Phasing 项目报告

目录

Phasing 项目报告 1
1. 问题描述 2
2. 问题分析和基本假设 2
3. 模型描述 3
4. F-B 算法细节 4
4.1 Alpha 参数 6
4.2 Beta 参数 8
4.3 F-B 算法总结 9
5. 观测序列的概率分布 10
6. 维特比算法 12
概率空间坍缩 12
最大似然分布 13
算法细节 13
7 MCMC 和吉布斯抽样 14
吉布斯抽样 14
问题 15
8 总结 16
9 缺陷和改进空间 16
超参数向量ρ 16
训练集 H 17
10 致谢 17

1. 问题描述

将一个人的基因序列 phase 为组成的两条单链是基于基因信息的生物学分析的基础。

给定一个人的基因序列，在这个长序列中选取若干位点作为关键位点，生成一条关键位点子序列。类似整张拼图中取出部分碎片。

接下来我们的任务是将这个子序列拆分为两个单链，目的是为了下一步 imputation 能够依据单链子序列的信息推断出完整的基因单链。

在这个问题中，首先我们并不尝试去判断哪一条单链遗传自父亲，哪一条单链遗传自母亲，我们仅仅尝试去把它们分开，这两条单链已经能够提供我们所需要的全部信息。同时由于下一步 imputation 并不基于完整的子序列进行推断，而是把子序列分成若干片段，针对每一个片段进行推断。因此我们并不需要把一个子序列完整的 phase 正确，而是只需要把子序列中的片段 phase 正确。

综上，我们的问题就是给定任意基因子序列，将其片段式地 phase 为两条正确的单链。

例 1 是一个文字上的描述，通过这个例子解释刚才部分的内容。更为详细的数学描述见例 2

例 1：

某人的 22 号染色体，我们测试出来获取到的信息是一个长度为 111 万的向量，即基因序列。我们在其中取了 8512 个位点的信息得到一个新的向量，即基因子序列。同时这个新的基因子序列含有 1511 个杂合子。那么接下来我们的任务是正确地将这个基因子序列片段式地 phase 为两条单链，根据遗传学原理纯合子我们是不需要进行 phase 的，那么我们所需要关注的就是那 1511 个杂合子的拆分方法。

2. 问题分析和基本假设

尽管如此，这个问题也是一个颇为复杂的问题。首先我们可以将这个问题看作某种拆分方法的估计问题，那么很自然的思想是尝试去找到某种拆分方法的最大似然估计。按照这个思路我们可以首先考虑暴力算法，但是维度的诅咒困扰着我们，一段含有 R 个杂合子的基因子序列的理论拆分方法有2^R种，指数级别的复杂度让我们无法通过穷举暴力算法去计算这个最大似然概率。因此我们需要一些更为巧妙地方式去绕开维度的诅咒，将这个 NP 完全问题转化为一个多项式时间内可解决的问题。

根据算法作者的假设，基因位点间的关系满足马尔可夫性质，即位点信息没有任何历史依赖性。同时如果我们拥有一些已经被证明为正确 phase 的结果，那么我们可以依赖这些先验知识对于拆分方法进行估计。因此，一个很自然的逻辑是去利用隐马尔可夫模型，HMM 对于这个问题进行建模分析。

在模型中，我们将这些先验单链信息看作马尔可夫链中的隐变量，即真实变量。同时将某种拆分方法下的单链视作观测变量，注意由于基因突变的缘故我们的观测并非完美观测。接下来我们尝试去解决隐马尔可夫模型中的最大似然观测估计问题，这是一个 HMM 中的经典问题。如果计算成功，我们可以将这个估计解作为我们原问题的解，那么我们就解决了原问题。接下来的部分我们会把完整的数学描述给出。

3. 模型描述

一个隐马尔科夫模型 HMM 有三组参数，两组变量，我们用 Z 表示隐变量，X 表示观测变量。注意 Z 和 X 是向量，因此我们得到了参数空间

$$HMM: \lambda = (\pi, \mathrm{A}, \mathrm{B})$$

其中π是 initial probability，A是 transition probability，B是 emission probability

同时由于这个马尔可夫链是一个非齐次马尔可夫链，因此对于每一个位点 m 我们都有不同 transition probability matrix。这个假设我个人认为非常合理，因为对于人类基因组来说每一段的重组率都不一样。

$$\pi = (\pi_1, \pi_2, \dots, \pi_J), \mathrm{A}_m = \begin{bmatrix} a_{11} & \cdots & a_{1J} \\ \cdots & \cdots & \cdots \\ a_{J1} & \cdots & a_{JJ} \end{bmatrix}, \mathrm{B} = \begin{bmatrix} b_{10} & b_{11} \\ \cdots & \cdots \\ b_{J0} & b_{J1} \end{bmatrix}$$

其中

$$\pi_j = P(z_1 = j)$$
$$a_{ij} = P(z_{m+1} = j | z_m = i)$$
$$b_{ij} = P(x_m = j | z_m = i)$$

注意到我们的任务是找出拆分方式的最大似然估计，那么在模型中可以写作

$$X^* = arg \max_{X \in \Omega} P(X \cdot \bar{X})$$

$$\text{such that } G = X + \bar{X}$$

其中Ω表示所有拆分方式的空间，X表示其中一条单链，$\bar{X}$表示另一条补链，而它们都是从一个人的基因中拆分出来的，所以满足方程$G = X + \bar{X}$

对于这个问题，其实可以很快发现一个人的真实单链遗传信息是由很多随机的因素所影响的，而根据中心极限定律，我们认为这个随机向量的分布满足一个多维高斯分布，但由于这个分布是一个离散的分布，我们认为最大似然估计就是这个分布中概率密度的最大值。

例 2：

$$G = \{2,0,0,1,1,0,1,2\} \quad X = \{1,0,0,1,0,0,1,1\} \quad \bar{X} = \{1,0,0,0,1,0,0,1\}$$
$$G = X + \bar{X}$$

其中

$$g_i = \{0,1,2\} \ \forall g_i \in G, \quad x_i = \{0,1\} \ \forall x_i \in X$$

通过例子可以清楚的看到这个问题的具象实例，在这个玩具例子中只展示了一种拆分方式，而我们要如何确定它是否为最大似然分布。我们会在接下来的一部分展开更为详细的讨论

4. F-B 算法细节

在刚才的一部分中我们讨论了模型的基础定义。我们在这一部分中将继续讨论如何计算这个最大似然估计的办法。

首先对于一个隐马尔科夫模型我们知道计算最大观测概率问题是其中的一个经典的问题，另外两个问题是译码问题和参数学习问题。但是对于我们这个实际情形来说，和传统最大观测概率问题一个显著的区别是观测值是成对出现且满足相加等于基因信息的约束。

