HMM Practice

1.1)实验过程见代码

1.2）参数最大似然参数估计，因为含有隐含变量，采用 EM 框架。

E-step 计算完全似然函数对隐含变量的后验分布求期望，计算 gama，epsilon 序列。

M-step 相对于参数 pi，T，theta 求 Q 函数的最大值。更新参数。

以上两步骤交替进行直到收敛。

推断 gama = P(y_n|X), epsilon = P(y_n-1,y_n|X)有两种方法：1.前向后向算法。2.吉布斯采样。

前向后向算法效果：

序列长度 1000，迭代次数 1000。

转移矩阵 T 的估计结果：

[[1.11722383e-01 1.20776559e-10 8.88277617e-01]
[1.07679434e-01 8.86488570e-01 5.83199538e-03]
[7.58864000e-01 2.13373319e-01 2.77626809e-02]]

原始转移矩阵：

[[0.8, 0.2, 0.0]
[0.1, 0.7, 0.2],
[0.1, 0.0, 0.9]]

通过对比发现，对估计的参数标定：

横轴依次是 T1 T2 T3；纵轴依次是 T3,T2,T1

最大加和算法（Vitebi）与加和乘积算法（前向后向算法）的区别只有：

他们都计算需要 alpha_list,只是 alpha_list 的迭代过程中，前向后向是：

alpha*T 是向量与矩阵相乘，内在就是先逐点乘积再求和，对应内积。

而在最大加和算法中，逐点相乘后，并不是求和，而是求最大值。这需要一个 parent_list 记录最大值的位置，以便回溯找到最大状态序列。

sampling list:

[0 0 0 1 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 1 1
1 0 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1
0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 0 1 1 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 1
1 0 1 1 1 1 1 1 1 0 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0
1 1 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 1 0 0 0 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 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 1 1 0 1 1 0 1 0 0 1
0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 1 1 0 0 0 0 0 1 1 1 1 1 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0
0 1 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 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 1 1 0 0 0 0 0 0 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 1 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 0
1 1 0 0 1 1 1 1 1 1 1 0 0 1 1 1 1 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
1 1 0 1 0 0 0 0 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 0 0 0 1
0 1 0 0 0 1 0 1 1 1 1 1 1 1 1 1 0 0 0 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 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 0 1 0 1 1 1 1
1 1 1 1 1 1 1 0 0 0 1 0 0 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1
1 0 0 0 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0 0 0 1 1 0 1 1 1 1
1 0 1 0 0 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 0 1 1 0 1 0 0 0 1 1 1 0
0 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 1 1 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 0
1 0 0 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 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 1 1 1 1 1 0 0 0 0 0 0 0 0 1
1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 1 0 1 0 1 0 0 1 1 1 1
1 0 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 1 1 1 0 1 1 1 1
1 0 1 1 1 0 1 1 1 1 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1
0 0 1 1 0 0 0 0 0 1 1 1 0 1 0 0 0 0 0 0 1 0 1 1 0 0 0 1 0 1 1 1 0 1 1 1 0

