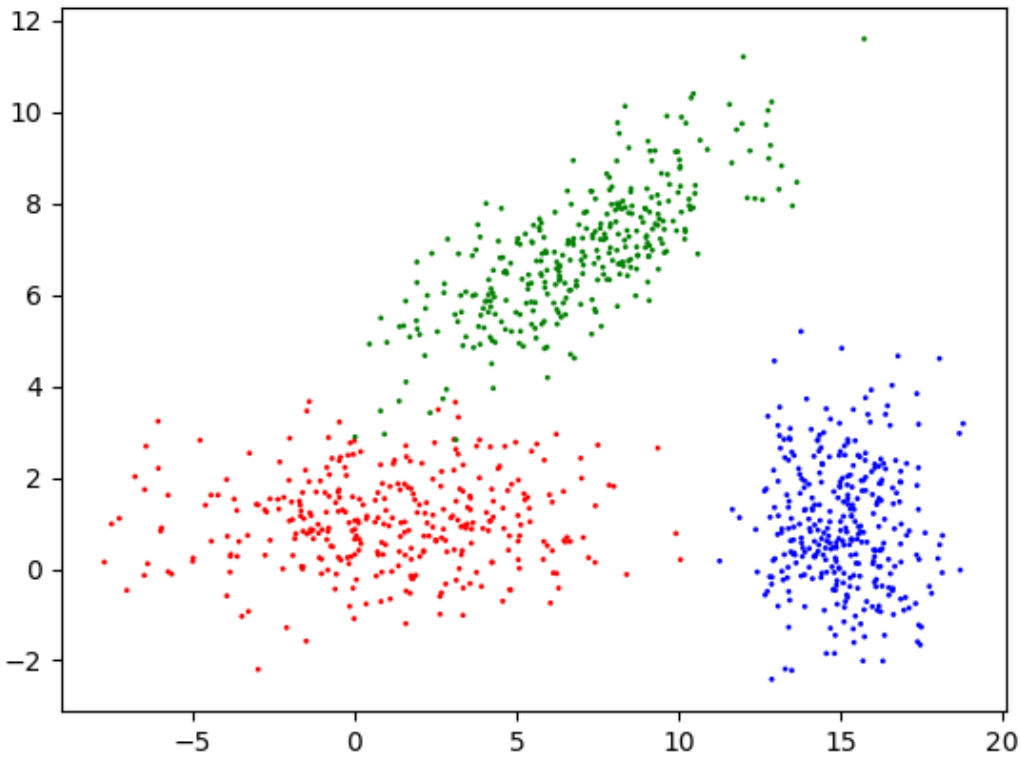
HW3 实验报告

2015011313 徐鉴劲 计54

Problem1

问题描述：给定一个数据集进行无监督学习，确定一个产生数据的分布。类别数量是通过假设预先确定的。

数据图(三类均等)



Maximum Likelihood Estimation

假设这个数据是由多个概率混合而成：$P(x) = \sum P(x|\omega_i)P(\omega_i)$。

其中每一个概率假设为高斯分布：$P(x|\omega_i) = \frac{1}{\sqrt{(2\pi)^d|\Sigma|}}e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}$.

它实际上是一个关于μ和Σ可导的函数，其中μ是N维向量，Σ是$N \times N$矩阵。

所以将它表示成一个函数的形式：$P(x|\omega_i) = f(x;\mu,\Sigma)$。

那么整个概率分布就是$P(x) = \sum t_i\mathcal{N}(x;\mu_i,\Sigma_i)$，其中$t_i$是满足$0 \leq t_i \leq 1$且加起来为1的标量。

构造损失函数$l = -\sum_k lnP(x_k)$，然后我们可以对参数求导，进而进行优化。

实现方式

由于求导数的步骤比较麻烦，我采用了tensorflow中的自动求导功能。

为了将上述表达式实现在tensorflow中，需要做以下必要的处理：

1. 矩阵化$l = -\sum_k lnP(x_k)$中的求和部分。因为tensorflow不支持挨个求和的操作。
2. 将Σ参数限制为正定矩阵。
3. 将t_i参数的限制完成。