但是从原理上来说我们依然可以利用经典的 forward-backward algorithm 来对于这个概率进行计算，那么我们开始。

首先在传统的 F-B 算法的复杂度为$O(MN^2)$, 在我们需要解决的问题中M是基因组长度，N是训练集的人数。因此我们很自然会想去尝试压缩训练集矩阵来减小N的数量，该算法的作者将其定义为去除矩阵中信息的冗余，那么我们首先需要对矩阵进行压缩。

压缩的规则是将原训练集 H 进行分段，分段的依据是每一段中的子链种类数小于等于某个阈值J，在段间可以进行自由链接。而段的权重以及节点的权重则表示通过这些段和节点的单链的数量，下面的例子清晰的展现了如何对训练集 H 进行压缩

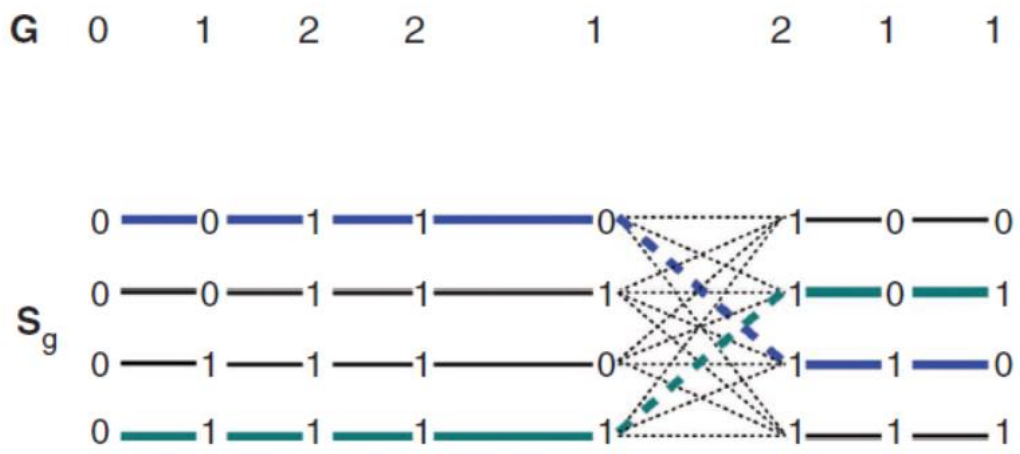
例 3：

在这个例子中可以看到矩阵压缩的过程，在拓扑结构上可以看到原训练集 H 中的任意一条单链都对应H_g中的一条路径（path）。

这样我们也就成功地在保留最大精度的条件下减小了计算复杂度。现在让我们再回想一下最开始的问题是片段式地对于一个人的基因 G 进行拆分，那么我们接下来尝试去思考如何定义这个片段

一个很自然的想法是把含有每 B 个杂合子的子序列分为一段，那么每一段则存在2^B种可能性，下面这个例子说明了如何将一个人的基因序列分成若干段的过程

例 4：



在这个例子中 B=2，因此我们将前 5 个位点划为第一段，后三个位点划为第二段
我们在之前阐述了为何要进行训练集 H 和测试集 G 的压缩与分段，也阐述了如何去这样操作。
那么接下来通过这样的方式对训练集和测试集进行处理之后，我们可以开始进一步分析 HMM 模型。
首先回到最初的参数空间

$$HMM = \lambda = (\pi, \mathrm{A}, \mathrm{B})$$

其中

$$\pi_j = P(z_1 = j | H_g, S_g) = \frac{w_j}{K}$$

$$a_{ij} = P(z_{m+1} = j | z_m = i, H_g, S_g) = (1-\rho_m)\frac{w_{ij}}{w_i}, +\rho_m \frac{w_j}{K}$$

$$b_{ij} = P(x_m = j | z_m = i, H_g, S_g) = \frac{K}{K+\theta}\mathbf{1}_{\{Hg_{m,i}=Sg_{m,j}\}} + \frac{\theta}{2(K+\theta)}$$

其中w_j, w_{ij}分别表示 Hg 中j节点和边(i,j)的权重，ρ_m表示第 m 个位点和第(m+1)个位点间的重组概率，θ表示所有基因位点突变的概率，K 表示训练集 H 中的人数

identity 函数$\mathbf{1}_{\{Hg_{m,i}=Sg_{m,j}\}}$表示如果在第 m 个位点测试集的第 j 个位点和训练集的第 i 个位点取值相同，比如同时为 0 或 1，则$\mathbf{1}_{\{Hg_{m,i}=Sg_{m,j}\}} = 1$，否则$\mathbf{1}_{\{Hg_{m,i}=Sg_{m,j}\}} = 0$
同时$\theta \approx \ln(K)$
至此，我们得到了$\lambda = (\pi, \mathrm{A}, \mathrm{B})$。一个比较有趣的结论是若$\rho_m = 0$，则$\mathrm{A}_m = \boldsymbol{I}$，因此$rank(A_m) = J$；若$\rho_m = 1$，则$\mathrm{A}_m = \begin{bmatrix} a_m \\ \dots \\ a_m \end{bmatrix}$,其中$a_m = \left[\frac{w_1}{K}, \frac{w_2}{K}, \dots, \frac{w_J}{K}\right]$，因此$rank(A_m) = 1$,
从这两个边界条件我们可以看到不同的ρ_m对于转移概率矩阵A_m有着非常巨大的影响，

同时在这个马尔科夫链中影响会被放大，在后面的分析中我们将看到这一点。可以说整体模型的估计结果对于ρ这个参数的取值敏感度非常高

到目前为止，可以看到依赖于训练集 Hg 以及测试集 Sg 我们定义了 HMM 的参数空间，依据此我们将对于最好的拆分方式进行估计。让我们再进一步利用 F-B algorithm 思考这个问题。F-B 算法本质上是一类动态规划算法，可以在多项式时间内计算出我们期待的分布，其相关变种算法有 Baum–Welch algorithm

我们将 forward 方向得到的参数称为 alpha，将 backward 方向得到的参数称为 beta，而计算这两个参数的目的是为了更深入地分析 HMM 模型里的序列分布进而得到我们所期望的拆分方式的似然函数。

