# EM_GMM

工程算法——EM算法实验作业要求

代码仓库： https://github.com/nchen909/EM_GMM

## 问题背景

当公司推出新产品时，他们通常想找出目标客户。如果他们有关于客户购买历史和购物偏好的数据，他们可以利用这些数据来预测哪些类型的客户更有可能购买新产品。高斯混合模型（GMM）模型便可以解决这个典型的无监督学习问题。

## GMM、EM 算法背景

GMM 是概率模型，它假设所有数据点都是从具有未知参数的几个高斯分布的混合中生成的。每个分布看做一个类别的话，它与 k-means 聚类的不同之处在于 GMM 包含有关每个聚类的中心（均值）和 variance（方差）的信息，并提供后验概率。

在前面提到的示例中，我们有 2 个集群（cluster）：**喜欢新产品的人和不喜欢新产品的人**。如果我们知道每个客户属于**哪个集群（标签）**，我们可以轻松估计**集群的参数（均值和方差）**，或者如果我们知道两个集群的参数，我们可以预测标签。不幸的是，我们都不知道其中任何一个。为了解决这个先有鸡还是先有蛋的问题，便需要使用期望最大化算法 (EM) 。

EM是在存在隐变量时寻找最大似然的迭代算法。该算法**在期望(E)步和最大化(M)步之间迭代**，前者使用参数的当前估计创建后验分布和对数似然的启发式（**估计样本的标签分布**），后者通过从E步最大化似然后验期望，**获得新一轮参数先验**，然后在接下来的E步中使用M步的参数估计。


（EM算法已知数据来自多个分布（在本例中为多个GMM），概率模型和参数不一定相同，每个样本属于哪个分布未知，作为隐变量）

## 实现

### 无监督GMM

#### 假设

我们将已知变量定义为$x$，将未知标签定义为$y$，并做了两个假设：先验分布$p(y)$是二项式的，每个集群中的$p(x|y)$是高斯分布。

$$\text{observed: } x, \text{ latent: } y$$

$$\begin{aligned}\text{prior: } & p(y;\phi) = \phi^{1\{y=1\}}(1-\phi)^{1\{y=0\}} \\ \text{evidence: } & p(x|y=0;\mu_0,\Sigma_0) = N(\mu_0,\Sigma_0) \\ & p(x|y=1;\mu_1,\Sigma_1) = N(\mu_1,\Sigma_1)\end{aligned}$$

$$\theta := \phi, \mu_0, \mu_1, \Sigma_0, \Sigma_1$$



## 数据读入

客户个人偏好未知，x1、x2表示客户的 2 个特征（假设客户数据服从高斯分布），我们的目标是预测客户是否喜欢该产品（y=1）或不喜欢（y=0）。

| x1 | x2 |
|---|---|
| -0.187 | 0.747 |
| 2.824 | 0.377 |
| 0.713 | 0.766 |
| 1.635 | 1.846 |
| 2.711 | 1.996 |



```
data_unlabeled = pd.read_csv("data/unlabeled.csv")
x_unlabeled = data_unlabeled[["x1", "x2"]].values
```

## 随机初始化参数

对于无监督GMM，所有参数都是随机初始化的。为简单起见，我们使用 θ 来表示以下等式中的所有参数。（当然，这里也可以探索使用kmeans等init_params方法初始化参数。）

$$\theta := \phi, \mu_0, \mu_1, \Sigma_0, \Sigma_1$$

```
def get_random_psd(n):#生成协方差矩阵
    x = np.random.normal(0, 1, size=(n, n))
    return np.dot(x, x.transpose())

def initialize_random_params():
    params = {'phi': np.random.uniform(0, 1),
              'mu0': np.random.normal(0, 1, size=(2,)),
              'mu1': np.random.normal(0, 1, size=(2,)),
              'sigma0': get_random_psd(2),
              'sigma1': get_random_psd(2)}
    return params
```

## E步

将初始化的参数传递给`e_step()`并计算每个数据点的启发式*Q(y=1|x) 和 Q(y=0|x)*以及我们将在 M 步中最大化的平均对数似然。（之所以成为启发式，是因为它们是用猜测的参数 θ 计算的。）

