CS4487 - Machine Learning

Lecture 6a - 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

Supervised Learning

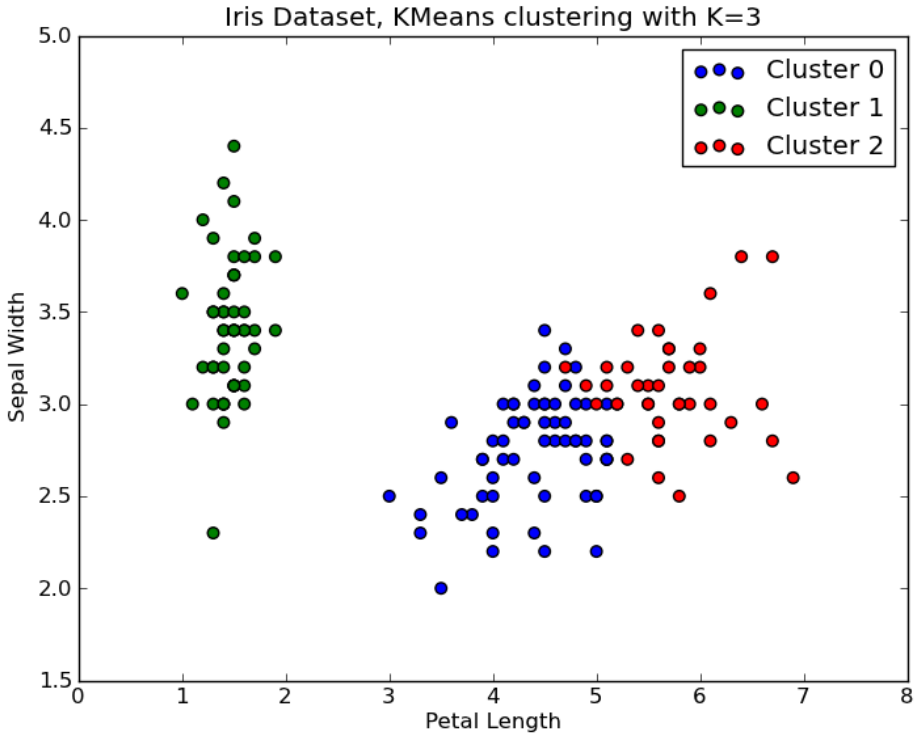
- *Supervised learning* considers input-output pairs $(\mathbf{x}, y)$
  - learn a mapping from input to output.
  - *classification*: output $y \in \pm 1$
  - *regression*: output $y \in \mathbb{R}$
- "Supervised" here means that the algorithm is learning the mapping that we want.

Unsupervised Learning

- Unsupervised learning only considers the input data $\mathbf{x}$.
  - There are no output values.
- **Goal:** Try to discover inherent properties in the data.
  - Clustering
  - Dimensionality Reduction
  - Manifold Embedding

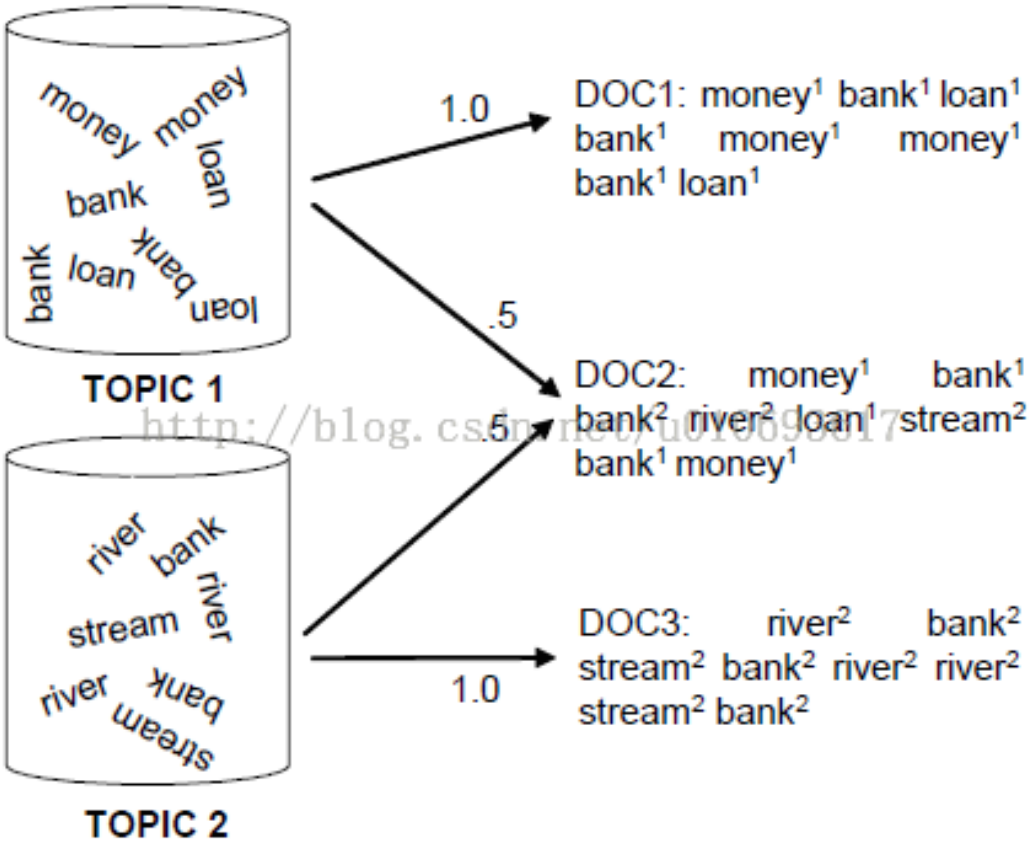
Clustering

- Find clusters of similar items in the data.
- Find a representative item that can represent all items in the cluster.
- **For example:** grouping iris flowers by their measurements.
  - Features are sepal width and petal length.



