School of Data and Computer Science, Sun Yat-sen University

E12 EM Algorithm (C++/Python)

18340215 张天祎

December 4, 2020

目录

1 Iris Dataset

Data Set Information:

This is perhaps the best known database to be found in the pattern recognition literature. Fisher's paper(The use of multiple measurements in taxonomic problems) is a classic in the field and is referenced frequently to this day. The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.

Attribute Information:

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm
5. class:
   - Iris Setosa
   - Iris Versicolour
   - Iris Virginica

Predicted attribute: class of iris plant. This is an exceedingly simple domain. More info please refer to "iris.names" file.

2 EM

2.1 The Gaussian Distribution

The Gaussian, also known as the normal distribution, is a widely used model for the distribution of continuous variables. In the case of a single variable x, the Gaussian distribution can be written in the form

$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}}\exp\{-\frac{1}{2\sigma^2}(x-\mu)^2\} \tag{2.1.1}$$

where μ is the mean and σ^2 is the variance.

For a D-dimensional vector $\mathbf{x}$, the multivariate Gaussian distribution takes the form

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}}\frac{1}{|\boldsymbol{\Sigma}|^{1/2}}\exp\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}\} \tag{2.1.2}$$

where $\boldsymbol{\mu}$ is a D-dimensional mean vector, $\boldsymbol{\Sigma}$ is a $D \times D$ covariance matrix, and $|\boldsymbol{\Sigma}|$ denotes the determinant of $|\boldsymbol{\Sigma}|$.

2.2 Mixtures of Gaussians

2.2.1 Introduction

While the Gaussian distribution has some important analytical properties, it suffers from significant limitations when it comes to modelling real data sets. Consider the example shown in Figure 1. This is known as the 'Old Faithful' data set, and comprises 272 measurements of the eruption of the Old Faithful geyser at Yel-lowstone National Park in the USA. Each measurement comprises the duration of the eruption in minutes (horizontal axis) and the time in minutes to the next eruption (vertical axis). We see that the data set forms two dominant clumps, and that a simple Gaussian distribution is unable to capture this structure, whereas a linear superposition of two Gaussians gives a better characterization of the data set.

Example of a Gaussian mixture distribution in one dimension showing three Gaussians (each scaled by a coefficient) in blue and their sum in red.

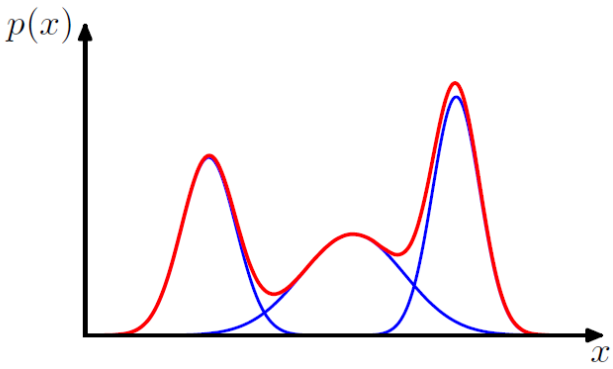


图 1: Example of a Gaussian mixture distribution

Such superpositions, formed by taking linear combinations of more basic distributions such as Gaussians, can be formulated as probabilistic models known as *mixture distributions*. In Figure 1 we see that a linear combination of Gaussians can give rise to very complex densities. By using a sufficient number of Gaussians, and by adjusting their means and covariances as well as the coefficients in the linear combination, almost any continuous density can be approximated to arbitrary accuracy.

We therefore consider a superposition of K Gaussian densities of the form

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{2.2.1}$$

which is called a mixture of Gaussians. Each Gaussian density $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ is called a component of the mixture and has its own mean $\boldsymbol{\mu}_k$ and covariance $\boldsymbol{\Sigma}_k$.

The parameters π_k in (2.2.1) are called *mixing coefficients*. If we integrate both sides of (2.2.1) with respect to $\mathbf{x}$, and note that both $p(\mathbf{x})$ and the individual Gaussian components are normalized,

we obtain

$$\sum_{k=1}^{K} \pi_k = 1. \tag{2.2.2}$$

Also, the requirement that $p(\mathbf{x}) \geq 0$, together with $\mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k) \geq 0$, implies $\pi_k \geq 0$ for all k. Combining this with condition (2.2.2) we obtain

$$0 \leq \pi_k \leq 1. \tag{2.2.3}$$

We therefore see that the mixing coefficients satisfy the requirements to be probabilities.