矩阵化

```
def lnp_x_mu_sigma(x, mu, sigma, t):
    res = 0
    N = mu.get_shape().as_list()[0]

    for i in range(N):
        m = mu[i:i+1, :]
        s = sigma[i, :, :]

        det = tf.matrix_determinant(s)
        inv = tf.matrix_inverse(s)
        n_sample = tf.cast(tf.shape(x)[0], tf.float64)

        a1 = tf.sqrt( (2 * np.pi) ** N * det)
        a2 =  -0.5 * tf.reduce_sum(tf.matmul(x - m, inv) * (x - m), axis=1)

        res += tf.exp(a2) / a1 * t[i]
    res = tf.reduce_sum(tf.log(res))
```

正定化

利用了A^TA是正定对称矩阵的事实。

```
sigma = tf.Variable(init_sigma)
psigma = tf.matmul(sigma, sigma, transpose_a=True)
```

t_i的限制

将实数范围内的变量通过函数$x^2 + 1$映射到$(1, +oo)$上，然后归一化。

```
t = tf.Variable(np.ones((N,), dtype="float64"))
et = (1+t*t) / tf.reduce_sum(1+t*t)
```

吉布斯采样优化

我发现同时优化t和μ和Σ容易导致t过早收敛到一个错误的直，所以我采用两步迭代优化：

在loss>8000的时候只优化μ和Σ，然后在loss<8000的时候交替优化μ, Σ，和t

求导与优化器

最开始的时候我采用的是朴素的梯度下降法，但是学习率的设置是一个问题，所以我使用了一种自适应学习率算法：Adam.

