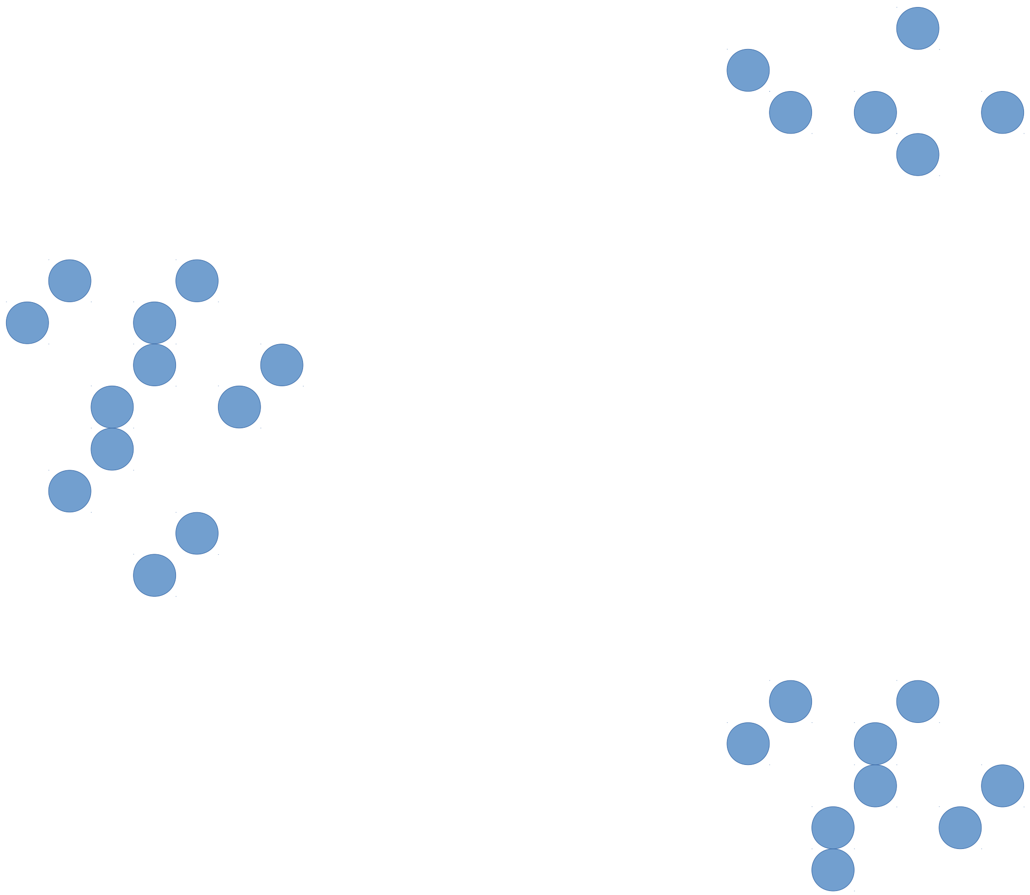
W4995 Applied Machine Learning

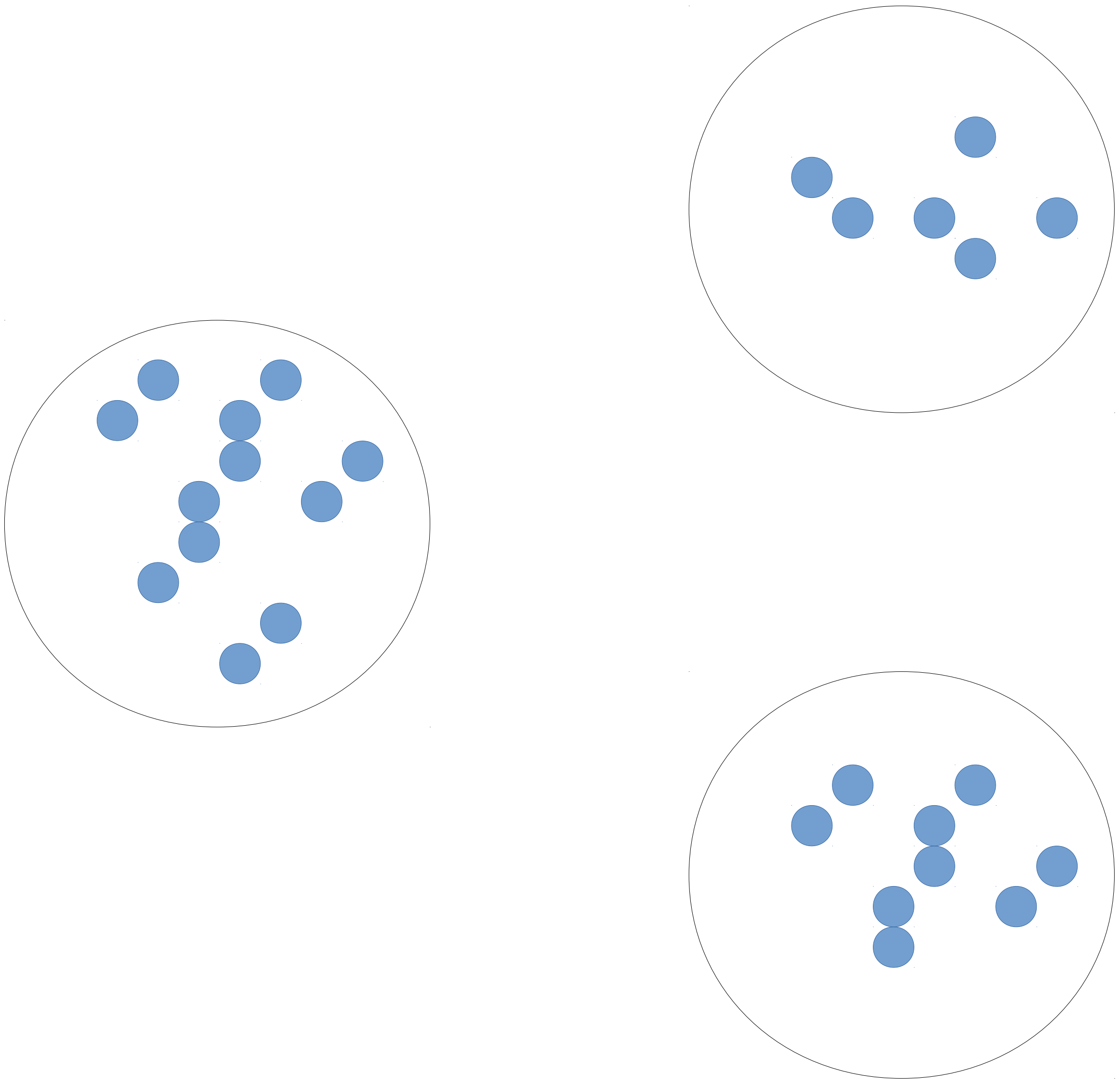
Clustering and Mixture Models

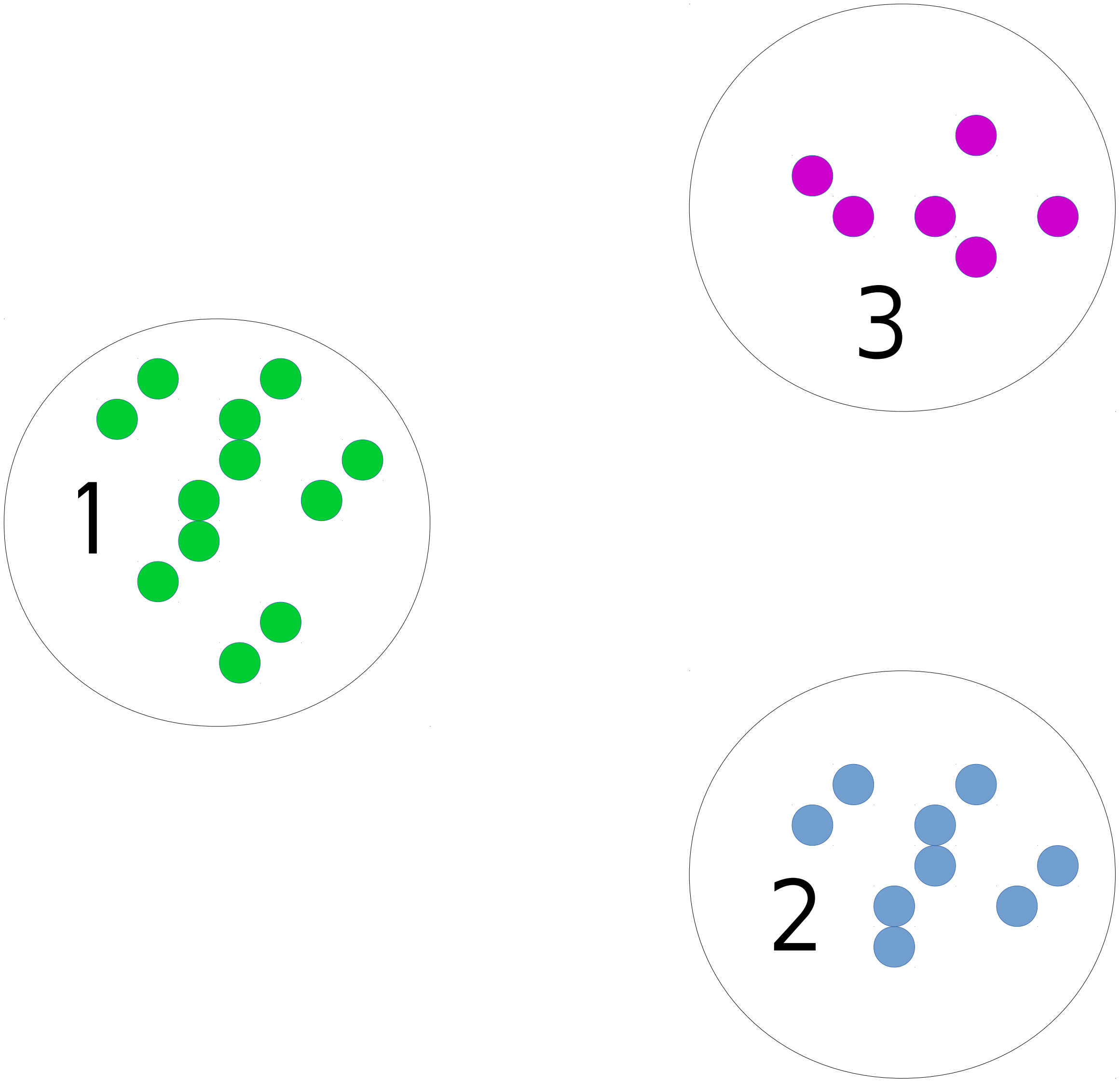
03/22/17

Andreas Müller

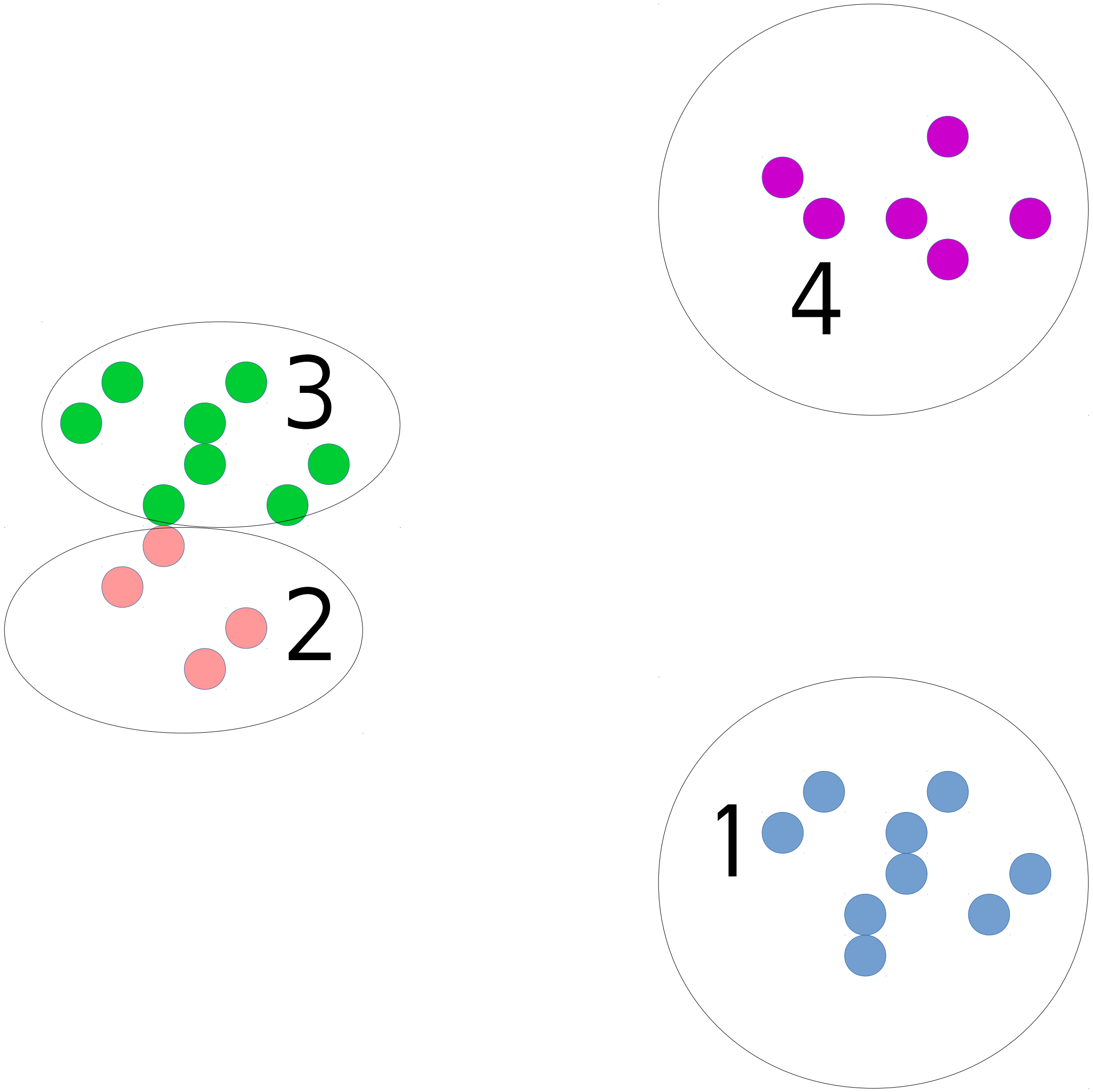
Clustering






3
1
2

Group into 4 clusters



Clustering

- Group data into a partitioning:
- Identify groups by assigning “cluster labels” to each data point.
- Points within a cluster should be “similar”.
- Points in different cluster should be “different”.
- Number of clusters might be pre-specified.
- Notations of “similar” and “different” depend on algorithm or user specified metrics.

Goals of Clustering

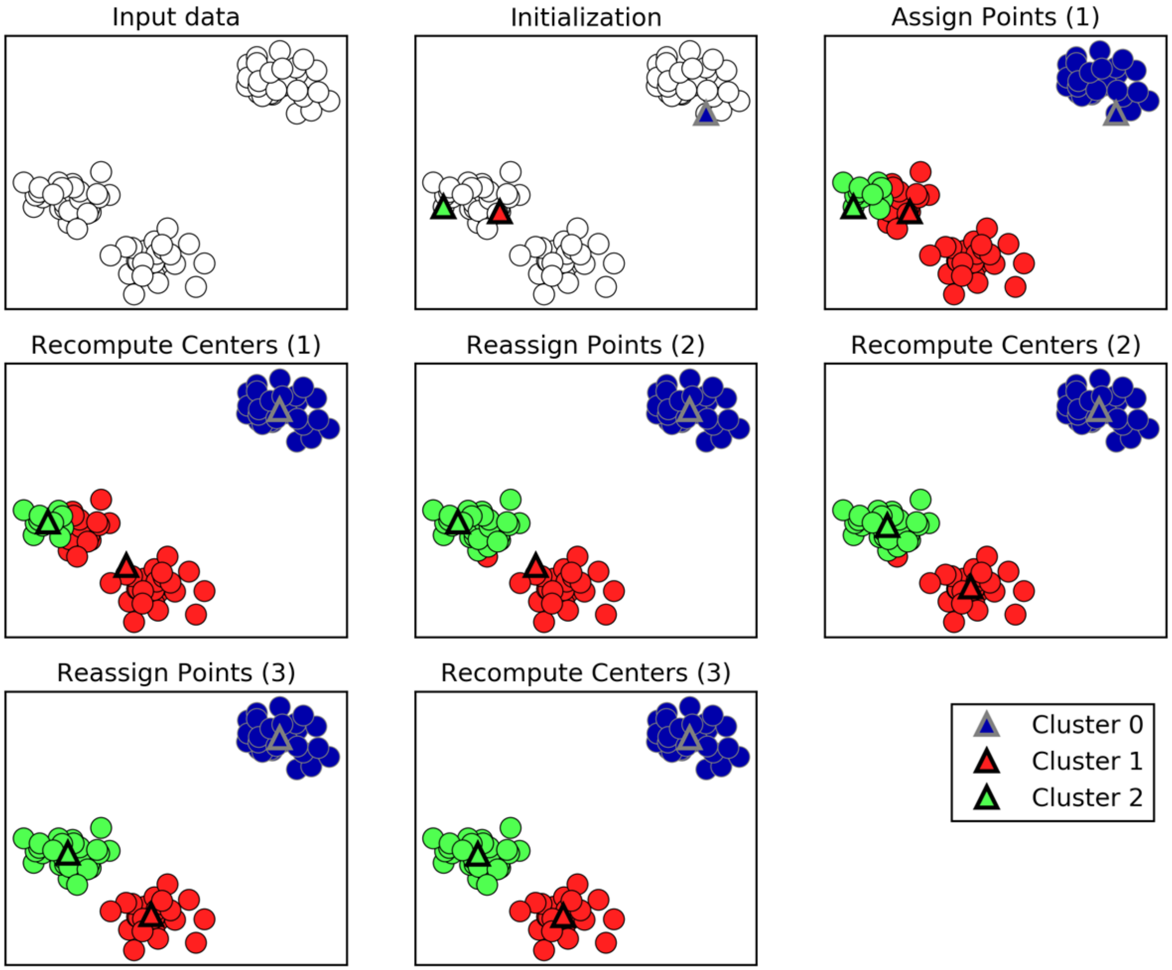
- Data exploration
  - are there coherent groups?
