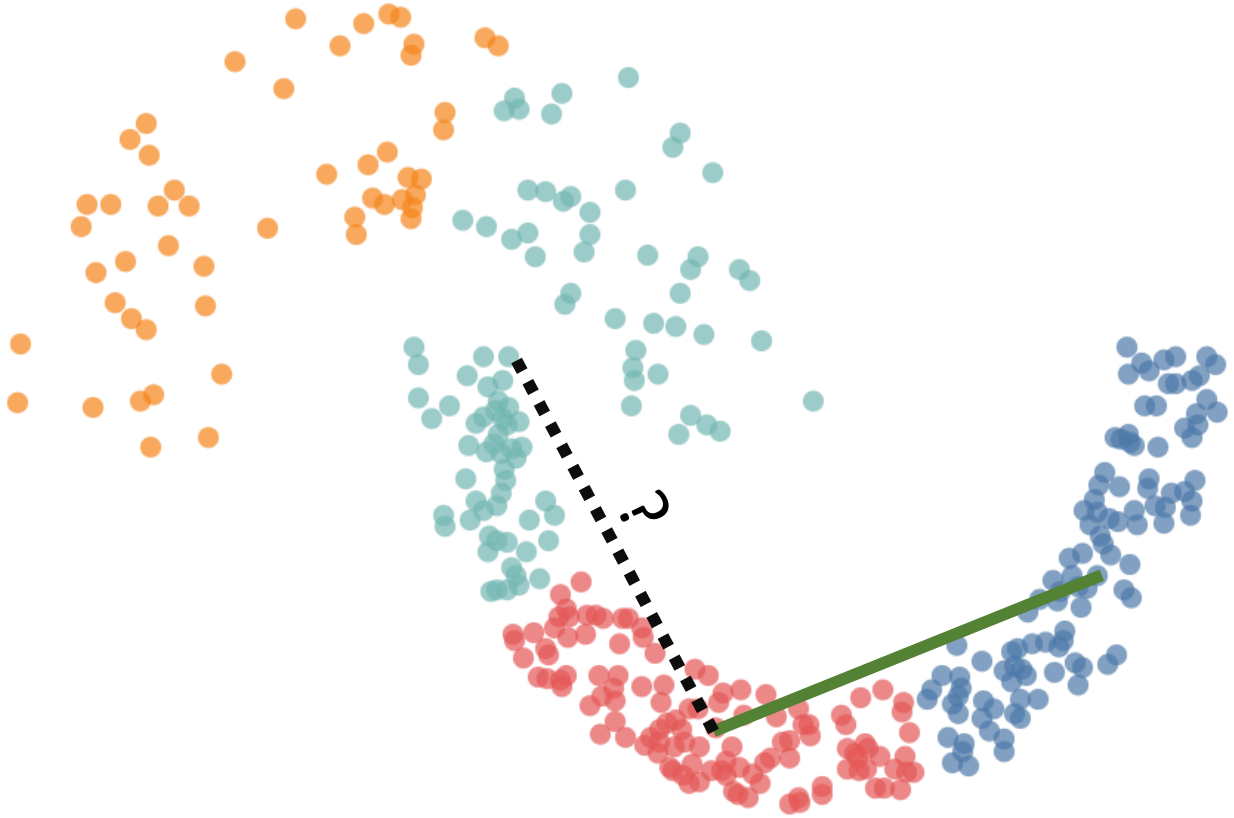
A state-of-the-art active semi-supervised clustering algorithm



Key idea 2: merge super-instances into clusters using constraints

cobras

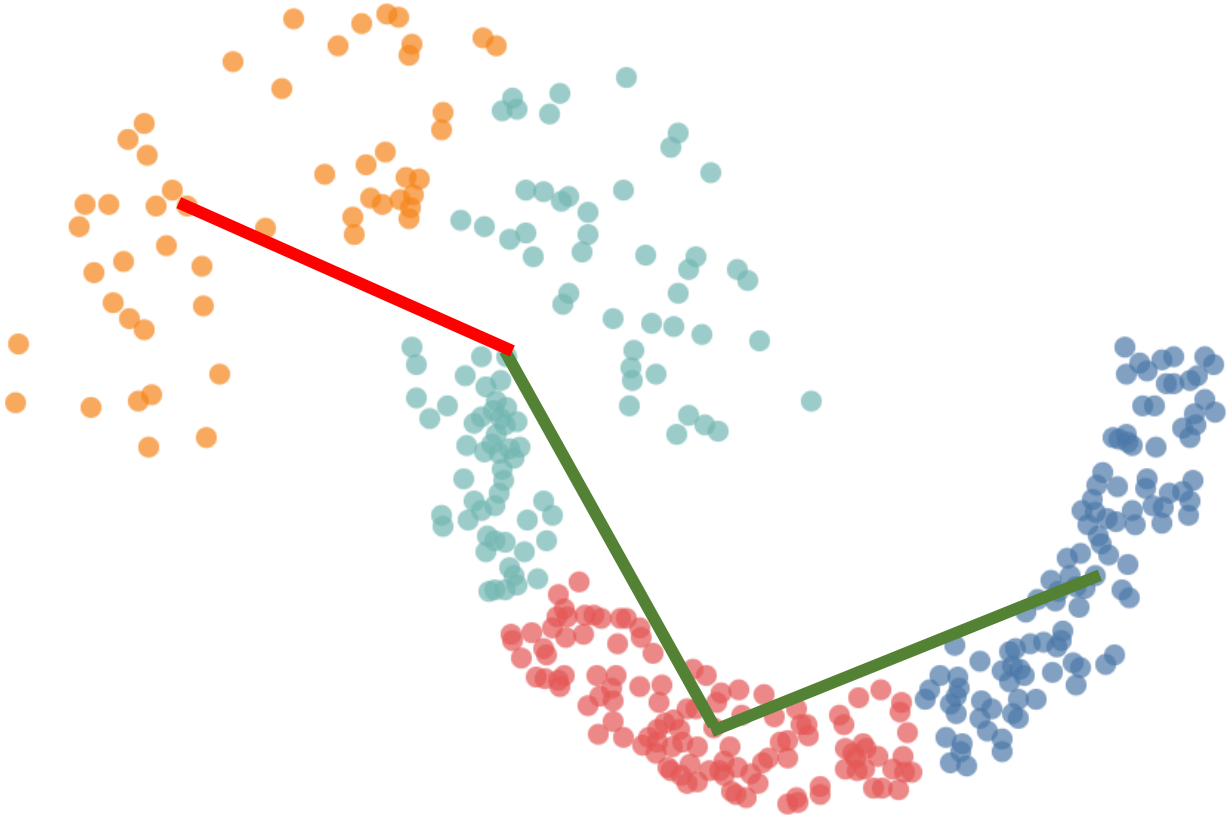
A state-of-the-art active semi-supervised clustering algorithm



Key idea 2: merge super-instances into clusters using constraints

cobras

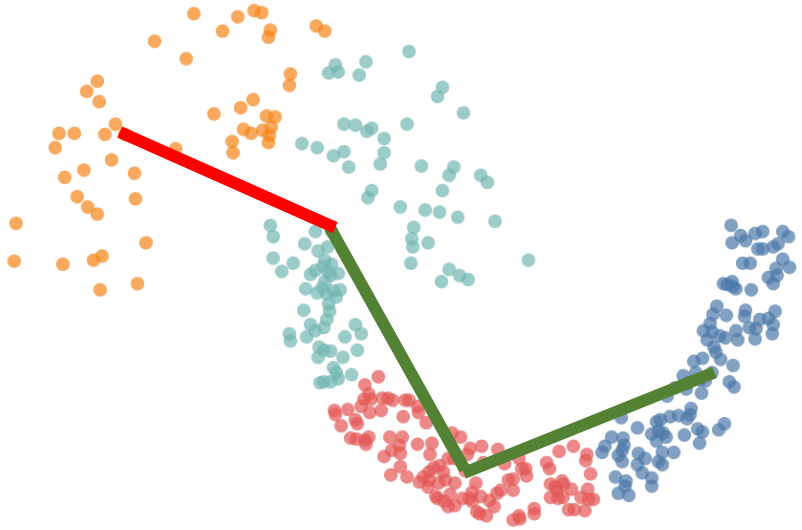
A state-of-the-art active semi-supervised clustering algorithm



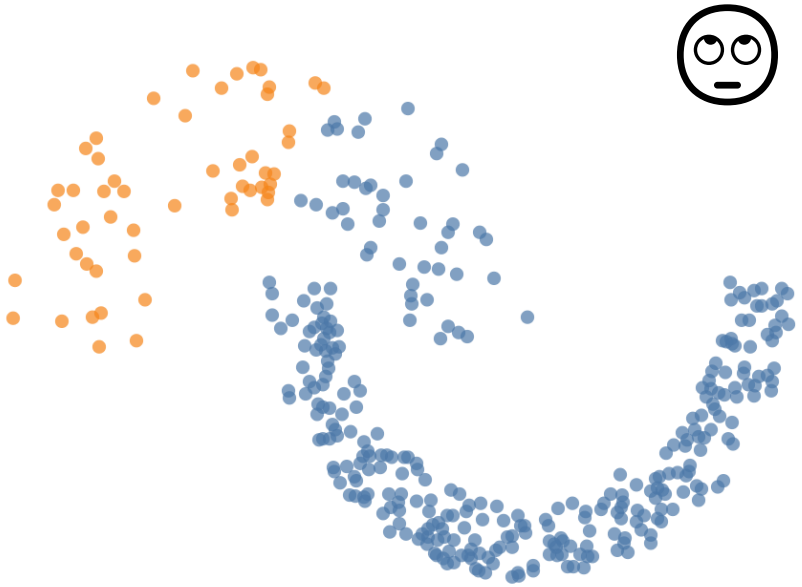
Key idea 2: merge super-instances into clusters using constraints