From the sum and product rules, the marginal density is given by

$$p(\mathbf{x}) = \sum_{k=1}^{K} p(k)p(\mathbf{x}|k) \tag{2.2.4}$$

which is equivalent to (2.2.1) in which we can view $\pi_k = p(k)$ as the prior probability of picking the k^{th} component, and the density $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = p(\mathbf{x}|k)$ as the probability of $\mathbf{x}$ conditioned on k. From Bayes' theorem these are given by

$$\gamma_k(\mathbf{x}) = p(k|\mathbf{x}) = \frac{p(k)p(\mathbf{x}|k)}{\sum_l p(l)p(\mathbf{x}|l)} = \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_l \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)}. \tag{2.2.5}$$

The form of the Gaussian mixture distribution is governed by the parameters π, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, where we have used the notation $\boldsymbol{\pi} = \{\pi_1, ..., \pi_K\}$, $\boldsymbol{\mu} = \{\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_k\}$ and $\boldsymbol{\Sigma} = \{\boldsymbol{\Sigma}_1, ..., \boldsymbol{\Sigma}_K\}$. One way to set the values of there parameters is to use maximum likelihood. From (2.2.1) the log of the likelihood function is given by

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln\{\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\} \tag{2.2.6}$$

where $X = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$. One approach to maximizing the likelihood function is to use iterative numerical optimization techniques. Alternatively we can employ a powerful framework called expectation maximization (EM).

2.2.2 About Latent Variables

We now turn to a formulation of Gaussian mixtures in terms of discrete *latent* variables. This will provide us with a deeper insight into this important distribution, and will also serve to motivate the expectation-maximization (EM) algorithm.

Recall from (2.2.1) that the Gaussian mixture distribution can be written as a linear superposition of Gaussians in the form

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{2.2.7}$$

Let us introduce a K-dimensional binary random variable $\mathbf{z}$ having a 1-of-K representation in which a particular element z_k is equal to 1 and all other elements are equal to 0. The values of z_k therefore satisfy $z_k \in \{0, 1\}$ and $\Sigma_k z_k = 1$, and we see that there are K possible states for the vector $\mathbf{z}$ according to which element is nonzero. We shall define the joint distribution $p(\mathbf{x}, \mathbf{z})$ in terms of a marginal distribution $p(\mathbf{z})$ and a conditional distribution $p(\mathbf{x}|\mathbf{z})$. The marginal distribution over $\mathbf{z}$ is specified in terms of the mixing coefficients π_k, such that

$$p(z_k = 1) = \pi_k \tag{2.2.8}$$

where the parameters $\{\pi_k\}$ must satisfy

$$0 \leq \pi_k \leq 1 \tag{2.2.9}$$

together with

$$\sum_{k=1}^{K} \pi_k = 1 \tag{2.2.10}$$

in order to be valid probabilities. Because $\mathbf{z}$ uses a 1-of-K representation, we can also write this distribution in the form

$$p(\mathbf{z}) = \prod_{k=1}^{K} \pi_k^{z_k}. \tag{2.2.11}$$

Similarly, the conditional distribution of $\mathbf{x}$ given a particular value for $\mathbf{z}$ is a Gaussian

$$p(\mathbf{x}|z_k = 1) = (\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{2.2.12}$$

which can also be written in the form

$$p(\mathbf{x}|\mathbf{z}) = \prod_{k=1}^{K} p(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_k}. \tag{2.2.13}$$

The joint distribution is given by $p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$, and the marginal distribution of $\mathbf{x}$ is then obtained by summing the joint distribution over all possible states of $\mathbf{z}$ to give

$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{z})p(\mathbf{x}|\mathbf{z}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{2.2.14}$$

where we have made use of (2.2.12) and (2.2.13). Thus the marginal distribution of $\mathbf{x}$ is a Gaussian mixture of the form (2.2.7). If we have several observations $\mathbf{x_1}, ..., \mathbf{x_N}$, then, because we have represented the marginal distribution in the form $p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{x}, \mathbf{z})$, it follows that for every observed data point $\mathbf{x}_n$ there is a corresponding latent variable $\mathbf{z}_n$.

We have therefore found an equivalent formulation of the Gaussian mixture involving an explicit latent variable. It might seem that we have not gained much by doing so. However, we are now able to work with the joint distribution $p(\mathbf{x}, \mathbf{z})$ instead of the marginal distribution $p(\mathbf{x})$, and this will lead

to significant simplifications, most notably through the introduction of the expectation-maximization (EM) algorithm.