  - How many groups are there?
- Data partitioning
  - divide data by group before further processing.
- Unsupervised feature extraction
  - Derive features from clusters or cluster distances

Clustering Algorithms

K-Means

K-Means algorithm

- Pick number of clusters k.
- Pick k random points as “cluster center”
- While cluster centers change:
  - Assign each data point to it’s closest cluster center
  - Recompute cluster centers as the mean of the assigned points.



Objective function for K-Means

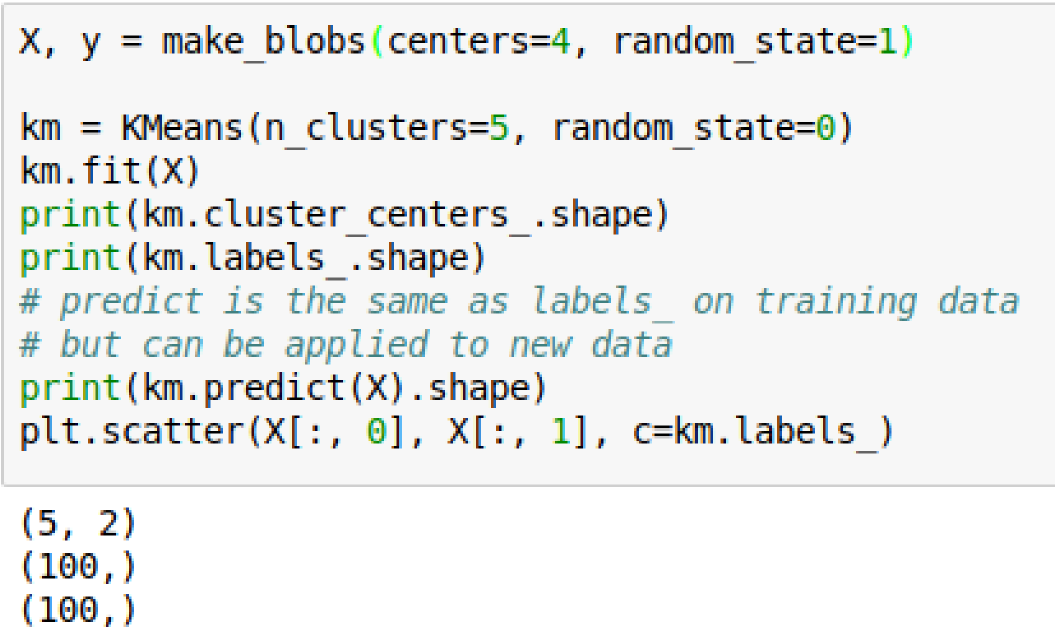
- Finds a local minimum of minimizing squared distances (exact solution is NP hard):

$$\min_{\mathbf{c}_j \in \mathbf{R}^p, j=1,..,k} \sum_i ||\mathbf{x}_i - \mathbf{c}_{x_i}||^2$$

$\mathbf{c}_{x_i}$ Is the cluster center c_j closest to x_i.