E step:

$$\begin{aligned}\text{for each } i \in \{1,\dots,n\}\text{: set}\\ Q(y^i=1|x^i) &:= p(y^i=1|x^i;\theta)\\ &= \frac{p(x^i|y^i=1;\theta)p(y^i=1;\theta)}{\sum_{y^i\in(0,1)} p(x^i|y^i;\theta)p(y^i;\theta)}\\ Q(y^i=0|x^i) &:= p(y^i=0|x^i;\theta)\\ &= \frac{p(x^i|y^i=0;\theta)p(y^i=0;\theta)}{\sum_{y^i\in(0,1)} p(x^i|y^i;\theta)p(y^i;\theta)}\end{aligned}$$



根据上述步骤完成`e_step()`过程。

```
def e_step(x):
    # *****START OF YOUR CODE (DO NOT DELETE/MODIFY THIS LINE)*****

    pass

    # *****END OF YOUR CODE (DO NOT DELETE/MODIFY THIS LINE)*****
```

## M步

在`m_step()`中，最大化似然期望，更新参数并获得新一轮参数先验。对于一般分布的EM，由于下式第三部计算复杂度为NP-hard，为将求和移出log，需要用Jensen不等式进行放缩。最后，如果后验期望的argmax没有封闭形式的解决方案，将需要使用梯度上升来解决优化问题并找到参数估计值。

M step:

$$
\begin{aligned}
\theta &= \text{argmax}\ell(\theta) \\
&= \text{argmax} \sum_{i=1}^{n} \log p(x^i; \theta) \\
&= \text{argmax} \sum_{i=1}^{n} \log \sum_{y^i \in (0,1)} p(x^i, y^i; \theta) \\
&\geq \text{argmax} \sum_{i=1}^{n} \sum_{y^i \in (0,1)} Q(y^i|x^i) \log p(x^i, y^i; \theta) \\
&= \text{argmax} \sum_{i=1}^{n} \text{E}_{Q(y^i|x^i)} \log p(x^i|y^i; \theta) p(y^i; \theta)
\end{aligned}
$$



而在本例中假设分布为GMM，它有上述问题的闭式解法，结果如下。

$$
\begin{aligned}
\phi &= \frac{\sum_{i=1}^{n} Q(y^i = 1|x^i)}{n} \\
\mu_0 &= \frac{\sum_{i=1}^{n} x^i Q(y^i = 0|x^i)}{\sum_{i=1}^{n} Q(y^i = 0|x^i)} \\
\mu_1 &= \frac{\sum_{i=1}^{n} x^i Q(y^i = 1|x^i)}{\sum_{i=1}^{n} Q(y^i = 1|x^i)} \\
\Sigma_0 &= \frac{\sum_{i=1}^{n} Q(y^i = 0|x^i)(x^i - \mu_0)(x^i - \mu_0)^T}{\sum_{i=1}^{n} Q(y^i = 0|x^i)} \\
\Sigma_1 &= \frac{\sum_{i=1}^{n} Q(y^i = 1|x^i)(x^i - \mu_1)(x^i - \mu_1)^T}{\sum_{i=1}^{n} Q(y^i = 1|x^i)}
\end{aligned}
$$



使用闭式解法完成`m_step()`。

```
def m_step(x):
    # *****START OF YOUR CODE (DO NOT DELETE/MODIFY THIS LINE)*****

    pass

    # *****END OF YOUR CODE (DO NOT DELETE/MODIFY THIS LINE)*****
    return params
```

