Clustering, prototypes and beyond

Ludwig Krippahl

Clustering, prototypes and beyond

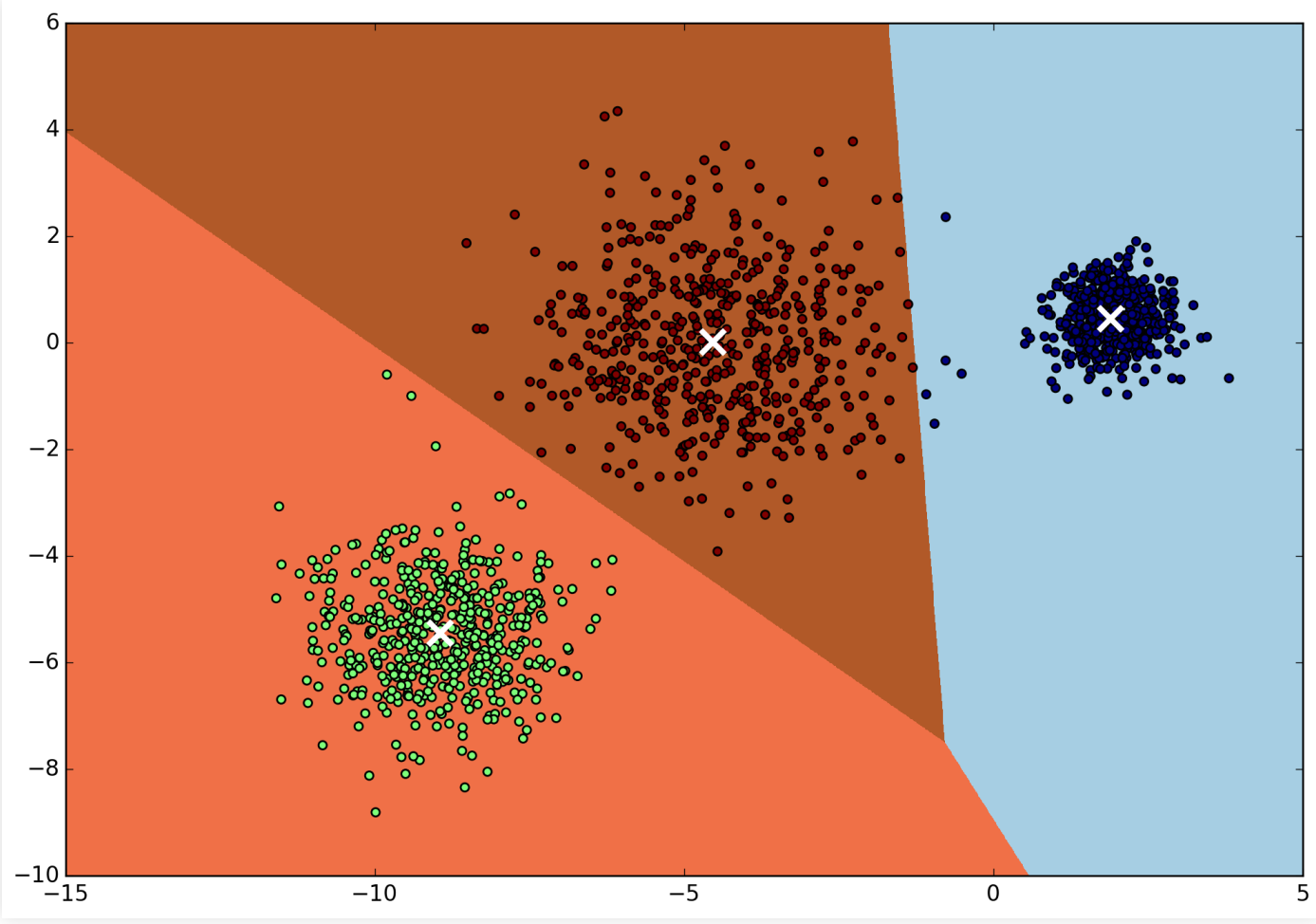
Summary

- Problems with prototype clustering
- Affinity propagation clustering
- Density-based clustering
- Clustering validation

Prototypes

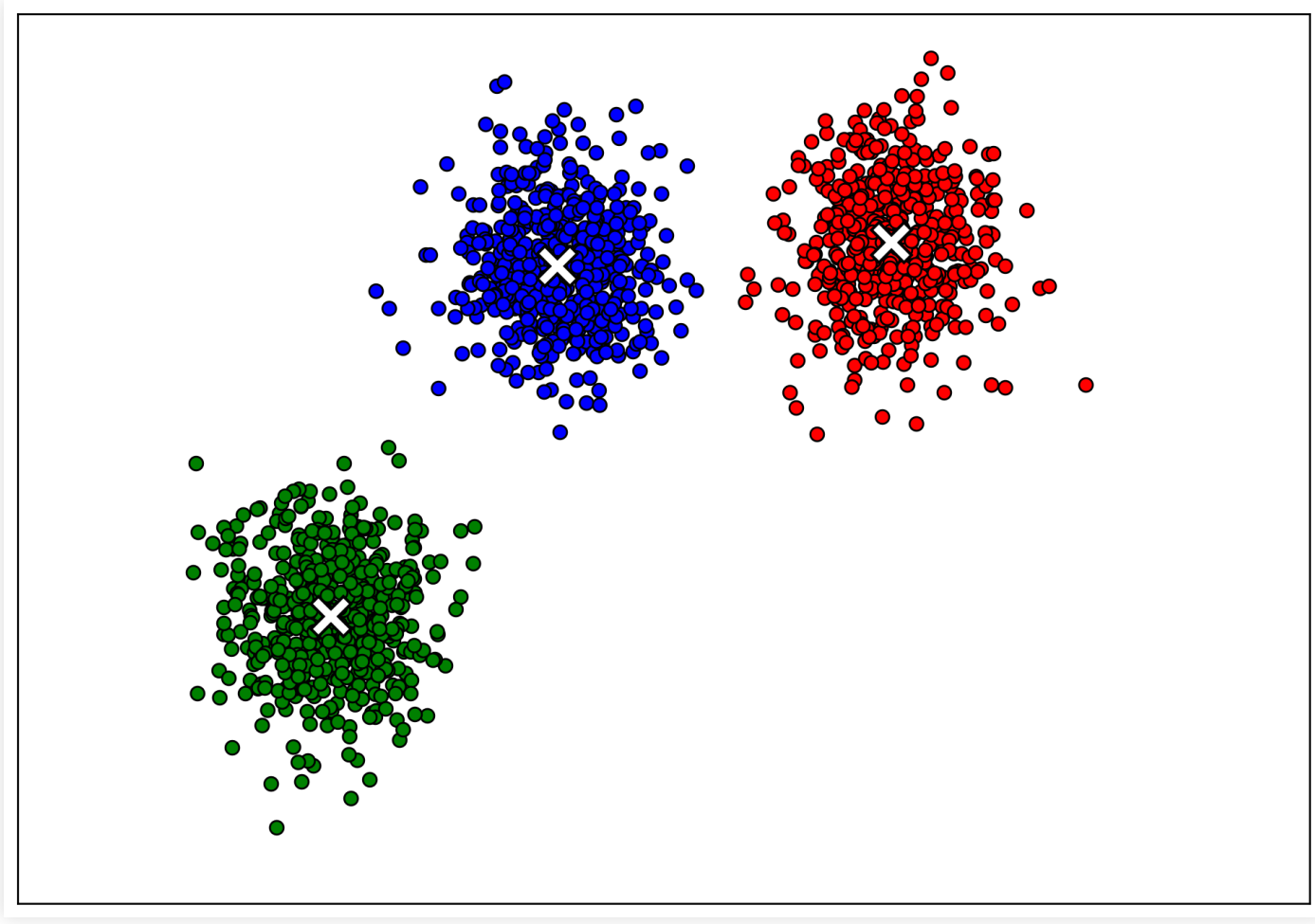
Prototypes

- K-means is an example of prototype-based clustering



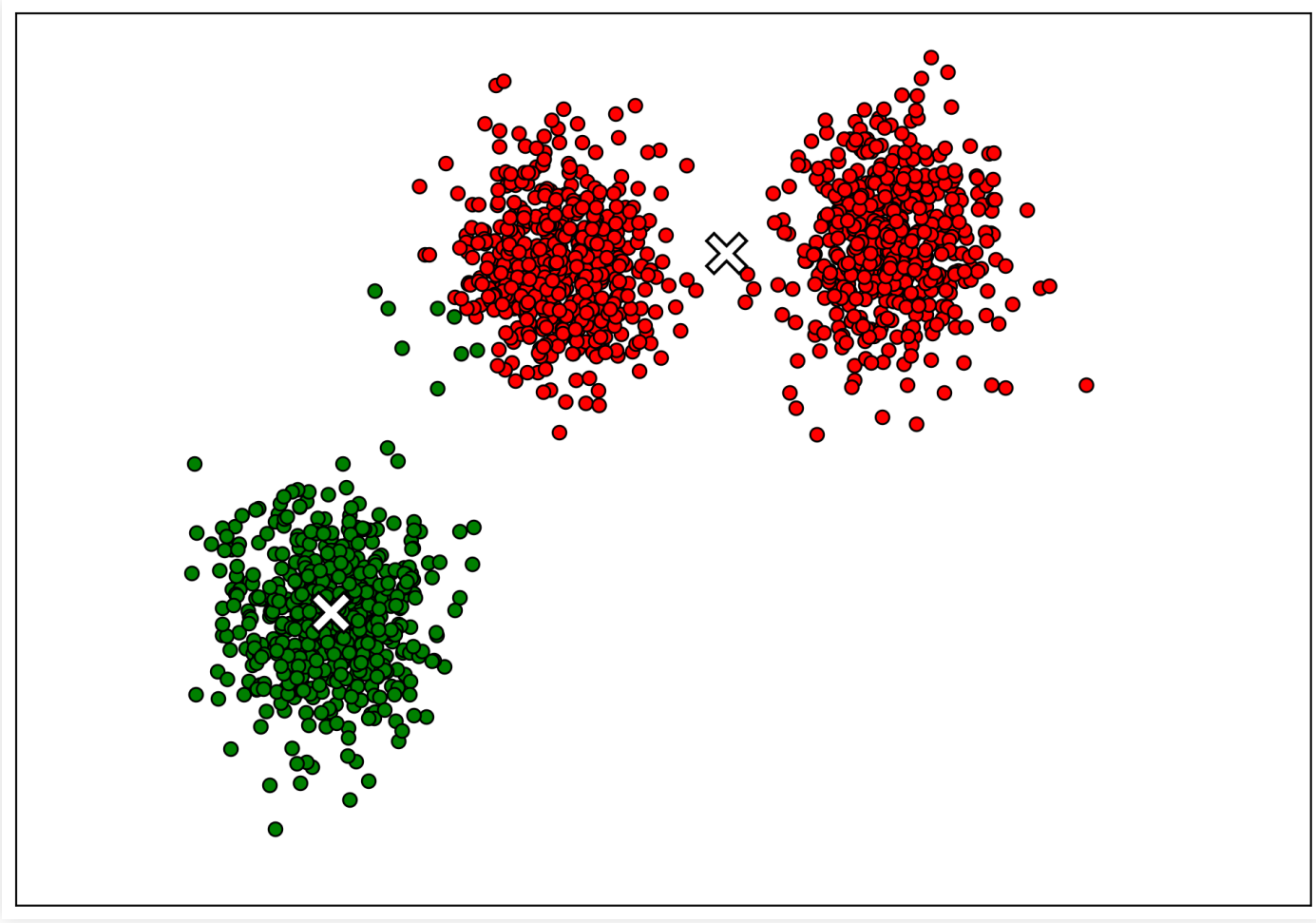
Prototypes (k-means)

