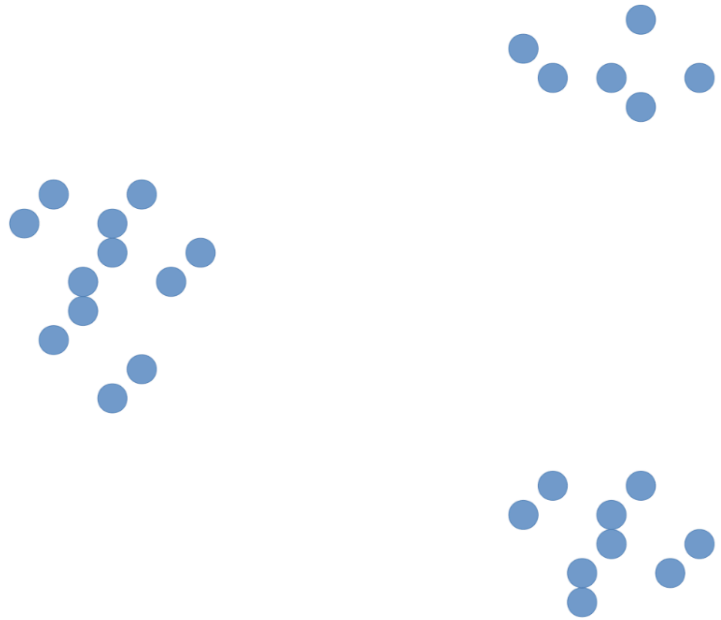
W4995 Applied Machine Learning

Clustering and Mixture Models

03/27/19

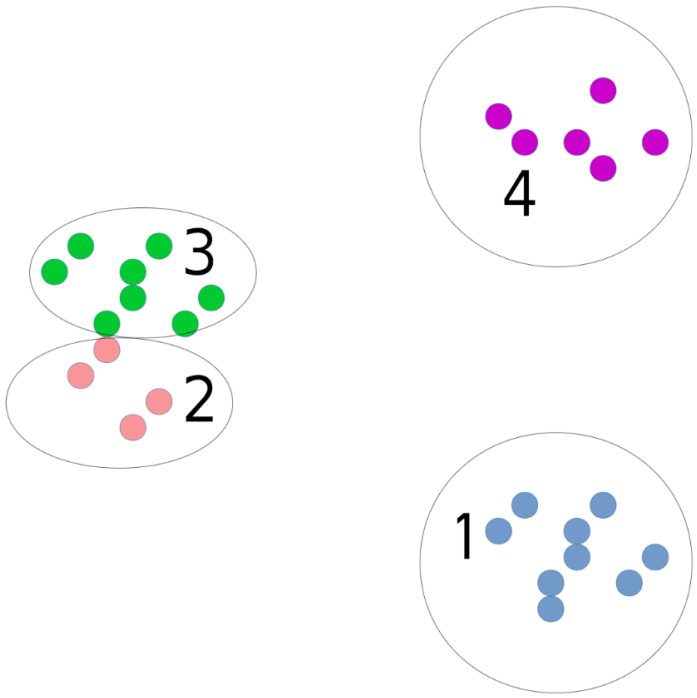
Andreas C. Müller

Clustering



Clustering

- Partition data into groups (clusters)
- Points within a cluster should be “similar”.
- Points in different cluster should be “different”.


4
3
2
1

Goals of Clustering

- Data Exploration
  - Are there coherent groups ?
  - How many groups are there ?

Goals of Clustering

- Data Exploration
  - Are there coherent groups ?
  - How many groups are there ?
- Data Partitioning
  - Divide data by group before further processing

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

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
- Evaluation and parameter tuning
  - Quantitative measures of limited use
  - Usually qualitative measures used
  - Best: downstream tasks

Clustering Algorithms

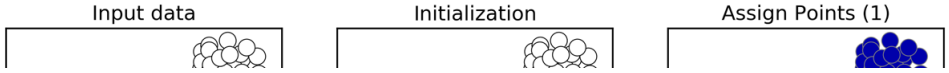
K-Means

Objective function for K-Means

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

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

K-Means algorithm



- Pick number of clusters k.