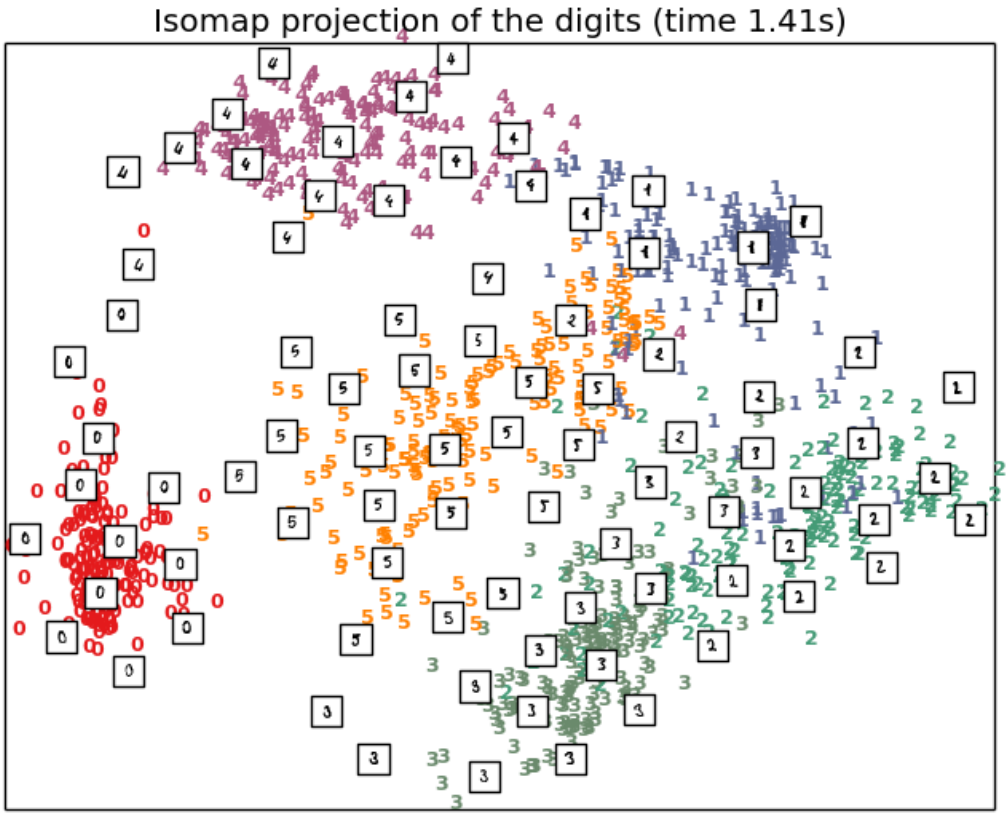
Dimensionality Reduction

- Transform high-dimensional vectors into low-dimensional vectors.
  - Dimensions in the low-dim data may have semantic meaning.
- **For example:** document analysis
  - high-dim: bag-of-word vectors of documents
  - low-dim: each dimension represents similarity to a topic.



Manifold Embedding

- Project high-dimensional vectors into 2- or 3-dimensional space for visualization.
  - Points in the low-dim space have similar pair-wise distances as in the high-dim space.
- **For example:** visualize a collection of hand-written digits (images).

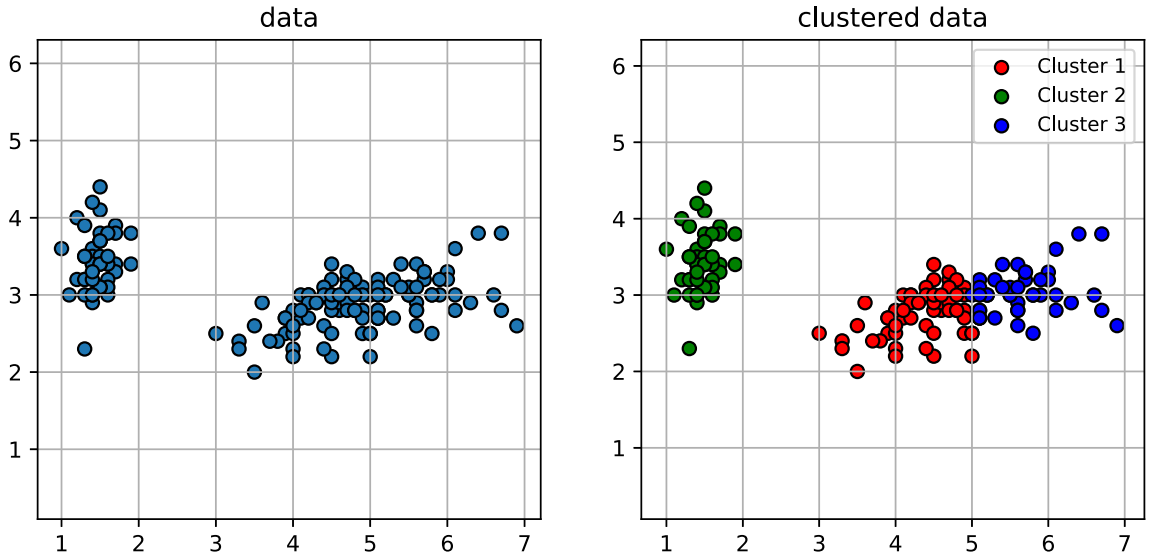


Clustering

- Each data point is a vector $\mathbf{x} \in \mathbb{R}^d$.
- Data is set of vectors $\{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$
- **Goal:** group similar data together.
  - groups are also called clusters.
  - each data point is assigned with a cluster index ($y \in \{1, \cdots, K\}$)
    - K is the number of clusters.

In [3]:
```
clusterfig
```

Out[3]:

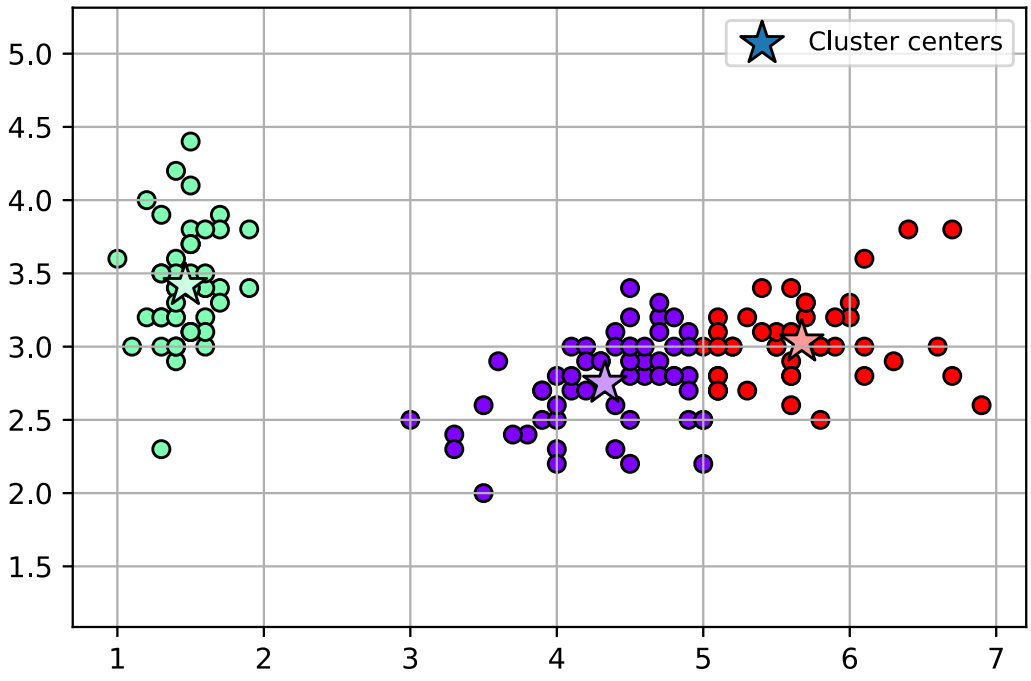


K-Means Clustering

- **Idea:**
  - there are K clusters.
  - each cluster is represented by a *cluster center*.
    - $\mathbf{c}_j \in \mathbf{R}^d, j \in \{1, \cdots, K\}$
  - assign each data point to the closest cluster center.
    - according to Euclidean distance: $\|\mathbf{x}_i - \mathbf{c}_j\|$