- Good for globular clusters with similar dispersion



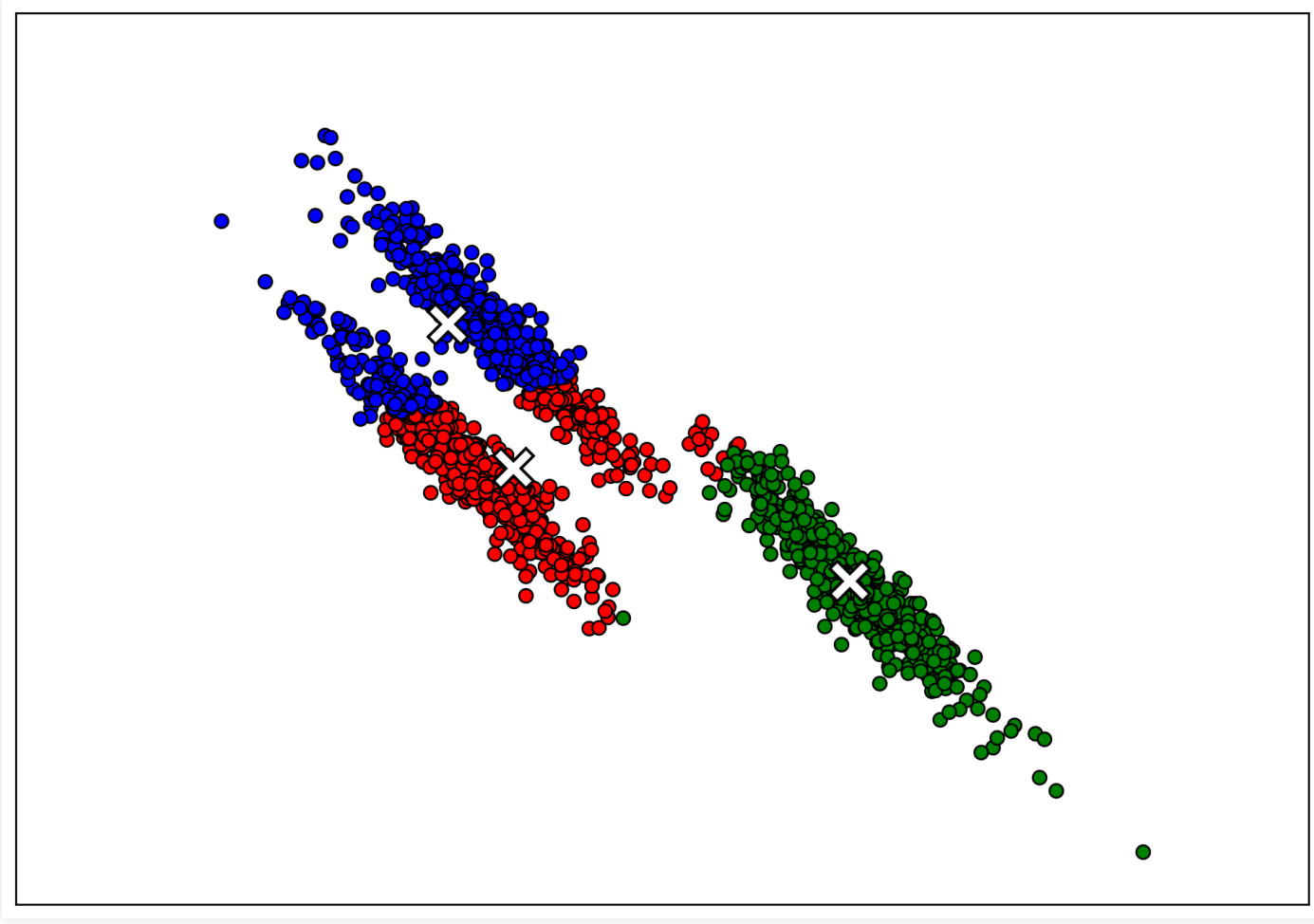
Prototypes (k-means)

- (assuming k is correct)



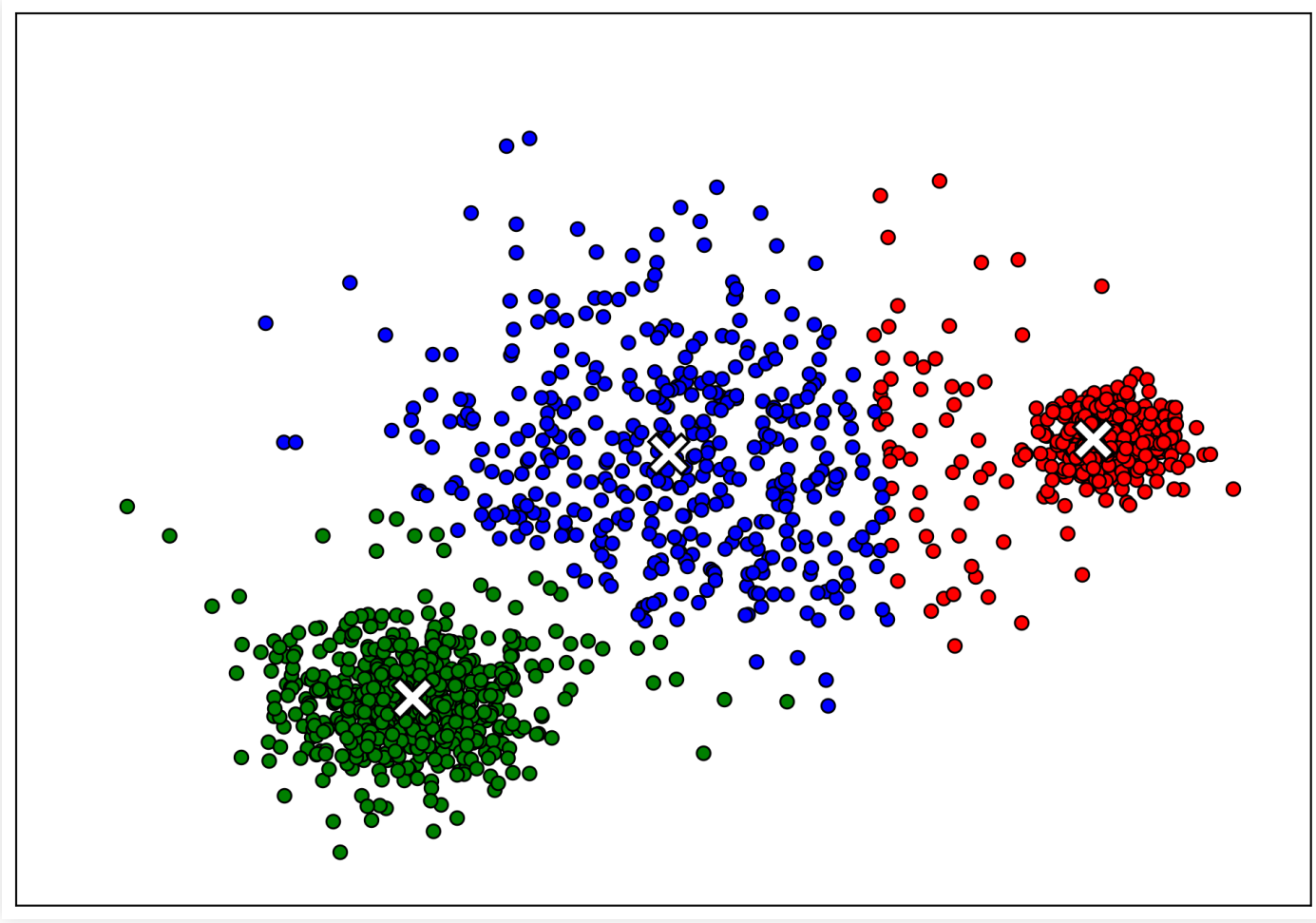
Prototypes (k-means)

- Not good for non-globular clusters



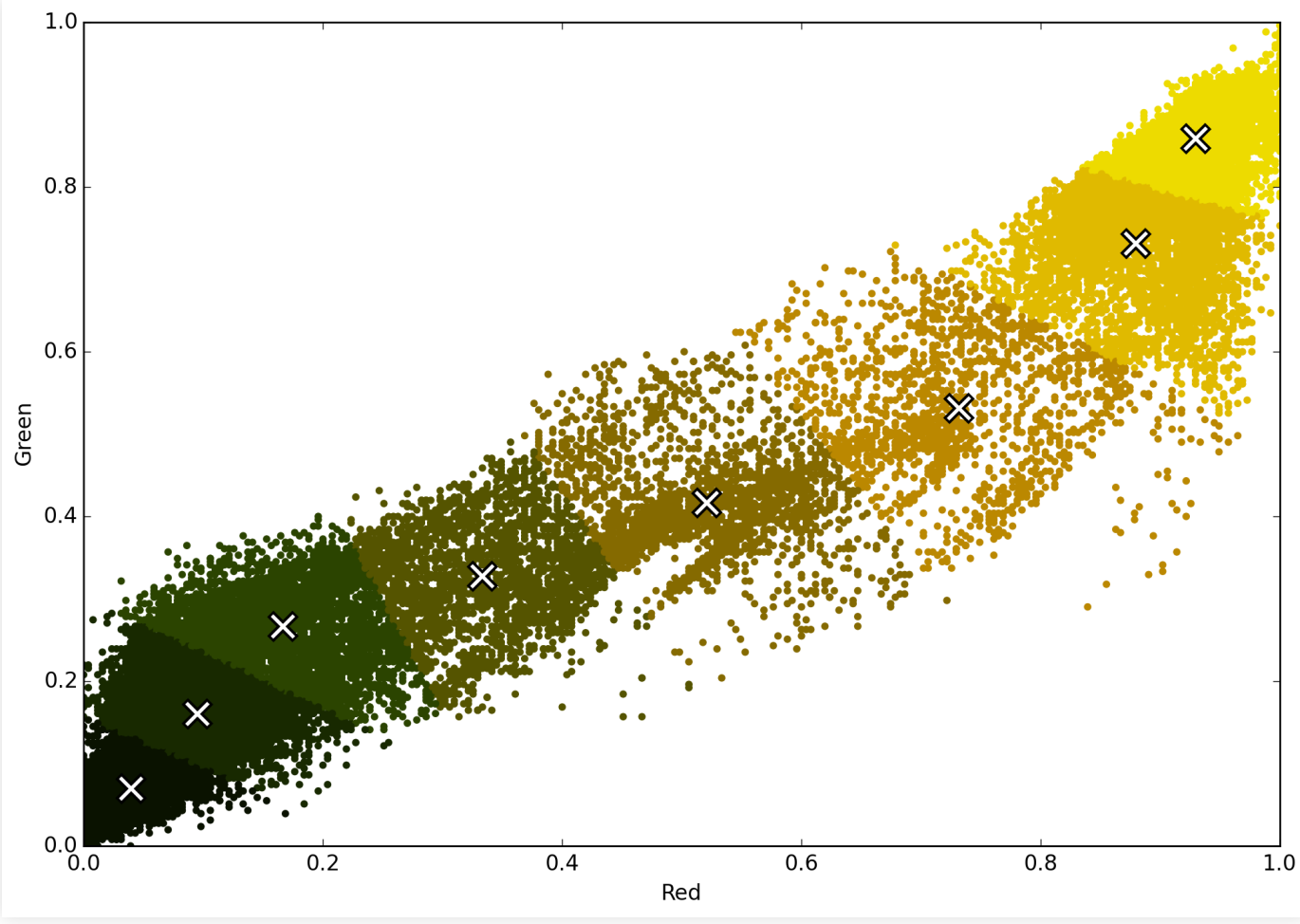
Prototypes (k-means)

