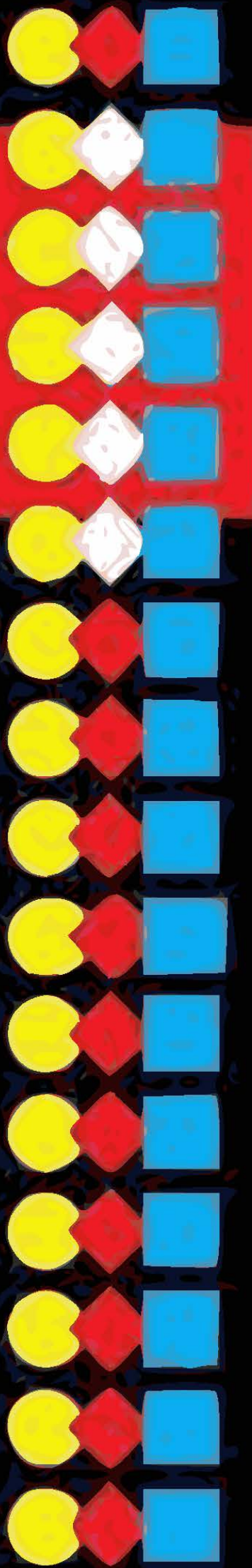
Biological sequence analysis

Probabilistic models of proteins and nucleic acids

R. Durbin
S. Eddy
A. Krogh
G. Mitchison



前言

1992 年在 Snowbird 举行的一个神经网络会议上, David Haussler 和他加州大学圣克鲁斯分校 (UCSC) 的同事们 (其中也包括本书作者之一 Anders Krogh, AK) 描述了使用概率论模型对蛋白质序列进行多序列联配建模的初步结果, 他们称这种模型为隐马模型 (HMM). 随后他们技术报告的复本被广泛地传播开来, 其中一些流传到剑桥大学的 MRC 分子生物学实验室. 在那里 Richard Durbin (RD) 和 Graeme Mitchison (GJM) 刚刚将自己的研究兴趣从神经建模转移到计算基因组序列分析上来. Sean R. Eddy (SRE) 当时是该实验室的一名实验分子遗传学背景的博士后学生, 他对计算分析感兴趣. 不久以后 AK 也到剑桥大学工作了一年.

我们都快速地接受了概率论建模的思想, 相信 HMM 及其随机文法对应物是优美的数学对象, 而且十分适于提取埋藏在生物序列中的信息. 圣克鲁斯小组和剑桥小组分别独立地开发了免费的 HMM 序列分析软件包, 并分别独立地将 HMM 方法推广到用于进行 RNA 二级结构分析的随机上下文无关文法上. 几乎与此同时, 在喷气推进实验室/加州工学院 (JPL/Caltech), 由 Pierre Baldi 领导的另一个小组也受 Snowbird 会议的成果所启发, 进行基于 HMM 方法的研究.

到 1995 年底, 我们自认为在概率论建模技术方面已经掌握了比较多的经验, 但另一方面也感到学界内此方面工作的交流相对缺乏. 虽然当时 HMM 已经引起广泛关注, 但它仍被很多人视为数学黑匣而非序列联配问题的自然模型. 许多详细描述 HMM 思想与方法的优秀文献仅限于语音识别领域, 这使得大量计算生物学家望而却步. 更重要的是, 我们和其他几个小组都越发清晰的意识到 HMM 方法可以应用于更多问题, 例如蛋白质结构建、基因识别和系统发育分析等多个领域. 1995 年圣诞节放假期间, 可能有点被雄心壮志、天真幼稚和假期闲暇所“怂恿”, 我们决定写一本关于生物序列分析的书, 主要强调概率论建模在其中的应用. 在过去的两年里, 当初那个宏伟的写作计划已经逐渐被精炼成我们对于一本实用书籍的期待.

因为作者固执己见, 所以本书带有很强的主观性, 并非序列分析的实用指南. 我们主要想对序列分析的基础进行通俗易懂地介绍, 并向读者展示为什么我们觉得概率论建模方法是有用的. 我们将尽量避免对特定计算机程序的讨论, 与此相对, 本书把更多的笔墨花费在程序背后的算法与原理上.

我们仔细地引用了大量学者的工作, 我们的见解深受其影响. 但可以确信其中必有遗漏, 这些文献我们本该阅读, 我们对此致以歉意. 同时, 本书内容必须涉及从进化生物学到概率论、再到生物物理学的诸多研究领域, 受时间、精力以及作者理解能力的限制, 我们处理一些问题时难免肤浅.

计算生物学是一门交叉学科. 包括我们在内, 研究者来自不同的研究背景, 例如分子生物学、数学、计算机科学以及物理学. 本书设想的读者是背景为这些学科之一的研究生或高年级本科生. 我们力求简洁而直观地呈现本书内容, 尽量避免令人望而生畏的数学推导或过于技术性的生物学细节.

我们假设读者已经熟悉分子遗传学的基本原理, 例如 DNA 到 RNA 到蛋白质的中心法则, 再例如核酸是由四种核苷酸子单元组成的序列, 蛋白质是由二十种氨基酸组成的序列. 必要时本书会给出分子遗传学的更多细节. 我们也假设读者具备一定的数学基础. 然而本书的部分章节仍包含一些数学性很强的描述. 我们已经尝试将这些内容安排在每章的最后, 并作为整体安排在全书的末尾. 特别是最后一章, 第 11 章涵盖了概率论中与本书前面许多内容相关的主题.

我们向爽快地接受邀请并对本书部分手稿进行校对的人们表达自己的感激之情. 我们特别感谢 Ewan Birney、Bill Bruno、David MacKay、Cathy Eddy、Jotun

Hein 和 Søren Riis. Bret Larget 和 Robert Mau 提供了他们一直在系统发育研究中使用的抽样方法的信息, 这些信息十分有帮助. David Haussler 勇敢地将本书尚未成熟的手稿作为 1996 年秋季的课堂讲义在 UCSC 讲授, 同时我们也十分感谢 David 以及他的整个班级提供了宝贵的反馈信息. 我们也要感谢 David 从一开始就将我们引入了这个研究领域. 在刊印本书的过程中, 我们与剑桥大学出版社的 David Tranah 及 Maria Murphy 还有 SG 出版公司的 Sue Glover 合作得十分愉快. 虽然本书充满方程、算法和伪代码, 但在编辑本书以及为本书进行 LaTeX 排版方面他们表现了非凡的业务能力; 他们对于我们异常乐观并毫无精准可言的时间表也展示出非凡的容忍. 书中难免存在错误, 但若没有以上所有人的努力, 相信错误的数量会更多.

我们也希望感谢为本书及相关工作提供资助的个人与集体, 他们包括维康基金会 (Wellcome Trust), 美国国家健康总署国家人类基因组研究所 (NIH National Human Genome Research Institute), 北大西洋公约组织 (NATO), Eli Lilly 公司 (Eli Lilly & Co.), 人类前沿科学项目组织 (Human Frontiers Science Program Organisation) 以及丹麦国家研究基金 (Danish National Research Foundation). 我们也感谢作者所在的研究所: RD 所在的 Sanger 研究中心 (Sanger Centre), SRE 所在的华盛顿大学医学院 (Washington University School of Medicine), AK 所在的生物序列分析中心 (Center for Biological Sequence Analysis), 还有 GJM 所在的 MRC 分子生物学实验室 (MRC Laboratory of Molecular Biology). Jim Durbin 和 Anne Durbin 于 1997 年 2 月将他们在伦敦的房子慷慨地借给我们, 本书最终草稿的合并与审定就是在此一气呵成. 我们感谢来自朋友, 家庭以及各个研究小组对写作过程的耐心等待以及对不远万里来到英格兰的 SRE与 AK 所带来不便的容忍. 我们保证不再承担新的如此宏伟的工程, 至少最近不打算这么做.

目录

第一章

绪论

当巴比伦人绘制星空图时, 天文学诞生了. 虽然我们的子孙一定不会认同生物学始于当前的基因组计划, 但是也许他们会清楚地认识到: 生物学知识的快速积累正是始于我们这个时代. 如何发掘这些知识的科学意义是一件极具挑战性的工作, 需要进一步理解细胞和有机体的生物学. 但其中的部分挑战只在于对极其丰富的生物学序列数据进行组织、分类和解析. 这不仅仅是一件抽象的字符串分析工作, 因为隐藏在碱基或氨基酸序列背后的是整个分子生物学的复杂性. 本书介绍的一些方法将不同的生物信息来源整合到一般的、清晰且可操作的序列分析概率论模型中, 原则上有能力把握部分上述的复杂性.

虽然这本书是关于计算生物学的, 但是我们有必要从一开始就明白: 要确定一个生物大分子的结构或功能, 最可靠的方法还是直接进行生物学实验. 可是在实际中, 得到一条与 RNA 或蛋白质相对应基因的 DNA 序列, 远比通过实验确定它的结构或功能要容易许多. 这种现状强烈地促使我们去发展一些计算学方法, 使得人们能仅从序列出发就可以推断其生物学的信息. 伴随着各种基因组计划的到来, 计算学方法变得尤其重要. 据估计, 仅人类基因组计划 就能产生 70000 到 100000 条人类基因的原始序列[†] , 而其中已经通过实验研究的仅占一小部分.

从本质上讲, 大多数计算序列分析问题都是统计学问题. 随机进化力作用于基因组, 并对其产生影响. 序列在随机突变、自然选择 和遗传漂变的混沌之中长期分化, 因此就如何辨识序列间的显著相似性提出了一个严重的信号噪声问题. 许多强大的现有分析方法都使用概率论. 本书强调概率论模型, 特别是隐马模型 的使用, 它能为各种序列分析问题的统计分析提供一个普适的结构.

1.1 序列的相似性、同源性及联配

大自然是一个修补匠, 不是一个发明家 Jacob [1977]. 新序列都是由已存在的序列变化而来, 而不是凭空产生. 这对基于计算的序列分析而言是一件很幸运的事. 通常可以识别新序列和已知序列间的显著相似性; 这时我们能把已知序列的结构和功能的信息传递到新序列上. 我们称这两条相关序列是*同源的* (homologous), 并且我们凭借*同源性* (homology) 传递信息.

粗略看来, 断定两条序列相似无异于断定两个文字串相似. 因此有一类生物序列分析的方法植根于计算机科学, 该领域有关于字符串比较方法的浩瀚文献. *联配* (alignment) 这一概念至关重要. 序列在进化中累积插入、删除以及替换, 因而在评价两条序列的相似性之前, 一般先要找出它们之间的某种可能的联配.

几乎所有的联配方法都是在一套计分系统下寻找两条字符串间的最优联配. 这些计分方式可以如此的简单, 即令“匹配为 $+1$, 失配为 -1”. 事实上, 早期的联配算法就是用这种形式描述的. 然而, 因为希望计分方案能给予生物学上最有可能的

[†]最新研究成果认为, 人类基因组中编码蛋白质的基因数目约为 20500 个, 见 Clamp, M., Fry, B., Kamal, M., Xie, X., Cuff, J., Lin, M. F., Kellis, M., Lindblad-Toh, K. and Lander, E. S. 2007. Distinguishing protein-coding and noncoding genes in the human genome. *Proceedings of the National Academy of Sciences of the USA* 104:19428–19433. ——译注

联配以最高分值, 所以我们需要考虑如下事实, 即生物分子具有进化历史、具有三维折叠结构以及限制其一级序列进化的其他特征. 因此, 除了联配机制与比较算法以外, 计分系统本身也需要仔细考虑, 而且可以很复杂.

发展更灵敏的计分系统以及评估联配分数的显著性, 与其说属于计算机科学, 不如归为统计学的范畴. 早先的一个进步便是为氨基酸二序列联配计分引入了一组概率矩阵 [Dayhoff, Eck & Park 1972; Dayhoff, Schwartz & Orcutt 1978], 用于量化某些替换的进化偏好. 更精密的概率论建模方法已经通过许多渠道逐渐引入计算生物学. 概率论建模方法为处理计算序列分析中的复杂推断问题提供了自然框架, 极大地扩展了能被有用且可靠理论所支持的应用的范围.

1.2 本书概述

本书在结构上大致可以分为四个部分, 每部分所覆盖的问题分别是: 二序列联配, 多序列联配, 系统发育树和 RNA 结构. 图 1.1 用状态机 (state machine) 的形式给出了作者推荐的各章阅读路线. 状态机是全书都会用到的一类模型.

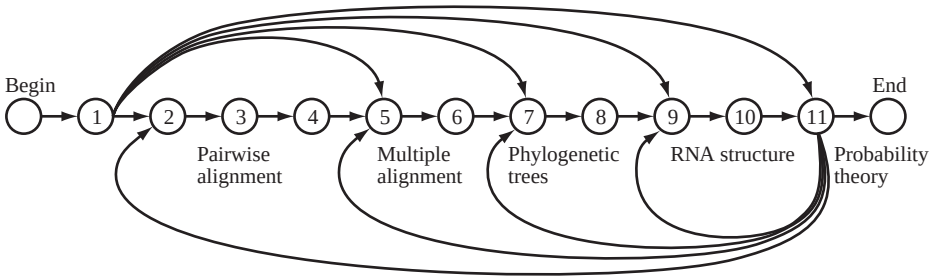


图 1.1: 本书概貌及推荐阅读路线.

各章内容如下:

2 **二序列联配**. 我们以判定一对序列是否在进化上相关的问题开篇. 然后研究传统的二序列联配与比较的算法, 这些算法主要使用动态规划寻找最优的含空位联配. 最后, 我们给出计分参数的概率论分析, 并对序列匹配的统计显著性进行讨论.

3 Markov **链与隐马模型**. 我们介绍隐马模型 (HMMs), 并说明在此模型下如何对一条序列或序列家族进行建模. 本章使用简单的例子, 给出所有的基本 HMM 算法与理论.

4 **使用** HMM **的二序列联配**. 继上一章的 HMM 理论后, 我们再次回到二序列联配问题. 我们发展了一类特别的 HMM 为已联配的序列对建模. 我们展示基于 HMM 的方法如何提供精细的方式以便对联配的精确度作出估计并对相似性给出不局限于特定联配的计分.

5 **用于序列家族的列型** HMM. 我们考虑已知进化家族或超家族的同源序列的寻找问题. 解决这个问题的一种标准方法是, 使用一个从多序列联配导出的含位置特异计分参数的“列型”. 我们描述 HMM 的一种标准形式, 称为列型 HMM, 以便对基于多序列联配的蛋白质和 DNA 序列家族建模. 我们特别关注对新家族成员最优搜索的参数估计, 其中包括对序列加权方案的讨论.

6 **多序列联配方法**. 一个密切相关的问题是如何构建一个序列家族的多序列联配. 在介绍基于列型 HMM 的多序列联配算法之前, 我们从概率论建模的观点出发研究现存的多序列联配算法.

7 **构造系统发育树**. 生物学中许多最有意思的问题都涉及到系统发育. 基因和物种是在什么时候、以何种方式进化的? 我们将概述推断进化树的一些常用方法, 包括聚类法、距离法和简约法. 在该章结尾, 我们讲解 Hein 的用于同时联配序列家族和推断其系统发育树的简约算法.

8 **系统发育的概率论方法**. 本章介绍概率论建模在系统发育中的应用, 包括树分值的最大似然估计法以及在树空间上从后验概率分布中抽样的方法. 我们也给出前一章所描述方法的概率论解释.

9 **转换文法**. 我们给出隐马模型为什么恰好是转换文法 Chomsky 层次结构的最低层次. 我们讨论如何使用更复杂的转换文法作为生物序列的概率论模型. 本章随后介绍随机上下文无关文法, 即 Chomsky 层次结构中的下一个层次.

10 RNA **结构分析**. 我们使用随机上下文无关文法理论处理 RNA 二级结构分析问题, 这类问题不能被 HMM 或其他基于一级序列的方法所解决. 这些问题包括 RNA 二级结构预测、基于结构的 RNA 联配以及基于结构的同源 RNA 数据库搜索.

11 **概率论背景**. 最后, 我们对书中其他部分以非正式指南风格提到的数学与统计学工具给出更正式的讲解.

1.3 概率与概率论模型

概率论的某些基本结论对理解本书绝大部分内容十分必要, 因此在接触序列之前, 我们将对概率论的关键思想和方法给出简要的介绍. 尽管许多读者可能对此领域十分熟悉, 但我们仍建议读者至少快速浏览一遍本节内容, 以便领会其后章节中将要出现的记号和部分思想. 除了这里的基本介绍以外, 我们在正文部分尽可能减少对抽象概率理论的讨论, 并把有关的数学推导和数学方法编入第 11 章. 在那里, 我们对相关理论作更详尽的介绍.

那么, 概率论模型 (probabilistic model) 是什么? 我们提到的模型 (model) 通常是指模拟研究对象的系统. 概率论模型是能够依照不同概率产生不同结果的模型. 因此它能模拟一整类研究对象, 并为每个对象分配相应的概率. 在本书中, 研究对象通常是序列, 而我们所建立的模型能够描述相关序列的家族.

我们先考虑一个简单例子. 一个为读者所熟悉的带有一组离散结果的概率论系统是六面骰子投掷实验. 投掷 (有可能作弊的 (loaded)) 骰子的模型有六个参数 $p_1, \ldots, p_6$; 其中掷得 i 的概率为 p_i. 作为概率, 参数 p_i 必须满足条件: $p_i \geq 0$ 并且 $\sum_{i=1}^{6} p_i = 1$. 在连续掷三次骰子的模型中, 假设每次投掷事件都相互独立, 那么掷得序列 $[1, 6, 3]$ 的概率就是单独概率的乘积 $p_1 p_6 p_3$. 我们会在本书的前面部分一直使用掷骰子模型, 以便为概率论建模给出简单而直观的例子.

再考虑一个更接近于生物学的例子, 这是关于任意蛋白质或 DNA 序列的极为简单的模型. 生物序列是字符串, 它们来自于一个有限的残基*字母表* (alphabet) , 该字母表的元素通常是 4 种核苷酸或 20 种氨基酸. 假设残基 a 随机出现的概率为 q_a, 并且与序列上的所有其他残基都相互独立. 如果蛋白质或 DNA 序列用 $x_1 \ldots x_n$ 表示, 那么整条序列的概率就是乘积 $q_{x_1} q_{x_2} \cdots q_{x_n} = \prod_{i=1}^{n} q_{x_i}$[1]. 在本书中, 当与其他模型进行比较时, 我们把这种“随机序列模型” 当作基准模型, 或零假设.

最大似然估计法

通常而言, 概率论模型的参数是从大量可靠事例组成的集合中*估计* (estimate) 而来的, 这种集合常被称为*训练集* (training set). 例如, 残基 a 的概率 q_a 可估计为该残基在某一已知蛋白序列库如 SWISS-PROT [Bairoch, Bucher & Hofmann 1997] 中的观测频率. 我们对该数据库的约 2 千万个残基一一计数得到 20 个频率. 我们有如此之多的数据, 因此只要训练序列没有针对奇特残基组分的系统偏差, 我们就可预料这些频率是模型中概率的合理估计. 这种估计模型的方式称为*最大似然估计法* (maximum likelihood estimation), 因为可以证明, 使用氨基酸在数据库中出现的频率作为概率 q_a, 能使给定模型下所有序列的总概率 (似然) 最大. 一般而言, 给定带参数 θ 的模型和数据集 D, θ 的最大似然估计是使 $P(D|\theta)$ 最大化之值. 更严格的讨论见第 11 章.

当我们从有限的数据中估计模型参数时, 存在发生*过拟合* (overfitting) 的危险, 即模型能很好的适应训练数据, 但是却不能很好的*推广* (generalise) 到新的数据. 观察下面这个例子: 假如我们连掷三次硬币得到的结果是 [**反面**, **反面**, **反面**], 那么使用最大似然估计法会有, **正面**出现的概率为 0, 而**反面**出现的概率为 1. 不久我们将会回过头讨论避免过拟合的方法.

[1] 严格地讲, 只有当所有序列长度相同时这个模型才是正确的, 因为只有这样, 所有可能序列的概率之和才是 1; 见第 3 章.

条件概率、联合概率与边际概率

假设我们有两个骰子, D_1 和 D_2. 记掷骰子 D_1 得到点数 i 的概率为 $P(i|D_1)$, 这是在给定骰子 D_1 的条件下掷得点数 i 的条件概率 (conditional probability). 如果我们以概率 $P(D_j), j = 1, 2$ 随机选取一个要掷的骰子, 那么选到骰子 j 并掷得点数 i 的概率是两个概率的乘积 $P(i, D_j) = P(D_j)P(i|D_j)$. 其中 $P(i, D_j)$ 称为联合概率 (joint probability). 命题

$$P(X, Y) = P(X|Y)P(Y) \tag{1.1}$$

对任意事件 X、Y 都成立.

当已知条件概率或联合概率时, 我们可以通过下式消去一个变量, 计算边际概率 (marginal probability):

$$P(X) = \sum_Y P(X, Y) = \sum_Y P(X|Y)P(Y),$$

即对所有可能的事件Y求和.

练习

1.1 考虑一个偶而作弊的赌博游戏, 它使用两类不同的骰子. 骰子有 99% 的可能是公平的 (D_{fair}) 但有 1% 的可能是作弊的 (D_{loaded}), 作弊时六点 (six) 出现的概率可高达 50%. 随机从桌上选出一个骰子掷出, 问概率 $P(\text{six}|D_{\text{loaded}})$ 和 $P(\text{six}|D_{\text{fair}})$ 分别是多少? $P(\text{six}, D_{\text{loaded}})$ 和 $P(\text{six}, D_{\text{fair}})$ 又是多少? 我们所选出的骰子掷出一个六点的概率是多少?

贝叶斯定理与模型比较

在与练习 1.1 中相同的偶尔作弊的赌博游戏中, 我们随机地选取一个骰子投掷三次, 结果得到三个连续六点 (3 sixes). 我们怀疑这可能是一个作弊的骰子, 但是如何评估到底是不是作弊呢? 我们想要知道的是 $P(D_{\text{loaded}}|3 \text{ sixes})$; 即在给定观测数据下假设骰子作弊的后验概率 (posterior probability). 但我们能直接计算的却是, 给定该假设后得到观测数据的概率, 即 $P(3 \text{ sixes}|D_{\text{loaded}})$. 这个概率称为该假设的似然 (likelihood). 然后我们可以通过 Bayes 定理

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)} \tag{1.2}$$

计算后验概率. 在 (1.2) 式中, X 对应事件“骰子为作弊的”; Y 对应事件“掷出三个六点”, 于是有

$$P(D_{\text{loaded}}|3 \text{ sixes}) = \frac{P(3 \text{ sixes}|D_{\text{loaded}})P(D_{\text{loaded}})}{P(3 \text{ sixes})}.$$

我们已知 (见练习 1.1) 选择作弊骰子的概率 $P(D_{\text{loaded}})$ 是 0.01, 同时知道作弊骰子连续掷出三个六点的概率 $P(3 \text{ sixes}|D_{\text{loaded}}) = 0.5^3 = 0.125$. 则“掷出三个六点”的总概率 $P(3 \text{ sixes})$ 恰好是 $P(3 \text{ sixes}|D_{\text{loaded}})P(D_{\text{loaded}}) + P(3 \text{ sixes}|D_{\text{fair}})P(D_{\text{fair}})$. 代入上式计算得:

$$\begin{aligned} P(D_{\text{loaded}}|3 \text{ sixes}) &= \frac{(0.5^3)(0.01)}{(0.5^3)(0.01) + (\frac{1}{6})^3(0.99)} \\ &= 0.21. \end{aligned}$$

因此事实上, 虽然我们掷出了三个连续六点, 但这个骰子更有可能是公平的.

再看一个更贴近生物学的例子. 假设我们相信, 平均来讲, 胞外蛋白 (extracellular proteins) 与胞内蛋白 (intracellular proteins) 在氨基酸的组成上存在细微差异. 例如, 我们通常认为半胱氨酸在胞外蛋白中比在胞内蛋白中出现的更普遍. 我们试

着通过这些信息判断一条新的蛋白序列 $x = x_1 \dots x_n$ 是属于胞内还是胞外. 为此, 我们先将取自 SWISS-PROT 的训练集分为胞内蛋白和胞外蛋白两类 (可以把无法判断的例子搁置一边).

然后, 我们可以估计出与胞内蛋白 (用 int 表示) 对应的一组频率 q_a^{int} 和与胞外蛋白 (用 ext 表示) 对应的一组频率 q_a^{ext}. 为了提供 Bayes 定理需要的全部信息, 我们还需要估计: 对于任意一条新序列, 它属于胞外蛋白的概率 p^{ext} 和属于胞内蛋白的概率 p^{int}. 此处, 我们还要假设每一条序列要么完全属于胞内蛋白, 要么完全属于胞外蛋白, 即 $p^{\text{int}} = 1 - p^{\text{ext}}$. 这里的 p^{ext} 和 p^{int} 称为*先验概率* (prior probability), 因为它们代表了在看到有关序列本身的任何信息之前, 我们能作出的关于一条序列的最好猜测.

现在可以得到 $P(x|\text{ext}) = \prod_i q_{x_i}^{\text{ext}}$ 和 $P(x|\text{int}) = \prod_i q_{x_i}^{\text{int}}$. 因为我们已经假设每条序列要么属于胞内蛋白, 要么属于胞外蛋白, 所以 $p(x) = p^{\text{ext}} P(x|\text{ext}) + p^{\text{int}} P(x|\text{int})$. 由 Bayes 定理可得:

$$P(\text{ext}|x) = \frac{p^{\text{ext}} \prod_i q_{x_i}^{\text{ext}}}{p^{\text{ext}} \prod_i q_{x_i}^{\text{ext}} + p^{\text{int}} \prod_i q_{x_i}^{\text{int}}}.$$

$P(\text{ext}|x)$ 正是我们需要的数, 称为序列是胞外蛋白的*后验概率* (posterior probability), 因为它表示我们在看到数据*之后*作出的最好猜测.

当然这个例子多少有点令人困惑, 因为许多跨膜蛋白 同时具有胞内组分和胞外组分. 我们真正需要的是能在序列上指出不同结构间的转换. 这需要更复杂的概率论模型, 我们将在后面 (第 3 章) 对此问题进行分析.

练习

1.2 在以上的掷骰子例子中, 需要看到多少个连续的六点, 才能认为我们更有可能使用了一个作弊骰子?

1.3 使用方程 (1.1) 证明 Bayes 定理.

1.4 人们发现了一种罕见的遗传疾病. 虽然平均在一百万人中, 才有一个人患这种疾病, 但是你仍然考虑作医学筛查. 你被告知该遗传检验是相当有效的, 其敏感性高达 100% (只要你有这种疾病, 它的检测结果就总正确), 而特异性也高达 99.99% (即每次只有 0.01% 的假阳性结果). 运用 Bayes 定理解释为什么你不该选择这种医学检验.

Bayes 参数估计

此前我们提到了过拟合的概念. 如果没有足够多的数据对参数进行可靠地估计, 我们可以利用*先验知识* (prior knowledge) 对估计进行约束, 而不至于放弃整个模型. 这可由 Bayes 参数估计方便地实现.

Bayes 定理除了可用于模型比较之外, 还可用于参数估计. 给定某数据集 D 后, 我们可以使用 Bayes 定理计算出任意特定参数集 θ 的后验概率

$$P(\theta|D) = \frac{P(\theta)P(D|\theta)}{\int_{\theta'} P(\theta')P(D|\theta')}. \tag{1.3}$$

请注意由于参数通常是连续变量, 而不是离散的, 所以上式等号右端中的分母不是求和而是求积分:

$$P(D) = \int_{\theta'} P(\theta')P(D|\theta').$$

从 (1.3) 式可以引出许多问题. 其中一个便是 "$P(\theta)$ 意味着什么?" 我们从哪里得到参数的先验分布? 有时, 没有很好的理论基础能让我们做出明确的选择, 此时通常选取*平坦* (flat) 的均匀先验或其他*不含信息的* (uninformative) 先验, 即那些尽可能无影响的先验. 对于其他情形, 我们却希望使用包含信息的 $P(\theta)$. 例如, 我们预先知道: 苯丙氨酸、酪氨酸、色氨酸三者在结构上是相似的, 通常在进化上它们是可

以互相替换的. 因此我们希望使用这样的 $P(\theta)$, 它倾向于选择使这三种氨基酸概率相近而不是差异较大的参数集. 第 5 章将详细研究这些问题.

再一个问题就是如何使用 (1.3) 式估计出好的参数. 有一种方法是选取使 $P(\theta|D)$ 最大的 θ 值. 这种方法称为最大后验估计法. 注意, (1.3) 式等号右端中的分母与特定的 θ 值无关, 因此最大后验估计法对应于似然和先验乘积的最大化. 如果先验是平坦的, 那么最大后验估计法就等价于最大似然估计法.

另一种参数估计的方法是选择后验分布的均值而不是最大值, 作为估计. 这种方法在操作上略有些复杂, 它要求后验概率要么能被解析计算, 要么能被抽样. 与之相关的一种方法根本不选取某个特定的参数集, 而是基于许多甚至所有不同参数值处的模型, 通过积分评估感兴趣的值, 并按照各自参数值的后验概率对结果进行加权. 当评估与加权可以解析地进行时, 这种方法将变得非常有吸引力——否则很难得到一个有效的结果, 除非参数空间非常小.

上述这些方法属于统计学的一个领域——Bayes 统计 [Box & Tiao 1992]. 虽然 Bayes 定理本身在处理条件概率上的合理性并无争议, 但像如何选择先验这样的问题所带有的主观性, 却导致一些人对 Bayes 方法持谨慎态度. 对此, 我们不采取过分严格的态度; 最大似然法和 Bayes 方法在本书的不同地方都得到了运用. 但是, 当从少量数据出发估计大的参数集时, 我们相信 Bayes 方法提供了一致的形式体系, 用于从相同类型数据的先前经验中产生额外的信息.

例子: 估计作弊骰子的概率

为了便于理解, 让我们回到掷骰子的例子. 假设我们有一个骰子, 并且猜测它是作弊的, 但是我们不知道它究竟是不是. 只允许掷十次这个骰子, 然后我们就必须做出对参数 p_i 的最好估计. 我们得到的观测序列是 $1, 3, 4, 2, 4, 6, 2, 1, 2, 2$. 基于观测频率, p_5 的最大似然估计应该是 0. 如果以此作为模型参数, 那么只须在某结果中观测到一个 5, 我们就能够断定该结果不是从这个骰子掷出的. 看起来这过于苛刻. 从直觉上讲, 我们并没有足够的数据以确定这个骰子真的就掷不出 5 点.

解决这一问题的常用方法是, 通过给每一种结果的真实观测数值加上一些假的额外计数, 以调整用于导出概率的观测频率. 在这个例子中, 我们可以为每个观测计数都加上 1, 现在估计出的概率 $\hat{p}_5$ 便是 $\frac{1}{16}$. 为每一类加上的这个额外计数被称作*伪计数* (pseudocount). 使用伪计数的方法对应于后验均值方法, 该方法使用 Bayes 定理以及来自于 Dirichlet 分布家族中的一个先验 (详见第 11 章). 不同的伪计数集合对应于关于骰子概率类型的不同先验假设. 如果先前的经验告诉我们绝大多数骰子都接近公平, 那么我们就可以多加入一些伪计数; 如果之前就在这个特定的赌局中看到有许多骰子都存在强烈的偏好性, 那么我们会更加相信在此特定例子中所收集到的数据并减少伪计数的权重. 当然, 如果我们收集了足够多的观测数据, 则与伪计数相比, 真实的计数将总会起决定性作用.

在图 1.2 中, 作为 p_5 的函数, 似然 $P(D|\theta)$ 在0处取得最大值是很明显的. 在同一幅图中, 我们还绘出了 p_5 的先验分布和每类都带 5 个伪计数的后验分布. 这种伪计数方法意味着, p_5 的先验分布 $P(\theta)$ 是一个 Dirichlet 分布. 请注意后验分布 $P(\theta|D)$ 是不对称的; p_5 的后验均值估计略大于 MAP 估计. □

练习

1.5 在上例中, 求掷出一个点数 2 的概率 p_2 的最大似然估计. 如果为每类结果都添加伪计数 1, 那么它的 Bayes 估计是多少? 如果为每类结果都添加伪计数 5, 它的 Bayes 估计又是多少?

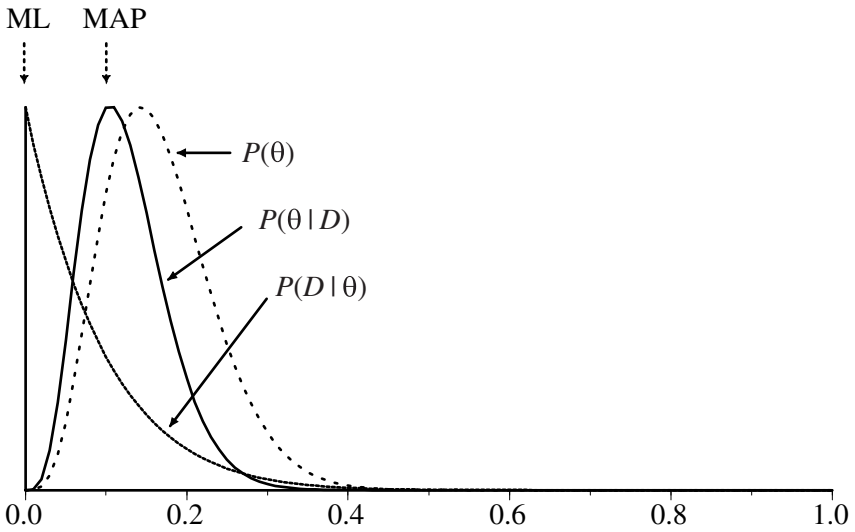


图 1.2: 例 1.1 中为每一类结果都增加 5 个伪计数后概率 p_5 (x 轴) 的最大似然估计法 (ML) 和最大后验估计法 (MAP) 之对比. 三条曲线均已经归一化为具有相同的最大值.

1.4 补充读物

计算分子生物学方面可供阅读的教材有: Waterman [1995] 的 Introduction to Computational Biology, Baldi & Brunak [1998] 的 Bioinformatics - The Machine Learning Approach 以及 Sankoff & Kruskal [1983] 的 Time Warps, String Edits, and Macromolecules. 我们向没有分子生物学背景的读者推荐 Watson *et al.* [1987] 的 Molecular Biology of the Gene, 这是一本大学本科水平的分子遗传学导论, 可读性强, 包罗万象. Branden & Tooze [1991] 的 Introduction to Protein Structure 是一本图文并茂的关于蛋白质三维结构的入门读物. MacKay [1992] 对 Bayes 概率论建模做过权威的讲解; Jefferys & Berger [1992] 对 Bayes 方法背后许多引人瞩目的思想作出了更初等的介绍.

第二章

二序列联配

2.1 引言

最基本的序列分析工作就是判断两条序列是否相关. 通常解决这种问题的方法是, 先将这两条序列 (或者它们的一部分) 进行联配, 然后判断这种联配的出现是否更有可能是由于序列间的相关性导致的, 或只是一种巧合. 关键问题有: (1) 应该考虑何种类型的联配; (2) 为联配排序的计分系统; (3) 找到最优或较好联配所使用的算法; (4) 评估联配分数显著性水平所使用的统计方法.

图 2.1 给出了三组二序列联配的实例, 它们都联配到人类 α 球蛋白序列 (SWISS-PROT 数据库中的检索代码是 `HBA_ HUMAN`) 的同一区域. 在每组联配的中间一行, 用字母表示等同 (identical) 位点, 用加号表示“相似” (similar) 位点. (“相似”的残基对是指为联配计分的替换矩阵 (substitution matrix) 中得正分的残基对; 我们稍后将讨论替换矩阵.) 先来看第一组联配, 这一组中有很多等同残基位点; 另外还有很多位点在功能上是保守的, 例如接近尾端的 `D-E` 配对, 表示天冬氨酸 (aspartic acid) 和谷氨酸 (glutamic acid) 的残基联配, 而两者都是带负电的氨基酸. 图 2.1b 也给出了一组具有生物学意义的联配结果, 因为我们知道这两条序列在进化上是相关的, 并且具有相同的三维结构以及氧结合功能. 但是此例有多个短等同段, 并且 α 球蛋白中某几个位置处插入了空位, 以便跨越豆血红蛋白 `LGB2_LUPLU` 中的额外残基区域而维持联配. 在图 2.1c 的联配中, 虽然等同位点和保守性替换的数目与上面的情形相类似, 但是我们看到的是一个假 (spurious) 联配, 因为它联配到一个结构和功能完全不同的蛋白.

如何才能区分图 2.1b 与图 2.1c 中的情形呢? 这对二序列联配方法来讲是一个挑战. 我们必须仔细推敲用于评估联配的计分系统. 下一节我们将会介绍如何给联配计分, 而随后的几小节将讨论如何根据计分方案找到最优联配的方法. 在本章最后, 我们将讨论匹配的统计显著性, 并给出计分方案参数化的更多细节. 即便如此, 有时我们仍然不能完全区分真假联配. 例如在图 2.1b 中, 想要通过二序列联配方法找到豆血红蛋白与人类 α 球蛋白之间的显著相似性, 实际上是极其困难的.

```
(a)
HBA_HUMAN  GSAQVKGHGKKVADALTNAVAHVDDMPNALSALSDLHAHKL
           G+ +VK+HGKKV  A+++++AH+D++ +++++LS+LH  KL
HBB_HUMAN  GNPKVKAHGKKVLGAFSDGLAHLDNLKGTFATLSELHCDKL

(b)
HBA_HUMAN  GSAQVKGHGKKVADALTNAVAHV---D--DMPNALSALSDLHAHKL
           ++ ++++H+ KV   + +A  ++          +L+ L+++H+ K
LGB2_LUPLU NNPELQAHAGKVFKLVYEAAIQLQVTGVVVTDATLKNLGSVHVSKG

(c)
HBA_HUMAN  GSAQVKGHGKKVADALTNAVAHVDDMPNALSALSD----LHAHKL
           GS+ + G +   +D L  ++ H+ D+  A +AL D    ++AH+
F11G11.2   GSGYLVGDSLTFVDLL--VAQHTADLLAANAALLDEFPQFKAHQE
```

图 2.1: 对人类 α 球蛋白片断的三组联配结果. (a) 与人类 β 球蛋白明显相似的联配. (b) 从结构上看, 联配到黄色羽扇豆 (yellow lupin) 的豆血红蛋白 (leghaemoglobin) 的似真 (plausible) 联配. (c) 联配到线虫谷胱甘肽 S-转移酶 (glutathione S-transferase) 同源蛋白 F11G11.2 的假高分联配.

2.2 计分模型

比较序列时, 我们寻找证据证实它们是经突变和选择过程由共同祖先分化而来的. 人们考虑的基本突变过程包括替换 (substitution) 即改换序列中的残基, 以及插入 (insertion) 和删除 (deletion) 即添加或去除残基. 插入和删除统称为空位 (gap)自然选择通过筛选突变对这个过程产生一定的影响, 以至于某些种类的改变被观察到的可能性更大.

我们赋予一个联配的总分, 是每个残基配对项与每个空位项之和. 如果用概率论的语言来阐述, 这就对应于序列相关之于无关的对数相对似然 (logarithm of the relative likelihood). 通俗地讲, 我们期望联配中等同位点和保守型替换比在偶然情况下出现的可能性要大, 所以它们贡献正分值项; 并且期望真实联配中的不保守改变比偶然情况下出现的可能性要小, 所以它们贡献负分值项.

使用此类可加的计分方案对应于这样一个假设, 即序列不同位点上突变的发生是相互独立的 (将任意长的连续空位当作一个突变来对待). 本章所列出的寻找最优联配的算法都基于这种计分方案. 尽管我们知道残基间的内在相互作用在决定蛋白质结构方面会产生极其重要的影响, 但基于这种独立性假设所得到的结果仍可以看作是 DNA 或蛋白序列联配的合理近似. 然而对于结构 RNA 而言, 这个计分方案就变得十分不精确, 因为在结构 RNA 中存在着重要的碱基对长程关联. 我们可以将这种相关性也考虑进来, 不过这会使计算复杂度显著上升; 因此我们将 RNA 联配问题安排在本书末尾 (第 10 章).

替换矩阵

我们需要每种联配残基对的计分项. 一个对蛋白质有敏锐直觉的生物学家可以为所有 210 种可能的氨基酸配对给出计分项, 但是尽管如此, 一个能指明计分具体意义的指导理论仍然是极其重要的. 接下来, 我们就从一个统计模型出发推导替换分值.

首先指明记号. 我们将考虑一对长度分别为 m 和 n 的序列 x 和 y. 令 x_i 为序列 x 的第 i 个字符, y_j 为序列 y 的第 j 个字符. 这些字符均来自某字母表 $\mathscr{A}$; 如果是 DNA 序列, 那么 $\mathscr{A}$ 就是 {`A, C, G, T`}; 而如果是蛋白序列, 则 $\mathscr{A}$ 就是全部 20 种氨基酸的集合. 我们用小写字母如 a, b 表示这个字母表中的字符. 现在我们只考虑无空位全局二序列联配 (ungapped global pairwise alignment), 即两条完全联配的等长序列, 如图 2.1a 所示.

给定一对已联配的序列, 我们想要为该联配计分, 以衡量这两条序列相关之于无关的相对似然. 首先我们使用模型分别对这两种情况的联配赋予概率值; 然后再考虑这两个概率之比.

无关或随机 (random) 模型 R 是一种最简单的模型. 该模型假设字母 a 以频率 q_a 独立地出现, 那么这两条序列的概率就是每个氨基酸概率的乘积:

$$P(x, y|R) = \prod_i q_{x_i} \prod_j q_{y_j}. \tag{2.1}$$

在另一种匹配 (match) 模型 M 中, 联配上的残基对以联合概率 p_{ab} 出现. 可以认为 p_{ab} 的值是残基 a 和 b 由它们共同祖先中某个未知的原始残基 c (c 可能就是 a 或 b) 衍变而来的概率. 这样整个联配的概率就是:

$$P(x, y|M) = \prod_i p_{x_i y_i}.$$

这两个似然的比就是我们所熟知的几率比 (odds ratio)

$$\frac{P(x, y|M)}{P(x, y|R)} = \frac{\prod_i p_{x_i y_i}}{\prod_i q_{x_i} \prod_i q_{y_i}} = \prod_i \frac{p_{x_i y_i}}{q_{x_i} q_{y_i}}.$$

为了得到可加的计分系统, 我们对该比值取对数, 称为对数几率比 (log-odds ratio)

$$S = \sum_i s(x_i, y_i), \tag{2.2}$$

其中

$$s(a, b) = \log\left(\frac{p_{ab}}{q_a q_b}\right) \tag{2.3}$$

是残基对 (a, b) 联配之于未联配的对数似然比.

如我们想要的一样, 方程 (2.2) 就是每个联配上残基对的分数 $s(a, b)$ 之和. 可以用矩阵形式列出所有 $s(a, b)$ 的值. 以蛋白序列为例, 它形成了一个 20×20 的矩阵, 其中第 i 行第 j 列的元素是 $s(a_i, a_j)$, 而 a_i 和 a_j 分别是 (在某种编号系统下) 第 i 个和第 j 个氨基酸. 这个矩阵就叫做计分矩阵 (score matrix) 或者替换矩阵 (substitution matrix). 图 2.2 给出了一个实例, 即 BLOSUM50 矩阵, 实际上它就是通过以上方法推演得到的. 我们可以利用这个矩阵为图 2.1a 中的联配计分, 结果是 130 分. 另一类经常会用到的替换矩阵叫做 PAM 矩阵. 有关 BLOSUM 和 PAM 矩阵推导过程的具体描述将在本章最后给出.

一个重要的结论是, 即使让某个富于直觉的生物学家写下一个特定的替换矩阵, 该替换矩阵也会按照上述理论暗含“目标频率” (target frequency) p_{ab} [Altschul 1991]. 任何替换矩阵都描述实际联配中观测到 ab 配对的概率.

练习

2.1 氨基酸D、E和K都带电; V、I和L都是疏水的. 那么在带电的三个氨基酸间平均 BLOSUM50 分值是多少? 在疏水的那一组里的平均分值又是多少? 在这两组间呢? 请对观测到的模式给出合理解释.

空位罚分

我们想对空位 (gap) 进行罚分 (penalise). 长度为 g 的空位的标准罚分 (cost) 可以是线性 (linear) 分值

$$\gamma(g) = -gd \tag{2.4}$$

或者是仿射 (affine) 分值

$$\gamma(g) = -d - (g - 1)e, \tag{2.5}$$

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	**5**	−2	−1	−2	−1	−1	−1	0	−2	−1	−2	−1	−1	−3	−1	1	0	−3	−2	0
R	−2	**7**	−1	−2	−4	1	0	−3	0	−4	−3	3	−2	−3	−3	−1	−1	−3	−1	−3
N	−1	−1	**7**	2	−2	0	0	0	1	−3	−4	0	−2	−4	−2	1	0	−4	−2	−3
D	−2	−2	2	**8**	−4	0	2	−1	−1	−4	−4	−1	−4	−5	−1	0	−1	−5	−3	−4
C	−1	−4	−2	−4	**13**	−3	−3	−3	−3	−2	−2	−3	−2	−2	−4	−1	−1	−5	−3	−1
Q	−1	1	0	0	−3	**7**	2	−2	1	−3	−2	2	0	−4	−1	0	−1	−1	−1	−3
E	−1	0	0	2	−3	2	**6**	−3	0	−4	−3	1	−2	−3	−1	−1	−1	−3	−2	−3
G	0	−3	0	−1	−3	−2	−3	**8**	−2	−4	−4	−2	−3	−4	−2	0	−2	−3	−3	−4
H	−2	0	1	−1	−3	1	0	−2	**10**	−4	−3	0	−1	−1	−2	−1	−2	−3	2	−4
I	−1	−4	−3	−4	−2	−3	−4	−4	−4	**5**	2	−3	2	0	−3	−3	−1	−3	−1	4
L	−2	−3	−4	−4	−2	−2	−3	−4	−3	2	**5**	−3	3	1	−4	−3	−1	−2	−1	1
K	−1	3	0	−1	−3	2	1	−2	0	−3	−3	**6**	−2	−4	−1	0	−1	−3	−2	−3
M	−1	−2	−2	−4	−2	0	−2	−3	−1	2	3	−2	**7**	0	−3	−2	−1	−1	0	1
F	−3	−3	−4	−5	−2	−4	−3	−4	−1	0	1	−4	0	**8**	−4	−3	−2	1	4	−1
P	−1	−3	−2	−1	−4	−1	−1	−2	−2	−3	−4	−1	−3	−4	**10**	−1	−1	−4	−3	−3
S	1	−1	1	0	−1	0	−1	0	−1	−3	−3	0	−2	−3	−1	**5**	2	−4	−2	−2
T	0	−1	0	−1	−1	−1	−1	−2	−2	−1	−1	−1	−1	−2	−1	2	**5**	−3	−2	0
W	−3	−3	−4	−5	−5	−1	−3	−3	−3	−3	−2	−3	−1	1	−4	−4	−3	**15**	2	−3
Y	−2	−1	−2	−3	−3	−1	−2	−3	2	−1	−1	−2	0	4	−3	−2	−2	2	**8**	−1
V	0	−3	−3	−4	−1	−3	−3	−4	−4	4	1	−3	1	−1	−3	−2	0	−3	−1	**5**

图 2.2: BLOSUM50 替换矩阵. 对数几率值已乘上因子并四舍五入为整数以提高计算效率. 主对角元对应于等同残基对, 用粗体强调.

其中 d 称为**空位开端** (gap-open) 罚分, e 称为**空位延伸** (gap-extension) 罚分. 通常来讲空位延伸罚分 e 要比空位开端罚分 d 小, 这便使得长插入和删除所罚掉的分数比在线性空位罚分法中少. 如果可以期望多残基空位和单残基空位几乎等频率, 那么这种罚分方式就是可取的.

空位罚分也对应于联配的一种概率论模型, 虽然它并不像替换矩阵的概率论基础那样被广泛认可. 我们假设在给定序列中特定位置上出现空位 (gap) 的概率是空位长度函数 $f(g)$ 和插入残基集合的整体概率之乘积:

$$P(\text{gap}) = f(g) \prod_{i \text{ in gap}} q_{x_i}. \tag{2.6}$$

式 (2.6) 取 $f(g)$ 和 q_{x_i} 项乘积的形式所对应的假设是, 空位长度与它所包含的残基无关.

此处, 概率 q_a 的自然值和在随机模型下所取值相同, 因为两者都对应于未匹配上的独立残基. 这样, 当除以随机模型下这个区域的概率而求几率比时, q_{x_i} 项就被消掉而只剩下一项依赖于空位长度的 $\gamma(g) = \log(f(g))$; 空位罚分也就对应于该长度空位的对数概率.

另一方面, 如果有证据表明空位区域有不同的残基分布, 那么空位区域的未联配残基处就应该有残基特异的分数, 而且应该等于这些残基在空位区域之于联配区域的出现频率对数比. 上述的这种情况有可能发生, 例如与在蛋白质序列中的平均频率相比, 极性氨基酸应该更有可能出现在蛋白质联配中的空位区, 因为与深埋的核心区 (buried core) 相比, 蛋白质结构表面的弯折 (loop) 中更有可能出现空位.

练习

2.2 证明方程 (2.4) 和 (2.5) 中给出的与线性和仿射空位方案相对应的概率分布 $f(g)$ 均是形如 $f(g) = k\mathrm{e}^{-\lambda g}$ 的几何分布

2.3 实际情况下, 线性空位罚分方案中使用的典型空位罚分是 $d = 8$, 仿射空位罚分方案使用的是 $d = 12, e = 2$, 此处它们都以半比特 (bit) 为单位. 比特是对概率值取以 2 为底的对数所相应的单位, 所以在自然对数单位下, 以上的三个值

分别为 $d = (8\log 2)/2$ 和 $d = (12\log 2)/2, e = (2\log 2)/2$. 请问从某一位置开始的 (任意长度的) 一个空位所对应的概率是多少? 如果存在空位, 那么它的长度应该服从怎样的分布?

2.4 用图 2.2 中的 BLOSUM50 矩阵和仿射空位罚分 $d = 12, e = 2$ 计算图 2.1b 和图 2.1c 中联配的分数. (读者也许碰巧会注意到, BLOSUM50 所用的单位是三分之一比特. 在 BLOSUM50 分数矩阵下使用空位开端为 12、空位延伸为 2 得到的罚分不同于上题中得到的空位开端/延伸概率, 因为上题假定分数的单位是二分之一比特. 我们之所以对空位罚分进行优化以便用于特定的替换矩阵, 其中一个原因是不同矩阵所用的缩放因子不同, 另一个原因是不同矩阵适用于两条序列间不同水平的期望进化分歧度.)

2.3 联配算法

给定一个计分系统, 我们需要能找到序列对最优联配的算法. 如果两条序列的长度均为 n, 那么整条序列的全局联配只可能有一种, 然而一旦允许联配中存在空位 (或在序列的子序列间寻找局部联配), 那么事情就会变得有点复杂了. 在两条长度同为 n 的序列间, 存在着

$$\binom{2n}{n} = \frac{(2n)!}{(n!)^2} \simeq \frac{2^{2n}}{\sqrt{\pi n}} \tag{2.7}$$

种可能的全局联配. 很明显即使对于比较适中的 n 值, 想要从计算上穷举所有这些联配也是不可行的. 给定我们已经描述的这种可加联配分值后, 寻找最优联配的算法称为**动态规划** (dynamic programming). 动态规划算法是计算序列分析的核心. 本书中除讲授数学方法的最后一章外, 所有其他章节都使用了动态规划算法. 最容易理解的动态规划算法是二序列联配算法. 读者应该确信理解本节所讲内容, 因为它是全书的重要基础. 动态规划算法能确保找到计分最优的联配或联配组. 多数情况下, 启发式 (heuristic) 方法也被开发而应用到同类搜索问题中. 启发式方法速度非常快，但却附加了额外假设, 并会丢失某些序列对的最好匹配. 本章稍后将对启发式搜索方法作简要探讨.

因为我们用对数几率比的形式给出计分方案, 越好的联配对应越高的分数, 所以我们要最大化分值以找到最优联配. 有时分数被赋予了其他的含义并被解释为代价 (cost) 或**编辑距离** (edit distance), 此时我们就需要最小化联配的代价. 在生物序列比较的相关文献中, 这两种方法都被使用过. 动态规划算法对这两者都适用; 其细微差别就在于前者是求最大 (max), 而后者是求最后者是求最小 (min).

我们介绍四种基本的联配. 所要寻找的联配类型取决于将要进行联配的序列. 不同类型联配的动态规划算法间存在稍许差异. 本节中, 我们只描述基于线性空位分值的二序列联配, 每个残基位点的空位罚分为 d. 但是, 这里所介绍的算法很容易扩展到更复杂的空位模型, 读者在本章稍后就会看到了.

我们将使用两条很短的氨基酸序列阐述各种不同的联配方法, 这两条序列是 `HEAGAWGHEE` 和 `PAWHEAE`. 为联配计分时, 我们使用 BLOSUM50 计分矩阵, 并指定每个未联配残基处的空位罚分为 $d = -8$. 图 2.3 中的矩阵 s_{ij} 给出了把这两条示例序列的每对残基联配到一起的局部分值 $s(x_i, y_j)$. 等同和保守的残基对用粗体强调. 通俗地讲, 联配算法的目的就是将具有正分的残基对尽可能多的包含在联配中, 同时使来自非保守残基对、空位和其他限制的损失达到最小.

练习

2.5 假设含空位的序列联配不允许第二条序列中的空位位于第一条序列中的空位之前; 即允许形如 `ABC/A-C` 和 `A-CD/AB-D` 的联配, 但不允许形如 `AB-D/A-CD` 的联配. (这种限制显得很自然, 因为已联配残基对间的区域能够以许多我们并不关心的方式构成联配.) 证明在保持各自序列中字符顺序不变的情况下, 共有 $\binom{n+m}{m}$ 种方法将两条长度分别为 n 和 m 的序列合并成为一条长度为 $n+m$ 的序列.

	H	E	A	G	A	W	G	H	E	E
P	−2	−1	−1	−2	−1	−4	−2	−2	−1	−1
A	−2	−1	**5**	0	**5**	−3	0	−2	−1	−1
W	−3	−3	−3	−3	−3	**15**	−3	−3	−3	−3
H	**10**	0	−2	−2	−2	−3	−2	**10**	0	0
E	0	**6**	−1	−3	−1	−3	−3	0	**6**	**6**
A	−2	−1	**5**	0	**5**	−3	0	−2	−1	−1
E	0	**6**	−1	−3	−1	−3	−3	0	**6**	**6**

图 2.3: 两条用来阐述动态规划联配算法的序列, 它们排成矩阵形式, 并且每个矩阵元对应于 BLOSUM50 中相应残基对的值. 正分用粗体表示.

I G A x_i	A I G A x_i	G A x_i − −
L G V y_j	G V y_j − −	S L G V y_j

图 2.4: 一个联配可以延伸到 (i, j) 的三种方式: x_i 联配上 y_j (左图), x_i 联配上一个空位 (中图), y_j 联配上一个空位 (右图).

2.6 通过先从联配的上下两条序列中交替地挑出字符, 再扔掉空位的方法, 证明含空位联配和练习2.5中描述的序列合并间存在一一对应. 并由此推导方程 (2.7) 的前半部分.

2.7 使用斯特林公式 (Stirling's formula) $x! \simeq \sqrt{2\pi}x^{x+\frac{1}{2}}\mathrm{e}^{-x}$ 证明方程 (2.7) 的后半部分.

全局联配: Needleman-Wunsch 算法

我们考虑的第一个问题是, 在允许空位存在的条件下如何得到两条序列的最优全局联配. 在生物序列分析领域, 解决这类问题的著名动态规划算法是 Needleman-Wunsch 算法 [Needleman & Wunsch 1970], 但是我们这里所描述的更高效算法由 Gotoh [1982] 所引入.

该算法的思想是, 用较短子序列最优联配的先前解决方案建立最优联配. 我们构造带下标 i, j 的矩阵 F, 其中两个下标分别对应两条序列, 矩阵元素 $F(i, j)$ 的值是 x 结束于 x_i 的初始片段 $x_{1\ldots i}$ 与 y 结束于 y_j 的初始片段 $y_{1\ldots j}$ 间最优联配的分值. 我们可以递归地计算 $F(i, j)$. 首先初始化 $F(0, 0) = 0$, 然后按从左上到右下的顺序填充矩阵元. 只要 $F(i-1, j-1)$、$F(i-1, j)$ 和 $F(i, j-1)$ 均已知, 我们就可以计算 $F(i, j)$. 对于结束于 x_i 和 y_j 的联配, 最优分 $F(i, j)$ 的来源有三种可能: x_i 联配上y_j, 那么 $F(i, j) = F(i-1, j-1) + s(x_i, y_j)$; 或者 x_i 联配上一个空位, 那么 $F(i, j) = F(i-1, j) - d$; 或者 y_j 联配上一个空位, 那么 $F(i, j) = F(i, j-1) - d$ (参见图2.4). 结束于 (i, j) 的最优分值就是这三种情况的最大值.

因此我们有

$$F(i,j) = \max \begin{cases} F(i-1, j-1) + s(x_i, y_j), \\ F(i-1, j) - d, \\ F(i, j-1) - d. \end{cases} \tag{2.8}$$

反复调用该公式便可填充矩阵元 $F(i, j)$: 如下图所示, 每次调用都从四个方格中左上、左侧或上侧的三个单元格中的一个值计算右下单元格中的值.

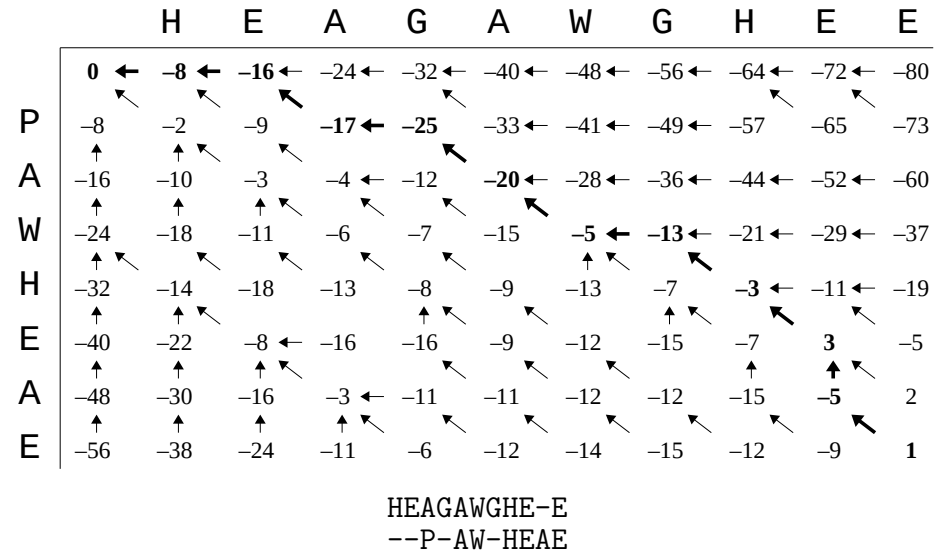
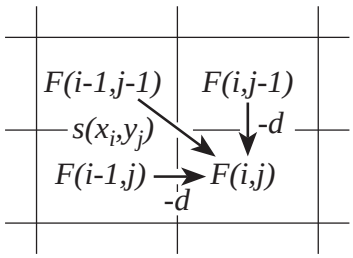


图 2.5: 上图: 示例序列的全局动态规划矩阵, 箭头表示回溯指针; 最优联配路径上的值被加粗显示. (在多条回溯路径具有相同最优分数的退化情形, 只显示一个箭头.) 下图: 与之对应的最优联配, 总分为 1.



在填充矩阵元 $F(i,j)$ 时, 我们同时在每个单元格中保存了一个指针, 用于指向导出 $F(i,j)$ 的上一单元格. 示例序列的完整动态规划矩阵参见图 2.5.

为了完善算法说明, 我们必须处理边界条件. 该矩阵的第一行处, 即 $j=0$ 时, $F(i,j-1)$ 和 $F(i-1,j-1)$ 都没有定义, 所以我们需要对 $F(i,0)$ 进行特殊的处理. $F(i,0)$ 代表 x 的前缀联配上 y 的所有空位, 所以我们可以定义 $F(i,0)=-id$. 同理可以定义矩阵第一列处 $F(0,j)=-jd$.

矩阵最后一个单元 $F(n,m)$ 的值被定义为 $x_{1\ldots n}$ 和 $y_{1\ldots m}$ 联配的最优分值, 也就是我们所求的 x 与 y 的最优全局联配分值. 要找到联配本身, 我们就需要由式 (2.8) 找到获得最终分值所走过的路径. 这一步称为**回溯** (traceback). 实现该过程需要先反向构建联配, 从终止元开始, 并按照我们在构建矩阵时所储存的指针路线走. 在每一步回溯的过程中, 我们都从当前元 (i,j) 走回到 $(i-1,j-1)$、$(i-1,j)$ 或 $(i,j-1)$ 中推出 $F(i,j)$ 的那个矩阵元. 与此同时我们在当前的联配前加入一对字符: 如果是 $(i-1,j-1)$ 元则加入 x_i 和 y_j, 如果是 $(i-1,j)$ 元则加入 x_i 和空位字符“$-$”, 而如果是 $(i,j-1)$ 元则加入“$-$”和 y_j. 直到最后我们会回到矩阵起点 $i=j=0$. 图 2.5 绘出了这一过程.

请注意, 实际上这里所描述的回溯过程只给出了一个带最优分联配; 如果在某一点上两个推导得分相同, 那么则该算法会任意选择其中一个. 可以很容易地修改回溯算法, 以便恢复带相等分值的多种最优联配. 用序列图结构 (sequence graph structure) 可以相当简洁的描述出所有可能的最优联配集 [Altschul & Erickson 1986; Hein 1989a]. 我们将在第7章中描述 Hein 的多序列联配算法时, 使用序列图结构.

此算法能够奏效的原因是, 总分值是相对独立片断分值的累加和, 所以到联配中某点的最优分值就等于到当前点上一步的最优分值加上当前步的分值增量.

算法复杂度的大 O 标记

了解一个算法所占用的 CPU 时间和内存大小如何根据问题规模变化是非常有用的. 在上述这个算法中, 我们知道一共储存了 $(n+1)\times(m+1)$ 个数, 每个数都需要一段固定的计算时间 (三次加法运算和一次求最大值的运算). 我们说这个算法占用 $O(nm)$ 的时间和 $O(nm)$ 的内存, 其中 n 和 m 是序列的长度. "$O(nm)$" 是一个标准记号, 称为大 O 标记, 意味着 "与nm同阶" (order), 即解决这个问题所需要的计算时间或内存大小随序列长度的乘积 nm 再乘上某常量而变化. 因为一般来讲 n 和 m 不会相差太多, 所以该算法通常也被认为是 $O(n^2)$ 的. n 的指数越大, 算法对于长序列的可操作性就越低. 对于生物学序列和标准的计算机而言, $O(n^2)$ 算法是可行的但却有点慢, 而 $O(n^3)$ 算法则只能应用到非常短的序列上.

练习

2.8 在图 2.5 的动态规划矩阵中找出第二个相等分值的最优联配.

2.9 对于给定序列 `GAATTC` 和 `GATTA`, 写出对应的动态规划矩阵和最优联配, 匹配 +2 分, 失配 -1 分, 使用线性空位罚分模型, $d=2$.

局部联配: Smith-Waterman 算法

到目前为止假设我们已知需要联配哪些序列, 而且假设我们要寻找它们之间从头到尾的最好匹配. 更为普遍的情形是在 x 和 y 的*子序列* (subsequence) 中搜索最优联配. 例如, 当怀疑两条蛋白质序列可能都含有同一种结构域 (domain), 或在比较基因组 DNA 序列的扩展区域时, 都会发生这种情况. 当比较两条高度分化的序列时, 搜索子序列最优联配的方法通常也是探寻其相似性的最灵敏方法, 该方法甚至还适用于两条完整序列拥有共同进化祖先的情形. 因为在这种情形下, 往往仅有一部分序列处于足够强的选择压力作用下而保留了能够被侦测到的相似性; 其余部分则由突变积聚了太多的噪声以至于无法形成联配. x 和 y 间子序列的最高分联配称为最优*局部* (local) 联配.

寻找最优局部联配的算法与上一节描述的全局联配算法密切相关. 但二者间有两点不同. 第一点是, 在动态规划矩阵表格的每个单元中我们都需要向式 (2.8) 额外附加一个选项, 即当所有其他可能值都小于0时令 $F(i,j)$ 为0:

$$F(i,j)=\max\begin{cases}0,\\ F(i-1,j-1)+s(x_i,y_j),\\ F(i-1,j)-d,\\ F(i,j-1)-d.\end{cases}\tag{2.9}$$

上式中选项 0 对应于开始一条新联配. 如果最好的联配在延伸到某一位置时出现负分, 那么最好开始一个新联配而不是继续延伸. 注意, 加入 0 值的一个结果便是将矩阵的第一行和第一列都清 0, 不再像全局联配中那样分别等于 $-id$ 和 $-jd$ 了.

第二点不同是联配可以终止于矩阵中的任意位置, 所以我们不再以矩阵右下角元素值 $F(n,m)$ 作为最好分值, 而是从整个矩阵中寻找 $F(i,j)$ 的最高值, 并从那里开始回溯. 遇到 0 值单元时我们就停止回溯, 因为这个单元对应于联配的起点. 图 2.6 显示了两条序列的最优局部联配, 相同序列的最优全局联配见图 2.5. 此例中的局部联配是全局联配的一个子集, 但情况并不总是这样.

为了让局部联配算法有效运作, 随机匹配的期望分值必须是负的. 因为如果不是这样, 那么两条完全无关序列间的长匹配就会因其长度而计很高的分数. 所以虽然算法是局部的, 但最大计分联配也将是全局的或近似全局的. 一个真实的子序列联配很可能因为长度的原因被一个更长但错误的联配所掩盖. 与此类似, 替换矩阵中必须存在某些大于 0 的 $s(a,b)$, 否则该算法将根本无法找到任何联配 (此算法只能找到最好分值或 0 中的较大者).

要求随机匹配的期望分值为负值的精确意义是什么呢? 在无空位情形中, 我们要考虑的相关量是固定长度联配的期望值. 因为相邻位置相互独立, 所以我们只需

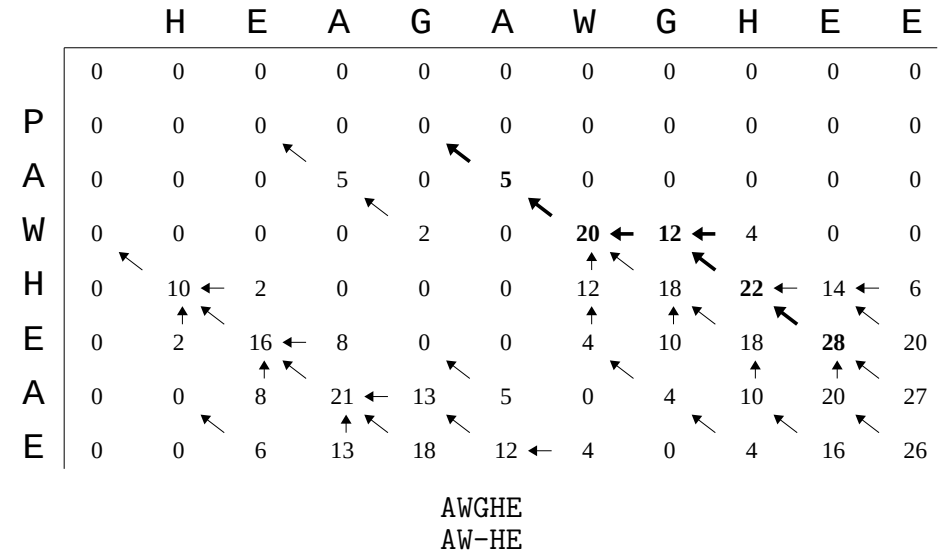


图 2.6: 上图: 示例序列的局部动态规划矩阵. 下图: 相应的局部最优联配, 总分是28.

要考虑单残基位置并给出条件

$$\sum_{a,b} q_a q_b s(a,b) < 0, \tag{2.10}$$

其中 q_a 代表字符 a 在序列中任意给定位置的概率. 当和上节一样, $s(a,b)$ 是推导出的对数似然比, 并使用相同的 q_a 作为随机模型的概率时, 式 (2.10) 恒成立. 这是因为

$$\sum_{a,b} q_a q_b s(a,b) = -\sum_{a,b} q_a q_b \log \frac{q_a q_b}{p_{ab}} = -H(q^2 \| p),$$

其中 $H(q^2 \parallel p)$ 是分布 $q^2 = q \times q$ 关于分布 p 的相对熵 , 该值恒为正除非 $q^2 = p$ (参见第 11 章). 实际上 $H(q^2 \parallel p)$ 是两个分布间区别程度的一种自然度量. 按定义, 它也是联配中每个联配上残基对包含我们所期望的信息多少的度量.

很不幸, 我们无法为最优含空位联配给出一个等价的分析. 不存在解析的方法以预测空位分值将导致何种局部或全局联配行为. 然而在为计分系统设定参数 (匹配分值 $s(a,b)$ 和空位分值 $\gamma(g)$) 时, 这却是一个实践上的重要问题, 并且人们已经为标准计分方案生成了表格以便揭示局部/全局行为, 以及其他一些统计特性 [Altschul & Gish 1996]. 我们将在考虑分值的统计显著性时再回到这个主题.

动态规划序列联配算法的局部形式早在 20 世纪 80 年代就已经诞生了. 人们经常使用的是以 Smith & Waterman [1981] 命名的 Smith-Waterman 算法. 我们通常所用的高效仿射空位罚分算法则由 Gotoh [1982] 给出 (仿射空位罚分算法见原书第 21 页).

重复匹配 (repeated match)

上一节给出了两条序列间单个最好局部匹配. 如果序列中一条或两条都较长, 那么就很有可能存在许多带显著分值的不同局部联配, 多数情形下, 我们对所有这些联配都感兴趣. 例如在一个蛋白质中可能存在很多重复的功能域 (domain) 或模体 (motif) 拷贝. 下面给出一种找到这些匹配的方法. 该方法是非对称的: 它只能在一条序列中找到另一条序列中某一部分 (如功能域或模体) 的一个或多个无交叠 (non-overlapping) 拷贝. 还有一种方法广泛用于寻找多重匹配 [Waterman & Eggert 1987] , 我们将会在第 4 章讲述.

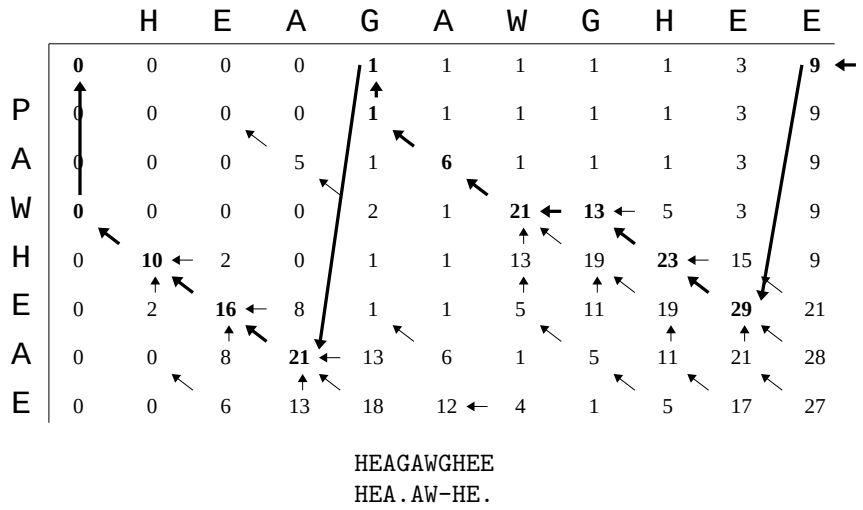


图 2.7: 上图: 示例序列的重复动态规划矩阵, $T = 20$. 下图: 总分是 $9 = 29 - 20$ 的最优联配. 本例中存在两组分离的匹配区域, 分数分别是 1 和 8. 下图中的点号表示 x 中未匹配的位点.

假设只对分值高于某一个阈值 (threshold) T 的匹配感兴趣. 通常这样做都是合理的, 因为即使是在两条完全无关的序列中, 我们仍然总能找到带小的正分的短局部联配. 令序列 y 为包含结构域或模体的序列, 而 x 为我们要从中找到多重匹配的序列.

图 2.7 给出了重复匹配算法的实例. 我们再次使用矩阵 F, 但这一次递归部分即 $F(i,j)$ 的含义有所不同. 在最终的联配中, x 将被划分为与序列 y 通过含空位联配而匹配上的部分, 以及未匹配的部分. 我们将讨论完整匹配区域的总分, 它等于标准的含空位联配分值减去阈值 T. 所有这些匹配的得分都会是正分. 现在假设 x_i 处于匹配区域内, 而且相应的匹配分别结束于 x_i 和 yj (如果这是匹配的一个含空位部分, 那么 x_i 和 y_j 可能并未真正地联配上), 那么当 $j \geq 1$ 时 $F(i,j)$ 就是到 $x_{1\ldots i}$ 的匹配分值之和的最好值. 而 $F(i,0)$ 则是到子序列 $x_{1\ldots i}$ 的完整匹配分值之和的最好值, 即假设 x_i 落入了未匹配区域.

为了达到预想的目标, 我们像往常一样先初始化 $F(0,0) = 0$, 然后按照如下递归关系填充矩阵:

$$F(i,0) = \max \begin{cases} F(i-1,0), \\ F(i-1,j) - T, \quad j = 1,\ldots,m; \end{cases} \tag{2.11}$$

$$F(i,j) = \max \begin{cases} F(i,0), \\ F(i-1,j-1) + s(x_i, y_j), \\ F(i-1,j) - d, \\ F(i,j-1) - d. \end{cases} \tag{2.12}$$

方程 (2.11) 处理未匹配区域和匹配末端, 并只允许匹配终止分值至少为 T. 方程 (2.12) 处理匹配的开始和延伸. 向矩阵添加一个额外元素 $F(n+1,0)$, 再使用 (2.11) 式就可以要得到所有匹配的总分值. 我们从每个匹配中都减去 T 以获得这个分值; 如果没有高于分值 T 的匹配, 则这个总分值取 0, 该结果通过重复调用式 (2.11) 的第一选项而得到.

每个匹配联配均可通过从矩阵元 $(n+1,0)$ 回溯到 $(0,0)$ 而得到, 每一步都退回到 max() 操作中当前单元格的源单元格. 该回溯过程是一个全局的过程, 它显示了 x 的每个残基位点所联配上的符号. 结果全局联配将包含 x 子序列与 y 子序列间更常规的含空位局部联配部分.

注意, 该算法能一次得到所有的局部匹配. 它能找到计分最大的一组匹配, 也就

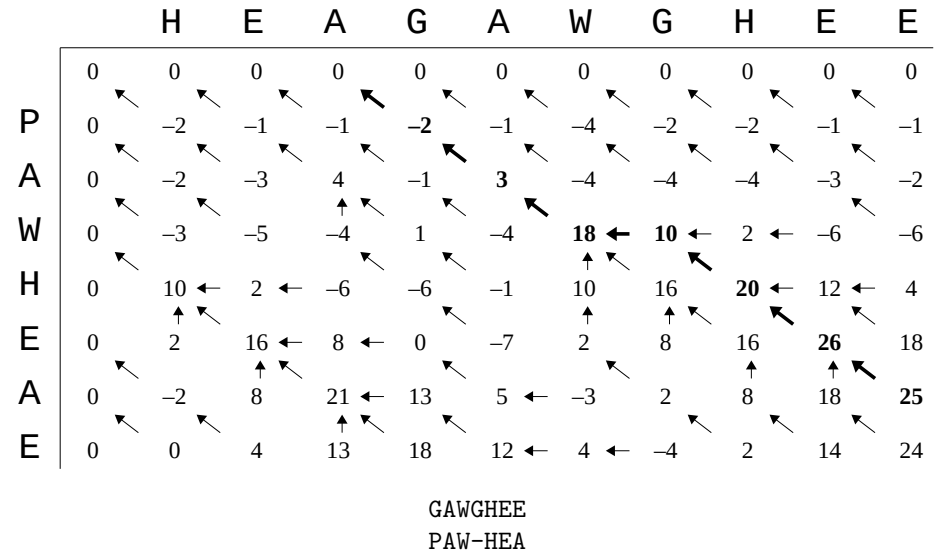
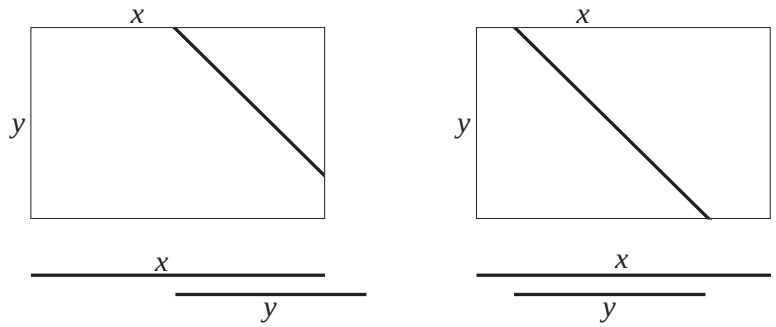


图 2.8: 上图: 示例序列的交叠动态规划矩阵. 下图: 最优交叠联配, 总分是 25.

是把每个匹配超出阈值 T 的部分求和并最大化. 改变 T 的大小将会改变算法所找出的结果. 增大 T 可能会排除某些匹配; 减小 T 则可能使某些匹配被分割成几段, 而且还会找到一些新的较差匹配. 从上一节的意义上讲, 如果局部最优的匹配段包含分值低于 $-T$ 的内部子联配, 那么它将被分割成若干小段. 然而, 这也许就是我们想要的: 例如有两个非常显著的相似高分片段, 它们被一段绝对值较大的负分值未匹配区域隔开, 这样我们就不清楚是给出一个整体匹配还是分别给出两段匹配更可取.

交叠匹配 (overlap match)

另一类搜索适用于下面的情形: 我们预期两条序列中的一条包含着另一条, 或者他们之间存在交叠. 这种情形通常发生在基因组 DNA 序列片段之间, 或它们与更长的染色体序列之间进行比较时. 下图列出了几种可能出现的情形:



其实我们想要的只是一种对两端多余部分不进行罚分的全局联配. 这为我们所使用的算法提供了线索: 让匹配起始于矩阵的第一行或左起第一列, 并结束于矩阵的最后一行或右起第一列. 因此初始化方程是: $F(i,0) = 0, i = 1,\ldots,n$ 和 $F(0,j) = 0, j = 1,\ldots,m$, 而矩阵中的递归关系就是全局联配中使用的式 (2.8). 令 $F_{\max}$ 为矩阵最后一行 $(i,m), i = 1,\ldots,n$ 和最右列 $(n,j), j = 1,\ldots,m$ 中的最大值. 回溯过程由最大值点开始, 直到第一行或左起第一列结束.

类似于 (2.11) 和 (2.12), 交叠匹配算法的重复 (repeat) 形式如下:

$$F(i,0) = \max \begin{cases} F(i-1,0), \\ F(i-1,m) - T; \end{cases} \tag{2.13}$$

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + s(x_i, y_j), \\ F(i-1,j) - d, \\ F(i,j-1) - d. \end{cases} \tag{2.14}$$

注意 (2.13) 中 $F(i,0)$ 的递归式不再像前一节 (2.11) 中那样需要遍历序列 y 中所有可能的子序列, 而是只需要查看到 $y_{1\ldots m}$ 为止的完整匹配. 然而, 我们仍然使用式 (2.11) 的初始形式计算 $F(n+1,0)$, 以得到 y 的初始子序列到 x 尾端的匹配.

混合匹配条件

现在应该清楚地认识到, 我们可以形式化地给出各种不同的动态规划变体. 所有此前介绍的联配方法都以矩阵 $F(i,j)$ 为表现形式, 只是边界条件和递归法则有所差别. 给定通用框架后, 我们就可以提出混合 (hybrid) 算法. 我们已经看到交叠算法重复形式的一个实例, 它还有很多可能的进一步变体.

例如当重复序列y容易出现在未被空位分割的串联拷贝 (tandom copy) 中时, 用下式替换 $j = 1$ 时的 (2.14) 式, 会很有用:

$$F(i,1) = \max \begin{cases} F(i-1,0) + s(x_i, y_1), \\ F(i-1,m) + s(x_i, y_1), \\ F(i-1,1) - d, \\ F(i,0) - d. \end{cases}$$

上式避免了式 (2.11) 的 $-T$ 罚分, 它只对每个串联的重复簇应用一次阈值, 而不是对每个重复都应用一次阈值.

另一个例子也许是这种情形: 我们想要找到的匹配始于两条序列的开始, 而可以终止于任何位置. 我们只需要设 $F(0,0) = 0$ 并用式 (2.8) 的递归公式, 然后让匹配终止于整个矩阵的最大值点就可以了.

实际上, 我们甚至还可以考虑混合边界条件, 例如这种情形: 人们认为在一条较长序列中出现某序列完整拷贝的先验概率是显著的, 但是也有可能只出现该序列一个片断. 此时我们可以在边界上设置罚分, 或为开始内部匹配设置罚分, 通过计算各自概率的对数得到罚分值. 这种模型适用于在基因组 DNA 序列中寻找某一重复序列家族的成员, 因为通常来讲它们是重复序列的完整拷贝, 但有时我们只能观测到某些重复序列的片断.

当进行序列相似性搜索时, 按理说应当总是考虑我们所要找的匹配类型, 继而选择对应这种匹配的最合适算法. 但实际上, 通常只有少数针对标准情形开发的可用执行程序, 因此通常来说为了方便起见, 我们先使用这些程序, 然后再对结果进行后续处理.

2.4 更复杂模型的动态规划

到目前为止, 我们仅仅考虑最简单的空位模型, 其中空位分值 $\gamma(g)$ 只是其长度倍数. 这种计分方案对生物序列而言并不理想: 延伸空位的罚分与出现第一个空位时相同, 然而当出现空位时, 它们的长度却往往不只一个残基. 假设已知 $\gamma(g)$ 的一般函数形式, 那么在对递归关系式作如下修正后, 我们就仍然可以使用 2.3 节中描述的所有动态规划算法形式:

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + s(x_i, y_j), & \\ F(k,j) + \gamma(i-k), & k = 0,\ldots,i-1, \\ F(i,k) + \gamma(j-k), & k = 0,\ldots,j-1. \end{cases} \tag{2.15}$$

这个式子代替了基本的全局动态关系式. 然而对于两条长度均为 n 的序列, 这一过程实施起来需要 $O(n^3)$ 步操作, 而不再是如线性空位罚分模型中的 $O(n^2)$, 因为对于每个矩阵元 (i, j) 我们都需要考虑前 $i + j + 1$ 个可能的指针, 而不再是先前策略中的仅仅三个指针. 这在多数情况下都会带来计算时间消耗上的惊人增长. 在某些条件下根据 $\gamma()$ 的属性, 关于 k 的搜索是有界的, 这样就可以使期望计算时间变回 $O(n^2)$, 尽管这些情况下比例常数会高一些 [Miller & Myers 1988].

使用仿射空位分值的联配

使用式 (2.15) 的标准方案之一是, 在式 (2.5) 中假设一种仿射空位罚分结构: $\gamma(g) = -d - (g-1)e$. 对于这种形式的空位罚分也存在一种 $O(n^2)$ 的动态规划方法. 对比先前只需要记录 $F(i, j)$ 一个值, 现在则要记录每一对残基 (i, j) 的多个值. 我们先用对应于图 2.4 中给出的三种不同情形的三个变量讲解这一过程, 为方便起见再把这三种情形列出来如下:

I G A x_i	A I G A x_i	G A x_i − −
L G V y_j	G V y_j − −	S L G V y_j

令 $M(i, j)$ 为 x_i 联配上 y_j 时 (上图左侧情形) (i, j) 元的最优得分, $I_x(i, j)$ 为 x_i 联配上一个空位时 (即对 y 而言是一个插入, 上图中间情形) 的最优得分, $I_y(i, j)$ 为 y_j 对 x 而言是一个插入时 (上图右侧情形) 的最优得分.

式 (2.15) 的递归关系式就变为:

$$
\begin{aligned}
M(i,j) &= \max \begin{cases} M(i-1,j-1) + s(x_i, y_j), \\ I_x(i-1,j-1) + s(x_i, y_j), \\ I_y(i-1,j-1) + s(x_i, y_j); \end{cases} \\
I_x(i,j) &= \max \begin{cases} M(i-1,j) - d, \\ I_x(i-1,j) - e; \end{cases} \\
I_y(i,j) &= \max \begin{cases} M(i,j-1) - d, \\ I_y(i,j-1) - e. \end{cases}
\end{aligned}
\tag{2.16}
$$

在这些方程中, 我们假设一个删除之后不会紧跟一个插入. 这在 $-d - e$ 小于最低失配 (mismatch) 分数时, 对于最优路径来讲是正确的. 像前面一样, 我们也可以通过回溯过程找到联配本身.

图 2.9 优雅地描述了方程 (2.16) 所定义的系统. 每种矩阵值都被显示为一个状态, 用箭头表示状态间的转移. 每个状态转移都伴随着一个分数增量, 每个状态都指定一个 $\Delta(i, j)$, 它用于决定下标为 i, j 时进入该状态的改变. 用于更新这些矩阵值的递归关系可以直接由图表得到 (比较图 2.9 和方程 (2.16)). 在矩阵 (i, j) 元上每个状态变量的新值就是达到这个状态的转移所对应的最大分值. 每个状态转移的分值, 都是目标状态的由 $\Delta(i, j)$ 指定的偏移量处源状态的值, 加上指定的分值增量得到. 此类描述对应于计算机科学中的**有限状态自动机** (finite state automaton, FSA) 问题. 一个联配就是一条由状态节点连接起来的路径 (path), 来自序列的字符根据 $\Delta(i, j)$ 在各个状态中的取值对应到联配中. 图 2.10 给出了一组短联配的实例及相应的仿射空位模型的状态路径.

实际上仿射空位罚分算法的常用实现只包含两个状态 M 和 I, 其中 I 表示落入空位区的情况. 从技术上讲, 这样做只能在最低失配分值不低于 $-2e$ 时确保给出正确的结果. 然而即使存在低于 $-2e$ 的失配罚分, 得到不同联配的机会也很小. 况且即便出现了不同的结果, 问题也不严重, 因为联配的不同之处一般都会落在匹配得很

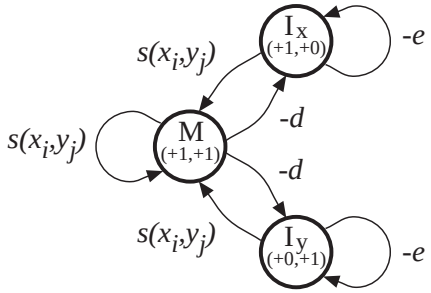


图 2.9: 仿射空位罚分联配中使用的三种状态间的相互关系图表.

差的空位区域. 这种形式的递归关系式为:

$$M(i,j) = \max \begin{cases} M(i-1,j-1) + s(x_i, y_j), \\ I(i-1,j-1) + s(x_i, y_j); \end{cases}$$

$$I(i,j) = \max \begin{cases} M(i,j-1) - d, \\ I(i,j-1) - e, \\ M(i-1,j) - d, \\ I(i-1,j) - e. \end{cases}$$

因为状态 I 可以是 $\Delta(1,0)$ 或 $\Delta(0,1)$, 所以这些方程并不存在与之对应的上述 FSA 图. 但是却存在另一种 FSA 形式化系统, 它将 $\Delta(i,j)$ 值与转移而不是状态关联起来. 这种自动机为删除和插入使用额外的转移, 可以解释二状态仿射空位算法. 实际上, 线性空位罚分的标准单状态算法可以表述为单状态转移发射 FSA (single-state transition emitting FSA), 它的三种转移对应于不同的 $\Delta(i,j)$ 值 ($\Delta(1,1)$、$\Delta(1,0)$ 和 $\Delta(0,1)$). 对致力于这方面研究的读者来讲, 计算机科学文献中存在一种更简单的基于状态的自动机称为 Moore 机 (Moore machines), 而转移发射系统则称为 Mealy 机 (Mealy machine) (参见第 9 章).

更复杂的 FSA 模型

用 FSA 描述动态规划算法的一个优势在于, 我们可以容易看出如何生成新的算法. 图 2.11 给出了一个实例, 显示了带两个匹配状态的四状态 FSA. 其背后的思想是, 可能存在与状态 A 相对应的无空位高保真 (high fidelity) 区域, 它们被与状态 B 相对应的含空位低保真 (low fidelity) 区域以及空位状态 I_x 和 I_y 隔开. 替换分值 $s(a,b)$ 和 $t(a,b)$ 可用来反映不同区域内的预期相似性. 与之类似, FSA 算法也可用于构建跨膜蛋白的联配, 用不同的匹配状态分别对应胞内区域、胞外区域或跨膜区域, 或其他更复杂情形所对应的区域 [Birney & Durbin 1997]. Searls & Murphy

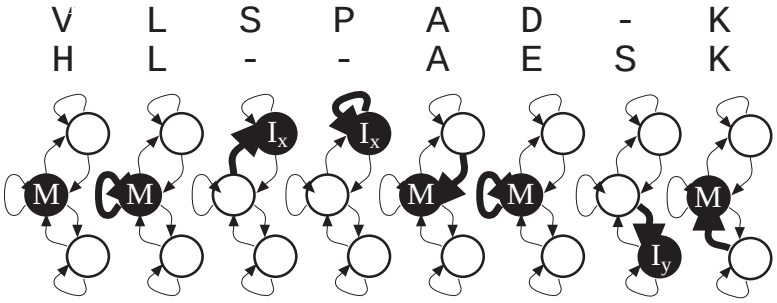


图 2.10: 使用仿射空位模型的联配状态分配实例.

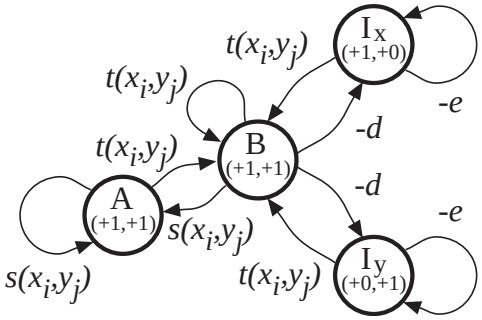


图 2.11: 四状态有限状态自动机, 其中匹配状态 A 和 B 分别对应于高保真和低保真区域. 值得注意的是, 此处 FSA 以罚分 $s(x_i, y_j)$ 和 $t(x_i, y_j)$ 在转移上发射, 而不在状态上发射, 前文已经讨论了这个差别.

[1995]为这类 FSA 给出了更加抽象的定义, 并开发了用于构建这类 FSA 的交互式工具.

这些更加复杂算法的一个特点是: 给定联配路径, 我们也就为原始序列中的字符赋予了隐含的标签, 用于表明匹配这些字符的状态. 以跨膜蛋白的匹配模型为例, 在找到最优联配的同时, 联配也就为每条蛋白指定了跨膜区、胞内区和胞外区. 在多数情况下, 这种序列标签可能和联配信息本身同样重要.

我们将在第 4 章重新回到二序列联配的状态模型这一问题上.

练习

2.10 计算图 2.10 中联配的计分, 其中 $d = 12, e = 2$.

2.5 启发式联配算法

到目前为止, 所有我们考虑的联配算法都是“正确的”, 这一判断的依据就是它们都保证能按照指定的计分方案找到最优分值. 特别地, 上一节中描述的仿射空位联配算法被广泛认为是现存最敏感的序列匹配方法. 但它们却不是最快的序列联配方法, 而在很多时候速度的确是一个问题. 到现在为止, 我们所描述的动态规划算法的时间复杂度都是 $O(nm)$, 即序列长度的乘积. 当前的蛋白数据库包含了上亿 (100 million) 量级的残基, 因此对长度为一千的序列, 要搜索整个数据库就必须计算大约 10^{11} 个矩阵元. 在撰写本书的年代, 正常情况下单独一台工作站每秒可计算千万个 (10 million) 矩阵元, 那么刚才这个搜索任务就需要消耗 10^4 秒, 约三小时. 而当我们要对许多不同序列进行搜索时, 时间消耗就立刻成为一个重要的问题.

出于这个原因, 人们做了多种尝试以设计出比直接动态规划更快的算法. 这些方法的目标是, 在考虑所有高分联配的同时, 搜索动态规划矩阵单元中尽可能小的部分. 如果序列十分相似, 那么就有很多方法被证实可以找到最优匹配, 这些方法都基于计算机科学领域中精确匹配字串搜索算法到非精确匹配情形的扩展 [Chang & Lawler 1990; Wu & Manber 1992; Myers 1994]. 但是对用于寻找疏远匹配的计分矩阵而言, 这些精确方法就变得难以胜任, 所以我们必须采用牺牲敏感性的启发式 (heuristic) 方法, 有时会遗漏掉最好计分联配. 存在很多可用的启发式搜索技术. 在这里我们简要描述两种最有名的算法: BLAST 和 FASTA , 以阐述不同方法和其中的取舍. 关于启发式联配算法的细节分析已经超出了本书的范畴.

BLAST

BLAST 软件包 [Altschul *et al.* 1990]提供了一个在查询序列 (query sequence) 和目标数据库 (target database) 之间寻找高分局部联配的程序, 二者皆可为 DNA 或蛋白质. BLAST 算法背后的思想是: 联配中的真实匹配很有可能包含一小段等同

位点或分值极高的短连续片段. 因此我们可以一开始就寻找这样的短连续片段并利用它们作“种子” (seed), 向外延伸以找到好的更长联配. 在维持种子片断足够短的条件下, 我们可以先对查询序列进行预处理来为所有可能的种子和它们对应的起始位置建立一个对照表.

BLAST 为所有定长 (默认情况下, 蛋白质序列为 3 个残基, 核酸序列为 11 个残基) 的“相邻字串” 建立一张表, 这些字串都以高于某个阈值的分数匹配上查询序列的某处, 典型的阈值设定为每残基 2 比特. 然后它搜索整个数据库, 一旦找到该表中的字串就开始“靶延伸” (hit-extension) 过程, 在两个方向上以无空位联配方式扩展可能的匹配, 直到延伸到最大分值 (实际上, 如果按照这样处理, 偶尔可能会在比真实最大扩展稍短的位置终止).

BLAST 通常仅用于寻找无空位联配. 可能很令人吃惊的是, 这种无空位联配的限制只丢掉了一小部分显著匹配, 部分原因是不相关序列的最优期望得分降低使得部分无空位分值仍然显著, 还有部分原因是由于对每对序列来讲, BLAST 可以找到并报告多个高分匹配, 并可为组合分值给出显著性值 [Karlin & Altschul 1993]. 不过新版 BLAST 最近出炉了, 它能给出含空位的联配 [Altschul & Gish 1996; Altschul *et al.* 1997].

FASTA

另一个广泛使用的启发式序列搜索软件包是 FASTA [Pearson & Lipman 1988]. 它使用多步 (multistep) 方法寻找局部的高分联配, 从精确短串匹配开始, 通过最大计分的无空位延伸最终确定含空位联配.

第一步使用查找表定位两序列间所有长度为 $ktup$ 的等同字 (word) 串. 对蛋白序列, $ktup$ 通常取值 1 或 2, 对 DNA 序列则取值 4 或 6. 然后查找带许多互为支持的字匹配的对角元. 这步操作运行很快, 例如可以通过对在不同下标 $(i-j)$ 处的匹配进行排序来实现.

第二步继续寻找最优对角元, 这一步类似于 BLAST 算法中的靶延伸过程, 即扩展精确匹配字串以找到最大计分无空位区域 (此过程中还可以将若干种子的匹配组合到一起).

接着, 第三步查看在允许空位罚分的条件下, 是否能够将这些无空位联配区域用空位区域连接起来. 最后一步, 对在整个数据库中搜索得到的最高计分候选匹配用完全动态规划算法进行重联配, 但将其限制在动态规划矩阵的一个子区域中, 围绕候选启发式匹配形成一个带状区域.

因为 FASTA 的最后一步使用标准的动态规划, 所以它处理联配分数的方式和本章前面介绍的那些算法完全一致. 在选择参数 $ktup$ 时, 我们需要在速度和敏感性间进行折衷: $ktup$ 取值越高速度越快, 但也更容易遗漏掉真实的显著匹配. 对蛋白序列而言, 要想接近完全局部动态规划的敏感性, 有必要设定 $ktup = 1$.

2.6 线性空间联配

除了时间因素, 另一个限制动态规划算法的计算资源是内存占用量. 目前为止我们所讲述的所有算法都在计算一个如 $F(i, j)$ 的分值矩阵, 这个矩阵的大小是 nm, 也就是序列长度的乘积. 对两条典型的几百残基长的蛋白序列, 其消耗的内存量还在现代台式电脑的能力范围之内; 但如果其中至少有一条是长度达到几十或几百 kb 的 DNA 序列, 那么整个矩阵所需要的内存量就会超出一台机器的物理内存了. 幸运的是, 内存方面的情形比速度方面要好一些: 我们已经有技术可以用有限的内存得到最优联配, 内存的占用是 $n + m$ 阶而不再是 nm 阶了, 而且所用时间也不会超过原来的两倍. 这通常被称为线性空间 (linear space) 法. 这一方法使用了二序列动态规划方法的重要基本技巧.

实际上, 如果只需要最大分值, 问题就简单了. 因为递归式对于 $F(i, j)$ 是局部的, 仅依赖于上一行的一个条目, 所以我们完全可以只保留当前计算点的上一行, 而将上边的其他诸行释放掉. 如果是寻找局部联配, 我们需要在整个矩阵中寻找最大分值, 但是在构建矩阵的过程中记住最大分值是很容易的. 然而这只能让我们得到

分值, 而无法得到联配; 如果我们释放掉那些行以避免 $O(nm)$ 阶的存储量, 我们也就失去了用来回溯的指针. 必须采用一种新方法获得联配.

假设我们正在使用线性空位计分方案寻找一个最优全局联配. 这一方法很容易扩展到其他类型的联配上. 我们使用**分治法** (divide and conquer) 策略.

令 $u = \lfloor \frac{n}{2} \rfloor$, 即 $\frac{n}{2}$ 的整数部分. 我们暂时假设可以确定一个 v 使得 (u, v) 元位于最优联配上, 即第 v 行就是最优联配经过矩阵第 $i = u$ 列时所在的那一行. 那么我们就可以将动态规划问题分为两个部分, 一个是从左上角的 $(0, 0)$ 到 (u, v), 另一个是从 (u, v) 到 (n, m). 整个矩阵的最优联配将是这两个独立的子矩阵最优联配串联起来的结果. (为了使这一步得到准确的结果, 我们定义这里的联配不含原点.) 一旦将联配拆分一次, 我们就可以通过对每一区域的再拆分而递归地构建出整个联配, 其中每步都多固定下来一对联配上的残基. 这一过程的终止条件可以定义为要联配的序列长度为 0, 这是比较直接的并意味着整个区域完全被指定了; 也可以定义为当序列足够短时终止, 这样就可以使用标准的 $O(n^2)$ 联配算法和回溯方法.

那么如何才能找到v呢? 当 $i \geq u$ 时, 我们定义一个 $c(i, j)$ 使 $(u, c(i, j))$ 是从 $(1, 1)$ 到 (i, j) 的最优路径. 我们可以在计算 $F(i, j)$ 的同时更新 $c(i, j)$. 如果 (i', j') 是导出 $F(i, j)$ 的单元 (i, j) 的上一个单元, 那么 $i = u$ 时令 $c(i, j) = j$, 否则令 $c(i, j) = c(i', j')$. 很明显这是一个局部的操作, 我们只需要保留 $c()$ 的上一行就可以了, 像我们只需要保留 $F()$ 的上一行一样. 现在我们就可以从矩阵最后的单元中得到我们想要的值: $v = c(n, m)$.

据我们所知, 上边描述的这种找到 v 的过程尚未被任何一个使用它的人在文章中发表. 另一个更有名的过程首先出现于计算机科学领域 [Hirschberg 1975], 而后由 Miller & Myers [1988] 引入计算生物学, 因此在序列分析领域一般被称为 Myers-Miller 算法. Myers-Miller 算法并不传递回溯指针 $c(i, j)$, 而是通过合并第 u 行处动态规划算法路径的向前和向后结果寻找联配中点 (u, v) (具体细节请见他们的论文). Myers-Miller 算法是优雅的递归算法, 但是具体解释起来有一点困难. Waterman [1995, p .211] 给出了第三种线性空间方法. Chao, Hardison & Miller [1994] 则对二序列联配中的线性空间算法做了综述.

练习

2.11 在算法的初始回合, 令 $u = 5$, 为图 2.5 中示例序列对的全局联配填充正确的 $c(i, j)$ 值.

2.12 证明线性空间算法所需时间仅约为标准 $O(nm)$ 算法所需时间的两倍.

2.7 分值的显著性

既然我们知道了如何找到最优联配, 那么如何评估分值的显著性呢? 也就是说, 如何判别该联配是一个能给出同源性证据的、有生物意义的联配, 或只是两条完全无关序列的最优联配?存在着两种可能的方法: 基于不同模型比较的 Bayes 方法, 以及另一种基于传统统计学的方法. 后者先假设一个零模型, 即这里的两条序列不相关, 然后再计算匹配分值高于观测值的机会.

Bayes 方法: 模型比较

在动机不是十分明确的情况下, 我们曾经给出对数几率比并将其作为所考虑的分值, 见原书第 11 页. 也许我们会坚持说真正想要的是两条序列相关之于无关的概率值 $P(M|x, y)$, 而不是上边算出的似然 $P(x, y|M)$. 一旦再给出某些假设, $P(M|x, y)$ 就可以使用 Bayes 法则计算出来. 首先我们指定两个模型的先验 (*a priori*) 概率. 它们反映了在实际看到这两条序列之前, 我们期望它们是相关的. 记序列相关的先验概率为 $P(M)$, 它表示匹配模型正确的概率, $P(R) = 1 - P(M)$ 表示随机模型正确的先验概率. 那么一旦我们看到实际数据, 匹配模型正确即这些序列

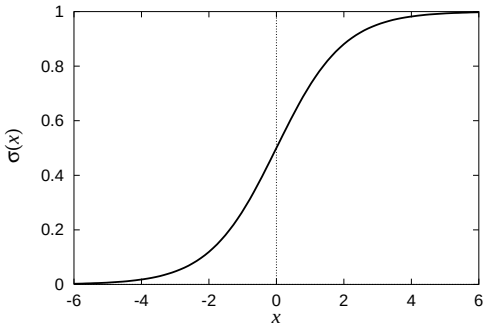


图 2.12: Logistic 函数.

相关的后验概率为:

$$\begin{aligned} P(M|x,y) &= \frac{P(x,y|M)P(M)}{P(x,y)} \\ &= \frac{P(x,y|M)P(M)}{P(x,y|M)P(M)+P(x,y|R)P(R)} \\ &= \frac{P(x,y|M)P(M)/P(x,y|R)P(R)}{1+P(x,y|M)P(M)/P(x,y|R)P(R)}. \end{aligned}$$

令

$$S' = S + \log\left(\frac{P(M)}{P(R)}\right), \tag{2.17}$$

其中

$$S = \log\left(\frac{P(x,y|M)}{P(x,y|R)}\right)$$

是联配的对数几率分值. 那么

$$P(M|x,y) = \sigma(S'),$$

其中

$$\sigma(x) = \frac{\mathrm{e}^x}{1+\mathrm{e}^x}.$$

$\sigma(x)$ 称为 logistic 函数. 它是一条 S 形的函数, 当 x 趋近于正无穷时其值趋近于 1, 当 x 趋近于负无穷时趋近于 0, 并且在 $x=0$ 点的值为 $\frac{1}{2}$ (见图 2.12). Logistic 函数被广泛应用于神经网络分析 (neural network analysis) 中, 它把由累加而得到的分数转化为概率——这不完全是一种巧合.

