EM_Solution

December 21, 2021

1 Fitting Gaussian Mixture Models with Expectation-Maximization Algorithm

1.0.1 Mixture Models

A mixture model is defined as:

$$p(\mathbf{x}) = \sum_{k=1}^{K} p(\mathbf{x} \mid \mathcal{G}_k) P(\mathcal{G}_k),$$

where

$$\begin{aligned} \mathcal{G}_k &= \text{mixture component (or clusters or groups)} \\ p\left(\mathbf{x} \mid \mathcal{G}_k\right) &= \text{component densities} \\ P\left(\mathcal{G}_k\right) &= \text{mixture proportions (or priors) s.t. } P\left(\mathcal{G}_k\right) \geq 0, \ \sum_i P\left(\mathcal{G}_k\right) = 1. \end{aligned}$$

The number of components, K, is a hyperparameter to be specified beforehand.

Given a sample set $\mathcal{X} = \{\mathbf{x}^n\}_{n=1}^N$, learning is to estimate the $p(\mathbf{x} \mid \mathcal{G}_k)$ and $P(\mathcal{G}_k)$.

1.0.2 Gaussian Mixture Models

Gaussian mixture model (GMM) assume the mixture components are Gaussian, i.e.,:

$$p\left(\mathbf{x} \mid \mathcal{G}_k\right) = \mathcal{N}\left(\mu_k, \Sigma_k\right).$$

Then the parameters need to be estimated are

$$\mathbf{\Phi} = \{P(\mathcal{G}_i), \mu_i, \Sigma_i\}_{k=1}^K.$$

Denote $\pi_k = P(\mathcal{G}_k)$ for brevity, the GMM has the following form:

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

where $\pi_1, \ldots, \pi_K$ are the mixing coefficients satisfy

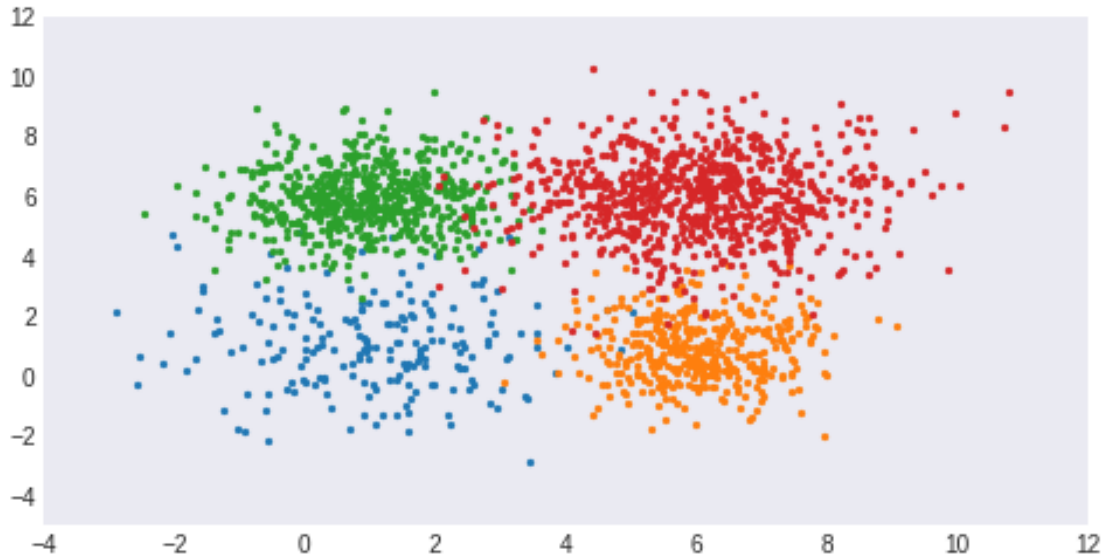
$$\sum_{k=1}^{K} \pi_k = 1 \ \text{ and } \ \pi_k \geq 0, \forall k = 1, \ldots, K.$$

Let us define a two-dimensional GMM with $K = 4$, where $\mu_1 = [1, 1], \mu_2 = [6, 1], \mu_3 = [1, 6], \mu_4 = [6, 6]$, and $\sigma_1^2 = [2, 2], \sigma_2^2 = [1, 1], \sigma_3^2 = [1, 1], \sigma_4^2 = [2, 2]$.

Sample 2000 points from the four distributions, we have the following dataset.

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.patches import Ellipse
from scipy.stats import multivariate_normal
plt.style.use('seaborn-dark')

n = 2000
K = 4
# Data Generation
X1 = np.random.multivariate_normal([1,1], np.diag([2,2]), 200)
X2 = np.random.multivariate_normal([6,1], np.diag([1,1]), 400)
X3 = np.random.multivariate_normal([1,6], np.diag([1,1]), 600)
X4 = np.random.multivariate_normal([6,6], np.diag([2,2]), 800)
X = np.vstack((X1, X2, X3,X4))
# Data Visualization
plt.figure(figsize=(8,4))
plt.axis([-4, 12, -5, 12])
plt.scatter(X1[:, 0], X1[:, 1], s=5)
plt.scatter(X2[:, 0], X2[:, 1], s=5)
plt.scatter(X3[:, 0], X3[:, 1], s=5)
plt.scatter(X4[:, 0], X4[:, 1], s=5)
plt.show()
```



Suppose we observe the 2000 points and our interest is to fit the underlying GMM model, then we need to find k paris of means $\mu_1, \ldots, \mu_k$ and variance $\sigma_1, \ldots, \sigma_k$.

1.0.3 Expectation-Maximization

Using GMM, the log likelihood is

$$\mathcal{L}(\mathbf{\Phi} \mid \mathcal{X}) = \log \prod_{n=1}^{N} p\left(\mathbf{x}^n \mid \mathbf{\Phi}\right) = \sum_{n=1}^{N} \log p\left(\mathbf{x}^n \mid \mathbf{\Phi}\right) = \sum_{n=1}^{N} \log \sum_{k=1}^{K} \pi_k \mathcal{N}\left(\mathbf{x} \mid \mu_k, \Sigma_k\right).$$

[2]:
```
def likelihood(X, Pi, Mu, Var):
    n_points, n_clusters = len(X), len(Pi)
    pdfs = np.zeros(((n_points, n_clusters)))
    for i in range(n_clusters):
        pdfs[:, i] = Pi[i] * multivariate_normal.pdf(X, Mu[i], np.diag(Var[i]))
    log_likelihood =  np.sum(np.log(pdfs.sum(axis=1)))
    return log_likelihood
```

Instead of maximize $\mathcal{L}(\mathbf{\Phi} \mid \mathcal{X})$ directly, the EM method introduces some hidden variables. In the case of mixture model, hidden variables are the source of the observations, namely, which observation belongs to which component.

Specifically, we define $\mathbf{z}^n = (z_1^n, \ldots, z_k^n)^T$ as the indicator variable which is the label for $\mathbf{x}^n$:

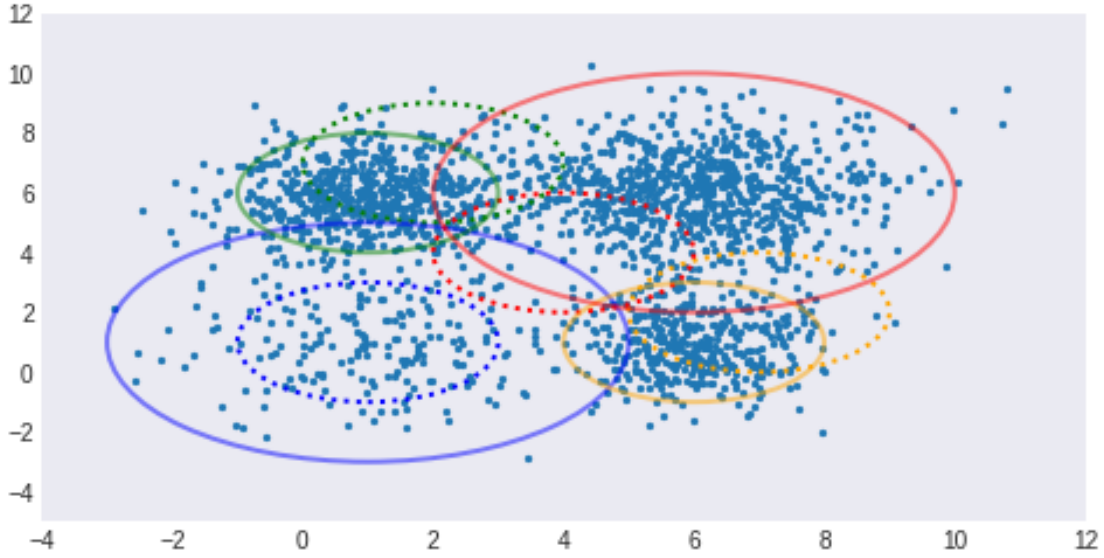
$$z_k^n = \begin{cases} 1 & \text{if } \mathbf{x}^n \text{ belongs to cluster } \mathcal{G}_k \\ 0 & \text{otherwise.} \end{cases}$$

The EM algorithm for mixture model contains two steps: 1. E-step: estimating the labels $\mathbf{z}^n$ of the observations given our current knowledge of the components. 2. M-step: updating the component knowledge $\mathbf{\Phi}$ given the labels estimated.

[3]:
```
z = np.ones((n, K)) / K
Pi = [1 / K] * 4
Mu = [[1,1], [7,2], [2, 7],[4,4]]
Var = [[1,1],[1,1],[1,1],[1,1]]
log_likelihood = []