Another quantity that will play an important role is the conditional probability of $\mathbf{z}$ given $\mathbf{x}$. We shall use $\gamma(z_k)$to denote $p(z_k = 1|\mathbf{x})$, whose value can be found using Bayes' theorem

$$\gamma(z_k) = p(z_k = 1|\mathbf{x}) = \frac{p(z_k = 1)p(\mathbf{x}|z_k = 1)}{\sum_{j=1}^{K} p(z_j = 1)p(\mathbf{x}|z_j = 1)} = \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \tag{2.2.15}$$

We shall view π_k as the prior probability of $z_k = 1$, and the quantity $\gamma(z_k)$ as the corresponding posterior probability once we have observed $\mathbf{x}$. As we shall see later, $\gamma(z_k)$ can also be viewed as the responsibility that component k takes for 'explaining' the observation $\mathbf{x}$.

2.3 EM for Gaussian Mixtures

Initially, we shall motivate the EM algorithm by giving a relatively informal treatment in the context of the Gaussian mixture model.

Let us begin by writing down the conditions that must be satisfied at a maximum of the likelihood function. Setting the derivatives of $\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ with respect to the means $\boldsymbol{\mu}_k$ of the Gaussian components to zero, we obtain

$$0 = -\sum_{n=1}^{n} \underbrace{\frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}}_{\gamma(z_{nk})} \sum_k (\mathbf{x}_n - \boldsymbol{\mu}_k) \tag{2.3.1}$$

Multiplying by $\boldsymbol{\Sigma}_k^{-1}$ (which we assume to be nonsingular) and rearranging we obtain

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n \tag{2.3.2}$$

where we have defined

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk}). \tag{2.3.3}$$

We can interpret N_k as the effective number of points assigned to cluster k. Note carefully the form of this solution. We see that the mean $\boldsymbol{\mu}_k$ for the k^{th} Gaussian component is obtained by taking a weighted mean of all of the points in the data set, in which the weighting factor for data point $\mathbf{x}_n$ is given by the posterior probability $\gamma(z_{nk})$ that component k was responsible for generating $\mathbf{x}_n$.

If we set the derivative of $\ln(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ with respect to $\boldsymbol{\Sigma}_k$ to zero, and follow a similar line of reasoning, making use of the result for the maximum likelihood for the covariance matrix of a single Gaussian, we obtain

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}} \tag{2.3.4}$$

which has the same form as the corresponding result for a single Gaussian fitted to the data set, but again with each data point weighted by the corresponding posterior probability and with the denominator given by the effective number of points associated with the corresponding component.

Finally, we maximize $\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ with respect to the mixing coefficients π_k. Here we must take account of the constraint $\sum_{k=1}^{K} \pi_k = 1$. This can be achieved using a Lagrange multiplier and maximizing the following quantity

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) + \lambda(\sum_{k=1}^{K} \pi_k - 1) \tag{2.3.5}$$

which gives

$$0 = \sum_{n=1}^{N} \frac{\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \tag{2.3.6}$$

where again we see the appearance of the responsibilities. If we now multiply both sides by π_k and sum over k making use of the constraint $\sum_{k=1}^{K} \pi_k = 1$, we find $\lambda = -N$. Using this to eliminate λ and rearranging we obtain

$$\pi_k = \frac{N_k}{N} \tag{2.3.7}$$

so that the mixing coefficient for the k^{th} component is given by the average responsibility which that component takes for explaining the data points.

2.4 EM Algorithm

Given a Gaussian mixture model, the goal is to maximize the likelihood function with respect to the parameters (comprising the means and covariances of the components and the mixing coefficients).

1. Initialize the means $\boldsymbol{\mu}_k$, covariances $\boldsymbol{\Sigma}_k$ and mixing coefficients π_k, and evaluate the initial value of the log likelihood.
2. **E step**. Evaluate the responsibilities using the current parameter values

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \tag{2.4.1}$$

3. **M step**. Re-estimate the parameters using the current responsibilities

$$\boldsymbol{\mu}_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n \tag{2.4.2}$$

$$\boldsymbol{\Sigma}_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \boldsymbol{\mu}_k^{new})(\mathbf{x}_n - \boldsymbol{\mu}_k^{new})^{\mathrm{T}} \tag{2.4.3}$$

$$\pi_k^{new} = \frac{N_k}{N} \tag{2.4.4}$$

where

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk}). \tag{2.4.5}$$

4. Evaluate the log likelihood

$$\ln p(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_{n=1}^{N} \ln\{\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\} \tag{2.4.6}$$

and check for convergence of either the parameters or the log likelihood. If the convergence criterion is not satisfied return to step 2.