In [5]:
```
kmfig
```

Out[5]:



K-means Clustering Objective

- *How to pick the cluster centers?*
  - Assume there are K clusters
  - Pick the cluster centers that minimize the squared distance to all its cluster members.

$$\min_{\mathbf{c}_1,\cdots,\mathbf{c}_K} \sum_{i=1}^{n} ||\mathbf{x}_i - \mathbf{c}_{z_i}||^2$$

  - where z_i is the index of the closest cluster center to $\mathbf{x}_i$.
    - $z_i = \text{argmin}_{j=\{1,\cdots,K\}} ||\mathbf{x}_i - \mathbf{c}_j||$
    - i.e., the assignment of point $\mathbf{x}_i$ to its closest cluster.

- Solution:
  - if the assignments $\{z_i\}$ are known...
    - let C_j be the set of points assigned to cluster j
      - $C_j = \{\mathbf{x}_i | z_i = j\}$
    - cluster center is the mean of the points in the cluster
      - $\mathbf{c}_j = \frac{1}{|C_j|} \sum_{\mathbf{x}_i \in C_j} \mathbf{x}_i$

Chicken and Egg Problem

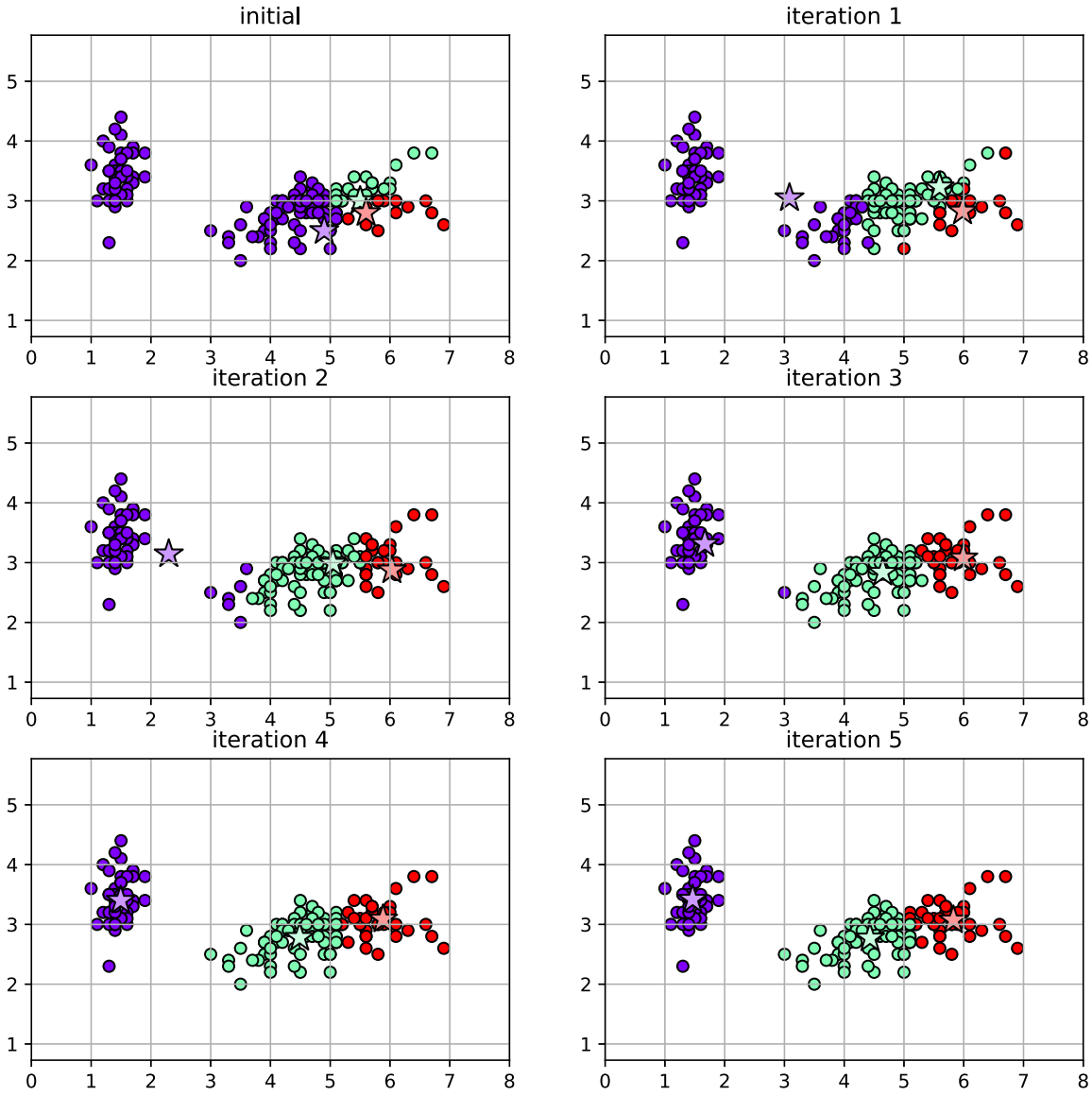
- Cluster assignment of each point depends on the cluster centers.
- Location of cluster center depends on which points are assigned to it.

K-means Algorithm

- Pick initial cluster centers
- Repeat:
  - 1) calculate assignment z_i for each point $\mathbf{x}_i$: closest cluster center using Euclidean distance.
  - 2) calculate cluster center $\mathbf{c}_j$ as average of points assigned to cluster j.
- This procedure will converge eventually.