cobras

A state-of-the-art active semi-supervised clustering algorithm



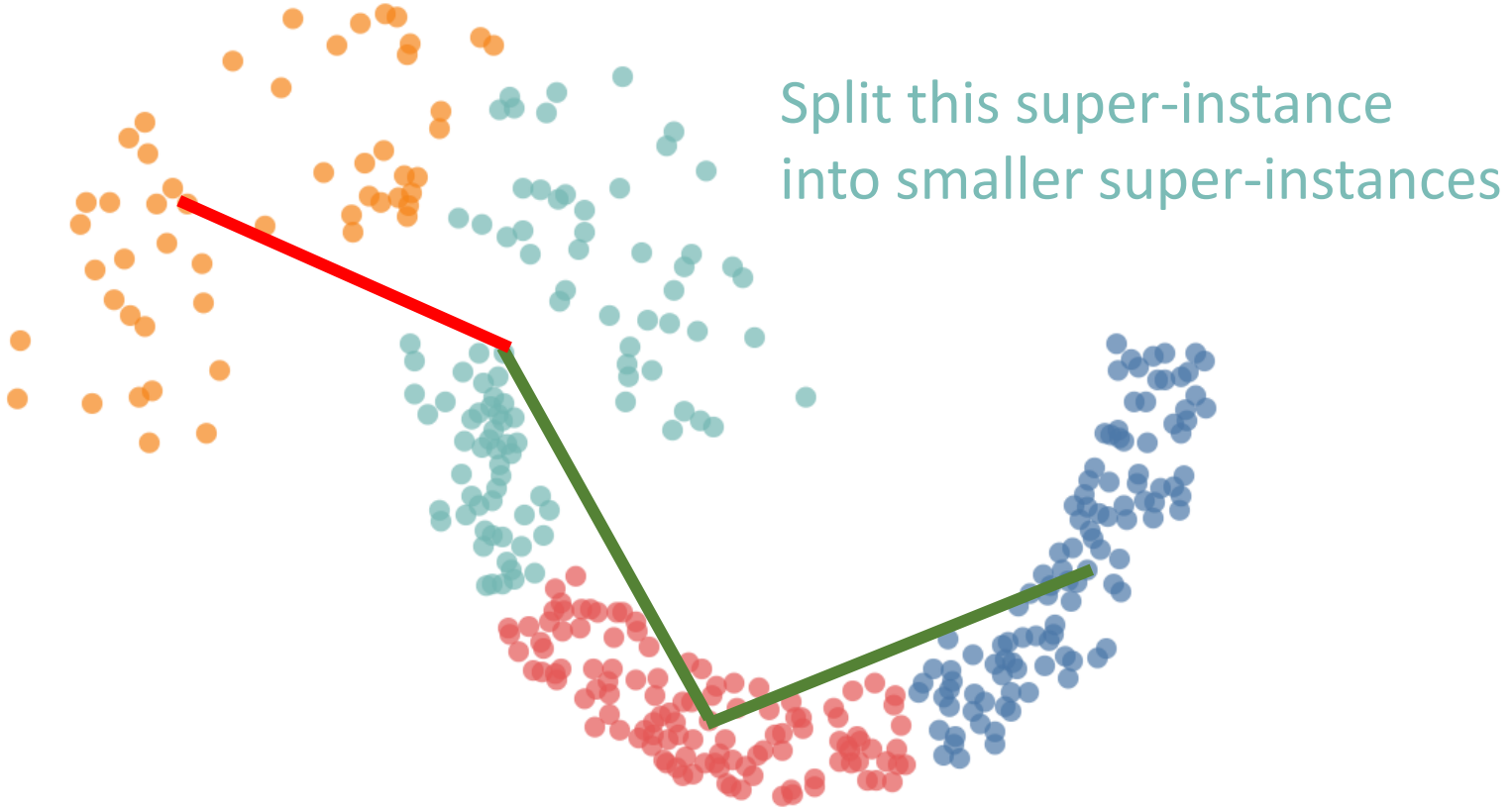
Super-instances and constraints



Clustering

cobras

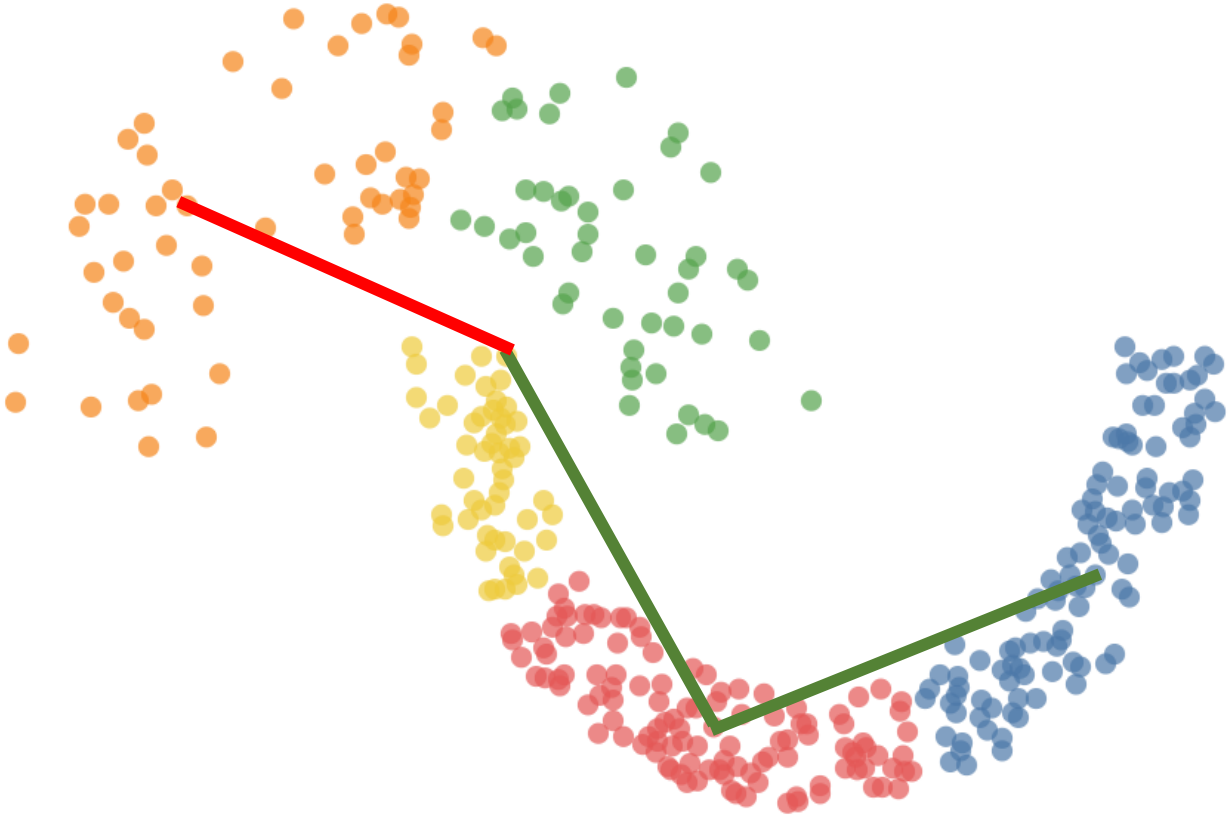
A state-of-the-art active semi-supervised clustering algorithm



Key idea 3: refine super-instances iteratively

cobras

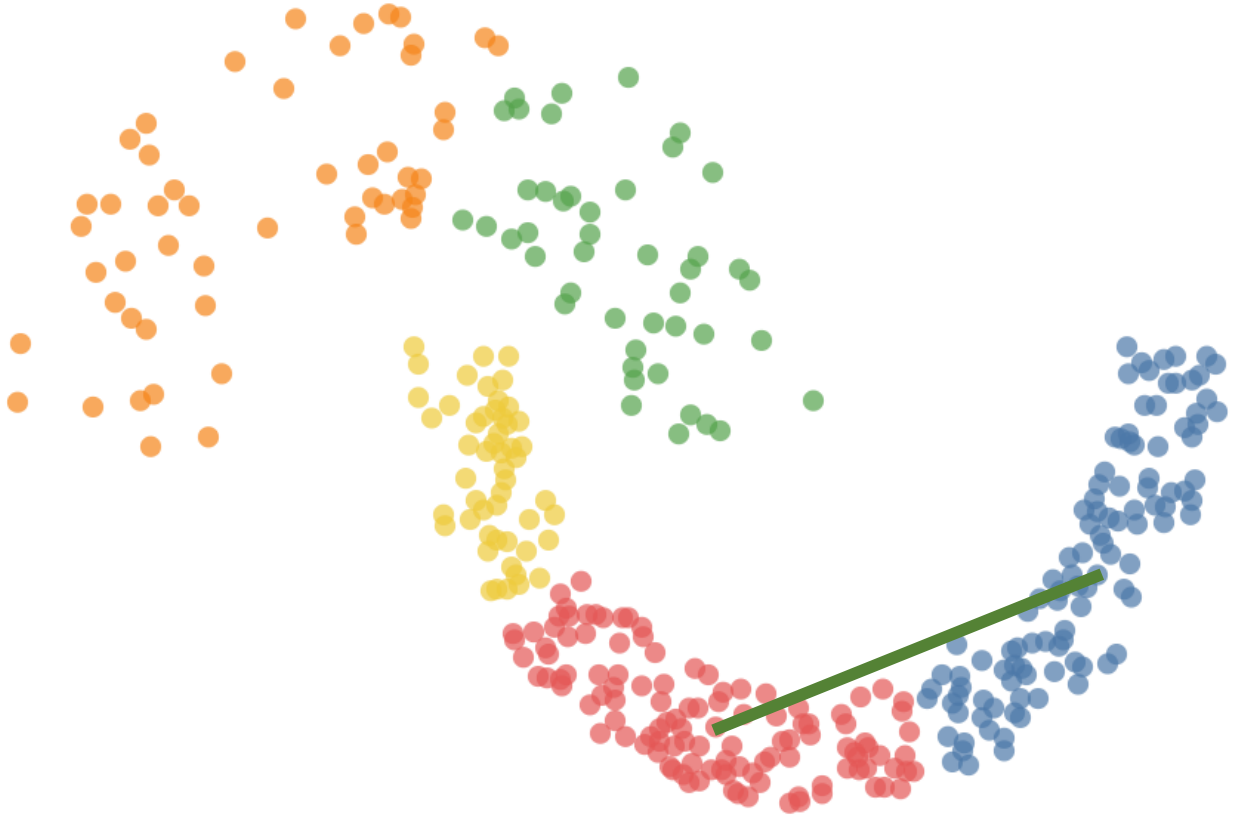
A state-of-the-art active semi-supervised clustering algorithm



Key idea 3: refine super-instances iteratively

cobras

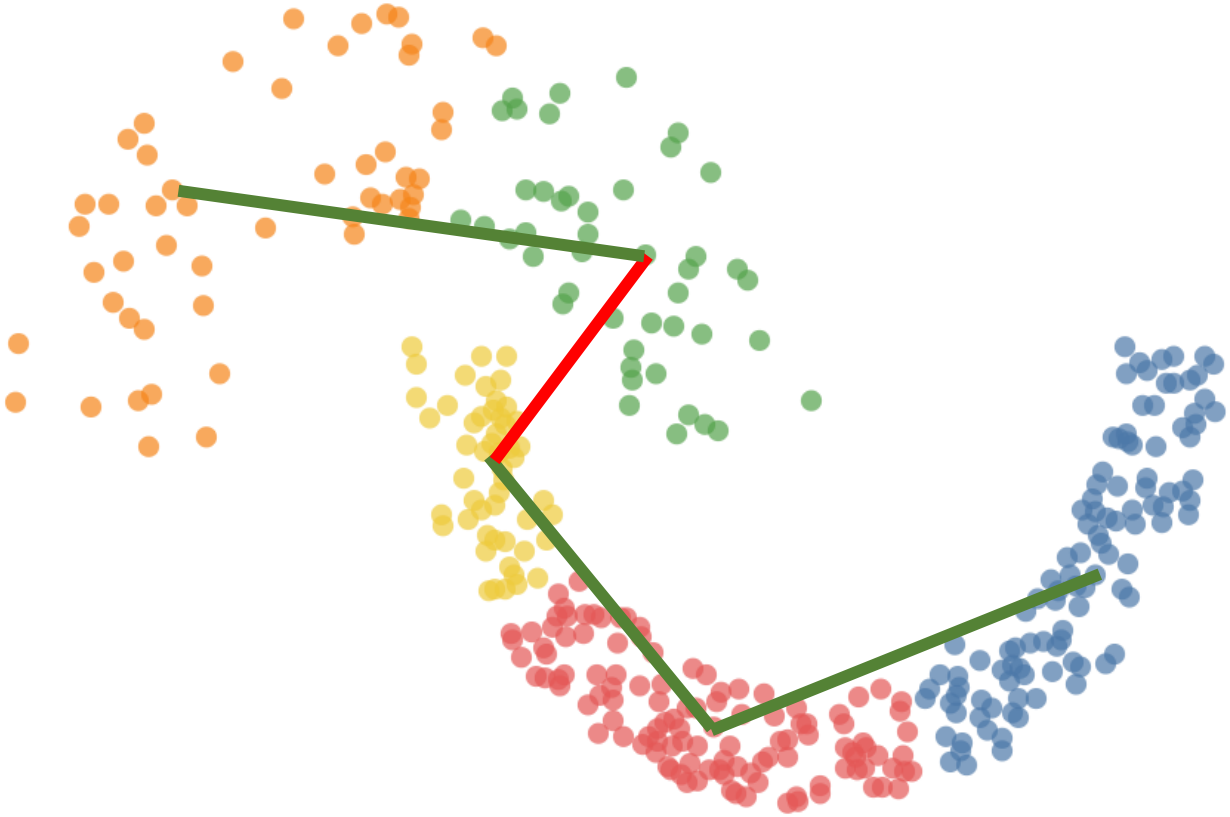
A state-of-the-art active semi-supervised clustering algorithm



Key idea 3: refine super-instances iteratively

cobras

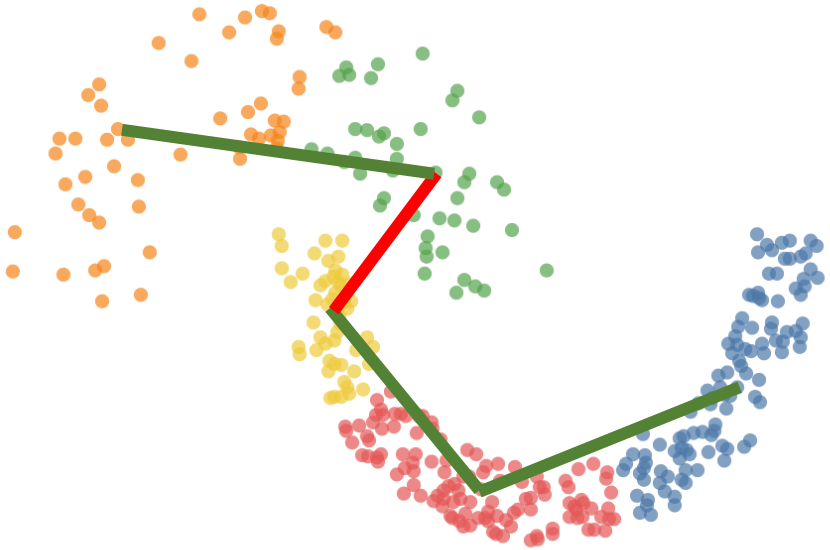
A state-of-the-art active semi-supervised clustering algorithm