然后我还对Adam的学习率进行了规划，根据不同loss的大小选择对应的学习率。

实验结果

题目中涉及到的数据集一共有四个：

1. 1000个数据点下，概率均等。
2. 1000个数据点下，概率不均等。
3. 300个数据点下，概率均等。
4. 300个数据点下，概率不均等。

但是题目中只要求做3个实验，于是我取了：

a. 1000个数据点下，概率均等。

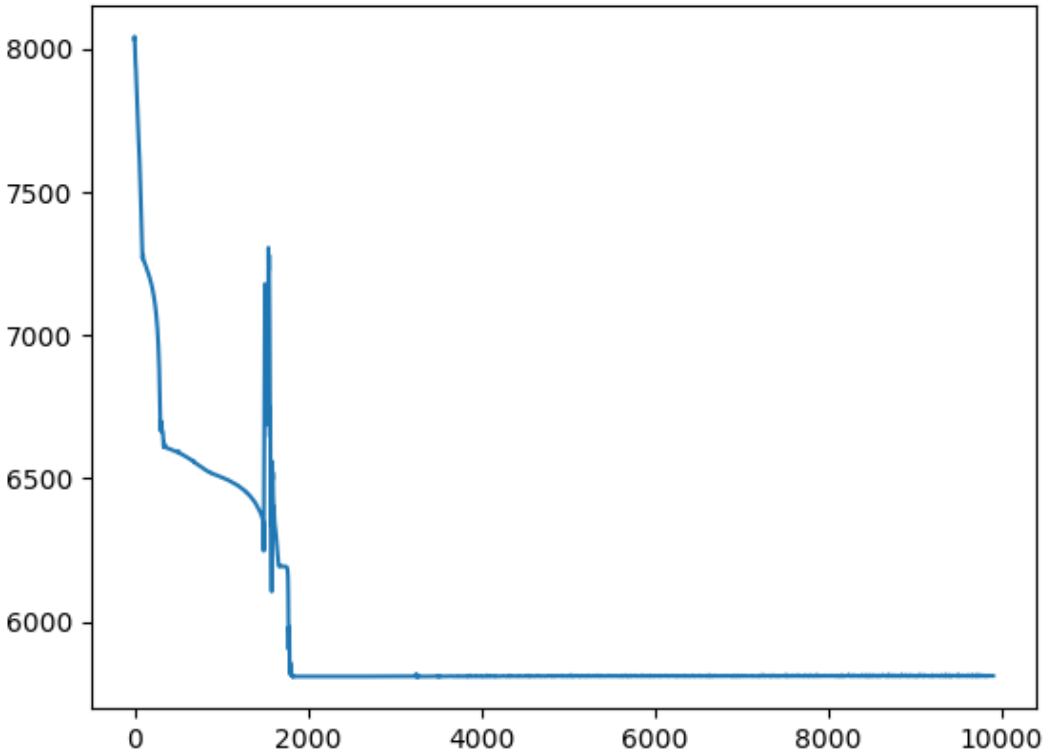
b. 1000个数据点下，概率不均等。

c. 300个数据点下，概率均等。

因为数据点很少的时候，概率是否均等并没有太大关系，因为本身数据的随机性就很大了。

对于每一个实验结果没有进行多次重复。

训练loss图



可以看到loss的下降并不是很平滑，这说明了MLE并不是凸优化，而且tf存在一定的数值不稳定

参数估计结果（a)

先验权重	**1**	**2**	**3**
t_i	0.39054	0.33358	0.27588

先验权重与数据吻合，正确。

μ	x	y
μ_1	7.28943	7.06694
μ_2	15.00692	1.01494
μ_3	0.89612	1.01104

结果与(7, 7), (15, 1), (1, 1)十分接近，剩下不准确的原因可能是数据的随机性与算法未收敛到最优解。

Σ_0	
7.88187	2.79020
2.79020	1.87916
Σ_1	
2.17459	-0.10019

-0.10019	1.79766
Σ_2	
11.56872	0.11756
0.11756	0.92613

生成数据的Σ如下

Σ_0	
8	3
3	2
Σ_1	
2	0
0	2
Σ_2	
12	0
0	1

他们十分接近，同样说明了估计的有效性。

参数估计结果（b)

先验权重	**1**	**2**	**3**
t_i	0.30055	0.55747	0.14198

与设定的先验很接近。

μ	x	y
μ_1	6.88429	6.95063
μ_2	0.79475	1.04674
μ_3	15.04197	0.91565

结果与(7, 7), (15, 1), (1, 1)十分接近，剩下不准确的原因可能是数据的随机性与算法未收敛到最优解。

Σ_0	
7.62127	2.50899
2.50899	1.70573

Σ_1	
10.92268	-0.10753
-0.10753	1.01599
Σ_2	
2.20195	0.18426
0.18426	2.11459

生成数据的Σ如下

Σ_0	
8	3
3	2
Σ_1	
12	0
0	1
Σ_2	
2	0
0	2

有一定误差。

参数估计结果（c)

μ	x	y
μ_1	7.20120	0.37176
μ_2	9.39782	1.55818
μ_3	6.40659	6.78052

Σ_0	
59.41734	-5.46793
-5.46793	1.70666
Σ_1	
48.23648	4.50168
4.50168	1.80242

Σ_2	
7.60814	2.79271
2.79271	1.87219

在只有300个数据点的情况下，所有参数的估计差距均很大。

Bayesian Estimation

问题描述：给定了一个具有类别的数据集，估计产生它的高斯分布。

课上介绍了单元高斯分布建模参数，并学习μ的过程。

但是数据集产生分布中的Σ并没有进行学习。

而且这道题中要求进行二维高斯分布建模。

多元BE理论基础

我们的目的是得到参数的后验概率。

$P(\theta|\mathcal{D}) = \alpha\Pi_k P(x_k|\theta)p(\theta).$

其中$P(x|\theta) \sim \mathcal{N}(x;\mu,\Sigma)$。

$p(\theta)$是一个先验假设，此处我们不知道任何情况，它是一个具有0均值和$+oo$方差的正态分布函数，$p(\theta_i) \sim \mathcal{N}(\theta_i; m_i, s_i^2)$，$m_i = 0$，$s_i = +oo$。

所以
$P(\theta|\mathcal{D}) = \left[\alpha\Pi_k\mathcal{N}(x_k;\mu,\Sigma)\right]\left[\Pi_i\mathcal{N}(\theta_i;m_i,s_i^2)\right] = \left[\alpha\Pi_k\mathcal{N}(x_k;\mu,\Sigma)\right]\left[\mathcal{N}(\theta_i;Mm_i,Ms_i^2)\right].$

