期末專題: 雙變量函數的繪圖 & 多變量函數的參數估計與極值計算

姓名: 周亨昆

學號: 410878016

系級: 統計三

目標

- ## 習題1 混合常態參數估計 Normal Mixture
- ## 習題2 限制式條件的最大值問題 Constraint optimization

習題 1：混合常態參數估計 Normal Mixture

工作敘述:

混合常態的參數估計。即，

$\max_{\Omega=\{\pi1,\pi2,\mu1,\sigma_1^2,\mu2,\sigma_2^2|\pi1+\pi2=1,\pi1,\pi2,\sigma_1^2,\sigma_2^2>0\}} L(\Omega)$

其中對數聯合概似函數為 $L(\Omega) = \ln \Pi_{i=1}^N(\pi_1 f(x_i|\mu_1,\sigma_1^2) + \pi_2 f(x_i|\mu_2,\sigma_2^2))$ $= \sum_{i=1}^N \ln(\pi_1 f(x_i|\mu_1,\sigma_1^2) + \pi_2 f(x_i|\mu_2,\sigma_2^2))$

$f(x|\mu,\sigma^2)$ 為常態分配的機率密度函數。自行產生適用的資料並運用 minimize估計參數 $\Omega = \{\pi_1,\mu_1,\sigma_1^2,\mu_2,\sigma_2^2\}$。

環境設定如下:

- #### 自行設定資料生成的參數 $\Omega = \{\pi_1,\mu_1,\sigma_1^2,\mu_2,\sigma_2^2\}$，設計兩組情況：一組之 μ_1,μ_2 較接近（視覺上好像只有一組），另一組分開較遠些（視覺上看出兩個常態混合）。

- #### 總樣本數 n= 50, 100, 300, 500, 1000, 10000。藉以評估樣本數大小對估計值的影響。

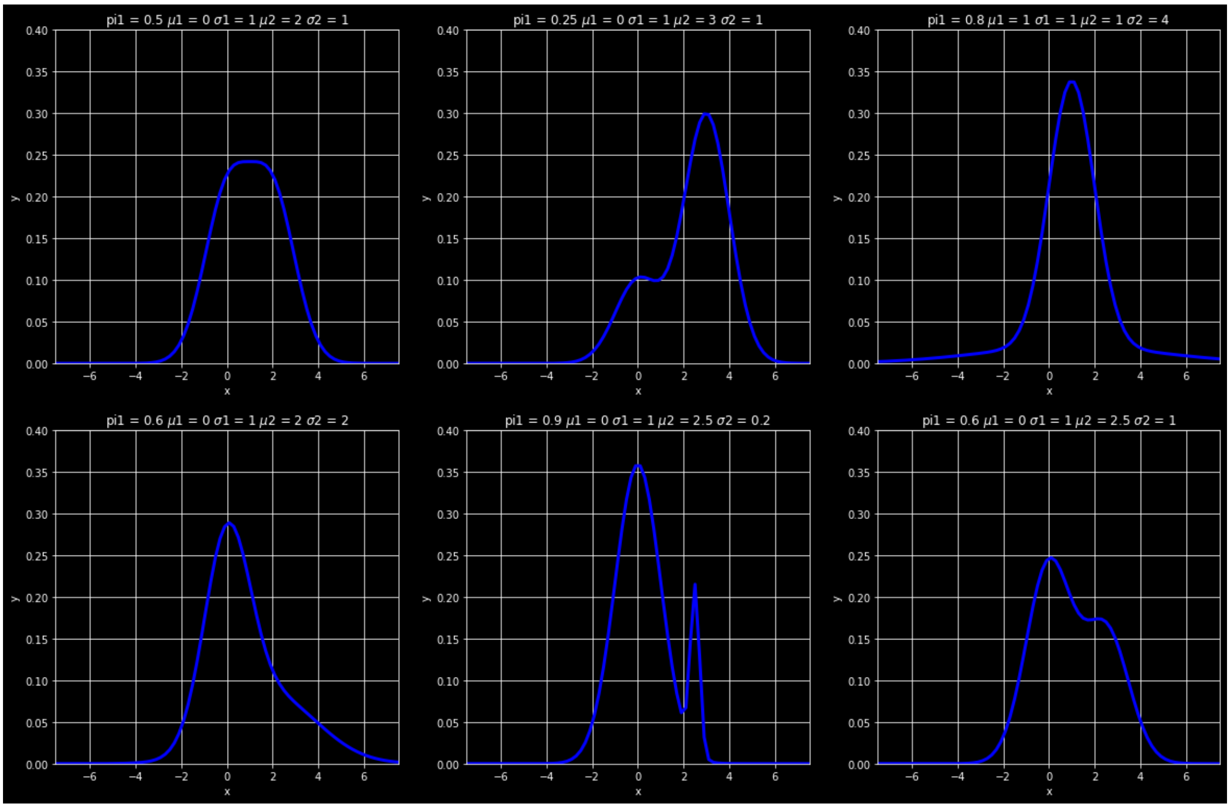
- #### 繪製執行結果，含資料的直方圖、真實的混合常態 PDF 與估計的混合常態 PDF。因每次執行的結果會因為樣本資料之不同而不同，選擇其中某一次即可。

- #### 混合常態的估計問題已經有一段歷史了，不難想像早已被寫成套件，譬如 sklearn.mixture.GaussianMixture。參照使用手冊。

1.先對六種混合常態的機率密度圖進行觀察

In [4]:

```python
import numpy as np
import scipy.optimize as opt
import matplotlib.pyplot as plt
from scipy.stats import beta, binom, norm
# 前置設定
x = np.linspace(-100, 100, 1000)
plt.figure(figsize = (20, 20))
# 用def定義繪圖
def g1_plot(pi1, a1, b1, a2, b2):
  f = lambda x: pi1 * norm.pdf(x, a1, b1) + (1-pi1) * norm.pdf(x, a2, b2)
  plt.plot(x, f(x), color = 'blue', linewidth = 3, label = 'True mixture')
  plt.grid(True)
  plt.xlim([-7.5, 7.5]), plt.ylim([0, 0.4])
  plt.xlabel("x"), plt.ylabel("y")
  plt.title('pi1 = {} $\mu 1$ = {} $\sigma 1$ = {} $\mu 2$ = {} $\sigma 2$ = {}'.for
# 開始進行繪圖
# pi1, a1, b1, a2, b2 = 0.5, 0, 1, 2, 1
plt.subplot(331)
g1_plot(0.5, 0, 1, 2, 1)

# pi1, a1, b1, a2, b2 = 0.25, 0, 1, 3, 1
plt.subplot(332)
g1_plot(0.25, 0, 1, 3, 1)

# pi1, a1, b1, a2, b2 = 0.8, 1, 1, 1, 4
plt.subplot(333)
g1_plot(0.8, 1, 1, 1, 4)

# pi1, a1, b1, a2, b2 = 0.6, 0, 1, 2, 2
plt.subplot(334)
g1_plot(0.6, 0, 1, 2, 2)

# pi1, a1, b1, a2, b2 = 0.9, 0, 1, 2.5, 0.2
plt.subplot(335)
g1_plot(0.9, 0, 1, 2.5, 0.2)

# pi1, a1, b1, a2, b2 = 0.6, 0, 1, 2.5, 1
plt.subplot(336)
g1_plot(0.6, 0, 1, 2.5, 1)
plt.show()
```



討論1

- ### plt.figure(figsize = (20, 20))是用來設定圖片大小，如果沒有設定會造成以上六張圖緊貼的狀況，不同張圖的標題及數字會疊在一起妨礙觀察。
- ### 用def去定義畫圖，可以省去時間眼力寫不段重複的程式碼，而且易於修改檢視。
- ### 六張圖設定一樣的 x y 範圍，更容易比較出不同混合常態的差異。
- ### plt.title部分使用了 = {}，可以包在def裡面讓程式自行填入。
- ### plt.subplot(331)是用來做一次化很多張圖的位置分配。plt.show()要在最後才能放上去，否則plt.subplot不能起作用。

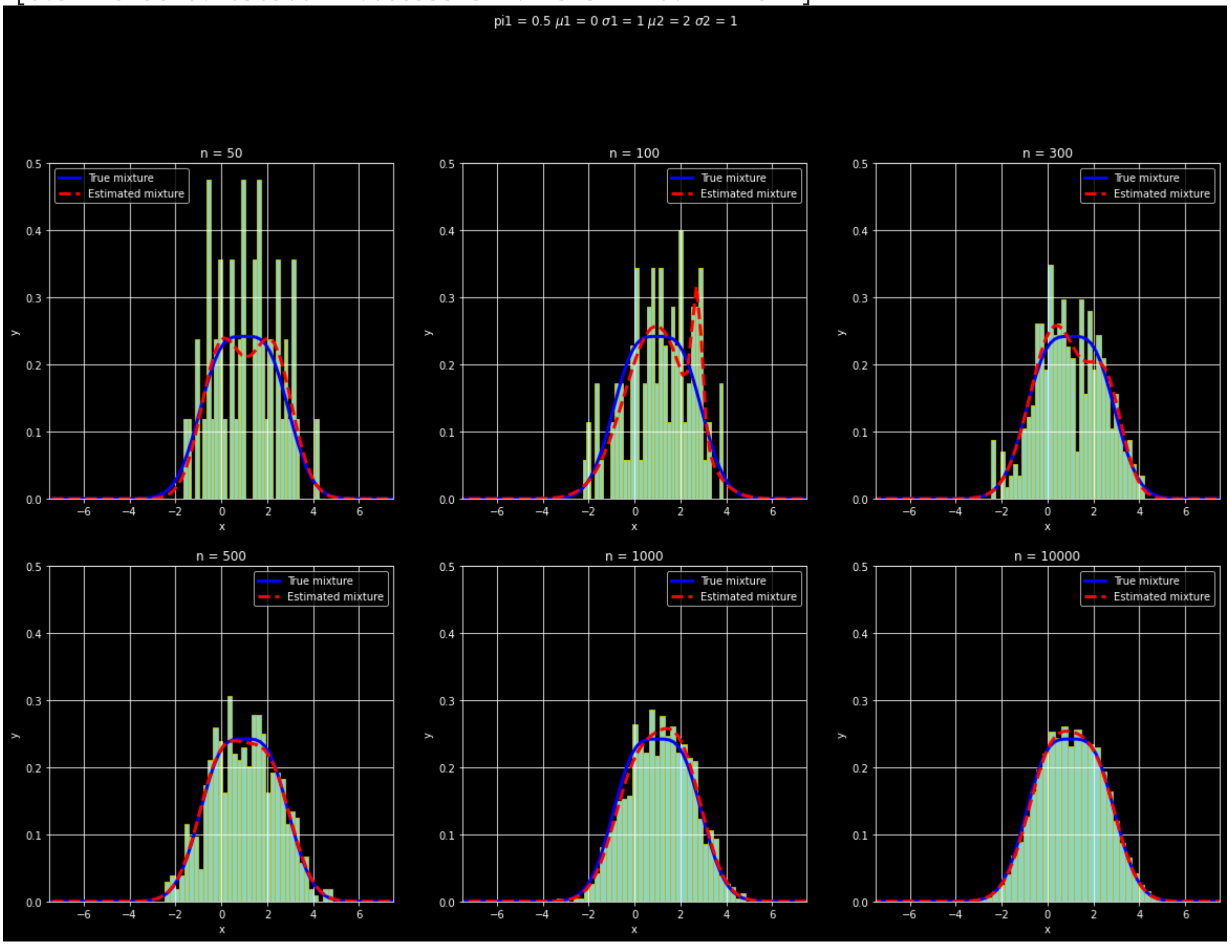
2. 不同樣本數的混合常態估計

In [5]:

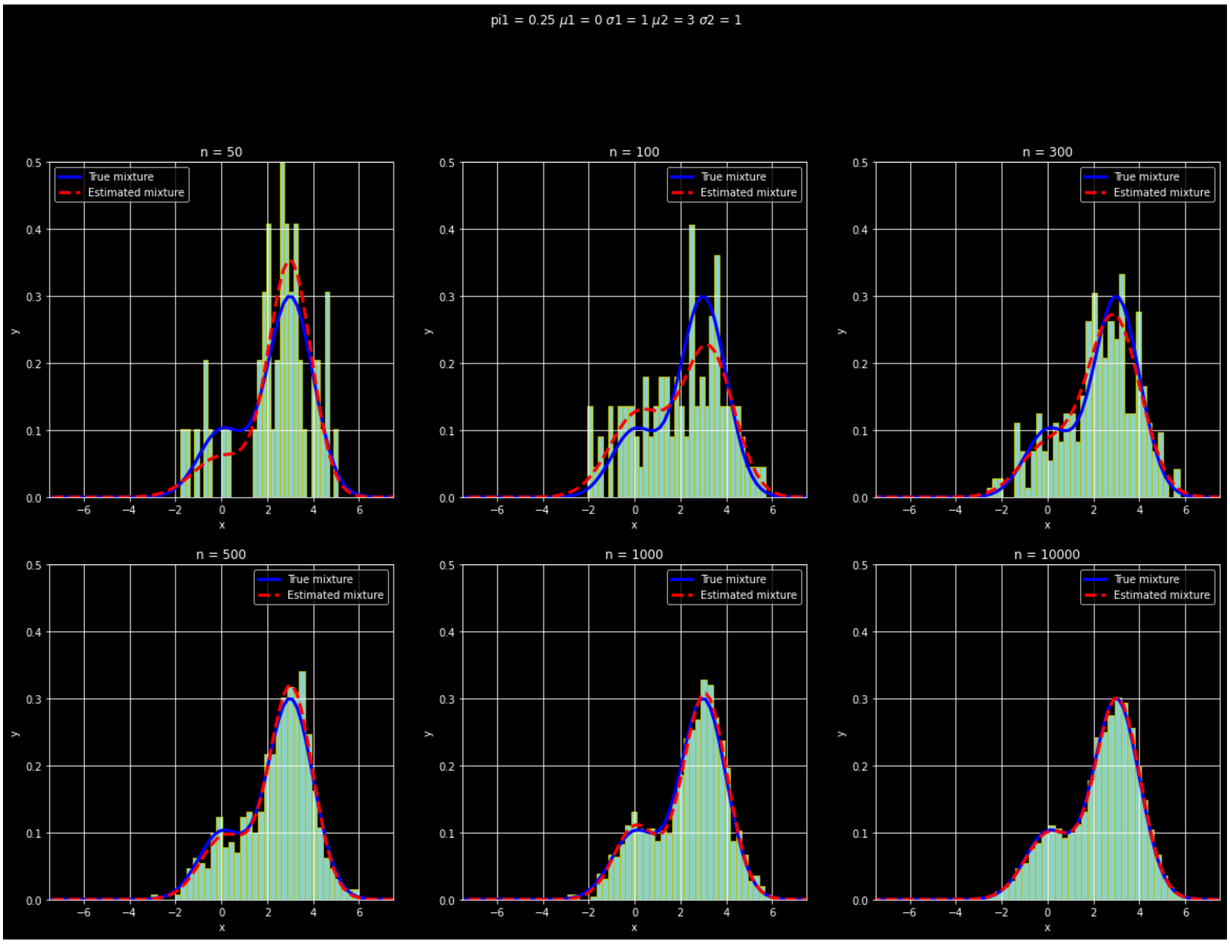
```
import numpy as np
import scipy.optimize as opt
import matplotlib.pyplot as plt
from scipy.stats import beta, binom, norm
# 前置設定
n = [50, 100, 300, 500, 1000, 10000]
x = np.linspace(-100, 100, 1000)
subplot = [331, 332, 333, 334, 335, 336]
# 用def定義繪圖
def g1_plot(n, pi1, a1, b1, a2, b2):
  f = lambda x: pi1 * norm.pdf(x, a1, b1) + (1-pi1) * norm.pdf(x, a2, b2)
  plt.plot(x, f(x), color = 'blue', linewidth = 3, label = 'True mixture')
  n1 = binom.rvs(n, pi1)
  n2 = n - n1
  sample = np.r_[norm.rvs(a1, b1, size = n1),
  norm.rvs(a2, b2, size = n2)]
  # 畫直方圖
  plt.hist(sample, 35, edgecolor = 'y', density = True)
  # max mle (min -mle)
  L = lambda x : -np.sum(np.log(x[0] * norm.pdf(sample, x[1], x[2]) + (1 - x[0]) * n
  # the constraints, bounds and options
  cons = []
```

```
  bnds = [(0, 1), (0, np.inf), (0, np.inf), (0, np.inf), (0, np.inf)]
  opts = dict(disp = True, maxiter = 1e4)
  # initial guess
  x0 = [pi1 - 0.1, a1 - 0.2, b1 + 0.2, a2 + 0.1, b2 - 0.1]
  res = opt.minimize(L, x0 = x0,
    bounds = bnds,
    constraints = cons,
    options = opts,
    tol = 1e-8)
  #估計參數
  print(res.x)
  # plot the estimated mixture pdf
  f_hat = lambda x: res.x[0] * norm.pdf(x, res.x[1], res.x[2]) + (1 - res.x[0]) * no
  plt.plot(x, f_hat(x), color = 'red', linestyle = '--',
  linewidth = 3, label = 'Estimated mixture')
  plt.legend()
  plt.xlabel("x"), plt.ylabel("y")
  plt.grid(True)
  plt.xlim([-7.5, 7.5]), plt.ylim([0, 0.5])
  plt.title("n = {}".format(n))
  plt.suptitle("pi1 = {} $\mu 1$ = {} $\sigma 1$ = {} $\mu 2$ = {} $\sigma 2$ = {}".

# pi1, a1, b1, a2, b2 = 0.5, 0, 1, 2, 1
plt.figure(figsize = (20, 20))
for i in range(len(n)):
 plt.subplot(subplot[i])
 g1_plot(n[i], 0.5, 0, 1, 2, 1)
plt.show()

# pi1, a1, b1, a2, b2 = 0.25, 0, 1, 3, 1
plt.figure(figsize = (20, 20))
for i in range(len(n)):
 plt.subplot(subplot[i])
 g1_plot(n[i], 0.25, 0, 1, 3, 1)
plt.show()

# pi1, a1, b1, a2, b2 = 0.8, 1, 1, 1, 4
plt.figure(figsize = (20, 20))
for i in range(len(n)):
 plt.subplot(subplot[i])
 g1_plot(n[i], 0.8, 1, 1, 1, 4)
plt.show()

# pi1, a1, b1, a2, b2 = 0.6, 0, 1, 2, 2
plt.figure(figsize = (20, 20))
for i in range(len(n)):
 plt.subplot(subplot[i])
 g1_plot(n[i], 0.6, 0, 1, 2, 2)
plt.show()

# pi1, a1, b1, a2, b2 = 0.9, 0, 1, 2.5, 0.2
plt.figure(figsize = (20, 20))
for i in range(len(n)):
 plt.subplot(subplot[i])
 g1_plot(n[i], 0.9, 0, 1, 2.5, 0.2)
plt.show()

# pi1, a1, b1, a2, b2 = 0.6, 0, 1, 2.5, 1
plt.figure(figsize = (20, 20))
for i in range(len(n)):
 plt.subplot(subplot[i])
 g1_plot(n[i], 0.6, 0, 1, 2.5, 1)
plt.show()
```