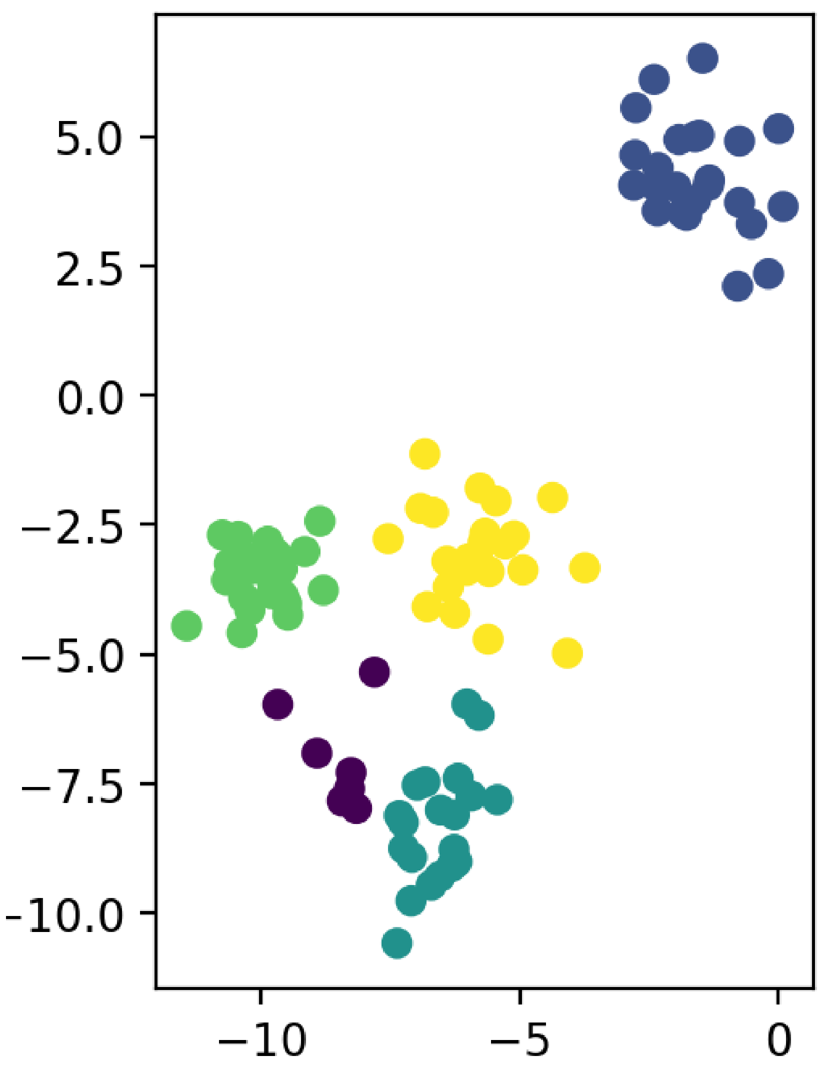
- New data points can be assigned cluster-membership based on existing clusters.

K-Means API



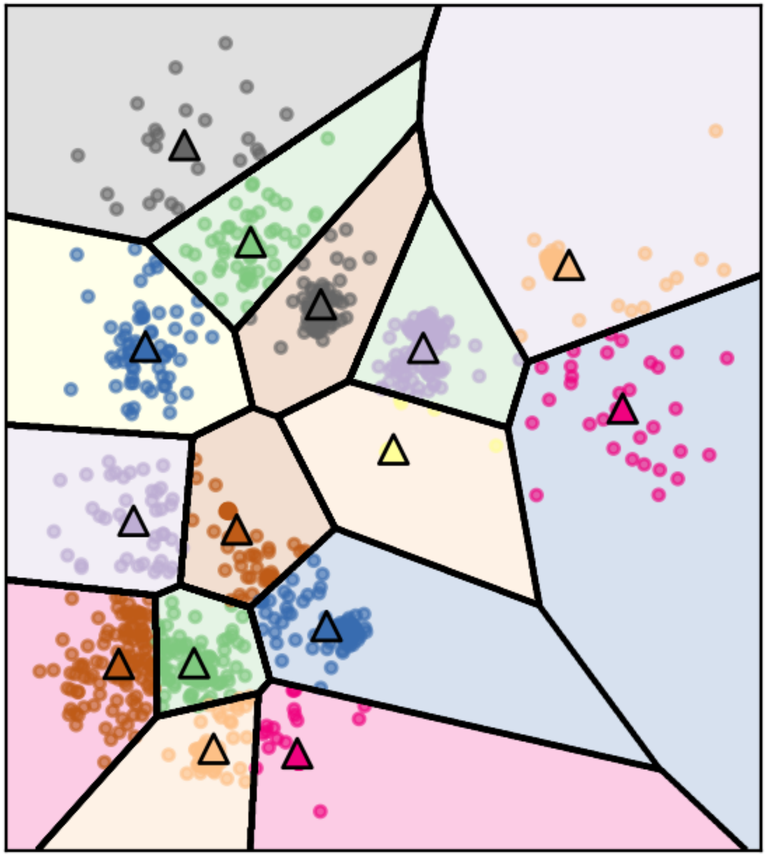
```
X, y = make_blobs(centers=4, random_state=1)

km = KMeans(n_clusters=5, random_state=0)
km.fit(X)
print(km.cluster_centers_.shape)
print(km.labels_.shape)
# predict is the same as labels_ on training data
# but can be applied to new data
print(km.predict(X).shape)
plt.scatter(X[:, 0], X[:, 1], c=km.labels_)
```

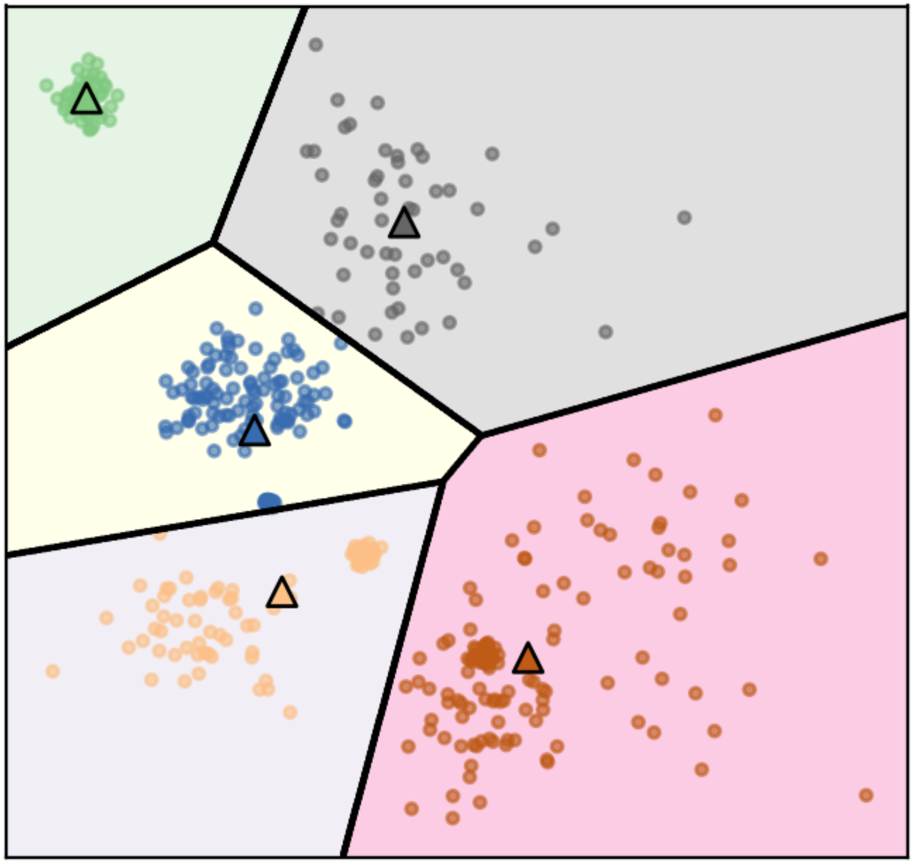
```
(5, 2)
(100,)
(100,)
```



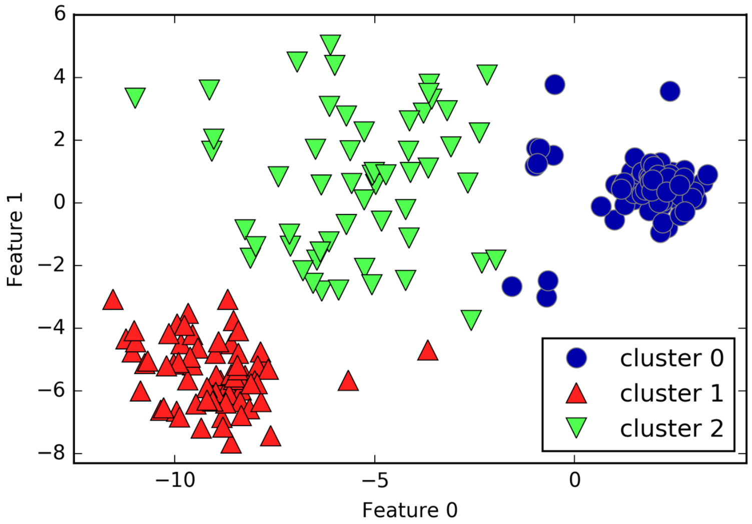
Restriction of Cluster Shapes

- Clusters are Voronoi-diagrams of centers.
- Clusters are always convex in space:

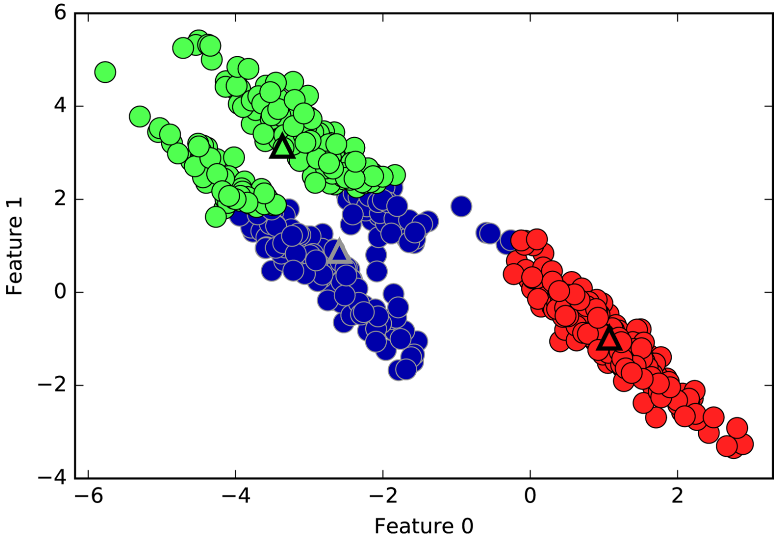




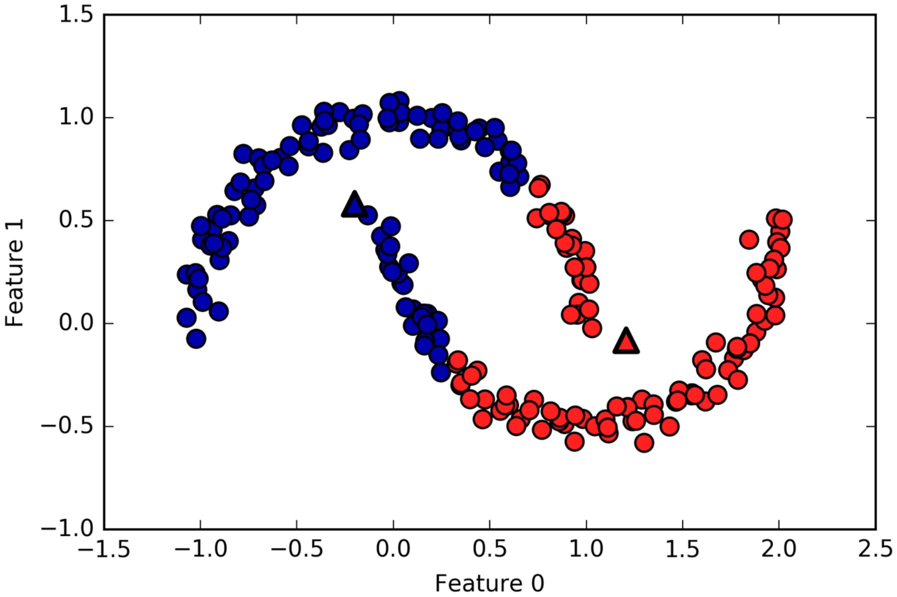
Limitations of k-means



Cluster boundaries are always in the middle of the centers.



Can’t model covariances well



Only simple cluster shapes

Computational properties

- Naive implementation:
  Each iteration computes n_cluster * n_samples distances.
- Fast exact algorithms:

  Elkans, Ying-Yang (compute cluster distances and use triangle inequality)
- Many approximate algorithms, in particular mini-batch K-Means: Much faster, online version (partial_fit)

Initialization

- Random centers fast but not great.
- K-means++ (default):

  Greedily add “furthest way” point
- By default K-means in sklearn does 10 random restarts with different initializations.
- For large datasets, K-means++ initialization may take much longer than clustering.
- Consider using random, in particular for MiniBatchKMeans

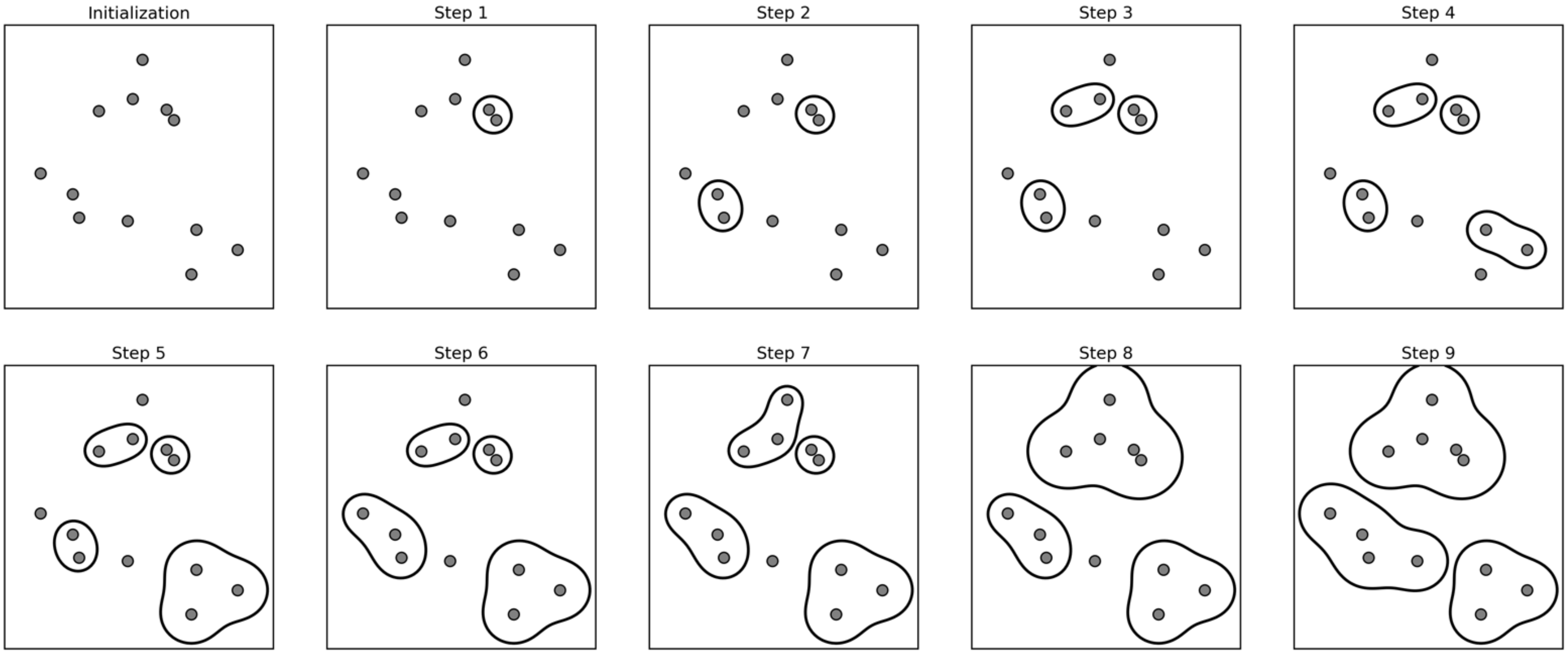
Feature Extraction using k-Means

- Cluster membership → categorical feature
- Cluster distanced → continuous feature
- Examples:
  - partitioning low-dimensional space (similar to using basis functions)
  - Extracting features from high-dimensional spaces, for image patches, see http://ai.stanford.edu/~acoates/papers/coatesleeng_aistats_2011.pdf

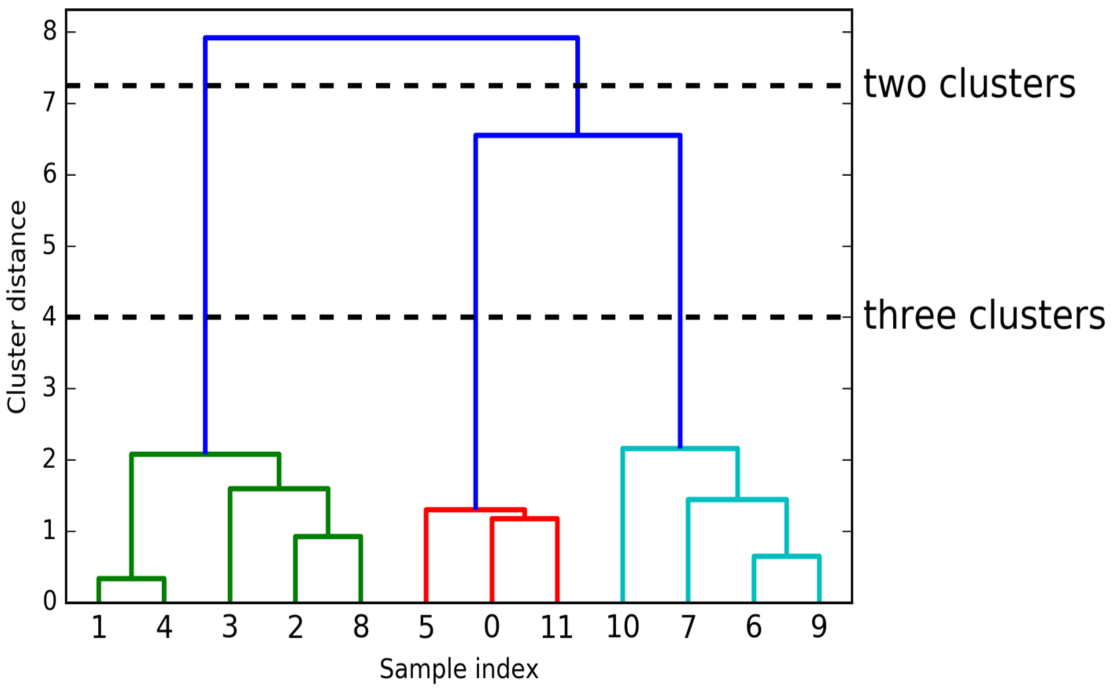
Agglomerative Clustering

Agglomerative Clustering

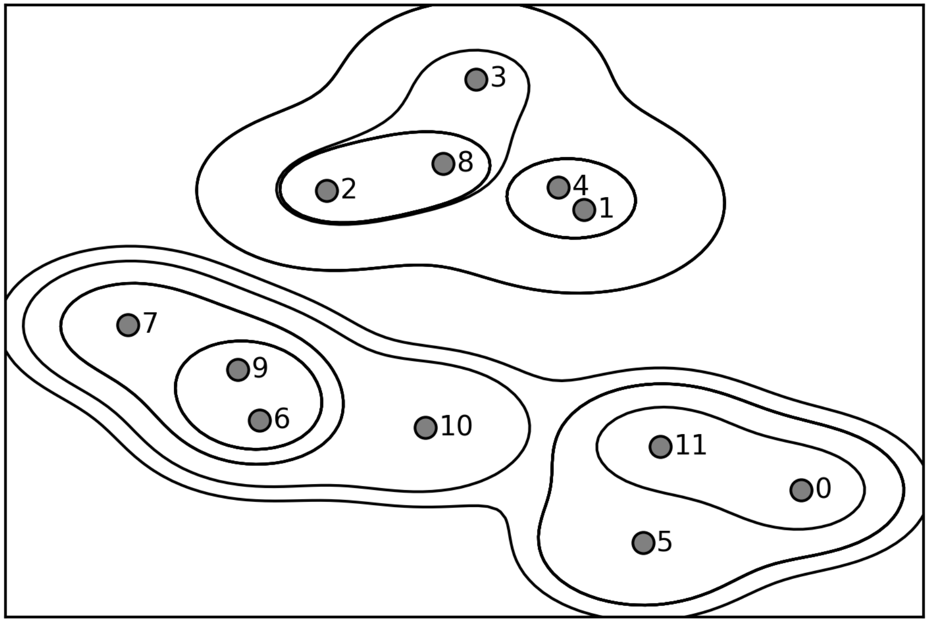
- Start with all points in their own cluster.
- Greedily merge the two most similar clusters.
- Creates a hierarchical clustering from with cluster sizes from n_samples to single cluster.