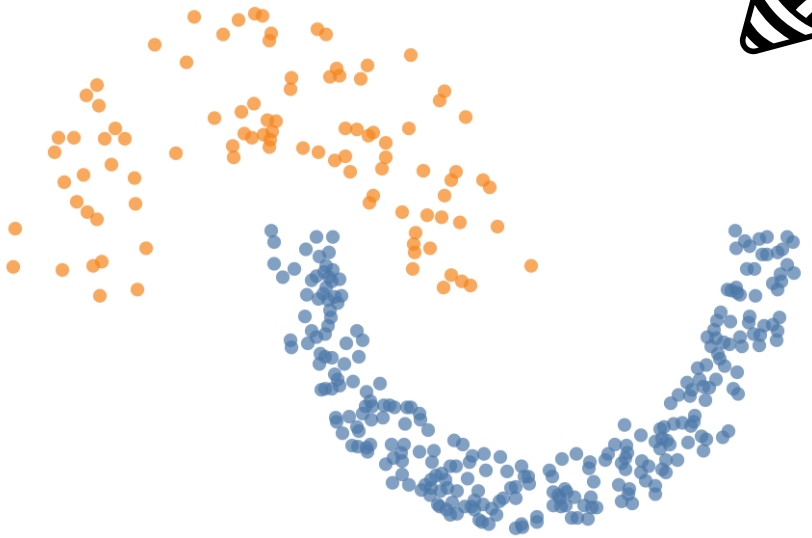


Key idea 3: refine super-instances iteratively

cobras

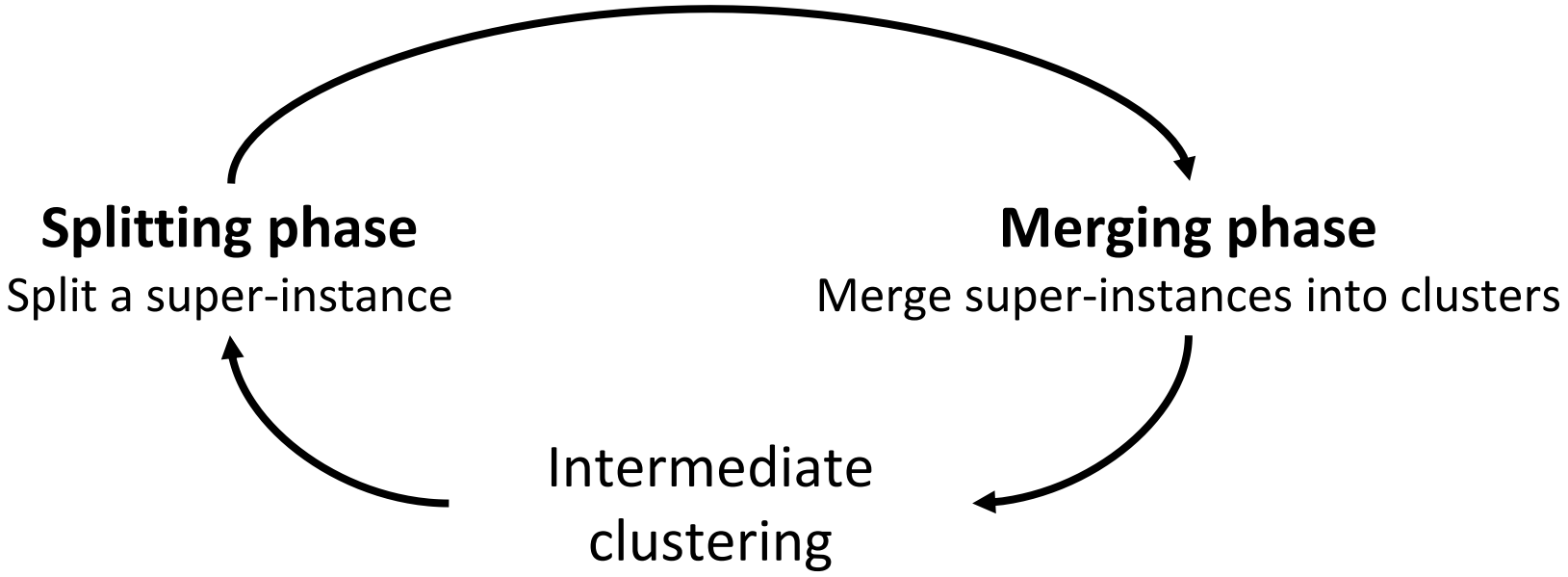
A state-of-the-art active semi-supervised clustering algorithm

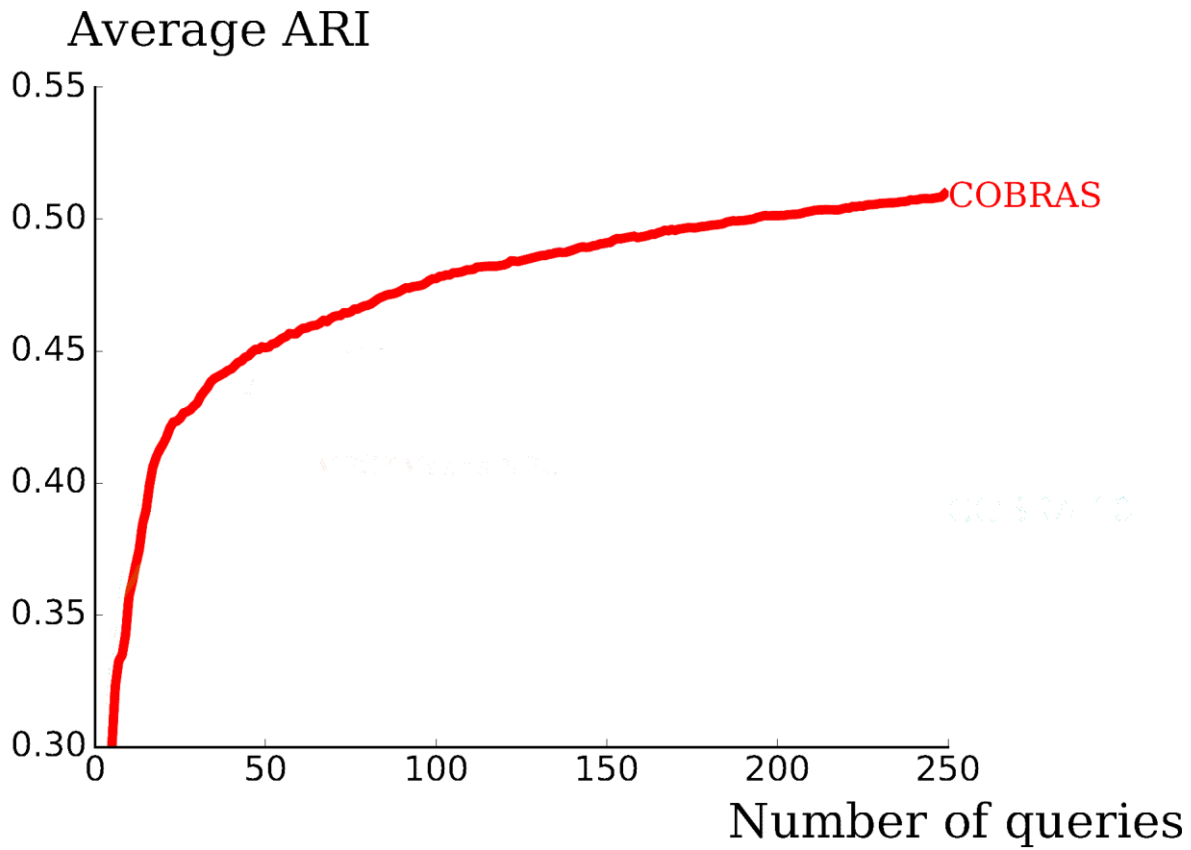




After 6 queries, COBRAS finds the perfect clustering

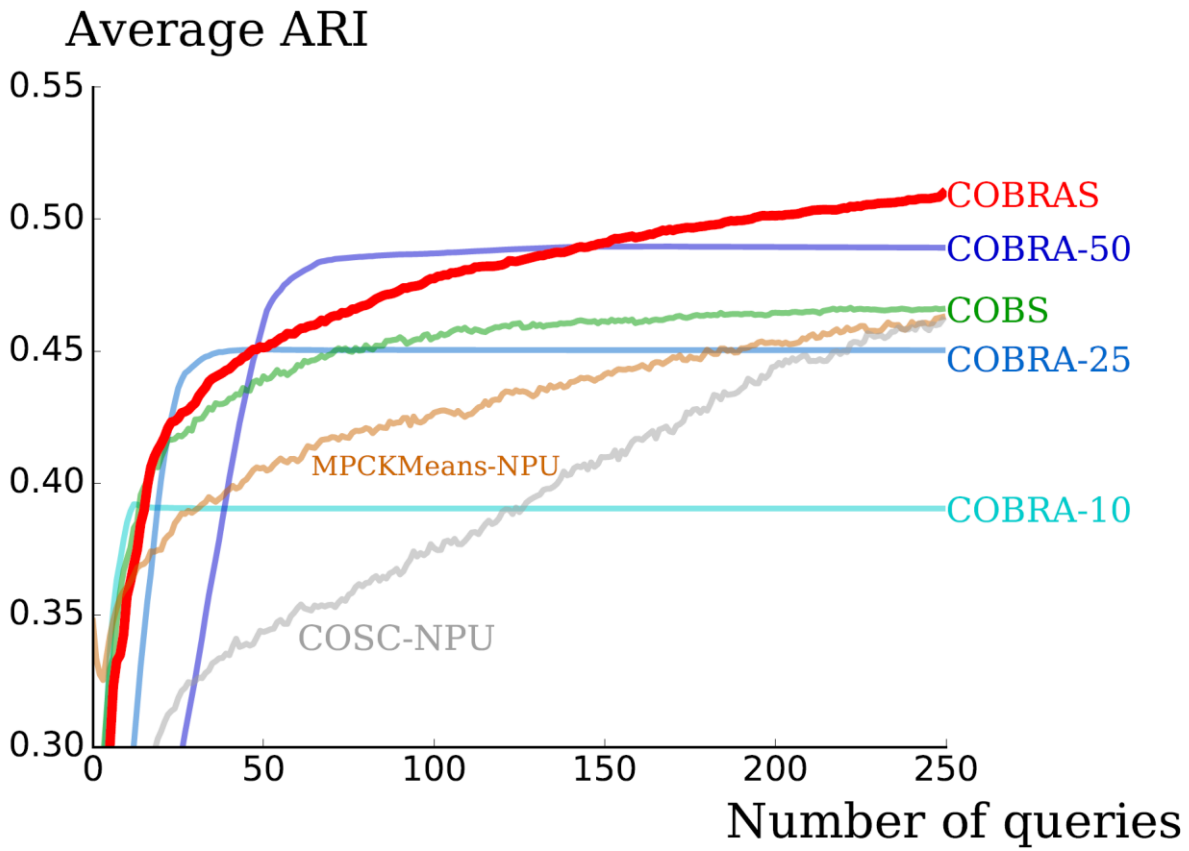
cobras Is anytime





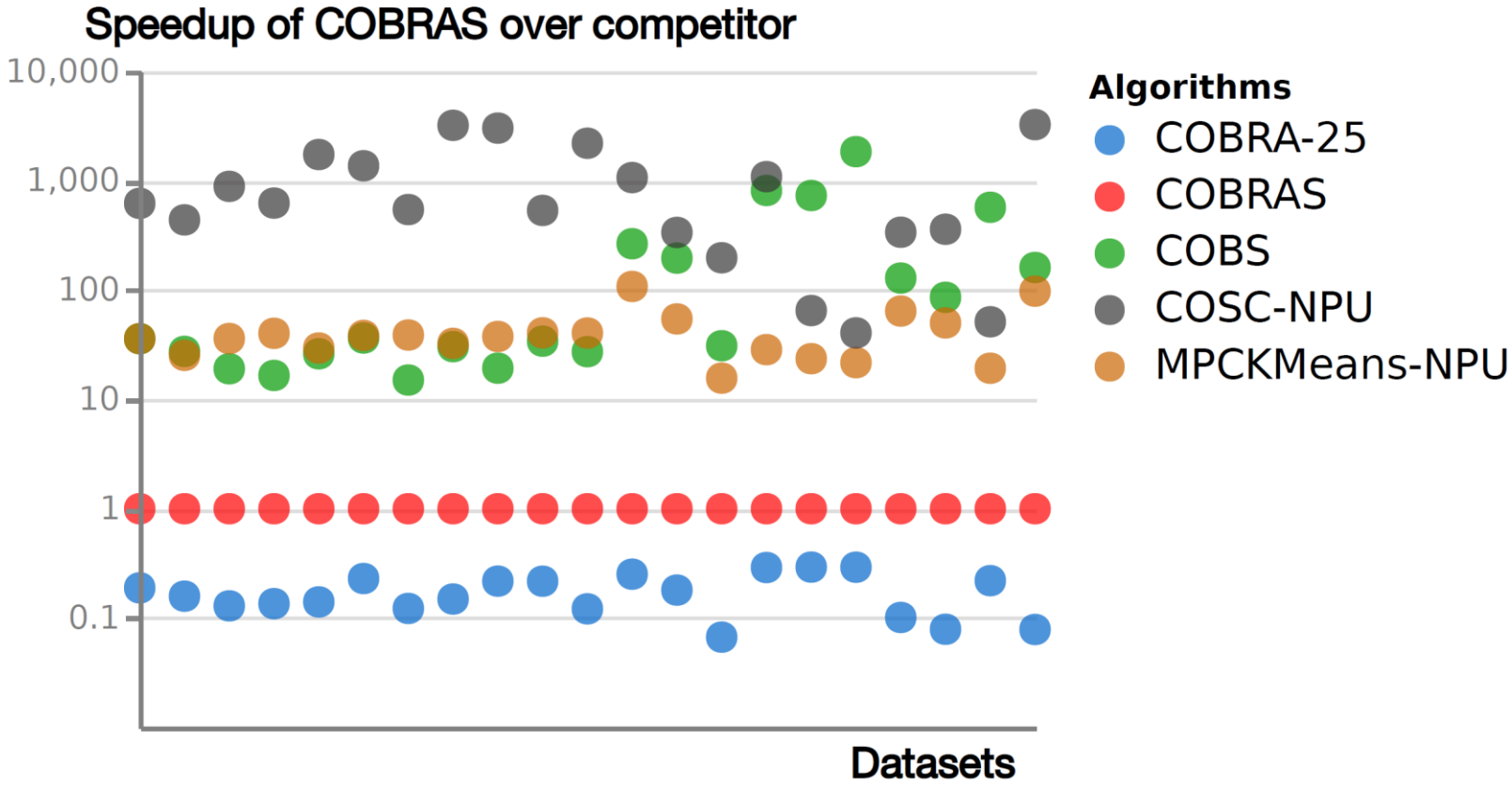
The more constraints, the better the clustering

