K-means and Expectation Maximization

- latent variable models
- k-means
- expectation maximization

Unsupervised learning

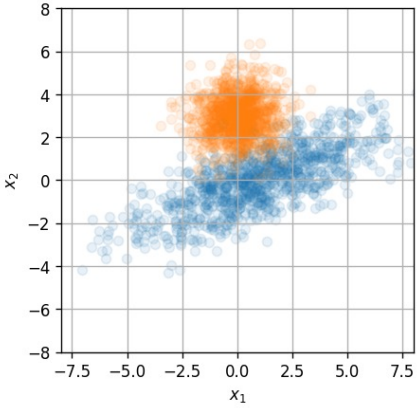
- so far we've focused on supervised learning
- *supervised* machine learning is about **learning functions from data**

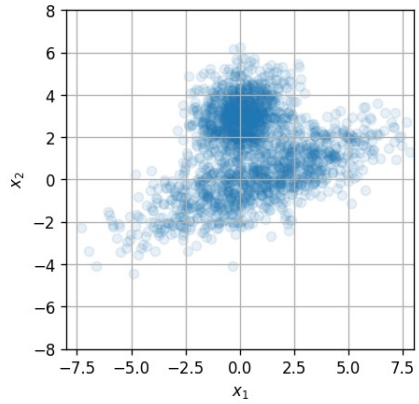
$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$$

- *unsupervised* machine learning is about finding interesting patterns in data

$$\mathcal{D} = \{(x_i)\}_{i=1}^n$$

- *knowlege discovery*
- *pattern recognition*
- under-specified





- learn $p(\boldsymbol{x})$ instead of $p(\boldsymbol{x}, y)$ or $p(y|\boldsymbol{x})$
- learn $p_\theta(\boldsymbol{x})$

- examples:

density estimation

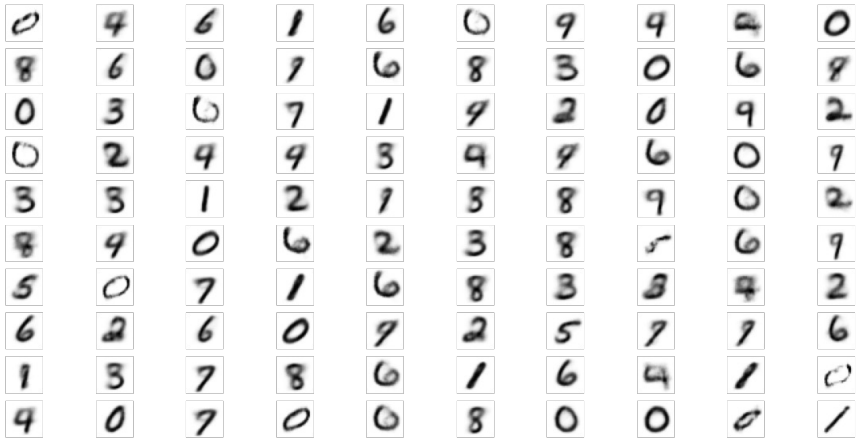
clustering

community detection in graphs

anomaly detection

Drawing Faces (or MNIST)

Variational Auto-Encoder (VAE)



Latent Variable Models

- goal: learn $p(\boldsymbol{x})$ from $\mathcal{D} = \{(\boldsymbol{x}_i)\}_{i=1}^n$

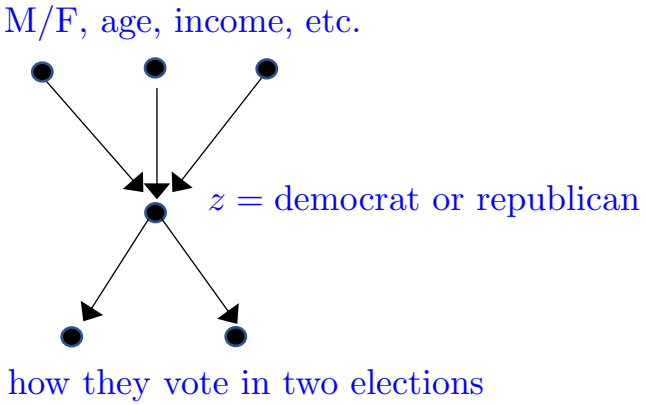
$$p(\boldsymbol{x}) = \int p(\boldsymbol{x}, z) dz \qquad p(\boldsymbol{x}) = \sum_z p(\boldsymbol{x}, z)$$

$$p(\boldsymbol{x}) = \int p(z) p(\boldsymbol{x}|z) dz \qquad p(\boldsymbol{x}) = \sum_z p(z) p(\boldsymbol{x}|z)$$