K-Means API

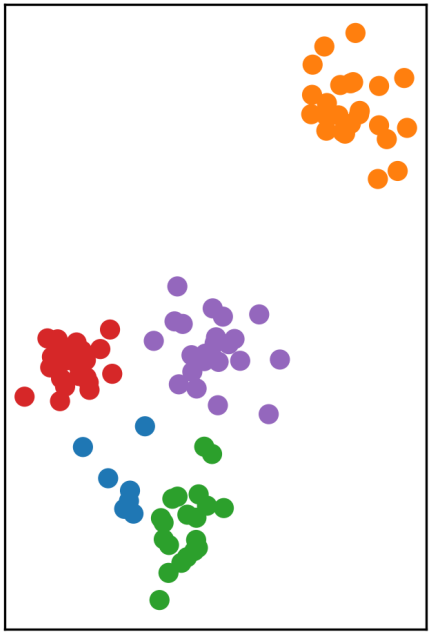
```
X, y = make_blobs(centers=4, random_state=1)

km = KMeans(n_clusters=5, random_state=0)
km.fit(X)
print(km.cluster_centers_.shape)
print(km.labels_.shape)
```

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

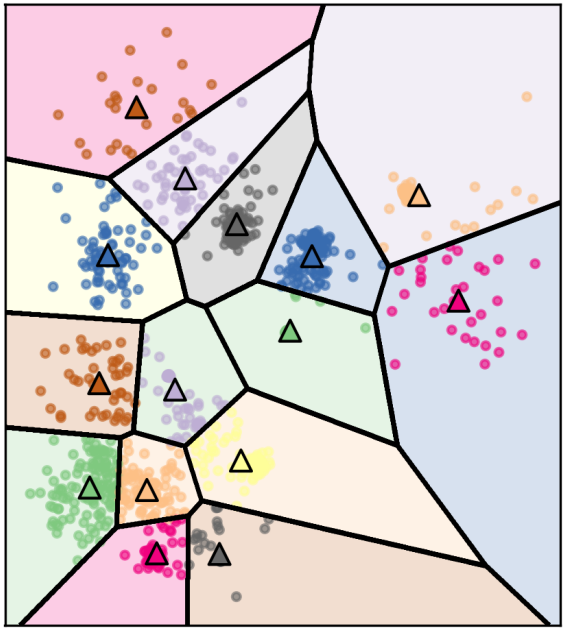
```
# predict is the same as labels_ on training data
# but can be applied to new data
print(km.predict(X).shape)
```

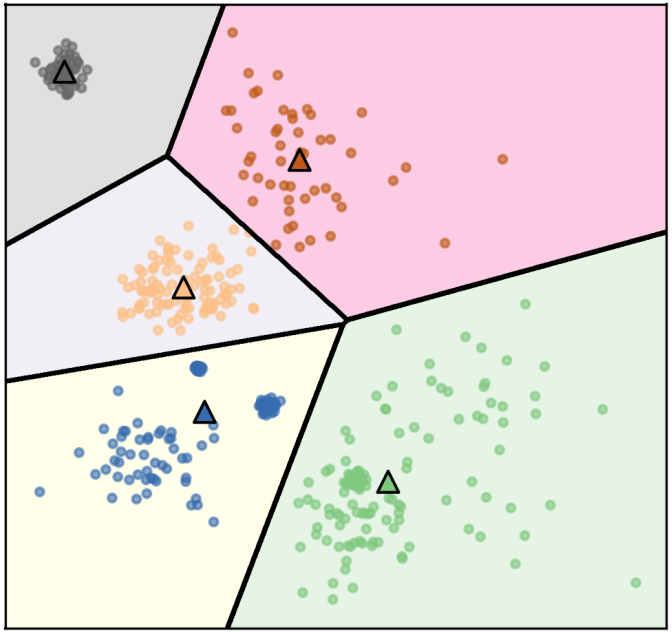
```
(100,)
```

```
plt.scatter(X[:, 0], X[:, 1], c=km.labels_)
```