cobras Is query efficient



Compared to competitors, COBRAS produces better clusterings

cobras Is time efficient



COBRAS is 10 to 1000 times faster than competitors

cobras in a 

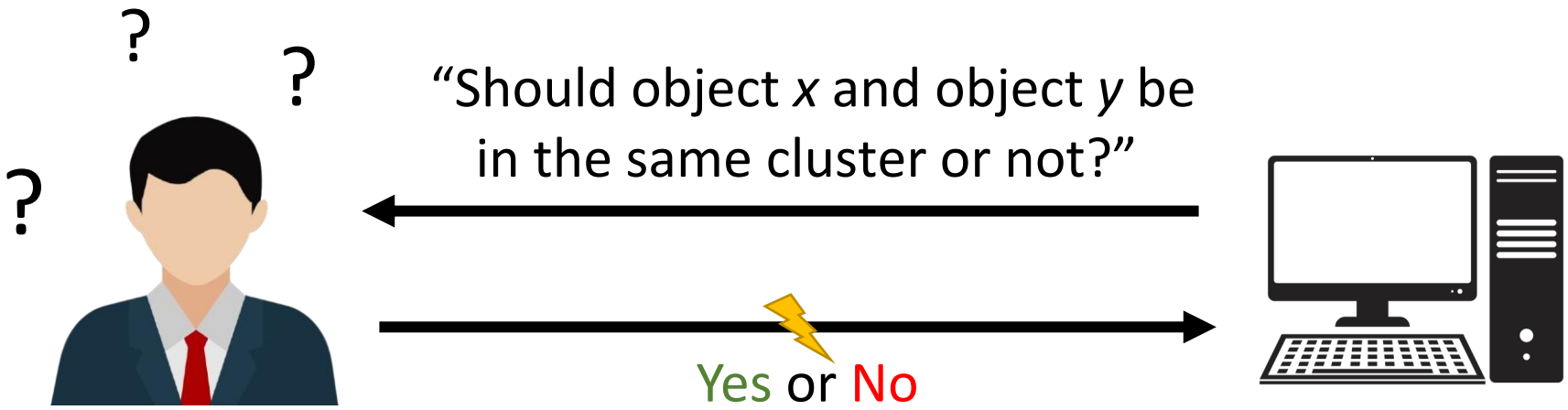
Anytime, Query efficient* and Time efficient

Very similar instances are part of the same cluster (super-instances)
Merging phase: merge super-instances into clusters using constraints
Splitting phase: refine super-instances iteratively

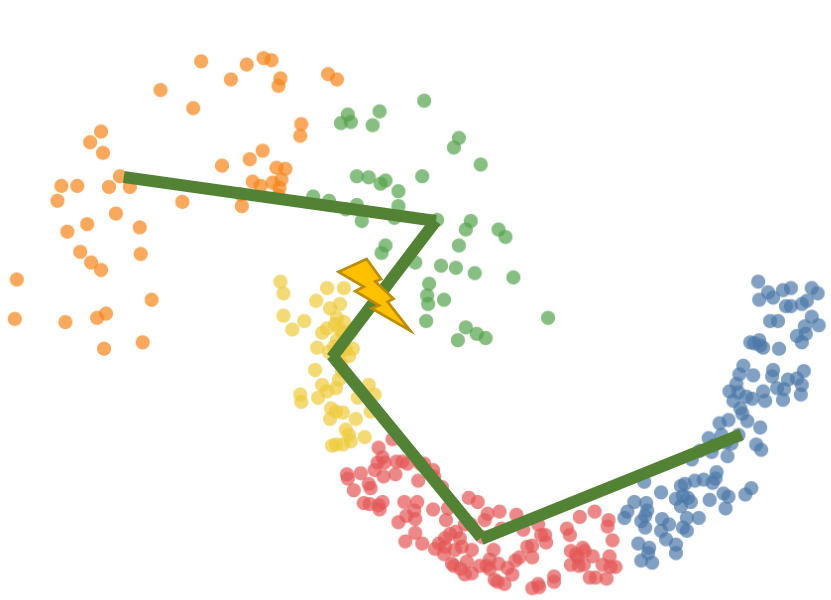
* Yurtman, Aras et al. 2021 propose a way to reuse more queries in COBRAS

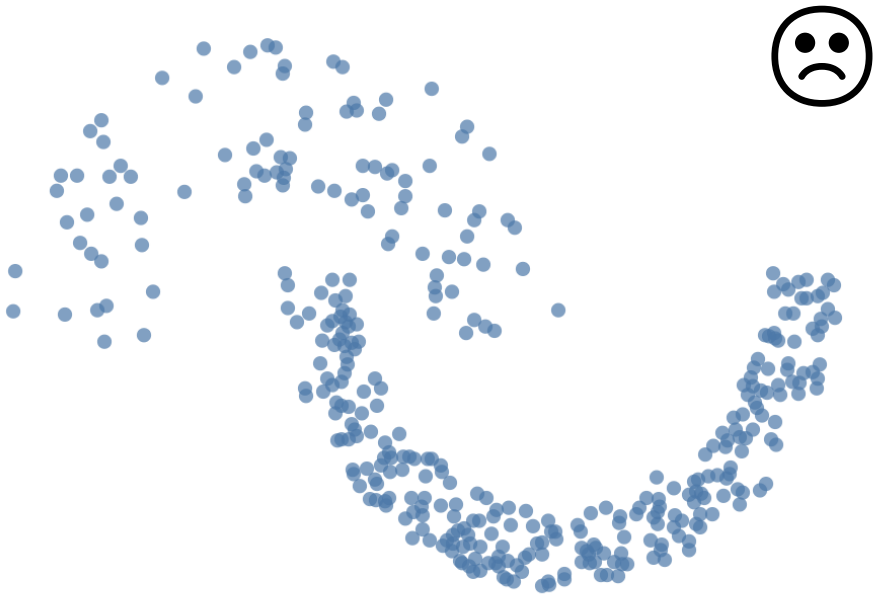
Semi-supervised, Active and Robust clustering

What if the user makes a mistake?



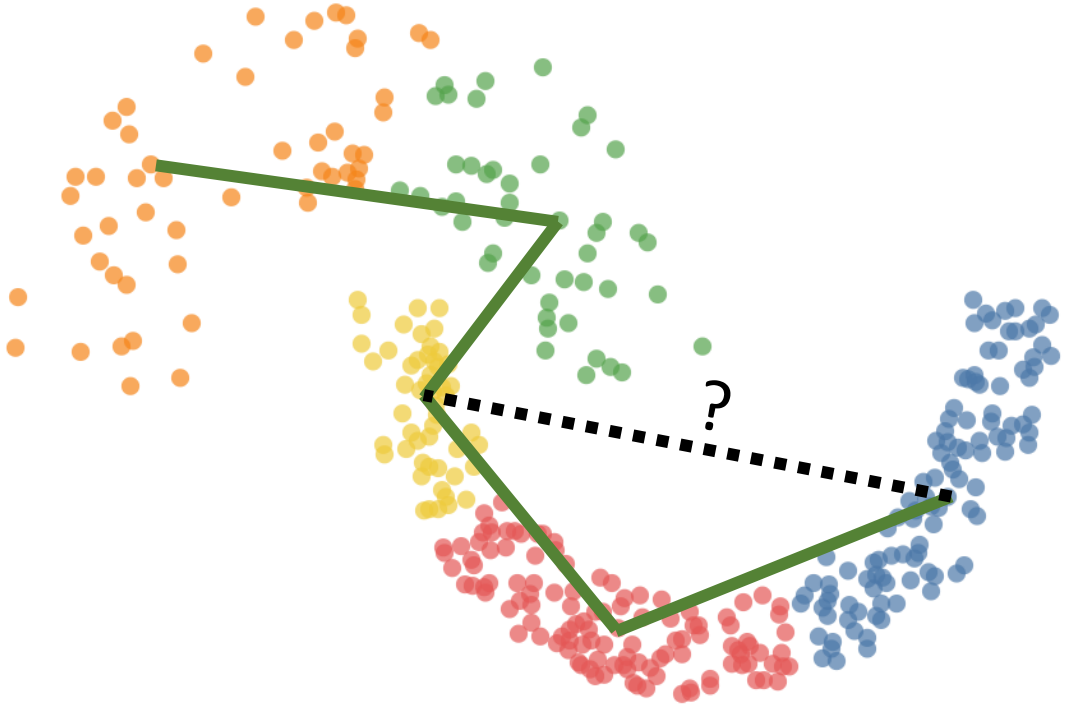
In cobras, a single incorrect constraint can have a detrimental impact





A single noisy constraint can impact multiple super-instances!

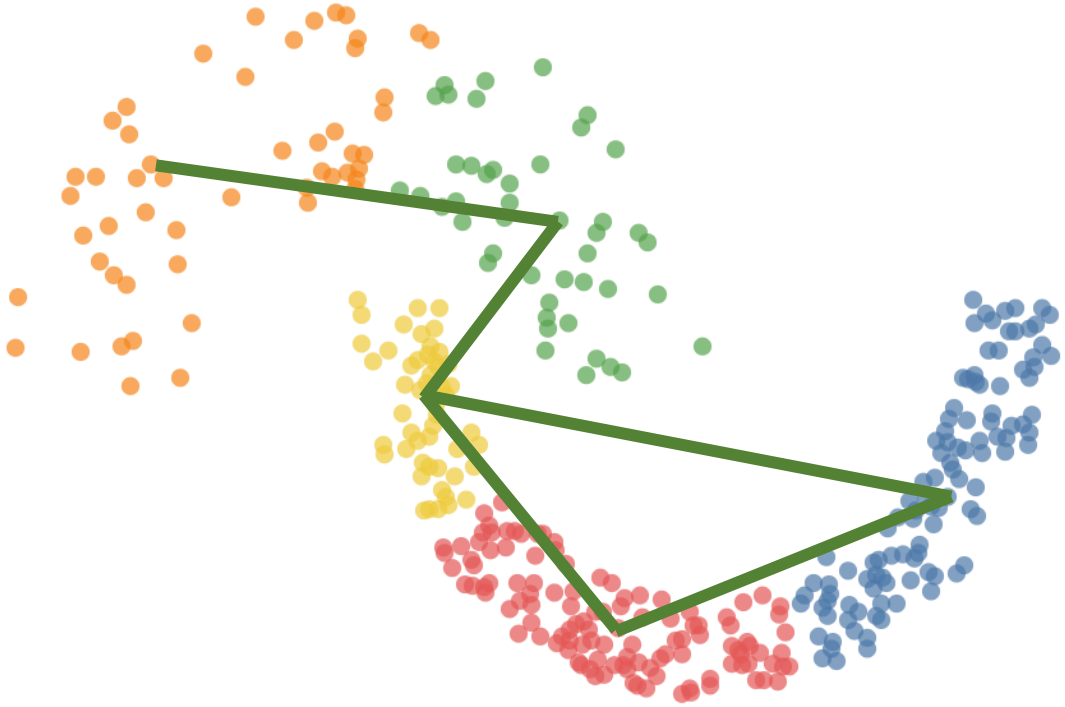
Noise robust cobras detects and corrects the constraints



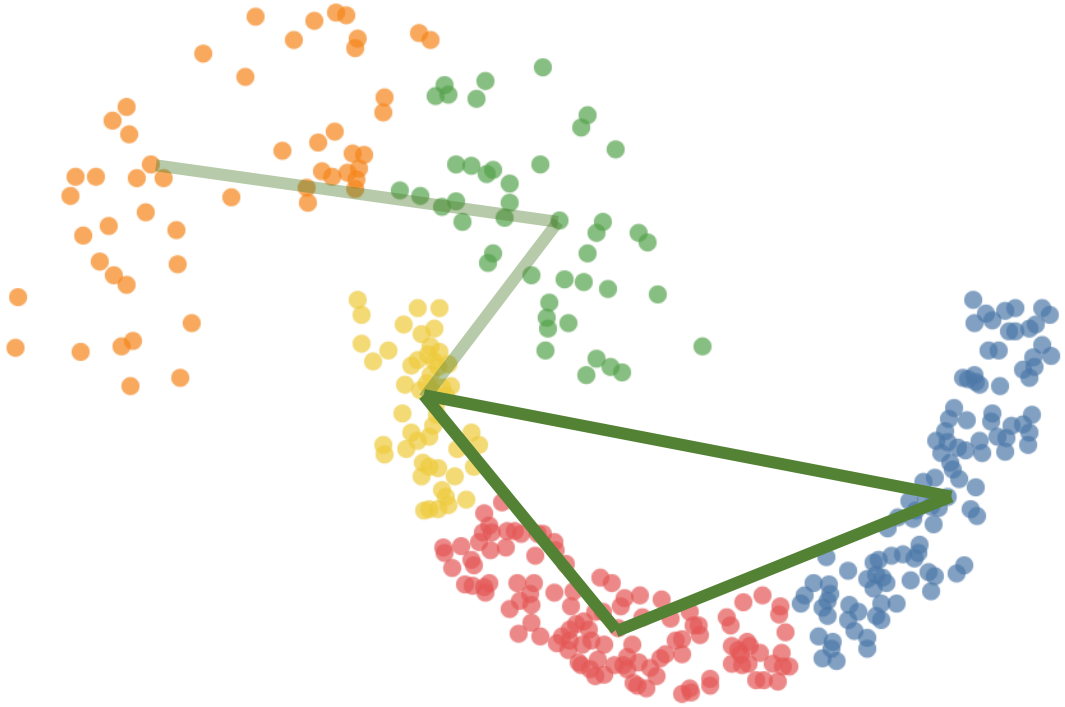
Key idea: ask additional **redundant** constraints and **reason** probabilistically