In [8]:
```
kmifig
```

Out[8]:

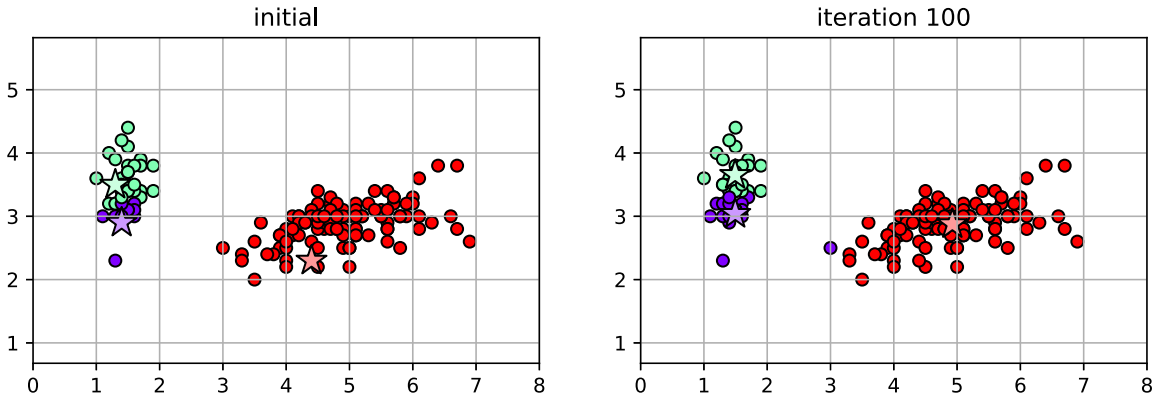


Important Note

- The final result depends on the initial cluster centers!
  - Some bad initializations will yield poor clustering results!
  - (Technically, there are multiple local minimums in the objective function)

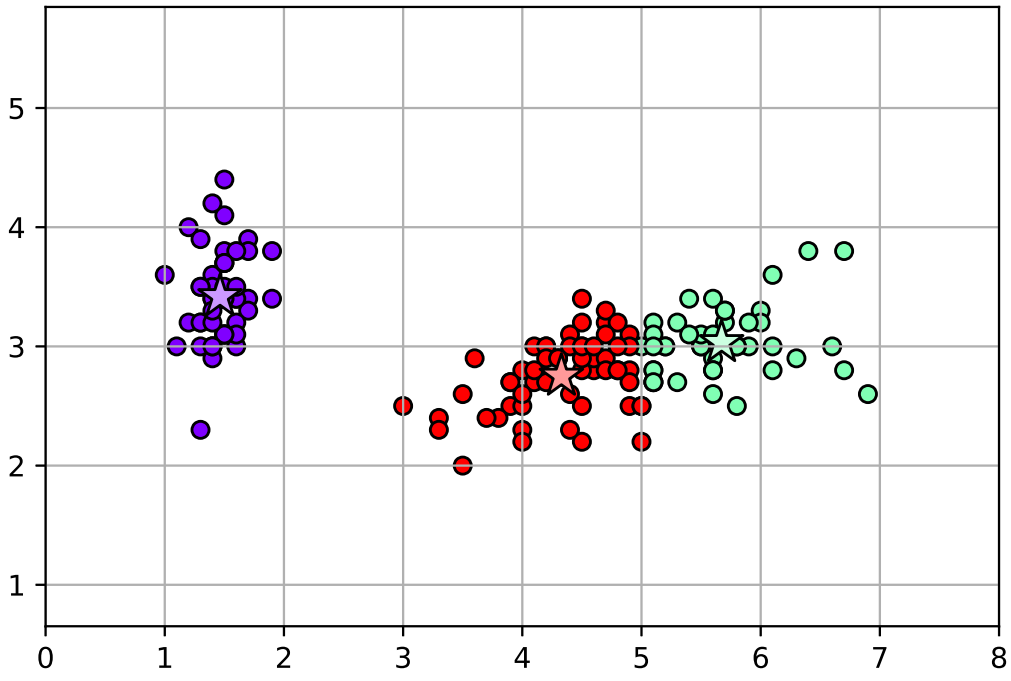
In [10]:
```
kmbadfig
```

Out[10]:



- **Solution:**
  - Try several times using different initializations.
  - Pick the answer with lowest objective score.
- In scikit-learn,
  - automatically uses multiple random initializations.
  - also uses a smart initialization method called "k-means++"
  - can run initialization runs in parallel (`n_jobs`)

In [11]:
```
# K-Means with 3 clusters
# (automatically does 10 random initializations)
km = cluster.KMeans(n_clusters=3, random_state=4487, n_jobs=-1)
Yp = km.fit_predict(X)   # cluster data, and return labels

cc = km.cluster_centers_   # the cluster centers
cl = km.labels_            # labels also stored here

plt.figure()
plot_clusters(km, axbox, X, Yp, rbow, rbow2);
```