从式 (2.17) 中我们看到, 标准的联配分值应该再加上一个先验对数几率比 $\log\left(\frac{P(M)}{P(R)}\right)$. 这正好直观地对应于似然比与先验几率比的乘积. 一旦这步完成后, 原则上我们就可以将得到的结果与 0 作比较以揭示序列是否相关. 为了使这一步有效实施, 应该注意到实际使用的表达式都是概率, 特别是当我们将所有已经得到的可能序列对的概率值累加到一起时, 要保证其和为 1. 然而当用特别的方式构建计分方案时, 这个结论就不一定对了.

当我们在大量不同的联配中寻找一个可能的显著匹配时, 先验几率比变得重要起来. 在对数据库进行搜索时, 这是一种典型的情形. 很明显, 如果有固定的先验几率比, 那么即使数据库中所有序列都无关, 随着需要匹配序列数目的增加, 随机情况下其中一条匹配看起来很显著的概率也会增大. 实际上在固定先验几率比的条件下, (假) 显著观测的期望数目呈线性增长. 如果我们想要使这个值固定, 那么就必须令先验几率比与数据库中的序列数目 N 成反比. 这样做的效果是, 为了维持一个固定的假阳性值, 我们要将 S 与 $\log N$ 而不再是0进行比较. 一种保守的策略是采用对应

于假阳性期望值的分数, 例如 0.1 或者 0.01. 当然这种方法并不一定特别合适. 例如我们可以确信, 在所有的蛋白中有 1% 的激酶 (kinase), 此时先验几率应该为 $\frac{1}{100}$, 可以预料到当考虑更多的序列时, 虽然假阳性会增加, 但是真阳性也一样会增加. 另一方面, 如果我们确信在整个数据库中只有一个显著的匹配, 那么与 $\log N$ 进行比较就更合理.

在这一点上, 我们可以转而考虑由局部匹配算法得到的计分统计显著性. 这里需要根据实际情况作一些修正, 我们要考虑的只是两条序列的子序列间, 众多可能的不同局部匹配中最好的那个. 局部匹配起始位点数目的一个简单估计是序列长度的乘积 nm. 如果全部匹配都是定长的而且所有的起始位点都给出相互独立的匹配, 那么我们就需要比较最优分 S 与 $\log(nm)$. 然而这两个假设很明显都是错误的 (例如, 沿着对角线的连续位点上的匹配片断就不独立), 因此我们就需要再对 S 加上一个的小修正因子, 它仅依赖于计分函数 s, 而不依赖于 n 和 m. 这个影响还不存在对应的解析理论, 而对于计分系统, 特别是在比较蛋白序列时所使用的计分系统, 看起来这一倍数 (multiplicative) 因子定在 0.1 左右比较合适. 因为我们所关心的是这一因子的对数加法项, 所以它的影响就相当小了.

经典方法: 极值分布

在这种情形下, 还有另一种在更经典统计框架下研究显著性的方法. 我们可以观察独立随机序列的 N 个最大匹配分值的分布. 如果这个最大分值高于观测到的最优分的概率足够小, 那么这一观测就被认为是显著的.

在固定无空位联配 (2.2) 的简单情形下, 与一条随机序列间的匹配分值是许多相似的随机变量之和, 所以它就可以用正态分布作为逼近. 一系列 N 个相互独立的正态随机变量之最大值 M_N 的渐近分布是已知的, 并且形如:

$$P(M_N \leq x) \simeq \exp(-KN\mathrm{e}^{\lambda(x-\mu)}), \tag{2.18}$$

其中 K 和 λ 为常数. 这种极限分布 (limiting distribution) 称为**极值分布** (extreme value distribution, EVD) (参见第 11 章). 我们可以用方程 (2.18) 计算搜索大量 (N 条) 无关序列所得最高分大于我们所观测到最高分 S 的概率. 如果这个概率低于某一个很小的数, 例如 0.05 或 0.01, 那么我们便可以得出结论: 能产生观测到的最大分值的序列不大可能是无关的, 即它可能是相关的.

经证实, 即使个别分值不服从正态分布, 极值分布仍然是大量单独分值最大值的正确极限分布 (见第 11 章). 由此, 同类显著性检验方法可用于从等概率大集合中寻找最优分值的任意搜索方法. 实际上对于来自局部联配算法的最优局部匹配分值而言, 两条 (相当长的) 序列间的最优分值本身都会服从极值分布, 因为在这种情形下我们可以有效地比较单个矩阵中 $O(nm)$ 个不同随机起始位置所对应的结果.

对于局部无空位联配, 通过使用 Dembo & Karlin [1991] 的更加完备的结果, Karlin & Altschul [1990] 解析地推导出恰当的 EVD 分布. 我们分两步给出这一方法. 第一步, 分数高于 S 的无关匹配数近似服从泊松 (Poisson) 分布, 其均值为

$$E(S) = Kmn\mathrm{e}^{-\lambda S}, \tag{2.19}$$

其中 λ 是

$$\sum_{a,b} q_a q_b \mathrm{e}^{\lambda s(a,b)} = 1 \tag{2.20}$$

的正根而 K 是一个常数, 它来自一个也仅依赖于 q_a 和 $s(a,b)$的 几何收敛级数. 这个 K 直接对应于上节末尾中我们所描述的倍数因子; 它对匹配之可能起始点的非独立性进行修正. λ 实际上是一个尺度 (scale) 参数, 它用来将 $s(a,b)$ 转化为自然尺度. 请注意, 如果最开始用方程 (2.3) 推导出对数似然 $s(a,b)$, 那么 $\lambda = 1$, 因为$\mathrm{e}^{\lambda s(a,b)} = p_{ab}/q_a q_b$.

于是, 存在一个得分高于 S 的匹配之概率为:

$$P(x > S) = 1 - \mathrm{e}^{-E(S)}, \tag{2.21}$$

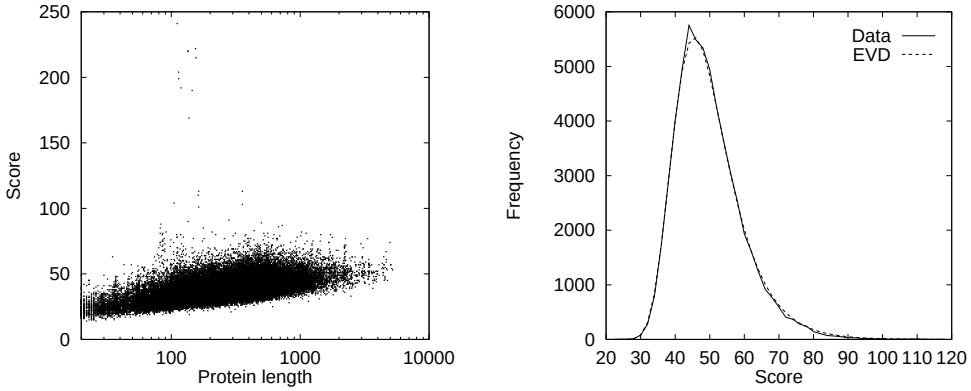


图 2.13: 左图: 人类细胞色素 C (SWISS-PROT 检索号 P000001) 与 SWISS-PROT34 蛋白质数据库的局部匹配分值之散点分布图, 此结果使用实现 Smith-Waterman 算法的 SSEARCH 程序 Pearson96 获得. 右图: 经长度归一化后的相应分值分布, 以及拟合的 EVD 分布.

很容易发现将方程 (2.19) 和方程 (2.21) 联立起来就可以得出与方程 (2.18) 形式相同但却不带 μ 的 EVD . 实际上, 通常不需要纠缠于概率的计算, 而只须要求 $E(S)$ 显著小于 1. 这可以转化为对某个固定的常数 T, 要求:

$$S > T + \frac{\log mn}{\lambda}. \tag{2.22}$$

这对应于上一节的 Bayes 分析, 它建议我们应该比较 S 和 $\log mn$, 但在这里, 我们可以对所使用的 T 值赋予精确的含义.

虽然人们还没有推导出含空位联配情形下与此相应的解析理论, 但 Mott [1992] 建议, 对随机序列的含空位联配计分仍然效仿无空位计分的极值分布形式, 并且现在已经有大量的经验证据支持这一理论. Altschul & Gish [1996] 使用大量随机生成的抽样数据, 为许多标准蛋白联配计分方案拟合了方程 (2.19) 中的 λ 和 K 值.

对长度的修正

在搜索序列长度参差不齐的数据库时, 即使所有序列相互间都无关, 较长数据库序列的最优局部匹配也比较短序列的最优局部匹配更容易获得高分. 图 2.13 就给出了一个例子. 这并不奇怪: 如果我们的搜索序列长度为 n, 数据库序列长度为 m_i, 那么在 nm_i 的矩阵中较大的 m_i 就对应更多的可能起始点. 但如果我们的先验预期是以等概率匹配到序列库中的任意序列, 那么就需要使随机匹配得分具有独立于长度的可比性.

一种在理论上对长度依赖性的合理修正是, 应该将每条数据库序列的最优分值减去 $\log(m_i)$. 该结论来自于上一节中 S' 的表达式. 而另一种看起来应用效果稍好, 并且当要搜索大量序列时也很容易实现的方法是: 将数据库中的序列按照其长度进行分组 (bin), 而后给出序列长度对数的一个线性拟合函数 [Pearson 1995] (这需要一些技巧才能实现从“背景”中分离出信号).

为什么使用联配分值作为检验的统计量?

到目前为止, 本节中我们一直假设, 作为联配显著性检验统计量而使用的联配分值, 与在搜索阶段中用于寻找最优匹配的相同. 先使用一种标准搜索匹配而后再利用另一种不相关的标准评估它, 这看起来可能很有吸引力. 在检验过程中, 这似乎防止了搜索阶段所引起的背景水平提高的问题. 然而我们需要搜索与显著性检测都具备尽可能强的辨别能力. 因此使用对此二者而言都是最优的可行统计量就很重要了. 如果在搜索阶段漏掉一个真正的相关联配, 那么显然我们就不会对它做显著性检验.

在搜索中使用检验统计量的后果是, 无关序列间的最优匹配从定性的角度看起来像真实的匹配. Karlin & Altschul [1990] 给出了一个突出的例子: 当找到随机序列间的最优局部无空位联配时, 在这些联配中残基 a 联配上残基 b 的观测频率为 $q_a q_b e^{\lambda s(a,b)}$, 即正好是我们期望在真实的进化匹配模型中观测到 a 联配上 b 的频率 p_{ab}. 我们能够用来区分真伪匹配的唯一属性就是分数的数量级, 它的期望正比于匹配长度.

当然, 在最敏感的计分方案中可能包含复杂的计算, 但实际上这些计算无法在搜索阶段实现. 此时, 恐怕必须使用更简单的分值进行搜索, 但同时需要保留一些候补的高分联配, 而不只是简单地留下最好的一个. 第4章将给出得到这些次优联配的方法.

2.8 从联配数据推导计分参数

本章最后, 我们回到首节中所讨论的主题: 如何确定计分模型的组成部分, 即替换分值和空位分值. 当时我们描述了如何由概率出发为二序列联配算法推导分值. 但这却留下了一个悬而未决的问题, 即如何估计概率. 要知道整个联配系统的性能依赖于这些参数的取值, 所以对它们的估计也就引起了相当的关注.

一种简单而明了的方法就是在已证实的联配中为联配上的残基对频率和空位频率进行计数, 并设概率 p_{ab}、q_a 和 $f(g)$ 为归一化后的频率. (这就相当于得到了概率的最大似然估计; 见第11章.) 使用这种简单的方法也存在两个难题. 第一难题是要获得已证实联配中一个好的随机抽样. 因为蛋白序列以家族形式出现, 所以它们间的联配倾向于相互不独立. 第二个难题更加微妙. 实际上, 不同序列对间的分化程度不同. 当两条序列不久前才由共同祖先分化而来时, 我们期望它们的多数残基都是等同的. 于是在 $a \neq b$ 时概率 p_{ab} 应该很小, 因此 $s(a,b)$ 应该是一个绝对值很大的负数, 除非 $a = b$. 而在另一个极端, 如果两条序列在很久以前便已经分化, 那么我们便期望 p_{ab} 趋向于背景频率 $q_a q_b$, 所以对所有的 a, b 而言, $s(a,b)$ 都应该接近于零. 这就意味着我们应该使用与要比较序列的期望分化程度相应的分值.

Dayhoff PAM 矩阵

Dayhoff, Schwartz & Orcutt [1978] 在定义他们的 PAM 矩阵时已经考虑了这两个难题, 该矩阵现在已广泛用于实际的蛋白序列联配. 他们方法的基础就是在考虑蛋白家族进化关系的同时, 找到非常相似的蛋白序列间联配的替换数据, 而后将这一信息拓展到更长的进化距离上.

开始时, 他们先构建了关联到 71 个家族的序列假想系统发育树, 其中每对序列间的差异不超过它们残基总数的 15%. 他们用简约法 (见第 7 章) 建树, 该方法给出一个残基列表, 它们分别是在每条祖先序列中每个位点上最有可能出现的那种残基. 由此可以得到一个包含序列与其树上最近祖先之间的所有成对残基 a 和 b 间的频率数组 A_{ab}. 这一配对过程的进化方向被忽略了: 无论是祖先序列中的残基 a 被后代序列中的 b 替换, 还是刚好相反的情形, A_{ab} 和 A_{ba} 每次都会增加. 基于树的计数方法避免了随进化相关性而来的替换过计数 (overcounting) 问题.

因为需要拓展到更长的时间跨度, 所以对于他们来讲需要估计的主要值并不是看到 a 联配上 b 的联合概率 p_{ab}, 而是残基 a 在 t 时刻被残基 b 替换的条件概率 $P(b|a,t)$. $P(b|a,t) = p_{ab}(t)/q_a$. 我们可以按照以下介绍的方法, 通过相乘短时间区间的条件概率得到长时间区间的条件概率. 这些条件概率称为**替换概率** (substitution probabilities); 它们在构建系统发育树时扮演重要部分 (见第 8 章). $P(b|a)$ 的短时间间隔估计可以由 A_{ab} 通过下式导出: $P(b|a) = B_{a,b} = A_{ab}/\sum_c A_{ac}$.

接下来这些值必须根据分化时间 t 作相应修正. 在残基 a 之出现频率为 q_a 的“典型”蛋白中, 替换的期望频率为 $\sum_{a \neq b} q_a q_b B_{ab}$. Dayhoff *et al.*定义, 如果替换次数的期望值为 1%, 即 $\sum_{a,b} q_a q_b B_{ab} = 0.01$, 那么就称此时的替换矩阵为 1 PAM 矩阵 (“已接受点突变” (point accepted mutation) 的首字头缩写). 为了将 B 矩阵转化为替换概率的1 PAM 矩阵, 他们利用因子 σ 按比例缩放了非对角元, 并调整对角元, 以保持每一行的和等于 1. 更精确地说, 当 $a \neq b$ 时他们定义 $C_{ab} = \sigma B_{ab}$, 且

$C_{aa} = \sigma B_{aa} + (1 - \sigma)$, 其中的 σ 使 C 成为一个 1 PAM 矩阵; 我们用 $S(1)$ 来表示 1 PAM C. 它的每一元素可以被看作单位时间内用 b 替换 a 的概率 $P(b|a, t = 1)$.

为了生成适合较长时间段的替换矩阵, 将 $S(1)$ 自乘 n 次得到它的 n 次幂, 记 $S(n) = S(1)^n$. 例如, $S(2)$ 是矩阵 $S(1)$ 与自己的乘积, 其中每一元素 $P(a|b, t = 2) = \sum_c P(a|c, t = 1)P(c|b, t = 1)$ 是 b 通过某一中间媒介 c 被 a 替换的概率. n 不太大时, 非对角元随 n 近似线性增长. 另一种理解它的方式为: 矩阵 $S(n)$ 表示 20 态 Markov 链经过 n 步后的结果, 其中这 20 态分别对应于 20 个氨基酸, 每一步的转移概率由 $S(1)$ 给出 (我们将在第3章全面讲解 Markov 链).

最后从 $S(t)$ 中得到一个分数矩阵. 已知 $P(b|a) = p_{ab}/q_a$, 则 t 时刻分数矩阵的元素为:

$$s(a, b|t) = \log \frac{P(b|a, t)}{q_b}.$$

为了方便计算, 这些值已乘上因子并四舍五入为整数. 使用最广泛的矩阵为 PAM250, 它已经乘上了因子 3/log2, 即分值的单位是三分之一比特.

BLOSUM 矩阵

Dayhoff 矩阵集已经成为序列比较技术的支柱之一, 但它们也存在局限性. $S(1)$ 的矩阵元通常源自短时间区间的替换, 而将 $S(1)$ 自乘为更高次幂, 以 PAM250 矩阵为例, 并没有获得短时间替换与长时间替换之间的真正区别 [Gonnet, Cohen & Benner 1992]. 对于前者, 三联密码子 (codon triplet) 中单碱基改变引起的氨基酸替换起主导作用, 如 `L` ↔ `I`、`L` ↔ `V` 或 `Y` ↔ `F`, 而后者则显示了所有类型的密码子变化.

自从 PAM 矩阵出现以来, 已经形成许多包含距离更远相关蛋白多序列联配的数据库, 它们可以更直接用于推导计分矩阵. 人们广泛使用的一种计分矩阵集是 BLOSUM 矩阵集 [Henikoff & Henikoff 1992]. 详细来说, 它们来自一组蛋白质家族中联配上的无空位区域, 这些蛋白家族均源于 BLOCKS 数据库 [Henikoff & Henikoff 1991]. 先对每个区块 (block) 中的序列进行聚类, 当两条序列间的等同残基百分比超过L%时, 就将这两条序列归为一类. Henikoff & Henikoff 然后计算了观测到一类中残基 a 联配上另一类中残基 b 的频数 A_{ab}, 并且为了修正聚类容量, 他们对每种组合都赋予权重 $1/(n_1 n_2)$, 其中 n_1 和 n_2 为类的容量.

他们从 A_{ab} 出发估计了 q_a 和 p_{ab}: $q_a = \sum_b A_{ab} / \sum_{cd} A_{cd}$ 即包含一个 a 的配对所占比例, 而 $p_{ab} = A_{ab} / \sum_{cd} A_{cd}$ 即 a, b 配对占所有观测到配对的比例. 这样他们就从标准方程 $s(a, b) = \log P_{ab}/q_a q_b$ (2.3) 计算出了计分矩阵的所有元素. 和前面一样, 所得对数几率分值矩阵被乘上因子并四舍五入为整数. 特别地, 在 $L = 62$ 和 $L = 50$ 条件下得到的两个矩阵被广泛用于二序列联配和数据库搜索, BLOSUM62 已经成为无空位匹配的标准, 而 BLOSUM50 则可能更适用于含空位联配 [Pearson 1996]. BLOSUM62 已经被乘上因子, 以便其元素值以半比特为单位, 即对数几率值都乘上了 2/log2, 而 BLOSUM50 的单位是三分之一比特. 注意越低的 L 值对应于越长的进化时间, 并适应于越远距离的搜索.

估计空位罚分

然而时间依赖 (time-dependent) 的含空位模型却不存在类似的标准集. 如果存在时间依赖的空位分值模型, 那么一个合理的假设是空位个数的期望会随时间线性增长, 而它们的长度分布保持不变. 在仿射空位模型中, 这就对应于使空位开端的罚分 d 与 log$t$ 呈线性关系, 而保持空位延伸罚分 e 不变. Gonnet, Cohen & Benner [1992] 由经验数据得到了一个类似的分布. 实际上, 他们提出了一个更好的拟合: $\gamma(g) = A + B\log t + C\log g$, 虽然在其方法中存在着某种循环论证, 因为他们使用序列联配算法从蛋白数据库与自身的完全比较结果中获得数据.

实际中, 一旦选好了替换分值, 人们就会凭经验选取空位罚分. 这可能是因为仿射空位模型仅有两个参数, 而蛋白替换矩阵有210个参数. 关于在选择空位罚分时需要考虑的影响因素, [Vingron & Waterman 1994] 给出了详细的讨论.

将替换模型和空位模型组合好以后, 我们还需要考虑最后一个问题. 既然在一条序列的某一给定位置有可能出现空位, 那么就不一定存在匹配. 这表明我们应该

在替换分值中包含不出现空位的概率项. 序列 x 中特定位置上出现空位的概率为 $\sum_{i\geq 1} f(i)$, 而与之类似, 序列 y 中该位置上出现空位的概率也一样. 于是我们得到不存在空位 (no gap), 即存在匹配的概率:

$$P(\text{no gap}) = 1 - 2\sum_{i\geq 1} f(i). \tag{2.23}$$

由此, 对应于匹配的替换分值不应该是 $s(a,b)$, 而应该是 $s'(a,b) = s(a,b) + \log P(\text{no gap})$. 这就使得更容易出现空位, 即空位罚分减少时, 序列对的分值降低. 然而这一修正的效果不大, 因此在从联配频率导出计分系统时一般也不会使用.

2.9 补充读物

Pearson [1996] 和 Pearson & Miller [1992] 都对生物序列比较中的动态规划方法做出了很好综述. Pearson [1995] 和 Shpaer *et al.* [1996] 给出了动态规划方法的敏感性评估以及此方法与快速启发式方法 BLAST 和 FASTA 的对比.

Bucher & Hofmann [1996] 描述了 Smith-Waterman 算法的概率论形式, 这与我们将在第 4 章讲解的内容有关.

在二序列动态规划联配领域中我们还未涉及到的有趣论题包括: 快速“捆绑” (banded) 动态规划算法 [Chao, Pearson & Miller 1992], 蛋白查询序列与 DNA 目标序列间的联配问题 [Huang & Zhang 1996], 还有不仅恢复最优、而且恢复“次优”或“近似最优” (near-optimal) 联配的问题 [Zuker 1991; Vingron 1996].

第三章

Markov 链与隐马模型 (HMM)

第 2 章已经介绍了二序列联配的方法, 本章重点将转向与单条序列相关的问题. 本章的主要目标是, 为符号序列的普适概率论模型建立一套理论, 我们称其为隐马模型 (hidden Markov model, 缩写为 HMM). 可用 HMM 及其简化变种 Markov 模型回答的问题种类有: “这条序列属于某个特定的家族吗?”或者“假设这条序列来自某个家族, 我们可以对它的内部结构做出怎样的推断?”识别蛋白序列中的 α 螺旋或 β 折叠区域, 就属于上述的第二类问题.

在给出生物序列例子的同时, 我们也将以更一般的形式对 HMM 的许多运算给出其数学和算法. 这些方法, 或与之相近的策略将用于本书的许多其他章节中. 因此本章包含了大量的数学专业内容. 我们对本章的安排是, 在前半段中用一个单独的生物学例子带领读者概览基本算法. 稍后, 我们将介绍其他例子以阐明基本方法的更复杂扩展.

在下一章中, 我们将看到 HMM 也可以应用于第 2 章讨论的不同类型的联配问题, 在第 5 章中它们将用于从数据库中搜寻蛋白家族, 在第 6 章中用于同时联配多条序列. 实际上, 在搜索和联配中的应用可能构成了 HMM 在生物序列分析中最著名的应用. 但为了强调 HMM 远超出序列联配范畴的更广阔适用性, 我们在这里以更通用的语言描述 HMM 理论.

20 世纪 70 年代初, HMM 首先用于语音识别 (speech recognition) 领域, 关于 HMM 的绝大多数文献均属于这个领域. 对 HMM 的最好、最全面介绍之一是综述 Rabiner [1989], 这篇文章也介绍了该主题的发展历史. 虽然本章与这篇综述在内容上有些重复, 但侧重点上会有重要的区别.

介绍 HMM 在生物序列分析中的应用之前, 我们先大致了解一下如何使用它们进行语音识别 [Rabiner & Juang 1993]. 录制完毕后, 一个语音信号将分为多个 10-20 毫秒长的片段 (称为帧, frame). 预处理后, 通过所谓的向量量子化 (vector quantisation) 将每帧分配到预先设定的众多类别之一. 典型的类别数目是 256 个. 于是语音信号就表示为一条由类别标签组成的长序列, 语音识别器需要由此找出真实的音素 (或单词) 序列. 问题在于真实的人类声音富于变化, 并且单词各部分的发音时间在不同语境中也不同.

在生物序列分析中, 许多问题具有与此相同的结构: 基于某字母表的符号序列, 找出它所代表的信息. 例如蛋白质序列的组成符号来自于包含 20 种氨基酸的字母表, 并且我们通常想知道一条给定序列属于哪个蛋白家族. 在这里, 氨基酸一级序列类似于语音信号, 蛋白家族类似于它所表示的发音单词. 而语音信号的时间变化则相当于蛋白序列中出现插入和删除.

让我们回到一个更简单的例子, 先用它引入不含隐状态的标准 Markov 模型, 然后再用它引入简单的隐马模型.

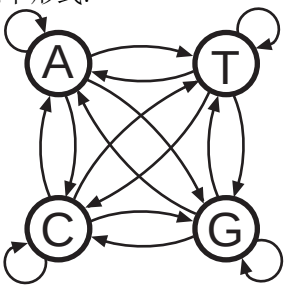
例子: CpG **岛** 在人类基因组中凡是出现双核苷酸序列 `CG` (经常写为 `CpG` 以便区分双链间的 `C-G` 互补碱基对), `C` 核苷酸 (胞嘧啶, cytosine) 在化学上一般会被甲基化 (methylation) 修饰. 这种甲基化的 `C` 转化为 `T` 的几率较高, 导致了通常情况下 `CpG` 双核苷酸序列在基因组中的出现频率要少于从 `C` 和 `G` 的独立概率出发所得到的

期望. 由于某些重要的生物学原因, 在诸如基因启动子 (promoter) 或“起始”区的短基因组片段附近, 这种甲基化过程被抑制. 于是在这些区域我们会看到比其他地方更多的 CpG 双核苷酸序列, 而事实上这些区域中 C 和 G 的总体含量也更多. 这样的区域称为 CpG 岛 (CpG island) [Bird 1987]. 它们的长度通常是几百到几千个碱基.

我们将考虑两个问题: 第一, 给出一条短基因组序列, 我们将如何判断它是否来自 CpG 岛? 第二, 给出一条长序列, 如果它含有 CpG 岛, 我们将如何发现? 让我们从第一个问题开始. □

3.1 Markov 链

我们该对 CpG 岛区域使用哪种概率论模型呢? 我们知道双核苷酸是重要的, 因此需要一个生成序列的模型, 使得一个符号的概率依赖于它前面的一个符号. 满足该条件的最简单模型就是一条经典的 Markov 链. 我们以图的形式把 Markov 链显示为“状态”的集合以及状态间的箭头, 其中每个状态表示一种特定的残基. DNA 的一条 Markov 链可以画成如下形式:



其中我们看到的每个状态都与 DNA 字母表里 A、C、G、T 中的一个相对应. 图中的每个箭头都与一个概率参数相关联, 它决定了一个特定残基跟随另一个残基的概率, 或一个状态跟随另一个状态的概率. 这些概率参数称为**转移概率** (transition probability), 用 a_{st} 表示:

$$a_{st} = P(x_i = t | x_{i-1} = s). \tag{3.1}$$

对于序列的任何概率论模型, 通过多次使用 $P(X,Y) = P(X|Y)P(Y)$ 我们可以把序列的概率写成

$$\begin{aligned} P(x) &= P(x_L, x_{L-1}, \ldots, x_1) \\ &= P(x_L|x_{L-1}, \ldots, x_1)P(x_{L-1}|x_{L-2}, \ldots, x_1) \cdots P(x_1). \end{aligned}$$

Markov 链的关键性质是每个 x_i 出现的概率仅依赖于前一个符号 x_{i-1} 的值, 而不是它前面的整条序列, 即 $P(x_i|x_{i-1}, \ldots, x_1) = P(x_i|x_{i-1}) = a_{x_{i-1}x_i}$. 于是前面的方程就变为

$$\begin{aligned} P(x) &= P(x_L|x_{L-1})P(x_{L-1}|x_{L-2}) \cdots P(x_2|x_1)P(x_1) \\ &= P(x_1) \prod_{i=2}^{L} a_{x_{i-1}x_i}. \end{aligned} \tag{3.2}$$

虽然我们从 DNA 序列的 CpG 岛中推导出这个方程, 但事实上它是描述来自任意 Markov 链的特定序列概率的普适方程. 关于 Markov 链的著作有很多, 例如 Cox & Miller [1965].

练习

3.1 长度为 L 的所有可能序列的概率和可写为 (使用 (3.2) 式)

$$\sum_{\{x\}} P(x) = \sum_{x_1} \sum_{x_1} \cdots \sum_{x_L} P(x_1) \prod_{i=2}^{L} a_{x_{i-1}x_i}.$$

证明: 上式等于 1.

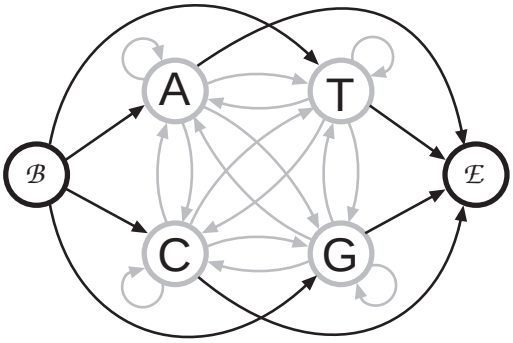


图 3.1: 为了给序列的两个端点建模, 可以向 Markov 链 (灰色模型) 添加起始和结束状态.

为起始和结束序列建模

注意, 像规定转移概率一样, 我们也应给出在特定状态起始的概率 $P(x_1)$. 为了避免 (3.2) 式中由起始概率引起的不一致性, 可以在模型中加入一个额外的起始状态 (begin state). 同时, 我们在字母表中加入一个字母 $\mathscr{B}$. 通过定义 $x_0 = \mathscr{B}$, 序列的起始也被加入到式 (3.2) 中, 这样序列中第一个字母的概率是

$$P(x_1 = s) = a_{\mathscr{B}s}.$$

类似地, 为了把结尾也纳入模型, 我们可以在序列结尾添加符号 $\mathscr{E}$. 这样以残基 t 结尾的概率是

$$P(\mathscr{E}|x_L = t) = a_{t\mathscr{E}}.$$

为了与新符号相一致, 我们在 DNA 模型中加入起始和结束状态 (end state) (见图 3.1). 事实上并不需要在字母表中明确地添加字母, 我们仅需将这两个新的状态作为“沉默” (silent) 状态以便标识起始点和结束点即可.

传统上, Markov 链并不对序列结尾建模; 我们假设序列可以结束于任何位置. 明确地加入结束状态是为了对序列的长度分布建模. 这样, 该模型就对所有可能的序列 (任意长度) 定义了概率分布. 长度分布呈指数衰减; 见下面的练习.

练习

3.2 假设模型有结束状态, 并且从任何状态到结束状态的转移概率都是 τ. 证明所有长度为 L (恰好转移到结束状态而终止) 的序列概率 (3.2) 之和是 $\tau(1-\tau)_{L-1}$.

3.3 证明所有可能的任何长度的序列的概率和是 1. 这一点证实了 Markov 链确实能够描绘整个序列空间的正确概率分布. (提示: $0 < x < 1$时, $\sum_{i=0}^{\infty} x^i = 1/(1-x)$.)

使用 Markov 链作判别

方程 (3.2) 的首要用途是计算似然比检验所需的值. 这里我们以 CpG 岛的真实数据为例阐述该用途. 我们从一组人类 DNA 序列中提取总共 48 个推测的 CpG 岛并建立 2 个 Markov 链模型, 其中一个针对标记为 CpG 岛的区域 (“+”模型), 而另一个则针对序列的其余部分 (“−”模型). 每个模型的转移概率按下面的方程设定

$$a_{st}^{+} = \frac{c_{st}^{+}}{\sum_{t'} c_{st'}^{+}}, \tag{3.3}$$

对于 a_{st}^{-} 也类似, 其中 c_{st}^{+} 是在标记为 CpG 的区域中字母 t 出现在字母 s 后面的次数. 这就是转移概率的最大似然 (ML) 估计值, 第 1 章中已经有所描述.

(这个例子含有将近 60000 个核苷酸, ML 估计已经足够了. 如果每种类型的计数较小, 那么使用 Bayes 估计过程或许更合适, 这正如第 11 章以及下面 HMM 部分所讨论的一样.) 结果如下表

+	A	C	G	T
A	0.180	0.274	0.426	0.120
C	0.171	0.368	0.274	0.188
G	0.161	0.339	0.375	0.125
T	0.079	0.355	0.384	0.182

−	A	C	G	T
A	0.300	0.205	0.285	0.210
C	0.322	0.298	0.078	0.302
G	0.248	0.246	0.298	0.208
T	0.177	0.239	0.292	0.292

其中每个表格第一行的 4 个数字分别代表 4 种碱基在 A 之后出现的频率, 其他行同理, 因此每一行之和都为 1. 这些数字是不一样的; 例如, G 在 A 后出现比 T 在 A 后出现更普遍. 同时请注意, 这两个表格是不对称的. 这两个表格中 G 出现在 C 之后的概率都低于 C 出现在 G 之后的概率, 而正如我们所料, 在 "−" 表中这种现象更为明显.

为了使用这些模型做判别, 我们计算对数几率比 (log-odds ratio)

$$S(x) = \log \frac{P(x|\text{model}+)}{P(x|\text{model}-)} = \sum_{i=1}^{L} \log \frac{a^{+}_{x_{i-1}x_i}}{a^{-}_{x_{i-1}x_i}}$$

$$= \sum_{i=1}^{L} \beta_{x_{i-1}x_i}$$

其中 x 表示序列, $\beta_{x_{i-1}x_i}$ 是相应转移概率的对数似然比. 下面按比特[1]给出一个 β 的表格:

β	A	C	G	T
A	−0.740	0.419	0.580	−0.803
C	−0.913	0.302	1.812	−0.685
G	−0.624	0.461	0.331	−0.730
T	−1.169	0.573	0.393	−0.679

图 3.2 显示了经长度归一化后的分值 $S(x)$ 分布, 即每个分子的平均比特数. 如果不做长度归一化, 该分布将更加分散.

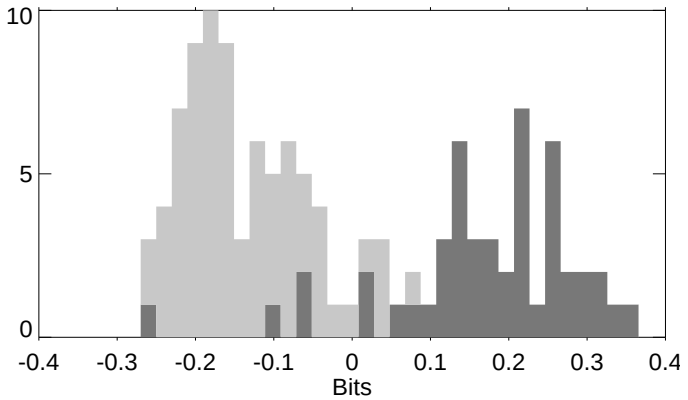


图 3.2: 所有序列经长度归一化后分值的直方图. CpG 岛用深灰色表示, 非 CpG 部分用浅灰色表示.

我们可以看到此方法在标记为 CpG 岛的区域和其他区域间做出了一个合理的判别. 是否做长度归一化 (normalisation) 对这种判别影响不大. 如果我们想进一步探究错误分类的情况, 那么请记住错误既可能来自于不充足或参数化不当的模型, 也可能来自训练数据的错误标记.

[1]使用以 2 为底的对数, 这样的单位称为比特. 见第 11 章. ——原注

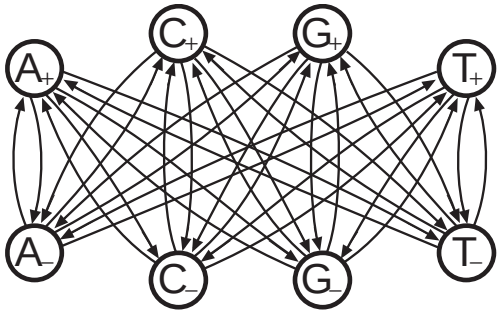


图 3.3: CpG 岛的 HMM. 除了图中画出的转移外, 像前面的简单 Markov 链一样, 每个集合 (“+”或“−”) 内部都包含一整套转移.

3.2 隐马模型

有很多经典 Markov 链的扩展形式, 我们会在本章后面的部分再做阐述. 这里, 我们将直接讨论隐马模型. 可以通过最初对 CpG 岛提出的第二个问题引出这个模型: 如何在一条没有注释的长序列中找到 CpG 岛? 可以使用刚刚建立的 Markov 链模型解决这个问题, 我们可以对序列中每个核苷酸周围 (例如长度为 100bp) 的窗口计算对数几率分值然后画到图上. 我们预计 CpG 岛部分将显现正值. 然而如果我们认为实际上 CpG 岛有明确的边界且长度不一, 那么这样的判别方法就不十分令人满意了. 为什么窗口大小为 100? 更令人满意的方法是为整条序列建立整合了两个 Markov 链的单一模型.

为了用一个模型模拟在基因组序列的非岛屿“海洋”之间出现“岛屿”, 我们需要将上一段中出现的两个 Markov 链展现在一个模型中, 并且设置两条链间每个转移点处发生转移的概率很小. 但是, 这样一来每个核苷酸符号就要对应两个状态. 我们重新标记状态以解决这个问题. 现在用 A_+、C_+、G_+ 和 T_+ 表示在 CpG 岛区域分别发射 `A`、`C`、`G` 和 `T`, 而用 A_-、C_-、G_- 和 T_- 表示在非岛屿区域的发射; 见图 3.3.

应该设置该模型中的转移概率使得每个集合内部的转移概率接近于原始子模型, 但是在两个子模型之间保持非常小但有限的转移概率. 整体上讲, 由“+”转换为“−”的几率大于相反的转换, 因此如果使模型自由运转, 那么它会在“−”的无岛屿状态停留更长时间.

重新标记是关键步骤. Markov 链和隐马模型的主要区别在于, 在隐马模型中不存在状态和符号间的一一对应. 在生成 x_i 时, 我们已经不可能仅通过观察 x_i 就判断出模型处于哪个状态. 在该例子中这就表现为, 我们无法仅通过一个符号 `C` 就区分出它是由状态 C_+ 还是 C_- 发射的.

HMM 的正式定义

现在让我们规范地描述隐马模型, 并且导出特定状态和符号序列的概率. 现在需要区分状态序列和符号序列. 我们将状态序列称为*路径* (path), 记为 π. 路径本身就是一条简单的 Markov 链, 因此每一状态的出现概率仅依赖于它的前一状态. 路径中第 i 个状态称为 π_i. 状态链的特性由下面的参数描述

$$a_{kl} = P(\pi_i = l | \pi_{i-1} = k). \tag{3.4}$$

就像前面的 Markov 链中为序列起始建模一样 (图 3.1), 为了给该过程的起始建模, 我们引入一个起始状态. 从这个起始状态到状态 k 的转移概率 a_{0k} 可以认为是由状态 k 开始的概率. 同样和前面类似, 我们通过总是将状态序列转移到一个结束状态来为序列结尾建模. 为了方便起见, 我们将起始和结束状态都标记为 0 (这样做不会发生冲突, 因为只能由起始状态转移出来, 也只能转移进入结束状态, 所以这些变量至多只会使用一次).

由于我们已经将符号 b 与状态 k 相互分离, 因此必须为模型引入一组新的参数 $e_k(b)$. 在 CpG 岛模型中, 每个状态只与唯一的符号相关联, 但这并不是必须的; 通常来讲, 一个状态可以从所有可能符号之分布中生成符号. 因此我们定义

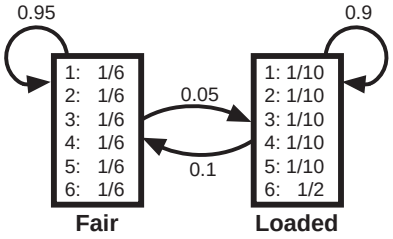
$$e_k(b) = P(x_i = b|\pi_i = k), \tag{3.5}$$

为处于 k 状态中时我们看到符号 b 的概率. 它们称为发射 (emission) 概率.

在 CpG 岛模型中, 发射概率都是 0 或者 1. 为了描述发射概率, 我们在这里再次使用第 1 章中赌场的例子.

例子: 偶尔作弊的赌场, 第一部分

让我们来看第 1 章中的例子. 在一个赌场里, 他们大部分情况下使用公平的骰子, 但有时换成作弊的骰子. 这种作弊骰子得到 6 的概率为 0.5, 而得到数字 1 到 5 的概率均为 0.1. 假设赌场在每次掷骰子之前将公平骰子换成作弊骰子的概率为 0.05, 而将骰子换回来的概率为 0.1. 这样两种骰子之间交换的过程就是一个 Markov 过程. 在这个 Markov 过程的每个状态中, 骰子掷出的结果有不同的概率, 这样整个过程就构成一个隐马模型. 我们可以把它画成下图的形式:



其中发射概率 $e()$ 标示在状态框里. □

上面的模型中隐藏的是什么呢? 如果只看到骰子掷出的结果序列 (观测序列), 那么你将无法得知哪几轮是由作弊骰子掷出的而哪些是由公平骰子掷出的, 因为赌场会对此秘而不宣; 也就是说, 状态序列是隐藏的. 而在一条 Markov 链 中, 你总能明确地知道某个特定观测结果所属的状态. 显然, 赌场不会说出他们使用了作弊骰子, 并告诉你数字出现的不同概率. 但是在这种更加复杂的情况下, 在上面的 HMM 中估计概率是可能的 (一旦你怀疑他们使用了两种不同的骰子), 对此我们会在后面做进一步讨论.

所谓发射概率源于人们容易把 HMM 看作是产生或发射序列的生成模型 . 例如我们可以用上面的公平/作弊骰子模型, 通过模拟连续地选择骰子并掷出骰子的过程, 来生成一条掷骰子结果的随机序列. 更一般地, 可以按如下方式由 HMM 生成序列: 首先, 以 a_{0i} 选取状态 π_1. 在该状态下, 按照该状态的分布 e_{π_1} 发射出一个观测值. 然后, 以转移概率 $a_{\pi_1 i}$ 选出新状态 π_2, 以此类推. 这样一条随机的人造观测序列就产生了. 因此, 我们有时会说 $P(x)$ 是 x 由此模型生成 (generate) 的概率.

现在可以容易地写出一条观测序列 x 和一条状态序列 π 的联合概率:

$$P(x, \pi) = a_{0\pi_1} \prod_{i=1}^{L} e_{\pi_i}(x_i) a_{\pi_i \pi_{i+1}}, \tag{3.6}$$

其中我们要求 $\pi_{L+1} = 0$. 例如在我们的模型中, 由状态序列 (C_+, G_-, C_-, G_+) 发射序列 CGCG 的概率为

$$a_{0,C_+} \times 1 \times a_{C_+,G_-} \times 1 \times a_{G_-,C_-} \times 1 \times a_{C_-,G_+} \times 1 \times a_{G_+,0}.$$

方程 (3.6) 相当于方程 (3.2) 的 HMM 形式. 然而, 由于通常我们不知道路径, 它在实际中的用处并不大. 在后面的章节中我们将介绍如何通过找到似然最大的解, 或通过使用状态的后验分布, 来估计路径. 然后我们再进一步介绍如何为 HMM 估计参数.

最可能的状态路径: Viterbi 算法

虽然通过观察相应的符号已经无法再判断系统处于哪个状态, 但是通常我们感兴趣的是潜在的状态序列. 通过考虑潜在状态而找到观测序列的“含义”, 在语音识别中的术语称为*解码* (decoding). 有几种不同的解码方法. 这里我们将介绍最常见的一种, 称为 Viterbi 算法. 它是一种动态规划算法, 与第 2 章中出现的算法紧密联系.

一般情况下, 任意一条特定的符号序列都可以由许多不同的状态序列生成. 例如在我们的 CpG 模型中, 状态序列 (C_+, G_+, C_+, G_+), (C_-, G_-, C_-, G_-) 和 (C_+, G_-, C_+, G_-) 都可以生成符号序列 `CGCG`. 然而, 它们这几个事件的发生概率却不同. 第三个的概率是在不同区域间来回转移的小概率之乘积, 因此其概率远小于前两个概率. 第二个概率显著小于第一个概率, 因为第二条状态序列包含两个从 `C` 到 `G` 的转移, 而这种转移在“$-$”区域发生的概率明显小于在“$+$”区域发生的概率. 所以在这三可能性中, 序列 `CGCG` 最可能来自于一连串“$+$”状态.

一条贯穿 HMM 的预测路径将告诉我们序列中的哪一部分会被预测为 CpG 岛, 因为此前我们假设每个状态都被指定为对 CpG 岛或其他区域建模. 如果要选择其中的一条路径作为预测结果, 也许我们该选择概率最大的, 即

$$\pi^* = \operatorname*{argmax}_{\pi} P(x, \pi). \tag{3.7}$$

最可能的路径 π^* 可以通过递归方法找到. 假设对所有状态 k 我们都已知以状态 k 及观测符号 i 结束的最可能路径的概率 $v_k(i)$. 那么可以用下面的公式计算对应于观测符号 x_{i+1} 的概率

$$v_l(i+1) = e_l(x_{i+1}) \max_k (v_k(i) a_{kl}). \tag{3.8}$$

所有序列必须以状态 0 (起始状态) 为开始, 因此初始条件为 $v_0(0) = 1$. 通过使指针不断后移, 我们可以用回溯的方法找到实际的状态序列. 其完整算法是:

算法: Viterbi
初始($i = 0$): $v_0(0) = 1, k > 0$时$v_k(0) = 0$.
递归($i = 1 \dots L$): $v_l(i) = e_l(x_i) \max_k(v_k(i-1)a_{kl})$;
$\mathrm{ptr}_i(l) = \operatorname{argmax}_k(v_k(i-1)a_{kl})$.
终止: $P(x, \pi^*) = \max_k(v_k(L)a_{k0})$;
$\pi^*_L = \operatorname{argmax}_k(v_k(L)a_{k0})$.
回溯($i = L \dots 1$): $\pi^*_{i-1} = \mathrm{ptr}_i(\pi^*_i)$. ◁

注意, 我们假设存在结束状态, 这就是在结束步骤使用 a_{k0} 的原因. 如果不对结束建模, 那么这个 a 就可以省略.

实现 Viterbi 算法以及我们后面将要介绍的算法时, 都存在一些值得注意的问题. 最严重的实际问题是多个概率之乘积往往是很小的数字, 从而在任何计算机上都会造成下溢 (underflow) 错误. 因此, 我们应该总是在对数空间中使用 Viterbi 算法,也就是计算 $\log(v_l(i))$, 这样乘积变为求和使得数字保持合适的大小. 这部分将在 3.6 节中讨论.

图 3.4 显示了序列 `CGCG` 和 CpG 岛模型所对应的所有 v 值的表格. 当我们把相同的算法应用于更长的序列时, 得到的最优路径 π^* 将在模型的“$+$”、“$-$”部分间切换, 从而给出所预测 CpG 岛区域的确切边界.

例子: 偶尔作弊的赌场, 第二部分

对于一条掷出骰子点数的序列, 我们现在可以找到贯穿原书第 38 页上模型的最可能路径. 我们从前面描述的模型生成了 300 个随机投掷的结果. 每次掷出的点数均来自公平骰子 (F) 或作弊骰子 (L), 如图 3.5 中数字下面的一行所示. Viterbi 算法用于预测状态序列, 即每个掷出点数来自哪个骰子. 通常情况下, 就像我们所看到的, Viterbi 算法很好地复原了真实状态序列. □

练习

v		C	G	C	G
$\mathscr{B}$	1	0	0	0	0
A_+	0	0	0	0	0
C_+	0	**0.13**	0	**0.012**	0
G_+	0	0	**0.034**	0	**0.0032**
T_+	0	0	0	0	0
A_-	0	0	0	0	0
C_-	0	0.13	0	0.0026	0
G_-	0	0	0.010	0	0.00021
T_-	0	0	0	0	0

图 3.4: 图 3.3 中描述的 CpG 岛模型和序列 CGCG 所对应 v 的结果表格. 最可能的路径用黑体标示.

```
Rolls   315116246446644245311321631164152133625144543631656626566666
Die     FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLL
Viterbi FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLL

Rolls   651166453132651245636664631636663162326455236266666625151631
Die     LLLLLLFFFFFFFFFFFFLLLLLLLLLLLLLLLLFFFLLLLLLLLLLLLLLFFFFFFFFF
Viterbi LLLLLLFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLFFFFFFFF

Rolls   222555441666566563564324364131513465146353411126414626253356
Die     FFFFFFFFLLLLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLL
Viterbi FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFL

Rolls   366163666466232534413661661163252562462255265252266435353336
Die     LLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Viterbi LLLLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

Rolls   233121625364414432335163243633665562466662632666612355245242
Die     FFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLLFFFFFFFFFFF
Viterbi FFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLFFFFFFFFFFF
```

图 3.5: 图中数字显示了例子中所描述的骰子投掷 300 次的结果. 下面一行显示了这些数字事实上是用哪个骰子掷出的 (F 表示公平骰子, L 表示作弊骰子). 再下面一行是 Viterbi 算法的预测结果.

3.4 证明 $\pi^* = \operatorname{argmax}_\pi P(\pi|x)$ 与 (3.7) 式等价.

前向算法

对于 Markov 链, 我们使用方程 (3.2) 计算一条序列的概率 $P(x)$. 其结果值可用于区分 CpG 岛和 DNA 的其他区域. 我们希望对 HMM 也能计算这样的概率. 由于许多不同状态路径均可以生成同一条序列 x, 我们必须将所有可能路径的概率相加, 从而得到 x 的全概率

$$P(x) = \sum_\pi P(x, \pi). \tag{3.9}$$

因为可能路径 π 的数量随序列长度的增加呈指数增长, 所以通过穷举所有路径强行计算 (3.9) 式是不可行的. 有一种方法是用上一节中的算法得到最可能的路径 π^*, 然后将其代入方程 (3.6) 以近似 $P(x)$. 这样做隐含假设了 π^* 是具有显著概率的唯一路径, 然而这个有些惊人的假设却在很多情况下都惊人地合理. 事实上没必要做近似, 因为全概率本身就可以用一种与 Viterbi 算法相似的动态规划过程计算得到, 只需要将取最大值的步骤替换为求和. 这种算法称为**前向** (forward) 算法.

前向算法中与 Viterbi 变量 $v_k(i)$ 对应的量是

$$f_k(i) = P(x_1 \dots x_i, \pi_i = k), \tag{3.10}$$

这是 $\pi_i = k$, 且观测序列结束于 x_i 的概率. 递归方程为

$$f_l(i+1) = e_l(x_{i+1}) \sum_k f_k(i) a_{kl}. \tag{3.11}$$

整个算法是:

算法: 前向算法
初始 ($i = 0$): $f_0(0) = 1$, $k > 0$ 时 $f_k(0) = 0$.
递归 ($i = 1 \dots L$): $f_l(i) = e_l(x_i) \sum_k f_k(i-1) a_{kl}$.
终止: $P(x) = \sum_k f_k(L) a_{k0}$. ◁

像 Viterbi 算法一样, 前向算法 (以及下一节中将要介绍的后向算法) 在计算机上实现时会出现下溢错误. 同样, 在对数空间上计算可以解决这个问题, 但是效果没有在 Viterbi 算法中那么好. 或者我们也可以选择使用缩放法 (scaling method). 这两种方法都将在 3.6 节中描述.

除了用于前向算法以外, $f_k(i)$ 还有很多其他用处, 包括将在后面两节中将要介绍的用法.

后向算法和后验状态概率

Viterbi 算法可以找出贯穿模型的最有可能路径, 但是正如我们当时所述, 它并不总是对序列进行进一步推断的最合适基础. 例如, 我们可能想要知道某观测 x_i 的最可能状态. 更一般地, 我们也许想得到已知观测序列时, 符号 x_i 来自于状态 k 的概率, 即 $P(\pi_i = k|x)$. 这就是发射序列已知的情况下, 状态 k 在时间 i 处的后验概率.

我们用稍微间接的方法得到后验概率. 我们首先计算状态 k 生成第 i 个符号时, 产生整条观测序列的概率:

$$\begin{aligned} P(x, \pi_i = k) &= P(x_1 \dots x_i, \pi_i = k) P(x_{i+1} \dots x_L | x_1 \dots x_i, \pi_i = k) \\ &= P(x_1 \dots x_i, \pi_i = k) P(x_{i+1} \dots x_L | \pi_i = k), \end{aligned} \tag{3.12}$$

上式的第二个等号成立是因为 k 以后的所有观测序列都仅依赖于 k 处的状态. 其中的第一个因子可以视为 (3.10) 中的 $f_k(i)$, 它可以由前一节中的前向算法得到. 第二个因子称为 $b_k(i)$,

$$b_k(i) = P(x_{i+1} \dots x_L | \pi_i = k). \tag{3.13}$$

它与前向变量相似, 但却通过从序列的结尾处开始反向递归计算得到:

算法: 后向算法
初始 ($i = L$): 对所有 k 令 $b_k(L) = a_{k0}$.
递归 ($i = L-1, \dots, 1$): $b_k(i) = \sum_l a_{kl} e_l(x_{i+1}) b_l(i+1)$.
终止: $P(x) = \sum_l a_{0l} e_l(x_1) b_l(1)$. ◁

我们很少用到终止步骤, 因为 $P(x)$ 通常可以从前向算法算出, 只是为了算法完整才在这里写出它.

现在可以将方程 (3.12) 写为 $P(x, \pi_i = k) = f_k(i) b_k(i)$, 由此我们就可以通过条件化直接得到所求的后验概率:

$$P(\pi_i = k|x) = \frac{f_k(i) b_k(i)}{P(x)}, \tag{3.14}$$

其中 $P(x)$ 可以由前向 (或后向) 算法计算得到.

例子: 偶尔作弊的赌场, 第三部分

图 3.6 为图 3.5 中的骰子点数序列画出了骰子公平的后验概率. 注意, 在某些地方后验概率并没有反应出骰子的实际种类. 只是因为一些误导性的点数序列可能随机出现, 所以这种结果也在预料之中. □

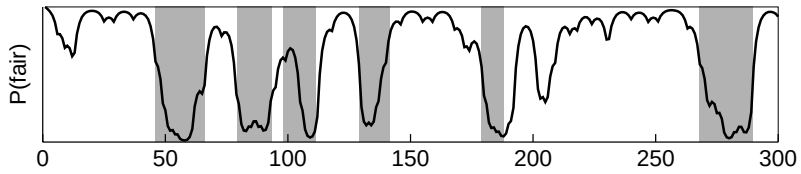


图 3.6: 在赌场的例子中, 处于公平骰子状态的后验概率. x 轴显示了掷骰子的次数. 阴影部分显示了哪些次是用作弊骰子掷出的.

后验解码

除了我们在前面章节中介绍的 Viterbi 解码以外, $P(\pi_i = k|x)$ 主要用于两种其他形式的解码方法. 当许多不同的路径与最可能的那条有几乎相同的概率时, 这些形式便显得尤为有用, 因为此时只考虑最可能的路径并不合理.

第一种方法是定义状态序列 $\hat{\pi}_i$ 以替代 π_i^*,

$$\hat{\pi}_i = \underset{k}{\operatorname{argmax}}\, P(\pi_i = k|x). \tag{3.15}$$

从其定义可知, 当我们关心特定点 i 的状态而不是整条路径时, 该状态序列也许更合适. 事实上, 由 $\hat{\pi}_i$ 所定义的状态序列的可能性也许并不像贯穿整个模型的路径那么大; 通常来讲有些转换会被禁止, 此时这条路径根本就不合格.

当我们关心的并不是状态序列本身, 而是由它引出的一些其他特性时, 第二种或许是更重要的新解码方法便会产生. 假设我们有定义在状态上的函数 $g(k)$. 于是我们将很自然地想求

$$G(i|x) = \sum_k P(\pi_i = k|x)g(k). \tag{3.16}$$

一个重要的特殊情况是, $g(k)$ 在某个状态子集上取值为 1, 而在其他状态上取 0. 此时 $G(i|x)$ 就是符号 i 来自于特定集合中的状态的后验概率. 以 CpG 岛模型为例, 我们关心的是一个碱基是否位于 CpG 岛中. 为此当$k \in \{\mathrm{A}_+, \mathrm{C}_+, \mathrm{G}_+, \mathrm{T}_+\}$时我们要定义 $g(k) = 1$, 而$k \in \{\mathrm{A}_-, \mathrm{C}_-, \mathrm{G}_-, \mathrm{T}_-\}$时 $g(k) = 0$. 这样 $G(i|x)$ 恰好就是在这个模型下碱基 i 位于 CpG 岛区域的后验概率.

标记所有状态以便定义其划分 (就像我们在 CpG 岛模型中所做的一样, 将它们标记为“+”或“−”) 时, 可以使用 (3.16) 式对序列的每个位置找到最可能的标记. 但这并不是给定序列的最可能的全局标记. 最可能的全局标记是难以直接算出的. 更深入的讨论见 Schwartz & Chow [1990] 和 Krogh [1998].

例子: CpG 岛预测

现在通过我们的模型便可以预测 CpG 岛了. 用 Viterbi 算法我们可以找到贯穿模型的最可能路径. 当路径经过“+”状态时, 一个 CpG 岛就被预测出来. 对于 41 条已知均包含 CpG 岛的序列, 除了 2 个 (假阴性, false negative) 以外所有的 CpG 岛都被找到了, 同时还预测到 121 个新 CpG 岛 (假阳性, false positive). 真正的 CpG 岛很长 (数量级在 1000 个碱基), 然而预测到的却较短, 并且 CpG 岛会被预测为几个短的片段. 通过实施以下两个简单的后续步骤后, 假阳性的数量减小到 67 个: (1) 将间隔小于 500 个碱基的预测片段连接到一起, (2) 删去短于 500 个碱基的预测片段.

使用后验解码方法, 与刚才相同的 2 个 CpG 岛被漏掉了, 同时预测出 236 个假阳性岛. 使用与上面相同的后续步骤处理后, 这个数字减小到了 83. 对这个问题, 除了后验解码预测出更多非常短的岛以外, 两种方法之间没有太大区别. 一些假阳性岛也可能是真的 CpG 岛. 那两个假阴性岛也许被错误标记了, 但也可能我们需要更精确复杂的模型来获得这些信号的所有特性. □

例子: 偶尔作弊的赌场, 第四部分

赌场模型发生了变化, 此时由公平骰子换成作弊骰子的概率仅有 0.01. 显然保持在公平骰子的概率为 0.99, 而其他的概率不变. 我们用这个模型生成了 1000 个随机投掷的结果. 根据这组结果, 由 Viterbi 算法找到的最可能路径始终没有经过作弊

骰子的状态. 图 3.7 显示了对应于这组结果的骰子公平的后验概率. 虽不完美, 但是后验解码确实可以预测出相当接近于真实情况的结果. □

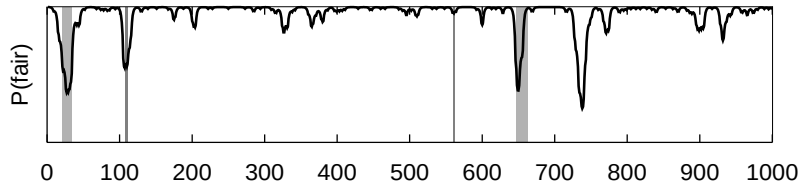


图 3.7: 当换成弊骰子的概率是 0.01 时的骰子公平的后验概率 (请与图 3.6 比较).

3.3 HMM 的参数估计

使用 HMM 时我们所面临的最困难问题也许便是首先要确定模型. 这包括两部分: 结构设计, 即都有哪些状态和它们是怎样连接的, 以及参数设定, 即转移概率 a_{kl} 和发射概率 $e_k(b)$ 的赋值. 在这一节中我们将讨论参数估计问题, 它已经有一套较完备的理论. 在下一节中我们将考虑模型结构设计, 那就更像是艺术了.

我们的工作框架是假设有一组示例序列, 我们的模型需要很好地拟合这些序列, 它们称为训练 (training) 序列. 令它们为 $x^1, \ldots, x^n$. 假设它们相互独立, 这样给定参数时, 所有序列的联合概率就是每条序列概率的乘积. 实际上, 我们在对数空间中进行计算, 因此使用序列的对数概率

$$l(x^1, \ldots, x^n|\theta) = \log P(x^1, \ldots, x^n|\theta) = \sum_{j=1}^{n} \log P(x^j|\theta), \tag{3.17}$$

其中 θ 表示模型中全部参数值的当前集合 (所有的 a 和 e). 上式等于模型的对数似然; 见第 11 章.

状态序列已知时的估计

正如路径已知时更容易写出序列的概率一样, 当所有示例序列的路径已知时也更容易估计概率参数. 通常, 情况确实如此. 例如给定一组基因组序列, 而其中的 CpG 岛已经通过实验数据标记出时, 就是这样. 其他的例子包括: 对于一个预测二级结构的 HMM, 我们从已知结构的蛋白中获得其训练序列; 或者对于一个从基因组序列预测基因的 HMM, 我们通过 cDNA 测序已经确定了其转录本 (transcript) 的结构.

当所有的路径都已知时, 我们可以对训练序列中用到的每个特定状态转移或符号发射进行计数. 将它们记为 A_{kl} 和 $E_k(b)$. 于是正如第 11 章中所述, a_{kl} 和 $e_k(b)$ 的最大似然估计分别是

$$a_{kl} = \frac{A_{kl}}{\sum_{l'} A_{kl'}} \text{和} e_k(b) = \frac{E_k(b)}{\sum_{b'} E_k(b')}. \tag{3.18}$$

a_{kl} 的估计方程与简单 Markov 链中完全相同.

与往常一样, 如果没有足够的数据, 最大似然估计很容易发生过拟合现象. 实际上如果状态 k 在示例序列中从未使用, 那么该状态的估计方程将无定义, 因为分子和分母都将为 0. 为了避免这种问题, 最好在使用式 (3.18) 前给 A_{kl} 和 $E_k(b)$ 加上一个预先设定的伪计数.

$$\begin{aligned} A_{kl} &= \text{训练数据中}k\text{到}l\text{的转移次数} + r_{kl}, \\ E_k(b) &= \text{训练数据中由}k\text{发射}b\text{的次数} + r_k(b). \end{aligned}$$

伪计数 r_{kl} 和 $r_k(b)$ 应该反映我们对概率值的先验偏向. 事实上它们有自然的概率解释, 即每个状态概率上 Bayesian Dirichlet 先验分布的参数 (见第 11 章). 它们必须是正值, 但无须是整数. 较小的总和 $\sum_{l'} r_{kl'}$ 或 $\sum_{b'} r_k(b')$ 表示较弱的先验知识, 而较大的总和表示更加确定的先验知识, 即需要更多的数据才能改变它.

路径未知时的估计: Baum-Welch 和 Viterbi 训练

当训练序列的路径未知时, 我们不能再用直接的闭式 (closed-form) 方程计算估计参数值, 而必须使用某种形式的迭代过程. 我们可以使用任意一种用于最优化连续函数的标准算法; 例如见 Press *et al.* [1992]. 然而, 有一种特殊的迭代算法作为标准方法而被使用, 称为 Baum-Welch 算法 [Baum 1972]. 它具有自然的概率解释. 简略地说, 算法先用 a_{kl} 和 $e_k(b)$ 的当前值通过考虑训练序列的可能路径来估计 A_{kl} 和 $E_k(b)$. 然后用 (3.18) 式导出 a 和 e 的新值. 迭代该过程, 直到满足某个停止标准.

我们可以证明迭代过程增加了模型的整体对数似然, 因此该过程将收敛于一个局部极大值. 不幸的是, 局部极大值通常有很多个, 你在哪一点上结束强烈依赖于参数的起始值. 当估计较大的 HMM 时, 局部极大值问题尤为严重, 后面我们将讨论几种方法以帮助解决这个问题.

严格来讲, 对给定的训练序列, Baum-Welch 算法计算 A_{kl} 和 $E_k(b)$ 作为使用每个转移和发射的期望 (expected) 次数. 与后验概率解码方法一样, 它使用相同的前向和后向值. 序列 x 中位置 i 处使用 a_{kl} 的概率是 (见练习 3.5)

$$P(\pi_i = k, \pi_{i+1} = l|x, \theta) = \frac{f_k(i)a_{kl}e_l(x_{i+1})b_l(i+1)}{P(x)}. \tag{3.19}$$

由上式, 我们可以对所有训练序列的所有位置求和得到使用 a_{kl} 的期望次数

$$A_{kl} = \sum_j \frac{1}{P(x^j)} \sum_i f_k^j(i)a_{kl}e_l(x_{i+1}^j)b_l^j(i+1), \tag{3.20}$$

其中 $f_k^j(i)$ 是为序列 j 计算的、由式 (3.10) 所定义的前向变量 $f_k(i)$, 而 $b_l^j(i)$ 是相应的后向变量. 类似地, 我们可以算出字母b出现在状态k的期望次数

$$E_k(b) = \sum_j \frac{1}{P(x^j)} \sum_{\{i|x_i^j=b\}} f_k^j(i)b_k^j(i), \tag{3.21}$$

其中内层求和仅对发射符号 b 的位置 i 求和.

算出这些期望后, 像前面一样, 使用式 (3.18) 计算新的模型参数. 我们可以像前面一样迭代使用新的参数值以得到新的 A 和 E, 但此时我们将在一个连续值的空间中收敛, 因此将永远无法真正达到最大值. 所以有必要设置一个收敛标准, 典型的方法是当整个对数似然的变化足够小时便停止运算. 除对数似然的改变以外, 也可以在迭代中使用其他终止标准. 例如对数似然可以用序列数目 n, 也许同时还可以用序列长度做归一化, 这样我们所考虑的就是每个残基的平均对数似然改变. 可以将 Baum-Welch 算法总结如下:

算法: Baum-Welch
起始: 任取模型参数.
递归:
  将所有的 A 和 E 变量设为它们的伪计数 r (或 0).
  对每条序列 $j = 1 \ldots n$:
    用前向算法对序列 j 计算 $f_k(i)$ (第 41 页).
    用后向算法对序列 j 计算 $b_k(i)$ (第 41 页).
    将序列 j 的贡献添加到 A (式 (3.20)) 和 E (式 (3.21)) 中.
  用 (3.18) 计算新的模型参数.
  计算模型的新对数似然.
终止:

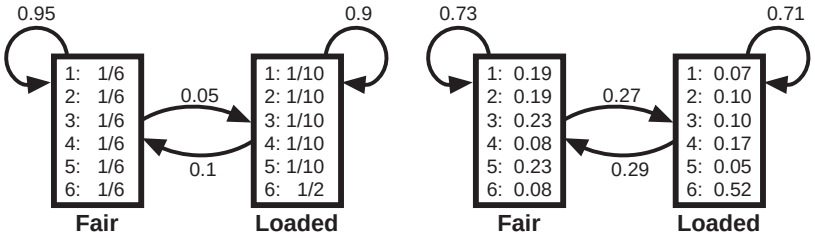
如果对数似然的变化小于某个预设的阈值
或者超过迭代次数的最大值, 则终止. ◁

正如这里所描述的, 和已知状态路径的情况一样我们通常给 A 和 E 值加上伪计数. 这种做法是可行的, 但却并不能用根据 Dirichlet 先验建立的正常 Bayes 理论所严格解释; 见第 11 章.

Baum-Welch 算法属于 EM 算法的一种特殊情形, 而 EM 算法是解决概率参数估计问题的十分强有力的通用方法. 本书第 11 章第 11.6 节给出了 EM 算法以及 Baum-Welch 算法的推导过程.

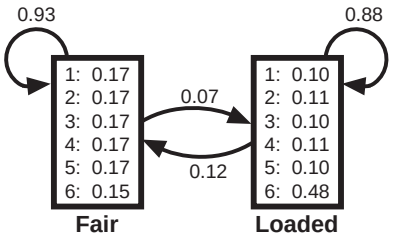
一种经常使用的 Baum-Welch 替代方法, 称为 Viterbi 训练 . 这种方法使用上面给出的 Viterbi 算法得到训练序列的最可能路径, 然后它们被用于前一节给出的重估计 (re-estimation) 过程. 得到新的参数后, 再次迭代这个过程. 此时算法精确收敛, 因为路径的指定过程是离散的, 而且我们可以继续迭代过程直至不再有路径发生改变. 在这一点上参数的估计值也将不再改变, 因为它们完全由路径决定. 与 Baum-Welch 不同, 该过程并不最大化真正的似然, 即模型参数 θ 的函数 $P(x^1, \ldots, x^n|\theta)$. 与之相对, 它找到的 θ 可以最大化所有序列之最有可能路径对似然 $P(x^1, \ldots, x^n, \pi^*(x^1), \ldots, \pi^*(x^n)|\theta)$的贡献. 很可能因为这个原因, Viterbi 训练通常表现的没有 Baum-Welch 好. 然而 Viterbi 训练被广范使用, 而且如果 HMM 被主要用于通过 Viterbi 联配进行解码时, 那么使用这种方法训练就是个好选择.

例子: 偶尔作弊的赌场, 第五部分 我们怀疑一个赌场按照原书第 54 页上描述的例子运作, 但并不确定. 日复一日, 我们通过简单地观测骰子点数收集数据. 得到足够多的数据后, 我们就希望估计一个模型. 假设我们收集到的数据构成图 3.5 中的 300 次结果. 由此观测序列, 我们可以通过 Baum-Welch 算法估计一个模型. 起始时设所有概率均为随机数. 下面画出生成数据 (与原书第 38 页例子中的相同) 的实际模型和估计出的模型.



可以看到两者非常相似, 虽然转移概率的估计值与真实值间差异较大. 该问题在某种程度上由局部极小值造成, 因此经过更多次尝试后, 我们实际上可能得到与正确模型相近的模型. 然而, 使用有限的数据永远不可能准确地估计参数.

为了阐明最后一点, 我们随机生成 30000 次投掷结果 (此处没有给出数据) 并据此估计一个模型. 该估计非常接近真实情况:



为了评估这些模型在多大程度上优于假设一直使用公平骰子的模型, 我们分别用三个模型的 300 次观测值计算出每次投掷的对数几率值:

正确的模型	0.101 比特
由 300 次结果估计的模型	0.097 比特
由 30000 次结果估计的模型	0.100 比特

由 300 次结果估计出的最差模型的对数几率值与其他两个模型几乎相同. 这是因为用于估计模型的数据同时又被用于测试模型. 如果用与之独立的投掷结果进行测试,

<table>
<tr><td colspan="2">Sequence</td><td>x_1</td><td>x_2</td><td>x_3</td><td>x_4</td><td>x_5</td><td>x_6</td><td>x_7</td><td>x_8</td><td>x_9</td><td>x_{10}</td><td>...</td></tr>
<tr><td colspan="2">Labels</td><td>−</td><td>−</td><td>−</td><td>+</td><td>+</td><td>+</td><td>+</td><td>−</td><td>−</td><td>−</td><td>...</td></tr>
<tr><td rowspan="8">State</td><td>1</td><td>−</td><td colspan="3" rowspan="4">f calculated as usual</td><td colspan="4" rowspan="4">$f = 0$</td><td colspan="4" rowspan="4">f calculated as usual</td></tr>
<tr><td>2</td><td>−</td></tr>
<tr><td>3</td><td>−</td></tr>
<tr><td>4</td><td>−</td></tr>
<tr><td>5</td><td>+</td><td colspan="4" rowspan="4">$f = 0$</td><td colspan="3" rowspan="4">f calculated as usual</td><td colspan="3" rowspan="4">$f = 0$</td></tr>
<tr><td>6</td><td>+</td></tr>
<tr><td>7</td><td>+</td></tr>
<tr><td>8</td><td>+</td></tr>
</table>

图 3.8: 标记有 4 个 + 状态和 4 个 – 状态的模型之前向表格. 每列对应于一个观测结果, 每行对应模型的一个状态. 其中前 10 个残基 $x_1, \ldots, x_{10}$ 被标记为$- - - + + + + - - -$.

其对数几率值将明显低于其他两个模型. □

练习

3.5 推导 (3.19) 式. 请使用下式

$$P(\pi_i = k, \pi_{i+1} = l|x, \theta) = \frac{1}{P(x|\theta)} P(x, \pi_i = k, \pi_{i+1} = l|\theta),$$

并且上式又可用 $P(x_1, \ldots, x_i, \pi_i = k|\theta)$ 和

$$\begin{aligned} P(x_{i+1}, \ldots, x_L, \pi_{i+1} = l|x_1, \ldots, x_i, \theta, \pi_i = k) \\ = P(x_{i+1}, \ldots, x_L, \pi_{i+1} = l|\theta, \pi_i = k) \end{aligned}$$

表示.

3.6 推导式 (3.21).

已标记序列的建模

在上面关于 CpG 岛的例子中, 我们已经看到如何用 HMM 为没有注释的序列预测标记. 在这些例子中, 我们需要分别训练 CpG 岛模型和非 CpG 岛模型, 然后把它们合并为一个更大的 HMM. 这种分别估计的方法非常烦琐, 尤其当类型多于两种时更是如此. 另外, 如果子模型间的转移是模糊的, 例如一条给定序列可能以多种方式从 CpG 岛子模型转移到另一个子模型, 那么对转移的估计就不再是简单的计数问题了. 然而有一种更直接的方法可以一次估计所有的参数, 下面我们就介绍这种方法.

此方法首先建立针对所有类型的整合模型, 我们为其中的每个状态指定类型标记. 为 CpG 岛建立模型, 岛状态的自然标记是“+”, 非岛状态的是“–”. 我们也对观测序列 $x = x_1, \ldots, x_L$ 做标记, 并称其为 $y = y_1, \ldots, y_L$. 如果 x_i 是 CpG 岛的一部分, 则 y_i 为“+”; 否则为“–”. Baum-Welch (或 Viterbi) 算法中, 当计算 f 和 b 时, 我们现在只允许贯穿模型的有效 (valid) 路径. 在有效路径中, 状态标记和序列标记相同, 即 π_i 标记为 y_i. 在前向和后向算法中, 这便对应于将所有与 y_i 标记不同的状态 l 设置为 $f_l(i) = 0$ 和 $b_l(i) = 0$ (见图 3.8).

辨别性估计

除非在子模型间有模糊的转移, 否则上面的估计过程将与用 Baum-Welch 算法分别估计子模型再用适当的转移将它们结合起来的方法得到相同结果. 这实际上对应于最大化似然

$$\theta^{ML} = \operatorname*{argmax}_{\theta} P(x, y|\theta).$$

通常我们主要关注于做好 y 的预测, 因此我们更喜欢代之以最大化 $P(y|x,\theta)$. 这称为条件最大似然 (conditional maximum likelihood, CML):

$$\theta^{CML} = \underset{\theta}{\operatorname{argmax}}\, P(y|x,\theta); \tag{3.22}$$

例如请参见 Juang & Rabiner [1991] 和 Krogh [1994]. 一个相关的准则称为最大互信息 (maximum mutual information) 或 MMI [Bahl *et al.* 1986].

似然 $P(y|x,\theta)$ 可以改写为

$$P(y|x,\theta) = \frac{P(x,y|\theta)}{P(x|\theta)},$$

其中 $P(x,y|\theta)$ 是上面描述的用前向算法对已标记序列算出的概率, 而 $P(x|\theta)$ 是忽略所有标记时用标准前向算法算出的概率. 不存在优化此概率的 EM 算法, 而且此估计过程更为复杂; 例如请参见 Normandin & Morgera [1991] 及上面提到的文献.

3.4 HMM 的模型结构

模型拓扑的选择

至此我们假设任何状态到任何状态间都可以发生转移. 虽然从全部连接的模型 (即所有转移都被允许, 而且“让模型自己找出”使用哪些转移) 出发是一种诱人的做法, 但这在实际中根本就不可行. 对于任意规模的实际问题, 即使拥有充足的训练数据, 从这种方式出发也经常得到很差的模型. 问题不在于过拟合, 而是局部极大. 模型的限制越少, 局部极大的问题就越严重. 某些方法试图根据数据, 并通过增加或删去转移和状态以调整模型拓扑 [Stolcke & Omohundro 1993; **?**]. 然而在实际中, 构建成功的 HMM 需要基于被研究问题的知识, 并谨慎地决定哪些转移在模型中是可以允许的.

禁止从状态k到状态 l 的转移对应于令 $a_{kl} = 0$. 如果我们使用 Baum-Welch 估计 (或 Viterbi 近似法), 那么重估计过程后 a_{kl} 将仍然是 0, 因为当概率是 0 时, 由 k 到 l 转移的期望次数也是 0. 因此即便并非所有的转移都被允许, 算法的所有数学部分也将保持不变.

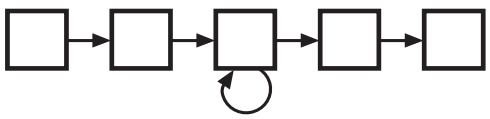
应该选择一个能够根据我们对该问题的知识, 对该问题给出合理解释的模型. 例如对 CpG 岛建模时, 很重要的一点就是模型应该能够分别为处于岛内状态和非岛状态中的 `CG` 双核苷酸给出不同的概率, 因为这被认为是区分 CpG 岛的主要决定因素.

时长建模

当为诸如碱基分布在一定长度的 DNA 中没有变化的现象建模时, 最简单的模型设计就是建立一个以概率 p 转移到自身的状态. 对 CpG 岛和作弊赌场的例子, 我们都是这样的. 进入状态后, 离开它的概率是 $1-p$, 所以该状态中存留 l 个残基 (residue) 的概率为

$$P(l \text{ residues}) = (1-p)p^{l-1}. \tag{3.23}$$

(忽略发射概率). 这种随长度呈指数衰减的分布 (称为几何分布, geometric distribution) 不适用于某些应用, 因为在这些应用中长度分布是重要的并且显著不同于指数分布. 我们可以通过引入几个具有相同残基分布的状态和状态间的转移, 对更加复杂的长度分布建模. 例如一个像这样的 (子) 模型:



可以给出最小长度为 5 个残基、并且随长度增加其分布呈指数衰减的序列. 类似地, 模型

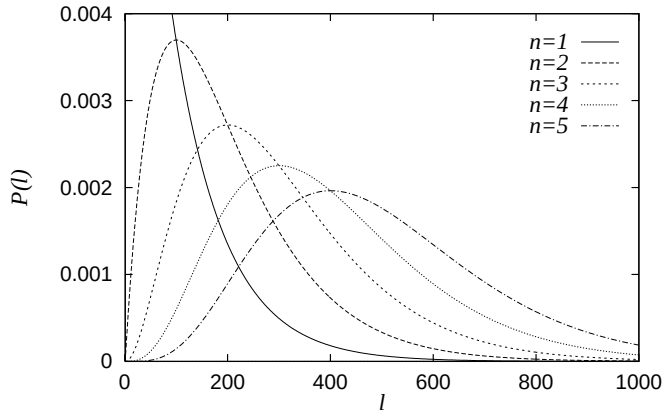
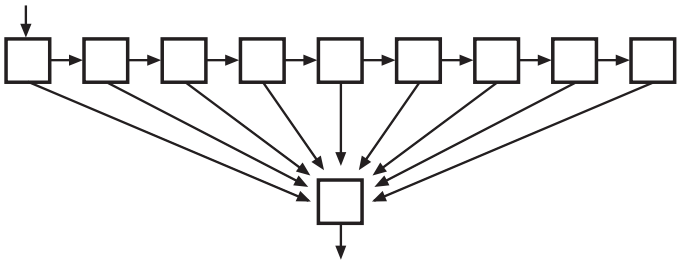
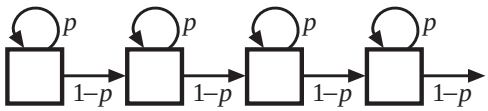


图 3.9: $p = 0.99$ 时, 包含 n (n 从 1 到 5) 个相同状态的模型的长度概率分布.



可以为长度从 2 到 10 的任意分布建模.

得到非几何长度分布的更精巧方法是使用 n 个状态, 其中每个状态均以概率 p 转移到自身, 并以概率 $1-p$ 转移到下一个状态:



显然这样的模型可以得到的最小序列长度为 n. 对长度为 l 的贯穿模型的任意给定路径, 其所有转移概率是 $p^{l-n}(1-p)^n$ (像上面一样, 我们暂时忽略发射概率). 经过状态的可能路径数为 $\binom{l-1}{n-1}$, 因此所有可能路径的总概率是

$$P(l) = \binom{l-1}{n-1} p^{l-n}(1-p)^n. \tag{3.24}$$

它称为**负二项** (negative binomial) 分布 , 当 $p = 0.99$ 且 $n \leq 5$ 时如图 3.9 所示. 对短序列而言, 贯穿模型路径数的增长比几何分布衰减还快, 因此该分布呈钟形 (bell-shaped). 路径数依赖于模型拓扑, 并且可以建立路径数与 n 和 l 有不同依赖关系的更普适模型. 对连续 Markov 过程, 可以得到的分布类型称为 Erlang 分布 , 通常也称为相位型 (phase-type) 分布, 例如见 Asmussen [1987].

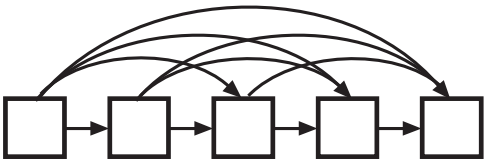
此外, 我们也可以直接对长度分布建模. 因为在许多信号处理应用中, 长度与时间等价, 所以这种方法称为**时长建模** (duration modelling). 但其代价是, 这些算法会慢许多. 更多细节见 Rabiner [1989].

沉默状态

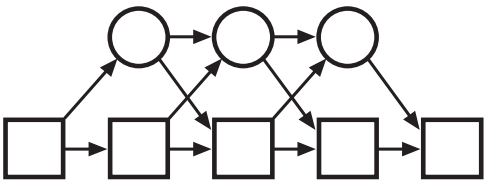
我们已经见过HMM 中不发射符号的状态, 即起始和结束状态. 这种状态称为**沉默状态** (silent states) 或**空状态** (null states), 它们也可以用于 HMM 的其他地方. 在第 5 章中我们将看到一个例子, 即状态链中的所有状态都需要与位于它们后面的所有状态相连. 这种链通常包含至少 200 个状态, 用转移将它们恰当地连到一起将

需要大概 20000 个转移概率 (假设是 200 个状态). 这个数字太大, 以至于无法从实际数据集中可靠地估计. 取而代之, 使用沉默状态后我们可以去掉大约 800 个转移.

具体情况如下: 为了允许任意的删除, 状态链需要被完全地“向前连接” (forward connected):



与此不同, 我们可以把所有的状态连接到一条平行的沉默状态链上, 在下图中用圆圈表示:



由于沉默状态不发射任何字母, 因此可以从任意“真实”状态在不发射任何字母的情况下到达它后面的任意“真实”状态.

减少参数的个数需要付出代价. 例如, 完全连接的模型可以做到从状态 1 到状态 5 和从状态 2 到状态 4 有较高的转移概率, 而从 1 到 4 和从 2 到 5 的转移概率较低. 这在使用沉默状态的模型中将无法实现.

只要不存在完全由沉默状态组成的环, 那么我们就容易将这些沉默状态整合进所有的 HMM 算法. 不存在环意味着可以为状态编号, 这样任何沉默状态间的转移都是从一个低编号的状态到一个高编号的状态. 对于前向算法, 变化如下:

(i) 对于所有的“真实”状态 l, 像前面一样为状态 k 从 $f_k(i)$ 计算 $f_l(i+1)$.

(ii) 对任何沉默状态 l, 令 $f_l(i+1) = \sum_k f_k(i+1)a_{kl}$, 其中 k 为“真实”状态.

(iii) 从编号最小的沉默状态 l 开始, 对于所有的沉默状态 $k < l$ 将 $\sum_k f_k(i+1)a_{kl}$ 加到 $f_l(i+1)$ 上.

针对 Viterbi 算法的改变完全相同 (当然连加变为取最大值), 除了第三步中逆向更新沉默状态外, 对于后向算法的改变也基本相同.

如果存在完全由沉默状态组成的环, 情况就变得有些复杂了. 通过 (仅) 计算模型中真实状态间的有效转移概率, 我们可以从计算中消去沉默状态, 该过程包括对沉默状态的 Markov 模型转移矩阵求逆 [Cox & Miller 1965]. 然而这些有效转移经常对应于一个完全连接的模型, 并将致使模型的复杂度大幅增加. 通常最好是直接确定这种环不存在.

练习

3.7 计算如上面那样长度为 L 的向前连接模型所需转移的总数. 该总数在 (如上面那样的) 含沉默状态模型中是多少?

3.8 证明经过 n 个状态的长度为 l 的路径总数为 $\binom{l-1}{n-1}$, 如式 (3.24) 中一样.

3.9 如果使用 Viterbi 算法, 存在自身环 (self-loop) 的包含 n 个状态的模型的长度概率分布是什么?

3.5 更复杂的 Markov 链

高阶 Markov 链

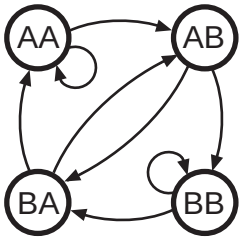
n 阶 Markov 过程是每个事件都依赖于它前面的 n 个事件的随机过程, 所以

$$P(x_i|x_{i-1}, x_{i-2}, \ldots, x_1) = P(x_i|x_{i-1}, \ldots, x_{i-n}). \tag{3.25}$$

我们前面讨论的 Markov 链都是 1 阶的.

字母表 $\mathscr{A}$ 上的 n 阶 Markov 链等价于 n 元字母表 $\mathscr{A}^n$ 上的 1 阶 Markov 链. 这基于一个简单事实: $P(x_k|x_{k-1}\ldots x_{k-n}) = P(x_k, x_{k-1}\ldots x_{k-n+1}|x_{k-1}\ldots x_{k-n})$ (给定条件 B 时 A 和 B 的概率, 就是给定 B 时 A 的概率). 即给定以 x_{k-1} 结尾的 n 元组的条件时 x_k 的概率, 等于给定以 x_{k-1} 结尾的 n 元组的条件时以 x_k 结尾的 n 元组的概率.

下面考虑一个 2 阶 Markov 链的简单例子, 它生成仅含字母 A 和 B 的序列. 序列被转化为字母对序列, 因此序列 ABBAB 将变为 AB-BB-BA-AB. 与之等价的四状态 1 阶 Markov 链如下:



在该等价模型中并非所有的转移都被允许 (或者说, 某些转移概率为 0). 这是因为一个特定字母后面只可能存在两种不同的字母对; 例如状态 AB 后面只能连接状态 BA 和 BB. 不存在能够从状态 AB 到状态 AA 的序列. 与之类似, DNA 序列的一个 2 阶模型等价于 16 个双核苷酸字母表上的 1 阶模型. 五碱基序列 `CGTCA` 在双核苷酸模型中对应于四状态链 CG-GT-TC-CA.

除了 n 阶模型和 1 阶模型在理论上等价外, 高 (即大于 1) 阶模型的框架有时更方便. 从理论上讲, 高阶模型可以完全按与 1 阶模型等价的方法处理.

寻找原核基因

这里给出识别原核 (prokaryotic) 基因的模型作为例子. 原核生物(细菌, bacteria) 的基因有非常简单的一维结构. 编码蛋白的基因以起始密码子 (codon) 开始, 随后是一些编码氨基酸的密码子, 最后以终止密码子结束; 见图 3.10. 密码子是 DNA 核苷酸的三元组, 其中有 61 个编码氨基酸, 而另外 3 个是终止密码子. 为了专注于建模, 我们在这里忽略诸如阅读框移位 (frame shift) 和非编码蛋白基因这样的复杂问题.

通过简单地寻找具有正确结构的 DNA 片段, 就可以很容易地找到候选基因片段, 正确结构即由 3 种可能的起始密码子之一开始, 紧接着一些非终止密码子, 最后以 3 种终止密码子之一为结束. 这样的候选基因片段称为开放阅读框 (open reading frame) 或 ORF. 通常许多交叠的 ORF 共有同一终止密码子而有不同的起始密码子. (ORF 通常指两个终止密码子间最大的开放阅读框, 但我们在这里指所有可能的候选基因片段.) ORF 远多于真正的基因, 在这里我们将简略介绍一些可能的方法以便区分非编码 ORF 与真正的基因.

在这个例子中我们使用来自大肠杆菌 (*E. coli*) 的 DNA (数据集细节在 Krogh, Mian & Haussler [1994] 中描述). 我们只考虑长于 100 个碱基的基因. 在数据集中共有 1100 个这样的基因. 该集合被任意地分为用于训练模型的、包含 900 个基因的训练集, 以及包含剩余 200 个基因的测试集.

像这章前面介绍的 CpG 岛一样, 我们估计一个1阶模型, 并测试它从其他的 ORF 中辨别出基因的效果如何. 在测试集中, 我们找出大约 6500 个长度大于 100 碱基的 ORF. 这其中未包含与已知真实基因共有同一终止密码子的 ORF , 因为它们通

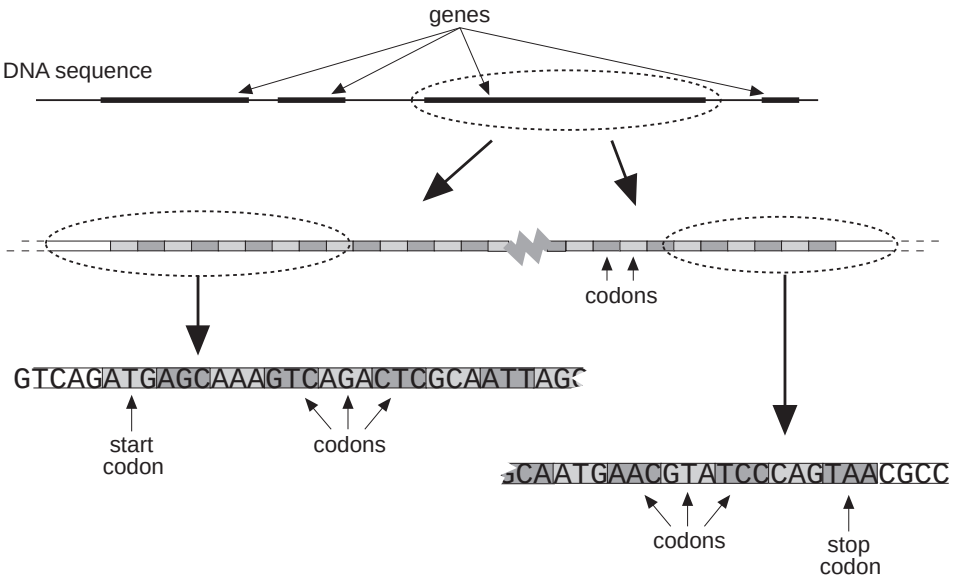


图 3.10: 原核生物的基因结构.

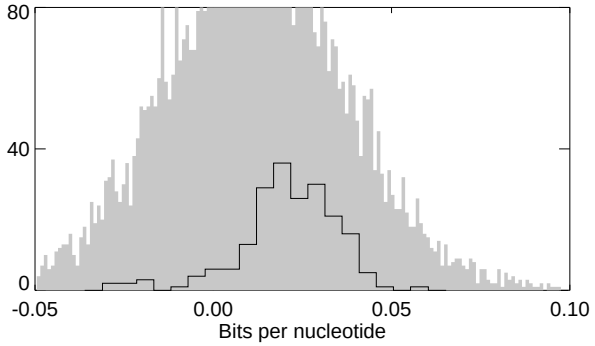


图 3.11: 根据 1 阶 Markov 链得到的所有 NORF (灰色) 和基因 (黑线) 中的每个核苷酸的对数几率柱状图. 由于 NORF 的数目很大, 所以它的组距 (bin size) 减小到五分之一.

常会得到很好的分值并使我们后续的分析更加困难. 剩下没有标记为编码区的 ORF 将称为 NORF (意为非编码 ORF, non-coding ORF).

图 3.11 的直方图表示了每个核苷酸的对数几率. 我们用可能的最简单方法作为零模型 (null model) 计算对数几率值, 即每种碱基的概率等于它在全部数据中出现的频率. 所有基因中每个核苷酸的平均对数几率值是 0.018, 而 NORF 的值只有一半 (0.009), 但它们的方差却使得此直方图对辨别几乎无法起到任何作用. 如果在画对数几率的直方图时没有除以序列长度, 你就会自欺欺人地认为模型具有足够的辨识力, 因为基因的平均长度大于 NORF, 所以 NORF 的总对数几率也相对更大. 这样几乎所有呈现出的关于基因的信息将全部来自于长度分布而不是这个模型.

值得注意的是直方图的平均值不在 0 比特, 并且两个分布 (基因和 NORF) 的均值非常接近. 这说明 Markov 链确实找到了碱基对之间的非随机联系, 但是它在编码和非编码区大体相同. 在 2 阶链中, 一个核苷酸的概率依赖于它前面的两个, 所以它跨越了密码子的长度. 因此我们也尝试了 2 阶模型, 但结果几乎和 1 阶模型相同, 所以这里没有给出直方图. 由于更高阶的模型也没有分开三联阅读框, 即密码子中三个碱基的不同位置, 因此换为更高阶的 Markov 链似乎将不会有大帮助.

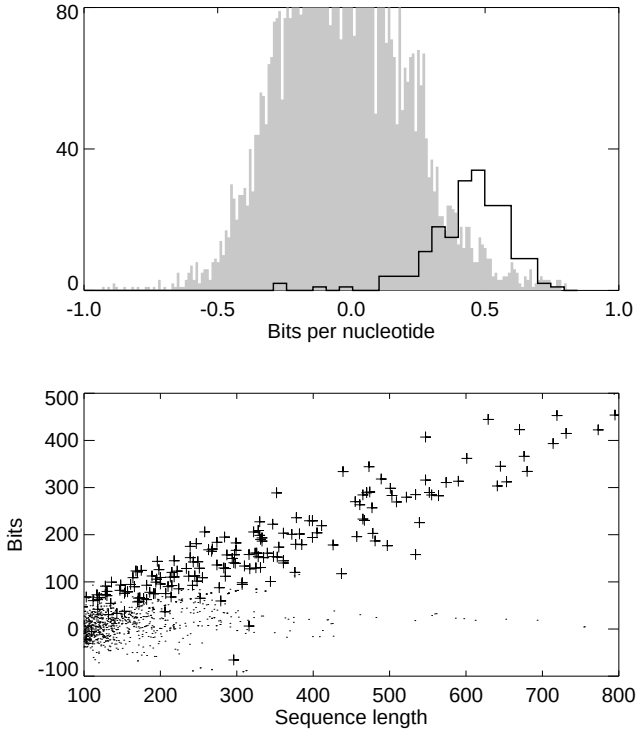


图 3.12: 上图: 密码子 Markov 链中 NORF 和基因的柱状图 (比较图 3.11). 下图: 作为基因 (+) 和 NORF (·) 长度函数的对数几率值.

虽然可以建立一个高阶非齐次 (inhomogeneous) Markov 链 (在下一节中讨论) 对三种不同阅读框中的碱基建模, 但由于目标是给 ORF 计分, 所以我们将使用不同的方法. 序列被转化为密码子序列. 为 64 个密码子中的每一个分配任意的符号, 于是所有基因和 NORF 也用这些字母表示 (这样序列长度将变为核苷酸序列的三分之一). 请注意由于三联组不交叠, 所以这种转化与上述由 n 阶模型向1阶模型的转化有些不同.

由翻译序列估计出含 64 个状态的 1 阶 Markov 链, 并且用与上面所述模型完全相同的方法, 用测试集里的基因和 NORF 对其进行测试. 结果显示在图 3.12 中. 虽然区别效果并不完美, 但我们可以看到它明显优于其他模型. 注意, 我们现在用对数几率分值作比较的分布是密码子的均匀分布. 灰色峰的中心在 0 点附近, 说明 Markov 链已经找到了编码区的特殊信号, 并且对平均的 NORF 来讲密码子的使用基本上是随机的, 而有一段明显得分较高的 NORF 反映了与我们数据中标记不一致的那些真实基因. 在这幅图中大多数得分高于 0.3$\tilde{0}$.35 比特的 ORF 可能与真实基因重叠. NORF 柱状图使用了较小的组距 (与图 3.11 中一样), 如果使用同样的组距, NORF 的柱状图将高出大约 5 倍.

如果对数几率没有被序列长度归一化, 那么区分能力将明显改善, 因为真实基因往往比 NORF 长, 见图 3.12.

练习

3.10 计算上面密码子模型中参数的个数. 数据集包含大约 300000 个密码子. 是否可以从该数据集估计一个 2 阶 Markov 链?

3.11 如何改进上面的基因模型?

非齐次 Markov 链

正如上文提到的, 一个成功的基因 Markov 模型需要对密码子的统计量建模. 即使不转化为另一个字母表, 这也可以实现. 众所周知基因的三个密码子位置有非常不同的统计量, 因此我们很自然地要使用三条不同的 Markov 链对编码区建模. 根据密码子里的位置, 三个模型被编号为 1 到 3. 假设 x_1 在密码子的第三位, 则 $x_2, x_3, \ldots$ 的概率为

$$a^1_{x_1x_2}a^2_{x_2x_3}a^3_{x_3x_4}a^1_{x_4x_5}a^2_{x_5x_6}\cdots$$

其中模型 k 的参数称为 a^k. 这称为非齐次 Markov 链 (inhomogeneous Markov chain). 此时我们假设该链是 1 阶的, 当然它也可以扩展为 n 阶. 此模型的参数估计由 3.1 节中描述的齐次模型参数估计直接扩展得到: 对类似于上面的 2 阶非齐次 Markov 链, 通过对其最后一个碱基位于密码子第一位的三联组进行计数来估算模型 1 的参数, 模型 2 和 3 也类似.

非齐次 Markov 链广泛用于 GENEMARK 基因预测程序 [Borodovsky & McIninch 1993], 这是现在使用最广泛的原核生物基因预测方法. 编码区的不超过五阶的非齐次模型已经与非编码区的齐次模型组合到一起, 以便在大量不同的细菌基因组中定位基因.

上面描述的 1 阶模型也可以被构建为一个 HMM , 其状态数等于字母表长度的三倍 (对 DNA 来说总数为 12). 向 HMM 中加入许多附加状态便可以构建更高阶的模型. 然而在 HMM 的状态中可以包含 n 阶 Markov 发射概率, 在这种情况下发射概率依赖于前面 n 个字母, 所以发射概率 (3.5) 变为

$$e_k(b|b_1,\ldots,b_n) = P(x_i|\pi_i = k, x_{i-1} = b_1, \ldots, x_{i-n} = b_n).$$

所有为标准 HMM 推得的算法经过明显的变换就都可以用于含有这种发射的模型. 人们也正用这种模型做基因预测 [Krogh 1998].

练习

3.12 画出与上面给出的 1 阶非齐次 Markov 链相对应的 HMM.

3.6 HMM 算法的数值稳定性

在 Viterbi、前向或后向算法中将多个概率相乘时, 即便使用现代的浮点处理器, 我们也会陷入到数值问题中. 例如对于 DNA, 我们也许要对 100000 碱基或更长的基因组序列建模. 假设一个发射概率和一个转移概率的乘积通常是 0.1, 则 Viterbi 路径的概率将在 $10^{-100000}$ 的量级. 多数计算机对这样的数字将表现很差: 它会产生下溢错误并导致程序崩溃; 更糟的是, 程序也许将继续运行并产生无法预料的错误数字. 现在有两种不同的方法可以处理这个问题.

对数变换

对 Viterbi 算法我们应该一直使用所有概率的对数. 因为乘积的对数就是对数的和, 所以所有乘法就都变成了求和. 假设对数以 10 为底, 上面的概率 $10^{-100000}$ 之对数值为 -100000. 这样, 下溢问题就基本解决了. 另外, 在一些计算机上求和运算比求乘积快, 所以该算法在这些计算机上也将运行得更快.

取对数后, 我们要给所有的模型参数加一个波浪号 (tilde), 例如 $\widetilde{a}_{kl} = \log a_{kl}$. 这样 Viterbi 算法 (3.8) 的递归关系变为

$$V_l(i+1) = \widetilde{e}_l(x_{i+1}) + \max_k(V_k(i) + \widetilde{a}_{kl}),$$

其中我们用 V 代表 v 的对数值. 只要对数的底大于 1 (例如 2, e, 或 10), 它具体是多少并不重要.

更加高效的做法是, 在运行 Viterbi 算法前对所有的模型参数都取对数值, 这样可以避免在动态规划递归中反复调用取对数函数.

对于前向和后向算法, 对数变换存在一个问题: 如果不使用求指数和取对数函数, 那么概率和的对数值就无法由概率的对数值计算, 而这两个函数均需要大量的计算. 然而, 实际情况并没有那么差. 假设你要从对数概率 $\widetilde{p} = \log p$ 和 $\widetilde{q} = \log q$ 计算 $\widetilde{r} = \log(p+q)$. 那么直接的方法是计算 $\widetilde{r} = \log(\exp(\widetilde{p}) + \exp(\widetilde{q}))$. 提出 $\widetilde{p}$后, 此式变为

$$\widetilde{r} = \widetilde{p} + \log(1 + \exp(\widetilde{q} - \widetilde{p})).$$

我们可以利用一个内插 (interpolation) 表格中的数据近似函数 $\log(1+\exp(x))$. 因为对于大的 $(\widetilde{p} - \widetilde{q})$ 而言, $\exp(\widetilde{q} - \widetilde{p})$将快速地趋近于零, 所以假如我们总是提出 $\widetilde{p}$ 和 $\widetilde{q}$ 中较大的一个, 那么对于一个合理的精度水平, 该表格实际上可以非常小.

概率的缩放

对数变换的一种代替方法是对变量 f 和 b 进行调整, 以便它们位于易处理的数值区间 [Rabiner 1989]. 对每个 i 定义一个缩放变量 s_i, 并且定义新的变量 f 为

$$\widetilde{f}_l(i) = \frac{f_l(i)}{\prod_{j=1}^{i} s_j}. \tag{3.26}$$

由此容易看出

$$\widetilde{f}_l(i+1) = \frac{1}{s_{i+1}} e_l(x_{i+1}) \sum_k \widetilde{f}_k(i) a_{kl},$$

这样前向递归 (3.11) 仅发生微小改变. 无论我们怎样定义 s_i 都可以, 但一种方便的选择是令 $\sum_l \widetilde{f}_l(i) = 1$, 这意味着

$$s_{i+1} = \sum_l e_l(x_{i+1}) \sum_k \widetilde{f}_k(i) a_{kl}.$$

变量 b 需要用同样的数字进行缩放, 这样 (3.3) 中的递归步骤变为

$$\widetilde{b}_k(i) = \frac{1}{s_i} \sum_l a_{kl} \widetilde{b}_l(i+1) e_l(x_{i+1}).$$

这种缩放方法通常效果不错, 但是在类似于第 5 章中我们所描述的那种含许多沉默状态的模型里, 该方法仍然可能导致下溢错误.

练习

3.13 用式 (3.26) 证明在按上面的方法选择 s_i 时 $P(x) = \prod_{j=1}^{L} s_j$. 当然更明智的方法是计算 $\log P(x) = \sum_j \log s_j$.

3.14 用上题结论证明, 用缩放过的 f 和 b 变量表示时, 方程 (3.20) 可以化简. 并且, 为缩放过的变量导出结论式 (3.21).

3.7 补充读物

对 HMM 更基础的介绍有 Rabiner & Juang [1986] 和 Krogh [1998].