Dendograms



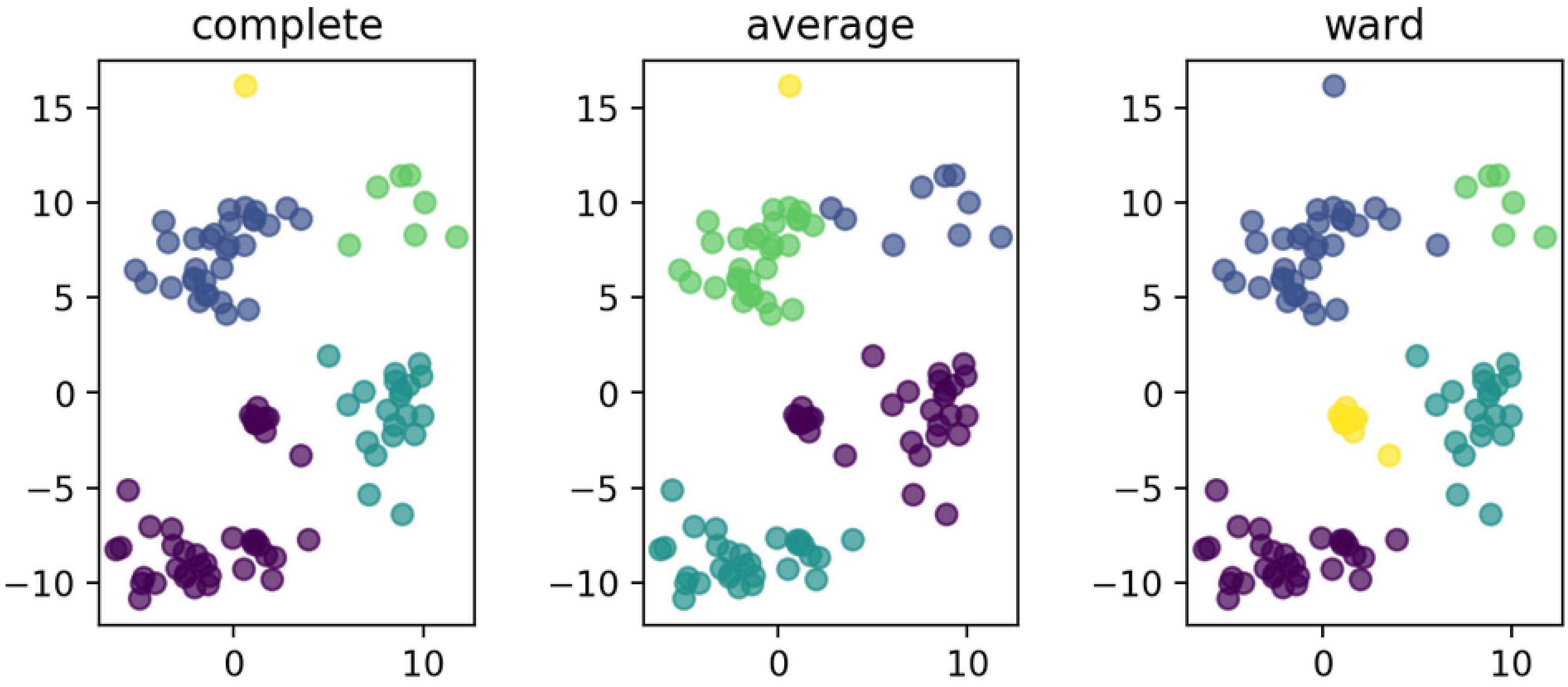
two clusters
three clusters
Cluster distance
0
1
2
3
4
5
6
7
8
1
4
3
2
8
5
0
11
10
7
6
9
Sample index



3
8
2
4
1
7
9
6
10
11
0
5

Merging Criteria

- Complete Link
  - Smallest maximum distance
- Average linkage
  - Smallest average distance between all pairs in the clusters.
- Single Link
  - Smallest minimum distance
- Ward (default in sklearn)
  - Smallest increase in within-cluster variance
  - Leads to more equally sized clusters.



```
complete : [41 31 20  7  1]
average : [31  9 30 29  1]
ward : [30 33 20  6 11]
```

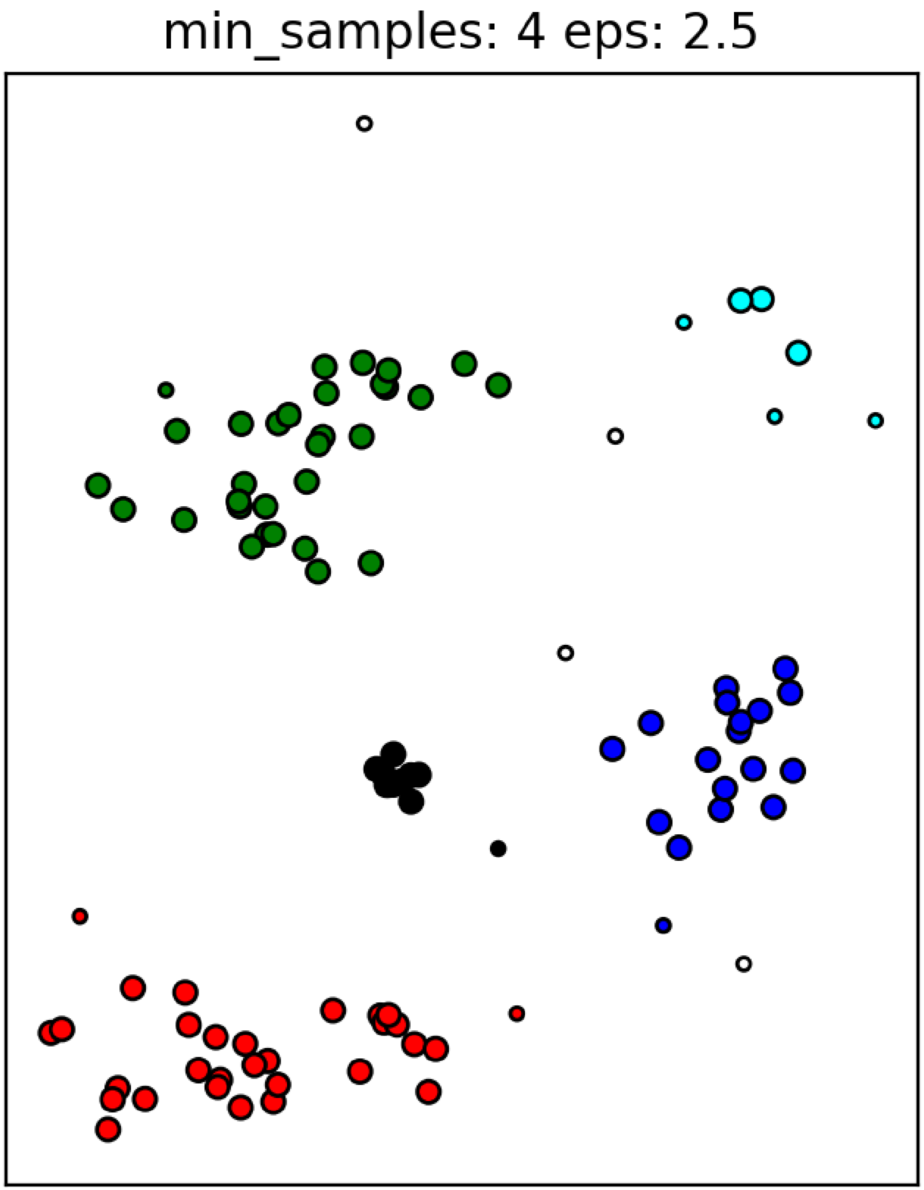
Pros and Cons

- Can restrict to input “topology” given by any graph, for example neighborhood graph.
- Fast with sparse connectivity, otherwise O(log(n) n ** 2)
- Some linkage criteria an lead to very imbalanced cluster sized (can be pro or con)
- Hierarchical clustering gives more holistic view, can help with picking the number of clusters.

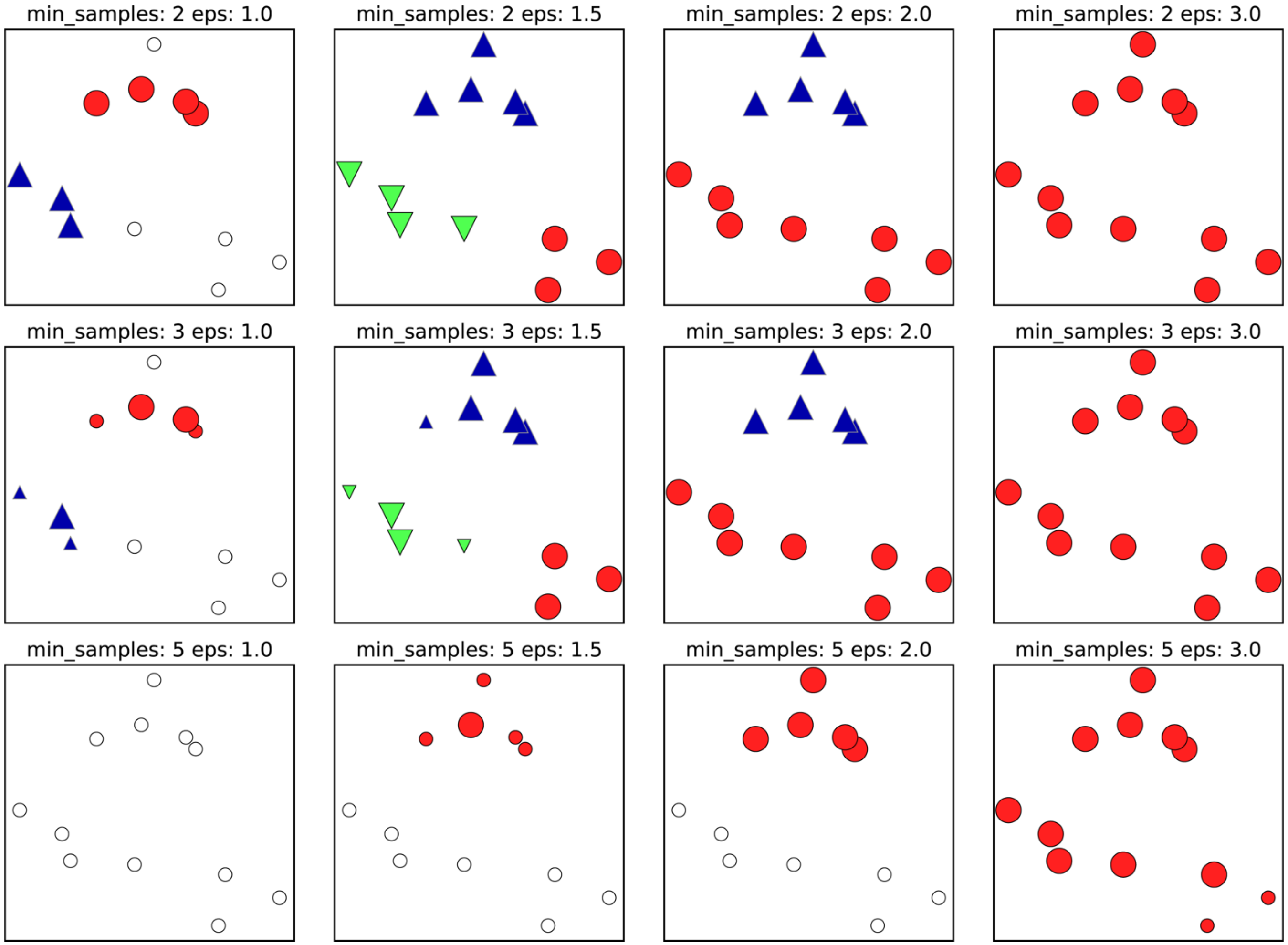
DBSCAN

Algorithm

- Clusters are formed by “core samples”
- Sample is “core sample” if more than min_samples is within epsilon - “dense region”
- Start with a core sample
- Recursively walk neighbors that are core-samples and add to cluster.
- Also add samples within epsilon that are not core samples (but don’t recurse)
- If can’t reach any more points, pick another core sample, start new cluster.
- Remaining points are labeled outliers