其中M是参数的个数。因为参数的分布假设成一样的了。

对于前一项：$\Pi_k\mathcal{N}(x_k;\mu,\Sigma)$

=$\Pi_k \frac{1}{\sqrt{(2\pi)^d|\Sigma|}} e^{-\frac{1}{2}(x_k-\mu)^T\Sigma^{-1}(x_k-\mu)}$

=$\left(\frac{1}{\sqrt{(2\pi)^d|\Sigma|}}\right)^N e^{-\frac{1}{2}\sum_k(x_k-\mu)^T\Sigma^{-1}(x_k-\mu)}$

我们需要化简指数项：$\sum_k(x_k-\mu)^T\Sigma^{-1}(x_k-\mu)$

=$\sum_k x_k^T\Sigma^{-1}x_k - x_k^T\Sigma^{-1}\mu - \mu^T\Sigma^{-1}x_k + \mu^T\Sigma^{-1}\mu$

=$\sum_k \left[x_k^T\Sigma^{-1}x_k - 2x_k^T\Sigma^{-1}\mu\right] + N\mu^T\Sigma^{-1}\mu$

首先我们以μ为主元进行整理：

$\sum_k (x_k-\mu)^T\Sigma^{-1}(x_k-\mu)$

$=N\mu^T\Sigma^{-1}\mu-2(\sum_k x_k^T)\Sigma^{-1}\mu+\sum_k x_k^T\Sigma^{-1}x_k$

$=(\mu-\frac{1}{N}\sum_k x_k)^T N\Sigma^{-1}(\mu-\frac{1}{N}\sum_k x_k)-\frac{1}{N}(\sum_k x_k)^T\Sigma^{-1}\sum_k x_k+\sum_k x_k^T\Sigma^{-1}x_k$

所以：

$P(\mu|\mathcal{D})=\alpha' p(\theta)e^{-\frac{1}{2}(\mu-\frac{1}{N}\sum_k x_k)^T N\Sigma^{-1}(\mu-\frac{1}{N}\sum_k x_k)}$

$=\alpha' p(\theta)\mathcal{N}(\frac{1}{N}\sum_k x_k, N\Sigma^{-1})$

$=\alpha'\mathcal{N}(\theta_i; Mm_i, Ms_i^2)\mathcal{N}(\frac{1}{N}\sum_k x_k, N\Sigma^{-1})$

$=\alpha'\mathcal{N}(\frac{1}{N}\sum_k x_k, \frac{\Sigma}{N})$

然后我们对Σ为主元进行整理。

设$(\Sigma^{-1})_{ij}=b_{ij}$，同理设$(x_k)_i=x_{ki}$，再令$\sum_k x_k=c$

$\sum_k (x_k-\mu)^T\Sigma^{-1}(x_k-\mu)$

$=\sum_k\sum_j\sum_i(x_{ki}-\mu_i)b_{ij}(x_{kj}-\mu_j)$

$=\sum_k\sum_j\sum_i b_{ij}(x_{ki}x_{kj}-x_{ki}\mu_j-x_{kj}\mu_i+\mu_i\mu_j)$

$=\sum_{ij} b_{ij}(\sum_k x_{ki}x_{kj}-c_i\mu_j-c_j\mu_i+N\mu_i\mu_j)$

$\sum_k x_{ki}x_{kj}=(X^TX)_{ij}$,

$c_i\mu_j=(c\mu^T)_{ij}, c_j\mu_i=(c\mu^T)_{ji}$,

$\mu_i\mu_j=(\mu\mu^T)_{ij}$

所以：

$\sum_k (x_k-\mu)^T\Sigma^{-1}(x_k-\mu)$

=

$\sum_{ij} b_{ij}(X^TX+N\mu\mu^T-c\mu^T-\mu c^T)_{ij}=\sum_{ij} b_{ij}(X^TX+N\mu\mu^T-c\mu^T-\mu c^T+cc^T-cc^T)_{ij}$

$=\sum_{ij}\Sigma^{-1}\odot(X^TX-cc^T+\frac{1}{N}(N\mu-c)(N\mu-c)^T)_{ij}$

将$\sum_{ij}$从指数项目上拆解下来，可以化成:

$P(\Sigma)=\alpha''\Pi_{ij}e^{-\frac{1}{2}\Sigma_{ij}^{-1}(X^TX-cc^T+\frac{1}{N}(N\mu-c)(N\mu-c)^T)_{ij}}$

实验结果

参数估计结果（a)

μ	x	y
μ_1	0.94192	1.03805
μ_2	7.00880	6.92195
μ_3	15.05649	1.01704

Σ_0	
11.83856	-0.33173
-0.33173	0.99943
Σ_1	
7.73065	3.08681
3.08681	2.12755
Σ_2	
1.83321	-0.11916
-0.11916	1.97818

参数估计结果(b)

μ	x	y
μ_1	0.94192	1.03805
μ_2	7.00880	6.92195
μ_3	15.05649	1.01704

Σ_0	
11.83856	-0.33173
-0.33173	0.99943
Σ_1	
7.73065	3.08681
3.08681	2.12755
Σ_2	
1.83321	-0.11916
-0.11916	1.97818

参数估计结果(c)

μ	x	y
μ_1	1.08155	1.00883
μ_2	7.21747	7.19486
μ_3	14.96565	1.21260

Σ_0	
12.15760	-0.11317
-0.11317	0.97244
Σ_1	
8.02456	3.13810
3.13810	1.99747
Σ_2	
1.89756	0.04821
0.04821	1.80511

实验结果对比

BE在大、小数据量上表现都不错，但是它是一个有监督的方法。

MLE在大数据集上表现不错，在小数据集上误差较大，其优点在于是基本无监督的，只用设置一个类的数量，和KNN是类似的。

Problem 2: BE

(a)

$P(D|\theta) = P(x_1, \ldots, x_n|\theta) = \Pi_i P(x_i|\theta)$

$=\Pi_i \Pi_j \theta_i^{x_{ij}} (1-\theta_i)^{1-x_{ij}}$

$=\Pi_i \theta_i^{\sum_j x_{ij}} (1-\theta_i)^{\sum_j (1-x_{ij})}$

$=\Pi_i \theta_i^{s_i} (1-\theta_i)^{n-s_j}$

(b)

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{\int_0^1 P(D|\theta)P(\theta)d\theta}$$

因为$P(\theta)$是均匀分布，所以分母中的θ可以提出来

$$P(\theta|D) = \frac{P(D|\theta)}{\int_0^1 P(D|\theta)d\theta}$$

$$= \frac{\Pi_i \theta_i^{s_i}(1-\theta_i)^{n-s_j}}{\int_0^1 \Pi_i \theta_i^{s_i}(1-\theta_i)^{n-s_j} d\theta}$$

$$= \Pi_i \frac{(n+1)!}{s_i!(n-s_i)!}\theta_i^{s_i}(1-\theta_i)^{n-s_j}$$

(c)