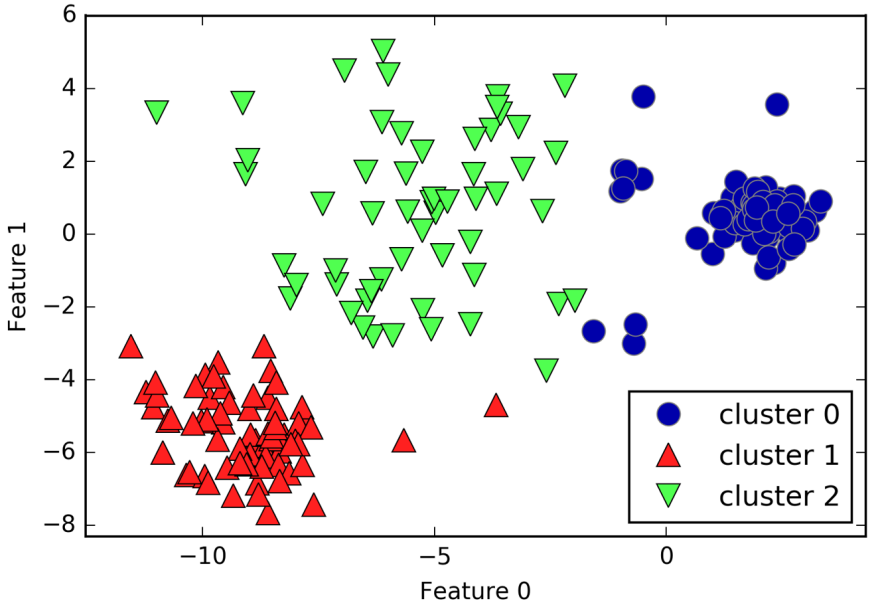
Restriction of Cluster Shapes





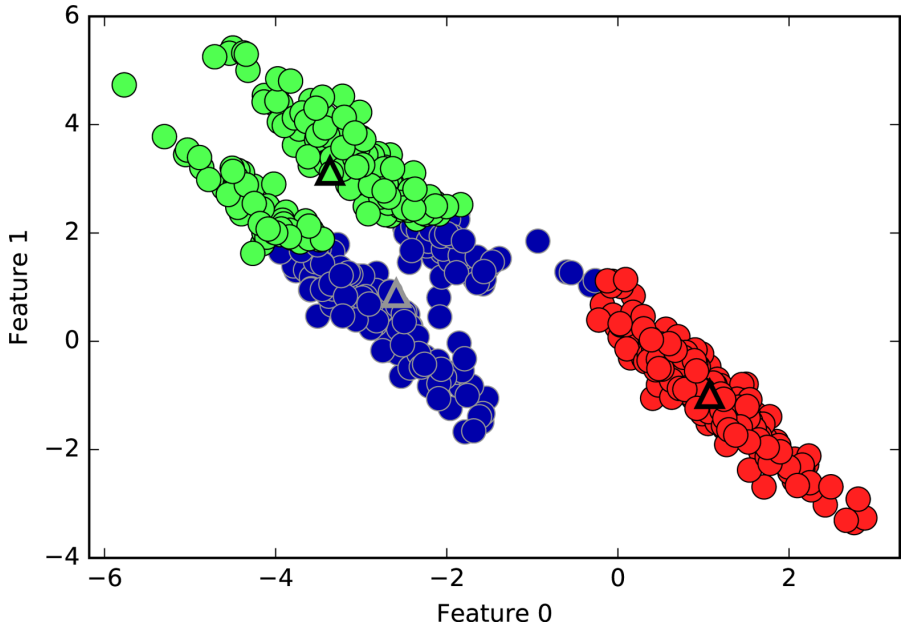
Clusters are Voronoi-diagrams of centers

Limitations of K-Means



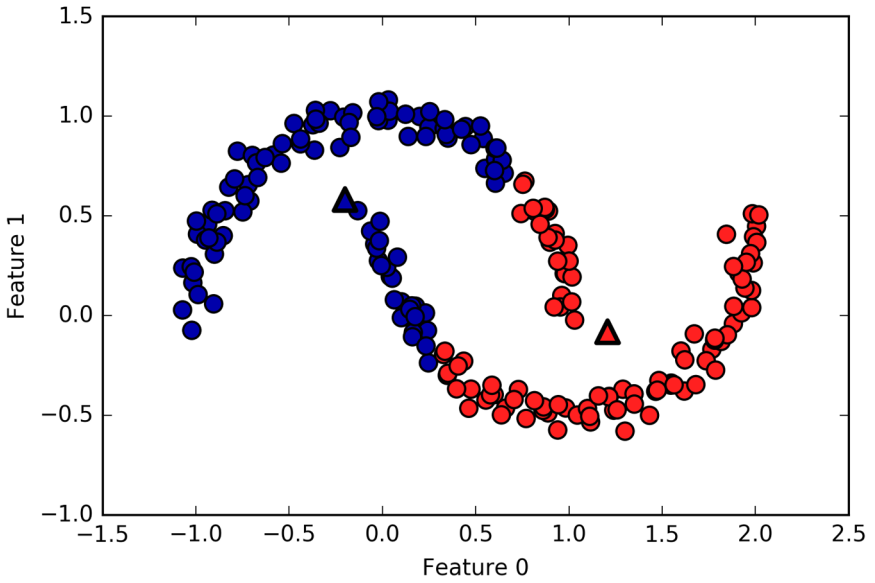
- Cluster boundaries equidistant to centers

Limitations of K-Means



- Can’t model covariances well

Limitations of K-Means



- Only simple cluster shapes

Computational Properties

- Naive implementation (Lloyd's):
  - n_cluster * n_samples distance calculations per iteration

Computational Properties

- Naive implementation (Lloyd's):
  - n_cluster * n_samples distance calculations per iteration
- Fast "exact" algorithms:
  - Elkan's, Ying-Yang

Computational Properties

- Naive implementation (Lloyd's):
  - n_cluster * n_samples distance calculations per iteration
- Fast "exact" algorithms:
  - Elkan's, Ying-Yang
- Approximate algorithms:
  - minibatch K-Means

Initialization

- Random centers fast.
- K-means++ (default): Greedily add “furthest way” point
- By default K-means in sklearn does 10 random restarts with different initializations.
- K-means++ initialization may take much longer than clustering.