## 迭代更新

在 E 步的下一次迭代中使用这些更新的参数，获得新的启发式并运行 M 步。EM 算法所做的是重复运行以上步骤直至平均对数似然收敛。

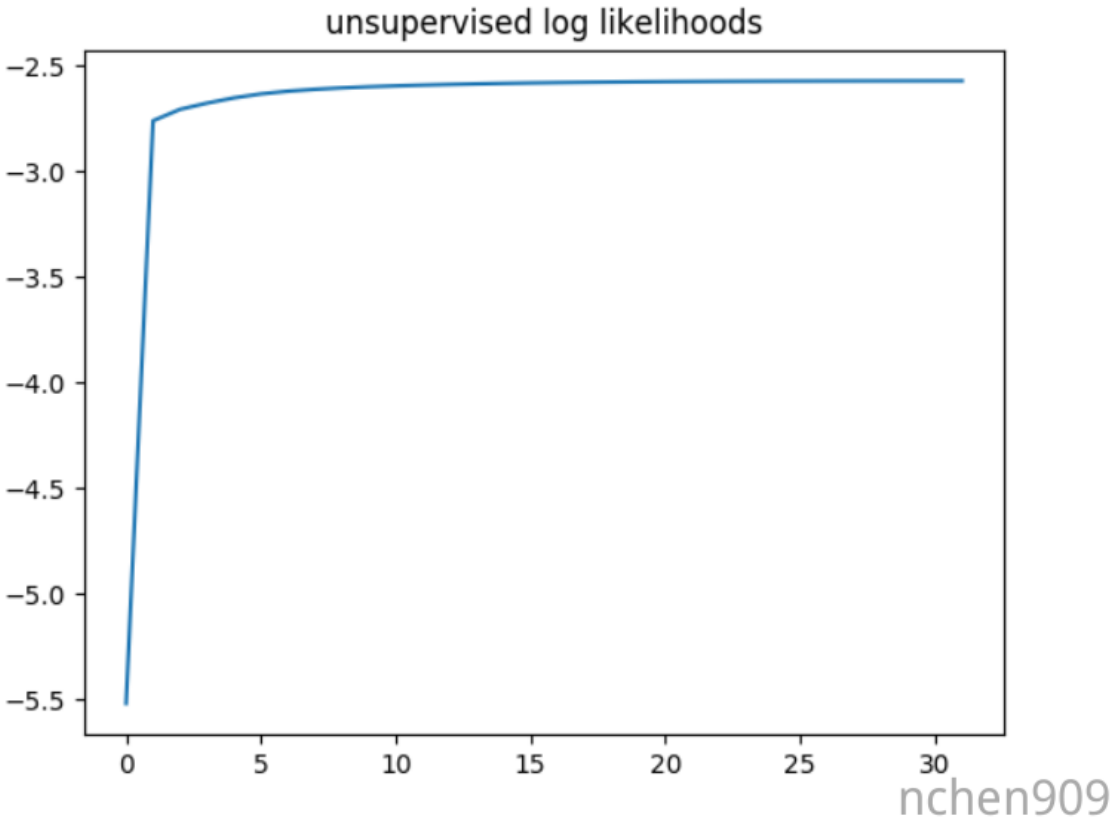
使用`e_step()`、`m_step()`函数完成`run_em()`并获得预测标签、后验概率和对数似然。

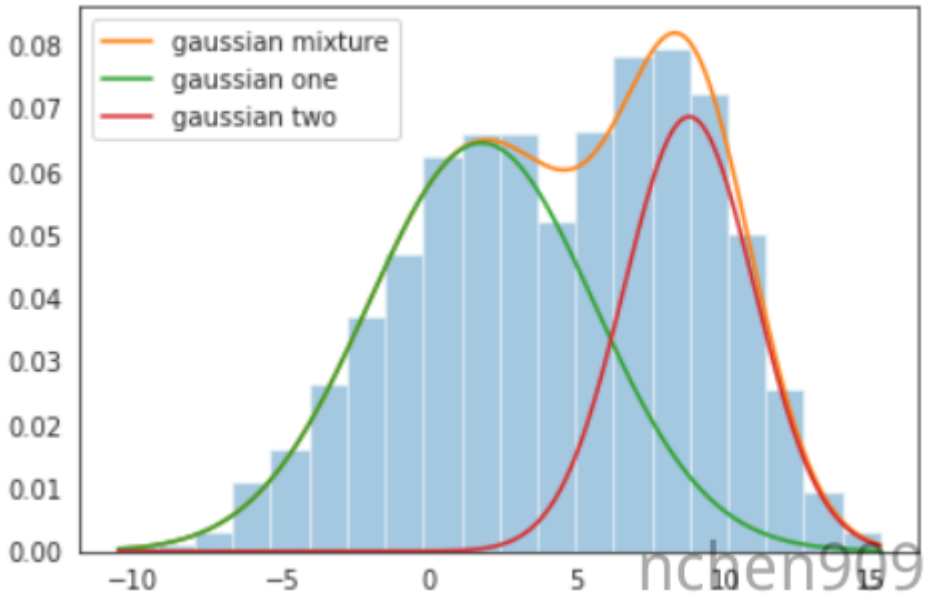
```
def run_em(x):
    # *****START OF YOUR CODE (DO NOT DELETE/MODIFY THIS LINE)*****

    pass

    # *****END OF YOUR CODE (DO NOT DELETE/MODIFY THIS LINE)*****
```