Mu_Star = [[1,1], [6,1], [1, 6],[6,6]]
Var_Star = [[2,2], [1,1], [1, 1],[2,2]]
colors = ['b', 'orange', 'g', 'r']
plt.figure(figsize=(8,4))
plt.axis([-4, 12, -5, 12])
plt.scatter(X[:, 0], X[:, 1], s=5)
ax = plt.gca()
for i in range(K):
    plot_args = {'fc': 'None', 'lw': 2, 'edgecolor': colors[i], 'ls': ':'}
    ellipse = Ellipse(Mu[i], 4 * Var[i][0], 4 * Var[i][1], **plot_args)
    ax.add_patch(ellipse)
for i in range(K):
    plot_args = {'fc': 'None', 'lw': 2, 'edgecolor': colors[i], 'alpha': 0.5}
```

```
    ellipse = Ellipse(Mu_Star[i], 4* Var_Star[i][0], 4 * Var_Star[i][1],␣
↪**plot_args)
    ax.add_patch(ellipse)
plt.show()
```



First, we compute the following expectation w.r.t. $\mathbf{Z}$ given $\mathcal{X}$ and $\mathbf{\Phi}^t$, i.e., computing the $\mathcal{Q}$ function:

$$\begin{aligned}
\mathcal{Q}(\mathbf{\Phi} \mid \mathbf{\Phi}^t) &= \mathbb{E}\left[\mathcal{L}(\mathbf{\Phi} \mid \mathcal{X}, \mathbf{Z}) \mid \mathcal{X}, \mathbf{\Phi}^t\right] \\
&= \mathbb{E}\left[\sum_{n=1}^{N} \log p(\mathbf{x}^n, \mathbf{z}^n \mid \mathbf{\Phi}) \mid \mathcal{X}, \mathbf{\Phi}^t\right] \\
&= \mathbb{E}\left[\sum_{n=1}^{N} \log p(\mathbf{z}^n \mid \mathbf{\Phi}) + log p(\mathbf{x}^n \mid \mathbf{z}^n, \mathbf{\Phi}) \mid \mathcal{X}, \mathbf{\Phi}^t\right] \\
&= \sum_{k=1}^{K}\sum_{n=1}^{N} \mathbb{E}\left[z_k^n \mid \mathcal{X}, \mathbf{\Phi}^t\right] \left(\log \pi_k + log \mathcal{N}\left(\mathbf{x} \mid \mu_k, \Sigma_k\right)\right).
\end{aligned}$$

E-step In E-step, we estimate the labels $\mathbf{z}^n$ of the observations given our current knowledge of the components.

Observe that

$$\begin{aligned}
\mathbb{E}\left[z_k^n \mid \mathcal{X}, \mathbf{\Phi}^t\right] &= p(z_k^n = 1 \mid \mathbf{x}^n, \mathbf{\Phi}^t) \\
&= \frac{p\left(\mathbf{x}^n \mid z_k^n = 1, \mathbf{\Phi}^t\right) P\left(z_k^n = 1 \mid \mathbf{\Phi}^t\right)}{p\left(\mathbf{x}^n \mid \mathbf{\Phi}^t\right)} \\
&= \frac{\pi_k \mathcal{N}\left(\mathbf{x}^n \mid \mu_k^n, \Sigma_k^n\right)}{\sum_{i=1}^{K} \pi_i \mathcal{N}\left(\mathbf{x}^n \mid \mu_i^t, \Sigma_i^t\right)}
\end{aligned}$$

M-Step I

- Maximization of $\mathcal{Q}(\boldsymbol{\Phi} \mid \boldsymbol{\Phi}^t)$:

$$\begin{aligned}\boldsymbol{\Phi}^{t+1} &= \arg\max_{\boldsymbol{\Phi}} \mathcal{Q}(\boldsymbol{\Phi} \mid \boldsymbol{\Phi}^t) = \arg\max_{\boldsymbol{\Phi}} \left[\sum_{\ell}\sum_{i} h_i^{(\ell)} \left[\log \pi_i + \log p_i(\mathbf{x}^{(\ell)} \mid \boldsymbol{\Phi})\right]\right] \\ &= \arg\max_{\boldsymbol{\Phi}} \left[\sum_{\ell}\sum_{i} h_i^{(\ell)} \log \pi_i + \sum_{\ell}\sum_{i} h_i^{(\ell)} \log p_i(\mathbf{x}^{(\ell)} \mid \boldsymbol{\Phi})\right] \\ &= \arg\max_{\boldsymbol{\Phi}} \left[\sum_{\ell}\sum_{i} h_i^{(\ell)} \log \pi_i + \sum_{\ell}\sum_{i} h_i^{(\ell)} \log p_i(\mathbf{x}^{(\ell)} \mid \boldsymbol{\Theta}_i)\right]\end{aligned}$$