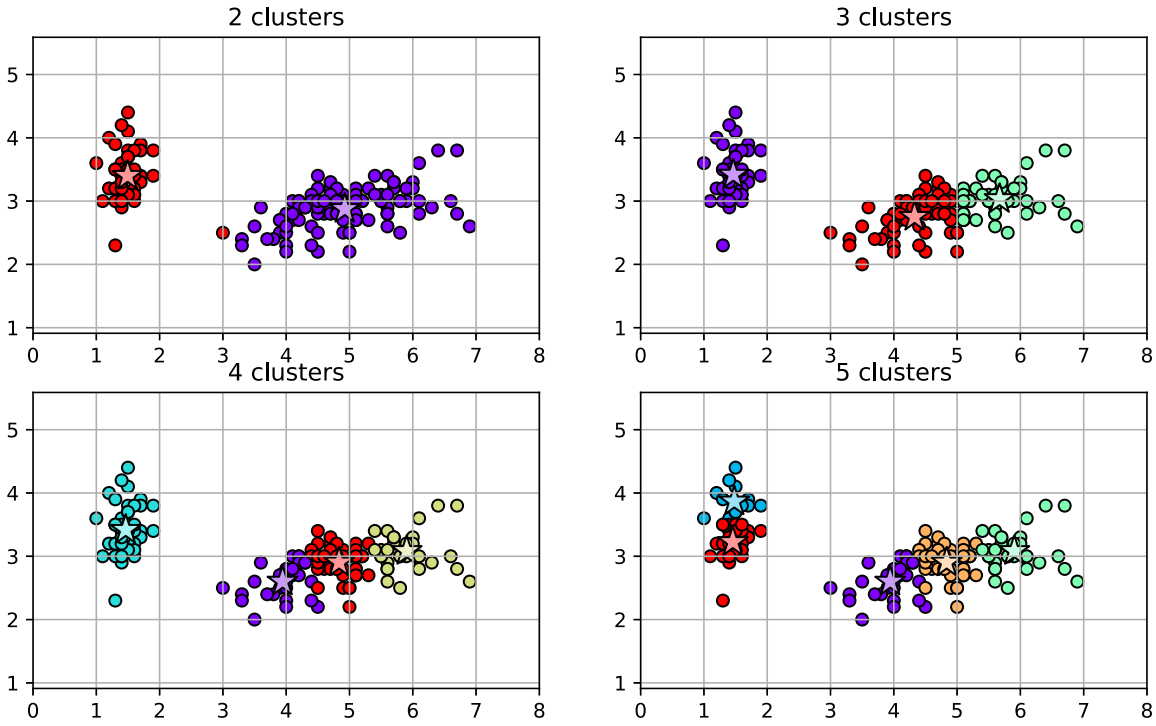


For different K

- We need to choose the appropriate K

In [13]:
```
ksfig
```

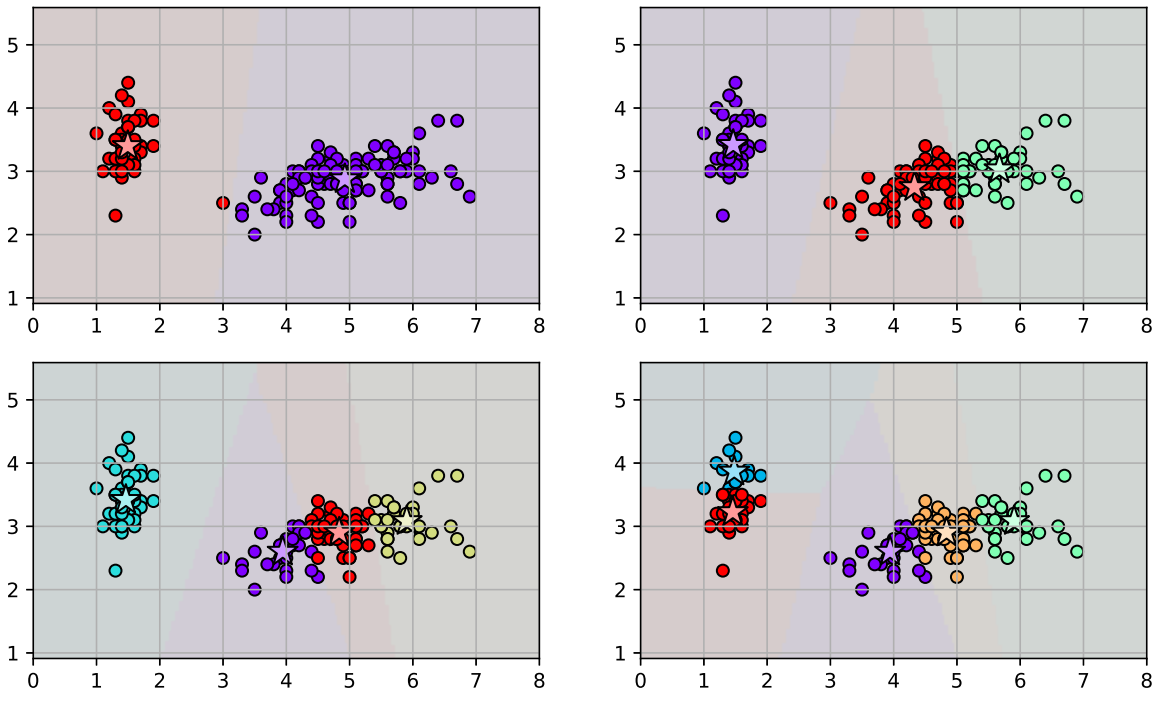
Out[13]:



- K-means splits the input space into different regions belonging to each cluster

In [15]:
```
krfig
```

Out[15]:

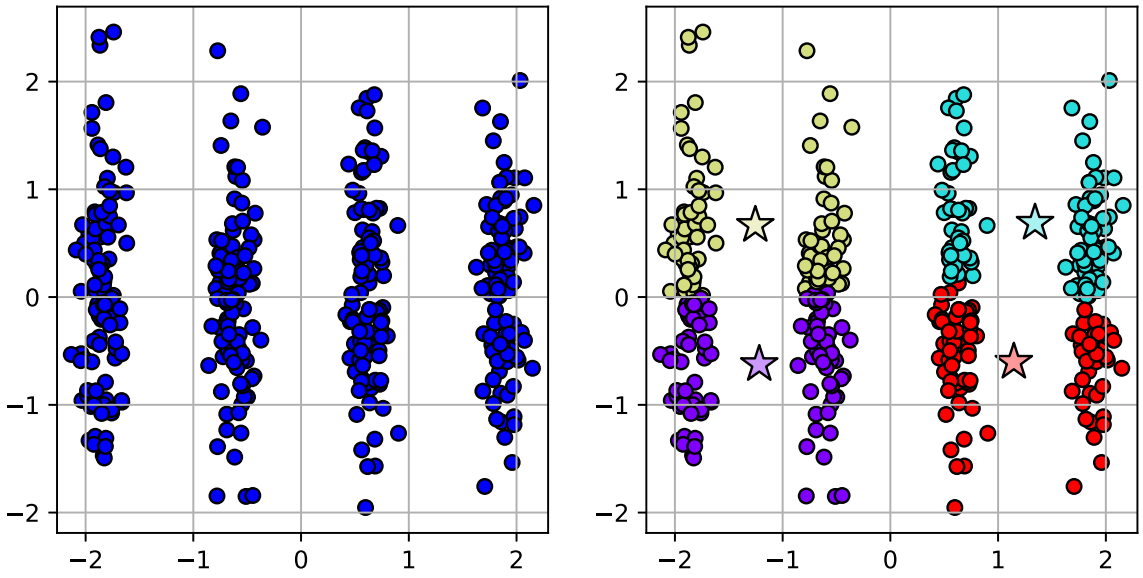


Circular clusters

- One problem with K-means is that it assumes that each cluster has a circular shape.
  - based on Euclidean distance to each center
  - Kmeans cannot handle skewed (elliptical) clusters.

In [17]:
```
efig
```

Out[17]:



Gaussian mixture model (GMM)

- A multivariate Gaussian can model a cluster with an elliptical shape.
  - the ellipse shape is controlled by the covariance matrix of the Gaussian
  - the location of the cluster is controlled by the mean.
- Gaussian mixture model is a weighted sum of Gaussians

$$p(\mathbf{x}) = \sum_{j=1}^{K} \pi_j N(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$$