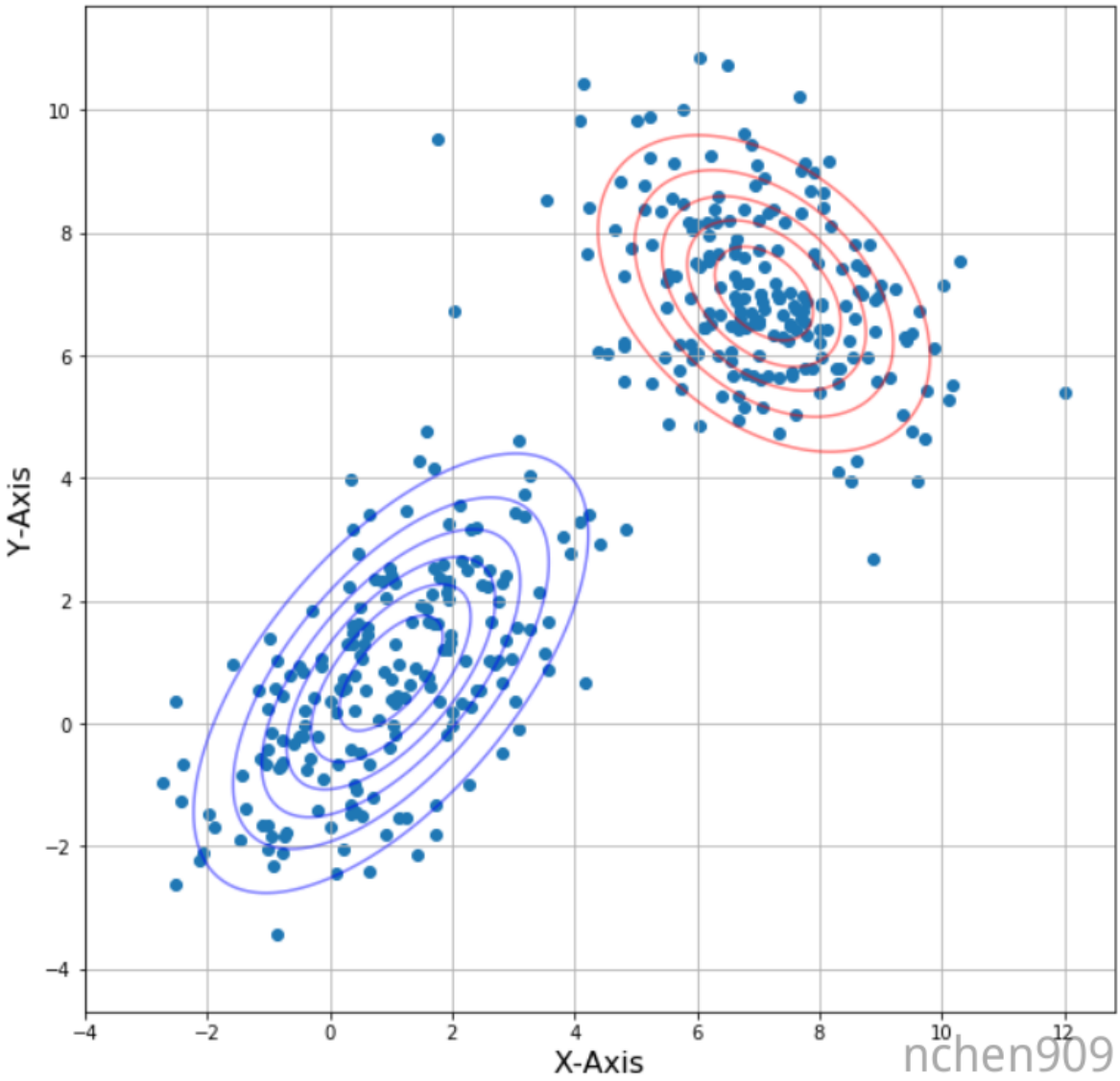
## 输出标签、可视化

通过`run_em()`得到的预测标签、后验概率和对数似然，输出对每个用户的预测标签。也可以探索似然收敛过程，迭代过程的分布图，最终收敛的效果等（加分项），示例效果如下。

```
predict: [1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
 1 1 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1
 1 1 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 1 1 1 1 1 0 1 1 1 1
 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1
 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0
```




gaussian mixture
gaussian one
gaussian two
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
−10
−5
0
5
10
15
nchen909