When n=1 and d=1, x = 0 or 1

$$P(\theta|D) = P(\theta|x) = \frac{2}{x!(1-x)!}\theta^x(1-\theta)^{1-x}$$

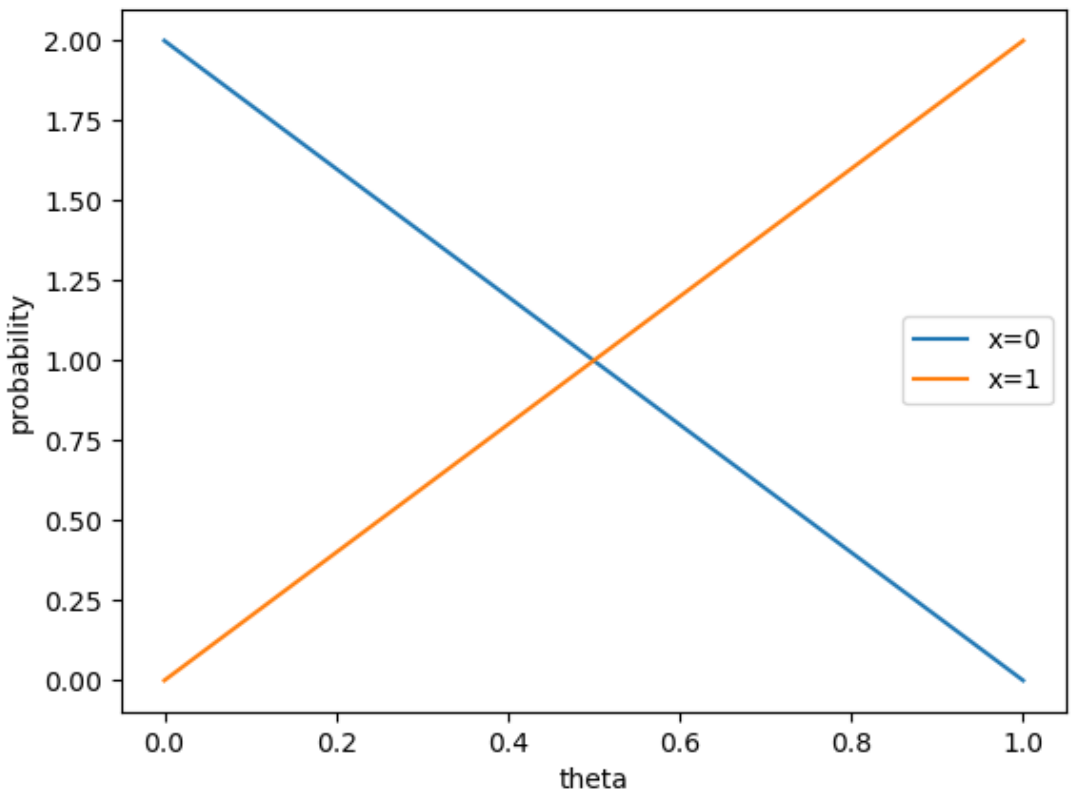
$$= 2\theta^x(1-\theta)^{1-x}$$

Thus,

$$P(\theta|x=0) = 2(1-\theta)$$

$$P(\theta|x=1) = 2\theta$$

The figure below shows the distribution



(d)

$P(x|D)$ is the classification confidence.

$P(x|D) = \int_0^1 P(x|\theta)P(\theta|D)d\theta$

$= \int_0^1 \Pi_i \theta_i^{x_i}(1-\theta_i)^{1-x_i} \frac{(n+1)!}{s_i!(n-s_i)!}\theta_i^{s_i}(1-\theta_i)^{n-s_i}d\theta$

$= \left[\Pi_i \frac{(n+1)!}{s_i!(n-s_i)!}\right] \int_0^1 \Pi_i \theta_i^{x_i+s_i}(1-\theta_i)^{1+n-x_i-s_i}d\theta$

As every θ_i is independent with each other, so the integration can be completely one by one

$= \left[\Pi_i \frac{(n+1)!}{s_i!(n-s_i)!}\right] \Pi_i \int_0^1 \theta_i^{x_i+s_i}(1-\theta_i)^{1+n-x_i-s_i}d\theta_i$

$= \Pi_i \frac{(n+1)!}{s_i!(n-s_i)!} \frac{(x_i+s_i)!(1+n-x_i-s_i)!}{(n+2)!}$

$= \Pi_i [\frac{s_i+1}{n+2}]^{x_i} [\frac{n+1-s_i}{n+2}]^{1-x_i}$

$= \Pi_i [\frac{s_i+1}{n+2}]^{x_i} [1-\frac{s_i+1}{n+2}]^{1-x_i}$

(e)

If we view $P(x|D)$ as some distribution obtained by $P(x|\hat{\theta})$, we can see from the formulae $\Pi_i [\frac{s_i+1}{n+2}]^{x_i} [1-\frac{s_i+1}{n+2}]^{1-x_i}$ that $\hat{\theta} = \frac{s_i+1}{n+2}$. In this way $P(x|D) = \Pi_i \hat{\theta}_i^{x_i}(1-\hat{\theta}_i)^{1-x_i}$.

However if you use an empirical estimation that each class probability is its frequency statistics, then we should obtain $\theta_i = \frac{s_i}{n}$, which is different than the estimation of BE.

Problem 3: HMM

(a)

Let transition matrix be $a_{ij} = P(x_{t+1} = j | x_t = i)$.