4.1 Alpha 参数

首先我们定义 alpha 如下

$$\alpha_m(i,j) = P(x_{1:m-1}, x_m = i; z_{1:m-1} z_m = j | H_g, S_g)$$

同时我们定义边界条件 m=1 时的 alpha 如下：

$$\alpha_1(i,j) = P(x_1 = i, z_1 = j) = P(x_1 = i | z_1 = j) * P(z_1 = j)$$

而它的动态规划方程以及证明如下(为了方便书写我们省略 Hg 以及 Sg)：

$$\begin{aligned}
\alpha_{m+1}(i,j) &= P(x_{1:m}, x_{m+1} = i; z_{1:m} z_{m+1} = j) \\
&= \sum_{z_m} P(x_{1:m}, x_{m+1} = i; z_{1:m-1}, z_m, z_{m+1} = j) \\
&= \sum_{z_m} P(x_{1:m-1}, x_m, x_{m+1} = i; z_{1:m-1}, z_m, z_{m+1} = j) \\
&= \sum_{z_m} P(x_{m+1} = i | z_{m+1} = j, z_m, x_{1:m-1}, x_m) * P(z_{m+1} = \\
&\quad j | z_{1:m-1}, z_m, x_{1:m-1}, x_m) * P(z_m, x_{1:m-1}, x_m) \\
&= P(x_{m+1} = i | z_{m+1} = j) \sum_{z_m} P(z_{m+1} = j | z_m) * P(x_{1:m-1}, x_m; z_{1:m-1}, z_m) \\
&= P(x_{m+1} = i | z_{m+1} = j) \sum_{z_m} \alpha_m(x_m, z_m) * P(z_{m+1} = j | z_m) \quad \dots\dots\dots(*)
\end{aligned}$$

从这个证明中我们可以看到对z_m的求和有 J 项，即$z_m = 1,2,\dots,J$。同时注意到x_m并没有给定具体值，回想 alpha 的计算依赖于 Hg 和 Sg 的拓扑结构，而x_m $\forall m$都是来自于 Sg，也就是说x_m必须服从 Sg 的拓扑结构。而我们在定义 Sg 的时候曾规定在段内x_{m+1}的父节点只有一个x_m，而在段间x_{m+1}的父节点则可以是任意x_m，因此我们在计算上应该区别对待。

在计算复杂度上可以看到对于任意i,j，我们都要进行至少J次计算，并且对于段间的特殊位点我们则需要进行IJ次运算，所以可以分析 forward 算法的复杂度为$O(MIJ^2)$，这符合之前我们的先验理论$O(MN^2)$。虽然按照之前的分析，$I = 2^B$，也就是如果以 B 个杂合子作为分段标准，那么 Sg 这个矩阵中的列数为2^B，这个数字可以视作一个常数。然而，我们的目标是以线性的速度去完成这一部分计算，我们可以尝试从转移概率矩阵A_m结构性上的特点去找到方案。

首先，对于 Hg 段间位点的计算。我们可以分析A_m中的元素

$$a_{ij} = P(z_{m+1} = j | z_m = i, H_g, S_g) = (1 - \rho_m)\frac{w_{ij}}{w_i}, + \rho_m \frac{w_j}{K}$$

此时可以看到分母w_i对于每一个上游节点都拥有不同的取值，因此我们对于这一类位点并不能进行简化。也就是说 Hg 的分段位点我们必须正常计算

那么，接下来我们来分析 Hg 段内位点。可以得到

$$a_{ij} = P\left(z_{m+1} = j \middle| z_m = i, H_g, S_g\right) = \begin{cases} \rho_2 + \rho_1 * \frac{w_j}{K} & if\ i \neq j \\ \rho_1 * \frac{w_j}{K} & if\ i = j \end{cases}$$

其中$\rho_2 = 1 - \rho_m$，$\rho_1 = \rho_m$

我们将这个式子代入$(*)$，那么可以得到

$[\alpha_{m+1}(i,1), \quad \dots\dots \quad ,\alpha_{m+1}(i,J)]$

$$= Emission * [\alpha_m(i,1), \quad \dots\dots \quad ,\alpha_m(i,J)] \begin{bmatrix} \rho_2 + \rho_1 * \frac{w_0}{K} & \cdots & \rho_1 * \frac{w_J}{K} \\ \vdots & \ddots & \vdots \\ \rho_1 * \frac{w_0}{K} & \cdots & \rho_2 + \rho_1 * \frac{w_J}{K} \end{bmatrix}$$

其中$Emission = P(x_{m+1} = i | z_{m+1} = j)$，我们给出如下定义

$$\Sigma_0 = [\alpha_m(i,1), \quad \dots\dots \quad ,\alpha_m(i,J)] \begin{bmatrix} \rho_2 + \rho_1 * \frac{w_0}{K} \\ \vdots \\ \rho_1 * \frac{w_0}{K} \end{bmatrix} \quad \dots\dots (a_1)$$

那么我们有

$$\alpha_{m+1}(i,1) = Emission * \Sigma_0$$

因此，我们可以尝试用递推的方式去计算$\alpha_m(\cdot\,,\,\cdot) \quad for\ \forall m$

注意到

$$\Sigma_1 = [\alpha_m(i,1), \quad \dots\dots \quad ,\alpha_m(i,J)] \begin{bmatrix} \rho_1 * \frac{w_1}{K} \\ \vdots \\ \rho_1 * \frac{w_1}{K} \end{bmatrix}$$

那么，我们定义

$$\Delta_0 = \Sigma_1 - \Sigma_0 = \left[\sum_{j=1}^{J} \alpha_m(i,j)\right] * \rho_1 * \frac{w_1 - w_0}{K} + [\alpha_m(i,1) - \alpha_m(i,0)] * \rho_2$$

到现在为止，我们应该已经发现了这个计算的 trick，那就是只计算Σ_0，而剩下的$\Sigma_j \ \forall j$都可以通过递推的方式给出。在段内位点中，我们可以给出

$$\Delta_j = \Sigma_{j+1} - \Sigma_j = \left[\sum_{j=1}^{J} \alpha_m(i,j)\right] * \rho_1 * \frac{w_{j+1} - w_j}{K} + [\alpha_m(i,j+1) - \alpha_m(i,j)] * \rho_2 \quad \dots\dots (a_2)$$