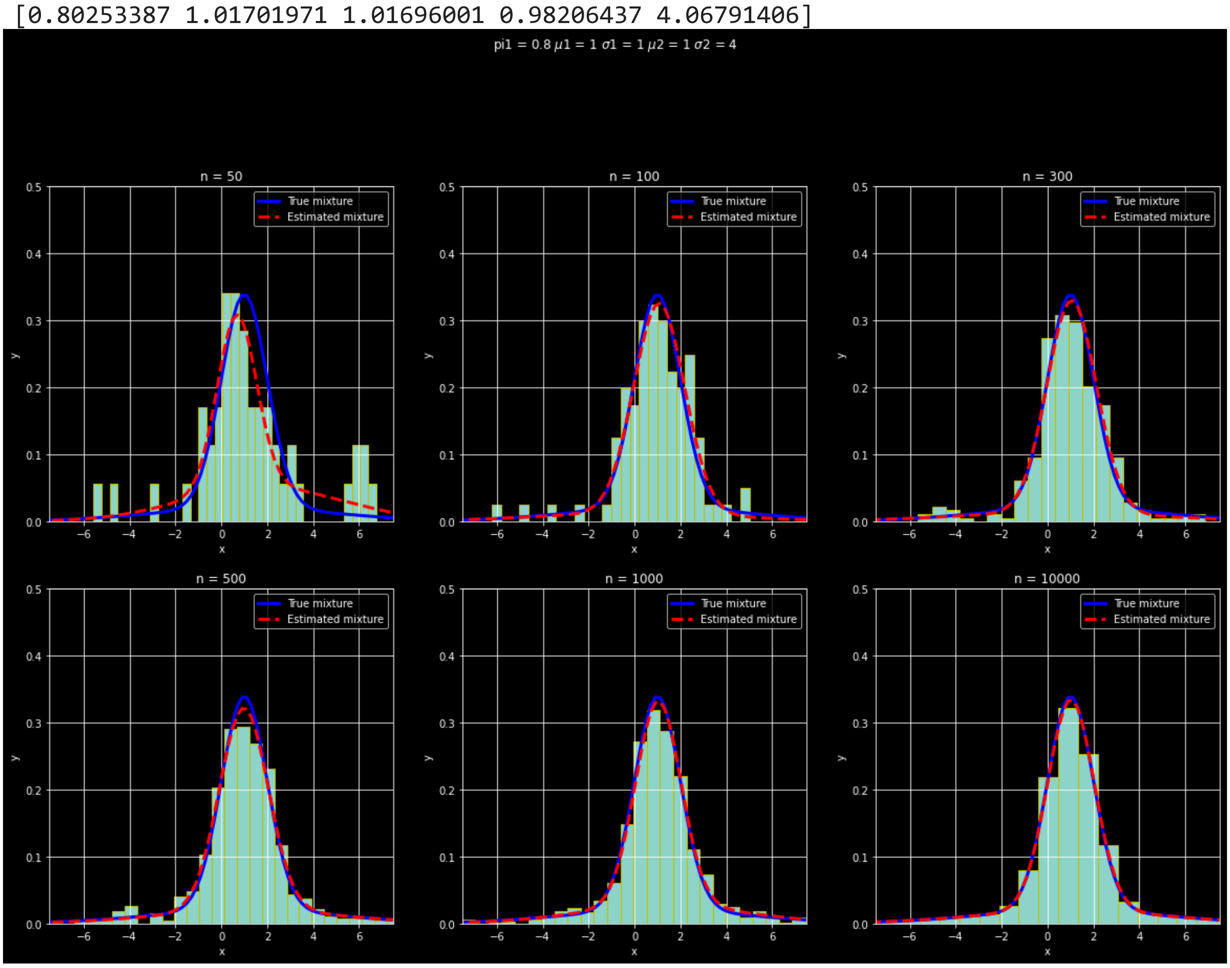
```
[0.4863209  0.         0.86183432 2.17502298 0.89819225]
[0.88852233 0.94504836 1.37693029 2.72966891 0.21800626]
[0.70776381 0.34169886 1.10183457 2.52940607 0.7424848 ]
[0.51253126 0.         1.00705399 2.04239091 1.007917  ]
[0.43087839 0.01506411 0.99148775 1.92016407 1.04126035]
[0.57495458 0.20505001 1.06853543 2.11373794 0.97177514]
```



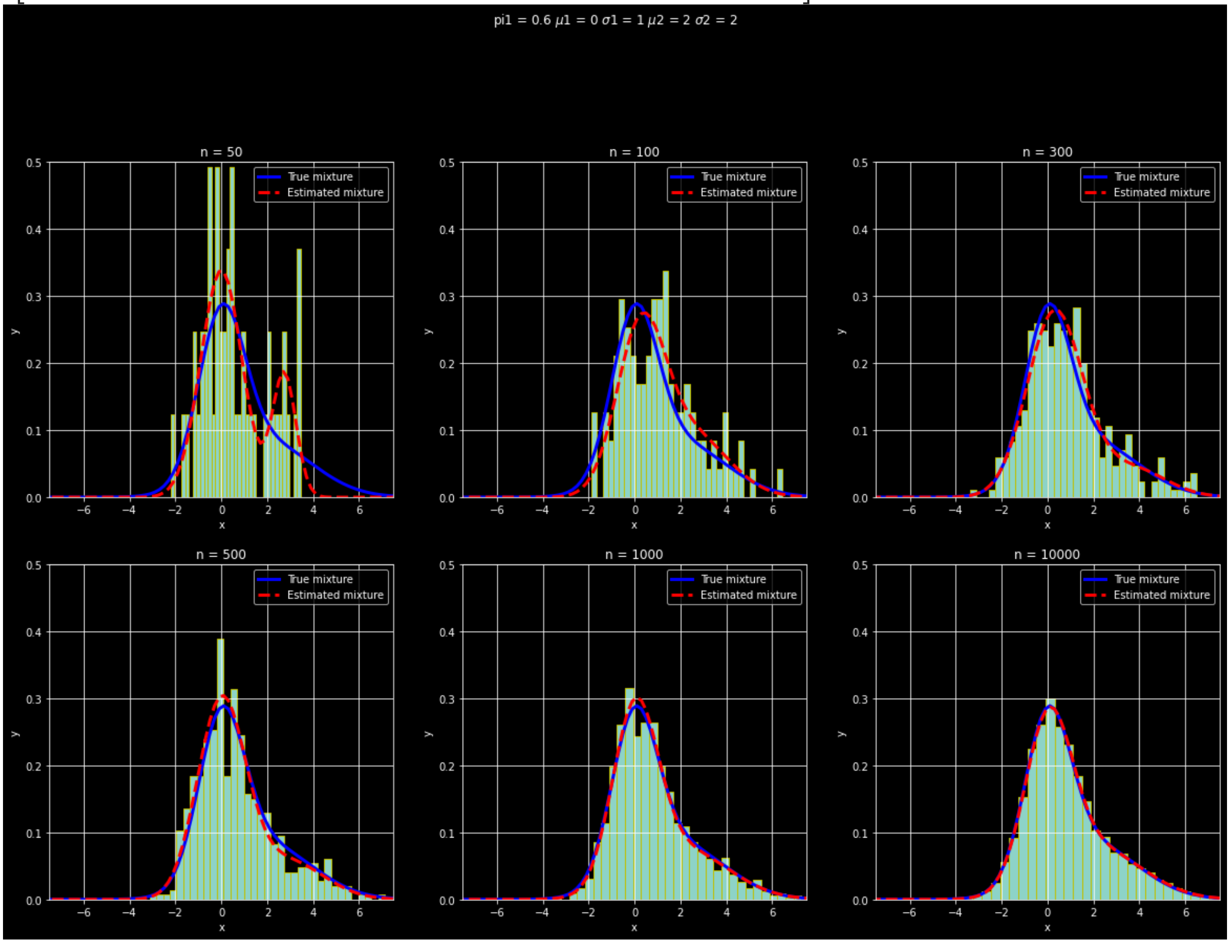

```
[0.18676115 0.         1.22227007 2.99790152 0.92363522]
[0.3933473  0.16677467 1.25029528 3.21542434 1.09511979]
[0.23157305 0.         1.19346874 2.8859197  1.13962024]
[0.24281527 0.12645486 1.03123756 3.02486019 0.94907255]
[0.25627452 0.04712909 0.93792656 3.07104792 0.96597005]
[0.25676051 0.05958661 1.03856075 3.04172484 0.98780022]
```



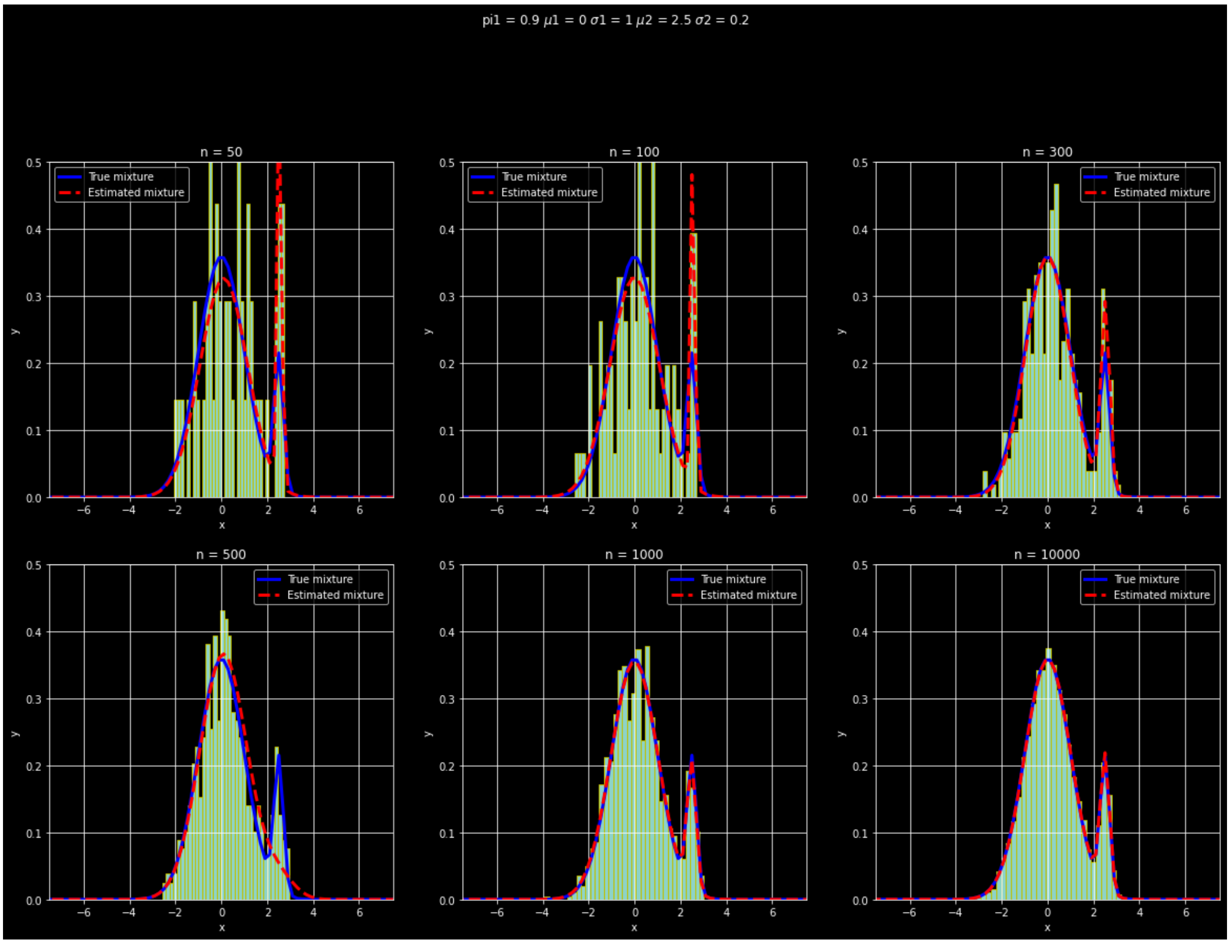
[0.55656292 0.67717827 0.85520888 1.82043995 3.43946269]
[0.86930961 1.07583655 1.10589375 0.   4.20655336]
[0.85496665 1.05014127 1.07552169 0.39703669 4.48703534]
[0.80645992 0.98121342 1.06538663 1.20409419 3.94859833]
[0.75884889 1.03693878 0.99047665 1.23729367 3.61469158]
[0.80253387 1.01701971 1.01696001 0.98206437 4.06791406]



```
[0.76286741 0.         0.89833443 2.73590232 0.51347485]
[0.63425283 0.27145008 1.05769454 2.51568259 1.5962543 ]
[0.83010389 0.31050793 1.20367413 3.57618303 1.53803704]
[0.7661206  0.         1.04729008 2.83873061 1.61118201]
[0.63730858 0.02817081 0.97960365 2.1531945  1.97326993]
[0.62434783 0.03739549 1.02452243 2.14617147 1.99943526]
```

