CS4487 - Machine Learning

Lecture 6b - Unsupervised Learning - Clustering

Dr. Antoni B. Chan

Dept. of Computer Science, City University of Hong Kong

Outline

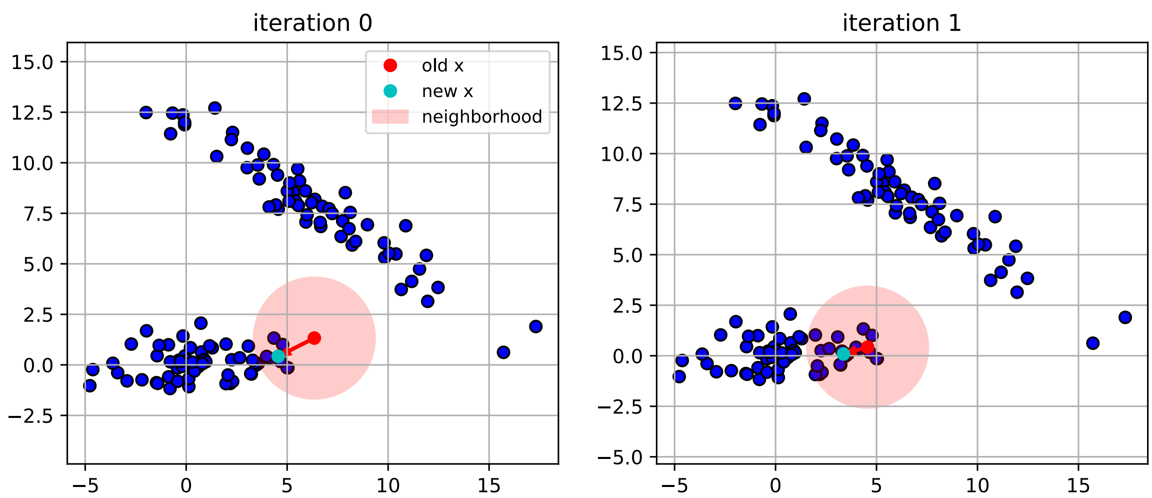
1. Unsupervised Learning
2. Parametric clustering
   A. K-means
   B. Gaussian mixture models (GMMs)
   C. Dirichlet Process GMMs
3. **Non-parametric clustering and Mean-shift**
4. Spectral clustering

Mean-shift algorithm

- Clustering algorithm that also automatically selects the number of clusters.
- **Idea:** iteratively shift towards the largest concentration of points.
  - start from an initial point $\mathbf{x}$ (e.g., one of the data points).
  - repeat until $\mathbf{x}$ is unchanged:
    - 1) find the nearest neighbors to $\mathbf{x}$ within some radius (bandwidth)
    - 2) set $\mathbf{x}$ to be the mean of the neighbor points.

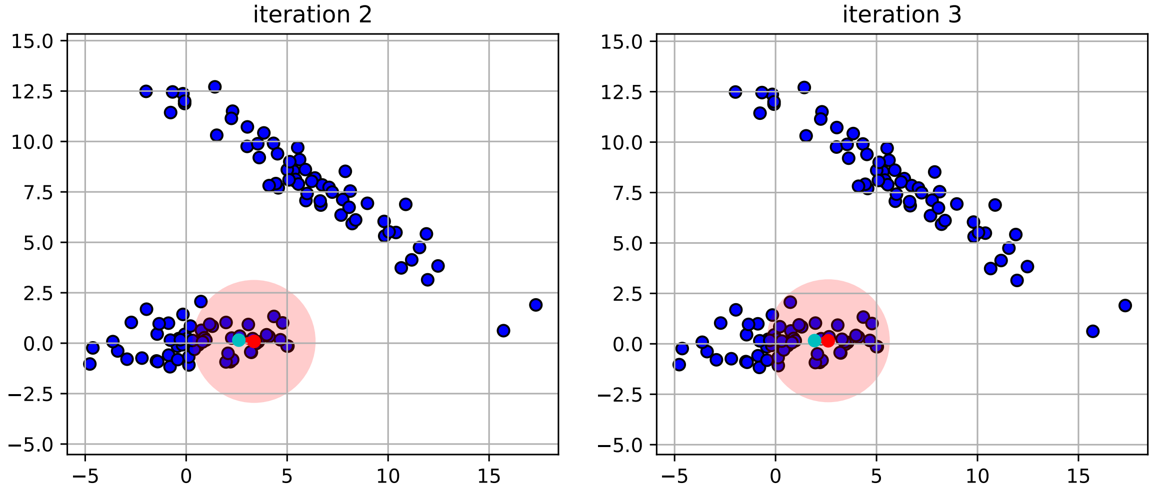
In [5]:
```
efigs[0]
```

Out[5]:



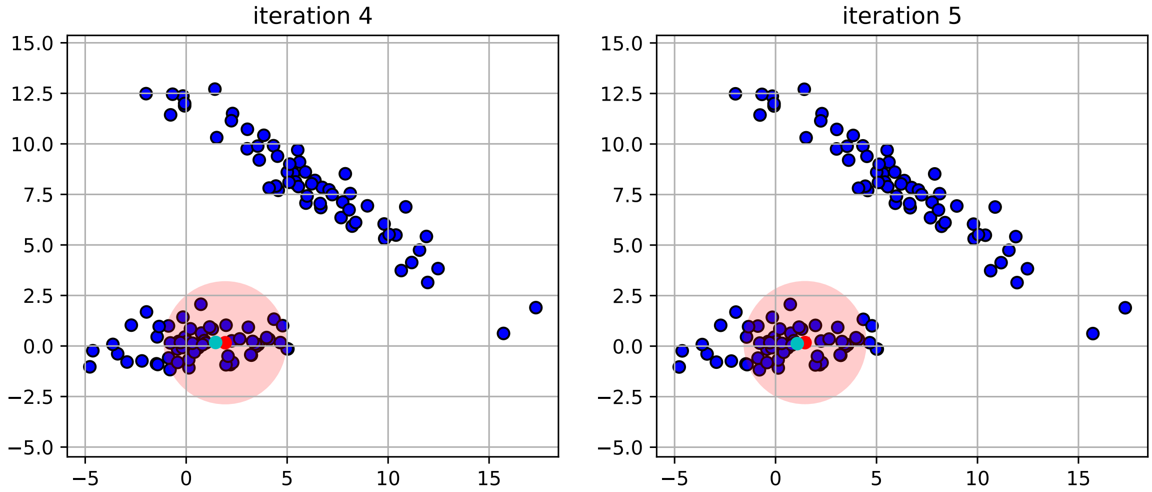
In [6]:
```
efigs[1]
```

Out[6]:



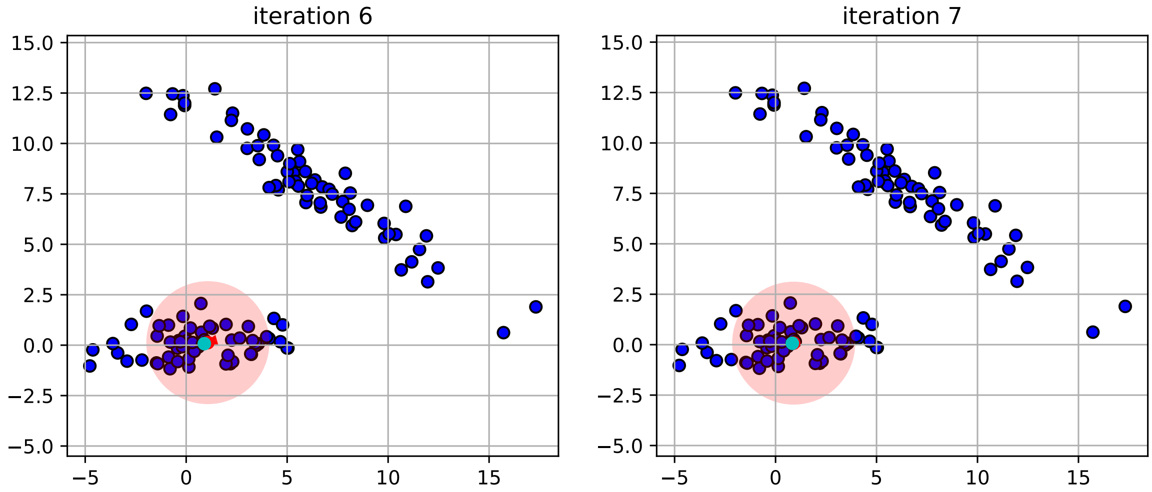
In [7]:
```
efigs[2]
```

Out[7]:



In [8]:
```
efigs[3]
```

Out[8]:



Getting the clusters

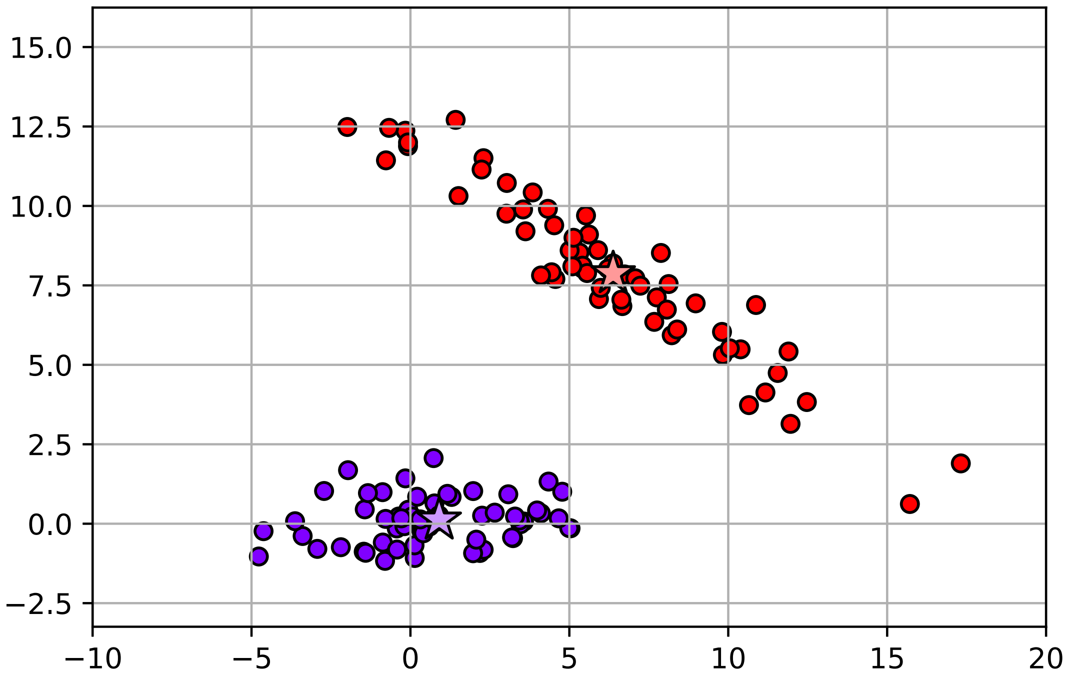
- Run the mean-shift algorithm for many initial points $\{x_i\}$.
  - the set of converged points contain the cluster centers.
    - need to remove the duplicate centers.
  - data points that converge to the same center belong to the same cluster.
  - different initializations can run in parallel (`n_jobs`)

In [9]:
```
# bin_seeding=True -- coarsely uses data points as initial points
ms = cluster.MeanShift(bandwidth=5, bin_seeding=True, n_jobs=-1)
Y = ms.fit_predict(X)

cc = ms.cluster_centers_  # cluster centers

plot_clusters(ms, axbox, X, Y, rbow, rbow2)
```

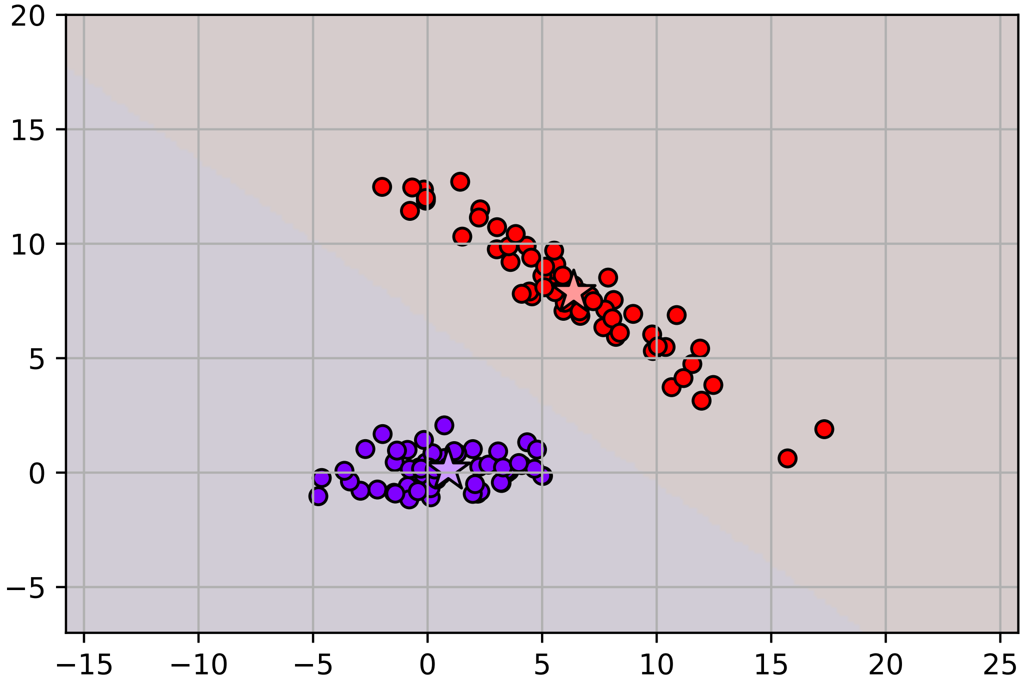
Out[9]: (2,)



- Cluster partitions
  - assign point based on convergence to same cluster center

In [10]:
```
plot_clusters(ms, axbox, X, Y, rbow, rbow2, showregions=True)
```

Out[10]: (2,)