Y-Axis
X-Axis
10
8
6
4
2
0
−2
−4
−4
−2
0
2
4
6
8
10
12
nchen909

```
em_unsupervised = EM()
# *****START OF YOUR CODE (DO NOT DELETE/MODIFY THIS LINE)*****

unsupervised_forecasts,unsupervised_posterior =
em_unsupervised.run_em(x_unlabeled)
#visulization
pass

# *****END OF YOUR CODE (DO NOT DELETE/MODIFY THIS LINE)*****
```

# 半监督GMM

在某些情况下，我们有少量的标记数据。例如，我们可能会从调查中了解一些客户的偏好。考虑到潜在客户群巨大，我们拥有的标记数据量不足以进行全监督学习，但我们**可以通过半监督的方式从数据中学习初始参数。**

这里使用与以前相同的未标记数据，但这次我们也有一些标记数据。

## 数据读入

```
data_labeled = pd.read_csv("data/labeled.csv")
x_labeled = data_labeled[["x1", "x2"]].values
y_labeled = data_labeled["y"].values
```

| x1 | x2 | y |
|---|---|---|
| -0.813 | -1.099 | 0 |
| -0.414 | -1.369 | 0 |
| -0.463 | 0.195 | 0 |
| -1.72 | -2.537 | 0 |
| -0.188 | -0.25 | 0 |



## 使用标签数据初始化参数

在半监督场景下，使用有标记数据学习初始化参数，可避免失败的随机初始化并以更少的步数收敛。可依据以下方式进行初始化。

$$\phi^{'} = \frac{\sum_{i=1}^{m} 1\{y^i = 1\}}{n}$$

$$\mu_0^{'} = \frac{\sum_{i=1}^{m} x^i 1\{y^i = 0\}}{\sum_{i=1}^{m} 1\{y^i = 0\}}$$

$$\mu_1^{'} = \frac{\sum_{i=1}^{m} x^i 1\{y^i = 1\}}{\sum_{i=1}^{m} 1\{y^i = 1\}}$$

$$\Sigma_0^{'} = \frac{\sum_{i=1}^{m} 1\{y^i = 0\}(x^i - \mu_0^{'})(x^i - \mu_0^{'})^T}{\sum_{i=1}^{m} 1\{y^i = 0\}}$$

$$\Sigma_1^{'} = \frac{\sum_{i=1}^{m} 1\{y^i = 1\}(x^i - \mu_1^{'})(x^i - \mu_1^{'})^T}{\sum_{i=1}^{m} 1\{y^i = 1\}}$$