Borodovsky *et al.* [1986a; 1986b; 1986c] 完成了类 HMM 模型在序列分析中的早期应用, 他使用了第 53 页中描述的非齐次 Markov 链. 这后来引出了 GENEMARK 基因预测程序 [Borodovsky & McIninch 1993]. Cardon & Stormo [1992] 引入了一个与 HMM 有很多相似之处的期望最大化 (EM) 方法, 并将其用于为蛋白结合模体建模. HMM 在基因预测中的随后应用包括 Krogh, Mian & Haussler [1994], Henderson, Salzberg & Fasman [1997], 和 Krogh [1997a, 1997b, 1998] 以及集成了神经网络和 HMM 的系统 [Stormo & Haussler 1994; Kulp *et al.* 1996; Reese *et*

al. 1997; Burge & Karlin 1997]. 这种混合系统在其他应用中也变得非常流行; 例如 Bengio *et al.* [1992], Frasconi & Bengio [1994], Renals *et al.* [1994], Baldi & Chauvin [1995] 和 Riis & Krogh [1997].

Churchill [1989] 使用 HMM 为分别来自线粒体、人的 X 染色体和噬菌体 λ 的 DNA 之间的组成差异建模, 后来又利用 HMM 研究基因组的组成结构 [Churchill 1992]. 其他应用包括用来预测蛋白二级结构的三状态 HMM [Asai, Hayamizu & Handa 1993], 用连成环状的十状态 HMM 为核小体中的振荡模式 (oscillatory pattern) 建模 [Baldi *et al.* 1996], 探测短蛋白编码区并分析蓝藻 (cyanobacteria) 的翻译起始位点 [Yada & Hirosawa 1996; Yada, Sazuka & Hirosawa 1997], 描述原核和真核启动子特征 [Pedersen *et al.* 1996], 以及识别分歧点 (branch point) [Tolstrup, Rouzé & Brunak 1997]. HMM 的一些其他应用将在第 5 章和第 6 章中的列型 HMM 部分讨论.

第四章

应用 HMM 的二序列联配

既然已经从隐马模型理论中学到新的技艺, 我们就用简短的一章谈论前面有关二序列联配的内容. 在第 2 章中, 我们引入了多状态的有限状态自动机, 以便对更复杂的二序列联配动态规划算法进行适当的描述. 把它们转化为 HMM 后, 我们可将其视为含空位联配 (gapped alignment) 过程概率论解释的基础. 使用该方法的一大优势在于, 我们不但可以利用已得到的概率论模型探讨由动态规划得出联配的可靠性, 而且还可以寻找备择 (次优) 联配. 的确, 通过对所有可能联配的概率加权, 我们可以评价二序列间不依赖于特定联配的相似性. 我们还可以从简单模型出发构建更为特殊的概率论模型, 以便对更为复杂的序列联配建模, 正如前面对 FSA 所讨论的一样.

我们先简单回顾为仿射空位罚分(affine gap penalties) 二序列联配所引入的有限状态自动机. 我们需要三个状态, M 对应匹配, 另外的两个状态就对应插入, 如图4.1中 X, Y 所示. 动态规划矩阵中更新各状态取值的迭代关系如下:

$$
\begin{aligned}
V^{\mathrm{M}}(i,j) &= s(x_i, y_j) + \max \begin{cases} V^{\mathrm{M}}(i-1,j-1), \\ V^{\mathrm{X}}(i-1,j-1), \\ V^{\mathrm{Y}}(i-1,j-1); \end{cases} \\
V^{\mathrm{X}}(i,j) &= \max \begin{cases} V^{\mathrm{M}}(i-1,j) - d, \\ V^{\mathrm{X}}(i-1,j) - e; \end{cases} \\
V^{\mathrm{Y}}(i,j) &= \max \begin{cases} V^{\mathrm{M}}(i,j-1) - d, \\ V^{\mathrm{Y}}(i,j-1) - e. \end{cases}
\end{aligned}
\tag{4.1}
$$

以上方程组适用于全局联配. 与前面一样, 我们通常给出全局联配的详细方程, 而指出作何改变就可适用于局部联配.

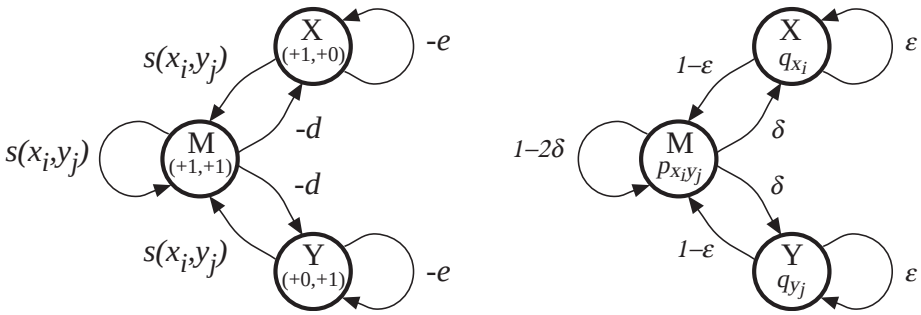


图 4.1: 仿射空位联配的有限状态机(左图) 及与之对应的概率论模型(右图).

4.1 成对 HMM

我们需要对图 4.1 中左边的 FSA 做两种改变, 才能将其转化为 HMM . 第一, 如图 4.1 的右图所示, 我们需要为状态中字符发射和状态间转移都赋予一定的概率. 例如, 状态 M 发射已联配字符对 a:b 的发射概率分布是 p_{ab}, 同时状态 X 和 Y 按分布 q_a 发射 a 和空位构成的字符对. 因为状态 X 发射序列 x 中的字符 x_i, 因此我们在代表状态 X 的圆圈里写上 q_{x_i}. 并且, 我们指定状态间的转移概率, 并使其满足从任一状态转移到其他所有状态的概率之和为 1. 考虑到对称性, 三个主状态间的转移概率有两个自由参数. 我们记从状态 M 到一个插入状态 (X 或 Y) 的转移概率为 δ, 停留在某个插入状态的概率为 ε.

但是, 图 4.1 中右边的模型并不能生成一个完全模型, 并给出在所有可能序列上的概率分布. 要实现这个想法, 我们需要定义一个开始状态和一个结束状态, 如图 4.2 所示. 实际上这样定义后, 我们就对第2章所使用的动态规划算法的初始及终止条件进行了形式化. 在下面的介绍中我们将看到, 开始与结束状态的更复杂组合可以对应于局部联配以及其他类型的联配. 加入明显的结束状态之后, 需要向模型引入另一个参数, 即转移到结束状态的概率, 在此我们假设这个概率对状态 M、X 和 Y 都相同, 并称之为 τ. 事实上, 它将决定模型联配出来的序列平均长度. 我们暂时设定从开始状态出发的转移概率等于从 M 状态出发的转移概率 (其实我们也可以说从 M 状态开始, 但是我们希望澄清一点: 和终止状态一样, 我们可以单独考虑初始状态).

这时我们得到的概率论模型与第 3 章定义过的隐马模型相当类似. 它们的差别在于这里模型发射的并不是单条序列, 而是二序列联配. 为了将其与发射出单条序列的标准 HMM 加以区别, 我们称这种模型为**成对 HMM** (pair HMM). 虽然因为这种模型多发射了一条序列, 搜索空间要比原来多一个维数, 但是第 3 章中的所有算法都可用于成对 HMM . 例如对 Viterbi 概率, 我们用$v^k(i,j)$代替$v_k(i)$. 下面我们将给出应用于如图 4.2 所示的、基本成对 HMM 的关键算法方程组.

正如标准的 HMM 可以发射一条序列那样, 成对 HMM 可以发射一对已联配的序列. 要达到这个目的, 我们从开始状态起始, 然后在下面的两步间不断循环: (1) 按照离开当前状态的转移概率分布选择下一个状态; (2) 按照新状态的发射分布选择符号对, 并将其加入联配中. 当转移到结束状态时, 此过程停止. 因为已知每个步骤的概率, 所以我们可以记录产生已有特定联配的总概率. 它就是每个单独步骤概率的乘积.

最可能路径是最优 FSA 联配

当给定序列 x 和 y 时, 第3章的 Viterbi 算法将允许我们找到贯穿成对 HMM 的最可能路径. 图 4.2 中全局成对 HMM 的正确形式如下. 为简化方程, 我们将开始状态定义为 M. 与前章一致, 我们使用小写字符 $v^{\cdot}(i,j)$ 表示概率值, 用大写字母

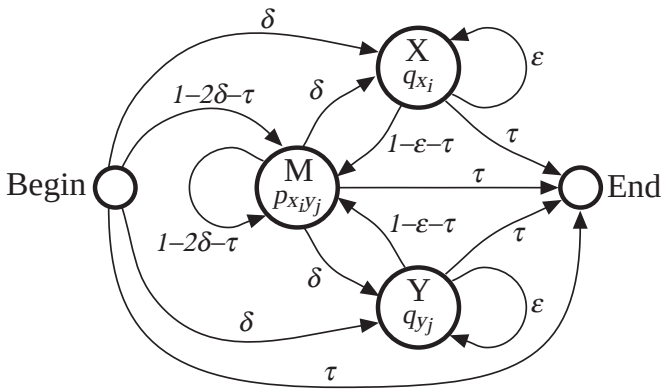


图 4.2: 图 4.1 的全概率形式.

$V^{\cdot}(i,j)$ 表示对数几率分值. 我们先给出概率形式的 Viterbi 算法:

算法: 成对 HMM 的 Viterbi 算法

初始: $v^{\mathrm{M}}(0,0)=1, v^{\mathrm{X}}(0,0)=v^{\mathrm{Y}}(0,0)=0$;

令所有$v^{\bullet}(i,-1), v^{\bullet}(-1,j)$为0, 令所有其余的$v^{\bullet}(i,0)$和$v^{\bullet}(0,j)$均为0.

迭代: 除$(i,j)=(0,0)$外, 对所有$i=0,\ldots,n, j=0,\ldots,m$:

$$v^{\mathrm{M}}(i,j)=p_{x_i y_j}\max\begin{cases}(1-2\delta-\tau)v^{\mathrm{M}}(i-1,j-1),\\(1-\varepsilon-\tau)v^{\mathrm{X}}(i-1,j-1),\\(1-\varepsilon-\tau)v^{\mathrm{Y}}(i-1,j-1);\end{cases}$$

$$v^{\mathrm{X}}(i,j)=q_{x_i}\max\begin{cases}\delta v^{\mathrm{M}}(i-1,j),\\n\varepsilon v^{\mathrm{X}}(i-1,j);\end{cases}$$

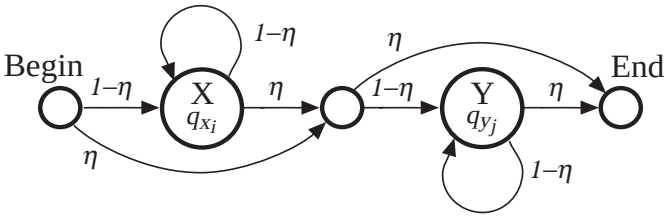
$$v^{\mathrm{Y}}(i,j)=q_{y_j}\max\begin{cases}\delta v^{\mathrm{M}}(i,j-1),\\\varepsilon v^{\mathrm{Y}}(i,j-1).\end{cases}$$

终止: $v^{\mathrm{E}}=\tau\max(v^{\mathrm{M}}(n,m),v^{\mathrm{X}}(n,m),v^{\mathrm{Y}}(n,m))$. ◁

要找到最优联配, 像往常一样我们需要记录指针然后回溯. 当然, 为了取得最终的联配结果, 我们必须像第 2 章那样在回溯的过程中记录路径中每一步所发射的残基, 以及 (或不记录) 第 3 章里描述的那类 HMM 的状态序列.

虽然成对 HMM 的 Viterbi 算法迭代方程与使用状态机的二序列联配 (4.1) 有着明显的相同形式, 但是弄清此二者间对应的确切形式是很有启发意义的.

首先, 我们必须变换为关于随机模型的对数几率比. 实际上我们已经有了联配的全概率论模型, 我们还需要一个包含适当终止条件的随机模型. 此前我们忽略了一个事实, 即随机模型并不能以适当的概率形式产生出变长序列. 下面是一个新的随机模型, 它也是一个成对HMM .



主要状态为 X 和 Y, 它们相互独立地交替发射两条序列. 它们都有以概率 $(1-\eta)$ 返回到自身的环. 与开始和结束状态一样, 在 X 和 Y 之间也有一个沉默状态, 上图中用小圆圈表示. 它并不发射任何符号, 但却可以用来从 X 和开始状态中收集输入字符(此用法的更多信息请参见原书第 p. 48 页中沉默状态的相关章节). 当这样定义之后, 就象图 4.2 的成对 HMM 模型那样, 该模型允许产生长度为 0 的序列 x 及 y, 并且可以生成序列随机模型分布的简单形式. 根据这个模型, 一对序列 x 和 y 的概率为:

$$\begin{aligned}P(x,y|R)&=\eta(1-\eta)^n\prod_{i=1}^{n}q_{x_i}\eta(1-\eta)^m\prod_{j=1}^{m}q_{y_j}\\&=\eta^2(1-\eta)^{n+m}\prod_{i=1}^{n}q_{x_i}\prod_{j=1}^{m}q_{y_j}.\end{aligned}\tag{4.2}$$

现在, 我们要将上式的每一项分配到组成 Viterbi 联配概率的乘积项中, 以便使整条联配的几率比值可以表示为每项几率比值的乘积 (相应地, 联配的对数几率比值等于每项的对数几率之和). 为此, 我们将因式 $(1-\eta)$ 和相应的因子 q_a 分配给 Viterbi 步骤中每步发射出的残基. 于是, 当 a 和 b 是两个匹配的残基时, 为匹配转移分配 $(1-\eta)^2 q_a q_b$, 当 a 为插入的残基时, 为插入状态分配 $(1-\eta)q_a$. 因为 Viterbi 路径需要考虑所有的残基, 所以刚好共有 $(n+m)$ 项, 于是式 (4.2) 中除初始因式 η^2 外的所有项都被考虑到了.

在对数几率项中, 现在我们可以按照包含对数几率发射分值与对数几率转移分值的可加 (additive) 模型进行计算. 实际上, 这通常也是实现成对 HMM 算法的最实用方法. 由此, 发射分值和转移分值可以按如下方式合并到一起,

$$s(a,b) = \log \frac{p_{ab}}{q_a q_b} + \log \frac{(1-2\delta-\tau)}{(1-\eta)^2},$$
$$d = -\log \frac{\delta(1-\varepsilon-\tau)}{(1-\eta)(1-2\delta-\tau)},$$
$$e = -\log \frac{\varepsilon}{1-\eta},$$

以便作为使用动态规划的序列联配标准项分值. 请注意, 由于 Viterbi 和随机模型中的因子被消掉, 所以 q_a 对 d 和 e 的贡献已经消失. 同时, 为了吸收来自匹配和空位状态转移间的差别, 我们在 s 和 d 的表达式里加入了一些小技巧. 我们想用 $s(a,b)$ 做为每个匹配的分值, 无论它前面是匹配还是插入. 为了正确实施这种想法, 我们已经对 d 作出调整, 以便从插入状态转移回来时校正匹配分值的差别. 这意味着对于插入, 动态规划矩阵的项不再精确对应于在该状态的对数几率比值, 尽管最后结果仍然是正确的.

现在, 我们可以给出 Viterbi 联配算法的对数几率形式, 该形式与标准的成对动态规划类似.

算法: 最优对数几率联配 起始:

令$V^{\mathrm{M}}(0,0) = -2\log\eta, V^{\mathrm{X}}(0,0) = V^{\mathrm{Y}}(0,0) = -\infty$.

所有的$V^{\bullet}(i,-1), V^{\bullet}(-1,j)$都设为$-\infty$.

迭代: 除了$(0,0)$, 对$i = 0,\ldots,n, j = 0,\ldots,m$:

$$V^{\mathrm{M}}(i,j) = s(x_i,y_j) + \max\begin{cases} V^{\mathrm{M}}(i-1,j-1), \\ V^{\mathrm{X}}(i-1,j-1), \\ V^{\mathrm{Y}}(i-1,j-1); \end{cases}$$
$$V^{\mathrm{X}}(i,j) = \max\begin{cases} V^{\mathrm{M}}(i-1,j) - d, \\ V^{\mathrm{X}}(i-1,j) - e; \end{cases}$$
$$V^{\mathrm{Y}}(i,j) = \max\begin{cases} V^{\mathrm{M}}(i,j-1) - d, \\ V^{\mathrm{Y}}(i,j-1) - e. \end{cases}$$

终止:

$V = \max(V^{\mathrm{M}}(n,m), V^{\mathrm{X}}(n,m) + c, V^{\mathrm{Y}}(n,m) + c)$. ◁

除了起始步骤中的常数项 $2\log\eta$ 以及终止步骤中的常数 $c = \log(1-2\delta-\tau) - \log(1-\varepsilon-\tau)$ 外, 这个算法与式 (4.1) 完全一致, c 就是前面描述的对 d 的修正. 实际上, 对 d 的修正仅仅是对匹配和插入状态赋予相同退出概率 τ 的结果. 如果改设从空位状态退出的转移概率为 $(1-\varepsilon)\tau/(1-2\delta)$, 那么 c 将变为 0, 因此当加上一项来自于起始条件得到的可加常数后, 对数几率算法就与标准二序列仿射空位联配算法具有完全相同的形式.

正如我们所介绍的, 这个流程显示了为获得最有可能的联配, 对图 4.2 所示的任何一个成对 HMM 而言, 我们如何才能推导出与之等价的 FSA. 这让我们看到序列联配中所使用项的基于概率的严格解释. 要是反过来, 从 FSA 形式的动态规划算法求得相应的成对 HMM, 将会更为复杂. 一般说来, 该算法还需要一个新的参数 λ 作为分值的全局缩放因子, 并且对于一组给定的分值, 也许还需要对 η 和 τ 的选取加以限制.

局部联配的成对 HMM

图 4.2 所示的模型适于找到序列间的全局匹配. 正如第 2 章所述, 许多最灵敏的成对搜索都是局部的. 当我们介绍局部联配算法以及其他变体算法, 如重复和交叠算法时, 我们都从更新方程及边界条件两方面解释其不同. 在成对 HMM 的形式体系中, 所有这些变化将通过添加状态和转移来体现. 于是, 我们可以为每种变体都画

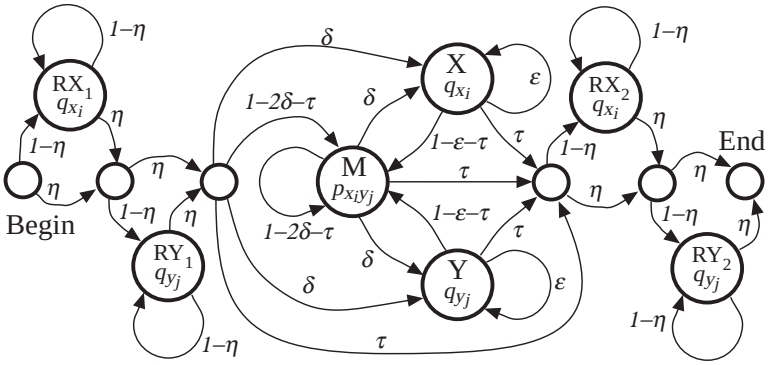


图 4.3: 局部联配的成对HMM . 它由全局模型(状态M , X 和Y) 和两侧相同的随机模型(状态RX_1, RY_1和RX_2, RY_2) 构成.

出一个单独的成对 HMM 模型. 图 4.3 画出了局部联配的模型. 看上去它比图 4.2 中的全局模型更为复杂,但它实际上仅是许多更简单小模块的组合.

一个完整的概率论模型必须考虑到所有的序列 x 及 y: 不仅仅包括 x 和 y 间的局部联配, 而且还有未联配的侧翼序列. 因此, 在图 4.2 的三状态匹配部分前后, 我们加入了额外的模型部分. 因为侧翼区序列都未联配, 所以每条序列的侧翼片段都是完整的随机背景模型. 当计算与随机模型相比得到的匹配对数几率分值时, 这些部分之似然贡献中的大多数项都会被随机模型中对应的项消去, 只剩下来自模型核心部分的局部匹配分值和一些额外的一次性 (one-off) 项. 类似的复合模型可以被用来构建交叠和重复模型, 以及第 2 章中讨论的众多混合模型.

练习

4.1 在完全随机模型下, 序列 x 长度为 t 的概率是多少?

4.2 从完全随机模型得到的序列期望长度为多少? 应该如何设置参数 η?

4.2 x 和 y 的全概率, 对所有路径求和

引入成对 HMM , 不仅为使用动态规划方法得到的标准二序列联配提供了另一种可选的基本原理, 而且允许我们做更多的事情. 在第 2 章讨论匹配的显著性时我们提到了一个问题, 即当相似性很低时, 很难确定需要计分的正确联配并检验其显著性. 现在, 我们可以计算一对序列根据 HMM 以**任何**联配形式关联起来的概率值, 这样就避开了该问题 (和整个第 2 章提到的方法). 为此, 我们对所有联配 (alignment) 求和:

$$P(x,y) = \sum_{\text{alignments } \pi} P(x,y,\pi).$$

如何计算这个和? 我们可以再次使用标准 HMM 算法, 例如第3章中提到的**前向**算法. 这种用来计算成对 HMM 的方法是, 我们可以再次使用曾用于寻找最大计分联配的相同动态规划思想, 但是在每步中以求和替代求最大值. 前向算法的概率形式如下, 令 $f^k(i,j)$ 为所有结束于 (i,j) 且终止于状态 k 的联配之联合概率. 与以前一样, 我们只为图 4.2 所示的全局模型给出该算法; 此算法也可以直接推广到其他类型的成对联配模型, 例如上面提到的局部模型.

算法: 成对 HMM 的前向算法

起始:

$f^{\mathrm{M}}(0,0) = 1, f^{\mathrm{X}}(0,0) = f^{\mathrm{Y}}(0,0) = 0.$

所有的$f^{\bullet}(i,-1), f^{\bullet}(-1,j)$都设为0.

```
HBA_HUMAN  KVADALTNAVAHVD-----DMPNALSALSDLH
           KV   + +A  ++          +L+ L+++H
LGB2_LUPLU KVFKLVYEAAIQLQVTGVVVTDATLKNLGSVH

HBA_HUMAN  KVADALTNAVAHVDDM-----PNALSALSDLH
           KV   + +A  ++          +L+ L+++H
LGB2_LUPLU KVFKLVYEAAIQLQVTGVVVTDATLKNLGSVH

HBA_HUMAN  KVADALTNA-----VAHVDDMPNALSALSDLH
           KV   + +A     V  V     +L+ L+++H
LGB2_LUPLU KVFKLVYEAAIQLQVTGVVVTDATLKNLGSVH
```

图 4.4: 空位位置不确定性的例子: 来自图 2.1 (b) 的球蛋白联配中三种计分高度相似, 但却明显不同的空位放置方法.

迭代: 除$(0,0)$以外, $i = 0,\ldots,n, j = 0,\ldots,m$:

$$\begin{aligned} f^{\mathrm{M}}(i,j) &= p_{x_i y_j}[(1-2\delta-\tau)f^{\mathrm{M}}(i-1,j-1)+ \\ &\qquad (1-\varepsilon-\tau)(f^{\mathrm{X}}(i-1,j-1)+f^{\mathrm{Y}}(i-1,j-1))]; \\ f^{\mathrm{X}}(i,j) &= q_{x_i}[\delta f^{\mathrm{M}}(i-1,j)+\varepsilon f^{\mathrm{X}}(i-1,j)]; \\ f^{\mathrm{Y}}(i,j) &= q_{y_j}[\delta f^{\mathrm{M}}(i,j-1)+\varepsilon f^{\mathrm{Y}}(i,j-1)]. \end{aligned}$$

终止:

$$f^{\mathrm{E}}(n,m) = \tau[f^{\mathrm{M}}(n,m)+f^{\mathrm{X}}(n,m)+f^{\mathrm{Y}}(n,m)]. \qquad \triangleleft$$

现在, 我们可以考虑作为结果的全概率 $P(x,y) = f^{\mathrm{E}}(n,m)$ 与式 (4.2) 给出的零模型概率间的对数几率比值. 这就是相对于无关而言, 两条序列通过某种联配相互关联之似然的度量. 在这个算法中我们没有假设任何特定联配. 当然, 如果存在一个明显的最好联配, 那么总和中几乎所有概率都由与该联配相应的路径所贡献. 然而总分值总是比最优联配分值高 (使用相同计分方案), 并且当存在与最优联配相当的可选联配或联配变体时, 其分数差别将更为显著.

全概率的一个重要用途是: 给定一对序列 x, y, 对联配 π 定义后验分布 $P(\pi|x,y)$. 它由下式给出,

$$P(\pi|x,y) = \frac{P(x,y,\pi)}{P(x,y)}. \tag{4.3}$$

在 (4.3) 中, 如果令 π 为 Viterbi 路径 π^*, 那么我们就可以得到根据 Viterbi 路径 $v^{\mathrm{E}}(n,m)/f^{\mathrm{E}}(n,m)$ 的后验概率, 这个后验概率可以被解释为最优计分联配"正确"的概率. 通常来说, 这个概率非常的小! 例如对图 2.1 (b) 中 α 球蛋白和豆血红蛋白的联配, 其后验概率为 4.6×10^{-6}. 虽然如果我们希望标准联配算法能找到"正确"的联配,那么这个数字会令人担忧, 但这不足为奇. 最优联配的许多小型变体都具有几乎相同的计分, 或者说它们具有几乎相等的可能性. 特别是当遇到一个空位时, 我们总要选择在哪里设置这个空位; 将空位向左或向右移动一个残基位置通常不会导致变化, 或是仅仅引起一些看上去随机的起伏.

图4.4展示了人类 α 球蛋白和豆血红蛋白序列的对应部分, 给出了体现上述现象的例子. 图中显示的第一个联配接近于经结构确认的联配 (structurally verified alignment), 其分值为 3 (BLOSUM50, 空位开端: –12, 空位延伸: –2 [1]). 下一个联配计分相同, 虽然空位偏移了两个位置. 第三个分值为 6, 虽然空位被错移了 5 个位置. 分值相差3对应着按照联配模型的相对似然增加两倍, 因为 BLOSUM50 分数的单位是 1/3 比特. 很明显在此情况下, 简单序列联配并不是确定联配的精确方法, 因为联配结果普遍被认为是高度分化的.

练习

[1] 原文为gap-extend minus –2, 可能有误. ——译注

4.3 如图 4.4 所示的空位位置变化的相对分数仅依赖于替换分值, 而不是空位分值. 这是为什么? 使用动态规划算法对联配的精确度有什么重要意义?

4.3 次优联配

既然许多联配与最优联配的概率基本相等 (或更一般地, 分数基本相等), 那么我们很自然地就会对它们产生兴趣. 这些联配称为**次优** (suboptimal) 联配. 有许多不同方法可以考察并表征次优联配. 首先, 我们要认真考虑我们期望找到什么.

有一类与最优分数相近的联配, 是上面提到的与最优联配只相差几个位置的联配 (例如图4.4所示). 因为联配中不同位置的微小差别可以独立地组合, 所以这些局部变体的数目随其分值与最优分值间的差别成指数增长. 因此要得到所有的变体是不切实际的. 但是随着联配位置的变化, 联配变化的弹性 (flexibility) 实质上也会发生变化. 存在呈现典型变体的抽样方法, 另外一些方法则可表示动态规划矩阵中的每个单元与它在联配中的"接近" 程度. 下面给出这两种方法的例子.

另一种形式的次优联配与最优联配本质不同, 或者可能完全不同. 当我们怀疑正确联配形式多于一种的时候, 例如一条或两条序列中存在重复, 就可以使用寻找这种次优联配的方法. 一般说来, 当寻找局部联配时这种方法更为恰当, 因为此时只需要联配每条序列中的一部分.

联配的概率论抽样

首先我们给出一种方法, 从4.3定义的后验分布中对联配进行抽样. 请回忆, 该方法根据联配在模型下是正确的 (being correct) 的似然, 为两条序列间的每种可能联配给出概率. 这些样本的总体将给出联配信息的描述, 而我们可以从给定序列对中确信无误地获得这类信息. 如 HMM 后验解码的章节(第 p. 42 页) 所述, 任何感兴趣的特定性质都可以通过对样本取平均而估计得到. 当联配的细节不确定时, 对使用相似性信息而言, 这是一种强有力的通用策略; 例如, 本书后面的第8章将会用到这种策略.

为产生一个样本联配, 我们在 $f^k(i,j)$ 值的矩阵中回溯, 但是在每步中不选择最高分, 而是基于三个部分的相对功效做出概率选择. 为进一步阐明这种方法, 我们假设现在处于回溯中状态M 的位置 (i,j), 称之为单元 $\mathrm{M}(i,j)$. 由前向算法可得:

$$f^{\mathrm{M}}(i,j) = p_{x_i y_j}[(1-2\delta-\tau)f^{\mathrm{M}}(i-1,j-1)+ \\ (1-\varepsilon-\tau)(f^{\mathrm{X}}(i-1,j-1)+f^{\mathrm{Y}}(i-1,j-1))].$$

在下一步

$$\text{以概率}\frac{p_{x_i y_j}(1-2\delta-\tau)f^{\mathrm{M}}(i-1,j-1)}{f^{\mathrm{M}}(i,j)}\text{选择}\mathrm{M}(i-1,j-1),$$
$$\text{以概率}\frac{p_{x_i y_j}(1-\varepsilon-\tau)f^{\mathrm{X}}(i-1,j-1)}{f^{\mathrm{M}}(i,j)}\text{选择}\mathrm{X}(i-1,j-1),$$
$$\text{以概率}\frac{p_{x_i y_j}(1-\varepsilon-\tau)f^{\mathrm{Y}}(i-1,j-1)}{f^{\mathrm{M}}(i,j)}\text{选择}\mathrm{Y}(i-1,j-1).$$

单元 $\mathrm{X}(i,j)$ 中相应的分布为

$$\text{以概率}\frac{q_{x_i}\delta f^{\mathrm{M}}(i-1,j)}{f^{\mathrm{X}}(i,j)}\text{选择}\mathrm{M}(i-1,j),$$
$$\text{以概率}\frac{q_{x_i}\varepsilon f^{\mathrm{X}}(i-1,j)}{f^{\mathrm{X}}(i,j)}\text{选择}\mathrm{X}(i-1,j),$$

对于单元 $\mathrm{Y}(i,j)$ 亦有类似的结果.

从我们的简单示例数据出发, 得到一个全局联配的样本集合如下:

```
HEAGAWGHEE   HEAGAWGHE-E   HEAGAWGHE-E
-P-A-WHEAE   -PA--W-HEAE   -P--AW-HEAE

HEAGAWGHEE    HEAGAWGHEE    HEAGAWGHEE
P---AWHEAE    -P--AWHEAE    --PA-WHEAE
```

由上可见, 当必须出现空位并且联配证据较弱时, 可选联配的可能性就更大, 正如以上联配中序列开始处的情形. 对分数贡献较多的配对, 如两个W, 或成块出现的配对, 如序列末尾处的情形, 都更为稳定. 这些样本中配对出现的频率可以做为该配对在联配中可靠性的一个自然指标. 下面我们将给出直接计算此频率期望值的方法, 即根据这个模型给出任意特定残基对应该联配上的概率.

在本书后面关于构建多序列联配的章节 (第 5 章) 中, 我们将使用与此处相同的抽样方法.

寻找截然不同的次优联配

如前所述, 有许多不同的方法可以找到与最优联配明显不同的联配变体. 其中有一种方法使用第2章中介绍的"重复" (repeat) 算法. 该方法可找到一条序列与另一序列中多个不交叠片断间计分最高匹配的最优集合. 但是该方法并不适用于目前的问题, 因为它按照不同的方式处理这两条序列. 而且, 最好的那条联配甚至有可能不在该集合中.

Waterman & Eggert [1987] 给出了使用最为广泛的寻找截然不同次优联配的方法,他们的算法可以找到次好 (the next best) 联配, 而该次好联配同任何已确定的联配均无共用联配残基对. 一旦取得最好的匹配, 该算法就会重新计算标准 (Viterbi) 动态规划矩阵, 同时在递归中加入一个步骤, 即将对应于包含在最好匹配中之残基对的单元清零, 以削除它们对下一个联配的贡献. 所得的矩阵和分数就包括了第二最优联配的信息. 这个过程可以不断重复, 为每次循环前已经得到的匹配将所有单元清零, 直至下一个分值小于 T (见图 4.5). 实际上如果矩阵存储于内存中, 则每次迭代时并不需要重新计算整个矩阵: 我们可以使用标记过程 (marking procedure) 来表示哪些单元需要更新. 4.6 节给出了其他寻找次优联配方法的相关文献.

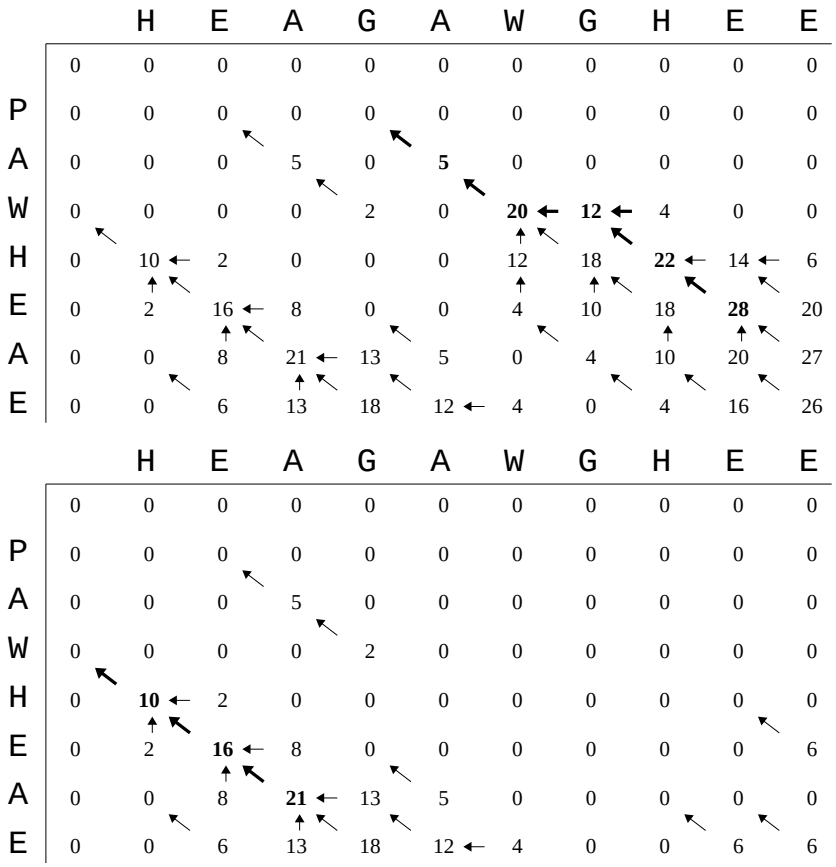


图 4.5: 应用于我们标准数据例子的 Waterman-Eggert 算法. 上图: 与图 2.6 完全一样的标准局部联配矩阵. 下图: 最好局部匹配已经被清零, 以便找到次好联配.

4.4 x_i 联配上 y_j 的后验概率

如果任何一条完全路径是完全正确的概率都很小, 我们能够对联配局部精确度吗? 通常, 我们得到的联配只有一部分十分明确, 而其他区域则不太确定. 保守性程度的变化依赖于结构及功能的限制, 于是核心序列会十分保守, 而回折区域 (loop region) 的联配就不是很可靠. 在这种情况之下, 为联配的每一部分都给出可靠性度量将会非常有用.

我们可以利用 HMM 的形式体系实现上述设想. 想法是: 计算通过一个特定匹配残基对 (x_i, y_j) 的所有联配之联合概率. 然后求该值与这对序列的所有联配之全概率 (由上节计算得到) 间的比值. 如果该比值接近 1, 则我们可以认为该匹配高度可靠; 若接近 0, 则该匹配不可靠. 用于此用途的该方法与第 3 章中为后验解码给出的算法十分相关.

我们引入一个新的记号 $x_i \diamond y_j$, 意为 x_i 联配上 y_j. 于是由标准的条件概率理论, 我们有:

$$\begin{aligned} P(x, y, x_i \diamond y_j) &= P(x_{1\ldots i}, y_{1\ldots j}, x_i \diamond y_j) P(x_{i+1\ldots n}, y_{j+1\ldots m} | x_{1\ldots i}, y_{1\ldots j}, x_i \diamond y_j) \\ &= P(x_{1\ldots i}, y_{1\ldots j}, x_i \diamond y_j) P(x_{i+1\ldots n}, y_{j+1\ldots m} | x_i \diamond y_j) \end{aligned}$$

其中第一项是通过前向算法计算得到的前向概率 $f^{\mathrm{M}}(i, j)$. 第二项是由相应的后向算法计算得到的相应后向概率 $b^{\mathrm{M}}(i, j)$.

算法: 成对 HMM 的后向计算 起始:

$$b^{\mathrm{M}}(n, m) = b^{\mathrm{X}}(n, m) = b^{\mathrm{Y}}(n, m) = \tau.$$

所有的 $b^\bullet(i,m+1), b^\bullet(n+1,j)$ 都设为0.

递归: 除了 (n,m), 对于所有的 $i=n,\ldots,1, j=m,\ldots,1$:

$$\begin{aligned} b^{\mathrm{M}}(i,j) &= (1-2\delta-\tau)p_{x_{i+1}y_{j+1}}b^{\mathrm{M}}(i+1,j+1) \\ &\quad + \delta[q_{x_{i+1}}b^{\mathrm{X}}(i+1,j) + q_{y_{j+1}}b^{\mathrm{Y}}(i,j+1)]; \\ b^{\mathrm{X}}(i,j) &= (1-\varepsilon-\tau)p_{x_{i+1}y_{j+1}}b^{\mathrm{M}}(i+1,j+1) + \varepsilon q_{x_{i+1}}b^{\mathrm{X}}(i+1,j); \\ b^{\mathrm{Y}}(i,j) &= (1-\varepsilon-\tau)p_{x_{i+1}y_{j+1}}b^{\mathrm{M}}(i+1,j+1) + \varepsilon q_{y_{j+1}}b^{\mathrm{Y}}(i,j+1). \end{aligned}$$ ◁

本算法不需要特别指定终止步骤, 因为我们只想求 $i,j\geq 1$ 时的 $b^{\cdot}(i,j)$ 值.

现在, 我们可以通过 Bayes 法则得到:

$$P(x_i\diamond y_j|x,y) = \frac{P(x,y,x_i\diamond y_j)}{P(x,y)},$$

我们也可以为使用特定插入状态的后验概率获得相似的值. 图 4.6 给出了该流程用于第 2 章中的示例序列所得的结果.

Miyazawa [1994] 也描述了本质上与之相同的方法, 并且他进一步定义了所谓的"概率联配" (probability alignment). 通过对每个 i 找到使 $P(x_i\diamond y_j)$ 最大化的 j, 可以定义一个 x 到 y 的联配, 这看上去很诱人(从现在开始我们省略条件 x,y, 因为该条件一直存在). 但是, 这样做并不保证能产生形式完好的联配; 它可能包含了与原序列顺序不一致的已联配位置对 $(i_1,j_1),(i_2,j_2)$, 也就是说可能有: $i_2>i_1$ 并且 $j_1<j_2$. Miyazawa 指出, 如果我们将自己限制为只寻找使 $P(x_i\diamond y_j)>0.5$ 的位置对 i,j, 那么该方法将总能得到与原序列顺序一致的联配, 并且每个 x_i 最多联配上一个 y_j. 联配明确的区域均满足此条件. 另一方面, 在联配不明确的区域, 例如远距离相关蛋白的相应回折区域, 在没有强烈证据表明特定残基对能够联配的位置上, 两条序列中都会插入空位.

联配的期望精确度

Miyazawa 的方法通常可以给出不完全联配, 因为可能有不满足 $P(x_i\diamond y_j)>0.5$ 的显著区域. 尽管这可能就是我们所需要的区域,但是我们也可以用后验匹配概率得到拥有全局最大精确度的完全联配,下面给出大体介绍. 首先, 注意到我们可以先计算给定联配 π 与从后验分布抽样得到的路径间的期望交叠数 $\mathscr{A}(\pi)$. 它与 π 中正确匹配的期望个数相等, 也就是对 π 之总体精确度的一个自然度量.

$$\mathscr{A}(\pi) = \sum_{(i,j)\in\pi} P(x_i\diamond y_j),$$

上式对 π 中所有联配上的位置对进行连加. 对于图 2.1 (b) 中 α 球蛋白和豆血红蛋白的联配, $\mathscr{A}(\pi)=16.48$, 或者说每个联配上的残基平均为0.40.

已知联配的这种新型分数之后, 我们是否可以找到两条序列间精确度最高的联配? 或许该方法并不能给出在区别两条序列是否相关方面判别能力最强的分数, 但是如果两条序列相关, 那么我们仍然希望利用它得到更精确的联配. 所需方法惊人地简单. 我们运行不考虑空位损失的标准动态规划方法, 其中的分值来自于成对匹配的后验概率. 递归方程如下:

$$A(i,j) = \max\begin{cases} A(i-1,j-1)+P(x_i\diamond y_j), \\ A(i-1,j), \\ A(i,j-1), \end{cases} \tag{4.4}$$

并且, 标准回溯流程可以给出最优联配. 显然该过程将优化合法联配中 $P(x_i\diamond y_j)$ 项的和. 有趣的是, 同样的算法还可以用于任何类型的空位分值; 随分值不同而变化的仅仅是 $P(x_i\diamond y_j)$ 的值本身, 这个分数可以通过以上介绍的计分方案特异的 (scoring scheme-specific) 标准动态规划过程得到.

图 4.6 展示了第 2 章中作为例子出现的短序列之最优精确度路径. 注意这里的路径与最有可能路径或 Viterbi 路径不同. 显然, 短序列开头的 P 优先联配长序列的 E, 而不是 A, 虽然 P 联配上 E 和 A 的分数都一样. 直观看来, 其原因是对放置随后的空位而言, 联配上 E 后可以有更多的选择.

Match

		H	E	A	G	A	W	G	H	E	E
	87	0	0	0	0	0	0	0	0	0	0
P	0	24	**36**	18	7	0	0	0	0	0	0
A	0	0	2	26	15	**43**	0	0	0	0	0
W	0	0	0	0	0	0	**85**	1	0	0	0
H	0	0	0	0	0	0	0	12	**73**	0	0
E	0	0	0	0	0	0	0	1	8	**65**	0
A	0	0	0	0	0	0	0	0	1	21	0
E	0	0	0	0	0	0	0	0	0	0	**86**

X insert

		H	E	A	G	A	W	G	H	E	E
	0	**62**	26	7	0	0	0	0	0	0	0
P	0	0	22	**32**	**36**	0	0	0	0	0	0
A	0	0	0	2	28	42	0	0	0	0	0
W	0	0	0	0	0	0	0	**72**	0	0	0
H	0	0	0	0	0	0	0	0	3	0	0
E	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	0	0	0	1	0
E	0	0	0	0	0	0	0	0	0	0	2

Y insert

		H	E	A	G	A	W	G	H	E	E
	0	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	0	0	0	0	0
W	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	1	0	0	0	0
E	0	0	0	0	0	0	0	0	12	0	0
A	0	0	0	0	0	0	0	0	0	**64**	0
E	0	0	0	0	0	0	0	0	0	0	10

图 4.6: 第 2 章中示例数据的后验概率. 这三个表格给出了在每个位置 (i, j) 处分别使用状态 M、X 和 Y 的后验概率. 表中给出的数值是百分数, 即相应概率乘以 100 后四舍五入的结果. 表中所指明的路径是 (4.4) 式意义下的最优精确路径.

4.5 用于搜索的成对 HMM 与 FSA 之对比

概率论建模方法的一个优势在于, 如果数据 D 对应于模型 M 中的样本, 则在数据无限多的极限条件下, 似然对 M 取其最大值, 即 $P(D|M) > P(D|\tilde{M})$, 其中 $\tilde{M}$ 为任意其他模型. 特别地, 如果 M 有一组参数, 例如 HMM 的转移及发射概率, 那么当把与样本相应的参数值赋予模型时, 数据的似然将被最大化.

于是, 如果一个成对 HMM 的参数能够很好的描述相关序列对的统计特性, 那么我们就应该用带有这些参数值的模型来搜索. 如果还存在一个模型 R 能够很好地描述随机序列的产生, 那么将 M 和 R 进行 Bayes 模型比较将是一个合适的过程(本书第 2 章第 p. 25 页). 按照这个想法, 我们应该在搜索时使用概率论模型. 但是, 当前使用的大多数算法(第 2 章) 都无法满足这个要求, 原因有二. 首先, 它们并不计算序列对的、通过将所有联配求和得到的全概率 $P(x, y|M)$, 而是找到最优匹配或 Viterbi 路径. 其次作为 FSA, 它们的参数并不能直接转化为概率值.

首先考虑使用 $Viterbi$ 路径所带来的影响. 在这种情况下我们可以看到, 其参数能匹配数据的模型并不一定是最优搜索模型. 图 4.7 给出了一个简单的 HMM . 状态 S 以概率 q_a 产生字符; S 以概率 α 转移到自身, 同时以概率 $1-\alpha$ 转移到状态的连续块 B, 这些状态在回到初始状态前发射长度为 4 的固定字符串 abac. 从 S 发射 abac 的概率为 $P_S(\texttt{abac}) = \alpha^4 q_\texttt{a} q_\texttt{b} q_\texttt{a} q_\texttt{c}$, 而从状态 B (始于 S) 发射 abac 的概率为 $1-\alpha$. 如果 $P_S(\texttt{abac}) > 1-\alpha$, 则任何数据集的最可能路径都使用 S, 因为向 B 的转移几乎不可能. 尽管如此, 如果数据中字符串 abac 出现的次数多于期望, 那么它就可用于区分该模型与按概率 q_a 发射字符之随机模型间的差别. 使用最优匹配而不是总概率所进行的模型间比较将不能检测数据来源, 甚至当数据集很大时也是如此. 我们可以改变参数以便在某种程度上修正这种不足. 例如, 如果到 B 的转移概率增长到 τ, 其中 $\tau > P_S(\texttt{abac})$, 那么该模型就将可以检测这些序列的类型. 但是此时每个 abac 都被归类为来自 B, 这个结果也不正确.

现在考虑这个问题, 将一个用于二序列联配的 FSA 转化为概率论模型. 图4.8 (a) 表示了局部匹配的 FSA ; 它包含以 0 损失发射未配对序列的初始和终止状态. 因为未配对序列的长度不定, 并且概率论模型总是对每次发射赋予非零损失, 所以不存在固定的比例变换过程将该模型的分数转化成 HMM 的对数概率. 另一方面, 如果我们使用 Bayes 模型比较法, 并定义一个随机模型 R, 它发射一条未配对序列的概率与局部联配模型 M 在初始及终止的未联配区域所使用概率相同, 那么该未配对序列的对数几率值将为零. 于是, 我们可以找到两个成对 HMM, 它们的对数几率比值与 FSA 分数吻合, 例如图4.8 (b). 注意, 这里的转移概率并不十分确切, 因为它们意味着非常短的序列. 但是为 FSA 所假设的参数却运行良好. 基于此, 我们猜想标准参数已经根据经验被设定为"下意识地" 给用作搜索方法的 Viterbi 之缺点提供补偿, 而这种缺点与图 4.8 中简单例子所展示的相同. 从这点我们可以推测, 如果将 Viterbi 用于数据库搜索, 那么概率论模型的性能就比不上标准联配算法, 但是如果

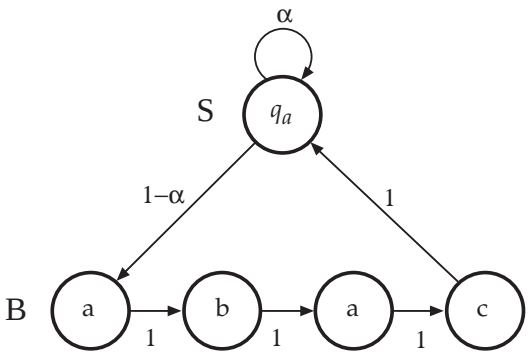


图 4.7: 以概率 q_a 从 S 发射序列, 并且从下面四个状态组成的块 B 发射字符串 abac的 FSA. 如果转移到 B 的概率很低, 那么最有可能路径就永远不可能通过 B , 即使序列包含模体 abac.

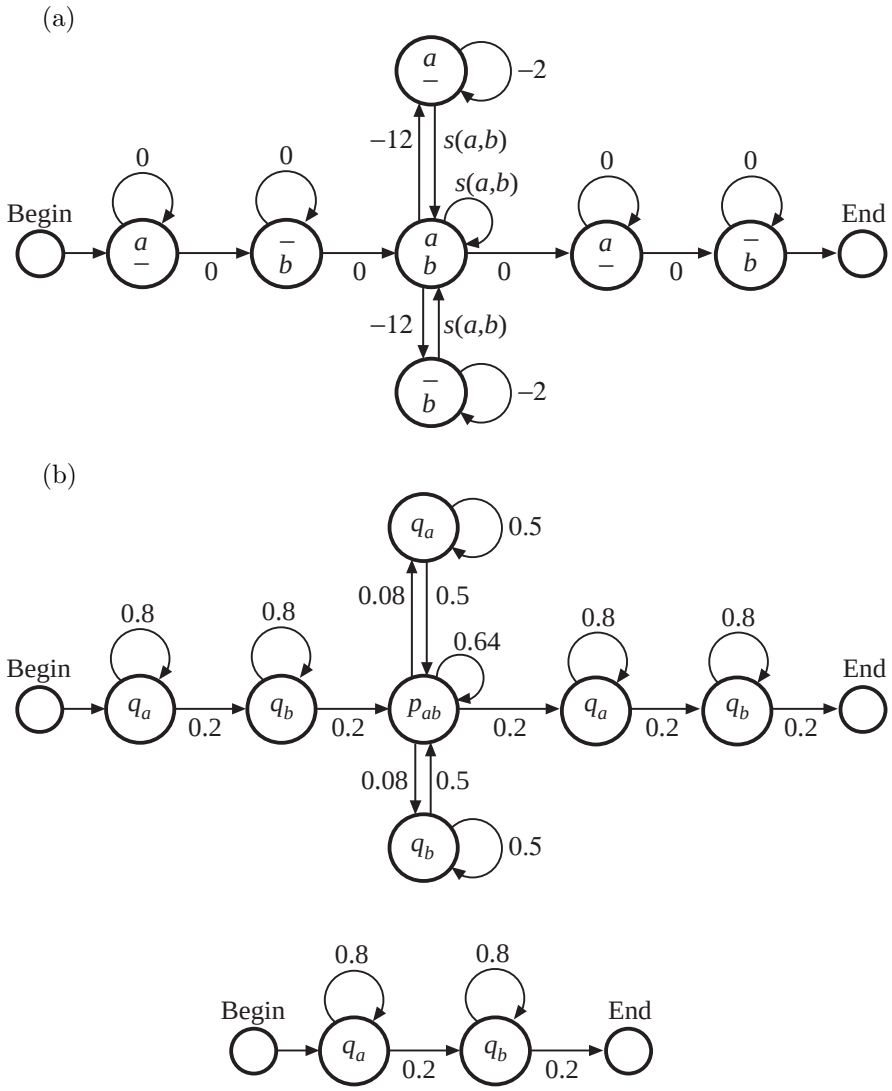


图 4.8: (a) 计算局部匹配算法的FSA . $s($ a , b) 使用BLOSUM50矩阵的分值. (b) 两个 HMM, 一个已联配序列的模型 (上图) 和一个随机模型 (下图), 它们的对数几率比分值同 (a) 中 FSA 的分值相同. 概率 p_{ab} 和 q_a 用于定义 $BLOSUM50$ 矩阵.

将前向算法用于提供独立于特定联配的完全分数, 那么和成对 HMM 相似的概率论模型也许会改进标准方法.

练习

4.4 证明使用图 4.7 中例子的全概率论模型可以区分出模型数据和随机数据.

4.5 将其与在模型中使用 Viterbi 路径进行比较, 该模型中到B 的转移概率增长到 τ , 其中 $\tau > P_S(\texttt{abac})$.

4.6 我们可以进一步改进模型: 将所有在 S 的发射概率都设为 $1/A$, 其中 A 是字母表长度. 这个模型与带相同发射概率的随机模型间的差别正好是数据中字符串 `abac` 的个数. 它是否具备与全概率论模型相同的辨别能力?

4.6 补充读物

虽然我们这里给出的根据成对 HMM 的二序列联配形式并不标准, 但是几位作者都已经考虑了一个与之等价的全概率论模型. Bucher & Hofmann [1996] 讨论了如何使用经配分函数 (partition function) 归一化后的局部概率论模型进行搜索. Bishop & Thompson [1986] 引入了进化分析框架下的一个相关模型, 最近 Thorne , Kishino & Felsenstein [1991; 1992] 在这方面有所发展, 他们为 DNA 序列含空位联配的概率论模型得到了参数估计的方法. 在第 8 章中, 我们将进一步讨论由进化研究所推动的模型.

Zuker [1991] 和 Barton [1993] 介绍了不同于 Waterman & Eggert [1987] 的寻找次优联配的方法. Mevissen & Vingron [1996] 给出了另一种方法, 用于量化动态规划联配的可靠性, 并且 Vingron [1996] 的最近综述回顾了寻找以及评估次优联配显著性的方法.

第五章

用于序列家族的 profile HMM

目前为止, 我们只关注单条序列的内部属性, 例如 DNA 中的 CpG 岛, 或二序列联配. 然而有生物功能的序列通常来自序列家族, 而且许多功能强大的序列分析方法都基于确定单条序列与序列家族间的关系. 一个家族中的序列会在进化过程中在一级序列上彼此分化, 它们或者通过基因组复制 (duplication) 而分离, 或者通过产生亲缘生物体内相应序列的物种形成 (speciation) 过程而分离. 一般来讲, 两种情况下它们均保持着相同或相关的功能. 因此, 认定一条序列确实属于某家族, 并将之与其他成员进行联配, 常可推断其功能。

如果已有属于某家族中的一组序列, 那么我们便可以使用数据库搜索的方法寻找更多家族成员, 即将已知的家族成员之一作为查询序列进行二序列联配. 更彻底办法是, 可以一条接一条地对所有已知成员序列进行搜索. 但是用家族的任何一条已知序列进行成对搜索可能找不到已有序列的远缘序列. 一种替代方法是在搜索中使用整个序列集的统计特性. 与此相似, 即使家族成员关系明确, 集中考虑整个家族的保守特征也可以显著改善精确联配的结果.

简单说来, 我们如何才能确定这些特征呢? 就像二序列联配能获取两条序列间的大量关系一样, 多序列联配可以展现出一个家族中序列间的相互关系. 图5.1给出了庞大的球蛋白 (globin) 家族中7条序列的多序列联配结果(某些蛋白质数据库已经可提供几百条此类球蛋白序列). 图中所示联配中每条蛋白质的三维结构都已知, 通过对齐球蛋白保守折叠的 8 个 α-螺旋以及序列中某些关键残基, 这些序列已联配好. 关键残基包括与球蛋白活性位点 (active site) 处氧结合亚铁血红素辅基 (oxygen-binding heme prosthetic group) 相互作用的两个保守组氨酸 (histitine , `H`) 残基.

显然与其他位置相比, 球蛋白联配中的某些位置具有更强的保守性. 一般来讲, 螺旋结构比它们间的回折区域(loop region) 更保守, 并且某些残基也极其保守. 在识别一条新的序列是否为球蛋白时, 需要检查该序列是否包含这些较保守的特征. 本章将具体阐述如何获取并利用这些特征信息.

可以预料, 我们对一致序列建模的方法是建立概率论模型. 特别地, 我们要引入十分适合为多序列联配建模的一种特殊隐 Markov 模型. 我们参照**profile**的标准叫法将其命名为 **profile HMM**, 前者是与后者关系密切的非概率论结构, 由 Gribskov, McLachlan & Eisenberg [1987] 此前为同样目的而引入. Profile HMM 可能是当前分子生物学界应用最为广泛的隐马模型 [Eddy 1996].

基于本章需要, 假设一个正确的多序列联配已给定, 我们可以据此建立模型以便找出新序列的潜在匹配并为这些匹配计分. 多序列联配可以根据结构信息创建, 例如这里提到的球蛋白家族的例子; 或来自基于序列的联配处理过程, 第6章将讨论这部分内容.

本章大部分内容利用第3章的一般 HMM 理论. 对于一些最重要的算法, 我们将重新呈现其与 profile HMM 相关的特定形式. 最后, 本章还会广泛探讨如何从多序列联配出发估计最优概率参数.

```
Helix                 AAAAAAAAAAAAAAAA   BBBBBBBBBBBBBBBBCCCCCCCCCCCC
HBA_HUMAN  ---------VLSPADKTNVKAAWGKVGA--HAGEYGAEALERMFLSFPTTKTYFPHF
HBB_HUMAN  --------VHLTPEEKSAVTALWGKV----NVDEVGGEALGRLLVVYPWTQRFFESF
MYG_PHYCA  ---------VLSEGEWQLVLHVWAKVEA--DVAGHGQDILIRLFKSHPETLEKFDRF
GLB3_CHITP ----------LSADQISTVQASFDKVKG------DPVGILYAVFKADPSIMAKFTQF
GLB5_PETMA PIVDTGSVAPLSAAEKTKIRSAWAPVYS--TYETSGVDILVKFFTSTPAAQEFFPKF
LGB2_LUPLU --------GALTESQAALVKSSWEEFNA--NIPKHTHRFFILVLEIAPAAKDLFS-F
GLB1_GLYDI ---------GLSAAQRQVIAATWKDIAGADNGAGVGKDCLIKFLSAHPQMAAVFG-F
Consensus            Ls....  v a W kv .  .    g . L.. f . P .   F  F

Helix          DDDDDDDEEEEEEEEEEEEEEEEEEE            FFFFFFFFFFFF
HBA_HUMAN  -DLS-----HGSAQVKGHGKKVADALTNAVAHV---D--DMPNALSALSDLHAHKL-
HBB_HUMAN  GDLSTPDAVMGNPKVKAHGKKVLGAFSDGLAHL---D--NLKGTFATLSELHCDKL-
MYG_PHYCA  KHLKTEAEMKASEDLKKHGVTVLTALGAILKK----K-GHHEAELKPLAQSHATKH-
GLB3_CHITP AG-KDLESIKGTAPFETHANRIVGFFSKIIGEL--P---NIEADVNTFVASHKPRG-
GLB5_PETMA KGLTTADQLKKSADVRWHAERIINAVNDAVASM--DDTEKMSMKLRDLSGKHAKSF-
LGB2_LUPLU LK-GTSEVPQNNPELQAHAGKVFKLVYEAAIQLQVTGVVVTDATLKNLGSVHVSKG-
GLB1_GLYDI SG----AS---DPGVAALGAKVLAQIGVAVSHL--GDEGKMVAQMKAVGVRHKGYGN
Consensus   .  t    .. . v..Hg kv. a   a...l   d   . a l. l   H  .

Helix       FFGGGGGGGGGGGGGGGGGG     HHHHHHHHHHHHHHHHHHHHHHHHHH
HBA_HUMAN  -RVDPVNFKLLSHCLLVTLAAHLPAEFTPAVHASLDKFLASVSTVLTSKYR------
HBB_HUMAN  -HVDPENFRLLGNVLVCVLAHHFGKEFTPPVQAAYQKVVAGVANALAHKYH------
MYG_PHYCA  -KIPIKYLEFISEAIIHVLHSRHPGDFGADAQGAMNKALELFRKDIAAKYKELGYQG
GLB3_CHITP --VTHDQLNNFRAGFVSYMKAHT--DFA-GAEAAWGATLDTFFGMIFSKM-------
GLB5_PETMA -QVDPQYFKVLAAVIADTVAAG---------DAGFEKLMSMICILLRSAY-------
LGB2_LUPLU --VADAHFPVVKEAILKTIKEVVGAKWSEELNSAWTIAYDELAIVIKKEMNDAA---
GLB1_GLYDI KHIKAQYFEPLGASLLSAMEHRIGGKMNAAAKDAWAAAYADISGALISGLQS-----
Consensus    v.   f  l . ..  ....    f   . aa. k. .     l sky
```

图 5.1: Bashford, Chothia & Lesk [1987] 中 7 条球蛋白的多序列联配. 左边一列是 SWISS-PROT 数据库 [Bairoch & Apweiler 1997] 中的蛋白质标识符. 联配上方给出了从A到H共8个α-螺旋. 联配下方给出了一致性序列, 如果同一位置上7条序列中至少6条有相同的残基则用大写字母表示, 4条或5条有相同的残基用小写字母表示, 3条的用圆点表示.

5.1 无空位计分矩阵

从图5.1中可以看出, 蛋白质家族多序列联配的一个普遍特征就是空位倾向于彼此对齐, 而留出在任一序列中都不包含插入或删除的完整块 (solid blocks). 我们先考虑关于这些无空位区域的模型.

作为例子, 请考查图5.1中的螺旋 E. 对这种区域来讲, 一个自然的概率论模型就是为位置 i 处观测到氨基酸 a 指定独立的概率 $e_i(a)$ (使用字母 e 的原因是, 引入空位后它们就是隐马模型中的**发射概率** (emission probability)). 根据这个模型, 一条新序列 x 的概率是

$$P(x|M) = \prod_{i=1}^{L} e_i(x_i),$$

其中 L 是块的长度, 此例子中为21. 事实上, 我们通常对这个概率与 x 在随机模型下概率的比值更感兴趣, 因此为了检测家族中的成员资格我们会评估对数几率比

$$S = \sum_{i=1}^{L} \log \frac{e_i(x_i)}{q_{x_i}}.$$

$\log \frac{e_i(a)}{q_a}$ 的工作方式与计分矩阵 $s(a,b)$ 中的元素一样, 但是其中第二个指标是位置 i 而非氨基酸 b. 出于这个原因, 此方法被称作**位置特异计分矩阵** (position

specific score matrix , PSSM). PSSM 可用于在长度为N的更长序列 x 中, 通过评估序列 x 中从 1 到 $N - L + 1$ 的每个开始点 j 的分值 S_j 而搜索匹配, 其中 L 是 PSSM 的长度.

5.2 添加插入与删除状态以获得 profile HMM

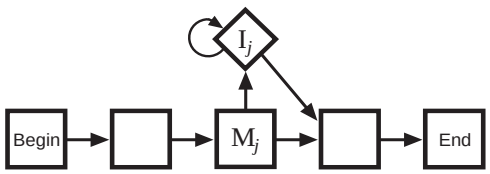
虽然 PSSM 可以获取某些保守信息, 但它显然不足以反映一个蛋白家族多序列联配的所有信息. 我们不得不寻找某种方法以便考虑空位. 可以将多个无空位块模型的分值组合起来, 这也正是 Henikoff & Henikoff [1991] 的BLOCKS 数据库所采用的方法. 然而, 在这里我们会为整个联配着手开发单一的概率论模型.

有一种方法是在联配的每个位置都允许空位, 像二序列联配中一样, 在每个位置都使用相同的空位分值 $\gamma(g)$. 但这样做也忽略了一些信息, 因为联配已经明确地指示出哪里更容易出现空位, 哪里不容易出现. 我们想获取这些信息以得到对位置敏感的空位分值, 就像我们从发射概率中得到对位置敏感的替换分值一样.

我们采用的方法是构造一个隐马模型 (HMM), 其中包含各个状态的重复结构, 但每个位置的概率不同. 这样就为序列家族中的成员提供了一个全概率论模型. 首先我们注意到 PSSM 可以看作是带一系列等同状态的平凡 (trivial) HMM, 我们称这些状态之为**匹配** (match) 状态, 它们被概率为1的转移分开.

因为无法选择转移, 所以该联配是平凡的. 我们将匹配状态的发射概率改称为 $e_{\mathrm{M}_i}(a)$.

下一步要处理空位. 我们必须区别对待插入与删除. 为处理插入, 即 x 中并不匹配模型任何符号的部分, 我们引入一组新的状态 I_i: 在匹配多序列联配第 i 列的那个残基之后, 我们将以 I_i 匹配插入. I_i 具有发射分布 $e_{\mathrm{I}_i}(a)$, 但通常将它们设定为背景分布 q_a, 就如同在二序列联配中看到一个未联配的插入残基所做的处理一样. 我们需要从 M_i 到 I_i 的转移, 为容纳多残基插入我们也需要从 I_i 到它本身的环, 并且, 我们还需要从 I_i 回到 M_{i+1} 的转移. 下面展示了单个此类插入状态的情形:

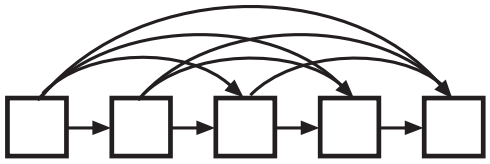


在图中, 我们用菱形代表插入状态. 一个插入的对数几率罚值是相关转移与发射的罚值之和. 如前所述, 假设 $e_{\mathrm{I}_i}(a) = q_a$, 那么发射对对数几率没有贡献, 并且长度为 k 的空位分值为

$$\log a_{\mathrm{M}_j\mathrm{I}_j} + \log a_{\mathrm{I}_j\mathrm{M}_{j+1}} + (k-1)\log a_{\mathrm{I}_j\mathrm{I}_j}.$$

从这里可以看出, 所示的插入状态类型对应于仿射空位计分模型.

对于删除, 即不与任何 x 的残基相匹配的多序列联配片断, 我们可以使用不相邻匹配状态间的向前"跳跃" 转移来处理:



但是, 为了允许长模型取任意长度的空位, 这种方法将需要很多转移. 我们改用

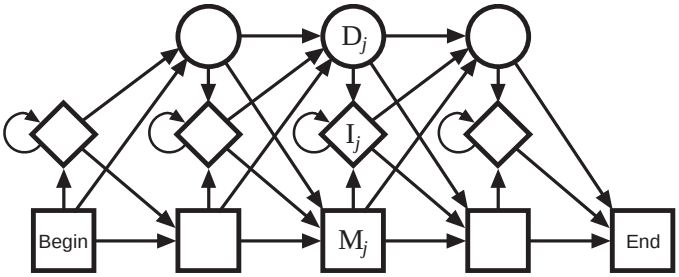
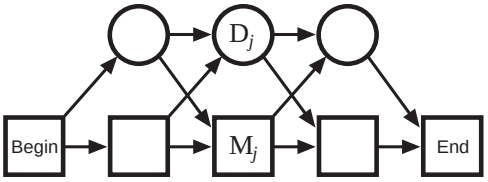


图 5.2: Profile HMM 的转移结构, 图中菱形表示插入状态,圆圈表示删除状态.

第3.4节中曾经介绍过的沉默状态 D_j 作为代替:



因为沉默状态并不发射任何残基, 所以在序列中的两个残基间, 我们可以利用一条沉默状态序列从任意匹配状态出发到达后面的任意匹配状态. 于是一个删除的罚值便是如下转移的罚值之和: 一个 M → D 转移, 后面接着若干 D → D 转移, 然后是一个 D → M 转移. 初看起来这与插入的罚值完全类似, 虽然贯穿模型的路径看上去有所不同. 从细节上讲, D → D 转移有可能具备不同的概率值, 因此对分值的贡献也就不同; 然而一个插入的所有 I → I 转移都包含相同状态, 这也就保证了相同的罚值.

所得完整的 HMM 结构如图5.2所示. 这种形式的模型称为 profile HMM, 它首先由 Haussler *et al.* [1993]和 Krogh *et al.* [1994]提出. 和他们的文章一样, 我们已经向插入与删除状态间添加了转移, 虽然这种情况极少发生. 忽略这些情况并不会对匹配的计分产生很大影响, 但却能在构建模型时制造麻烦.

Profile HMM 能够推广二序列联配

我们已经看到 profile HMM 中所使用的空位状态罚值如何反映带仿射空位的二序列联配中搜使用的那些状态罚值. 为了弄清其间的关系, 考虑退化 (degenerate) 情形将会有所帮助, 其中构造 HMM 的多序列联配只有一条序列.

比较图5.2与图4.2. 令示例序列为 y, 那么图5.2就是图4.2展开后的形式, 其中每个发射 y_i 都来自于一个单独的成对 HMM 副本. 状态 M_j 对应于一条匹配状态 M 的序列, I_j 对应于 X 的具体化, D_j 对应于 Y 的具体化. 为了达到尽可能接近的对应, 匹配的发射概率 $e_{M_i}(a)$ 的自然值就是 $p_{y_i a}/q_{y_i}$, 即成对联配中给定 y_i 时看到 a 的条件概率, 并且对转移概率而言, 对所有 i 有 $a_{M_i I_i} = a_{M_i D_{i+1}} = \delta$ 和 $a_{I_i I_i} = a_{D_i D_{i+1}} = \varepsilon$.

事实上严格来讲, profile HMM 就是由调整图4.2的成对 HMM 得到的隐 Markov 模型, 它将发射序列 y 作为其联配中的序列之一. 正因为如此, 寻找 x 到 profile HMM 的最有可能联配的 Viterbi 方程, 与第4章中寻找x和y到成对 HMM 的最有可能联配的方法在本质上是一致的. 下面会看到, 如果将它们转换为对数几率比形式, 我们就会重新获得式 (2.16) 的标准仿射空位罚值的二序列联配方程. 任何差异都由开始或终止条件的微小差别造成.

5.3 从多序列联配中导出 profile HMM

Profile HMM 与此前的二序列联配使用相同的动态规划方法, 尽管这看上去很好, 但这并不是我们引入它的原因. Profile HMM 背后的关键思想是, 我们可以使用图5.2中的相同结构, 只设置转移与发射概率来获取整个家族的多序列联配中每个位

```
HBA_HUMAN    ...VGA--HAGEY...
HBB_HUMAN    ...V----NVDEV...
MYG_PHYCA    ...VEA--DVAGH...
GLB3_CHITP   ...VKG------D...
GLB5_PETMA   ...VYS--TYETS...
LGB2_LUPLU   ...FNA--NIPKH...
GLB1_GLYDI   ...IAGADNGAGV...
                ***  *****
```

图 5.3: 这十列来自图 5.1 的七条球蛋白序列的多序列联配.带星号的列将被 profile HMM 看作是"匹配" .

置的特定信息. 本质上我们想为这个家族构建表现一致序列的模型, 而不是针对某个特定成员的序列.

有很多不同方法可以从家族中的多序列联配来计算参数值. 为了用实例说明这些模型, 图5.3给出了图5.1中球蛋白联配的一小部分.

非概率论profile

类似于 profile HMM 的模型首先由 Gribskov, McLachlan & Eisenberg [1987]提出, 他们最先使用了"profile" 一词(另见 Gribskov, Lüthy & Eisenberg [1990]). 但是他们并没有构造作为基础的概率论模型, 而是为每个匹配状态和空位惩罚直接指定了位置特异分值, 这样便可应用于标准的"最优匹配" 动态规划中. 他们为每个一致序列位置设定了分值, 它等于相应多序列联配列中出现的所有残基的标准替换分值取平均. 例如, 他们会为我们例子中第一列上的残基 a 赋予分值

$$\tfrac{5}{7}s(\mathtt{V},a)+\tfrac{1}{7}s(\mathtt{F},a)+\tfrac{1}{7}s(\mathtt{I},a)$$

其中 $s(a,b)$ 是标准替换矩阵. 他们同时也用启发式方程为每列设定空位损失, 该方程根据多序列联配中观测到的横跨该列的最大空位长度递减空位 (插入或删除) 的罚值.

虽然这种方法从直观上看可以将信息组合起来, 并且也被有效地用于寻找某些家族的新成员, 但它的确会导致一些异常情况. 例如图5.3中第一列比第二列更加保守, 但是第一列中被替换矩阵涂抹掉的信息和第二列中涂抹掉的差不多. 如果我们的联配包含100条序列, 并且它们都在某位置包含一个半胱氨酸 (cysteine , C), 那么对一个"平均" 的 profile 而言, 该列隐含的概率分布就将正好与从单一序列生成的分布相同. 这与我们的期望不相符, 因为观测到的确认例子越多, 半胱氨酸的似然就应该越大.

除了这些对替换分值的结论以外, 空位分值也和我们的预期不同. 例如图5.3联配中第二列中一个删除的分值将被设定为与第四列中的一样, 其中第二列中有一条序列HBB_ HUMAN包含一个删除, 而第四列里七条序列中有五条是删除开端. 为第四列中新空位开端设置更高的概率将更为合理.

人们已经对非概率论的 profile 做出调整, 以便应对上述以及其他问题 [Thompson, Higgins & Gibson 1994b; Gribskov & Veretnik 1996], 我们将在本书的后面章节进行介绍.

基本profile HMM 的参数化

现在回到隐马模型的 profile. 与所有 HMM 一样, 它们拥有发射与转移概率. 假设这些概率都非零, 那么一个 profile HMM 就可以为来自给定字母表的残基之任意可能序列进行建模. 因此它在整个序列空间上定义了一个概率分布. 参数化过程的目的就是使这个分布在家族的成员周围达到峰值.

我们现在已有的用于控制分布形状的参数是概率值以及模型长度. 我们要详细解释如何最优地设定这些值. 这里我们给出 Krogh *et al.* [1994]的基本方法. 在介绍

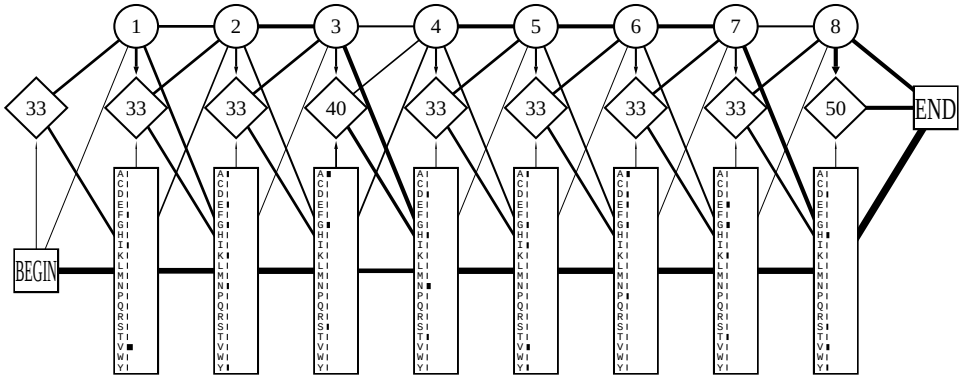


图 5.4: 使用Laplace 法则, 从图 Figure 5.3的小联配推导出的隐马模型. 每个匹配状态中, 在氨基酸的后面用线条的长短表示发射概率大小, 而状态间线段的粗细表示转移概率的大小. I → I的转移概率乘以 100 后显示在插入状态中. (本图由SAM 软件包自动生成.)

完数据库搜索和局部联配变体的部分后, 我们会回过头对其他参数估计技术进行广泛的讨论.

更准确地说, 模型长度的选择对应于一个决策 (decision), 即需要将多序列联配中的哪些列指定为匹配状态, 哪些列指定为插入状态. 上面从单一序列 y 得到的 profile HMM 对每个残基 y_i 都有一个匹配状态. 但是图5.3清楚地告诉我们, 该区域的一致序列应该只有八个残基, 而对一致序列来讲, `GLB1_ GLYDI` 中未标记星号的两个残基应该被看作是插入. 我们暂时先使用启发式规则决定哪列对应匹配状态, 哪列对应插入状态. 一个有效的简单规则是将那些超过一半字符为空位的列建模为插入.

第二个问题就是如何为概率参数赋值. 我们认为该联配提供了序列 x 到我们 HMM 间联配的一组独立样本. 因为那些联配已知, 所以我们就能用第3.3节的方程 (3.18) 直接估计参数. 我们只需要记录每个转移或发射的使用次数, 然后按照

$$a_{kl} = \frac{A_{kl}}{\sum_{l'} A_{kl'}} \text{ 和 } e_k(a) = \frac{E_k(a)}{\sum_{a'} E_k(a')}$$

给概率赋值即可, 其中 k 和 l 是状态的下标, a_{kl} 和 e_k 是转移和发射概率而 A_{kl} 和 E_k 是相应的频率.

当在我们的训练联配中包含十分大量序列的极限情况下, 该方法将给出概率之准确且相容 (consistent) 的估计. 然而当序列较少时, 此方法存在一些问题. 一个主要困难是, 许多转移或发射可能在训练联配中并不出现, 因此就得到零概率, 这就意味着以后永远不允许它们再出现了. 更广泛地说, 我们并没有使用蛋白联配的任何已有知识, 这就像早期的非概率论方法隐含地使用独立导出的替换矩阵一样. 一种避免零概率的最简单方法是, 我们可以向观测频数中添加伪计数(见第1章和第3章). 最简单的伪计数方法是 Laplace 规则: 为每个频数都添加1. 我们将在后面的第5.6节中讨论选择伪计数值的更好办法以及其他估计参数的方法.

例子: 图5.3中 HMM 的参数

假设我们采用 Laplace 规则获得对应于图5.3中联配的 HMM 参数. 于是有 $e_{\mathrm{M}_1}(\mathtt{V}) = 6/27$, $e_{\mathrm{M}_1}(\mathtt{I}) = e_{\mathrm{M}_1}(\mathtt{F}) = 2/27$, 并且对所有除 `V`、`I` 和 `F` 之外的残基 a 有 $e_{\mathrm{M}_1}(a) = 1/27$. 相似地有 $a_{\mathrm{M}_1\mathrm{M}_2} = 7/10$, $a_{\mathrm{M}_1\mathrm{D}_2} = 2/10$ 和 $a_{\mathrm{M}_1\mathrm{I}_1} = 1/10$ (从第一列开始有六个匹配到匹配的转移, 一个到删除状态的转移, 它发生在 `HBB_ HUMAN` 中, 而且没有插入). 图5.4给出了图表形式的 HMM 完整参数集. □

5.4 基于 profile HMM 的搜索

发明 profile HMM 的一个主要目的, 是为了通过获得某序列与 profile HMM 间

的显著匹配而探测家族中的潜在成员. 我们暂时假设正在搜索全局匹配. 在实践中, 就像二序列联配那样, 有一种局部联配方法也许会在寻找远距离匹配方面更加敏感. 对此我们将在下一节讨论.

我们选择一种方法为到隐马模型的匹配计分. 我们既可以用 Viterbi 方程给出一条序列 x 的最可能联配 π^* 及其概率 $P(x,\pi^*|M)$, 也可以使用前向方程 (forward equation) 计算 x 在所有可能路径上求和的全概率 $P(x|M)$.

出于实际目的, 当使用以上两种方法评估潜在匹配时, 我们需要考虑的值都是所得概率同给定标准随机模型下 x 概率间的对数几率比

$$P(x|R) = \prod_i q_{x_i}.$$

因此我们在这里列出专门用于 profile HMM 的 Viterbi 和前向算法, 它们能直接算出我们想要的对数几率值. 注意, 转变到对数几率并不改变结果; 在后面我们完全可以把随机模型的对数分值减掉. 然而, 对数几率值更明了有效. 使用对数几率单位的另一个实际原因是, 它可以避免计算原始概率时出现的下溢问题, 这在第3.6节中讨论过.

Viterbi 方程

令 $V_j^{\mathrm{M}}(i)$ 是将子序列 $x_{1\ldots i}$ 匹配上结束于状态 j 的子模型之最优路径的对数几率分值, 该路径以 M_j 状态发射出 x_i 作为结束. 相似地, $V_j^{\mathrm{I}}(i)$ 表示以 I_j 状态发射出 x_i 作为结束之最优路径的分值, 而 $V_j^{\mathrm{D}}(i)$ 对应于以 D_j 状态结尾的最优路径. 于是有如下通式:

$$\begin{aligned}
V_j^{\mathrm{M}}(i) &= \log \frac{e_{\mathrm{M}_j}(x_i)}{q_{x_i}} + \max \begin{cases} V_{j-1}^{\mathrm{M}}(i-1) + \log a_{\mathrm{M}_{j-1}\mathrm{M}_j}, \\ V_{j-1}^{\mathrm{I}}(i-1) + \log a_{\mathrm{I}_{j-1}\mathrm{M}_j}, \\ V_{j-1}^{\mathrm{D}}(i-1) + \log a_{\mathrm{D}_{j-1}\mathrm{M}_j}; \end{cases} \\
V_j^{\mathrm{I}}(i) &= \log \frac{e_{\mathrm{I}_j}(x_i)}{q_{x_i}} + \max \begin{cases} V_j^{\mathrm{M}}(i-1) + \log a_{\mathrm{M}_j\mathrm{I}_j}, \\ V_j^{\mathrm{I}}(i-1) + \log a_{\mathrm{I}_j\mathrm{I}_j}, \\ V_j^{\mathrm{D}}(i-1) + \log a_{\mathrm{D}_j\mathrm{I}_j}; \end{cases} \\
V_j^{\mathrm{D}}(i) &= \max \begin{cases} V_{j-1}^{\mathrm{M}}(i) + \log a_{\mathrm{M}_{j-1}\mathrm{D}_j}, \\ V_{j-1}^{\mathrm{I}}(i) + \log a_{\mathrm{I}_{j-1}\mathrm{D}_j}, \\ V_{j-1}^{\mathrm{D}}(i) + \log a_{\mathrm{D}_{j-1}\mathrm{D}_j}. \end{cases}
\end{aligned} \tag{5.1}$$

在典型例子中, 计算 $V_j^{\mathrm{I}}(i)$ 的方程里不出现发射分值 $e_{\mathrm{I}_j}(x_i)$, 因为我们假设从插入状态 I_j 出来的发射分布与背景分布相同, 所以在对数几率形式中概率值可以被消去. 另外, $\mathrm{D}\to\mathrm{I}$ 和 $\mathrm{I}\to\mathrm{D}$ 的转移项可能不出现, 就像上面所讨论的.

我们需要稍微注意动态规划过程中的起始和终止部分. 我们打算允许联配以删除或插入状态作为开始或结束, 以免序列的开始或结束不能匹配上模型的第一或最后一个匹配状态. 从机制上保证这点的最简单办法就是, 将开始状态重命名为 M_0, 并令 $V_0^{\mathrm{M}}(0) = 0$ (就像我们在第3章所做的那样). 然后我们允许转移到 I_0 和 D_1. 相似地, 最后我们可以把所有以插入或删除状态结尾的可能路径归为一类, 具体讲是重命名结束状态为 M_{L+1} 并使用不带发射项的 (5.1) 式中的第一个关系式来计算 $V_{L+1}^{\mathrm{M}}(n)$ 作为最终值.

如果将这些递归方程与式 (2.16) 的标准空位动态规划方程相比较, 那么我们就会发现, 除了给变量重命名以及替换、空位开端和空位延伸的分值都依赖于其在模型中的位置 j 以外, 算法的其他部分都相同.

前向算法

前向算法的递归方程与 Viterbi 方程类似, 但其中的求最大值操作被求和取代. 我们为部分全对数几率比 (full log-odds ratios) 定义变量 $F_j^{\mathrm{M}}(i)$、$F_j^{\mathrm{I}}(i)$ 和 $F_j^{\mathrm{D}}(i)$,

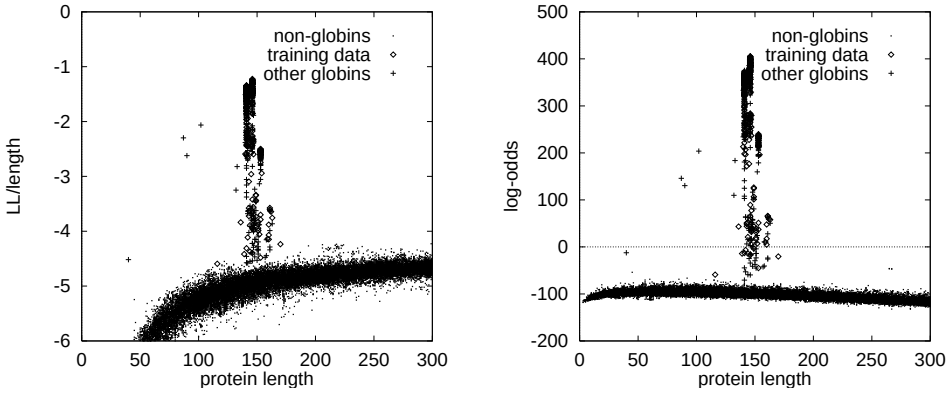


图 5.5: 左图: 作为序列长度函数的、经长度归一化后的 LL 分值. 右图: 相同情形的对数几率分值.

它们与 $V_j^{\mathrm{M}}(i)$、$V_j^{\mathrm{I}}(i)$ 和 $V_j^{\mathrm{D}}(i)$ 相对应. 这样递归方程便是:

$$\begin{aligned}
F_j^{\mathrm{M}}(i) &= \log \frac{e_{\mathrm{M}_j}(x_i)}{q_{x_i}} + \log\left[a_{\mathrm{M}_{j-1}\mathrm{M}_j} \exp\left(F_{j-1}^{\mathrm{M}}(i-1)\right)\right. \\
&\qquad \left. + a_{\mathrm{I}_{j-1}\mathrm{M}_j} \exp\left(F_{j-1}^{\mathrm{I}}(i-1)\right) + a_{\mathrm{D}_{j-1}\mathrm{M}_j} \exp\left(F_{j-1}^{\mathrm{D}}(i-1)\right)\right]; \\
F_j^{\mathrm{I}}(i) &= \log \frac{e_{\mathrm{I}_j}(x_i)}{q_{x_i}} + \log\left[a_{\mathrm{M}_j\mathrm{I}_j} \exp\left(F_j^{\mathrm{M}}(i-1)\right)\right. \\
&\qquad \left. + a_{\mathrm{I}_j\mathrm{I}_j} \exp\left(F_j^{\mathrm{I}}(i-1)\right) + a_{\mathrm{D}_j\mathrm{I}_j} \exp\left(F_j^{\mathrm{D}}(i-1)\right)\right]; \\
F_j^{\mathrm{D}}(i) &= \log\left[a_{\mathrm{M}_{j-1}\mathrm{D}_j} \exp\left(F_{j-1}^{\mathrm{M}}(i)\right) + a_{\mathrm{I}_{j-1}\mathrm{D}_j} \exp\left(F_{j-1}^{\mathrm{I}}(i)\right)\right. \\
&\qquad \left. + a_{\mathrm{D}_{j-1}\mathrm{D}_j} \exp\left(F_{j-1}^{\mathrm{D}}(i)\right)\right].
\end{aligned}$$

起始和终止条件与 Viterbi 算法中相同, $F_0^{\mathrm{M}}(0)$ 被初始化为 0.

虽然它们看起来有些复杂, 但在实际执行中我们可以使用查找 (lookup) 和内插 (interpolation) 函数将运算 $\log(\mathrm{e}^x + \mathrm{e}^y)$ 高效地执行到足够精度; 见3.6节.

对数几率计分的替代方法

在某些较早的 HMM 文献中, 人们不计算相对于随机模型的对数几率分值, 而是直接使用给定模型下序列概率的对数. 它称为 LL 分值即"对数似然" (log likelihood): $\mathrm{LL}(x) = \log P(x|M)$. LL 分值强烈依赖于长度, 因此就搜索而言使用简单阈值并不足够好. 最好使用除以序列长度的 LL 分值, 但即使这样也不总是完美, 因为 LL 与序列长度间的依赖关系并不是线性的(见下面的例子).

能够避开这个问题的一种方法是, 估计一个标准分值以及作为长度函数的标准差, 然后使用每条序列远离平均值的标准差数目. 它被称为Z-分值, 下面的例子中也会给予说明.

例子: 对球蛋白建模并进行搜索

从 300 个随机选取的球蛋白序列中从头估计 profile HMM, 即使用第6章将要介绍的过程从未联配的序列开始. 我们采用一种简单的伪计数规则. 我们从多次估计中选出全局 LL 值最高的模型. (使用 SAM 软件包1.2版本的默认设置, 见 Hughey & Krogh [1996].)

在此模型下我 LL 和对数几率分值. 对于零模型, 我们使用了训练集中300条序列的氨基酸频率. 图5.5中显示了经过长度归一化后训练集中所有球蛋白的分值、数据库中所有其他球蛋白的分值以及所有剩余的长度不超过300个氨基酸的球蛋白分值. [1]除了在"模糊地带" (twilight zone) 内的少数序列外, 球蛋白序列都与非球蛋白清晰地分开.

[1]少量可疑球蛋白和其他一些不常见的序列已经从这些数据中移除. ——原注

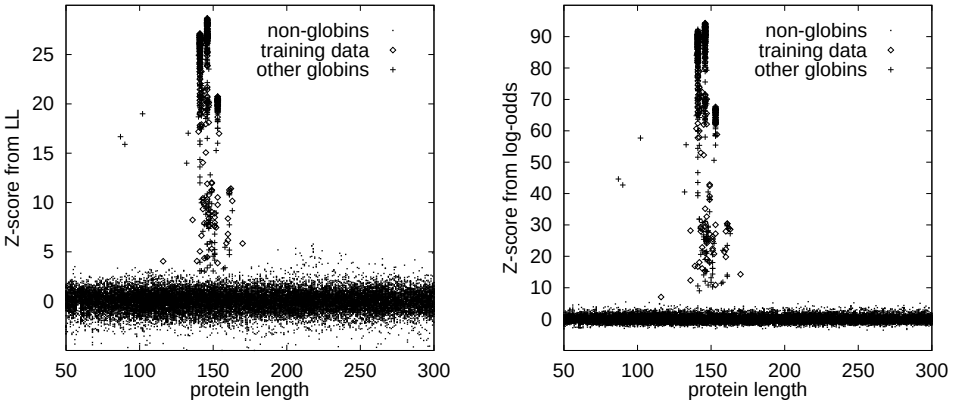


图 5.6: 从 LL 分值(左) 和对数几率值(右) 算出的 Z-分值.

此二者间的主要区别在于非球蛋白分值的方差上, 它们的对数几率分值较低, 因此与球蛋白产生了明确的分化. 但是只简单选择对数几率值等于零的分界线将在搜索中损失大量真实的球蛋白. 这是因为 profile HMM 覆盖的范围不够宽: 它过于集中在球蛋白的一个子集上. 虽然我们将在本章后面介绍可以直接处理这种问题的方法, 但我们也可以对给定的搜索结果采用更实用的方法把信号从噪音中分离出来, 并对每个搜索结果计算 Z-分值.

计算 Z-分值时, 我们会为非球蛋白序列的 LL 或对数几率分值拟合一条平滑曲线(Krogh *et al.* [1994] 给出了方法概述). 随后对每个长度(确切地说是其周围的一个小邻域) 都估计标准差, 接着以此标准差为单位计算每个分值到平滑曲线间的距离. 这就是所谓的 Z-分值. 图5.6画出了结果(它仍然是序列长度的函数). [2]

显然现在我们就可以确定某个阈值将大多数球蛋白和所有其他序列分开. 同时我们也注意到由对数几率算出的分值更适于做判别, 它的信噪比大概是 LL 分值的3倍. 其原因是, 通过除以随机模型的概率我们可以调整序列残基的组份. 如果不这么做, 和球蛋白残基组成相似的序列就会比包含不同残基的序列更倾向于得到较高的分值, 这就增加了噪音的方差. □

联配

除了寻找匹配以外, profile HMM 的另一个主要用途是给出序列到家族的联配, 或更精确地说是将序列添加到家族的多序列联配中. 这是下一章"多序列联配方法"的主要内容, 其中详细地阐述了带 profile HMM 的联配. 目前我们仅指出, 自然的解决方案是选择最高计分或 Viterbi 联配. 通过回溯 Viterbi 变量 $V_j^{\bullet}(i)$ 我们可以得到这个解, 这与二序列联配中的方法完全一样. 除了这个以外第4章中的所有方法都能应用于探究变体以及评估联配的可靠性.

5.5 用于非全局联配的profile HMM 变体

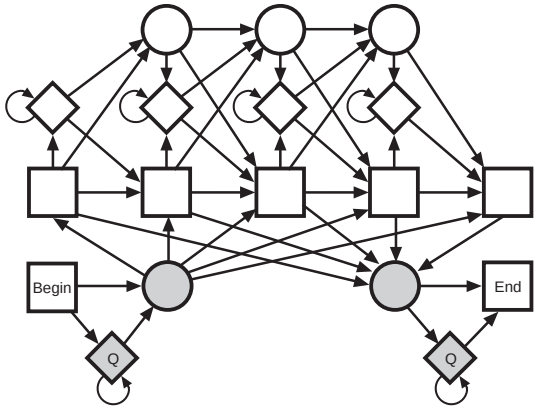
我们已经看到, 一条序列到一个 profile HMM 的 Viterbi 联配与第2章中描述过的两条序列间使用仿射空位罚分的全局动态规划比较之间存在十分紧密的联系. 因此使用 profile HMM, 我们可以推广所有动态规划的变体, 例如寻找局部、重复或交叠匹配的变体.

但是从第2章开始我们已经发展了更完整的概率论模型, 这次我们想更加关注于保证转换到局部算法的结果仍然是适当的概率论模型, 即每条序列都被赋予一个真正的概率以便对所有序列求和时有 $\sum_x P(x|M) = 1$. 我们的做法是为完整序列 x 指

[2]不存在关于这些分值分布形状的解析结果. 全局联配分布很可能不是一个严格的 Gauss 分布(见 [Waterman 1995]), 但看起来它是一个很好的近似. 对局部联配而言, 极值分布可能更合理, 本书第2章已经讨论过. ——原注

定一个新模型, 它合并了原有的 profile HMM 与某简单自循环模型的一个或多个复本, 该自循环模型用于解释无法联配的序列区域. 这些行为与向 profile 本身添加的插入状态很相似. 我们称其为**侧翼模型** (flanking model) 状态, 因为它们用于把侧翼序列的模型构建为实际 profile 匹配本身.

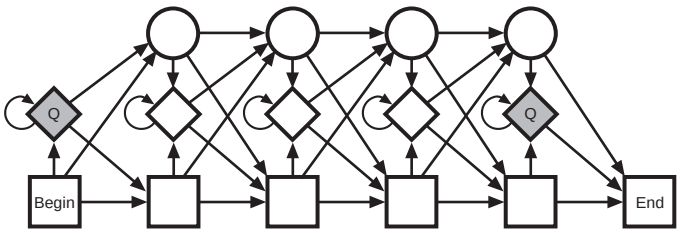
局部(Smith-Waterman风格) 联配的模型如下:



侧翼模型状态用带阴影的菱形表示. 注意, 和指定通常为 q_a 的新状态发射概率一样, 我们必须指定许多新的转移概率. 侧翼状态的环状概率应该接近于 1, 因为它们必须能够解释序列的长片段. 令其为 $(1-\eta)$. 同时请注意, 我们也使用了沉默状态作为"切换点" , 即图中带阴影的圆圈, 以便减少转移的总数.

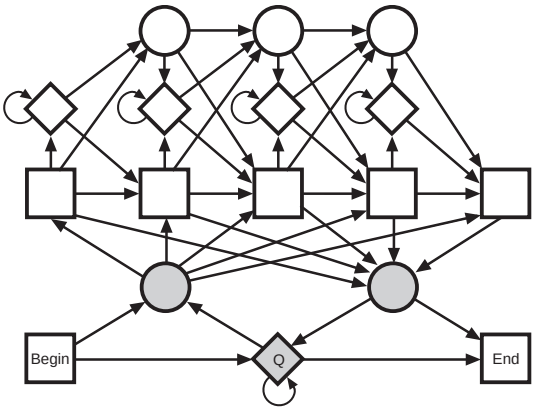
下一个问题就是如何设置模型中从左边的侧翼状态到不同开始点的所有转移概率. 有种方法是令它们都具有相同的概率 η/L. 另一种方法是为模型开始的起始点赋予更大的概率. HMMER 软件包中 [Eddy 1996] 的缺省设置是, profile 起始概率为 $\eta/2$, 其他位置为 $\eta/(2(L-1))$, 即更加偏向于模型开始的起始部分匹配.

如果第一个模型状态被赋予全部的概率, 那么这就迫使此模型只能匹配被搜索序列中 profile 的完整拷贝, 这保证了一种"交叠" 匹配约束. 在某些情况下, 例如当 HMM 代表我们期望全部出现或彻底消失的蛋白质结构域时, 这种方法是合适的. 然而考虑到罕见情形, 如其中第一个残基可能缺失, 此时允许一个从侧翼状态到删除状态的直接转移将是很明智的, 如图所示:



显然通过调整转移连接与概率, 我们可以制造出很多不同种类的模型, 它们可能在

不同环境下是有用的. 我们给出最后一个例子, 它类似于第一个局部匹配模型:



它允许到 profile 模型子部分的重复匹配, 这与第2章中重复算法的变体有些类似.

注意, 所有这些转移连接性与概率设置的变体都不仅仅影响到模型所允许的匹配类型, 而且还影响分值. 如果找到了一个好的匹配, 那么更严格的转移分布将给出更高的匹配分值, 因此当能够设计出这样的分布以代表期望中正确的匹配类型时, 它们更可取.

练习

5.1 证明如果随机模型与第4章中介绍的一样(两个连续状态以概率 $(1-\eta)$ 自循环), 并且 η 值与侧翼模型中的相同, 那么局部联配模型将给出类似2.9的新方程.

5.2 解释上述方程间差别的原因.

5.6 对概率估计的深入说明

因为前文许诺过, 所以我们现在回过头详细地讨论参数估计问题. 虽然本节的讨论主要关注发射概率, 但类似的方法也可用于转移概率. 我们的目的是介绍能够使用的方法. 本书第 11 章(第 p. 213页) 将给出从样本计数中得到概率估计的更详细数学讨论.

最直接的估计参数方法是给出参数的最大似然估计. 我们将对以前使用的记号做微小改变. 假设给定联配中位置 j 处残基 a 的观测频率是 c_{ja}, 那么相应模型参数 $e_{\mathrm{M}_j}(a)$ 的最大似然估计就是

$$e_{\mathrm{M}_j}(a) = \frac{c_{ja}}{\sum_{a'} c_{ja'}}. \tag{5.2}$$

如前所述, 以上估计的一个明显问题是, 如果某特定结果没有被观测到, 那么它的概率将被估计为零. 而这种情况经常发生. 例如, 图5.3联配的第一列只出现了 `V`、`I` 和 `F`. 但在生物学中, 其他球蛋白序列很有可能在那个位置出现别的氨基酸残基. 解决此问题的最简单办法就是向观测计数 c_{ja} 中添加伪计数. 下面我们首先详细讨论伪计数方法, 然后介绍更多复杂的备选方法.

简单伪计数

一种简单且常用的伪计数方法是向所有计数都添加一个常数, 这就避免了零概率问题. 当此常数是 1 时, 该方法称为"Laplace 法则" , 我们在前面的例子中就使用了这种方法. 另一种稍微有些复杂的方法是添加一个正比于背景概率的量, 如

$$e_{\mathrm{M}_j}(a) = \frac{c_{ja} + Aq_a}{\sum_{a'} c_{ja'} + A}, \tag{5.3}$$

其中 c_{ja} 是真实计数, 而 A 是和真实计数相比赋予伪计数的权重. 对蛋白质联配而言, A 的值在 20 附近似乎比较合适.

这种形式的正规化(regularization) 具有诱人的特征, 因为如果数据量很小, 即与 A 相比所有真实计数都很小, 那么 $e_{\mathrm{M}_j}(a)$ 近似等于 q_a. 在另一个极端, 即数据量十分大时, 正规化子(regulariser) 的作用就变得微不足道, 并且 $e_{\mathrm{M}_j}(a)$ 在本质上就等于最大似然解. 因此从直觉上说, 伪计数很有意义.

添加伪计数相当于就是在一般的蛋白质知识基础上向联配中添加假想数据 (fake imagined data), 以便代表所有其他可能出现的情况. 因此它们就对应于观测到以联配形式出现的家族特定数据之前, 蛋白质家族的先验信息. 这一陈述可以被纳入到 Bayes 框架下. Bayes 方程告诉我们如何将数据 D 与参数上的先验概率分布 $P(\theta)$ 整合到一起, 并给出 θ 上的后验分布, 我们可以由此后验分布取最大值或均值作为最好估计:

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)}.$$

在我们的情况里, 参数 θ 是模型概率. 上文提到的伪计数方法在 Bayes 框架中对应于假设一个在所有概率上带参数 $\alpha_a = Aq_a$ 的 Dirichlet 先验分布; 其数学细节请见第11章.

Dirichlet 混合

和基于替换矩阵的方法相比, 简单伪计数方法存在的问题是单个伪计数向量仅能包含最基本的先验知识. 因此我们需要很多联配的示例数据才可以得到参数的合理估计. 经验告诉我们为了达到良好的判别效果, 通常在对蛋白质建模时至少要有50条示例序列才令人满意.

因此为了包括更好的先验信息, Brown Brown *et al.* [1993]建议使用 Dirichlet 分布的**混合** (mixture) 作为先验. 其原理是, 也许存在一些对应于不同类型联配环境的不同伪计数先验集 $\alpha_\bullet^1, \ldots, \alpha_\bullet^K$, 其中 α_a^k 对应于上例中的 Aq_a. 一个集合可能与外露 (exposed) 的回折环境有关, 而另一个可能与内含 (buried) 的小残基环境有关, 等等. 给定计数 c_{ja}, 我们首先 (基于它与观测数据的拟合程度) 估计每个先验分布 k 的似然有多大, 然后按照这些后验概率将它们的结果组合到一起:

$$e_{\mathrm{M}_j}(a) = \sum_k P(k|\boldsymbol{c}_j)\frac{c_{ja} + \alpha_a^k}{\sum_{a'}(c_{ja'} + \alpha_{a'}^k)},$$

其中 $P(k|\boldsymbol{c}_i)$ 是**后验混合系数** (posterior mixture coefficients). 我们可以用 Bayes 法则计算之,

$$P(k|\boldsymbol{c}_j) = \frac{p_k P(\boldsymbol{c}_j|k)}{\sum_{k'} p_{k'} P(\boldsymbol{c}_j|k')},$$

其中 p_k 是每个混合组分的先验概率, 而 $P(\boldsymbol{c}_j|k)$ 是依据 Dirichlet 混合 k 的数据概率. $P(\boldsymbol{c}_j|k)$ 的方程看起来令人恐惧, 但事实上求解起来却十分简单:

$$P(\boldsymbol{c}_j|k) = \frac{(\sum_a c_{ja})!}{\prod_a c_{ja}!}\frac{\prod_a \Gamma\left(c_{ja} + \alpha_a^k\right)}{\Gamma\left(\sum_a (c_{ja} + \alpha_a^k)\right)}\frac{\Gamma\left(\sum_a \alpha_a^k\right)}{\prod_a \Gamma\left(\alpha_a^k\right)},$$

其中 $\Gamma(x)$ 是 gamma 函数, 它是一个定义在实数上的标准函数, 与定义在整数上的阶乘函数相关. 本书第11章将给出此方程的详细解释以及如何得到混合组分的分布 $\alpha_\bullet^k$.

使用这类方法后, 我们似乎可以用少到10或20个示例, 将好的 profile HMM 拟合到联配 [Sjölander *et al.* 1996].

替换矩阵混合

Dirichlet 混合的一种替代方法是, 利用从观测计数与替换矩阵中得到的信息, 在单一 Dirichlet 公式体系内调整伪计数. 虽然这种方法的理论基础并不牢固, 但作

为启发式方法, 它在直观上是合理的, 它整合了非概率论 profile 方法和 Dirichlet 伪计数方法的特点.