0]
最大状态序列：
[0 0 0 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 2 2 2 2 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0
2 2 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 2 2 0 0 0 0 0 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 2 2 0 0 0 0 0 0
0 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0
0 0 0 0 0 0 2 2 0 0 0 0 0 0 0 0 0 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 2 2 0 0 0 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2
2 2 2 2 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0
0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 2
2 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 2 2 2 2
2 2 2 2 2 2 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 2 2 2
2 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 0 0 0 0 0 2 2 2 2 2 0 0 0 0 0 0 2 2 2 2 0 0 0 2 2 2 2 2 2 2 2 2 0
0]
真正状态序列：
[0 0 1 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 2 2 2 2 0 0 1 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 1 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 0 0 0 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 1 0 0 0 1
1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1
0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0
0 1 2 2 2 2 2 2 2 2 2 2 2 2 0 1 1 1 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 1 1 2 0 0 1 1 1 2 2 2 2 2 2 0 0 0 0 0 0 0 0 1 2 0 0 0 0 0 1 0
0 1 1 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 1 1 1 2 2 0 0 0 0 1 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0
1 1 0 0 0 1 1 2 2 0 0 0 0 0 1 1 1 1 2 2 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 0 0 0 0 0 0 0 0 1 1 2 2 0 0 1 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
2 2 0 1 2 2 2 2 2 2 0 0 0 1 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 1 1 2 2
2 2 2 2 0 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 1 1 1
1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 0 0 0 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 1 1 1 1 1 1 2 2 2
2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 1 2 2 2 2 2 2 2 2 2 0 0 0 1 1 1 1 1 2
2 0 0 0 0 0 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 1 1 1 1 1 1 2
2 0 0 0 0 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 1 1 1 0 0 0 0 0 1 2 2 2
2 2 2 2 2 2 0 0 0 1 1 1 1 2 2 2 2 2 2 0 0 1 1 1 2 2 2 2 2 2 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
0 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 1

1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 1 1 1 1 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 0 1 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 1 2 2 2 2 2 2 2 2 2 2 2
0 0 0 0 0 0 1 1 2 2 2 2 0 0 0 0 0 0 0 0 1 1 1 2 0 1 1 1 1 1 2 2 0 0 0 0 0
0]
matching rate：0.78

长度为 10000 时，迭代次数 100，转移矩阵收敛到：
[[0.46835248 0.50233874 0.02930877]
 [0.44579876 0.31914616 0.23505508]
 [0.04998488 0.08124883 0.86876629]]
与真实矩阵相差较大。
运行维特比算法时，相应的 match rate 在 0.73 左右徘徊。

1.3）每次估计出来的参数属于哪个状态都是变动的，没有想到合适的标定方法。方差没做。

2.1）用 Gibbs 采样的方法估计的参数很不准。长度 1000 的序列，MLE 每次迭代中吉布斯采样 200 次，取后 100 次结果作为计算 gama，epsilon 的数据。下面为估计转移概率矩阵 MLE 算法迭代 100 次时的结果：
[[0.08409778 0.39429735 0.52160487]
 [0.29867453 0.42026743 0.28105804]
 [0.34233917 0.26247855 0.39518228]]

伪代码：
```
MLE：
	初始化 pi, T, theta
	for :
		gama_list, epsilon_list = get_gama_epsilon(pi, T, theta)；
		利用 gamma, epsilon 更新 pi，T，theta；
	for end
MLE end


get_gama_epsilon(pi, T, theta):
	随机初始化一个 Y 序列；
	for ：（吉布斯采样循环）
		初始化 t = N；
		for t > 0:  (采样每一个 y_t）
			计算 P(y_t|X)  正比于 P(Y_t|Y_t-1)*P(Y_t+1|Y_t)*P(X_t|Y_t)；
			根据上 P(y_t|X)采样 Y_t；
		存储本次采样序列 Y；
	统计所有采样序列 Y，得到 gamma，epsilon：
	gamma[i]即采样序列 Y 在位置 i 处出现的所有状态的频率；
	epsilon[i]即在采样序列 Y 在 i-1 到 i 状态跳变的频率；
return ：gamma，epsilon；
```
2.2）链长度 100，最大似然迭代 200 次，每一次 MLE 迭代里吉布斯采样整个链 200 次，取后 100 次的所为统计对象。实验结果如下：

转移概率矩阵 T 的估计：
iter：100
T：
```
[[ 0.45193303  0.21905853  0.32900844]
 [ 0.28953442  0.41363768  0.2968279 ]
 [ 0.48279304  0.30971734  0.20748962]]
```

iter：200
T：
```
[[ 0.40068952  0.24902822  0.35028226]
 [ 0.2766782   0.43997596  0.28334584]
 [ 0.39892087  0.35158258  0.24949655]]
```