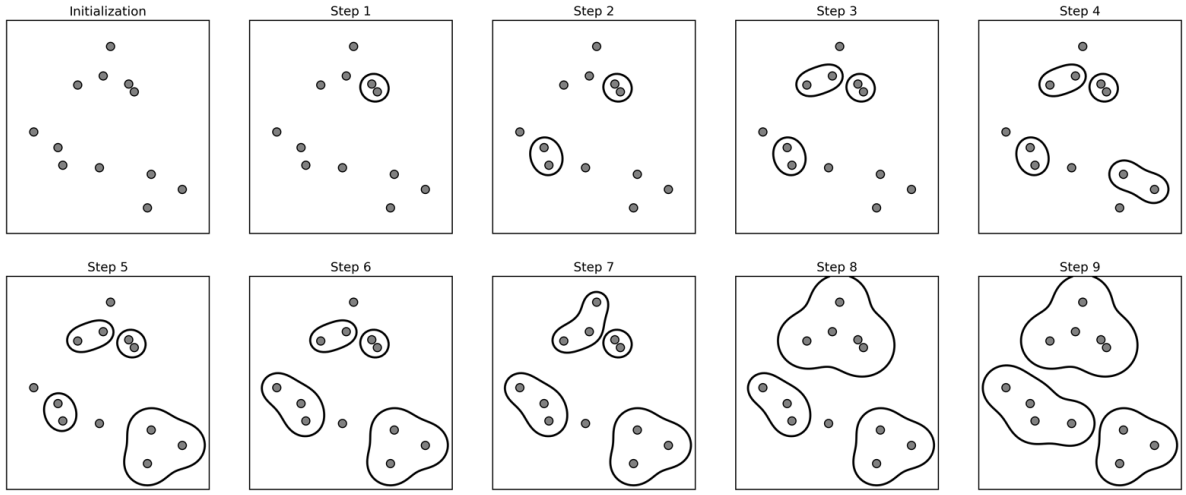
Feature Extraction using K-Means

- 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

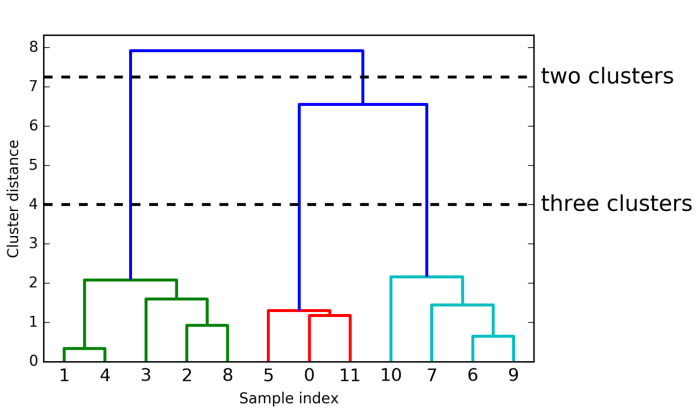
Agglomerative Clustering

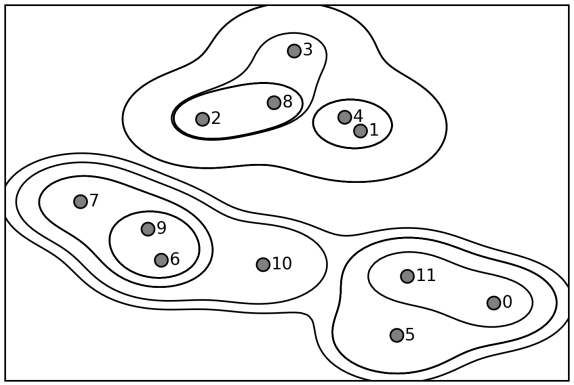
Agglomerative Clustering

- Start with all points in their own cluster.
- Greedily merge the two most similar clusters.