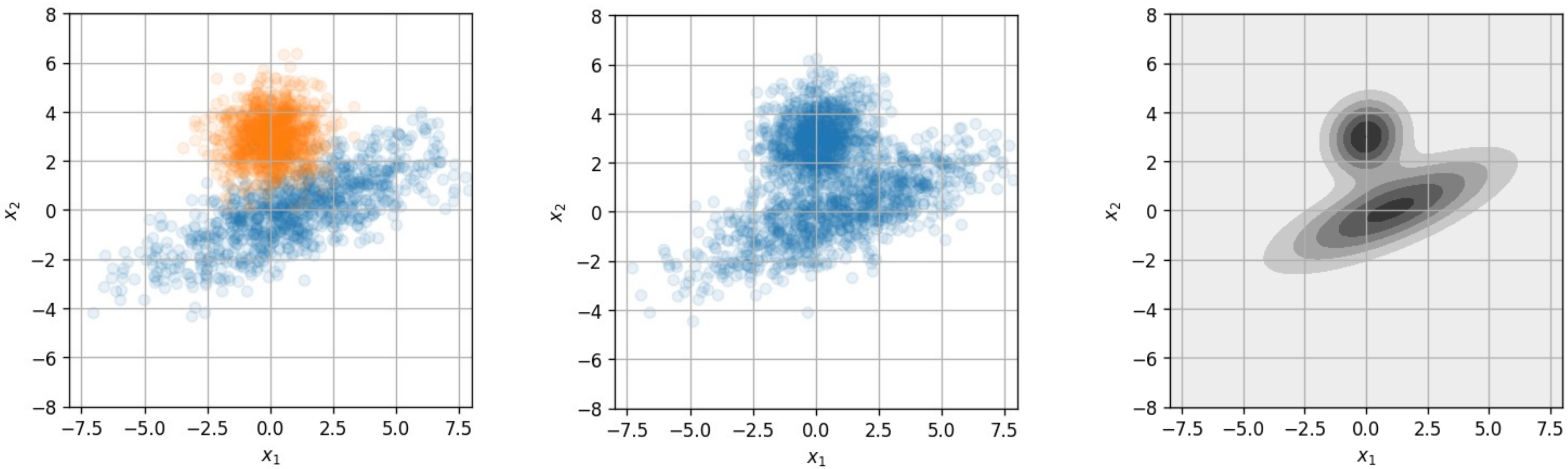
  - z is a *latent* variable

- *might* be mathematically convenient
- maybe $p(\boldsymbol{x}|z)$ has a simple form
- z might be meaningful, or maybe just convenient
- MNIST: z = digit, pen width, slant, etc.
- *latent* means hidden or concealed

  - if z is discrete, then *mixture* model



Gaussian Mixture Models (GMM)



$$p(\boldsymbol{x}) = 0.5\,\mathcal{N}\left(\boldsymbol{x}\,\middle|\,\begin{bmatrix}0\\3\end{bmatrix},\begin{bmatrix}1 & 0.1\\0.1 & 1\end{bmatrix}\right) + 0.5\,\mathcal{N}\left(\boldsymbol{x}\,\middle|\,\begin{bmatrix}1\\0\end{bmatrix},\begin{bmatrix}8 & 3\\3 & 2\end{bmatrix}\right)$$

- Gaussian mixture model:

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

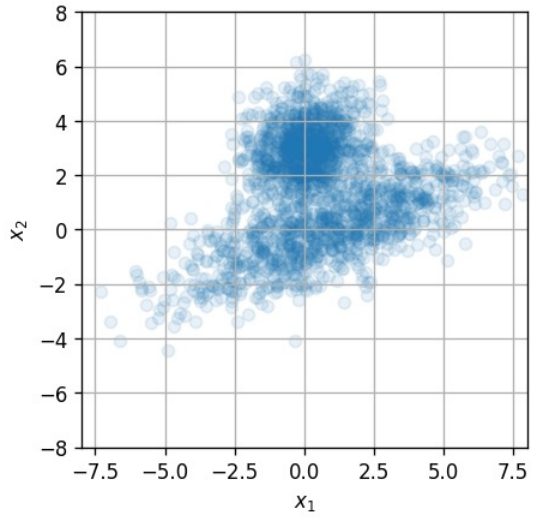
$$\mathcal{N}(x|\mu,\Sigma) = \frac{e^{-(x-\mu)^T\,\Sigma^{-1}\,(x-\mu)/2}}{(2\pi)^{n/2}\sqrt{\det\Sigma}}$$

- need to learn:

$$\pi_1, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1, \pi_2, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2, \ldots$$

- uses of GMMs: density estimation, clustering

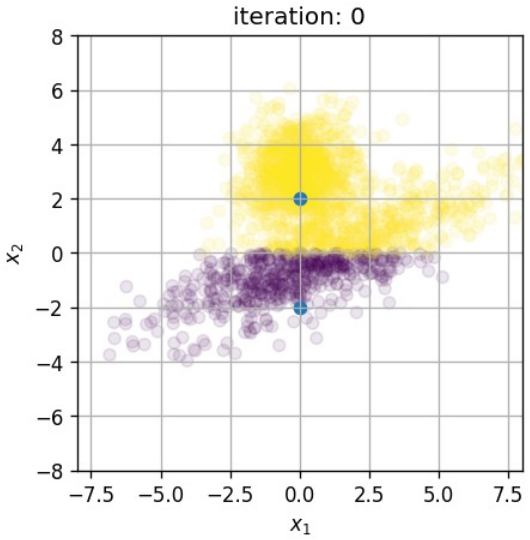
Clustering with k-Means

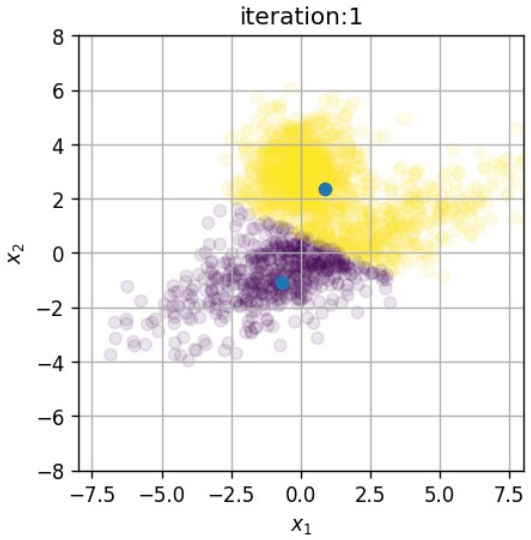


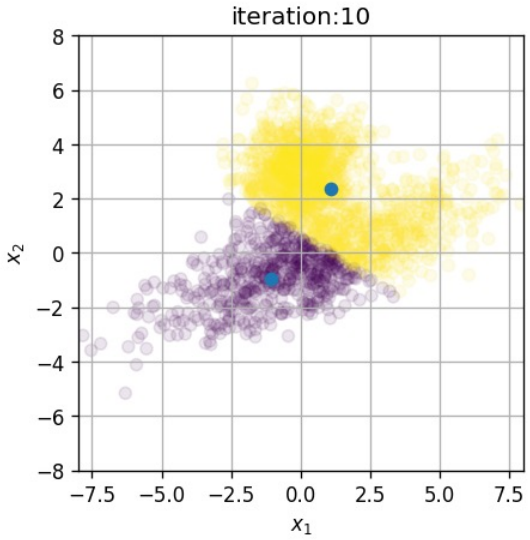
initialize cluster centers $\boldsymbol{\mu}_k$, $k = 1, \ldots, K$