Soenen, Jonas et al. 2021

Noise robust cobras detects and corrects the constraints

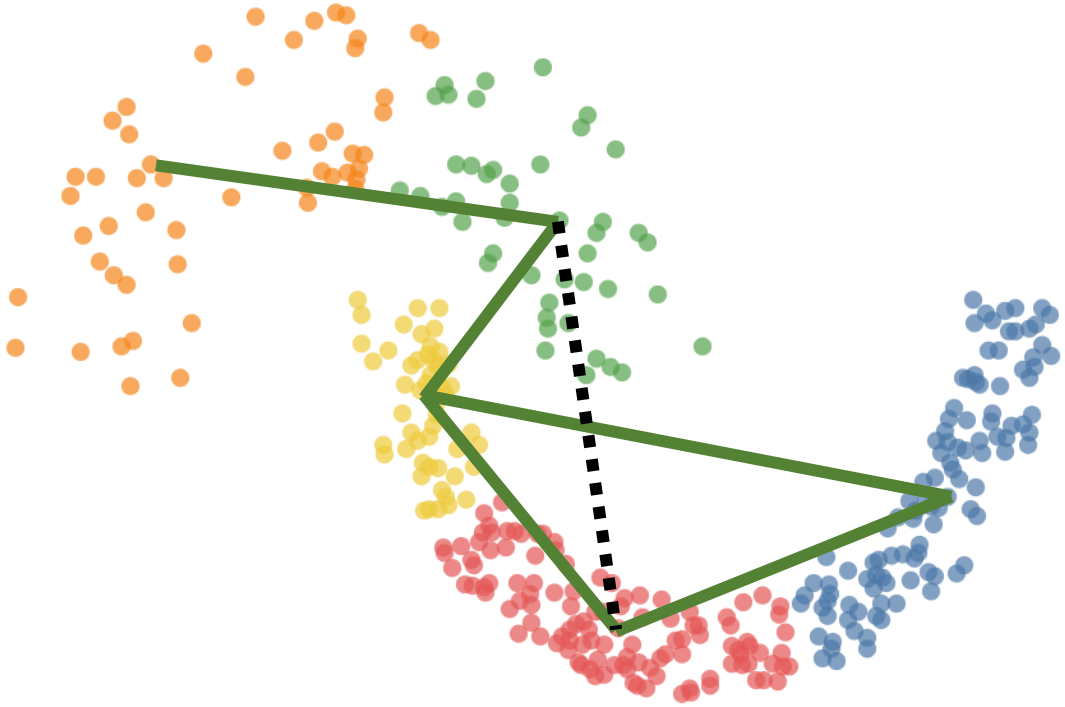


Noise robust cobras detects and corrects the constraints

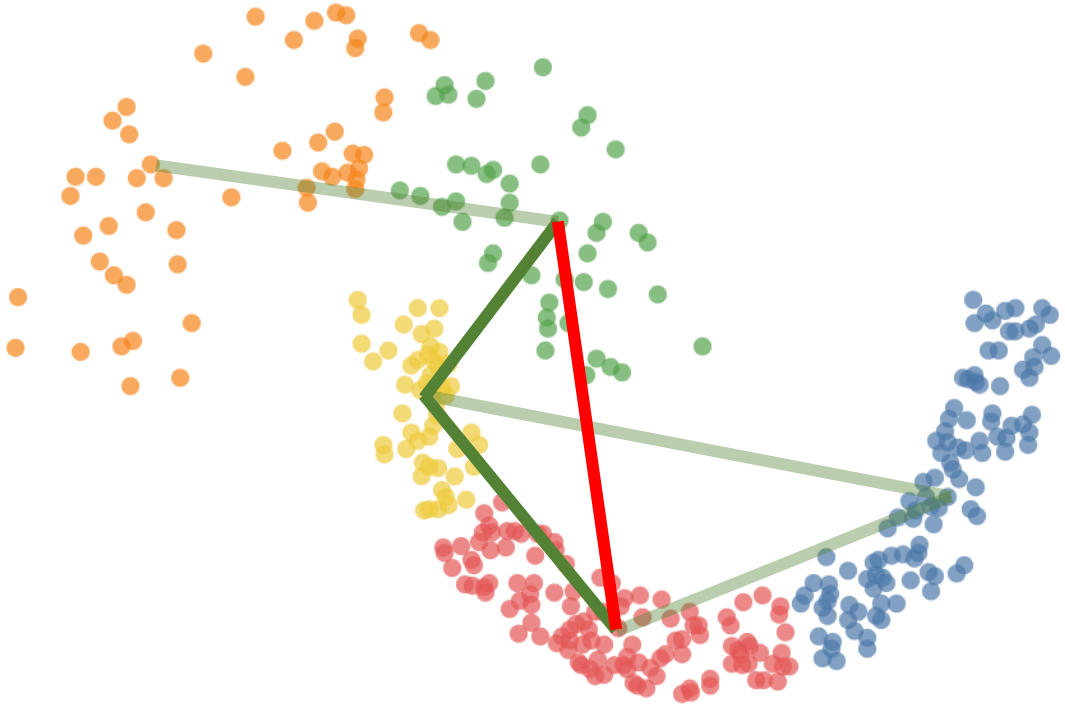


A cycle of must-links increases our confidence in the involved constraints

Noise robust cobras detects and corrects the constraints

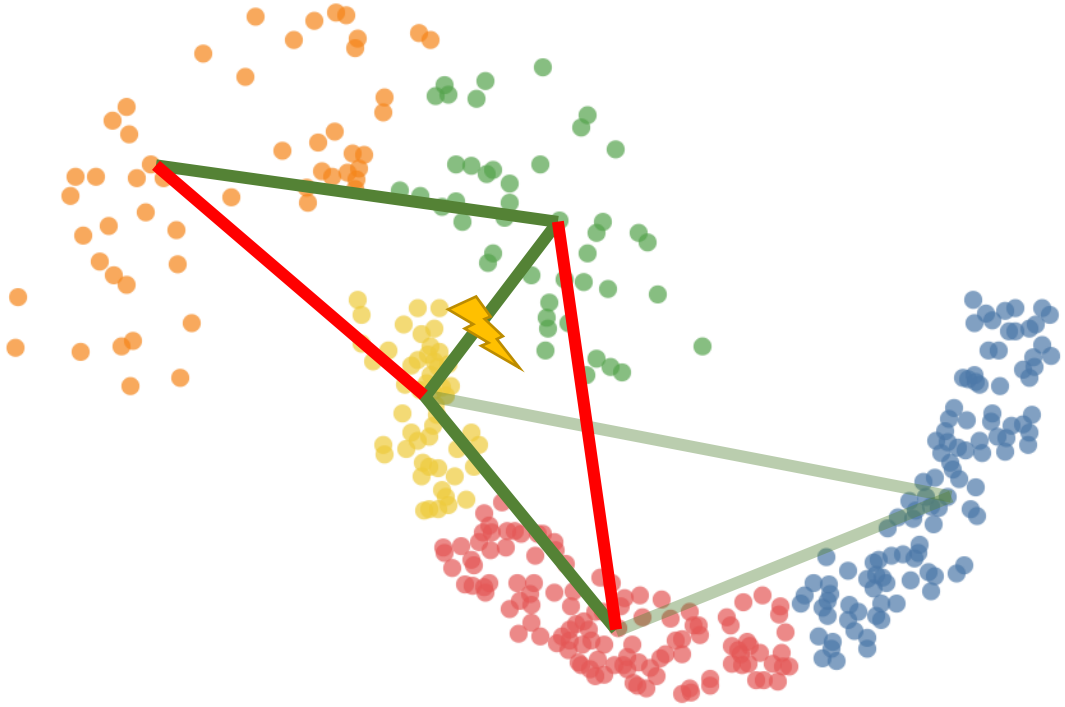


Noise robust cobras detects and corrects the constraints



A cycle with exactly one cannot-link is inconsistent.
There must be a noisy constraint!

Noise robust cobras detects and corrects the constraints



An additional constraint helps us decide which constraint is noisy

A simple probabilistic model is used to reason about correctness of constraints

$$P(C\ is\ correct|U\ ) = \frac{P(U|C\ correct)\ P(C\ correct)}{\sum_{all\ C'} P(U|C'correct)\ P(C'\ correct)}$$

Probability that constraints C are correct given noisy user constraints U

A simple probabilistic model is used
to reason about correctness of constraints

$$P(C \text{ is correct} | U) = \frac{P(U|C \text{ correct}) \, P(C \text{ correct})}{\sum_{\text{all } C'} P(U|C' \text{correct}) \, P(C' \text{ correct})}$$

Assumption 1

$P(\text{user makes mistake}) = \nu$

Assumption 2

Every consistent constraint set is equally likely

≈ Every clustering is equally likely

A simple probabilistic model is used to reason about correctness of constraints

$$P(C\ is\ correct|U\ ) = \frac{\nu^{d}(1-\nu)^{n-d}}{\sum_{consistent\ C'} \nu^{d'}(1-\nu)^{n-d'}}$$

where:

$n\ =\ \#constraints$

$d\ =\ \#constraints$ where U and C disagree

$d'\ =\ \#constraints$ where U and C' disagree

A simple probabilistic model is used to reason about correctness of constraints

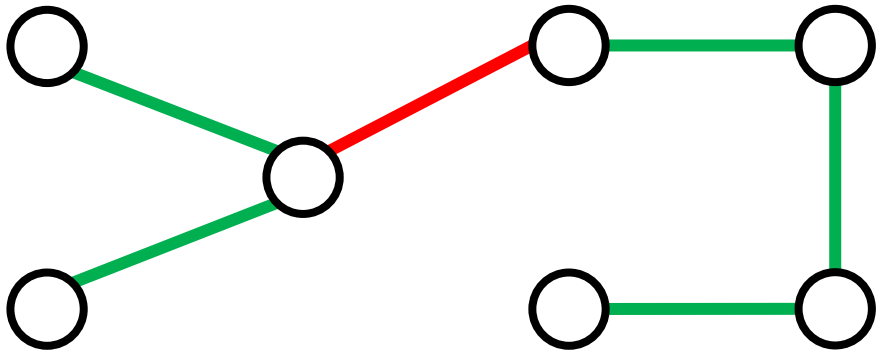
$$P(C \text{ is correct} | U) = \frac{\nu^d (1-\nu)^{n-d}}{\sum_{\text{consistent } C'} \nu^{d'} (1-\nu)^{n-d'}}$$

Expensive to compute → approximate

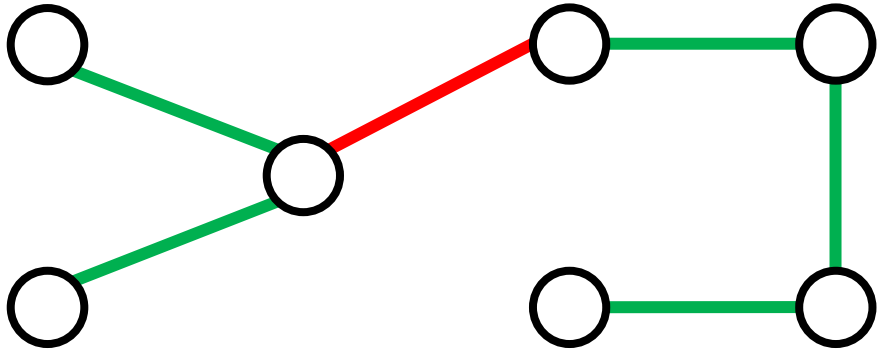
Only count consistent C' with highest likelihood