- Or differing variances



Prototypes (k-means)

- Note: this may be useful for applications like vector quantization



Prototypes

K-Means (and K-medoids), fixed number of prototypes

- This may be useful in some applications
  - For example, for vector quantization or when we know how many clusters we want
- But we must take this into consideration when deciding where to apply k-means and other prototype-based clustering algorithms

Affinity Propagation

Affinity Propagation

Three matrices:

- Similarity matrix $s_{i,k}$ (example, negative of the distance)
  - $s_{i,i}$ propensity for being a prototype; lower results in fewer clusters
- Responsibility, $r_{i,k}$, "sent" from x_i to x_k
  - Evidence for suitability of x_k as prototype for x_i
- Availability $a_{i,k}$, "sent" from candidate x_k to point x_i
  - How much x_k is available to represent x_i as a prototype

Affinity Propagation algorithm

- Initialise $\mathbf{R}$ and $\mathbf{A}$ to zero
- Update $r_{i,k}$

$$r_{i,k} \leftarrow s_{i,k} - \max_{k' \neq k} \left(a_{i,k'} + s_{i,k'} \right)$$

- First iteration, $r_{i,k}$ is just the similarity relative to others
- Afterwards, discounts $a_{i,k'}$ as other prototypes become available
- Responsibility update: candidate prototypes compete for points

Affinity Propagation algorithm

- Next, we update availability:

$$a_{i,k(i\neq k)} \leftarrow \min\left(0, r_{k,k} + \sum_{i' \notin \{i,k\}} \max(0, r_{i',k})\right)$$

- Self responsibility plus sum of positive responsibilities k receives from other points

$$a_{k,k} \leftarrow \sum_{i' \neq k} \max\left(0, r_{i',k}\right)$$

- Self availability, evidence that k is prototype

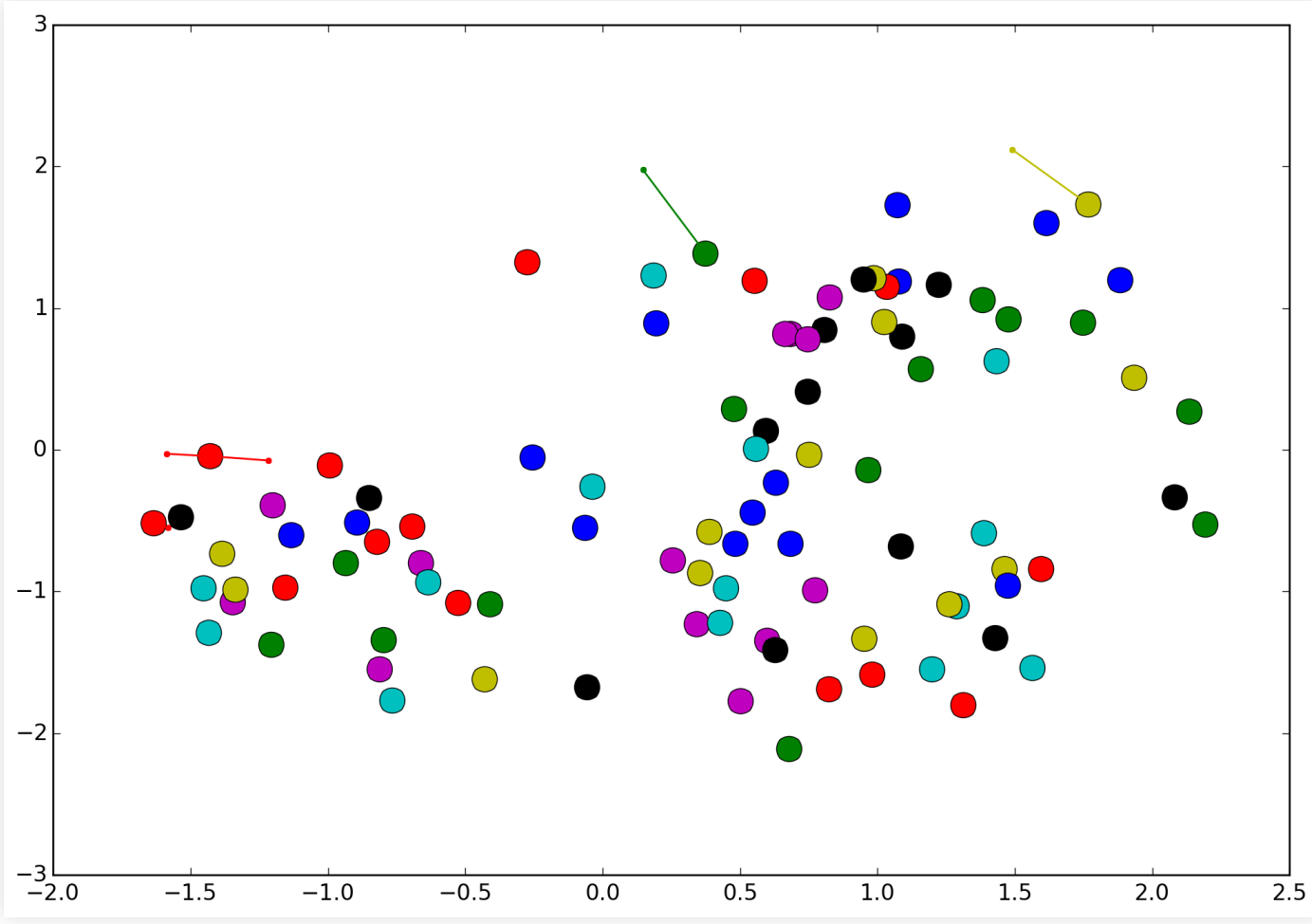
Affinity Propagation algorithm

- Identifying prototypes and clusters:
- For each point x_i, compute:

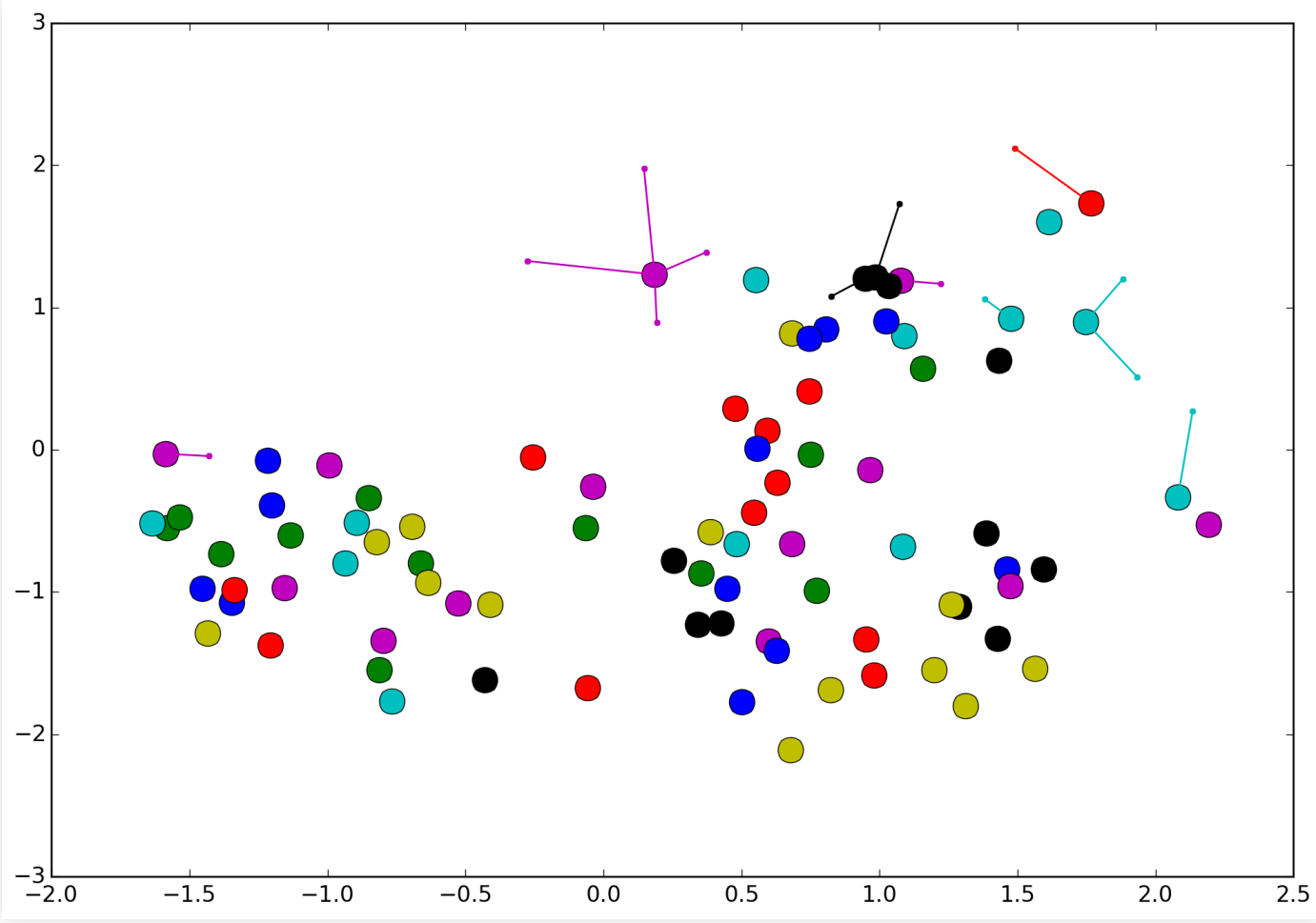
$$k' = \arg\max_k \; a_{i,k} + r_{i,k}$$