第一步先将矩阵元素 $s(a,b)$ 转换为条件概率 $P(b|a)$. 若我们假设像第2章一样, 将对数几率比作为推导出的替换矩阵元素, 则 $s(a,b) = \log(P(a,b)/q_a q_b)$, 这与 $\log(P(b|a)/P(b))$ 相等, 于是有 $P(b|a) = q_b \mathrm{e}^{s(a,b)}$. 事实上给定背景概率 q_a 后, 我们可以从任意计分矩阵 $s(a,b)$ 推导 $P(b|a)$ 的值; 见下面的说明.

给定条件概率 $P(b|a)$, 我们可以生成伪计数如下. 令 f_{ja} 是从计数算出的最大似然概率, 那么 $f_{ja} = c_{ja}/\sum_{a'} c_{ja'}$. 使用这些值, 将伪计数值设为

$$\alpha_{ja} = A \sum_b f_{jb} P(a|b),$$

其中 A 是一个正常数, 相当于简单伪计数方法中使用的那个正常数 [Tatusov, Altschul & Koonin 1994; Claverie 1994; Henikoff & Henikoff 1996]. 然后我们使用本质上与5.3相同的方程获得模型参数:

$$e_{\mathrm{M}_j}(a) = \frac{c_{ja} + \alpha_{ja}}{\sum_{a'} c_{ja'} + \alpha_{ja'}}.$$

这种伪计数没有明显的统计学解释, 但是它的想法却十分自然: 氨基酸 i 对伪计数 j 的贡献正比于它在该列中的丰度 (abundance) 以及它变为氨基酸 j 的概率. 该公式相当于在二序列连配处理和最大似然解间进行内插 (interpolate). 如果序列数量很少(特别是当 $A \gg 1$ 时), 那么公式中的替换矩阵项就将起主要作用, 而当计数很大(更精确地说是计数的总和 $C_j \gg A$) 时此公式将给出接近最大似然估计的值.

对于伪计数的比例常数 A, 我们有各种不同的选择. 例如, Lawrence *et al.* [1993] 中 $A = 1$, 但实践中其效果显得很弱. Claverie [1994]推荐使用 $A = \min(20, N)$, 而 Henikoff & Henikoff [1996] 则推荐使用 $A = 5R$, 其中 R 是某列中观测到的不同残基种类数(即所有满足 $c_{ja} > 0$ 的 a 的数目).

从任意矩阵导出 $P(b|a)$

只要满足某些条件, 即使计分矩阵 $s(a,b)$ 不是对数几率矩阵, 我们也可以找到某个比例因子 λ 使得 $\lambda s(a,b)$ 能够按照对数几率矩阵的方式正确工作 [Altschul 1991]. 这些条件就是矩阵负偏 (negatively biased), 即 $\sum_{ab} q_a q_b s(a,b) < 0$ 并且它至少包含一个正元素.

我们想要得到一组值 r_{ij}, 并使

$$s(a,b) = \frac{1}{\lambda} \log \frac{r_{ab}}{q_a q_b},$$

其中 r_{ab} 可以解释为 a, b 对的概率. 由上述方程容易得到用替换矩阵表示的位置对概率 $r_{ab} = q_a q_b \exp(\lambda s(a,b))$. 要想成为真正的概率, 所有 r_{ab} 加起来必须等于1. 因此我们需要找到 λ 满足

$$f(\lambda) = \sum_{a,b} q_a q_b \mathrm{e}^{\lambda s(a,b)} - 1 = 0. \tag{5.4}$$

满足上式的一个值是 $\lambda = 0$, 但显然它并非我们所求. 已经证明, 上面给出的两个条件足够保证此方程有另一个正数解; 请见后面的练习.

所得 λ 值称作替换矩阵的自然比例因子. 这种替换矩阵的概率论解释引出了矩阵的熵度量 $\sum_{ab} r_{ab} \log(r_{ab}/q_a q_b)$, 它对替换矩阵的特征描述和替换矩阵间的比较都很有用 [Altschul 1991].

练习

5.3 用负偏条件证明: 对足够小的 λ, $f(\lambda)$ 为负. 提示: 计算 $f(\lambda)$ 在 $\lambda = 0$ 处的导数 $f'(0)$.

5.4 用第二个条件, 即至少存在一个正的 $s(a,b)$, 证明存在足够大的 λ 使 $f(\lambda)$ 为正.

5.5 最后, 证明 $f(\lambda)$ 的二阶导数为正, 并用此结论和上面两题的结论证明有且仅有一个正数 λ 满足 (5.4) 式.

基于祖先的估计

已经存在比上面介绍的方法原则性更强, 也更直接的方法使用替换矩阵信息估计 HMM 概率. 此方法不使用伪计数而是假设所有观测序列都从一个共同祖先独立演化而来, 此方法还可以估计那个祖先中给定位置处出现的残基(更确切地说是该残基之类型的后验概率分布). 这样我们就能从中估计在祖先的新后代里观测到每个残基的概率, 它们与样本中的概率不同.

假设存在示例序列 x^k, 而联配 (alignment) 的第 j 列是残基 x_j^k (我们稍微调整了记号; 如果存在空位, 那么该 x_j^k 就不是序列 x^k 中的第 j 个残基, 但这种记号对现在的讨论很方便). 这里我们再次需要从替换矩阵中导出的条件概率 $P(b|a)$. 令共同祖先中的残基是 y_j. 然后我们可以用 Bayes 法则计算 $y_j = a$ 的后验概率

$$P(y_j = a|\text{alignment}) = \frac{q_a \prod_k P(x_j^k|a)}{\sum_{a'} q_{a'} \prod_k P(x_j^k|a')}. \tag{5.5}$$

注意, 我们需要共同祖先残基之先验分布, 并令其为 q_a, 因为它是在缺乏更多信息的情况下氨基酸的背景概率.

现在我们就能用一条新序列之预测概率计算 HMM 的发射概率

$$e_{\mathrm{M}_j}(a) = \sum_{a'} P(a|a')P(y_j = a'|\text{alignment}). \tag{5.6}$$

上文已经提到, 本方法存在的一个问题是, 在保守度方面, 列与列之间差别明显. 事实上, 这也正是我们在使用联配估计 profile HMM 时想利用的性质之一. 然而使用单一替换矩阵就意味着假设了固定的保守度. 在第2章我们曾讨论过, 矩阵通常总是以家族形式出现, 而且它们蕴涵各不相同的保守性水平. PAM [Dayhoff, Schwartz & Orcutt 1978] 和 BLOSUM [Henikoff & Henikoff 1992] 系列矩阵就是例子. 因此如果对一个家族中的矩阵进行优化选择, 那么我们便可以显著改善5.5和5.6中的方法. 这样的话, 非常保守的列就可以使用保守矩阵, 如 PAM30, 而非常不保守的列就可以使用发散矩阵, 如 PAM500.

那么如何选择最优矩阵呢? 一个自然的办法是最大化观测数据的似然

$$P(x_j^1, \ldots, x_j^N|t) = \sum_a q_a \prod_k P(x_j^k|a, t), \tag{5.7}$$

其中 t 是矩阵族参数(t 是进化**时间** (time)). 这里也可以使用 Bayes 方法, 提出 t 上的先验分布, 然后将其与 (5.7) 结合起来, 使用 Bayes 法则得到 t 的后验分布, 最后在 (5.6) 式中求和. 但这就使得计算任务变得十分繁重.

最大似然时间依赖 (time-dependent) 方法与 PROFILE 软件包中 [Gribskov & Veretnik 1996] 的"进化权重" (evolutionary weight) 方法有紧密联系. 但是该进化权重方法为每个可能的祖先氨基酸估计不同的进化时间 t, 而且对一组底线 (baseline) 概率调整结果权重; 细节请参见 Gribskov & Veretnik [1996]. 本小节介绍的方法与随后在第8章中应用最大似然方法构建系统发育树之方法也存在强烈关联.

检验伪计数方法

人们用各种不同方法对上述所有方法进行了检验. Henikoff & Henikoff [1996]进行了广泛的直接检验, 他们建立了 profile 并将其用于搜索. 结果表明, 如上文所说 $A = 5R$ 时基于替换矩阵的 (5.6) 是最好的方法; Dirichlet 混和正规化子紧随其后. 但其他检验给出不同的结果 [Tatusov, Altschul & Koonin 1994; Karplus 1995], 因此究竟哪种方法最好还不清楚, 而且这很有可能依赖于应用, 以及所使用的混和组分或替换矩阵之细节.

Karplus [1995]给出了检验不同正规化子 (regulariser) 的一种有趣方法. 他没有进行海量的数据库搜索, 而是提出下面的问题[3]: 从小样本出发我们到底可以将氨基酸分布近似到什么程度? 他从一个很大的深层联配集合中抽出若干列(见

[3]原书第二次印刷前, 对原书第121页进行了修订. ——原注

BLOCKS 数据库; Henikoff & Henikoff [1991]). 想像一下我们从带完全计数 C_a 的一列中取出大小为 n、计数为 s_a 的一个小样本. 使用上面提到的方法之一, 我们可以从样本计数 s_a 估计在相同列中可能出现的其他符号之概率 $e_s(a)$ (下标 s 用于提醒我们: 此估计依赖于样本计数). 我们也能根据其他符号在数据库所有列中出现的频率, 以及从某列 (column) C 中抽出 s 的概率 $P(s|C)$ (由多项分布给出), 直接估计它们的概率. 该估计便是:

$$P(a|s) = \frac{\sum_{\text{columns } C} P(s|C)C_a}{\sum_{\text{columns } C} P(s|C)\,|C|},$$

其中 $|C|$ 表示列 C 中符号的个数. 只有当样本容量 $n \leq 5$ 时才能计算出 $P(a|s)$, 但这也是最有意思的地方, 因为正规化过程正是对小样本才最重要. 我们现在可以使用相对熵 $-\sum_a P(a|s)\log e_s(a)$ 比较"理想" (ideal) 概率 $P(a|s)$ 与由正规化子给出的概率. 对所有容量为 n 的样本 s 求和可以给出一个度量

$$E_n = \sum_{s,|s|=n} P(s)\left(-\sum_a P(a|s)\log e_s(a)\right), \tag{5.8}$$

其中 $P(s)$ 是对数据库中所有列求平均后抽出样本 s 的概率, 即 $P(s) = \sum_C P(s|C)\,|C| \,/\, \sum_C |C|$.

Karplus 提出, 一个好的正规化子应该使 E_n 达到最小. 他证明了上述比较复杂的正规化子中有几个已经能形成十分接近最优解的估计值, 即当直到 $n = 5$ 时 E_n 已经很小. 当然我们最终关心的是数据库搜索, 而且没有证据显示获得最小 E_n 值的正规化子在实际中最适合搜索. 源联配数据库中的典型相似性有可能与使用我们的 HMM 要进行搜索的那些相似性有所不同.

像评估方法一样, Karplus 的方法也可以用于给上述不同的方法设定自由参数, 例如式 (5.3) 中使用的伪计数总数 A. 对任意的 A 值, 可以直接地或通过某种随机抽样, 从我们的列数据库中计算 E_n, 事实上我们还能够计算关于 A 的相对熵梯度. 因此借助梯度下降 (gradient descent) 法 [Press *et al.* 1992] 或其他最优化方法, 我们能找到使这个平均相对熵达到最小的 A 值. 原则上, 这对任意样本容量 n 都可行, 而随后得到的参数将依赖于 n.

5.7 最优模型的构建

第一次讨论 profile HMM 的参数化时, 我们曾指出像估计概率参数一样, 有必要决定联配中的哪些列应该被指定为插入状态, 哪些列为匹配状态. 我们称此过程为模型构建. 我们当时提议使用简单启发式方法, 但是有更好的办法. 有一种高效的动态规划算法可以找到使模型后验概率达到最大的列状态分配, 同时它还能拟合最优概率参数.

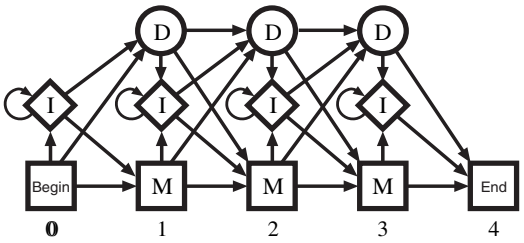
在 profile HMM 的形式体系中, 我们假设一列联配上的符号要么对应于从相同匹配状态出来的发射, 要么对应于从相同插入状态出来的发射. 因此为了指定一个 profile HMM 的体系结构和联配中所有序列的状态路径, 标记出哪些列来自匹配状态就足够了, 如图5.7所示. 在被标记的列中, 符号被指定于匹配状态而空位被指定于删除状态. 在未被标记的列中, 符号被指定于插入状态而空位则被忽略. 从状态路径得到状态转移和符号发射计数, 然后可以用这些计数通过前面一节介绍的方法估计概率参数. 顺便提及, 注意到这个模型估计过程隐含假设了多序列联配是正确的, 即隐含的状态路径总概率是1而所有其他状态路径总概率是0, 这与 Viterbi 算法的假设相似. 下一章将讨论同时进行联配与模型估计的问题.

含 L 列的联配总共有 2^L 种不同的标记组合方法, 因此我们也就可以从 2^L 种不同的 profile HMM 中进行选择. 至少有三种方法可用于确定标记. 在**手工** (manual) 构建方法中, 用户可以手工标记联配的列. 这也许是允许用户手工指定某个给定联配之模型架构的最简单办法. 在**启发式** (heuristic) 构建方法中, 使用某种规则以决定某一列是否要进行标记. 例如当某列空位符号的比例低于某个阈值时, 该列可能被标记. 在 **MAP** (maximum *a posteriori*) 构建法中, 动态规划给出一个后验最大的选择, 下面我们就介绍这种方法.

(a) Multiple alignment:

	x	x	.	.	.	x
bat	A	G	-	-	-	C
rat	A	-	A	G	-	C
cat	A	G	-	A	A	-
gnat	-	-	A	A	A	C
goat	A	G	-	-	-	C
	1	2	.	.	.	3

(b) Profile-HMM architecture:



(c) Observed emission/transition counts

		model position			
		0	1	2	3
match emissions	A	-	4	0	0
	C	-	0	0	4
	G	-	0	3	0
	T	-	0	0	0
insert emissions	A	0	0	6	0
	C	0	0	0	0
	G	0	0	1	0
	T	0	0	0	0
state transitions	M-M	4	3	2	4
	M-D	1	1	0	0
	M-I	0	0	1	0
	I-M	0	0	2	0
	I-D	0	0	1	0
	I-I	0	0	4	0
	D-M	-	0	0	1
	D-D	-	1	0	0
	D-I	-	0	2	0

图 5.7: 从联配构建模型的例子. 图中给出了一小段 DNA 多序列联配 (a), 其中三列的上面标记了 x. 在模型架构中, 我们为这三列编号位置1到位置3 (b). 把列赋予模型位置的做法决定了符号发射和状态转移的计数 (c), 我们可以由此估计概率参数.

MAP 匹配-插入分配

MAP 构建算法递归地计算 S_j, 它是结束于第 j 列的联配之最优模型的对数概率, 这里我们假设第 j 列被标记. 我们从以标记列 $i\,(i<j)$ 为结尾的较小子联配计算 S_j, 具体方法是向 S_i 递增第 i 列与第 j 列间的转移和发射之概率对数和. 相关概率参数可以从由标记列 i 和列 j 但不标记其间的列(如果存在) 所蕴涵的计数中快速 (on the fly) 估计出来.

因为限制在已标记的列 i 与列 j 间的联配部分之转移和发射计数与列 i 前与列 j 后的标记情况无关, 所以这就使动态规划的递归成为可能. 递归式中仅考虑被标记的列, 因为未标记列的转移与发射计数与相邻列的状态分配无关; 单一插入状态可能代表联配中的多个列.

例如令 $\mathscr{T}_{ij}$ 是标记列 i 和 j 间所有状态转移的对数概率和. 我们可以从观测状态转移计数 c_{xy} 和概率 a_{xy} 确定 $\mathscr{T}_{ij}$:

$$\mathscr{T}_{ij} = \sum_{x,y\in \mathrm{M}\ ,\ \mathrm{D}\ ,\ \mathrm{I}} c_{xy} \log a_{xy}.$$

可以从标记 i 和 j 时所导致的部分状态路径得到转移计数 c_{xy}. 例如, 如果在某序列中我们看到列 i 是一个空位, 列 $i+1$ 到 $j-1$ 有五个残基, 列 j 有一个残基, 那么我们就如此计数: 一个删除-插入转移, 四个插入-插入转移, 以及一个插入-匹配转移. 转移概率 a_{xy} 按通常方法从 c_{xy} 估计得到, 它很可能包含 Dirichlet 先验项 α_{xy} (或事实上, 与 $i,\ldots,j$ 以外的标记方式无关之任何形式的先验):

$$a_{xy} = \frac{c_{xy}+\alpha_{xy}}{\sum_y c_{xy}+\alpha_{xy}}.$$

令 $\mathscr{M}_j$ 是列 j 处匹配状态符号发射的类似对数概率贡献, 而 $\mathscr{I}_{i+1,j-1}$ 是列 $i+1,\ldots,j-1$ (其中 $j-i>1$) 处插入状态发射的对数概率贡献. 现在我们就能够给出此算法:

算法: MAP 模型构建

初始:

令 $S_0 = 0, \mathscr{M}_{L+1} = 0.$

递归: 对 $j = 1, \ldots, L+1$:

令 $S_j = \max_{0 \le i < j} S_i + \mathscr{T}_{ij} + \mathscr{M}_j + \mathscr{I}_{i+1,j-1} + \lambda;$

且 $\sigma_j = \operatorname*{argmax}_{0 \le i < j} S_i + \mathscr{T}_{ij} + \mathscr{M}_j + \mathscr{I}_{i+1,j-1} + \lambda.$

回溯: 从 $j = \sigma_{L+1}$ 开始, 当 $j > 0$ 时循环:

将第 j 列标记为匹配列;

令 $j = \sigma_j$. ◁

然后我们就能从这个已标记的联配出发建立 profile HMM. 额外项 λ 是一个罚值, 用于偏向含有更少匹配状态的模型. 按照 Bayes 理论, λ 就是对每一列都进行标记的先验概率之对数值, 它意味着模型长度的一个简单却足够按指数递减的先验分布.

如果注意一下算法实现, 那么对于包含 L 列的联配, 此算法的内存复杂度能达到 $O(L)$, 而时间复杂度能达到 $O(L^2)$.

5.8 训练序列的加权

到此为止我们始终回避估计参数时如何为序列加权的问题. 在一个典型的联配中, 经常有很多彼此联系紧密的序列. 直观上看, 从这些序列中得到的很多信息是共享的, 因此我们在估计过程中赋予它们的影响, 不应该与严重偏离所有其他序列的单条序列一样. 在两条序列完全相同的极端情形, 二者都应该得到其他序列权重的一半才合理, 这样使得其净影响就像只有一条序列一样. 从统计上讲, 问题是通常而言, 我们的例子并不能构成属于某家族的所有序列之好的随机抽样; 独立性假设不成立. 为了处理这种情况, 人们开发了很多不同办法为序列赋予权重. 原则上, 它们中的任意一种方法都可以和前几节介绍的模型参数拟合与模型构建方法组合起来使用.

从树导出的简单加权方案

很多加权方法都基于构建一棵与序列相关的树. 因为家族中的序列通过进化树相互关联, 所以一个十分自然的办法就是当估计每条观测序列的独立贡献时, 设法重建并利用这棵树, 且越晚分化的序列权重越少. 就像下一章详细介绍多序列联配一样, 我们将在第7和第8章重点讨论系统发育树的构建. 对于现在的问题, 此类方法的细节并不十分重要, 因此假设我们已知一棵可将这些序列联系起来的树, 树上的边长表示树上每条边的相对分化程度.

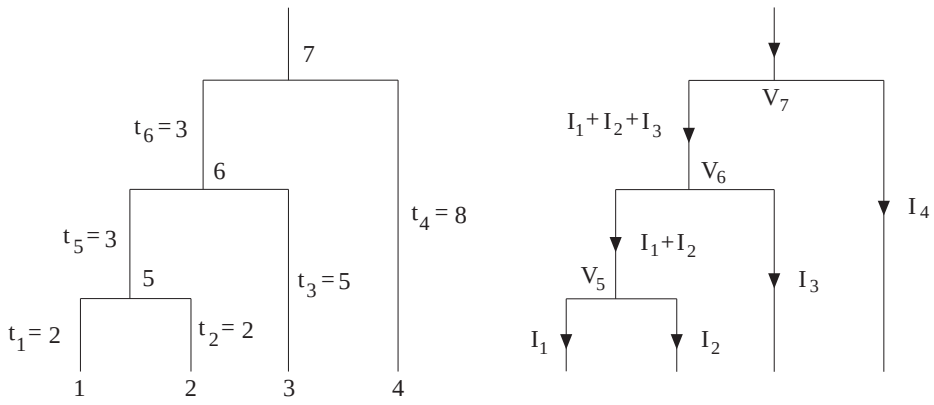


图 5.8: 左图: 带有边长的序列树. 右图: 用于序列加权的" Kirchhoff 定律" 方法中相应的"电流" 与"电压" 值(见正文).

一种直观上最简单的加权方案 [Thompson, Higgins & Gibson 1994b]可以描述如下. 给定一棵由粗细均匀导线做成的树, 我们为树根施加电压 V. 所有叶节点的电势设为零, 我们测量从它们流出的电流并把这些电流记为权重. 显然此树高度分化部分的电流较小, 因此这些权重具备正确的定性性质. 我们可以应用 Kirchoff 定律计算这些电流的大小. 例如在图5.8所示的树上, 令节点 n 处的电流和电压分别是 I_n 和 V_n. 因为常数因子不影响计算, 所以我们令电阻等于边沿时间 (edge-time). 然后得到 $V_5 = 2I_1 = 2I_2$、$V_6 = 2I_1 + 3(I_1 + I_2) = 5I_3$ 还有 $V_7 = 8I_4 = 5I_3 + 3(I_1 + I_2 + I_3)$. 共有3个方程将这四个电流联系起来, 于是有 $I_1 : I_2 : I_3 : I_4 = 20 : 20 : 32 : 47$.

另一种引人注目的简单想法由 Gerstein, Sonnhammer & Chothia [1994]提出. 他们的算法从叶节点开始沿树向上, 逐渐递增权重. 最开始序列的权重被设定为紧挨其上的边之边沿时间. 现在假设到达节点 n. n 上面的边之边沿时间是 t_n, 它将被平分为所有 n 以下叶节点 (leaves below n) 序列的权重, 然后向这些权重递增与它们现有权重值成正比的部分. 形式上, 权重 w_i 的增量 Δw_i 由下式给出

$$\Delta w_i = t_n \frac{w_i}{\sum_{\text{leaves } k \text{ below } n} w_k}. \tag{5.9}$$

直到树根都执行这一操作.

很明显这是一种既简单又有效的算法. 例如图5.8中, 树的权重计算如下: 先令权重为叶节点的边长, 即 $w_1 = w_2 = 2$、$w_3 = 5$ 且 $w_4 = 8$. 在节点5, 它上面的边长3被平均分到 w_1 和 w_2, 使得此二者都是3/2, 因此现在 $w_1 = w_2 = 2 + 3/2 = 3.5$. 在节点6我们发现它上面的边长3被节点1、2和3按比例 3.5 : 3.5 : 5 分开, 使得 $w_1 = w_2 = 3.5 + 3.5/12$ 且 $w_3 = 5 + 3 \times 5/12$. 因为 $w_4 = 8$, 由此得 $w_1 : w_2 : w_3 : w_4 = 35 : 35 : 50 : 64$. 尽管这些权重与 Kirchoff 定律给出的相差不多, 但这两种方法在某种意义上是相反的, 因为如果一棵树只有两个叶节点, 并且一条边比另一条边长, 那么 Kirchoff 会下调较长边的权重而(5.9) 式则上调它的权重.

从 Gauss 参数得到根的权重

有种关于加权的看法认为, 权重应该表示叶节点对树根分布的影响. 如Altschul , Altschul, Carroll & Lipman [1989]所做的一样, 我们可以使这种观点变得精确. 他们方法的基础是由 [Felsenstein 1973] 提出的、应用到连续参数的"修剪" (pruning) 算法. 与字母表中的离散成员不同, 我们拥有一个连续的实值变量, 它就像一个生命体的重量. 同时, 代替替换矩阵的是概率密度, 它定义了将此变量的一个值 x 替换为另一个值 y 的概率. 这种密度的一个简单例子是 Gauss 密度, 其中如果沿着时间为 t 的边那么 $x \to y$ 的概率是 $\exp(-(x-y)^2/(2\sigma^2 t))$. "修剪" 算法随后就像对有限字母表进行操作一样, 只是用积分代替了离散求和 [Felsenstein 1973]. [4]

Felsenstein 的算法为树根的所求参数产生一个 Gauss 分布, 其均值μ线性依赖于叶节点的参数值 x_i, 于是有 $\mu = \sum w_i x_i$. Altschul, Carroll & Lipman [1989] 指出, 应该用这些 w_i 作为权重. 因为它们表现了每个叶节点对于树根的影响.

例子: 三叶树的Altschul-Carroll-Lipman 权重

为了说明如何推导这些权重, 考虑图5.9中简单的三叶树, 其中叶节点 i 有权重值 x_i. 节点 4 (at node 4) 的概率分布由下式给出

$$P(x \text{ at node } 4 \mid L_1, L_2) = K_1 \mathrm{e}^{-\frac{(x-x_1)^2}{2t_1}} \mathrm{e}^{-\frac{(x-x_2)^2}{2t_2}}$$

其中 K_1 是归一化常数. 上式可改写为

$$P(x \text{ at node } 4 \mid L_1, L_2) = K_1 \mathrm{e}^{-\frac{(x-v_1x_1-v_2x_2)^2}{2t_{12}}}$$

[4]从历史上说, 首先出现的是面向连续情形的方法, Felsenstein 为实值参数的 Gauss 分布定义了修剪算法. 在我们引用的文献中, 作者考虑了每个叶节点处的参数分布, 例如每个生命体重量的均值和方差. 令人费解的是, 作者同时引入了不同叶节点间数值的协方差. 我们至今都不清楚如何计算, 例如例如奶牛和猫的体重间的协方差, 对蛋白质而言, 在联配中包含多个相应位点将在原则上允许我们考虑它们之间的相关性. ——原注

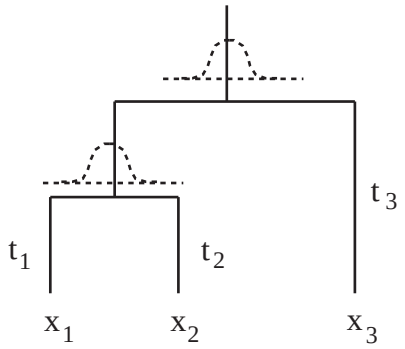


图 5.9: 在推导 Gauss 权重时正文中所提到的树.

其中 $v_1 = t_2/(t_1 + t_2)$、$v_2 = t_1/(t_1 + t_2)$ 并且 $t_{12} = t_1 t_2/(t_1 + t_2)$. 如果我们考虑只有两个叶节点的树, 且其树根在节点4, 那么根分布的均值将是 $\mu = v_1 x_1 + v_2 x_2$, 且权重将是 v_1和v_2. 接着讨论三叶树, 我们进一步发现节点5的分布是

$$P(y \text{ at node } 5 \mid L_1, L_2, L_3) = K_2 \mathrm{e}^{-\frac{(y-x_3)^2}{2t_3}} \int \mathrm{e}^{-\frac{(x-v_1x_1-v_2x_2)^2}{2t_{12}}} \mathrm{e}^{-\frac{(x-y)^2}{2t_4}}\, dx$$

其中 K_2 是归一化常数, 而且此式对节点4处所有可能的 x 值求积分(这就严格等价于在离散字母表的情况下对所有可能的祖先残基分配求和). 这是一个标准的 Gauss 积分, 它可以化简为

$$P(y \text{ at node } 5 \mid L_1, L_2, L_3) = K_3 \mathrm{e}^{-\frac{(y-w_1x_1-w_2x_2-w_3x_3)^2}{2t_{123}}},$$

其中 K_3 是一个新的归一化常数, 且 $t_{123} = t_3\{t_1t_2 + t_4(t_1 + t_2)\}/\Omega$, 而 $\Omega = t_1t_2 + (t_3 + t_4)(t_1 + t_2)$. y 分布, 即树根分布的均值是

$$\mu = w_1x_1 + w_2x_2 + w_3x_3,$$

其中 $w_1 = t_2t_3/\Omega$、$w_2 = t_1t_3/\Omega$ 且 $w_3 = \{t_1t_2 + t_4(t_1 + t_2)\}/\Omega$. 它们就是三叶树的 Altschul-Carroll-Lipman 权重. □

Voronoi 权重

人们还发明了不基于树的加权方案. 例如有一种方法的基础就是"序列空间" (sequence space) 中某个家族之序列的图像. 一般来讲, 有些序列将成簇出现而其他的将广泛分开. Voronoi 方案 [Sibbald & Argos 1990]的基本原理是假设这种不均匀性表现了抽样的影响, 其中就包括自然选择做出的偏向某些门 (phylum) 的"抽样" . 对蛋白质家族中的所有合格序列进行更彻底的搜索, 或重新进行多次进化, 都应该使得某些区域内的分布变平. 为了补偿空位, 我们要使序列的权重正比于其周围空白空间的体积.

如果序列空间是二维的, 或甚至是低维的, 那么我们就可以用计算几何学中的标准方法将空间划分为若干个围绕示例点的区域. 这种标准方法是连接相邻两个点, 然后画出这些连线的垂直平分线并延长, 直到它们相交. 这样划分形成的多边形(二维情形) 称为**Voronoi 图** (Voronoi diagram) [Preparata & Shamos 1985], 它具有如下性质: 围绕每个点的多边形是一个集合, 该集合包括了所有到这个点的距离比到其他点距离都近的点.

序列空间当然是一个高维结构, 因此其 Voronoi 几何很难描绘或计算. 然而我们能够实施它的基本原理, 即从序列空间中对序列进行随机抽样并测试, 以便找到每条序列最接近于家族序列中的哪些序列. 窍门就在抽样中. 其实现如下: 对联配中的每个位置, 从任意序列该位置处出现过的残基中均匀地选择. 如果将与第 i 个家族成

员最近的抽样序列数计为 n_i (如果出现相等情况, 那么就将其平分), 那么我们就可以定义第 i 个权重为 $n_i/\sum_k n_k$.

最大值判别权重

另一种加权方法通过间接方式得到, 它来源于最开始就集中精力重新用公式描述构建模型过程中的主要目标 [Eddy, Mitchison & Durbin 1995]. 相对于最大化家族中序列的似然, 或甚至是最大化从 Bayes 先验得到后验概率, 我们通常更关心对序列是否属于家族做出正确的决策. 因此我们对下面的概率更感兴趣

$$P(M|x) = \frac{P(x|M)P(M)}{P(x|M)P(M) + P(x|R)(1 - P(M))},$$

其中 x 是家族中的一条序列, M 是要对此家族建立的模型, R 是不在此家族中的序列之备择随机模型, $P(M)$ 是一条新序列属于该家族的先验概率. 给定示例训练序列 x^k, 我们愿意最大化将它们都正确分类的概率, 即

$$D = \prod_k P(M|x^k),$$

而非通常情况下基于最大似然方法的 $\prod P(x^k|M)$. 我们称 D 为序列集 x^k 上模型的**判别** (discrimination). 最大化 D 将强调此方法对于家族中关系遥远或不易找到的成员之作用. 易分类序列的 $P(M|x)$ 值将十分接近于1; 为提高其似然 $P(x|M)$ 而改变参数将对 D 产生很小的影响. 而另一方面, 提高 $P(M|x)$ 较小的序列之似然则可能导致较大影响.

使 D 达到最大的参数值被证实可以最大化加权形式的似然, 其中的权重正比于 $1 - P(M|x_i)$, 即将序列 i 错误分类的概率. 对此可以这样理解: 已知如果 $y = \mathrm{e}^x/(K + \mathrm{e}^x)$, 那么

$$\frac{\partial \log y}{\partial x} = \frac{K}{K + \mathrm{e}^x} = (1 - y).$$

如果令 $x = \log\left(\frac{P(x|M)}{P(x|R)}\right)$, 即序列 x 的对数似然比, 那么 $y = P(M|x)$. 因此 $\log D$ 取最大值时, 我们也就达到了对数似然比加权和的最大值, 其中权重是 $1 - P(M|x_i)$, 并且因为随机模型已经固定, 所以这就与模型 M 的加权对数似然的最大值等价. 由此, 最大判别标准实际上就形成了另一套序列加权系统.

但是与此前的系统不同, 这些权重是在有些循环的方式下定义的; 它们依赖于正要被拟合的模型. 当使用最大判别加权方法时, 我们必须使用迭代方法; 初始权重集给出一个模型, 我们从该模型中计算出后验概率 $P(M|x)$, 然后此概率导致新的权重, 于是产生一个新的模型, 如此继续直到此方法收敛. 这种迭代的重估计过程类似于拟合未标记序列集合的 HMM 参数的 EM 算法(见原书第 p. 44 页和第 p. 221 页).

最大判别训练方法的一大优势在于: 它直接优化某种运算的性能, 而我们将把模型用于该运算, 这就确保我们为识别最远序列付出了最大的努力. 但另一方面, 这一点也会导致问题. 如果训练集中存在一条错误分类的序列, 那么为其计较好分值所导致的失真将损害算法在处理分类中正确成员的性能. 不过从某种程度上讲, 所有加权方案都存在相同的问题: 在从示例序列构建的树中, 被错误指派的序列将成为最远的一条序列.

最大熵权重

最后, 我们介绍两种加权方法, 其思想基础都是试图让模型的统计伸展性尽可能好.

假设多序列联配的第 i 列有 k_{ia} 个 a 类残基, 并且总共有 m_i 种不同残基. 为了从这些计数出发通过对每条序列加权得到尽量均匀的分布, 我们可以为序列 k 选择权重 $1/(m_i k_{ix_i^k})$. 然后最大似然估计会给出一个分布 $p_{ia} = k_{ia}/(m_i k_{ia}) = 1/m_i$, 即所有出现在这一列的残基将有相同概率. 为了说明这种思想, 假设有十条序列在某位点都是残基 A, 而另一条序列在该位点是残基 B, 于是 A和 B的未加权频率分别是 $c_{\mathtt{A}} = \frac{10}{11}$ 和 $c_{\mathtt{B}} = \frac{1}{11}$. 这十条序列的权重是 $w_1 = w_2 = \cdots = w_{10} = 1/(2 \times 10) = 0.05$, 而 $w_{11} = 1/(2 \times 1) = 0.5$, 这就使 A 和 B 的全部权重的效果相同.

以上只考虑了一列. 当每条序列只有一个权重时, 我们当然不可能使联配中所有列都达到均匀分布. 但是对所有列取平均后, 我们就希望得到合理的权重. 此时权重按下式计算:

$$w_k = \sum_i \frac{1}{m_i k_{ix_i^k}},$$

然后归一化使其和为1. 这种加权方案由 [Henikoff & Henikoff 1994]提出.

除了求平均外, 还有另一种方法可将不同列的信息组合到一起, 而且它的理论证明简单. 分布"均匀性" (uniformity) 的一种标准度量是熵 (entropy) (见 11.8 节), 熵越大, 分布就越均匀. 事实上容易发现, 上述选出的基于单列的权重使该列的分布 p_{ia} 的熵最大化. 一个 HMM 定义了序列的概率分布, 因此从单列加权到所有列的一个自然扩展就是最大化整个 HMM 分布的熵 [Krogh & Mitchison 1995]. 也许令人吃惊的是, 我们将看到这与最大判别加权密切相关.

让我们考虑一个无空位联配中的所有位点. 然后对每个位点的熵求和, 并选取使此和达到最大的权重; 即最大化 $\sum_i H_i(w_\bullet) + \lambda \sum_k w_k$, 其中 $H_i(w_\bullet) = -\sum_a p_{ia} \log p_{ia}$, 且 p_{ia} 是在第 i 个位点处残基 a 的加权频率, 计算方法同上.

例如假设我们有序列 $x^1 =$ `AFA`、$x^2 =$ `AAC` 和 $x^3 =$ `DAC`. 令它们的权重分别为 w_1、w_2 和 w_3, 于是每个位点处的熵为

$$\begin{aligned}
H_1(w_\bullet) &= -(w_1 + w_2)\log(w_1 + w_2) - w_3 \log w_3, \\
H_2(w_\bullet) &= -w_1 \log w_1 - (w_2 + w_3)\log(w_2 + w_3), \\
H_3(w_\bullet) &= -w_1 \log w_1 - (w_2 + w_3)\log(w_2 + w_3).
\end{aligned}$$

我们假设权重之和是1, 因此当对熵求微分和求最大值的时候我们必须使用 Lagrange 乘数项 $\lambda \sum_k w_k$. 令 $H_1(w_\bullet) + H_2(w_\bullet) + H_3(w_\bullet) + \lambda \sum_k w_k$ 的导数为零得到 $(w_1 + w_2)w_1^2 = (w_1 + w_2)(w_2 + w_3)^2 = w_3(w_2 + w_3)^2$, 由此得到 $w_1 = w_3 = 0.5$, $w_2 = 0$. 这使得每列的频率都相同, 即我们想要的目的. 如果认为给一条序列赋权重0显得有些奇怪, 那么请注意 x^2 中每个位点处的残基总可以在其他两条序列中找到. 从直观上看, x^2 位于 x^1 和 x^3 "之间" (实际上, 如果从简约法导出的进化重构关系看, 它很可能就是 x^1 和 x^3 的祖先序列; 见第 7 章).

从另一个角度看待这个例子的结果就是, 如果我们假设模型概率是加权计数频率, 就像加权最大似然过程所做的一样, 那么得到的模型就为所有原始序列 x^1、x^2 和 x^3 都分配相等的概率. 这看起来十分合理, 因为对正在建立其模型的家族而言, 所有的示例序列都应该被等同地看作是该家族中的好成员. 事实上 Krogh & Mitchison [1995]证明了最大熵过程为示例序列分配的权重使得序列的某个子集(或全部) 在所得模型中有非零权重与相等的概率, 或者它们有较大的概率与零权重. 前者可以被看作是被整个序列集所占据的序列空间区域之边界点, 而后者则是内点.

此外, 经验测试表明, 最大熵权重在如下意义上是最优的, 即它们能够使为任意示例序列分配的最小分值达到最大 [Krogh & Mitchison 1995]. 这是上一节最大判别权重所指明的标准之绝对形式; 与简单地为最弱匹配赋予最强权重不同, 所有参数拟合工作都用于增加其分值, 直到等于其他非零已加权序列的分值为止. 虽然最大熵加权方法满足了一个吸引人的目标, 但是它面临与最大判别方法相同的问题: 如果某条序列是离群的 (outlier), 即它不是该家族的正式成员, 那么此方法将强制性地将其纳入该家族, 并且有可能在处理所有其他序列的性能上付出可观代价. 另外从直观上看, 拒绝接受某些序列的所有信息也许不太可取.

练习

5.6 使用除 Voronoi (它需要序列的随机抽样) 以外, 上文介绍过的所有加权方法, 计算下面序列集的权重: `AGAA`, `CCTC`, `AGTC`.

5.9 补充读物

为了寻找序列家族中的新成员, 人们在二十世纪80年代提出了 PSSM 方法, 虽然当时人们并不总是使用显式的、基于概率的推导过程得到其矩阵值. PSSM 还有

其他名称, 例如**权重矩阵** (weight matrices) [Staden 1988]. 最近, 使用相关方法的文献包括 Stormo [1990], Henikoff & Henikoff [1994]和 Tatusov, Altschul & Koonin [1994].

非概率论形式的 profile 已经有很长一段历史, 人们也提出并检验了 profile 方法的很多变体. Thompson, Higgins & Gibson [1994b] 和 Luthy, Xenarios & Bucher [1994] 都报道了使用 BLOSUM 矩阵 [Henikoff & Henikoff 1992] 而非 PAM 矩阵为加权序列时得到的改进. Thompson, Thompson, Higgins & Gibson [1994b] 同时改进了空位处理方法.

有些方法已经建议将结构信息包括到 profile 中. Luthy, McLachlan & Eisenberg [1991] 在六种不同的结构环境中估计了替换矩阵: 三种二级结构元素 α-螺旋、β-折叠和"其他" , 与内/外分类组合到一起, 这种内外分类取决于某氨基酸暴露于溶剂的情况. Bowie, Luthy & Eisenberg [1991] 和 Wilmanns & Eisenberg [1993] 包含了结构 profile 的其他变化.

早些时候, Baldi *et al.* [1994] 将 profile HMM 用于为球蛋白、免疫蛋白和激酶建模. 他们的工作同时也引入了一种不同的估计方法: 基于梯度下降的方法, 另见 Baldi & Chauvin [1994]. 相同的 profile HMM 基本结构也已经用于其他一些领域. 有人以 PFAM [Sonnhammer, Eddy & Durbin 1997] 为名称建立了所有大蛋白质家族的 HMM 库. 称作 PROSITE [Bairoch, Bucher & Hofmann 1997] 的正则表达式 (regular expression) 库正在向本质上与 profile HMM 相同的方向扩展 [Bucher *et al.* 1996]. Profile HMM 在 DNA 方面也有一些应用, 例如它可以在大规模基因组序列中用于寻找 DNA 重复序列家族的成员.

第六章

多序列联配方法

在第5章中, 我们假设合理的多序列联配已知, 并且它为构建 profile HMM 提供了起点. 现在考虑什么是合理的多序列联配, 以及从未联配序列中自动构建它的方法.

通常, 我们仅从一级序列推断多序列联配. 使用蛋白序列进化的专业知识, 生物学家可以手工获得高质量的多序列联配. 这些知识主要来自经验, 其中的重要因素包括: 联配中特定种类的列, 如高度保守的残基或内部的疏水残基; 二级结构和三级结构的影响, 例如外露的 β 折叠片上疏水区与亲水区的交替; 还有一些预期的插入型和缺失型, 它们容易同保守区域块交替出现. 另外, 序列间的系统发育关系会对列间以及空位型间发生的变化产生约束. RNA 联配也包含相似的知识, 但是除此之外它们通常还会受到二级结构模型的强烈约束, 而在许多情况下这种二级结构模型也是由一级序列数据推断出来的(第10章) .

手工多序列联配过程相当枯燥乏味. 因此自动多序列联配方法就成为计算生物学中广泛研究的课题. 通常, 一种自动方法必须含有计分方法, 使得较好的多序列联配能获得较好的分数. 我们应该仔细区分多序列联配计分问题与从所有可能多序列联配中找出最优联配的问题. 多序列联配程序的描述更倾向于强调联配算法而非计分函数. 然而现在需要明确的是, 在概率论建模中计分函数才是我们首要关注的, 其次才是算法, 尽管它也很重要. 在概率论建模中我们的目标之一就是将尽可能多的专家评价标准整合到我们的计分过程中.

因此, 在讨论自动多序列联配问题之初, 先仔细考虑一下我们到底想要做什么. 首先, 我们分别从结构和进化的角度考查一下多序列联配的含义. 然后我们考虑如下问题: 怎样才能以最好的方式将各种生物学标准转变为一个数值计分方案, 以便程序能够识别一个好的多序列联配. 我们对不同多序列联配程序所使用的各种方法进行了测试. 本章最后将描述基于第5章的 profile HMM 的全概率论多序列联配方法, 并比较 profile HMM 之于其他方法的优缺点. 我们将主要关注蛋白质联配, 尽管大部分讨论也适用于 DNA 联配. (RNA 联配 因碱基配对的长程相互作用而变得复杂, 至第10章再讨论.)

6.1 多序列联配的含意是什么

在多序列联配中, 序列间的同源(homologous)残基被联配到相同的列上. "同源" 指结构上和进化上的双重含义. 理想情形是, 一列联配上的残基应该处于相似的三维结构位置并且全都应该由共同的祖先残基分化而来. 例如图6.1显示了来自免疫球蛋白超家族(immunoglobulin superfamily)的10条序列之人工多序列联配结果. 其中一条序列 (1tlk, 端素蛋白 (telokin)) 的晶体结构已知. 它的结构以及它于其他相关序列间的联配揭示了 I-型(I-set) 免疫球蛋白超家族折叠的保守特性, 包括8条保守的 β 链与序列中的特定关键残基, 例如在形成折叠结构核心二硫键(disulfide bond) 的 b 链和 f 链中存在的2个完全保守的半胱氨酸. 我们根据这些专业的结构知识, 将来自不同神经细胞黏着分子(neural cell adhesion molecule) 的其他9条序列手工联配到 1tlk 上.

```
structure:  ...aaaaa...bbbbbbbbbb.....ccccccCCC..C........ddd
1tlk        ILDMDVVEGSAARFDCKVEGY--PDPEVMWFKDDNP--VKESR----HFQ
AXO1_RAT    RDPVKTHEGWGVMLPCNPPAHY-PGLSYRWLLNEFPNFIPTDGR---HFV
AXO1_RAT    ISDTEADIGSNLRWGCAAAGK--PRPMVRWLRNGEP--LASQN----RVE
AXO1_RAT    RRLIPAARGGEISILCQPRAA--PKATILWSKGTEI--LGNST----RVT
AXO1_RAT    ----DINVGDNLTLQCHASHDPTMDLTFTWTLDDFPIDFDKPGGHYRRAS
NCA2_HUMAN  PTPQEFREGEDAVIVCDVVSS--LPPTIIWKHKGRD--VILKKDV--RFI
NCA2_HUMAN  PSQGEISVGESKFFLCQVAGDA-KDKDISWFSPNGEK-LTPNQQ---RIS
NCA2_HUMAN  IVNATANLGQSVTLVCDAEGF--PEPTMSWTKDGEQ--IEQEEDDE-KYI
NRG_DROME   RRQSLALRGKRMELFCIYGGT--PLPQTVWSKDGQR--IQWSD----RIT
NRG_DROME   PQNYEVAAGQSATFRCNEAHDDTLEIEIDWWKDGQS--IDFEAQP--RFV
consensus:  ........G..+.+.C.+.........+.W........+........++

structure:  ddd.....eeeeee.......fffffffff.......gggggggggggg.
1tlk        IDYDEEGNCSLTISEVCGDDDAKYTCKAVNSL-----GEATCTAELLVET
AXO1_RAT    SQTT----GNLYIARTNASDLGNYSCLATSHMDFSTKSVFSKFAQLNLAA
AXO1_RAT    VLA-----GDLRFSKLSLEDSGMYQCVAENKH-----GTIYASAELAVQA
AXO1_RAT    VTSD----GTLIIRNISRSDEGKYTCFAENFM-----GKANSTGILSVRD
AXO1_RAT    AKETI---GDLTILNAHVRHGGKYTCMAQTVV-----DGTSKEATVLVRG
NCA2_HUMAN  VLSN----NYLQIRGIKKTDEGTYRCEGRILARG---EINFKDIQVIVNV
NCA2_HUMAN  VVWNDDSSSTLTIYNANIDDAGIYKCVVTGEDG----SESEATVNVKIFQ
NCA2_HUMAN  FSDDSS---QLTIKKVDKNDEAEYICIAENKA-----GEQDATIHLKVFA
NRG_DROME   QGHYG---KSLVIRQTNFDDAGTYTCDVSNGVG----NAQSFSIILNVNS
NRG_DROME   KTND----NSLTIAKTMELDSGEYTCVARTRL-----DEATARANLIVQD
consensus:  ..........L.+..+...+.+.Y.C................+.+.+..
```

图 6.1: 10个 I-型免疫球蛋白超家族结构域的多序列联配, 修改自Harpaz & Chothia [1994]. 序列左边是序列在PDB或SWISS-PROT数据库中的标识符. 序列上面注释了端素蛋白的结构, 即 1tlk 的8条 β 链(a-g ; C 表示c ’链). 联配好的列在序列的下面一行注释, 如果所有残基完全一样那么就用字母表示, 高度保守则用+号表示.

除了高等同度序列的平凡例子外, 我们不可能毫不含混地认定结构上或进化上的同源位置并创建单一的”正确” 多序列联配. 因为蛋白结构也在进化(尽管比蛋白序列要慢的多), 所以我们不能期望两条不同序列的蛋白结构完全重合(superposable). Chothia & Lesk [1986]检查了几个蛋白家族的成对结构联配, 并发现对于存在分歧但却有明确同源性(30%等同) 的一对已知蛋白序列来讲, 一般只有50%的单个残基在这两个结构上是重合的(图6.2). 球蛋白家族在计算工作中经常被当作”典型的”蛋白家族, 但事实上它是一个例外: 在各条分化序列中, 球蛋白的整个结构几乎都是保守的. 甚至就连”结构重合” 之定义都是主观的, 而且不同专家对其的理解也不尽相同.

原则上, 即使结构发生分化, 我们也总可以找到一个清晰正确、能反映进化关系的联配. 但事实上, 推断进化上正确的联配比推断结构联配要困难很多. 结构联配有独立的参考点(晶体或 NMR 结构的重合), 然而一个序列家族的残基进化历史却不可能从任何来源独立获得; 它只能从序列联配中推断. 因为序列容易比结构分化地更快, 所以在结构水平无法联配的蛋白部分通常在序列水平同样也不能联配.

因此, 我们定义单个”正确” 联配的能力依赖于联配序列间的相关性. 非常相似的序列间的联配通常来说是明确的, 但是我们对这类联配并不感兴趣; 一个简单的程序就可以得到正确联配. 对我们感兴趣的情况(例如成对序列等同度平均约为30%的一个蛋白家族), 我们必须注意到没有客观方法能够定义清晰正确的联配. 通常来讲, 无论家族中的序列如何分化, 我们几乎都能鉴别出关键残基的某个小子集, 而这些关键残基在家族的所有序列中都能形成清晰的联配 [Harpaz & Chothia 1994]; 核心结构元件也倾向于保守并能构成有意义的联配; 但由于结构进化和序列分化, 其他区域可能无法形成有意义的联配.

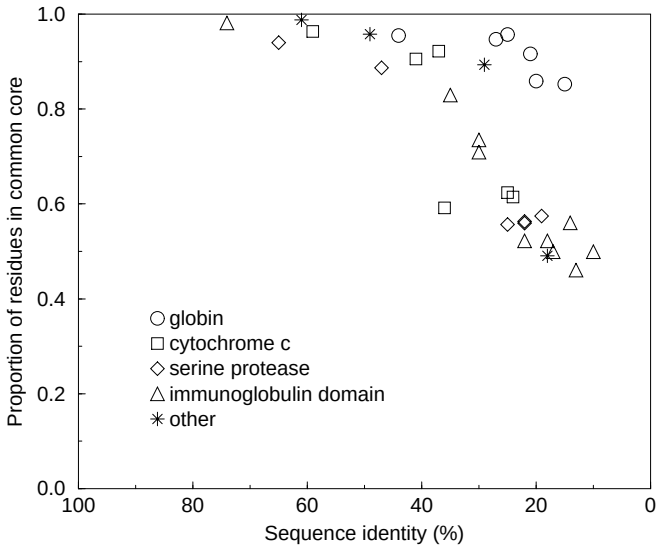


图 6.2: 二序列联配中结构重合残基所占比例是序列等同度(sequence identity) 的函数; 本图根据 Chothia & Lesk [1986]中的数据重新绘制. "其他" (other) 结构联配包括两个二氢叶酸还原酶(dihydrofolate reductase) 、两个溶菌酶(lysozyme) 、质体蓝素(plastocyanin) /天青蛋白(azurin) 和木瓜蛋白酶 (papain) /猕猴桃蛋白酶 (actinidin) 的二序列联配.

多序列联配的质量评估必须考虑以上问题. 因此, 例如要求序列联配程序产生出与手工结构联配**完全**相同的联配, 就意味着向联配中加入与手工结果中相同的、关于如何"联配" 结构上无法联配区域的无意义偏差. 相比之下, 我们更应该关注对应于关键残基和核心结构元件的列子集, 我们对其联配更有把握 [McClure, Vasi & Fitch 1994].

6.2 为多序列联配计分

我们的计分系统最少要考虑多序列联配的两个重要特征: (1) 某些位点比其他位点更保守, 例如采用位置特异的计分方案; (2) 序列间并不独立, 而是通过一棵系统发育树相互关联. 因此为多序列联配计分的一种理想方法应该是, 指定分子序列进化的一个全概率论模型. 给定序列的正确系统发育树, 多序列联配的概率为: 通过祖先中间序列产生该联配的所有必需进化事件之概率乘积, 乘以根祖先序列之先验概率. 我们所需的进化模型很复杂. 进化变化的概率将取决于沿树各个分支的进化时间和自然选择所带来的位置特异的结构与功能约束, 以便关键残基和结构元件是保守的. 因此, 该模型下高概率的联配将是结构上与进化上的好联配.

不幸的是, 我们没有足够数据以确定如此复杂进化模型的参数. 因此必须简化一些假设. 本章中, 我们主要关注的是具备操作可行性的近似方法, 在为结构上相容的残基之联配进行位置特异计分的同时, 部分或全部忽略系统发育树. 在第7章和第8章中, 我们将会考查更明确的系统发育树和分子进化模型, 其中大多数使用位置独立而非位置特异的近似进化模型.

几乎所有的联配方法都会假设联配的各列之间在统计上是独立的. 于是计分函数就可以写为:

$$S(m) = G + \sum_i S(m_i), \tag{6.1}$$

其中 m_i 是多序列联配 m 的第 i 列, $S(m_i)$ 是第 i 列的分数, G 是联配中空位的计分函数.

这里的 G 以非特异函数的形式出现, 因为在多序列联配中为空位计分的方法间存在很大区别. 最简单的方法是将空位符当作一种额外的残基, 于是 $S(m) = \sum_i S(m_i)$. 然而, 大多数多序列联配方法使用仿射计分函数, 即打开空位比延长空位罚分要多, 这样连续的空位残基就不再被视为独立了. 为简单起见, 下面几个段落将关注为无空位已联配残基列计分的 $S(m_i)$ 的定义.

最小熵

我们现在定义一些符号. 如上, m 是一个多序列联配, m_i^j 是序列 j 上第 i 列的符号. c_{ia} 是残基 a 在第 i 列的观测计数; $c_{ia} = \sum_j \delta(m_i^j = a)$, 其中当 $m_i^j = a$ 时 $\delta(m_i^j = a)$ 为1, 否则为0. 令 m_i 为第 i 列联配上的符号列 $m_i^1, \ldots, m_i^N$, 对于有 K 个不同残基的字母表, 令 c_i 为第 i 列中观测字符的计数向量 $c_{i1}, \ldots, c_{iK}$.

如果这些序列的系统发育树有许多中间祖先, 那么序列间的统计相关性就会相当复杂(见第7章). 如果我们假设所有序列独立产生, 那么计分问题就会变得十分简单. 如果我们假设在每列**内**以及各列**间**的残基都独立, 那么每列 m_i 的概率就是

$$P(m_i) = \prod_a P_{ia}^{c_{ia}}, \tag{6.2}$$

其中 p_{ia} 是残基 a 在第 i 列的概率, 我们可以定义列的分数为这个概率的负对数:

$$S(m_i) = -\sum_a c_{ia} \log p_{ia}. \tag{6.3}$$

这是一种**熵**度量, 它与信息论中的 Shannon 熵方程(第11章) 有直接关系. 它可以方便地度量在已联配残基列中观测到的可变性. 列变化越剧烈, 其熵值也就越大. 完全保守的列计分为0. 我们可以将好的联配定义为使总熵(例如 $\sum_i S(m_i)$) 达到最小的那个联配.

与前面(第5章) 一样, 可以用计数 c_{ia} 估计参数 p_{ia}; 例如其最大似然估计是

$$p_{ia} = \frac{c_{ia}}{\sum_{a'} c_{ia'}}. \tag{6.4}$$

在实践中, 我们通常用伪计数和 Dirichlet 先验为此概率估计实施正规化.

这明显与该问题的 HMM 表示法相似. Profile HMM 更进一步, 在联配中用概率论方法建立插入和缺失的模型. 在放弃进化树并假设序列间独立的情况下, 我们能直接估计包括列内残基的概率和插入、缺失概率的位置特异模型. 标准 profile 也做了相似的假设.

如果精心挑选序列家族的代表序列, 那么假设序列相互独立就是合理的. 虽然序列的样本有偏(biased) 并且某些进化子家族相对其他子家族而言有过多或过少的代表序列, 但是这个假设通常都成立. 针对此问题, 人们提出多种基于树的加权方案, 以便从某种程度上补偿序列独立假设的缺陷(见第5章).

配对和: SP 分数

为多序列联配计分的标准方法虽然不属于 HMM 形式体系, 但却与之相似, 因为它也没有借助系统发育树而且假设列在统计上是独立的. 我们用所谓的"配对和" (sum of pairs , SP) 函数为列计分, 该函数使用替换计分矩阵. 列的 SP 分数定义为:

$$S(m_i) = \sum_{k<l} s(m_i^k, m_i^l), \tag{6.5}$$

其中分数 $s(a, b)$ 来自类似于 PAM 或 BLOSUM 的替换计分矩阵. 使用简单线性空位罚分时, 处理空位如下: 定义 $s(a, -)$ 和 $s(-, a)$ 为空位罚分, 定义 $s(-, -)$ 为0. 否则空位要单独计分(例如使用仿射空位罚分时).

把某列的所有成对替换分数加起来似乎是一件很自然的事情. 然而, 替换分数通常来自成对比较的对数几率分数. 将其扩展到多序列联配, 例如三序列联配(three-way alignment) 的 SP 分数应该是 $\log(p_{abc}/q_aq_bq_c)$ 而非 $\log(p_{ab}/q_aq_b)+\log(p_{bc}/q_bq_c)+\log(p_{ac}/q_aq_c)$. SP 分数没有概率论证明; 为每条序列计分时就好像它来自其他 $N-1$ 条序列而不是一个单独祖先. 随着序列数目的增加, 进化事件被重复计数的问题也随之出现. Altschul, Carroll & Lipman [1989] 认识到这个问题并提出了一套加权方案, 以便在某种程度上补偿 SP 分数的缺陷(第5章).

例子: 关于 SP 分数的一个问题

关于标准 SP 多序列计分系统所存在的问题, 我们列举一个直观而具体的例子: 考虑包含 N 条序列的多序列联配, 由于功能上的某种重要原因, 它们在某特定位置都含有一个亮氨酸 (L). 根据 BLOSUM50 替换矩阵(图2.2), L 联配上 L 的分数是5, 所以这列的 SP 分数是 $5\times N(N-1)/2$, 其中 $N(N-1)/2$ 是该列中符号对的数目. 如果换成该列有一个甘氨酸 (G) 和 $N-1$ 个 L, 那么由于 G-L 配对是-4分而不是+5分, 该列的分数要低 $9\times(N-1)$, 而且 $N-1$ 个对都受到了影响. 也就是含一个 G 的列的 SP 分值比全是 L 的列的分值要低

$$\frac{9(N-1)}{5N(N-1)/2}=\frac{18}{5N}.$$

注意它与 N 成反比; 正确联配和错误联配间分值的相对差异随联配中序列数目的增加而**减少**. 这明显是违反直觉的. 随着我们拥有保守亮氨酸证据的增加, 这种相对差异也应该增加. 见原书第105页的另一个例子. □

6.3 多维动态规划

我们先把计分问题放到一边, 现在考虑构造多序列联配的算法.

可以将二序列动态规划联配方法(第2章) 推广到 N 条序列的联配. 然而我们马上就会看到, 当序列数目稍微增加时, 这种方法就不可行了. 假设联配的列之间在统计上是独立的, 我们暂时还假设用线性空位罚分 $\gamma(g)=gd$ 是长度为 g 的空位计分, 每个空位的罚分是 d. 于是我们可以对每列 i 的分数 $S(m_i)$ 求和, 以计算联配的总分 $S(m)$:

$$S(m)=\sum_i S(m_i). \tag{6.6}$$

带仿射空位罚分和多个状态的多维动态规划方法是可行的, 它使用与第2章中类似的方法, 只是高维情形下其形式变得更加冗长.

定义 $\alpha_{i_1,i_2,\ldots,i_N}$ 为结束于$x^1_{i_1},x^2_{i_2},\ldots,x^N_{i_N}$的子序列的联配的最大分数, 则动态规划算法如下:

$$\alpha_{i_1,i_2,\ldots,i_N}=\max\begin{cases}
\alpha_{i_1-1,i_2-1,\ldots,i_N-1} & + & S(x^1_{i_1},x^2_{i_2},\ldots,x^N_{i_N}),\\
\alpha_{i_1,i_2-1,\ldots,i_N-1} & + & S(-,x^2_{i_2},\ldots,x^N_{i_N}),\\
\alpha_{i_1-1,i_2,i_3-1,\ldots,i_N-1} & + & S(x^1_{i_1},-,\ldots,x^N_{i_N}),\\
 & \vdots & \\
\alpha_{i_1-1,i_2-1,\ldots,i_N} & + & S(x^1_{i_1},x^2_{i_2},\ldots,-),\\
\alpha_{i_1,i_2,i_3-1,\ldots,i_N-1} & + & S(-,-,\ldots,x^N_{i_N}),\\
 & \vdots & \\
\alpha_{i_1,i_2-1,\ldots,i_{N-1}-1,i_N} & + & S(-,x^2_{i_2},\ldots,-),\\
 & \vdots &
\end{cases} \tag{6.7}$$

除了所有残基都被空位所代替的一种情况外, 其他所有的空位组合方式都出现了. 共有 2^N-1 种此类组合. 我们省略了算法的起始、终止和回溯步骤, 它们与一般的二序列动态规划算法类似.

通过引入取0或1的 Δ_i 可以简化记号, 我们还可以定义"乘积"

$$\Delta_i \cdot x = \begin{cases} x & \text{if } \Delta_i = 1, \\ - & \text{if } \Delta_i = 0. \end{cases} \tag{6.8}$$

现在递归式可写为 [Sankoff & Cedergren 1983; Waterman 1995]:

$$\alpha_{i_1,i_2,\ldots,i_N} = \max_{\triangle_1+\ldots+\triangle_N>0} \{\alpha_{i_1-\triangle_1,i_2-\triangle_2,\ldots,i_N-\triangle_N} + S(\triangle_1 \cdot x^1_{i_1}, \triangle_2 \cdot x^2_{i_2}, \ldots, \triangle_N \cdot x^N_{i_N})\}. \tag{6.9}$$

该算法需要计算含 $L_1 L_2 \ldots L_N$ 个输入的全局动态规划矩阵. 为了计算每个输入, 我们需要对一列中空位的全部 $2^N - 1$ 种组合求最大值, 但这不包括所有 $\triangle_k$ 均为0的情况. 假设所有序列长度基本均为 $\bar{L}$, 那么多维动态规划算法的空间复杂度为 $O(\bar{L}^N)$, 时间复杂度为 $O(2^N \bar{L}^N)$.

注意我们并没有指定列分数 $S(m_i)$ 的函数形式. 多维动态规划正确运行的唯一必要假设是列分数独立. 原则上讲, 我们可以使用某种进化模型计算 $S(m_i)$[Sankoff 1975].

练习

6.1 假设我们有一些长度为50个残基的序列, 并且每2个残基间的比较需要在电脑上花费1秒的CPU 时间. 那么4条序列的联配就要花费 $(2L)^{N-2} = 10^{2N-4} = 10^4$ 秒(几个小时). 如果内存无限大并且我们愿意在太阳烧尽前的50亿年时间中一直等待计算结果, 请问我们的计算机能联配多少条序列?

MSA

由 Carrillo & Lipman [1988]设计的一种巧妙算法可以减少我们需要考虑的多维动态规划矩阵的体积. 多序列联配软件 MSA [Lipman, Altschul & Kececioglu 1989]实现了该算法. 最优情况下 MSA 可以联配不多于5到7条长度合理 (200-300个残基) 的蛋白序列.

Carrillo & Lipman 采用 SP 计分系统为残基和空位计分. 此处我们假设一个多序列联配的分数是由多序列联配定义的所有二序列联配的分数之和; 关于这种计分的更广泛定义可见[Altschul 1989]. 令 a^{kl} 表示序列 k 和 l 间的二序列联配. 那么完整联配的分数就是

$$S(a) = \sum_{k<l} S(a^{kl}). \tag{6.10}$$

令 $\hat{a}^{kl}$ 表示 k 与 l 间的最优**二序列** (pairwise) 联配结果, 我们可以在 $O(\bar{L}^2)$ 时间内用标准动态规划算法算出. 显然 $S(a^{kl}) \le S(\hat{a}^{kl})$.

将此简单结论与 SP 计分系统的定义结合起来, 我们就可以得到最优多序列联配中任意二序列联配的分数下界. 如果临时假设最优多序列联配的分数下界是 $\sigma(a)$, 那么 $\sigma(a) \le S(a)$. 由以上讨论和 SP 分数的定义可知, 对最优多序列联配 a 有

$$\sigma(a) \le S(a^{kl}) - S(\hat{a}^{kl}) + \sum_{k'<l'} S(\hat{a}^{k'l'})$$

于是有

$$S(a^{kl}) \ge \beta^{kl}$$

$$\text{其中} \quad \beta^{kl} = \sigma(a) + S(\hat{a}^{kl}) - \sum_{k'<l'} S(\hat{a}^{k'l'}).$$

由此便知, 我们只需考虑 k 与 l 间分数比 β^{kl} 高的二序列联配. 容易计算下界 β^{kl}. 我们可以用任何一个快速启发式多序列联配算法(例如下面提到的某种渐进联配算法) 得到一个好的边界 $\sigma(a)$. 使用标准二序列联配, 我们可以计算出所有 $N(N-1)/2$ 个最优二序列联配 $\hat{a}^{kl}$, 并为其计分. 这些边界越高, 需要计算的动态规

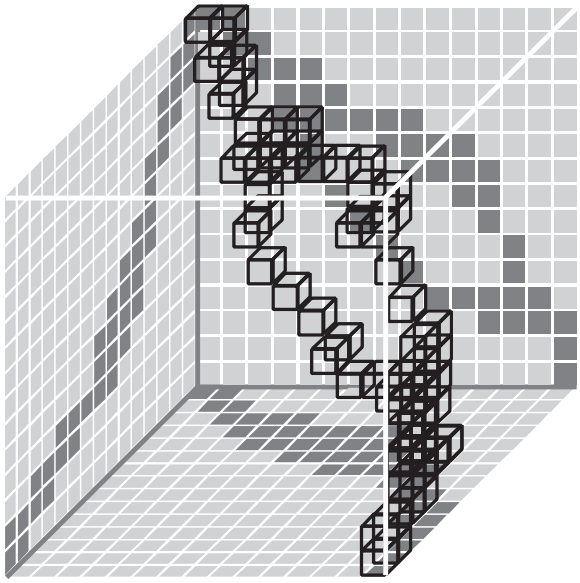


图 6.3: Carrillo & Lipman 的算法可以将最优联配的搜索过程限制在多维动态规划矩阵的子集中, 图中显示了三维情形. 深灰色的是集合 B^{kl}, 而搜索过程被限制在用黑色做轮廓的单元中.

划矩阵的体积就越小, 算法也就越快. (实际上, MSA 默认地以启发式方法选取一个较高的 β^{kl}, 因此它并不能保证得到最优联配.)

现在对每对 k, l, 我们都能找到坐标对 (i_k, i_l) 的完全集 B^{kl} 使得经过 (i_k, i_l) 的 x^k 到 x^l 的最优联配分数比 β^{kl} 大. 在整个二序列动态规划表格里的每个单元中, 该方法都对前向和后向 Viterbi 分值求和并检测此结果是否比 β^{kl} 大, 于是计算这个集合所需的时间是 $O(\bar{L}^2)$. 随后费时的多维动态规划算法就仅限于评估所有这些集合的交集中的单元: 即对所有 k, l 使得 (i_k, i_l) 在 B^{kl} 之内的单元 $(i_1, i_2, \ldots, i_N)$ (图6.3). 处理交集矩阵以及高效地执行动态规划计算都需要很多技巧, Gupta, Kececioglu & Schaffer [1995] 给出了其中的细节.

Altschul & Lipman [1989] 将 Carrillo-Lipman 的算法理论扩展到更接近现实的计分系统, 其方法基于进化星和树 (evolutionary stars and trees) 而非 SP 方法, 但现在我们并不清楚这些想法是如何实现的.

6.4 渐进联配方法

渐进联配 (progressive alignment) 可能是最常用的多序列联配方法. 它由一系列二序列联配构成. 最初, 选出两条序列并用标准二序列联配方法对其实施联配; 固定该联配结果. 然后选出第三条序列并且将其与第一个联配结果进行联配, 然后反复迭代该过程直到所有序列都被联配为止.

渐进联配策略是由许多作者一起提出的 [Hogeweg & Hesper 1984; Waterman & Perlwitz 1984; Feng & Doolittle 1987; Taylor 1987; Barton & Sternberg 1987; Higgins & Sharp 1989]. 这些算法略有不同: (1) 用于联配的先后次序之挑选方法不同; (2) 渐进过程是只包含将序列与单个增长的联配结果做联配, 或是将序列子家族建立到树结构上, 并在某一点对序列的联配结果做联配; (3) 用于将序列或联配结果同已有的联配结果进行联配和计分的过程不同.

渐进联配是启发式的: 它并未将计分过程从优化算法中分离出来. 它也不能直接优化联配正确性的任何全局计分函数. 渐进联配的优势是快速高效, 并且大多数情况下联配结果是合理的.

渐进联配算法中最重要的启发式规则是先联配最相似的序列对. 这些序列间的联配最可靠. 许多算法会构造一棵"指导树" (guide tree) . 它是一棵二叉树, 叶节点

表示序列, 内部节点表示联配. 根节点表示一个完整的多序列联配. 离根最远的节点表示最相似的序列对. 构造指导树的方法和构建系统发育树的方法很相似(第7章), 但是一般来讲指导树都是"快速而粗放" (quick and dirty) 的, 因此不适合做严谨的系统发育推断.

Feng-Doolittle 渐进多序列联配

Feng-Doolittle 算法是一种早期多序列联配算法 [Feng & Doolittle 1987]. 总体上, 算法如下:

算法: Feng-Doolittle 渐进联配

(i) 用标准二序列联配方法计算 N 条序列中所有 $N(N-1)/2$ 个二序列间的距离, 并将其填入一个上三角矩阵[1], 这里的成对"距离" 由转换原始联配分数近似得到.

(ii) 用 Fitch & Margoliash [1967a] 中提到的聚类算法, 从距离矩阵构建指导树.

(iii) 从添加到树上的第一个节点开始, 联配各子节点(可能是二条序列, 也可能是一条序列和一个联配, 也可能是两个联配). 依照其他节点添加到树上的顺序(即按相似性由高到低的顺序), 对其重复该步骤直到所有的序列都已经被联配. ◁

将联配分数转换为距离的方法不需要十分精确, 因为我们的目的只是构建一棵近似的指导树, 而不是一棵进化树. Feng & Doolittle 计算距离 D 为

$$D = -\log S_{\text{eff}} = -\log \frac{S_{\text{obs}} - S_{\text{rand}}}{S_{\max} - S_{\text{rand}}}, \tag{6.11}$$

其中 S_{obs} 是观测到的二序列联配分数; $S_{\max}$ 是最大分数, 即两条序列分别与自己联配的分数之平均; S_{rand} 是长度和残基含量都相同的两条随机序列之期望联配分数. S_{rand} 既可以由随机打乱的两条序列产生, 也可以用 Feng & Doolittle [1996] 的近似计算得到. 于是有效分数 S_{eff} 就可以看作是归一化后的相似性百分比; 随着进化距离的增长, 它按指数衰减到0, 因此取 $-\log$ 使其值与进化距离更近似于线性关系. 在系统发育树的构建中, 从联配计算距离时要更加小心.

Fitch-Margoliash 算法是一种从距离距阵出发构建进化树的快速聚类算法. 聚类算法将在第7章讨论.

序列-序列的联配由常用的二序列动态规划算法完成. 通过把一条序列和一个已存在的序列群中的每条序列依次做二序列联配, 此方法将该序列加入这个序列群. 分数最高的二序列联配会决定该序列将如何联配到这个序列群. 而对序列群间的联配, 两序列群间的所有序列对都会进行二序列联配; 得分最高的二序列联配将决定这两个序列群间的联配. 于是, 计分系统本质上就是带仿射仿空位罚分的标准 PAM 分数. 联配完成之后, 空位被不确定字符 `X` 取代. Feng & Doolittle 将此规则命名为"一旦有空位, 始终是空位" (once a gap , always a gap). 此规则允许将二序列联配用于指导序列与序列群的联配或序列群间的联配; 否则, 任一给定的二序列联配都不一定同序列群中的已有联配一致. 因为 `X` 与任何残基(包括空位) 联配都时不用付出罚分, 所以此规则会产生我们想要的副作用: 在随后二序列联配的相同列中鼓励出现空位. 在基于 profile 的渐进联配算法中不需要将空位符重写为 `X` (见下文).

Profile 联配

Feng & Doolittle 方法的一个缺陷是, 所有的联配都由二序列联配决定. 一旦构建了已联配的序列群, 那么使用从该序列群之多序列联配中获得的位置特异信息, 以便将一条新序列联配到该序列群的做法就是有益的. 我们应该考虑序列位于每个位点处的保守程度, 而且高度保守位点处的失配也应该比易变位点处的失配获得更多罚分. 在成簇联配中经常出现空位的位置, 空位罚分应该减少, 而没有空位的位

[1] 原文为 diagnoal matrix (对角矩阵), 可能有误. ——译注

置罚分要增加. 以上主张也促成了序列 profile 在数据库搜索中应用的发展. 这也使 profile 在渐进多序列联配算法中的应用变得合理.

许多渐进联配方法使用序列到 profile 的二序列联配方法 [Thompson, Higgins & Gibson 1994a; Gribskov, McLachlan & Eisenberg 1987], 或将 profile 到 profile 的二序列联配作为子程序在进程中多次调用(见 Gotoh [1993]). Profile-序列或 profile-profile 联配中使用的计分函数没有统一的精确定义. 已联配残基的分数通常以配对和(sum-of-pairs) 的形式出现, 但是不同方法对空位的处理却大不相同.

就像先前所讨论的那样, 对线性空位计分来说 profile 联配是简单的, 因为通过设置 $s(-,a) = s(a,-) = -g$和$s(-,-) = 0$, 我们可以将空位分数包含到 SP 分数 (6.5) 中. 假设我们有二个多序列联配(或 "profile"), 其中一个包含序列 1 到 n, 另一个则包含序列 $n+1$ 到 N. 对这两个 profile 做联配就意味向整条列中插入空位, 于是其中一个 profile 的联配将保持不变. 此时全局联配分数 (6.5) 即为

$$\begin{aligned}\sum_i S(m_i) &= \sum_i \sum_{k<l\le N} s(m_i^k, m_i^l) \\ &= \sum_i \sum_{k<l\le n} s(m_i^k, m_i^l) + \sum_i \sum_{n<k<l\le N} s(m_i^k, m_i^l) + \sum_i \sum_{k\le n, n<l\le N} s(m_i^k, m_i^l).\end{aligned}$$

在上式中, 我们所做的就是将总和分成三部分, 其中两部分仅考虑 profile, 而另一部分则包含所有的交叉项. 前两项不受全局联配影响, 因为向 profile 中添加只含空位字符的列不改变其分数 ($s(-,-) = 0$). 因此仅优化带交叉项的最后一个和就可以得到两个 profile 间的**最优联配**. 我们可以用同标准二序列联配完全一样的方式完成该步骤, 其中列与列间的分数由相加配对分数得到. 很明显, profile 也可以只由一条序列构成, 此时这就对应于将一条序列联配到一个 profile.

CLUSTALW

被人们广泛应用的 CLUSTALW 程序 [Thompson, Higgins & Gibson 1994a] 实现了基于 profile 的渐进多序列联配算法, 它继承了一个更早期的流行程序 CLUSTALV [Higgins, Bleasby & Fuchs 1992]. 除开仔细调整了对 profile 联配方法的使用外, CLUSTALW 与 Feng-Doolittle 算法的工作方式几乎相同. 从总体上说, CLUSTALW 算法如下:

算法: CLUSTALW 渐进联配

(i) 通过二序列动态规划联配方法构建所有 $N(N-1)/2$ 对序列的距离矩阵, 然后使用 Kimura [1983] 的模型将相似性分值近似地变换为进化距离.

(ii) 用 Saitou & Nei [1987] 的邻接聚类 (neighbor-joining clustering) 算法构建一棵指导树.

(iii) 使用序列-序列、序列-profile 以及 profile-profile 联配, 在节点处以相似性从高到低的次序渐进地进行联配. ◁

在联配构建和计分阶段方面, CLUSTALW 显然是很特别的. 除了常见的 profile 构建与联配方法以外, CLUSTALW 使用如下一系列额外的启发式规则提高了精确度:

- 为序列加权以补偿大子家族中的有偏代表 (biased representation). 从根本上讲, CLUSTALW 的 profile 计分函数是配对和. 与 Carrillo-Lipman 算法一样, 为序列加权对补偿配对和的缺陷是重要的.
- 根据联配的期望相似性挑选为联配计分的替换矩阵; 关联相近的序列使用"硬"矩阵(例如 BLOSUM80) 进行联配, 而关联远的序列用"软" 矩阵(例如 BLOSUN50) 进行联配.
- 用一个修正因子乘上位置特异的空位开端 profile 罚分, 该因子是该位置处观测残基的函数. 我们可以从大量结构联配中的观测空位频率出发获得这些罚分. 一般来说, 疏水 (hydrophobic) 残基(更可能埋在蛋白内部) 比亲水

(hydrophilic) 残基或柔性 (flexible) 残基(更可能从蛋白表面得到) 有更高的空位罚分.

- 当一个位置跨越连续五个或更多的亲水残基时, 空位开端罚分也会减小.
- 当联配的某列中没有空位而它附近存在空位时, 空位开端和空位延伸的罚分均会增加. 这条规则试图强制所有空位都在联配中相同的位置出现.
- 在渐进联配阶段, 如果某个联配的分数较低, 那么指导树可能会马上进行调整以推迟低分联配, 直到后面的渐进联配过程中积累更多的 profile 信息再开始.

从概率论建模角度看, 学习这种精雕细琢的启发式方法是很有趣的. 将启发式方法融入更正式的概率论模型也许是一种好办法, 因为它为全概率论模型赋予了优化大自由参数集的能力.

迭代求精方法

渐进联配算法存在的一个缺陷是, 其子联配是"冻结" (frozen) 的. 即当更多数据到来时, 我们不能改变一组已联配序列间的现存联配. 而迭代求精方法试图避开这个问题[Barton & Sternberg 1987; Berger & Munson 1991; Gotoh 1993].

在迭代求精方法中, 我们可以使用如上所述的方法先生成一个初始联配; 然后取出一条 (或一组) 序列与剩下的已联配序列之profile 做重联配. 如果我们正在优化一个有意义的分值, 那么这样做要么增加了总分数, 要么会导致分数都相同. 选择另一条序列并对对其进行重联配, 依此类推直到联配结果不再改变. 假设所有序列都被尝试过并且存在最高分, 那么因为序列空间有限, 所以该过程确保收敛到分值的局部最大值.

Barton & Sternberg [1987]的算法向我们展示了如何将上面提到的某些方法整合到一起. 它的具体步骤如下:

算法: Barton & Sternberg 多序列联配

(i) 找到成对相似性最高的两条序列, 并用标准二序列动态规划联配算法对它们进行联配.

(ii) 找到一条与最初两条序列联配之 profile 最相似的序列, 通过 profile-序列联配将其与最初两条序列进行联配. 重复此过程直到所有序列都被包含进多序列联配中.

(iii) 移出序列 x^1, 并用 profile-序列联配方法将其与其他已联配序列$x^2, \ldots, x^N$的 profile 进行重联配. 对$x^2, \ldots, x^N$ 重复此步骤.

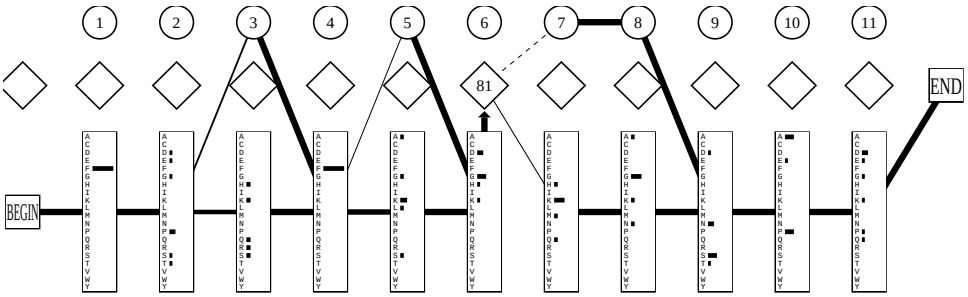
(iv) 重复上述重联配步骤若干次, 或直到联配分数收敛. ◁

Profile 联配和迭代求精方法的思想与多序列联配问题之概率论隐马模型方法的形式很相似. 下面我们再回到 HMM 方法.

6.5 由 profile HMM 训练的多序列联配

在第5章, 我们展示了序列 profile 可以改写为概率论形式的 profile HMM . 因此, 我们就可以简单地使用 profile HMM 替代渐进联配或迭代联配方法中的标准 profile. HMM 的形式体系具有几个优势. 特别地, 从本质上讲特定的 SP 计分方案可以用更明确的 profile HMM 假设所替代, 即假设所有序列均由单一"根" 概率分布独立产生.

我们也可以使用第3章的 Baum-Welch 期望最大化算法, 从初始未联配序列中训练得到 profile HMM. 从 HMM 文献中涌现出的这类方法事实上是人们最早应用的基于 HMM 的多序列联配方法. 如果将训练好的模型用于每条独立序列 Viterbi 联配的最后一步, 那么除了模型外训练过程还将产生一个多序列联配 [Krogh *et al.* 1994].




```
FPHF-DLS-----HGSAQ
FESFGDLSTPDAVMGNPK
FDRFKHLKTEAEMKASED
FTQFAG-KDLESIKGTAP
FPKFKGLTTADQLKKSAD
FS-FLK-GTSEVPQNNPE
FG-FSG----AS--DPG
```

图 6.4: 从联配 (下图) 估计出模型 (上图). 联配中阴影区的残基被视为插入. 对绘制该模型的描述见图5.4.

带已知 profile HMM 的多序列联配

在解决从初始未联配训练序列中同时估计模型和多序列联配的问题之前, 我们先考虑从已知模型中得到多序列联配这个更简单的问题. 这个问题在序列分析中经常出现, 例如当面对一个多序列联配以及家族中序列的小代表集之模型时, 我们想用该模型联配家族中的其他大量序列.

我们已经知道如何将序列和 profile HMM 进行联配: 用 Viterbi 算法找到贯穿模型的最有可能路径. 构造多序列联配时, 我们只需要为每条独立序列计算 Viterbi 联配. 联配到相同的 profile HMM 匹配状态的残基被联配到同一列上. 这便暗示了 profile HMM 多序列联配与传统多序列联配间的一个重要不同, 这一点会在下面的例子中变得更加清晰.

图6.4显示了一个小 profile HMM 以及导出它的多序列联配. 在这个例子中, 阴影部分的残基被人为地定义成插入状态, 其他十列对应十个 profile HMM 的匹配状态. 我们将相同的七条序列与这个模型进行重联配, 其最优 Viterbi 路径显示在图6.5中. 从这些路径得到的多序列联配显示在图6.6中的左边, 其中小写字母残基表示插入状态而大写字母残基表示匹配状态.

我们观测到的重要事实是, 原始联配 (图6.4) 和新得到的联配 (图6.6左图) 是一样的. Profile HMM 并不试图联配表示插入状态的小写字母残基. 我们可以任意选择如何在联配中放置插入残基; 某些实现 profile HMM 的方法只是简单地将插入区域进行左对齐, 就像图6.6一样. 插入状态残基经常代表序列中的非典型 (atypical)、不保守 (unconserved) 和无意义的可联配 (not meaningfully alignable) 部分. 就像前面所讨论的一样, 这是从生物学意义上讲对多序列联配的实际看法. 例如我们期望同源蛋白质结构的环在结构上是有所差异的并且不可联配. 与之相反, 很多多序列联配算法对整条序列进行联配, 完全忽略序列联配的意义.

图6.6的右图展现了一条联配到相同模型的新序列. 因为在指定为插入状态6的阴影区域中, 这条序列相比其他七条序列含有更多的插入残基, 所以必须调整原来七条序列的联配以便为这两个新的残基分配空间. 在算法实现时, 我们通常关注所有的 Viterbi 路径并在构建多序列联配前为每个插入状态寻找最大插入残基数, 这样我们就事先知道需要多少空间以容纳这些插入.

概览: 从未联配序列出发进行 profile HMM 训练

现在我们考察更难的问题, 即从初始未联配序列中估计模型和多序列联配. 方

Position	1	2	3	4	5	6	insert	7	8	9	10	11
	F	P	H	F	–	D	LS	H	G	S	A	Q
	F	E	S	F	G	D	LSTPDAV	M	G	N	P	K
	F	D	R	F	K	H	LKTEAEM	K	A	S	E	D
	F	T	Q	F	A	G	KDLESI	K	G	T	A	P
	F	P	K	F	K	G	LTTADQL	K	K	S	A	D
	F	S	–	F	L	K	GTSEVP	Q	N	N	P	E
	F	G	–	F	S	G	AS	–	–	D	P	G

图 6.5: 七条序列的贯穿模型的最有可能路径. 如果路径在模型的位置 i 处经过一个匹配状态, 那么相应的残基就被放置于标记为 i 的列. 如果经过删除状态, 则在表格中放置一个"–" 号, 而如果是在位置6经过插入状态, 那么相应的残基就被放置在标有"插入" (insert) 的列.

```
FPHF-Dls.....HGSAQ
FESFGDlstpdavMGNPK
FDRFKHlkteaemKASED
FTQFAGkdlesi.KGTAP
FPKFKGlttadqlKKSAD
FS-FLKgtsevp.QNNPE
FG-FSGas.....--DPG
```

```
FS-FLKngvdptaai--NPK
FPHF-Dls.......HGSAQ
FESFGDlstpdav..MGNPK
FDRFKHlkteaem..KASED
FTQFAGkdlesi...KGTAP
FPKFKGlttadql..KKSAD
FS-FLKgtsevp...QNNPE
FG-FSGas.......--DPG
```

图 6.6: 七条序列的联配, 用小写字母表示插入. 点号是空位填充字符, 以便对齐联配的列. 右图: 加入一条新序列后的联配. 新加入的序列在最上面, 因为这条序列有更多的插入, 所以需要增加更多的点号以填充空位.

法总结如下:

算法: 用 profile HMM 做多序列联配

(i) 起始: 选择 profile HMM 模型的长度并初始化参数.

(ii) 训练: 用 Baum-Welch 算法(第 p. 44页) 或 Viterbi 训练(第 p. 45页) 估计模型. 通常有必要使用启发式方法以避免局部最优值 (见下文).

(iii) 多序列联配: 用 Viterbi 算法(第 p. 39页) 将所有序列联配到最终模型上并用前一节介绍的方法构建多序列联配. ◁

我们现在考虑初始化和训练的细节.

初始模型

profile HMM 是包含三种状态 (匹配、删除和插入) 的重复线性结构. 为 Baum-Welch 估计法选择初始体系结构时, 我们唯一要做的决定是模型的长度 M. 这里 M 是 profile HMM 中的**匹配**状态数而不是总状态数, 在第5章中 profile HMM 的总状态数应该是 $3M + 3$. 一个常用的规则是设 M 为训练序列的平均长度 (或根据先验知识来设定).

因为 Baum-Welch 估计能找出局部最优而非全局最优, 所以要慎重选择初始模型. 我们应该鼓励模型使用"敏感的" (sensible) 转移; 例如, 转移到匹配状态的概率应该大于其他转移概率. 同时我们想要从多个点开始 Baum-Welch 方法, 以便看它是不是收敛于近似相同的最优值, 因此我们希望初始模型的参数选择具有一定的随机性 (randomness).

一种合理的方法是从模型参数的 Dirichlet 先验中对模型的初始参数进行抽样 (第11章). 或者我们可以使用来自先验的频率初始化该模型, 用该模型生成少量随机

序列, 然后用它们的计数作为"数据" 来估计出一个初始模型. 如果更进一步, 我们可以通过模型构建, 从对部分或全部序列之多序列联配的现有猜测中估计初始模型.

Baum-Welch 期望最大化

基本的参数估计可直接应用第3章的 Baum-Welch 算法完成. 下面以第5章中的符号做参考给出了算法的公式.

算法: Profile HMM 的前向算法

起始:

$$f_{\mathrm{M}_0}(0) = 1.$$

递归:

$$\begin{aligned}
f_{\mathrm{M}_k}(i) =& e_{\mathrm{M}_k}(i)\left[f_{\mathrm{M}_{k-1}}(i-1)a_{\mathrm{M}_{k-1}\mathrm{M}_k} + f_{\mathrm{I}_{k-1}}(i-1)a_{\mathrm{I}_{k-1}\mathrm{M}_k}\right.\\
&\left. + f_{\mathrm{D}_{k-1}}(i-1)a_{\mathrm{D}_{k-1}\mathrm{M}_k}\right];\\
f_{\mathrm{I}_k}(i) =& e_{\mathrm{I}_k}(i)\left[f_{\mathrm{M}_k}(i-1)a_{\mathrm{M}_k\mathrm{I}_k} + f_{\mathrm{I}_k}(i-1)a_{\mathrm{I}_k\mathrm{I}_k}\right.\\
&\left. + f_{\mathrm{D}_k}(i-1)a_{\mathrm{D}_k\mathrm{I}_k}\right];\\
f_{\mathrm{D}_k}(i) =& f_{\mathrm{M}_{k-1}}(i)a_{\mathrm{M}_{k-1}\mathrm{D}_k} + f_{\mathrm{I}_{k-1}}(i)a_{\mathrm{I}_{k-1}\mathrm{D}_k} + f_{\mathrm{D}_{k-1}}(i)a_{\mathrm{D}_{k-1}\mathrm{D}_k}.
\end{aligned}$$

终止:

$$\begin{aligned}
f_{\mathrm{M}_{M+1}}(L+1) =& f_{\mathrm{M}_M}(L)a_{\mathrm{M}_M\mathrm{M}_{M+1}} + f_{\mathrm{I}_M}(L)a_{\mathrm{I}_M\mathrm{M}_{M+1}}\\
&+ f_{\mathrm{D}_M}(L)a_{\mathrm{D}_M\mathrm{M}_{M+1}}.
\end{aligned}$$

◁

算法: Profile HMM 的后向算法

起始:

$$\begin{aligned}
b_{\mathrm{M}_{M+1}}(L+1) &= 1;\\
b_{\mathrm{M}_M}(L) &= a_{\mathrm{M}_M\mathrm{M}_{M+1}};\\
b_{\mathrm{I}_M}(L) &= a_{\mathrm{I}_M\mathrm{M}_{M+1}};\\
b_{\mathrm{D}_{M+1}}(L) &= a_{\mathrm{D}_M\mathrm{M}_{M+1}}.
\end{aligned}$$

递归:

$$\begin{aligned}
b_{\mathrm{M}_k}(i) =& b_{\mathrm{M}_{k+1}}(i+1)a_{\mathrm{M}_k\mathrm{M}_{k+1}}e_{\mathrm{M}_{k+1}}(x_{i+1})\\
&+ b_{\mathrm{I}_k}(i+1)a_{\mathrm{M}_k\mathrm{I}_k}e_{\mathrm{I}_k}(x_{i+1}) + b_{\mathrm{D}_{k+1}}(i)a_{\mathrm{M}_k\mathrm{D}_{k+1}};\\
b_{\mathrm{I}_k}(i) =& b_{\mathrm{M}_{k+1}}(i+1)a_{\mathrm{I}_k\mathrm{M}_{k+1}}e_{\mathrm{M}_{k+1}}(x_{i+1})\\
&+ b_{\mathrm{I}_k}(i+1)a_{\mathrm{I}_k\mathrm{I}_k}e_{\mathrm{I}_k}(x_{i+1}) + b_{\mathrm{D}_{k+1}}(i)a_{\mathrm{I}_k\mathrm{D}_{k+1}};\\
b_{\mathrm{D}_k}(i) =& b_{\mathrm{M}_{k+1}}(i+1)a_{\mathrm{D}_k\mathrm{M}_{k+1}}e_{\mathrm{M}_{k+1}}(x_{i+1})\\
&+ b_{\mathrm{I}_k}(i+1)a_{\mathrm{D}_k\mathrm{I}_k}e_{\mathrm{I}_k}(x_{i+1}) + b_{\mathrm{D}_{k+1}}(i)a_{\mathrm{D}_k\mathrm{D}_{k+1}}.
\end{aligned}$$

◁

随后, 前向和后向变量可以结合起来用于重新估计发射和转移概率参数, 算法如下:

算法: Profile HMM 的 Baum-Welch 重估计方程

来自序列 x 的期望发射计数为:

$$E_{\mathrm{M}_k}(a) = \frac{1}{P(x)}\sum_{i|x_i=a} f_{\mathrm{M}_k}(i)b_{\mathrm{M}_k}(i);$$

$$E_{\mathrm{I}_k}(a) = \frac{1}{P(x)}\sum_{i|x_i=a} f_{\mathrm{I}_k}(i)b_{\mathrm{I}_k}(i).$$

来自序列 x 的期望转移计数为:

$$A_{\mathrm{X}_k\mathrm{M}_{k+1}} = \frac{1}{P(x)} \sum_i f_{\mathrm{X}_k}(i) a_{\mathrm{X}_k\mathrm{M}_{k+1}} e_{\mathrm{M}_{k+1}}(x_{i+1}) b_{\mathrm{M}_{k+1}}(i+1);$$

$$A_{\mathrm{X}_k\mathrm{I}_k} = \frac{1}{P(x)} \sum_i f_{\mathrm{X}_k}(i) a_{\mathrm{X}_k\mathrm{I}_k} e_{\mathrm{I}_k}(x_{i+1}) b_{\mathrm{I}_k}(i+1);$$

$$A_{\mathrm{X}_k\mathrm{D}_{k+1}} = \frac{1}{P(x)} \sum_i f_{\mathrm{X}_k}(i) a_{\mathrm{X}_k\mathrm{D}_{k+1}} b_{\mathrm{D}_{k+1}}(i). \qquad \triangleleft$$

和往常一样, Baum-Welch 重估计过程可以被第 p. 45 页介绍的 Viterbi 算法所替换 (也可以看下文). 其他类型的估计方法也已经用于 profile HMM 的估计, 例如梯度下降法 [Baldi *et al.* 1994].

避免局部最大

Baum-Welch 算法保证在概率"表面" (surface) 上找到一个局部最大值, 但是却并不能保证这个局部最优就在全局最优的附近, 或者它就是一个生物学上合理的解. 任意实用的分数优化 (score optimising) 多序列联配方法都存在几乎相同的问题(多维动态规划能找到全局最优解, 但却不够实用). 其中的一部分原因是, 通常这些模型都相当长, 所以有很多机会使得算法陷入错误的解. 例如, 同一个保守模体 (motif) 的两个变种最后有可能被建模为两个不同的模体, 或者, 被两个其他区域夹在中间的一个保守区域最后可能被建模为一个插入. 搜索参数空间的一种简单方法是重复地从不同 (随机) 初始模型开始很多次, 并保留最后的一个最高分值.

一种更加相关的方法是采用某种随机搜索算法把 Baum-Welch 从局部最大值里"撞" (bump) 出来. (这两种算法可以结合起来, 人们通常也是这样做的). 最常用的随机算法是**模拟退火** (simulated annealing) 算法 [Kirkpatrick, Gelatt & Vecchi 1983]. 我们将描述如何进行模拟退火, 然后讨论由模拟退火所引出的一种 profile HMM 训练算法.

模拟退火的理论基础

某些化合物只有在从高温到低温缓慢地退火时才会结晶 (crystallise). 如果温度下降过快, 其结构最终就会停留于自由能的局部最小值并产生紊乱. 在最优化问题中, 我们有某个需要最小化的函数并称其为"能量" $E(x)$, 其中 x 表示对其最小化所需的所有变量. (最大化一个函数就是最小化这个函数的相反数). 受到物理例子的启发, 我们可以引进人工"温度" T, 并且按照统计物理学定律, 设置 (configuration) (或"状态") x 的概率应该由 Gibbs 分布给出: [2]

$$P(x) = \frac{1}{\mathrm{Z}} \exp\left(-\frac{1}{T} E(x)\right). \tag{6.12}$$

归一化 (normalising) 项 $Z = \int \exp(-\frac{1}{T}E(x))dx$ 在统计物理学中称作**配分函数** (partition function) 因为 x 经常是多维的, 所以这是一个复杂积分而且经常不可能计算出 Z 值.

在 $T \to 0$ 的极限下, 除了具有最低能量外的所有设置之概率均为 0 (系统被"冻结" (frozen)). 在 $T \to \infty$ 的极限下, 所有设置概率均相同(系统被"融化" (molten)). 从结晶过程类推, 通过先在高温时对这个概率分布进行抽样, 然后再逐渐降低温度的方法可以找到一个 (或多个) 最小值. 该方法就称作模拟退火. 在本书没有考虑到的其他应用中, 模拟退火经常由所谓的 Monte Carlo 方法 [Binder & Heerman 1988]完成.

对 HMM 来说, 一个自然的能量函数是负对数似然 $-\log P(\mathrm{data}|\theta)$, 因此概率 (6.12) 就是

$$\frac{\exp\left(-\frac{1}{T}[-\log P(\mathrm{data}|\theta)]\right)}{Z} = \frac{P(\mathrm{data}|\theta)^{1/T}}{Z} = \frac{P(\mathrm{data}|\theta)^{1/T}}{\int P(\mathrm{data}|\theta')^{1/T} d\theta'}. \tag{6.13}$$

[2] 在物理学中, 温度要乘上 Boltzmann 常数, 但这里的温度并不是真实的物理温度, 所以不需要这么做. ——原注

从这个分布中挑选出一个模型看起来显然是非常重要的. 我们下面提到的两种方法都是近似方法.

Baum-Welch HMM 重估计中的噪音注射 (noise injection) 受到模拟退火的启发, Krogh *et al.* [1994]介绍了一种特别的方法. 模拟退火的要点是, 随机选择设置(与总是选取更低能量的算法相对) 使得逃离局部最大值成为可能. 通过向从前向-后向过程中估计出的计数里添加噪音, 我们也能得到类似的效果, 然后逐渐减小噪音, 就像在模拟退火中降低温度一样. 在Krogh *et al.* [1994]中, 噪音由初始模型中的一个随机游走 (random walk) 产生. Hughey & Krogh [1996]给出了关于这种方法有效性的系统研究.

HMM 的模拟退火 Viterbi 估计

第二种方法由Eddy [1995]提出, 使用 Baum-Welch 估计的 Viterbi 近似的模拟退火变体来训练模型. Allison & Wallace [1993] 描述了一个与之相似, 但却基于有限状态自动机而非 HMM 的算法.

请回忆在 Viterbi 估计(第 p. 45 页) 中, 每条序列的最有可能路径均被用于计算估计新模型时所需的计数, 而不是对所有路径求和以便得到这些计数的期望. 如果存在 N 条序列, 那么就有一个从 N 条路径 $\pi^1, \ldots, \pi^N$ 到模型参数的准确对应. 因此, 可以将路径看作是在其上进行似然最大化的基础参数, 所以模拟退火就可以在这些(离散的) 变量而不是(连续的) 模型参数 θ 中完成.

Viterbi 估计与模拟退火变体间的关键区别是, Viterbi 为每条序列 x 选择概率最高的路径 π, 而模拟退火则根据给定当前模型下由温度 T 修正的路径似然对每条路径 π 进行抽样:

$$\mathrm{Prob}(\pi) = \frac{P(\pi, x|\theta)^{1/T}}{\sum_{\pi'} P(\pi', x|\theta)^{1/T}}.$$

上式中的分母就是配分函数 Z. 但它只对所有路径求和, 因此可以使用包含**取幂**的转移和发射参数的修正前向算法计算得到. 为提高计算效率, 可以提前算好取幂的参数: $\hat{a}_{ij} = a_{ij}^{1/T}$ 和 $\hat{e}_j = e_j(x)^{1/T}$ 用来取代第 p. 41页中描述的前向计算的未修正概率参数. 随后前向算法的结果就是配分函数 Z, 而当参数没有取幂时它就是 $P(x)$.

然后, 我们从前向动态规划网格 (lattice) 中通过**随机** (stochastic) 回溯方法选出一条次优路径 π. 这个联配包含一系列状态 π_i, 它们依照由前向变量确定的概率递归地选出. 因为这个算法适用于任何 HMM, 所以我们使用与第3章中前向算法相同的通用符号.

算法: HMM 的随机抽样回溯算法

起始: $\pi_{L+1} = \mathrm{End}$.

递归: 对于$L + 1 \geq i \geq 1$,

$$\mathrm{Prob}(\pi_{i-1}|\pi_i) = f_{i-1,\pi_{i-1}} \hat{a}_{\pi_{i-1},\pi_i} / \left(\sum_k f_{i-1,k} \hat{a}_{k,\pi_i}\right). \qquad \lhd$$

换句话说, 对回溯中达到的每个状态而言, 根据达到当前状态的路径(取幂) 概率和之份额选出前一个状态. [3]

随后该次优联配算法用于实现 Viterbi 训练的一种模拟退火变体. 与在每次迭代的每一步中都确定关于当前模型的一个最优多序列联配不同, 次优多序列联配由抽样得到. 次优的程度由温度因子 T 控制, 开始时它很高(给出十分随机化的联配), 然后缓慢下降. 因为新联配是从给定**前一个**模型的联配概率分布中选出(就像期望最大化方法中求期望的步骤一样), 而不是按照其最优模型通过联配的概率选出, 所以按照模拟退火方法的正式统计力学基础, 此过程并非完全正确 [Kirkpatrick, Gelatt & Vecchi 1983].

找到以何种速度降低温度的最好"方案" 本身就是一门科学(或者说是艺术). 有一种模拟退火理论的结果表明, 如果温度降低地足够慢, 那么它就保证能找到最优值, 但这样所用的时间却长得惊人. 实践中经常使用一种简单的按指数或线性递减的温度方案, 这就相当于每一步要么将 T 乘以某个小于1的数要么用 T 减去某个小常量.

与Gibbs 抽样的比较

[3] 直接遵循取幂运算对乘法满足分配律, 便可以从代数上证明此算法能够正确地计算配分函数, 而且能够从 Boltzmann / Gibbs 分布下的所有可能路径(已知当前模型 θ) 中正确地抽样出一条路径, 例如 $(a_{1,2}a_{2,3}a_{3,4})^{1/T} = (a_{1,2})^{1/T}(a_{2,3})^{1/T}(a_{3,4})^{1/T} = \hat{a}_{1,2}\hat{a}_{2,3}\hat{a}_{3,4}$. ——原注

由 Lawrence *et al.* [1993]描述的" Gibbs 抽样器" (Gibbs sampler) 算法具有很大的相似性. Lawrence *et al* .使用的统计模型是一种很短的无空位模体模型, 它本质上是不含插入及删除状态的profile HMM (虽然他们并没有称它为HMM). 训练数据由一组序列组成, 这些序列(在最简单的情况) 只包含某种模体的一个实例, 例如 DNA 上的一个特异蛋白结合位点, 而最初这些模体的位置是未知的. 现在的问题就是要在找到这些模体所在位置的同时为它们的一致统计模型估计参数(通过知道一个, 我们可以找到另一个, 这就如同一个联配意味着一个 HMM 并且反之亦然). 对于期望最大化 (EM; 第11章) 而言这是一个很自然的问题, 其中缺失数据是模体的位置, 我们可以简单地将其指定为它们在序列中的起始点; 它们对应于联配, 即我们在 HMM 训练中试图推断的缺失数据. 事实上, 早期的算法 [Lawrence & Reilly 1990]将期望最大化应用于此问题, 但是这些方法却被证明倾向于达到较差的局部最优值.