Or, equivalently, the problem is

$$\begin{aligned}\underset{\{\pi_i\},\{\boldsymbol{\Theta}_i\}}{\text{maximize}} \quad & \mathcal{Q}(\boldsymbol{\Phi} \mid \boldsymbol{\Phi}^t) = \sum_{\ell}\sum_{i} h_i^{(\ell)} \log \pi_i + \sum_{\ell}\sum_{i} h_i^{(\ell)} \log p_i(\mathbf{x}^{(\ell)} \mid \boldsymbol{\Theta}_i) \\ \text{subject to} \quad & \sum_i \pi_i = 1\end{aligned}$$

M-step

- The second term of $\mathcal{Q}(\boldsymbol{\Phi} \mid \boldsymbol{\Phi}^t)$ does not depend on π_i. The problem for $\{\pi_i\}$ is

$$\begin{aligned}\underset{\{\pi_i\}}{\text{maximize}} \quad & \sum_{\ell}\sum_{i} h_i^{(\ell)} \log \pi_i \\ \text{subject to} \quad & \sum_i \pi_i = 1\end{aligned}$$

Using the constraint $\sum_i \pi_i = 1$ to define the Lagrangian, we solve for

$$\frac{\partial}{\partial \pi_i}\left[\sum_{\ell}\sum_{i} h_i^{(\ell)} \log \pi_i - \lambda\left(\sum_i \pi_i - 1\right)\right] = 0$$

to get

$$\pi_i = \frac{\sum_{\ell} h_i^{(\ell)}}{N}$$

which is analogous to the prior estimation in classification.

M-Step III

- The first term of $\mathcal{Q}(\boldsymbol{\Phi} \mid \boldsymbol{\Phi}^t)$ does not depend on $\boldsymbol{\Theta}_i$. The problem for $\{\boldsymbol{\Theta}_i\}$ is

$$\underset{\{\boldsymbol{\Theta}_i\}}{\text{maximize}} \quad \sum_{\ell}\sum_{i} h_i^{(\ell)} \log p_i(\mathbf{x}^{(\ell)} \mid \boldsymbol{\Theta}_i)$$

We solve for

$$\frac{\partial}{\partial \boldsymbol{\Theta}_i} \sum_{\ell}\sum_{i} h_i^{(\ell)} \log p_i\left(\mathbf{x}^{(\ell)} \mid \boldsymbol{\Phi}\right) = \frac{\partial}{\partial \boldsymbol{\Theta}_i} \sum_{\ell} h_i^{(\ell)} \log p_i\left(\mathbf{x}^{(\ell)} \mid \boldsymbol{\Theta}_i\right) = \mathbf{0}$$

M-Step IV

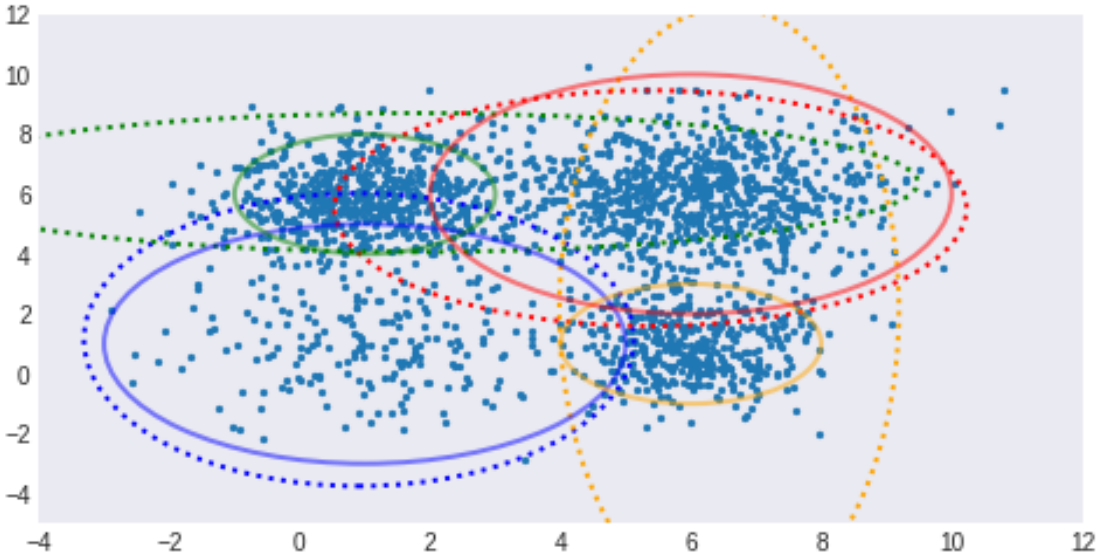
- Specifically, for Gaussian components, $\hat{p}_i(\mathbf{x}^{(\ell)} \mid \boldsymbol{\Theta}_i) = \mathcal{N}(\mathbf{m}_i, \mathbf{S}_i)$, and so we get

$$\mathbf{m}_i^{t+1} = \frac{\sum_\ell h_i^{(\ell)} \mathbf{x}^{(\ell)}}{\sum_\ell h_i^{(\ell)}} \quad \text{and} \quad \mathbf{S}_i^{t+1} = \frac{\sum_\ell h_i^{(\ell)} (\mathbf{x}^{(\ell)} - \mathbf{m}_i^{t+1})(\mathbf{x}^{(\ell)} - \mathbf{m}_i^{t+1})^T}{\sum_\ell h_i^{(\ell)}}$$

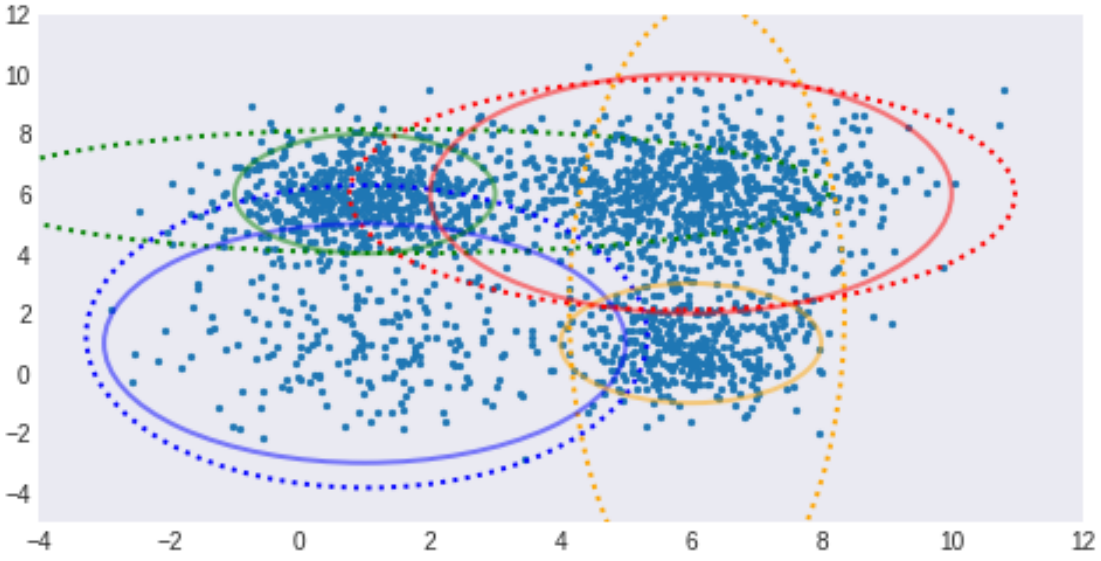
[4]:
```
def EM(X, z, Pi, Mu, Var):
    pdfs = np.zeros(((n, K)))
    for i in range(K):
        pdfs[:, i] = Pi[i] * multivariate_normal.pdf(X, Mu[i], np.diag(Var[i]))
    z = pdfs / pdfs.sum(axis=1).reshape(-1, 1)
    Pi = z.sum(axis=0) / z.sum()
    for i in range(K):
        Mu[i] = np.average(X, axis=0, weights=z[:, i])
        Var[i] = np.average((X - Mu[i]) ** 2, axis=0, weights=z[:, i])
    return z, Pi, Mu, Var