- If $k' = i$ then point x_i is a prototype
- If $k' \neq i$ then point x_i belongs to cluster of prototype x'_k

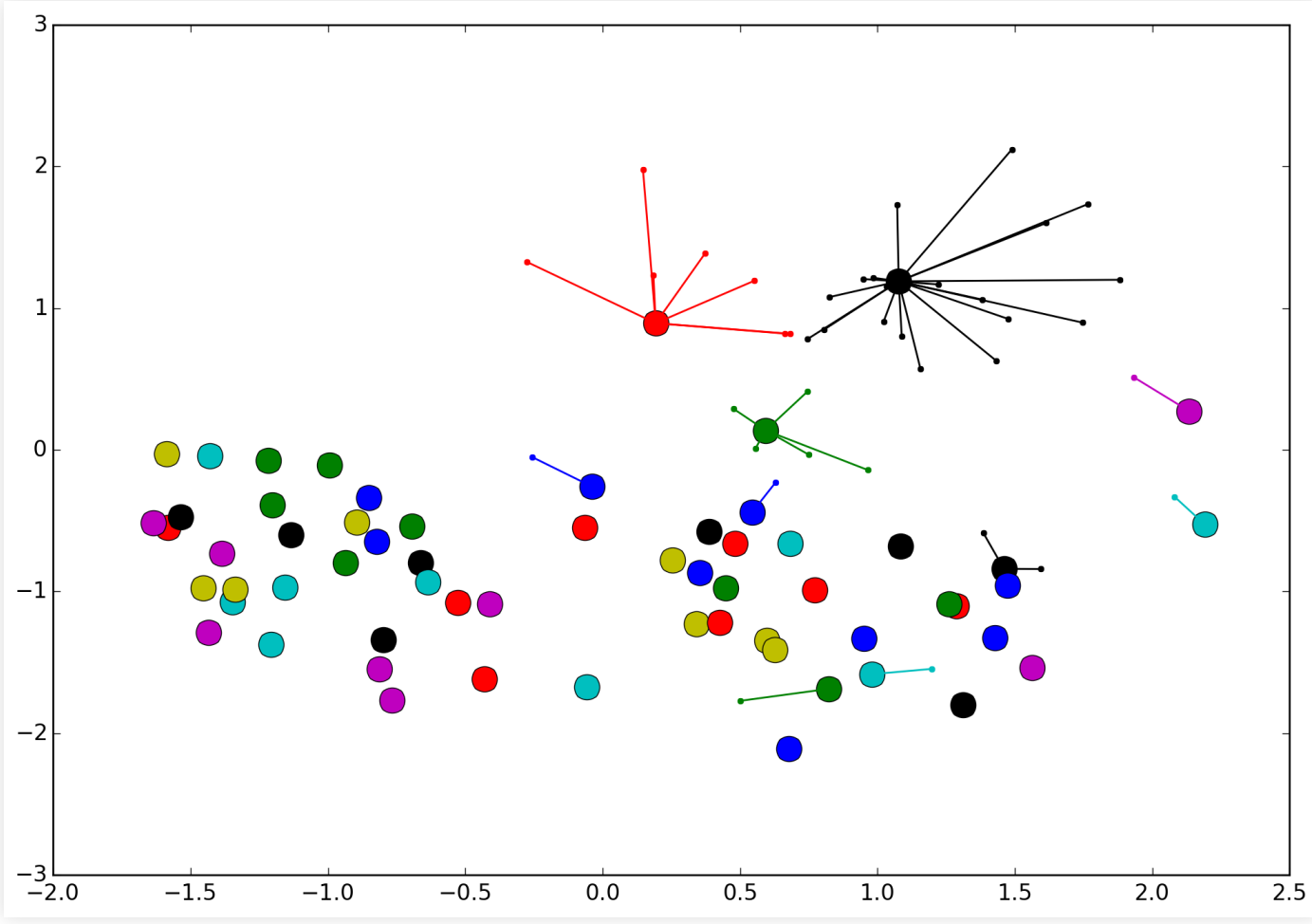
Affinity Propagation



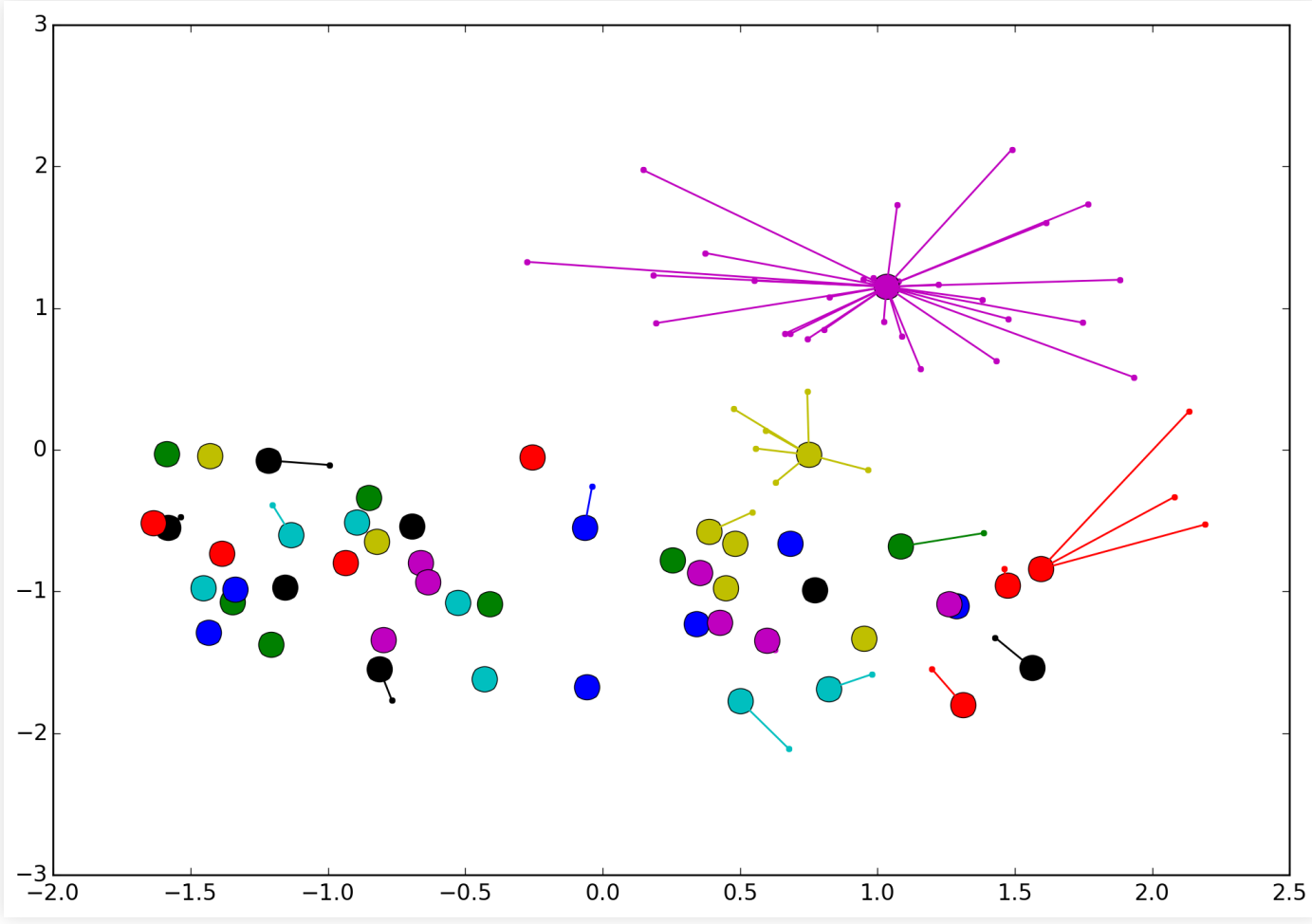
Affinity Propagation



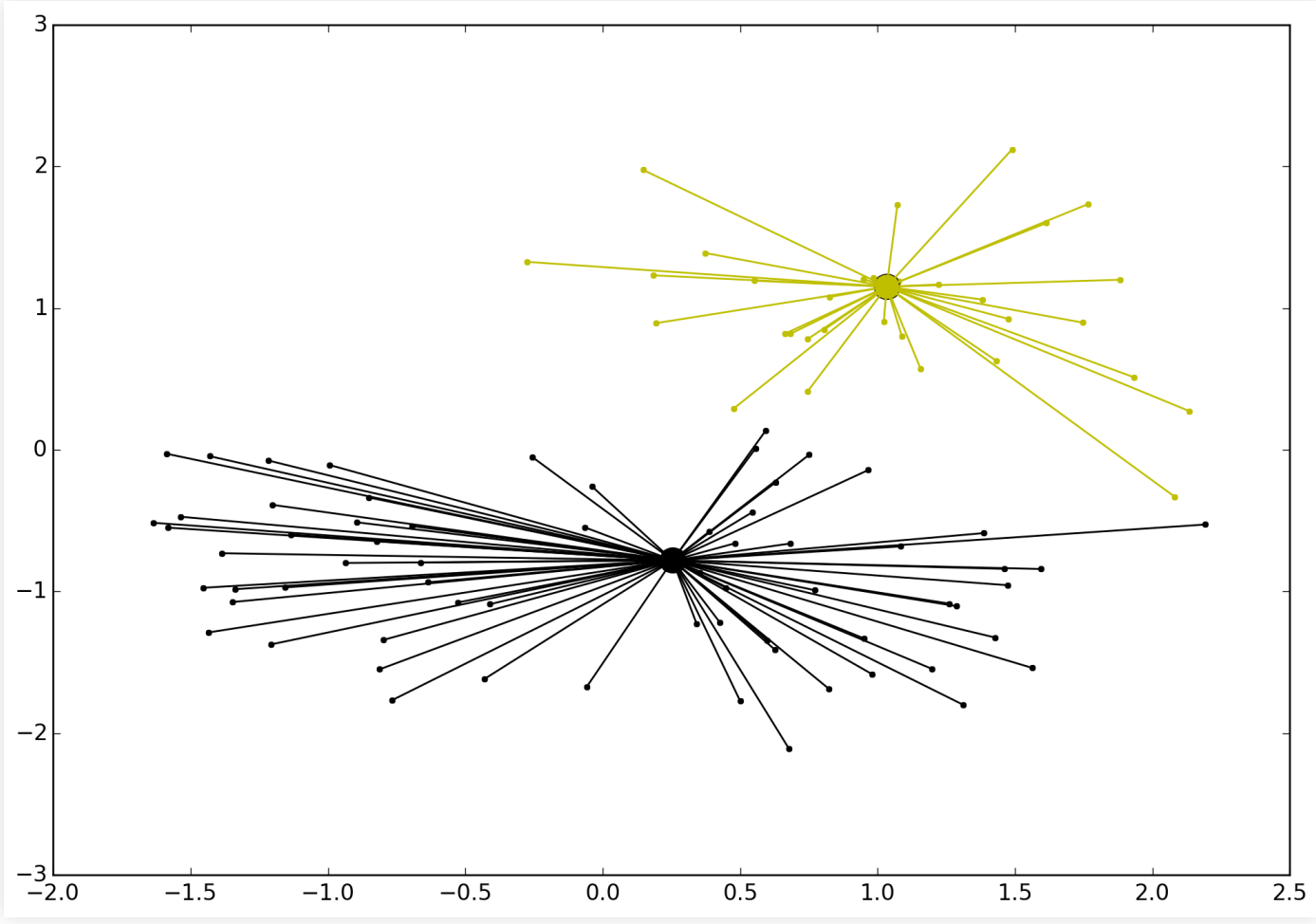
Affinity Propagation



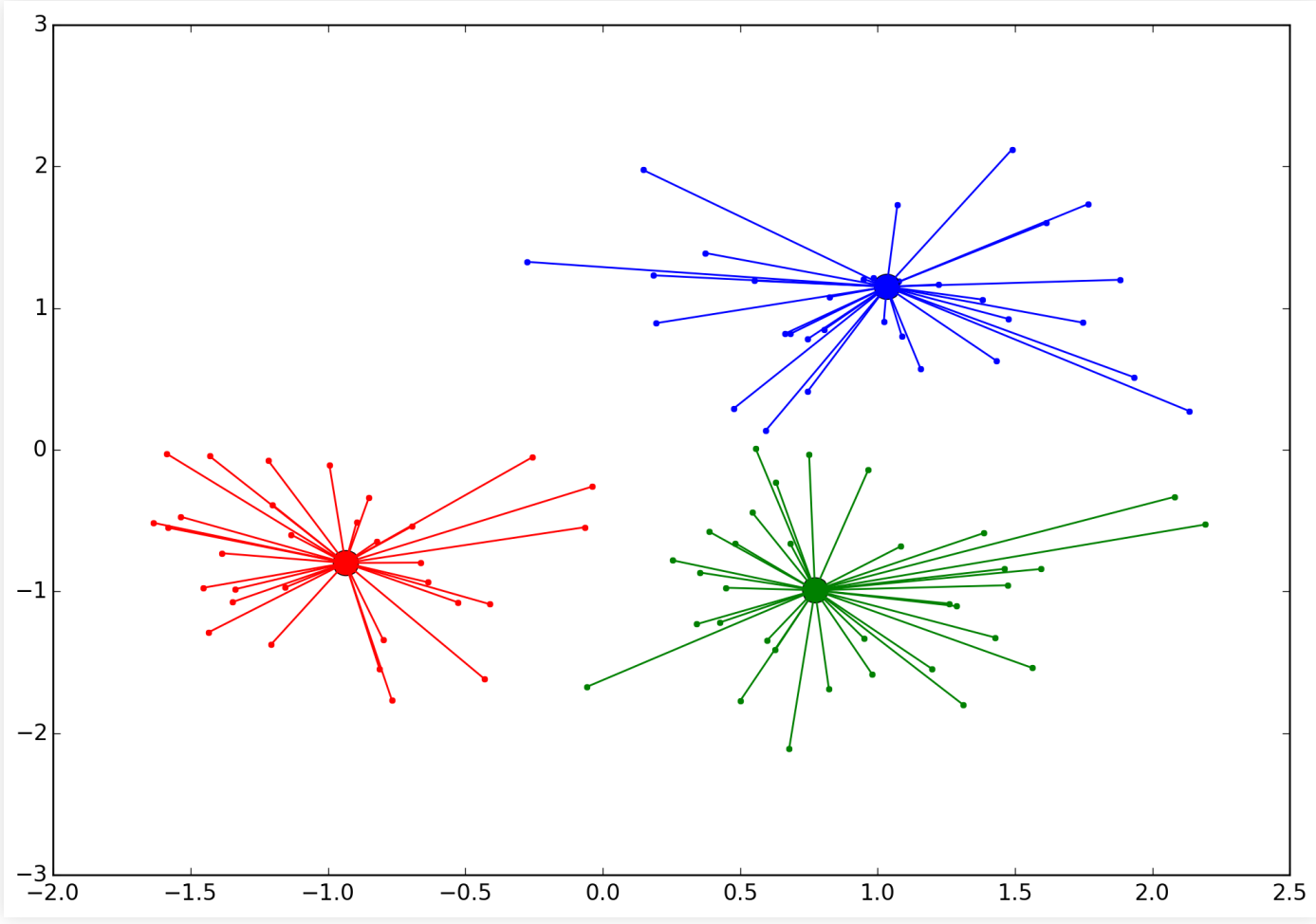
Affinity Propagation



Affinity Propagation

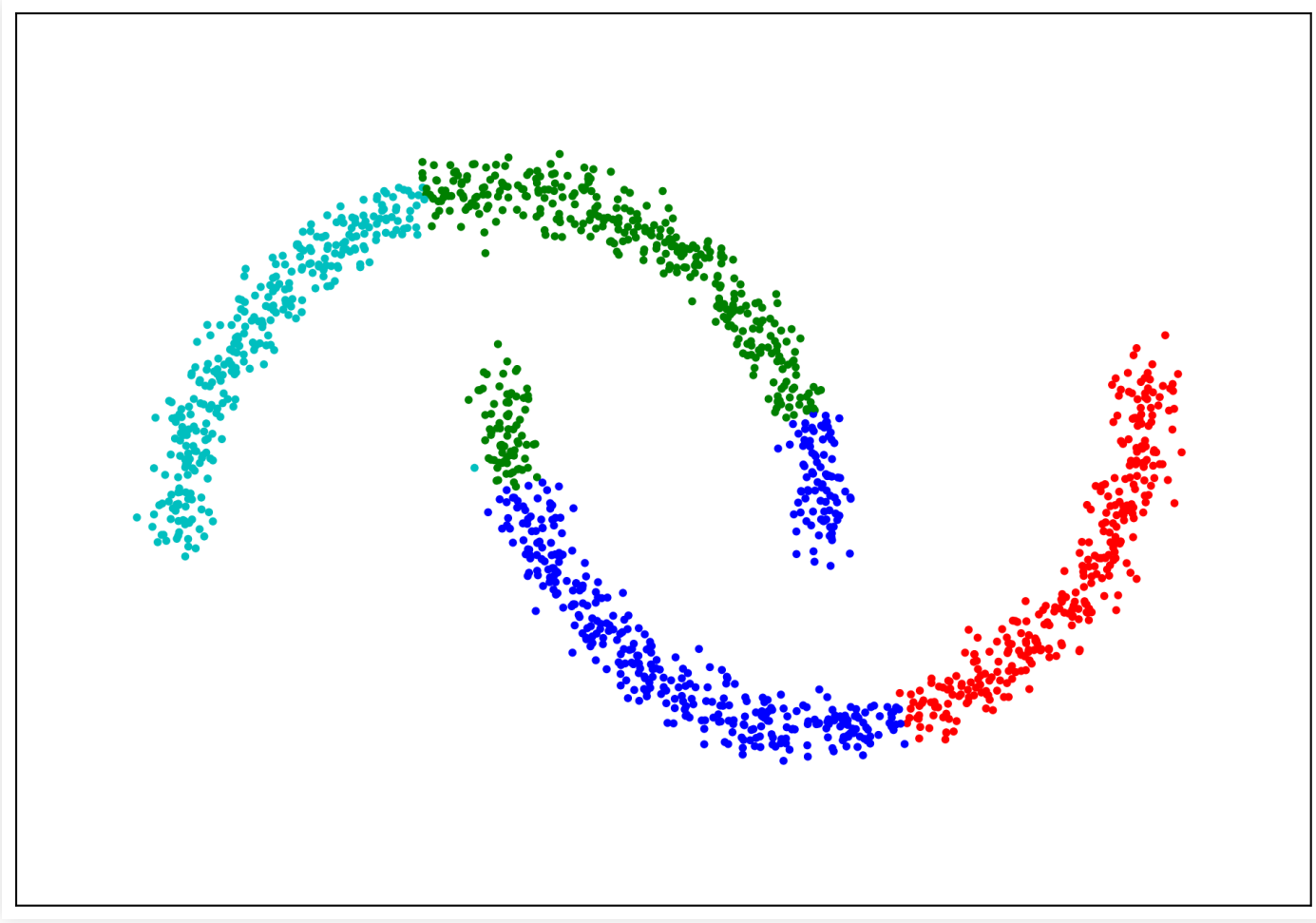


Affinity Propagation



Affinity Propagation

- Automatic k, but still based on prototypes



Affinity Propagation with Scikit-Learn

```
class sklearn.cluster.AffinityPropagation:
   damping=0.5  # to avoid oscilations when updating
   max_iter=200  convergence_iter=15, #stop criteria
   preference=None                    #set s(k,k)
   affinity='euclidean'               #or 'precomputed'

.cluster_centers_indices_
.labels_
.affinity_matrix_
.n_iter_
```

DBSCAN

DBSCAN

- Density-based spatial clustering of applications with noise

- Neighbourhood N_ϵ: all points within distance ϵ
- p is a core point if at least minPts in N_ϵ (those points are directly reachable from core point p)
- q is reachable from p if directly reachable from p or from core point p' reachable from p