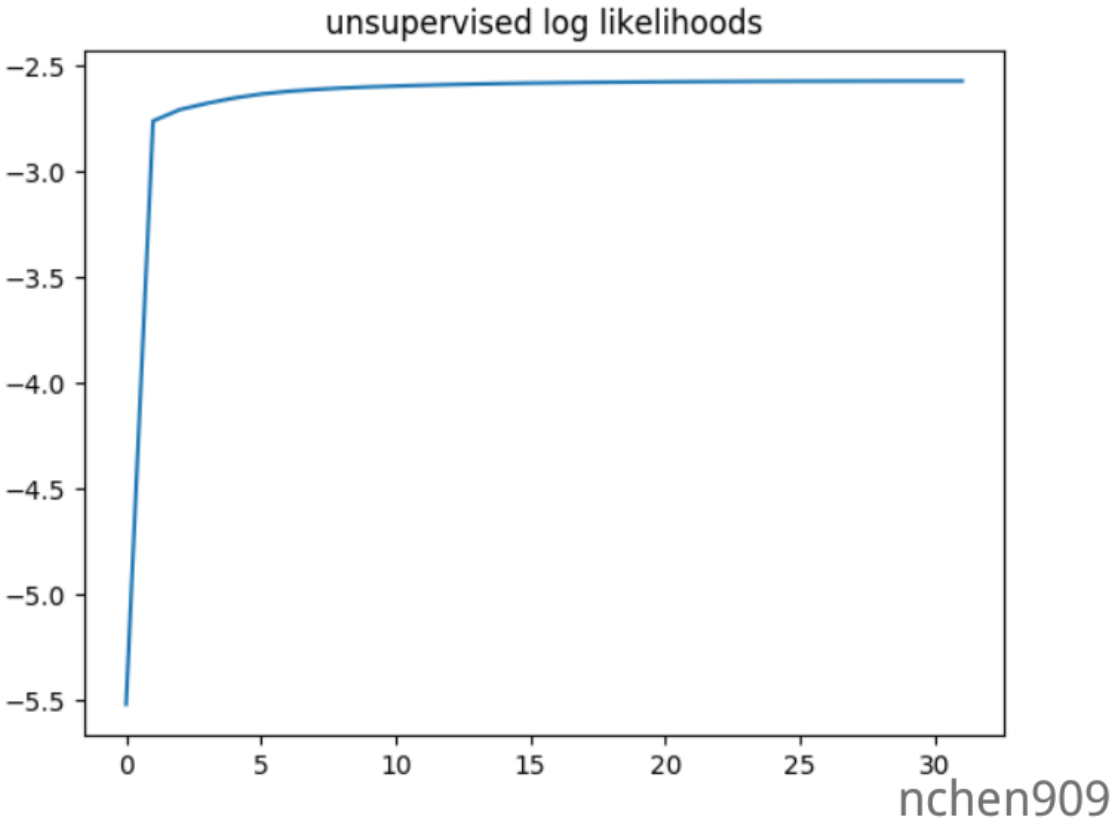


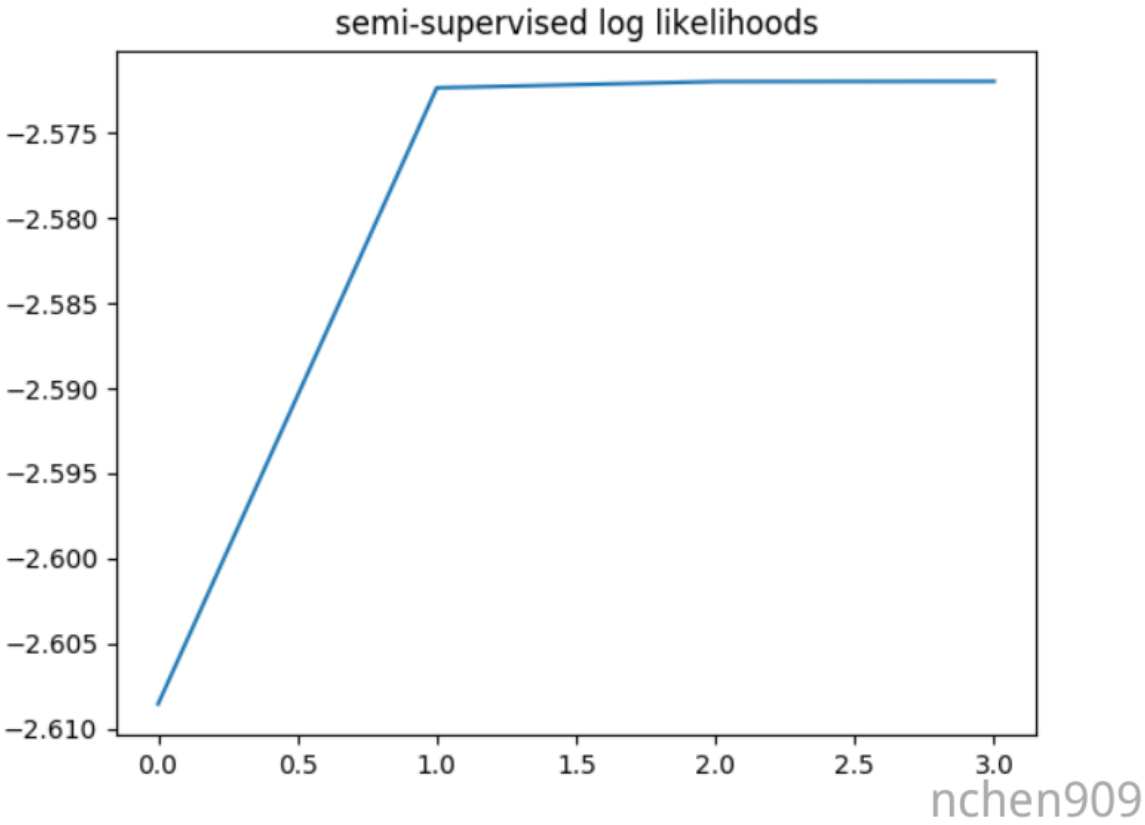
在`learn_params()`中实现从标记数据中学习初始参数。这些学习到的参数将在第一个 E 步骤中使用。

```
def learn_params(x_labeled, y_labeled):
    # *****START OF YOUR CODE (DO NOT DELETE/MODIFY THIS LINE)*****

    pass

    # *****END OF YOUR CODE (DO NOT DELETE/MODIFY THIS LINE)*****
    return params
```