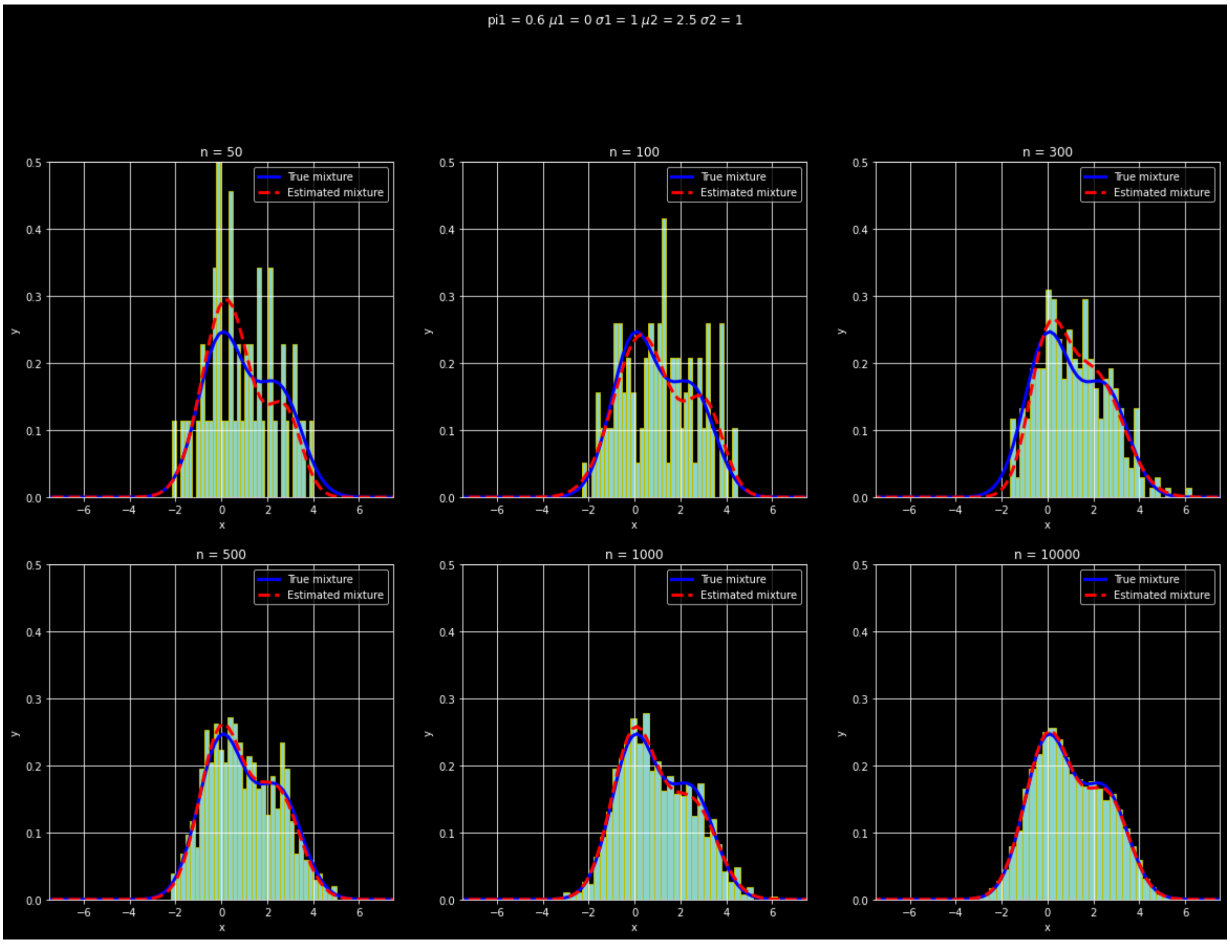



```
[0.84919896 0.10312775 1.03787975 2.54006289 0.0952895 ]
[0.87822354 0.         1.06799367 2.55181509 0.091208  ]
[0.86512293 0.         0.96174841 2.51716331 0.19173008]
[0.94733565 0.0918376  1.03212108 2.48286954 0.68828194]
C:\Users\User\AppData\Local\Programs\Python\Python39\lib\site-packages\scipy\stats\_
distn_infrastructure.py:1870: RuntimeWarning: divide by zero encountered in true_div
ide
  x = np.asarray((x - loc)/scale, dtype=dtyp)
C:\Users\User\AppData\Local\Temp/ipykernel_12324/2121117963.py:20: RuntimeWarning: d
ivide by zero encountered in log
  L = lambda x : -np.sum(np.log(x[0] * norm.pdf(sample, x[1], x[2]) + (1 - x[0]) * n
orm.pdf(sample, x[3], x[4])))
[0.90704052 0.         1.01908412 2.4931889  0.19394412]
[0.89893783 0.00767918 1.0010482  2.50127614 0.19878292]
```




```
[0.76940335 0.19795345 1.0398991  2.76276509 0.73464508]
[0.73557891 0.30275126 1.21121474 3.0771685  0.80757   ]
[0.42980765 0.         0.81970701 2.02101045 1.24036064]
```

```
C:\Users\User\AppData\Local\Programs\Python\Python39\lib\site-packages\scipy\stats\_
distn_infrastructure.py:1870: RuntimeWarning: divide by zero encountered in true_div
ide
  x = np.asarray((x - loc)/scale, dtype=dtyp)
C:\Users\User\AppData\Local\Temp/ipykernel_12324/2121117963.py:20: RuntimeWarning: d
ivide by zero encountered in log
  L = lambda x : -np.sum(np.log(x[0] * norm.pdf(sample, x[1], x[2]) + (1 - x[0]) * n
orm.pdf(sample, x[3], x[4])))
[0.59980355 0.         0.94936667 2.40231571 0.99625258]
[0.6213711  0.         0.99639267 2.52457032 1.0606873 ]
[0.63242571 0.04138673 1.02505371 2.57919365 0.96796033]
```



討論2

- ### def g1_plot(pi1, a1, b1, a2, b2)這樣的目的是為了保持函數的機動性，此函數可以快速變換成想要的混合函數，只要變更參數。
- ### initial guess部份"x0 = [pi1 - 0.1, a1 - 0.2, b1 + 0.2, a2 + 0.1, b2 - 0.1]"，這邊是estimated mixture的參數。因為這整段我想寫出可以隨時變更true mixture參數的def，所以x0是用直接拿true mixture參數進行增減，而不是用純數字表示。
- ### plt.title、plt.suptitle寫法搭配 = {}，讓程式可以自動填入數字，方便且不會出錯。
- ### 最後迴圈的部分是同時跑兩個東西。一個是subplot，所以最一開始有subplot = [331, 332, 333, 334, 335, 336]的設計，一次畫六張圖，plt.figure(figsize = (20, 20))是為了不讓六張圖彼此緊貼影響判斷。另一個是n，因為要用不同的樣本去跑看看，所以最一開始有設定n = [50, 100, 300, 500, 1000, 10000]，代表不同的樣本數。
- ### 在完成所有畫圖設定後，才使用plt.show()，完成收尾。
- ### 由上方圖中可看出在true mixture和estimated mixture參數有些微差異下，樣本數少的時候會有一定程度的差異；但隨著樣本數增加，true mixture 和 estimate mixture會逐漸相同。
- ### print(res.x)可以用來看估計的參數，可以看出當n越大時，估計的參數越靠近真實值。

3.加入 sklearn.mixture.GaussianMixture 的 EM 演算法做比較

In [6]:

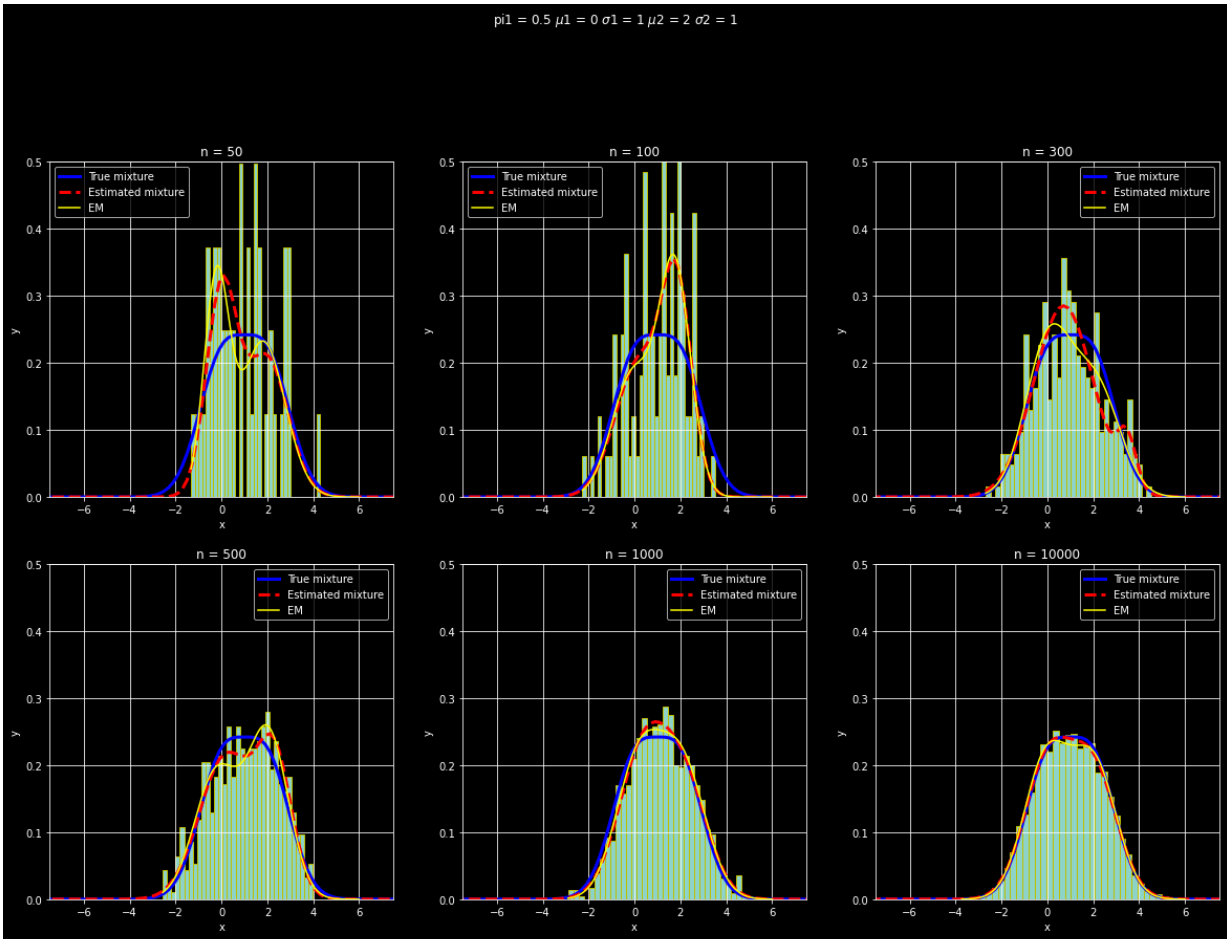
```
import numpy as np
import scipy.optimize as opt
import matplotlib.pyplot as plt
from scipy.stats import beta, binom, norm
from sklearn import mixture
# 前置設定
```

```
n = [50, 100, 300, 500, 1000, 10000]
x = np.linspace(-100, 100, 1000)
subplot = [331, 332, 333, 334, 335, 336]
# 用def定義繪圖
def g1_plot(n, pi1, a1, b1, a2, b2):
  f = lambda x: pi1 * norm.pdf(x, a1, b1) + (1-pi1) * norm.pdf(x, a2, b2)
  plt.plot(x, f(x), color = 'blue', linewidth = 3, label = 'True mixture')
  n1 = binom.rvs(n, pi1)
  n2 = n - n1
  sample = np.r_[norm.rvs(a1, b1, size = n1),
  norm.rvs(a2, b2, size = n2)]
  # 畫直方圖
  plt.hist(sample, 35, edgecolor = 'y', density = True)
  # max mle (min -mle)
  L = lambda x : -np.sum(np.log(x[0] * norm.pdf(sample, x[1], x[2]) + (1 - x[0]) * n
  # the constraints, bounds and options
  cons = []
  bnds = [(0, 1), (0, np.inf), (0, np.inf), (0, np.inf), (0, np.inf)]
  opts = dict(disp = True, maxiter = 1e4)
  # initial guess
  x0 = [pi1 - 0.1, a1 - 0.2, b1 + 0.2, a2 + 0.1, b2 - 0.1]
  res = opt.minimize(L, x0 = x0,
    bounds = bnds,
    constraints = cons,
    options = opts,
    tol = 1e-8)
  #估計參數
  print(res.x)
  # plot the estimated mixture pdf
  f_hat = lambda x: res.x[0] * norm.pdf(x, res.x[1], res.x[2]) + (1 - res.x[0]) * no
  plt.plot(x, f_hat(x), color = 'red', linestyle = '--',
  linewidth = 3, label = 'Estimated mixture')
  plt.legend()
  plt.xlabel("x"), plt.ylabel("y")
  plt.grid(True)
  plt.xlim([-7.5, 7.5]), plt.ylim([0, 0.5])
  plt.title("n = {}".format(n))
  plt.suptitle("pi1 = {} $\mu 1$ = {} $\sigma 1$ = {} $\mu 2$ = {} $\sigma 2$ = {}".
  # EM [mixture.GaussianMixture]
  gmm = mixture.GaussianMixture(n_components=2, covariance_type='full',\
            verbose = 0, max_iter = 1000, tol = 1e-6)
  gmm.fit(sample.reshape(-1, 1))

  weights = gmm.weights_
  means = gmm.means_
  covars = gmm.covariances_

  model = mixture.GaussianMixture(2).fit(sample.reshape(-1,1))
  x_range = np.linspace(np.min(sample), 6, 1000)
  pdf = np.exp(model.score_samples(x_range.reshape(-1, 1)))
  responsibilities = model.predict_proba(x_range.reshape(-1, 1))
  pdf_individual = responsibilities * pdf[:, np.newaxis]
  plt.plot(x_range, pdf, label='EM', color="yellow")
  plt.legend()

# pi1, a1, b1, a2, b2 = 0.5, 0, 1, 2, 1
plt.figure(figsize = (20, 20))
for i in range(len(n)):
 plt.subplot(subplot[i])
 g1_plot(n[i], 0.5, 0, 1, 2, 1)
plt.show()

# pi1, a1, b1, a2, b2 = 0.25, 0, 1, 3, 1
plt.figure(figsize = (20, 20))
```

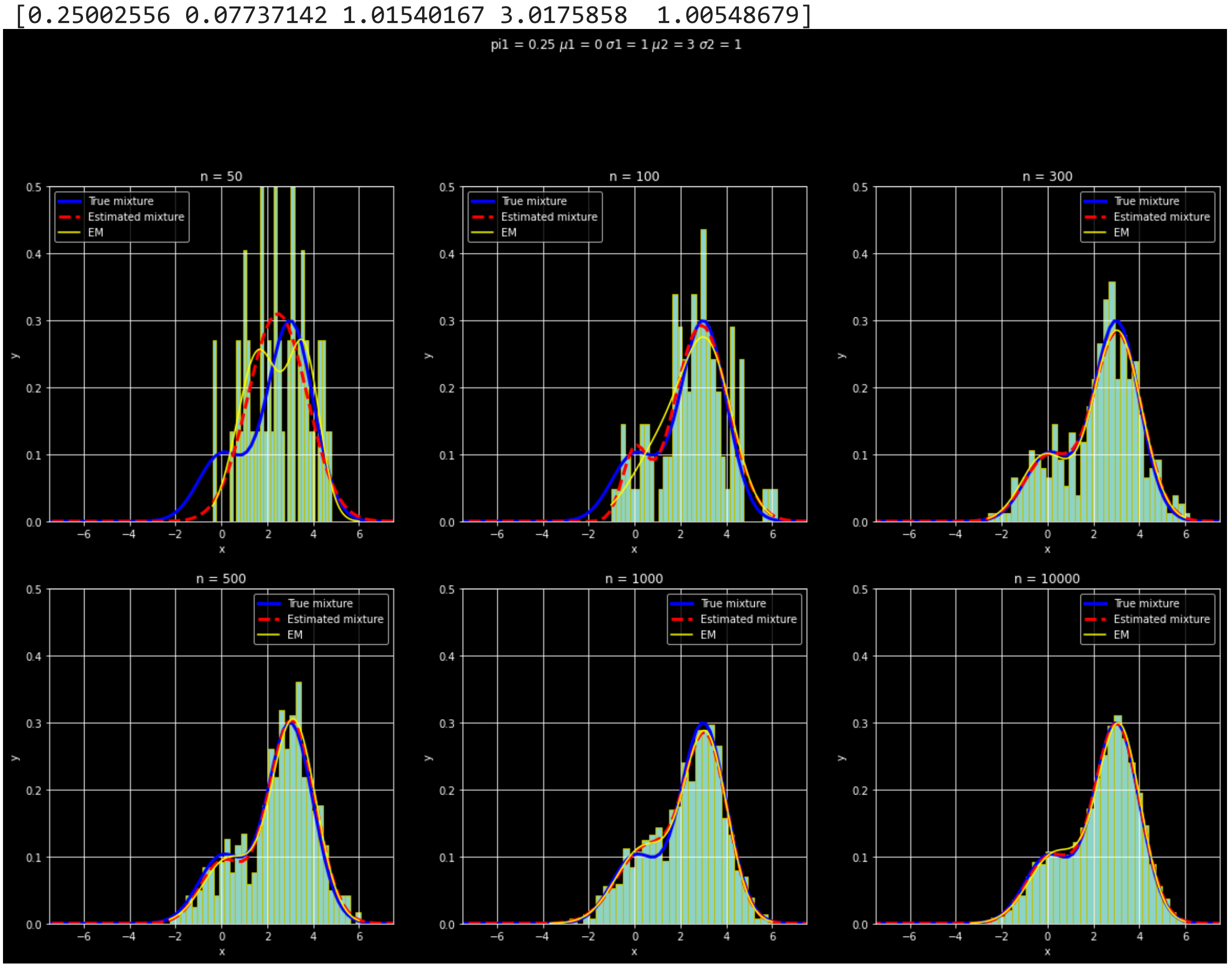
```python
for i in range(len(n)):
    plt.subplot(subplot[i])
    g1_plot(n[i], 0.25, 0, 1, 3, 1)
plt.show()

# pi1, a1, b1, a2, b2 = 0.8, 1, 1, 1, 4
plt.figure(figsize = (20, 20))
for i in range(len(n)):
    plt.subplot(subplot[i])
    g1_plot(n[i], 0.8, 1, 1, 1, 4)
plt.show()

# pi1, a1, b1, a2, b2 = 0.6, 0, 1, 2, 2
plt.figure(figsize = (20, 20))
for i in range(len(n)):
    plt.subplot(subplot[i])
    g1_plot(n[i], 0.6, 0, 1, 2, 2)
plt.show()

# pi1, a1, b1, a2, b2 = 0.9, 0, 1, 2.5, 0.2
plt.figure(figsize = (20, 20))
for i in range(len(n)):
    plt.subplot(subplot[i])
    g1_plot(n[i], 0.9, 0, 1, 2.5, 0.2)
plt.show()

# pi1, a1, b1, a2, b2 = 0.6, 0, 1, 2.5, 1
plt.figure(figsize = (20, 20))
for i in range(len(n)):
    plt.subplot(subplot[i])
    g1_plot(n[i], 0.6, 0, 1, 2.5, 1)
plt.show()
```

```
[0.51233497 0.         0.67085607 1.9559914  0.93126655]
[0.5199326  0.19896905 1.03459022 1.85307958 0.65162367]
```