1. assign each point to the closest cluster center: $z_i = \text{argmin}_k \|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|_2^2$
2. update cluster centers: $\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i:z_i=k} \boldsymbol{x}_i$
3. repeat 1, 2

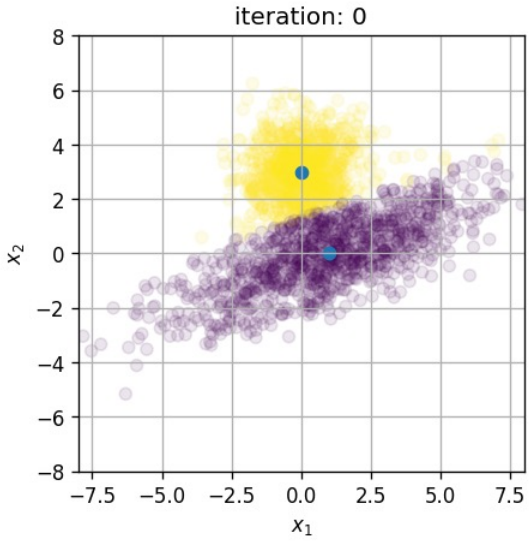
K-means

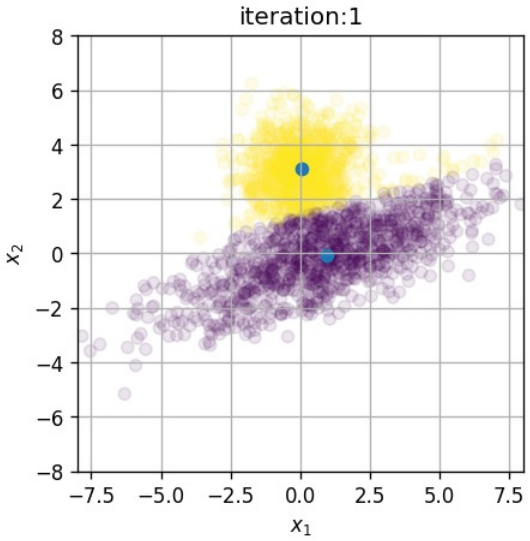

iteration: 0
8
6
4
2
0
−2
−4
−6
−8
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
x_1
x_2

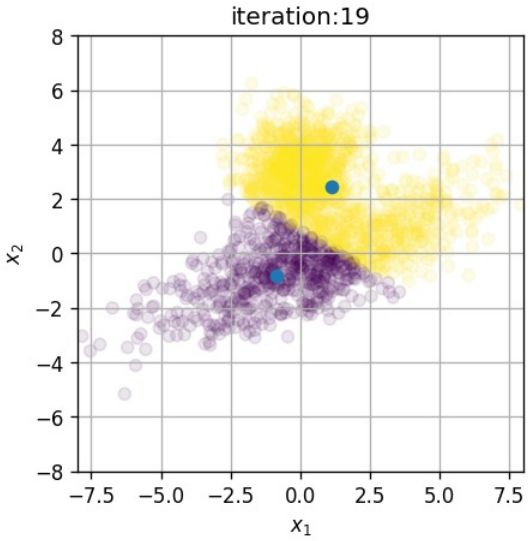

iteration:1
8
6
4
2
0
−2
−4
−6
−8
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
x_1
x_2


iteration:10
8
6
4
2
0
−2
−4
−6
−8
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
x_1
x_2