3 Tasks

- Assume that score vectors of teams in the same class are normally distributed, we can thus adopt the Gaussian mixture model. Please classify the teams into 3 classes by using EM algorithm. If necessary, you can refer to page 430-439 in the book Pattern Recognition and Machine Learning.pdf and the website https://blog.csdn.net/jinping_shi/article/details/59613054 which is a Chinese translation.
- You should show the values of these parameters: $\boldsymbol{\gamma}$, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. You can also visualize the values of these parameters to see the convergence of EM algorithm. Give the results and simple analysis in your report.
- Please submit a file named E12_YourNumber.pdf and send it to ai_2020@foxmail.com

4 Codes and Results

```
1  import csv
2  import numpy as np
3  import matplotlib.pyplot as plt
4  from collections import Counter
5
6  EPOCH = 50
7
8  def load_data():
9          with open("iris.data","r") as f:
10                 reader = list(csv.reader(f))
11                 return [[float(j) for j in  i[:-1]] for i in reader[:-1]], [
                       j[-1] for j in reader[:-1]]
12
```

```
13
14  def N(x, mu, sigma):
15          D = len(x[0])
16          delta = x - mu
17          return (1/(((2*np.pi)**(D/2))*(np.linalg.det(sigma)**0.5))*(np.exp
                (-1/2*delta*(sigma.I)*delta.T)))[0][0]
18
19
20  def likelihood(data, pi, mu, sigma, n, K):
21          _sum = 0
22          for i in range(n):
23                  sum = 0
24                  for j in range(K):
25                          sum += pi[j] * N(data[i], mu[j], sigma[j])
26                  _sum += np.log(sum)
27          return _sum[0, 0]
28
29
30  def EM(data, K):
31          paint = [[],[]]
32          data = np.matrix(data)
33          n, m = np.shape(data)
34
35          # 初始化
36          pi = [1/K for i in range(K)]
37          mu = [data[np.random.randint(0, n)] for i in range(K)]
38          # mu = [data[0], data[50], data[100]]
39          sigma = [np.matrix(np.eye(m)) for i in range(K)]
40          gamma = np.matrix(np.zeros((n,K)))
41          hold = likelihood(data, pi, mu, sigma, n, K)
42          _likelihood = 0
43
44          epoch = 0
45          while(epoch < EPOCH and abs(hold-_likelihood)>np.exp(-3)):
46                  hold = _likelihood
47                  # E
48                  for i in range(n):
```

```
49                          sum = 0
50                          for j in range(K):
51                                  gamma[i , j] = pi[j] * N(data[i], mu[j],
                                        sigma[j])
52                                  sum += gamma[i, j]
53                          for j in range(K):
54                                  gamma[i, j] /= sum
55
56                  sum_gamma = np.sum(gamma, axis=0) # N_new
57                  # M
58                  for i in range(K):
59                          mu[i] = np.matrix(np.zeros((1, m)))
60                          sigma[i] = np.matrix(np.zeros((m,m)))
61
62                          for j in range(n):
63                                  mu[i] += gamma[j, i] * data[j]
64                          mu[i] /= sum_gamma[0, i] # mu_new
65
66                          for j in range(n):
67                                  sigma[i] += gamma[j, i] * (data[j] - mu[i]).
                                        T * (data[j] - mu[i])
68                          sigma[i] /= sum_gamma[0, i] # sigma_new
69
70                          pi[i] = sum_gamma[0 , i] / n # pi_new
71                  _likelihood = likelihood(data, pi, mu, sigma, n, K)
72                  epoch += 1
73                  paint[0].append(epoch)
74                  paint[1].append(_likelihood)
75                  # print("EPOCH:",epoch)
76                  # print(Counter([np.argmax(gamma[i]) + 1 for i in range(n)])
                        )
77                  # print(_likelihood)
78          print("gamma:\n", gamma)
79          print("mu:\n", mu)
80          print("sigma:\n", sigma)
81          print("likelihood:", _likelihood)
82          return [np.argmax(gamma[i]) + 1 for i in range(n)], paint
```

```
83
84  def main():
85          data, labels = load_data()
86          rst_labels, paint = EM(data, len(Counter(labels)))
87          plt.scatter(paint[0], paint[1])
88          plt.show()
89          count = 0
90          map1 = {}
91          for i, label in enumerate(labels):
92                  if map1.get(label, 0):
93                          map1[label].append(i)
94                  else:
95                          map1[label] = [i]
96          set_list1 = [set(i) for i in map1.values()]
97          map2 = {}
98          for i, label in enumerate(rst_labels):
99                  if map2.get(label, 0):
100                         map2[label].append(i)
101                 else:
102                         map2[label] = [i]
103         set_list2 = [set(i) for i in map2.values()]
104         for _set in set_list2:
105                 count += max([len(_set & __set) for __set in set_list1])
106
107         print("Accuracy:",count/len(labels))
108
109 if __name__ == "__main__":
110         main()
```