And suppose A, B, C, D = 1, 2, 3, 4.

Take the statistics of the training data:

1. ω_1

$P(x_{t+1}/x_t, \omega_1)$	$x_{t+1}=1$	$x_{t+1}=2$	$x_{t+1}=3$	$x_{t+1}=4$
$x_t = 1$	0.23529	0.47059	0.17647	0.11765
$x_t = 2$	0.25000	0.20000	0.40000	0.15000
$x_t = 3$	0.00000	0.31250	0.25000	0.43750
$x_t = 4$	0.00000	0.11111	0.22222	0.66667

Prior:

1	2	3	4
0.14516	0.29032	0.27419	0.29032

2. ω_2

$P(x_{t+1}/x_t, \omega_2)$	$x_{t+1}=1$	$x_{t+1}=2$	$x_{t+1}=3$	$x_{t+1}=4$
$x_t=1$	0.57143	0.21429	0.07143	0.14286
$x_t=2$	0.44444	0.27778	0.16667	0.11111
$x_t=3$	0.21429	0.35714	0.14286	0.28571
$x_t=4$	0.17391	0.13043	0.26087	0.43478

Prior:

1	2	3	4
0.14516	0.29032	0.27419	0.29032

(b)

Let s denote the sequence and c_i be its i-th element. Use minimum error probability classification:

If $P(s|\omega_1) > P(s|\omega_2)$ then decide ω_1, otherwise decide ω_2.

$P(s|\omega) = P(x_1, \ldots, x_n|\omega) = P(x_1 = c_1)\Pi_{i=2}P(x_{i+1} = c_{i+1}|x_i = c_i, \omega)$.

To avoid underflow, use log-probability:

$J = -lnP(s|\omega) = \sum -lnP(x_{i+1} = c_{i+1}|x_i = c_i, \omega) - lnP(x_1 = c_1)$

Choose the smaller one.

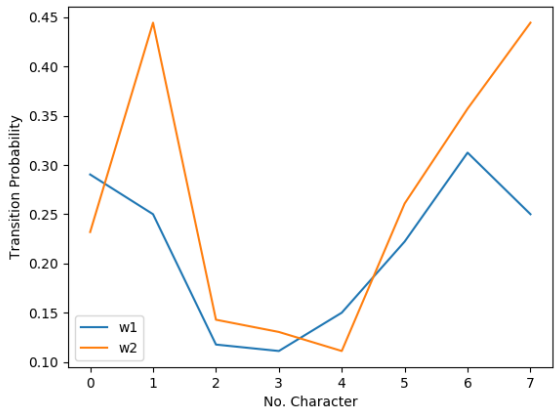
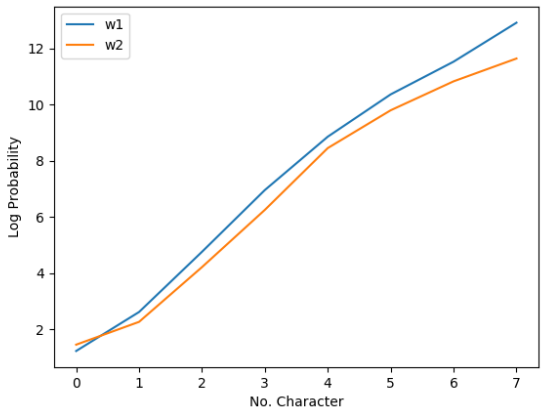
Classification result:

1: ABBBCDDD	2: DADBCBAA	3: CDCBABA	4: ADBBBCD
2	1	2	1

(c)

The classification process is plot as the figure below:

Probability of each product factor	Probability of prefix string

Probability of each product factor	Probability of prefix string
	

We can see that the same sequence will obtain different `step probability` at each step w.r.t. to different category.

Note that changing prior will not affect other `step probabilities`, so change the prior will make the log probability shift up-and-down as a whole.