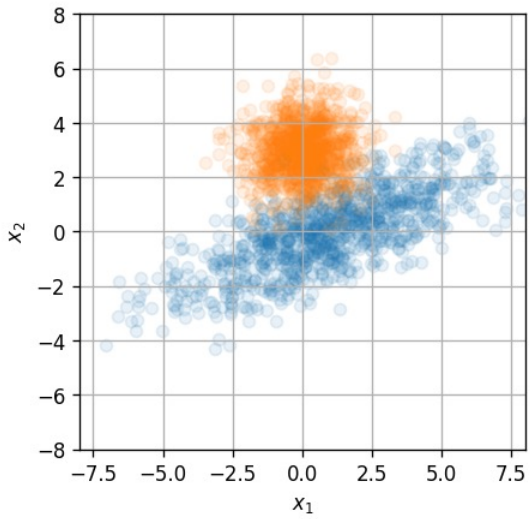
K-means

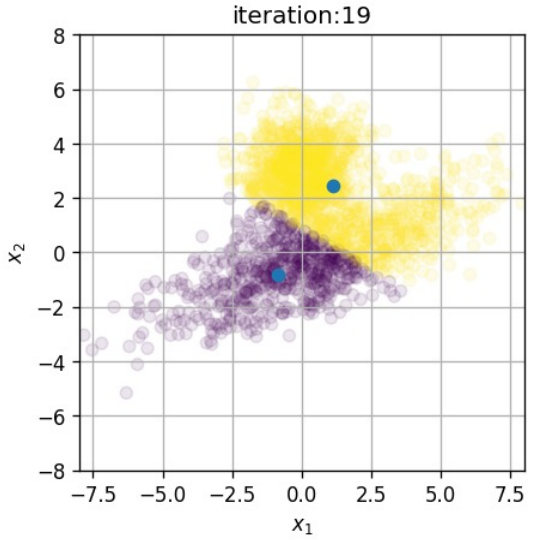

iteration: 0
8
6
4
2
0
−2
−4
−6
−8
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
x_1
x_2


iteration:1
8
6
4
2
0
−2
−4
−6
−8
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
x_1
x_2


iteration:19
8
6
4
2
0
−2
−4
−6
−8
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
x_1
x_2

K-means


8
6
4
2
0
−2
−4
−6
−8
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
x_1
x_2


iteration:19
8
6
4
2
0
−2
−4
−6
−8
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
x_1
x_2

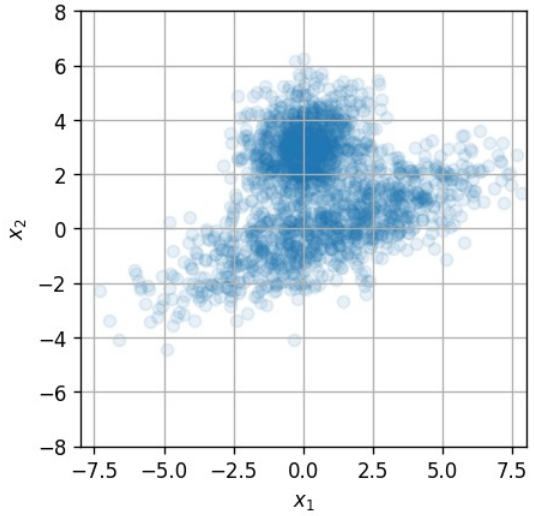
Expectation Maximization (EM) for GMMs

Bayes: $p(z|\boldsymbol{x}) = \frac{p(z)p(\boldsymbol{x}|z)}{p(\boldsymbol{x})}$

initialize means: $\boldsymbol{\mu}_k,\ j = 1, \ldots, K$

initialize covariances: $\boldsymbol{\Sigma}_k = \boldsymbol{I}$

initialize priors: $\pi_k = \frac{1}{K}$



E-step: 1. compute the responsibility of each point to each cluster (*soft* assign each point):

$$p(z = k|\boldsymbol{x}_i) = \quad r_{ik} = \frac{\pi_k \mathcal{N}(\boldsymbol{x}_i|\mu_k, \Sigma_k)}{\sum_{k'} \pi_{k'} \mathcal{N}(\boldsymbol{x}_i|\mu_{k'}, \Sigma_{k'})}$$