- 结果如下图

```
1           gamma:
2           [[1.00000000e+000 1.17403794e-042 2.14774544e-035]
3           [1.00000000e+000 4.87084011e-030 9.14251908e-029]
4           [1.00000000e+000 8.08843454e-035 1.60603745e-030]
5           [1.00000000e+000 9.82288781e-031 1.11645704e-026]
6           [1.00000000e+000 5.42844715e-045 9.63104448e-036]
7           [1.00000000e+000 1.07825805e-043 1.20463999e-035]
8           [1.00000000e+000 7.49768699e-035 2.56920202e-029]
```

```
[1.00000000e+000 6.12089401e-039 2.21203143e-032]
[1.00000000e+000 1.14915299e-026 1.18427151e-024]
[1.00000000e+000 1.37110415e-034 2.80948241e-029]
[1.00000000e+000 6.22160546e-048 1.87689095e-038]
[1.00000000e+000 1.93169778e-037 1.21511136e-029]
[1.00000000e+000 7.66625754e-033 6.42928732e-029]
[1.00000000e+000 4.36341389e-033 3.05308820e-028]
[1.00000000e+000 1.48576561e-060 9.27752889e-049]
[1.00000000e+000 7.97867230e-062 3.79548753e-047]
[1.00000000e+000 5.80898510e-048 9.10974961e-041]
[1.00000000e+000 3.53018715e-039 2.74606522e-034]
[1.00000000e+000 2.32550120e-045 1.37632987e-037]
[1.00000000e+000 5.00243205e-046 2.44179251e-036]
[1.00000000e+000 1.65598722e-037 8.56253698e-032]
[1.00000000e+000 1.22084531e-039 8.31073269e-034]
[1.00000000e+000 1.13141417e-045 8.28488441e-036]
[1.00000000e+000 9.50824166e-025 3.16258441e-025]
[1.00000000e+000 3.17940184e-034 5.13009634e-025]
[1.00000000e+000 2.18803565e-028 9.13699793e-027]
[1.00000000e+000 4.19245957e-031 1.18950740e-028]
[1.00000000e+000 5.34619196e-042 8.49704061e-035]
[1.00000000e+000 2.85204302e-040 6.52903248e-035]
[1.00000000e+000 2.14548119e-032 5.05098034e-027]
[1.00000000e+000 2.32109126e-030 1.03264738e-026]
[1.00000000e+000 3.93361690e-033 2.00253557e-032]
[1.00000000e+000 2.18313608e-063 2.27426133e-042]
[1.00000000e+000 2.36900665e-064 9.35906064e-047]
[1.00000000e+000 1.37110415e-034 2.80948241e-029]
[1.00000000e+000 2.33952866e-036 1.48288835e-033]
[1.00000000e+000 2.73405042e-044 4.74677813e-039]
[1.00000000e+000 1.37110415e-034 2.80948241e-029]
[1.00000000e+000 2.29636144e-029 1.84534328e-026]
[1.00000000e+000 3.57398845e-039 4.92061306e-033]
[1.00000000e+000 1.11862345e-039 1.28367846e-034]
[1.00000000e+000 2.00543592e-014 6.38180681e-020]
[1.00000000e+000 1.27854498e-033 4.21741018e-028]
[1.00000000e+000 2.03948557e-026 8.65670396e-026]
```