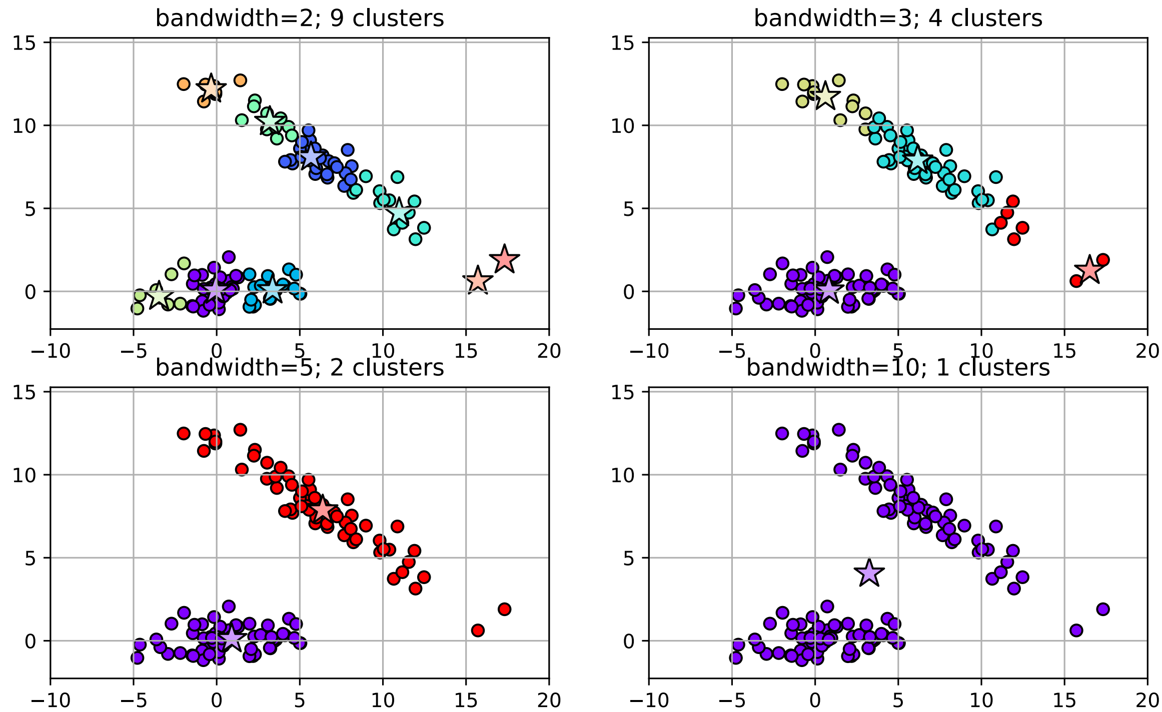


Number of clusters

- Number of clusters is implicitly controlled by the bandwidth (radius of the nearest-neighbors)
  - larger bandwidth creates less clusters
    - focuses on global large groups
  - smaller bandwidth creates more clusters
    - focuses on local groups.

In [12]:
```
msfig
```

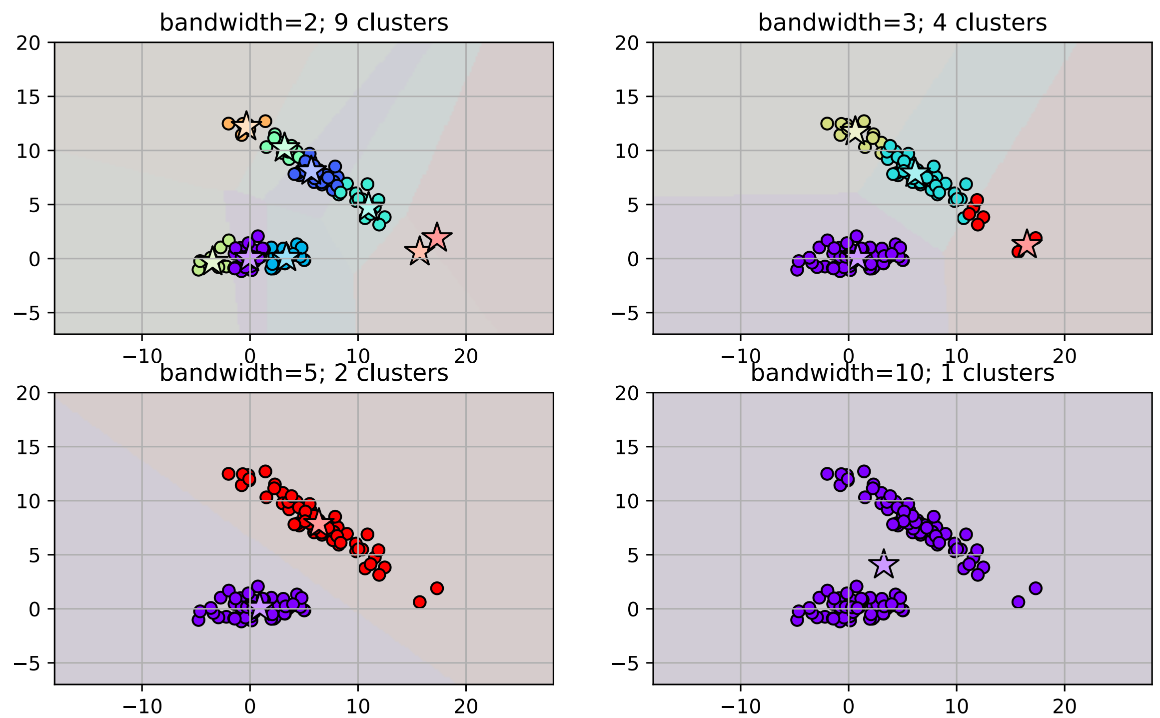
Out[12]:



- Cluster partitions
  - assign points based on convergece to same cluster center.

In [14]:
```
msfig
```

Out[14]:



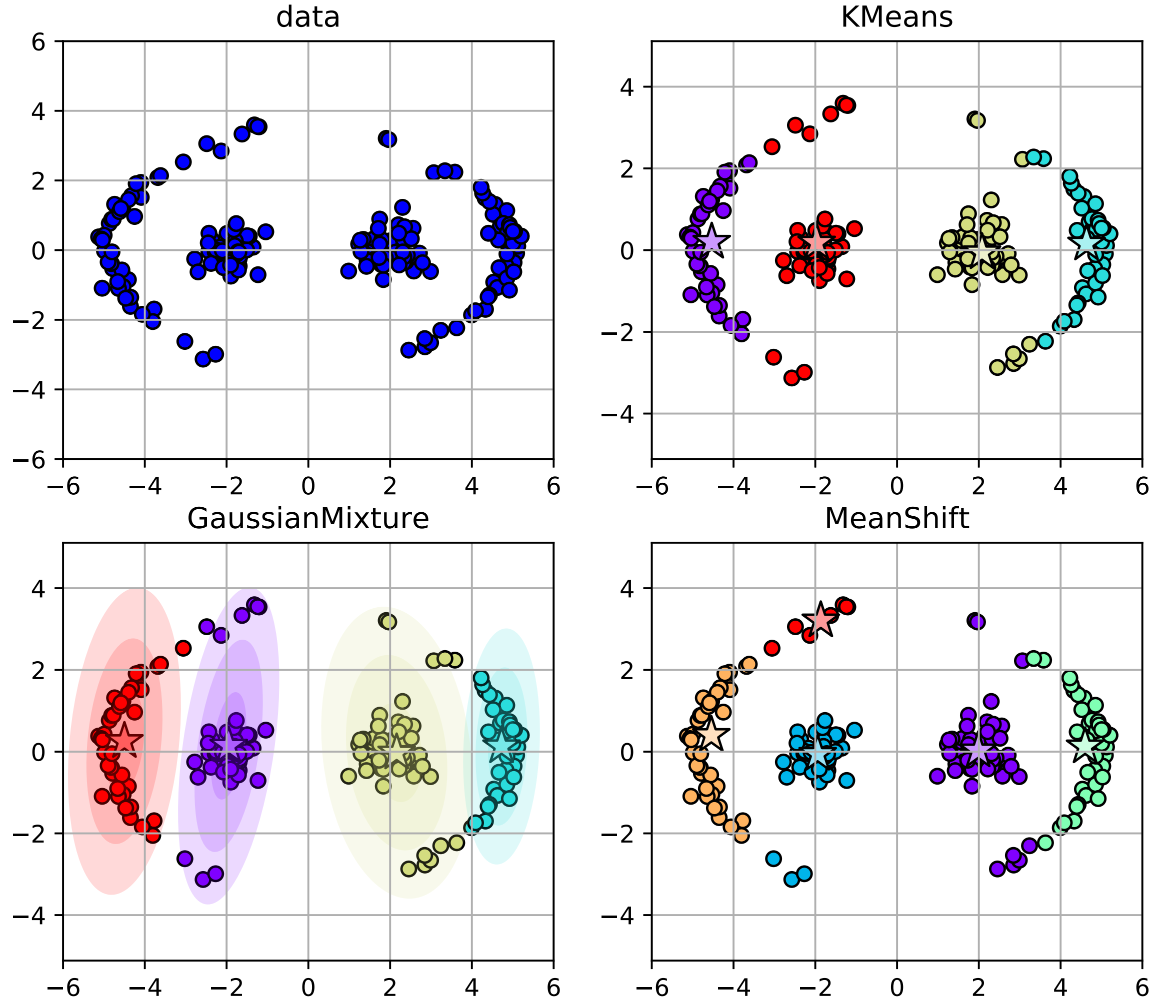
Non-compact clusters

- K-means, GMM, and Mean-Shift assume that all clusters are compact.
  - i.e., circles or ellipses
- What about clusters of other shapes?
  - e.g., clusters not defined by compact distance to a "center"