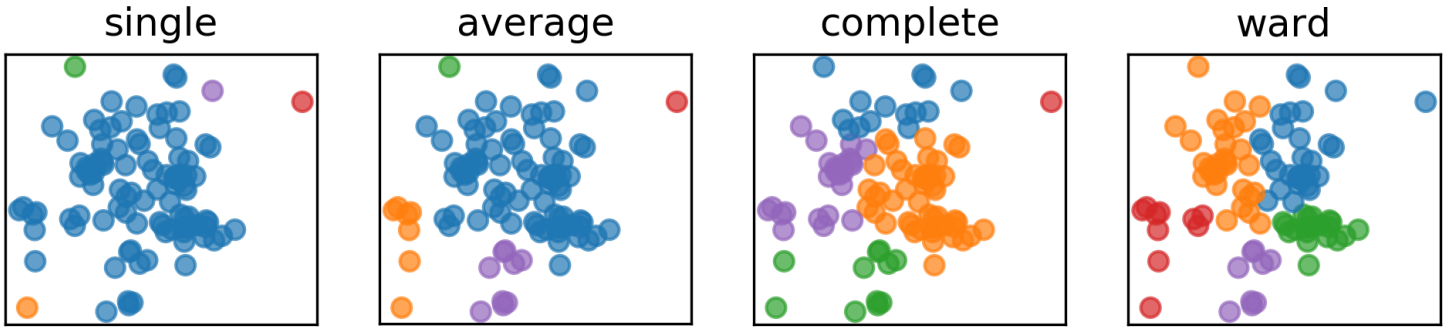


Dendograms


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


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

Linkage Criteria



Cluster sizes:

```
single :   [96  1  1  1  1]
average :  [82  9  7  1  1]
complete : [50 24 14 11  1]
ward :     [31 30 20 10  9]
```

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

Pros and Cons

- Can restrict to input “topology” graph.

Pros and Cons

- Can restrict to input “topology” graph.
- Fast with sparse connectivity, otherwise O(log(n) n ** 2)

Pros and Cons

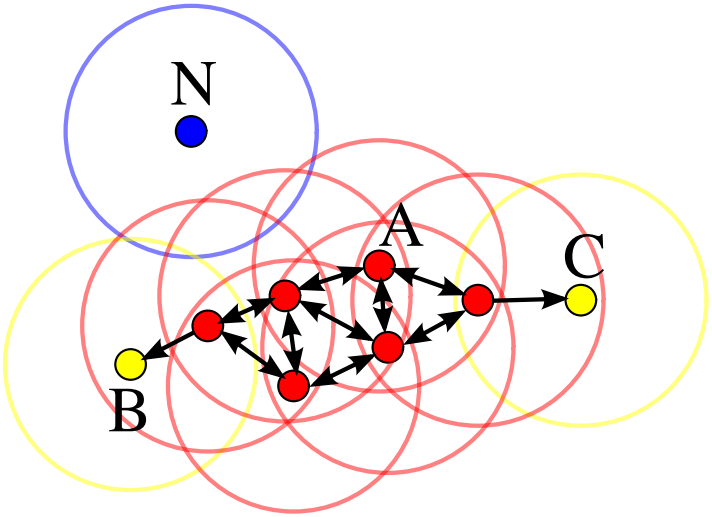
- Can restrict to input “topology” graph.
- Fast with sparse connectivity, otherwise O(log(n) n ** 2)
- Some linkage criteria an lead to very imbalanced cluster sizes

Pros and Cons

- Can restrict to input “topology” graph.
- Fast with sparse connectivity, otherwise O(log(n) n ** 2)
- Some linkage criteria an lead to very imbalanced cluster sizes
- Hierarchical clustering gives more holistic view than single clustering.