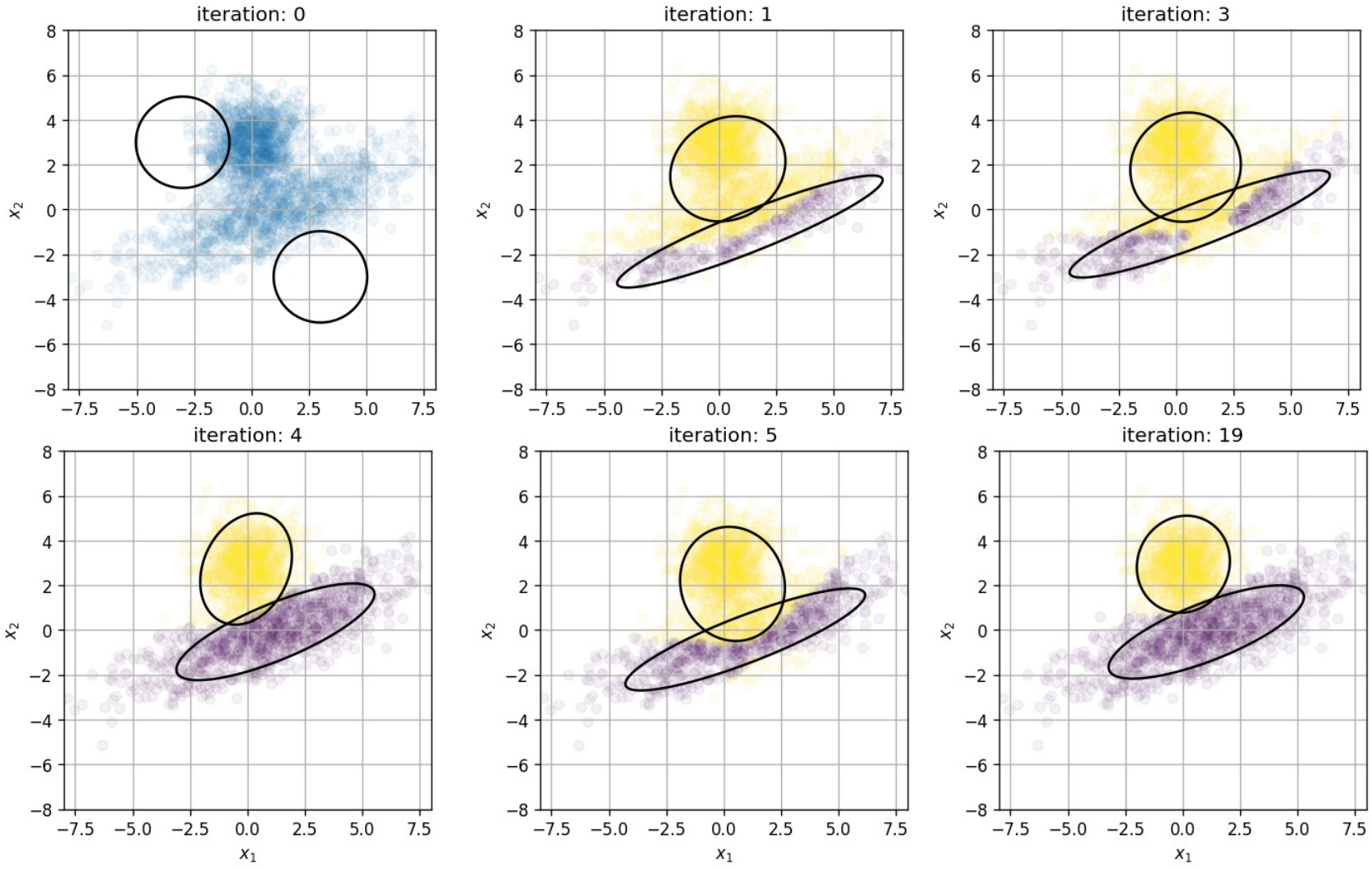
M-step: 2. estimate prior, mean and covariance for each mixture component

$$\pi_k = \frac{1}{N} \sum_i \mathbb{I}\{z_i = k\} \qquad \pi_k = \frac{1}{N} \sum_i E[\mathbb{I}\{z_i = k\}] \ = \frac{1}{N} \sum_i p(z = k|\boldsymbol{x}_i)$$