In [16]:

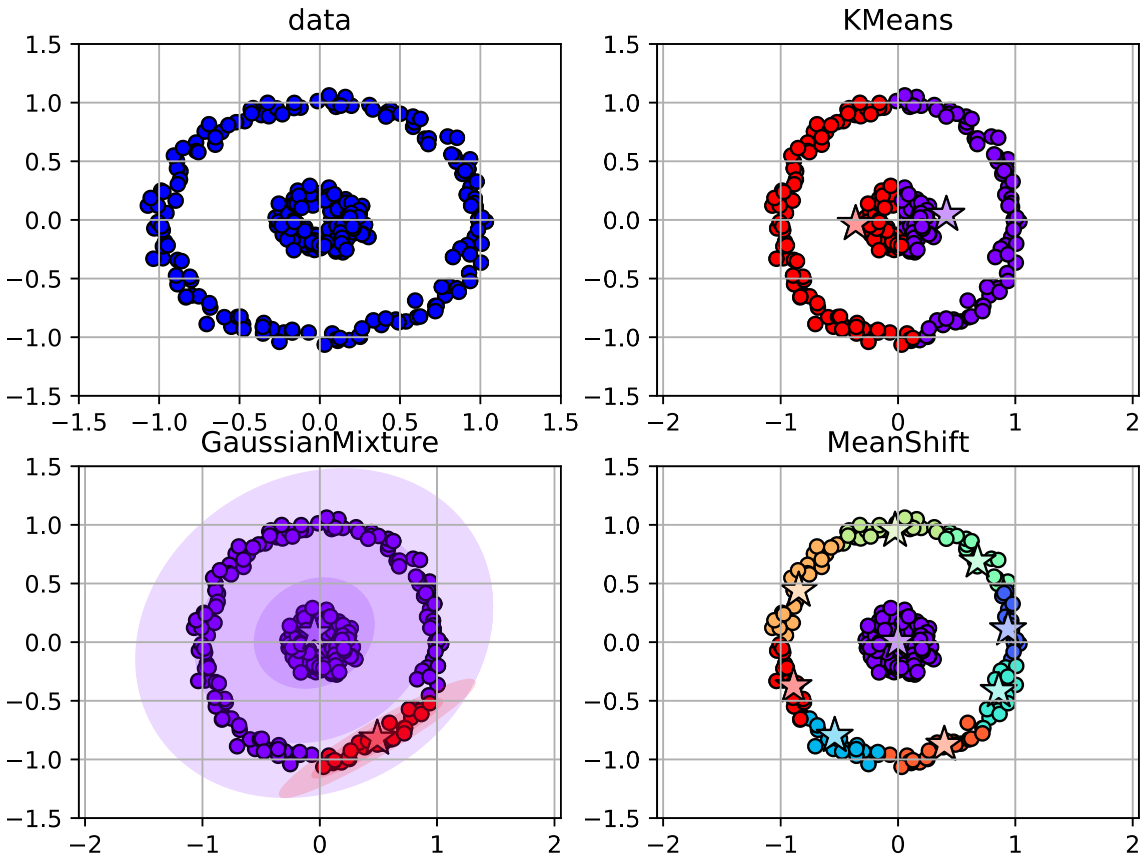
```
tiefig
```

Out[16]:



```
In [18]: circfig
```

Out[18]:

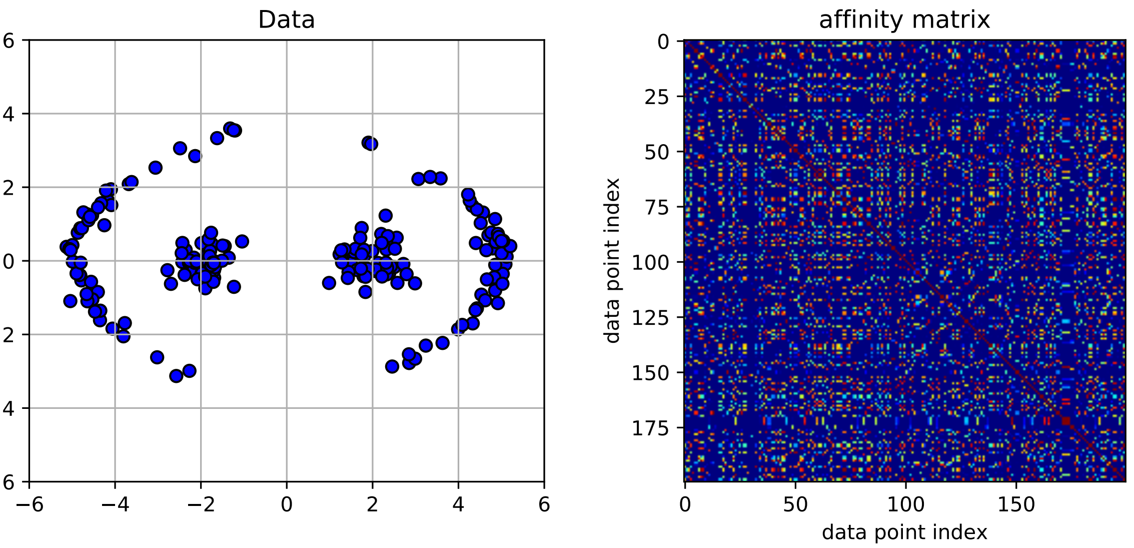


Spectral Clustering

- Estimate the clusters using the pair-wise affinity between points.
- Affinity (similarity) between points
  - kernel function: $k(\mathbf{x}_i, \mathbf{x}_j)$ -- RBF kernel
  - number of nearest neighbors within a radius (bandwidth)

```
In [20]: afig
```

Out[20]:

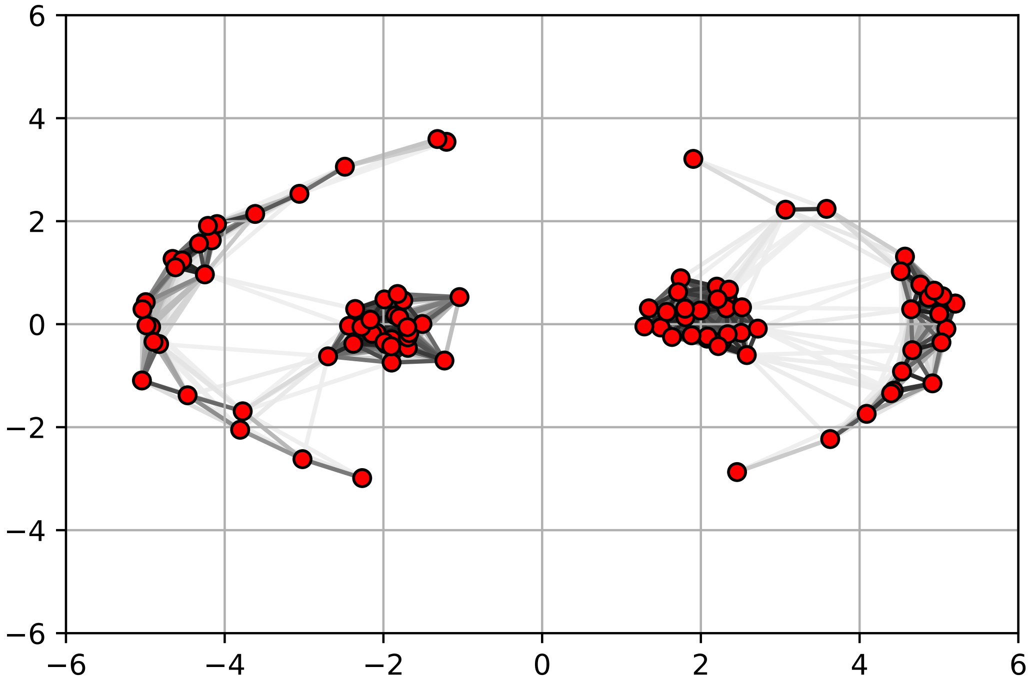


Spectral Clustering

- **Idea:** clustering with a graph formulation
  - each data point is a node in a graph
  - edge weight between two nodes is the affinity $k(\mathbf{x}_i, \mathbf{x}_j)$
    - (darker colors indicate stronger weights)