回想之前的分析，对于任意段间位点，我们需要考虑 Sg 的拓扑结构，因此需要考虑第二个对于x_m的 Σ，那么我可以得到段间位点的Δ_j如下

$$\Delta_j = \Sigma_{j+1} - \Sigma_j = \sum_{i=1}^{2^B}\left[\sum_{j=1}^{J} \alpha_m(i,j)\right] * \rho_1 * \frac{w_{j+1} - w_j}{K} + \left[\sum_{i=1}^{2^B} \alpha_m(i,j+1) - \sum_{i=1}^{2^B} \alpha_m(i,j)\right] * \rho_2$$

$$\dots\dots (a_3)$$

至此，我们可以看到对于每一个$\alpha_m(i,j)$，除了$j=1$我们需要$O(J)$次计算，其余的任意(i,j)我们只需要常数级别的运算。那么 alpha 的计算复杂度就变成了$O(MIJ)$，我们完成了计算速度的优化

4.2 Beta 参数

首先我们定义 beta 如下

$$\beta_m(i,j)=P(x_{m+1:M};z_{m+1:M}|x_m=i;z_m=j,H_g,S_g)$$

同时我们定义边界条件 m=M 时的 beta 如下：

$$\beta_M(i,j)=1\ for\ \forall i,j$$

而它的动态规划方程以及证明如下(为了方便书写我们省略 Hg 以及 Sg)：

$$\beta_m(i,j)=P(x_{m+1:M};z_{m+1:M}|x_m=i;z_m=j)$$

$$=\sum_{z_{m+1}}P(x_{m+1:M};z_{m+1},z_{m+2:M}|x_m=i;z_m=j)$$

$$=\sum_{z_{m+1}}P(x_{m+2:M}|x_m=i,z_m=j,x_{m+1},z_{m+1})P(x_{m+1}|z_{m+1},z_m=j,x_m=i)\,P(z_{m+1}|x_m=i;z_m=j)$$

$$=\sum_{z_{m+1}}P\left(x_{m+2:M}\middle|x_{m+1;}z_{m+1}\right)*P(x_{m+1}|z_{m+1})*P(z_{m+1}|z_m=j)$$

$$=\sum_{z_{m+1}}\beta_{m+1}\left(x_{m+1,}z_{m+1}\right)*P(x_{m+1}|z_{m+1})*P(z_{m+1}|z_m=j)\quad\ldots\ldots\ldots(**)$$

从这个证明中我们可以看到对z_{m+1}的求和有 J 项，即$z_{m+1}=1,2,\ldots,J$。同时注意到x_{m+1}并没有给定具体取值，回想 beta 的计算依赖于 Hg 和 Sg 的拓扑结构，而$x_{m+1}\ \forall m$都是来自于 Sg，也就是说x_m必须服从 Sg 的拓扑结构。而我们在定义 Sg 的时候曾规定在段内x_m的子节点只有一个x_{m+1}，而在段间x_m的子节点则可以是任意x_{m+1}，因此我们在计算上应该区别对待。

在计算复杂度上可以看到对于任意i,j，我们都要进行至少J次计算，并且对于段间的特殊位点我们则需要进行IJ次运算，所以可以分析 backward 算法的复杂度为$O(MIJ^2)$，这符合之前我们的先验理论$O(MN^2)$。虽然按照之前的分析，$I=2^B$，也就是如果以 B 个杂合子作为分段标准，那么 Sg 这个矩阵中的列数为2^B，当然这个数字可以视作一个常数。然而，我们的目标是以线性的速度去完成这一部分计算，类似于之前 alpha 部分的分析，我们可以尝试从转移概率矩阵A_m结构性上的特点去找到方案。

首先，对于 Hg 段间位点的计算。我们可以分析A_m中的元素

$$a_{ij}=P\left(z_{m+1}=i\middle|z_m=j,H_g,S_g\right)=(1-\rho_m)\frac{w_{ji}}{w_j},+\rho_m\frac{w_i}{K}$$

此时可以看到分母w_j对于每一个上游节点都拥有不同的取值，因此我们对于这一类位点并不能进行简化。也就是说 Hg 的分段位点我们必须正常计算

那么，接下来我们来分析 Hg 段内位点。可以得到

$$a_{ij}=P\left(z_{m+1}=j\middle|z_m=i,H_g,S_g\right)=\begin{cases}\rho_2+\rho_1*\dfrac{w_j}{K} & if\ i\neq j\\ \rho_1*\dfrac{w_j}{K} & if\ i=j\end{cases}$$

其中$\rho_2=1-\rho_m$，$\rho_1=\rho_m$

我们将这个式子代入$(**)$，那么可以得到

$$[\beta_m(i,1),\quad \dots\dots\quad ,\beta_m(i,J)]$$

$$= [\beta_{m+1}(i,1)\cdot Emis(i,1),\quad \dots\dots\quad ,\beta_{m+1}(i,J)Emis(i,J)]\begin{bmatrix}\rho_2+\rho_1*\frac{w_0}{K} & \cdots & \rho_1*\frac{w_0}{K}\\ \vdots & \ddots & \vdots\\ \rho_1*\frac{w_J}{K} & \cdots & \rho_2+\rho_1*\frac{w_J}{K}\end{bmatrix}$$

其中$Emis(i,j)=P(x_{m+1}=i|z_{m+1}=j)$，接下来我们给出如下定义

$$\Sigma_0=[\alpha_m(i,1)Emis(i,1),\quad \dots\dots\quad ,\alpha_m(i,J)Emis(i,J)]\begin{bmatrix}\rho_2+\rho_1*\frac{w_0}{K}\\ \vdots\\ \rho_1*\frac{w_J}{K}\end{bmatrix}\dots\dots(b_1)$$

那么我们有

$$\beta_m(i,1)=\Sigma_0\quad for\ \forall i=\{1,2,\dots,2^B\}$$

因此，我们可以尝试用递推的方式去计算$\alpha_m(\cdot,\cdot)\ \ for\ \forall m$

注意到

$$\Sigma_1=[\alpha_m(i,1),\quad \dots\dots\quad ,\alpha_m(i,J)]\begin{bmatrix}\rho_1*\frac{w_0}{K}\\ \vdots\\ \rho_1*\frac{w_J}{K}\end{bmatrix}$$

那么，我们定义

$$\Delta_0=\Sigma_1-\Sigma_0=[\beta_{m+1}(i,2)Emis(i,2)-\beta_{m+1}(i,1)Emis(i,1)]*\rho_2\quad \dots\dots(b_2)$$