Mu_Star = [[1,1], [6,1], [1, 6],[6,6]]
Var_Star = [[2,2], [1,1], [1, 1],[2,2]]
colors = ['b', 'orange', 'g', 'r']
for i in range(20):
    log_likelihood.append(likelihood(X, Pi, Mu, Var))
    z, Pi, Mu, Var = EM(X, z, Pi, Mu, Var)
    print('log-likehood:%.3f'%log_likelihood[-1])
    plt.figure(figsize=(8,4))
    plt.axis([-4, 12, -5, 12])
    plt.scatter(X[:, 0], X[:, 1], s=5)
    ax = plt.gca()
    for i in range(K):
        plot_args = {'fc': 'None', 'lw': 2, 'edgecolor': colors[i], 'ls': ':'}
        ellipse = Ellipse(Mu[i], 4 * Var[i][0], 4 * Var[i][1], **plot_args)
        ax.add_patch(ellipse)
    for i in range(K):
        plot_args = {'fc': 'None', 'lw': 2, 'edgecolor': colors[i], 'alpha': 0.
↪5}
        ellipse = Ellipse(Mu_Star[i], 4* Var_Star[i][0], 4 * Var_Star[i][1],␣
↪**plot_args)
        ax.add_patch(ellipse)
    plt.show()
```

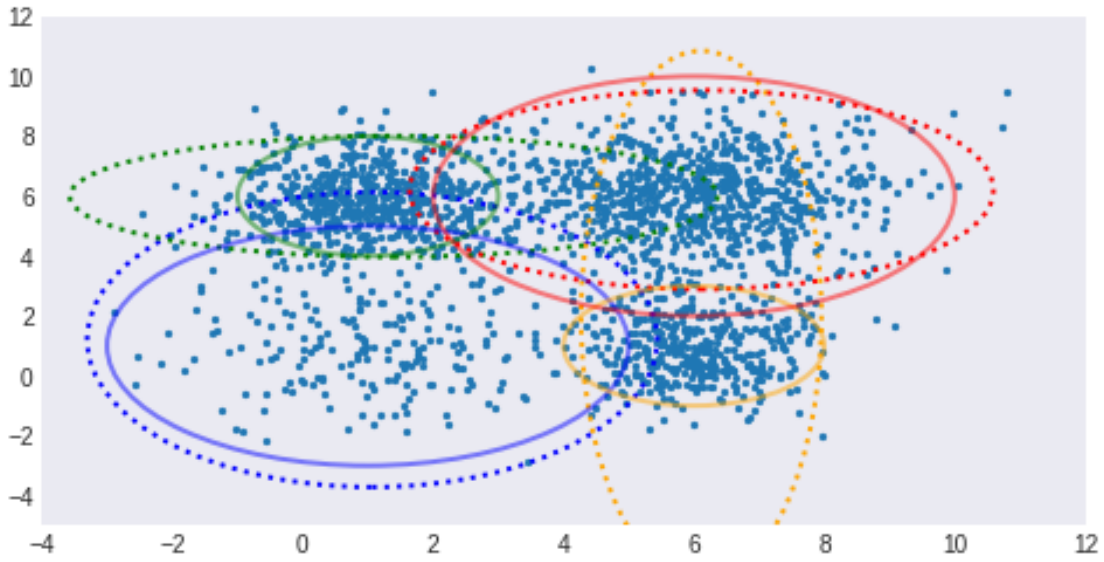
```
log-likehood:-12562.725
```



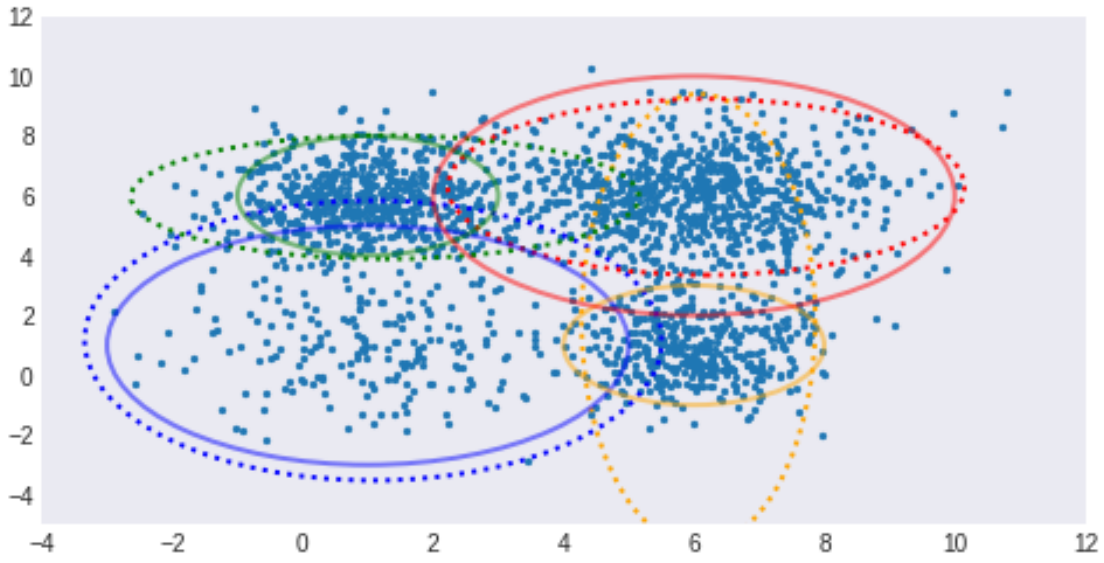
log-likehood:-9150.175



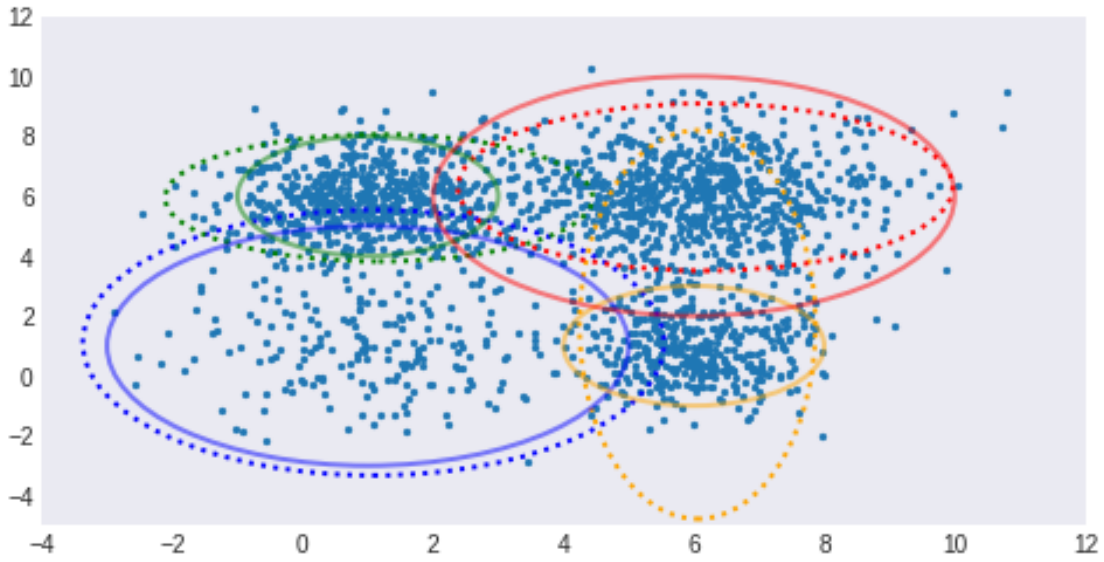
log-likehood:-8958.559



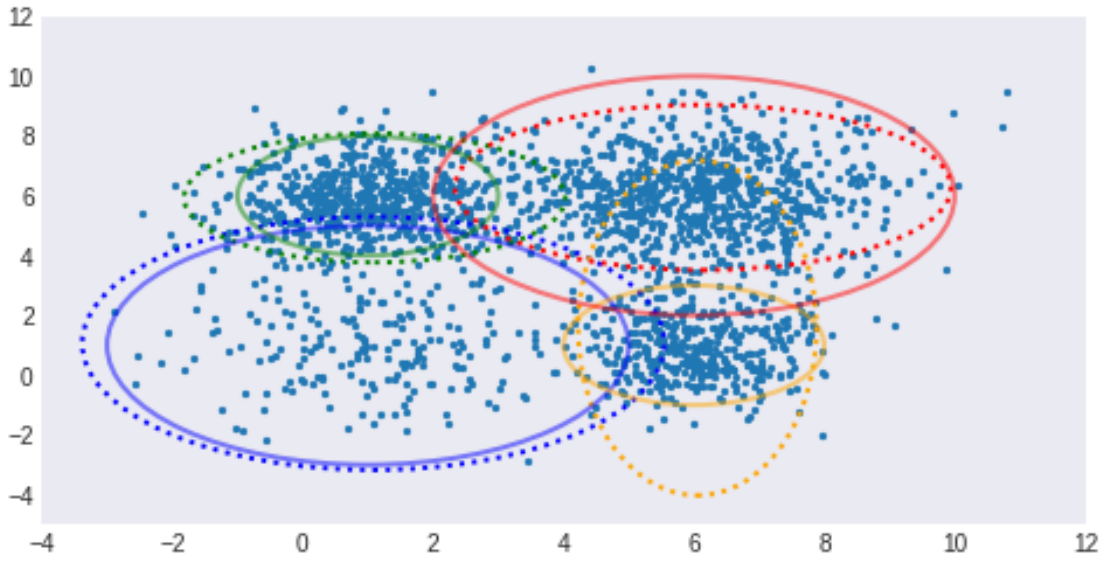
log-likehood:-8878.288



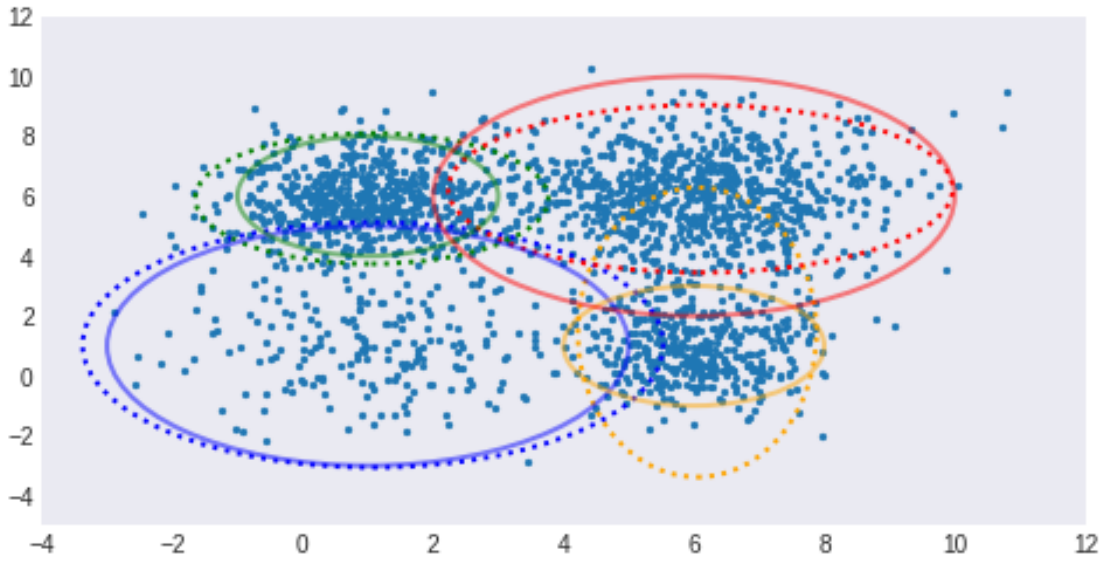
log-likehood:-8839.246



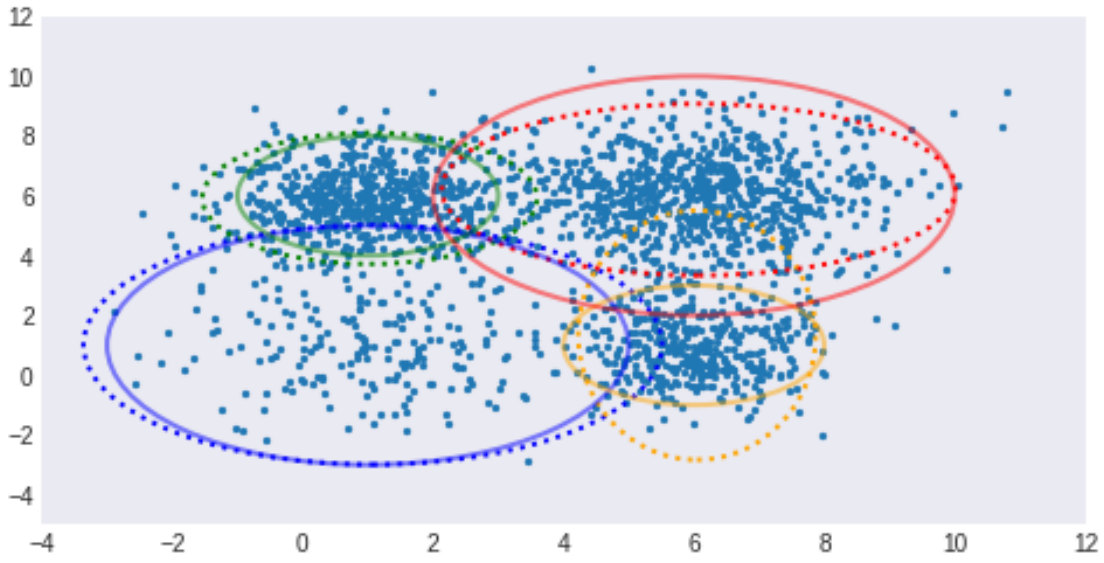
log-likehood:-8817.664



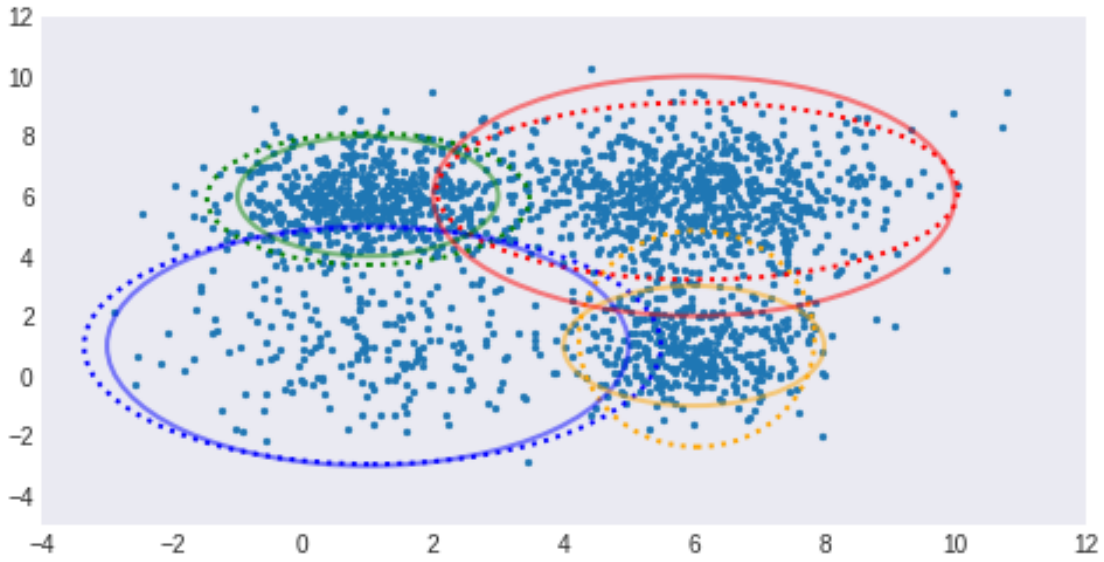
log-likehood:-8802.158



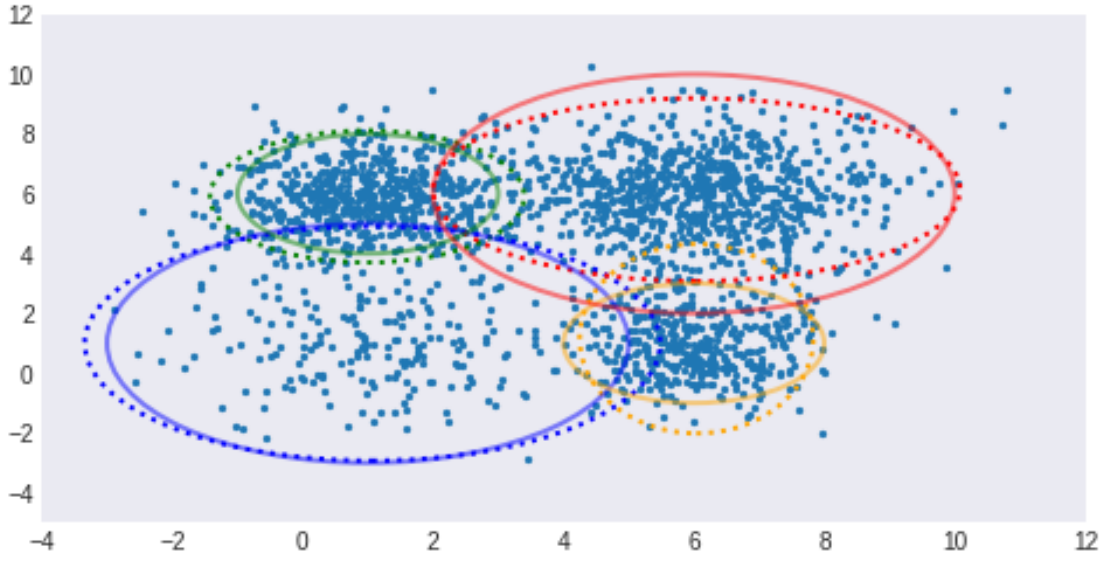
log-likehood:-8789.305



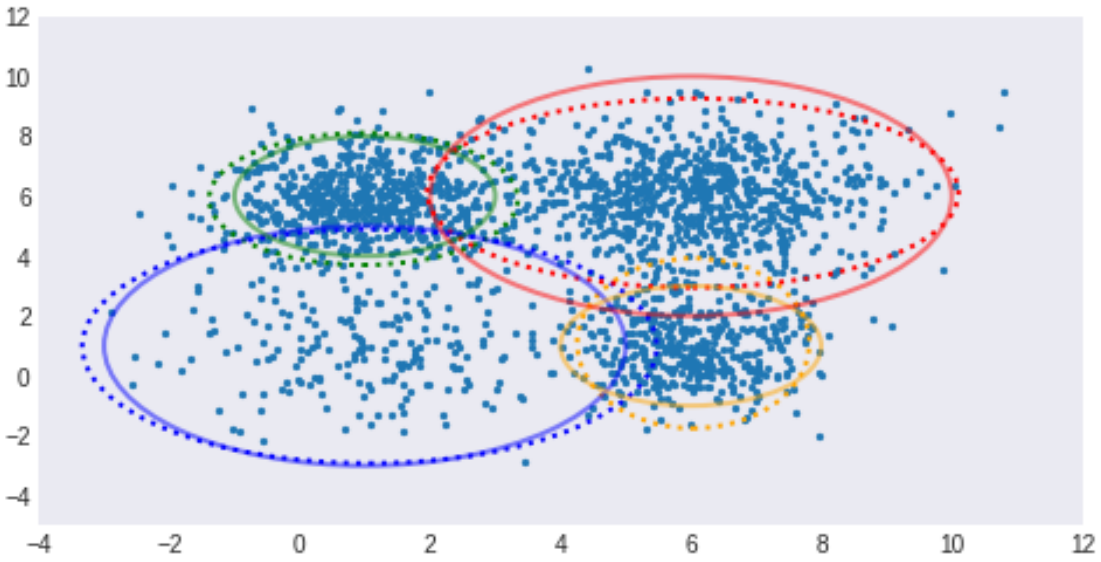
log-likehood:-8778.319



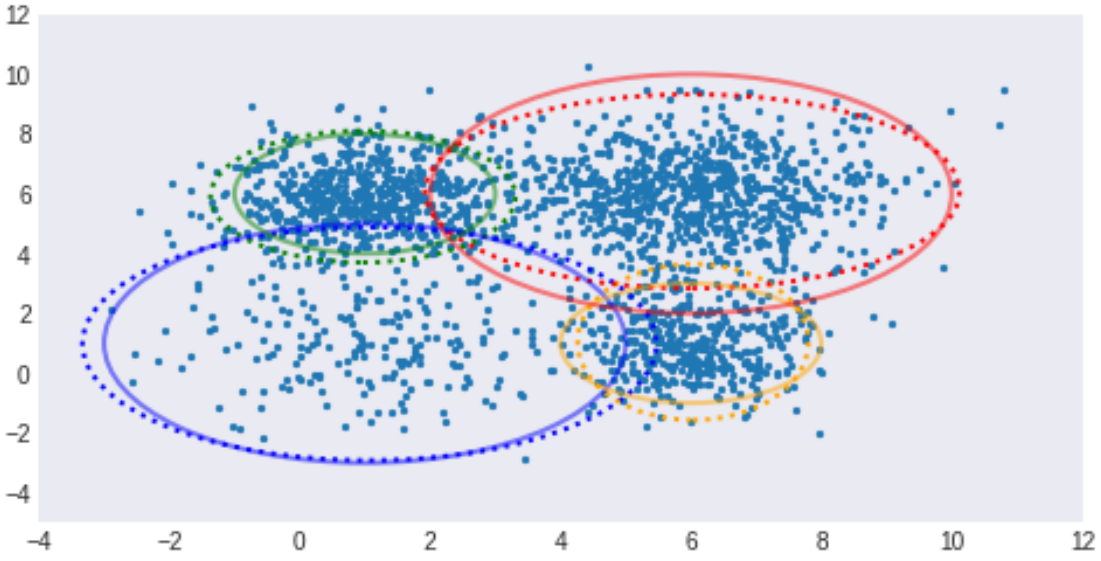
log-likehood:-8769.264



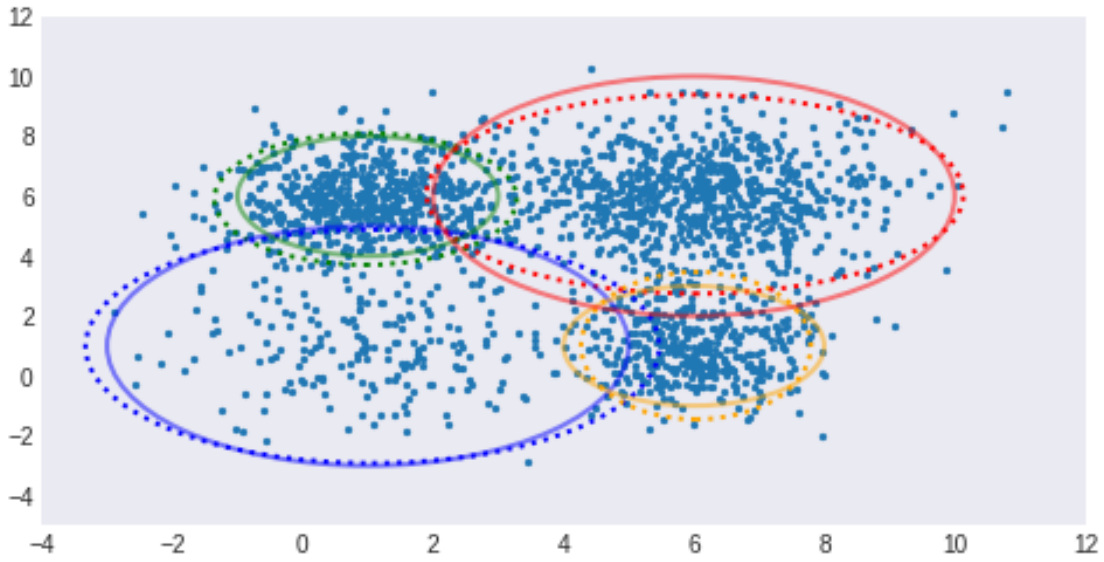
log-likehood:-8762.449



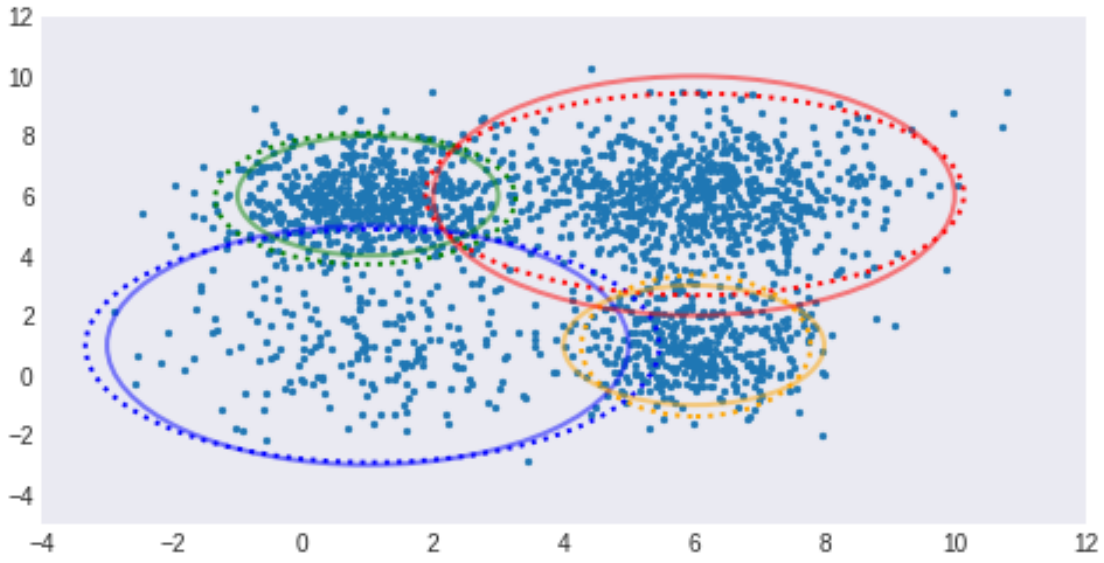
log-likehood:-8757.924



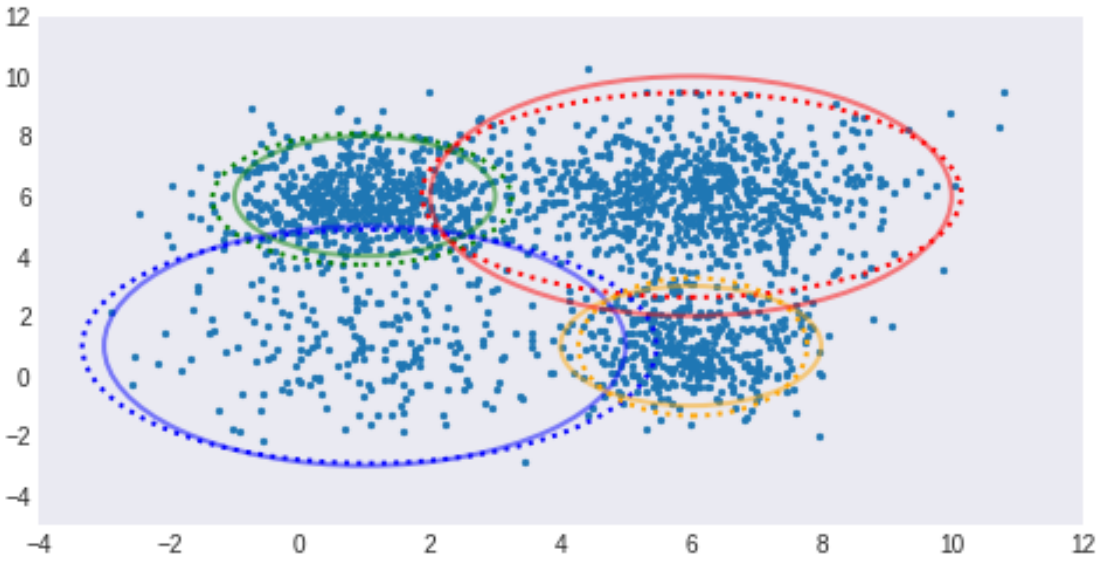
log-likehood:-8755.292



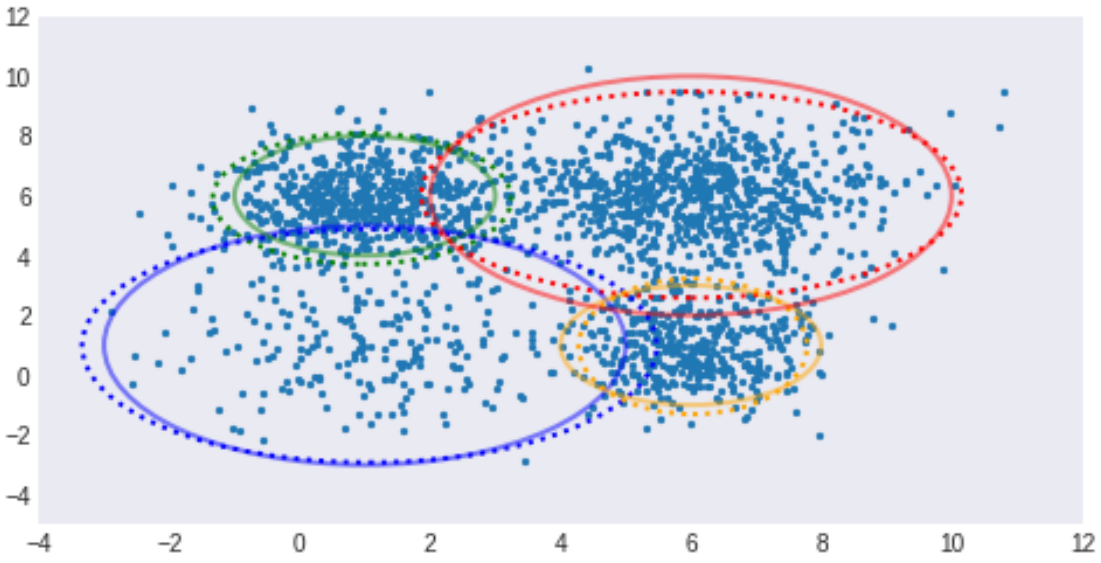
log-likehood:-8753.922



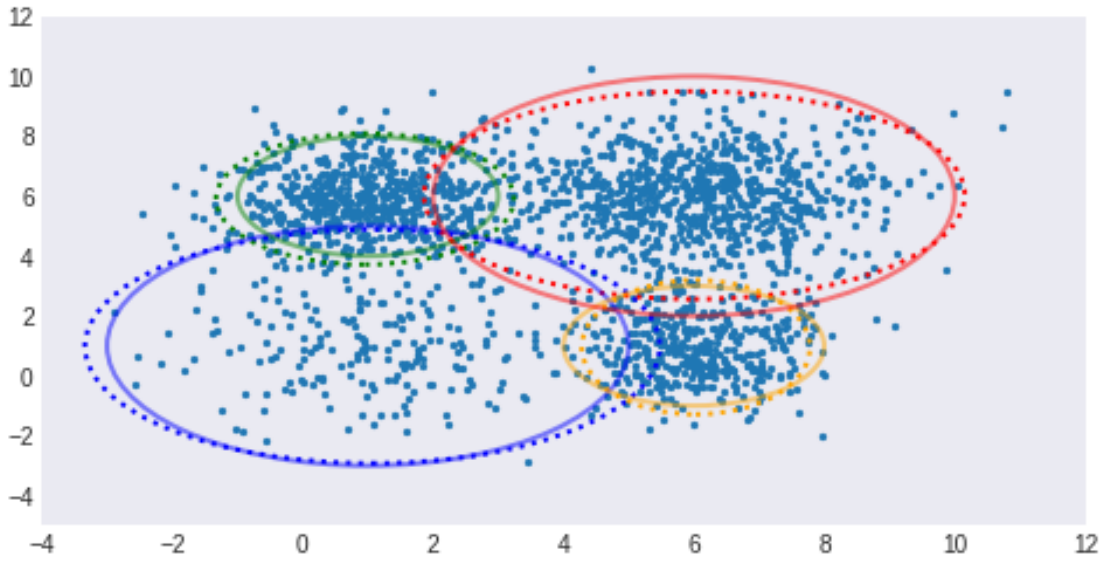
log-likehood:-8753.262



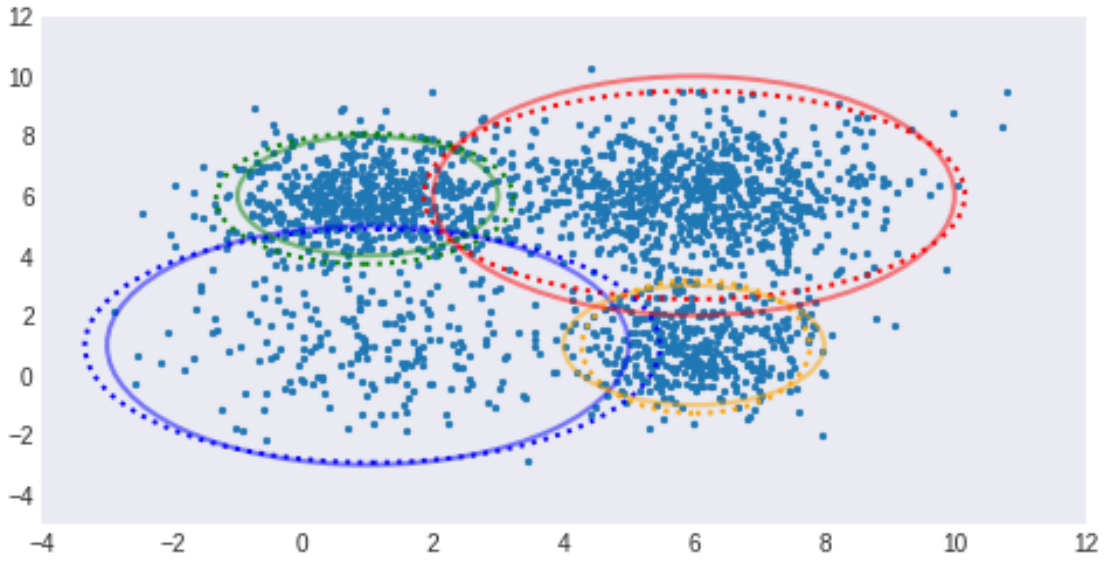
log-likehood:-8752.959



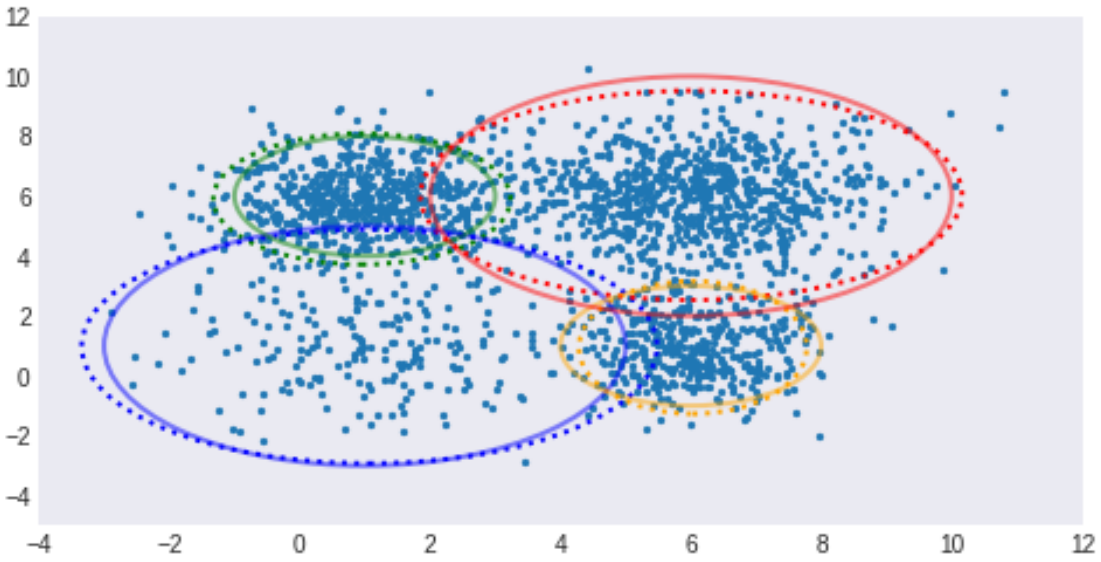
log-likehood:-8752.823



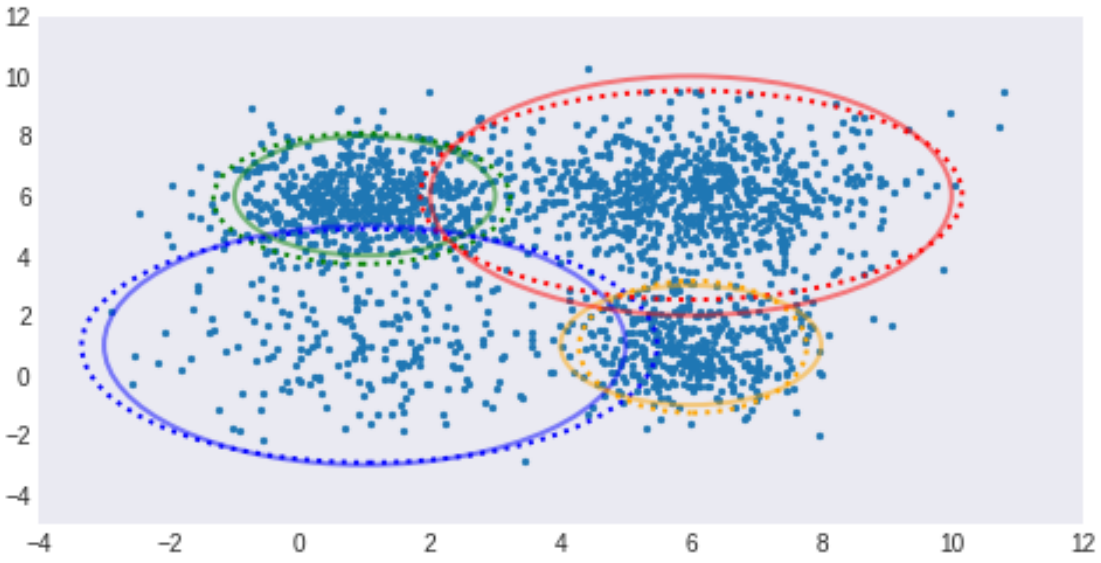
log-likehood:-8752.763



log-likehood:-8752.736

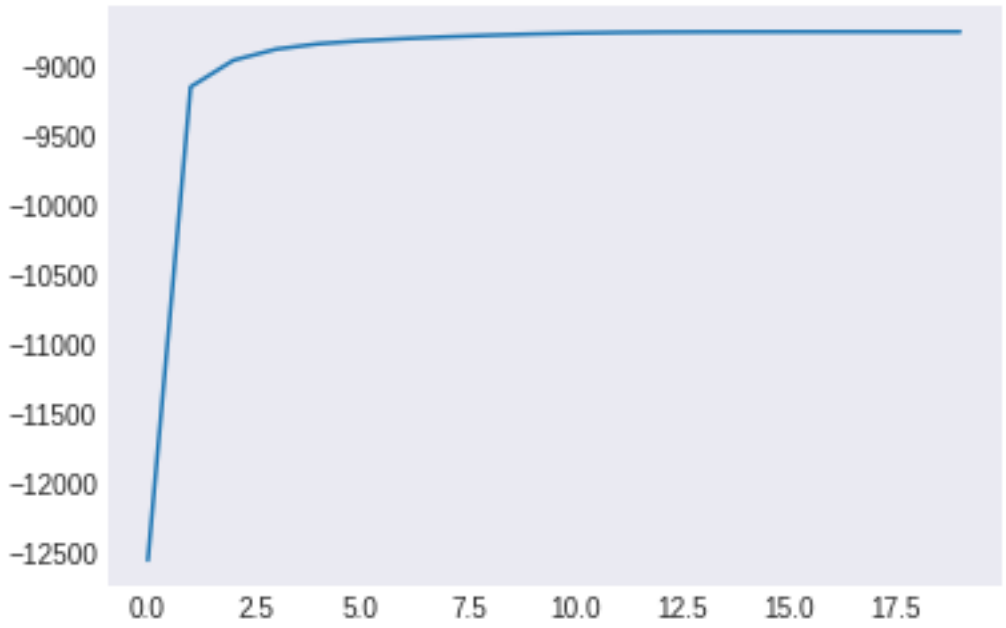


log-likehood:-8752.724



[5]:
```
plt.plot(range(20),log_likelihood)
```

[5]: [<matplotlib.lines.Line2D at 0x7fbb2a14a5e0>]


−9000
−9500
−10000
−10500
−11000
−11500
−12000
−12500
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

[ ]:

Nonparametric methods

December 21, 2021

Parametric: $p(\mathbf{x}|C_i)$ is represented by a single global parametric model.

Nonparametric: $p(\mathbf{x}|C_i)$ cannot be represented by a single parametric model or a mixture model but the data itself.

Let the i.i.d. sample $\mathcal{X} = \{x_i\}_{i=1}^n$ be drawn from some unknown probability density $p(x)$ with $x \in [0, 1]$. We can obtain the histogram estimator $\hat{p}(x)$ for $p(x)$ with bin width specified as h. We define the loss of estimation error in the L^2 space:

$$L(h) = \int_0^1 \left(\left(\hat{p}(x) - p(x)\right)^2 \mathsf{d}x = \int_0^1 \hat{p}^2(x)\mathsf{d}x - 2\int_0^1 \hat{p}(x)p(x)\mathsf{d}x + \int_0^1 p^2(x)\mathsf{d}x.\right.$$

Considering the last term $\int p^2(x)\mathsf{d}x$ is uncorrelated with $\hat{p}(x)$ and replacing the integral with the average, we get

$$L'(h) = \int_0^1 \hat{p}^2(x)\mathsf{d}x - \frac{2}{n}\sum_{i=1}^n \hat{p}(x_i).$$

1. Why we can use $L'(h)$ to replace $L(h)$.
   The reason that we introduce the $L(h)$ is to find a h to minimize the loss of estimation. So it can be an optimization problem, i.e.,

   $$\underset{h}{\mathsf{minimize}} \quad L(h)$$

   Because $\int_0^1 p^2(x)\mathsf{d}x$ is uncorrelated with h, we can drop it. Then we turn to the term $\int_0^1 \hat{p}(x)p(x)\mathsf{d}x$. Because we have

   $$\mathsf{E}\left[\frac{1}{n}\sum_{i=1}^n \hat{p}(x_i)\right] = \frac{1}{n}\sum_{i=1}^n \mathsf{E}\left[\hat{p}(x_i)\right] = \frac{1}{n}\cdot n\cdot\int_0^1 \hat{p}(x)p(x)\mathsf{d}x = \int_0^1 \hat{p}(x)p(x)\mathsf{d}x,$$

   the $\frac{1}{n}\sum_{i=1}^n \hat{p}(x_i)$ is an estimator of $\int_0^1 \hat{p}(x)p(x)\mathsf{d}x$.

2. The expression of $\hat{p}(x)$ is

   $$\hat{p}(x) = \frac{\#\left\{x^{(\ell)} \text{ in the same bin as } x\right\}}{nh}.$$

Since

$$\frac{1}{n}\sum_{i=1}^{n}\hat{p}(x_i)=\frac{1}{n}\sum_{j=1}^{m}\sum_{i\in B_j}\hat{p}(x_i)=\frac{1}{n}\sum_{j=1}^{m}Z_j\hat{p}(x_i)=\frac{1}{n}\sum_{j=1}^{m}\frac{Z_j^2}{nh},$$

we have

$$\begin{aligned}L'(h)&=\int_0^1\hat{p}^2(x)\mathsf{d}x-\frac{2}{n}\sum_{i=1}^{n}\hat{p}(x_i)\\&=\sum_{j=1}^{m}\int_{(j-1)h}^{jh}\hat{p}^2(x)\mathsf{d}x-\frac{2}{n}\sum_{j=1}^{m}\frac{Z_j^2}{nh}\\&=\sum_{j=1}^{m}\int_{(j-1)h}^{jh}(\frac{Z_j}{nh})^2\mathsf{d}x-\frac{2}{n}\sum_{j=1}^{m}\frac{Z_j^2}{nh}\\&=-\frac{1}{n^2h}\sum_{j=1}^{m}Z_j^2.\end{aligned}$$

Because $\sum_{j=1}^{m}Z_j^2\geq\sum_{j=1}^{m}Z_j=n$ and $\sum_{j=1}^{m}Z_j^2\leq\sum_{j=1}^{m}Z_jn=n^2$, we have

$$-\frac{1}{h}\leq L'(h)\leq-\frac{1}{nh}.$$

Therefore, when $h\to 0$, $L'(h)\to-\infty$.

3. Recall the leave-one-out in cross-validation. We define the leave-one-out estimator, i.e.,

$$\hat{p}_{(-i)}(x)=\frac{\#\left\{x^{(\ell)}\text{ in the same bin as }x\text{ without }x_i\right\}}{nh}.$$

In this situation, the loss function is

$$L'(h)=\int_0^1\hat{p}^2(x)\mathsf{d}x-\frac{2}{n}\sum_{i=1}^{n}\hat{p}_{(-i)}(x_i).$$

For the second term, we have

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{n}\hat{p}_{(-i)}(x_i)&=\frac{1}{n}\sum_{j=1}^{m}\sum_{i\in B_j}\hat{p}_{(-i)}(x_i)\\&=\frac{1}{n}\sum_{j=1}^{m}\sum_{i\in B_j}\frac{Z_j-1}{(n-1)h}\\&=\frac{1}{n(n-1)h}\sum_{j=1}^{m}Z_j(Z_j-1).\end{aligned}$$

Therefore, the $L'(h)$ becomes

$$\begin{aligned}
L'(h) &= \int_0^1 \hat{p}^2(x)\mathtt{d}x - \frac{2}{n}\sum_{i=1}^{n}\hat{p}_{(-i)}(x_i) \\
&= \frac{1}{n^2 h}\sum_{j=1}^{m} Z_j^2 - \frac{2}{n(n-1)h}\sum_{j=1}^{m} Z_j(Z_j-1) \\
&= \frac{2}{n(n-1)h}\sum_{j=1}^{m} Z_j + (\frac{1}{n^2 h} - \frac{2}{n(n-1)h})\sum_{j=1}^{m} Z_j^2 \\
&= \frac{2}{(n-1)h} - \frac{n+1}{n^2(n-1)h}\sum_{j=1}^{m} Z_j^2 \\
&= \frac{2n^2 - (n+1)\sum_{j=1}^{m} Z_j^2}{n^2(n-1)h}.
\end{aligned}$$

And $2n^2 - (n+1)\sum_{j=1}^{m} Z_j^2 \leq 2n^2 - (n+1)n^2 = n^2(1-n) \leq 0$, when $n \geq 1$. So $L'(h) \to -\infty$ as $h \to 0$.