In [22]:
```
graphfig
```

Out[22]:

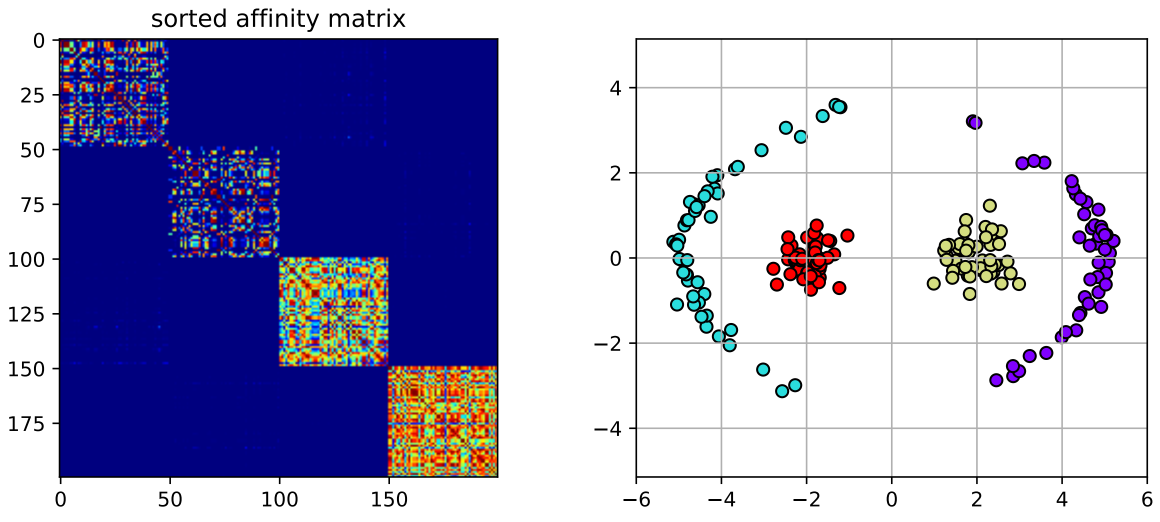


- **Goal:** cut the graph into clusters such that weights of cut edges is small compared to the total edge weight within each cluster.
  - find "blocks" of high affinity in the affinity matrix.

In [24]:
```
# spectral clustering
# rbf affinity
sc = cluster.SpectralClustering(n_clusters=4, affinity='rbf',
                                gamma=1.0, assign_labels='discretize', n_jobs=-1
)
Y = sc.fit_predict(X)
```

In [26]:
```
scfig
```

Out[26]:

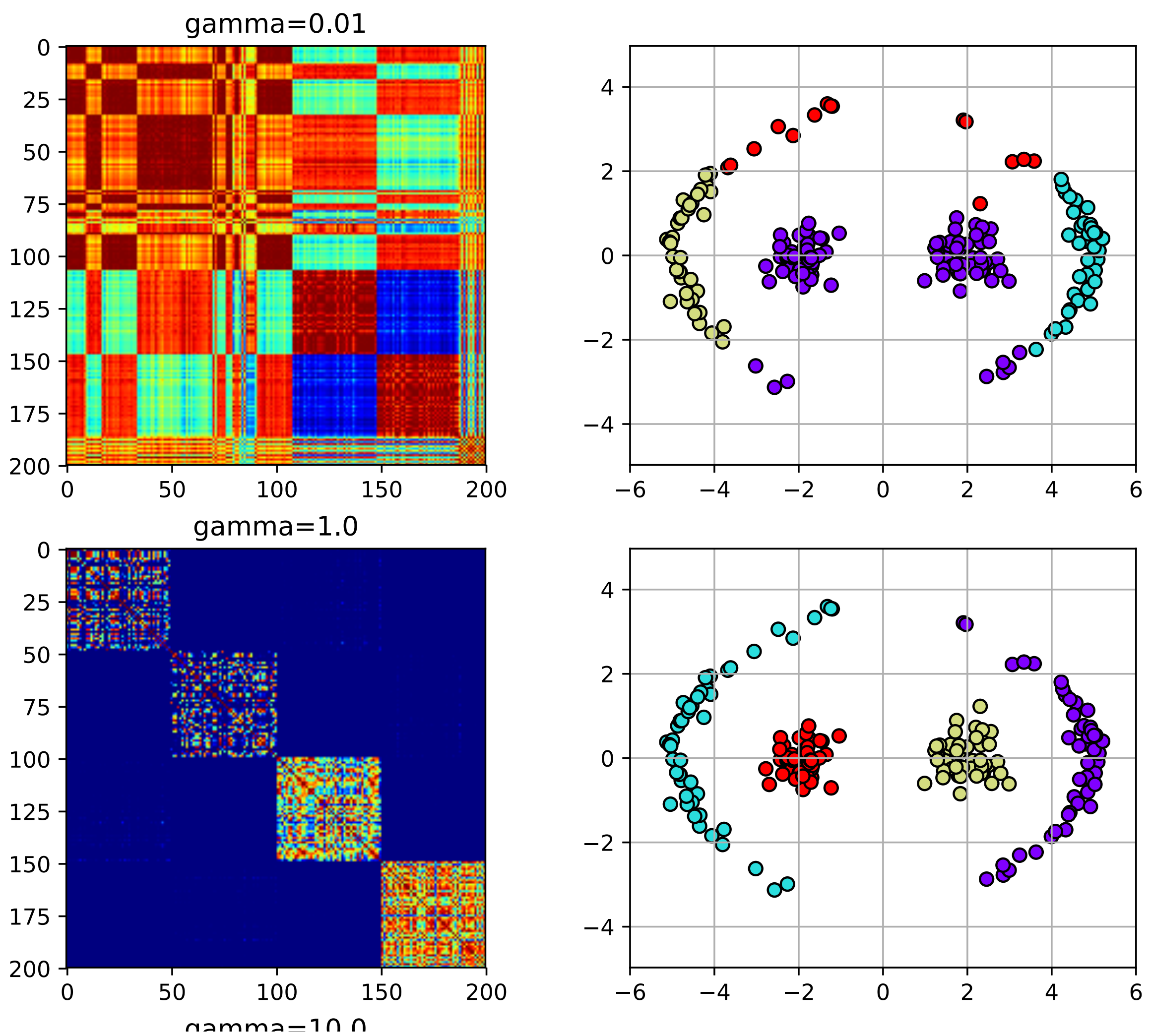


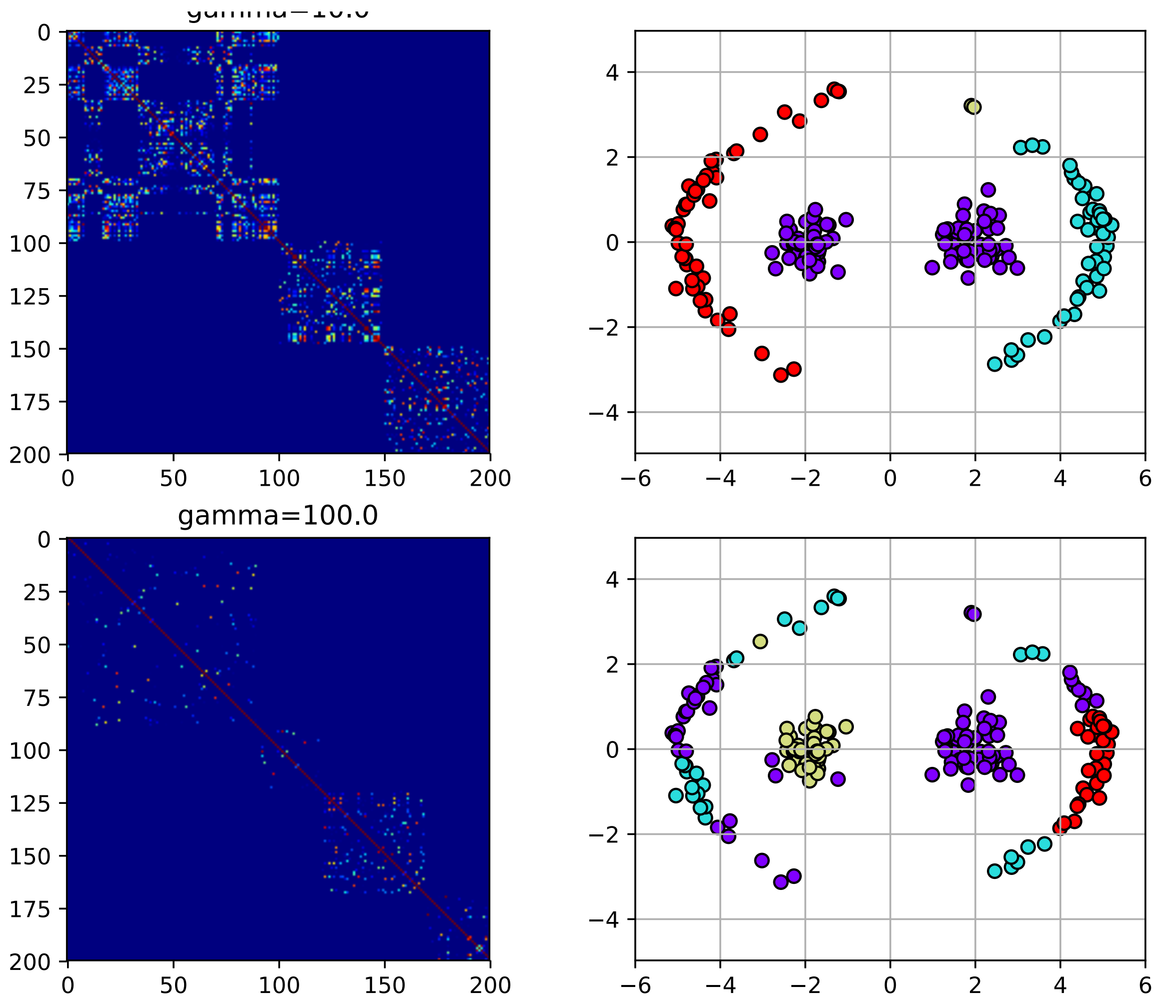
Sensitivity to gamma

- gamma controls which structures are important
    - small gamma - far away points are still considered similar
    - large gamma - close points are not considered similar

In [28]:
```
scfig2
```

Out[28]:



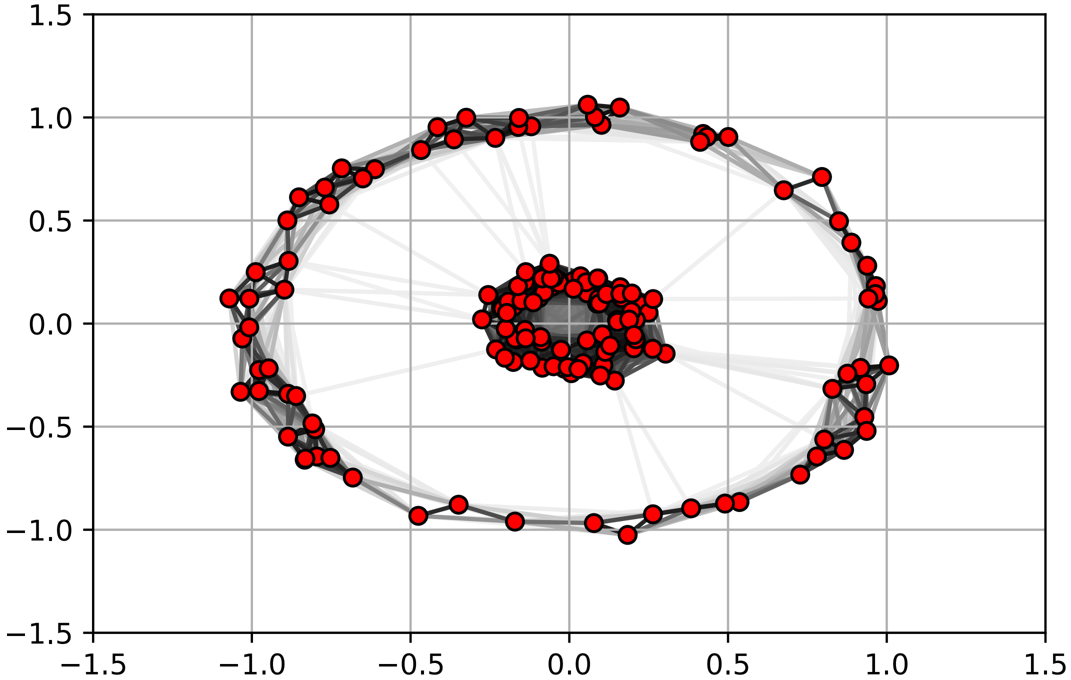


Another Example

In [30]:

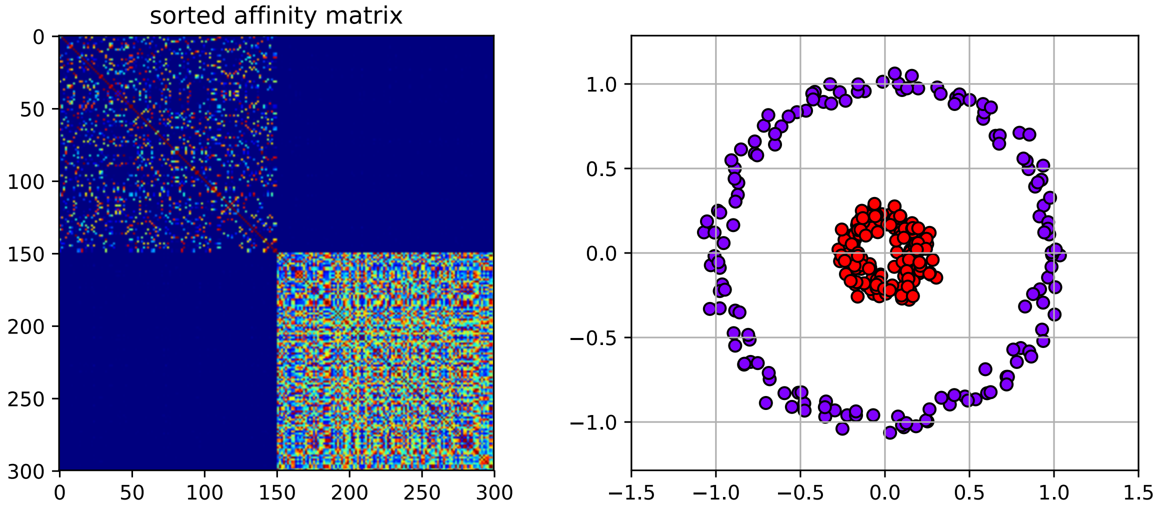
```
graphfig2
```

Out[30]:



In [32]:
```
scfig
```

Out[32]:

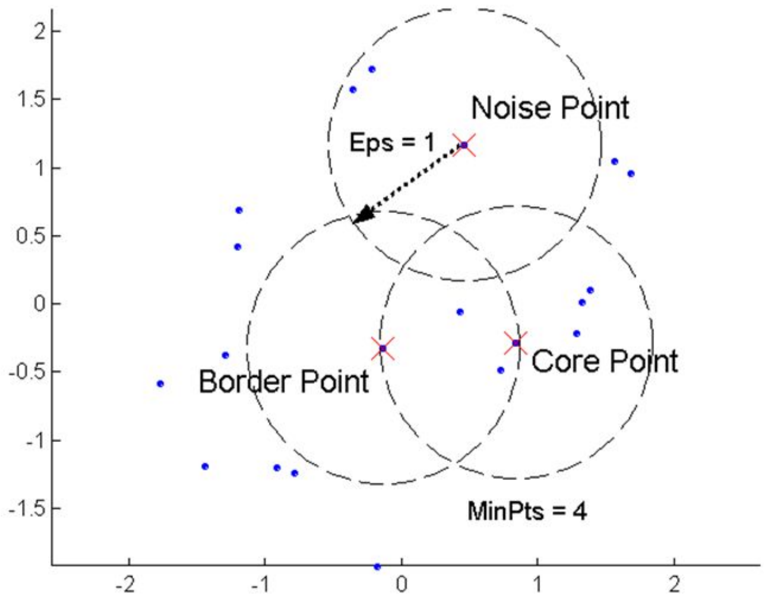


DBSCAN

- "Density-Based Spatial Clustering of Applications with Noise"
  - Assumption: clusters are regions of high density of points separated by areas of low density.
  - Algorithm Idea:
    - Find a *core* point of high density.
    - Recursively label the neighbors as core points.
    - Neighbors that are not core samples are called *boundary* or *non-core* points.
    - Points that are not boundary and not core points are *outliers*.

- Define two parameters:
  - `eps`: the maximum distance to be considered a neighbor.
  - `min_samples`: the minimum number of neighbors (including point itself) to be considered a core sample.

DBSCAN: Core, Border, and Noise Points



Ch. Eick: Introduction to Hierarchical Clustering and DBSCAN

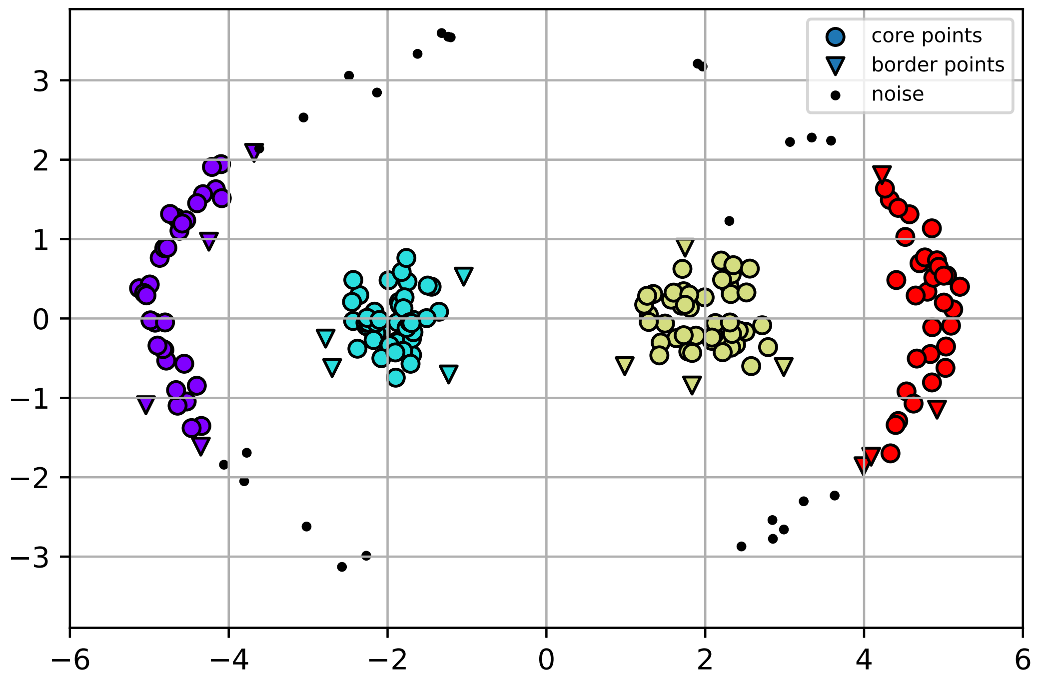
In [34]:
```python
# eps = the max distance to be considered a neighbor
# min_samples = min number of neighbors to be a core sample
dbs = cluster.DBSCAN(eps=0.5, min_samples=5, n_jobs=-1)
Y = dbs.fit_predict(X)

# labels: -1 means outlier
print(Y)

# indices for core samples
print(dbs.core_sample_indices_)
```

```
[-1  0  0  0  0  0  0  0 -1  0  0  0 -1 -1  0  0 -1  0  0  0  0  0  0 -1
  0 -1  0  0 -1 -1 -1  0  0  0 -1  0  0  0  0  0 -1  0  0  0 -1  0  0  0
  0 -1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
  1  1  1  1  2  2  2  2  2 -1  2  2  2  2  2  2  2  2  2  2  2  2  2  2
  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
  2  2  2  2  2  2  3  3  3  3 -1  3 -1  3  3  3  3  3  3  3 -1 -1  3  3
  3  3  3 -1  3  3  3  3  3  3 -1  3  3  3  3 -1  3  3  3  3  3 -1  3  3
 -1  3  3 -1  3 -1  3  3]
[  1   2   3   4   6   7   9  10  11  14  15  19  20  21  22  24  26  27
  31  32  33  35  36  37  38  39  41  43  45  46  47  48  50  51  52  53
  54  55  56  57  58  59  60  61  62  63  64  65  66  68  69  70  71  72
  73  75  76  77  78  79  80  81  82  83  84  85  86  87  89  90  91  92
  93  94  95  96  97  99 100 101 102 103 104 106 107 108 109 110 112 113
 114 115 116 117 118 119 120 121 122 123 124 125 126 129 130 131 132 133
 134 135 136 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152
 155 157 158 159 160 161 162 163 166 167 168 169 170 172 173 174 175 176
 177 179 180 182 184 185 187 188 190 191 193 196 198 199]
```