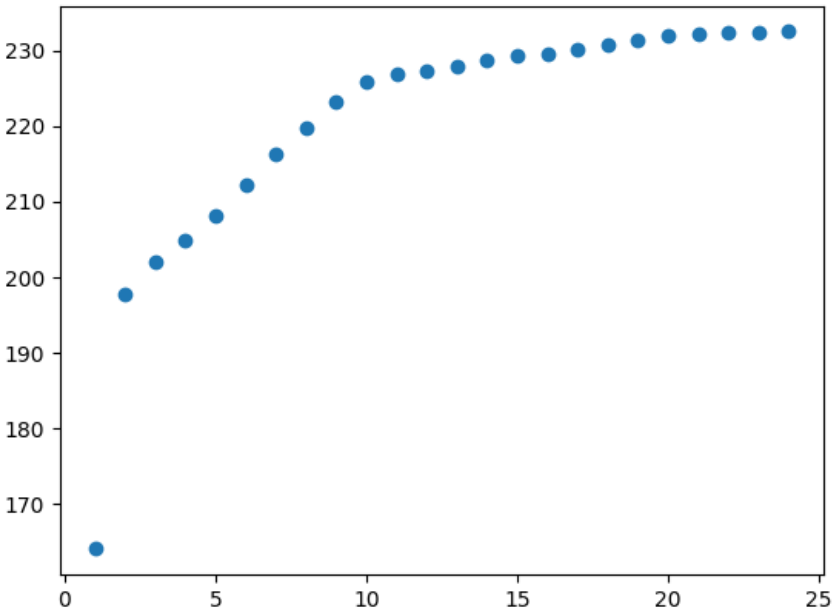
```
[1.00000000e+000 1.75528979e-037 4.64824170e-029]
[1.00000000e+000 6.70473016e-027 4.30222592e-027]
[1.00000000e+000 5.91120592e-049 4.72559536e-036]
[1.00000000e+000 9.46119825e-034 1.00682161e-028]
[1.00000000e+000 1.03647822e-047 8.62113404e-038]
[1.00000000e+000 2.61872871e-037 1.53661523e-032]
[1.84001479e-094 9.99789281e-001 2.10718772e-004]
[9.90304920e-086 9.98880806e-001 1.11919399e-003]
[5.06006038e-107 9.95538027e-001 4.46197310e-003]
[1.33806415e-065 9.47130247e-001 5.28697531e-002]
[1.12596125e-094 9.77615834e-001 2.23841662e-002]
[8.78333018e-081 9.72460582e-001 2.75394182e-002]
[6.71515507e-096 9.88362263e-001 1.16377367e-002]
[5.03280754e-035 9.99888763e-001 1.11237468e-004]
[1.50256782e-088 9.98917311e-001 1.08268881e-003]
[6.36433540e-062 9.71646359e-001 2.83536408e-002]
[2.03170768e-043 9.98587381e-001 1.41261913e-003]
[2.22658824e-074 9.94069244e-001 5.93075580e-003]
[1.80198439e-062 9.99381583e-001 6.18416500e-004]
[1.20137288e-092 9.71700227e-001 2.82997727e-002]
[3.48813618e-048 9.99857870e-001 1.42129629e-004]
[2.17486745e-081 9.99908591e-001 9.14090501e-005]
[3.11533378e-085 9.32520562e-001 6.74794379e-002]
[9.60436935e-059 9.97646534e-001 2.35346609e-003]
[3.83448229e-095 5.50508362e-003 9.94494916e-001]
[3.73821507e-056 9.99619355e-001 3.80644956e-004]
[4.68890908e-108 6.70338797e-002 9.32966120e-001]
[6.12696537e-064 9.99885178e-001 1.14821702e-004]
[9.01144134e-110 6.45275451e-002 9.35472455e-001]
[1.29358193e-087 9.26251711e-001 7.37482890e-002]
[2.41392014e-075 9.99836523e-001 1.63477422e-004]
[4.36652355e-082 9.99790930e-001 2.09070203e-004]
[1.22366723e-102 9.90666160e-001 9.33383986e-003]
[9.94388012e-118 4.14376972e-001 5.85623028e-001]
[3.76668026e-087 9.70532931e-001 2.94670695e-002]
[1.92081721e-040 9.99992040e-001 7.96032397e-006]
[1.83915106e-053 9.99623213e-001 3.76787374e-004]
```