到现在为止，我们应该已经发现了这个计算的 trick，那就是只计算Σ_0，而剩下的$\Sigma_j\ \forall j$都可以通过递推的方式给出。在段内位点中，我们可以给出

$$\Delta_j=\Sigma_{j+1}-\Sigma_j=[\beta_{m+1}(i,j+1)Emis(i,j+1)-\beta_{m+1}(i,j)Emis(i,j)]*\rho_2\quad \dots\dots(b_3)$$

回想之前的分析，对于任意段间位点，我们需要考虑 Sg 的拓扑结构，因此需要考虑第二个对于x_m的 Σ，那么我可以得到段间位点的Δ_j如下

$$\Delta_j=\Sigma_{j+1}-\Sigma_j=\left[\sum_{i=1}^{2^B}\beta_{m+1}(i,j+1)Emis(i,j+1)-\sum_{i=1}^{2^B}\beta_{m+1}(i,j)Emis(i,j)\right]*\rho_2$$

至此，我们可以看到对于每一个$\beta_m(i,j)$，除了$j=1$我们需要$O(J)$次计算，其余的任意(i,j)我们只需要常数级别的运算。那么 beta 的计算复杂度就变成了$O(MIJ)$，我们完成了计算速度的优化

4.3 F-B 算法总结

我们依据之前的分析给出 alpha 和 beta 的线性时间计算算法如下：

- Alpha

$for\ \forall m > 1, \forall i \in \{1,2,\dots,2^B\}, compute\ \Sigma_0\ by\ equation\ (a_1)$

$for\ j = 2\ to\ J$

Do

$compute\ \Delta_j\ by\ equation\ (a_2)\ or\ (a_3)$

$compute\ \Sigma_j = \Sigma_{j-1} + \Delta_j$

$for\ j = 1\ to\ J$

Do

$compute\ \ \alpha_m(i,j) = \Sigma_j * Emission$

- Beta

$for\ \forall m < M, \forall i \in \{1,2,\dots,2^B\}, compute\beta_m(i,0) = \ \Sigma_0\ by\ equation\ (b_1)$

$for\ j = 2\ to\ J$

Do

$compute\ \Delta_j\ by\ equation\ (b_2)\ or\ (b_3)$

$compute\ \beta_m(i,j) = \Sigma_j = \Sigma_{j-1} + \Delta_j$

同时还应该注意到无论是计算 alpha 还是 beta。对于 Sg 分段处的位点，我们可以只计算$i = 1$的情形，其他i所对应的函数值和$i = 1$没有区别，具体细节如果需要可以自行推导，分析方法和之前类似，这可以进一步少量地提升计算速度。

至此，我们已经从理论到计算完整的得到了 forward-backward algorithm 的所有细节信息，我们将在接下来的部分继续深入讨论我们该如何使用已经计算出来的信息。

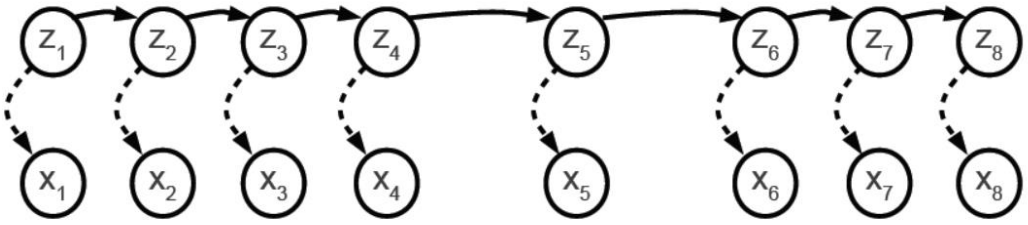
5. 观测序列的概率分布

回忆在之前的讨论中，我们强调的任务是拿到拆分方式的最大似然分布，那么我们需要一种办法去计算每一种拆分方式的分布。在我们目前使用以及定义的 HMM 中我们知道

$$X = \{x_1, x_2, \dots\dots, x_M\}$$

这样一条序列为 HMM 中的观测序列，那么我们希望能够得到任意观测序列的概率分布。分析过程如下：

首先观察 HMM 的变量关系



可以看出对于任意m，如果给定z_m，x_m只依赖于z_m；同时z_{m+1}也只依赖于z_m。和其他的变量无关，这是模型马尔可夫性质所决定的。

接着我们定义

$$\begin{aligned}\xi(i_1,j_1,i_2,j_2) &= \alpha_m(i_1,j_1)\cdot P(z_{m+1}=j_2|z_m=j_1)\cdot P(x_{m+1}=i_2|z_{m+1}=j_2)\cdot \beta_{m+1}(i_2,j_2)\\ &= P(x_{1:m-1}, x_m=i_1; z_{1:m-1}, z_m=j_1)P(z_{m+1}=j_2|z_m=j_1)\\ &\quad P(x_{m+1}=i_2|z_{m+1}=j_2)P(x_{m+2:M}; z_{m+2:M}|x_{m+1}=i_2; z_{m+1}=j_2)\end{aligned}$$

$$= P(x_{1:m-1}, x_m = i_1, x_{m+1} = i_2; z_{1:m-1}, z_m = j_1, z_{m+1} = j_2)$$
$$P(x_{m+2:M}; z_{m+2:M} | x_{m+1} = i_2; z_{m+1} = j_2)$$

因此我们得到

$$\xi(i_1, j_1, i_2, j_2) = P(x_{1:m-1}, x_m = i_1, x_{m+1} = i_2, x_{m+2:M}; z_{1:m-1}, z_m = j_1, z_{m+1} = j_2, z_{m+2:M})$$

接下来我们有

$$P(x_m = i_1, x_{m+1} = i_2) \propto \frac{\sum_{j_1=1}^{J}\sum_{j_2=1}^{J}\xi(i_1, j_1, i_2, j_2)}{\sum_{i_1=1}^{2^B}\sum_{i_2=1}^{2^B}\sum_{j_1=1}^{J}\sum_{j_2=1}^{J}\xi(i_1, j_1, i_2, j_2)} \dots\dots (*)$$

同时我们也可以得到

$$P(x_m = i_1) \propto \frac{\sum_{i2=1}^{2^B}\sum_{j_1=1}^{J}\sum_{j_2=1}^{J}\xi(i_1, j_1, i_2, j_2)}{\sum_{i1=1}^{2^B}\sum_{i2=1}^{2^B}\sum_{j1=1}^{J}\sum_{j2=1}^{J}\xi(i_1, j_1, i_2, j_2)} \dots\dots (**)$$