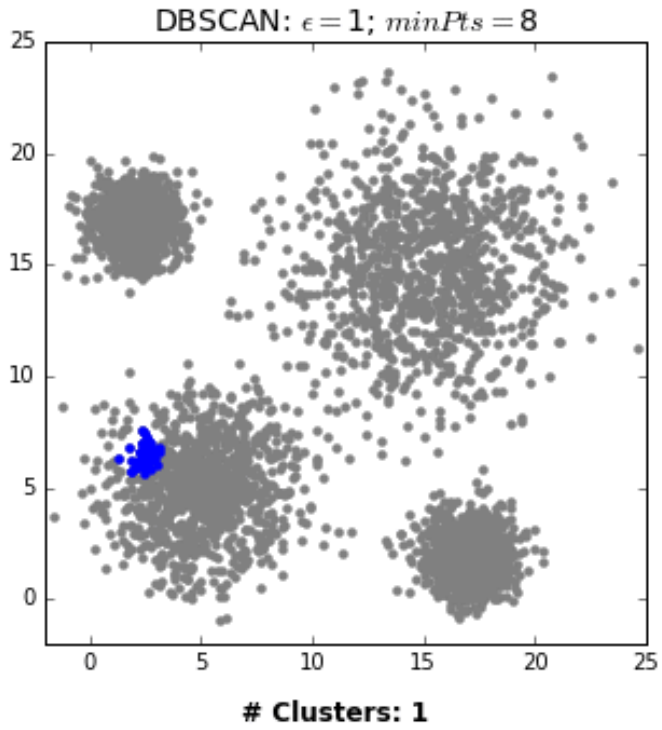
DBSCAN

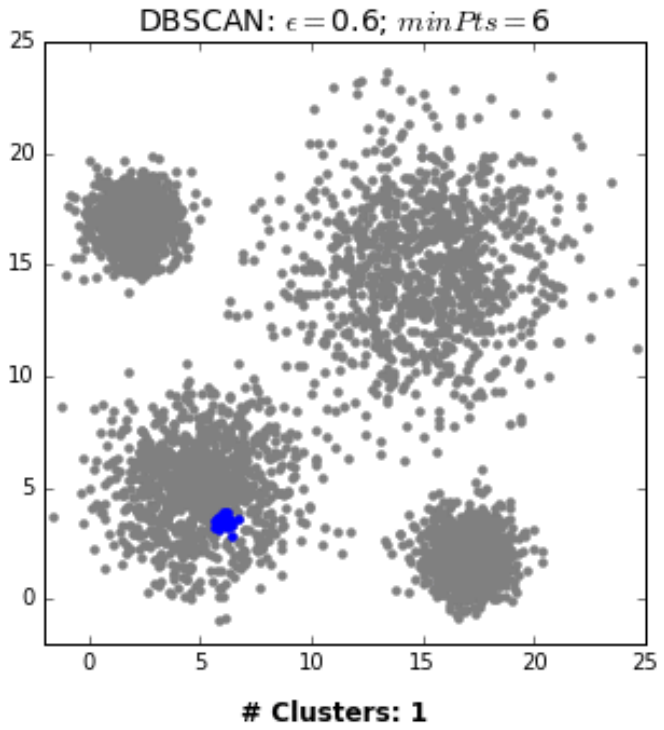
Algorithm



- eps: neighborhood radius
- min_samples: 4

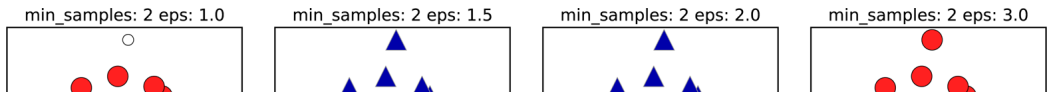
- A: Core
- B, C: not core
- N: noise





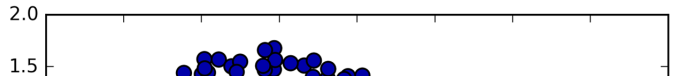
by David Sheehan dashee87.github.io

Illustration of Parameters



Pros and Cons

- Pro: Can learn arbitrary cluster shapes
- Pro: Can detect outliers
- Con: Needs two (non-obvious?) parameters to adjust



(Gaussian) Mixture Models

Mixture Models

- Generative model: find p(X).

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

Gaussian Mixture Models

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

$$k \sim \text{Mult}(\pi), x \sim \mathcal{N}(\mathbf{x}, \mu_k, \Sigma_k)$$

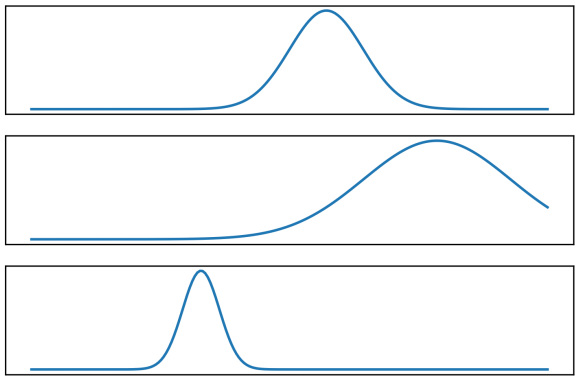
Gaussian Mixture Models

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

$$k \sim \text{Mult}(\pi), x \sim \mathcal{N}(\mathbf{x}, \mu_k, \Sigma_k)$$

- Non-convex optimization.
- Initialized with K-means, random restarts.
- Optimization (EM algorithm):
  - Soft-assign points to components.
  - Compute mean and variance of components.
  - Iterate

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



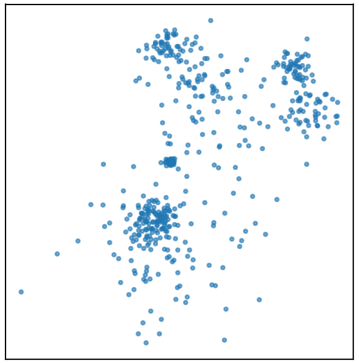
Why Mixture Models?