Full procedure

User constraints U



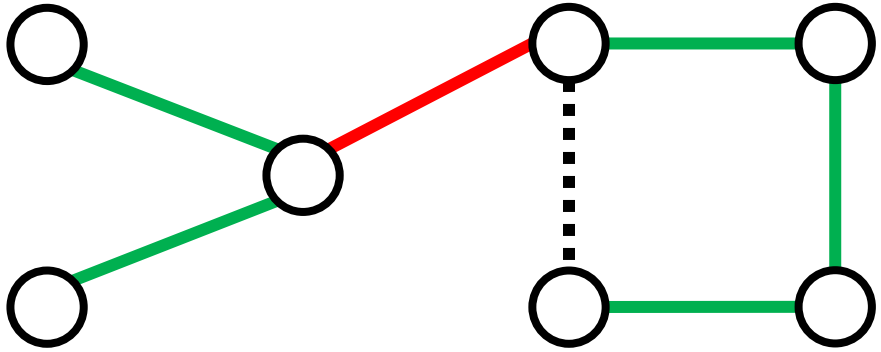
Most likely constraint set C



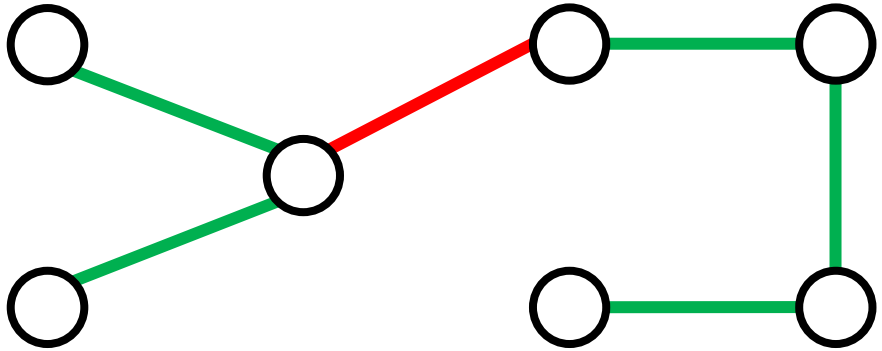
P(correct) = 0.532

Full procedure

User constraints U



Most likely constraint set C

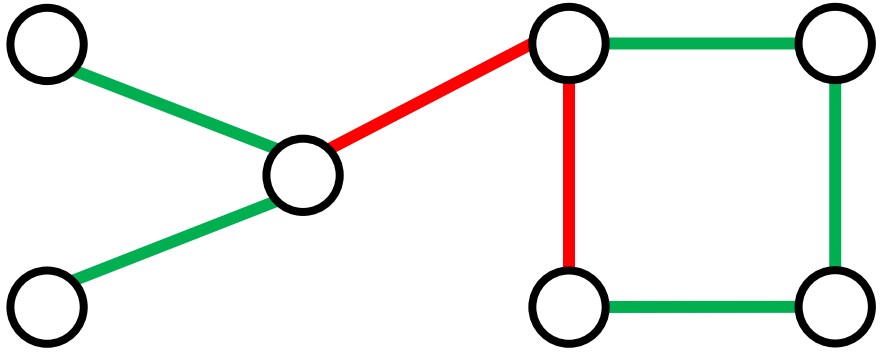


P(correct) = 0.532

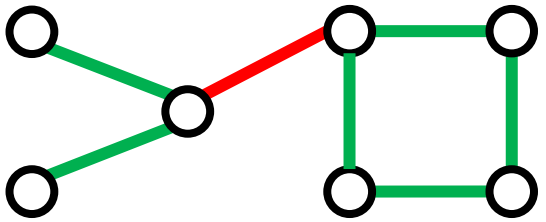
Ask a redundant constraint that increases the confidence in C

Full procedure

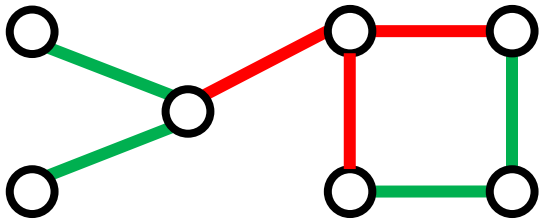
User constraints U



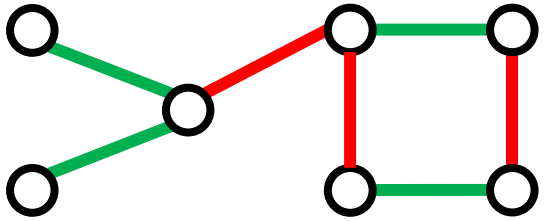
Most likely constraint set C



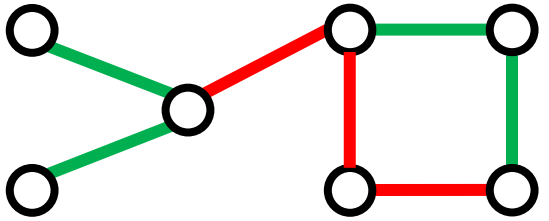
P(correct) = 0.16



P(correct) = 0.16



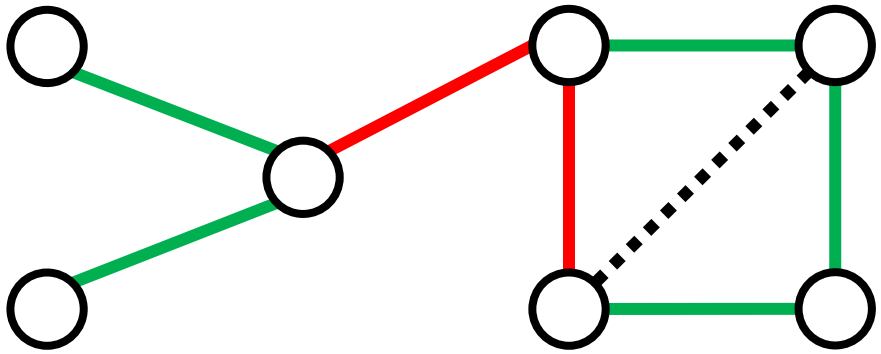
P(correct) = 0.16



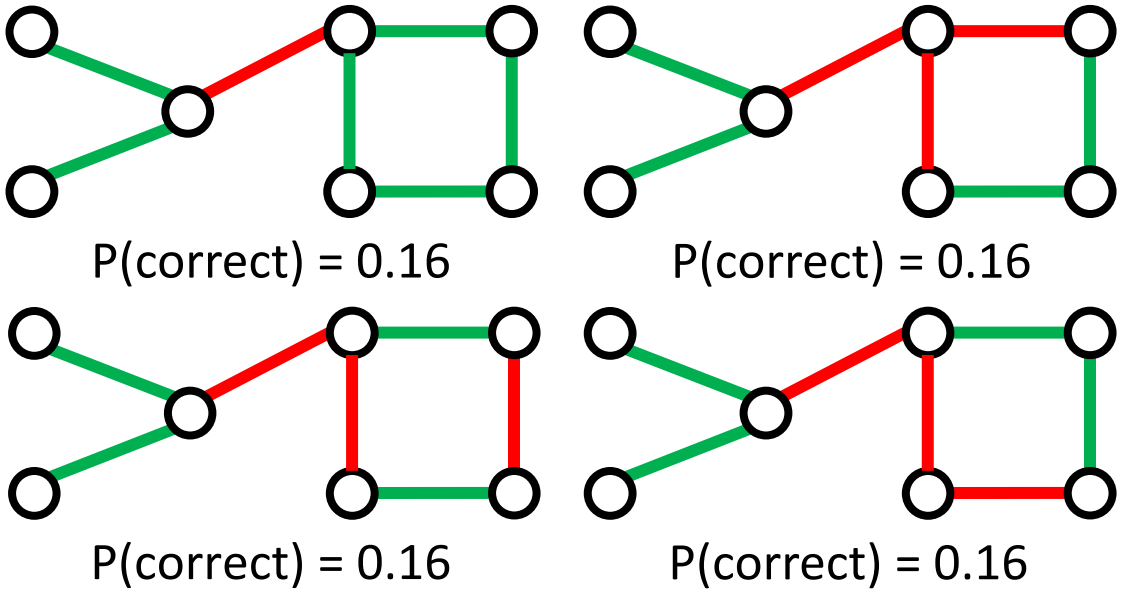
P(correct) = 0.16

Full procedure

User constraints U



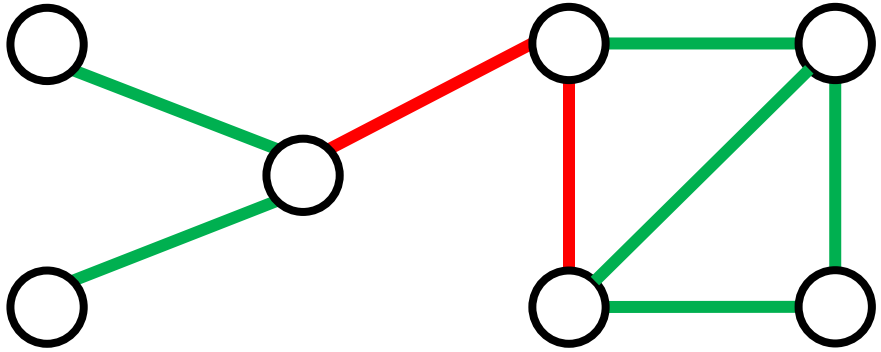
Most likely constraint set C



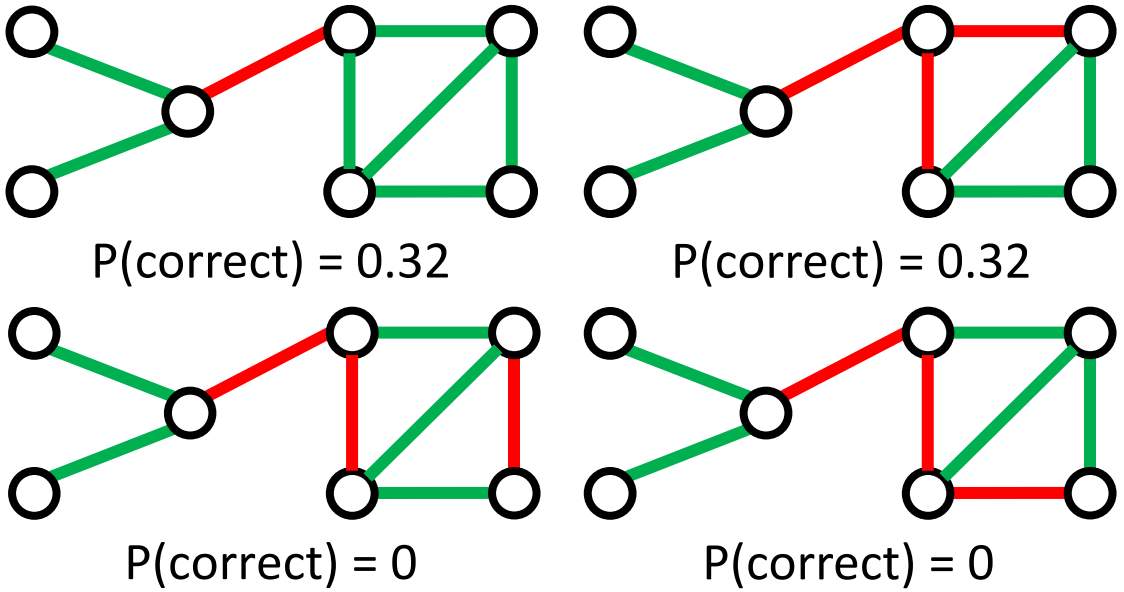
Ask a redundant constraint that eliminates half of the possibilities

Full procedure

User constraints U



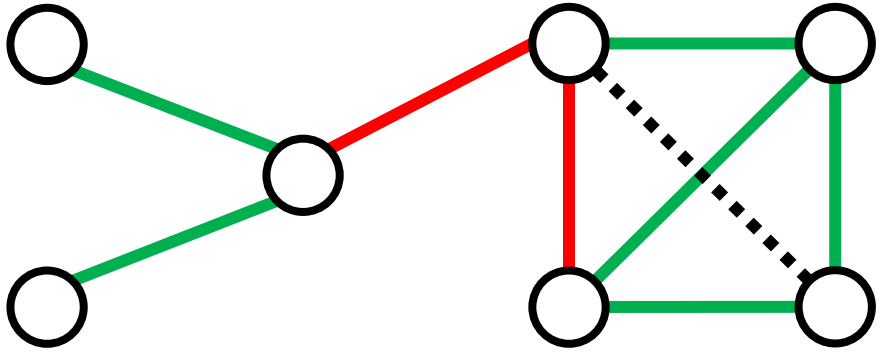
Most likely constraint set C



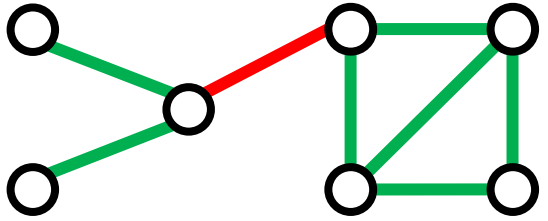
Ask a redundant constraint that eliminates half of the possibilities

Full procedure

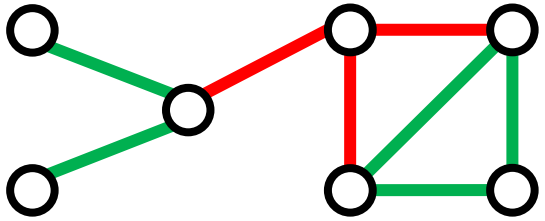
User constraints U



Most likely constraint set C



P(correct) = 0.32

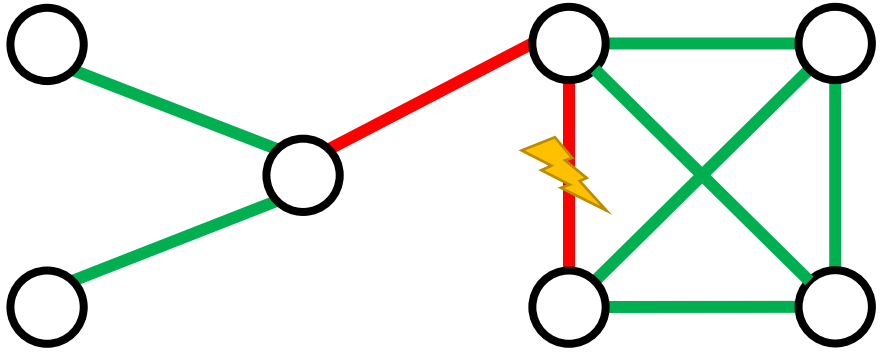


P(correct) = 0.32

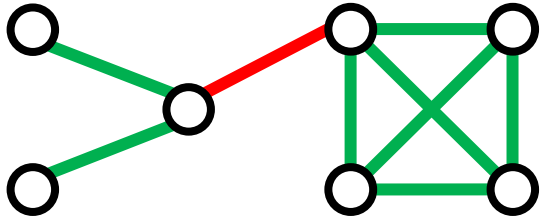
Ask another redundant constraint

Full procedure

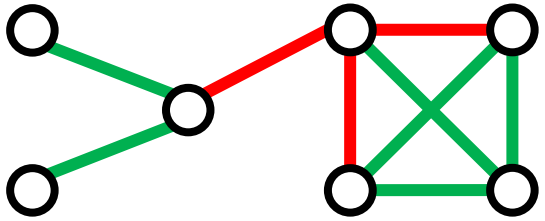
User constraints U



Most likely constraint set C



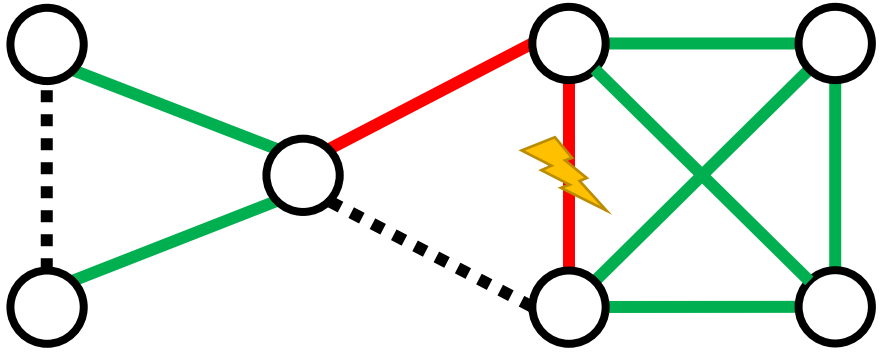
P(correct) = 0.57



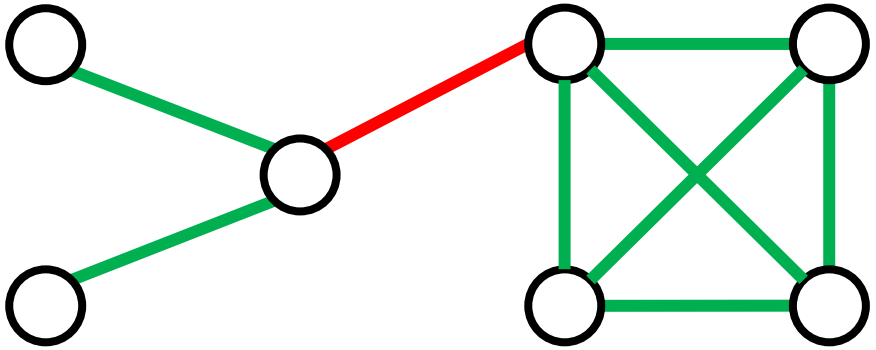
P(correct) = 0

Full procedure

User constraints U



Most likely constraint set C

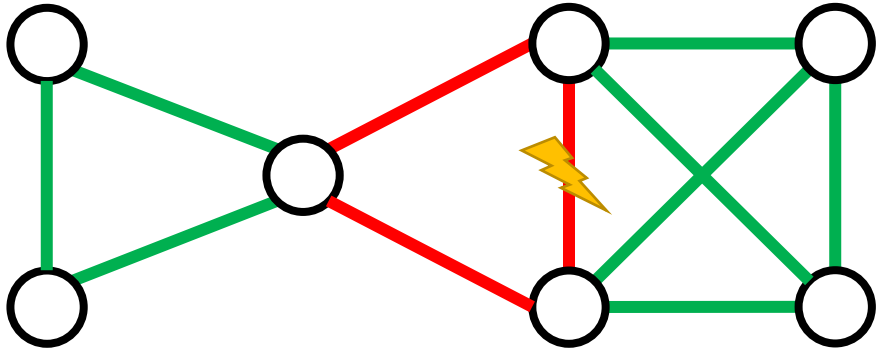


P(correct) = 0.57

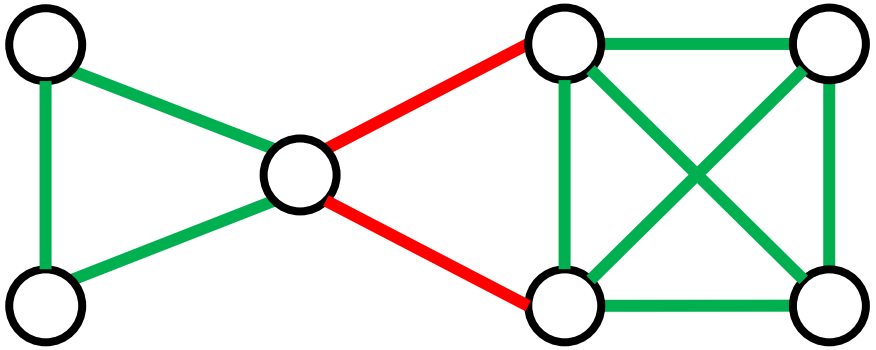
You guessed it: more redundant constraints!

Full procedure

User constraints U



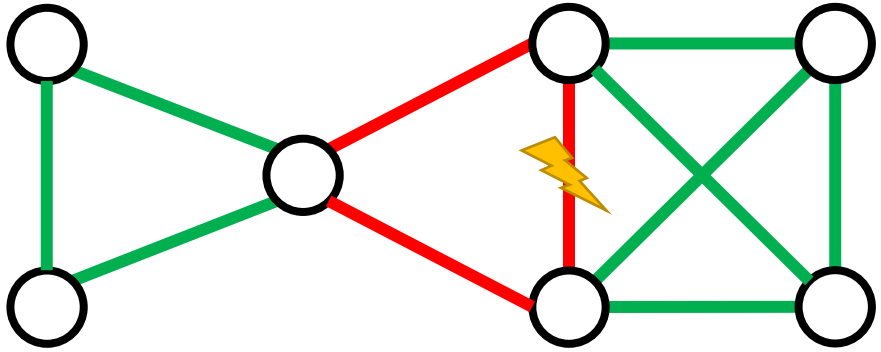
Most likely constraint set C



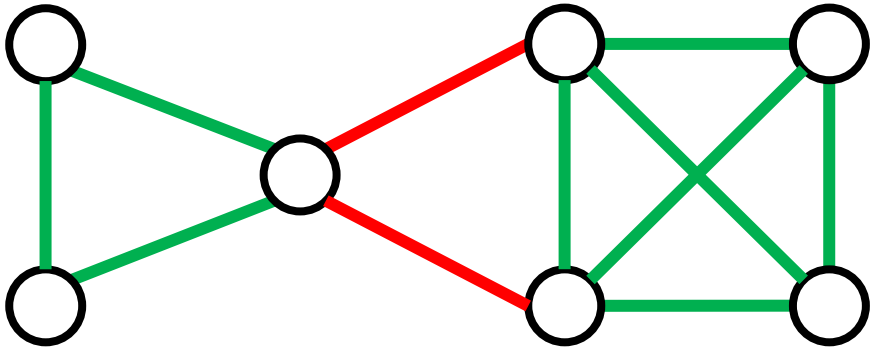
P(correct) = 0.72

Full procedure

User constraints U

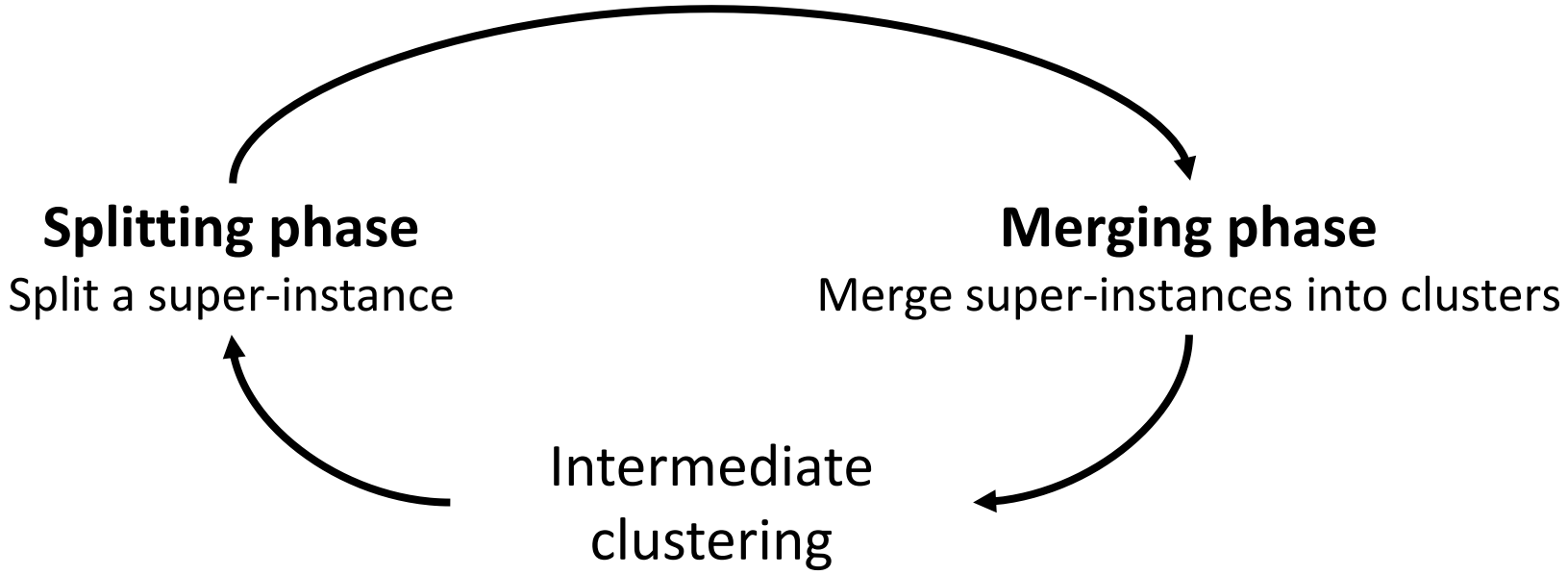


Most likely constraint set C

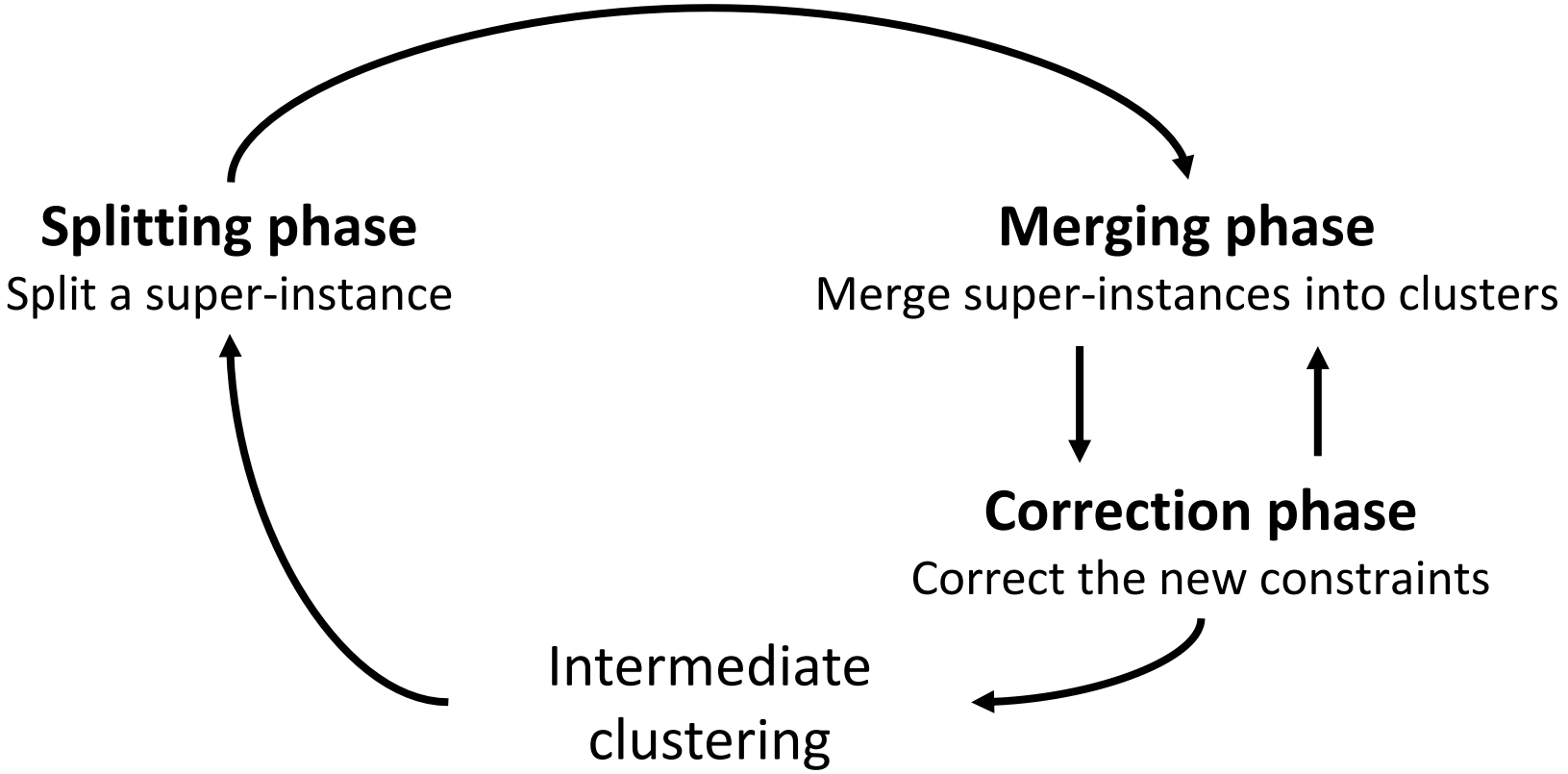


P(correct) = 0.92

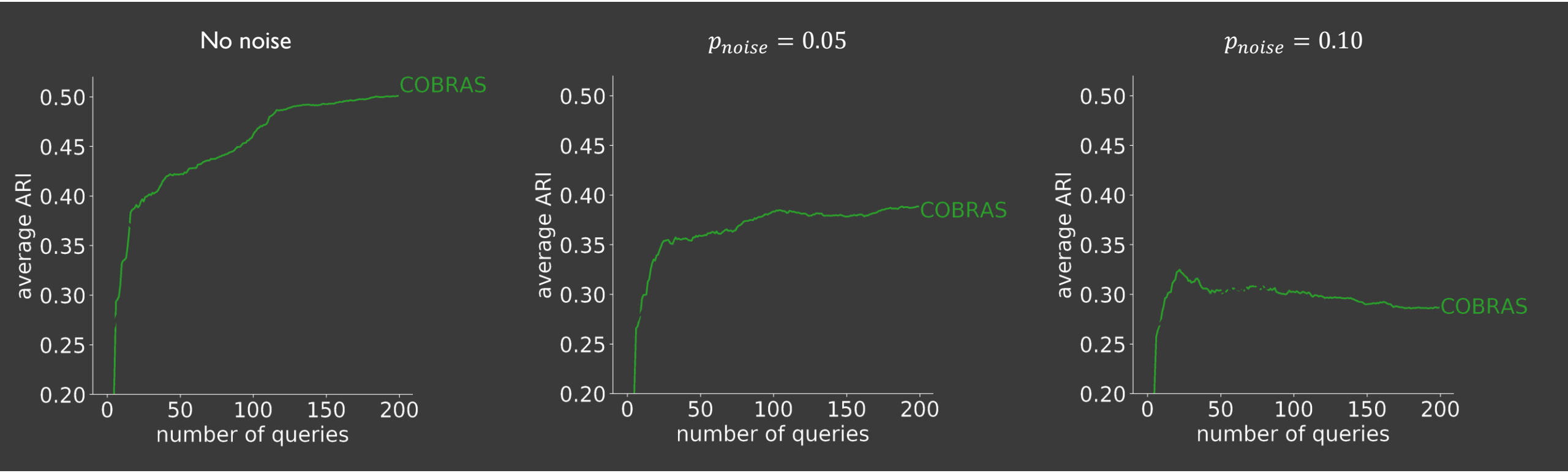
Remark: fully connected must-link components are ignored in the probability calculation


Splitting phase
Split a super-instance
Merging phase
Merge super-instances into clusters
Intermediate clustering

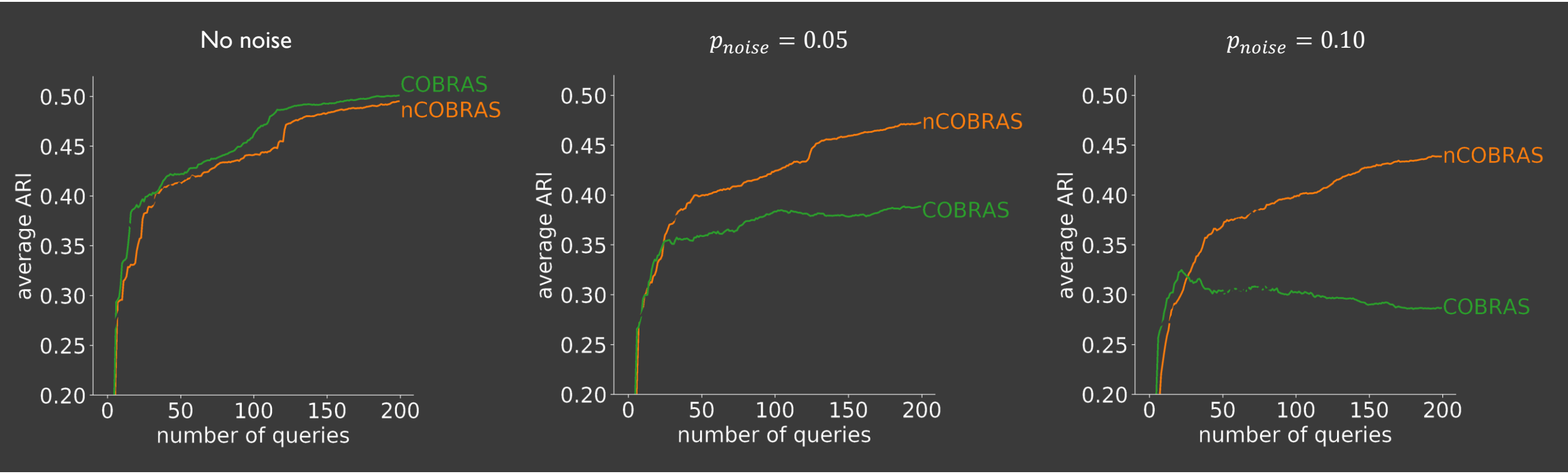
Noise robust cobras is anytime



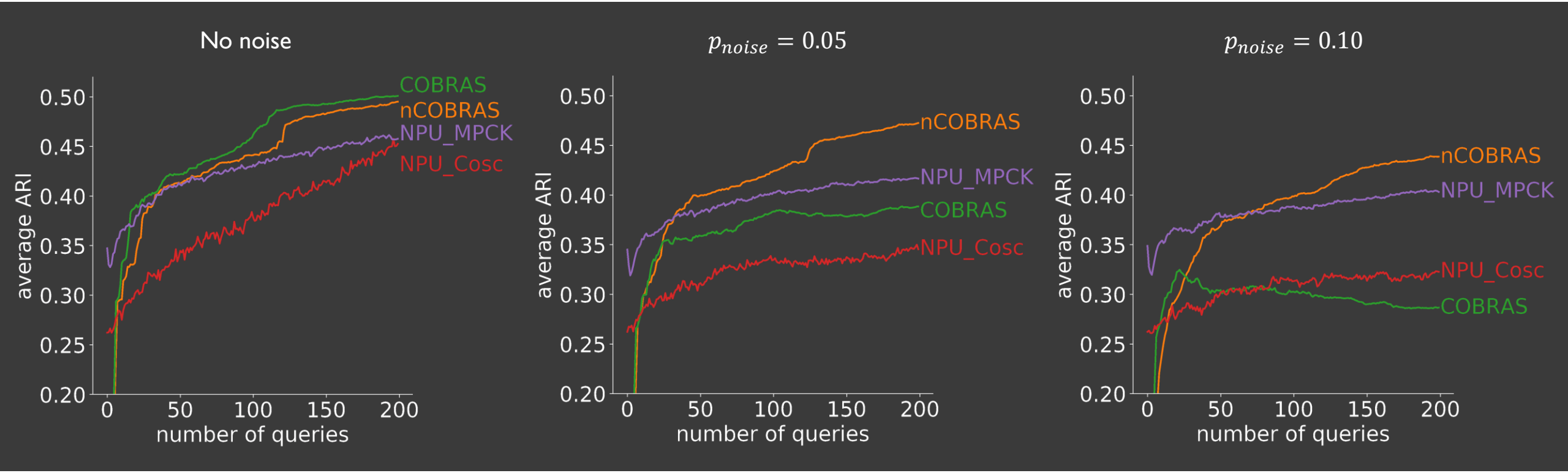
cobras is sensitive to noise



Noise robust cobras is robust against noise

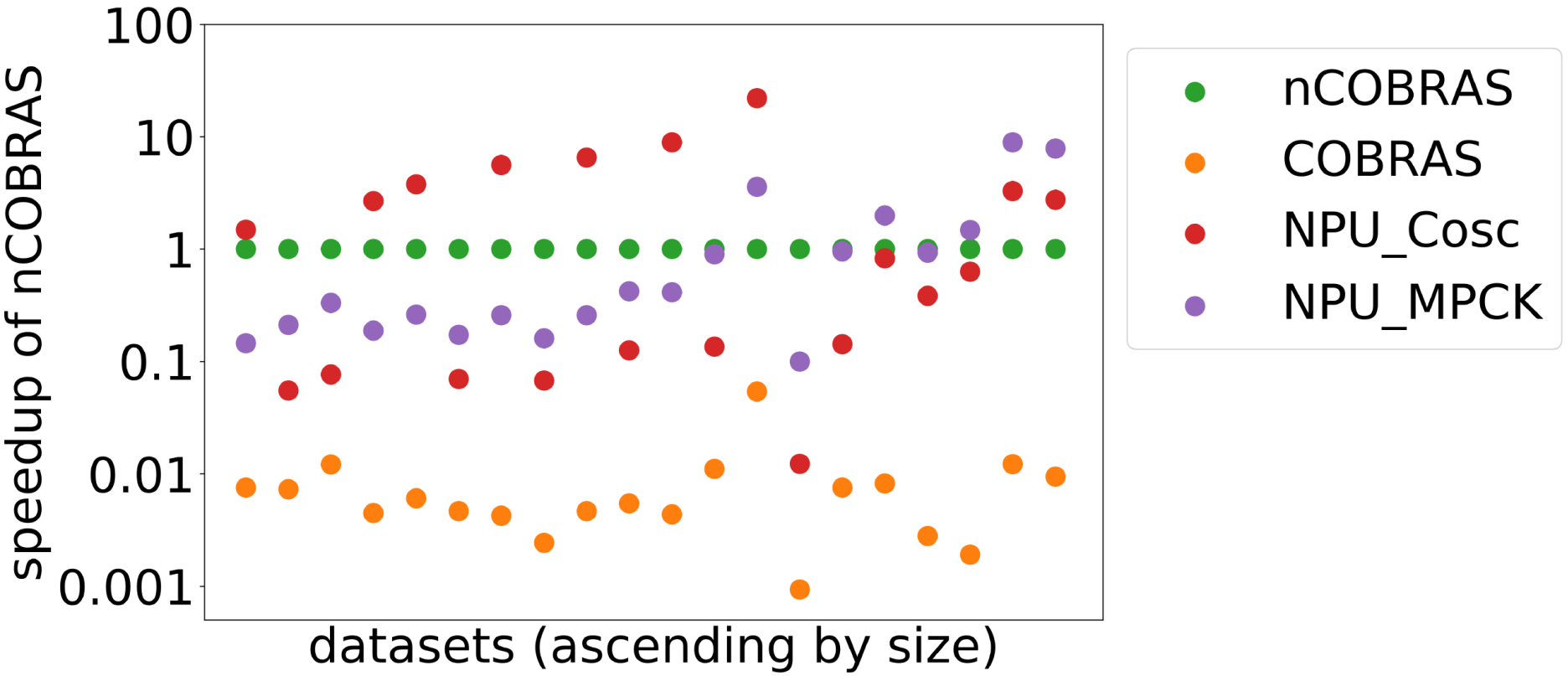


Noise robust cobras is query efficient



nCOBRAS outperforms NPU_MPCK and NPU_COSC

Noise robust cobras is time efficient?



nCOBRAS' correction phase makes it substantially slower than COBRAS, but it is still about as fast as other systems.

Noise robust cobras in a 

Noise robust, Anytime, Query efficient and Time efficient (?)

Detect and **correct** the constraints by
asking additional **redundant** constraints
and **reasoning** probabilistically


Questions?