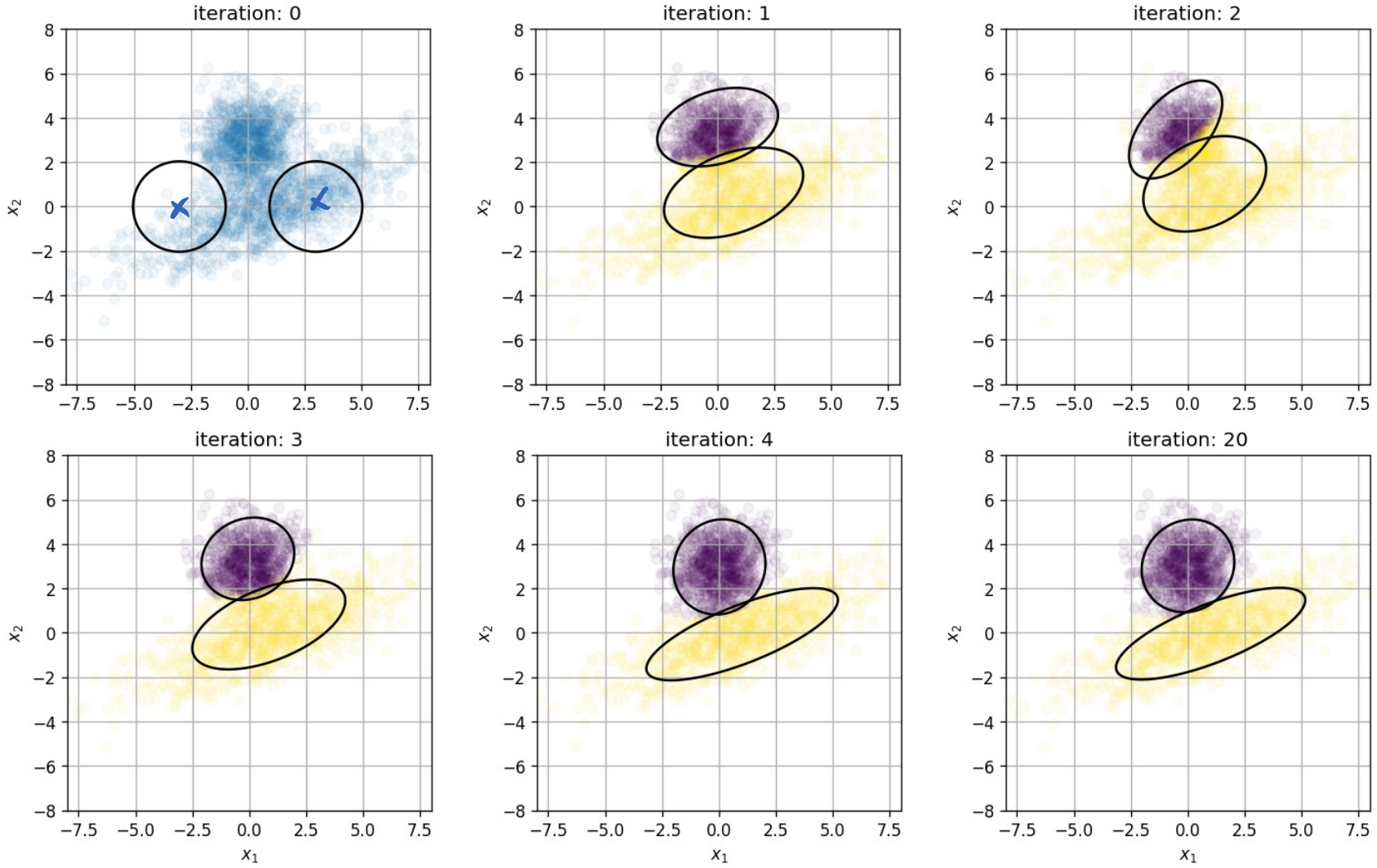
$$\pi_k = \frac{1}{N} \sum_i r_{ik} \qquad \boldsymbol{\mu}_k = \frac{1}{\sum_i r_{ik}} \sum_i r_{ik} \boldsymbol{x}_i \qquad \boldsymbol{\Sigma}_k = \frac{1}{\sum_i r_{ik}} \sum_i r_{ik} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^T$$