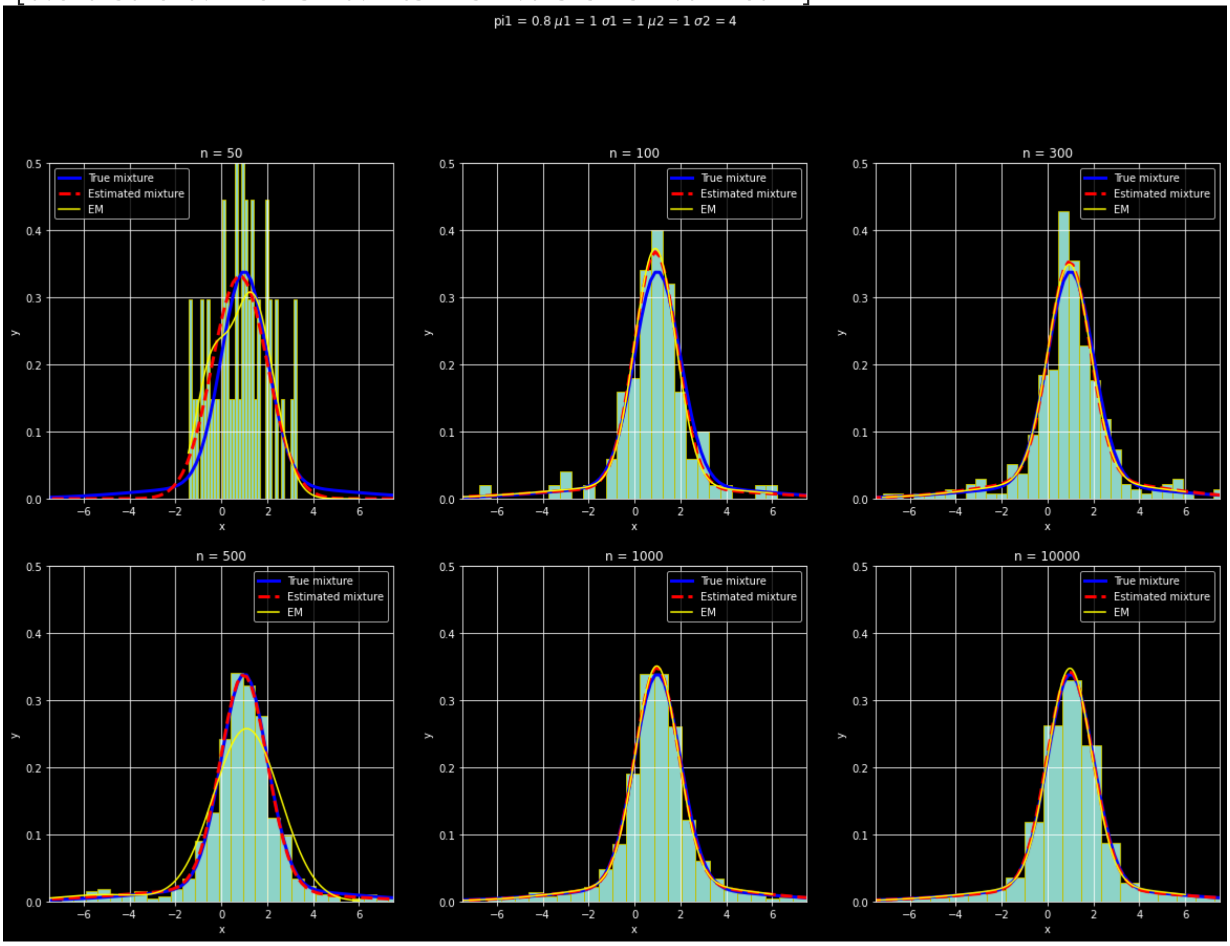
```
C:\Users\User\AppData\Local\Programs\Python\Python39\lib\site-packages\scipy\stats\_
distn_infrastructure.py:1870: RuntimeWarning: divide by zero encountered in true_div
ide
  x = np.asarray((x - loc)/scale, dtype=dtyp)
C:\Users\User\AppData\Local\Temp/ipykernel_12324/410409333.py:21: RuntimeWarning: di
vide by zero encountered in log
  L = lambda x : -np.sum(np.log(x[0] * norm.pdf(sample, x[1], x[2]) + (1 - x[0]) * n
orm.pdf(sample, x[3], x[4])))
[0.92646501 0.70558751 1.29581653 3.46677194 0.39631723]
[0.61968716 0.19793777 1.15452275 2.31335538 0.76116575]
[0.60710398 0.41027907 1.09121995 2.24076981 1.0066736 ]
[0.51737    0.         1.01828872 2.0207243  1.01898246]
```



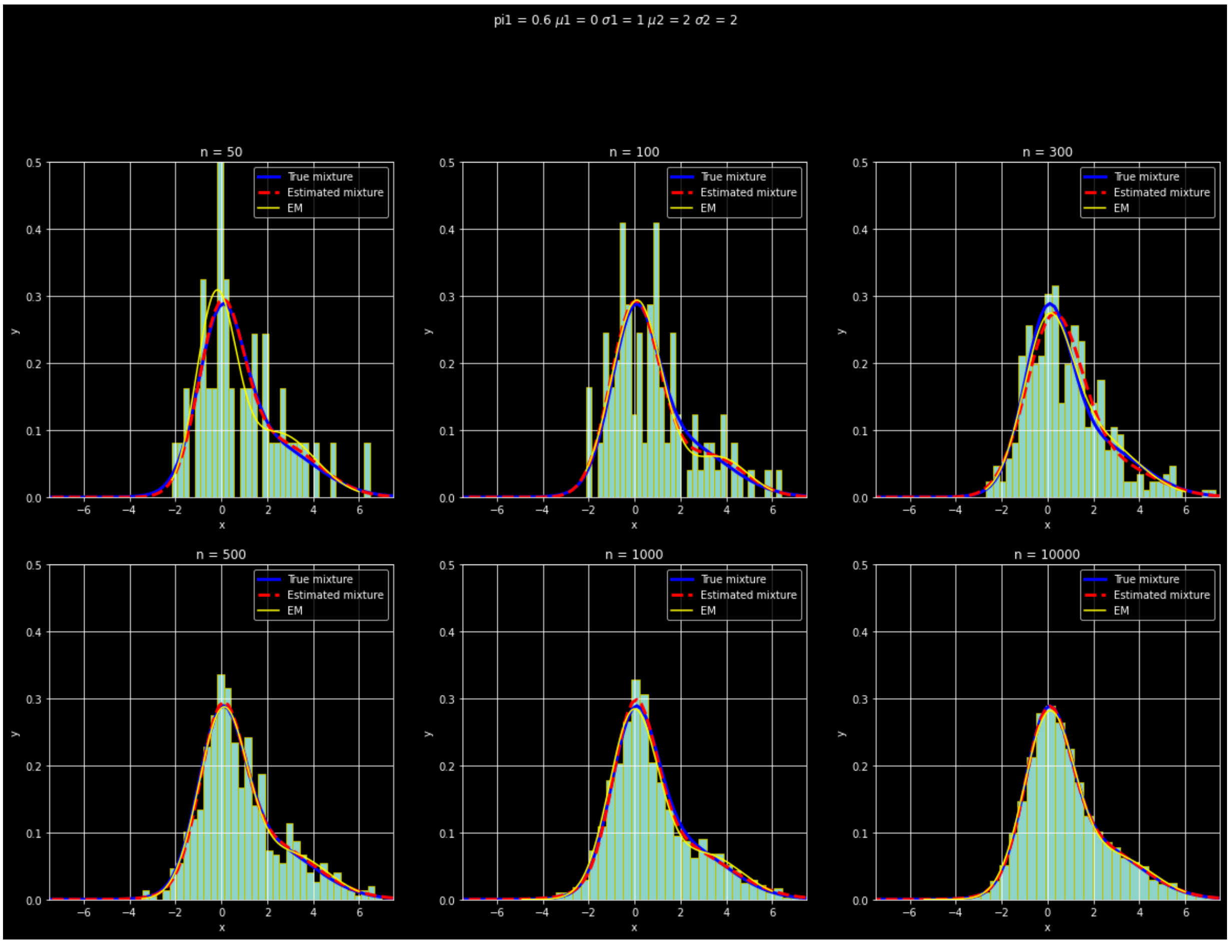

```
[0.         0.28569874 1.09647876 2.4518189  1.28554204]
[0.1340592  0.         0.54168318 2.91776132 1.17951037]
[0.25124318 0.         1.04675998 3.03405311 1.05708345]
[0.21629525 0.02126459 0.9452988  3.06335921 1.03001037]
[0.34234309 0.43914081 1.22374124 3.10056666 0.95380469]
[0.25002556 0.07737142 1.01540167 3.0175858  1.00548679]
```



```
[1.         0.80310527 1.20030401 1.08752849 3.69138086]
[0.81962586 0.91984116 0.93022382 0.50402576 4.32255749]
[0.73506802 0.94705044 0.90659023 1.00038291 3.55795624]
[0.82102332 0.97593656 1.02089521 0.         4.15845959]
[0.77472178 0.97208933 0.95176127 1.16880425 3.70370145]
[0.81015078 0.9728943  0.99637745 1.07395195 4.07748049]
```

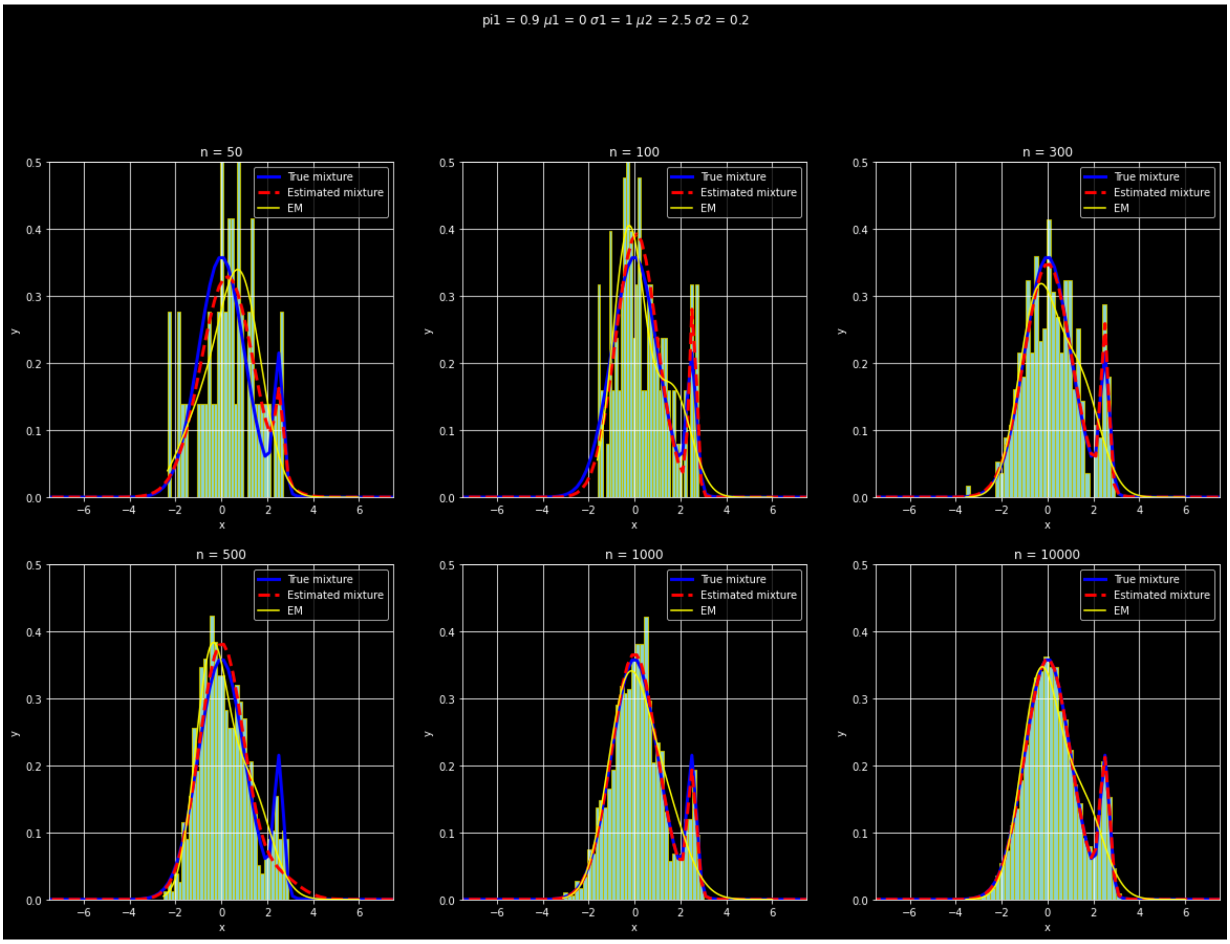



```
[0.65506412 0.         0.98064294 2.4890872  1.73583138]
[0.76934739 0.02771781 1.06791761 3.24973642 1.48878362]
[0.76100755 0.24390616 1.20671757 2.63983862 1.87951031]
[0.575981   0.01944024 0.95010621 2.0005513  2.03499912]
[0.57430896 0.         0.93896462 1.90414685 2.16273113]
[0.5882058  0.         0.99172669 1.96850157 2.03197296]
```