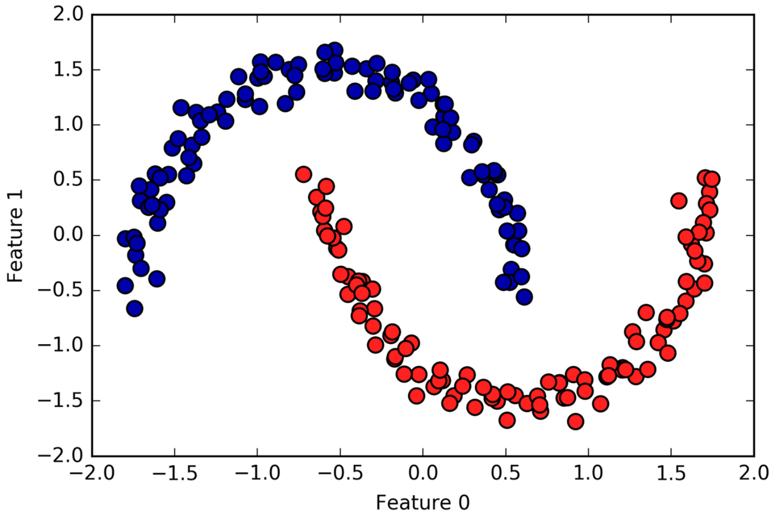


Illustration



Pros and Cons

- Allows complex cluster shapes
- Can detect outliers
- Needs two parameters to adjust, eps is hard to pick (can be done based on number of clusters though).
- Can learn arbitrary cluster shapes



(Gaussian) Mixture Models

Mixture models

- Generative model – find p(X).
- Assume form of data generating process
- Mixture model assumption:
  - Data is mixture of small number of known distributions.
  - Each mixture component distribution can be learned "simply"
  - Each point comes from one particular component
- We learn the component parameters and weights of components

$$p(\mathbf{x}) = \sum_{j=1}^{k} \pi_k p_k(\mathbf{x}|\theta)$$

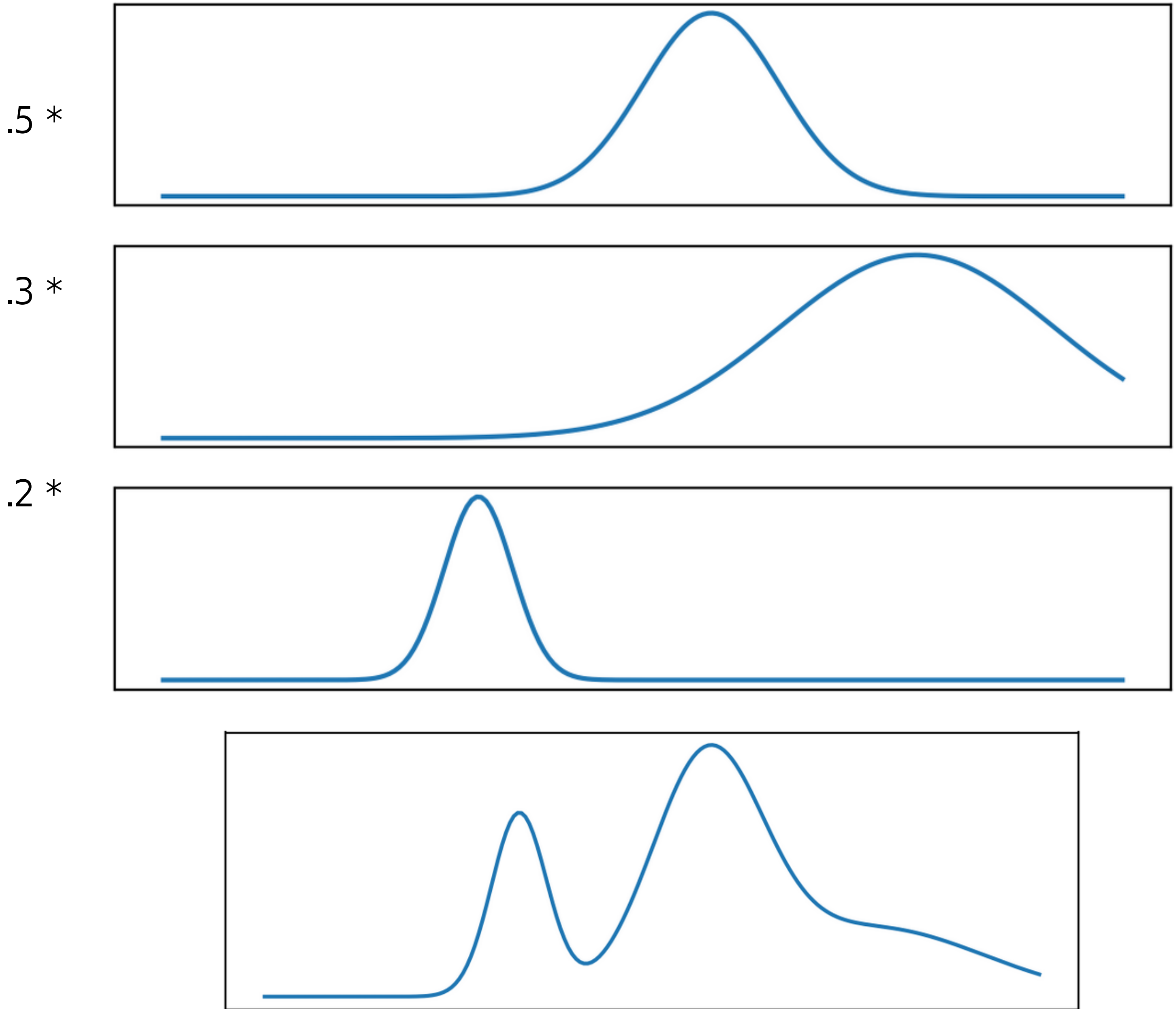
Gaussian Mixture Models

- Each component is created by a Gaussian distribution.
- There is a multinomial distribution over the components.

$$p(\mathbf{x}) = \sum_{j=1}^{k} \pi_k \mathcal{N}(\mathbf{x}, \mu_k, \Sigma_k)$$

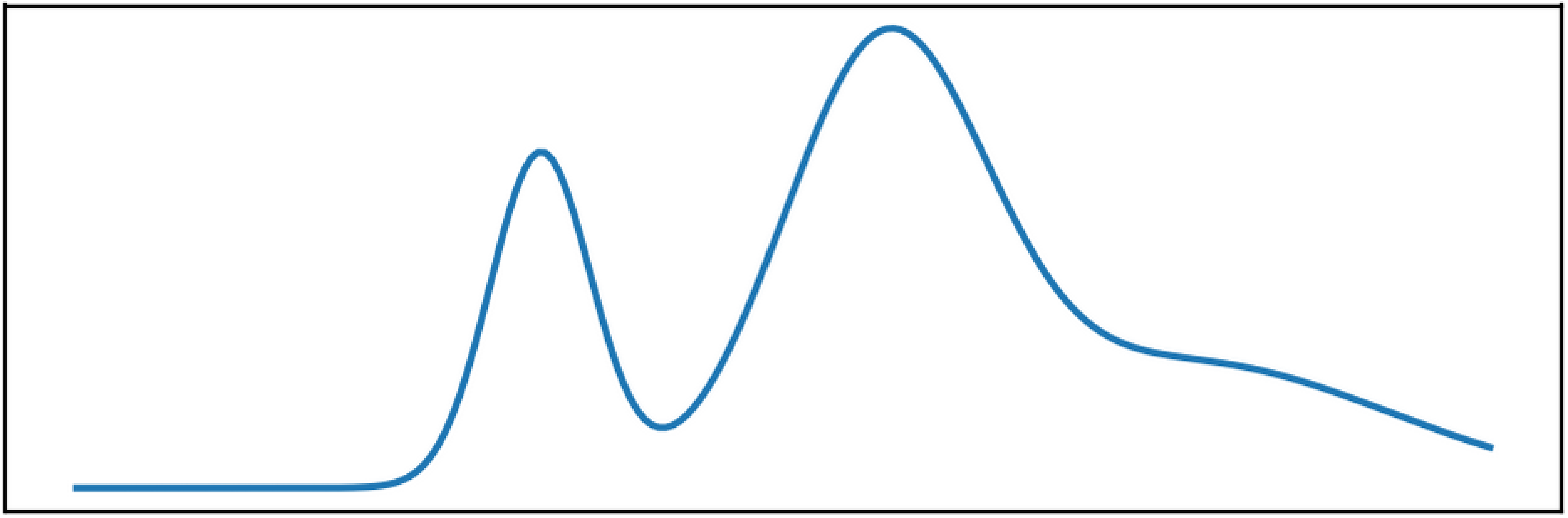
- Non-convex optimization.
- Alternately assign points to components and compute mean and variance (EM algorithm).
- Initialized with K-means, random restarts.

$$p(\mathbf{x}) = \sum_{j=1}^{k} \pi_k \mathcal{N}(\mathbf{x}, \mu_k, \Sigma_k)$$