3. repeat 1, 2

EM


iteration: 0
iteration: 1
iteration: 3
iteration: 4
iteration: 5
iteration: 19
x_1
x_2

EM


iteration: 0
iteration: 1
iteration: 2
iteration: 3
iteration: 4
iteration: 20
x_1
x_2

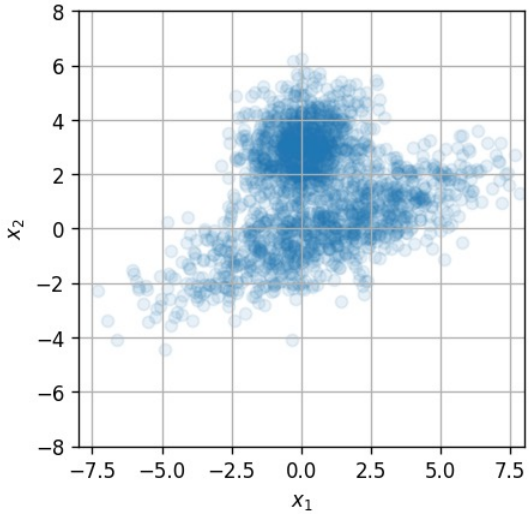
Expectation Maximization (EM)

- observe $\mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$

- goal: maximize the log-likelihood:

$$\log L(\theta) = \sum_{i=1}^{n} \log p_\theta(\boldsymbol{x}_i) = \sum_{i=1}^{n} \log \sum_{z_i} p_\theta(z_i, \boldsymbol{x}_i)$$

$$= \sum_{i=1}^{n} \log \sum_{z_i} p(z_i) p_\theta(\boldsymbol{x}_i | z_i)$$



E-step:

- for each data point, compute $p_{\theta_{t-1}}(z_i | \boldsymbol{x}_i)$.

  i.e, compute the posterior of the latent variable for fixed θ_{t-1}

M-step:

- maximium likelihood for paramaters θ given the expected value of z_i from E-step

$$\theta_t = \arg\max_\theta \sum_i E_{z \sim p_{\theta_t}(z|\boldsymbol{x}_i)} \log p_\theta(\boldsymbol{x}_i, z)$$

- repeat

K-means and EM

- K means

  initialize cluster centers $\boldsymbol{\mu}_k$, $k = 1, \ldots, K$

  1. assign each point to the closest cluster center: $z_i = \arg\min_k ||\boldsymbol{x} - \boldsymbol{\mu}_k||_2^2$
  2. update cluster centers: $\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i:z_i=k} \boldsymbol{x}_i$
  3. repeat 1, 2

- K means and EM

  fix covariances: $\boldsymbol{\Sigma}_k = \boldsymbol{I}$

  fix priors: $\pi_k = \frac{1}{K}$

  E-step: 1. compute the **most** responsible cluster for each each point

$$z_i = \arg\max_k \frac{\pi_k \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \Sigma_k)}{\sum_{k'} \pi_{k'} \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_{k'}, \Sigma_{k'})}$$

hard assignment to each cluster

  M-step:

  2. estimate mean for each mixture component
  3. repeat 1, 2

EM

```
from sklearn import mixture
gmm = mixture.GaussianMixture(n_components=2, covariance_type='full')
gmm.fit(X_rows)

print('covariances: ', gmm.covariances_)
print('means: ',  gmm.means_)
```

```
covariances:
[[[8.22408446 3.03067247]
  [3.03067247 1.97311564]]

 [[1.04096518 0.073664  ]
  [0.073664   1.18887948]]]

means:
[[ 1.02671502 -0.08439335]
 [ 0.02955104  2.94435719]]
```

$$\mathcal{N}\left(\begin{bmatrix}0\\3\end{bmatrix}, \begin{bmatrix}1 & 0.1\\0.1 & 1\end{bmatrix}\right)$$

$$\mathcal{N}\left(\begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}8 & 3\\3 & 2\end{bmatrix}\right)$$