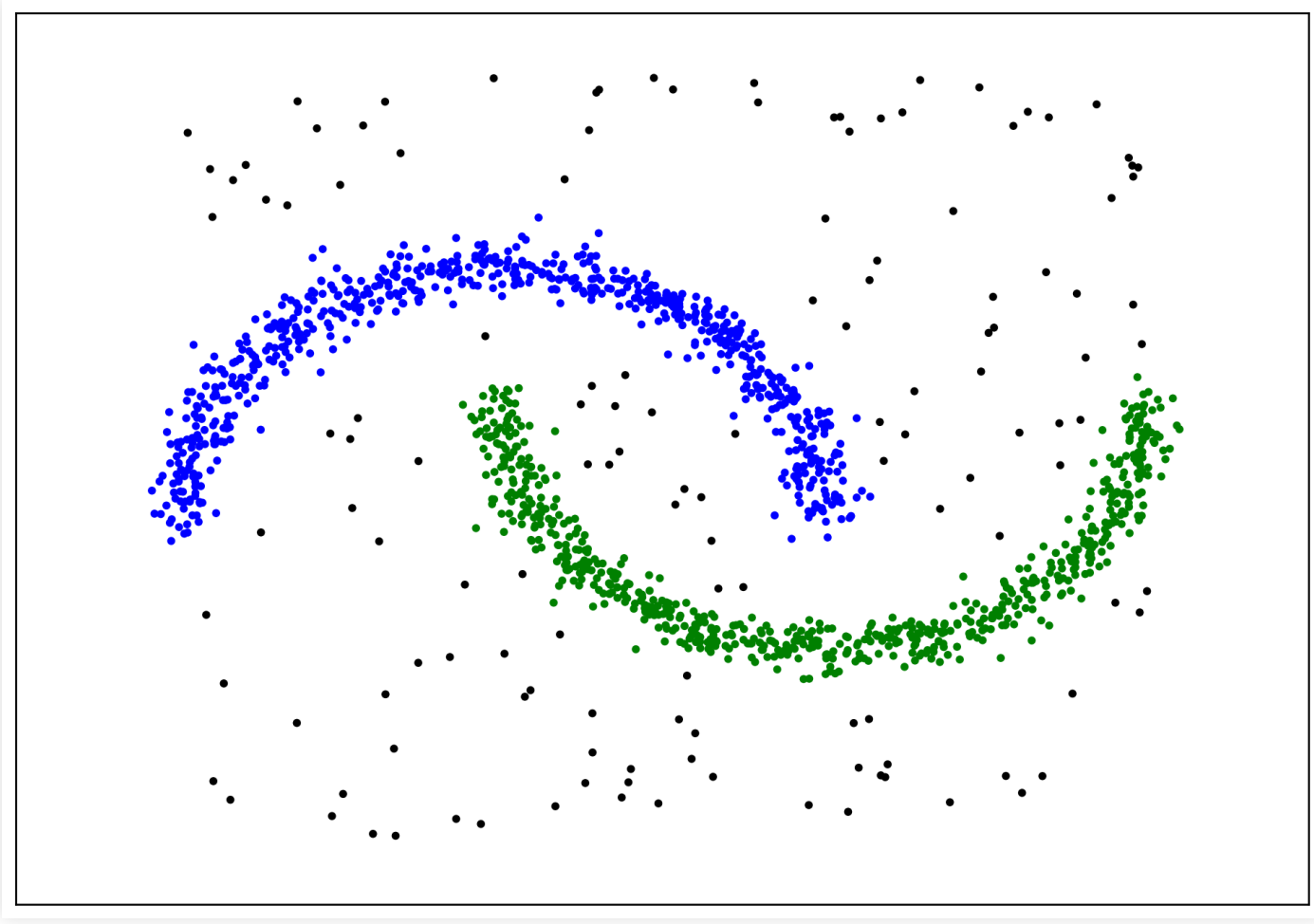
DBSCAN algorithm

- Visit each point p
- If $|N_\epsilon(p)| < \text{minPts}$ then p is (presumed) noise
- Else, create cluster of core point p
- For each $q \in N_\epsilon(p)$, add q to cluster of p

- if $|N_\epsilon(q)| \geq \text{minPts}$ merge clusters of p and q

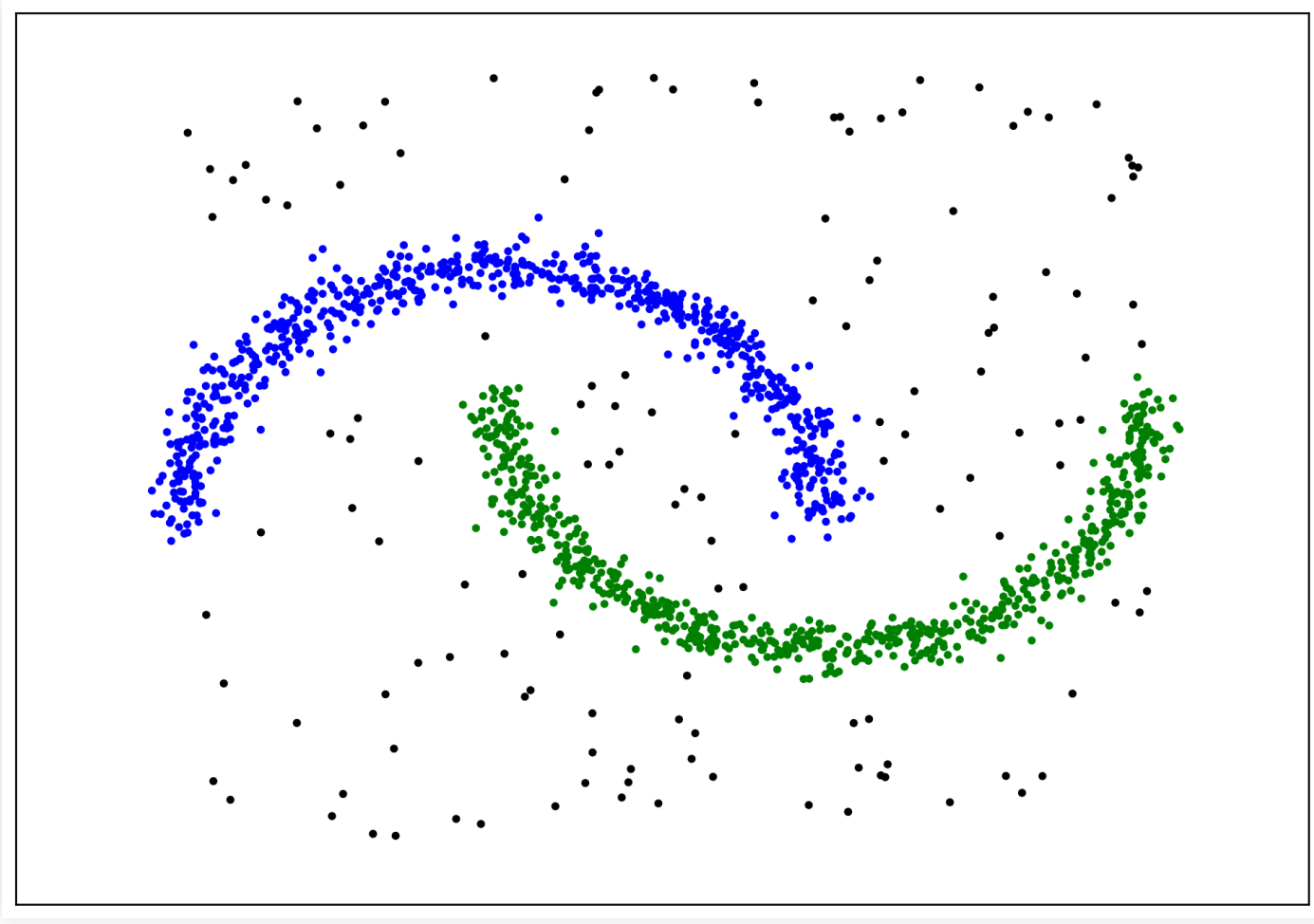
DBSCAN

- Density-based clustering solves cluster shape problems



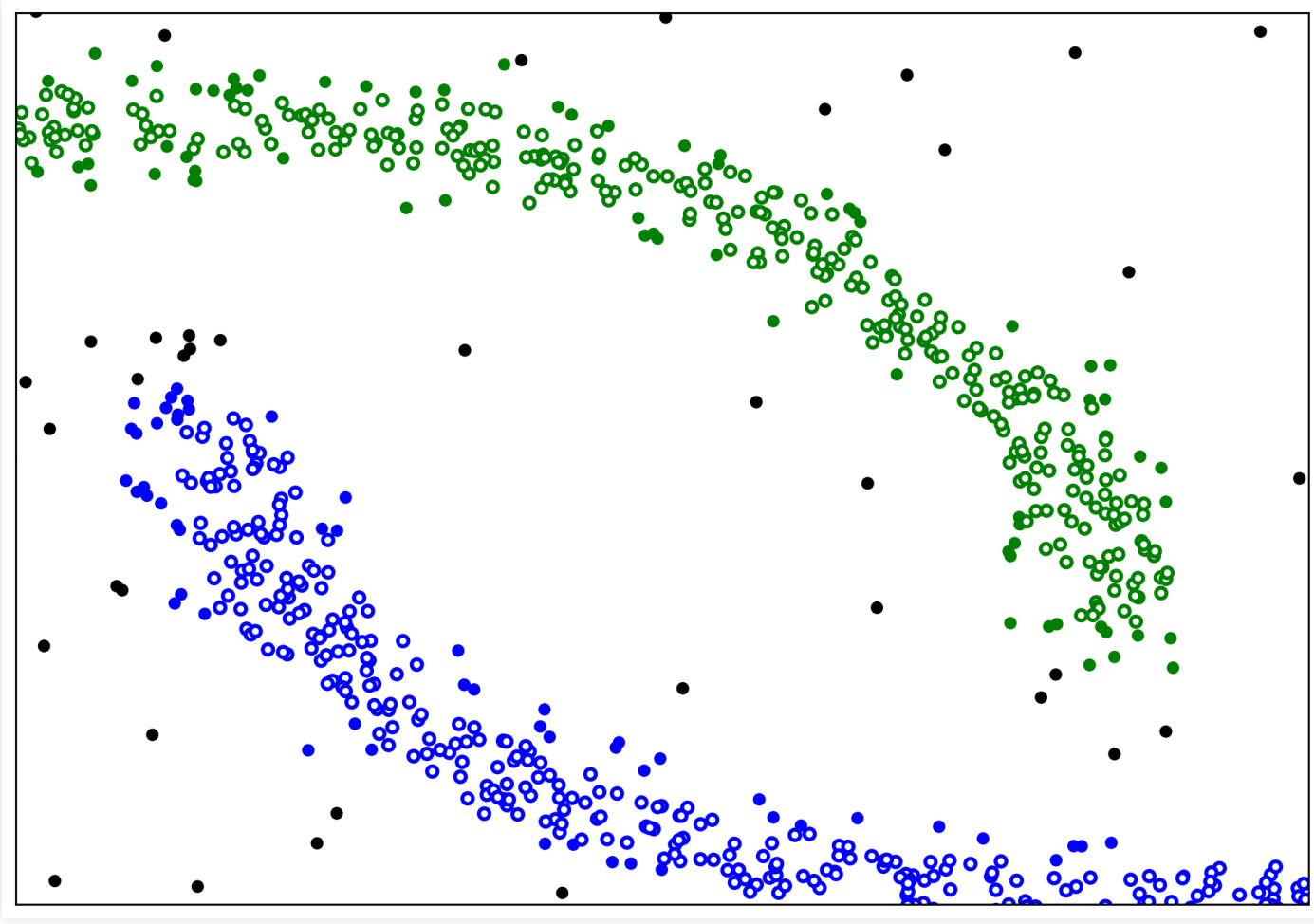
DBSCAN

- And can handle noise



DBSCAN

- Note: core points are not prototypes



DBSCAN with Scikit-Learn

```
class sklearn.cluster.DBSCAN:
   eps=0.5,              #epsilon for neighb.
   min_samples=5,        #minPts
   metric='euclidean'    #can be callable or precomputed

.core_sample_indices_
.labels_              #-1 for noise
```

Clustering validation

Clustering validation

- In supervised learning we can measure error
- Generaly, for clustering we can only assess how "good" clusters are

- What is "good"? Depends on what we want to do.