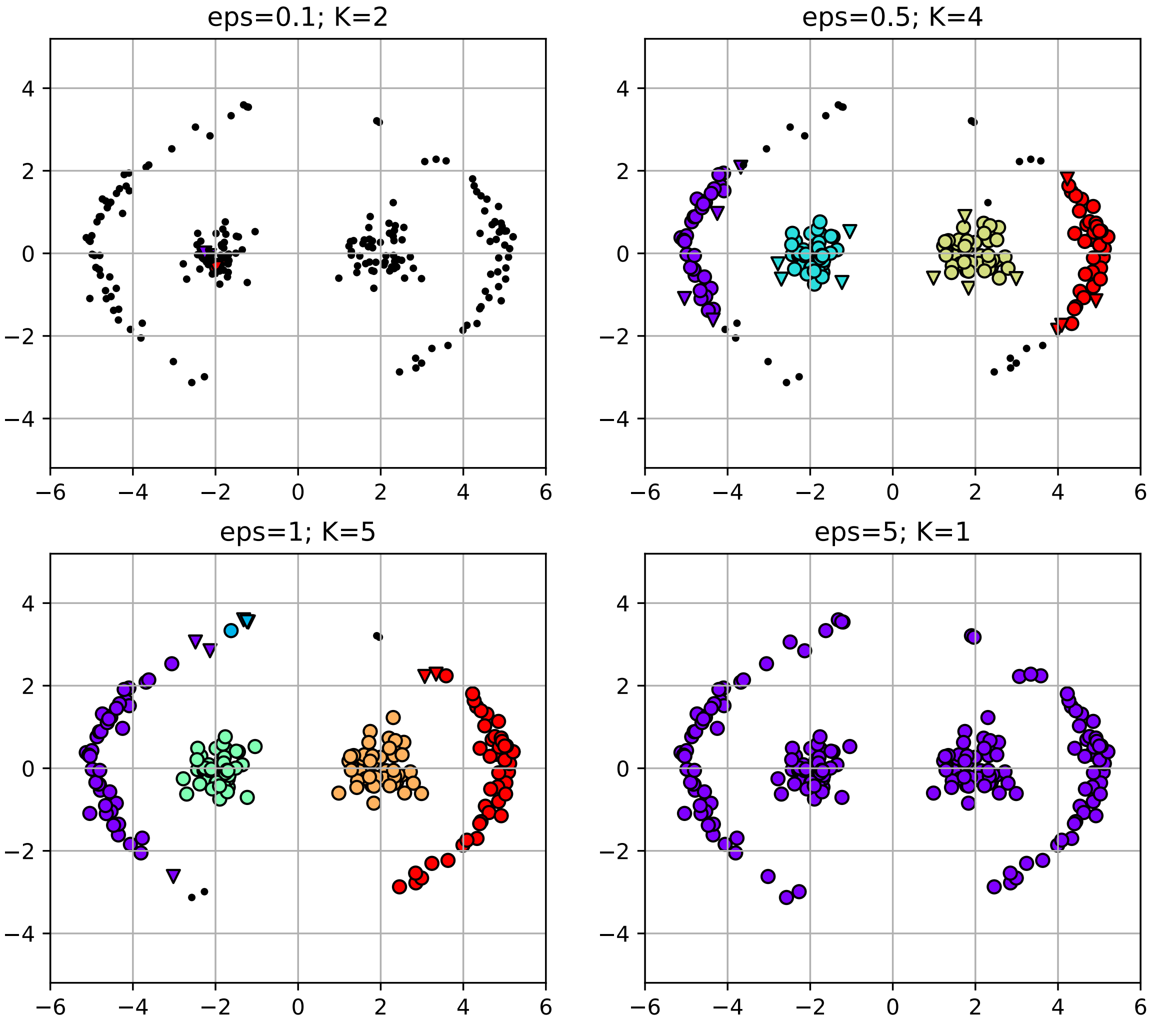
In [35]:
```python
plot_clusters(dbs, axbox, X, Y, rbow, rbow2)
plt.legend(fontsize=7);
```



- Effect of `eps`
    - smaller `eps` - high density required to make a single cluster
    - larger `eps` - encourages collapsing of clusters
    - `min_samples` is 5.

In [37]:
```
dbfig
```

Out[37]:

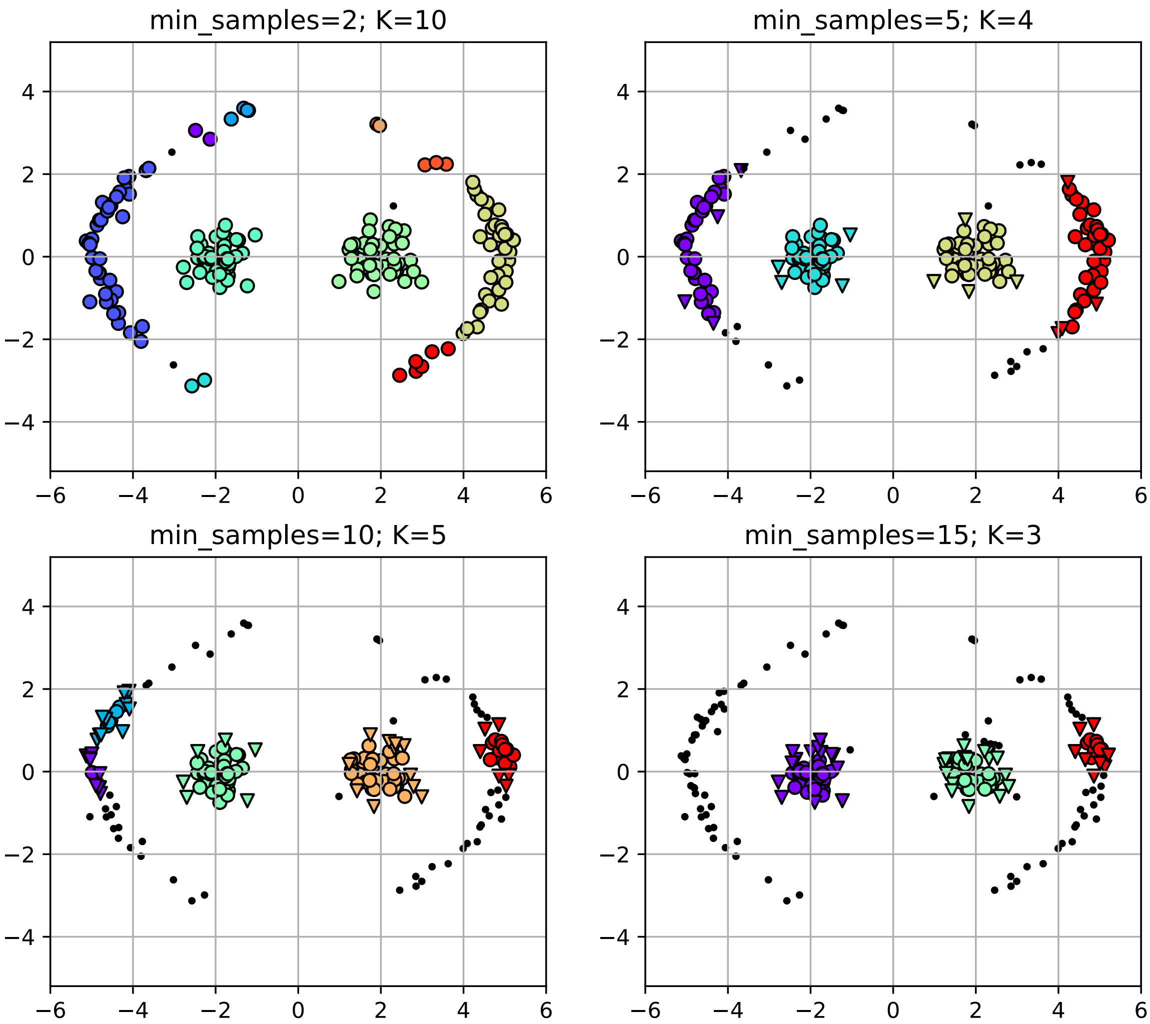


- Effect of `min_samples`
    - smaller `min_samples` - encourages forming small clusters
    - larger `min_samples` - forms clusters only in very high-density regions.
    - `eps` is 0.5 here.

In [39]:

```
dbfig
```

Out[39]:



Clustering Summary

- **Goal:** given set of input vectors $\{\mathbf{x}_i\}_{i=1}^n$, with $\mathbf{x}_i \in \mathbb{R}^d$, group similar x_i together into clusters.
  - estimate a cluster center, which represents the data points in that cluster.
  - predict the cluster for a new data point.

Name	**Cluster Shape**	**Principle**	**Advantages**	**Disadvantages**
K-Means	circular	minimize distance to cluster center	- scalable (MiniBatchKMeans)	- sensitive to initialization; could get bad solutions due to local minima. - need to choose K.
Gaussian Mixture Model	elliptical	maximum likelihood	- elliptical cluster shapes.	- sensitive to initialization; could get bad solutions due to local minima. - need to choose K.
Dirichlet Process GMM	elliptical	maximum likelihood	- automatically selects K via concentration parameter.	- can be slow. - sensitive to initialization; could get bad solutions due to local minima.
Mean-Shift	concentrated compact	move towards local mean	- automatically selects K via bandwidth parameter.	- can be slow.
Spectral clustering	irregular shapes	graph-based	- can handle clusters of any shape, as long as connected.	- need to choose K. - cannot assign novel points to a cluster. - can be slow (kernel matrix)
DBSCAN	irregular shapes	density-based	- can handle clusters of any shape, as densely sampled. - can detect outliers	- sensitive to parameters - cannot assign novel points to a cluster.

Other Things

- *Feature normalization*
  - feature normalization is typically required clustering.
  - e.g., algorithms based on Euclidean distance (Kmeans, Mean-Shift, Spectral Clustering)