在 HMM 框架下, 模拟退火和 Gibbs 抽样器都是 EM 的 Viterbi 近似的随机抽样变体. 在 Gibbs 抽样的每步迭代中, 我们会从联配中移除一条序列; 由剩下的已联配序列建立一个 HMM; 然后使用参数 $T = 1$ 时的随机抽样算法对这条序列到其他序列的新联配进行概率抽样. 不断地重复这个迭代过程直到模型到达某个高概率区域. 于是 Gibbs 抽样器类似于在常数 $T = 1$ 时运行上面的模拟退火 Viterbi 算法, 它从不用任何温度因子效应修改的概率分布中抽样得到联配. 第11章对 Gibbs 抽样做出了一般性的描述.

自适应地修正模型的体系结构; 模型外科手术

在训练一个模型后 (或其过程中), 我们可以查看它产生的联配并做出决定: (a) 某些匹配状态是冗余的, 且应归为一个插入状态; 或 (b) 看上去一个或多个插入状态包含了太多的序列, 此时它们应该展开(例如在插入状态的前后可以插入更多的匹配模块). 发生这种情况既是因为模型长度的初始选择不够好, 同时也是因为训练时遇到了局部最优. 设计出能够在训练过程中以及训练刚好完成时自适应地修正模型体系结构的过程对我们是有利的.

Krogh *et al.* [1994] 提出了一种称为**模型外科手术** (model surgery) 的方法. 通过由前向-后向算法(或相似的 Viterbi 算法) 估计出的"计数" , 我们可以看到某个转移被训练序列使用的程度. 匹配状态的用处是它可以对该状态中所有字符的计数求和. 如果某个匹配状态被少于一半(或另外某个预定义的比例) 的序列所使用, 那么与之对应的模块将被删除. 类似地, 如果超过一半 (或另外某个预定义比例) 的序列使用到某个插入状态的转移, 那么它将被展开为几个新的模块. 新模块的数量取决于插入的平均长度. 虽然这种方法是特设 (*ad hoc*) 的, 但效果却很好.

另外一种方法是用第5章给出的最大后验 (maximum a posteriori, MAP) 模型构建 算法, 重估计模型结构和模型参数. 因为这个过程需要联配而不需要期望的"计数" , 所以它不适用于通常的 Baum-Welch 期望最大化过程. 通过对 Baum-Welch 的 Viterbi 近似, 它可以正确地应用于训练过程, 并且事实上它可以完全取代常用的参数重估计过程, 产生一个 (局部) 收敛的最优化算法——这种算法能够同时优化 HMM 的体系结构和参数. 通过插入一个包含 Viterbi 联配和 MAP 模型构造过程的迭代, 我们也可以周期性地 (就像模型外科手术一样) 将其应用于完全的 Baum-Welch 估计. 这种用法并不一定保证改进数据的总似然, 但是同模型外科手术一样, 它是一种相当好的启发式方法.

6.6 补充读物

长篇多序列联配算法的综览文献包括 Carrillo & Lipman [1988], Chan, Wong & Chiu [1992]以及 Gotoh [1996].

我们尚未讨论过的一类多序列联配算法也属于模拟退火 算法, 它们定义"移动" (move, 候选联配中的小改动) 和用于确定某个被提议的 (proposed) 移动是否应该被接受之概率的目标函数. 这些抽样算法是 Monte Carlo 型的模拟退火算法, 与我们讨论过的 Viterbi 型 HMM 估计的模拟退火变体十分不同[Lukashin, Engelbrecht & Brunak 1992; Hirosawa *et al.* 1993; Kim & Pramanik 1994; Kim, Pramanik & Chung 1994].

类似于 Gibbs 抽样器的一致模体查找算法与多序列联配算法十分相似, 上文已经简略讨论过. 除 Gibbs 抽样器外, 其他模体查找算法的例子还包括 Stormo &

Hartzell [1989], Hertz, Hartzell & Stormo [1990], Bailey & Elkan [1994]和Bailey & Elkan [1995]. 三维结构 的多序列联配问题也与多序列联配问题相关 (但是更难) [Russell & Barton 1992; Holm & Sander 1993; Gerstein & Levitt 1996].

有几篇文献系统地测试了各种多序列联配算法与通过结构或手工产生之联配相比的准确度, 其中包括McClure, Vasi & Fitch [1994]和Gotoh [1996].

第七章

构造系统发育树

上一章我们考虑了一组序列的多序列联配问题. 有人认为序列联配应考虑其进化关系 [Sankoff, Morel & Cedergren 1973]. 例如, 与在近缘序列之间含有很多替换的联配相比, 其大多数改变只出现在大进化距离上的联配就比较可信.

一些多序列联配算法用到树, 例如我们已经知道有几种渐进联配算法使用"引导树" . 从其名称就可判断, "引导树" 用于引导聚类过程, 而不是满足某个分类学家的需要. 本章我们将转移重点, 开始关注如何构造树. 但本章不会忽略联配: 最后一节将描述同步实现联配和构树的方法.

本章着重介绍两种一般性的构树方法: 距离法与简约法; 下一章将从概率论的角度阐述系统发育.

7.1 生命之树

对有机体分子机制相似性的研究充分说明: 地球上的所有有机体均来自同一祖先. 因此, 任意一组物种都相关, 这种关系称为**系统发育** (phylogeny). 通常这种关系用一棵**系统发育树** (phylogenetic tree) 表示. 系统发育学的任务就是从对现存有机体的观测出发推断出这棵树.

传统上, 人们用形态学特征 (来自活的有机体或化石) 推断系统发育关系. Zuckerkandel & Pauling 的开创性论文 [1962] 指出, 分子序列能够提供许多富含信息的特征. 因此如果有来自不同物种的一组序列, 那么我们就能根据它们推断出所讨论物种间的可能系统发育关系. 以上我们假设这些序列都从一个共同祖先物种的某个共同祖先基因演化而来.

基因复制 (gene duplication) 的普遍发生意味着我们需要仔细检查上述假设. 一组序列的系统发育树未必真实地反映它们所属物种间的系统发育关系, 因为基因复制是与物种分化不同的另一机制, 这两种机制都可使两条序列从共同祖先分化并分离. 由于物种分化而分开的基因称为**直系同源基因** (orthologue), 由基因复制作用分化而来的基因称作**旁系同源基因** (paralogue)如果有兴趣推断带这些基因的物种之系统发育树, 那么我们必须使用直系同源序列. 当然我们可能对复制事件的系统发育关系也感兴趣, 此时就应当建立旁系同源基因的系统发育关系, 即使这些旁系同源基因属于同一物种. 图7.1给出了旁系同源基因和直系同源基因间的区别.

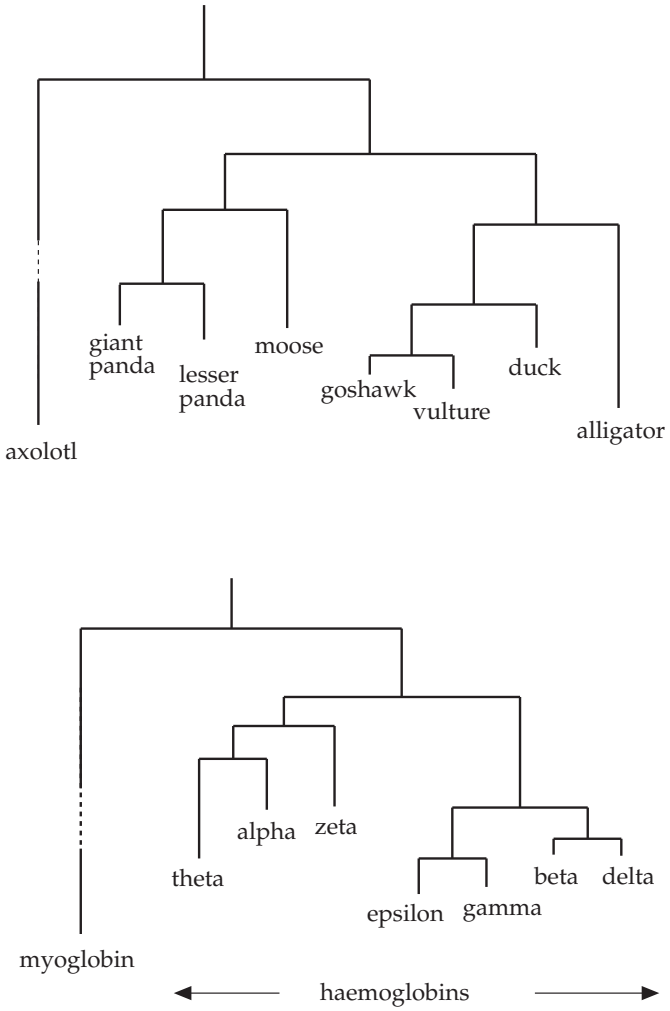


图 7.1: 上图: 基于一组 α-血红蛋白 (haemoglobin) 构建的直系同源基因树. 下图: 旁系同源基因树, 包括人类血红蛋白的 α、β、γ、δ、ε、ζ 和 θ 链, 以及人类肌红蛋白 (myoglobin). α-血红蛋白的直系同源基因在 SWISS-PROT 中的标识符分别为 HBA_ACCGE、HBA_AEGMO、HBA_AILFU、HBA_AILME、HBA_ALCAA、HBA_ALLMI、HBA_AMBME和HBA_ANAPL, 它们是 PFAM 数据库[Sonnhammer, Eddy & Durbin 1997] (http://genome.wustl.edu/Pfam/) 中按字母排序的前八个 α-蛋白质. 球蛋白旁系同源基因在 SWISS-PROT 中的标识符分别为HBAT_HUMAN、HBAZ_HUMAN、HBA_HUMAN、HBB_HUMAN、HBD_HUMAN、HBE_HUMAN、HBG_HUMAN和MYG_HUMAN. 这两棵树均用 J.Felsenstein 的软件包 PHYLIP (http://evolution.genetics.washington.edu/phylip.html) 通过第7.3节介绍的邻接法构造. 邻接法使用基于 PAM 的 ML 距离(见第p. **??**页), 这些距离从 PHYLIP 软件包中的 PROTDIST 程序得到.

7.2 有关树的背景知识

本章假设所有树均为二叉 (binary) 树, 即每个分叉的边都有两个子边(见图7.2). 等价的说, 每个分支节点均有三条边汇合, **节点** (node) 是边的端点. 树的二叉性假设不是严重的限制, 因为其他任意分枝形式都可近似为某些分支很短的二叉树.

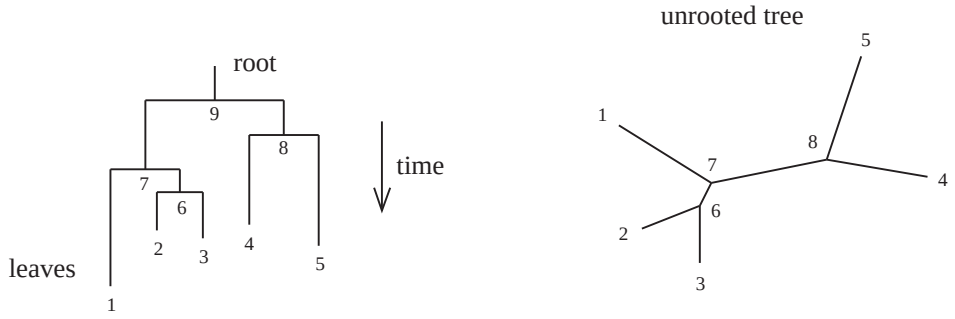


图 7.2: 二叉树的例子(左), 画出了根、叶节点以及进化时间的方向(距离现在最近的时间画在图的底部). 图中也画出了相对应的无根树(右), 而时间的方向却没有确定.

树的每条边都与一定量的进化分歧 (evolutionary divergence) 相关联, 这种分歧由序列间距离的某种度量所定义, 或根据某种进化历程的残基替换模型所定义. 我们在这里采用常用术语"长度" 或"边长" 描述所画图中边的长度. Langley & Fitch [1974]研究了系统发育学定义的长度与古生物学年代间隔之间的关系, 并发现不同蛋白质能以非常不同的速率发生改变, 而且即使相同的序列也可能在某些有机体中更迅速地发生进化. 然而对更大量的蛋白质取平均证实, 系统发育树上的长度与进化时间间隔之间存在广泛的一致性 [Doolittle *et al.* 1996; Wray, Levinto & Shapiro 1996].

一棵真实的生物学系统发育树是有"根" 的, 根就是所有序列的最原始祖先. 有些算法提供了关于根位置的信息, 或至少是一种推测. 其他的算法, 诸如下章讨论的简约法与概率论模型方法, 都完全不提供树根的位置信息, 因此必须用其他标准为树定根. 现在我们考虑如何表示有根树与无根树.

图7.2画出一棵无根树以及它的有根树形式. 请注意图中的有根树: 我们将根画在树的顶部, 同时将末端节点即**叶节点** (leaves) 对应于观测到的序列并画在底部.

树的叶节点有名字或编号. 有时可以交换编号而不改变系统发育关系(例如图7.2中的编号4和5), 但它们往往交换不得(例如交换图7.2中的编号1和2会改变系统发育关系). 加了标记的树称为**有标分支型** (labelled branching pattern). 更粗略地讲, 可称之为树的**拓扑** (topology) [1], 并以符号T 记之. 为了完整地定义系统发育树, 必须指定树的边长[2]; 在 i 的适当编号方案下, 通常以 t_i 记之.

树的计数与标注

可以按以下方法为有根树的节点和边计数. 假设树有 n 个叶节点. 沿树向上移动, 当遇到新节点时边就会合并, 这样边的数目每次就减少1. 因此除 n 个叶节点外, 还应有 $(n-1)$ 个节点, 于是共 $(2n-1)$ 个节点; 而边数还要少一, 即 $(2n-2)$ 条, 因为根节点上面的边不算在内. 我们会用从 1 到 n 的数字标注叶节点, 用从 $n+1$ 到 $2n-1$ 的数字标识分支节点, 并保留表示根节点的 $2n-1$. 边的长度将用其底部的节点数字作下标, 因此 d_1 表示节点 1 之上直接相联的边长度, 依此类推.

一棵有 n 个叶节点的无根树共有 $2n-2$ 个节点和 $2n-3$ 条边. 可以在此无根树的任何边上添加根节点, 因此我们可以由此构造 $(2n-3)$ 棵有根树. 图7.3展示了 $n=3$ 的情形; 根的 3 个可能位置分别对应 3 棵有根树. 因此对已知的叶节点数目 n 来讲, 无根树的数目是其有根树数目的 $(2n-3)$ 倍.

除根节点外, 我们还可以为图7.3中带 3 个叶节点的无根树添加额外的、带不同标记的边或"分支" (即"4"), 于是得到带 4 个叶节点的无根树. 共有 3 棵这种树, 它们都有 $(2n-3)=5$ 条边, 容易看出它们是不同的有标分支型. 此时有 5 种方法再为该树添加一个不同标记的分支(即"5"), 则带 5 个叶节点的无根树总共有 $3\times 5=15$

[1]拓扑学家可能会用这个名词表示无标分支型 (unlabelled branching pattern), 即不能通过在节点处边的置换或边的缩短及延伸来相互重排的树之不同类别. ——原注

[2]"时间" 这一最终感兴趣变量的刻意写照. ——原注

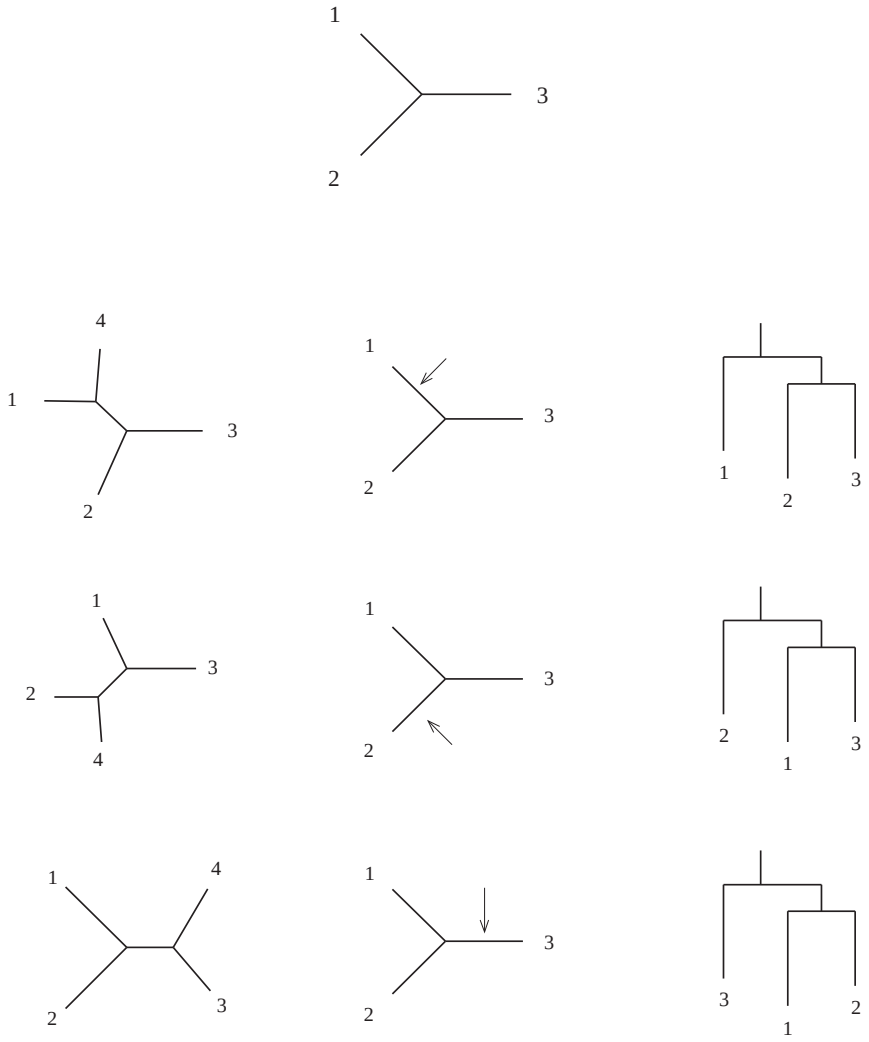


图 7.3: 通过选择不同边作为根节点的位置(见箭头), 从含3条序列的无根树导出有根树(右侧一列).

棵. 以此类推, 可以看出带 n 个叶节点的无根树共有 $(3)\cdot(5)\cdot\ldots\cdot(2n-5)$ 棵; 这个数也可以记为 $(2n-5)!!$. 由上面的讨论可知, 共有 $(2n-3)!!$ 棵相应的有根树. 树的个数随 n 快速增长; 当 $n=10$ 时大约有 200 万棵无根树, 而当 $n=20$ 时, 约有 2.2×10^{20} 棵无根树. 如果想要了解更多有关树的计数问题, 请参考 Felsenstein [1978b].

练习

7.1 向图7.2中无根树的7个可能位置添加根节点, 依此画出有根树.

7.2 图7.3中带三个和四个叶节点的树均有相同的无标分支型. 对有根树和无根树而言, 为确保获得多于 1 个的无标分支型, 分别需要多少个叶节点? 请为有根树写出该数目的递归关系. (提示: 考虑这样的树, 它由两棵树在根节点处合并而得到.)

7.3 迄今为止所有我们考虑的树均是二叉的, 很容易想象在有根形式下三叉(ternary) 树会有三个向下分支来自一个分支节点. 无根三叉树则是在每个分支节点处辐射出四条边. 若无根三叉树有 m 个分支节点, 则它带多少个叶节点, 有多少条边?

7.4 下一步考虑混合情形的无根树, 它有 m 个三叉分支节点和 n 个二叉分支节点. 则它带多少个叶节点, 有多少条边? 令 $N_{m,n}$ 表示该树的不同有标分支型数目, 推广二叉树的计数结论并证明:

$$N_{m,n} = (3m + 2n - 1)N_{m,n-1} + (n + 1)N_{m-1,n+1}.$$

(提示: "=" 后面第一项计算有多少种方法可以将一个新边添加到一个已经存在的边上, 也就是增加一个额外二叉节点; 第二项对应于在二叉节点上添加新边, 形成三叉节点.)

7.5 对于很小的 m 值, 使用上面的递归关系计算 $N_{m,0}$, 即含有 m 个分支节点的不同纯三叉树的个数. (提示: 我们知道 $N_{0,i} = (2i - 1)!!$, 故递归关系可将 $N_{m,0}$ 用 $N_{0,i}$ 表示, 其中 $i \le n$. 程序员们将会很愿意编写一个递归程序实现该过程.) 检查计算结果是否满足 $N_{m,0} = \prod_{i=1}^{m}(1 + 9i(i - 1)/2)$, 你能证明该式吗?

7.3 用成对距离建树

许多直观上更容易被接受的建树方法之起点都是给定数据集中每个序列对 i, j 间距离 d_{ij} 的集合. 有多种定义距离的方法. 例如我们可以把 d_{ij} 定义为残基 x_u^i 与 x_u^j 间不同位点 u 的比例 f (假设已经存在两条序列间的联配). 对小的 f 来讲, 这是一种敏感的定义. 然而对两条无关序列而言, 随机替换将导致 f 接近随机期望下的差异位点比例, 并且我们希望当 f 趋近这个值时两序列间的距离变大. 残基替换的 Markov 模型, 如 DNA 的 Jukes-Cantor 模型(第 p. **??**页) 就可以用来定义具有如此特性的距离. Jukes-Cantor 距离是 $d_{ij} = -\frac{3}{4}\log(1 - 4f/3)$, 当 f 接近其平衡值(有 75% 的残基互不相同) 时该距离趋近于无穷.第8.6节我们将继续讨论距离的定义.

聚类方法: UPGMA

让我们从一种叫作 UPGMA 的聚类 (clustering) 方法 [Sokal & Michener 1958]开始, 它表示"非加权对组算术平均法" (unweighted pair group method using arithmetic averages). 尽管它的缩写名令人望而生畏, 但此方法既简单而且从直观上看又很吸引人. 它对序列进行聚类, 在每步都融合两个类, 同时在树上创建一个新节点. 这棵树可以被想象为自下而上地聚集, 每个节点都被添加到其他节点之上, 边的长度由位于该边顶部与底部的节点高度 (height) 差决定.

首先我们定义两个类 C_i 和 C_j 间的距离 d_{ij} 为对所有来自不同类的两条序列间距离之和求平均:

$$d_{ij} = \frac{1}{|C_i||C_j|}\sum_{p \text{ in } C_i, q \text{ in } C_j} d_{pq}, \tag{7.1}$$

其中 $|C_i|$ 和 $|C_j|$ 分别表示第 i 个和第 j 个类中序列的数目. 注意, 如果 C_k 是两个类 C_i 和 C_j 的并集, 即如果 $C_k = C_i \cup C_j$, 且 C_l 是任意其他类, 则有 (练习7.6):

$$d_{kl} = \frac{d_{il}|C_i| + d_{jl}|C_j|}{|C_i| + |C_j|}. \tag{7.2}$$

聚类过程如下: **算法: UPGMA**

初始:

指定每条序列 i 自身作为一个类 C_i,

为每条序列定义树 T 的一个叶节点, 高度设为 0.

迭代:

找到两个类 C_i, C_j 使 d_{ij} 最小, 如果有几对类距离都最小，则随机选一对.)

定义一个新类 k: $C_k = C_i \cup C_j$, 并按式 (7.2) 对所有 l 定义 d_{kl}.

定义一个带子节点 (daughter node) i 和 j 的节点 k, 并置其高度为 $d_{ij}/2$.

将 k 添加到现有的类集合中, 然后删除 i 和 j.

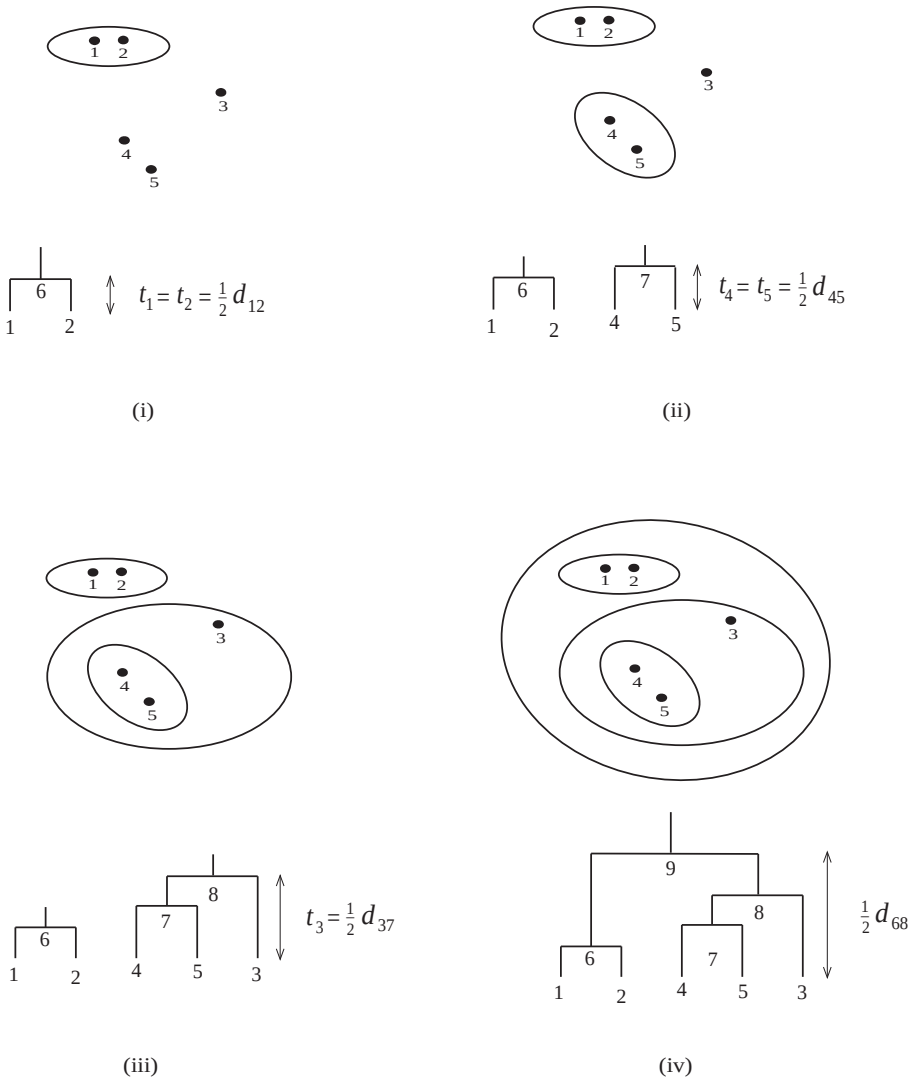


图 7.4: 使用 UPGMA 对序列逐步聚类, 以便构建有根树. 其中5条序列间的距离如平面上的点所示 (对一组距离而言, 通常这种表示方法不一定可行).

终止:

当只剩两个类 i 和 j 时, 置根节点的高度为 $d_{ij}/2$. ◁

为了检查此过程可以产生正确定义的边长, 我们必须证明父节点 (parent node) 总是位于其子节点的上方 (见练习7.7). UPGMA 方法有一些变体, 它们把类间距离定义为其构成序列 (constituent sequences) 距离的最大值或最小值而非算术平均, 但似乎 UPGMA 的性能表现最好.

例子: 对5条序列应用 UPGMA

5条序列间的距离被图示为平面上的距离(见图7.4). UPGMA 运行过程如下: 首先, 找到最近的两条序列, 假设为 x^1与x^2. 它们的父节点标记为 6, 边长 t_1 与 t_2 定义为 $t_1 = t_2 = \frac{1}{2}d_{12}$. 然后我们将序列 x_i 与表示类 $\{x^1, x^2\}$ 的新分支节点 6 间的距离 d_{i6} 定义为平均值 $\frac{1}{2}(d_{1i} + d_{2i})$, 再在剩余序列与节点 6 间寻找最近的类别对. 这一类别对是 $\{x^4, x^5\}$; 按上面的方法构建它们的父节点 7, 并定义边长 t_4 和 t_5 为 $t_4 = t_5 = \frac{1}{2}d_{45}$. [3]重复这个过程. 下一个最近类别对是 x^3 和节

[3]原文为 $d_4 = d_5 = \frac{1}{2}d_{45}$, 可能有误. ——译注

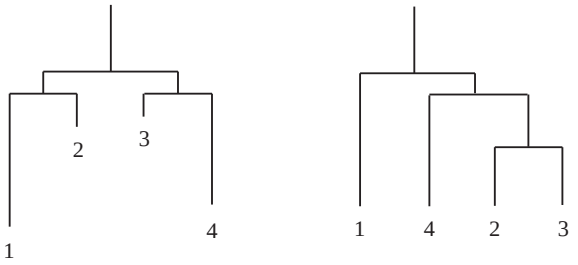


图 7.5: 一棵树(左图), 及其被 UPGMA 错误重建的树(右图).

点 7. 引入节点 8 作为 x^3 和 7 的父节点, 并定义 x^3 以上的边长度为 $t_3 = \frac{1}{2}d_{37}$, 节点 7 以上的边长度 $t_7 = \frac{1}{2}d_{37} - \frac{1}{2}d_{45}$, 这样从每个分支下来的时间总和都相同. 最后一个融合发生在节点6($\{x^1, x^2\}$)与节点8 ($\{x^3, x^4, x^5\}$)之间, 其距离为 $d_{68} = \frac{1}{6}(d_{13} + d_{14} + d_{15} + d_{23} + d_{24} + d_{25})$. □

练习

7.6 证明, 如果类间的距离按(7.1) 式定义, 且 $C_k = C_i \cup C_j$, 则对任意 l, d_{kl} 由(7.2)式给出.

7.7 证明任意节点总是位于其子节点的上面. (提示: 否则, 证明在形成某个子节点时, 我们会错误选择距离最近的类.)

分子钟与距离的超度量属性

用 UPGMA 可以生成一种特别的有根树, 我们能够将这种树的边长看作是用固定速率的**分子钟** (molecular clock)度量的进化时间. 我们假设树上所有点的序列分化按相同的固定速率发生, 即从任意节点沿向下路径到达叶节点的时间之和都一样, 无论选择哪一条路径. 在树 T 中, 如果我们按分子钟通过边长相加得到距离数据, 那么 UPGMA 将正确地重建树 T. 为了明白起见, 想象一条水平线从树 T 的叶节点处升起: 每当这条线越过一个节点, 该节点左支的所有叶节点到该节点右支所有叶节点间的距离都将是当前的最小距离, 因此这个节点被准确地添加到它在原来树的相应位置上.

如果原来的树不具有这种优良性质, 而是到其叶节点有不同的边长, 就像图7.5 (左) 所示, 那么它就有可能被 UPGMA 错误地重建(见图7.5右). 此时错误在于最近的叶节点是不相邻的, 即它们没有一个共同的父节点. 我们可以使用**超度量** (ultrametric)条件检验重建过程是否可能是正确的. 距离 d_{ij} 被称为超度量的, 如果对任意三条序列 x^i, x^j, x^k, 距离 d_{ij}, d_{jk}, d_{ik} 或者都相等, 或者其中两个相等而第三个较小. 从带分子钟的树得到的距离满足这个条件.

练习

7.8 能够证明, 如果距离 d_{ij} 是超度量的而且用 UPGMA 方法从这些距离构建树, 那么令从这棵树上得到的 i 到 j 间路径上那个节点的高度之两倍作为距离时, 该距离与 d_{ij} 相等. 对应用于 5 条序列的 UPGMA 例子, 如果其距离是超度量的, 验证上述结论的正确性. (提示: 证明当两个类 C_k 和 C_l 发生融合的时候, 超度量条件使得 C_k 中任意叶节点与 C_l 中任意叶节点间的距离相等.)

可加性与邻接法

在描述由 UPGMA 产生的树具有分子钟性质时, 我们隐含假设了另一个重要性质: 可加性 (additivity). 给定一棵树, 其边长被称为**可加的** (additive), 如果任意一对叶节点间的距离是连接它们的路径上的边长之和. 这种性质在 UPGMA 树构建时就自动建立. 然而有可能不满足分子钟性质但满足可加性, 在这种情况下有些算法能够用来正确地重建树.

给定一棵具有可加边长 $\{d_\bullet\}$ 的树 T, 我们可以按如下方法从叶节点的成对距离

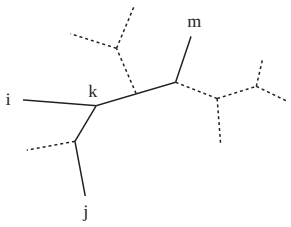


图 7.6: 对任意3个叶节点 i、j 和 m, 都有一个节点即这里的 k, 使得到它们的分支在此相遇. 由可加性可知 $d_{im} = d_{ik} + d_{km}$、$d_{jm} = d_{jk} + d_{km}$、$d_{ij} = d_{ik} + d_{jk}$, 于是$d_{km} = \frac{1}{2}(d_{im} + d_{jm} - d_{ij})$, 此即方程(7.3).

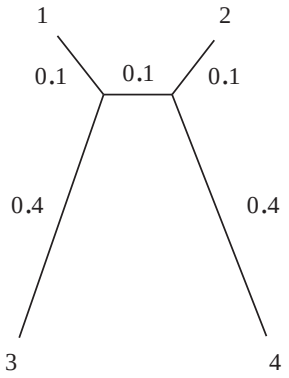


图 7.7: 树中最近的一对叶节点不是邻近叶节点. 边长如图所示. 如果边长具有可加性, 则有 $d_{12} = 0.3$ 和 $d_{13} = 0.5$, 因此相邻节点对1, 3可以比不相邻节点对1, 2分开得更远.

$\{d_{ij}\}$ 出发重建这棵树: 找到一对**相邻叶节点** (neighbouring leaves), 即有相同父节点 k 的叶节点. 设它们的编号为 i, j. 把它们从叶节点的列表中删除, 并将节点 k 添加到当前节点的列表, 定义它到叶节点 m 的距离为

$$d_{km} = \tfrac{1}{2}(d_{im} + d_{jm} - d_{ij}). \tag{7.3}$$

由可加性, 刚定义的距离 d_{km} 恰好等于原树中对应节点间的距离 (见图7.6). 用这种方法可以去掉叶节点, 每次操作使节点数减1, 直到仅剩余一对叶节点.

如果可以只从距离确定一对邻近叶节点, 那么我们就能精确地重建具有可加边长的树. 值得注意的是, 我们可以使用一种方法挑选邻近叶节点, 该方法先由 Saitou & Nei [1987]提出, 后被 Studier & Keppler [1988]改进.

首先, 请注意单纯选择两个最近叶节点是不够的, 即达到最小距离 d_{ij} 的 i, j 对. 图7.7说明了为什么会这样. 如果一对邻居中的一个边较短, 而另个边较长, 那么与真的邻居相比, 有短边的那个就可能与另一个叶节点更接近, 如图所示. 为了避免这种情况, 我们使用的技巧是: 减去一个到所有其他叶节点间距离的平均值; 事实上, 它为长边提供了补偿. 我们定义

$$D_{ij} = d_{ij} - (r_i + r_j),$$

其中

$$r_i = \frac{1}{|L| - 2} \sum_{k \in L} d_{ik}, \tag{7.4}$$

且 $|L|$ 表示叶节点集合 L 的大小. 现在我们称 D_{ij} 最小的一对叶节点 i, j 为邻居, 本章最后将给出证明. 可以检验该结论对图7.7中的树是正确的, 而且我们能够从中得到启发(见练习7.9).

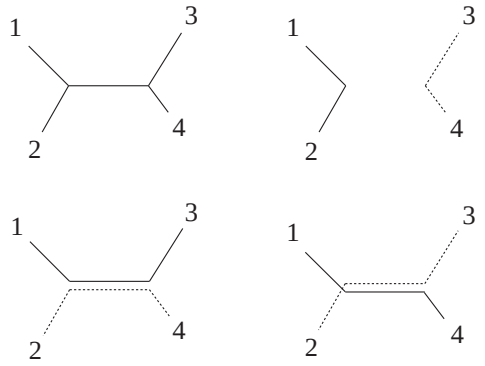


图 7.8: 可加性意味着长度之和 $d_{12}+d_{34}$、$d_{13}+d_{24}$ 和 $d_{14}+d_{23}$ 中的两个必须相等并且大于第三个. 如果通过边长相加得到成对距离, 则这个结论成立, 如图所示.

完整的邻接 (neighbour-joining)算法分步构建树 T, 并保存树的活动节点列表 L. 如果有一棵事先存在的可加树, L 就是邻近对被去掉后剩余的叶节点集合, 而 T 是由这些去掉的节点构成的树.

算法: 邻接法

初始:

定义 T 为叶节点集合, 每个叶节点对应一条给定序列, 并令 $L=T$.

迭代:

从 L 中选择一对 i,j 使由 (7.4) 定义的 D_{ij} 最小.

定义新节点 k, 并对 L 中的所有 m 令 $d_{km}=\frac{1}{2}(d_{im}+d_{jm}-d_{ij})$.

将 k 以边长 $d_{ik}=\frac{1}{2}(d_{ij}+r_i-r_j)$、$d_{jk}=d_{ij}-d_{ik}$ 加入到 T 中, 分别连接 k 到 i 和 j.

从 L 中删掉 i 和 j, 添加进 k.

终止:

当 L 只剩两个叶节点 i 和 j 时, 添加 i 与 j 间的剩余边, 其边长为 d_{ij}. ◁

如果满足可加性, 则由 $\frac{1}{2}(d_{ij}+r_i-r_j)$ 定义的 d_{ik} 是正确的长度, 因为这个表达式是所有叶节点 m 上 $\frac{1}{2}(d_{ij}+d_{im}-d_{jm})$ 的平均值, 且这种项中的每个都正好是 d_{ik} (请比较 (7.3) 式).

可加性是一种依赖于距离度量的性质: 对于一种距离度量, 某棵树可能是可加的, 但对另一种可能就不是. 在8.6节我们会看到, 某种类型的最大似然距离度量在大量数据的极限条件下被认为可以给出可加性, 如果它所基于的模型假设是正确的. 当然, 真实数据至多是近似可加的.

即使边长不可加我们仍然可以使用邻接法, 但不再保证能重建正确的树. 就像超度量条件能检验分子钟性质一样, 我们可以使用下面的距离属性作为可加性的一种检验: 对每四个叶节点 i、j、k和 l, 距离 $d_{ij}+d_{kl}$、$d_{ik}+d_{jl}$ 和 $d_{il}+d_{jk}$ 中的两个必须相等且比第三个大. 这种**四点条件** (four-point condition) 是可加性的一个推论, 因为其中两个和包含了连接叶节点对的"桥" (bridge) 长度 (见图7.8).

练习

7.9 证明在图7.7的树上最短距离 D_{ij} 对应邻近叶节点.

7.10 证明对一棵带四个叶节点的树, 一对邻居的 D_{ij} 要比所有其他节点对的 D_{ij} 小"桥长" , 即树中连接两个分支节点的边长.

为树定根

与 UPGMA 不同, 邻接法仅产生无根树. 寻找树根便成为第二个任务, 这可以通过添加**外群** (outgroup) 或借助某些物种来实现, 这些物种相对于剩下每个物种的距离要比它们彼此之间的距离远. 于是原树与外群的连接点便成为根的最好候选

位置. 例如在图7.1上方的树中, 蝾螈 (axolotl) 可以被认为是一个外群, 因为它是两栖动物而其他物种都属于羊膜动物 (amniote, 或称脊椎动物). 因此合理的做法是将蝾螈与其他物种间的分支置于其他所有分支之前.

如果没有合适的外群, 我们可以使用一些特别的策略, 例如如果与分子钟偏差不大, 我们就可以选择连续边之最长链的中点作为期望的树根.

7.4 简约法

现在我们介绍可能是使用最广泛的建树算法: **简约法** (parsimony). 它的工作原理是找到一棵树, 以便能够用最少的替换数目解释观测序列. 它使用的策略与前面考虑的基于距离的算法有所不同. 与构建一棵树不同, 它为一棵已知树赋罚值, 然后有必要搜索所有的拓扑或采用更高效的搜索策略以便确定"最好" 的树(见第 p. 123页). 因此我们可以把该算法分为两部分:

(1) 计算给定树 T 的罚值;

(2) 对所有树进行搜索, 以便找到这个罚值的全局最小值.

让我们从一个例子开始. 假如我们有下面四条已联配的核酸序列:

```
AAG
AAA
GGA
AGA
```

我们可以尝试用这四条序列构造不同的树, 对每棵树中所需的替换进行计数并在所有位点对它们求和. 图7.9展示了用以上四条序列构造的三棵可能的树; 它们在序列与叶节点的对应顺序上有所不同. 在每棵树中, 我们已经给出了祖先节点处的假设序列以便找到整棵树所需的最小改变数目. 稍后我们会看到此过程是如何实现的. 最左边的树(共三个改变) 比它右边的两棵树(均含四个改变) 所需的改变要少.

正如在这个例子中看到的, 简约法独立对待每个位点, 然后将所有位点的替换相加. 因此基本步骤就是在给定拓扑和叶节点残基分配的情况下, 对每个位点需要做出的最小改变进行计数. 有一种简单算法可以执行这一步骤. 首先考虑简约法的一种简单扩展, 称为**加权简约法** (weighted parsimony) 它不仅对替换进行计数, 还为每个 a 到 b 的替换赋予罚值 $S(a,b)$; 我们现在的目标是最小化这个罚值 [Sankoff & Cedergren 1983]. 当对所有 a 有 $S(a,a)=0$, 且对所有 $a \neq b$ 有 $S(a,b)=1$ 时, 加权简约法就简化为传统的简约法.

为了计算位点 u 处的最小罚值, 我们进行如下操作: 令 $S_k(a)$ 表示将 a 分配到节点 k 时的最小罚值.

算法: 加权简约法

初始:

令 $k=2n-1$ 为根节点的编号.

递归: 所有的 a 计算 $S_k(a)$ 如下:

如果 k 是叶节点则:

当 $a=x_u^k$ 时令 $S_k(a)=0$, 否则 $S_k(a)=\infty$.

如果 k 不是叶节点则:

对所有位于子节点 i,j 处的 a 计算 $S_i(a)$ 和 $S_j(a)$,

并定义 $S_k(a)=\min_b(S_i(b)+S(a,b))+\min_b(S_j(b)+S(a,b))$.

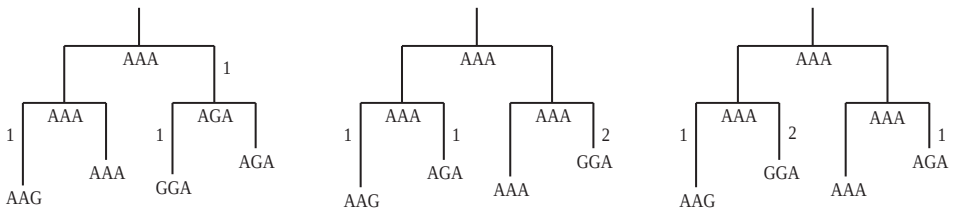


图 7.9: 用简约法构树.

终止:
树的最小罚值 $= \min_a S_{2n-1}(a)$. ◁

注意"递归" 步骤要求计算 k 的子节点 i 和 j 处的 S_i 和 S_j 值, 而这需要对 i 和 j 都"递归" 计算才能实现. 此算法相当于从叶节点开始向上直到根. 这种遍历树的方法称为**后序遍历** (post-order traversal) , 它是许多树算法计算机实现的重要组成部分.

有时我们对能给出最小罚值的祖先残基分配感兴趣. 例如一种定义边长的方法可以是, 记录沿某条边上发生的失配数目, 而这些失配出现于树中所有可能的最小罚值祖先分配中. 我们可以保存从节点 k 处每个残基 a 分别到子节点 i 和 j 处的残基 b 和 c 的指针以便执行该操作, 而这些指针是加权简约算法里定义 $S_k(a)$ 的方程中的最小选择. 我们定义节点 d 的左右子节点指针分别为 $l_k(a)$ 和 $r_k(a)$ (如果有多个使罚值达到最小的残基, 则这两个指针也许会指向多个目标), 然后在加权简约法中"递归" 部分的末尾添加下面的步骤:

$$\text{令} l_k(a) = \text{argmin}_b(S_i(b) + S(a,b)),$$

$$\text{且} r_k(a) = \text{argmin}_b(S_j(b) + S(a,b)). \tag{7.5}$$

为了得到祖先残基的分配, 我们选择给出最小罚值 $S_{2n-1}(a)$ 的根残基 a, 然后通过指针回溯到叶节点, 当指针指向多个目标时任意选择一个进行回溯.

在传统简约法中我们只对残基替换进行计数, 为了获得树的罚值我们只需保存每个节点处最小罚值残基的列表以及当前罚值 C. **算法: 传统简约法 [Fitch 1971]**

初始:
令 $C = 0$ 且 $k = 2n - 1$.
递归: 为了得到集合 R_k:
如果 k 是叶节点则:
令 $R_k = x_u^k$.
如果 k 不是叶节点则:
计算 k 的子节点 i 和 j 的 R_i 和 R_j,
如果交集 $R_i \cap R_j$ 非空, 则令 $R_k = R_i \cap R_j$
否则令 $R_k = R_i \cup R_j$, 并自加 C.
终止:
树的最小罚值 $= C$. ◁

在传统简约法中存在一个找到祖先分配的回溯过程: 我们从 R_{2n-1} 中选择一个残基, 然后沿树向下进行. 从 R_k 中选出一个残基后, 如果有可能就从其子节点的集合 R_i 中选择相同残基, 否则从 R_i 中随机选择一个残基(对 k 的另一个子节点的集合 R_j 做同样处理).

图7.10中的树 T 显示了用这种回溯过程得到的祖先残基的两个可能集合(中间的两个图). 左下角的图展示了另一种分配, 它不能用这种回溯方法得到. 其原因是在每个节点保存最小罚值残基列表, 但忽略了这样一种可能性, 即一个残基失配的罚值可能在树的某个层次获得支付但在更高层次上得到补偿. 加权简约算法自动考虑了这种情形. 右下角的图展示了每个节点处 `A` 和 `B` 的最小罚值, 并标出了由(7.5)定义的导致左下角图中树的残基分配之回溯指针特定选择. 注意, 从顶端节点引出的左指针指向残基 B, 其罚值比最小值多一. 这个差值可以得到补偿, 因为 B 不一定需要为这个转移支付失配罚值. 一般而言, 由回溯 R_k 无法得到的分配可以通过如下方法获得: 保存节点 k 处的一组残基 Q_k, 而其罚值比 R_k 中残基的罚值多一. 我们可以很容易地扩展传统简约算法, 使得在计算 R 的同时计算出 Q.

我们已经阐明了在有根树情形下的简约法. 然而传统简约法中树的最小罚值与树根位置无关. 事实上在最优树中, 根节点下方的两条边不可能同时拥有残基替换, 否则我们就可以将根节点的残基分配变为其下方一个节点的残基分配, 因为这会减少罚值. 原则上, 这意味着根节点可以被删除, 罚值可以沿着无根树的边而计算. 如果碰巧发生这种情形, 那么在有根树上计算罚值就变得十分简单, 因为在应用简约算法时, 根节点定义了一个方向, 也就是定义了唯一的亲子 (parent-daughter) 关系. 但树根位置的独立性意味着需要搜索的树之数目减少了.

练习

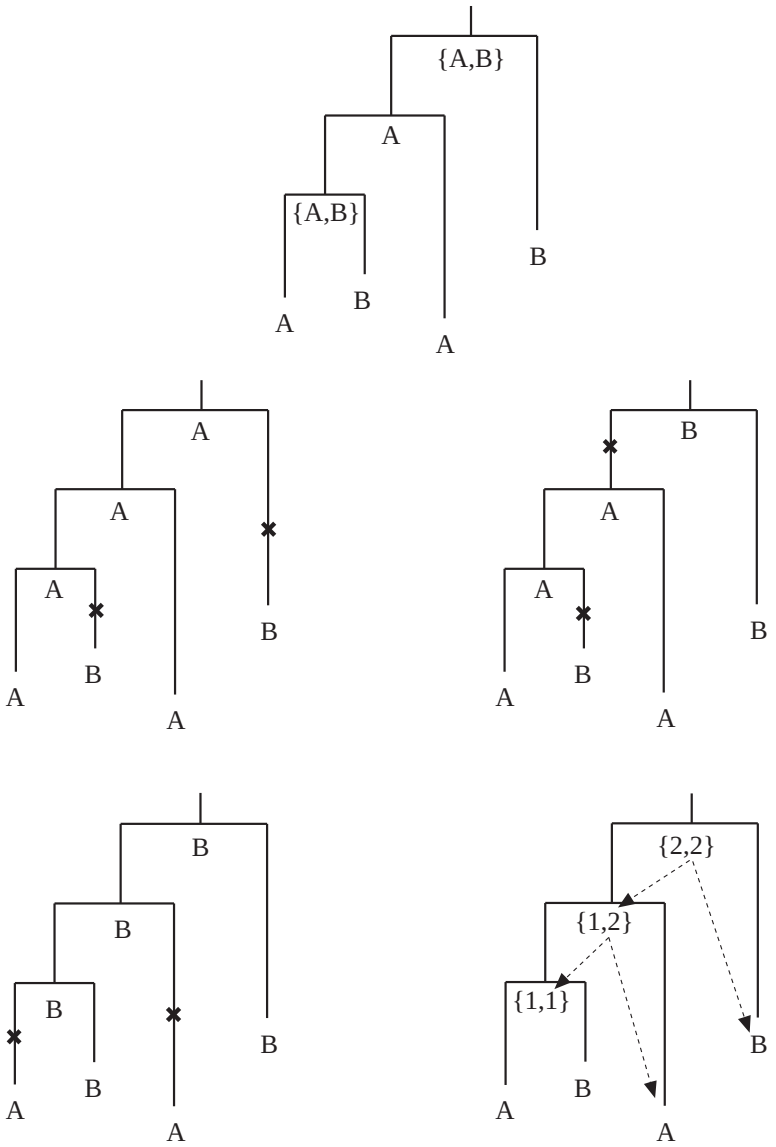


图 7.10: 传统简约法, 每个替换罚1分(在边上用" X " 标记), 其他情况罚0分. 集合 R_k 显示在顶部的树中, 中间两棵树画出了用集合 R_k 回溯得到的祖先残基分配. 左下角的树给出了另一个符合条件的祖先残基集合, 但它不能由这个方法得到; 右下角的树展示了如何用加权简约法(7.5) 的回溯过程得到这个残基分配.

7.11 证明传统简约法给出与加权简约法相同的罚值, 如果对所有 a 权重 $S(a,a)=0$ 且对所有 $a \neq b$ 权重 $S(a,a)=1$. (提示: 证明, 要得到每个节点 k 处有最小罚值 $S_k(a)$ 的残基 a, 只要在每个节点处保存列表 R_k 即可.)

7.12 假设替换罚值是一种度量, 即对所有 a,b,c 都有 $S(a,a)=0$, 对称性 $S(a,b)=S(b,a)$ 以及三角不等式 $S(a,c) \leq S(a,b)+S(b,c)$, 那么请证明加权简约法的

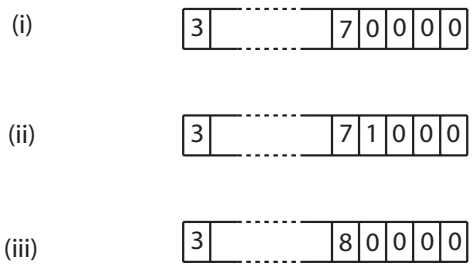


图 7.11: 用于对无根树进行计数的里程表.

最小罚值同样与根节点位置无关. [4] (提示: 如果根节点有一个残基A而其两个子节点有不同的残基 B 和 C, 证明从三角不等式可以推导出此罚值并非最小. 使用度量的另外两个性质证明根节点可以移动到其任意子节点而不增加罚值.)

用分支定界法 (branch and bound) 选择有标分支型

我们已经明白因为只需要考虑无根树, 所以可以减少简约法搜索树的数目. 即便如此, 随着叶节点数目的增加, 拓扑的数目仍急剧变大. 因此我们需要比简单枚举更有效的搜索策略.

有些方法随机地对树进行搜索; 例如我们可以交换一棵树上随机选取的分支, 如果改变后的树比当前树得分更高, 那么就选择改变后的树. 然而这不能保证找到全局最优树. 另一种策略用每次添加一条边的方法构建树. 随机选择三条序列, 并将其放置到一棵无根树上. 接着选择另一条序列并把它添加到一条边上, 使得由这四条序列构成的树计分最高. 再随机选一条序列, 把它添加到计分最高的位置, 如此继续直到建树完成. 这种方法同样不能保证找到全局最优树, 而且事实上按不同顺序添加序列可能得到不同的最终树[Felsenstein 1981a].

在简约法中有一种替代策略可以保证找到最优树; 它利用了如下事实: 树中替换数目的增加只能由添加额外边获得. **分支定界法** (branch and bound) 背后的思想是从不断增加的叶节点开始系统地建树, 然而一旦当前不完全树之罚值已经超过至今得到的完全树之最小罚值, 它就丢弃特定的建树途径.

我们通过一个数组 $[i_3][i_5]\ldots[i_{2n-5}]$ 枚举所有的无根树, 其中每个 i_k 取值 $1\ldots k$. 相应的树按如下方法得到: 选取有三条序列 x^1、x^2 和 x^3 的无根树, 在标有 i_3 的边上为 x^4 添加一条边. 因为这条新的边将一条已经存在的边分成两部分, 所以边的总数现在变为 $3+2=5$. i_5 的值决定了我们将 x^5 添加到哪里, 于是给出 $5+2=7$ 条边. 如此继续直到 x^n, 它有 $(2n-5)$ 个可以选择的位置.

现在将 $[i_3][i_5][i_7]\ldots[i_{2n-5}]$ 看作汽车仪表盘上的里程表 (milometer, 在美国称为 odometer). 最右端的数字一直增加直到 $2n-5$ 后, 它们回到 1, 并且右边第二个数组的指标进位 1. 当右边第二个数组的指标达到 $2n-7$ 后, 它又从 1 开始, 且让右边第三个数组的指标进位 1, 如此反复进行.

这样就枚举出了某特定序列之所有带 n 个叶节点的树, 但我们仍然想对少于 n 个叶节点的树进行计数, 因为我们要构建可变大小的树. 于是可以向每个计数器添加一个"0" , 它表示计数器所指定顺序的边不存在, 而且我们让每个指标都从0到 i_k 进行循环. 但是这样会产生一些无意义的值, 因为我们不能向一条不存在的边添加额外的边, 即我们不能使非零计数器位于 0 的右边. 因此, 当计数器右侧有一串0的时候, 我们就不得不把最左边的"0" 进位成"1" , 以进行下一步(例如图7.11中从 (i) 到 (ii)).

[4] Sankoff & Cedergren [1983] 假设其罚值是一个度量, 但此性质仅在这里是必须的. ——原注

该过程开始时, 里程表设为 [1][0][0]...[0]. 令至今为止完全树的最小罚值为 C. 一旦我们当前树T的罚值比 C 大, T 就不是最优树. 但 (这里有个诀窍) 如果这种情况发生在某个非零计数器右边都是 0 时, 那么我们就可以直接把最右边的非零计数器加 1 而不是让这些0都变成"1" . 这样做的原因是, 最右边非零计数器定义了一棵带 $k < n$ 个叶节点的树, 而添加更多的叶节点只能增加罚值. 因此我们能处理下一棵带 k 个叶节点的树(例如图7.11中从 (i) 到 (iii)). 这种方法可以节省大量搜索.

7.5 树的评估: 自举法 (bootstrap)

我们此前描述的建树方法只产生一棵树, 或在简约法中只产生几棵最优树, 但并未给出我们能够在多大程度上信任这些树. Felsenstein [1985]建议使用自举法 [Efron & Tibshirani 1993]作为评估某些系统发育特征显著性的方法, 例如某一特定物种集合在它们分支(一个"进化枝" (clade)) 上的分离特性.

自举法这样进行: 已知一个包含序列联配的数据集, 通过从联配中随机有放回 (with replacement) 地挑选列, 生成另一个相同大小的人工数据集. (于是原数据集的某列可以在人工数据集中重复出现.) 然后对这个新的数据集实施建树算法, 并将随机选列与构树的整个过程重复若干次, 一般来讲重复次数要达到 1000 次的数量级. 某个特定的系统发育学特征出现的频率可以作为我们对此特征之置信度的一个度量.

对某些概率论模型而言, 系统发育学特征 F 的自举法频率可以被证明用于近似后验分布 $P(F|\text{data})$ (见第 p. 146 页). 当自举法用于非概率论形式的模型, 例如简约法时, 我们可以用统计假设检验的术语对其进行解释, 然而若要将自举法纳入到置信区间的标准概念中去, 就需要对此过程进行更加详尽的阐述 [Efron, Halloran & Holmes 1996].

7.6 同步联配与系统发育

我们现在将所考虑的问题转到同时进行序列联配并找到它们间看起来合理的系统发育树. 有两种简约型 (parsimony-type) 算法可以解决这个问题, 第一种使用空位的字符替换模型, 第二种使用仿射空位罚分. 两种方法都能为给定树找到最优联配; 有必要搜索所有树以找到全局最优.

Sankoff & Cedergren 的空位替换算法

Sankoff & Cedergren 的算法保证可以找到祖先序列以及它们与叶节点序列间的联配, 所有这些结果使得基于树的简约型罚值达到最小 [Sankoff & Cedergren 1983]. 事实上, 该算法是组合了本书曾经介绍过的两种方法 (图7.12). 在第6章中(第 p. 97页), 我们描述了联配一组 N 条序列 $x^1, x^2, \ldots, x^N$ 的动态规划方法 [Sankoff & Cedergren 1983; Waterman 1995]. 用取最小函数min 代替取最大函数max 后(我们在这里使用罚值而不是计分), 以 $x^1_{i_1}, x^2_{i_2}, \ldots, x^N_{i_N}$ 结尾的联配之最小罚值 $\alpha_{i_1,i_2,\ldots,i_N}$ 变为

$$\alpha_{i_1,i_2,\ldots,i_N} = \min_{\Delta_1+\ldots+\Delta_N>0}\{\alpha_{i_1-\Delta_1,i_2-\Delta_2,\ldots,i_N-\Delta_N} + \sigma(\Delta_1 \cdot x^1_{i_1}, \Delta_2 \cdot x^2_{i_2}, \ldots, \Delta_N \cdot x^N_{i_N})\}. \qquad (7.6)$$

其中 Δ_i 的值是 0 或 1, 且当 $\Delta_i = 1$ 时 $\Delta_i \cdot x = x$, 而当 $\Delta_i = 0$ 时 $\Delta_i \cdot x =$ "$-$" . σ 是为联配扩展字母表中的一组符号而设置的加权简约法罚值. 我们可以使用加权简约法(第 p. 120页) 沿树向上计算出这个罚值, 其中现在定义的 $S(a, b)$ 不只局限于 a, b 都是残基的情形, 也包括了 a, b 中的一个或两个是空位符"–" 的情形.

因此 Sankoff & Cedergren 的程序过程如下: 当我们取到 $(i_1, i_2, \ldots, i_N)$ 时, 对 $\Delta_1, \ldots, \Delta_N$ 的所有 $2^N - 1$ 种组合而言, 式 (7.6) 中每一项 $\alpha_{i_1-\Delta_1,i_2-\Delta_2,\ldots,i_N-\Delta_N}$ 都已经预先计算好了, 并且 $\sigma(\Delta_1 \cdot x^1_{i_1}, \Delta_2 \cdot x^2_{i_2}, \ldots, \Delta_N \cdot x^N_{i_N})$ 可以沿树向上计算出来, 这需要数量级为 N 步的计算(每条边一步). 因此整个计算需要 $N(2n)^N$ 步, 其中 n

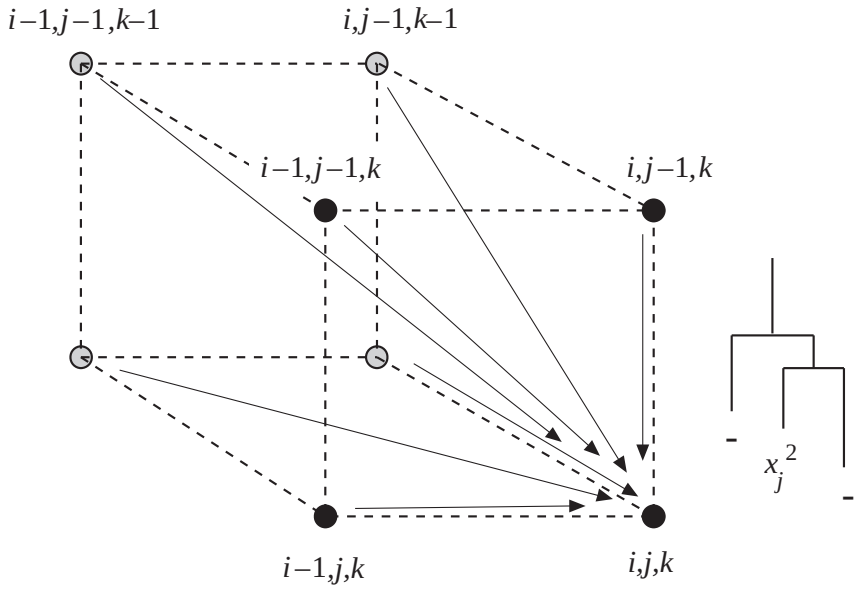


图 7.12: 用 Sankoff & Cedergren 算法联配 3 条序列时的动态规划矩阵. 矩阵中的转移用箭头表示. 对 3 条序列而言, 每个点都有 7 个转移. 为每个转移都分配一个罚值, 它是由简约算法导出的最小罚值, 树的叶节点由这个转移定义如下: 如果一个坐标减少 1, 那么相关叶节点就被分配为输入序列的前一个字符; 如果坐标不变, 它的叶节点就被分配一个"–" . 例如, 从 $(i, j-1, k)$ 到 (i, j, k) 的转移被分配为如右图所示的树.

是序列的长度. 令人遗憾的是, 即使对于 6 条普通长度 (100个残基的量级) 的序列来讲这个计算量也还是太巨大了.

Hein 的仿射罚值算法

Hein 的算法 [Hein 1989a]使用仿射空位罚值, 这比使用空位的简单替换处理要现实得多. 在大多数实际情况中, 这个算法也比 Sankoff & Cedergren 的算法要快很多, 快到允许对一个大小适中的序列集搜索树的拓扑. 这是现在唯一能够有效联配序列并寻找其备选系统发育树的实用算法. 在获得上述可观突破的同时, 该算法也付出了代价: 它从简化的假设出发选择祖先序列, 而这并不能保证找到全局最简约选择.

假设我们给定一棵树. 回想一下, 传统简约算法是沿树向上, 为每个节点分配一个可能的残基列表. 这些残基恰好使沿每条边到达两个子节点的替换数目最小. 此时通过最小化每个节点, 我们有可能找到整棵树的最小替换数. Hein 算法也使用了相同的步骤: 在向上通过树时, 如果给定两个子节点的序列, 那么在该节点处我们只需考虑那些导致最小罚值的序列. 我们以后会发现, 与传统简约法不同, 这种方法并不能保证为整棵树找到最小罚值. 但首先还是让我们看一下如何确定每个节点的最小罚值序列.

我们的目标是在一个给定节点处找到序列 z, 使它可以与该节点的子节点序列 x 和 y 都联配上, 并满足

$$S(x, z) + S(z, y) = S(x, y), \tag{7.7}$$

其中 S 表示两条序列的已知联配之总罚值. 假设失配的罚值为1, 否则罚值为0, 而且如果在每个位点 z 与 x 或 y 中的一个有相同残基(当 x, y 残基相同时, 与两者都相同), 则式 (7.7) 在任意位点处都成立. Hein 算法也可扩展为一般的加权简约法 (见练习7.13).

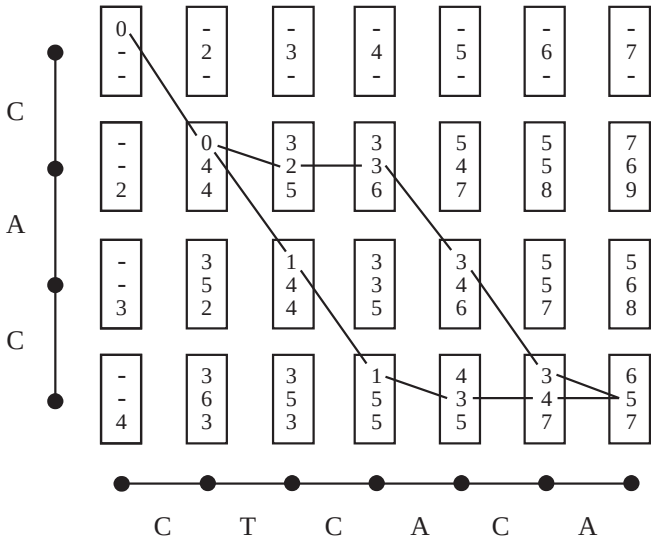


图 7.13: 两条序列的动态规划矩阵, 每个单元格从上到下依次为 V^{M}、V^{X} 和 V^{Y} 的值. 最优路径用单元格间的连线表示. 这里 $d = 2, e = 1$, 且失配的罚值是1. 注意我们还包括了这样的罚值, 即在一条序列中是空位而后紧跟着在另一条序列中也是空位的罚值(例如正数第二行, 左起第二列的单元格元素 $V^{\mathrm{Y}} = 4$, 它来自其上的单元格元素 $V^{\mathrm{X}} = 2$), 即使这样的匹配是非最优的.

因为还必须对空位进行处理, 所以我们没有证明一定能找到满足 (7.7) 式的序列 z. 为此, 我们使用第2章里描述过的仿射空位动态规划方法. 令 $V^{\mathrm{M}}(i,j)$ 、 $V^{\mathrm{X}}(i,j)$ 、 $V^{\mathrm{Y}}(i,j)$ 分别表示如下三种情况下, 结束于序列 x 中位置 i 和序列 y 中位置 j 的联配之最小罚值: (1) 序列 x 中第 i 个残基联配上序列 y 中第 j 个残基, (2) 序列 x 中第 i 个残基联配上序列 y 中的一个空位, (3) 序列 y 中第 j 个残基联配上序列 x 中的一个空位. 这些分别与到达匹配状态 M(i,j)、插入状态 X 和 Y 的 Viterbi 罚值相对应. 我们把 V^{M}、V^{X} 和 V^{Y} 三个数字写到动态规划矩阵的单元格 (i,j) 中(图7.13). 令长度为 k 的空位之仿射空位罚值为 $d + (k-1)e$, 其中 $e \leq d$. 于是递归公式为:

$$\begin{aligned} V^{\mathrm{M}}(i,j) &= min\{V^{\mathrm{M}}(i-1,j-1), V^{\mathrm{X}}(i-1,j-1), V^{\mathrm{Y}}(i-1,j-1)\} \\ &\quad + S(x_i, y_j), \\ V^{\mathrm{X}}(i,j) &= min\{V^{\mathrm{M}}(i-1,j) + d, V^{\mathrm{X}}(i-1,j) + e\}, \\ V^{\mathrm{Y}}(i,j) &= min\{V^{\mathrm{M}}(i,j-1) + d, V^{\mathrm{Y}}(i,j-1) + e\}. \end{aligned} \tag{7.8}$$

在这里我们假设失配的罚值比 $2e$ 小, 保证最优联配将不包含如下情况: 一条序列的一个空位紧跟着另一条序列的一个空位, 如 $\begin{smallmatrix}\mathtt{TTAC--}\\ \mathtt{TT--GG}\end{smallmatrix}$, 而是更倾向于将残基进行配对, 如此例中 $\begin{smallmatrix}\mathtt{TTAC}\\ \mathtt{TTGG}\end{smallmatrix}$.

我们标识了出现在最小罚值路径上的所有转移(如图7.13所示). 连接这些转移的任意路径都将给出 x 和 y 的最优联配. 现在假设序列 x 和 y 是节点 n 的两个子节点序列. 使用我们已标记转移的任意路径也都可以用于定义 n 处符合条件的祖先序列, 其过程如下. 如果一个转移对应 x 和 y 中两个残基的匹配, 那么我们就选择这两个残基中的一个作为祖先序列的残基. 如果一个转移对应残基和空位的匹配, 那么选择空位或者那个残基作为祖先序列的残基.

这将产生一条联配到 x 和 y 的序列 z. 从 z 的构造过程来看, 很明显如果在某个位点处 x 和 y 都是残基, 那么 z 将与 x 或 y 的残基相同 (或可能与二者都相同). 因此 (7.7) 式两边将会得到相等的贡献. 假如留意对空位开端罚值的处理, 那么当 x 或 y 有一个是空位时我们也会得到同样的结论. 事实上我们需要如下的额外规则以构

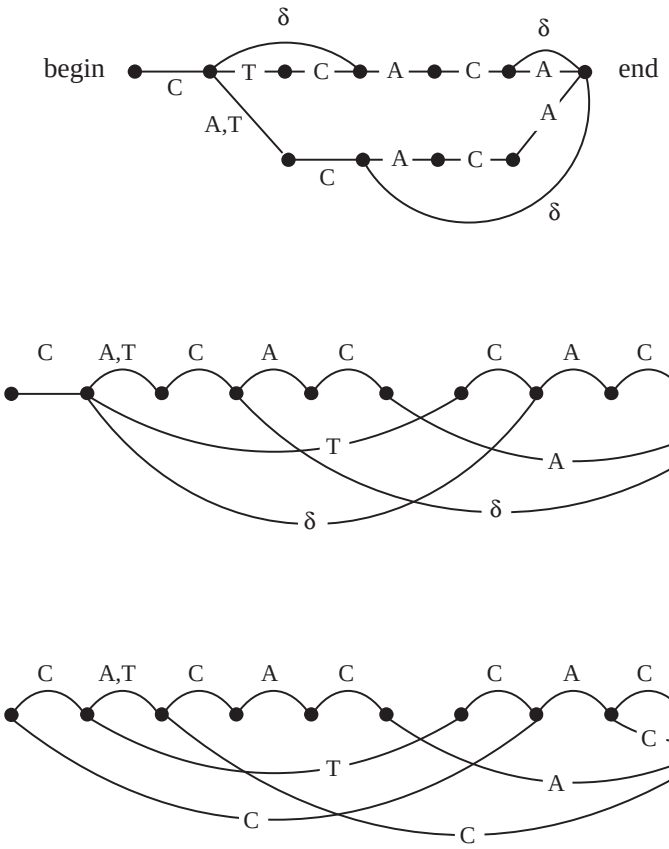


图 7.14: 从能够贯穿图7.13中动态规划矩阵之路径导出的序列图. 上图: 带空边(用 δ 标记) 的图. 中图: 同样的图, 只是节点排成了一行. 下图: 空边被这样一条边代替: 它返回到"前驱" 顶点并发射出连接到该边的一个残基.

造祖先序列: 如果路径上的某条序列出现连续空位, 并且它们都联配上另一条序列的一组残基, 那么祖先序列必须全部跳过这些残基, 或者全部包含这些残基.

例如给定两条序列 `CAC` 和 `CTCACA` (见图7.13), 序列 `CTC` 可以这样导出: 首先沿着矩阵中靠下的路径, 接着对应于联配 $\frac{\texttt{CAC---}}{\texttt{CTCACA}}$, 然后在第二个位置选择 `T`, 最后跳过连续三个空位. 这是一条可能的祖先序列, 因为它能与 `CAC` 构成联配 $\frac{\texttt{CTC}}{\texttt{CAC}}$, 并且由于 `A , T` 的失配而获罚值1; 它也能与 `CTCACA` 构成联配 $\frac{\texttt{CTC---}}{\texttt{CTCACA}}$, 获罚值 $d+2e$. 两个罚值的和 $d+2e+1$ 就是原来联配 $\frac{\texttt{CAC---}}{\texttt{CTCACA}}$ 的罚值. 类似地, `CACACA` 也是一条符合条件的祖先序列, 选择与空位匹配的连续残基时便可以得到它. 这里不允许只使用其中的部分残基. 例如 `CACAC` 就不是一条祖先序列. 因为此时到子节点序列的最优联配是罚值为 $d+e$ 的 $\frac{\texttt{CACAC}}{\texttt{CAC--}}$ 以及罚值为 $d+1$ 的 $\frac{\texttt{CACAC-}}{\texttt{CTCACA}}$, 而此二者的罚值都包含空位开端项. 我们假设 $d > e$, 因而这两个罚值的和 $2d+e+1$ 超过了原来联配的罚值 $d+2e+1$.

现在我们从动态规划矩阵导出一张图, 以便正式描述沿贯穿该矩阵之路径而行的思想. 一旦动态规划矩阵某单元中三个数字里的某项被最优路径采纳, 我们就在图中用一个顶点表示. 必须注意, 相同单元中不同的项需要不同的顶点与之对应. 图7.13中的情况可以解释其原因: 两条最优路径在倒数第二个单元发生交叉, 一条使用 `M` 状态, 罚值为 3, 另一条使用 `X` 状态, 罚值为 4. 如果交换这个单元里的路径, 我们就会无法辨别当时是正在打开一个空位还是正在延伸一个空位, 以至失去回溯的线索 (见 Altschul & Erickson [1986]).

图中的有向边是发生在最优联配中的转移. 我们为所有这些边分配了残基: 如果矩阵的转移在一对残基处停止, 那么这两个残基就都要连接到该边上; 如果是一个残基对一个空位, 那么只把残基连接到该边上. 最后将对应于两个或多个连续空

位块的"空边" (dummy edge) 添加进去. 这些空边从该空位块的起始顶点指向块的结尾, 无需为其分配残基.

现在我们考虑贯穿该图的路径, 它从初始点出发, 而其终点能够匹配上每条序列的最后一个残基. 规则是任意路径都把符号发射到它所使用的边上, 当有多个可用符号时, 任选其一. 显而易见, 图中任意路径都发射出一条满足条件的序列. 该图称为**序列图** (sequence graph) .

如果两个子节点都是树的叶节点, 也就是只有一条序列与之相连, 该构造方法同样适用. 当沿树向上, 且存在多条可以分配给子节点的符合条件之序列时, 又会怎么样呢? Hein 想出了一个聪明的办法, 即仍然执行相同的构建过程, 但用图而非序列作为动态规划矩阵中要被匹配的对象.

为了达到这个目的, 我们首先将每个图展开使其顶点位于一条直线上 (图7.14的中图); 这一步总是能实现, 于是所有边都指向同一方向. 假设有两个图 G_1 和 G_2. 同样, 我们也知道了每个单元格中 V^{M}、V^{X} 和 V^{Y} 的值. 然而, 与考虑来自序列中前一个残基的转移不同, 我们现在用图中的入边 (incoming edge) 定义"前驱" (preceding). 当这些边中包含一条空边时, 我们就可以跳回到它起始的顶点, 然后用前驱非空边定义一个前驱顶点. 注意, 因为空边会覆盖整个空位块, 并且失配罚值小于 $2e$ 的条件排除了一条序列中的一个空位块紧接着另一条序列中的一个空位块的情形, 所以不可能存在连续的空边链. 因此不可能发生前驱节点个数的组合爆炸 (combinatorial explosion).

对空边的处理可以这样执行: 首先修改每个序列图, 删除所有的空边(图7.14中用 δ 标出), 并将每条空边都替换为这样的边: 它的起始点在该空边起始点的前一位, 而且它所带有的符号与该空边前驱边的符号相同. 图7.14的中图和下图画出了这个替换过程. 现在我们可以简单地说, 通过沿修改后的 G_1 或 G_2 中的边而行, 就可以得到动态规划矩阵中的那些转移.

定义过矩阵中单元格里的 V 值后, 我们就能够像以前一样通过回溯过程得到最优路径, 于是可以构造下一个序列图 G_3. 这里出现了一个新的特点. 序列图中的边可以带有多个符号, 它们就是传统简约法中的集合 R_k. 与传统简约法相同的过程控制了多组符号的合并: 如果 V^{M} 由 G_1 的一条边和 G_2 的一条边定义, 且存在一个共用符号(即如果使用这两条边的路径上无失配罚值), 那么只有共用符号被指定到将要生成的 G_3 的边上. 如果没有共用符号, 那么生成的边会从两条边获得所有符号.

现在, 我们便可以在矩阵上执行递归式 (7.8). 贯穿矩阵的最优路径定义另一个图, 由此继续, 沿着树向上直到树的根节点. 然后我们可以沿着树向下, 重构对应于一个给定祖先序列的子节点序列. 为此, 在追溯祖先序列的同时, 我们跟踪每个子序列图中边的序列, 选择子序列中与祖先序列相一致的的符号. 如果一个删除或一系列删除跳过了某个子序列图中的一些连续节点, 那么必须用任意选择的符号填充被跳过的边. 例如当追溯图7.13中下面一条路径的祖先序列 `CAC` 时, 先产生子序列 `CTCACA` 的前三个符号 `CTC ...`, 然后必须添加上后三个符号 `... ACA`, 即使在祖先路径中它们被空边(图7.14上图中长度为 3 的 δ 边) 跳过. 图7.16展示了一棵树的可能祖先序列; 该树的序列图见图7.13和图7.15.

我们已经描述了给定一棵树后, 序列是如何联配的. 我们也有必要在树中进行搜索, Hein [1989b]对此提出了他自己的高效搜索算法. 如果序列图不太复杂, 那么整个过程就容易处理. 如果我们要包括动态规划矩阵中的大多数转移, 那么计算的复杂度就会增长到 Sankoff & Cedergren 算法的水平. 这里的假设是, 大多数联配都会包含给出最小罚值的几条主要路线. 如果序列足够相似, 那么该假设便成立, 因为这样将存在能够定义贯穿矩阵之单一路径的大段明确匹配.

练习

7.13 Hein 的算法可以扩展到一般权重 $S(a,b)$, 只需对序列图中每个边都附加上一个最小罚值集合 $S_k(a)$ (就像加权简约法算法里一样) 以代替集合 R_k. 假设对所有 a 有 $S(a,a)=0$, 且 z 与 x 或 y 有一个共用残基, 请证明式 (7.7) 成立. 评估图7.13、图7.14和图7.15中序列图的最小罚值 (假设某个核苷酸字母表).

Hein 模型的局限

现在让我们回到 Hein 算法的过程, 它在向上过程中的每个节点处选择最小罚

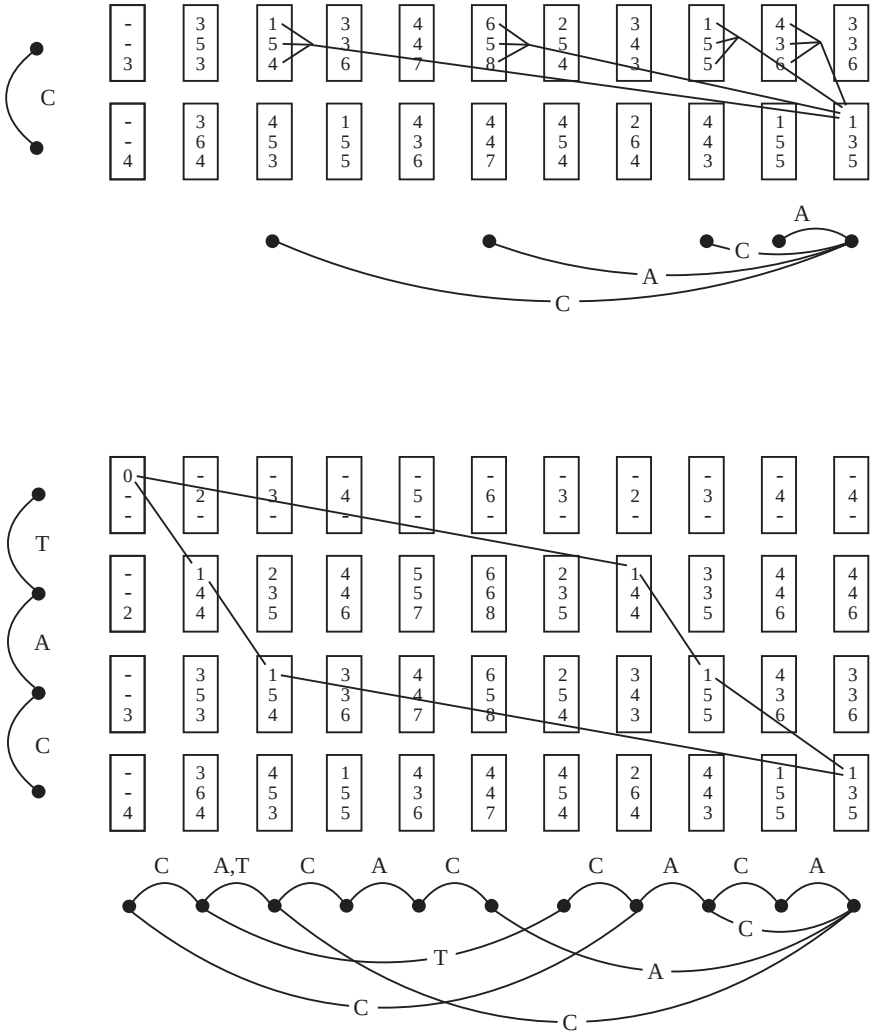


图 7.15: 序列图相对于序列 TAC 的动态规划矩阵, 该序列图来自图7.14的下图. 此矩阵中的序列图为图7.16中树的顶端节点产生可能的祖先序列. 单元格中 V^{M}、V^{X} 和 V^{Y} 的值通过对所有"前驱" 顶点取最小值确定, 我们可以通过到达当前顶点的边返回这些前驱顶点. 上图示例了如何计算某个单元格[6] 中V^{M}的值.

值序列. 为了说明这可能导致无法找到树的全局最优, 假设长度为 k 的空位罚值是 $13+3(k-1)$, 失配罚值是 4 (Hein 用这个值对 5SRNA 进行示例联配). G 和 GTT 的合格祖先序列就是 G 和 GTT 本身; 它们都要求其子序列之一带有连续的空位对, 罚值是 $13+3=16$. 序列 GT 是不合格的祖先序列, 因为它在两个联配中都需要单个空位, 于是总的罚值是 $2\times 13=26$, 即 $\frac{\texttt{G-}}{\texttt{GT}}$ 和 $\frac{\texttt{GT-}}{\texttt{GTT}}$. 但假设我们有这样一棵树, 它的根节点分叉为 G 和 GTT 的祖先, 以及第三条序列为 GT 的叶节点 (见图7.17). 于是该树对不合格祖先 GT 的总罚值就更小, 因为在那种情况下该树需要两个大小为 1 的空位, 相比之下不管使用哪一条合格祖先序列, 我们都需要大小为 1 和 2 的两个空位.

警告 B . Schwikowski 最近已经证明空边的构建有缺陷. 使用 Hein 的序列图, 但避免该缺陷的一种替代办法见 Schwikowski , B . and Vingron , M . 1997. The deferred path heuristic for the generalized tree alignment problem . *Journal of Computational Biology* 4:415-431.

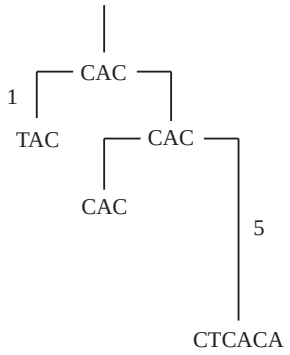


图 7.16: 给定如图所示的树, 叶节点序列 `TAC , CAC , CTCACA` 的可能祖先序列.

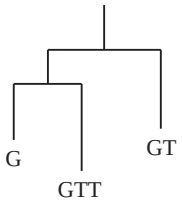


图 7.17: 在选择最优祖先序列时 Hein 规则失效的例子, 此时无法为祖先产生最优全局分配.

7.7 补充读物

简约法首先由 Edwards & Cavalli-Sforza [1963; 1964]在连续参数的框架下阐述, 它被归结为在欧氏空间里寻找最小长度的树之合并点. Camin & Sokal [1965], Eck & Dayhoff [1966], Fitch [1971]和其他人引入了基于序列或其他离散变量, 并用于简约法的计数算法. 这些算法的简洁性以及丰富的序列数据共同促进了这些方法的广泛使用.

有时人们宣称简约法是 Occam 剃刀 (Occam ' s razor) 的一个直接哲学继承, 并宣称它不需要特定的进化假设, 例如"在系统发育分类学中的简约法使用与其他生物学分支或任意其他科学中的应用间不存在区别, (并且) 也不涉及任何特别的进化机制" [7] [Brooks & McLennan 1991, p. 65]. Edwards [1996] 深入地揭示了这种思想的本质. 另外, 本书8.6节对简约法的解释将简约法与本书介绍的概率论模型联系了起来.

系统发育距离方法最早也是在上面提到的 Edwards & Cavalli-Sforza 的论文中描述. 他们提出观测距离与树上长度之和间的一种最小二乘匹配(least-square match). 也可以为邻接法给出最小二乘解释: 此时, 我们将观测到的距离与简化树上的距离进行比较 [Saitou & Nei 1987]. 除了本章讨论过的以外, 还存在许多别的距离方法; 例如 Fitch & Margoliash [1967a]将式 (7.3) 的距离定义与聚类结合到一起.

带可加长度的树具有许多数学结果. 7.8节给出了 Studier & Keppler 的定理, 此外 Buneman [1971]证明, 当一组距离满足四点条件 (见第 p. **??** 页) 时, 我们就可以找到一棵树与一组边长, 使得它们能够生成这些作为边长之和的距离.

遗传物质的水平转移 (horizontal transfer) 给系统发育学提出了有趣的难题. 起初看来, 树的简单结构不再适用, 因为重组事件 (recombinational event) 似乎会在一条序列与它的两个祖先之间制造连接, 因此产生一个环. 但是重组点两侧的每个片段都有单一的父节点, 而基因组可以看作序列片断的连接, 每个片断都有其自

[7] 原文为: The use of parsimony in phylogenetic systematics is no different from its use in any other branch of biology or any other science , [and] does not invoke any particular evolutionary mechanism .——译注

身的树 [Hein 1993]. 重组事件似乎在病毒和原核生命中特别重要, 水平转移也频繁发生. (二倍体"家族树" (family tree) 发生的重组当然更加频繁, 但它需要另一种不同的模型, 该模型由几乎不含序列进化的交联事件 (crossing-over event) 所主导.)

树的一般化理论由 Bandelt & Dress [1992]提出. 他们阐述了如何从距离数据构建像树一样进行分支的网络, 该网络具有很强的拓扑证据, 并且能够产生覆盖模糊区域 (region of ambiguity) 的网格 (mesh).

实用的概括性系统发育学文献包括 Waterman [1995]和 Swofford & Olsen [1996]; 最近的综述见 Saitou [1996]和 Felsenstein [1996].

7.8 附录: 邻接 (neighbour-joining) 定理的证明

为完备起见, 也因为该定理是很令人满意的数学结果, 所以我们引述了 Studier & Keppler [1988]对最短邻接距离的叶节点是邻居的证明. 它保证了带可加长度的树将被邻接法正确地重建. 最近的一篇文章推广了这个结果, 证明邻接法同样可以在可加性仅近似满足的条件下正确地重构树 [Atteson 1997].

定理: 对有可加边长的树, D_{ij} 最小意味着 i 和 j 是邻居叶节点.

证明: 假设最小的 D 是 D_{ij}, 且进一步假设 i 和 j 不是邻居节点, 我们将导出一个矛盾.

因为 i 和 j 不是邻居, 所以连接它们的路径上至少有两个节点(见图7.18). 称这些节点为 k 和 l, 并令 L_k 是从 k 的第三个分叉上导出的叶节点集合, 即不指向 i 或 j 的边, 并令 l 的相应集合为 L_l. 令 m 和 n 是 L_k 中在节点 p 相互连接的一对邻居叶节点(如果不存在这样的叶节点对, 那么就有另一个论断; 见练习7.14). 令 d_{uv} 表示连接任意两个节点 u 和 v 的路径上的边长之和. 由可加性, 当它们两个都是叶节点时, d_{uv} 就是正确的距离. 对 L_k 中的任意 y, 显然有 $d_{iy} + d_{jy} = d_{ij} + 2d_{ky}$ (见图7.18的下图). 类似地有 $d_{my} + d_{ny} = d_{mn} + 2d_{py}$. 因此

$$d_{iy} + d_{jy} - d_{my} - d_{ny} = d_{ij} + 2d_{ky} - 2d_{py} - d_{mn}. \tag{7.9}$$

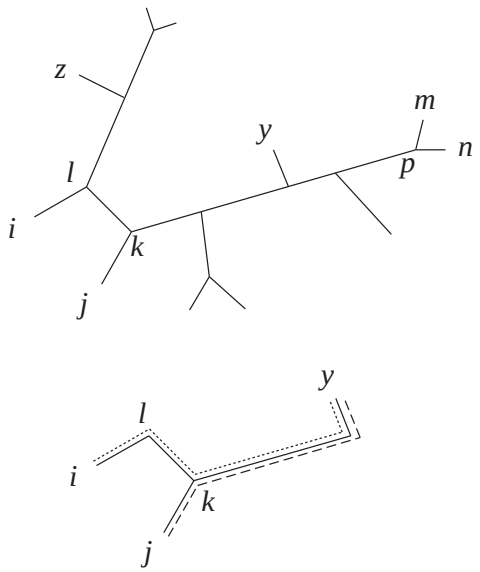


图 7.18: 如果叶节点 i 和 j 不是邻居, 那么它们之间的路径上就至少有两个节点, 这里记为 k 和 l. 从 k 出发但不指向 i 或 j 的分支称为 L_k, 图中用其上的一对邻居 m 和 n 表示. 从 l 出发的分支称为 L_l, 从它又引出一个到达叶节点 z 的分支. 下图: 从 i 到 y (点线) 和从 j 到 y (虚线) 的路径, 给出了 $d_{iy} + d_{jy} = d_{ij} + 2d_{ky}$ 的一个直观证明.

同样, 对 L_l 中的 z, 我们有

$$d_{iz} + d_{jz} - d_{mz} - d_{nz} = d_{ij} - d_{mn} - 2d_{pk} - 2d_{lk}. \tag{7.10}$$

由 D_{ij} 的定义可知,

$$D_{ij} - D_{mn} = d_{ij} - d_{mn} - \frac{1}{N-2}\left(\sum_{\text{all leaves } u} d_{iu} + d_{ju} - d_{mu} - d_{nu}\right),$$

而且容易从式 (7.9) 和式 (7.10) 验证, 对树上所有叶节点 (all leaves) u 求和后, d_{ij} 和 d_{mn} 的系数都是 $(N-2)$ (见练习7.15). 因此可以消去 $d_{ij} - d_{mn}$ 项, 故[8]

$$D_{ij} - D_{mn} = \frac{1}{N-2}\left(\sum_{y \text{ in } L_k} (2d_{py} - 2d_{ky}) + \sum_{z \text{ in } L_l} (2d_{pk} + d_{lk})\right) + C, \tag{7.11}$$

其中 C 是从 i 到 j 之间的路径上, 包括 k 和 l 在内的所有来自其他分支的额外正项之和. 令 $|L_l|$ 和 $|L_k|$ 分别表示 L_l 和 L_k 中节点的个数, 注意到 $d_{py} - d_{ky} > -d_{pk}$ 的事实, 我们有

$$D_{ij} - D_{mn} > 2d_{pk}(|L_l| - |L_k|)/(N-2).$$

我们必定有 $D_{mn} > D_{ij}$, 因为 D_{ij} 是最小值, 所以 $|L_l| < |L_k|$. 但交换节点 l 和 k 也会得到同样论断, 所以也必定有 $|L_k| < |L_l|$. 所以最开始的假设错误, 即 i 和 j 是邻居叶节点.

练习

7.14 假设从 k 出发的分支只有一个叶节点 m (因此在 L_k 中不可能找到一对邻居), 证明在从 i 到 j 的路径上出现除 k 以外的其他节点就意味着 $D_{ij} > D_{jm}$, 这与 D_{ij} 是最小值的假设矛盾.

7.15 证明当 $y = m$ 或 $y = n$ 时, (7.9) 不含 $2d_{py}$ 项. 证明这意味着 $2d_{py}$ 项可以被包含到 (7.11) 中对 L_k 的所有 y 之求和项 $\sum(2d_{py} - 2d_{ky})$ 中, 而当我们从这个和里减去 $2d_{pm} + 2d_{pn}$ 时, 该结论也包括 $y = m$ 和 $y = n$ 的情形. 证明此后 d_{mn} 项能被消去, 并检查 $y = i$ 和 $y = j$ 时的情形.

[8]原文缺下式编号, 中文版已经改正. ——译注

第八章

系统发育的概率论方法

8.1 引言

本章的目标是阐明系统发育 (phylogeny) 的概率论模型, 揭示如何用最大似然法或抽样法从序列集出发推断系统发育树. 我们也会回顾前一章的系统发育方法, 并指明它们通常有概率论解释, 尽管我们通常并不这样描述.

系统发育中概率论方法的概述

基于概率论的系统发育学之基本目标是, 依据树的似然 $P(\text{data}|\text{tree})$ 为树排序, 或者如果更倾向于 Bayes 观点, 则依后验概率 $P(\text{tree}|\text{data})$ 为树排序. 可能还有次要目标, 例如寻找某些特定分类特征的似然或后验概率值, 或者将一组生物体归于单一分支. 为了达到这些目标, 我们必须能够定义并计算给定一棵树后数据的概率 $P(x^\bullet|T,t_\bullet)$. 这里数据是 n 条序列 x^j 的集合, 其中 $j=1\ldots n$, 我们将其简写为 $x^\bullet$. T 是一棵带 n 个叶节点的树, 其中序列 j 在叶节点 j 上, 并且 $t_\bullet$ 是树的边长. 为了定义 $P(x^\bullet|T,t_\bullet)$, 我们还需要一个进化模型, 即沿着树的边使序列发生改变的突变和选择事件之模型.

假设我们将 $P(x|y,t)$ 定义为祖先序列 y 沿着边长 t 进化到序列 x 的概率. 把特定祖先集合分配到树的节点上后, 树 T 的概率可以由所有进化概率的乘积来计算, 树的每条边都对应一个概率. 例如, 图8.1中树的概率为

$$P(x^1,\ldots,x^5|T,t_\bullet)=P(x^1|x^4,t_1)P(x^2|x^4,t_2)P(x^3|x^5,t_3)P(x^4|x^5,t_4)P(x^5),$$

其中 $P(x^5)$ 表示 x^5 位于树根节点的概率. (除了像 Hillis *et al.* [1992] 的实验室进化实验,) 祖先序列通常是未知的, 为了计算给定树上已知序列的概率 $P(x^1,\ldots,x^3|T,t_\bullet)$,

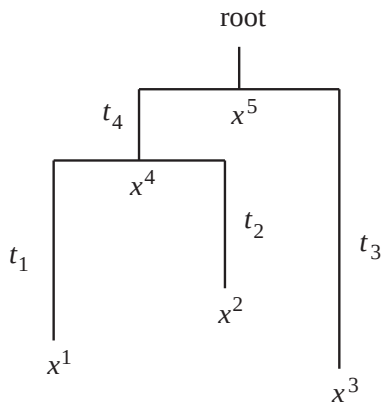


图 8.1: 一棵含三条序列的树.

我们需要在所有可能的祖先x^4、x^5上求和. 这类似于 HMM 中在所有不同的路径上求和以获得观测数据的概率 (见第3章).

给定这个模型后, 我们就可以搜索最大似然树 (maximum likelihood tree), 也就是使 $P(x^\bullet|T,t_\bullet)$ 最大化的拓扑为 T、边长为 $t_\bullet$ 的树. 找到这个最大值需要: (1) 按照指定叶节点处序列的分配顺序, 对拓扑进行搜索; (2) 对每个拓扑, 搜索所有可能的边长 $t_\bullet$.

正如我们已经知道的 (本书第 p. 114 页), 含 n 个叶节点的有根二叉树有 $(2n-3)!!$ 棵, 当序列超过六条时这个数目会变的非常大. 因此我们需要有效的搜索过程(如见本书第 p. 123 页及 Felsenstein [1981a]) 以便实现 (1). 最大化边长似然的部分 (2) 则可以由不同的优化技术完成 (见8.3节).

另外一种策略是, 按后验分布 $P(T,t_\bullet|x^\bullet)$ 抽样而随机地搜索树 (见8.4节). 这是最近的研究成果, 但很有前景.

8.2 进化的概率论模型

对序列 x 经过边长 t 从祖先序列 y 进化而来的概率 $P(x|y,t)$, 我们还没有指定其具体形式. 为此我们需要一个进化模型. 我们知道在进化过程中, 残基会被替换, 成组残基的插入删除也会出现, 并且核酸和蛋白质结构也会施加更为复杂的约束. 后面我们会考虑插入删除的模型, 但开始时我们要作一些最基本的简化假设: 给定数据序列的每一位点均被视为相互独立并且不发生删除和插入. 因此, 我们的序列形成一个无空位联配, 每个位点都独立进化.

令 $P(b|a,t)$ 表示残基 a 在边长 t 上被替换成残基 b 的概率. 我们的假设意味着, 对两条已联配上且无空位的序列 x 和 y, 有 $P(x|y,t)=\prod_u P(x_u|y_u,t)$, 其中 u 为每个位点在序列中的编号.

现在考察残基 a 和 b 之替换概率 $P(b|a,t)$ 的可能形式. 给定一个大小为 K 的残基字母表, 我们能把它们写成依赖 t 的 $K\times K$ 的矩阵, 用 $S(t)$ 表示:

$$S(t)=\begin{pmatrix} P(A_1|A_1,t) & P(A_2|A_1,t) & \dots & P(A_K|A_1,t) \\ P(A_1|A_2,t) & P(A_2|A_2,t) & \dots & P(A_K|A_2,t) \\ \dots & \dots & \dots & \dots \\ P(A_1|A_K,t) & P(A_2|A_K,t) & \dots & P(A_K|A_K,t) \end{pmatrix}.$$

一些重要的替换矩阵族是**可乘的** (multiplicative), 即对所有长度 s 和 t 均有

$$S(t)S(s)=S(t+s). \tag{8.1}$$

这等价于替换概率

$$\sum_b P(a|b,t)P(b|c,s)=P(a|c,s+t)$$

对所有的 a、c、s和 t 均成立. 如果我们采用如下观点, 即把 t 看作一种"时间" 变量[1], 那么可乘性就是替换过程的一个推论, 该替换过程具有 Markov 性和平稳性 (stationary), 而后者意味着 t 时刻的 a 在 s 时刻被替换成 b 的概率仅依赖于时间间隔 $(s-t)$ (练习8.2).

对核酸序列, 存在一种 Jukes & Cantor [1969] . 这里假设替换**速率** (rate) 矩阵 R 的形式为

$$\begin{array}{c} \\ \mathtt{A} \\ \mathtt{C} \\ \mathtt{G} \\ \mathtt{T} \end{array}\!\!\begin{array}{c} \begin{array}{cccc} \mathtt{A} & \mathtt{C} & \mathtt{G} & \mathtt{T} \end{array} \\ \begin{pmatrix} -3\alpha & \alpha & \alpha & \alpha \\ \alpha & -3\alpha & \alpha & \alpha \\ \alpha & \alpha & -3\alpha & \alpha \\ \alpha & \alpha & \alpha & -3\alpha \end{pmatrix} \end{array}, \tag{8.2}$$

[1]例如, t 可能正比于突变速率 × 进化时间. ——原注

这意味着所有核苷酸以相同的速率 α 转移. 短时间的替换矩阵 $S(\varepsilon)$ 可以近似为 $S(\varepsilon) \simeq (I + R\varepsilon)$, 其中, I 为单位矩阵, 其对角元为1, 其他元均为0. 由此可得,

$$I + R\varepsilon = \begin{pmatrix} 1-3\alpha\varepsilon & \alpha\varepsilon & \alpha\varepsilon & \alpha\varepsilon \\ \alpha\varepsilon & 1-3\alpha\varepsilon & \alpha\varepsilon & \alpha\varepsilon \\ \alpha\varepsilon & \alpha\varepsilon & 1-3\alpha\varepsilon & \alpha\varepsilon \\ \alpha\varepsilon & \alpha\varepsilon & \alpha\varepsilon & 1-3\alpha\varepsilon \end{pmatrix}.$$

由矩阵的可乘性可得 $S(t+\varepsilon) = S(t)S(\varepsilon) \simeq S(t)(I + R\varepsilon)$. 我们可以写成 $(S(t+\varepsilon) - S(t))/\varepsilon \simeq S(t)R$, 而当 ε 趋于无限小时, 有 $S't = S(t)R$. 由此, 我们可以推导出时间 t 的替换矩阵. 鉴于速率矩阵的对称性, 将 $S(t)$ 写成以下形式:

$$S(t) = \begin{pmatrix} r_t & s_t & s_t & s_t \\ s_t & r_t & s_t & s_t \\ s_t & s_t & r_t & s_t \\ s_t & s_t & s_t & r_t \end{pmatrix}. \tag{8.3}$$

将上式代入 $S'(t) = S(t)R$, 我们可以得到方程

$$\begin{aligned} \dot{r} &= -3\alpha r + 3\alpha s, \\ \dot{s} &= -\alpha s + \alpha r, \end{aligned}$$

并且很容易检验下面两式是上面两式的解:

$$\begin{aligned} r_t &= \tfrac{1}{4}(1 + 3e^{-4\alpha t}), \\ s_t &= \tfrac{1}{4}(1 - e^{-4\alpha t}). \end{aligned} \tag{8.4}$$

矩阵8.3用参数 r_t 和 s_t 构建了**Jukes-Cantor 模型**. 注意当 $t = \infty$ 时, $r_t = s_t = \frac{1}{4}$. 这也意味着该模型的平衡 (equilibrium) 核酸频率为 $q_{\mathtt{A}} = q_{\mathtt{C}} = q_{\mathtt{G}} = q_{\mathtt{T}} = \frac{1}{4}$.

Jukes-Cantor 模型并未考虑核甘酸替换中的一些重要特点. 例如转换 (transition), 即嘌呤 (purine) 到嘌呤或嘧啶 (pyrimidine) 到嘧啶的替换通常比颠换 (transversion) 更常见, 而颠换改变了核甘酸的类型. [2]为了解决这一问题, Kimura [1980]提出了一个带有如下速率矩阵的模型.

$$\begin{pmatrix} -2\beta-\alpha & \beta & \alpha & \beta \\ \beta & -2\beta-\alpha & \beta & \alpha \\ \alpha & \beta & -2\beta-\alpha & \beta \\ \beta & \alpha & \beta & -2\beta-\alpha \end{pmatrix}. \tag{8.5}$$

采用与推导 Jukes-Cantor 模型相同的方法, 我们可以对矩阵进行积分以获得依赖时间的一般形式:

$$S(t) = \begin{pmatrix} r_t & s_t & u_t & s_t \\ s_t & r_t & s_t & u_t \\ u_t & s_t & r_t & s_t \\ s_t & u_t & s_t & r_t \end{pmatrix}, \tag{8.6}$$

其中

$$\begin{aligned} s_t &= \tfrac{1}{4}(1 - e^{-4\beta t}) \\ u_t &= \tfrac{1}{4}(1 + e^{-4\beta t} - 2e^{-2(\alpha+\beta)t}), \\ r_t &= 1 - 2s_t - u_t. \end{aligned}$$

尽管该模型被广泛使用, 但它仍然与现实有很大差距, 因为其平衡频率相等, 即 $q_{\mathtt{A}} = q_{\mathtt{C}} = q_{\mathtt{G}} = q_{\mathtt{T}} = \frac{1}{4}$, 然而许多物种在AT对GC的比值上表现出很强的偏向性. 针对

[2]转换包括 A ↔ G 和 C ↔ T, 而颠换包括 A ↔ T、G ↔ T、A ↔ C 和 C ↔ G. ——原注

此问题, Hasegawa, Kishino & Yano [1985] 所提出的模型既考虑了核酸频率不相等的特点, 同时也考虑了转换和颠换不一致的问题.