- Clustering (components are clusters)
- Parametric density model

Examples

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

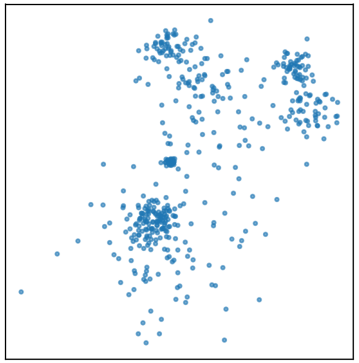
```
[[-2.286 -4.674]
 [-0.377  6.947]
 [ 8.685  5.206]]
[[[  6.651   2.066]
  [  2.066  13.759]]
 [[  5.467  -3.341]
  [ -3.341   4.666]]
 [[  1.481  -1.1  ]
  [ -1.1     4.191]]]
```

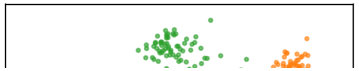


Examples

```
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]]]
```





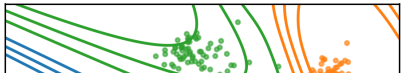
Probability Estimates

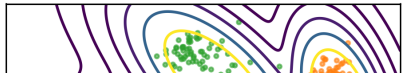
```
gmm.predict_proba(X)
```

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

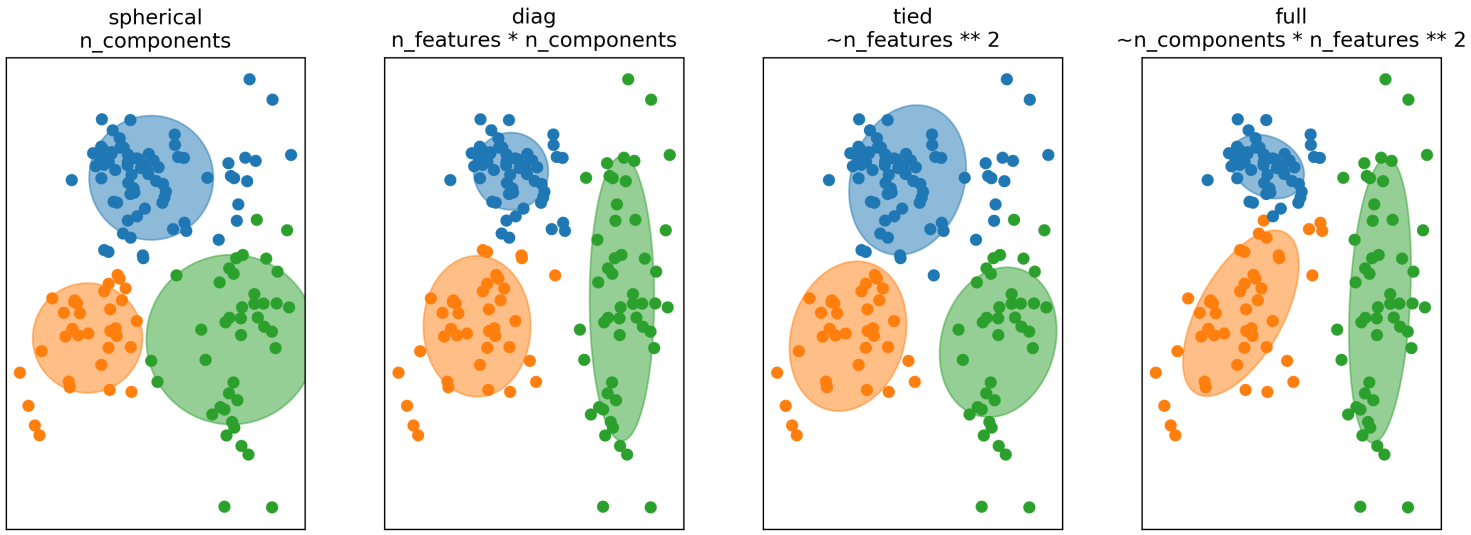
```
# log probability under the model
print(gmm.score(X))
print(gmm.score_samples(X).shape)
```

```
-5.508
(500,)
```

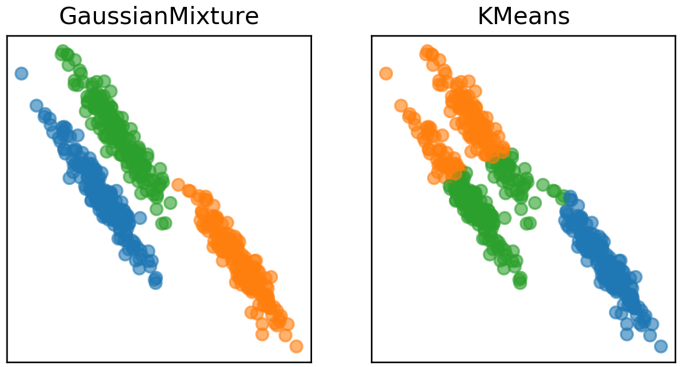




Covariance restrictions



GMM vs KMeans



Summary

- KMeans
  Classic, simple. Only convex cluster shapes, determined by cluster centers.