因此，我们有

$$P(x_{m+1} = i_2 | x_m = i_1) = \frac{\sum_{j_1=1}^{J}\sum_{j_2=1}^{J}\xi(i_1, j_1, i_2, j_2)}{\sum_{i2=1}^{2^B}\sum_{j_1=1}^{J}\sum_{j_2=1}^{J}\xi(i_1, j_1, i_2, j_2)} \dots\dots (***)$$

至此，我们可以依据$(*)$，$(**)$和$(***)$计算出我们所需要得一切碎片信息。那么接下来我们将所有碎片组装起来可以得到

$$P(X) = P(x_1, x_2, \dots\dots, x_M) = P(x_1)P(x_2|x_1) \dots . P(x_M|x_{M-1})$$

回忆我们之前的论述，任意x_m是 Sg 中的抽样，那么它们的转移性质就必须服从 Sg 的拓扑结构，因此对于 Sg 段内的位点，我们有

$$P(x_{m+1}|x_m) = \begin{cases} 1 \ if \ x_{m+1} \ and \ x_m \ belong \ to \ the \ same \ path \\ 0 \quad otherwise \end{cases}$$

因此，我们只需要考虑段间连接处的计算。也就是说拆分的过程是一段接一段的方式来进行的，这也大幅度地减小了我们的计算时间复杂度。

从刚才的分析中我们可以看到，对于任意一个 Sg 分段处任意的上下游组合$x_m = i_1, x_{m+1} = i_2$，我们需要进行J^2次计算，在接下来的部分我们讨论如何将这个时间进行压缩

线性时间计算：

相似地，我们可以使用之前 alpha 和 beta 部分类似的技巧来进行分析。

假设$j_1 = 1$，我们有

$$\Sigma_1 = [Emis(i_2, 1)\beta_{m+1}(i_2, 1), \dots, Emis(i_2, J)\beta_{m+1}(i_2, J)] \begin{bmatrix} \rho_2 + \rho_1 \frac{w_1}{K} \\ \dots \\ \rho_1 \frac{w_1}{K} \end{bmatrix}$$

假设$j_1 = 2$，我们有

$$\Sigma_2 = [Emis(i_2, 1)\beta_{m+1}(i_2, 1), \dots, Emis(i_2, J)\beta_{m+1}(i_2, J)] \begin{bmatrix} \rho_1 \frac{w_1}{K} \\ \dots \\ \rho_1 \frac{w_1}{K} \end{bmatrix}$$

因此，我们得到线性时间计算Σ_j的方法

$$\Delta_1 = \Sigma_2 - \Sigma_1 = [Emis(i_2,1)\beta_{m+1}(i_2,1) - Emis(i_2,0)\beta_{m+1}(i_2,0)] * \rho_2$$

类似的，我们有

$$\Delta_j = \Sigma_{j+1} - \Sigma_j = [Emis(i_2,j+1)\beta_{m+1}(i_2,j+1) - Emis(i_2,j)\beta_{m+1}(i_2,j)] * \rho_2 \dots\dots(1)$$

同时注意到方程$(*)$中的分子

$$Numerator = \sum_{j=1}^{J} f(j) \dots\dots(2)$$

其中

$$f(j) = \alpha_m(i_1,j)\Sigma_j \dots\dots(3)$$

并且我们观察$(*)$还可以发现

$$Dominator = \sum_{i_1=1}^{2^B}\sum_{i_2=1}^{2^B} Numerator(i_1,i_2) \dots\dots(4)$$

所以可以做出推论，如果我们能实现对于Σ_j的常数时间计算，那么也就实现了$f(j)$的常数时间计算，那么也就实现了$Numerator$的线性时间计算

依据于此，我们可以得到线性时间计算$P(x_m = i_1, x_{m+1} = i_2)$的算法

$Compute\ \Sigma_1$
$for\ \forall m \in S_g\ cut$
    $for\ j = 2\ to\ J$
        $compute\ \Delta_j\ by\ equation\ (1)$
        $compute\ \Sigma_j = \Sigma_{j-1} + \Delta_j$
    $for\ j = 1\ to\ J$
        $compute\ f(j)\ by\ equation\ (3)$
    $compute\ Numerator\ by\ equation\ (2)$
    $for\ i_1 = 2\ to\ 2^B$
        $for\ i_2 = 1\ to\ 2^B$
            $compute\ Numerator(i_1,i_2)$
    $compute\ Dominator\ by\ equation\ (4)$

至此，我们已经利用 alpha 和 beta 完成了 xi 的计算，并且通过 xi 完成了所有碎片条件概率的计算。那么从理论上我们已经可以计算出任意一条 path 的概率，也就是

$$P(X)\ for\ \forall X \in \Omega$$

但是回想我们的目标是找出最大的一条概率所对应的最优路径，接下来的部分我们将对此进行讨论。

6. 维特比算法

概率空间坍缩

回想我们的目标是找出最优的拆分方案估计

$$X^* = arg\max_{X\in\Omega} P(X\cdot\bar{X})$$

那么我们首先要对于概率空间进行坍缩。即把单链空间坍缩为对链空间。这个变换的过程实际上非常简单，依据作者的描述，我们将X和$\bar{X}$视为独立从我们已经拥有的 HMM 里计算找出的，那么我们有

$$P(X\cdot\bar{X}) = P(X)\cdot P(\bar{X})$$

因此对于任意 m，任意 i，我们只需要把它的补链找出来同时把它们的概率分布直接相乘就完成了对这个概率空间的坍缩步骤。

最大似然分布

在上一部分的分析中，我们得到

$$P(X) = P(x_1, x_2, \dots\dots, x_M) = P(x_1)P(x_2|x_1)\dots.P(x_M|x_{M-1})$$

因此我们的任务是在 Sg 上找出一条最大概率路径，对于这个典型的最短路图论问题我们可以使用动态规划的算法进行求解。同时观察到在这个问题中邻接矩阵是一个非常稀疏的矩阵并且是一个分块矩阵，除了对角块元素不为 0，其余区块元素均为 0。那么我们如果用传统的 Dijkstra 算法或者 Bellman-Ford 算法来进行求解效率肯定是很低的，具体原因是比如 Dijkstra 贪心算法的算法时间复杂度为$O(NlogN)$, N为邻接矩阵的节点数，在这个问题中$N = \left(\#\ of S_g\ Cut\right) * 2^B$，这并不是一个很小的数字，因此计算时间会变得难以接受。而 Bellman-Ford 作为动态规划算法，其算法复杂度要比贪心算法更大一个数量级，因此我们不需要继续进行分析。