接下来, 请回顾在2.8节用于蛋白质序列的模型, 我们为整数 n 定义的条件概率的 PAM 矩阵为 $S(n) = S(1)^n$, 即 PAM1 的 n 次幂. 我们可将此扩展, 用于 t 的所有取值 (所有的正实数, 而不仅仅是整数), 从而获得一个矩阵, 其形式十分类似于在 Jukes-Cantor 和 Kimura 的 DNA 模型中的矩阵.

为阐明这点, 对 $S(1)$ 进行对角化 (diagonalize). 通常, 该操作对由条件概率组成的形如 $S(1)$ 的"随机矩阵" 都可行. 因此我们可以得到 $S(1) = UD(\lambda_i)U^{-1}$, 其中 U 为坐标变换矩阵而 $D(\lambda_i)$ 是对角元依次为特征值 (eigenvalue) $\lambda_1 \dots \lambda_{20}$ 的对角矩阵. 这些特征值的取值范围是从0到1, 所以可写为 $\lambda_i = \exp(-\mu_i)$. 至此, $S(1)$ 的幂可以用对角矩阵坐标系统中的简单方式来表示; 例如 $S(2) = S(1)S(1) = UD(\lambda_i)U^{-1}UD(\lambda_i)U^{-1} = UD(\lambda_i^2)U^{-1}$, 并且一般地有 $S(t) = UD(\lambda_i^t)U^{-1}$. 于是

$$S(t) = U \begin{pmatrix} e^{-\mu_1 t} & 0 & \dots & 0 \\ 0 & e^{-\mu_2 t} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & e^{-\mu_{20} t} \end{pmatrix} U^{-1}.$$

从这里可以看出, $S(t)$ 的每一项可以表示为一组指数的和: 假设 A_i 表示第 i 个氨基酸, 那么 $P(A_j|A_i, t) = \sum_k u_{ik} \exp(-\mu_k t) v_{kj}$, 其中 u_{ik} 和 v_{kj} 分别是 U 和 U^{-1} 中的元素.

这与 DNA 模型中的速率矩阵有点类似, 我们能容易的看出为何如此. 这是因为假如 Jukes-Cantor 速率矩阵可以被对角化, 那么对某个恰当的坐标变换矩阵 U 有 $R = UD(\lambda_i)U^{-1}$, 当 $S(t) = UT(t)U^{-1}$ 时, 等式 $S'(t) = S(t)R$ 变为 $T'(t) = T(t)D$, 其中 $S(t) = UT(t)U^{-1}$. 当初始情况 $S(0)$ 是对角化矩阵时, 方程 $T'(t) = T(t)D$ 很容易求解; 实际上 $T(t)$ 必须是依次以 $\exp(\lambda_i t)$ 为对角元素的对角阵. 由于特征值是0或 -4α, 所以我们可以很容易的用类似 PAM 矩阵的推导方法获得 Jukes-Cantor 的矩阵项8.4.

令 $t = \infty$, PAM 矩阵可以写成如下形式,

$$\begin{pmatrix} q_{A_1} & q_{A_2} & \dots & q_{A_{20}} \\ q_{A_1} & q_{A_2} & \dots & q_{A_{20}} \\ \dots & \dots & \dots & \dots \\ q_{A_1} & q_{A_2} & \dots & q_{A_{20}} \end{pmatrix}$$

其中 q_{A_i} 为氨基酸的平衡频率, 接近于 Dayhoff, Schwartz & Orcutt [1978] 最初构建其矩阵时所用数据库中的氨基酸频率.

练习

8.1 证明 Jukes-Cantor 和 Kimura 的替换矩阵都是可乘的.

8.2 令 $P(a(t_2)|b(t_1))$ 为 t_1 时刻的残基 b 被 t_2 时刻的残基 a 所替换的概率. 平稳性意味着可将其写为 $P(a|b, t_2 - t_1)$. Markov 性是指如果 $t_0 < t_1$ 那么 $P(a(t_2)|b(t_1)) = P(a(t_2)|b(t_1), c(t_0))$, 即 a 替换 b 的概率不受较早某一时刻 t_0 时残基 c 的影响. 请证明

$$\sum_{b(t)} P(a(s+t)|b(t), c(0)) P(b(t)|c(0)) = P(a(s+t)|c(0)),$$

并推导可乘性8.1成立.

8.3 计算无空位联配的似然

本节我们阐述如何使用前面的模型计算树的似然. 我们首先以两条序列为例, 之后再扩展到 n 条序列的一般情形.

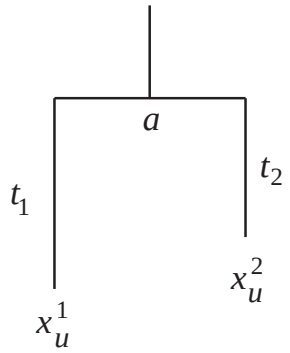


图 8.2: 一棵简单的树.

两条序列的例子

假设我们有两条序列 x^1 和 x^2. 此时只有一棵树, 它有两个分支和一个根节点, 其中根节点表示 x^1 和 x^2 的假设 (hypothetical) 共同祖先 (图8.2). 因此, 我们只需考察似然如何随长度 t_1 和 t_2 变化.

考虑一个位点 u. 此时叶节点1和2上的残基分别为 x_u^1, x_u^2. 我们为根节点分配一个残基 a, 这里使用和叶节点不同的标记是为了强调它是一个变量, 而不能由数据集 x^1 和 x^2 确定:

$$P(x_u^1, x_u^2, a|T, t_1, t_2) = q_a P(x_u^1|a, t_1)P(x_u^2|a, t_2).$$

这是从根节点分布 (假设替换矩阵族为平衡分布) 中得到 a, 并且 a 被 x_u^1 和 x_u^2 所替换的概率. 注意, 我们包括 x_u^1 和 x_u^2 里有一个或两个都为 a 的情况. 由于通常并不知道根节点的残基, 所以我们必须对所有可能的 a 求和以得到 x_u^1 和 x_u^2 的概率. 如下:

$$P(x_u^1, x_u^2|T, t_1, t_2) = \sum_a q_a P(x_u^1|a, t_1)P(x_u^2|a, t_2). \tag{8.7}$$

假如有 N 个位点, 那么全似然公式为

$$P(x^1, x^2|T, t_1, t_2) = \prod_{u=1}^{N} P(x_u^1, x_u^2|T, t_1, t_2). \tag{8.8}$$

例子: 两条核酸序列的似然

假设我们有两条核酸序列, 并且为了简单起见, 令仅有两种核苷酸C和G. 例如, 序列可能是

```
CCGGCCGCGCG
CGGGCCGGCCG
```

那么假设在 Jukes-Cantor 模型下, 序列的似然 $P(x_1, x_2|T, t_1, t_2)$ 是多少?

使用替换概率 (8.4), (8.7) 式给出了C出现在树 T 的两个叶节点上的概率为:

$$P(\mathtt{C}, \mathtt{C}|T, t_1, t_2) = q_{\mathtt{C}} r_{t_1} r_{t_2} + q_{\mathtt{G}} s_{t_1} s_{t_2} + q_{\mathtt{A}} s_{t_1} s_{t_2} + q_{\mathtt{T}} s_{t_1} s_{t_2} = \tfrac{1}{4}(r_{t_1} r_{t_2} + 3 s_{t_1} s_{t_2}).$$

由对称性可得 $P(\mathtt{G}, \mathtt{G}|T, t_1, t_2) = P(\mathtt{C}, \mathtt{C}|T, t_1, t_2)$. 类似地,

$$P(\mathtt{C}, \mathtt{G}|T, t_1, t_2) = P(\mathtt{G}, \mathtt{C}|T, t_1, t_2) = \tfrac{1}{4}(r_{t_1} s_{t_2} + s_{t_1} r_{t_2} + 2 s_{t_1} s_{t_2}).$$

替换 r 和 s 可得

$$P(\mathtt{C}, \mathtt{C}|T, t_1, t_2) = \tfrac{1}{16}(1 + 3e^{-4\alpha(t_1+t_2)}),$$
$$P(\mathtt{C}, \mathtt{G}|T, t_1, t_2) = \tfrac{1}{16}(1 - e^{-4\alpha(t_1+t_2)}).$$

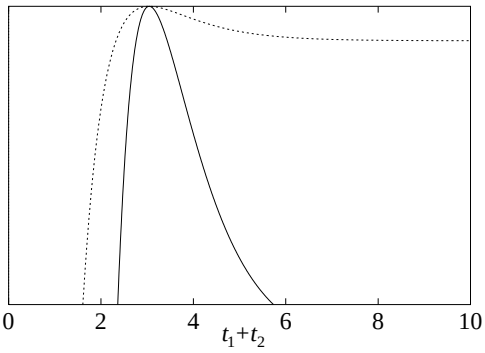


图 8.3: 由 (8.9) 式给出的对数似然 $P(x^1, x^2|T, t_1, t_2)$, 其中虚线取 $n_1 = 100, n_2 = 250$, 而实线取 $n_1 = 1000, n_2 = 2500$. 画实线的曲线更陡, 因为有更多的数据定义了这个最大似然的峰. 为了使它们的峰重合, 这些曲线均已经过平移.

现在假设两条序列间有 n_1 个位点残基相同, n_2 个位点发生了替换. 此时 (8.8) 式为

$$P(x^1, x^2|T, t_1, t_2) = \frac{1}{16^{n_1+n_2}}(1 + 3e^{-4\alpha(t_1+t_2)})^{n_1}(1 - e^{-4\alpha(t_1+t_2)})^{n_2}. \qquad (8.9)$$

注意该似然仅仅依赖于 t_1 和 t_2 的和. 这是因为 Jukes-Cantor 的替换过程是时间对称 (time-symmetrical) 的, 所以没有可用的信息来确定根节点的位置. 当 $t_1 + t_2$ 保持恒定时, 即使滑动根节点该似然仍不改变. 对于根节点的这种不确定性, 我们将在本书第 p. 141 页中进行更加详尽地讨论. 图8.3示例了该似然(图中是对数似然的形式) 如何随 $t_1 + t_2$ 变化. □

任意条序列的似然

现在, 我们能将这些计算扩展到 n 条序列的情形. 假设我们有一棵边长为 $t_\bullet$ 的树 T. 令 $\alpha(i)$ 为 i 的直接祖先节点, 即 i 向上分支顶端的节点. 像通常一样, 令 $x_u^1 \dots x_u^n$ 表示 n 条序列 $x^1, \dots, x^n$ 在第 u 个位点处的残基. 在树 T 的 n 个叶节点处产生这些残基的概率 $P(x_u^1 \dots x_u^n|T, t_\bullet)$ 可以通过相乘该树所有分支的替换概率得到 .从而

$$P(x_u^1 \dots x_u^n|T, t_\bullet) = \sum_{a^{n+1}, a^{n+2}, \dots, a^{2n-1}} q_{a^{2n-1}} \prod_{i=n+1}^{2n-2} P(a^i|a^{\alpha(i)}, t_i) \prod_{i=1}^{n} P(x_u^i|a^{\alpha(i)}, t_i), \qquad (8.10)$$

其中该连加号对非叶节点 k (节点编号从 $n + 1$ 到 $2n - 1$) 的所有可能残基分配 a^k 求和.

我们可以后序遍历该树, 从叶节点开始向上逐步算出这个概率 [Felsenstein 1981a]. 令 $P(L_k|a)$ 表示节点 k 以下的所有叶节点概率, 假设 k 处的残基为 a. 此时, 我们可以从所有 b 和 c 的概率 $P(L_i|b)$ 和 $P(L_j|c)$ 出发, 计算 $P(L_k|a)$, 其中 i 和 j 是 k 的子节点(图8.4):

算法: Felsenstein 的似然算法 起始:

令 $k = 2n - 1$.

迭带: 对所有 a 计算 $P(L_k|a)$ 如下:

如果 k 是叶节点则:

$a = x_u^k$ 时令 $P(L_k|a) = 1$,

$a \neq x_u^k$ 时令 $P(L_k|a) = 0$.

如果 k 不是叶节点则:

对所有子节点 i 和 j 上的 a 计算 $P(L_i|a)$ 和 $P(L_j|a)$,

并令 $P(L_k|a) = \sum_{b,c} P(b|a, t_i)P(c|a, t_j)P(L_j|c)$.

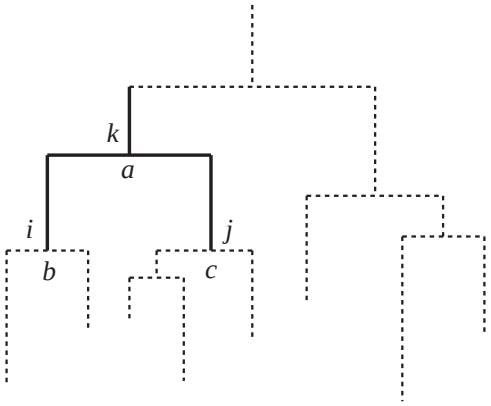


图 8.4: 标记树的分支.

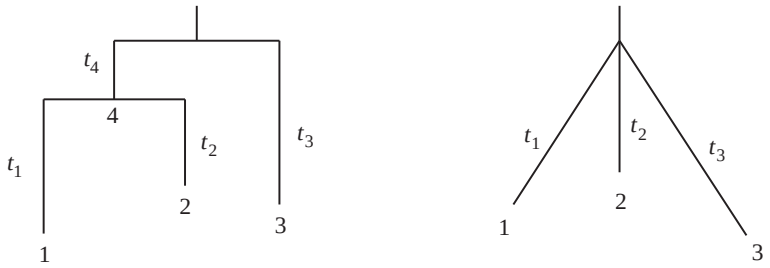


图 8.5: 带三个叶节点的树(左); 以及为了计算似然, 由它简化出的三叉树(右).

终止:

位点 u 的似然 $= P(x_u^\bullet|T,t_\bullet) = \sum_a P(L_{2n-1}|a)q_a$. ◁

请注意上述算法类似于加权简约算法(本书第 p. 120页), 我们将在本书第 p. 154页中进一步阐述.

计算似然的最后一步使用由位点间的独立性假设所导出的

$$P(x^\bullet|T,t_\bullet) = \prod_{u=1}^{N} P(x_u^\bullet|T,t_\bullet). \tag{8.11}$$

例子: 三条核酸序列的树

现在我们将本书第 p. **??**页的例子扩展至带三个叶节点的进化树(图8.5, 左图). 数据为三条仅含C和G的核酸序列, 如:

```
CCGGCCGCGCG
CGGGCCGGCCG
GCCGCCGGGCC
```

我们使用 Jukes-Cantor 模型计算似然. 正如之前一样, 我们独立考察分配了不同残基的位点. 考虑C出现在所有叶节点的情形. 于是我们有

$$\begin{aligned}
P(\mathtt{C},\mathtt{C},\mathtt{C}|T,t_1,t_2,t_3) &= q_{\mathtt{C}} r_{t_3}\left(r_{t_4 r_{t_1}} r_{t_2} + 3 s_{t_4} s_{t_1 s_{t_2}}\right) \\
&\quad + (q_{\mathtt{A}} + q_{\mathtt{G}} + q_{\mathtt{T}}) s_{t_3}\left(r_{t_4} s_{t_1} s_{t_2} + 2 s_{t_4} s_{t_1} s_{t_2} + s_{t_4} r_{t_1} r_{t_2}\right) \\
&= \tfrac{1}{4} r_{t_1} r_{t_2}(r_{t_3} r_{t_4} + 3 s_{t_3} s_{t_4}) \\
&\quad + \tfrac{3}{4} s_{t_1} s_{t_2}(2 s_{t_3} s t_4 + s_{t_3} r_{t_4} + s_{t_4} r_{t_3}) \\
&= \tfrac{1}{4}\left(r_{t_1 r_{t_2} r_{t_3+t_4}} + 3 s_{t_1} s_{t_2} s_{t_3+t_4}\right),
\end{aligned}$$

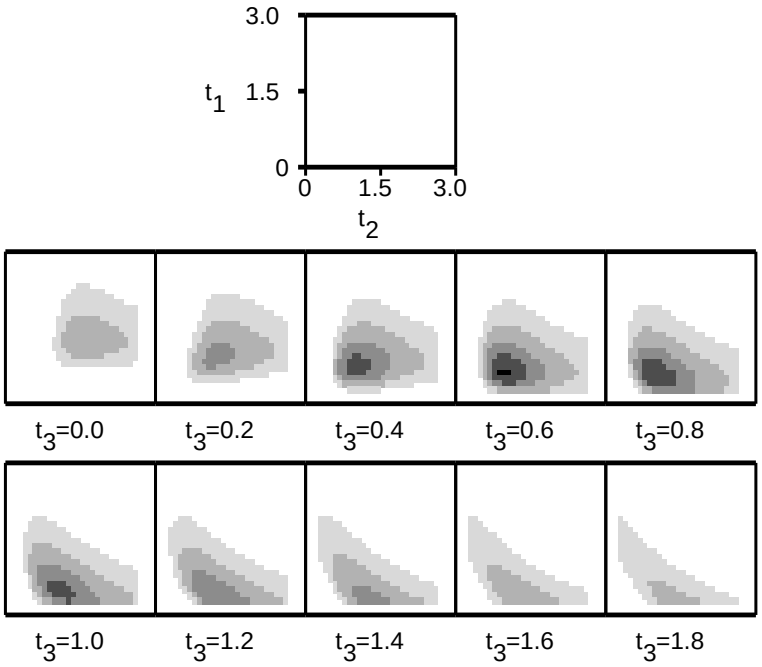


图 8.6: 公式 (8.12) 给出的似然函数, 令 $n_1 = 10, n_2 = 20, n_3 = 15, n_4 = 17$. 白色和五个灰度级别分别表示由0, 0.001, 0.01, 0.07, 0.3, 0.9和1分成的区间上的似然. 每个方块显示了 t_3 取特定值(见方框下面) 时的似然, 方块的两个坐标轴代表 t_1 和 t_2, 它们的取值范围显示在图顶部的方块中.

其中第一个等号只计算了8.10中的项, 并在根节点以平衡概率开始, 第二个等号重新组合化简, 而第三个等号能够依据 Jukes-Cantor 矩阵的可乘性得到(练习8.3). 我们再次发现, 与根节点相连的边长, 即此处的 t_3 和 t_4, 仅以它们和的形式出现. 这适用于所有叶节点上的值, 不仅适用于"C , C , C" ; 因此对整体的似然同样适用, 即允许我们将根节点换至节点4, 从而产生一棵三叉树(图8.5, 右图). 为简单起见我们将第三个边长用 t_3 表示(而非 $t_3 + t_4$), 从而现在就可以对根的所有残基分配与三个边长的概率乘积求和, 以便计算其似然:

$$P(x_u^1, x_u^2, x_u^3|T, t_1, t_2, t_3) = \sum_a q_a P(x_u^1|a, t_1) P(x_u^2|a, t_2)) P(x_u^3|a, t_3).$$

上式右端存在四种可能的项, 即所有残基相同或其中一个与另外两个不同; 例如:

$$P(\mathtt{C}, \mathtt{C}, \mathtt{C}|T, t_1, t_2, t_3) = \tfrac{1}{4}(r_{t_1} r_{t_2} r_{t_3} + s_{t_1} s_{t_2} s_{t_3})$$
$$P(\mathtt{C}, \mathtt{C}, \mathtt{G}|T, t_1, t_2, t_3) = \tfrac{1}{4}(r_{t_1} r_{t_2} s_{t_3} + s_{t_1} s_{t_2} r_{t_3} + 2 s_{t_1} s_{t_2} s_{t_3}).$$

如果序列间残基完全相同的位点有 n_1 个, CCG或GGC类型的位点有 n_2 个, CGC或GCG类型的位点有 n_3 个, GCC或CGG类型的位点有 n_4 个, 那么通过对称性可得

$$P(x^1, x^2|T, t_1, t_2) = 4^{-3(n_1+n_2+n_3+n_4)} a(t_1, t_2, t_3)^{n_1} b(t_1, t_2, t_3)^{n_2} \times b(t_1, t_3, t_2)^{n_3} b(t_3, t_2, t_1)^{n_4}, \quad (8.12)$$

其中 $a(t_1, t_2, t_3)$ 和 $b(t_1, t_2, t_3)$ 为指数项的和 (见练习8.4). 该似然函数的具体描述见图8.6. □

练习

8.3 证明 $r_{t_3} r_{t_4} + 3 s_{t_3} s_{t_4}$ 和 $2 s_{t_3} s_{t_4} + s_{t_3} r_{t_4} + s_{t_4} r_{t_3}$ 是从时间为 t_3 和 t_4 的 Jukes-Cantor 矩阵之乘积中产生的项, 并推导它们可以分别写为 $r_{t_3+t_4}$ 和 $s_{t_3+t_4}$.

8.4 证明 $a(t_1,t_2,t_3)$ 和 $b(t_1,t_2,t_3)$ 可分别写为

$$a(t_1,t_2,t_3)=1+3\mathrm{e}^{-4\alpha(t_1+t_2)}+3\mathrm{e}^{-4\alpha(t_1+t_3)}+3\mathrm{e}^{-4\alpha(t_2+t_3)}+6\mathrm{e}^{-4\alpha(t_1+t_2+t_3)}$$

和

$$b(t_1,t_2,t_3)=1+3\mathrm{e}^{-4\alpha(t_1+t_2)}-\mathrm{e}^{-4\alpha(t_1+t_3)}-\mathrm{e}^{-4\alpha(t_2+t_3)}-2\mathrm{e}^{-4\alpha(t_1+t_2+t_3)}.$$

根节点位置的可逆性与独立性

在简约法中, 我们只需搜索无根树的拓扑. 尽管很难明显地观察到似然与根节点位置间的独立性, 但在一定合理的假设下事实的确如此. 事实上我们只需要两个假设: 其一是替换矩阵是可乘的 (8.1), 正如此前所述, Jukes-Cantor 矩阵和 PAM 矩阵就满足此性质; 其二是替换矩阵是可逆的, 即对于所有的 a、b 和 t, 有

$$P(b|a,t)q_a=P(a|b,t)q_b. \tag{8.13}$$

从对称性可以看出 Jukes-Cantor 替换矩阵和 PAM 矩阵均满足可逆性. PAM 矩阵的可逆性可从这样的事实导出: 在计数时, 我们丢弃了有关进化时间方向的信息, 即祖先残基 a 被替换为后代残基 b 的事件与其相反的替换过程被视为等价(见2.8节以及下述关于 PAM 矩阵族的例子).

为证明可乘性与可逆性意味着根节点在所有位置上的似然相等, 我们假设在根节点 $2n-1$ 以下存在两个节点 i,j. 从 Felsenstein 算法中用 $P(L_i|\bullet)$ 和 $P(L_j|\bullet)$ 定义 $P(L_{2n-1}|\bullet)$ 可知, 序列 $x^\bullet$ 在位点 u 的似然

$$P(x_u^\bullet|T,t_\bullet)=\sum_a q_aP(L_{2n-1}|a)=\sum_{b,c,a}q_aP(b|a,t_i)P(c|a,t_j)P(L_i|b)P(L_j|c),$$

于是根据可逆性, 有

$$P(x_u^\bullet|T,t_\bullet)=\sum_{b,c}\left(\sum_a P(c|a,t_j)P(a|b,t_i)\right)q_bP(L_i|b)P(L_j|c).$$

通过矩阵的可乘性, 我们可以将内部的求和写为 $\sum_a P(c|a,t_j)P(a|b,t_i)=P(c|b,t_i+t_j)$, 此即 P 的概率独立于节点 $2n-1$ 上分配的符号 a, 而仅依赖于根节点以下两分枝的边长和. 因此根节点可以自由移动于 i 和 j 之间, 并且可以移至树中的任何地方. 这就是 Felsenstein [1981a] 的"滑轮原理" (pulley principle). 它意味着, 当使用具有可乘性和可逆性的矩阵族时, 我们只需在无根树上进行最优树的搜索.

例子: PAM 矩阵族的可逆性

如2.8节所述, 用于构建 PAM 矩阵族的矩阵 A 为对称矩阵, 即对所有的 a 和 b 满足 $A_{ab}=A_{ba}$. 由于 $p_a=\sum_b A_{ab}/\sum_{cd}A_{cd}$, 归一化后的矩阵 B 满足

$$p_aB_{ab}=A_{ab}/\sum_{cd}A_{cd}=A_{ba}/\sum_{cd}A_{cd}=p_bB_{ba}.$$

因为 B 经过比例变换后得到 $S(1)$, 所以 $t=1$ 时满足可逆性. 为证明对任意 n, $S(n)$ 均可逆, 不妨假设对所有小于 n 的数 k 都它满足可逆性, 那么根据 $n-1$ 和 1 时的可逆性, 有:

$$\begin{aligned} p_aP_n(b|a) &= \sum_c p_aP_{n-1}(c|a)P_1(b|c) &&= \sum_c P_{n-1}(a|c)p_cP_1(b|c)\\ &= \sum_c P_{n-1}(a|c)P_1(c|b)p_b &&= P_n(a|b)p_b. \end{aligned}$$

□

例子: 非可逆的矩阵族

非可逆的矩阵具有怎样的形式呢? 假设对两残基A和B而言, A → B 的替换多于 B → A 的替换. 为保持残基频率不变, 其他残基间的替换将处于平衡状态. 最简单的情况是字母表里只有三种残基且残基间存在循环(cyclic) 替换模式, 则瞬时替换矩阵为

$$\begin{pmatrix} -\alpha & \alpha & 0 \\ 0 & -\alpha & \alpha \\ \alpha & 0 & -\alpha \end{pmatrix}. \tag{8.14}$$

这将导出一个依赖于时间 t 的矩阵族

$$S(t) = \begin{pmatrix} r_t & s_t & u_t \\ u_t & r_t & s_t \\ s_t & u_t & r_t \end{pmatrix},$$

其中

$$\begin{aligned} r_t &= \tfrac{1}{3}\left(1 + 2\mathrm{e}^{-3\alpha t/2}\cos(\sqrt{3}\alpha t/2)\right), \\ s_t &= \tfrac{1}{3}\left(1 - \mathrm{e}^{-3\alpha t/2}\cos(\sqrt{3}\alpha t/2) + \sqrt{3}\mathrm{e}^{-3\alpha t/2}\sin(\sqrt{3}\alpha t/2)\right), \\ u_t &= 1 - r_t - s_t. \end{aligned}$$

□

练习

8.5 证明上述矩阵族具有可乘性, 它带有正项, 并且 $t = 0$ 时的替换率由 (8.14) 式给出. 计算上述矩阵极限的分布, 并证明当 $t > 0$ 时它不满足可逆性8.13.

8.6 我们已证明可逆性可使根节点移至树中的任意位置. 那么当根节点移到叶节点时将会发生什么情况呢?

8.4 用似然做推断

现在, 我们开始介绍概率论系统发育学的核心内容. 我们已经用公式阐述了进化模型, 并且为了计算来自这个模型的似然我们也定义了一个算法, 现在我们需将这套方法应用于数据集上以便推断出其系统发育学性质. 现在我们将概述概率论推断方法, 并首先介绍其中最为经典并广泛使用的最大似然法 (maximum likelihood). 我们也可使用概率论方法评估概率论模型以及所设计出的变量之质量, 但对于此问题的讨论将放在几个精巧设计的模型之后 (见8.5节).

最大化似然

有一种候选的"最优" 树是使似然最大化的进化树. 回顾搜索进化树的方法, 它对于任一拓扑 T 寻找使似然 $P(x_u^\bullet|T, t_\bullet)$ 达到最大的边长 $t_\bullet$. 给出总体最大似然的拓扑与边长即为所求的最优树.

对于少量序列, 例如二至五条序列, 我们可以很容易地枚举出所有的树. 对于每棵树, 我们都可以明确地将似然写为边长的函数, 并通过适当的数值技术使似然最大. 从本质上说这正是 Kishino, Miyata & Hasegawa [1990] 使用牛顿最优化方法所做的工作 [Press *et al.* 1992]. 他们打算将该方法用于蛋白序列的最大似然系统发育研究, 而且他们使用了 PAM 矩阵.

对于大批量的序列, 可以使用 Felsenstein 算法 (本书第 p. 138页) 计算其似然. 针对这种情况, Felsenstein [1981a] 还为寻找最优边长提出了一种 EM 算法. 或者, 我们可以使用标准的优化器 (optimizer), 例如共轭梯度 (conjugate gradient) [Press *et al.* 1992], 虽然它需要似然对边长的导数, 但我们可直接计算这些导数: 将 (8.10) 式中所有出现的 $P(y^k|y^{\alpha(k)}, t_k)$ 替换为其导数 $\partial P(y^k|y^{\alpha(k)}, t_k)/\partial t_k$ 即可.

即便使用最好的优化器, 最大似然方法的计算量还是很大, 而且对于蛋白序列更为如此, 因为其主要的核心计算部分使用一个 20×20 的替换矩阵, 而不是 4×4 的矩阵. 我们需要另一种策略以应对大数据集上的问题; 有一种办法是抽样法.

练习

8.7 在某些简单的情况下, 我们可以计算得到最大似然的边长. 在给定两条核酸序列(本书第 p. **??**页) 时, 证明最大似然可由下式计算

$$t_1 + t_2 = -\tfrac{3}{4} \ln \frac{3n_1 - n_2}{3n_1 + 3n_2}.$$

从后验分布中抽样

如前所述, 最大似然法的计算量非常大. 而且我们不清楚它是否是最优的策略. 如果已知先验 $P(T, t_\bullet)$, 那么我们就可以依据 Bayes 方法计算其后验概率 $P(T, t_\bullet|x^\bullet)$, 即

$$P(T, t_\bullet|x^\bullet) = \frac{P(x^\bullet|T, t_\bullet)P(T, t_\bullet)}{P(x^\bullet)}.$$

后验提供了我们想要的信息, 即在给定该数据的情况下, 每个系统发育模型的概率是多大.

已经有几位学者将 Bayes 方法应用于小数据集上, 但事实上我们可以很容易地枚举这种小数据集上的所有树拓扑(例如 Rannala & Yang [1996]). 最近, Mau, Newton & Larget [1996] 用 Metropolis 算法在树和边长的空间中对后验分布进行抽样, 实现了对大序列集合的处理.

从树空间里抽样就是按某种分布 (此处为树的后验分布) 中的概率随机地抽取树. 如果样本量很大, 那么在大样本的极限情况下, 树的某种性质出现在样本中的频率将会收敛于模型中该性质的后验概率. 例如, 假设一个特定的树拓扑以比例 f 出现在样本里, 那么 f 就是此拓扑后验概率的一个估计. 我们可以对样本中满足相关条件的情形之比例进行计数, 以便确定出某个给定组是单源 (monophyletic) 的概率, 或某个分支点存在于其他两个节点间的概率. 此类问题并不能简单地依靠似然方法解决, 因为它们需要对变量求积分, 而且似然并非概率分布(见本书第 p. 213页).

Mau *et al* .采用了一种特殊的抽样方法, 即 Metropolis 算法, 其抽样过程是产生一条树的序列, 其中每棵树都从前一棵而来. 该算法假设存在一种机制, 在给定一棵树的条件下通过从**建议** (proposal) 分布中抽样而随机生成另一棵树. 令 $P_1 = P(T, t_\bullet|x^\bullet)$ 为当前树的后验概率, 且令 $P_2 = P(\tilde{T}, \tilde{t}_\bullet|x^\bullet)$ 为被建议生成的新树之后验概率, 则 Metropolis 规则是: 当 $P_2 \geq P_1$ 时, 接受新树作为树序列中的下一项; 而当 $P_2 < P_1$ 时, 以概率 P_2/P_1 接受新树为序列中的下一项. 否则最初的树构成下一个样本(T 和 $t_\bullet$ 的这种重复如果发生的话, 就是该过程的一个重要部分, 因为我们需要对具有特定性质的样本进行计数). [3]

只要建议分布是对称的, 即从 $T, t_\bullet$ 建议 $\tilde{T}, \tilde{t}_\bullet$ 的概率与从 $\tilde{T}, \tilde{t}_\bullet$ 建议 $T, t_\bullet$ 的概率相同(见11.4节), 那么该过程就保证能从后验分布中正确地抽样.

练习

8.8 考察由概率分别为 $P(T)$ 和 $P(\tilde{T})$ 的两棵树 T 和 $\tilde{T}$ 组成的简化系统发育空间. 如果建议过程总是选择与当前树不同的另一棵树, 那么请证明 Metropolis 算法会产生一条序列, 其中 T 和 $\tilde{T}$ 的频率都分别收敛于其概率.

系统发育树的建议分布

为了使 Metropolis 算法正确运行, 建议分布的选择极其重要. 如果只是从包含所有树的空间中随机选取建议树, 那么其后验概率通常会很小, 并且某些重复过程会被浪费掉. 在另一方面, 假如建议树过于接近当前树, 那么我们就需要重复很多步骤以充分探索系统发育空间. 技巧就在于寻找出一种给出建议树的方法, 以便这些建议树是当前树的较优 (promising) 变体.

Mau *et al* .提出了由两部分组成的建议机制: 一部分调节边长, 它可以使拓扑以一种有趣的方式发生转换; 而另一部分则将序列重新分配给叶节点. 其中第一部分采用所谓**遍历 profile** (traversal profile) 的树之表示法. 它是一个完全等价于最初

[3] 也需要注意, Metropolis 规则仅使用了 P_1 和 P_2 的比值, 这是幸运的, 因为在 Bayes 公式中分母只能通过在整个树空间上积分得到, 而且它通常是一个未知因子. ——原注

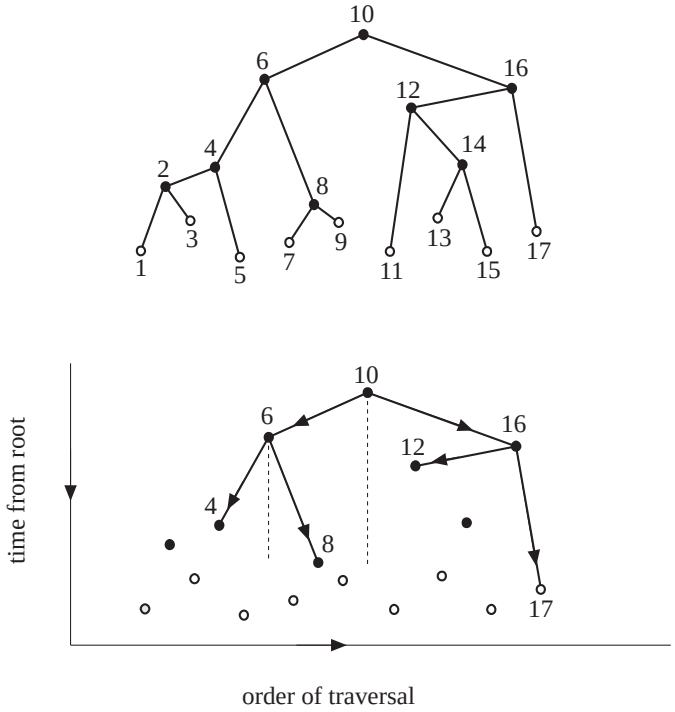


图 8.7: 上图: 以遍历 profile 顺序为节点编号的树. 下图: 从遍历 profile 重建树.

树的图, 但允许对拓扑进行更方便的操作. 在遍历 profile 中, [4]节点的摆放高度对应于它到根节点的边长总和. 节点在水平上被规则地隔开, 并按照遍历该树时每个节点出现的次序排列. 遍历的定义如下: 从最左边的叶节点开始, 我们从左至右以深度优先 (depth first) 遍历这棵树, 并当我们第一次到达某节点时为其编号. 这就可以保证对任一编号为 k 的节点, 所有左边的子节点编号都小于 k, 而所有右边的子节点编号都大于 k. 图8.7中的上图表现了如何采用遍历 profile 方法对树中所有节点进行编号.

给定一个遍历 profile, 我们可以按照如图8.7所描述的过程重建树. 根节点放在树中最高的节点位置(图中的节点10).然后向根节点左、右两边的最高节点 (图中的节点6和16) 画边. 假设我们已经到达节点 k, 那么节点 k 的子节点必须位于某水平延伸范围内, 该范围以所有高于 k 的节点为边界; 与此前一样, 在该延伸范围内, 向最高节点的左右两子节点画边. 因此在图中, 节点6右边的子节点所在区域由垂直的虚线分隔开. 当到达叶节点时, 此过程终止(图中用空心圆圈表示叶节点).

Mau *et al* .之建议过程的一部分是: 对当前树采用遍历 profile 并对节点位置进行上下改变, 而这些改变的量从具有特定边界的均匀分布选出. 只要节点的相对高度发生变化, 新的拓扑就会产生(图8.8). 然而此过程永远都不允许遍历顺序不相邻的节点成为邻居(但是请见练习8.10). 因此他们给出了实现此过程的一种补充建议机制. 在每个节点, 该机制通过随机改变分支方向对叶节点进行重排. 此方法没有改变后验概率值(因此总是被接受), 但却会引入树空间的一个新区域. 调整遍历 profile 的高度当然会引起后验的变化, 但即使拓扑发生改变, 这种变化也随调整规模发生连续的改变(练习8.9). 在此方面, 建议机制比分支交换 (branch-swopping) 更具优势. 分支交换是一种直观明了的修改树的方法, 但它的缺点是有可能使后验概率发生很大改变.

为定义后验, 我们必须从树中选取先验. 由于几乎没有关于进化树分布的可靠信息, Mau *et al* .假设了一个平坦的 (flat) 先验: 对任意的树拓扑, 为 n 条序列的

[4]Mau, Newton & Larget [1996] 用固定斜率的线段连接节点, 而不是像这里我们所表现的那样, 在用相等间隔把节点水平地分开. ——原注

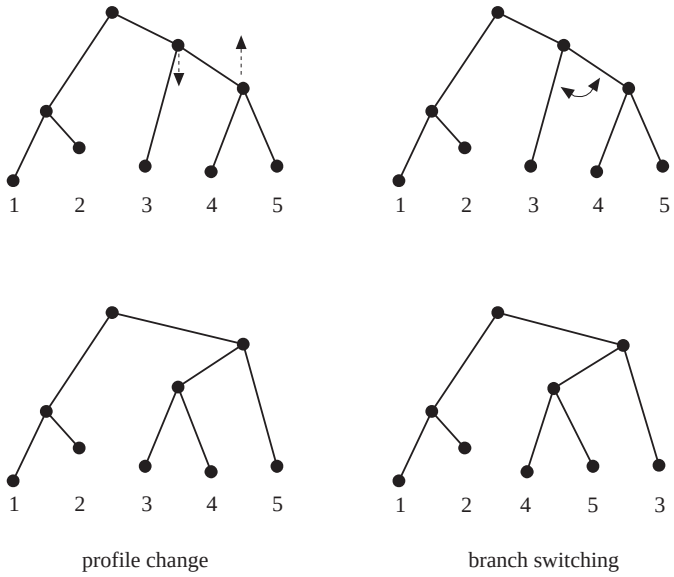


图 8.8: 建议机制的两部分: 改变 profile 中节点的高度(左) 以及通过交换分支重排叶节点(右). 如图所示, 前者可以造成拓扑上的改变. 而后者则不会; 它仅重排已有的拓扑. 但是, 先改变叶节点顺序然后再应用第一种步骤也会产生新的拓扑.

所有边长集 $t_1, \ldots, t_{2n-1}$ 分配相同的概率(注意这并不意味着所有的树拓扑都有相等的先验概率; 见练习8.11). 为保证概率分布可以被归一化, 他们为从根节点到任意叶节点的总边长强制设定了上界. [5] 他们发现对大到包含32条序列的数据集, 他们能可重复地 (reproducibly) 识别概率最大的拓扑. 当满足分子钟假设, 即当遍历 profile 中的所有叶节点都在同一高度时, 这种方法看上去表现得最好.

练习

8.9 考虑图8.8中左边两图所示的 profile 改变. 假设上面一图中带箭头的两个节点高度分别为 h_1 和 h_2, 然后交换其高度, 使之分别为 h_2 和 h_1. 证明当 $h_1 - h_2$ 趋于零时, 此交换所造成似然改变也趋于零.

8.10 证明遍历 profile 中最远端的两个叶节点能够成为邻居, 但其他不相邻的一对节点都不能成为邻居.

8.11 边长的平坦先验为任意拓扑都分配了一个先验, 它是对该拓扑的所有可能边长求积分得到的. 受 Mau *et al* . 的方法所启发, 如果为从根节点到任意叶节点的总边长强制设定一个边界, 那么该积分存在; 称此上界为 B. 当存在分子钟时, 请证明对带四个叶节点并且拓扑为((01)(23))的树, 从积分得到的先验概率为 $B^3/3$; 请证明对拓扑结构为((0(12))3)的树, 此积分为 $B^3/6$. (提示: 定义每棵树中从三个分支节点到现在的时间, 然后对这三个变量积分.) 这便证明了不同的拓扑可以有不同的先验. 然而请证明, 假如将相对于当前时间的分枝节点时间之特定顺序(假设分子钟) 定义为**有标历史** (labelled history), 那么所有带四个叶节点的有标历史均有相同的先验概率. 将此结论扩展为 n 个叶节点的情形.

抽样法在系统发育学中的其他应用

抽样法不仅已经用于物种或基因的系统发育, 而且也已经用于从当前的个体集合出发推断群体历史. Kuhner, Yamato & Felsenstein [1995] 使用抽样法挑选可能的

[5] 在先验中为边长所设的限制看起来可能是一个人为的约束, 但它确实没有太大的效果, 因为通常来讲带极长边的树之似然都很小. 这是因为长边上的替换概率倾向于平衡频率 q_a; 因此所有与其他序列间的关联就都被忽略了. ——原注

树 T, 以便将当前个体集合中的个体联系起来. 现在, 树的先验依赖于群体大小 θ. 直观地说, 这是因为群体越大, 寻找任意两个体的共同祖先时我们就需要向回走越远的距离. 因此对每棵树 T, 我们都可以用先验 $P(T|\theta)$ 作为似然以便估计 θ. 于是抽样过程就允许我们积累正比于 $\int P(\text{data}|T)$ 的似然数据, 在大量样本的极限情况下它将给出我们所求的似然函数 $\int P(\text{data}|T)P(T|\theta)dT = P(\text{data}|\theta)$.

Kuhner, Yamato & Felsenstein [1995] 所使用的建议机制与 Mau, Newton & Larget [1996] 的方法紧密相连. 与在遍历profile 中调整所有节点的高度不同, Kuhner *et al* .调整**两个**节点的相对高度, 并且允许重新标记其子节点. 因此这就是Mau *et al* .方法中两步骤的局部形式. 确定哪种机制能更有效地抽样是很有趣的.

先验 $P(T|\theta)$ 似乎难于计算, 因为它需要对群体中的所有树求和, 而该群体可以为现存个体集合提供可能的系统发育关系. 然而存在一种评估 $P(T|\theta)$ 的极为简单的方法, 其思想基础是回溯时间并且让树枝发生**融合**(coalesce) [Kingman 1982a; 1982b; Hudson 1990]. 对于一个固定的大群体而言, 在时间上融合的概率密度为 $2/\theta$. 想像一条水平线从叶节点开始沿着树 T 升起. 于是对每一次融合, 边长的个数都将减少1. 假设从 k 条边减少为 $k-1$ 条边所用的融合时间为 τ_k. 那么在时间间隔 dt, $k(k-1)/2$ 对边中的一对间发生融合的概率为 $k(k-1)dt/\theta$, 从而可得 $(2/\theta)\exp(-\tau_k k(k-1)/\theta)$ 即为融合出现在时间间隔 θ_k 之末尾而非之前的概率. 对于所有的时间间隔 θ_k 求连乘积, 可得该树的总概率

$$P(T|\theta) = \left(\frac{2}{\theta}^{n-1}\right) \exp\left(-\sum_{k=2}^{k=n} \frac{k(k-1)\tau_k}{\theta}\right).$$

有一种从简单进化模型得到的先验与融合密切相关, 在此模型中我们认为一棵树是由一系列分裂 (splitting) 事件形成的. 假如单个分裂事件出现在一条正在增长的边上之概率密度是一个常数, 比如 λ, 那么我们就说该分裂事件服从**Yule 过程**(Yule process). 这样所得的树之先验就有简单形式, 它正比于 $\exp(-\lambda\sum t_i)$, 其中 t_i 是边长(见练习8.12). 这不同于融合先验, 因为它假设根序列的所有后代都出现在叶节点上, 且不存在省略 (omission) 或灭绝 (extinction) 的情况, 而与之相对反, 融合先验将所有物种或基因都视为随机地从一个大储备池 (pool) 中选出. 哪一种先验更为合适取决于分类学家是正在关注一个关系相近的小家族, 还是一个覆盖广阔区域 (wide-ranging) 的选择.

练习

8.12 按照 Yule 过程, 时间间隔0到 t 内不发生分裂事件的概率密度由当 $\delta t \to 0$ 时 $(1-\lambda\delta t)^{t/\delta t}$ 的极限, 即 $\exp(-\lambda t)$ 给出. 请推导: 带 n 个叶节点的树之 Yule 先验正比于 $\exp(-\lambda\sum t_i)$, 其中 t_i 都是边长. 按照同练习8.11中相同的推导过程, 证明在 Yule 先验下, 四个叶节点上的所有有标历史的先验都相等. 将上述结论扩展至 n 个叶节点的情形.

8.13 假设存在一个分子钟, 分别在分裂率为 λ 的 Yule 先验以及群体大小为 θ 的融合先验下, 计算带两个、三个和四个叶节点的有根树之所有边的期望长度. (提示: 考虑三叶情形. 令两个短边的长度均为 s, 长边的长度为 t. 则该树总边长为 $2t+s$, 于是 Yule 过程下该树的概率正比于 $\exp(-\lambda(2t+s))$. 证明该树的融合概率正比于 $\exp(-(2t+4s)/\theta)$. 现在用标准方法, 也就是对所有 $0 \le t \le \infty$ 以及 $0 \le s \le t$ 求积分, 计算 s 和 t 在这些分布下的均值 (mean).)

再现自举法

本书第 p. 124页中讲到的自举法可以应用于最大似然, 就如它可以应用于其他建树方法一样. 先从真实数据集中随机有放回地选出列以便产生人工数据, 然后我们为人工数据集找出最大似然树. 此过程的多次重复中, 某种特性的出现频率可以作为我们用最大似然法推断该特性的置信度之度量.

因此我们就可以获得类似于从后验中抽样得到的信息, 而事实上两种方法是相关的: 对某些系统发育学模型而言, 当假设树上的平坦先验时, 某个特性的自举置信

度近似于该特性的后验概率. 为了获得该结论的直观感觉, 请考察一种简单情形: 基于一组数据用最大似然法估计投掷一次硬币时得到正面的概率.

例子: 用自举法处理硬币投掷实验的结果

投掷硬币 N 次, 获得 m 次正面 (head , H) 和 n 次背面 (tail , T). 假设一个平坦的先验, 则正面概率 p 的后验分布可由 Dirichlet 分布

$$P(p|m\mathrm{H}, n\mathrm{T}) = p^m(1-p)^n \frac{(N+1)!}{m!n!} \tag{8.15}$$

给出. 自举法的试验始于从数据中选出一组共 N 次投掷硬币的结果, 其中 H 概率为 m/N、T概率为 n/N. 假如在此数据集中共有 k 个 H, 则通过此数据得到的最大似然概率估计为 $p^{ML} = k/N$. 从而,

$$P(p^{ML} = k/N) = \left(\frac{m}{N}\right)^k \left(\frac{n}{N}\right)^{N-k} \binom{N}{k}. \tag{8.16}$$

对于较大的 N 而言, 我们可以用下述分布将其近似为

$$P(p^{ML} = p) = \left(\frac{m}{N}\right)^{Np} \left(\frac{n}{N}\right)^{N-Np} (N+1)\binom{N}{Np}, \tag{8.17}$$

其中出现因子 $(N+1)$ 是因为我们把二项展开式中的 $(N+1)$ 项都替换为 $[0,1]$ 上的密度. 对于 N 较大的情况, 我们可以通过正态分布(本书第 p. 206页) 将 (8.15) 式近似为:

$$P(p|m\mathrm{H}, n\mathrm{T}) \simeq \frac{(N+1)}{\sqrt{2\pi Np(1-p)}} \exp\left(-\frac{(m-Np)^2}{2Np(1-p)}\right), \tag{8.18}$$

与此类似, 8.17可改写为

$$P(p^{ML} = p) \simeq \frac{(N+1)}{\sqrt{2\pi mn/N}} \exp\left(-\frac{(m-Np)^2}{2mn/N}\right). \tag{8.19}$$

一个很直接的练习就可以证明, 当 N 很大即如果有很多数据时, 这两个分布会彼此逼近 (见练习8.14). □

此结果也可以很容易地扩展到多项分布. 在系统发育的例子中, 为获得自举数据集而选择特定叶节点分配的概率服从多项分布. 下面我们考虑这种情况, 即当前分布退化为二项分布时, 我们便可直接应用上述投掷硬币的例子.

例子: 用于简单树的自举法

假设存在两条核酸序列, 并且我们想采用带两个叶节点的树以及 Jukes-Cantor 替换矩阵对其进行建模. 我们可以将两叶节点之一设定为根节点, 并令 t 为连接两叶节点的边长. 令 n_s 表示两叶节点有相同核酸的原始数据集位点数, n_d 为两叶节点有不同核酸的原始数据集位点数; 若共有 N 个位点则 $n_d + n_s = N$. 当扩展 (8.9) 式中的计算, 并且假设 $e^{-\alpha t}$ 上的平坦先验时, 后验概率可写为

$$P(e^{-\alpha t}|\mathrm{data}) = (1+3e^{-\alpha t})^{n_s}(1-e^{-\alpha t})^{n_d}/Z, \tag{8.20}$$

其中 Z 是从 Bayes 定理得出的归一化因子 (normalising factor). 假设 $n_{\mathtt{XY}}$ 为原始数据集中XY型叶节点出现的次数, $m_{\mathtt{XY}}$ 为自举数据集中相对应的次数. 则取出自举数据集的概率可通过多项分布给出:

$$P(m_\bullet|n_\bullet) = \left(\frac{n_{\mathtt{AA}}}{N}\right)^{m_{\mathtt{AA}}} \left(\frac{n_{\mathtt{AC}}}{N}\right)^{m_{\mathtt{AC}}} \cdots \frac{N!}{m_{\mathtt{AA}}!m_{\mathtt{AC}}!\cdots}. \tag{8.21}$$

至此, 自举数据集的最大似然可通过练习8.7的一个明显扩展给出, 即

$$\exp(-\alpha t^{ML}) = \frac{3m_s - m_d}{3N}, \tag{8.22}$$

其中 m_s 和 m_d 分别为自举数据集中相同和不同的叶节点个数. 因此 t^{ML} 仅依赖于 m_s 和 m_d, 而与单个计数 $m_{\mathtt{XY}}$ 无关, 并且我们可以对所有由 (8.21) 式给出的带相等 m_s 和 m_d 的 $P(m_\bullet|n_\bullet)$ 求和以便确定自举值 t^{ML} 出现的频率. 对所有这些项求和, 有

$$P\left(\mathrm{e}^{-\alpha t^{ML}} = \frac{3m_s - m_d}{3N}\right) = \left(\frac{n_s}{N}\right)^{m_s}\left(\frac{n_d}{N}\right)^{m_d}\binom{N}{m_s}. \tag{8.23}$$

把式 (8.20) 与式 (8.15) 以及式 (8.16) 与式 (8.23) 做对比, 并注意到式 (8.22) 意味着 $m_s = N \times (1 + 3\exp(-\alpha t^{ML}))/4$ 和 $m_d = 3N(1 - \exp(-\alpha t^{ML}))/4$, 我们就可知该树的后验近似于自举结果, 这与投掷硬币的例子相同. □

因此对某些系统发育模型和足够多的数据(N 足够大) 而言, 自举分布可以给出后验的很好近似. 然而评估大量最大似然树所花费的工作量, 使得用自举法替代抽样法时并不很令人满意. 自举法很可能更适用于非概率论建树方法. 但是自举与后验间的关系的确给出了一些深层领悟. 它尤其有助于反驳 Hillis & Bull [1993] 所提出的异议. 这些作者从树中产生抽样数据集, 并找出给定树拓扑由简约法重建的频率分布; 对每个抽样数据集, 他们同时给出了该拓扑的自举频率. 他们发现自举频率的分布比原始抽样的分布宽得多. 然而这并不奇怪, 因为先抽样再自举的过程引入了这两个步骤的方差. 事实上在他们的模拟中, 自举分布在**给定数据下** [Efron, Halloran & Holmes 1996] 有正确的方差, 因此可被看作为后验分布.

练习

8.14 证明对于足够大的 N, 式 (8.18) 中的 $P(p|m\mathrm{H}, n\mathrm{T})$ 和式 (8.19) 中的 $P(p^{ML} = p)$ 的值要么都非常小要么几乎相等.

8.5 更现实的进化模型

迄今为止, 我们所使用的进化模型使用了许多极端的简化假设(本书第 p. 134页). 对无空位联配的限制抛弃了由插入与删除模式给出的有用系统发育学信息. 8.11中的假设, 即用相同替换矩阵对序列中的每个位点建模显然也是不正确的, 因为蛋白结构和RNA 的碱基配对等因素向不同位点施加了不同的约束. 为了关注于位点的单一基本属性, 人们早就知道某些位点的替换比其他位点发生得更快 [Fitch & Margoliash 1967b]. 通过允许可变进化速率, 我们首先描述为这些情况建模的一些尝试, 然后我们再转向处理含空位联配的方法.

允许不同位点具有不同进化速率

最大似然法的基本策略是, 选取树 T 和一组长度 $t_\bullet$, 然后利用下式计算所有位点的似然:

$$P(x^\bullet|T, t_\bullet) = \prod_{u=1}^{N} P(x_u^\bullet|T, t_\bullet).$$

Yang [1993] 建议引入位点相关(site-dependent) 变量 r_u, 它可以为位点 u 处的所有 $t_\bullet$ 进行比例调整. 如果已知每一位点处的 r_u 值, 那么我们可以将似然写为

$$P(x^\bullet|T, t_\bullet, r_u) = \prod_{u=1}^{N} P(x_u^\bullet|T, r_u t_\bullet).$$

因为通常情况下 r_u 未知, 所以最好的办法就是为其假设先验, 然后对每个 r 的所有值求积分. Yang [1993] 使用 gamma 分布 $g(r, \alpha, \alpha)$ 作为其先验; 此分布的均值为1方差为 $1/\alpha$, 因此它所允许的分布范围就在紧密的分布 (α很大) 和宽散的分布 (α很小) 之间. 于是似然为

$$P(x^\bullet|T, t_\bullet, \alpha) = \prod_{u=1}^{N} \int_0^\infty P(x_u^\bullet|T, rt_\bullet) g(r, \alpha, \alpha) dr. \tag{8.24}$$

对每个固定的 T, 该似然关于 $t_\bullet$ 和 α 达到最大值. Yang 使用一组球蛋白得到了包括四种哺乳动物的最优树, 并指出与 Felsenstein [1981a] 最初的方法即将所有位点上时间固定相比, 当允许可变时间时对数似然显著下降.

我们可以解析地求出式 (8.24) 中的积分(它们只是 gamma 函数的积分), 但是结果表达式里项的个数却会随序列个数呈指数增长, 因此从计算上讲该最优化过程运行较慢. 为此 Yang [1994] 提出了一种近似方法, 把积分替换为离散求和. 间隔 $(0,\infty)$ 被分割为 m 个小区间, 使 gamma 分布 $g(r,\alpha,\alpha)$ 在每个区间中都有相同的面积. 令 r_k 表示 gamma 分布在第 k 个间隔内的均值. 于是我们可以定义

$$P(x^\bullet|T,t_\bullet,\alpha)=\prod_{u=1}^{N}\sum_{k=1}^{k=m}P(x_u^\bullet|T,r_kt_\bullet)/m. \tag{8.25}$$

Yang 发现如果 $m=3,4$, 那么上式就足以给出模型连续形式的一个很好近似. 因为计算量只是非变化 (non-varying) 位点方法的 m 倍, 所以该算法更易于处理.

同连续模型中一样, 此处的 α 可以通过最大似然法从给定数据中估计出来. 当数据量较大时上述方法是可接受的, 但对较小的数据集, 该方法会受困于从计数中估计概率时 profile HMM 所面临的诸多问题. 因此, 我们最好还是从大量的可信系统发育关系数据出发推断 α 值.

Felsenstein & Churchill [1996] 提出一个类似于 Yang 的算法, 但是却采用隐马模型的形式. 在他们的模型中, 每一位置都对应于联配中的一个位点. 每个位置处都存在一些状态, 它们对应于不同的进化速率. 在相邻的位置上, 所有速率-状态 (rate-state) 间都存在转移. 假如在位点 u 选出状态 i, 那么它就为似然贡献项 $P(x_u^\bullet|T,r_it_\bullet)$, 其中 r_i 为状态 i 的速率. 随后可以用前向算法 (forward algorithm) 的一个变体为所有可能的速率分配给出概率和. 与本书第 p. 41页中 HMM 的前向算法不同, 在此方法中: (1) 贯穿模型的路径是所选速率的集合, 而非序列到模型的联配; (2) 概率是某位点处整个序列集的似然, 而非总和为1的来自某状态的发射概率. 然而从形式上看, 当我们令 HMM 前向算法中的 $e_l(x_i)$ 为位点 i 处速率 l 的似然, 并令 HMM 前向算法中的 a_{kl} 为速率 k 到 l 的转移概率时, 这两种算法却是一致的.

除了隐马模型包括状态间的转移概率 a_{kl} 外, Felsenstein & Churchill 隐马模型中的总似然与 Yang 离散模型(8.25) 所给出的完全相同. 这些转移概率能够捕捉在相继位点处出现速率特定模式的任意改变趋势. 在蛋白质中, 替换会更自由地发生在表面位点, 而其速率的模式将依赖于其二级结构的类型. 环型结构中应该出现一系列的外露位点, 因此我们可以使得模型中快速率之状态间的转移具有更大概率. 但是 β 折叠片将会显示内含 (buried) 和外露 (exposed) 残基的另一种模式, 而螺旋结构会呈现一种简易的三联体模式. 为了表现这些结构特性, 我们需要更精巧设计的模型体系结构.

含空位的进化模型

我们现在关注叶节点序列之联配结果中含有空位的情况. 可以通过下述方法简略地将空位引入模型: 把"–" 作为含 K 个残基之字母表中的一个额外字符, 并将 $K\times K$ 大小的残基替换矩阵更换为含有空位字符大小为 $(K+1)\times(K+1)$ 的替换矩阵. 在这里我们常犯的错误是, 认为相邻位点处的空位相互独立, 因此并不考虑空位成块出现的趋势.

引入插入和删除状态可以生成更好的进化模型: Allison, Wallace & Yee [1992b] 给出了原则上如何这样做, 以便在系统发育中给出仿射类型的空位. 他们的方法使用最小描述长度 (minimum description length), 这与最大似然法非常相近. 然而不幸的是, 就目前来讲, 用这种方法处理仿射空位罚分在计算上难以完成 (computationally intractable).

另一种方法通过片段的替换模型 (model of fragment substitution) [Thorne, Kishino & Felsenstein 1992] 实现, 它在一定程度上表现出了生物学的似真性 (plausibility), 但至今为止却只能应用到两条序列的情况.

现在, 我们描述一种以计算上合理的方式处理仿射类型空位罚分的模型. 这便是**树 HMM** (tree HMM) [Mitchison & Durbin 1995], 它使用 profile HMM 体系结构并将贯穿模型的路径视为经历进化改变的对象 [Mitchison 1998].

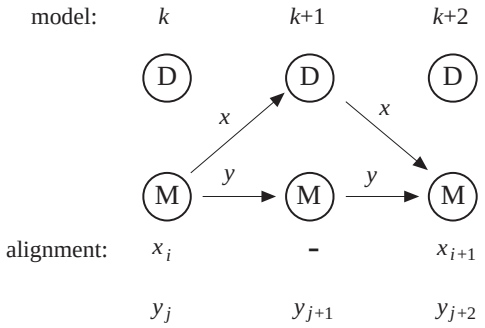


图 8.9: HMM 的一个片段, 图中显示了 x 和 y 的路径.

假设使用比 Krogh *et al.* [1994]中profile HMM 简单的体系结构, 我们可以将序列联配到某隐马模型; 该体系结构只含有匹配和删除两种状态, 分别用 M_k 和 D_k 表示, 其中 k 表示模型中的位置. 假设序列 y 为序列 x 的祖先; 我们可以认为这两条序列分别位于两个节点, 并且在树中这两个节点以长度为 t 的边相连. 假设两序列都联配到我们的模型, 那么每条都将对应一条规定好的贯穿模型之路径, 并在匹配状态发射特定的残基. 考察图8.9中该模型的一部分. 两序列都使用匹配状态 M_k, x 在 M_k 发射残基 x_i, y 在 M_k 发射残基 y_j. 替换 $y_j \to x_i$ 的概率可表示为 $P(x_i|y_j, t)$, 这与标准最大似然的系统发育中所用的一致.

接下来考虑这种可能性, 即 y 使用不同于 x 的**转移**, 所以它并不经过与 x 相同的状态序列. 在图8.9中的位置 k, x 从匹配状态转移到了删除状态: $M_k \to D_{k+1}$, 而 y 则转移到了下一个匹配状态: $M_k \to M_{k+1}$. 与从 y_j 到 x_i 之发射的替换类似, 我们为这种转移的替换设定概率值. 令" MM " 表示匹配到匹配的转移, " MD " 表示匹配到删除的转移, 于是此概率值可记为 $P(\mathrm{MD}|\mathrm{MM}, t)$.

在图8.9中的位置 $k+1$ 处, x 发生了转移 $D_{k+1} \to M_{k+2}$, 简写为" DM " , 而 y 发生了 MM 转移 $M_{k+1} \to M_{k+2}$. 此处我们假设 x 的变化独立于 y, 无论两序列的起始状态是否相同. 因此假如在 x 中出现一个相对于 y 的删除, 那么在此过程中对 DD 或 DM 的选择就决定了删除的长度, 并且我们认为这些选择受独立于序列 y 的突变过程所控制. 假设独立路径中 x 所使用的转移概率由先验得到; 因此 x 的转移 $D_{k+1} \to M_{k+2}$ 具有概率 q_{DM}.

我们可以在 4×4 的矩阵里表示转移的替换和先验, 它们分别对应该特定 HMM 体系结构中所出现的四种转移, 即 MM、MD、DM 和 DD . 然而这并不是一个标准的替换矩阵, 因为矩阵中每一行的概率之和并不都等于1. 然而它可以分块为四个 2×2 的矩阵, 这些小矩阵均取决于祖先和后代序列开始它们转移的起始状态 (匹配或删除):

$$\begin{pmatrix} \begin{pmatrix} P(\mathrm{MM}|\mathrm{MM},t) & P(\mathrm{MD}|\mathrm{MM},t) \\ P(\mathrm{MM}|\mathrm{MD},t) & P(\mathrm{MD}|\mathrm{MD},t) \end{pmatrix} & \begin{pmatrix} q_{\mathrm{DM}} & q_{\mathrm{DD}} \\ q_{\mathrm{DM}} & q_{\mathrm{DD}} \end{pmatrix} \\ \begin{pmatrix} q_{\mathrm{MM}} & q_{\mathrm{MD}} \\ q_{\mathrm{MM}} & q_{\mathrm{MD}} \end{pmatrix} & \begin{pmatrix} P(\mathrm{DM}|\mathrm{DM},t) & P(\mathrm{DD}|\mathrm{DM},t) \\ P(\mathrm{DM}|\mathrm{DD},t) & P(\mathrm{DD}|\mathrm{DD},t) \end{pmatrix} \end{pmatrix}.$$

现在我们考察图8.9所示的树 HMM . 在位置 k, 我们有发射项 $q_{y_j} P(x_i|y_j, t)$, 其中 q_{y_j} 来自 y 的根先验. 转移贡献了 $q_{\mathrm{MM}} P(\mathrm{MD}|\mathrm{MM}, t)$ 项, 其中 q_{MM} 为转移 MM

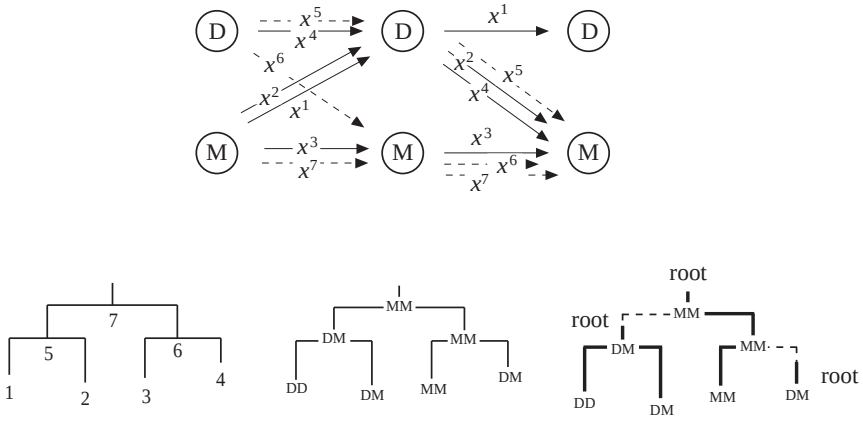


图 8.10: 带4个叶节点的树 HMM 片断. 叶节点序列的路径用实箭头表示, 一种可能的祖先路径用虚箭头表示. 下面画出了树, 其中数字表示节点 (左), 还画出了模型中心位置发生的转移 (中) 以及使用先验代替替换概率处的根 (root) 节点位置 (右), 即父序列和子序列起始于不同状态处的根节点位置.

的根先验. 假如对于在上述替换矩阵和树概率的表达式中均包含先验存有疑惑, 那么请注意它们的来源相同: 当某种序列行为无法用祖先解释时, 我们就只好求助于先验. 因此在位置 $k+1$, 转移给出 $q_{\mathrm{MM}}q_{\mathrm{DM}}$, 出现这两个先验项是因为此处这两条序列的行为都独立于其祖先(尽管已知 y 是 x 的祖先, 而 y 的祖先未知).

现在, 假设存在一棵带边长 $t_\bullet$ 的任意树 T, 其叶节点序列 $x^\bullet$ 全部联配上 HMM. 根据无空位联配的概率论模型8.10进行类推, 我们通过相乘 T 中所有边的替换概率以及根节点的先验项, 来定义 $P(x^\bullet|T,t_\bullet)$. 然而树 HMM 中存在两种用于相乘的替换概率, 即发射概率和转移的概率. 为了得到无空位模型的全概率, 我们在每个位置都对祖先节点的所有可能残基分配求和. 同样, 在树 HMM 中我们对相关变量的所有可能分配求和, 此时既包括发射也包括转移. 如果我们不仅通过它所使用的转移, 而且还通过它所发射的字符来定义一条路径, 那么我们就相当于对祖先序列使用的所有可能路径求和.

图8.10描述了似然中某些项的计算过程. 在模型的中间位置上, 转移 DD 、DM、 MM 和 DM 分别依次出现在叶节点1、2、3和4上. 假设它们都由这些叶节点序列的已知联配给出. 然而祖先转移未知, 因此必须求和算出. 虚线箭头表示一组祖先路径; 还存在许多其他可能的组合. 在模型的中间位置处, 我们可以从祖先路径的给定集合推导出画在其下面的转移树 (transition tree, 正中位置). 可以从转移矩阵计算其概率如下:

$$\begin{aligned}\text{树的概率} &= q_{\mathrm{MM}}P(\mathrm{MM}|\mathrm{MM},t_6)P(\mathrm{MM}|\mathrm{MM},t_3)\\ &\quad\times q_{\mathrm{DM}}\times q_{\mathrm{DM}}P(\mathrm{DD}|\mathrm{DM},t_1)P(\mathrm{DM}|\mathrm{DM},t_2).\end{aligned}$$

上式中被乘号隔开的三项因子可视为沿树枝断裂 (break) 所形成的子树之概率. 转移的开始状态出现改变时, 这样的断裂便会产生.

所有这些项均可通过动态规划算法求和得到; 然而其速度比 profile HMM 的前向算法要慢很多, 因为它需要同时记录当前路径所使用的前驱状态和其祖先路径所使用的状态. 这所导致的计算负荷随序列条数的增加呈指数增长, 因此该算法只适用于少量序列. 但是存在对此似然的较好近似 [Mitchison 1998], 其计算所需的资源与 Felsenstein [1981a] 的最初算法相当.

评估不同的概率论模型

进化模型发展得越来越复杂, 但却存在一个问题, 那就是通过模型结构的添加, 我们可能不清楚到底得到了什么. 假如模型 M_2 比模型 M_1 复杂, 那么 M_2 似然的最

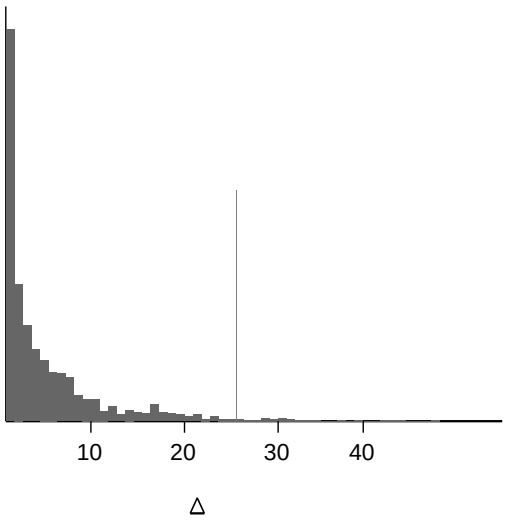


图 8.11: 正文中给出的两个替换矩阵的例子. 直方图显示模拟数据中对数似然差 Δ_i 的分布. 细的竖线是最初数据的 Δ 值.

大值可能比 M_1 的大 (事实上一般来讲, 当 M_2 包含 M_1 为特例, 并且 M_2 是在 M_1 基础上的更精细模型时, 该论断都正确). 但是, 假如参数仅在很小范围内取值时似然才是不可忽略 (non-negligible) 的, 那么 M_2 有就可能是一个较差的模型. 因此相对于比较其最大似然, 有一种更好的方法是比较概率 $P(D|M_1)$ 和 $P(D|M_2)$, 它们可以通过对每个模型的所有参数求积分得到. 更确切地讲, 假如 M_1 含有参数 $\theta_\bullet$ 且其先验概率为 $P(\theta)$, 那么我们有

$$P(D|M_1) = \int P(D|M_1,\theta)P(\theta_\bullet)d\theta_1 \ldots d\theta_n,$$

类似地有 $P(D|M_2)$. 概率 $P(D|M)$ 有时被称为给定数据下模型 M 的**证据** (evidence) [MacKay 1992]. 如果 $P(D|M)$ 的不可忽略贡献部分以先验概率 P_r 来自参数空间的一小块区域, 那么这就设定了边界 $P(D|M) < \max_\theta P(D|M,\theta)P_r$, 而且 $P(D|M)$ 的值会很小. 考虑先验概率 $P(M_1)$ 和 $P(M_2)$ 时比较两个模型 M_1 和 M_2 的一种自然的方法就是, 计算由下式给出的 M_1 的后验概率:

$$P(M_1|D) = \frac{P(D|M_1)P(M_1)}{P(D|M_1)P(M_1) + P(D|M_2)P(M_2)}. \tag{8.26}$$

Goldman [1993] 根据 Cox [1962] 的方法, 提出了另一种评价模型的方法. 令 $\hat{L}_1(D)$ 和 $\hat{L}_2(D)$ 分别表示模型 M_1 和 M_2 对数据 D 的最大似然, 评估每个最大似然时均独立于另一模型(最大似然有可能出现于其共用参数的不同点). 令

$$\Delta = \log(\hat{L}_2(D)) - \log(\hat{L}_1(D)).$$

鉴于上述原因, Δ 值本身并不能很好反映模型 M_2 的优越性. 但如果现在使用数据 D 下最大似然的 M_1 的参数值从 M_1 中模拟数据集 D_i, 那么我们就可以知道, 在模拟数据集 D_i 下 Δ_i 值的分布中, 最初 Δ 值是典型的 (typical , 例如落在该分布95%的界限内) 还是几乎超过了所有的 Δ_i. 如果是后者, 那么更复杂的模型 M_2 就已经反映了 M_1 无法展现之数据集所包含的某方面信息, 因此我们就可以拒绝 M_1.

该检验法有时称为**参数自举** (parametric bootstrap), 并且比前面所定义的常用自举法(本书第 p. 124页) 功能更强大,因此更适用于概率论模型的显著性检验.Goldman 描述了如何应用参数自举将我们现在所关心的系统发育模型 M_1 与包含非常多参数的模型 M_2 进行比较, 其中 M_2 为每一位点上所有可能的残基集合都

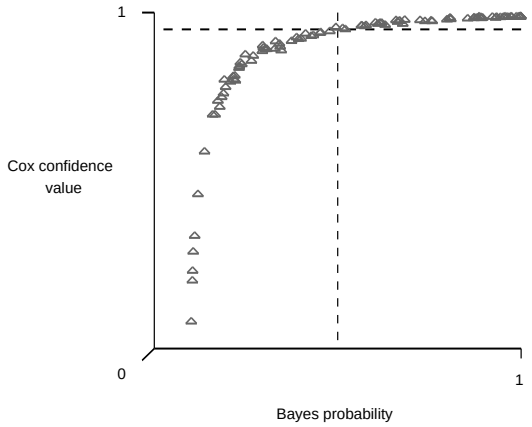


图 8.12: 单参数和双参数模型间的比较. 对100个大小为500的数据集 D, 图中画出了 Beyes 概率 $P(M_2|D)$ 以及从图8.11之柱状图中估计出的 Cox 置信值 $P(\Delta_i < \Delta)$. 水平虚线是95%的置信限, 而垂直虚线是 $P(M_2|D) = 0.5$.

赋予概率值. 下述例子描述了在一种简单的情况下如何使用该方法, 以及在相同数据上的 Bayes 模型比较方法.

例子: 两种替换矩阵模型间的比较

假设存在两种残基A和B, 以及两种替换模型: 带一个参数 p 的 M_1

$$\begin{pmatrix} 1-p & p \\ p & 1-p \end{pmatrix}, \tag{8.27}$$

和带两个参数 p_1, p_2 的模型 M_2

$$\begin{pmatrix} 1-p_1 & p_1 \\ p_2 & 1-p_2 \end{pmatrix}, \tag{8.28}$$

首先以参数 $p_1 = 0.5$ 和 $p_2 = 0.4$ 从 M_2 中抽样, 构造一个基本数据集 D. 假设我们以等概率随机地选取总共 $N = 500$ 个A和B, 并用矩阵8.28中的条件概率得出残基. 我们用 $n_{\mathtt{AA}}$ 和 $n_{\mathtt{AB}}$ 分别表示从A得到A和B的数目, 并用 $n_{\mathtt{BA}}$ 和 $n_{\mathtt{BB}}$ 分别表示从 B 得到A和B的个数. p_1 和 p_2 的值十分接近, 因此用数据拟合 M_1 和 M_2 时应该都不太差; 问题是我们的两种检验是否可以识别出对 M_2 的潜在更好拟合.

给定该数据集 D 后, 我们为 M_1 确定 p 值的最大似然, 并且模拟产生了来自 M_1 的1000个数据集 D_i. 我们对每个数据集都计算 Δ_i 并累积其分布, 图8.11画出了其柱状图. 细竖线标出了基本数据集 D 的 Δ 值. 此时它便给出了 $P(\Delta_i < \Delta)$ 的估计0.985. 于是这也让我们知道 D 落在了分布的 95% 边界以外, 因此 M_1 被拒绝, 故而有双参数模型 M_2 更为合适.

如果重复整个实验, 从对 M_2 抽样以获得新数据集 D 开始, 我们将得到 $P(\Delta_i < \Delta)$ 值的分布. 现在, 我们就用此分布与 Bayes 概率的分布进行比较. 为此我们假设所有参数的平坦先验, 于是

$$\begin{aligned} P(D|M_1) &= \int P(D|M_1, p)dp \\ &= \beta \int p^{n_{\mathtt{AB}}+n_{\mathtt{BA}}}(1-p)^{n_{\mathtt{AA}}+n_{\mathtt{BB}}}dp, \end{aligned}$$

其中 β 为二项因子, $P(D|M_1)$ 和 $P(D|M_2)$ 都包含此项. $P(D|M_2)$ 的相应表达式为

$$P(D|M_2) = \beta \int p_1^{n_{\mathtt{AB}}}(1-p_1)^{n_{\mathtt{AA}}} p_2^{n_{\mathtt{BA}}}(1-p_2)^{n_{\mathtt{BB}}}dp_1 dp_2.$$

上面的积分均可改写为阶乘形式; 见11.6:

$$P(D|M_1) = \frac{\beta(n_{\mathtt{AB}} + n_{\mathtt{BA}})!(n_{\mathtt{AA}} + n_{\mathtt{BB}})!}{(N+1)!},$$
$$P(D|M_2) = \frac{\beta n_{\mathtt{AB}}! n_{\mathtt{BA}}! n_{\mathtt{AA}}! n_{\mathtt{BB}}!}{(n_{\mathtt{AB}} + n_{\mathtt{AA}})!(n_{\mathtt{BA}} + n_{\mathtt{BB}})!},$$

如果假设 $P(M_1)$ 和 $P(M_2)$ 有相等的先验概率, 那么我们就可以用8.26从上式中计算出 $P(M_2|D)$. 图8.12显示了从100 个数据集 D 中获得的 $P(M_2|D)$ 值分布, 其中也标识出 $P(\Delta_i < \Delta)$ 的估计值. 对后一个概率超过0.95, 即以95% 的置信度拒绝 M_1 的那些点, 我们也发现 $P(M_2|D) > 0.5$, 这表明了对 M_2 的偏好. 这些检验在本质上十分不同, 因此表现出这些特定数据上的某种一致性. 然而当 D 中的数据点个数 N 增加时, Bayes 方法经常倾向于偏好于被参数自举法拒绝的 M_1, 而当数据点数目变小时却会出现相反的趋势. 我们需要深入探索上述两种方法间的关系, 尤其要考虑到似然比方法已经逐渐被广泛使用 [Huelsenbeck & Rannala 1997]. □

8.6 概率论方法与非概率论方法的比较

本章的剩余部分将回到前一章所讲到的系统发育方法上, 即简约法和成对距离法, 并给出其概率论解释.

简约法的概率论解释

假如给定一组替换概率 $P(b|a)$, 并忽略它们与长度 t 的相关性. 令 $S(a,b) = -\log P(b|a)$, 我们便可以得到一组替换罚值. 假如将这些罚值用于加权简约法, 那么正如 Felsenstein [1981b]指出的, 从加权简约法(本书第 p. 120页) 算出的整棵树 T 中位点 u 的最小罚值可视为似然的一个近似. 事实上, 它就是式(8.10), 即全概率 $P(x_u^1, \ldots, x_u^n|T)$ 的 Viterbi 近似. 正如在 HMM 中全概率需要对所有路径求和, 而 Viterbi 法搜索最大概率路径一样, 式 (8.10) 中的概率对祖先节点的所有残基分配求和, 而简约法则通过最小化负概率 $-\log P(b|a)$ 之和来搜索一组能最大化此概率的祖先残基分配. 上述对应并不完全, 因为简约法常常不包括与概率论模型之根分布相对应的等价量. 但是如果假设其分布是平坦的, 那么它就向公式中贡献一个常数项, 当计算树的简约法最优值时此常数项可以忽略.

并非所有的罚值函数 $S(a,b)$ 都可以用上述方法从概率算出. 然而传统简约法的罚值, 例如令残基替换为1、残基相同为0, 可以很容易的解释为对数概率. 事实上, 对角元是 α 而其他元是 β 且 $\beta < \alpha$ 的任意替换矩阵都可以解释为对数概率. 这是因为使用 $S(a,a) = -\log P(\alpha)$ 和 $S(a,b) = -\log P(\beta)$ 其中 $a \neq b$ 的简约法, 就等价于传统的简约法(见练习8.15).

简约法的诱人之处在于其运行速度. 事实上, 简约法在计算上的主要进步就是它无须像最大似然法那样优化边长. 如果将简约法理解为最大似然法的 Viterbi 近似, 那么舍弃 $P(a|b,t)$ 中的时间参数 t 就可以完成这种简化. 然而正如下面的例子所说明, 这样会导致不利的后果.

例子: 简约法和最大似然法的比较

检验建树算法性能的一种简单方法是通过抽样随机地生成树, 然后考虑给定算法正确建树的频度. 抽样过程首先在根节点以概率 q_a 选取残基 a, 然后以概率 $P(b|a,t_i)$ 接受沿节点 i 上的边从 a 到 b 的替换, 依此类推沿树向下进行. 从而这就产生出叶节点上的一种残基分配. 对于无根树, 我们可以选择任意节点作为根并开始上述过程. 只要生成模型可逆, 根节点的选取就无关紧要.

如果使用相同的概率论模型建树, 那么最大似然法就会因其一致性而在大数据量的极限情况下正确地重建树. 一个有趣的问题是, 其他算法对此任务的表现如何.

图8.13中的四叶树已经成为许多此类模拟研究的必备工具. 其中特别引人好奇的情形是, 两兄弟叶节点 ‖ 的边长较短, 而剩余两边较长. 对这棵树的研究最早由 Felsenstein [1978a] 和 Cavender [1978] 完成, 他们发现即使在大数据量的情况下, 简

‖原文为 two sister leaves (两姐妹叶节点), 即相邻的两个叶节点. ——译注

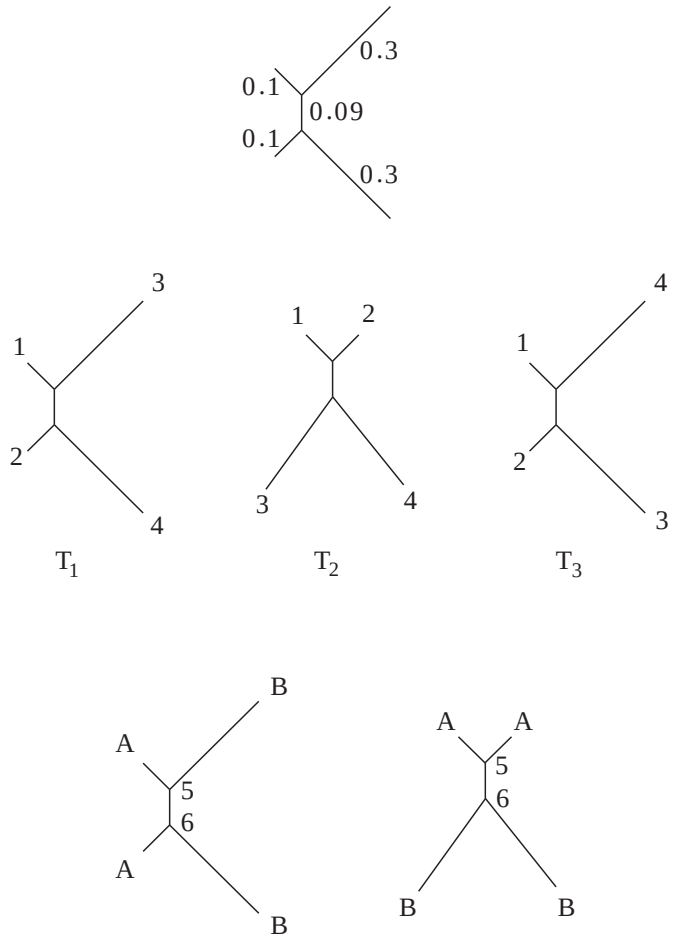


图 8.13: 图: 边长明显不等的无根树. 中图: 原树 T_1 以及另外两种无根树 (T_2 和 T_3). 下图: 对拓扑 T_1 和 T_2, 为已编号的叶节点进行特定的残基分配.

约法仍然会产生错误结果. 依据 Felsenstein 的方法, 为简单起见我们假设字母表仅包含两字符 {A, B}, 并且令替换矩阵为[7]

$$\begin{pmatrix} 1-p & p \\ p & 1-p \end{pmatrix} \tag{8.29}$$

令叶节点1和3的 $p = 0.3$ 而叶节点2和4的 $p = 0.1$, 并且连接叶节点的边长 $p = 0.09$. 如图8.13所示.

存在三种可能的四叶无根树 (本书第 p. 114 页); 我们称原树为 T_1, 而另外两棵为 T_2 和 T_3. 下面的表格显示了在不同序列长度 N 的情况下, 对最大似然法或简约法重建抽样树进行1000次检验的结果. 列元素表示选择树 T_i 的次数.

最大似然法的建树结果:

[7]令 $p = \frac{1}{2}(1 - \exp(-\alpha t))$ 可将其转化为可乘矩阵族, 但此处我们不用这种形式. ——原注

N	T_1	T_2	T_3
20	419	339	242
100	638	204	158
500	904	61	35
2000	997	3	0

简约法的建树结果:

N	T_1	T_2	T_3
20	396	378	224
100	405	515	79
500	404	594	2
2000	353	646	0

值得注意的是, 正如我们所料, 当 N 增加时, 最大似然法结果中的 T_1 也随即增加. 但对简约法而言, 情况却并非如此, 其中一个明显的偏差是 T_2 的被选次数随N的增加而增加. 为查明简约法出错的原因, 考虑将残基A、A、B、B分别分配给叶节点1、2、3、4 (图8.13中左下图); 对此给定边长而言这种情况是很常见的, 因为连接叶节点3与4的长边上更易发生替换, 而叶节点1和2间距离较近.

假如沿节点5和6间的"桥" 发生一次替换, 那么这就会使树 T_1 产生两个失配的简约法罚值, 而树 T_2 中只产生一个失配(图8.13中右下图). 最大似然不能通过这种方法得到. 当具有正确边长时, 节点5和6间的替换因其边长过短而不大可能发生. 因此, 该最有可能的解释就是在 T_2 中也需要两个替换以便和 T_1 中的一致. 这也表明了隐藏在简约法中的时间无关性 (time-independence) 的弊端.

本例中的树也许显得有些病态 (pathological), 因为终端边长的差异十分悬殊, 而且该树严重偏离分子钟假设. 然而的确存在满足分子钟假设的五叶树, 但简约法却仍无法正确重建之 [Hendy & Penny 1989]. □

练习

8.15 证明使用罚值 $S(a,a) = -\log P(\alpha)$ 和 $S(a,b) = -\log P(\beta)$, 其中 $a \neq b$ 寻找最简约的树, 等价于使用失配罚值为1的传统简约法.

最大似然法与成对距离法

接下来我们回过头讲述成对距离法, 并找出它与概率论建模之间的关系.

假设给定边长为 $t_\bullet$ 的树 T, 使用可乘可逆的替换矩阵对叶节点处长度为 N 的序列进行抽样, 如本书第 p. 154页所述.选取两个叶节点 i, j. 容易看出, 这些叶节点处的抽样序列 同时也是"拆解" (stripped-down) 树的抽样, 而删除连接 i, j 的路径外之所有其他边便可以得到拆解树(见图8.14的左图). 这是因为只有在从根到 i 或 j 的边上之抽样步骤才与 i, j 处的残基选择有关. 另外, "拆解" 树顶端节点(图8.14中的节点8) 以上的原树部分与 i, j 处的残基选择无关, 这是因为由可逆性可知, 该顶端节点处的分布与根节点的分布相同.

运用矩阵的可乘性, 我们可以对从顶端节点到 i 或 j 的所有路径中的边长求和. 例如给定图8.14中的树且令 $i = 1, j = 3$, 那么从可乘性就有

$$P(a^1|a^8, t_1 + t_6) = \sum_{a^6} P(a^1|a^6, t_1)P(a^6|a^8, t_6),$$

其中 a_k 为节点 k 处的残基. 这就说明在给定 a^8 的情况下, 沿长度为 $t_1 + t_6$ 的边所得到的抽样在叶节点1处选择残基的概率, 将与抽样先在节点6选择残基继而在叶节点1选择残基所依概率相同.

可逆性意味着, 我们可以进一步通过颠倒树枝将拆解树"拉直" (straighten out). 例如给定图8.14的中间那棵树以及根节点分布 q, 那么残基 a^1 与 a^3 的概率就等

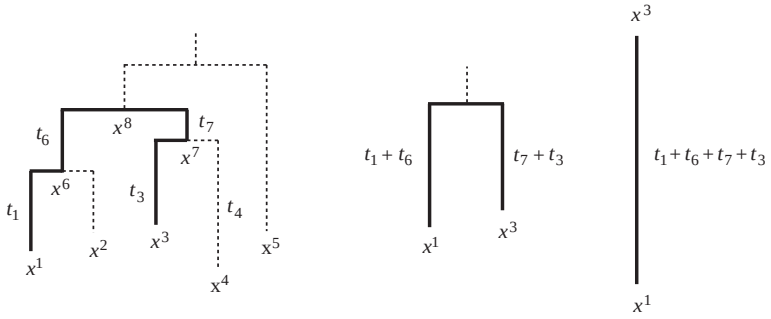


图 8.14: 连接两个叶节点1和3的最短路径用粗线表示.

于先以概率 q 选取 a^3, 然后从边长为 $t_1+t_6+t_7+t_3$ 的单叶树中抽样得到 a^1 一样(图8.14中的右图). 这是因为

$$\begin{aligned}&\sum_{a^8} P(a^1|a^8, t_1+t_6)P(a^3|a^8, t_7+t_3)q_{a^8}\\ =&\sum_{a^8} P(a^1|a^8, t_1+t_6)P(a^8|a^3, t_7+t_3)q_{a^3}\\ =&P(a^1|a^3, t_1+t_6+t_3+t_7)q_{a^3}.\end{aligned}$$

对一般树来说, 假设连接 i, j 的边长为 $t_{k_1}, t_{k_2}, \ldots, t_{k_r}$. 那么我们的抽样理论就使得

$$P(x_u^i, x_u^j|T, t_\bullet) = q_{x_u^j} P(x_u^i|x_u^j, t_{k_1}+t_{k_2}+\ldots+t_{k_r}).$$

定义**最大似然距离** (maximum likelihood distance) [Felsenstein 1996] 为

$$d_{ij}^{ML} = \underset{t}{\operatorname{argmax}} \left\{ \prod_u q_{x_u^j} P(x_u^i|x_u^j, t) \right\},$$

上式大括号中对所有位点 u 求连乘积. 由于项 $q_{x_u^j}$ 独立于 t, 故我们可以将其改写为

$$d_{ij}^{ML} = \underset{t}{\operatorname{argmax}} \left\{ \prod_u P(x_u^i|x_u^j, t) \right\}. \tag{8.30}$$

于是当 N 很大时, 最大似然的一致性(本书第 p. 213页) 意味着

$$d_{ij}^{ML} \simeq t_{k_1}+t_{k_2}+\ldots+t_{k_r}. \tag{8.31}$$

如果概率论模型是正确的, 那么对于给定的大量数据而言, 叶节点序列间的最大似然距离应当十分接近可加. 现在我们已经知道邻接法可以正确的重建可加树, 于是当最大似然距离从可乘可逆的模型中得到并且存在大量数据(当然, 并且作为其基础的概率论模型也正确) 时, 邻接法将会正确的重建任何树. 下述的例子表明, 对图8.13中简约法明显无法胜任的树, 邻接法表现得确实与最大似然法一样好.

通常情况下, 邻接法比任意概率论方法都快很多, 因为事实上它无需搜索树空间, 所以一种诱人的想法便是完全摒弃概率论. 然而如果这样做, 那么我们同时也就忽略了概率论方法在评价树之可信度以及使用模型后验概率评价模型本身之似真性方面的能力. 因此不应将邻接法或其他距离法视为概率论方法的替代, 而应该将它们看作在给定模型下生成可信树的方法. 例如, 它所提供的树很有可能成为抽样过程的好起点.

例子: 用邻接法重建树

作为邻接法成功建树的例子, 我们使用矩阵(8.29) 的替换概率按照本书第 p. 154页的方法从图8.13的树中模拟了数据. 我们使用相同矩阵推导出最大似然距离, 然后使用邻接法建树. 该过程生成三种可能无根树的次数显示如下:

邻接法的建树结果:

N	T_1	T_2	T_3
20	477	301	222
100	635	231	134
500	896	85	19
2000	995	5	0

显而易见, 在给定的大量数据下, 邻接法以较高的可靠度生成正确的树 T_1. 实际上在此特定检验中, 我们几乎没有理由偏爱最大似然法胜于邻接法. □

作为本部分的结束, 我们简单考查最大似然距离的某些特殊例子. 对 DNA 而言, Jukes-Cantor 模型给出了一个简单的距离公式, 因为练习8.7意味着 $d^{ML} = -\frac{1}{4\alpha}\log_{\mathrm{e}}(1-\frac{4f}{3})$, 其中 f 为核酸不同的位点所占比例. **Jukes-Cantor 距离**常常不用时间单位来表示, 而是按照长度 d^{ML} 上的期望替换数表示. 根据速率矩阵(8.2), 我们可知期望替换数为 $3\alpha d^{ML} = -\frac{3}{4}\ln(1-\frac{4f}{3})$.

Kimura 矩阵 (8.6) 给出了距离的紧致表达式. 在二序列联配中, Kimura [1980] 定义 Q 为颠换 (transversion) 的比例, P 为转移 (transition) 的比例. 然后令 (8.6) 中的符号 $s_t = Q/2, u_t = P$, 经过变换有 $\alpha t = -\frac{1}{2}\log(1-2P-Q) + \frac{1}{4}\log(1-2Q), \beta t = -\frac{1}{4}\log(1-2Q)$. 再根据式 (8.5), 长度为 t 的边上发生替换的期望数目为 $K = (2\beta+\alpha)t$, 于是得

$$K = (2\beta+\alpha)t = -\frac{1}{2}\log(1-2P-Q) - \frac{1}{4}\log(1-2Q).$$

其中 K 为**Kimura 距离** (Kimura distance). 可以将其推导过程表述如下: 将 (8.30) 中的对数似然写为

$$\sum_u \log P(x_u^i|x_u^j,t) = N((1-P-Q)\log r_t + Q/2\log s_t + P\log u_t + Q/2\log s_t),$$

其中 N 为已联配位点的总数. 这是出现在 Kimura 矩阵 (8.6) 中某行的概率 r_t、s_t、u_t 和 s_t 关于其相应替换类型的频率 $1-P-Q$、$Q/2$、P 和 $Q/2$ 的相对熵. 我们知道 (图11.5所示) 当这几组概率相等, 即 Kimura 方程 $s_t = Q/2$和$u_t = P$ 成立时, 相对熵达到最大.

现在, 通常来讲如果只在 t 上进行最大化, 那么我们就无法得到最大相对熵. 也许不存在同时满足上面两个方程的 t 值. 但是如果我们同时在 t 和比值 α/β 上进行最大化并保持 $\alpha+\beta$ 不变, 那么未知数个数就与方程个数相同, 从而就可以满足 Kimura 方程. 当数据量很大时, 用这种方法从数据中估计 α/β 也许是一种合理的方案, 但是当比较两条不是很长的序列时, 我们可能更倾向于将 α/β 的先验包括在内. 例如, 我们或许可以使用 gamma 函数并定义 $\tilde{K} = \mathrm{argmax}_t \max_{\alpha/\beta}\{g(\alpha/\beta,a,b)\prod_u P(x_u^i)|x_u^j,t,\alpha,\beta\}$, 其中 a,b 为适当的常量, 并且 $P(x_u^i|x_u^j,t,\alpha,\beta)$ 表示从 Kimura 矩阵得到的替换概率.

最后, 对蛋白质序列, PAM 矩阵 $S(t)$ 可被用于定义式 (8.30) 中的 $P(x_u^i|x_u^j,t)$. 我们无法给出最大值处 t 的解析表达式, 但我们却可以很容易的用梯度上升 (gradient ascent) 或某些更有效的优化技术达到这些 t 值.

练习

8.16 从最大相对熵原理 (图11.5) 出发推导 Jukes-Cantor 距离.

Sankoff & Cedergren 方法的概率论解释

如果将 Sankoff & Cedergren 算法中的分值解释为对数概率, 并且如果将该算法中的" max " 替换为"+" , 那么所得算法就将计算全似然 (full likelihood), 正如 Allison, Wallace & Yee [1992a] 所指出的一样. 树的分值 $S(\Delta_1\cdot x_{i_1}^1, \Delta_2\cdot x_{i_2}^2, \ldots, \Delta_N\cdot x_{i_N}^N)$ 将变为对祖先节点的所有残基分配求和, 而递归式 (7.6) 也将对前面的 α 求和, 因此也就是对所有可能的联配求和. 与 Sankoff & Cedergren 最初的算法相同, 这种算法

对大多数问题而言并不实际.

Hein 算法的概率论解释

正如前面所述(本书第 p. 154页), 如果将分值解释为 $\log P(x|y)$, 其中 $P(x|y)$ 为不依赖于时间的替换概率, 那么我们就可以认为简约法是全概率的 Viterbi 近似. 用这种方法导出的分值通常会为不同的残基替换产生不同的数值. 这意味着两条序列间通常仅存在唯一最优联配, 因此 Hein 的序列图将仅包含一条路径. 然而一般会存在大量只比最优稍差的次优路径. 所以在此种情况下, 简约法就会给出全概率的一个较差近似.

如果尝试用"+" 替换" max " 以便进行改进, 那么我们就不得不在序列图中包含贯穿动态规划矩阵的所有路径. 在叶节点以上的第一个节点, 该图的大小为 N^2, 而在第二高的节点该图的大小为 N^3 或 N^4, 依此类推. 很明显, 我们失去了所有已经获得的、全面但缓慢的 Sankoff-Cedergren 方法所不具备的优势.

作为折衷, 我们可以尝试选择近似最优 (near-optimal) 的路径, 以便该路径能够近似全概率同时使得序列图能够保持在可控的规模. 这种策略或许可以产生一个不错的联配或系统发育算法, 但更可能需要精巧的启发式方法以便选择路径.

8.7 补充读物

Edwards & Cavalli-Sforza [1963; 1964] 首次将最大似然法应用于系统发育领域, 他们检验了连续变量的情况, 例如某一物种骨骼特征的大小或群体中某些基因的频率. 他们使用随机游走 (random walk) 以及允许二分叉 (bifurcation) 的 Yule 过程描述了这些变量的进化历程 [Edwards 1970]. Thompson [1975] 为实现该方法设计出了计算方法, 并将其应用到某些有趣的例子上.

Felsenstein [1981a] 所写的一篇重要文章阐述了如何将最大似然法应用于离散字符, 例如序列中残基的情形. 他在这篇文章中为计算任意大小的树的似然引入了基本算法(本书第 p. 138 页), 为关于边长最大化似然给出了高效方法(本书第 p. 142 页), 并说明了如何用可逆性将问题化简为无根树情形(本书第 p. 141 页). 这为似然法在当代分子系统发育领域的广泛应用奠定了基础.

在本章和前一章中, 除了字母表的大小不一样外, 我们将 DNA 和蛋白质序列当作是类似类型的数据. 当然它们在生物学中所扮演的角色却非常不同, 这使得它们适用于不同的目的. 例如密码子第三位的快速变化可以使得我们发现最近的进化事件, 而蛋白质的更保守区域则可能携带着地球历史中早期物种形成事件的信息 [Doolittle *et al.* 1996]. 在许多情况下, 我们应该同时处理DNA 和蛋白质两个水平的问题. Goldman & Yang [1994] 已经证明展示了如何利用 Markov 模型完成这一任务, 在该模型中状态为密码子而转移概率既反映了 DNA 的替换模式, (当编码的氨基酸发生改变时) 也反映了氨基酸的属性.

系统发育的未来似乎非常有前途. 基因组科学方面激动人心的发展意味着我们会得到海量的序列数据, 并且我们很有可能将新型的序列信息应用于系统发育. 我们已经清楚的看到, 和染色体倒位 (inversion) 及其他基因组重排 (rearrangement) 一样 [Shimamura *et al.* 1997], 各种重复家族的出现可以用作系统发育学标记 (marker) [Hannenhalli *et al.* 1995]. 这样, 数据的森林 (the forest of data) 就会使我们更加清晰地考察进化树.

第九章

转换文法

到目前为止, 我们一直将生物序列看作是独立无关的一维字符串. 该假设使计算变得简便, 但在结构上却不真实. 蛋白质与核酸的三维折叠涉及到一级序列上非邻近残基间的大量物理相互作用. 能否发展蛋白质和核酸序列的概率论模型以便允许存在长程相互作用? 我们能否用这样的模型进行高效地计算? 本章将远离特定序列问题的模型, 退一步去讨论这些更理论化的问题. 我们将会见到, 如何用序列建模的更普适观点看待前面章节叙述过的许多方法.

计算语言学家为字符串建模发展了一套一般理论 [Chomsky 1956; 1959]. 该理论称为**转换文法的 Chomsky 层次结构** (Chomsky hierarchy of transformational grammars). 到目前为止, 本书使用过的大多数模型都处于 Chomsky 层次结构中从复杂度及描述能力角度而言之四种类型中的最低层次. 为了理解自然语言的结构, 人们发展了转换文法. 随后它们在理论计算机科学中的地位越发重要 [Hopcroft & Ullman 1979; Gersting 1993], 这是因为和自然语言不同, 计算机语言可以被精确地阐明为形式文法. 最近, 转换文法已经在分子生物学的序列分析问题中得以应用 [Searls 1992; Dong & Searls 1994; Rosenblueth *et al.* 1996].

文法理论在生物序列分析之更高阶结构中的一个应用实例, 是用**随机上下文无关文法** (stochastic context-free grammars , SCFG) 分析 RNA 二级结构 [Eddy & Durbin 1994; Sakakibara *et al.* 1994; Grate 1995; Lefebvre 1995; 1996]. 虽然许多计算分子生物学中的序列联配方法都隐含地符合**随机正则文法** (stochastic regular grammar), 但这些联配方法却有自己悠久的历史, 并可以独立于 Chomsky 层次结构而存在. 与之相反, 把 SCFG 应用到 RNA 二级结构的概率论建模则刚刚发展起来, 因而 RNA SCFG 的术语仍然很接近于其计算语言学根源. 为了理解 RNA 的 SCFG, 我们需要掌握计算语言学的基本内容. 这一章的主要目标是打牢基础, 以便将基于 SCFG 的概率论模型应用到 RNA 二级结构问题中. 我们以概述非概率论形式下的转换文法开始. 然后我们介绍作为形式化系统的随机文法, 它将用于带长程关联与限制的序列之全概率论建模. 最后我们给出随机上下文无关文法的广义联配算法, 下一章的 RNA 模型便是它的一个子集.

9.1 转换文法

尽管荒谬, 但"Colourless green ideas sleep furiously" (没有颜色的绿色理想猛烈地睡着) 仍是语法上正确的一句英语. 大多数说英语的读者 (读过 Chomsky 著作的除外) 在这之前从来都没有见过这个句子, 或者甚至见过其中相邻单词的任意组合. 然而他们都能认出它, 正确地分析语法, 并且能用正确的英语语调念出来.

Chomsky 对大脑或计算机程序如何在算法上判定一个新句子是否符合语法十分感兴趣. 他构造了一个称为"文法" (grammar) 的有限形式机器, 递归地枚举属于某种语言的无数个句子. 在文法理论中, "语言中包含这个句子吗?" 的问题往往被替换为"文法能产生这个句子吗?" 的问题. 第一个问题是难处理的 (句子的集合可能无限大), 但对于许多文法的有用形式而言, 第二个问题在实践中是可回答的. 回答的

好坏依赖于文法对语言限制 (constraints on the language) 建模的好坏; 即多少合乎文法的句子不能被文法产生, 以及多少不合文法的句子被文法错误地产生.