- Different contexts:

- Determine if the data actually has structure
- Determine the number of clusters
- Evaluate how well computed clusters fit data structure
- (without external information)
- Evaluate how well computed clusters fit known classification
- (with external indeces)
- Compare sets of clusters

Unsupervised clustering validation

- Measure how good clusters are without referring to external data (internal indices):

- Cluster cohesion
- Cluster separation
- Sum of Squares Error
- Silhouette score

Clustering validation

Unsupervised clustering validation

- Cohesion of cluster i:

$$cohesion(C_i) = \sum_{x \in C_i} \sum_{y \in C_i} S(x, y)$$

- Separation between clusters i, j:

$$separation(C_i, C_j) = \sum_{x \in C_i} \sum_{y \in C_j} S(x, y)$$

For prototype-based clusters:

- Cohesion of cluster i:

$$cohesion(C_i) = \sum_{x \in C_i} S(x, \mathbf{c}_i)$$

- Separation between clusters i, j:

$$separation(C_i, C_j) = S(\mathbf{c}_i, \mathbf{c}_j)$$

Clustering validation (unsupervised)

- Sum of squares error, for prototype-based clusterings:

$$SSE = \sum_{k} \sum_{x \in C_k} dist(x, \mathbf{c}_k)$$

- Silhouette Score:

- $a(i)$: average distance of i to all points in same cluster
- $b(i)$: average distance of i to all points in nearest cluster (lowest average distance)

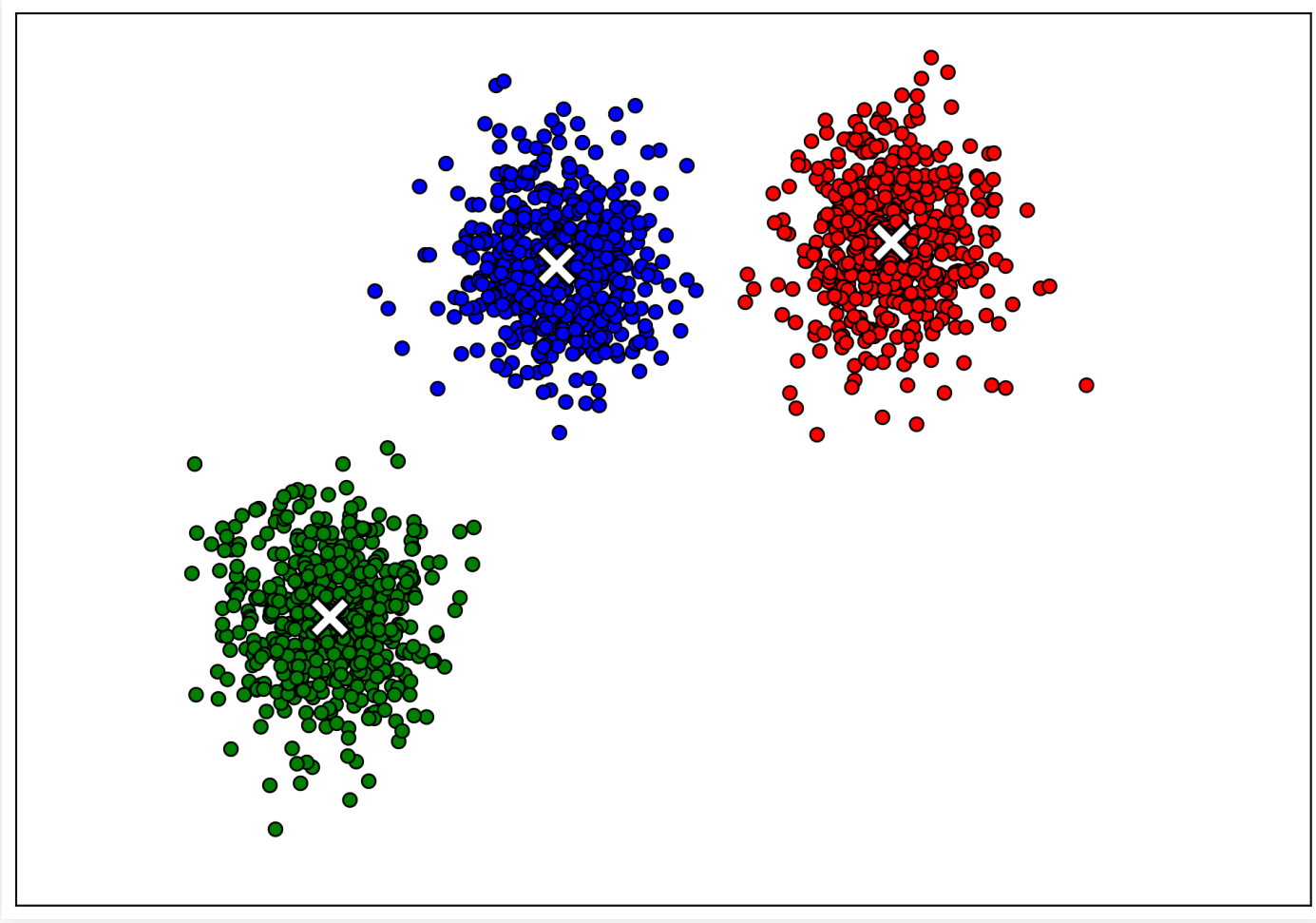
$$s(i) = \frac{b(i) - a(i)}{\max\left(a\left(i\right), b\left(i\right)\right)}$$

- $-1 \leq s(i) \leq 1$, with $s(i)$ greater if $a(i) \ll b(i)$

```
from sklearn.metrics import silhouette_score
silhouette_score(data, labels)
```