Summary

- KMeans
  Classic, simple. Only convex cluster shapes, determined by cluster centers.

- Agglomerative
  Can take input topology into account, can produce hierarchy.

Summary

- KMeans
  Classic, simple. Only convex cluster shapes, determined by cluster centers.

- Agglomerative
  Can take input topology into account, can produce hierarchy.

- DBSCAN
  Arbitrary cluster shapes, can detect outliers, often very different sized clusters.

Summary

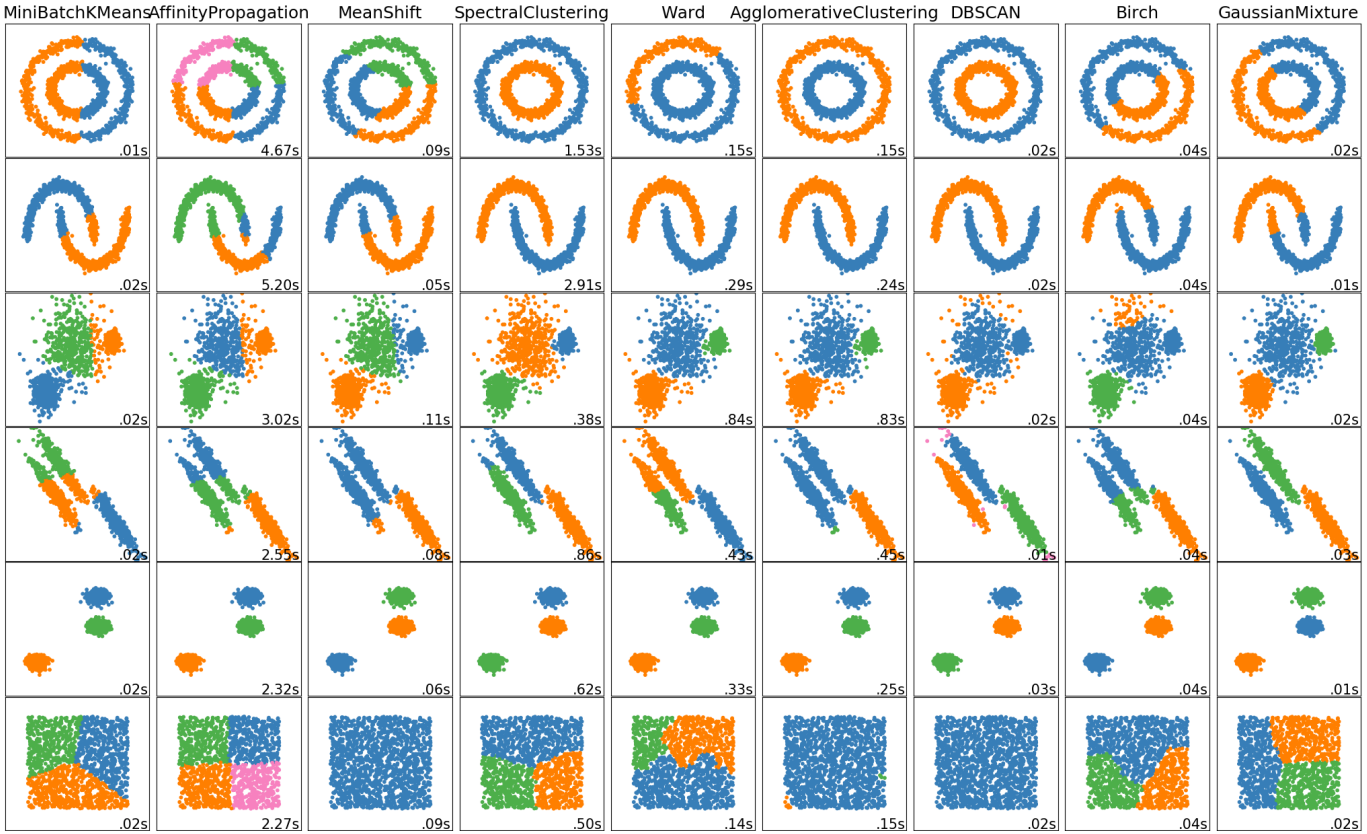
- KMeans
  Classic, simple. Only convex cluster shapes, determined by cluster centers.

- Agglomerative
  Can take input topology into account, can produce hierarchy.

- DBSCAN
  Arbitrary cluster shapes, can detect outliers, often very different sized clusters.

- Gaussian Mixture Models

  Can model covariance, soft clustering, can be hard to fit.



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

55 / 56

Questions ?