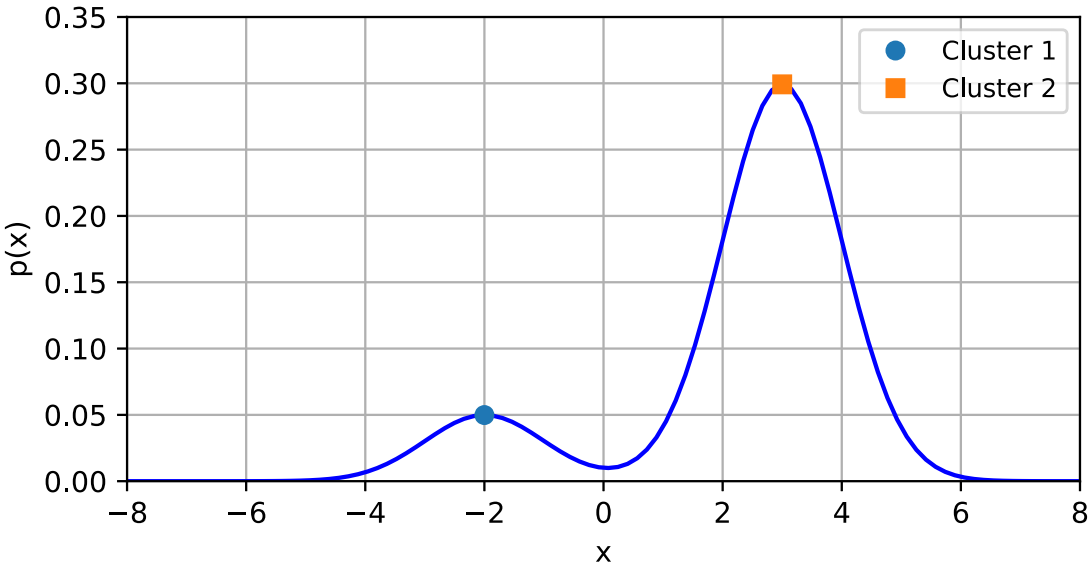
  - Each Gaussian represents one elliptical cluster
    - $\boldsymbol{\mu}_j$ is the mean of the j-th Gaussian. (the location)
    - $\boldsymbol{\Sigma}_j$ is the covariance matrix of the j-th Gaussian. (the ellipse shape)
    - π_j is the prior weight of the j-th Gaussian. (how likely is this cluster)

1-D example of GMM

- each Gaussian is a "mountain"

In [19]:
```
eggmm
```

Out[19]:

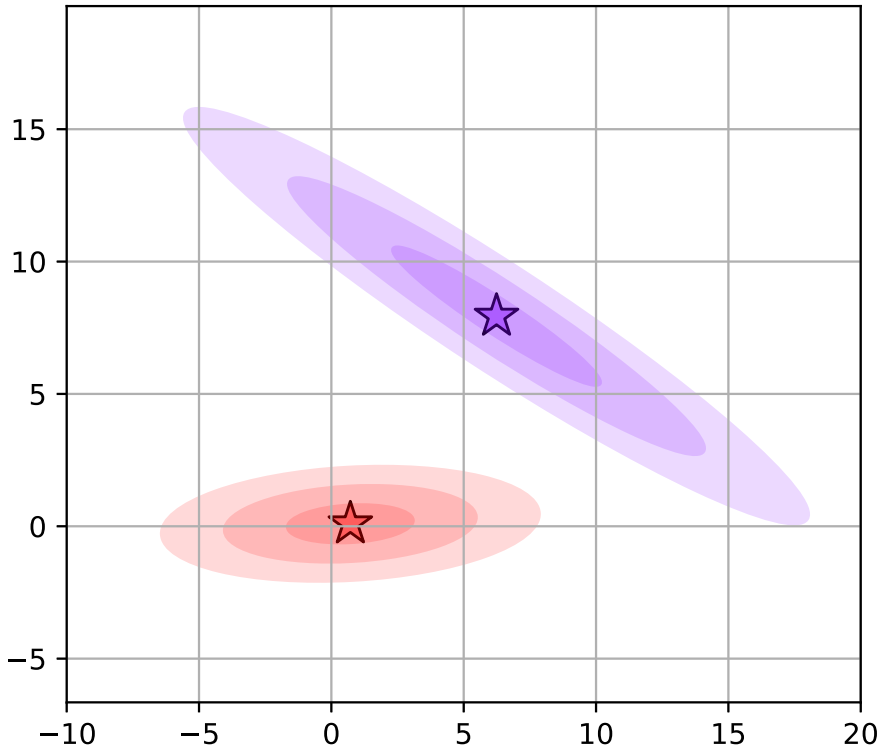


2D example of GMM

- Each Gaussian defines a "mountain"
  - contours are ellipses

In [21]:
```
gmm2fig
```

Out[21]:



Clustering with GMMs

- Using the data, learn a GMM using maximum likelihood estimation:

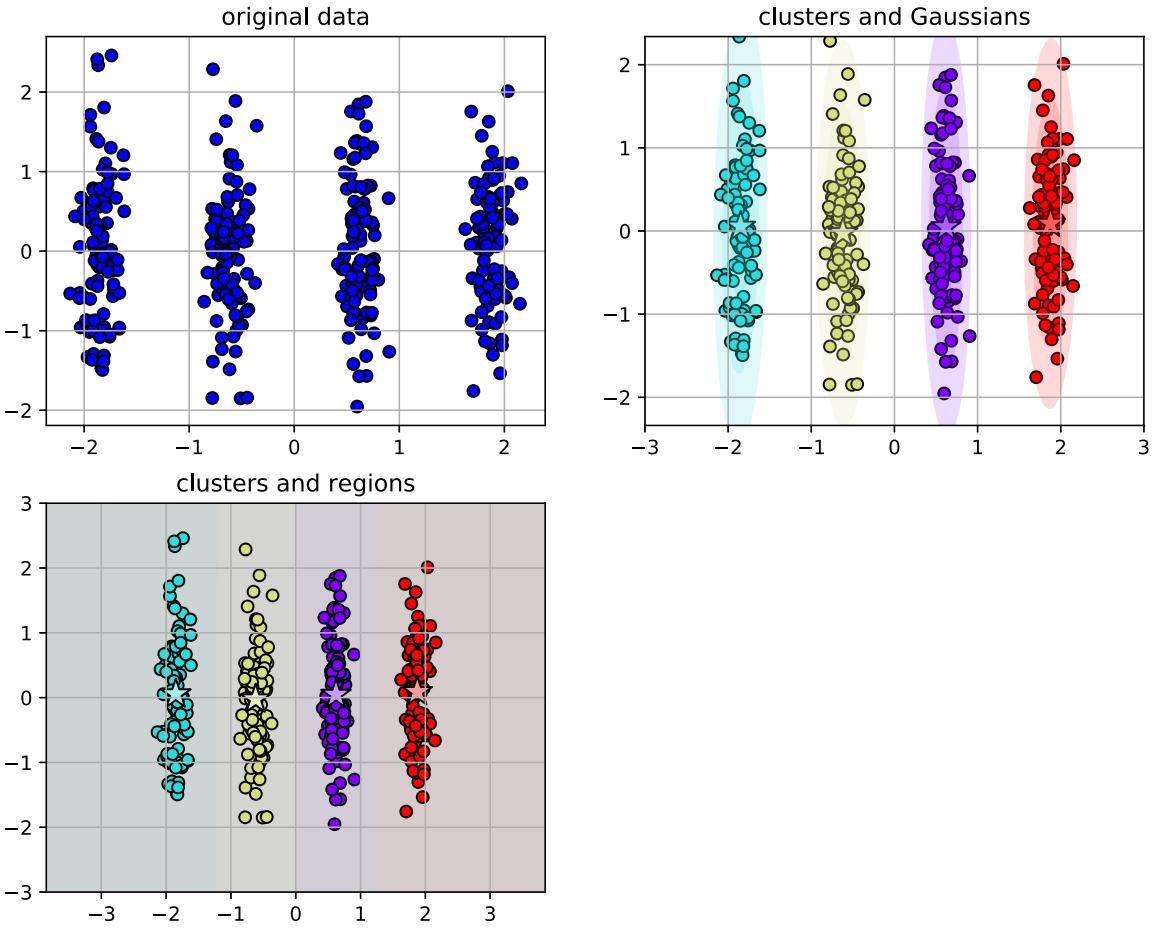
$$\max_{\pi,\boldsymbol{\mu},\boldsymbol{\Sigma}} \sum_{i=1}^{N} \log \sum_{j=1}^{K} \pi_j N(\mathbf{x}_i|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$$