## 输出标签、可视化

除了初始参数外，其他一切都相同，因此我们可以重用之前定义的函数。训练模型、输出对每个用户的预测标签，并与无监督GMM比较收敛步数与速度。下为比较无监督GMM与半监督GMM收敛过程的参考图。




## 与 scikit-learn API结果比较

为验证我们的实现，比较 scikit-learn API 的预测和你的预测结果。要在 scikit-learn 中构建模型，只需调用 GaussianMixture API 并使用未标记的数据拟合模型。`GMM_sklearn()`从 scikit-learn 返回预测和后验。

```
def GMM_sklearn(x):
    model = GaussianMixture(n_components=2,
                            covariance_type='full',
                            tol=0.01,
                            max_iter=1000,
                            weights_init=weights,
                            means_init=means,
                            precisions_init=covariances)
    model.fit(x)
    print("\nscikit learn:\n\tphi: %s\n\tmu_0: %s\n\tmu_1: %s\n\tsigma_0:
%s\n\tsigma_1: %s"
              % (model.weights_[1], model.means_[0, :], model.means_[1, :],
model.covariances_[0, :], model.covariances_[1, :]))
    return model.predict(x), model.predict_proba(x)[:,1]


learned_params = learn_params(x_labeled, y_labeled)
em_api = EM(params=learned_params)
sklearn_forecasts, posterior_sklearn = em_api.GMM_sklearn(x_unlabeled)
print("predict:",sklearn_forecasts)
output_df = pd.DataFrame({'semisupervised_forecasts': semisupervised_forecasts,
                          'semisupervised_posterior':
semisupervised_posterior[:, 1],
                          'sklearn_forecasts': sklearn_forecasts,
                          'posterior_sklearn': posterior_sklearn})

print("\n%s%% of forecasts matched." %
(output_df[output_df["semisupervised_forecasts"] ==
output_df["sklearn_forecasts"]].shape[0] /output_df.shape[0] * 100))
```

```
semi-supervised:
    phi: 0.586349881794546
    mu_0: [-1.04546727 -1.02704636]
    mu_1: [0.98763329 0.99661118]
    sigma_0: [[0.36018609 0.30853357]
 [0.30853357 0.75384027]]
    sigma_1: [[0.7196797  0.1437903 ]
 [0.1437903  0.30853791]]
total steps:  4

scikit learn:
    phi: 0.59647894226803
    mu_0: [-1.06169376 -1.0563389 ]
    mu_1: [0.96408565 0.98206315]
    sigma_0: [[0.35027155 0.29629092]
 [0.29629092 0.73083581]]
    sigma_1: [[0.74510804 0.16156928]
 [0.16156928 0.32021029]]

99.4% of forecasts matched.nchen909
```

结果基本一致。

# 作业要求

(#展示数据、代码)

1. 完成E、M步及迭代的函数实现，输出对每个用户的预测标签，比较无监督GMM与半监督GMM
2. 探索似然收敛过程，迭代过程的分布图，最终收敛的效果等，并可视化（加分项）
3. 思考EM与Kmeans的区别，探索EM算法更多的应用场景，如缺失值填补等（加分项）
4. 撰写实验报告（pdf格式提交）
5. 不可互相抄袭
6. 将代码与运行结果、实验报告打包提交至 https://workspace.jianguoyun.com/inbox/collect/2d69cd0085884fd4923ee54b7195a6cd/submit