转换文法有时也叫作**生成文法** (generative grammars). 这是就生成序列而言的, 即使该模型的主要用途是识别、评价、和/或分析字符串. 在第3章, 我们已经将隐马模型描述为"发射" 序列的生成 (generative) 概率论模型. 一条给定序列是否属于某个家族是通过计算由隐马模型生成该序列的概率来推断的. 当一个隐马模型构建者谈及序列生成时, 生物学家常常发现这个概念令人困惑. 很明显是生物进化生成序列, 而不是隐马模型. 将术语"生成" 和"发射" 包含进方便的形式化体系之中, 在很大程度上应归功于 Chomsky.

转换文法的定义

一个转换文法包括若干**符号** (symbols) 与**改写规则** (rewriting rule) $\alpha \to \beta$ (有时也称为**产生式** (productions)), 其中α和β都是字符串. 这里有两种字符: 抽象的**非终端** (nonterminal) 字符, 以及在观测到的字符串中实际出现的**终端** (terminal) 字符. 上面产生式左手边的α至少应该包含一个非终端字符, 它通常要转化成产生式右边可由终端和/或非终端字符组成的新字符串. 如果我们对句子建模, 那么终端字可能是单词; 如果我们对蛋白序列建模, 那么终端字符可能是氨基酸符号. 我们将用小写字母表示终端字符, 用大写字母表示非终端字符.

了解转换文法如何工作的最简单方法是考察实例. 我们将使用两字母的终端字母表 $\{a, b\}$ 和单个非终端字符 S. 一个特殊的空白终端字符 ϵ 用于结束产生过程. 下面是一个生成包含 a 和 b 之任意字符串的转换文法:

$$S \to aS, \quad S \to bS, \quad S \to \epsilon.$$

为获得包含 a 与 b 的字符串, 我们可以从某初始字符串开始按照文法规则执行一系列转换. 为方便起见, 我们通常从特殊的起始非终端字符 S (本例中它是我们仅有的非终端字符) 开始. 选择一个可用的产生式, 使其左边有字符串 S, 同时 S 可以被该产生式右边的字符串替换. 该过程选择一个子字符串, 并根据文法允许的改写规则连续地改写, 直到字符串完全由终端字符组成且不能再被改写为止. 由此过程所得到的连续字符串称为文法的**推导** (derivation). 我们简单示例文法的推导如下:

$$S \Rightarrow aS \Rightarrow abS \Rightarrow abbS \Rightarrow abb.$$

为方便起见, 我们通常使用缩写表示法, 如 $S \to aS|bS|\epsilon$ 以表示多种可能的产生式, 其中符号 | 表示"或" . 在这个例子中, 我们能把 S 转换为三种可能的字符串.

当使用转换文法处理序列分析问题时, 我们常常想到一条特殊的序列. 问题是这条序列能否"匹配" 文法(能被文法生成). 我们反过来确定该序列的推导是否存在. 如果推导存在, 那么该序列就是由该文法建模之语言中的有效成员. 找到给定序列的有效推导称为**分析** (parsing), 此时推导称为该序列的一个**分析** (parse). 可以把分析看作是文法与序列间的**联配** (alignment)就像序列到 HMM 的 Viterbi 联配是将序列位置分配到 HMM 状态一样, 用文法对序列进行的分析本质上是将序列位置分配到文法的非终端字符.

Chomsky 层次结构

Chomsky [1959] 描述了文法改写法则上的四种限制. 由此得到的四种文法组成一个层次结构, 称为 Chomsky 转换文法的层次结构. 在下面的例子中, 我们使用 W 表示任意非终端字符, a 表示任意终端字符, α 和 γ 表示包括空字符串在内的、由非终端字和/或终端字组成的任意字符串, β 代表不包括空字符串在内的、终端字和/或非终端字组成的任意字符串.

正则文法 (regular grammar): 只允许形如 $W \to aW$ 或 $W \to a$ 的产生式法则.

上下文无关文法 (context-free grammar): 允许任意形如 $W \to \beta$ 的产生式法则. 产生式左边必须仅含有一个非终端字符而产生式右边可以是任意字符串.

上下文相关文法 (context-sensitive grammar): 产生式形如 $\alpha_1 W \alpha_2 \to \alpha_1 \beta \alpha_2$. 其中被允许的非终端字符 W 之转换依赖于上下文 α_1和α_2. 可以证明, 该规则等

价于要求右边至少包含和左边一样多的符号; 上下文相关文法的产生式不会收缩 (shrink) [Chomsky 1959]. 这使得上下文相关文法可以产生形如 $AB \to BA$ 的例子.

非限制性(片语结构) 文法 (unrestricted (phrase structure) grammar): 允许任意形如$\alpha_1 W \alpha_2 \to \gamma$的产生式.

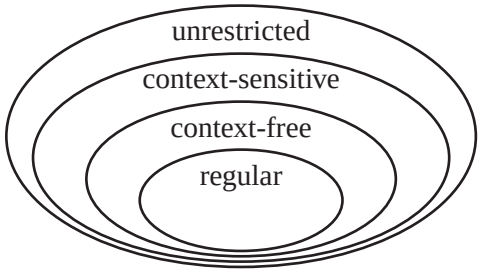


图 9.1: 转换文法的 Chomsky 层次结构, 根据文法中置于产生法则上之限制的增加而嵌套. 就其所允许的产生式而言, 正则文法是最简单、最严格的文法, 因此最容易分析. 然而, 正则文法在描述字串上"结构"约束的能力也最低.

自动机

在计算机科学中, 每个文法都对应一种抽象的计算设备, 称为**自动机** (automaton). 人们将文法描述为生成模型, 而将通常将自动机描述为能够接受或拒绝给定序列的分析器. 这里, 我们会发现自动机在两个受限用途上的有用性. 首先, 直观上讲自动机通常比与其对应的文法更容易描述和理解. 特别地, 有限状态自动机有很好的图形表示, 这比费力地枚举正则文法的改写规则更容易理解. 其次, 自动机给出了如何使用形式文法识别序列的更具体思想.

文法	分析自动机
正则文法	有限状态自动机
上下文无关文法	下推自动机
上下文相关文法	线性有界自动机
非限制性文法	图灵机

表 9.1: 分析器抽象与文法层次的关系

正则文法

正则文法中的所有产生式规则都形如 $W \to aW$ 或 $W \to a$, 其中 W 和 a 分别表示文法中的任意非终端字符和终端字符. 为了终止推导, 我们有时也允许一个额外的产生式 $W \to \epsilon$, 其中 ϵ 是空字符串. [1]本质上讲, 正则文法从左到右生成句子. 正则文法不能有效地描述终端字符之间的长程关联. 正则文法是"一级序列" (primary sequence) 模型. [2]

例子: 奇数正则文法

本章介绍的第一个文法是一个正则文法, 生成由 a 和 b 组成的任意字串: 一种相当枯燥的语言. 正则文法还有许多更有趣且令人惊奇的行为. 下面的正则文法用来

[1] $W \to \epsilon$的规则是"收缩" 的产生式, 即右边比左边短. 从技术上讲, 这使它成为一种非限制性文法规则. 然而, 可以证明正则文法总能扩展以便吸收 ϵ. 例如, 近似正则文法 $S \to aS|bS|\epsilon$ 和正则文法 $S \to aS|bS|a|b$ 等价. ϵ 产生式没有为正则文法和上下文无关文法的分析算法带来严重问题, 但在证明上却存在某些技术困难. ——原注

[2] 我们也可以允许从右到左的文法, 仅允许形如 $W \to Wx$ 或 $W \to x$ 的产生式. 它们也是正则文法. 但在同样的文法中同时允许 $W \to Wx$ 和 $W \to xW$ 则是一个上下文无关文法. ——原注

(a) Human FMR-1 mRNA sequence, fragment

```
... GCG CGG CGG CGG CGG CGG CGG CGG CGG
CGG CGG AGG CGG CGG CGG CGG CGG CGG CGG
CGG CGG AGG CGG CGG CGG CGG CGG CGG CGG
CGG CGG CTG ...
```

(b)

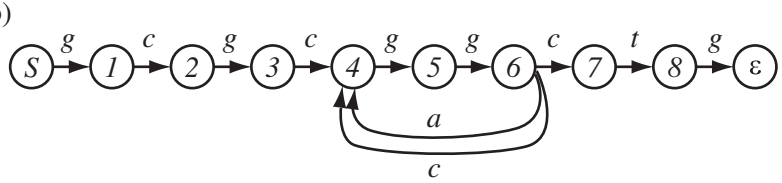


图 9.2: (a) FMR-1 三联重复区域的序列, 来自 GENBANK HSFMR1A, 访问号是 X69962. 重复区中三联体的两个变体AGG用下划线标识. (b) 识别带任意数目三联体之 FMR-1 三联重复区域的有限状态自动机. 注意这里存在一个转移, 它能接受三联体变体AGG.

生成仅含 a 和 b、且有奇数个 a 的字符串 [Searls 1992]:

从S 开始,
$S \to aT|bS,$
$T \to aS|bT|\epsilon.$

当一个字串含有奇数个 a 时, 这个推导处于非终端字符 T; 而当含有偶数个 a 时, 它处于非终端字符 S. 因为只能从非终端字符 T 终止, 所以它只能获得含奇数个 a 的字符串. □

有限状态自动机

对应于正则文法的分析自动机是**有限状态自动机** (finite state automaton). 第2章中我们看到有限状态自动机可以作为二序列联配算法的一般模型. 我们现在对其进行更一般的考察. 有限状态自动机每次从输入序列中读入一个字符. 如果这个字符被接受, 自动机就进入一个新的状态; 或者如果没有接受, 自动机就停止并拒绝字符串. 如果这个自动机到达最终的"接受" (accepting) 状态, 那么这个输入的字符串就已经被自动机成功地分析和识别.

有限状态自动机是由许多**状态** (state) 组成的模型, 这些状态由**状态转移** (state transition) 互相连接起来. 状态及状态转移对应于非终端字符即等价的正则文法产生式. 人们通常抽象地表示有限状态自动机, 用圆圈表示状态, 箭头表示转移.

例子: FMR-1三联重复区域

人的 FMR-1 基因序列中包括一个三联重复区域, 其中序列CGG会重复若干次. 该三联体的数目在个体之间差异显著, 其拷贝数的增加与脆性X 综合症 (frigile X syndrome) 密切相关. 这是一种引起智力低下及其他症状的遗传病, 每2000个儿童中就有一例. 图9.2所示的有限状态自动机对 FMR-1 的CGG重复区域进行了简洁地建模, 通过一个环状转移回到一个新的CGG上.

为检验一条序列是否符合对 FMR-1 中对CGG重复的描述, 该序列每次向自动机输送一个字符. 如果第一个字符是G, 那么自动机就进入状态1; 否则退出并拒绝该序列. 如果自动机处于状态1并且读到一个字符C, 那么它就顺利地移动到状态2, 以此类推, 直到达到结束状态E且再没有字符需要检查为止, 此时自动机便顺利地识别了序列.

有限状态自动机可以匹配来自"语言" 的任意字符串, 该语言包含GCG CGG CTG, CGG CGG CGG CTG, GCG CGG CGG CGG CTG, 以及任意数目的CGG之拷贝. 与此有限

状态自动机等价的正则文法是:

$$
\begin{array}{ll}
S \to gW_1 & W_5 \to gW_6 \\
W_1 \to cW_2 & W_6 \to cW_7|aW_4|cW_4 \\
W_2 \to gW_3 & W_7 \to tW_8 \\
W_3 \to cW_4 & W_8 \to g \\
W_4 \to gW_5 &
\end{array}
$$

□

Moore 机 vs. Mealy 机

在图9.2的FMR-1 自动机中, 终端字符与自动机的转移相关联. 在转移上接受字符的有限自动机叫作**Mealy 机** .与之相对,在第3章提到的隐马模型中, 我们把状态与终端字符相关联, 并把发射事件的字符与状态转移的事件分离. 这种在状态上接受字符的有限自动机叫作**Moore 机**这两种类型的机器可以相互转换, 例如我们可以把 FMR-1 自动机的一个G标记为状态1, 有这个状态就接受G, 而不是转移到这个状态. 在 Mealy 机中, 对应于 FMR-1 自动机状态1的文法产生式是 $S \to gW_1$, 而在 Moore 机中它可以写作 $S \to \hat{W}_1, \hat{W}_1 \to gW_1$, 其中 $\hat{W}_1$ 是一个额外的中间非终端字符. (因为这两种形式是等价的, 所以我们就不太需要关心在 Moore 机形式下的法则 $S \to \hat{W}_1$ 并不严格符合正则文法的规则.)

确定性自动机 vs. 非确定性自动机

FMR-1 自动机是一种**非确定性** (nondeterministic) 有限自动机. 当自动机处于状态6且下一个输入字符是C 时, 自动机接受C并且移动到状态4**或**状态7. 在**确定性** (deterministic) 有限自动机中, 对任意状态和任意输入字符而言, 仅有一种可能的接受转移. 可以证明, 任意非确定性有限自动机可以转化为确定性有限自动机.

用确定性有限自动机做分析是极为高效的. 快速的 BLAST 数据库搜索程序 [Altschul *et al.* 1990] 之核心中就使用确定性有限自动机算法. 非确定性有限自动机分析算法必须在拒绝一条序列前检查所有可能的路径, 但它仍然十分高效. GREP、SED、AWK 和 VI 中的 UNIX 的文本模式匹配工具都实现了高效的非确定性有限自动机; UNIX 的"正则表达式" (regular expression) 即等价于正则文法.

练习:

9.1 把图9.2中的 FMR-1 自动机转换为一个 Moore 机, 用状态接受特定字符, 而不是用转移接受字符.

9.2 把 FMR-1 自动机转换为一个确定性自动机.

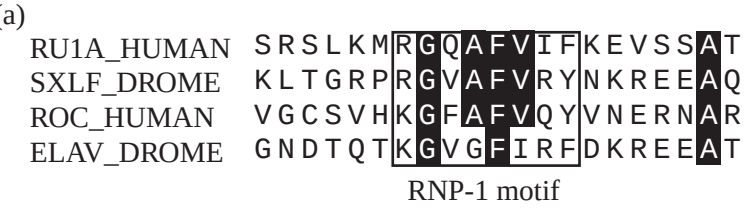
PROSITE 模式

PROSITE 数据库是正则文法应用在生物学上的一个非常优秀的例子, 它由日内瓦的 Amos Bairoch 及其同事编写 [Bairoch, Bucher & Hofmann 1997]. 每条 PROSITE 记录包括一个序列模式, 表示某蛋白家族的所有或几乎所有成员共有的一个高度保守特征模体. 与联配采用的计分方法不同, PROSITE 模式要么匹配一条序列, 要么就不匹配; 它们是用有限状态自动机进行序列匹配的正则文法.

一个 PROSITE 模式包含若干模式元素, 它们由破折号分开并用句点作为结尾. 在模式元素中, 字母表示氨基酸的单字母代码; 方括号表示其内任意一种残基都允许出现; 花括号表示**除了**其中的残基外, 其他的残基都被允许; x表示在这个位置上, 任意残基都能出现. 长度或长度范围在圆括号里给出, 如-x (4)-匹配四个任意类型的残基, 而-x (2,4)-匹配二、三或四个任意类型的残基. 图9.3给出了1995年2月版的 PROSITE 数据库中1029个模式里的一条作为例子.

任意 PROSITE 模式都是一个正则文法, 且能与一个非确定性有限状态自动机相对应. PROSITE 模式的语法 (syntax) 与标准正则表达式的语法相似. 某些常用的 PROSITE 模式搜索工具使用 UNIX GREP 作为其搜索引擎, 首先把 PROSITE 模式转换成一个 UNIX 正则表达式, 然后 GREP 为该表达式建立自动机.

(a)



(b)

```
[RK]-G-{EDRKHPCG}-[AGSCI]-[FY]-[LIVA]-x-[FYM].
```

图 9.3: (a) 多序列联配的一部分展示了 RNA 结合蛋白某主家族的高度保守序列模体" RNP-1" . (b) RNP-1 的 PROSITE 模式 PS00030.

例子: PROSITE 模式的正则文法形式

下面给出了与图9.3中 PROSITE 的 RNP-1 模式所对应的正则文法. 我们用初始非终端字符S和8个非终端字符$W_1, \ldots, W_8$ 对应保守模体的8个位置. 为简单起见, 某些产生式使用方括号, 就像 PROSITE 里描述的一样: 例如$[ac]W$表示$aW|cW$.

$$
\begin{aligned}
S &\to rW_1|kW_1 \\
W_1 &\to gW_2 \\
W_2 &\to [afilmnqstvwy]W_3 \\
W_3 &\to [agsci]W_4 \\
W_4 &\to fW_5|yW_5 \\
W_5 &\to lW_6|iW_6|vW_6|aW_6 \\
W_6 &\to [acdefghiklmnpqrstvwy]W_7 \\
W_7 &\to f|y|m
\end{aligned}
$$

□

练习

9.3 重要的 DNA 结合蛋白模体 C2H2 锌指 (zink finger) 的 PROSITE 模式为`C-x(2,4)-C-x(3)-[ LIVMFYWC ]-x(8)-H-x(3,5)-H`. 请画出接受该模式的有限自动机.

正则文法不能做什么

当以下两种情况发生时, 我们就会遇到不能用正则文法描述之语言 L 的两个经典例子 [Chomsky 1956]:

(i) L 包含所有形如 aa、bb、$abba$、$abaaba$ 等的字串, 它们正向读和反向读都一样(即回文 (palindrome) 语言).

(ii) L 包含所有形如 aa、$abab$、$aabaab$ 等的字串, 它们由相同的两部分组成 (即复制 (copy) 语言).

正则文法可以生成回文串作为其语言的一部分. 关键是, 正则文法不能有效地生成**仅含**回文的语言, 因此它不能区分正确的回文和非回文. 我们需要比正则文法更复杂的文法, 以便描述在语言中文法串的更多特殊限制.

如图9.4所示, 回文语言的相互作用是**嵌套的** (nested), 即相互作用的连线不交叉; 而在复制语言中则可以出现交叉的相互作用. 此区别在确定语言的生成文法类型时有十分重要的作用.

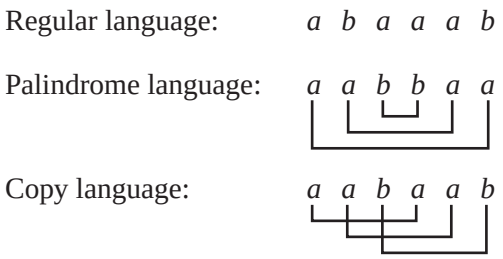


图 9.4: 与正则语言不同, 回文和复制语言在较远位置间存在关联. 连线表示回文和复制语言中字符串上的相关位置.

9.2 上下文无关文法

我们可以使用处于 Chomsky 层次结构中下面一级的**上下文无关文法** (context-free grammars , CFG) 处理回文语言. 很明显, 在计算生物学中很少出现分析句子"Doc , note . I dissent . A fast never prevents a fatness . I diet on cod ." [3]的问题. 我们仔细考察上下文无关文法的原因是, 如下面的例子所示, RNA 二级结构是一种回文语言. RNA 二级结构中存在这样的问题: 只要某些嵌套位置对之间保持碱基对强关联, 那么序列本身是什么就并不重要.

上下文无关文法允许额外的法则, 让文法在终端字符间产生嵌套的长程配对关联. 产生式法则的左边必须仍是一个非终端字符, 而右边则可以是终端字和非终端字的任意组合. 因此右边可以从一个单独非终端字符产生一个相关碱基的配对, 而不像正则文法产生式那样必须从两个不同的非终端字符独立地产生一个符号对. 可以生成回文语言的一个 CFG 例子是:

$$S \rightarrow aSa|bSb|aa|bb$$

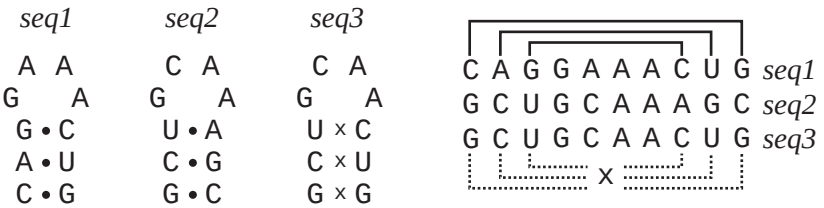
从这个 CFG 出发的回文"aabaabaa" 之推导是:

$$S \Rightarrow aSa \Rightarrow aaSaa \Rightarrow aabSbaa \Rightarrow aabaabaa.$$

正则文法从左到右生成字符串, 而上下文无关文法则从外到内生成字符串. 这种由外到内的生成方式使得我们只能获得嵌套关联. 复制语言的交叉关联(图9.4) 破坏了这种嵌套约束, 所以复制语言不是上下文无关语言.

例子: RNA 茎环 (stem loop) 结构的上下文无关文法

在下图中, *seq*1 和 *seq*2 虽然在序列上不同, 但却共享相同的模式对 (`A-U`和`C-G`), 因此可以折叠成相同的 RNA 二级结构. 而 *seq*3 虽然前半部分和 *seq*1 相同而后半部分和 *seq*2 相同, 但却不能折叠为类似的结构. 一致 RNA 二级结构需要一组像回文语言那样的嵌套配对限制, 除非相关配对是互补的而非相同的.



像*seq*1和*seq*2那样带三个碱基配对和一个`GCAA`或`GAAA`环的 RNA 茎环, 可以用

[3]这句英语回文由 Peter Hilton 提供, 他是二战时期破解德军密码 (the German Enigma code) 的英国密码队成员. ——原注

CFG 建模为:

$$
\begin{aligned}
S &\to aW_1u|cW_1g|gW_1c|uW_1a,\\
W_1 &\to aW_2u|cW_2g|gW_2c|uW_2a,\\
W_2 &\to aW_3u|cW_3g|gW_3c|uW_3a,\\
W_3 &\to gaaa|gcaa.
\end{aligned}
$$

□

练习

9.4 用上下文无关文法写出上例中 $seq1$ 和 $seq2$ 的推导.

9.5 写出一个可以产生上例中 $seq1$ 和 $seq2$ 但不能产生 $seq3$ 的正则文法.

9.6 考虑上例中用 CFG 生成的完全语言. 描述一个正则文法, 使它产生完全相同的语言. 用正则文法描述该序列家族是否明智?

分析树

上下文无关文法到序列的联配(也就是分析) 有一种优雅的表现形式, 称为**分析树** (parse tree). 其树根是起始非终端字符 S. 叶节点是序列的终端字符. 中间节点是非终端字符. 中间节点的子节点是其非终端字符的产物, 且从左到右排列.

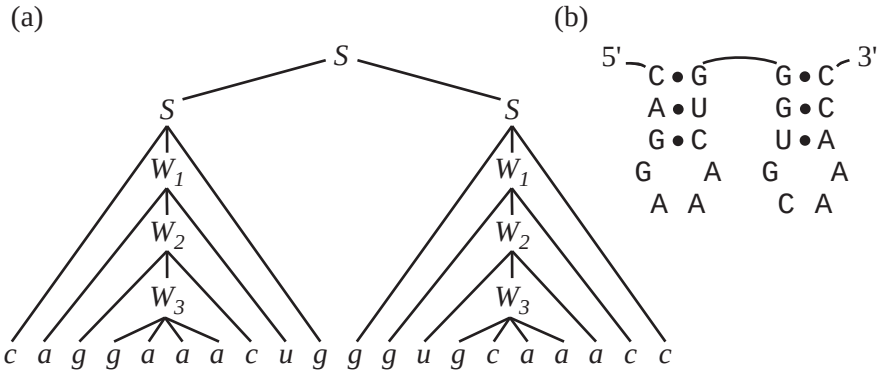


图 9.5: (a) CAG GAA ACU GGG UGC AAA CC的一棵分析树以及茎环文法, 我们用法则 $S \to SS$ 扩展它以便构造一棵更有趣的树. (b) 相同序列的 RNA 二级结构, 它与其分析树表示间存在密切的对应.

分析树中以某中间节点为根的片段称为**子树** (subtree). 任何子树都可推导出该观测序列的一条连续片段. 这个性质很重要. 它允许算法为序列构建最优分析树, 其方法是对越来越长的子序列递归地构建越来越大的最优分析子树. 图9.5示例了 CFG 和某小 RNA 的分析树.

例子: PROSITE 模式的分析树

正则文法是上下文无关文法的一个子集. 因此, 正则文法与序列间的联配也可用分析树表示. 图9.6为图9.3中 RNP-1 的 PROSITE 模式显示了正则文法分析树. 联配与分析树之间的对应关系一目了然. □

下推自动机

为 CFG 作分析的自动机叫作**下推自动机** (push-down automaton). 有限状态自动机除了记录当前状态外不需要内存, 而下推自动机用下推栈 (stack) 的形式为符号保留有限的内存.[4]

[4]下推栈是一种后进先出 (last in , first out) 的数组或列表. 元素在栈的顶部"压入" (push) 和"弹出" (pop), 像一堆盘子一样. ——原注

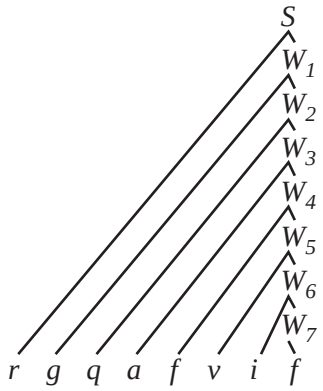


图 9.6: 使用本书第 **??**页中的正则文法为 RNP-1 模体RGQAFVIF建立的分析树. 正则文法是上下文无关文法的线性特殊情况, 因此正则文法的分析树本质上就是文法非终端字符到序列终端字符上的一个标准线性联配.

下推自动机根据如下算法从左到右分析序列. 对自动机的栈进行初始化时, 压入起始非终端字符. 接下来的步骤是迭代直到处理完所有的输入字符. 当没有输入字符余下时, 如果栈是空的, 那么这条序列就已经分析成功.

算法: 用下推自动机分析
从栈里弹出一个符号.
如果弹出的符号是非终端字符:

- 在输入中从当前位置出发向后取字符, 并为非终端字符选择一个有效的生成规则. 对于确定性下推自动机, 至多有一个可能的选择. 对于非确定性自动机, 所有可能的选择都需要单独评估. 如果不存在有效的产生式, 那么终止算法并拒绝该序列.

- 从最右边的字符开始, 把所选产生式法则的右边压入栈中.

如果弹出的字符是终端字符:

- 比较该字符和输入的当前符号. 如果匹配, 那么将自动机移动到输入的右边(输入符号被接受). 如果不匹配, 那么终止算法并拒绝该序列. ◁

对非确定性上下文无关文法而言, 下推自动机不是有效的识别器. 所有有效的自动机移动都必须经过彻底尝试, 直到要么输入字符串被成功接受, 要么不再有更多的移动可以被尝试为止. 尽管我们可以使用许多不太复杂的非确定性 CFG 通过这种蛮力 (brute-force) 算法识别字符串, 但需要检验的不同推导可能会导致组合爆炸 (combinatorial explosion). 本章的后面部分, 我们会为上下文无关文法描述更复杂深奥的多项式时间算法: Cocke-Yonger-Kasami (CYK) 分析算法.

例子: 用下推自动机分析 RNA 的茎环结构

现在用本书第 168页中三碱基配对的 RNA 茎环结构之上下文无关文法分析序列`GCC GCA AGG C`. 下表显示了分析序列时发生在自动机栈中的一系列操作. 自动机在输入 (左列) 中的位置用方框表示. 表中也包括下推栈中的符号 (中间列), 左侧为栈顶. 基于输入中的当前位置和当前栈, 该表还描述了下一步自动机操作 (右列). 为简单起见, 非终端字符用其下标数字表示, 例如数字1表示W_1等等.

输入序列	栈	在栈和输入上的自动机操作
G CCGCAAGGC	S	弹出S. 读取输入; 使用产生式$S \to g1c$.
G CCGCAAGGC	$g1c$	弹出g. 接受g; 输入右移
G C CGCAAGGC	$1c$	弹出1. 读取输入; 使用产生式$1 \to c2g$.
G C CGCAAGGC	$c2gc$	弹出c. 接受c; 输入右移.
GC C GCAAGGC	$2gc$	弹出2. 读取输入; 使用产生式$2 \to c3g$.
GC C GCAAGGC	$c3ggc$	弹出c. 接受c; 输入右移.
GCC G CAAGGC	$3ggc$	弹出3. 读取输入; 使用产生式$3 \to gcaa$.
GCC G CAAGGC	$gcaaggc$	弹出g. 接受g; 输入右移.
	⋮	(几次接受操作)
GCCGCAAGG C	c	弹出c. 接受c; 输入右移.
GCCGCAAGGC	$-$	栈清空. 输入字符串为空. 接受整条字符串.

□

练习

9.7 修改下推自动机算法, 使它能随机**产生**上下文无关文法语言中一条可能的有效序列.

9.3 上下文相关文法

虽然最初看来复制语言并不比回文语言复杂, 但复制语言不是上下文无关语言. 通常来讲, 复制语言需要上下文相关文法. 即使想要产生复制语言的简单例子, 其上下文相关文法也很复杂. 例如考虑由字符串 cc、$acca$、$abaccaba$ 或 $bbabccbbab$ 组成的复制语言; 即所有字符串都包含由 a 和 b 组成之字符串的两个拷贝, 并且拷贝间有一对 c. 生成该语言的上下文相关文法是:

起始:

$S \to CW$

产生非终端字符:

$W \to A\hat{A}W \mid B\hat{B}W \mid C$

重排非终端字符:

$\hat{A}B \to B\hat{A}$

$\hat{A}A \to A\hat{A}$

$\hat{B}A \to A\hat{B}$

$\hat{B}B \to B\hat{B}$

产生终端字符:

$CA \to aC$

$CB \to bC$

$\hat{A}C \to Ca$

$\hat{B}C \to Cb$

终止:

$CC \to cc$

我们有7个非终端字符, S、A、$\hat{A}$、B、$\hat{B}$、C 和 W. 其中 A 和 $\hat{A}$ 用来生成字符 a (同理, B 和 $\hat{B}$ 用来生成字符 b, C 用来生成 c). 非终端字符 A 和 B 生成字符串的左半边, 而 $\hat{A}$ 和 $\hat{B}$ 用来生成字符串的右半边.

在复制语言中, 上下文相关文法并不能直接产生字符间的交叉成对相互作用. 相反, 它用非终端字符 W 产生带非交叉相互作用的字符对, 然后文法通过检查它们的局部上下文以适当重排这些非终端字符. 重排规则交换非终端字符, 把带尖号之非终端字符移动到不带尖号之非终端字符的右边. 因为在推导过程中只要出现产生式法则的左边符号, 我们就能使用该产生式, 所以文法需要仔细构造, 以便直到非终端字符都排妥后才开始产生终端字符.

下面是一个字串 $aabccaab$ 的文法推导:

$$S \Rightarrow CW \Rightarrow CA\hat{A}W \Rightarrow CA\hat{A}A\hat{A}W \Rightarrow CA\hat{A}A\hat{A}B\hat{B}W \Rightarrow CA\hat{A}A\hat{A}B\hat{B}C$$
$$\Rightarrow CAA\hat{A}\hat{A}B\hat{B}C \Rightarrow CAA\hat{A}B\hat{A}\hat{B}C \Rightarrow CAAB\hat{A}\hat{A}\hat{B}C \Rightarrow CAAB\hat{A}\hat{A}Cb$$
$$\Rightarrow CAAB\hat{A}Cab \Rightarrow CAABCaab \Rightarrow aCABCaab \Rightarrow aaCBCaab$$
$$\Rightarrow aabCCaab \Rightarrow aabccaab.$$

上下文相关文法的分析自动机是**线性有界自动机** (linear bounded automaton). 这种机制从后向前系统地操作观测字符串的所有可能推导, 直到某个推导到达起始非终端字符, 或者穷尽所有可能的推导都不能找到有效的推导为止. 因为上下文相关文法是受限的, 所以产生式法则的左边不能比右边长, 需要检验的可能推导数目必须有限. 推导过程的中间产物不可能比观测序列本身更长. 在计算机科学的教科书中, 线性有界自动机被描述为抽象的线性内存"磁带" (tape) 以及一个读/写头; 术语"有界" 是指所需磁带的总量保证小于等于观测字符串的长度. 尽管如此, 所有可能的推导数目也随序列长度呈指数增长. 至今仍不存在用于分析上下文相关文法的普适多项式时间算法. 当考虑上下文相关文法的任何实际应用时, 我们需要密切关注这一点. 此时必须使用诸如模拟退火等近似算法.

非限制性文法与图灵机

对非限制性文法, 其产生式法则的左边和右边都可以是任意的符号组合. 与之等价的分析自动机是图灵机. **不**存在普适的算法以保证能在有限时间内确定某字符串是否可以从非限制性文法中得到有效推导. 直观上看, 这是因为产生式的右侧能够收缩为更少的字符. 在从后向前操作图灵机的可能推导时, 中间字符串可以变得比输入更长, 因此可能的推导之数目可以无限增长. 相反, 在上下文相关文法推导中, 中间字符串的数目必定是有限的, 因为在线性有界自动机的磁带上, 当自动机从后向前操作、接近可能的结论时, 中间字符串只会变得越来越小. 在计算机科学中图灵机的性质有很大的理论价值, 但缺少能确保其停止的算法, 这使得非限制性文法对实际应用没有吸引力, 除非这些文法有更受限的特殊情况. 许多可以描述为非限制性文法的问题, 例如前面讨论过的上下文相关文法和 NP 问题, 都已经被人们转换为最优化问题并以非精确方式用 (例如) 模拟退火方法完成"分析" 过程.

9.4 随机文法

仔细考虑 PROSITE 模式, 我们会发现在计算生物学中使用简单有限自动机的缺陷. 随着更多序列的确定和家族成员的增加, 创建一个特殊模式也就越发困难. 模式规则的例外情形可能发生在任意位置. 例如, SRP55蛋白SR55_DROME是与果蝇mRNA 剪接有关的另一个 RNA 结合蛋白, 它的 RNP-1 模体是NGYGFVEF. 第一个N不符合 PROSITE 模式, 因为该位置要求R或K. 现在这个模式不得不修改为允许N. 但由于例外的积累和模式的放松, 这个模式的特异性已经减弱. 结果, 信息太少可能使得我们匹配到不相关的随机序列. 对于某些多变的蛋白家族, 已经证明不可能获得具有辨别能力的 PROSITE 模式. 合乎逻辑的解决办法是允许例外的存在, 但与等同地考虑所有可能不同, 给例外的计分应该比一致序列的强匹配要低. 这一思想导致了随机 (概率论的) 正则文法, 就像序列 profile (第5章) 与隐马模型(第3章)一样.

Chomsky 层次结构中的任意文法都能以其随机形式用于作为序列概率论建模系统的基础. 随机文法模型 θ 以概率 $P(x|\theta)$ 产生不同的字符串 x, 而非随机文法要么产生一个字符串 x 要么什么都不产生.

在随机正则文法和随机上下文无关文法中, 从任意给定非终端字符出发的所有可能产生式之概率和均为1. 所得的随机文法在序列 x 上定义了一个概率分布, 即 $\sum_x P(x|\theta) = 1$. 例如对 PROSITE 例子中的第一个产生式法则 $S \to rW_1|kW_1$, 某随机正则文法可能为每个产生式均赋予概率0.5:

$$\underset{(0.5)}{S \to rW_1}, \quad \underset{(0.5)}{S \to kW_1}.$$

通过为例外情况赋予很低但非零的概率, 随机正则文法就能容许例外, 但这在整体上并不会降低对更可信模体的识别能力. 例如对SR55_DROME中 RNP-1 模体第一个位置的非一致N, 可以用如下的产生式法则建模:

$$\underset{(0.45)}{S \to rW_1}, \quad \underset{(0.45)}{S \to kW_1}, \quad \underset{(0.10)}{S \to nW_1}.$$

如果产生式法则允许所有可能符号 (所有20个氨基酸) 都带有概率, 并且我们将此文法设计为能生成任意长度的序列, 那么被随机文法指定的语言就包括**所有**可能的字符串而不只是它们的一个子集. 因此随机文法可以用于在所有无限序列的空间上指定概率分布.

随机上下文相关或非限制性文法

本书不会更详细地探讨随机上下文相关文法或随机限制性文法, 因为我们对其在计算生物学中的实际应用一无所知. 尽管如此, 应该注意到相比于正则文法和上下文无关文法的描述, 我们必须更仔细地描述上下文相关文法和非限制性文法在随机形式下的产生式法则. 同一个非终端字符 W 在不同的上下文中可能有不同的产生式法则, 并且这些上下文也不一定唯一. 例如考虑带概率 $p_1, \ldots, p_5$ 的上下文相关文法 $S \to aW, S \to bW, bW \to bb, W \to a, W \to b$. 由这个文法生成的语言是 $\{aa, ba, ab, bb\}$, 它带有概率 $\{p_1p_4, p_1p_5, p_2p_4, (p_2p_3 + p_2p_5)\}$. 从代数上容易证明, 简单地要求 S 和 W 的产生式之和是1, 也就是 $p_1 + p_2 = 1, p_3 + p_4 + p_5 = 1$, 并不能给出一个该语言上的概率分布, 除非出现 $p_1 = 0$ 或 $p_3 = 0$ 的特殊情况. 可以这样解决此问题, 首先重排文法, 使得某非终端字符的上下文能唯一地确定一组可能的产生式法则, 且非终端字符都不可能有多个左边形式. 然后在给定的上下文中为非终端字符的转换设置概率并使其和为1, 从而形成一个随机文法. 例如上面的文法可被改为 $S \to aW, S \to bW, bW \to bb, bW \to ba, aW \to aa, aW \to ab$ 分别带概率 $p_1, \ldots, p_6$, 于是现在的条件 $p_1 + p_2 = 1, p_3 + p_4 = 1, p_5 + p_6 = 1$ 便给出一个合适的随机文法.

隐马模型是随机正则文法

隐马模型等价于随机正则文法. 仅有的不同是这两种模型在传统上的表示不同. HMM 通常描述为 Moore 机, 在状态上发射字符, 并且独立于转移. 而随机正则文法产生式则对应于 Mealy 机, 当转移到新的非终端字符时发射终端字符 (即产生式形如 $W_1 \Rightarrow aW_2$). 在前面的章节我们看到, Moore 机和 Mealy 机可以相互转化. 例如, 如果某一 HMM 状态有 N 个到新状态的转移且每个新状态均发射 M 个字符中的一个, 那么该状态也就可以建模为 NM 个随机正则文法产生式. 因此用于联配、计分和训练随机正则文法的算法, 均与用于隐马模型的算法相同 (第3章).

练习

9.8 G-U 配对在 RNA 茎的碱基配对上能够被接受, 但比 G-C 和 A-U 的 Watson-Crick 配对发生频率要低. 把 RNA 茎环结构的上下文无关文法 (本书第 168 页) 改写为随机上下文无关文法, 并允许在茎部的 G-U 配对概率是 Watson-Crick 配对概率的一半.

9.9 扩展本书第 169 页的下推自动机算法, 并根据其相应的概率由随机上下文无关文法生成序列. (注意: 这为从任意 SCFG 中对序列进行抽样提供了一种高效算法, 包括下章中更复杂的 RNA SCFG.)

9.10 考虑一个简单的 HMM, 它为 DNA 中的两种碱基组分建模. 该模型的两个状态完全由四种状态转移相互连接. 状态1以概率 $(p_a, p_c, p_g, p_t) = (0.1, 0.4, 0.4, 0.1)$ 发射富含 CG 的序列, 状态 2 以概率 $(p_a, p_c, p_g, p_t) = (0.3, 0.2, 0.2, 0.3)$ 发射富含 AT 的序列. (a) 画出这个 HMM. (b) 设定转移概率以便状态 1 的运行期望长度为 1000bp, 状态 2 的运行期望长度是 100bp. (c) 以正则文法形式给出相同的模型, 并指明终端字符、非终端字符以及带相关概率的产生式法则.

9.5 用于序列建模的随机上下文无关文法

我们现在可以写下作为序列模型的随机上下文无关文法. 然而这只是对序列分析问题创建有用的概率论建模系统的第一步. 就像 HMM 一样, 我们也必须用算法处理下面三个问题:

(i) 计算从序列到参数化之随机文法的最优联配(联配问题).

(ii) 计算给定参数化之随机文法的条件下序列的概率(计分问题).

(iii) 给定一组示例序列或结构, 估计未参数化之随机文法的最优概率参数(训练问题).

在第3章, 我们已经知道了每个问题在隐马模型 (因此也就是随机正则文法) 下的解. Viterbi 算法解决联配问题. 前向-后向算法中的前向步骤解决计分问题. Baum-Welch 期望最大化方法中应用的前向-后向算法解决训练问题. 对随机上下文无关文法, 类似的动态规划算法也存在.

随机上下文无关文法的标准形式

CFG 对改写规则右侧的符号串变化不做限制. 为了表述一般的 CFG 分析算法, 对改写规则采用限制性的"标准形式" 是很有用的. 其中一种标准形式是**Chomsky 标准型** (Chomsky normal form). Chomsky 标准形要求所有 CFG 产生式法则具有形式$W_v \to W_y W_z$或$W_v \to a$. 把非一致的改写规则扩展为一系列从额外非终端字符出发的标准形式产生式后, 我们可以将任意 CFG 重构为 Chomsky 标准型. 因此, 适用于 Chomsky 标准形式的 CFG 分析算法一般来讲对任意 CFG 都适用. 例如, 本书第 **??** 页中回文 CFG 的生成规则 $S \to aSa$ 可以在 Chomsky 标准形式下展开为 $S \to W_1 W_2, W_1 \to a, W_2 \to SW_1$.

练习

9.11 将生成规则 $W \to aWbW$ 转换为 Chomsky 标准形. 设原产生式的概率是 p, 求标准形式下的产生式概率.

9.12 将本书第 168页的 RNA 茎结构模型文法中的产生式法则 $W_3 \to gaaa|gcaa$ 转换为 Chomsky 标准形. 假设 $W_3 \to gaaa$ 有概率 p_1, $W_3 \to gcaa$ 有概率 $p_2 = 1 - p_1$, 请为你的标准形式产生式分配概率. 证明 `GAAA` 环和 `GCAA` 环的标准形式的正确概率分别是 p_1 以及 p_2.

内向算法

用于 Chomsky 标准形式下 SCFG 的内向-外向算法 [Lari & Young 1990] 是用于 HMM (第三章) 中的前向-后向算法的自然对应物. 给定 SCFG 后, 内向算法计算一条序列的概率 (分值), 如同用于 HMM 的前向算法一样. 最好的内向算法变体是Cocke-Younger-Kasami (CYK) 算法, 它用于寻找从 SCFG 到序列的最大概率联配, 就像用于 HMM 的 Viterbi 算法一样. 像前向-后向算法一样, 内向-外向是一种递归的动态规划算法, 但内向-外向的计算复杂度要大得多.

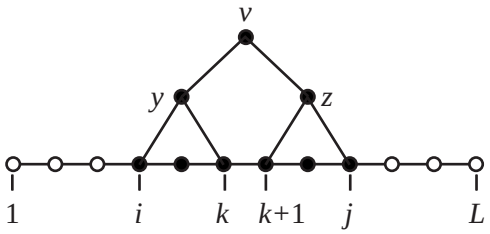


图 9.7: $\alpha(i, j, v)$的内向计算迭代步骤图解, 它是从 i 到 j 的子序列之以状态v为根节点的分析子树概率. 该计算递归地进行, 即对所有的 y、z 和 k, 对状态 y 和 z 以及更短的子序列 i 到 k 和 $k+1$ 到 j 的、由转移概率 $v \to yz$ 加权的分析子树求和.

下面定义记号. 考虑含有 M 个不同非终端字符 $W = W_1, \ldots, W_M$ 的 Chomsky 标准形式 SCFG . 起始非终端字符是 W_1. 令 u, y, z 是非终端字符 W_v, W_y, W_z 的下标. 产生式法则形如 $W_v \to W_y W_z$ 和 $W_v \to a$ (其中 a 是终端字母表中的符号). 令这些产生式的概率参数分别为 $t_v(y, z)$ 和 $e_v(a)$ (对**转移**和**发射**而言). 序列 x 有 L 个符号, 用 $x_1, \ldots, x_L$ 表示. 令 i、j 和 k 分别表示序列 x 中符号 x_i、x_j 和 x_k 的下标.

对所有的 i、j 和 v, 内向算法为子序列 $x_i, \ldots, x_j$ 计算以非终端字符 W_v 为根的分析子树之概率 $\alpha(i,j,v)$ [Lari & Young 1990]. 此计算需要 $L \times L \times M$ 的三维动态规划矩阵. 该计算从长度为1的子序列 ($i = j$) 开始, 然后是长度为2的子序列, 如此向外递归地计算越来越长的子序列, 直到能够确定以起始非终端字符为根的完全分析树之概率为止. 该算法的递归本质在图9.7中给出. 形式上, 内向算法为:

算法: 内向

起始: 从 $i = 1$ 到 L, $v = 1$ 到 M:

$\alpha(i,i,v) = e_v(x_i)$.

迭代: i 从 $L-1$ 减小到1, j 从 $i+1$ 增加到 L, v 从 1 增加到 M:

$\alpha(i,j,v) = \sum_{y=1}^{M} \sum_{z=1}^{M} \sum_{k=i}^{j-1} \alpha(i,k,y)\alpha(k+1,j,z)t_v(y,z)$.

终止:

$P(x|\theta) = \alpha(1,L,1)$. ◁

因此内向算法可以用SCFG 计算序列的概率(分值). 它的内存复杂度是 $O(L^2M)$, 这很明显来自 α 的3个下标. 该算法的时间复杂度是 $O(L^3M^3)$, 这很明显来自递归循环的三个序列位置下标 i,j,k 以及三个文法非终端字符下标 v,y,z.

外向算法

对所有 i、j 和 v, 外向算法计算概率 $\beta(i,j,v)$, 即**排除** (excluding) 所有子序列 $x_i, \ldots, x_j$ 之以 W_v 为根的分析子树外, 整条序列 x 之以起始非终端字符为根的完全分析树概率 [Lari & Young 1990]. 和内向算法一样, 该计算在 $L \times L \times M$ 的三维矩阵中进行. 计算外向概率 $\beta(i,j,v)$ 需要此前内向计算的结果 $\alpha(i,j,v)$. 外向算法从最大的排除 (excluded) 子序列 $x_1, x_2, \ldots, x_L$ 开始, 向内递归. 外向算法的图示见图9.8. 其形式描述如下:

算法: 外向

起始:

$\beta(1,L,1) = 1$;

$\beta(1,L,v) = 0$, 从$v = 2$到M.

迭代: s 从 $L-1$ 减小到 1, j 从 s 增加到 L, v 从 1 增加到 M, 令 $i = j - s + 1$:

$$\beta(i,j,v) = \sum_{y,z} \sum_{k=1}^{i-1} \alpha(k,i-1,z)\beta(k,j,y)t_y(z,v) + \sum_{y,z} \sum_{k=j+1}^{L} \alpha(j+1,k,z)\beta(i,k,y)t_y(v,z).$$

终止:

$P(x|\theta) = \sum_{v=1}^{M} \beta(i,i,v)e_v(x_i)$, 对所有 i. ◁

使用期望最大化的参数重估计

通过期望最大化方法, 内向变量 α 和外向变量 β 可用于重新估计 SCFG 的概率参数, 就像通过 EM 方法在 HMM 训练中使用前向-后向变量一样 [Lari & Young 1990]. 推导中状态 v 被使用的期望次数是:

$$c(v) = \frac{1}{P(x|\theta)} \sum_{i=1}^{L} \sum_{j=i}^{L} \alpha(i,j,v)\beta(i,j,v).$$

可以推广上式以便寻找 W_v 被占用 (occupied) 且产生式法则$W_v \to W_yW_z$被使用的期望次数:

$$c(v \to yz) = \frac{1}{P(x|\theta)} \sum_{i=1}^{L-1} \sum_{j=i+1}^{L} \sum_{j-1}^{k=i} \beta(i,j,v)\alpha(i,k,y)\alpha(k+1,j,z)t_v(y,z).$$

于是, 产生式法则 $W_v \to W_yW_z$ 的概率 EM 重估计方程是:

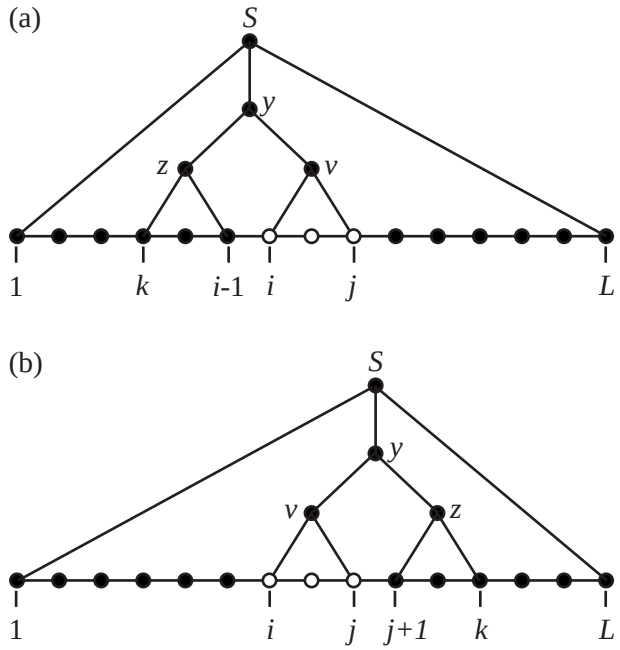


图 9.8: $\beta(i,j,v)$的递归计算图解, 它是**排除** (excluding) 以非终端字v为根且生成子序列 i,j (空心圆圈) 的子树外之所有分析树的概率求和. 图 (a) 对应外向迭代方程的第一部分, 对 $\beta(i,j,v)$ 有贡献项的包括: 非终端 y 和子序列 $1,\ldots,k-1,j+1,\ldots,L$ 的外向值, 填充到子序列 $k,\ldots,i-1$ 之非终端字 z 的内向值以及 $y \to zv$ 的转移概率. 图 (b) 对应迭代方程的第二项, 它包括: 排除子序列 $i,\ldots,k$ 上非终端字 y 的外向概率, 填充进子序列 $j+1,\ldots,k$ 中之状态 z 的内向概率以及 $y \to vz$ 的转移概率.

$$\begin{aligned}\hat{t}_v(y,z) &= \frac{c(v \to yz)}{c(v)} \\ &= \frac{\sum_{i=1}^{L-1}\sum_{j=i+1}^{L}\sum_{k=i}^{j-1}\beta(i,j,v)\alpha(i,k,y)\alpha(k+1,j,z)t_v(y,z)}{\sum_{i=1}^{L}\sum_{j=1}^{L}\alpha(i,j,v)\beta(i,j,v)}.\end{aligned}$$

对另外的产生式法则 $W_v \to a$, 类似方程成立:

$$\hat{e}_v(a) = \frac{c(v \to a)}{c(v)} = \frac{\sum_{i|x_i=a}\beta(i,i,v)e_v(a)}{\sum_{i=1}^{L}\sum_{j=i}^{L}\alpha(i,j,v)\beta(i,j,v)}.$$

这些重估计方程可以从单一观测序列 x 出发, 直接扩展到多条独立观测序列的情形. 对所有序列简单求和便可得到期望计数.

CYK 联配算法

剩下的问题是为序列找到一棵最优分析树 (联配). 这可以应用 Cocke-Younger-Kasami (CYK) 算法解决, 它是内向算法的变体, 用取最大操作替换求和操作. [5]我们计算变量 $\gamma(i,i,v)$, 最终获得 $\log P(x,\hat{\pi}|\theta)$, 其中 $\hat{\pi}$ 是最可能的分析树. 我们还要

[5]CYK 算法最初由 Cocke、Younger 和 Kasami 独立地描述, 它是用于非随机 CFG 的严格匹配算法. 因此对 SCFG 分析算法我们使用"CYK" 算法这个名字有一点不确切, 但我们在文献中没有找到 SCFG 形式算法的其他名字. ——原注

保存一个回溯"变量" $\tau(i,j,v)$, 它是数字三元组 (x,y,z), 用于回溯三维动态规划矩阵并恢复最优联配. 算法中矩阵填充步骤的形式描述如下:

算法: CYK

起始: 从 $i=1$ 到 L, $v=1$ 到 M:

$\gamma(i,i,v) = \log e_v(x_i)$;

$\tau(i,i,v) = (0,0,0)$.

迭代: i 从 $L-1$ 减小到1, j 从 $i+1$ 增加到 L, v 从 1 增加到 M:

$\gamma(i,j,v) = \max_{y,z} \max_{k=i\ldots j-1}\{\gamma(i,k,y)+\gamma(k+1,j,z)+\log t_v(y,z)\}$;

$\tau(i,j,v) = \operatorname{argmax}_{(y,z,k),k=i\ldots j-1}\{\gamma(i,k,y)+\gamma(k+1,j,z)+\log t_v(y,z)\}$.

终止:

$\log P(x,\hat{\pi}|\theta) = \gamma(1,L,1)$. ◁

下面的算法通过回溯以恢复最优联配, 它通过向下推栈中压入和弹出三元组 (i,j,k) 来实现:

算法: CYK 回溯

起始:

压入 $(1,L,1)$ 进栈.

迭代:

弹出 (i,j,v).

$(y,z,k) = \tau(i,j,v)$.

如果 $\tau(i,j,v) = (0,0,0)$ (意即 $i=j$), 连接 x_i 作为 v 的子节点;

否则:

y,z 作为 v 的子节点.

压入 $(k+1,j,z)$.

压入 (i,k,y). ◁

就像 Viterbi 联配算法可以作为对 HMM 之 EM 训练算法的近似一样, CYK 可以作为对内向-外向训练的近似. 我们没有使用内向-外向的概率方法计算计数的期望, 而是为训练序列计算最优的 CYK 联配, 然后对这些联配中的转移和发射进行计数.

SCFG 算法总结

使用内向-外向算法和 CYK 算法, SCFG 可作为全概率论建模系统使用, 就像我们使用 HMM 一样. 下表总结了 SCFG 算法的属性并将其和它们在 HMM 中的对应物作了比较.

目标	HMM 算法	SCFG 算法
最优算法	Viterbi	CYK
$P(x\|\theta)$	前向	内向
EM 参数估计	前向-后向	内向-外向
内存复杂度	$O(LM)$	$O(L^2M)$
时间复杂度	$O(LM^2)$	$O(L^3M^3)$

SCFG 算法具有令人生畏的计算复杂性, 但这主要由其通用性所导致. 受限制更多的 SCFG 有更快的算法. 下一章中 RNA 之 SCFG 算法的时间复杂度是 $O(L^3M)$. 虽然仍然不够理想, 但它已经比 $O(L^3M^3)$ 好了许多.

有时人们说内向-外向算法只能应用于 Chomsky 标准形式的 SCFG, 这意味着在做任何分析之前我们必须先把 SCFG 转化成 Chomsky 标准型. 但这只对于内向-外向算法的呆板定义而言才是正确的. 内向-外向算法给出 Chomsky 标准型的 SCFG 仅仅是为了普适性和记号方便(请回忆, 无论产生式如何复杂, 任意 SCFG 都能改写为 Chomsky 标准型). 限制产生式右边形式的其他 SCFG "标准型" 也有本质上相同的算法. 我们在下一章中将会看到用于 RNA 建模过程之区别于 Chomsky 标准型的另一种自然形式.

9.6 补充读物

我们在这一章中描述的形式语言理论并不是很严格. 对更多细节感兴趣的读者

可以参考 Harrison [1978] 的 Introduction to Formal Language theory 或 Hopcroft & Ullman's [1979] 的 *Introduction to Automata Theory , Languages , and Computation.* 这两本书都主要阐述了非随机上下文无关文法、下推自动机和快速 CFG 分析算法, 因为它们在计算机语言和高效语言编译器的设计上极为重要. Gene Myers [1995] 也撰写了有关上下文无关文法分析算法的专题.

我们描述的 SCFG 算法基于 Lari & Young [1990;1991] 在语音识别领域的工作.

转换文法理论已经应用到有用程度不同的生物学问题之公式化描述, 而不仅仅局限于序列分析. 这些问题包括代谢途径建模 [Collado-Vides 1989; 1991] 以及发育途径建模 [Lindenmayer 1968]. 另外, 人们还将其他"语言学" (linguistic) 方法用于计算序列分析, 它们基于 k 串 ("字" , k-tuple) 频率而非转换文法理论 [Brendel, Beckmann & Trifonov 1986; Pesole, Attimonelli & Saccone 1994; Pietrokovski, Hirshon & Trifonov 1990].

第十章

RNA 结构分析

对于许多有趣的 RNA 而言, 碱基配对相互作用所形成的二级结构比它们的序列更为保守. 这使得 RNA 的序列分析比蛋白质或 DNA 的序列分析更为复杂困难. RNA 二级结构问题是第9章引入的随机上下文无关文法概率论模型的一个自然应用. 在这一章, 我们考察两个在生物学上有重要意义的 RNA 分析问题.

第一个问题是单条序列的 RNA 二级结构预测. 我们将要概述用于 RNA 二级结构预测的两种著名的动态规划算法: Nussinov 算法和 Zuker 算法. 然后我们开发一个实现了概率论形式之 Nussinov 算法的小型 SCFG, 并将 RNA 二级结构预测作为使用 SCFG 分析 RNA 的一个介绍性例子.

第二个是一组相关问题, 它们与分析相关 RNA 家族的多序列联配有关. 与第5章中 profile HMM 用于多序列联配和数据库搜索相同, 我们构建 RNA 结构 profile, 称为"协方差模型" (covariance model , CM), 以便处理受二级结构约束的 RNA 多序列联配. 协方差模型既用于 RNA 多序列联配也用于数据库搜索. 从 RNA 多序列联配中预测一致性结构, 即所谓的比较 RNA 序列分析, 在某种程度上也能由 RNA 协方差模型的训练算法实现自动化.

阅读本章时请读者注意, 基于 SCFG 的 RNA 分析方法既不广为人知也没得到广泛使用. 我们描述的所有 SCFG 方法都处于发展初期, 而且在计算复杂性上存在相当严重的问题. 为 RNA 分析改进的 SCFG 算法或许很快就会出现. 这里, 我们试图给出基于 SCFG 用来进行 RNA 分析的概率论方法基础, 而不陷入那些可能很快就会发生变化的细节中. RNA SCFG 至少为我们提供了一个对应于 profile HMM 的范例. 我们可以看到, 为 HMM 而构建的概率论体系在很大程度上也可以应用于一类不同并且更加复杂的模型.

10.1 RNA

在很多人看来, RNA 仅仅是 DNA 基因和蛋白质翻译机器之间的中间被动信使. 信使 RNA (messenger RNA , mRNA) 通常被描述为一条线性的无结构序列, 除了它所编码的氨基酸序列外并不使人感兴趣. 然而存在许多非编码 RNA , 它们具有高级的三维结构, 并且其中的一些甚至能催化生化反应. 自从20世纪80年代早期催化 RNA 的惊人发现以来[Cech & Bass 1986], 人们发现了许多带有趣新结构的催化 RNA . 最近, 人们已经使用体外进化 (*in vitro* evolution) 技术发明了新的 RNA, 以便在随机 RNA 序列库中筛选新的催化剂和新的特异配体 [Gold *et al.* 1995].

RNA 催化作用的发现, 再次唤起了以 "RNA 世界" 假说而广为人知的生命起源观点 [Gilbert 1986; Gesteland & Atkins 1993]. RNA 世界假说认为在 DNA 基因组和催化蛋白质之前存在一个原始世界, 当时 RNA 基因组由 RNA 催化复制. 时有争论认为许多现存的结构和催化 RNA 是从一个已经消亡的 RNA 世界, 经过漫长的进化时间而流传下来的"分子化石" .

结构 RNA 及催化 RNA 在现代生命体的分子生物学中也十分重要. 核糖体的肽基转移酶活性被认为是由核糖体 RNA (ribosomal RNA) 催化的 [Noller, Hoffarth & Zimniak 1992]. RNA 剪接过程 (从真核 mRNA 前体转录本 (transcript) 中去除

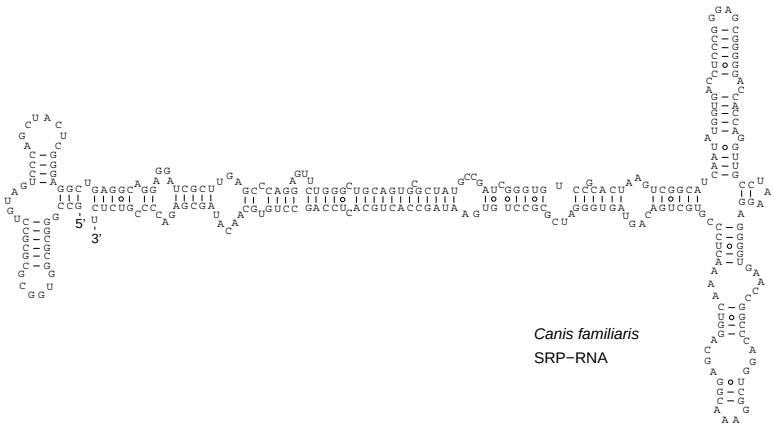


图 10.1: 狗(*canis familiaris*) 信号识别颗粒(signal recognition particle , SRP) 的RNA 二级结构.

内含子) 由一个复杂的 RNA/蛋白质机器(剪接体, spliceosome) 催化, 它包含5种主要的小核 RNA (small nuclear RNA) [Baserga & Steitz 1993]. 参与蛋白跨质膜转运的信号识别颗粒是一个 RNA/蛋白质复合体 [Larsen & Zwieb 1993]. 核糖体 RNA 的正确加工和修饰过程需要许多小核仁 RNA (small nucleolar RNA) [Maxwell & Fournier 1995]. 在信使 RNA 转录本中, RNA 结构(特别在5'和3'非转录区域) 以很多方式影响后转录 (post-transcriptional) 遗传调控. 已知的后转录调控机制包括 mRNA 可变剪接控制 [McKeown 1992]、翻译效率调节 [Melefors & Hentze 1993]以及 mRNA 稳定性调控 [Peltz & Jacobson 1992].

RNA 二级结构的术语

RNA 是四种不同核苷酸亚单位的聚合物. 这四种核苷酸`A`, `C`, `G`和`U`分别是腺嘌呤 (adenine), 胞嘧啶 (cytosine), 鸟嘌呤 (guanine) 和脲嘧啶 (uracil) 的缩写. 在 DNA 中, 胸腺嘧啶 (thymine) `T`代替脲嘧啶.

`G-C`和`A-U`形成以氢键结合的碱基对并称作是互补 (complementary) 的. `G-C`对形成三个氢键并且比形成两个氢键的`A-U`对更趋于稳定. 在 RNA 结构中, 碱基对近似共平面 (coplanar) 且几乎总是**堆积** (stack)在其他碱基对的上面. 连续堆积的碱基对称为**茎** (stem). 在三维空间中, RNA 茎一般会形成一个规则 (A 型) 的双螺旋 (deouble helix). 和 DNA 不同, RNA 通常以单链分子产生, 然后在分子内折叠形成许多短的碱基配对茎. 这种碱基配对结构称为 RNA 的**二级结构** (secondary structure). RNA 二级结构通常用二维图描述, 如图10.1所示.

RNA 二级结构的元件按照图10.2所示的命名. 以碱基对为界的单链子序列称为**环** (loop). 茎末端的环称为**发夹环** (hairpin loop). 由一个简单的茎和环构成的简单子结构称为**茎环** (stem loop) 或者**发夹** (hairpin) (因为这个结构类似发夹). 出现在茎中的单链碱基称为**凸起** (bulge); 如果这些单链碱基只位于茎的一侧, 则称作**凸环** (bulge loop); 如果单链碱基打断了茎两侧的连续碱基, 则称作**中间环** (interior loop). 最后, 存在能辐射出三个或更多茎的**多分支环** (multi-branched loop).

除了正则 (canonical) 的`A-U`和`G-C`配对以外, RNA 二级结构还包含非正则 (non-canonical) 的配对. 最常见的非正则配对是`G-U`配对, 从热力学上讲, 对它的偏好几乎和 Watson-Crick 配对一样. 其他配对也存在. 非正则配对可以扭曲规则的 A 式 RNA 螺旋. 这些扭曲似乎是专门识别 RNA 的蛋白质所偏好的靶位点.

RNA 二级结构中的碱基对几乎总是以嵌套 (nested) 的方式出现. 通俗地讲, 这意味着如果在 RNA 序列上画弧线连接碱基对, 那么弧线就都不相互交叉. 更正式的讲, 位置 i 和 j 之间的碱基配对和位置 i' 和 j' 之间的碱基配对是嵌套的当且仅当 $i < i' < j' < j$ 或 $i' < i < j < j'$ (回想一下, 它符合第9章回文语言的限制条件——

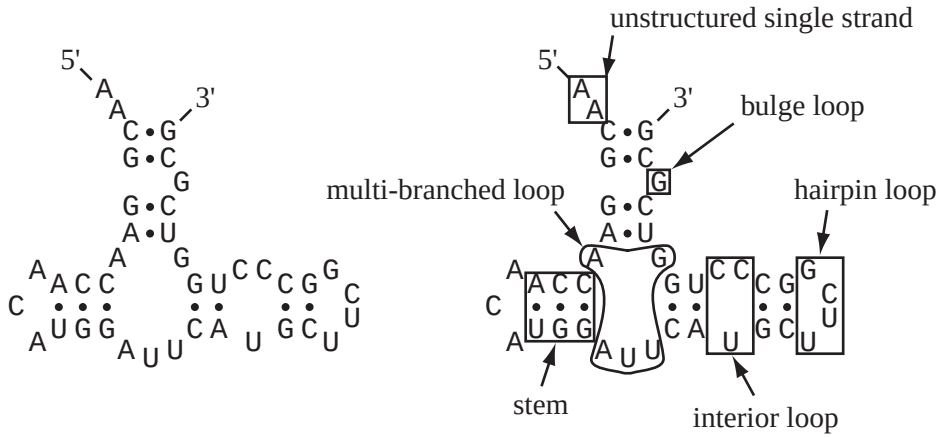


图 10.2: 在假想的例子上标示 RNA 二级结构的基本元件.

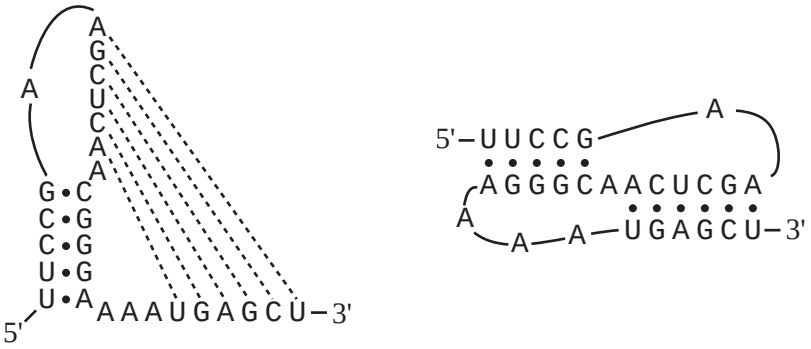


图 10.3: 位于一个环和封闭茎外面位置之间的碱基配对称为假结 (左图). 相同假结的另一描述如右图所示. 在三维空间中, 两个茎能共轴 (coaxially) 堆积并模拟一个临近的 A-型螺旋. 这个特殊的例子是人类免疫缺陷病毒反转录酶 (human immunodeficiency virus reverse transcriptase) 的一种人工选择 RNA 抑制剂 [Tuerk, MacDougal & Gold 1992].

这就是为什么上下文无关文法可以应用到 RNA 二级结构上). 当出现非嵌套碱基对的情况时, 我们称之为**假结** (pseudoknot) 图10.3给出一个假结的例子.

我们描述的动态规划算法, 包括 Zuker 和 Nussinov 的 RNA 折叠算法以及 SCFG 算法, 都不能处理假结. 我们在前一章已经看到, 描述假结的普适交叉相互作用需要用到上下文相关文法. 因为假结存在于许多重要的 RNA 中, 所以我们忽略了很多重要的生物学信息. 所幸的是, 通常来讲嵌套二级结构中假结碱基对的总数比碱基配对数少. 例如, *E . coli* SSU RNA 的一个较权威的二级结构模型包含447个有比较序列分析支持的 Watson-Crick 或 `G-U` 配对, 而它们中只有8个是非嵌套的假结相互作用 [Gutell 1993]. 对包括同源 RNA 数据库搜索在内的很多目的而言, 通常来讲为了获得高效的动态规划算法而牺牲假结所含的信息是可以接受的. 然而对三维结构预测等其他目的而言, 我们必须考虑假结而不能做同样的牺牲.

RNA 序列进化受其结构约束

通常比较容易发现, 某些同源 RNA 有共同结构却没有明显的序列相似性. 剧烈的序列改变通常能被容忍, 只要有补偿性的突变能维持碱基配对的互补即可. 因此在数据库中搜索同源 RNA 时, 能够在搜索保守序列的同时也搜索保守二级结构对

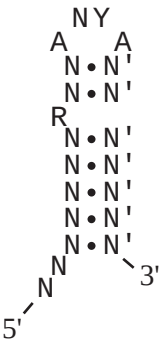


图 10.4: R17噬菌体外壳蛋白的一致性结合位点. N, Y和R是多个可能核苷酸的标准"简并" 符号. N表示{A , C , G , U}, Y表示{C , U}, R表示{A , G}. N ’表示与N配对的互补碱基.

我们是有利的.

图10.4显示了细菌 RNA 病毒 R17 衣蛋白质的一致 (consensus) RNA 结合位点 [Witherell, Gott & Uhlenbeck 1991]. R17 外壳蛋白质结合这个位点并抑制其复制酶的翻译, 这是 R17 溶解循环周期正常时间安排 (timing) 的一部分. 该一致性结构中只有四个一级序列的位置是特定的, 并且其中两个是简并 (degenerate) 的. 如果我们有兴趣在一条核酸序列中搜索 R17 外壳蛋白的结合位点, 那么标准的序列联配方法就不再适用.

到底有多不适用? 我们可以从 Shannon 信息论的经验法则中得到启发. 按照信息论的术语, 一个一致碱基对和一个保守碱基包含相同的信息. 一个完全保守碱基 ($p_x = 1$) 所贡献的信息量 (information) (相对熵, relative entropy) 是 $\sum_x p_x \log_2 \frac{p_x}{f_x} = 2$ 比特 (bit) (假设等概率的初始期望碱基频率, $f_x = \frac{1}{4}$). 类似地, 图10.4中的简并字符R和Y中的每个都含有1比特的信息量, N的是0. 任意序列中的 Watson-Crick 配对所贡献的信息量也是2比特, 这是因为 $\sum_x \sum_y p_{xy} \log_2 \frac{p_{xy}}{f_{xy}} = 2$ (仍假设初始期望是等概率的即 $f_{xy} = \frac{1}{16}$, 并且观测到的 Watson-Crick 配对也以等概率出现即 $p_{AU} = p_{CG} = p_{GC} = p_{UA} = \frac{1}{4}$).

只考虑一级序列的保守性时, R17一致序列的信息量是6比特. 我们期望在每64 (2^6) 个核苷酸中就会随机地找到一个匹配. 向一致结构添加7个碱基对会增加14比特的信息量, 使信息量增加到20比特, 并且使找到一个假匹配的几率降低到百万 (2^{20}) 分之一. 如果在近缘细菌噬菌体 MS2 的基因组 (GenBank MS2CG; 数据库中没有 R17 的基因组) 中搜索NNN NNN NRN NAN YAN NNN NNN, 那么在 3569bp 的基因组中我们就会找到38个匹配, 其中37个是假的. 如果在要求有7个碱基配对的情况下进行重新搜索, 那么我们只能找到位于外壳蛋白结合位点处的一个真实匹配.

以上的搜索使用了类似于 RNMOT [Gautheret, Major & Cedergren 1990].的 RNA 模式匹配程序. 该程序搜索确定的(非随机的) 模体, 并且用二级结构约束作为限制条件. 它对短小且定义明确的模式很有效, 但为那些保守性并不很好的结构寻找匹配时, 它在某种程度上就不是很敏感而且存在问题. 当前, 更灵敏、更基于统计的RNA 数据库搜索的流行做法是为每个感兴趣的RNA 结构都写一个仔细的专用程序 [Dandekar & Hentze 1995]. 存在许多用于寻找转移RNA 基因的此类程序 [Fichant & Burks 1991; Pavesi *et al.* 1994; Lowe & Eddy 1997]; 而且有一个程序用于寻找催化组I 的内含子 (intron) [Lisacek, Diaz & Michel 1994]. 但是随着让人有不同兴趣之 RNA 数量的增加, 现状越来越令人不满意.

通过比较序列分析推断结构

由碱基配对引起的序列约束使数据库搜索变得困难, 却使一致的 RNA 二级结构预测变得相对容易一些——至少相对于蛋白质结构预测而言是这样. 在一个结构正确的 RNA 多序列联配中, 保守的碱基配对通常表现为频繁关联的补偿突变. 尽管只是一种结构预测的理论方法, 但在没有解析三维晶体或 NMR 结构的情况下, 由**比**

较序列分析 (comparative sequence analysis) 过程预测 RNA 二级结构仍然被看作是最为可信的方法. 大多数研究透彻的 RNA 之已接受一致结构均由比较分析给出 [Woese & Pace 1993]. (见图10.5).

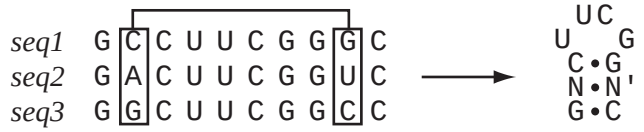


图 10.5: 比较序列分析识别出, 该多序列联配(左图) 中用方框标记的两个位置是共变的以便保持 Waston-Crick 互补. 这种共变意味着一对碱基, 从而得到一致的二级结构预测(右图).

比较分析是一门艰苦的艺术. 通过比较分析推断正确的结构需要知道结构正确的多序列联配, 而推断结构正确的多序列联配需要知道正确的结构. "解析" 结构需要借助于迭代优化过程, 在现有对多序列联配之最好预测的基础上猜测结构; 然后在这个新结构预测的基础上重新联配序列. 这些用来比较的序列必须足够相似, 以便最开始仅由其一级序列的等同度 (identity) 就可以使其发生联配并开始整个过程; 但它们又必须足够不相似, 以便我们可以检测到一些共变的替换.

二序列协方差的一个定量度量源自信息论 [Chiu & Kolodziejczak 1991; Gutell *et al.* 1992]. 已联配的两列 i 和 j 间的互信息 M_{ij} 由下式给出

$$M_{ij} = \sum_{x_i,x_j} f_{x_i x_j} \log_2 \frac{f_{x_i x_j}}{f_{x_i} f_{x_j}}. \tag{10.1}$$

f_{x_i} 是在第 i 列观测到之四碱基(`A` , `C` , `G` , `U`) 中一个的频率. $f_{x_i x_j}$ 是第 i 列和第 j 列16种可能碱基对中一个的联合(成对) 频率. M_{ij} 衡量联合频率分布偏离两列独立变化之期望分布的程度. 对四字母的 RNA 字母表, M_{ij} 在0到2比特之间变化. 当 i 和 j 分别完全随机 ($f_i = f_j = 0.25$) 时, M_{ij} 达到最大, 但是有时 i 和 j 却关联地很好, 例如在 Watson-Crick 配对中.

直观上讲, M_{ij} 告诉我们从某个位置的残基等同度能得到多少信息量, 假如我们已知处于另一个位置的残基等同度. 一对没有序列约束的碱基之信息量是2比特: 例如已知 i 位置是`G`, 我们对于 j 位置处的不确定性 (uncertainty) 就从4种减少到只有一种可能 (`C`), 因此我们得到2比特的信息量. 如果 i 和 j 是不相关的, 则它们的互信息是零. 如果 i 或 j 是高度保守的位置, 那么我们就得到很少或得不到互信息: 如果一个位置不变, 那么我们就不能从它同伴 (partner) 的等同度中得到关于它的任何知识.

图10.6展示了从1415条 tRNA 多序列联配计算得到的 M_{ij} 值的等高线 (contour) 图. 三叶草结构的四个碱基配对茎十分明显. 相对于在一级序列上极为可变的反密码子 (anticodon) 茎和受体 (acceptor) 茎来讲, 在一级序列上相对高度保守的 D 茎和 TψCG 茎并不十分明显.

练习

10.1 式(10.1) 的互信息计算需要对所有16种不同的碱基对频率进行计数. 这样做的好处是不需要假设 Watson-Crick 配对, 因此我们可以检测共变的非正则配对之互信息, 如 `A-A`和 `G-G`配对. 另一方面, 该计算需要大量的联配序列以便获得这16种概率的可信频率. 请写出碱基配对关联的另一种信息论度量, 它只考虑 i, j 的两种等同分类, 而不是原来的16种: 令 Watson-Crick 和 `G-U` 配对在一类, 其他的配对归为另一类. 对少量序列和无穷多条序列的极限两种情况, 比较这种计算方法和 M_{ij} 计算的性质.

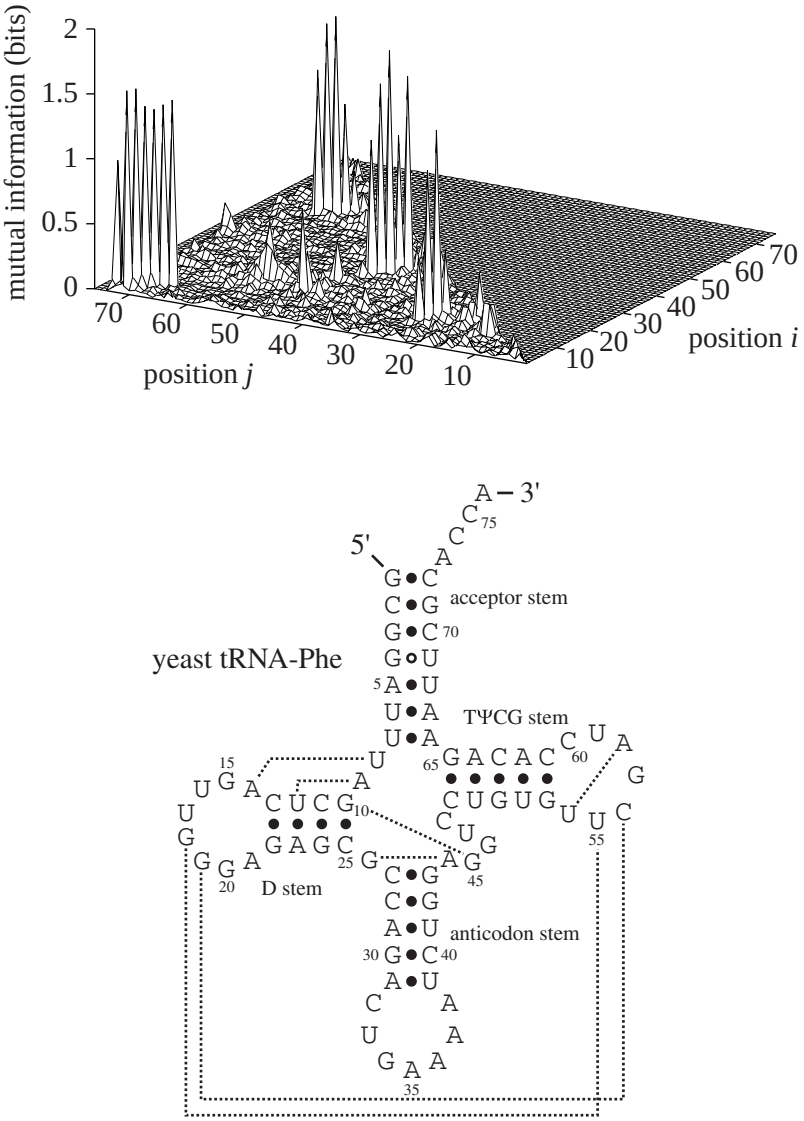


图 10.6: 某 tRNA 联配的互信息图(上图), 以及酵母苯丙氨酸 (phenylalanine) tRNA 的二级结构(下图). 上图位置i, j之平面的对角线上分别对应了 tRNA 三叶草结构4个茎的4个强共变位置. 虚线表示在酵母 tRNA-Phe 晶体结构中观测到的某些其他三级连接 (tertiary contacts). 其中的某些三级结构产生相互关联的碱基对, 我们可以在互信息图中隐约地看到.

10.2 RNA 二级结构预测

假设我们希望预测单个 RNA 的二级结构. 从一条序列出发我们可以得出多个貌似合理的二级结构. 这个数目随序列的长度呈几何增长. 一条长度仅为200碱基的 RNA 可以有超过 10^{50} 种可能的碱基配对结构. 我们必须区分生物学上正确的结构和所有不正确的结构. 我们同时需要一个为正确结构打最高分的函数, 以及一个为所有可能结构计分的算法.

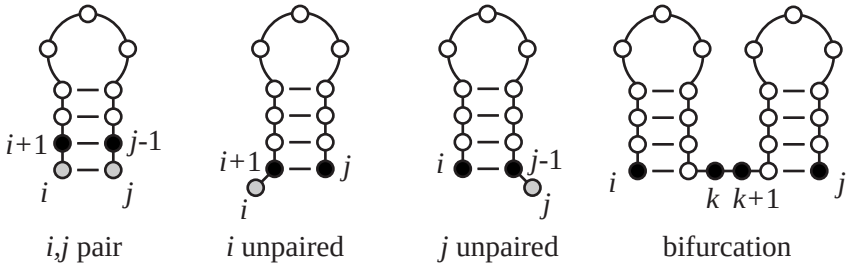


图 10.7: Nussinov 算法所考虑的四种方式, 我们可以通过这些方式把 i 和/或 j 添加到已经计算出的更短子序列之最优结构上, 以便得到子序列 i, j 的最优 RNA 结构. 此处不考虑假结.

碱基配对最大化与 Nussinov 折叠算法

有一种方法可以用来找到含最多碱基对的结构. Nussinov 为这个问题引入了一种高效的动态规划算法 [Nussinov *et al.* 1978]. 尽管对精确结构的预测而言该标准过于简单, 但这个例子是有指导意义的, 因为 Nussinov 算法的机制和那些更精密的能量最小化折叠算法以及基于概率论模型的 SCFG 算法是一致的.

Nussinov 以递归的方式计算. 它先对短序列计算最优结构, 然后再向外拓展到越来越长的子序列. 递归计算的关键在于我们只能以四种可能的方式从较小子序列的最优结构中得到对 i, j 的最优结构(图10.7):

(1) 向子序列 $i+1, j$ 的最优结构上增加一个非配对位置 i;
(2) 向子序列 $i, j-1$ 的最优结构上增加一个非配对位置 j;
(3) 向子序列 $i+1, j-1$ 的最优结构上增加一个 i, j 的配对;
(4) 将两个最优子结构 i, k 和 $k+1, j$ 组合到一起.

我们将更正式地描述 Nussinov 的 RNA 折叠算法如下. 给定一条带符号 $x_1, \ldots, x_L$, 长度为 L 的序列 x. 令 $\delta(i,j) = 1$, 如果 x_i 和 x_j 是互补碱基对; 否则 $\delta(i,j) = 0$. 我们可以递归地计算分值 $\gamma(i,j)$, 它是子序列 $x_i, \ldots, x_j$ 中能够形成的碱基对数目之最大值.

算法: Nussinov RNA 折叠, 填充阶段

起始:

$$\begin{aligned} \gamma(i, i-1) &= 0 \qquad && i = 2\text{到}L; \\ \gamma(i, i) &= 0 && i = 2\text{到}L. \end{aligned}$$

递归: 从长度为2到长度 L 的所有子序列:

$$\gamma(i,j) = \max \begin{cases} \gamma(i+1, j), \\ \gamma(i, j-1), \\ \gamma(i+1, j-1) + \delta(i,j), \\ \max_{i<k<j}[\gamma(i,k) + \gamma(k+1, j)]. \end{cases}$$ ◁

图10.8给出了 Nussinov 矩阵填充操作的一个例子.

$\gamma(1, L)$ 的值是最大碱基配对结构中碱基对的数目. 经常有许多不同的结构带相同的碱基配对数. 为了找到最大碱基配对结构之一, 我们从 $\gamma(1, L)$ 开始对动态规划矩阵中计算得到的值进行回溯. 其回溯算法的伪代码如下:

算法: Nussinov RNA 折叠, 回溯阶段

起始: 压 $(1, L)$ 入栈.

递归: 重复下列过程, 直到栈变空:

- 弹出 (i, j);
- 如果 $i >= j$, 则继续;
  否则如果 $\gamma(i+1, j) = \gamma(i, j)$, 则压入$(i+1, j)$;
  否则如果 $\gamma(i, j-1) = \gamma(i, j)$, 则压入$(i, j-1)$;

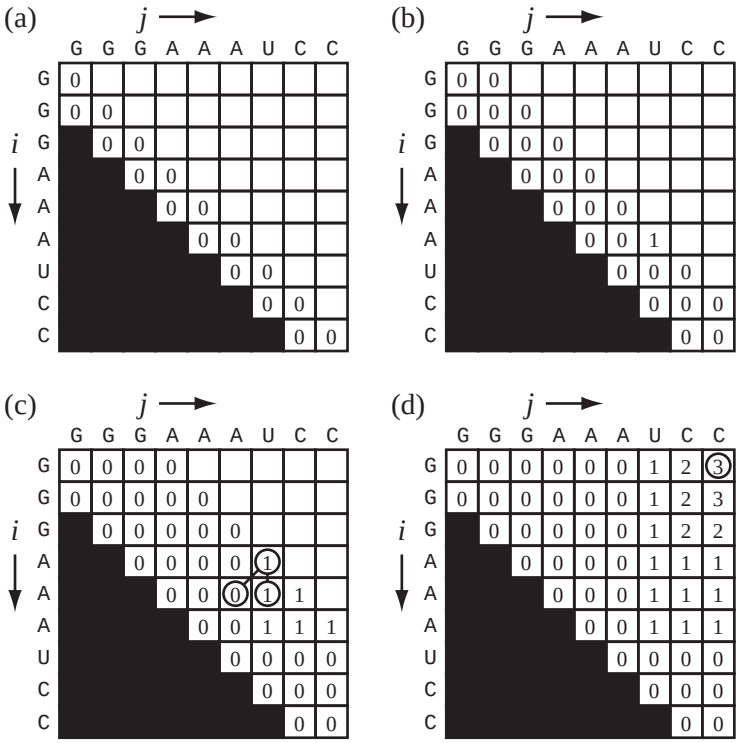


图 10.8: 用一条示例序列GGGAAAUCC显示了 Nossinov 折叠算法的矩阵填充阶段. (a) 已初始化的半对角矩阵. (b) 计算出长度为2的子序列之分值后的矩阵. (c) 相同子序列的两种不同最优子结构. 对子序列AAAU, 要么令 i 处的A和 j 处的U配对(对角路径), 要么将 i 添加到 $i+1$ 处的A和 j 处的U 已经配对的子结构中(垂直路径). (d) 最终矩阵. 右上角的值指最大的配对结构有三个碱基对.

否则如果 $\gamma(i+1,j-1)+\delta_{ij}=\gamma(i,j)$则:
- 记录碱基对 i,j;
- 压入 $(i+1,j-1)$.

否则对 $k=i+1$ 到 $j-1$: 如果 $\gamma(i,k)+\gamma(k+1,j)=\gamma(i,j)$ 则:
- 压入 $(k+1,j)$;
- 压入 (i,k);
- 结束. ◁

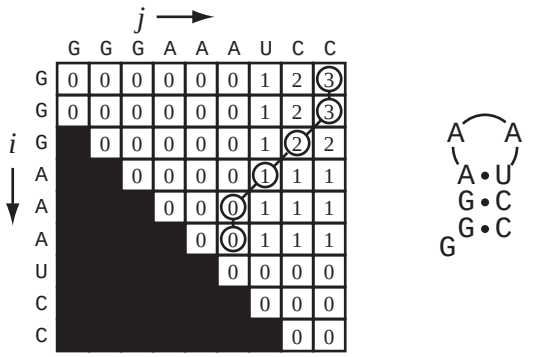


图 10.9: 对图10.8中已填充的矩阵, Nussinov 折叠算法的回溯阶段演示. 用圆圈表示一条最优回溯路径. 右边是对应这条路径的最优结构.†

回溯过程在时间和内存上都是线性的. 填充步骤是受限步骤 (limiting step), 因为它在空间上是 $O(L^2)$, 而时间上是 $O(L^3)$. 回溯的例子见图10.9. 图10.9中的回溯没有分支, 所以它对回溯算法中下推栈的要求并不明显. 当回溯有分叉结构时, 下推栈就变得很重要. 当在分叉的一边进行回溯时, 下推栈记下分叉的另一边, 请读者回忆第9章的下推自动机.

练习

10.2 上面给出的回溯算法事实上并不能给出图10.9所示的结构. 那么, 用该算法复原出的另一种包含三个碱基对的最优结构是什么? 是否存在其他的最优结构? 修改回溯算法, 使其能找到一种不同的结构.

10.3 正如我们所给出的, Nussinov 算法能在相邻互补残基间产生无意义的碱基对(例如, 前面练习中的一种可能结构包含这样的AU碱基对). 修改 Nussinov 折叠算法, 使发夹环至少含有 h 个碱基, 并给出用于填充和回溯的新递归方程.

10.4 证明 Nussinov 折叠算法能够扩展到找出一种最大**分值**的结构, 其中残基 a 和 b 的碱基对有分值 $s(a,b)$ (例如, 我们可以设 $s(\texttt{G , C}) = 3, s(\texttt{A , U}) = 2$ 以便更好地反映GC对具有更好的热力学稳定性).

SCFG 形式的 Nussinov 算法

Nussinov 算法和第9章的 SCFG 算法在本质上是相似的. 作为 SCFG 在 RNA 二级结构分析中应用的例子, 考虑下面简单的 RNA 折叠 SCFG 之产生式法则:

$$\begin{array}{ll} S \to aS|cS|gS|uS & (i\text{未配对}), \\ S \to Sa|Sc|Sg|Su & (j\text{未配对}), \\ S \to aSu|cSg|gSc|uSa & (i,j\text{配对}), \\ S \to SS & (\text{分叉情况}), \\ S \to \epsilon & (\text{终止推导}). \end{array} \tag{10.2}$$

该 SCFG 有单个非终端字符 S 以及14个带相关概率参数的产生式规则. 现在假设概率参数都已知. 使用该 SCFG 的一条序列之最大概率分析 (parse) 就是序列位置到产生式的分配. 因为这些产生式对应于二级结构元件(碱基对和单链碱基), 所以这个最大概率分析就等价于最大概率的二级结构. 如果碱基配对产生式有相对高的概率值, 那么 SCFG 将青睐那些倾向于在结构上有最大碱基配对数目的分析.

尽管这个 SCFG 的产生式法则不是 Chomsky 标准形式, 但是我们可以容易地写出一个 CYK 分析算法以寻找最大概率的二级结构. 我们也可以把 SCFG 转化为 Chomsky 标准形式, 然后应用第9章中的算法. 尽管 Chomsky 标准形式方法以它的普遍性引人注目, 但对特殊的 SCFG 采用特殊算法却更有效. 改进的 CYK 算法如下. 令 SCFG 产生式的概率参数用 $p(aS), p(aSu)$ 等表示.

算法: 用于 Nussinov 风格 RNA SCFG 的 CYK 算法

起始:

$$\gamma(i,i-1) = -\infty \qquad i = 2\text{到}L;$$

$$\gamma(i,i) = \max \begin{cases} \log P(x_i S) \\ \log P(S x_i) \end{cases} \qquad i = 1\text{到}L.$$

递归: i 从 $L-1$ 减小到1, j 从 $i+1$ 增加到 L:

$$\gamma(i,j) = \max \begin{cases} \gamma(i+1,j) + \log p(x_i S); \\ \gamma(i,j-1) + \log p(S x_j); \\ \gamma(i+1,j-1) + \log p(x_i S x_j); \\ \max_{i<k<j} \gamma(i,k) + \gamma(k+1,j) + \log p(SS). \end{cases}$$

◁

当此过程结束时, $\gamma(1, L)$ 就是给定 SCFG 模型 θ 之最优结构 $\hat{\pi}$ 的对数似然 $\log P(x, \hat{\pi}|\theta)$. 为了寻找对应于最优分值的结构, 回溯过程可以和 Nussinov 算法中的回溯过程相似地实现, 也可以和第9章中所描述的 CYK 算法类似, 在填充阶段保留额外的回溯指针.

现在这个算法和原 Nussinov 算法的主要不同在于, 该 SCFG 的描述是一个概率论模型. 我们会有一些原则性更强的选择用于优化模型参数. 我们需要设置 SCFG 的参数, 为此可以主观地估计有关概率, 也可以对已知 RNA 结构中的状态转移进行计数然后将计数转化为概率. 我们甚至可以从**未知**结构的示例 RNA 中学习概率, 使用"期望最大化" (EM) 和内向-外向训练 (inside-outside training) 迭代地推断结构和参数(即在 EM 算法中, 这些结构是隐数据). 一旦我们写下了 SCFG 作为 RNA 折叠问题的一个全概率论模型, 那么我们就能"打开开关" , 自动地应用前几章我们学习过的所有概率论机制.

与 Nussinov 算法一样, 这个小型 SCFG 是个很好的入门例子, 但它太过简单以至于不能用作精确的 RNA 折叠器. 它没有考虑一些重要的结构特征, 例如对某些长度的环的偏好, 也没有考虑由茎部邻近碱基配对间堆积的相互作用而引起的对结构中某些最近邻居的偏好. †

练习

10.5 写出一个回溯算法, 以便在上面的算法完成后决定最优的 RNA 二级结构.

10.6 设计一个 SCFG , 使它用不同的非终端字符对凸起环、发夹环、多分支环和单链建模.

能量最小化与 Zuker 折叠算法

RNA 折叠被生物物理学原理, 而非计数和最大化的碱基配对数所支配. 对于单链 RNA, 最复杂的二级结构预测方法是 Zuker 算法. 它是一种能量最小化算法, 假设正确的结构有最低的平衡自由能 (ΔG) [Zuker & Stiegler 1981; Zuker 1989a].

RNA 二级结构的 ΔG 近似地等于来自各种环、碱基对及其他二级结构元件的独立贡献之和. 与简单的 Nussinov 计算的一个重要不同是, 在计算茎的能量时, 对邻近碱基对间接触面处的**堆积** (stacking)贡献求和, 而不使用每对碱基的单独贡献. 换句话说, 含 n 对碱基的茎之能量是 $n-1$ 个碱基堆积项的和, 而不是 n 对碱基项的和. 该结果能更好地符合实验观测的 RNA 结构之 ΔG 值, 但同时这也使动态规划算法变得复杂. RNA 结构预测的 ΔG 参数表与小模型 RNA 的实验热力学研究结果相吻合 [Freier *et al.* 1986; Turner *et al.* 1987]. 这些参数包括堆积参数、发夹环长度、凸起环长度、中间环长度、多分支环长度、单摆 (single dangling) 核苷酸数以及茎终端失配数等.

图10.10给出的例子描述了如何预测 RNA 结构的 ΔG. 假设单碱基凸起不打断茎的堆积, 因此图中包含了一个堆积项. 假设更长的凸起打断堆积, 从而不增加堆积项. 发夹环能量是两项之和: 只和环长度有关的环扰动能量, 以及依赖于关闭碱基对和茎之首、末碱基的末端失配能量. 图10.10中使用的能量来自早期 37°C 时的" Freier 规则" [Freier *et al.* 1986]. [1]

我们可以用动态规划算法迭代地计算最小能量结构(假设不存在假结), 这与上面计算最大碱基配对的结构十分相似. 二者间的主要区别是, 因为堆积参数的缘故, 我们使用两个矩阵(称为 V 和 M) 而不是一个. $W(i, j)$ 是 i, j 处最好结构的能量. $V(i, j)$ 是 i, j 配对时 i, j 处最好结构的能量. 此算法能通过仅向矩阵 V 中增加新的碱基配对来记录堆积相互作用. 在借助仿射空位罚值 (第2章) 记录插入延伸的二序列动态规划联配中, 我们使用了额外的插入状态, 而从概念上讲, 这与此处所使用的两状态计算十分相似. Zuker 算法的完整描述见 Zuker & Stiegler [1981].

---

† 在此我们需要讨论文法的模糊性 (ambiguity). 显然任意实用的 RNA 结构预测文法在形式上都必须是模糊的: 对给定序列而言, 存在许多可能的结构, 而我们只对其中的最大似然结构感兴趣. 同时我们也需要避免第二种模糊性: 每个结构的分析树必须是明确的. 否则我们就不能断言(将由 CYK 算法给出的) 最大似然分析树对应于最大似然结构. 式 (10.2) 的文法就具有这第二种模糊性, 从而也构成了它不是恰当 RNA 折叠文法的另一个原因. ——译注

[1] 当前最新的参数可以从国际互联网上获得: `http://www.bioinfo.rpi.edu/~zukerm/rna/energy/`. ——原注

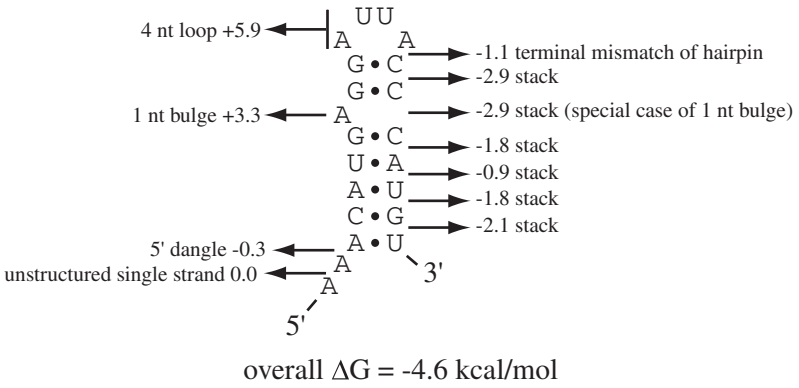


图 10.10: 计算 RNA 茎环(野生型 R17 外壳蛋白的结合位点) 的 ΔG.

我们可以写下遵循相似规则的 SCFG. 最简单的堆产生式法则可能是, 例如 $cVg \rightarrow cgVcg$ 表示在茎上的CG配对之后(堆积上) 产生一个GC配对, 使用 V 作为一个碱基对的生成非终端字符(像 Zuker 的 V 矩阵中那样). 为了生成GC我们把CG作为上下文放置于产生式的左边, 从技术上讲这是一个上下文相关的产生式, 因此我们不能使用此规则用作 SCFG 的基础. 但是我们可以使用四种不同的非终端字符 $V^{au}, V^{cg}, V^{gc}, V^{ua}$, 将其转换为上下文无关产生式, 并使用形如 $\rightarrow gV^{gc}c$ 的右边来产生一个G-C配对, 例如——非终端标记 V^{gc} 可以"记住" 一个G-C配对刚刚生成. (换句话说, 我们所做的一切就是使模型变为一个更高阶的 Markov 过程.) 例如, 产生式 $V^{cg} \rightarrow gV^{gc}c$ 的概率就是一个C-G对堆积在一个G-C对上的概率. [2] Zuker 算法的其他细节及其两个矩阵 V 和 W 均可以整合到带两个(对最近相邻上下文扩展而得到的) 非终端字符 V 和 W 的类似全概率论模型中. 其中 SCFG 形式 Zuker 算法的 CYK 算法及内向-外向算法都有与 Zuker 算法本身相同的算法复杂度.

次优 RNA 折叠

最初的 Zuker 算法只能找到最优结构. 生物学上正确的结构通常不是我们计算出的最优结构, 而是在计算出的最小能量百分之几范围内(即在误差条 (error bar) 之内) 的一个结构. 引入有效的**次优** (suboptimal) 折叠算法是一种显著地进步. Zuker 次优折叠算法 [Zuker 1989b]与在内向和外向两个方向运行的 CYK 算法相似. 其中一个矩阵(完全就是 CYK 算法) 为所有 i, j 配对的子序列 i, j 寻找最优结构的 ΔG, 而第二个矩阵 (事实上是外向 CYK 算法) 为所有 i, j 配对且排除子序列 $i+1, j-1$ 的序列寻找最优结构[3]对于给定的 i, j, 这两个数字的和就是使用 i, j配对的最优结构的 ΔG. 然后, 次优折叠算法根据 ΔG 对碱基对 i, j "随机" 地进行一次抽样, 并在内向和外向两个矩阵中回溯, 以便找到使用该碱基对的最优结构(因此更准确地说, 此算法次优地对**单碱基对** (one base pair) 进行抽样. 结构的其余部分是给定该碱基配对条件下的最优结构.)

SCFG 版本的 RNA 折叠算法也能够通过依概率回溯内向矩阵并按照结构之似然对结构进行抽样, 这与第6章中从前向矩阵抽样得到次优 profile HMM 联配的方法类似.

碱基对的置信度估计

McCaskill [1990] 为能量最小化折叠算法引入了配分函数计算, 我们可以用它计算特定碱基对或结构的概率. McCaskill 算法用 Gibbs-Boltzmann 方程把 ΔG 转换成概率, 并对所有结构的概率求和, 而不是选择能量最低的单一结构. 所有包括碱基

[2]因为对给定的 x_i, x_j 配对, 只有一个非终端字符是可能的, 其他三个的概率都为零, 因此这四个非终端字符在分析算法的时间和空间复杂度上与一个非终端字符表现相同. ——原注

[3]Zuker 实际上使用了双序列并将其看作一个环, 然后计算 $j, \ldots, L/1, \ldots, i$ 上最优结构的能量. 对于环状 RNA, 这就给出了与外向算法相同的结果. 而对于线性 RNA, Zuker 算法必须将3'端和5'端之间并不存在的交接点作为一个特殊情况来处理. 外向算法在实现上可能更容易一些. ——原注

对 i, j 的结构概率之和除以所有结构的概率之和被认为是碱基对 i, j 的置信度估计.

从 SCFG 的观点看, McCaskill 算法本质上是一种内向-外向算法, 而 Zuker 算法本质上是一种 CYK 算法. 从概念上讲, SCFG 的碱基对置信度估计与第4章中我们描述过的用于成对 HMM 之二序列联配置信度估计类似.

练习

10.7 为方程 (10.2) 中 Nussinov 风格的 RNA 折叠之 SCFG 写出内向算法、外向算法以及内向-外向重估计方程.

10.8 参考 profile HMM 次优联配抽样, 从你的内向矩阵中给出一个对结构进行概率抽样的算法.

10.9 说明如何利用你的内向-外向变量计算位置 i, j 配对之对所有结构求和的概率. 结果的函数形式应该与内向-外向重估计方程类似.

10.3 协方差模型: 基于 SCFG 的 RNA profile

假设有一族相关 RNA ——转移RNA 或催化内含子组I ——它们有通用的一致二级结构以及一级序列模体, 我们希望在序列数据库中搜索同源 RNA . 在第5章里, 我们使用基于 HMM 的 profile 为 DNA 和蛋白质家族建模, 但我们已经在第9章中证明了, HMM 是一级结构模型因此它不能有效地处理 RNA 二级结构限制. 在这一节, 我们将描述基于 SCFG 的 RNA 结构 profile, 称为"协方差模型" (covariance model , CM), 它是与 profile HMM 类似的 SCFG. Profile HMM 指定了很适合为多序列联配建模的重复线性 HMM 构架, 而 CM 则为一致 RNA 二级结构建模指定了重复的树状 SCFG 构架.

虽然我们这里沿用的协方差模型方法是由 Eddy & Durbin [1994] 所开发的, 但同样的一般性思想和算法也同时出现在由Sakakibara *et al* . [1994] 所独立提出的基于 SCFG 之相似 RNA 模型中.

CM 是详细且相当复杂的概率论模型, 我们首先从直观上考虑小 RNA 联配的更简单模型.