Using original prior, we have:

$$J_1 = J_1 + lnP(x_0 = c_0|\omega_1) - lnP(x_0 = c_0|\omega_1) = \hat{J}_1 - lnP(x_0 = c_0|\omega_1)$$

and

$$J_2 = J_2 + lnP(x_0 = c_0|\omega_2) - lnP(x_0 = c_0|\omega_2) = \hat{J}_2 - lnP(x_0 = c_0|\omega_2)$$

subtract them:

$$\Delta = \hat{J}_1 - \hat{J}_2 + lnP(x_0 = c_0|\omega_2) - lnP(x_0 = c_0|\omega_1)$$

To make two category output the same log probility based on a different prior, we have:

$$\Delta' = \hat{J}_1 - \hat{J}_2 + lnP_2 - lnP_1 = 0$$

This leads to

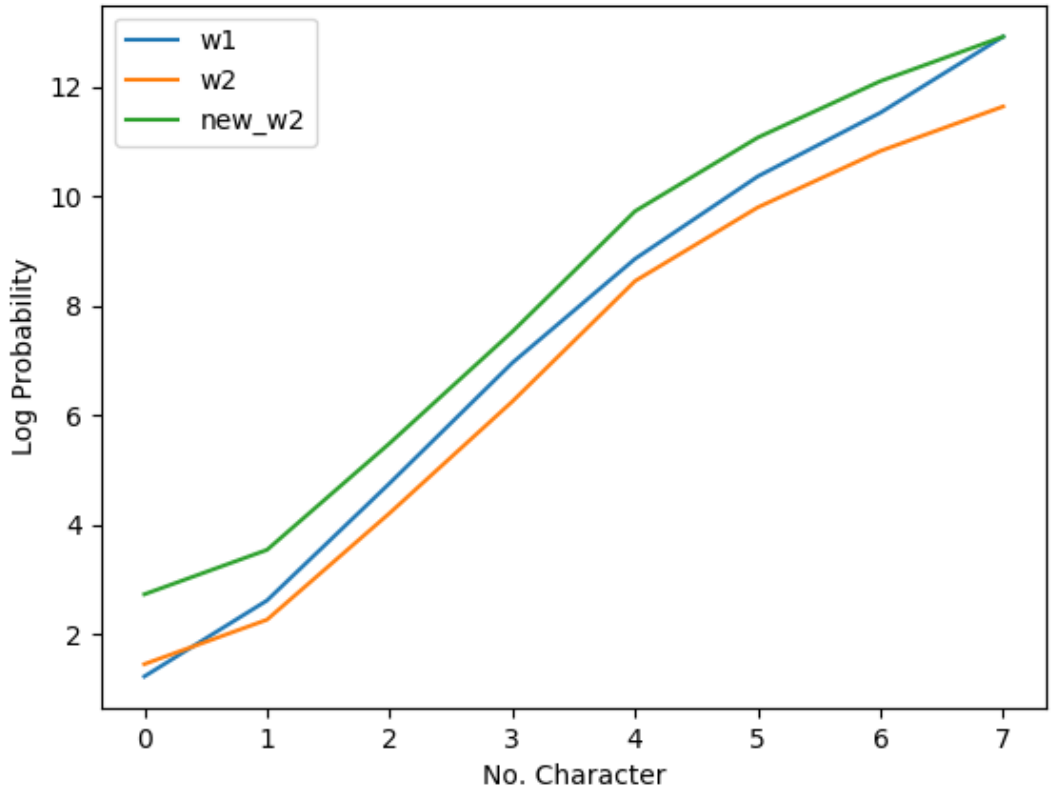
$$lnP_1 - lnP_2 = -lnP(x_0 = c_0|\omega_2) + lnP(x_0 = c_0|\omega_1) + \Delta$$

As long as new prior P_1 and P_2 satisty this equation, the two category's probability will be the same.

Suppose that category 1's prior remains the same, now we can get an exact solution of category 2's new prior.

$P_2 = exp(lnP(x_0 = c_0|\omega_1) + \Delta)$, in which $\Delta = J_1 - J_2$.

Now the classification process looks like



From the figure we confirm that this two category have the same output probability.

运行代码

第一题：

python prob1.py

有可能因为初始化太过极端，造成计算溢出（inf），此时重新运行一遍，问题一般就没有了。

其他题目：

```
python prob2.py
python prob3.py
```