相比之下，用前向后向算法估计，相同链长度，最大似然迭代 200 次，后转移概率矩阵的估计效果：
iter：200
T：
```
[[  7.96880292e-01   2.03119692e-01   1.54557459e-08]
 [  5.92132553e-08   1.00914553e-01   8.99085388e-01]
 [  2.45660536e-01   5.57285032e-01   1.97054432e-01]]
```
大约就是：
```
[[0.8   0.2    0]
 [0    0.1    0.9]
 [0.24   0.56  0.2]]
```
从上面可以看出前向后向算法参数估计效果很好，吉布斯采样近似估计很差。
可能的原因有
1.采样长度不够。
2.多链取平均。

因此实验吉布斯采样迭代次数变成 500，300 次以后链作为统计对象。其余参数不变，转移矩阵在 MLE 迭代 200 次时候估计效果：
iter: 200
T:
```
[[ 0.28937043  0.39387948  0.31675009]
 [ 0.28361764  0.44474342  0.27163894]
 [ 0.32577332  0.27939762  0.39482906]]
```
迭代次数 400 次：
iter: 400
T:
```
[[ 0.37265166  0.28619676  0.34115158]
 [ 0.27365409  0.45117668  0.27516922]
 [ 0.34325026  0.28619666  0.37055308]]
```
吉布斯链变长了效果依然很差。
把链变长到 1000，吉布斯迭代 2000 次，MLE 迭代 55 次的效果依然很差。
iter: 55
T:
```
[[ 0.37590303  0.33152869  0.29256828]
 [ 0.27029092  0.4491388   0.28057028]
 [ 0.38866919  0.4022148   0.20911601]]
```

实验吉布斯采样分 10 个链，均迭代 200 次，取后一百次作为统计对象。EM 迭代 100 次与 200 次结果如下：

iter: 100
T :
[[9.99979881e-01 1.00596377e-05 1.00596377e-05]
[3.45952378e-01 3.27444652e-01 3.26602971e-01]
[3.33333333e-01 3.33333333e-01 3.33333333e-01]]
iter: 200
T :
[[9.99979783e-01 1.01085682e-05 1.01085682e-05]
[3.47627319e-01 3.29136842e-01 3.23235838e-01]
[3.33333333e-01 3.33333333e-01 3.33333333e-01]]
实验第二次：
iter :200
T:
[[9.98879430e-01 1.11027249e-03 1.02974584e-05]
[3.55299598e-01 3.28193881e-01 3.16506521e-01]
[3.33333333e-01 3.33333333e-01 3.33333333e-01]]
从两次实验看出，多链模拟也不能提升效果。

2.3)
由 MEMM 的马尔科夫毯可知，吉布斯采样中 Yt 的条件概率分布如下图：

$$P(Y_t = y \mid X, Y_{-t}) \propto P(Y_t \mid Y_{t-1}, X_t) \cdot P(Y_{t+1} \mid X_{t+1}, Y_t)$$
$$= P(Y_t = y \mid Y_{t-1} = y_{t-1}, X_t = x_t) \cdot P(Y_{t+1} = y_{t+1} \mid X_{t+1} = x_{t+1}, Y_t = y)$$

因此用吉布斯采样推断进行 MEMM 参数估计算法如下：
伪代码：
```
MLE：
	初始化 pi, T, theta
	for :
		gama_list, epsilon_list = get_gama_epsilon(pi, T, theta)；
		利用 gamma, epsilon 更新 pi，T，theta；
	for end
MLE end


get_gama_epsilon(pi, T, theta):
	随机初始化一个 Y 序列；
	for ：（吉布斯采样循环）
		初始化 t = N；
		for t > 0:  (采样每一个 y_t )
			计算 P(y_tlX,Y_-t) ；
			根据上 P(y_tlX)采样 Y_t；
		存储本次采样序列 Y；
	统计所有采样序列 Y，得到 gamma，epsilon：
	gamma[i]即采样序列 Y 在位置 i 处出现的所有状态的频率；
	epsilon[i]即在采样序列 Y 在 i-1 到 i 状态跳变的频率；
return ：gamma，epsilon；
```