- Results in an algorithm similar to K-means:
  - 1) Calculate cluster membership
    - uses "soft" assignment - a data point can have a fractional contribution to different clusters.
    - contribution of point i to cluster j
      - $z_{ij} = p(z_i = j|\mathbf{x}_i)$
  - 2) Update each Gaussian cluster (mean, covariance, and weight)
    - uses "soft" weighting
      - "soft" count of points in cluster j: $N_j = \sum_{i=1}^{N} z_{ij}$
      - weight: $\pi_j = N_j/N$
      - mean: $\boldsymbol{\mu}_j = \frac{1}{N_j} \sum_{i=1}^{N} z_{ij}\mathbf{x}_i$
      - variance: $\Sigma_j = \frac{1}{N_j} \sum_{i=1}^{N} z_{ij}(\mathbf{x}_i - \boldsymbol{\mu}_j)^2$

In [22]:
```
# fit a GMM
gmm = mixture.GaussianMixture(n_components=4, random_state=4487, n_init=10)

gmm.fit(X)
Y = gmm.predict(X)

cc = gmm.means_     # the cluster centers
```

In [24]:
```
efig
```

Out[24]:



Covariance matrix

- The covariance matrix is a $d \times d$ matrix.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

- For high-dimensional data, it can be very large.
  - requires a lot of data to learn effectively.

- Solution:
  - use *diagonal* covariance matrices (d parameters):

$$\begin{bmatrix} a_{11} & 0 & 0 \\ 0 & a_{22} & 0 \\ 0 & 0 & a_{33} \end{bmatrix}$$

    - Axes of ellipses will be aligned with the axes.
  - use *spherical* covariance matrices (1 parameter)

$$\begin{bmatrix} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & a \end{bmatrix}$$

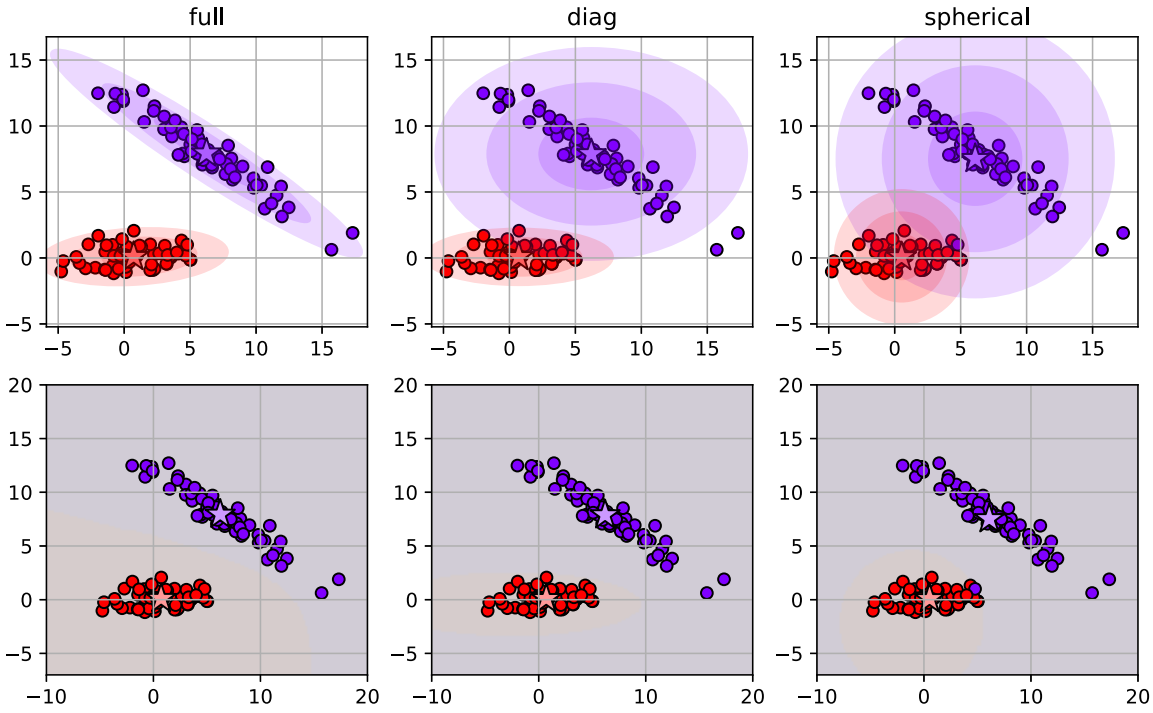
    - Clusters will be circular (similar to K-means)

In [26]:
```
# full covariance (d*d parameters)
gmmf = mixture.GaussianMixture(n_components=2, covariance_type='full',
                               random_state=4487, n_init=10)
gmmf.fit(X)

# diagonal convariance (d parameters)
gmmd = mixture.GaussianMixture(n_components=2, covariance_type='diag',
                               random_state=4487, n_init=10)
gmmd.fit(X)

# spherical covariance (1 parameter)
gmms = mixture.GaussianMixture(n_components=2, covariance_type='spherical',
                               random_state=4487, n_init=10)
gmms.fit(X)
```

Out[26]:
```
GaussianMixture(covariance_type='spherical', init_params='kmeans',
        max_iter=100, means_init=None, n_components=2, n_init=10,
        precisions_init=None, random_state=4487, reg_covar=1e-06,
        tol=0.001, verbose=0, verbose_interval=10, warm_start=False,
        weights_init=None)
```

In [28]:
```
efig
```

Out[28]:



How to select K?