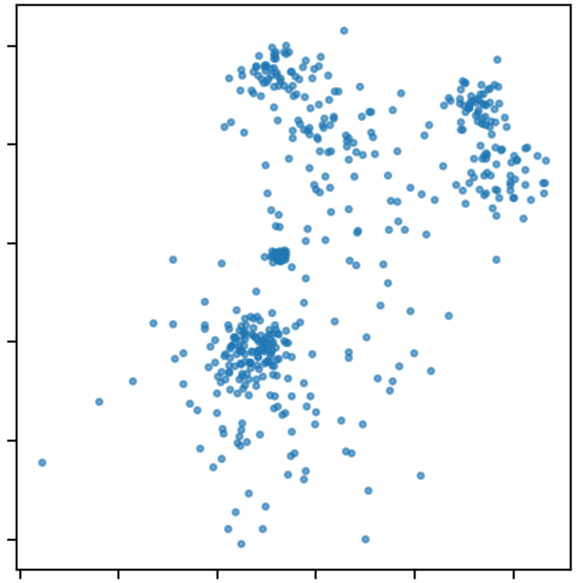


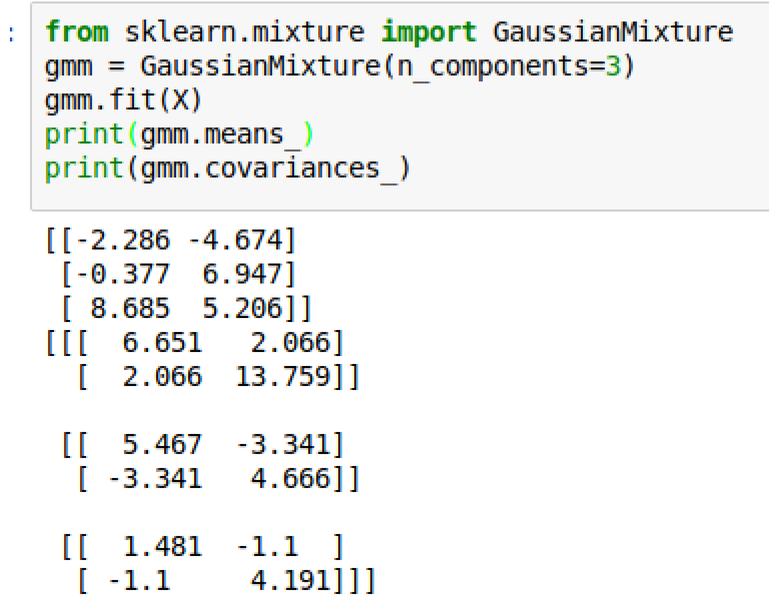
Goals of Mixture Models

- Create parametric density model.
- Allows for testing how “likely” a new point is.
- Clustering (each components is one cluster).
- Feature Extraction



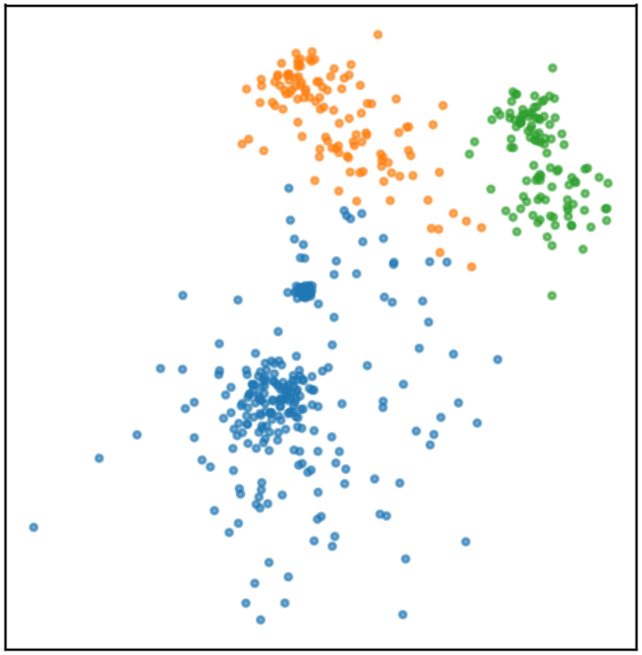
Examples



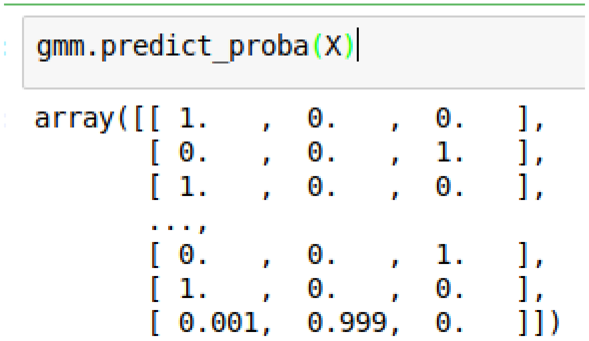


```
from sklearn.mixture import GaussianMixture
gmm = GaussianMixture(n_components=3)
gmm.fit(X)
print(gmm.means_)
print(gmm.covariances_)
```

```
[[-2.286 -4.674]
 [-0.377  6.947]
 [ 8.685  5.206]]
[[[  6.651   2.066]
  [  2.066  13.759]]

 [[  5.467  -3.341]
  [ -3.341   4.666]]

 [[  1.481  -1.1  ]
  [ -1.1     4.191]]]
```

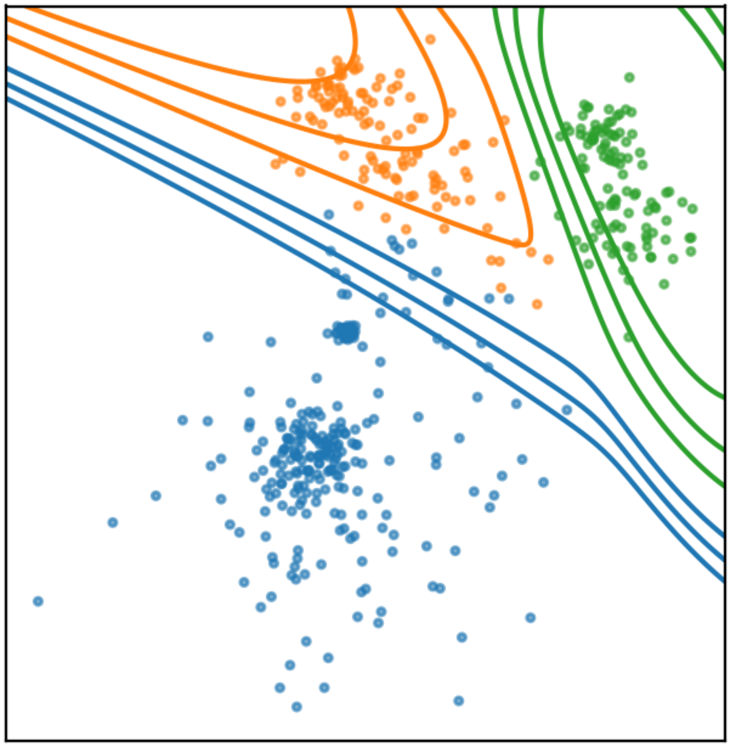


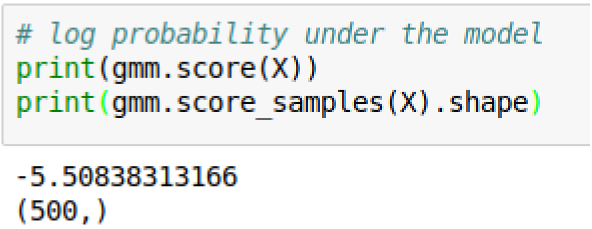
Probability estimates



```
gmm.predict_proba(X)
```

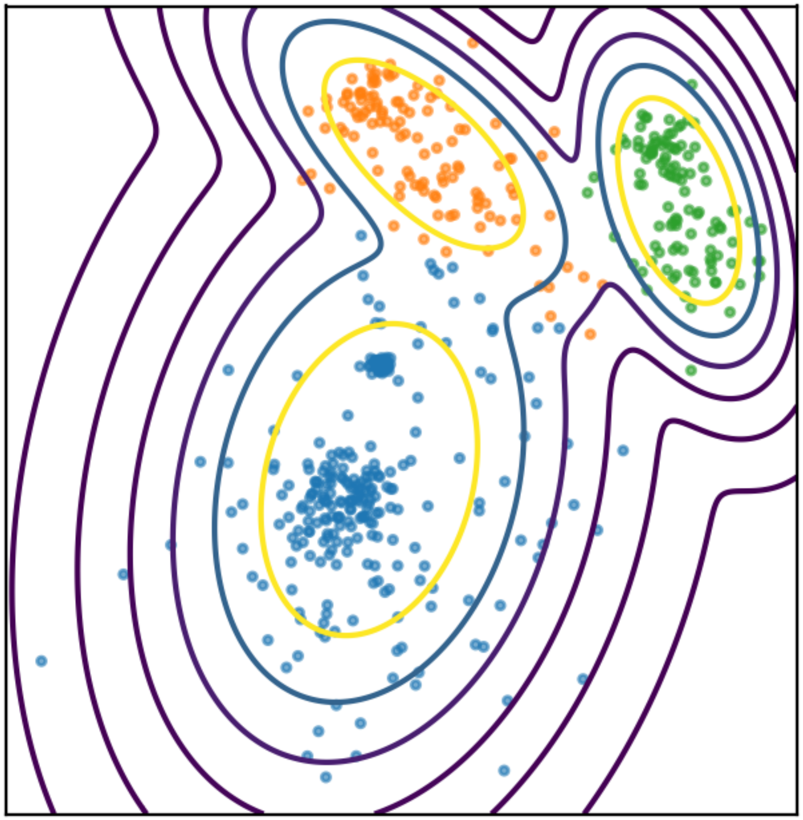
```
array([[ 1.   ,  0.   ,  0.   ],
       [ 0.   ,  0.   ,  1.   ],
       [ 1.   ,  0.   ,  0.   ],
       ...,
       [ 0.   ,  0.   ,  1.   ],
       [ 1.   ,  0.   ,  0.   ],
       [ 0.001,  0.999,  0.   ]])
```





```
# log probability under the model
print(gmm.score(X))
print(gmm.score_samples(X).shape)
```