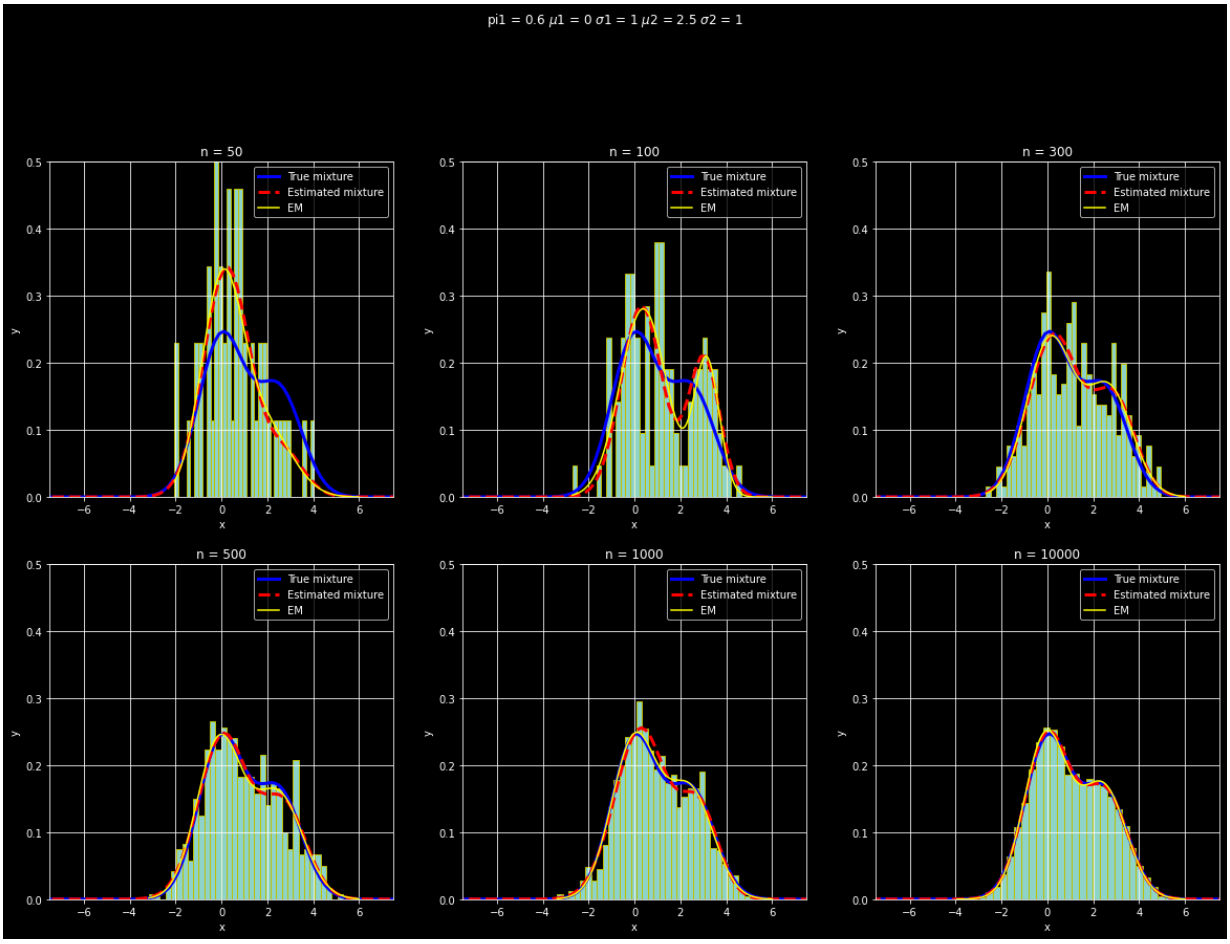



```
C:\Users\User\AppData\Local\Programs\Python\Python39\lib\site-packages\scipy\stats\_
distn_infrastructure.py:1870: RuntimeWarning: divide by zero encountered in true_div
ide
  x = np.asarray((x - loc)/scale, dtype=dtyp)
[0.95058449 0.25528403 1.15142188 2.529997   0.17151348]
[0.89136205 0.08699021 0.90531693 2.5460889  0.15471269]
[0.89755033 0.01056128 1.02736936 2.4910537  0.16956732]
```

```
C:\Users\User\AppData\Local\Programs\Python\Python39\lib\site-packages\scipy\stats\_
distn_infrastructure.py:1870: RuntimeWarning: divide by zero encountered in true_div
ide
  x = np.asarray((x - loc)/scale, dtype=dtyp)
C:\Users\User\AppData\Local\Temp/ipykernel_12324/410409333.py:21: RuntimeWarning: di
vide by zero encountered in log
  L = lambda x : -np.sum(np.log(x[0] * norm.pdf(sample, x[1], x[2]) + (1 - x[0]) * n
orm.pdf(sample, x[3], x[4])))
[0.93152837 0.01103939 0.97356219 2.53393747 0.86410474]
[0.91268877 0.         0.99326925 2.50910306 0.19577902]
[0.89459775 0.00364526 0.99461657 2.49242554 0.21337145]
```




```
[0.86594165 0.22953379 1.00939939 2.74711885 0.8583967 ]
C:\Users\User\AppData\Local\Programs\Python\Python39\lib\site-packages\scipy\stats\_
distn_infrastructure.py:1870: RuntimeWarning: divide by zero encountered in true_div
ide
  x = np.asarray((x - loc)/scale, dtype=dtyp)
[0.64102061 0.31050411 0.89653381 3.02218663 0.68555032]
[0.66186577 0.2712031  1.09350883 2.90026955 0.92186431]
[0.65322457 0.06068771 1.06953554 2.69214092 0.98304985]
[0.72049787 0.27082645 1.13015611 2.81719251 0.83721517]
[0.60245683 0.         0.98449369 2.49899393 0.99350982]
```



討論3

- ### 這邊使用了EM，[sklearn.mixture.GaussianMixture]套件，所以最一開始需要from sklearn import mixture。
- ### 出來的結果是，在樣本小時，EM會和其他兩條線(TRUE MIXTURE、ESTIMATE MIXTURE)有差異，但隨著樣本數越大，三條線會逐漸吻合。

習題 2：限制式條件的最大值問題 **Constraint optimization**

工作敘述

計算下列最大概似估計 **MLE** 問題的參數 α, β：

$$\max_{\alpha,\beta>0} \ln L(\alpha,\beta)$$

其中的聯合概似函數為

$$L(\alpha,\beta) = \prod_{i=1}^{n} f_t(v_i|\alpha,\beta) F_T(u_i|\alpha,\beta)^{-1}$$

where

$$f_t(v|\alpha,\beta) = \alpha\beta v^{\beta-1} exp(-\alpha v^{\beta})$$

$$F_T(u|\alpha,\beta) = 1 - exp(-\alpha u^\beta)$$

變數 **u,v** 的 **n** 個樣本已知並存在檔案 **UV.txt**

建議依下列程序，逐步進行：

1.先下載資料檔，取出資料並觀察資料的樣子。

2.目標函數 $\ln L(\alpha,\beta)$ 需要進一步推導到比較適合的樣子，也就是將 $\prod$ 透過 $\ln$ 換成 Σ。並不是連乘的 $\prod$ 不能計算，非要換成連加的 Σ 不可，而是當樣本數多時，連乘的計算比較不穩定，太大或太小的數值連乘可能超過硬體的極限。所以典型的最大概似估計問題，往往會在原目標函數前加上對數 $\ln$ 轉換成連加模式。請盡量將式子推到最精簡。

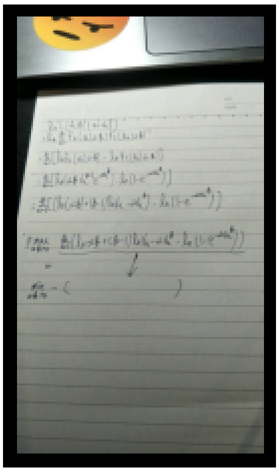
3.利用推導到精簡的目標函數，繪製立體圖與等高線圖。繪圖時，需要摸索參數的範圍，找到最佳的觀察位置。畫得好，隱約可以看出最大值的位置

4.接著開始部署 **minimize** 的各項停止條件及計算。有了等高線圖的幫助，通常答案已經呼之欲出，計算的結果只是得到一組更明確的數據。

1.推導MLE的數學式至可程式化的階段。

In [8]:

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import numpy as np

lena = mpimg.imread('hw6.jpg')
lena.shape
plt.imshow(lena)
plt.axis('off')
plt.show()
```



討論1

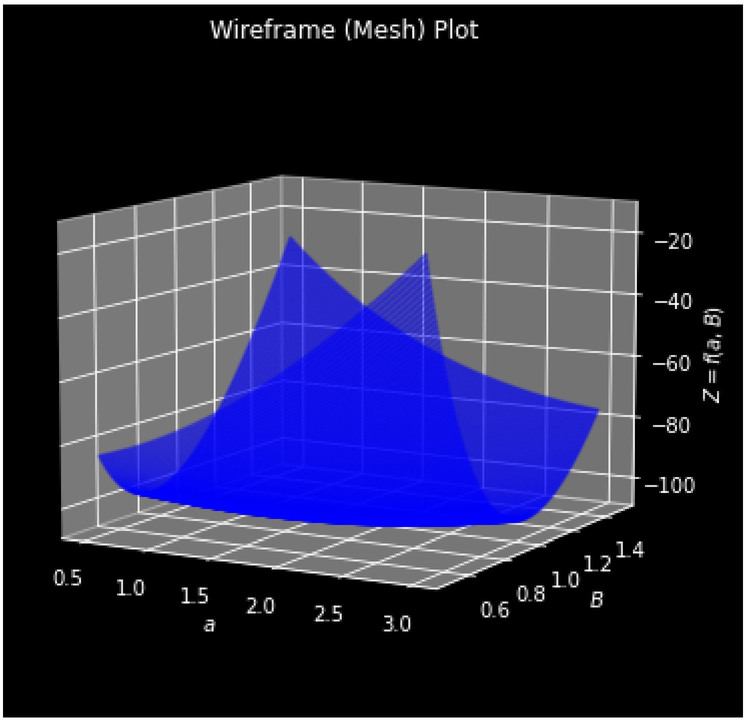
- ### 用mpimg.imread讀取照片。

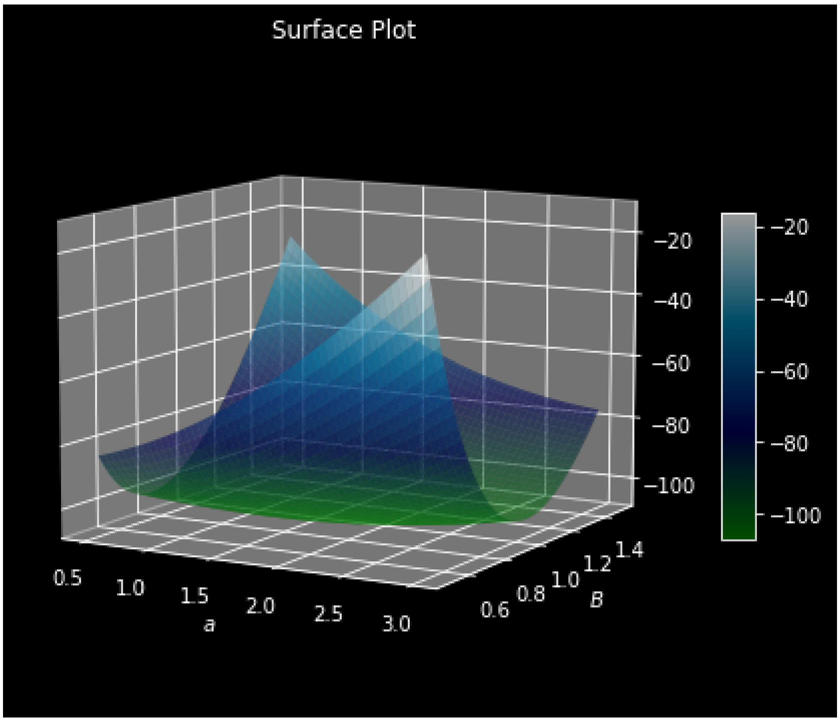
2.撰寫程式

In [9]:

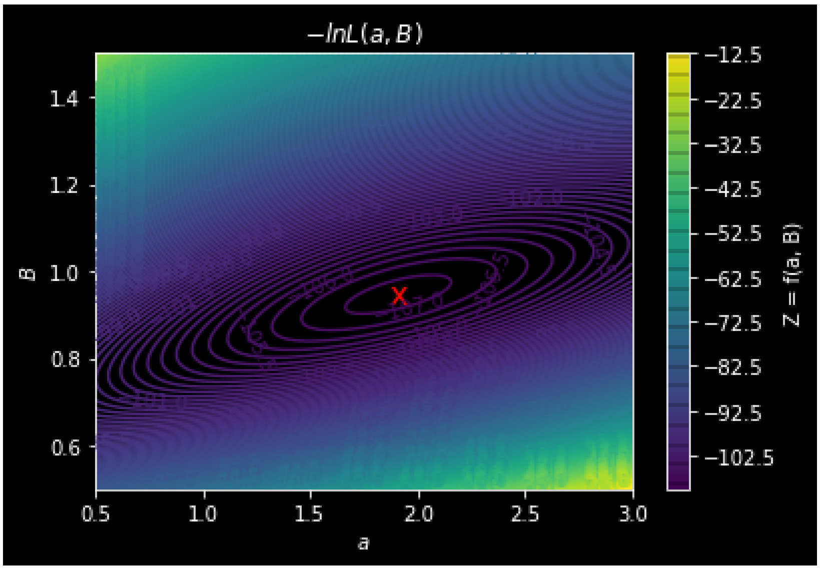
```
import numpy as np
import matplotlib.pyplot as plt
import scipy.optimize as opt
# 寫函式
def f(a):
    l = np.zeros([206, 206])
    for i in range(n):
        l = l + (-1) * (np.log(a[0] * a[1]) + (a[1] - 1) * np.log(v[i]) - a[0]*v[i]*
    return l
# 讀檔
D = np.loadtxt("UV.txt", comments = "%")
u, v = D[:, 0], D[:, 1]
n = D.shape[0]

a = np.linspace(0.5, 3, n); b = np.linspace(0.5, 1.5, n)
# 網格網格矩陣
X, Y = np.meshgrid(a, b)
Z = f([X, Y])
# 對於線框，通過 rstride 和 cstride 控制線密度
fig = plt.figure(figsize=(9, 6))
ax = plt.axes(projection = '3d')
ax.plot_wireframe(X, Y, Z, color ='blue',
    alpha=0.3, rstride = 1, cstride = 1)
ax.set_xlabel('$a$'), ax.set_ylabel('$B$')
ax.set_zlabel('$Z = f(a, B)$')
ax.view_init(10, -60)
plt.title('Wireframe (Mesh) Plot')
plt.show()

# for surface 3D plot
fig = plt.figure(figsize=(9, 6))
ax = plt.axes(projection = '3d')
surf = ax.plot_surface(X, Y, Z, color = 'r', \
    rstride=4, cstride=4, alpha =0.6, cmap='ocean')
fig.colorbar(surf, ax=ax, shrink=0.5, aspect=10)
ax.view_init(10, -60)
ax.set_xlabel('$a$'), ax.set_ylabel('$B$')
plt.title('Surface Plot')
plt.show()

# 計算最大值
ff = lambda x: -np.sum((np.log(x[0] * x[1]) + (x[1] - 1) * np.log(v) - x[0] * (v **
opts = dict(disp = 1, xtol = 1e-6, ftol = 1e-6, maxfun = 1e4, maxiter = 1e4, full_ou
Optval = opt.fmin(func=ff, x0=[1,1], **opts)
print(Optval[0][0], Optval[0][1], Optval[1])

levels = np.arange(-110, -12, 0.5)
contours = plt.contour(X, Y, Z, levels=levels)
# 在每一行添加函數值
plt.clabel(contours, inline = 0, fontsize = 10)
cbar = plt.colorbar(contours)
plt.text(Optval[0][0], Optval[0][1], 'x', color ='red', fontsize = 16,
    horizontalalignment='center',
    verticalalignment='center')
plt.xlabel('$a$'), plt.ylabel('$B$')
# set colorbar label
cbar.ax.set_ylabel('Z = f(a, B)')
plt.title('$-lnL(a, B)$')
plt.show()
```

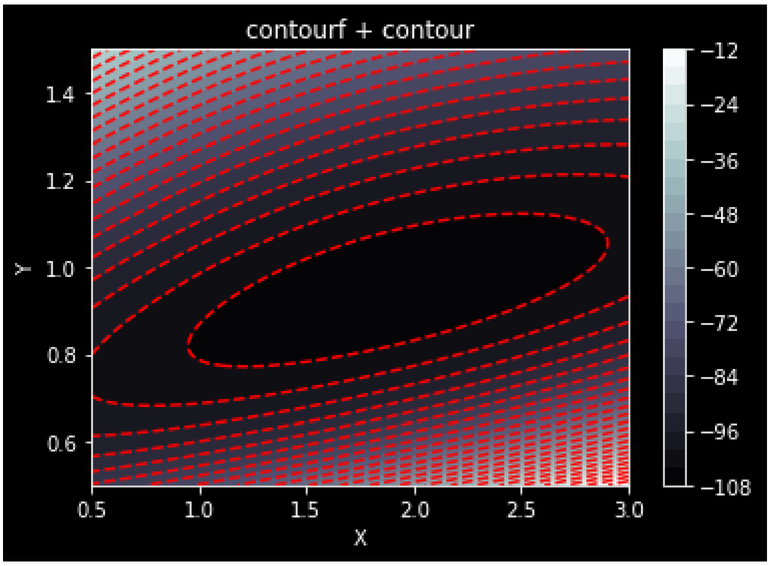
```python
# 用contourf繪製等高線圖
C1 = plt.contourf(X, Y, Z, 30, \
    cmap = plt.cm.bone)
C2 = plt.contour(C1, levels = C1.levels, \
    colors = 'r')
plt.colorbar(C1)
plt.xlabel('X'), plt.ylabel('Y')
plt.title('contourf + contour')
plt.show()
```






```
Optimization terminated successfully.
         Current function value: -107.208541
         Iterations: 53
         Function evaluations: 107
1.9073503546844888 0.9464390188940687 -107.20854069643369
```





討論2

- ### 要看u v的資料數，可以用len(u)語法查看。
- ### 最一開始寫函式部分np.zeros([206, 206])，原因是u v 資料數各是206個。
- ### np.loadtxt("UV.txt", comments = "%")，comments="%"，代表第一行有%的部分不要讀，從第二行開始讀。
- ### a = np.linspace(0.5, 3, n); b = np.linspace(0.5, 1.5, n)是設定alpha beta的範圍。
- ### levels = np.arange(-110, -12, 0.5)是可以設定"bar"的範圍。
- ### 計算最大值的部分，因為def f(a)的部分有涉及迴圈，所以"opts、Optval"的程式部分會執行錯誤，所以我再寫一個不含迴圈的函式，專門用來計算此題的最大值。
- ### 最大值可由"Optval[0][0], Optval[0][1], Optval[1]"看出，在alpha=Optval[0][0]，beta=Optval[0][1]時，會有最大值-Optval[1]，Optval[1]其中要記得加負號。意即，當alpha = 1.9時 beta = 0.9時 會有最大值107。