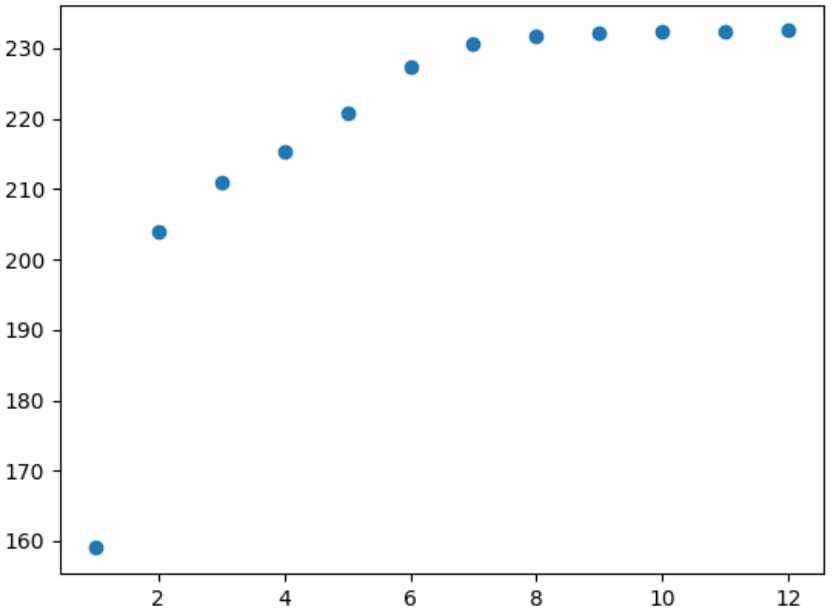
```
[7.12990765e-048 9.99871295e-001 1.28705153e-004]
[1.37751935e-057 9.99872341e-001 1.27658672e-004]
[4.83987897e-119 1.02272345e-002 9.89772766e-001]
[5.57555736e-085 8.61654603e-001 1.38345397e-001]
[8.76589030e-086 9.88947645e-001 1.10523552e-002]
[7.04255332e-097 9.98027657e-001 1.97234288e-003]
[8.06071121e-085 9.43676290e-001 5.63237101e-002]
[8.99181545e-064 9.98827874e-001 1.17212611e-003]
[2.43809907e-064 9.91813716e-001 8.18628361e-003]
[4.78291021e-075 9.48379795e-001 5.16202050e-002]
[1.66529926e-087 9.92313170e-001 7.68682975e-003]
[5.39406670e-062 9.99622348e-001 3.77652023e-004]
[6.06079670e-036 9.99915884e-001 8.41162632e-005]
[3.90267212e-070 9.94725270e-001 5.27473010e-003]
[2.66470166e-064 9.97964916e-001 2.03508394e-003]
[1.36359835e-068 9.98315792e-001 1.68420779e-003]
[2.93881845e-074 9.99608239e-001 3.91761242e-004]
[3.06843223e-029 9.99963398e-001 3.66018518e-005]
[5.50690382e-066 9.98861982e-001 1.13801811e-003]
[2.25594941e-207 3.75733216e-016 1.00000000e+000]
[1.35759779e-131 2.57472619e-007 9.99999743e-001]
[5.16898520e-186 8.88653551e-009 9.99999991e-001]
[9.62381827e-152 7.94195374e-005 9.99920580e-001]
[4.43466894e-182 3.28204861e-011 1.00000000e+000]
[6.20753949e-234 1.20840359e-010 1.00000000e+000]
[3.52866352e-097 1.96751663e-005 9.99980325e-001]
[1.72357768e-200 4.50732889e-007 9.99999549e-001]
[1.01956168e-170 1.81957191e-008 9.99999982e-001]
[2.08683720e-212 2.07965313e-012 1.00000000e+000]
[6.91734308e-133 1.03436526e-004 9.99896563e-001]
[1.27204993e-143 4.28490744e-007 9.99999572e-001]
[3.99852421e-162 1.73798017e-008 9.99999983e-001]
[6.30175953e-134 8.59581741e-012 1.00000000e+000]
[3.77581322e-160 2.68368589e-021 1.00000000e+000]
[1.48156336e-159 3.36990459e-012 1.00000000e+000]
[4.22229639e-146 1.27764808e-003 9.98722352e-001]
[7.28761281e-233 1.02021003e-005 9.99989798e-001]
```