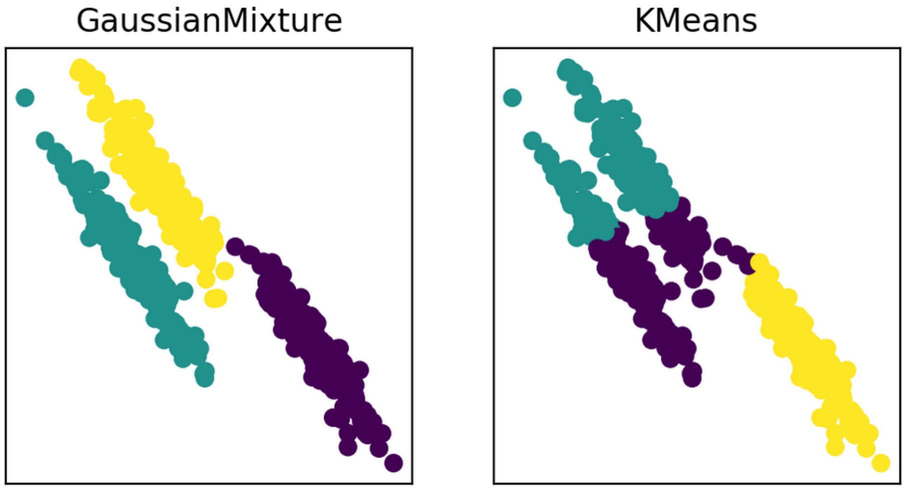
```
-5.50838313166
(500,)
```

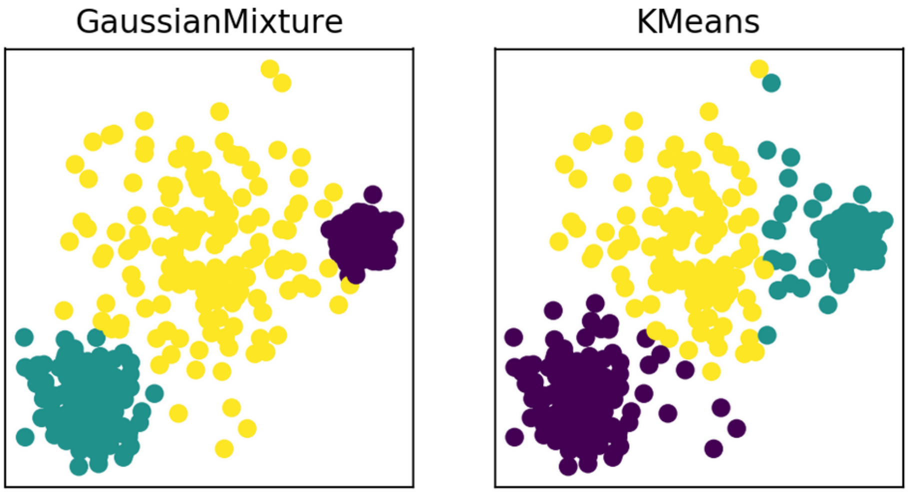


Notes

- In high dimensions, covariance=”full” might not work.
- Try covariance=”diagonal”, covariance=”tied”.
- Restart with different initializations (default=1)

GMM vs KMeans

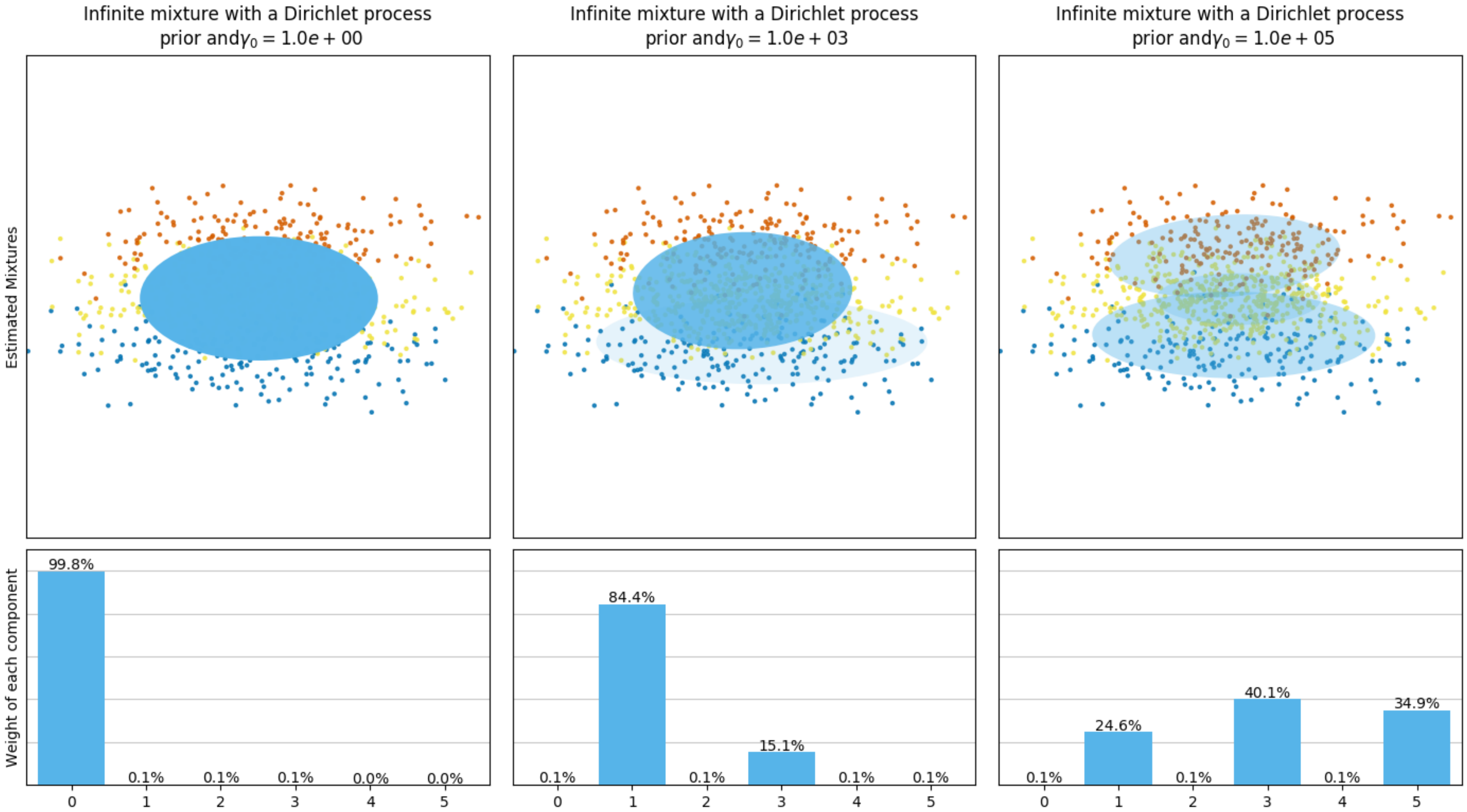




GMM can be more unstable, is more expensive to compute.

Bayesian Infinite Mixtures

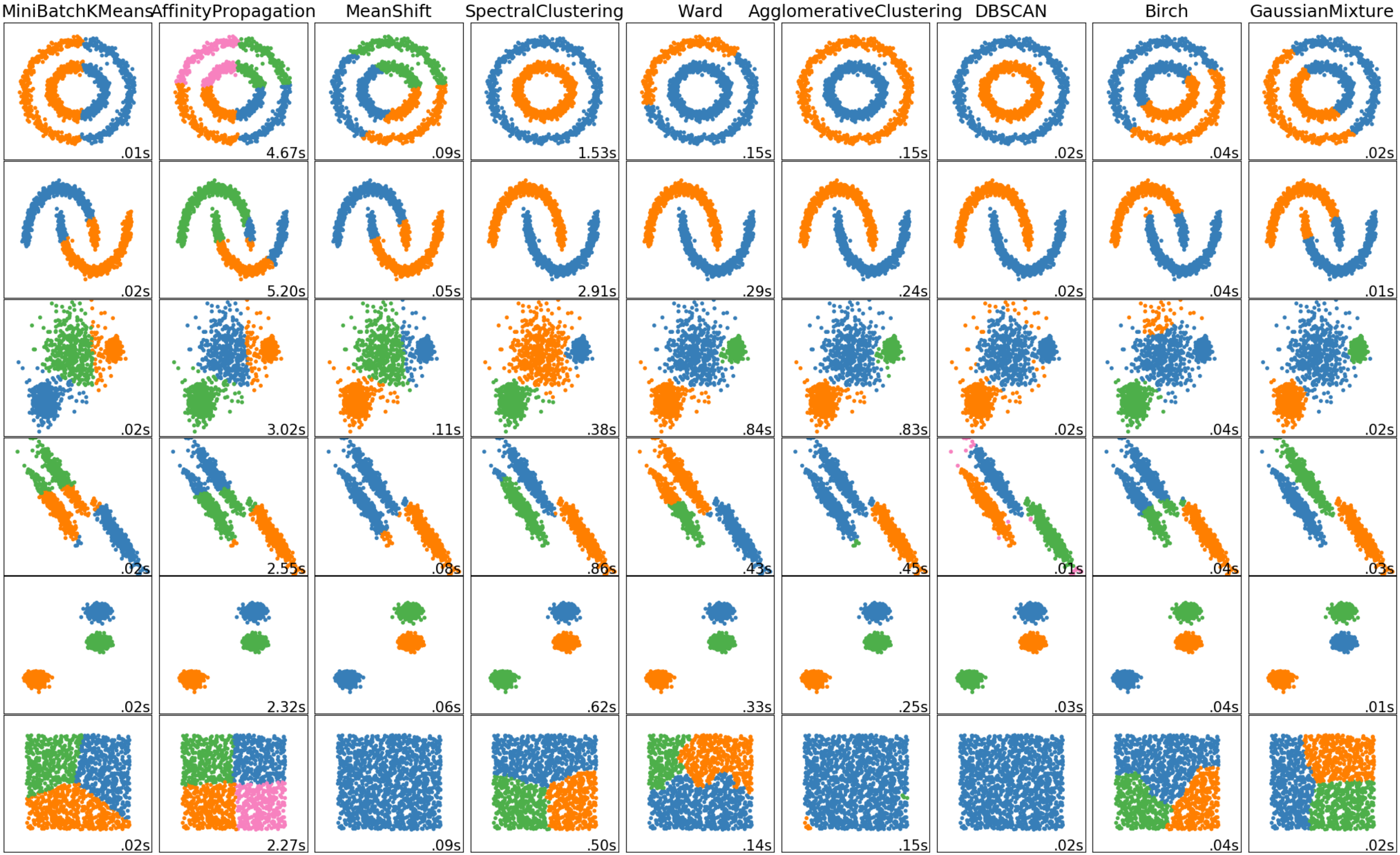
- Bayesian treatment:
  - Add priors on mixture coefficients and Gaussians.
  - Can “unselect” components if they don’t contribute.
  - Possibly more robust
- Infinite Mixtures:
  - Replace Dirichlet Prior over mixture coefficients by Dirichlet Process.
  - “automatically” finds number of components (based on parameter of the prior).
- Both implemented in BayesianGaussianMixture
- Use variational inference (as opposed to gibbs sampling)
- In practice: need to specify upper limit of components



Infinite mixture with a Dirichlet process prior and$\gamma_0 = 1.0e + 00$

Infinite mixture with a Dirichlet process prior and$\gamma_0 = 1.0e + 03$

Infinite mixture with a Dirichlet process prior and$\gamma_0 = 1.0e + 05$



http://scikit-learn.org/dev/auto_examples/cluster/plot_cluster_comparison.html