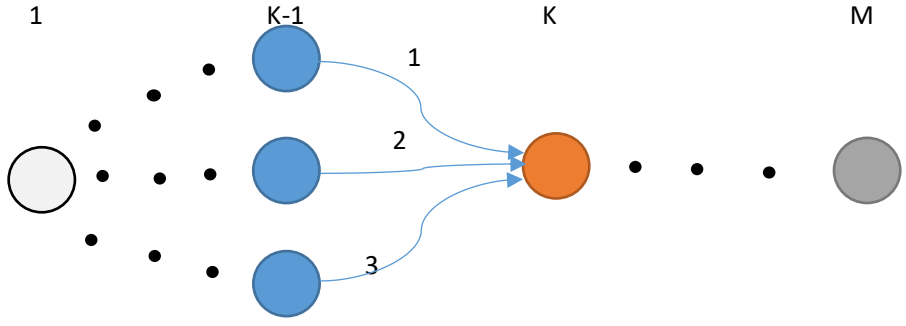
幸运的是，我们可以用 Viterbi 算法来对这个问题进行求解，其原理是一个动态规划算法，但是其不需要将所有节点都考虑并生成一个巨大的邻接矩阵。而是利用马尔可夫性质生成若干 (具体数量为$\#\ of S_g\ Cut$) 个分块的小矩阵，从而大大降低了算法的时间复杂度。

算法细节

Viterbi 算法的基本逻辑与 Viterbi decoding 大同小异，即一种典型的动态规划算法，利用最优子结构的性质来对一个复杂问题进行拆分。具体分析如下

我们首先将这个问题抽象，理解成一个从第一段开始一直到最后一段，经过若干条边，每一条边对应一个权重。那么概率成绩最大可以通过取$-ln$转换为权重加和最小，也即为求最短路径，也即为求所有边的最短权重和。

同时我们知道，在非循环无负权重图中，经过某中间节点到达终点的最短路必定包含从起始点到达中间节点的最短路。



在上图中白点表示第一段起始点，灰点表示第 M 段终点，蓝点表示第(K-1)段上游点，橙色点表示第 K 段当前节点。黑色省略号表示某条路径。
假设橙色节点的上游只有三个节点，那么我们可以清晰地观察到从橙色节点到终点的最短路径必定包含从起始点到橙色节点的最短路径，同时假设我们已经找到了白色节点到三个蓝色节点的最短路径，并记录权重为

$$(a, b, c)$$

那么从白色节点到橙色节点的最短路径权重为

$$min\ (a + w_1, b + w_2, c + w_3)$$

而其对应的最短路径为

$$argmin\ (a + w_1, b + w_2, c + w_3)$$

这是第 K 段的情形，如果我们让 K 从 1 到 M，应用相同的计算，那么我们就可以找出从第 0 段节点一直到第 M 段节点的最短路径。
同时注意到在我们这个问题中，每一段实际上含有2^B个节点，因此当我们计算到最后一段时，我们需要在2^B个候选人中找寻最小值，然后进行回溯找到完整的最优轨迹。
事实上这一段论述并不是一个严谨的数学论述，但是我觉得离散数学里面那些符号打起来非常累，所以没有采用那种方式。不过通过这个简单的例子和分析应该已经了解了 Viterbi 算法的核心
回到正题，我们至此已经通过一系列复杂的计算分析完成了对于最大似然估计的计算，我们已经找到了目标拆分方式。因此我们可以将这个拆分方式作为最终的估计结论。

7 MCMC 和吉布斯抽样

但是，根据这个作者的论述，他似乎是认为之前的所有分析中所利用的这个 HMM 由于我们参考库的大小以及样本多样性等限制原因，我们无法拿到一个非常完美的模型，因此对于任意m, i_1, i_2，这个方程

$$P(x_m = i_1, x_{m+1} = i_2) \propto \frac{\sum_{j_1=1}^{J}\sum_{j_2=1}^{J}\xi(i_1, j_1, i_2, j_2)}{\sum_{i_1=1}^{2^B}\sum_{i_2=1}^{2^B}\sum_{j_1=1}^{J}\sum_{j_2=1}^{J}\xi(i_1, j_1, i_2, j_2)}$$

对于真值的估计是有偏的，同时注意到$\xi(i_1, j_1, i_2, j_2)$实际上是 alpha 和 beta 的函数，而 alpha 和 beta 实际上是 (π, A, B)的函数，因此我们可以理解他的意思是认为模型存在误差。他所给出的解决方案是利用马尔可夫蒙特卡洛算法中的 Gibbs Sampling 来解决

吉布斯抽样

从原理上来说吉布斯抽样是一种纯数值模拟的方法，从头到尾不进行任何的计算，唯一要做的事情是不断的扔筛子和计数。而这个方法的目的是在解决一个复杂问题的时候，我们需要对定义在复空间上的多个随机变量或者向量进行估计，此时我们并没有办法获取全部随机变量的联合概率的信息，因此我们做出的估计毫无疑问是不精确的，但是我们可以获取它们的条件概率分布，依据马尔可夫性质进行抽样的话，抽样结果会最终收敛于它们的联合概率密度。
其中的数学原理相当复杂，分析方法是马尔可夫链中的各态经历理论(Ergotic Theory)，数学分析的形式在于马尔可夫链的 transition probability matrix 可以通过若尔当标准型分解为

$$A = P\Lambda P^{-1}$$

其中

P为A的特征向量所构成的空间，而Λ为一个 Jordan 矩阵，其对角线上的元素为A的所有特征根。对于马尔可夫链的转移矩阵A，如果其所有特征根的 magnitude 均小于等于 1，即

$$|\lambda_i| = |r_i + jc_i| \leq 1 \; for \; i \in \{1,2,\dots,n\}$$

同时如果存在唯一的一个特征根的 magnitude 等于 1

那么根据控制理论，我们可以认为这个系统是稳定的，即

$$\lim_{n\to\infty} A^n = \lim_{n\to\infty} (P\Lambda P^{-1})^n = \lim_{n\to\infty} P\Psi P^{-1}$$