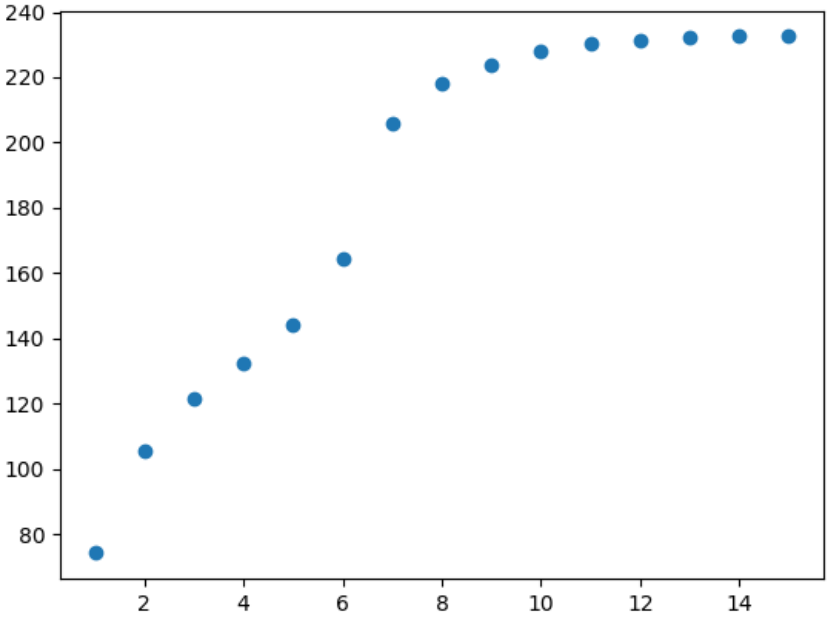
```
120             [7.23626581e-272 7.79001477e-021 1.00000000e+000]
121             [2.08883925e-116 1.04014109e-004 9.99895986e-001]
122             [8.84468821e-182 1.74667948e-011 1.00000000e+000]
123             [1.54948907e-126 7.97325420e-009 9.99999992e-001]
124             [6.29729071e-241 9.26578657e-012 1.00000000e+000]
125             [2.74673123e-119 1.01802344e-004 9.99898198e-001]
126             [3.13083726e-168 4.21123046e-006 9.99995789e-001]
127             [3.84727550e-176 1.13602637e-003 9.98863974e-001]
128             [3.80183871e-113 6.09930821e-004 9.99390069e-001]
129             [5.31451001e-115 8.69712914e-003 9.91302871e-001]
130             [7.49036061e-168 7.77650493e-011 1.00000000e+000]
131             [2.96528715e-160 6.47311966e-003 9.93526880e-001]
132             [1.46666834e-194 9.61082961e-008 9.99999904e-001]
133             [3.91336174e-205 7.78107402e-003 9.92218926e-001]
134             [2.25578811e-173 1.86091435e-013 1.00000000e+000]
135             [1.11346508e-115 2.64431846e-001 7.35568154e-001]
136             [3.76174421e-140 4.24873748e-005 9.99957513e-001]
137             [1.34509763e-213 1.26120964e-013 1.00000000e+000]
138             [2.07593678e-178 1.27905863e-012 1.00000000e+000]
139             [2.28614722e-144 3.68163692e-003 9.96318363e-001]
140             [2.59555507e-110 8.28129241e-003 9.91718708e-001]
141             [4.39091799e-156 1.96412367e-007 9.99999804e-001]
142             [4.06728882e-183 6.81486756e-016 1.00000000e+000]
143             [4.91933038e-153 1.36414331e-013 1.00000000e+000]
144             [1.35759779e-131 2.57472619e-007 9.99999743e-001]
145             [3.04730284e-192 1.74039846e-011 1.00000000e+000]
146             [2.34586443e-192 1.32061163e-016 1.00000000e+000]
147             [2.91006516e-158 1.39384083e-014 1.00000000e+000]
148             [3.80591191e-131 7.18609263e-009 9.99999993e-001]
149             [2.78591688e-140 2.57303408e-006 9.99997427e-001]
150             [7.46351045e-162 3.53070478e-010 1.00000000e+000]
151             [6.61155367e-124 2.56404749e-003 9.97435953e-001]]
152           mu:
153             [matrix([[5.006, 3.418, 1.464, 0.244]]), matrix([[5.91721992,
                  2.77806046, 4.20565461, 1.29858127]]), matrix([[6.54649015,
                  2.94951281, 5.48387957, 1.98742145]])]
154           sigma:
```

```
155             [matrix([[0.121764, 0.098292, 0.015816, 0.010336],
156                      [0.098292, 0.142276, 0.011448, 0.011208],
157                      [0.015816, 0.011448, 0.029504, 0.005584],
158                      [0.010336, 0.011208, 0.005584, 0.011264]]), matrix
                             ([[0.27553297, 0.09660565, 0.18553291,
                             0.05481874],
159                      [0.09660565, 0.0925449 , 0.09102806, 0.04299919],
160                      [0.18553291, 0.09102806, 0.20248081, 0.06176557],
161                      [0.05481874, 0.04299919, 0.06176557, 0.03235987]]),
                             matrix([[0.38748036, 0.09223532, 0.30243962,
                             0.06083684],
162                      [0.09223532, 0.11041221, 0.08382644, 0.05572773],
163                      [0.30243962, 0.08382644, 0.32580136, 0.07267231],
164                      [0.06083684, 0.05572773, 0.07267231, 0.08479206]])]
165             likelihood: 232.51276198223593
166             Accuracy: 0.9666666666666667
```

- EM 算法是初始化敏感的，不同的初始化结果可能导致不同的结果。
- 其中主要是初始 μ 值对结果影响较大，这里使用的是随机初始化，有可能导致还没有找到结果提前终止，但大多数情况表现良好。也可以使用固定值初始化，如初始化部分代码的注释部分那样。
- 不同的初始化对收敛速度的影响也很大，下面三图横轴表示迭代次数，纵轴是最大似然对数，三例都收敛到了最优的结果，精度 0.967，但收敛速度差别很大。


230
220
210
200
190
180
170
0
5
10
15
20
25


230
220
210
200
190
180
170
160
2
4
6
8
10
12


240
220
200
180
160
140
120
100
80
2
4
6
8
10
12
14