- Clustering results depends on the number of clusters used.
- We don't typically know this information beforehand.

Dirichlet Process GMM

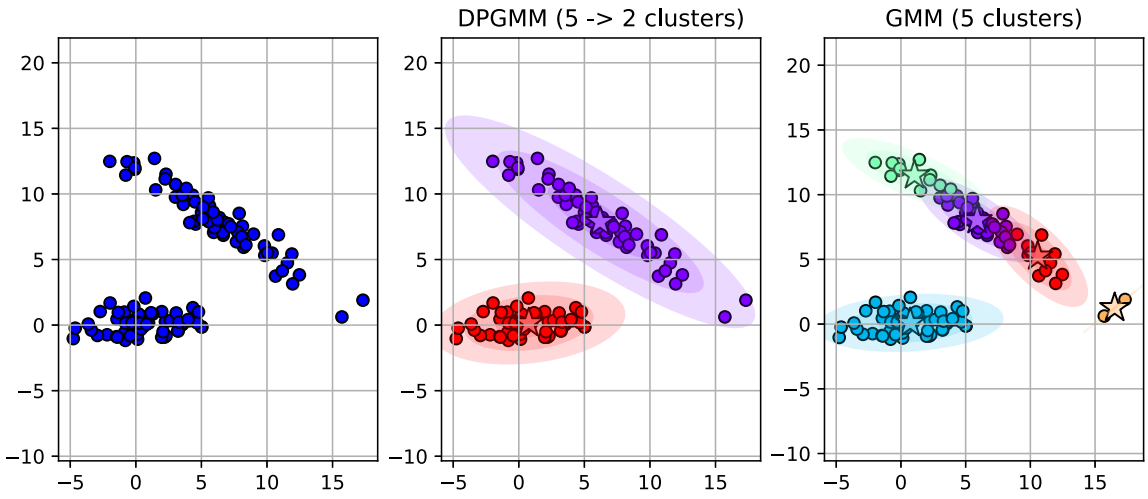
- GMM is extended to automatically select the value of K
  - use a "Dirichlet Process" to model K as a random variable.
- *concentration* parameter α controls the range of K values that are preferred
  - higher values encourage more clusters
  - lower values encourage less clusters
  - expected number of clusers is $\alpha \log N$, where N is the number of points.

In [29]:
```
# alpha = concentration parameter
# n_components = the max number of components to consider
dpgmm = mixture.BayesianGaussianMixture(covariance_type='full',
                    weight_concentration_prior=1,
                    n_components=5, max_iter=100, random_state=4487)
dpgmm.fit(X)
Y = dpgmm.predict(X)
cl = unique(Y)          # find active clusters
newK = len(cl)          # number of clusters
cc = dpgmm.means_[cl]   # get means
```

- DPGMM automatically selects 2 components from 5
  - for comparison, GMM with 5 clusters looks messy

In [31]:
```
dfig
```

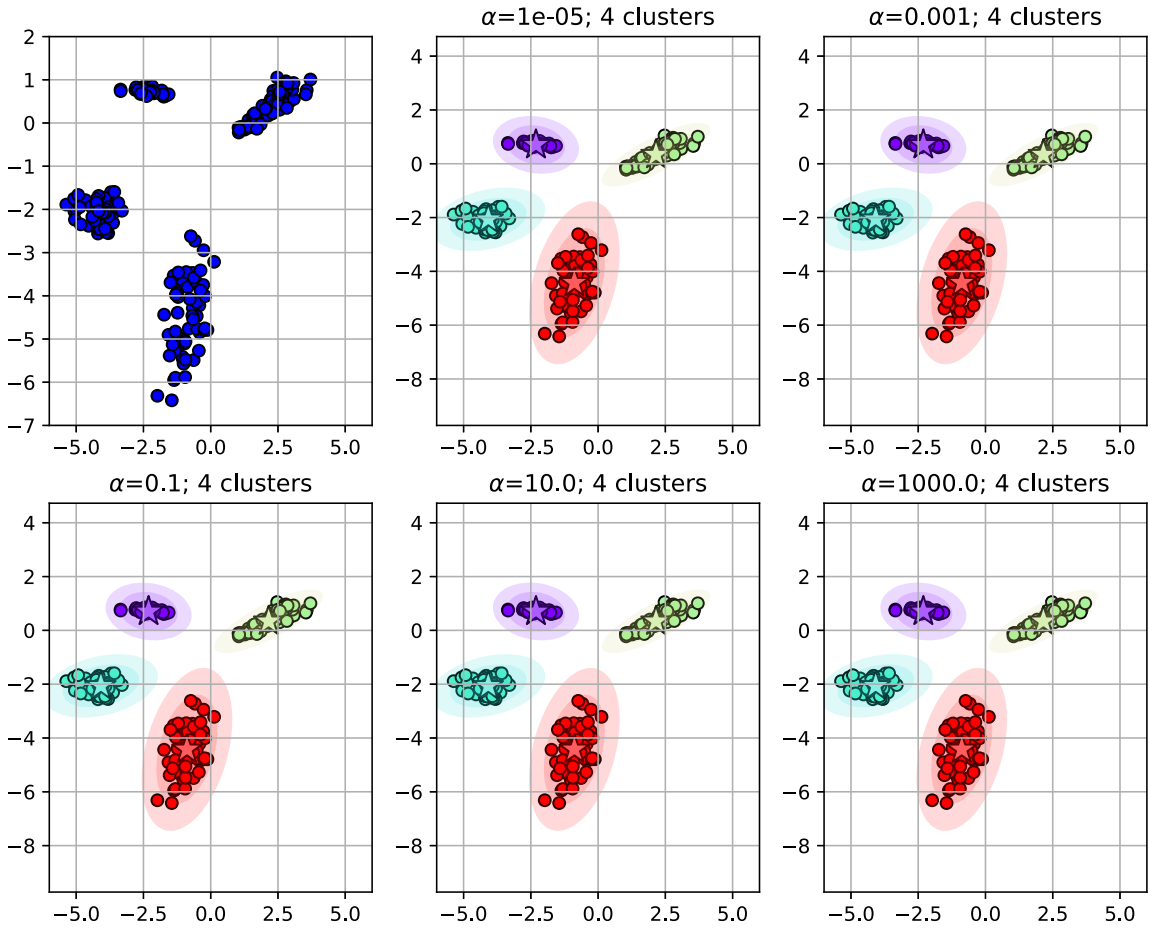
Out[31]:



- For different concentration parameter α
  - larger α may yield more clusters

In [35]:
```
dpfig
```

Out[35]:



- Choice of α is not that critical
  - same number of clusters for large ranges of α

In [37]:
```
dpgmmafig
```

Out[37]: