Machine Learning for Data Science (CS4786)
Lecture 13

Clustering + K-Means Clustering

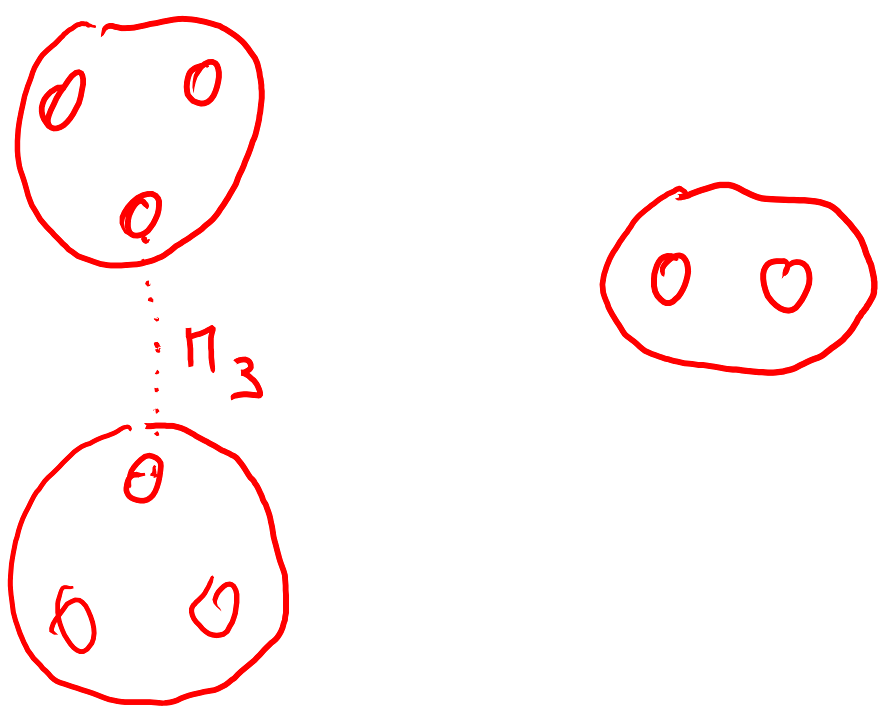
Thorsten Joachims

What does Single-Link HAC optimize?

- Single-link HAC finds the clustering $C_1, \ldots, C_K$ that maximizes the following objective:

$$M_3 = \min_{\boldsymbol{x}_s, \boldsymbol{x}_t : c(\boldsymbol{x}_s) \neq c(\boldsymbol{x}_t)} dissimilarity(\boldsymbol{x}_s, \boldsymbol{x}_t)$$

[Maximize smallest between-cluster distance]

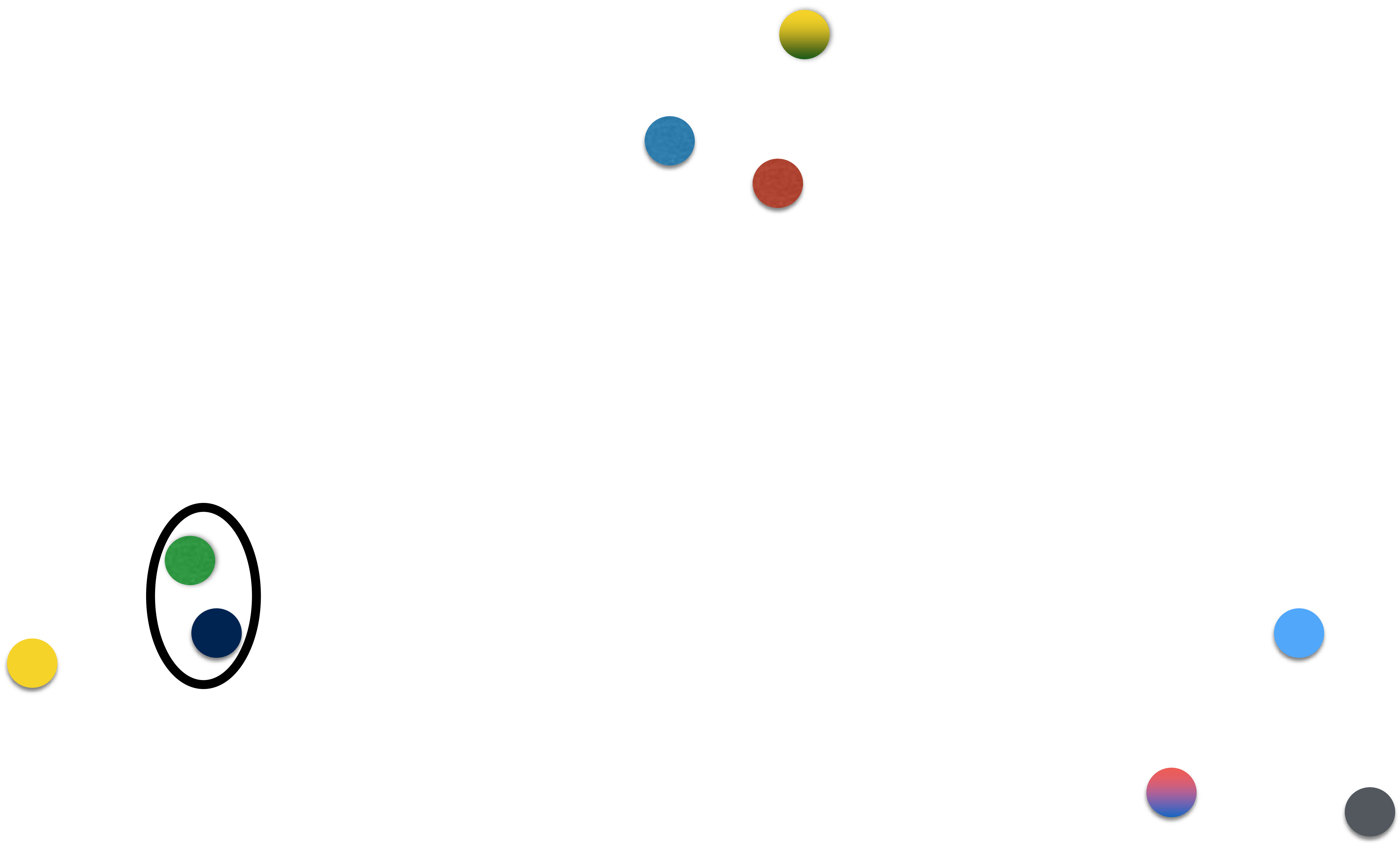


Observation

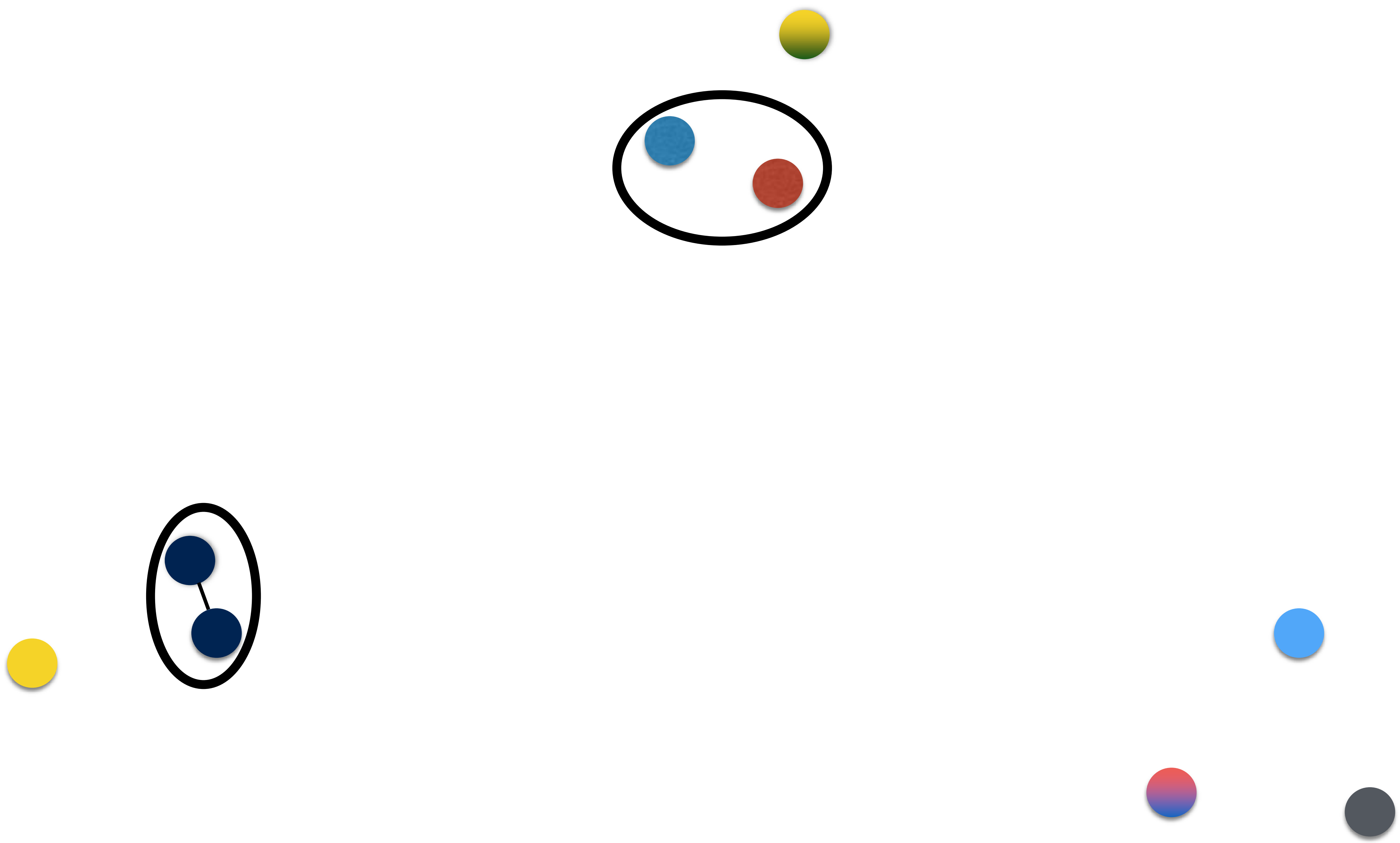
Say c is the clustering produced by single-link HAC. Then it always holds

$$\min_{\boldsymbol{x}_s, \boldsymbol{x}_t : c(\boldsymbol{x}_s) \neq c(\boldsymbol{x}_t)} dissimilarity(\boldsymbol{x}_s, \boldsymbol{x}_t) > \text{merged distances in tree}$$

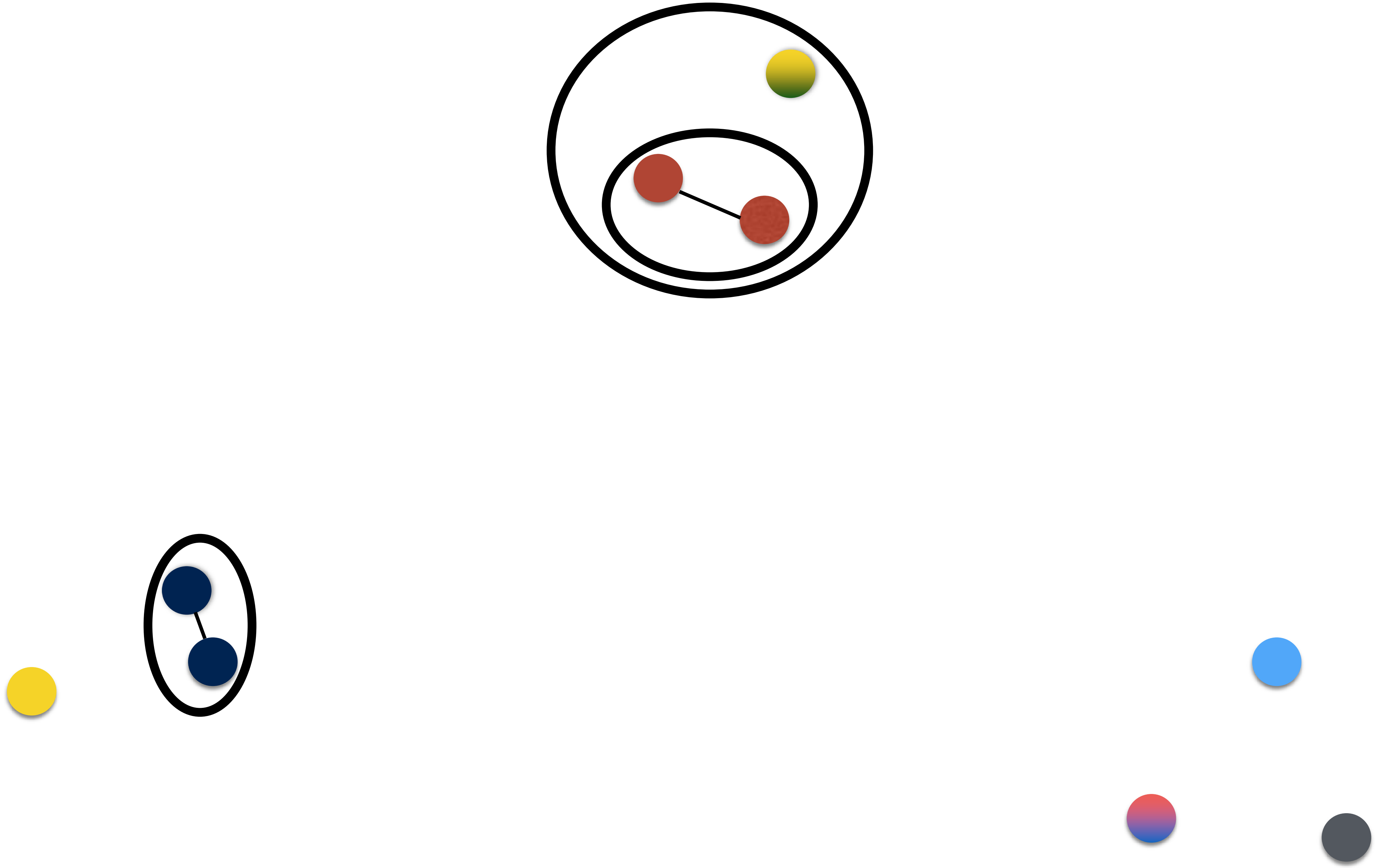
Demo



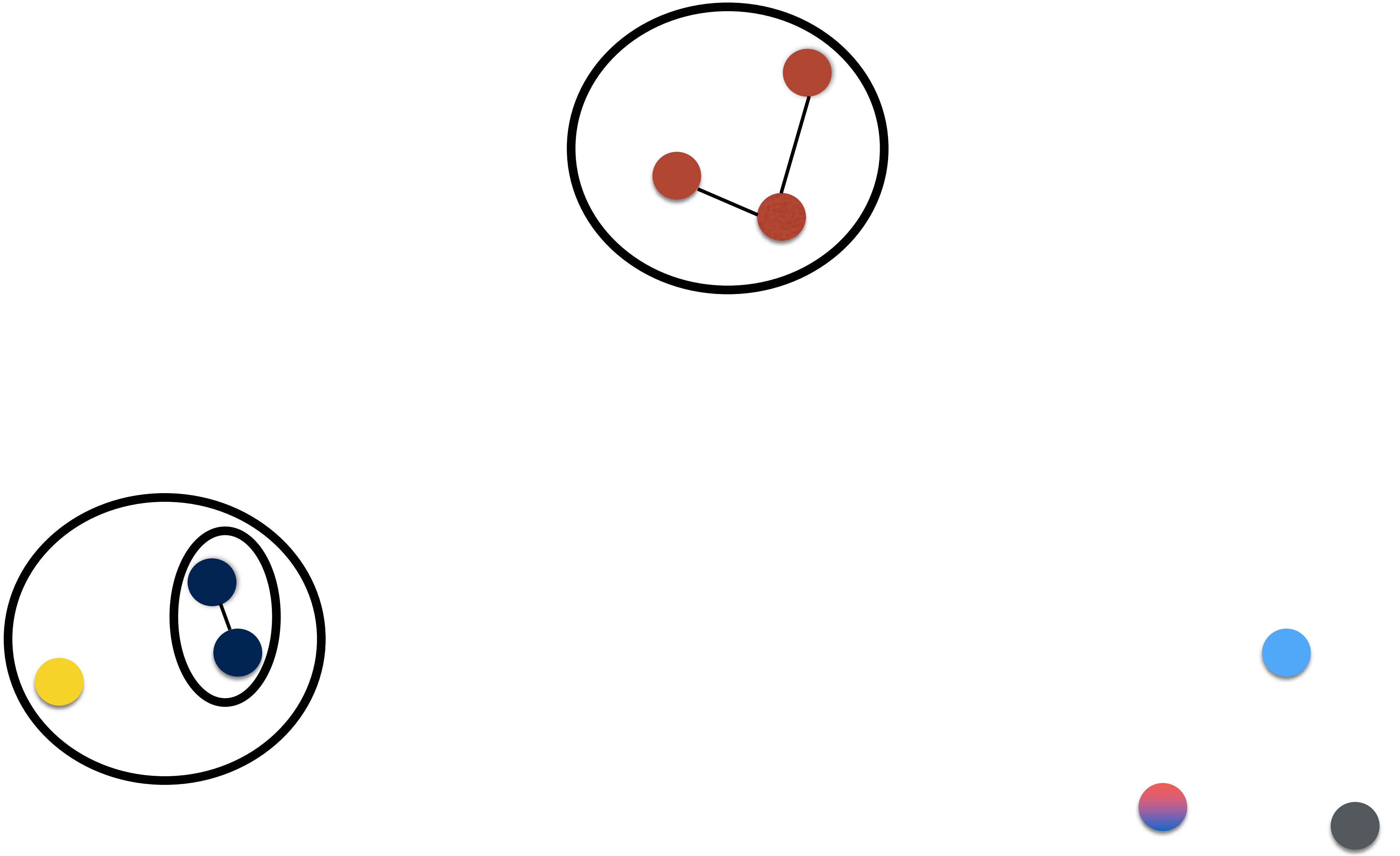
Demo



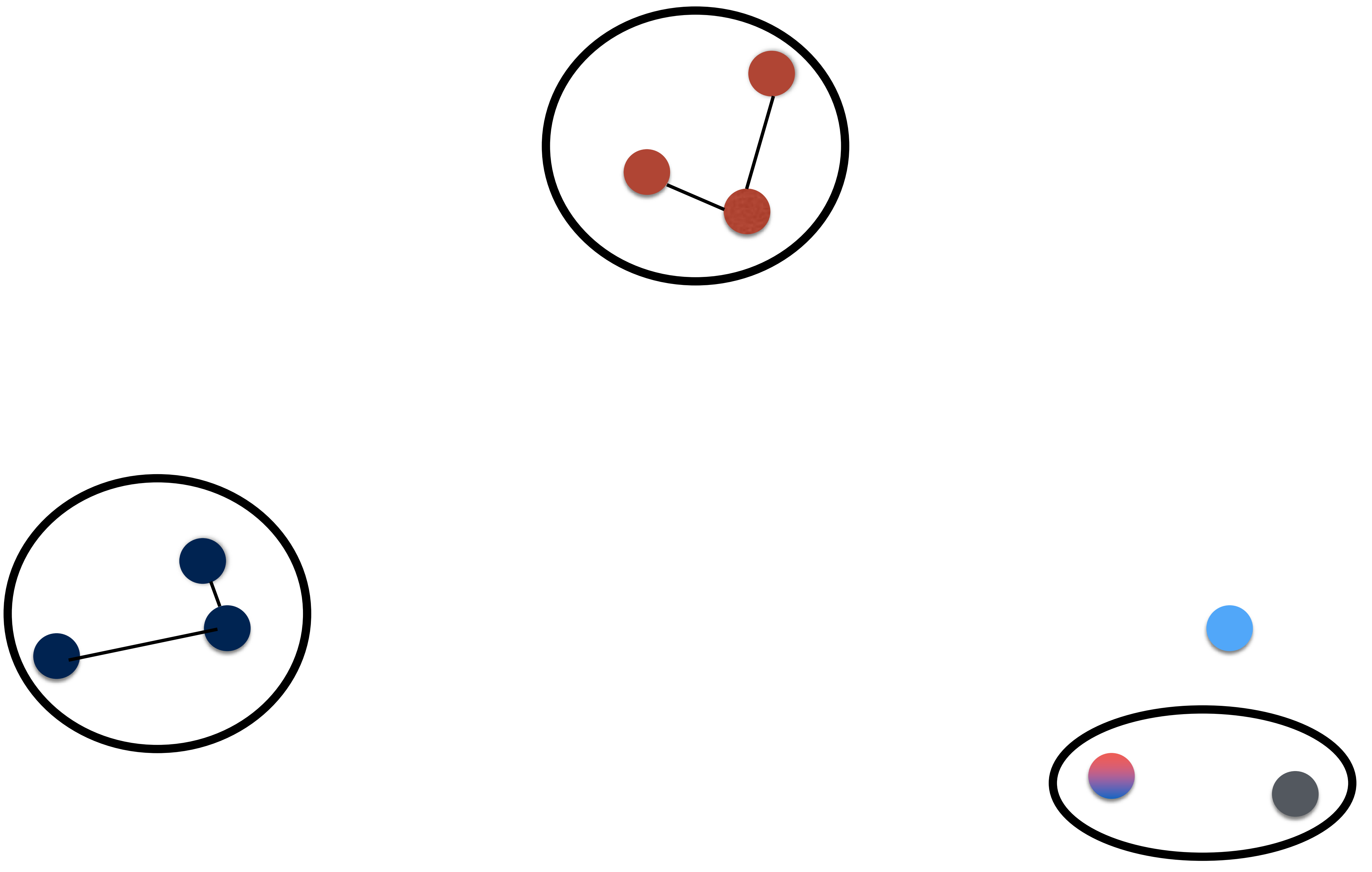
Demo



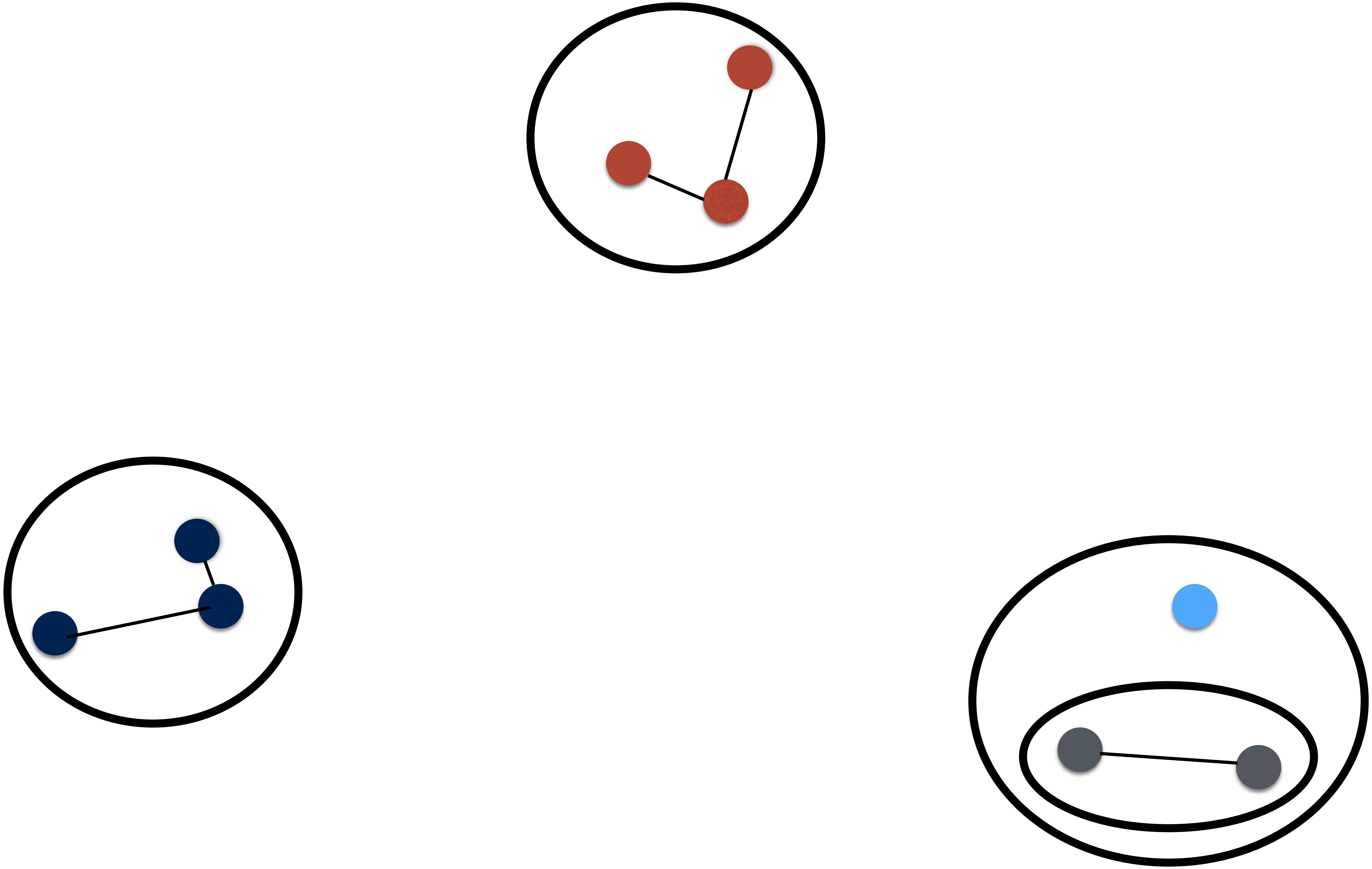
Demo



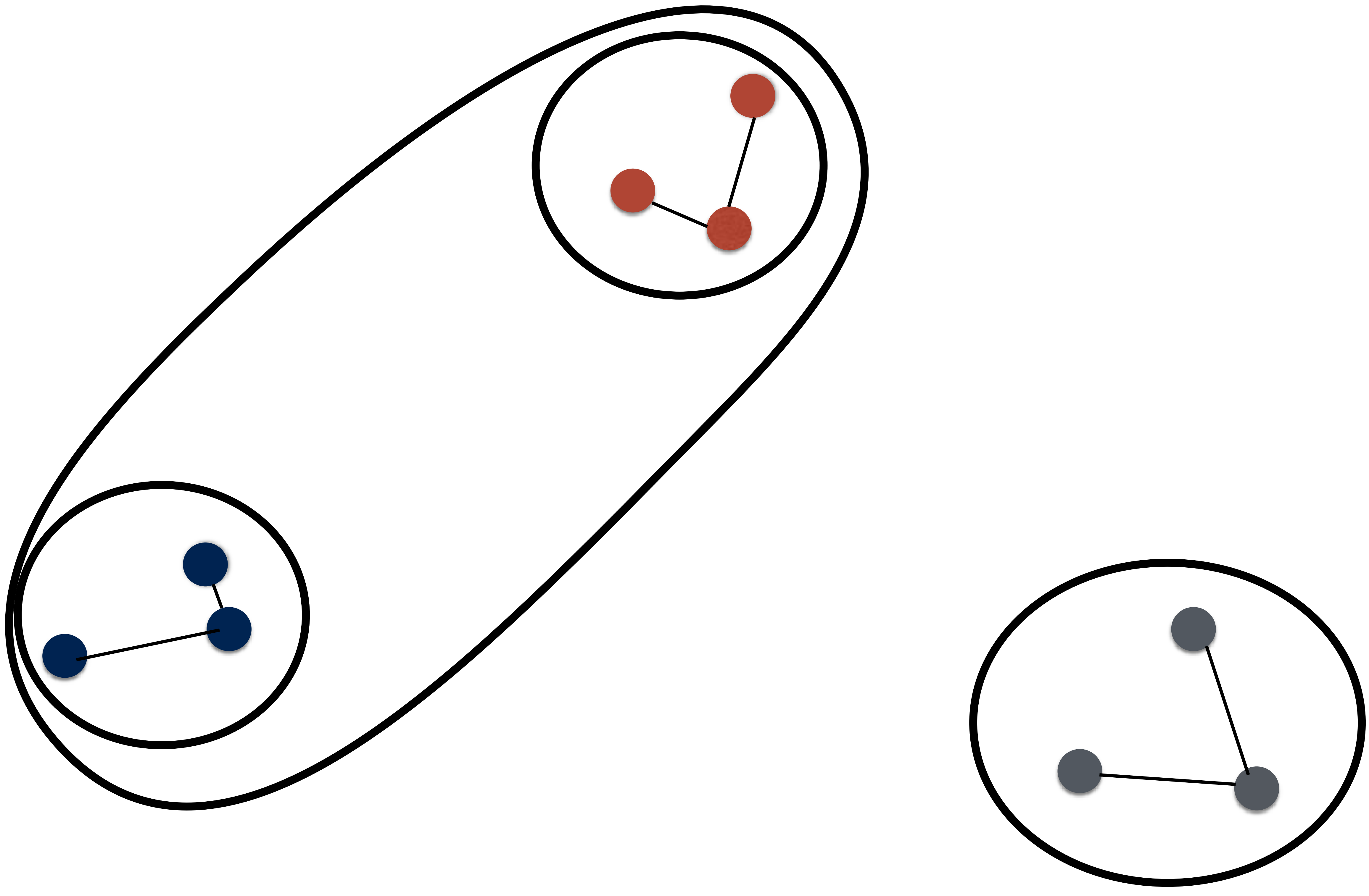
Demo



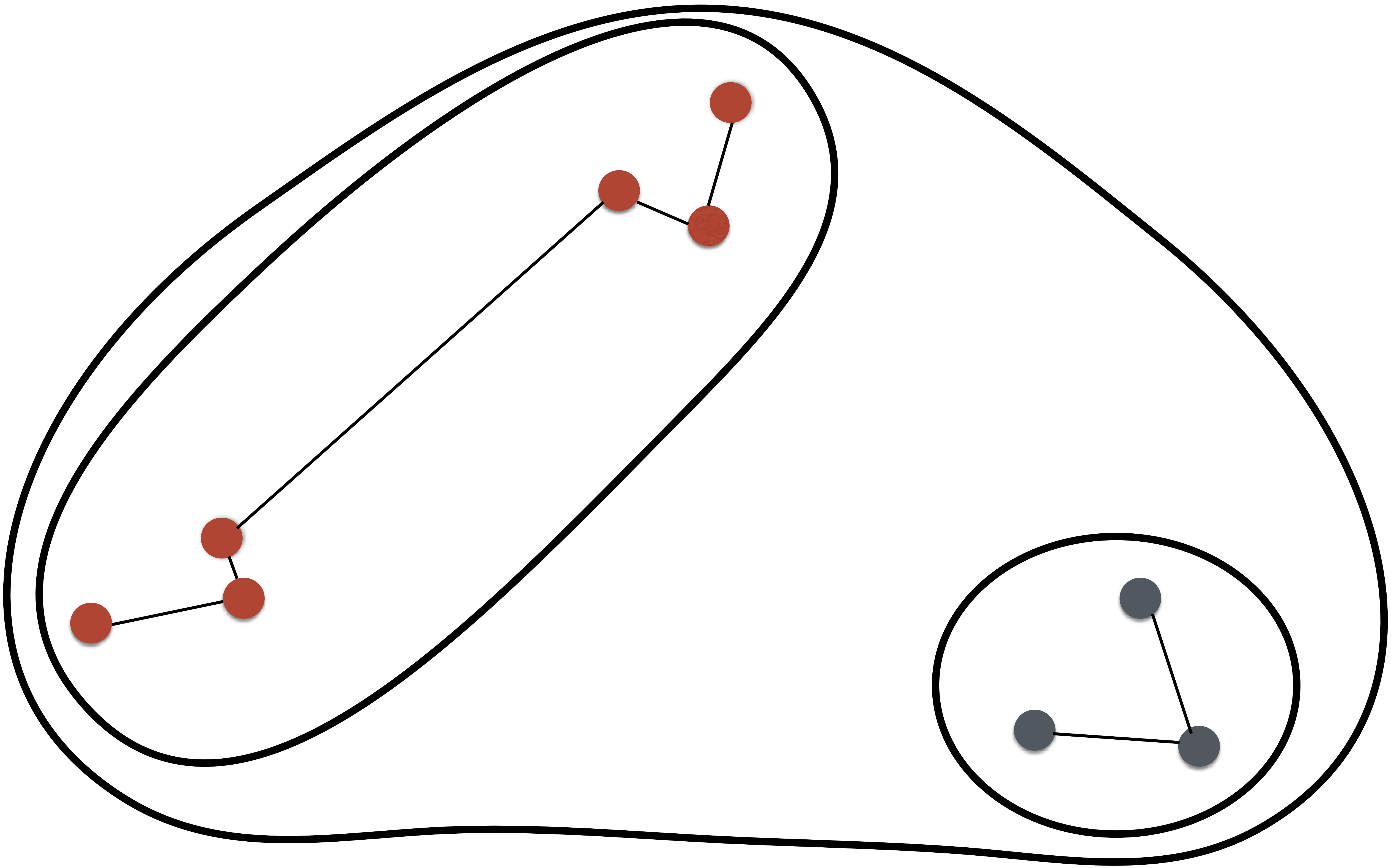
Demo



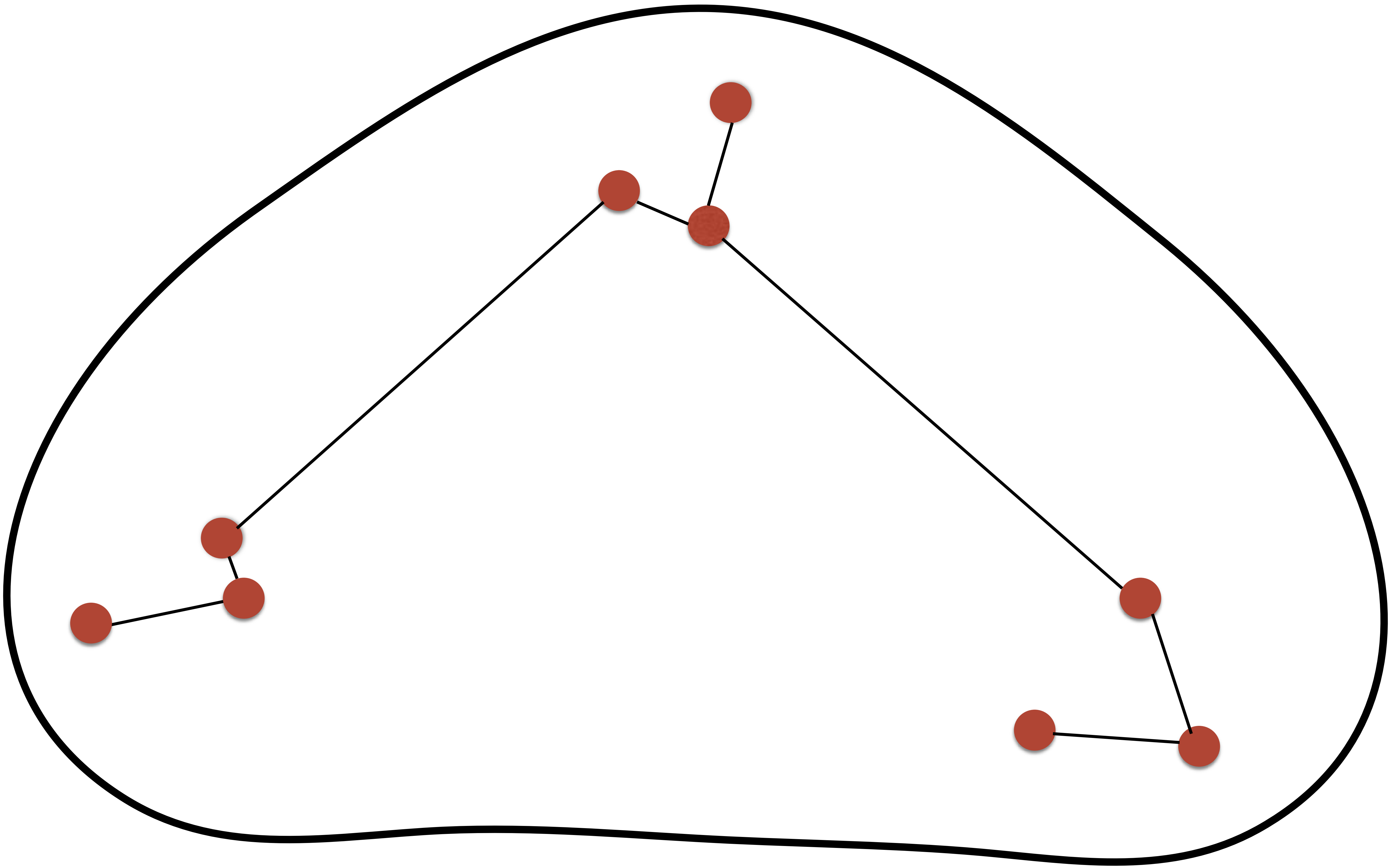
Demo



Demo



Demo



What does Single-Link HAC optimize?

- Single-link HAC finds the clustering $C_1, \ldots, C_K$ that maximizes the following objective:

$$M_3 = \min_{\boldsymbol{x}_s, \boldsymbol{x}_t : c(\boldsymbol{x}_s) \neq c(\boldsymbol{x}_t)} dissimilarity(\boldsymbol{x}_s, \boldsymbol{x}_t)$$

[Maximize smallest between-cluster distance]

Single Link Objective

Proof:

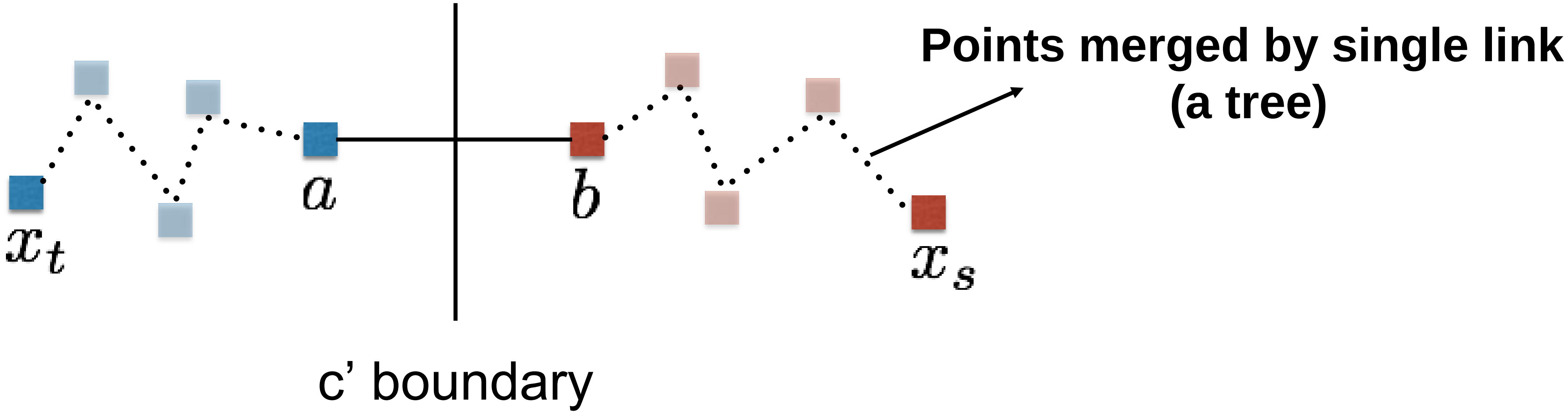
Say c is solution produced by single-link clustering

Key observation:

$$\min_{t,s:c(x_i)\neq c(x_j)} \text{dissimilarity}(x_i, x_j) >$$ Merged distances in tree

Say $c' \neq c$ then,

$$\exists\ t, s \text{ s.t. } c'(x_t) \neq c'(x_s) \text{ but } c(x_t) = c(x_s)$$



Summary

- Clustering: Find partitioning of the data points.
- HAC: Repeatedly merge clusters bottom up.
  - Design: Distance measure and merge criterion.
  - Efficiency: $O(n^2 \log n)$
- Single-link optimizes M_3 criterion.

K-means Clustering

- For all $j \in [K]$, initialize cluster centroids $\hat{\mathbf{r}}_j^0$ randomly and set $m = 1$
- Repeat until convergence (or until patience runs out)
  1. For each $t \in \{1, \ldots, n\}$, set cluster identity of the point

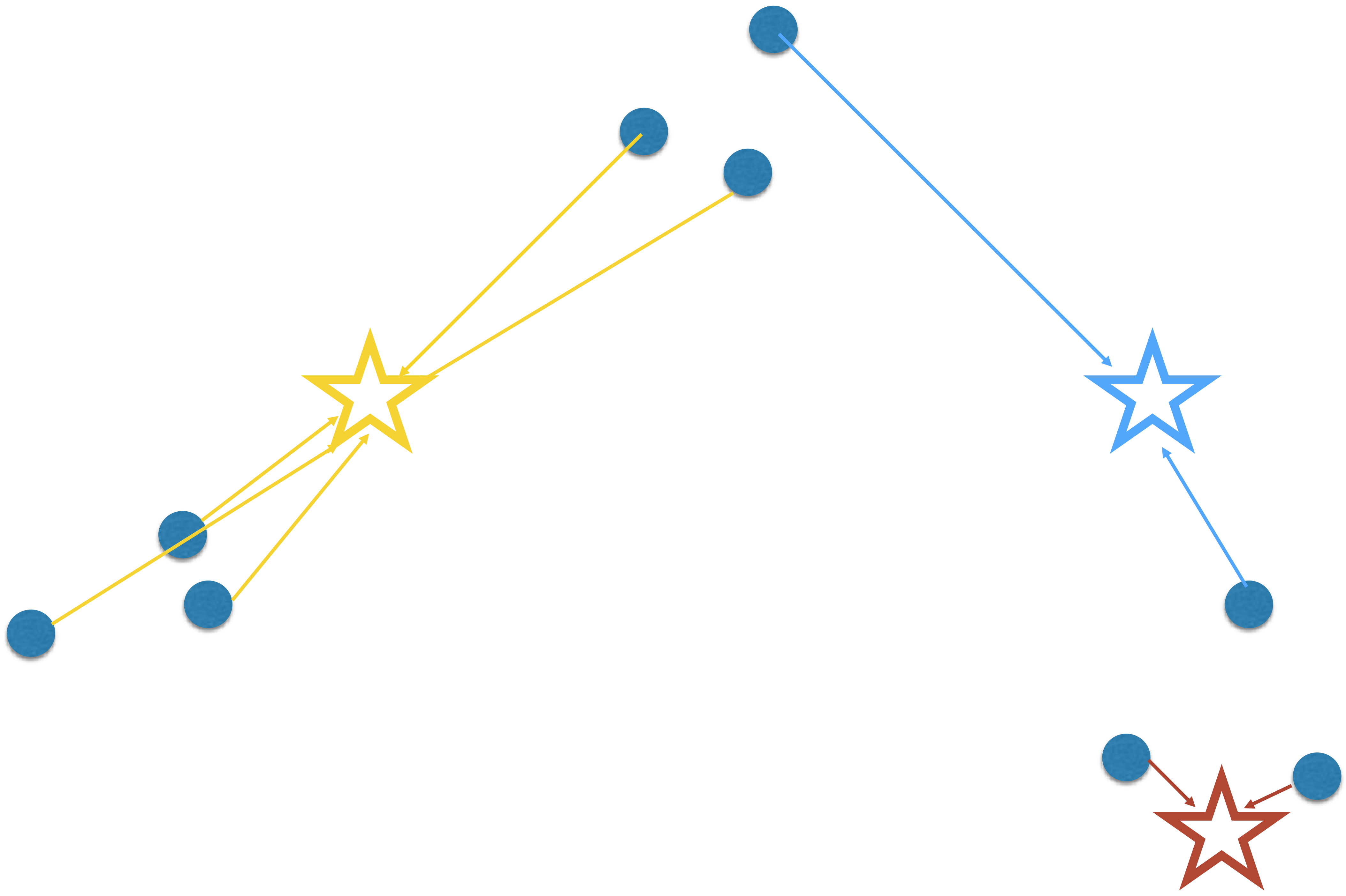
$$\hat{c}^m(\mathbf{x}_t) = \operatorname*{argmin}_{j \in [K]} \|\mathbf{x}_t - \hat{\mathbf{r}}_j^{m-1}\|$$

  2. For each $j \in [K]$, set new representative as

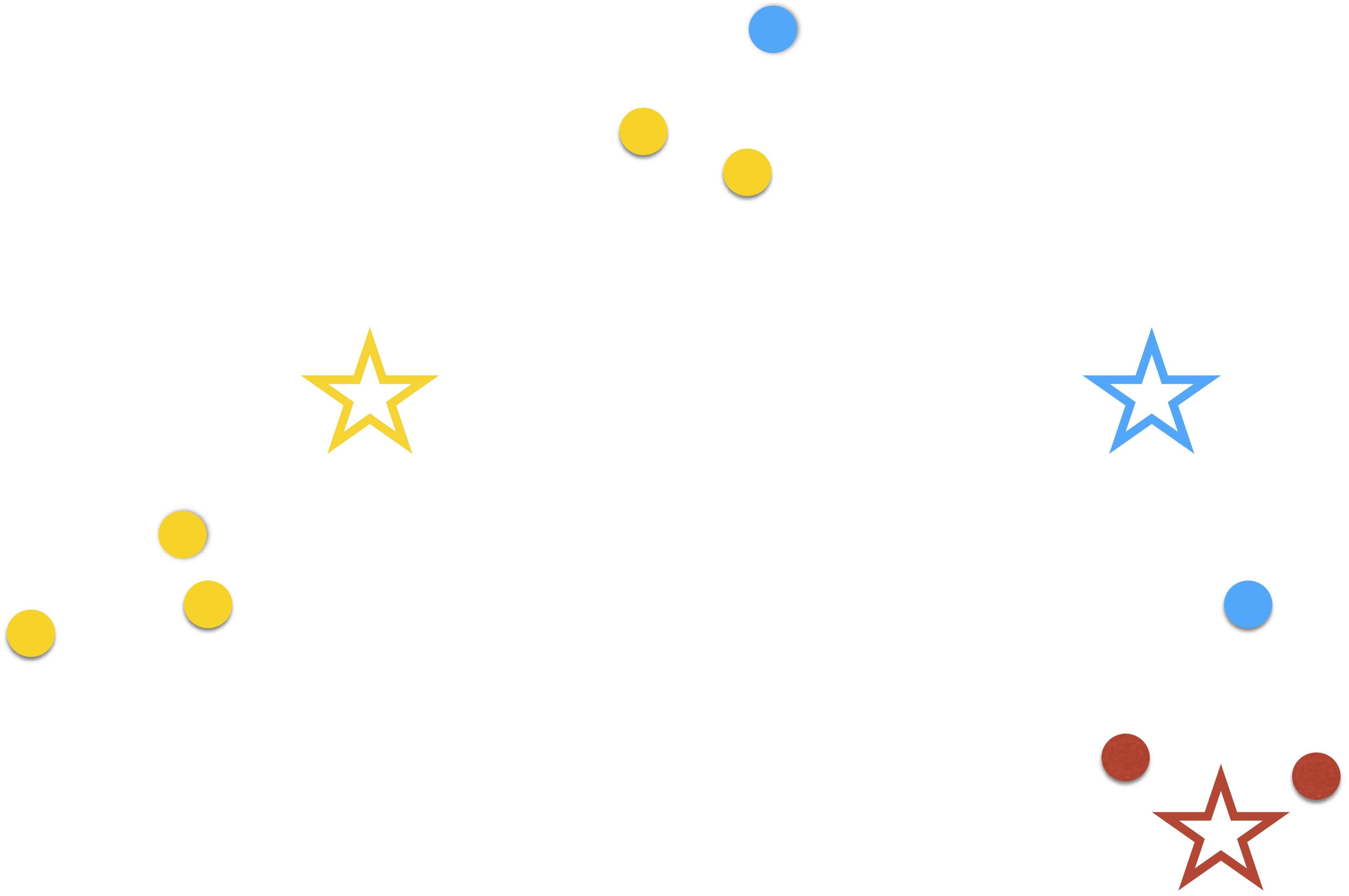
$$\hat{\mathbf{r}}_j^m = \frac{1}{|\hat{C}_j^m|} \sum_{\mathbf{x}_t \in \hat{C}_j^m} \mathbf{x}_t$$

  3. $m \leftarrow m + 1$

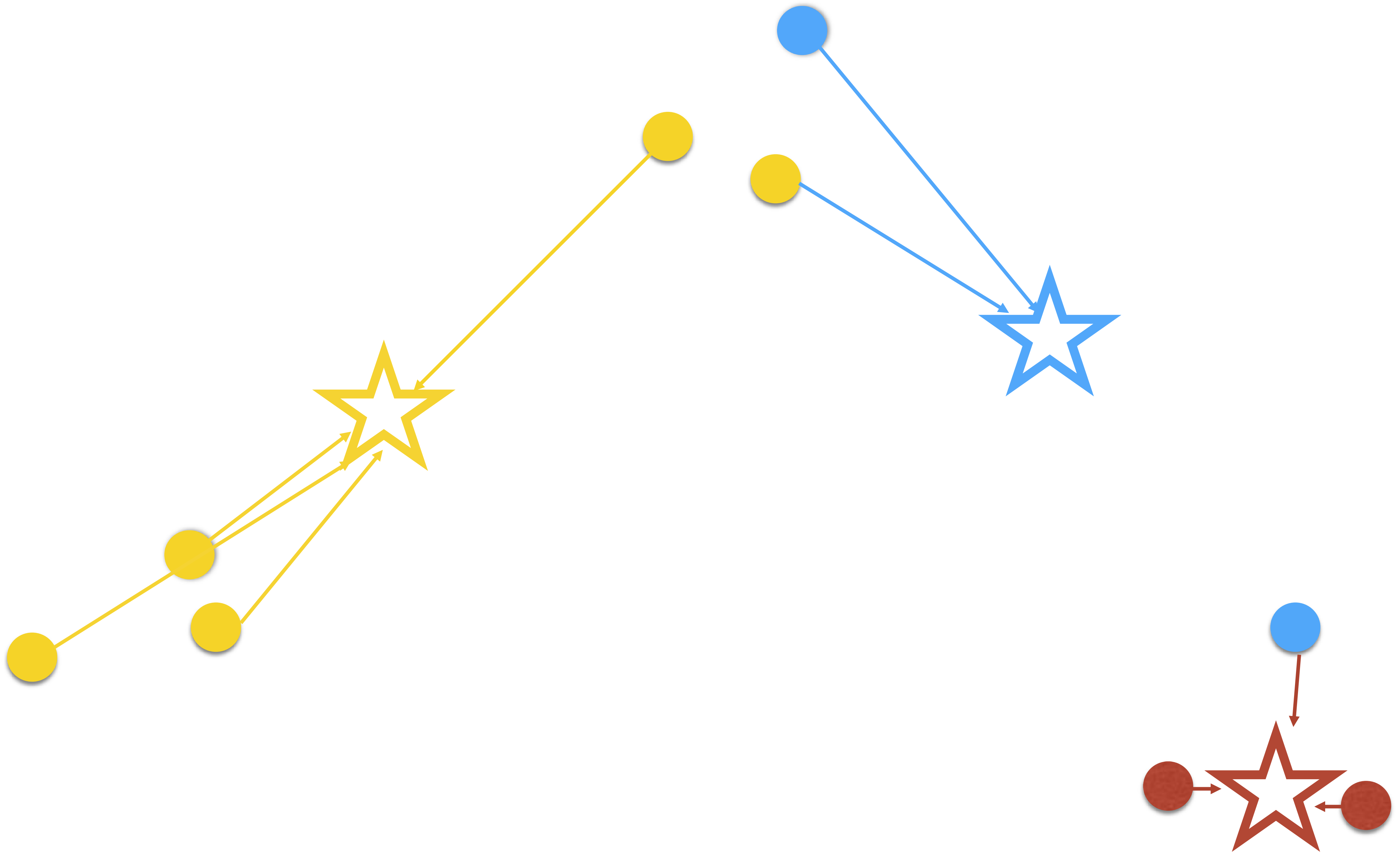
Demo



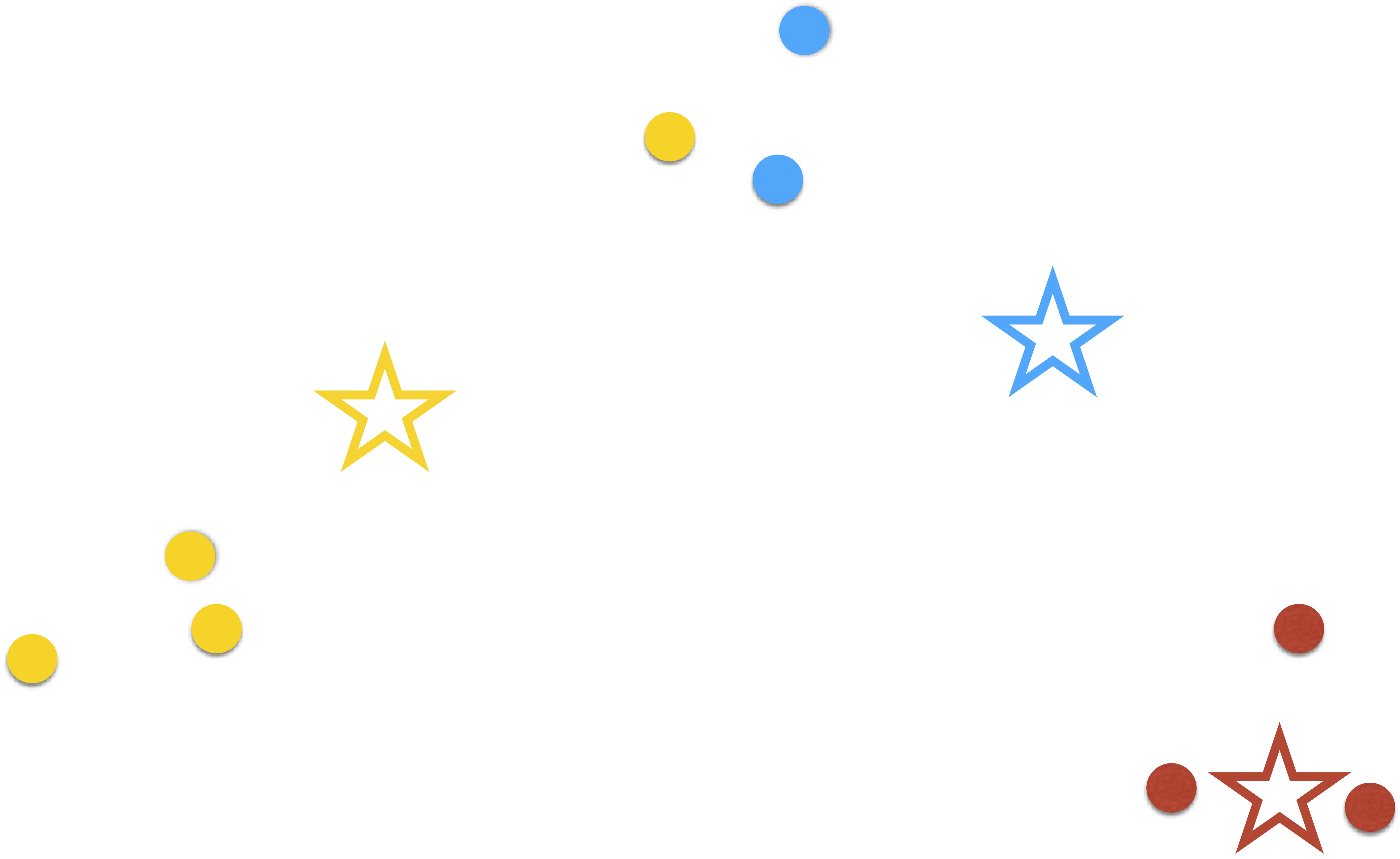
Demo



Demo



Demo



Demo

Demo

Demo

K-MEANS CLUSTERING

- For all $j \in [K]$, initialize cluster centroids $\hat{\mathbf{r}}_j^0$ randomly and set $m = 1$
- Repeat until convergence (or until patience runs out)
  1. For each $t \in \{1, \ldots, n\}$, set cluster identity of the point

$$\hat{c}^m(\mathbf{x}_t) = \underset{j \in [K]}{\operatorname{argmin}} \|\mathbf{x}_t - \hat{\mathbf{r}}_j^{m-1}\|$$

  2. For each $j \in [K]$, set new representative as

$$\hat{\mathbf{r}}_j^m = \frac{1}{|\hat{C}_j^m|} \sum_{\mathbf{x}_t \in \hat{C}_j^m} \mathbf{x}_t$$

  3. $m \leftarrow m + 1$

Question: Does the K-Means algorithm always terminate, or can it run into an infinite loop?

K-MEANS CONVERGENCE

- K-means algorithm converges to local minima of objective

$$O(c; \mathbf{r}_1, \ldots, \mathbf{r}_K) = \sum_{j=1}^{K} \sum_{c(\mathbf{x}_t)=j} \|\mathbf{x}_t - \mathbf{r}_j\|_2^2$$

- Proof:

$$\hat{c}^m(x_t) = \underset{j \in \{1..K\}}{\arg\min} \left(x_t - \hat{r}_j^{m-1}\right)^2$$

K-means Convergence

- K-means algorithm converges to local minima of objective

$$O(c; \mathbf{r}_1, \ldots, \mathbf{r}_K) = \sum_{j=1}^{K} \sum_{c(\mathbf{x}_t)=j} \|\mathbf{x}_t - \mathbf{r}_j\|_2^2$$

- Proof:
  Clustering assignment improves objective:

$$O\left(\hat{c}^{m-1}; \mathbf{r}_1^{m-1}, \ldots, \mathbf{r}_K^{m-1}\right) \geq O\left(\hat{c}^{m}; \mathbf{r}_1^{m-1}, \ldots, \mathbf{r}_K^{m-1}\right)$$

  (By definition of $\hat{c}^m(\mathbf{x}_t)$)
  Computing centroids improves objective:

$$O\left(\hat{c}^{m}; \mathbf{r}_1^{m-1}, \ldots, \mathbf{r}_K^{m-1}\right) \geq O\left(\hat{c}^{m}; \mathbf{r}_1^{m}, \ldots, \mathbf{r}_K^{m}\right)$$

  (By the fact about centroid)

Fact: Centroid is Minimizer

$$\forall \mathbf{r}_j, \quad \sum_{t \in C_j} \left\| \mathbf{x}_t - \overbrace{\frac{1}{|C_j|} \sum_{s \in C_j} \mathbf{x}_s}^{\text{centroid}} \right\|^2 \leq \sum_{t \in C_j} \|\mathbf{x}_t - \mathbf{r}_j\|^2$$

$$0 = \frac{\partial \sum_{t \in C_j} (x_t - r_j)^2}{\partial r_j} = \sum_{t \in C_j} 2(x_t - r_j) = -2|C_j| r_j + 2 \sum_{t \in C_j} x_t$$

$$\Leftrightarrow \cancel{2} |C_j| r_j = \cancel{2} \sum_{t \in C_j} x_t$$

$$r_j = \frac{1}{|C_j|} \sum_{t \in C_j} x_t \quad \leftarrow \text{minimum } r_j$$

Time Complexity

- Assume computing distance between two instances is O(d) where d is the dimensionality of the vectors.
- Reassigning clusters for n points: O(Kn) distance computations, or O(Knd).
- Computing centroids: Each instance gets added once to some centroid: O(nd).
- Assume these two steps are each done once for i iterations: O($iKnd$).
- Linear in all relevant factors, assuming a fixed number of iterations.

Buckshot Algorithm

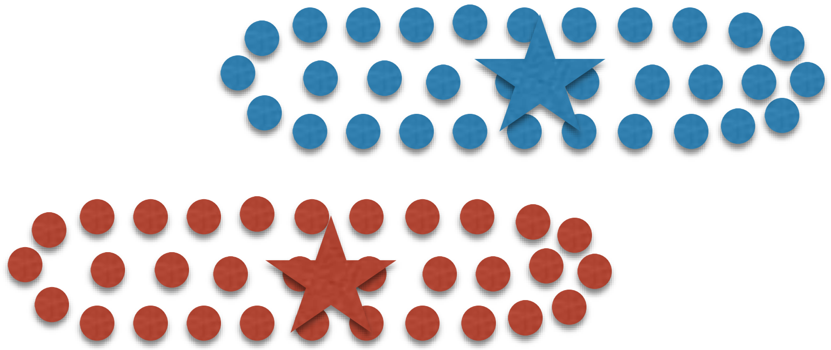
Problem

- Results can vary based on random seed selection, especially for high-dimensional data.
- Some seeds can result in poor convergence rate, or convergence to sub-optimal clusterings.

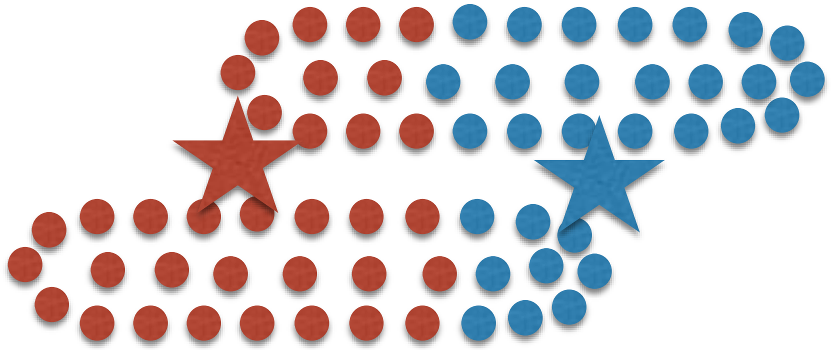
Idea: Combine HAC and K-means clustering.

- First randomly take a sample of instances of size $n^{1/2}$
- Run average-link HAC on this sample
- Use the results of HAC as initial seeds for K-means.
- Overall algorithm is efficient and avoids problems of bad seed selection.

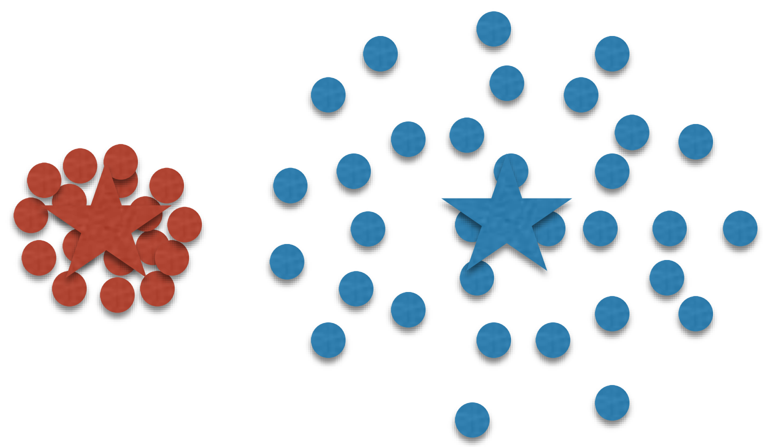
K-means: pitfalls



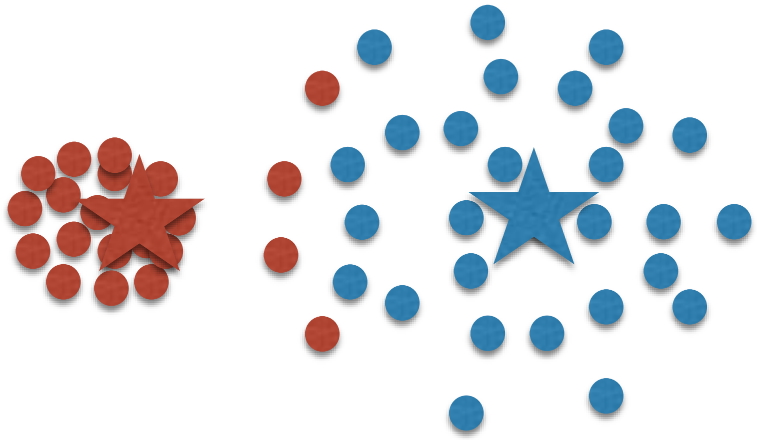
K-means: pitfalls



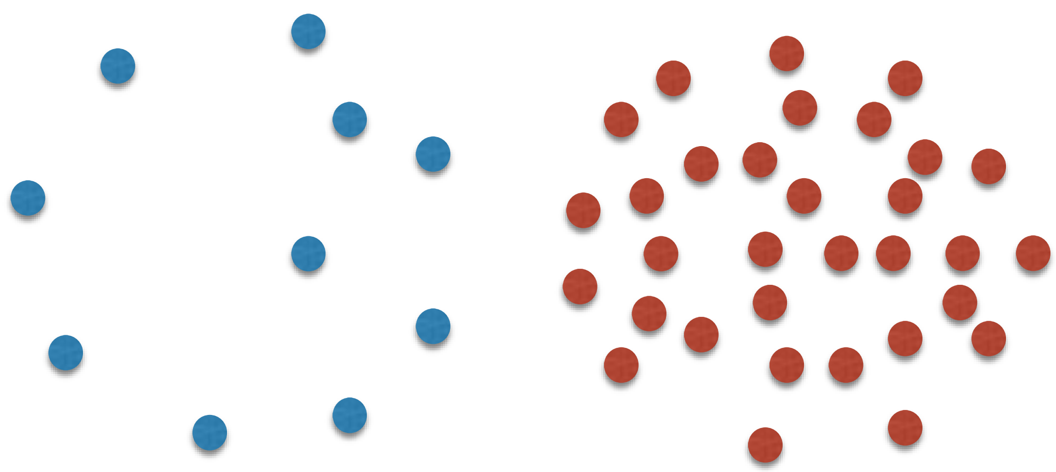
K-means: pitfalls



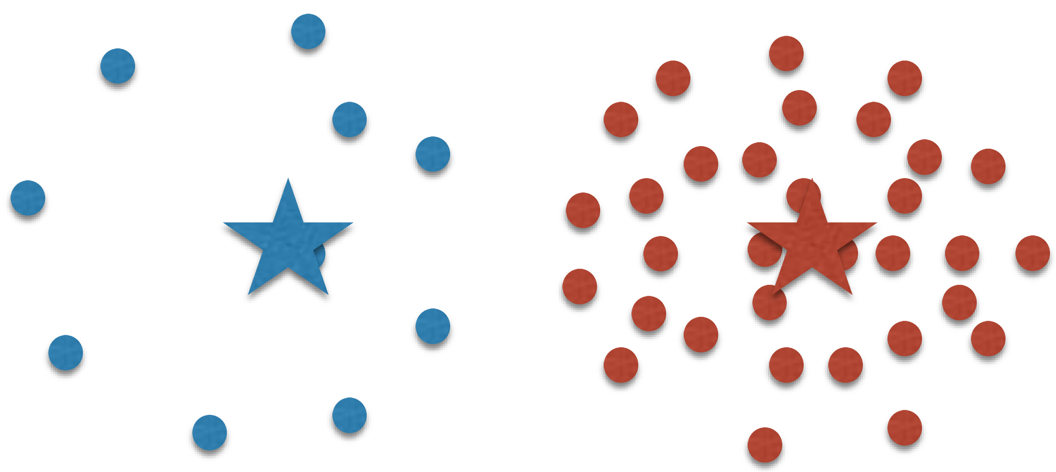
K-means: pitfalls



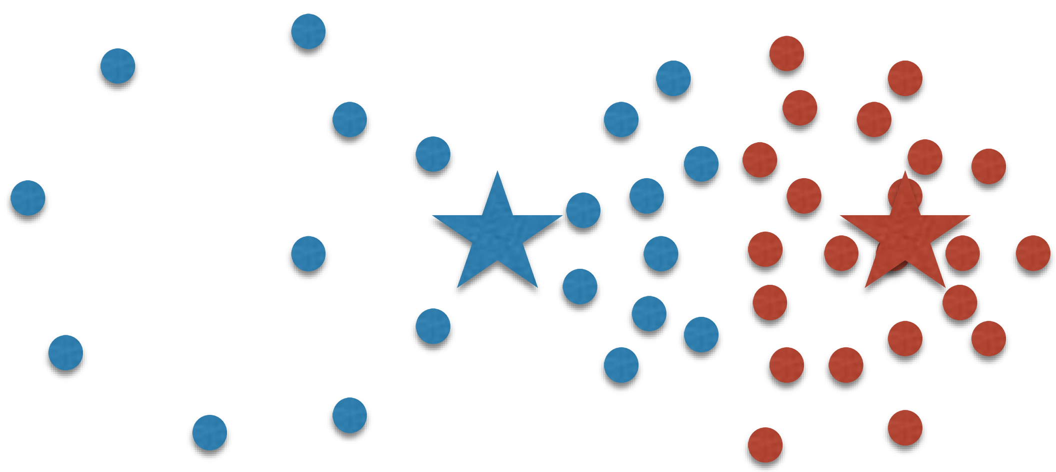
K-means: pitfalls



K-means: pitfalls



K-means: pitfalls



K-means: pitfalls

- Looks for spherical clusters
- Of same radius
- And with roughly equal number of points

K-means: pitfalls

- Can we design algorithm that can address these shortcomings?