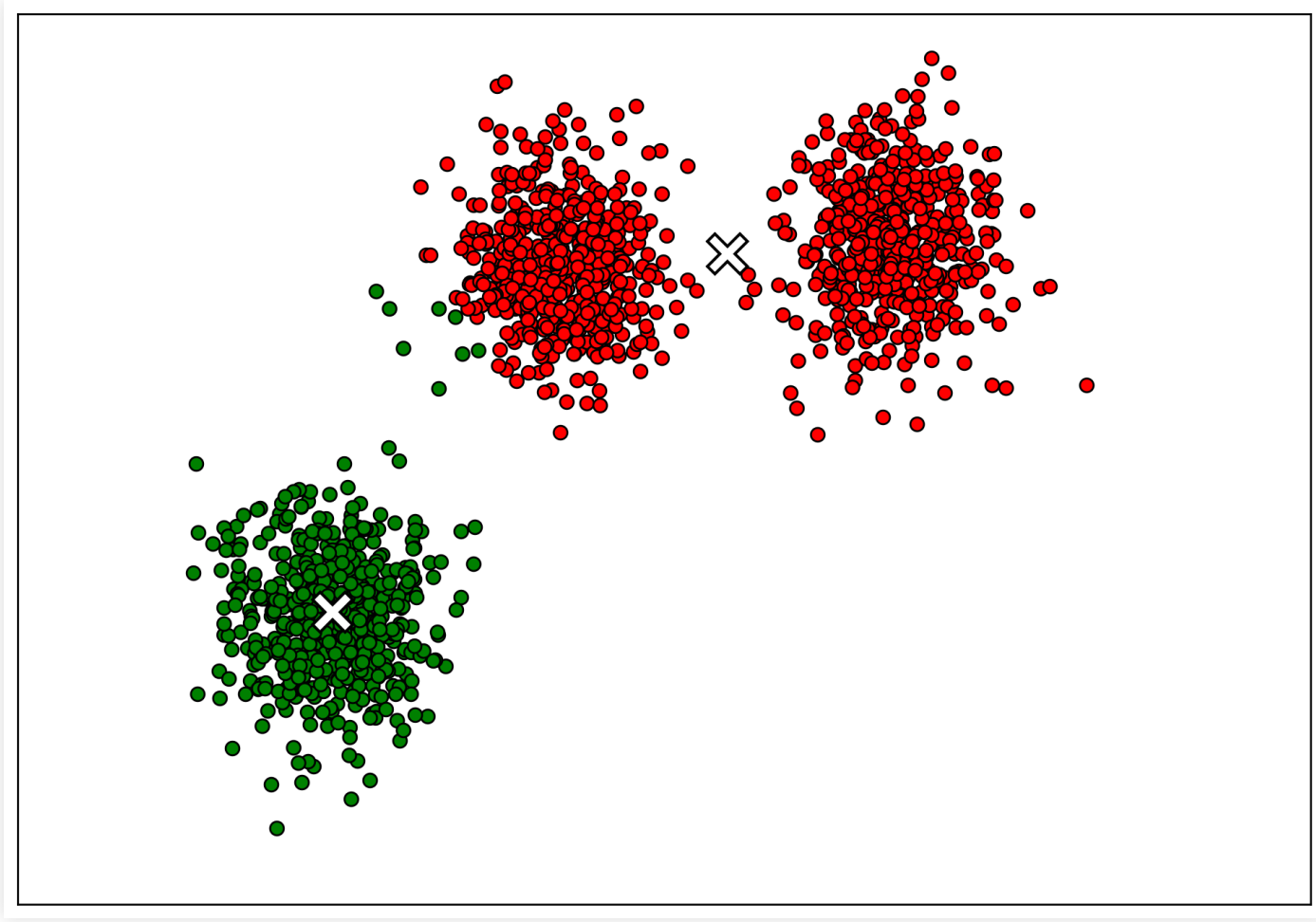
Clustering validation

- Silhouette Score: 0.733



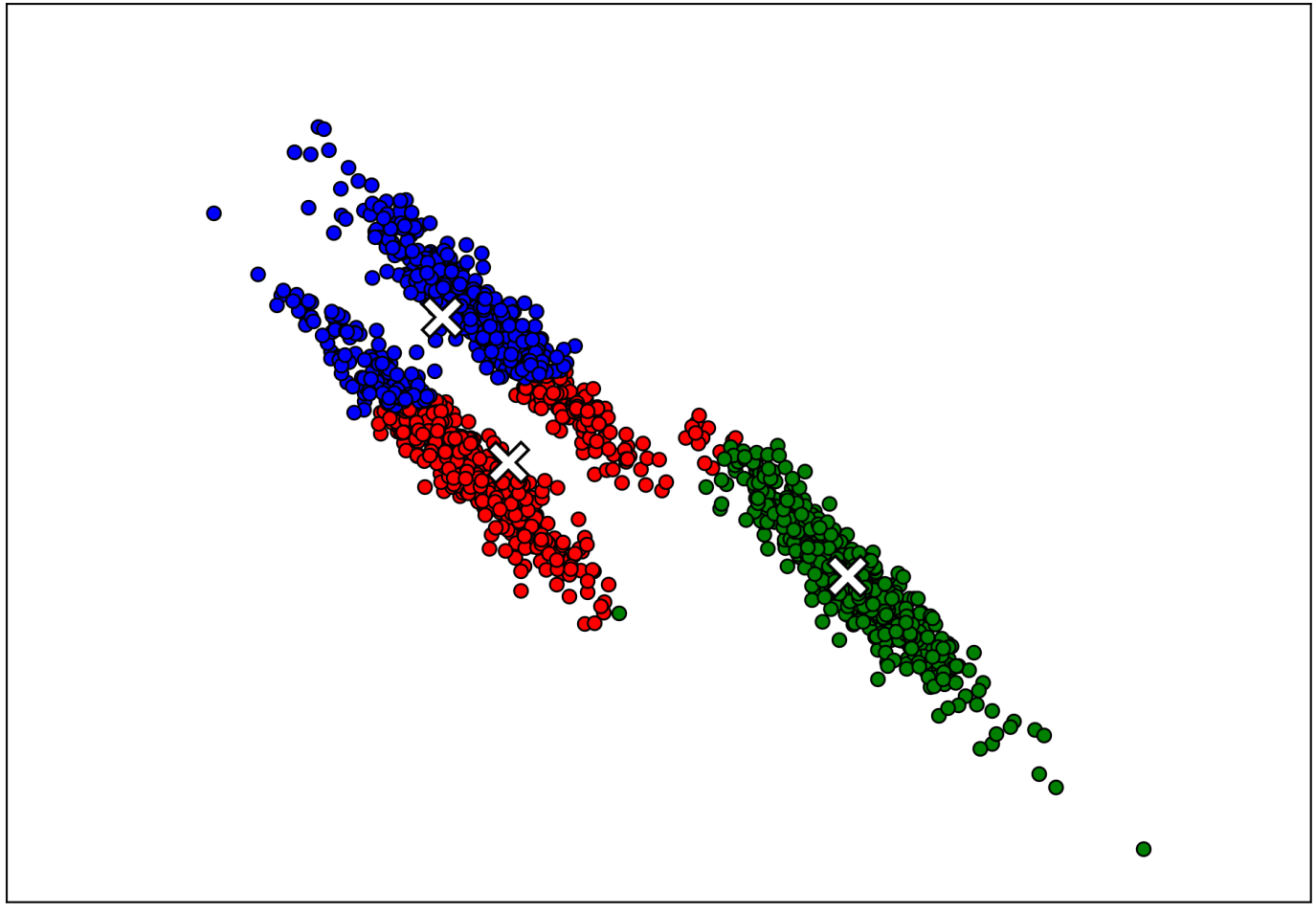
Clustering validation

- Silhouette Score: 0.628



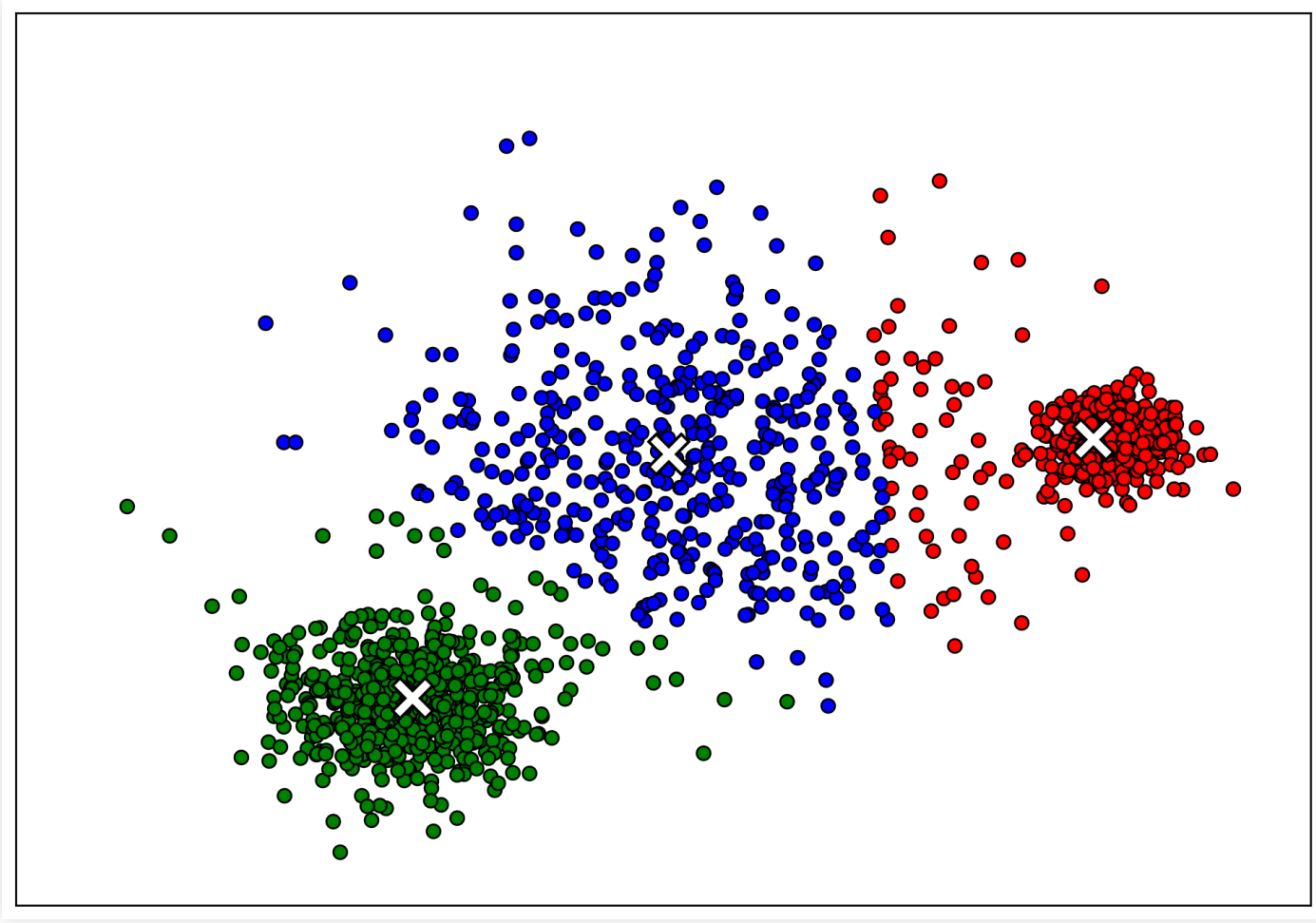
Clustering validation

- Silhouette Score: 0.439



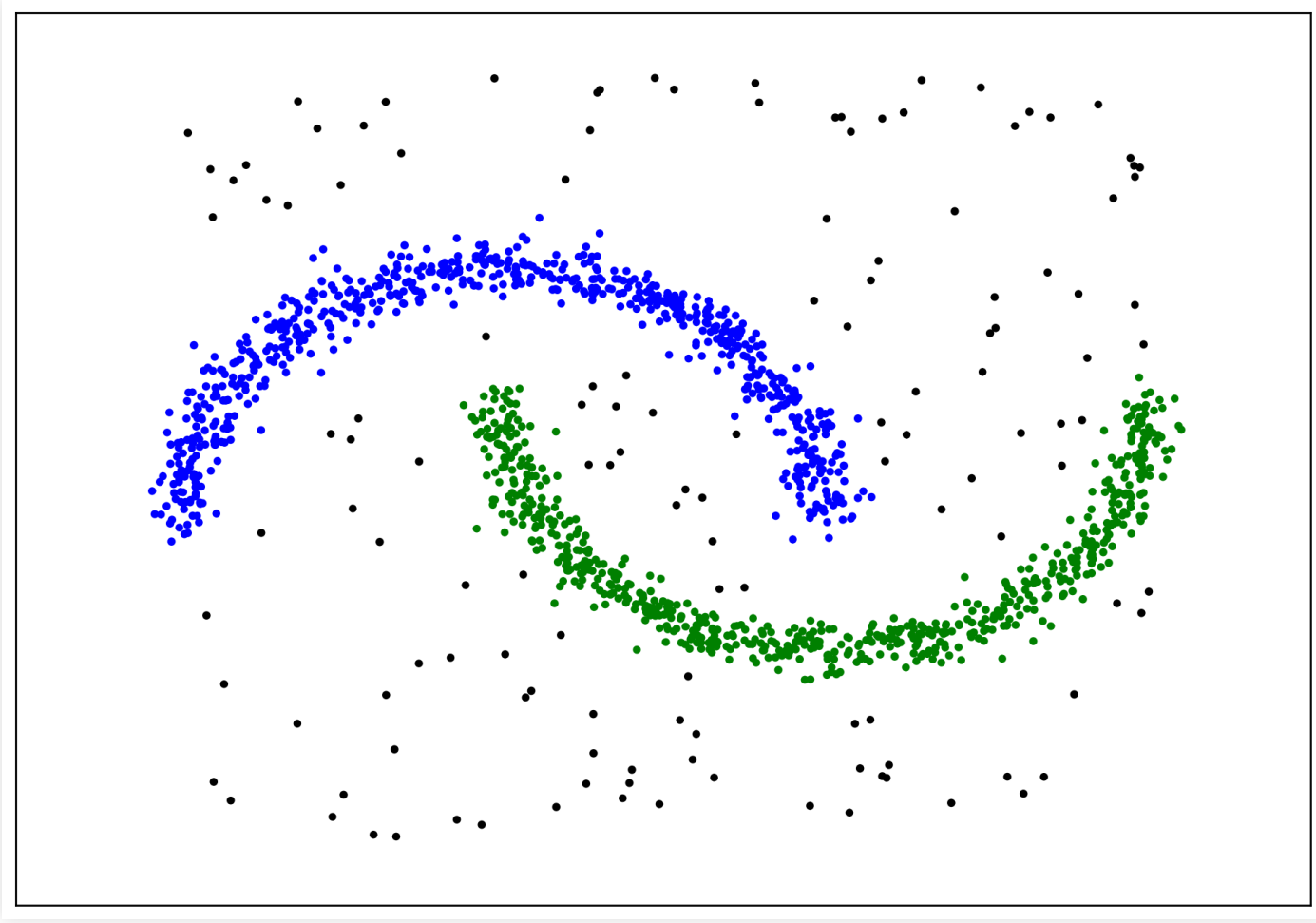
Clustering validation

- Silhouette Score: 0.639



Clustering validation

- Silhouette score: 0.336 (better for compact clusters)



Clustering validation

- Internal index:
  - unsupervised, internal indicators of cohesion, separation, etc
- External index:
  - supervised, compare clusters to some known structure
- Relative index:
  - Compare different clusterings

“The validation of clustering structures is the most difficult and frustrating part of cluster analysis. Without a strong effort in this direction, cluster analysis will remain a black art accessible only to those true believers who have experience and great courage.”

- Algorithms for Clustering Data, Jain and Dubes (1988)

Summary

Clustering, prototypes and beyond

Summary

- Prototype-based: k-means, AP
- Density-based: DBSCAN
- Cluster validation, unsupervised
- (Cluster validation, supervised)

Further reading

- Affinity Propagation: Frey et al. "Clustering by passing messages between data points." Science 2007
- DBSCAN: Ester, Martin, et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." KDD'96, 1996.
- Scikit-learn documentation on clustering