其中

$$\Psi = \lim_{n\to\infty} \Lambda^n$$

根据矩阵地性质，我们得到Ψ这个矩阵只有一行不全为 0，其余皆为 0。

那么我们可以看到一个马尔可夫链的稳态分布在这种情形下一定存在。

但是要注意到的一点是，如果存在多个特征根的 magnitude 等于 1，那么这个马尔可夫链不存在稳态分布即有周期性会不停地摆动。

但是在我们这个问题中我们可以认为满足之前地假设，也就是说通过不断地抽样，我们实际上在模拟这个马尔可夫链。那么最终我们可以得到真实的稳态联合概率分布

具体算法为

- Initialize, $x_0(1), \dots, x_0(l)$, arbitrarily.
- **For** $n = 1, 2, \dots, N$, **Do**
  - **For** $d = 1, 2, \dots, l$, **Do**
    - Draw

$$x_n(d) \sim p\left(x \mid \{x_n(i), i < d \neq 1\}, \{x_{n-1}(i),\ i > d \neq l\}\right)$$

  - **End For**
- **End For**

本质上为通过迭代的形式不断进行抽样，最后统计抽样结果得到联合概率分布的无偏估计估计。

问题

我们很容易可以想到，吉布斯采样的缺点在于在很多实际问题中我们难以获取所有随机变量的条件概率分布，同时其收敛速度并不能有效的保证，因此我们需要大量的迭代次数来保证算法的精度。

但是在作者提出的算法中，他并不是通过统计的方法来得到最终的联合概率分布，因此我认为他的这个事实上是伪 MCMC。他的算法核心在于每次迭代通过抽样获得一对拆分的对链，随后把这个对链与 H 取并集随后重新对 H 进行压缩得到 Hg，那么也就是说每一次的 HMM 中的参数空间

$$\lambda = (\pi, A, B)$$

都会有所改变，随后将每一次依据参数空间计算出的段间碎片分布

$$P(x_m = i_1, x_{m+1} = i_2, \overline{x_m} = \overline{i_1}, \overline{x_{m+1}} = \overline{i_2})$$

进行统计，当迭代结束后用均值作为估计，即

$$\hat{P} = \frac{1}{N}\sum_{n=1}^{N} P(x_m = i_1, x_{m+1} = i_2, \overline{x_m} = \overline{i_1}, \overline{x_{m+1}} = \overline{i_2})$$

但是，这并不是吉布斯抽样，我怀疑这是为了发论文而所引入的噱头
回到主题，至此，我们已经全部讨论完了这个基因 phasing 算法

8 总结

这个基因 phasing 算法的一次迭代步骤如下。

- 进行训练集 H 的矩阵压缩，将其变换为 Hg。将测试集 G 按每 B 个杂合子为一段的方式进行拆分，将其变换为 Sg
- 通过 Hg 定义隐马尔科夫模型的参数空间(π, A, B)
- 依据 Hg 和 Sg 的拓扑结构和(π, A, B)以及 Forward-Backward 算法计算 alpha 和 beta
- 依据 alpha 和 beta 计算 xi
- 依据 xi 计算所有的单链空间条件概率分布
- 将单链空间条件概率分布按 Sg 中的配对相乘形成双链空间条件概率分布
- 依据双链空间条件概率分布进行抽样，得到一对结果
- 将抽样结果加入训练集 H 并重新进行压缩

在多次迭代结束之后我们最后还需要做

- 将多次计算的双链空间条件概率分布进行统计，用均值来对自身进行估计
- 将估计结果转换为权重
- 将权重输入维特比算法解析出最终的最大似然估计

对于不同的测试个体，黑色部分是大家公用的计算部分，而蓝色部分对于每一个人都需要单独计算。在整个算法中，alpha 和 beta 是计算量最大的一个部分，大约占到了全部算法 90%左右的计算量，按照之前的论述我们可以对其进行线性计算，时间复杂度为$O(MIJ)$，其中M为基因位点总数，$I = 2^B$为每一段种类数，J为 Hg 的压缩比例。因此这应该就是这个算法的速度最优解了

至此，我们完成了全部的分析计算！

9 缺陷和改进空间

超参数向量ρ

针对于这个算法，事实上如前文所论述，模型本身对于超参数向量ρ是非常敏感的，由于转移概率矩阵A取决于ρ，那么ρ的微小变化会导致最终整条链的计算结果发生不小的变化。而对于ρ的计算，目前使用的是方程为

$$\rho_m = 1 - \exp\left(\frac{-4N_e r_m}{K}\right)$$

其中，r_m是变量，目前r_m的外界输入中有很多位点 m 缺失了真实数据，因此我们使用的是线性插值的方法进行近似，因此这个值可能是不准的，期待未来数据库的补充

其次对于这个方法论我不清楚原理，如果原理没有问题那么自然很好。但是如果原理存在问题那么对于结果是毁灭性的。
目前我们的算法，还有作者的算法只能实现片段式的正确，并不能整条链全部 phase 正确。所谓片段式的正确是指一个片段内完全相同，下一个片段则有可能估计到它的影子链上，也就是说发生了完全的翻转。这其实根据维特比算法不难分析出来，但是根本原因我认为是在于超参数r_m存在一定的误差。

训练集 H

同时我们注意到关键步骤 alpha 和 beta 的计算也非常依赖于训练集 H，那么我们目前的训练集所达到的精度我认为是不够的。目前的程序计算精度可以这么近似估计，503 段 phase 正确或镜像 phase 正确的结果有 452 段，那么错误了 51 段，平均错误一段我们按 B=3 来计算则期望为$\frac{B}{2}=1.5$个，那么正确率为

$$rate=\frac{1.5*51}{503*3+2}=95\%$$

如果我们可以提高训练集 H 的精度，那么我们不仅可以改善上一部分提到的翻转镜像问题，同时还可以提高正确率。
那么提高训练集 H 的精度的有效办法是增加训练集人数，在作者论文中他提到使用了 15K 以上人数的训练集，我认为我们这一块应该尝试去改进

10 致谢

感谢超钧哥和心意姐在项目过程中提供的帮助以及理解支持，两个月前怀着忐忑的心情开始研究这个问题，在项目最初遇到了很多生物问题，在项目中段遇到了很多代码问题，在项目的后段遇到了很多综合的问题。感谢二位在整个过程中提供的各种建议，帮助以及理解。
最终能做出来也真可谓屡败屡战了，也感谢自己一直没有放弃。
翻过这座山，他们才会听到你的故事，不论这故事结局会如何。