无空位 RNA 联配的 SCFG 模型

图10.11显示了某 RNA 的一致结构以及适合该一致结构的 RNA 家族之无空位多序列联配. 为描述此基于 SCFG 模型的多序列联配, 我们需要一些不同类型的非终端字符以生成不同类型的二级结构和序列元件.

我们**成对地** (pairwise) 发射能生成两配对碱基的非终端字符, 以便为碱基配对的列建模. 如果可能, 我们通过**向左** (leftwise) 发射非终端字符, 以便为单链列建模. 对于茎上3'端的凸起和中间环, 有时需要**向右** (rightwise) 发射非终端字符. **分叉** (bifurcation) 非终端字符用于分离多茎和多分支环. 定义一个特殊的**起始** (start) 非终端字符, 它既是初始非终端字符, 又是从分叉产生的直接子结构. [4] 再定义一个特殊的**结尾** (end) 非终端字符, 它以概率1生成 ϵ 并终止推导. 下面总结了这些状态的产生式法则, 其中使用 W 作为一般的非终端字符, 代表六个状态中的任意一个:

$P \to aWb$	成对(16对发射概率),
$L \to aW$	向左(4个单发射概率),
$R \to Wa$	向右(4个单发射概率),
$B \to SS$	分叉(概率为1),
$S \to W$	起始(概率为1),
$E \to \epsilon$	结尾(概率为1).

[4] 没必要为开始某些非终端字符而进行分叉, 但这样却会简化许多子序列算法. 其中的一个原因是, 我们可以切断以分叉起始的连接, 并将此结构的每个分支都当作一个独立 RNA 结构域 (domain) 的一个独立 SCFG 模型来对待. ——原注

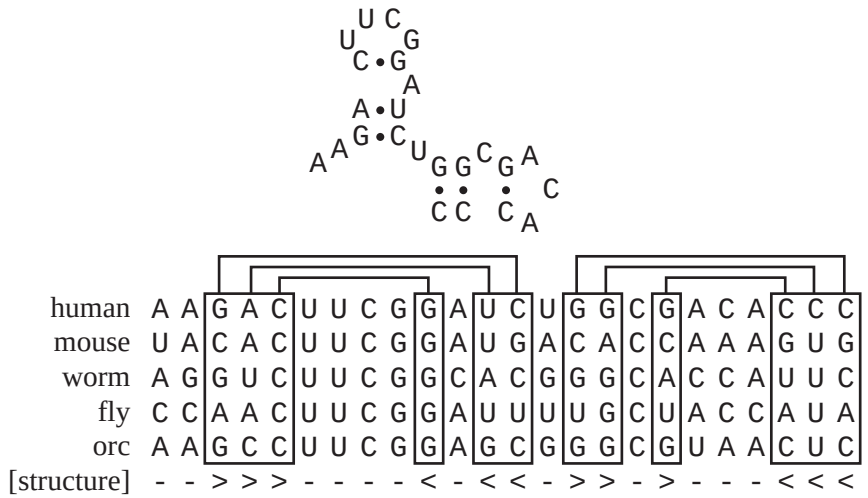


图 10.11: 顶端显示不带插入和删除的示例 RNA 家族之一致结构. 来自不同有机体的五条示例序列有相同的结构, 如多序列联配下方的结构所示. 联配中的碱基配对位置用方框表示, 配对的碱基用线条连接. 联配的最后一行是结构的一致表示, 我们用这种格式注释 RNA 结构联配 [Konings & Hogeweg 1989].

图10.11中的结构可以简化为一个 SCFG, 细节如下所述. 为了清楚起见, 我们为每个非终端字符只写出一种可能的产生式(即对应于图10.11中人类 (human) 序列结构的产生式). 成对产生式总共有16种产生式和16种可能配对的产生式概率; 向左和向右的产生式各有4种.

	茎1	茎2
$S_1 \to L_2 \ldots$	$S_5 \to P_6$	$S_{15} \to L_{16}$
$L_2 \to aL_3 \ldots$	$P_6 \to gP_7c \ldots$	$L_{16}n \to uP_{17} \ldots$
$L_3 \to aB_4 \ldots$	$P_7 \to aR_8u \ldots$	$P_{17} \to gP_{18}c \ldots$
$B_4 \to S_5S_{15}$	$R_8 \to P_9a \ldots$	$P_{18} \to gL_{19}c \ldots$
	$P_9 \to cL_{10}g \ldots$	$L_{19} \to cP_{20} \ldots$
	$L_{10} \to uL_{11} \ldots$	$P_{20} \to gL_{21}c \ldots$
	$L_{11} \to uL_{12} \ldots$	$L_{21} \to aL_{22} \ldots$
	$L_{12} \to cL_{13} \ldots$	$L_{22} \to cL_{23} \ldots$
	$L_{13} \to gE_{14} \ldots$	$L_{23} \to aE_{24} \ldots$
	$E_{14} \to \epsilon$	$E_{24} \to \epsilon$

该模型有一个重要性质: 它的非终端字符可以通过一棵树连接起来. 这棵 SCFG 树的结构准确地反映了 RNA 结构及其分析树的结构. (这并非 SCFG 的必要性质; 例如前面我们看到, RNA 折叠的 SCFG 中就不具备此性质.) 这使得我们可以采用一种简便的 SCFG 图形表示, 它能直观且简洁地反映正被建模的 RNA 家族之结构. 从前面的文法可以清楚地看到, 即使是简单的 RNA SCFG, 我们所写出的产生式形式也是十分冗长的. 相同 SCFG 的图形表示如图10.12所示.

这个模型总共有24种非终端字符, 它们对24种核苷酸的 RNA 联配建模. 这两个数字并不总相同, 但模型中非终端字符的个数大致上和联配的长度呈线性关系. 每个配对都需要一个非终端字符, 每个单链核苷酸也需要一个非终端字符, 非终端字符 B、S 和 E 的分类 (assortment) 使得这个模型变完整.

产生式法则列表和图10.12的图形化模型间存在十分重要的区别. SCFG 通常在

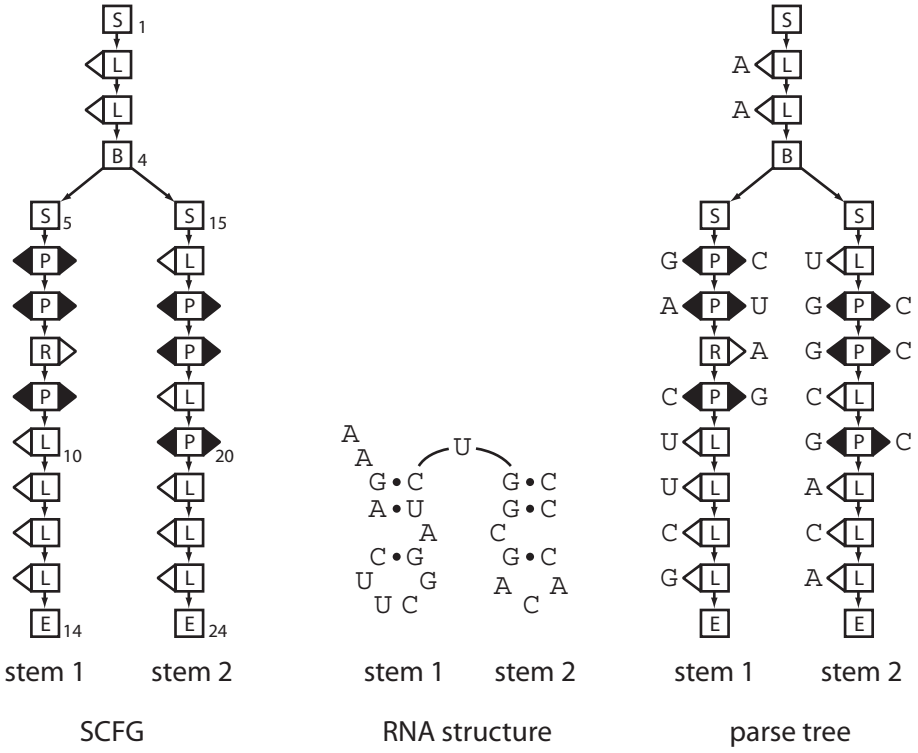


图 10.12: 无空位 RNA SCFG 示例的图形在左边表示. 用 P 标记的方框代表16个成对产生式概率; 用 L 和 R 标记的方框分别表示4个向左和向右的产生式法则; 用 S、B 和 E 标记的方框分别表示起始、分叉和结尾非终端字符. RNA 一致序列结构被重绘(中间) 为对应更接近 SCFG 的树结构. 对这个结构的分析树 (右边) 如图所示, 其中 RNA 核苷被分配到 SCFG 的状态.

转移上发射符号(即 $W_1 \rightarrow aW_2b$, 发射符号与转移到一个新的非终端状态同时进行). 我们也可以将转移从发射中分离出来, 并且独立于前一个状态转移, 在状态上发射符号. 第9章已经讨论过在转移上发射 (emit-on-transition , Mealy 机) 和在状态上发射 (emit-on-state , Moore机) 间的这种区别. 在大多数我们已经描述过的 HMM 中, 人们倾向于使用 Moore 机, 其中包括 profile HMM. 协方差模型作为第5章中 profile HMM 思想基于 SCFG 的扩展, 也使用在状态上发射的形式. 同样地, 我们通常称 CM 的非终端字符为"状态" . 因此, 图10.12中无空位 SCFG 的每个成对状态都有16个发射概率, 每个向左或向右状态都有4个发射概率, 并且所有状态转移的概率均为1 (无空位模型中不存在可选路径). 当模型允许插入和删除时, 转移概率才会令人感兴趣.

将RNA 结构联配映射到 SCFG 时还存在一些不明确的地方, 例如发夹环能够通过向右的状态从右向左地生成. 在可能的情况下, 我们都无条件地选择向左状态, 因为向左状态与我们前面提到的对 profile HMM 的处理非常相似.

只要不发生插入或删除, 图10.12的模型就是图10.11中 RNA 家族的一个合理模型. 该模型相当于只含匹配状态不含插入/删除状态的 HMM. 我们可以将其作为全概率论模型对 RNA 数据库进行搜索. [5]然而, 我们很可能会漏掉许多同源结构; 如果要对大多数真实的 RNA 结构进行建模, 那么允许插入和删除就是十分重要的. 我们现在回到协方差模型, 它将无空位的 SCFG 示例推广到容许插入和删除的模型, 就像我们曾经把只含匹配状态 (match-state-only) 的 HMM (也就是无空位权重矩阵)

[5]如果我们使用这个 SCFG 对 RNA 进行分析, 那么对联配使用内向-外向及 CYK 算法将会变得小题大作. 因为不含空位, 所以只可能有单一联配. 因此我们可以用一个概率论模型在 $O(L)$ 的时间内为无空位 RNA 的模体计分. ——原注

推广到带插入和删除状态的 profile HMM 一样.

练习

10.10 重写来自无空位 RNA 模型的产生式规则列表, 使符号能像 HMM 那样独立于先前状态而发射. 这是对应于图10.12中图形 SCFG 表示的正式随机转换文法.

协方差模型的设计

CM 的设计目标很直接, 但要实现它们却不容易. 首先, CM 围绕一棵一致 RNA 结构树而构建起来, 它和我们上面讨论过的用于无空位 RNA 的模型树很相似. 第二, CM 允许联配中的任意位置出现任意长度的插入或删除. CM 使用与 profile HMM 相同的策略处理与一致结构相关的插入和删除. 请回忆 profile HMM 重复使用三个状态(匹配、插入、删除) 对多序列联配中的每个位置建模. 我们可以将 profile HMM 看作是无空位一致模型的固定 (stereotyped) 扩展: 无空位模型中的每个匹配状态都可以扩展为 profile HMM 中的匹配、删除和插入. 类似地, CM 将无空位一致模型扩展为单个状态的固定模式. 我们称模型结构的重复单元为**节点**. Profile HMM 是单一类型三状态节点的线性串, 而 CM 是节点的分支树, 它含有带不同状态数目的不同类型节点.

用于单链一致位置的向左节点可以像 HMM 节点那样扩展到匹配、插入和删除状态, 向右的单态节点也是如此. 因此向左节点变为状态 ML 、IL 和 D 的三元组, 而向右节点则变为状态 MR、IR 和 D 的三元组.

当成对节点扩展时, 它们必须考虑插入和删除的几种可能性. 一个删除可能会移除碱基配对中的两个碱基, 或仅仅是5'或3'的一方, 使剩下的一个非配对碱基成为凸起. 碱基配对茎上的插入可能发生在配对的5'端、3'端或两端. CM 将一个成对节点扩展到六种状态: 一个 MP 状态、一个 D 状态(用于碱基配对的完全删除) 、ML 和 MR 状态(用于单碱基删除, 分别移除3'或5'的碱基) 以及 IL 和 IR 状态, 分别允许配对的5'或3'端上的插入.

根的起始节点扩展到一个起始状态S 以及对5'和3'端的插入状态 IL 和 IR. 分叉下方的左子起始节点恰好扩展到一个单 S 状态, 而分叉下方的右子起始节点则扩展到一个 S 状态和一个左插入状态 IL. 如此安排插入状态确保我们可以明确地指定任意位置的插入. [6]

一致树中的分叉节点和结尾节点在 CM 中简单地变为 B 和 E 状态.

然后, 状态被状态转移连接起来. 和 profile HMM 一样, 状态能够连接当前节点的所有插入状态以及下一个节点中的所有非插入状态. 插入状态有一个到自己的状态转移, 并允许多于一个碱基的插入. 在成对节点处, IL 能连接到 IR, 但反之不然, 所以我们可以明确地将插入指定到贯穿模型的单一路径上. 图10.13形象地总结了状态转移的连通性.

完全 CM 是状态的有向图, 它根据作为其基础的一致树组织起来. CM 的"主线" 就是该一致树, 但 CM 同时也允许路径通过状态处理删除和插入.

这只是对 RNA 结构 profile 的一种可能设计. 茎中的插入通常是碱基配对而不是单链(即茎的长度可变). 因此我们可以选择在成对节点处包含一个成对插入 (IP) 状态. 我们可以从成对节点中移除 ML 和 MR 状态, 并把一个配对中单个碱基的删除(它们会留下一个凸起) 建模为完全的删除紧跟带 IL 或 IR 状态的凸起插入. 一个精密的设计甚至可能尝试对这样的事实建模: RNA 中的长插入经常是包含结构的. 假设给定无限的计算资源, 我们可以想象使用如第1部分中所描述的广义 SCFG 的 RNA 折叠模型替代每个插入状态.

从 RNA 联配构建 CM

给定 RNA 序列联配和一致二级结构的注释, 并且给定哪些列应被视为插入以及哪些列应被视为一致列, 我们就可以精确地定义一个 CM 并且可以马上将其构造出来. 使用联配中非插入列上的结构注释, 我们首先可以构造一棵一致结构树. 然后用 CM 的状态填充树的节点, 并且通过上面介绍的方法用状态转移连接各个状态.

[6] 另外一种可能的安排是, 左子节点扩展到 S 和 IR, 而右子节点扩展到 S; 这种安排是因为面对选择时, 我们采纳与 profile HMM 相似的向左生成. ——原注

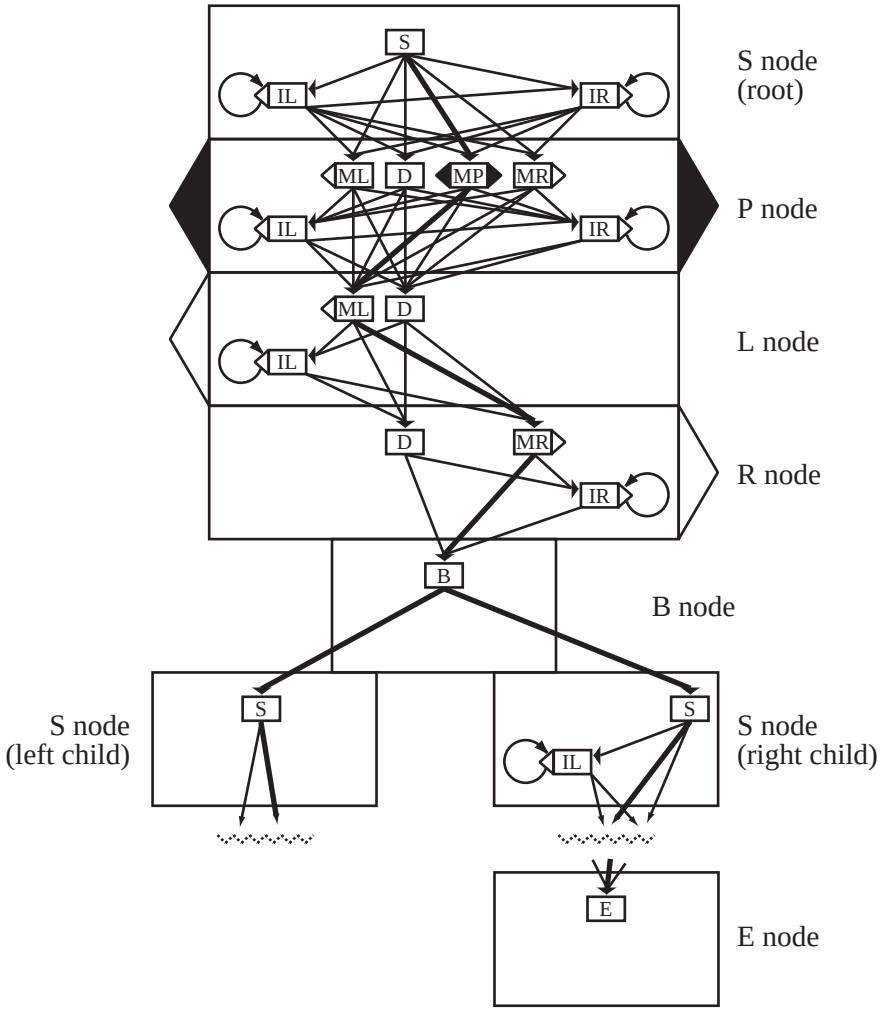


图 10.13: 根据一致 RNA 结构树的节点(大方框) 为 CM 的状态分组. 此例使用一个严格的假设, 这样做只是为了在同一个图里放入所有八种 CM 节点. 状态转移用箭头表示. 一致路径的"主线" 用粗线表示. 这条主线就是一致树本身. 底部的锯齿线表示模型中没有显示的更多节点.

基于联配列到树节点的这种分配, 单个符号就能被指定到单个 CM 状态上, 而且联配中的每条序列就都能被指定一棵唯一的 CM 分析树. 我们随后对这些分析树上的发射与转移事件进行计数. 使用这些观测计数按通常方式为 CM 估计符号发射和状态转移的概率, 一般将 Dirichlet 或混合 Dirichlet 先验包含在内并进行均值的后验估计. 因为 CM 结构直接反映了一致 RNA 结构, 所以该过程明确而又迅速.

如果一致列没有经过注释, 那么一种简单的启发式操作将用于分配, 例如将任意空位字符多于50%的列称为插入列. 如果一致结构未知, 那么这就是一个比较困难的问题; 我们将在本章后面详细讨论.

这样, 我们就可以从一个结构被注释的 RNA 多序列联配出发建立 CM. 我们现在转而描述一些联配算法, 它们的用途包括搜索数据库、构建新的基于结构之联配、以及从以前未联配上和未确定结构的序列出发训练 CM.

CM 联配算法

第9章中我们给出了通用的 SCFG 算法, 并将其应用到 Chomsky 标准形式的

SCFG. 但就如我们前面所讨论的一样, 在 RNA 分析中使用 Chomsky 标准形式的 SCFG 将会十分繁琐. Chomsky 标准形式不允许形如 $W \to cWg$ 的成对产生式法则. 因此我们会针对 CM 给出一组类似的联配算法.

记号

一个 CM 由 M 个不同的状态(非终端字符) 组成, 记为 $W_1, W_2, \ldots, W_M$. 令 v、y 和 z 分别是状态 W_v、W_y 和 W_z 的下标. 我们有7种类型的状态标记P 、L 、R 、D 、S 、B 和E , 依次表示成对发射、向左 (5') 发射、向右 (3') 发射、删除、起始、分叉和结尾状态. W_1 是整个CM 的(根) 起始状态. 通常有多个结尾状态, 因为 CM 通常是反映多分支 RNA 二级结构的多分支模型. 表10.1列出了这7种状态, 并将其与符号发射概率和状态转移概率联系到一起.

状态(s_v)	产生式	Δ_v^L	Δ_v^L	发射	转移
P	$W_v \to x_i W_y x_j$	1	1	$e_v(x_i, x_j)$	$t_v(y)$
L	$W_v \to x_i W_y$	1	0	$e_v(x_i)$	$t_v(y)$
R	$W_v \to W_y x_j$	0	1	$e_v(x_j)$	$t_v(y)$
D	$W_v \to W_y$	0	0	1	$t_v(y)$
S	$W_v \to W_y$	0	0	1	$t_v(y)$
B	$W_v \to W_y W_z$	0	0	1	1
E	$W_v \to \epsilon$	0	0	1	1

表 10.1: 协方差模型的七种状态

我们定义 Δ_v^L 和 Δ_v^R 为通过状态 v 向左和向右发射的字符个数. 这简化了算法描述. 我们在第6章 Sankoff-Cedergren 的 N 维动态规划算法描述中也使用过相似的简化记号.

为了记号和实现的简便, 每个 CM 中的状态还包含另外的信息. 令 s_v 是**状态类型** (state type), 它从 P、L、R、D、S、B 或 E 中取值并表示产生式法则的七种可能形式之一. 令 $\mathscr{C}_v$ 是状态的**子状态** (children), 它由一个或多个状态 W_y 之下标 y 的列表来表示, 而状态 W_v 可以经过状态转移到达状态 W_y. 令 $\mathscr{P}_v$ 为状态的父状态 (parent), 它由一个或多个状态 W_y 之下标 y 的列表来表示, 而状态 W_y 能转移到状态 W_v.

分叉 (B) 状态在 CM 中被特殊处理. 一个分叉状态 W_v 总以概率1转移到两个 S 状态 W_y 和 W_z, 而 y 和 z 都只能有一个选择. B 状态的子状态列表 $\mathscr{C}_v$ 是这两个 S 子状态之 (y, z) 对. 这两个 S 子状态的父状态列表 $\mathscr{P}_y$ 和 $\mathscr{P}_z$ 都是 $\{v\}$, 因为只有 W_v 能转移到这两个状态. 用于分叉转移的这种单一选择性与 Chomsky 的标准形式差别很大, 在 Chomsky 标准形式中我们几乎全都将其描述为对所有 y 和 z, 按概率随机地选择分叉法则 $W_v \to W_y W_z$. 在 RNA 模型中, 分叉状态只需要用来描述结构中出现的多分支环或多茎情况. 模型的大部分由 P、L 和 R 状态所组成. 这些施加在分叉上的限制大大地降低了 CM 算法的计算复杂度.

在联配算法中, 起始状态 (S) 和删除状态 (D) 被等同对待. 它们间仅有的不同存在于结构上. 起始状态只发生在分叉的直接子状态, 或作为根状态 W_1 出现; 而删除状态则发生在 CM 的P、L和R节点中.

存在 CM 的另外三种约束. 第一, 每个状态可能只使用一种产生式法则(s_v 既指状态又指产生式类型). 第二, 与在 profile HMM 中一样, 状态并不是完全相互连接 (interconnected) 的; $\mathscr{C}_v$ 中连接的状态数目是常数, 与状态的数目 M 无关. 与 Chomsky 标准形式的 SCFG 相比, 这进一步降低了联配算法的计算复杂度. 最后我们给出一种最重要的约束. 状态编号使得对所有的 $y \in \mathscr{C}_v$ 都有 $y > v$; 除插入状态外, 对所有 $y \in \mathscr{C}_v$ 都有 $y \geq v$. 该条件对非发射状态 (S , D , B) 意义重大, 因为它确保了不存在非发射的循环. 对 HMM 中的删除状态, 我们也描述了相似的约束 (第3章).

现在考查操控 RNA 协方差模型的重要算法.

计分: 内向算法

给定一条 RNA 观测序列 x, 它由 L 个符号 $x_1,\ldots,x_i,\ldots,x_j,\ldots,x_L$ 组成. 首先考虑**计分** (scoring)) 问题, 即在给定协方差模型 θ 的情况下, 计算对 x 之所有可能结构求和的序列似然 $P(x|\theta)$. 此概率可用内向算法求出.

内向算法用 $\alpha_v(i,j)$ 递归地填充三维动态规划矩阵. $\alpha_v(i,j)$ 是子序列 $x_i,\ldots,x_j$ 以 v 为根的所有分析子树之概率和. $\alpha_v(i+1,i)$ 是长度为0的空子序列概率; 因为存在非发射状态D、S和B, 所以我们必须包含它作为边界条件. 为了记号方便, 我们对所有发射概率均使用 $e_v(x_i,x_j)$: L 状态的 $e_v(x_i,x_j)=e_v(x_i)$, R 状态的 $e_v(x_i,x_j)=e_v(x_j)$, 而非发射态的 $e_v(x_i,x_j)=1$.

算法: CM 的内向算法

起始: 从 $j=0$ 到 L, $v=M$ 到 1;

$$\alpha_v(j+1,j)=\begin{cases} s_v=E: & 1;\\ s_v\in S,D: & \sum\limits_{y\in\mathscr{C}_v} t_v(y)\alpha_y(j+1,j);\\ s_v=B: & \alpha_y(j+1,j)\alpha_z(j+1,j);\\ s_v\in P,L,R: & 0. \end{cases}$$

递归: 从 $j=1$ 到 L, $i=j$ 到 1, $v=M$ 到 1:

$$\alpha_v(i,j)=\begin{cases} s_v=E: & 0;\\ s_v=P, j=i: & 0;\\ s_v=B: & \sum\limits_{k=i-1}^{j}\alpha_y(i,k)\alpha_z(k+1,j);\\ \text{否则}: & e_v(x_i,x_j)\sum\limits_{y\in\mathscr{C}_v} t_v(y)\alpha_y(i+\Delta_v^L,j-\Delta_v^R). \end{cases}$$ ◁

当算法完成时, 概率 $P(x|\theta)$ 就在 $\alpha_1(1,L)$ 中. 如果有 b 个分叉状态和 a 个其他状态 ($M=a+b$), 那么算法的空间复杂度为 $O(L^2M)$, 时间复杂度为 $O(aML^2+bML^3)$.

CM 的外向算法

为了实现下一节的内向-外向参数估计, 我们需要外向算法. 外向算法计算值 $\beta_v(i,j)$, 它是所有根在状态 v 且能生成除了子序列 $x_i,\ldots,x_j$ 以外之完全序列 x 的所有分析树概率. 外向算法需要首先计算内向变量 α. [7]所有三维动态规划矩阵的单元格均被初始化为0, 外向算法是:

算法: CM 的外向算法

起始:

$\beta_1(1,L)=1$.

递归: 从 $i=1$ 到 $L+1$, $j=L$ 到 $i-1$, $v=2$ 到 M:

$$\beta_v(i,j)=\begin{cases} \text{对}s_v=S,\mathscr{P}_v=y,\mathscr{C}_y=\{v,z\}:\\ \quad\sum\limits_{k=j}^{L}\beta_y(i,k)\alpha_z(j+1,k);\\ \text{对}s_v=S,\mathscr{P}_v=y,\mathscr{C}_y=\{z,v\}:\\ \quad\sum\limits_{k=1}^{i}\beta_y(k,j)\alpha_z(k,i-1);\\ \text{对}s_v\in P,L,R,D,B,E:\\ \quad\sum\limits_{y\in\mathscr{P}_v} e_y(x_{i-\Delta_y^L},x_{j+\Delta_y^R})t_y(v)\beta_y(i-\Delta_y^L,j+\Delta_y^R). \end{cases}$$ ◁

外向算法的空间和时间复杂度与内向算法相同.

CM 参数的内向-外向期望最大化

如果模型的结构已知(即这个家族的一致 RNA 结构已知) 但概率参数未知, 那么概率参数可以使用一种称为内向-外向算法的期望最大化方法从未联配的示例RNA 中估计出来. 在实践中, 人们很少使用这种算法. 几乎所有的 RNA 一致结

[7]实际上只需对 $s_v=S$ 时的 α_v 保存内向路径. 这在内存有限时很有用. ——原注

构都通过比较序列分析从多序列联配中导出, 而且我们此前也描述了如何将结构已注释的多序列联配立即转化为参数化的 CM. 但出于完备性考虑, 我们仍然要给出内向-外向算法的 CM 版本. 可以想像这样的情况: 一致结构由另外的手段获得. 而且, 我们可能也不希望假设多序列联配的结果完全正确, 这样我们可能就不想直接使用联配中的计数信息, 而使内向-外向算法更合适.

序列 x 的一个推导中, 在 i,j 处使用状态 v 的概率是 $\frac{1}{P(x|\theta)}\alpha_v(i,j)\beta_v(i,j)$. 在多个方向上对这些项求和, 我们就可以获得许多有用的概率或期望计数, 其中就包括发射与转移概率参数的 EM 重估计过程所需的期望计数, 就如我们对 Chomsky 标准形式 SCFG 所做的一样.

单条序列 x 中状态 v 被使用 (used) 的期望次数是

$$c(v \text{ used}) = \frac{1}{P(x|\theta)} \sum_{i=1}^{L+1} \sum_{j=i-1}^{L} \alpha_v(i,j)\beta_v(i,j).$$

单条序列 x 中转移 $t_v(y)$ 被使用的期望次数是

$$c(v \to y \text{ used}) = \frac{1}{P(x|\theta)} \sum_{i=1}^{L+1} \sum_{j=i-1}^{L} \beta_v(i,j)e_v(x_i,x_j)t_v(y)\alpha_y(i+\Delta_v^L, j-\Delta_v^R).$$

当用 N 条独立观测序列 $x^1,\ldots,x^h,\ldots,x^N$ 而不是单条序列作训练集时, 就像我们通常从一个 RNA 序列家族出发训练模型那样, 期望数是在单条序列上的求和, 而且需要使用为每条序列计算的内向、外向变量 α^h 和 β^h:

$$c(v \to y \text{ used}) = \sum_{h=1}^{N} \frac{1}{P(x^h|\theta)} \sum_{i=1}^{L+1} \sum_{j=i-1}^{L} \beta_v^h(i,j)e_v(x_i^h,x_j^h)t_v(y)\alpha_y^h(i+\Delta_v^L, j-\Delta_v^R).$$

当给定 N 条训练序列 $x^1,\ldots,x^N$ 时, 从状态 v 到状态 y 的 CM 转移概率之内向-外向 EM 重估计方程是[8]

$$\hat{t}_v(y) = \frac{\sum_{h=1}^{N} \frac{1}{P(x^h|\theta)} \sum_{i=1}^{L+1} \sum_{j=i-1}^{L} \beta_v^h(i,j)e_v(x_i^h,x_j^h)t_v(y)\alpha_y^h(i+\Delta_i^L, j-\Delta_v^R)}{\sum_{h=1}^{N} \frac{1}{P(x^h|\theta)} \sum_{i=1}^{L+1} \sum_{j=i-1}^{L} \alpha_v^h(i,j)\beta_v^h(i,j)}.$$

同理可得状态 v 的CM 发射概率之内向-外向重估计方程, 其中表达式 $\delta()$ 取1, 如果括号中的条件满足; 否则取0:

当 $s_v = P$ 时:

$$\hat{e}_v(a,b) = \frac{\sum_{h=1}^{N} \frac{1}{P(x^h|\theta)} \sum_{i=1}^{L} \sum_{j=i}^{L} \delta(x_i^h, x_j^h = a,b)\beta_v^h(i,j)\alpha_v^h(i,j)}{\sum_{h=1}^{N} \frac{1}{P(x^h|\theta)} \sum_{i=1}^{L} \sum_{j=i}^{L} \beta_v^h(i,j)\alpha_v^h(i,j)};$$

当 $s_v = L$ 时:

$$\hat{e}_v(a) = \frac{\sum_{h=1}^{N} \frac{1}{P(x^h|\theta)} \sum_{i=1}^{L} \sum_{j=i}^{L} \delta(x_i^h = a)\beta_v^h(i,j)\alpha_v^h(i,j)}{\sum_{h=1}^{N} \frac{1}{P(x^h|\theta)} \sum_{i=1}^{L} \sum_{j=i}^{L} \beta_v^h(i,j)\alpha_v^h(i,j)};$$

当 $s_v = R$ 时:

$$\hat{e}_v(a) = \frac{\sum_{h=1}^{N} \frac{1}{P(x^h|\theta)} \sum_{i=1}^{L} \sum_{j=i}^{L} \delta(x_j^h = a)\beta_v^h(i,j)\alpha_v^h(i,j)}{\sum_{h=1}^{N} \frac{1}{P(x^h|\theta)} \sum_{i=1}^{L} \sum_{j=i}^{L} \beta_v^h(i,j)\alpha_v^h(i,j)}.$$

[8]在定义中状态B 的转移概率为1, 因此不再需要重估计. ——原注

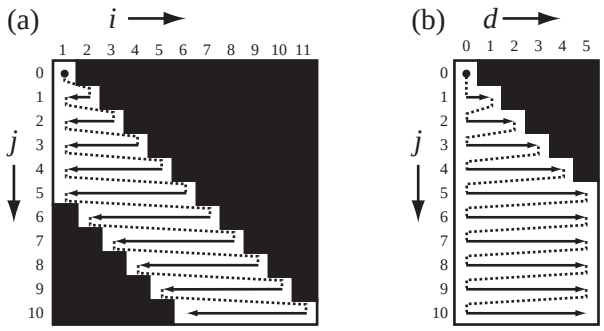


图 10.14: (a) 标准 CYK 动态规划矩阵的其中一层, 其中数据库序列长度 $L = 10$, 且矩阵用起始位置 i 和结尾位置 j 作下标. (三维矩阵中共有 M 层, 每一层都对应模型中的一个状态.) 白色显示了当限制最大匹配子序列长度为 $D = 5$ 时, 矩阵中需要计算的部分. 箭头显示了在彻底检查数据库序列(即递增 j) 的搜索算法中, 矩阵单元格的计算顺序. (b) 相同 CYK 计算的另一种坐标系统, 它以结尾位置 j 和子序列长度 d 为下标, 其中 $d = j - i + 1$. 当要求此坐标系统下的内存用量与 L 无关时, 我们更容易实现平滑扫描的 CYK 数据库搜索算法, 因为 (b) 中的矩阵大小是 $D \times L$ 而非 $L \times L$.

内向-外向产物也能用于估计其他有趣的量. 例如在单条序列 x 中, x_i 和 x_j 通过任意成对产生式发生碱基配对 (paired) 的概率是:

$$P(x_i, x_j \text{ paired}) = \frac{1}{P(x|\theta)} \sum_{v|s_v=P} \alpha_v(i,j)\beta_v(i,j).$$

数据库搜索: CYK 算法

假设给定一条很长的序列(例如整个基因组), 而我们的任务是找到一条或多条匹配 RNA 模型的子序列. 我们已经给出的算法很适合全局联配, 但却不适合从一条或多条子序列到 CM 上的局部联配. 我们当然不想让数据库搜索算法的时间和空间需求达到数据库序列长度 L 之平方或立方的水平. 将最长的已联配子序列之长度限制到小于常数 D, 并对动态规划矩阵的坐标系统进行转换后, 我们便可以实现用于序列数据库搜索的高效 CYK (或内向, 或外向) 算法. 动态规划矩阵用 v, j, d 而不是 v, i, j 作下标, 其中 d 是子序列 $i, \ldots, j$ 的长度 ($d = j - i + 1$) 且 $d \le D$. 图10.14显示了如何使得此新坐标系统可以直接迭代地计算长度为 $0, \ldots, D$、并以序列位置 j 为结尾之子序列最优联配的一行分值.

用于 SCFG 联配的标准 CYK 算法返回给定模型 θ 的序列 S, 以及最优分析 $\hat{\pi}$ 的概率 $P(S, \hat{\pi}|\theta)$ 之对数. 该分值是强烈依赖于已联配序列长度的函数, 这潜在地提高了从数据库里不同长度之交叠 (overlapping) 子序列中选出最优匹配子序列的难度. 如同在 HMM 中讨论的那样, 解决这个问题的一种好方法是计算之于随机序列"零" (null) 模型的对数几率分值. 如果随机模型是一个独立同分布模型, 零假设下序列似然是单残基频率 f_a 的乘积, 那么与 HMM 联配计分类似, 对数概率发射项就可以被对数几率碱基对分值或孤立核苷酸分值所替换, 使 CYK 算法直接得到对数几率分值. [9]在下面的算法中, 我们使用记号 $\log \hat{e}$ 表示对数几率发射分值而非对数概率值 $\log e$:

$$\begin{aligned}
&\text{当} s_v = P\text{时} \quad \log \hat{e}_v(a,b) = \log(e_v(a,b)/f_a f_b);\\
&\text{当} s_v = L\text{时} \quad \log \hat{e}_v(a,b) = \log(e_v(a)/f_a);\\
&\text{当} s_v = R\text{时} \quad \log \hat{e}_v(a,b) = \log(e_v(b)/f_b).
\end{aligned}$$

[9]为完成此全概率论模型, 我们还应该为该随机模型指定一个长度分布, 但这一项通常被忽略. ——原注

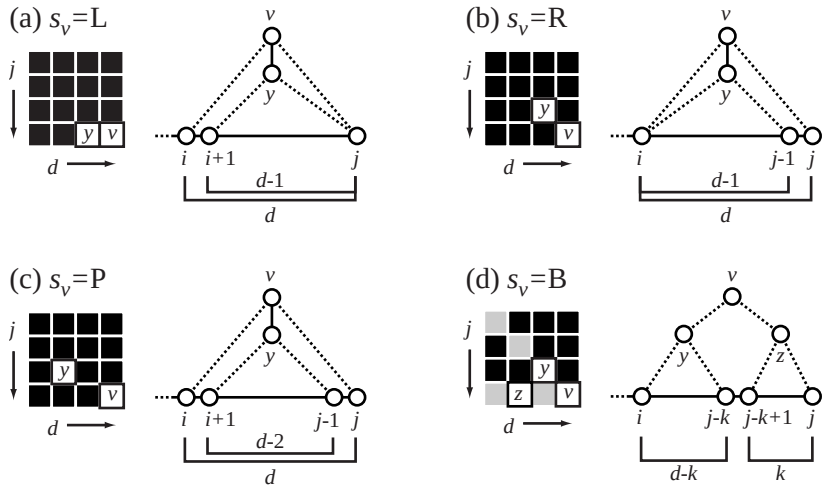


图 10.15: 为四种不同的状态显示 CYK 数据库搜索算法的递归步骤. 图中只显示了一层三维动态规划矩阵. 例如在左上方, 标记为 v 的单元格的值 $\gamma_v(j,d)$ 依赖于一个或多个标记为 y 的可能单元格的值 $\gamma_y(j,d-1)$, 其中不同的状态 y 与状态 v 相连接. 动态规划矩阵右侧的图用不同方法显示了该过程, 即如果 v 向左产生单个残基, 那么我们就将以状态 v 为根、以 j 结尾且长度为 d 的子序列之分析子树添加到这三个参数分别为 $y, j, d-1$ 的子树中. 对 R (右上图) 和 P (左下图) 状态的计算也类似. 当 v 是一个分叉 (B) 状态时, 该计算依赖于对最优分叉点的选择. 我们用动态规划矩阵中标为 y 和 z 的两个单元格显示这种点; 灰色显示了所有其他此类相连单元的集合.

CYK 数据库搜索算法如下:

算法: CM 数据库搜索的 CYK 算法

起始: 从 $j=0$ 到 L, $v=M$ 到1:

$$\gamma_v(j,0) = \begin{cases} s_v = E\text{时}: \\ \quad 0; \\ s_v \in D, S\text{时}: \\ \quad \max_{y\in\mathscr{C}_v}[\gamma_y(j,0) + \log t_v(y)]; \\ s_v = B, \mathscr{C}_v = (y,z)\text{时}: \\ \quad \gamma_y(j,0) + \gamma_z(j,0); \\ \text{否则}: \\ \quad -\infty. \end{cases}$$

递归: 从 $j=1$ 到 L, $d=1$ 到 D (并且 $d \le j$), $v=M$ 到 1:

$$\gamma_v(j,d) = \begin{cases} s_v = E\text{时}: \\ \quad -\infty; \\ s_v = P\text{并且}d<2\text{时}: \\ \quad -\infty; \\ s_v = B, \mathscr{C}_v = (y,z)\text{时}: \\ \quad \max_{0\le k\le d}[\gamma_y(j-k, d-k) + \gamma_z(j,k)]; \\ \text{否则}: \\ \quad \max_{y\in\mathscr{C}_v}[\gamma_y(j-\Delta_v^R, d-\Delta_v^L-\Delta_v^R) + \log t_v(y)] \\ \qquad + \log \hat{e}_v(x_i, x_j). \end{cases}$$

◁

图10.15展示了该算法中递归关键步骤的图解.

某些进一步的实现细节是十分重要的. 与初始化所有 L 行不同, 更好的办法是将初始化步骤和迭代步骤合并起来, 每次沿着第 j 行移动, 首先初始化 $\gamma_v(j,0)$, 然后对 $d=1,\ldots,D$ 计算 $\gamma_v(j,d)$. (因为对长度为0的子序列所进行的初始化计算独立于序列, 所以对所有 v 而言, $\gamma_v(0,0)$ 只需要计算一次, 而这些值可以被复制并用于

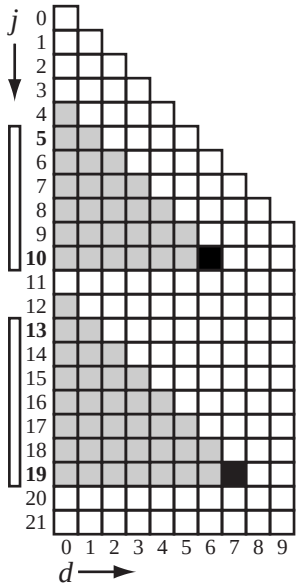


图 10.16: CYK 数据库搜索矩阵中的两个无交叠匹配. 这两个匹配 (从位置5到10和13到19) 用矩阵左侧的两个方框表示. 高分矩阵元 $\gamma_0(j,d)$ 用黑色表示. 必须放到内存中以便重建整条联配的矩阵部分用灰色表示. 灰色三角形同时也表明了单元格 (j,d) 的坐标如何决定每个匹配事件的起始点 i, 即使整个矩阵并没保存在内存中. 为简单起见, 图中只画出了一层三维矩阵.

初始化随后的 $\gamma_v(j,0)$.) 在上面的算法和图10.15中, 显然除去分叉状态 B 外, 第 j 行的所有分值都只依赖于第 j 和第 $j-1$ 行的分值. 而分叉状态只依赖于前 D 行的起始状态分值, 因为在 CM 结构的定义中 B 状态总是分叉为两个 S 状态. 于是只有 $D+1$ 行 S 状态分值和两行其他状态的分值需要被放入内存.

在第 j 行中, 分值 $\gamma_0(j,d)$ 是到以位置 j 结尾的模型(即对一棵始于根状态 $v=0$ 的分析树) 之完全联配的对数几率分值. 如图10.16所示, 匹配的起始点 i 能从 d 中算出来: $i=j-d-1$. 这也就是说, 获得联配的起始点并不需要对动态规划矩阵进行回溯. 找到一个高分值的 $\gamma_v(j,d)$ 后, CYK 搜索算法不仅能立即输出这个分值, 而且还能输出给出这个高分值的子序列之起始位置 i 以及结尾位置 j.

一种实现可能会报告所有高于某个分数阈值的匹配. 然而一个高分联配会带有"阴影区" (shadow), 即在起始和结尾点处带微小差异的联配也能获得较好的分值. 因此较好的方法是报告大于阈值的非交叠匹配. 下面介绍一种简单的分值后处理(post- processing) 算法, 它可以用固定大小的内存扫描无限数量的序列. 在计算完行 j 之后, 行 j 中对某 d 的最好分值 $\gamma_0(j,d)$ 就被确定了. 如果它比报告阈值大, 则将其存入一个列表; 如果它和列表中某个已存在的匹配交叠, 那么分值较低的匹配就被删除; 列表中所有结尾点 j 小于当前最小起始点 $j-D$ 的匹配都被报告为非交叠匹配.

CYK 搜索算法的时间复杂度是 $O(M_aLD+M_bLD^2)$, 其中 M_a 是模型的非分叉状态, M_b 是模型的分叉状态, L 是数据库残基长度, D 是残基最大匹配数. 算法的空间复杂度是 $O(M_aD+M_bD^2)$. 计算时间随数据库的增大而线性增长, 而内存需求则与数据库大小无关.

练习

10.11 另一种相同的矩阵坐标系统可以应用到内向-外向算法中. 与 CYK 相比, 内向算法的优势在于它对子序列所有可能的结构和联配之概率进行求和, 但又不比 CYK 版本的计算复杂度大. 请给出一个内向算法用于搜索长度不大于 D 的局部子序列匹配.

结构联配: 带回溯的 CYK 算法

因为出于对内存效率的考虑, 矩阵中的大部分元素都被舍弃掉了, 所以我们不能像前面所描述的搜索算法一样进行回溯. 因此我们无法恢复联配; 我们只能恢复分值和起始结尾的位置. 如果多花费一些内存, 此算法就能马上修改为能够进行回溯并且能够恢复匹配子序列的最优 SCFG 分析树. 因为 CM 结构反映 RNA 的一致二级结构, 所以 CM 的分析树既表现了到模型的最优联配, 也表现了最优二级结构预测. 分析树中指派为 P (成对) 的状态表示预测出的碱基配对.

实现 CYK 的回溯时, 我们既可以使用回溯指针的第二个矩阵, 也可以重建分值计算过程, 就像第2章中所讨论的其他动态规划算法一样. 所有 $D+1$ 行的回溯指针或分值都必须保留在内存中以保证可以回溯匹配. 如果使用前面的交叠处理算法, 那么就要在内存中保留 $2D+1$ 条完整行, 因为在决定一个匹配与后面的匹配不交叠之前, 我们还需要额外的 D 行. 如果我们既需要分值也需要联配, 那么同时实现局部 CYK 搜索算法与带回溯的全局 CYK 联配算法将会相当有效, 并且此方法可以简单地分为两步: 首先进行一个局部搜索过程以找到匹配上的子序列; 第二步在带回溯的全局联配模型中, 每次都把每条这样的子序列最优地联配到模型.

我们从长度为 d 并以 j 结尾的高计分子序列之 $\gamma_0(j,d)$ 开始回溯, 然后反向运行. 对于序列的全局联配而非局部联配来讲, 回溯过程的起点是 $\gamma_0(L,L)$. 该过程本质上和用于恢复 HMM 动态规划的回溯, 或本章前面介绍的从更简单的 RNA 模型出发的 SCFG 回溯是相同的, 因此我们不再给出详细阐述.

练习

10.12 修改 CYK 算法, 使它能在每个矩阵元素中保存回溯信息以协助恢复最优分析树. 从一个分叉状态回溯, 需要保留的最小信息是多少? 如果从其他状态回溯, 需要保留的最小信息又是多少?

使用 CM 的"自动"比较序列分析

假设给定一组未联配的 RNA 序列, 它们的一致二级结构未知. 多序列 RNA 联配和一致二级结构预测的组合属于比较序列分析的范畴, 这在很大程度上仍然由手工完成. 我们已经给出了内向-外向训练算法, 但前提是我们已知模型的结构, 由此能得到 RNA 家族的一致结构. 我们还给出一个算法, 不需要知道一致结构而从多序列联配出发构建 CM . 我们现在将描述如何把这两种想法结合成一个自动化的比较序列分析算法.

基本想法是在下面两个步骤间迭代: (a) 对给定的当前联配构建最优(或近似最优的) CM 结构; 和(b) 对给定的当前 CM 构建最优多序列联配.

对(a), 在给定的 RNA 多序列联配下, 我们有几种可能的方法来找到一致结构. 受比较序列分析方法的直接影响, Eddy & Durbin [1994] 使用了一种启发式方法. 他们首先对所有已联配列中的残基配对计算互信息项 M_{ij}. 然后使用动态规划折叠算法(本质上就是 Nussinov 算法) 寻找使 M_{ij} 项之和最大的结构树. 寻找最大和 S_{ij} 的算法填充步骤如下:

$$S_{i,j} = \max \begin{cases} S_{i+1,j} & \text{列}i\text{未配对;} \\ S_{i,j-1} & \text{列}j\text{未配对;} \\ S_{i+1,j-1} + M_{i,j} & \text{列}i,j\text{配对.} \\ \max_{i<k<j} S_{i,k} + S_{k+1,j} & \text{分叉.} \end{cases}$$

回溯这个矩阵就能得到一致结构树, 它能扩展为一个 CM. 该方法的好处是通过使用互信息项, 只有被比较分析支持得最好的那些碱基才能形成配对. 该方法的缺点之一是过配对 (overpair), 因为 $M_{ij} \geq 0$ 所以即使增加一个假 (spurious) 配对, S_{ij} 也会有少许增加. 第二个缺点是因为此方法只能寻找共变 (covariation), 因此高度保守结构或序列变动很小的结构域很可能会被错误预测.

随后用一致结构为每条联配上的序列找到各自的分析树, 从这些分析树上收集转移和发射计数, 并按通常的方式把这些计数转换为概率参数.

还有一种严格的 CM 构造算法, 它使用未注释的 RNA 多序列联配, 同时能推导出一个最优的(最大后验, maximum *a posteriori*) CM 结构及其参数[S . R . Eddy , 未发表]. 此算法是第5章中用于 profile HMM 之 MAP 构造算法的一个扩展.

对 (b), 为了在给定当前模型下构造一个最优的多序列联配, 我们可以对每条序列都应用 CYK 联配算法. 与 HMM 中一样, 每条序列到一个共同模型的单独联配都意味着这些序列间的多序列联配.

这类似于 EM 算法, 只是在期望步骤中用 CYK 算法代替了内向-外向算法, 并且在最大化步骤中使用了一个模型构建算法(此算法同时重估计模型的结构和参数, 而不只是估计参数).

这种训练算法与人工比较序列分析工作十分相似. 在某些情况下它甚至能给出相同的答案. 此算法首先猜测未联配上的序列之联配(或随机, 或基于序列的联配), 然后算法对结构作初始猜测, 然后根据这个猜测重新联配所有序列, 接着对结构作出一个新的猜测, 迭代下去直到算法收敛于一致解. 对于理想数据集, 如 Eddy & Durbin [1994] 中的100条转运 RNA 序列, 这种办法很有效.

然而现实中很难找到如此理想的数据集. 要完成工作, 算法需要大量的短 RNA 序列; 必须有足够的一级序列分化使共变能够揭示大多数的碱基配对; 一致二级结构必须高度保守; 并且这些序列必须能构成全局联配, 而不仅仅是局部联配. 此外, 该算法倾向于局部最优, 但它的计算密集程度已经如此之高, 以至于再要求使用第5章中对 profile HMM 的模拟退火方法就毫无吸引力. 尽管上述都只是是需要突破的技术极限, 但必须说明基于 SCFG 分析方法的自动化比较分析仍然更偏向于理论而非实践. 协方差模型方法最实际的应用是在数据库中搜索同源的 RNA 结构, 下一小节会讨论这个问题.

CM 算法实际应用的一个例子

为了更好地描述如何将本理论应用到实际的序列分析问题中, 我们有必要考查协方差模型的实际应用.

在许多基因组中最大的基因家族不是蛋白质基因家族而是结构 RNA ——转运 RNA (transfer RNA , tRNA) 基因家族.例如, 在啤酒酵母*Saccharomyces cerevisiae*中有274个 tRNA 基因, 而在人类基因组中有大约1500个不同的 tRNA 基因. 人们编写了许多程序用于在基因组中寻找 tRNA 基因. 这些 tRNA 检测程序往往是大规模基因组注释项目的一部分. 这些经手工仔细调整过的程序之假阳性率 (false positive rate) 量级通常在每兆 (megabase , Mb) DNA 碱基中出现0.2-0.4个假预测 [Fichant & Burks 1991; Pavesi *et al.* 1994]. 这样的假阳性率对于像酵母 (14Mb) 这样的小基因组而言是可以接受的, 但对于 3000Mb 的人类基因组来说, 这样的程序会产生大约1000个假阳性, 这就意味着有相当部分的 tRNA 基因被预测错误.

转运 RNA 是检验协方差模型方法的理想候选对象: 因为它们都比较短(通常有75-95个残基); 它们几乎没有共享的一级序列, 但有一种高度保守的"三叶草" (cloverleaf) 二级结构; 并且现在已有数千条 tRNA 序列可以用于训练统计模型.

[Steinberg, Misch & Sprinzl 1993] 从1415条 tRNA 序列的多序列联配出发构建 tRNA 的协方差模型, 这些序列来自不同的组织, 并且包含细胞器和病毒 tRNA. 其中有38条序列在反编码子环上包含短内含子, 所以训练出的模型能够"意识" (aware) 到许多真核 tRNA 基因含有短内含子. 许多不同位置处的长内含子(通常是催化组I和II的内含子) 在联配中被排除了, 因为它们太长以至于不值得用这种算法进行判断.

他们使用被接受的一致二级结构之注释信息, 从这个联配中直接构造模型. 模型构建的计算时间可以忽略不计. 结果模型有285个状态. 它们被分配到一致模型的72个节点上: 3个分叉节点, 28个配对节点, 33个向左节点, 1个向右节点和7个起始节点. 其中这28个配对节点对应于 tRNA 三叶草结构的四个螺旋和可变臂 (variable arm) 螺旋上共28个一致碱基配对.

[Eddy & Durbin 1994] 实现了 CYK 数据库扫描算法 (COVE 程序套件中的 COVELS), 当限制了最大匹配长度 $D = 150$ 后, 它在一台 SGI Indigo2 R4400 工作站上搜索 DNA 的速度达到了每秒20个残基. 搜索矩阵需要大概 500K 的内存空间, 而搜索 tRNA 所需的 CPU 时间与内存要求相比就更有限. 在一个 MasPar 多处理器上并行地实现该算法时, 搜索加速到每秒大约2000个残基, 但这种使用特殊设备的方法并不太实用.

为此 [Lowe & Eddy 1997] 编写了一个称为 `TRNASCAN-SE` 的混合程序, 它使用两

种不同的 tRNA 检测过程作为快速预筛选器 (pre-filter). 当这两种过程中至少有一个提出某候选 tRNA 后, 该 tRNA 就在第二步中和 CM 进行匹配. 最后, 统计显著(对数几率分值 ≥ 20 比特) 的匹配被认为是 tRNA. 表10.2将这个混合程序和其他 tRNA 检测程序以及单纯的 CM 进行了比较.

程序	速度 (碱基/秒)	真阳性 (%)	假阳性 (每Mb)
TRNASCAN 1.3 [Fichant & Burks 1991]	400	95.1	0.37
POL3SCAN [Pavesi *et al.* 1994]	373 000	88.8	0.23
单纯 CM [Eddy & Durbin 1994]	20	99.8	<0.002
TRNASCAN-SE [Lowe & Eddy 1997]	30 000	99.5	<0.000 07

表 10.2: tRNA 基因检测方法的比较

与其他两种人工调整方法相比, TRNASCAN-SE 采用了基于 CM 的(总体上) 自动化方法, 因此在特异性 (specificity) 方面有很大改进, 而在敏感性 (sensitivity) 上也有少许提高. 在 3000Mb 的人类基因组中, 假阳性 tRNA 的期望数目从多于一千下降到了不到一个.

除了特异性和敏感性, CM 方法的另一优势是它的通用性 (generality) ——我们可以对任意 RNA 序列联配构建此模型. 例如在 TRNASCAN-SE 中, 我们可以很容易地从硒半胱氨酸 tRNA (selenocysteine tRNA, 与大多数 tRNA 在许多方面都存在不同) 的联配出发构造一个专门的额外模型, 使程序能检测这些变体基因.

CM 方法应用到 tRNA 序列分析中的最主要不足是它对内存和 CPU 的要求较高. 在 TRNASCAN-SE 中, 我们可以使用其他现存的程序作为快速筛选器, 从而克服时间问题. 然而这种方法并不普遍适用, 因为此类程序对大多数其他RNA 家族并不适用. 因此我们需要更好的算法和/或更好的计算机以便将 CM 方法应用于更大的 RNA.

10.4 补充读物

除了动态规划算法外, 许多其他算法也应用于单序列 RNA 结构预测问题, 它们包括遗传算法 (genetic algorithm) [Shapiro & Wu 1996; van Batenburg, Gultyaev & Pleij 1995], 约束满足算法 (constraint satisfaction algorithm) 和 Monte Carlo 抽样算法 (Monte Carlo sampling algorithm) [Abrahams *et al.* 1990; Gultyaev 1991]. 这些算法中有很多都试图预测除经典二级结构之外的假结. 对 RNA 结构建模而言, RNA 假结仍然是算法发展上非常棘手的问题. 通常, 这些算法都不保证能够找到一个最优假结结构. 但是 Cary & Stormo [1995] 所提出的图论最大权重匹配算法 (graph theoretic maximum weighted matching algorithm) 却是一个著名的例外. Brown & Wilson [1995] 描述了一种基于 SCFG 的方法对假结建模, 它使用假结的分离 SCFG 模型与其他结构的交集.

人们已经将单序列二级结构预测算法的精度和手工比较分析的结果进行了系统的对比 [Fields & Gutell 1996; Konings & Gutell 1995]. 一段时间以前, 人们就认识到需要使用树形表现并对RNA 结构进行比较 [Margalit *et al.* 1989; Shapiro & Zhang 1990]. 还有一篇关于" RNA 结构空间" (RNA structure space) 的有趣文献, 它使用理论方法和计算建模来处理二级结构如何限制 RNA 的进化与功能这个问题 [Schuster *et al.* 1994; Schuster 1995].

为 RNA 结构家族和多序列联配建模的基于 SCFG 的方法由 Eddy & Durbin [1994] 和Sakakibara , Haussler 以及在UC Santa Cruz 的同事 [Sakakibara *et al.* 1994] 所提出. Lefebvre [1995; 1996] 开发了与之相当的算法. Corpet & Michot [1994] 描述的非概率论 RNA 结构联配算法与 SCFG 算法有些类似.

第十一章

概率论背景

为使本书自成一体, 我们在最后章整理出前面使用过的概率论思想与方法. 本章各节均相当独立, 可供读者随意翻阅. 本章的某些部分比本书其他章包含更多的数学技巧.

11.1 概率分布

我们在这里介绍本书出现的各种概率分布. 如果观测值属于某个有限集 X, 而我们想为这些观测值赋予概率, 那么概率分布就是为 X 中每个观测值 x 均赋予概率 p_x 的一种赋值. 例如投掷一个公平骰子所得观测值的概率分布, 对其 6 个观测值 $x = 1, \ldots, 6$可以有$p_x = 1/6$.

如果我们有一个连续变量 x (如某物体的重量), 则该变量取特定数值例如重量恰好是 1 磅的概率为 0 . 但 x 在某区间内取值的概率, 如 $P(x_0 \le x \le x_1)$, 可以明确定义且为正值. 随着区间长度趋近于 0, 我们可记 $P(x - \delta x/2 \le x \le x + \delta x/2) = f(x)\delta x$, 其中 $f(x)$ 是一个函数, 称为**概率密度** (probability density) 或简称**密度** (density). 而区间的概率可以通过积分得到: $P(x_0 \le x \le x_1) = \int_{x_0}^{x_1} f(x)dx$. 密度必须满足: 对所有 x, $f(x) \ge 0$, 且 $\int_{-\infty}^{+\infty} f(x)dx = 1$. 但是应注意到, 可以有 $f(x) > 1$. 例如一个明确定义的密度可以是: 当 $0 \le x \le 0.1$时$f(x) = 10$, 而对其他 x 有 $f(x) = 0$.

二项分布

我们考虑的第一个分布也许是最为简单且最为熟悉的**二项分布** (binomial distribution). 它定义在一个有限集上, 该集合由二元 (仅含 "0" 和 "1" 的) 输出实验的 N 次试验的所有可能结果组成. 若 p 为得到 "1" 的概率, $1 - p$ 为得到 "0" 的概率, 则 N 次试验得到 k 个 "1" (k '1' s out of N) 的概率为:

$$P(k \text{ '1' s out of } N) = \binom{N}{k} p^k (1-p)^{N-k}, \tag{11.1}$$

其中 $\binom{N}{k}$ 表示从 N 个物体中选出 k 个的方法数, 即 $N!/((N-k)!k!)$, 阶乘函数对非负整数定义为 $n! = n(n-1)\cdots 1$, 且 $0! = 1$.

对任意分布 P, 均值 (mean) m 与方差 (variance) σ^2 分别定义为 $m = \sum kP(k)$ 及 $\sigma^2 = \sum (k-m)^2 P(k)$. 方差的正平方根 σ 称为标准差 (standard deviation). 对于二项分布有:

$$m = \sum_{k=1}^{N} k \binom{N}{k} p^k (1-p)^{N-k}$$

且

$$\sigma^2 = \sum_{k=1}^{N} (k-m)^2 \binom{N}{k} p^k (1-p)^{N-k}$$

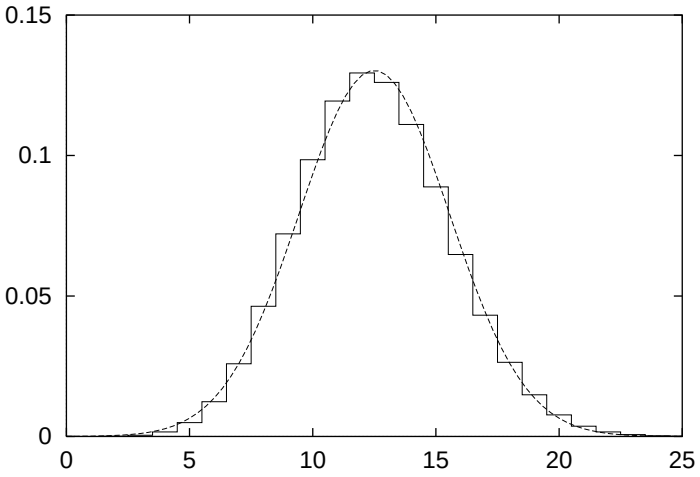


图 11.1: N 很大时二项分布(实线) 之极限趋近于 Gauss 分布(虚线). 此例中式 (11.1) 的 $N = 40, p = 1/4$.

可以证明 $m = Np, \sigma^2 = Np(1-p)$ (见练习 11.1).

练习

11.1 计算二项分布的均值和方差. (提示: 为计算 m, 我们可以求二项展开式 $(p+q)^N = \sum_0^N \binom{N}{k} p^k q^{N-k}$ 关于 p 的微分, 并令 $q = 1-p$. 对于方差, 可以求关于 p 的二阶微分.)

Gauss 分布

考虑令 $N \to \infty$ 时会发生什么情况. 均值与方差都随 N 线性增长, 但是通过定义新的变量 $u = (k-m)/\sigma = (k-Np)/\sqrt{Np(1-p)}$, 我们可以重新标度, 给出固定的均值和方差. 一个经典的结果 [Keeping 1995] 是, 在大量事件的极限下, 二项分布就化为 Gauss 分布(见图11.1), 并且重新标度后的密度为:

$$f(u) = \frac{1}{\sqrt{2\pi}} \exp(-u^2/2) \tag{11.2}$$

这可以视为中心极限定理 (central limit theorem) 的一个特例. 该定理可以描述为: N 个独立随机变量, 若均值与方差归一化成等同, 则当 $N \to \infty$ 时, 其和的分布趋于高斯分布. 若单个变量以 $1-p$ 和 p 的概率分别取值 0 和 1, 则该变量 N 个复本之和的分布为 $P(k) = P(X_1 + \ldots + X_N \le k)$, 正好是前面考虑的二项分布.

多项分布

如果实验以概率 $\theta_i, i = 1, \ldots, K$ 产生 K 个独立结果, 那么二项分布就推广为**多项分布** (multinomial distribution). 出现 n_i 次结果 i 的概率为:

$$P(n|\theta) = M^{-1}(n) \prod_{i=1}^{K} \theta_i^{n_i}. \tag{11.3}$$

这里我们将分布的参数 θ 作为概率的条件, 这在 Bayes 框架中这是很自然的, 因为此时参数本身也是随机变量. 例如, 在经典统计框架下, n 的概率可记为 $P_\theta(n)$. 归一化常数仅依赖于观测结果的总数 $\sum_k n_k$. 对固定的 $\sum_k n_k$ 有:

$$M(n) = \frac{n_1! n_2! \ldots n_k!}{(\sum_k n_k)!} = \frac{\prod_i n_i!}{(\sum_k n_k)!}. \tag{11.4}$$

$k = 2$ 时多项分布将退化为二项分布.

例子: 掷骰子

一个骰子掷 N 次的结果可以由多项分布来描述. 令 6 种结果的各自概率分别为 $\theta_1, \ldots, \theta_6$. 对于 $\theta_1 = \cdots = \theta_6 = 1/6$ 的公平骰子, 投掷 12 次后每种结果均出现两次的概率为:

$$\frac{12!}{2!^6}\left(\frac{1}{6}\right)^{12} = 3.4 \times 10^{-3}.$$

□

Dirichlet 分布

在 Bayes 统计里, 我们需要使用概率参数的分布作为先验分布. 对概率密度的一种自然选择是 Dirichlet 分布:

$$\mathscr{D}(\theta|\alpha) = Z^{-1}(\alpha)\prod_{i=1}^{K}\theta_i^{\alpha_i-1}\delta(\sum_{i=1}^{K}\theta_i - 1). \tag{11.5}$$

其中 $\alpha = \alpha_1, \ldots, \alpha_K, \alpha_i > 0$ 是一组确定 Dirichlet 分布的常数, θ_i 满足 $0 \le \theta_i \le 1$ 且和为 1, 即如 Delta 函数项 $\delta(\sum_i \theta_i - 1)$ 所示. 参数 θ_i 的代数表述与多项分布中一致. 我们不在结果出现的次数 n_i 上进行归一化, 而是在 θ_i 的所有可能值上进行归一化. 也就是说, 多项分布是关于指数 n_i 的分布, 而 Dirichlet 分布则是关于底数 θ_i 的分布. 这两种分布互称共轭分布 (conjugate distribution) [Casella & Berger 1990], 二者形式上的密切关系导致它们在许多估计问题中彼此协调影响.

(11.5) 式所定义的 Dirichlet 分布中的归一化因子 Z 可以通过 gamma 函数表示 [Berger 1985]:

$$Z(\alpha) = \int\prod_{i=1}^{K}\theta_i^{\alpha_i-1}\delta(\sum_i \theta_i - 1)d\theta = \frac{\prod_i \Gamma(\alpha_i)}{\Gamma(\sum_i \alpha_i)}. \tag{11.6}$$

Gamma 函数是阶乘函数在实数域的推广. 对于整数 n, $\Gamma(n) = (n-1)!$. 而对任意正实数 x 有

$$\Gamma(x+1) = x\Gamma(x). \tag{11.7}$$

可以证明, Dirichlet 分布的均值等于其归一化后的参数值, 即 θ_i 的均值为 $\alpha_i/\sum_k \alpha_k$. 例如图 11.2 所示的三个分布都有相同的均值(1/8, 1/4, 5/8), 即使右上图的 α 值比左上图的大 10 倍. 注意 α 值越大, 分布就越密集. 并且还应看到, 如左下图所示, 当某些 $\alpha_i < 1$ 时, 对于相应的 θ_i, 分布在 0 处达到峰值.

当变量数为 2 ($K = 2$) 时, Dirichlet 分布退化成大家更为熟悉的 beta 分布, 此时归一化常数是 beta 函数.

例子: 骰子工厂

再考虑第 1 章与第 3 章讨论过的概率论模型, 其中包含概率参数为 $\theta = \theta_1, \ldots, \theta_6$ 的作弊骰子. 从参数为 $\alpha = \alpha_1, \ldots, \alpha_6$ 的 Dirichlet 分布中对概率向量 θ 抽样, 就好像是一个"骰子工厂" 以不同的 θ 生产不同的骰子 [MacKay & Peto 1995].

假设骰子工厂 A 的所有 α_i 均为 10, 而骰子工厂 B 的所有 α_i 均为 2. 那么从平均上看, 两个工厂都生产公平骰子; 两种情况下 θ_i 的均值都为 $\frac{1}{6}$. 但如果我们发现一个作弊骰子, 其参数 $\theta_6 = 0.5, \theta_1 = \ldots = \theta_5 = 0.1$, 那么它更有可能是工厂 B 生产的:

$$\mathscr{D}(\theta|\alpha_A) = \frac{\Gamma(60)}{(\Gamma(10))^6}(0.1)^{5(10-1)}(0.5)^{10-1} = 0.119,$$
$$\mathscr{D}(\theta|\alpha_B) = \frac{\Gamma(12)}{(\Gamma(2))^6}(0.1)^{5(2-1)}(0.5)^{2-1} = 199.6.$$

参数 α 值越大的工厂生产偏向于公平骰子的分布就越密集. $\sum \alpha_i$ 与 Dirichlet 的方差成反比. (不要被值为 199.6 的 Dirichlet 密度吓倒; 请回忆, 连续概率密度在任何一点的值都有可能大于1.)

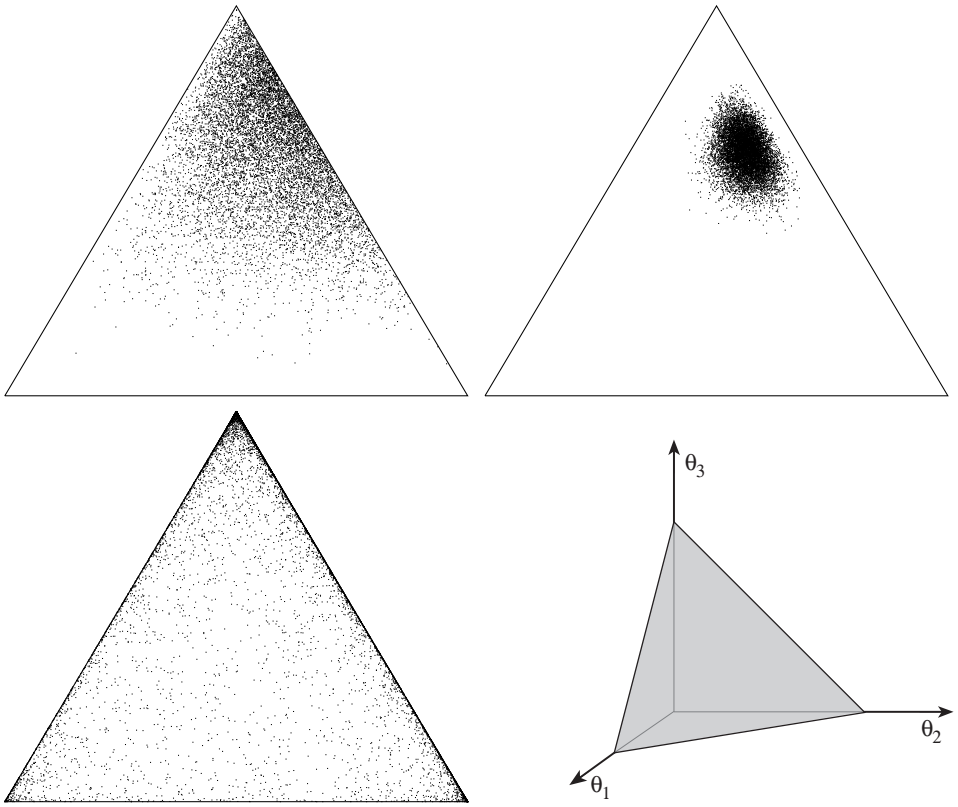


图 11.2: 由 10000 个样本点组成的三维 Dirichlet 分布, 即以概率 $\mathscr{D}(\theta|\alpha)$ 选出点 θ. 左上图使用的 α 值为 (1,2,5), 右上图为 (10,20,50), 左下图为 (0.1,0.2,0.5). 概率 θ 表示为三维空间 $(\theta_1,\theta_2,\theta_3)$ 中满足 $\sum\theta_i = 1$ 的截面; 见右下图. 即将点 $(\theta_1,\theta_2,\theta_3)$ 映射为平面中的点 $((\theta_2-\theta_1)/\sqrt{3},\theta_3)$.

如果一个工厂能够生产出接近完美的公平骰子, 那么它的 α_i 值就都相等且非常大. 如果生产出的骰子不可靠, 但从平均上看又是公平的, 那么其 α_i 值就都相等但较小. □

Gamma 分布

Gamma 分布 $g(x,\alpha,\beta)$ 由下式给出:

$$g(x,\alpha,\beta) = \frac{\mathrm{e}^{-\beta x}x^{\alpha-1}\beta^{\alpha}}{\Gamma(\alpha)},$$

上式对 $0 < x,\alpha,\beta < \infty$ 有定义. 它的均值为 α/β, 方差为 α/β^2. β 仅仅是标度参数.

Gamma 分布与 Poisson 分布互为共轭分布, Poisson 分布 $f(n) = \mathrm{e}^{-p}p^n/n!$ 给出了某时间间隔内看到 n 个事件的概率, 其中 p 表示该时间间隔内发生一个事件的概率. 因为间隔内的事件数量是一个速率, 所以 gamma 分布适于为速率的概率建立模型, 这如同 Dirichlet 分布适用于为发射概率作先验, 而与其共轭的多项分布则被用于为计数赋以概率(第 p. 218 页). Gamma 分布已应用于为 DNA 序列中不同位点处的进化速率进行建模(第 p. 148 页).

极值分布

假设我们按照密度 $g(x)$ 抽取 N 个样本, 那么样本最大值小于 x 的概率就为 $G(x)^N$, 其中 $G(x) = \int_{-\infty}^{x} g(u)du$. N 个样本最大值的密度可由对其取 x 的微分得到, 即 $Ng(x)G(x)^{N-1}$. N 趋近于无穷大时, $Ng(x)G(x)^{N-1}$ 的极限称为 $g(x)$ 的

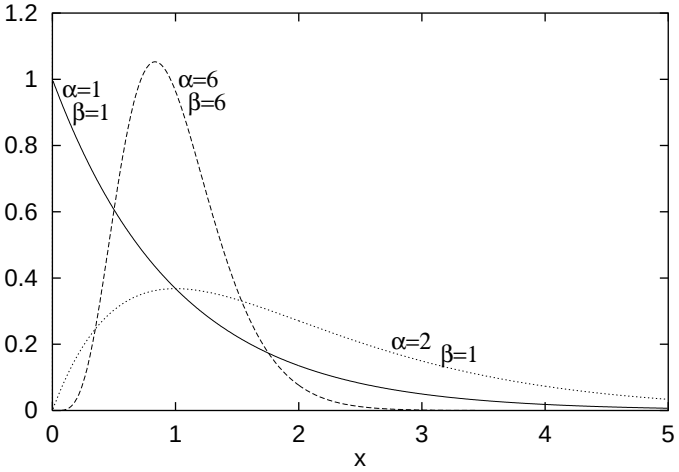


图 11.3: $\alpha = \beta = 1.0$、$\alpha = \beta = 6.0$ 和 $\alpha = 2.0, \beta = 1.0$ 时的 gamma 分布 $g(x, \alpha, \beta)$.

极值密度 (extreme value density , EVD). 它在实际中有广泛的应用, 例如为链的断点(由最弱连接决定) 建模, 以及评估从一组联配得到的最大分值的显著性(见第2章).

当 $g(x)$ 为指数型密度 $g(x) = \alpha \mathrm{e}^{-\alpha x}$ 时, 可以计算 EVD 如下. 积分可得 $G(x) = 1 - \mathrm{e}^{-\alpha x}$. 令 y 使得 $\mathrm{e}^{-\alpha y} = 1/N$, 并记 $z = x - y$, 则有:

$$\begin{aligned} Ng(x)G(x)^{N-1} &= N\alpha \mathrm{e}^{-\alpha x}(1 - \mathrm{e}^{-\alpha x})^{N-1} = \alpha \mathrm{e}^{-\alpha z}(1 - \mathrm{e}^{-\alpha z}/N)^{N-1} \\ &\to \alpha \mathrm{e}^{-\alpha z} \exp(-\mathrm{e}^{-\alpha z}) \quad (N \to \infty), \end{aligned}$$

上式使用了著名的极限 $(1 - X/N)^N \to \mathrm{e}^{-X} \quad (N \to \infty)$. [1]累积概率 (cumulative probability, 极值 $\le x$ 的概率) 是 $\exp(-\mathrm{e}^{-\alpha z})$, 它称为Gumbel 分布 [Gumbel 1958]. 对适中的 N 值, 上述密度通常可以给出极值分布的很好近似. 当使用指数型密度时, 由图11.4可见容量为10的样本最大值能够给出 EVD 的很好近似.

令人吃惊的是, Gumbel 分布是多种原始密度 $g(x)$ 的 EVD; 例如 $g(x)$ 为 Gauss 分布时也成立. 更一般地, EVD 必须形如 $\exp(-f(a_N x + b_N))$, 其中 a_N 与 b_N 为依赖于 N 的常数, 而 $f(x)$ 或为指数形式 e^{-x} 或为 $|x|^{-\lambda}$, 其中 λ 为某一正常数(关于这个定理的精确陈述请见 Waterman [1995]).

[1]以上论述中有细微之处. 我们必须小心, 因为 $\mathrm{e}^{-\alpha z}$ 不会随着 N 的增大而快速增长, 以至于极限 $(1 - \mathrm{e}^{-\alpha z}/N)^N \to \exp(-\mathrm{e}^{-\alpha z})$ 不再成立. 更准确地讲, 必须证明依照分布 $Ng(x)G(x)^{N-1}$, $\mathrm{e}^{-\alpha z}$ 值很大的概率逐渐趋近于 0. ——原注

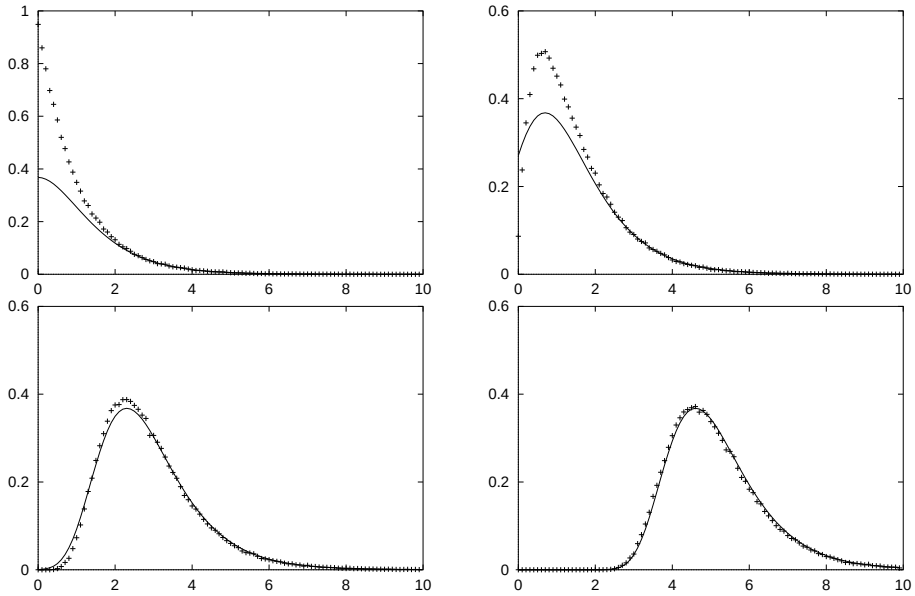


图 11.4: 对极值分布的近似: 先从定义在 $0 \le x < \infty$ 上的分布 e^{-x} 中抽样 N 个点, 然后选出最大值. 按照从左上到右下的顺序, N 分别为 1、2、10 和 100.

11.2 熵

本书中的某些术语借用自信息论(例如见 Cover & Thomas [1991]). 信息论与概率论建模有着密切的联系.

熵 (entropy) 是结果平均不确定度的一种度量. 给定一个随机变量 X, 它在 K 个事件 $x_1, \ldots, x_K$ 的离散集合上有概率 $P(x_i)$, 则 Shannon 熵定义为:

$$H(X) = -\sum_i P(x_i) \log P(x_i). \tag{11.8}$$

在此定义中, 如果 $P(x_i) = 0$, 则令 $P(x_i) \log P(x_i)$ 为 0. 通常我们假设 log 为自然对数 (有时记为 ln). 但我们通常也使用以2为底的对数 (记为 $\log_2$), 这种情况下熵的单位是"比特" (bit).所有对数间都互成比例, 例如 $\log_2(x) = \log_{\mathrm{e}}(x)/\log_{\mathrm{e}}(2)$, 所以从理论上讲使用哪种对数都没有关系. 通常来讲, 我们考虑的总是概率分布 P 的熵 $H(P)$, 而不是随机变量 X 的熵 $H(X)$.

当所有概率 $P(x_i)$ 都相等 ($P(x_i) = 1/K$) 时熵最大, 此时随机样本结果的不确定度就达到最大. 这个最大值为 $-\sum_i \frac{1}{K} \log \frac{1}{K} = \log K$. 如果我们明确地知道分布中某抽样的结果, 即如果对某个 k 有 $P(x_k) = 1$, 而其他的 $P(x_i) = 0$, 那么熵为0.

某些概率论模型生成的序列之期望分值当被定义为对数概率时, 也可以用作熵. 例如我们假设一条序列某一位置处出现残基 a 的概率为 P_a, 于是概率 P_a 的分值是 $\log P_a$, 而期望分值为 $\sum_a P_a \log P_a$, 也就是负熵. 当模型在相互独立的位点集合中定义概率时, 上述结论都成立(见练习11.2).

若事件的结果事先已知, 那么因为已经获得了信息, 所以不确定度由H降为0. 因此, 通常情况下熵与信息等价. 此处容易产生混淆; 某事物越随机(熵越高), 它所包含的信息就越多, 这与我们的直觉正好相反. 但如果我们将信息看成是熵的差异, 就不会混淆了. 更一般地说, **信息量**(information content) 或**信息** (information) 是指收到某种"消息" 之后, 对不确定度减少的一种度量; 因此, 收到消息前 (before) 后 (after), 熵的差异为:

$$I(X) = H_{\mathrm{before}} - H_{\mathrm{after}}. \tag{11.9}$$

不确定度并不总是减小到 0; 例如信道中可能有噪声, 我们可能仍会保留结果的某些不确定性, 在这种情况下 H_{after} 为正数并且信息少于原来的熵.

信息论中往往假设概率分布是完全已知的. 但在许多应用中真正的分布是未知的, 因此熵是从事件的频率而不是真实的分布计算得到; 见下例.

例子: 随机DNA 的熵

如果 DNA 序列中的每种符号 (A、C、G 或 T) 均以等概率出现 ($p_a = 1/4$), 那么每种 DNA 符号的熵就为 $-\sum_a p_a \log_2 p_a = 2$ 比特.

我们可以将熵看成是为发现观测结果我们所要回答的问题个数, 这些问题都只有是或否两种答案. 例如对于随机 DNA, 我们需要回答两个问题: 首先问"是嘌呤还是嘧啶?" 若是嘌呤, 则然后问"是 A 还是 G?" ; 否则然后问"是 C 还是 T?" □

例子: 保守位点的信息量

信息量可以用来衡量 DNA 或蛋白质序列联配中某一位点处的保守程度. 设想我们预料研究的 DNA 序列是随机的 ($p_a = 0.25$; $H_{\text{before}} = 2$ 比特), 但我们观测到几条相关序列的某一特定位点总是 A 或 G, 且其出现概率分别为 $p_A = 0.7$ 和 $p_G = 0.3$. 因此 $H_{\text{after}} = -0.7\log_2 0.7 - 0.3\log_2 0.3 = 0.88$ 比特. 于是, 这个位置的信息量为 $2 - 0.88 = 1.12$ 比特. 位点越保守, 信息量就越大.

但是, 请注意如果观测值分布的熵比期望要高(即更"随机"), 那么信息量可以是负值. 因此为了找到不寻常的模式, 更好的办法是使用下面介绍的相对熵来度量分布间的差异. □

练习

11.2 假设一个模型, $p_i(a)$ 是氨基酸 a 在长为 l 的序列上第 i 个位点出现的概率, 这些氨基酸被认为是独立的. 那么特定序列 $x = x_1, \ldots, x_l$ 的概率 $P(x)$ 为多少? 试证明此概率的对数平均值就是负熵 $\sum P(x)\log P(x)$, 这里是对长度为 l 的所有可能序列 x 求和.

相对熵与互信息

我们回到不同类型的熵定义. 对于两个分布 P 和 Q, 定义**相对熵** (relative entropy, 也称为 Kullback-Leibler "距离") 为:

$$H(P||Q) = \sum_i P(x_i)\log\frac{P(x_i)}{Q(x_i)}. \tag{11.10}$$

如果 Q 为均匀"背景分布" ($Q(x_i) = \frac{1}{K}$), 表示 H_{before} 的完全单纯的起始状态, 那么信息量就与相对熵等价. 这两个名词有时可以相互替换.

相对熵有如下性质: 它总是大于或等于0. 容易证明, $H(P||Q) \geq 0$, 当且仅当对所有 i 有 $P(x_i) = Q(x_i)$ 时等号成立 (见图 11.5). 将相对熵 $H(P||Q)$ 看成是概率分布 P 与 Q 间的距离, 经常很有用. 但是它并不满足对称性, 即$H(P||Q) \neq H(Q||P)$, 而且它也不满足数学上距离度量的正式要求.

相对熵常作为模型的期望分值出现, 该分值定义为**对数几率** (log-odds), 即 $P(\text{data}|M)/P(\text{data}|R)$, 其中 M 为该模型, R 为零(null) 模型. 若 p_a 为模型 M 下残基 a 在一条序列中某一位置的概率, q_a 为模型 R 下残基 a 在一条序列中某一位置的概率, 则残基 a 的分值是 $\log(p_a/q_a)$, 期望分值 $\sum_a p_a \log(p_a/q_a)$ 是相对熵.

另一种重要的熵度量是**互信息** (mutual information). 称两个随机变量 X 和 Y 独立, 如果 $P(X,Y) = P(X)P(Y)$. 我们会对其独立的**程度**感兴趣, 而这可以用分布 $P(X,Y)$和$P(X)P(Y)$ 间的相对熵"距离" 来度量:

$$M(X;Y) = \sum_{i,j} P(x_i,y_i)\log\frac{P(x_i,y_i)}{P(x_i)P(y_i)}, \tag{11.11}$$

其中 $\{x_i\}$和$\{y_i\}$ 为 X 和 Y 的可能取值. $M(X;Y)$ 就是互信息, 它可以解释为已知结果 Y 的前提下可获得的关于 X 的信息量.

当 X 和 Y 总是共变 (或协变, covary) 的时候, 互信息取得最大值. 例如, 如果某两条联配好的 DNA 序列上位置 i 和 j 处, 除 AT、TA、GC 以及 CG 外的所有残基对概率皆为0, 那么这时就出现最大共变 (covariation). 在这种情况下, 我们总能得

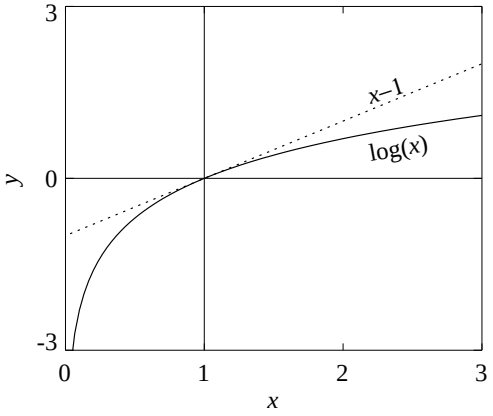


图 11.5: 相对熵 (11.10) 总大于等于零的证明. 如果对所有 i 有 $P(x_i) = Q(x_i)$, 那么相对熵总为零. 从此图可以看出, $\log(x) \leq x - 1$ 且仅当 $x = 1$ 时等式成立. 于是有 $-H(P||Q) = \sum_i P(x_i) \log(Q(x_i)/P(x_i)) \leq \sum_i P(x_i)(Q(x_i)/P(x_i) - 1) = 0$, 且仅当对所有 i 有 $Q(x_i) = P(x_i)$ 时等号成立.

到 $P(x_i, y_j) = P(x_i) = P(y_j)$ 或 $P(x_i, y_j) = 0$, 因此 $M = -\sum_i P(x_i) \log P(x_i)$.这就是 X (或 Y) 的熵, 所以它对均匀分布达到最大, 最大值是 $\log K$ (假设 X 和 Y 有相同的可能结果数 K). 因此 DNA 序列的最大互信息是 $\log_2 4 = 2$ 比特.

图 11.6 给出了一个 RNA 联配中(从频率计算得到的) 每两列间的互信息.

例子: 受体位点 (Acceptor site) 寻找生物序列中的不寻常模式时, 相对熵十分有用. 为了阐明这点, 我们从人类基因数据库中抽出 757 个受体位点. 受体位点就是内含子3'端的剪切位点, 在该处内含子被剪切后形成信使 RNA . 内含子的最后两个碱基几乎总是 AG, 并且在这个数据集中序列都遵循此规律. 我们只取出现在两个密码子间的内含子受体位点, 即剪切不发生在密码子内. 我们取出了剪切位点上游 (upstream) 的 30 个碱基和下游 (downstream) 的 20 个碱基. 图 11.6 显示了从这些序列中任意选取的一个样本.

在序列的每个位置 i 上, 我们可以得到四种核苷酸的频率 $p_i(a)$, 并可以计算相对熵 $\sum_a p_i(a) \log_2[p_i(a)/q_a]$, 其中 q_a 为序列中四种核苷酸的总体分布, 见图11.6. 在一致序列 AG 处, 相对熵非常高(分别等于 $-\log_2(q_A)$ 和 $-\log_2(q_G)$). 在上游序列中存在着一个有趣的现象: 在 AG 的前两位碱基处相对熵达到最小. 在编码区存在着相对熵的一个弱周期信号 (几乎不可见), 这是因为三种读码框间存在不同的碱基构成. 关于剪切位点处信息的更多讨论, 请参阅 Brunak, Engelbrecht & Knudsen [1991] 和 Hebsgaard *et al.* [1996]; 关于显示各种熵度量的更多方法, 请参阅 Schneider & Stephens [1990].

为了考察相邻位置是否独立, 我们可以计算每列间的互信息. 对于相邻两列(记为 i 和 $i + 1$), 可通过计算列 i 中 a 出现的次数以及列 $i + 1$ 中 b 出现的次数来得到残基对出现的频率 $p_i(a, b)$. 由此便可计算互信息 $\sum_{a,b} p_i(a, b) \times \log_2[p_i(a, b)/p_i(a)p_{i+1}(b)]$, 见图 11.6.

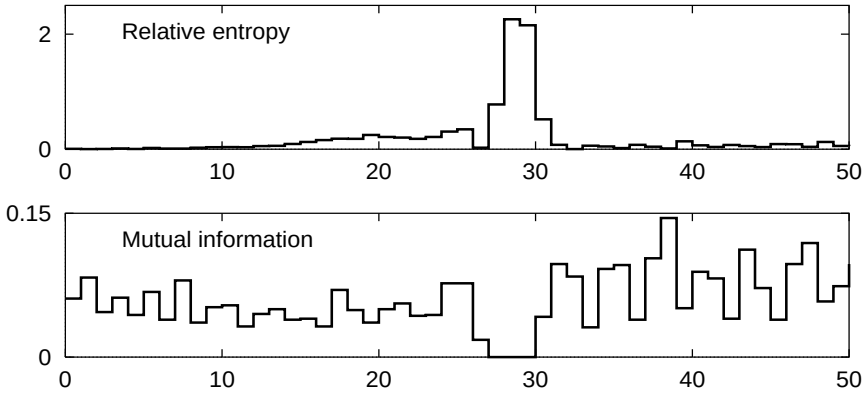
请注意在一致序列 AG 处互信息为 0: 已知该位置为 A 并不能传递关于下一位置的任何信息, 因为下一个碱基总会是 G. 受体位点周围的互信息总是远小于最大值 2 比特, 但却非 0, 这说明相邻位置间存在着相关性. 该现象普遍存在于 DNA 序列中. 在编码区中可以观察到明显的周期模式, 这表明核苷酸在三种读码框中是相关的. □

练习

11.3 证明前述命题: 当 q 为均匀分布时, 信息量等价于相对熵.

11.4 证明 $M(X;Y) = M(Y;X)$.

11.5 证明 $M(X;Y) = H(X)+H(Y)-H(Y,X)$, 其中 $H(Y,X)$ 为联合分布 $P(X,Y)$ 的熵.



Example sequences

```
CTTCTCAAATAACTGTGCCTCTCCCTCCAGATTCTCAACCTAACAACTGA
CTGCTCACCGACGAACGACATTTTCCACAGGAGCCGACCTGCCTACAGAC
GGTTCCCTCTTGGCTTCCATGTCCTGACAGGTGGATGAAGACTACATCCA
ACTAACTCTCCTCCTCGTGTGTCTCCCCAGCCCGTGTCCCAGCCCACCCA
TTGATAACATGACATTTTCCTTTTCTACAGAATGAAACAGTAGAAGTCAT
TCTACCGTCCCTTTCCCACACACTCTGCAGAAGGTGGTGTTGTCTTCTGG
CTTTTTTCTCTCCTATGTGCATCCCCCCAGGAGCTGGCTGAATATGAATA
GCTAATAGCTTGCTTATTTATTTAACATAGGGCTTCCGTTACAAGATGAG
AATTTAGTTTTATTCCCATGTGACCTGCAGGTAAATGAAGAAGGCAGTGA
ACTCTGCTCACTGTCACTTTGCTCCCACAGCGTCCGCTCTGCAATGGCAG
ACCTCCTAACGTTGTTGGGTTTCTTTGCAGAACTTTGCTGCCCAGATGGC
GTAAACCCCTCATTTTCTGTTCCGATGCAGGGCCCCATGGGACCTCGAGG
AGAAGTGACATTTTTCCTATATGTTGACAGGGTGGTGACTTCACACGCCA
CTGGTGTGAGGACCTGCCTCTCTTTTCAAGGGTGAACCTGGTATTGCTGG
ACCTTGGGCACTGTGTTCCTTTGTTTCTAGCACTGGCAGATCCCCCTGAG
TTTTGTTATGCAATTATTGTTTTCTTACAGGGCCCTCTACTAAAGAAGGA
GCATCACCTGTCAGCTCCCTGTGTCCACAGGCTCTGCAGCGGCTCAGGGA
```

图 11.6: 受体位点的相对熵和互信息图. 下面是这些序列的一个样本. 注意在保守的 AG 处, 相对熵隆起一个峰, 而互信息则出现一个凹陷.

11.3 推断

概率论模型是本书关注的重点. 模型可以呈现很多形式: 从简单的分布到带许多概率分布的复杂随机文法. 一旦选定模型种类, 我们需要从数据中**推断** (infer) 该模型的参数. 例如, 我们可以用多项分布对掷骰子的结果进行建模. 假设产生 i 的观测的数目是 $n_i (i = 1, \ldots, 6)$. 我们并不知道骰子是否公平, 因此我们需要估计多项分布的参数, 即投掷一次骰子后得到 i 的概率 θ_i. 下面, 我们将考虑通常能够用于推断的不同策略. 更多的背景介绍请参阅 Ripley [1996] 及 MacKay [1992].

最大似然

假设我们想要从一组数据 D 中为模型 M 估计参数 $\theta = \{\theta_i\}$. 最直观的策略就是对所有可能的 θ, 最大化 $P(D|\theta, M)$. 这称为**最大似然** (maximum likelihood , ML) 判据. 形式上我们记为:

$$\theta^{\mathrm{ML}} = \operatorname*{argmax}_{\theta} P(D|\theta, M). \tag{11.12}$$

一般来说, 当 $P(x|y)$ 被看作是 x 的函数时表示概率; 而被看作是 y 的函数时称为似然. 注意, 似然并不是概率分布或概率密度, 而仅仅是关于变量 y 的一个函数.

最大似然具有许多优良性质. 例如它在下述意义下是一致的, 即用于生成数据集的参数值 θ_0 在大量数据的极限下也是使似然最大化之值. 为了说明这点, 假设模

型 M 有 K 个可观测结果 $\omega_1, \ldots, \omega_K$ (例如, 在一组已联配序列中的某一位点处, 存在 4^n 种可能的核苷酸分配形式). 随着数据量的增加, ω_i 出现的频率 $n_i/\sum n_k$ 将趋近于 $P(\omega_i|\theta_0, M)$ (见练习 11.6). 因此参数 θ 的对数似然 $\sum_i (n_i/\sum n_k) \log P(\omega_i|\theta, M)$ 将趋近于 $\sum_i P(\omega_i|\theta_0, M) \log P(\omega_i|\theta, M)$. 由相对熵的正定性可知, 对所有 θ 有 $\sum_i P(\omega_i|\theta_0, M) \log P(\omega_i|\theta_0, M) \geq \sum_i P(\omega_i|\theta_0, M) \log P(\omega_i|\theta, M)$. 因此似然在 θ_0 处取最大值.

最大似然的一个缺点是: 当数据量不大时, 它会给出较差的结果; 此时依赖于更多的先验知识就更为明智. 再次考虑掷骰子的例子, 我们想要从三次不同的投掷结果中估计出多项分布的参数. 在本书第 p. 218 页我们指出 θ_i 的最大似然估计为 $n_i/\sum n_k$, 即至少有三个参数被估计为0. 显然, 对于大多数骰子来说这是很差的估计, 此时我们想要整合先验知识以期望所有参数都接近于 1/6.

练习

11.6 **弱大数定理** (weak law of large numbers) 指出, 容量为 N 的样本的均值与真实均值之差大于等于 d 的概率为 $\sigma^2/(Nd^2)$, 其中 σ^2 为分布的方差. 试证明由此可得, 当 $\sum n_k \to \infty$ 时, $n_i/\sum n_k$ 趋近于 $P(\omega_i)$, 其中 n_i 为 ω_i 的出现频率.

后验概率分布

我们可以利用 Bayes 定理引入先验知识. 假设参数 θ 服从某种概率分布. 通过以 M 为条件, 可以给出如下形式的 Bayes 定理:

$$P(\theta|D, M) = \frac{P(D|\theta, M)P(\theta|M)}{P(D|M)}. \tag{11.13}$$

先验 $P(\theta|M)$ 必须以某种合理的形式选出, 这就是 Bayes 估计法的精妙之处. 对先验的自由选取使得 Bayes 统计不时引起人们的争议, 但我们相信该方法能够提供十分方便的框架, 以便将先验(生物学) 知识整合到统计学估计法中.

给定数据及模型后, $P(\theta|D, M)$ 是参数的后验概率. 可以采取多种方法将后验用于推断. 我们可以从中抽样(见11.4节), 进而找到模型参数的高概率区域. 在8.4节中, 我们介绍了该方法在系统发育的概率论模型中的应用. 如果想为模型得到一组特定的参数值, 我们可以仿照 ML 并选择**最大后验**概率 (maximum a posteriori probability , MAP) 估计,

$$\theta^{\mathrm{MAP}} = \underset{\theta}{\operatorname{argmax}}\, P(D|\theta, M)P(\theta|M). \tag{11.14}$$

注意, 我们略去了数据先验 $P(D|M)$, 这是因为它不依赖于参数 θ, 所以最大值点 θ^{MAP} 与它无关. 另一种方法是使用**后验均值**估计 (posterior mean estimator , PME), 它对经后验加权的所有参数集取平均:

$$\theta^{\mathrm{PME}} = \int \theta P(\theta|n) d\theta. \tag{11.15}$$

即对所有有效的概率向量, 即所有和为1的向量进行积分. 下面, 我们将为带某一先验的多项分布推导 PME.

MAP 和 PME 估计量的正确性有些值得怀疑, 这是因为参数的非线性变换往往会改变结果. 从技术上讲, 它们并不是**等变化的**(equivariant) [Ripley 1996]. 要探究其中的原理, 我们需要考虑在密度上变量替换所造成的影响.

变量替换

给定密度 $f(x)$, 假设有一个变量替换 $x = \phi(y)$. 于是我们可以由 $g(y) = f(\phi(y))|\phi'(y)|$ 定义密度 $g(y)$. 式中含有 ϕ 的导数 $\phi'(y)$ 是因为在变换 ϕ 之下, 区间 δx 对应于区间 $\delta y \phi'(y)$, 于是密度 f 中在 ϕ 下扫过的量正比于这个导数; 对导数取绝对值确保密度非负†. 该定义产生一个经过正确归一化的密度, 因为 $\int g(y)dy =$

†原文为正 (positive), 中文版修改为"非负" (non-negative) 更为严格. ——译注

$\int f(\phi(y))|\phi'(y)|dy = \int f(x)dx = 1$, 其中 f 是一个密度. 形式上我们记这个变换规则为:

$$g(y) = f(\phi(y))|\phi'(y)|. \tag{11.16}$$

显然, 函数 $f(\phi(y))$ 与 $f(x)$ 有相同的最大值. 然而当乘以 $|\phi'(y)|$ 之后, 最大值可能改变(见练习11.7). 现在后验 $P(\theta|D,M)$ 是一个密度, 因此用 MAP 选出的峰值可以在变换下作相同的改变. 类似的论证显示, PME 也可以在坐标变换之后发生改变.

相比之下, 似然 $P(D|\theta,M)$ 不会像密度一样发生变换; 它仅仅是关于 θ 的一个函数且坐标变换并不改变峰的位置, 这正如 $f(\phi(y))$ 与 $f(x)$ 有相同的峰一样 [Edwards 1992].

练习

11.7 令 $f(x) = 2(1-x)$ 为[0,1]上的密度. 试求在 $x = y^2$ 下它如何变换到 y 的密度. 证明该变换下密度的峰与 PME 都发生改变.

11.4 抽样

给定定义在有限集 X 里元素 x_i 上的概率 $P(x_i)$, 从该集合中抽样 (sample), 即以概率 $P(x_i)$ 随机地抽取元素 x_i.

抽样的基本实用工具是一种函数, 它源于计算机中的伪随机数生成器 (pseudo-random number generator), 即 rand [] 或类似名字的函数), 该生成器按均匀密度在区间 [0,1] 内随机地选数. 记此函数为 rand [0,1]. 我们可以用该函数以频率 $P(x_i)$ 选取元素 x_i. 我们令 $y = \text{rand}[0,1]$, 然后寻找满足 $P(x_1)+\ldots+P(x_{i-1}) < \text{rand}[0,1] < P(x_1)+\ldots+P(x_{i-1})+P(x_i)$ 的 i 以便选取我们所要的元素 x_i. 显然, rand [] 落在该区间的概率为 $P(x_i)$, 由此可见 x_i 是以正确的概率被选取.

实际上, 用计算机产生随机数并不简单. 伪随机数的标准函数往往非常原始, 而且在某些应用中并不完善. 例如许多 UNIX 计算机的标准 rand [] 函数返回 0 到 $2^{15}-1$ 间的一个整数, 我们期待通过对这个函数的返回值取模 2 (modulo 2) 得到"随机" 的二进制位(0或1). 然而这便给出一串 0 和 1 交替出现的序列, 很显然并不随机. 在大多数系统中也有其他(更好的) 随机数函数. 例如 Press *et al.* [1992] 就给出了关于随机数生成器的讨论.

通过变换均匀分布实现抽样

抽样的概念同样适用于密度: 从给定密度 f 中抽样, 就是从定义了 f 的空间中选取元素 x, 使得在点 x 的任意小邻域 δR 内选取一个点的概率为 $f(x)\delta R$. 对密度的抽样可以这样完成: 先从定义在 [0,1] 上的均匀分布中抽样得到伪随机数, 然后应用可适当改变密度的变量替换.

上述想法的理论描述如下: 假设我们给定密度 $f(x)$ 以及映射 $x = \phi(y)$, 由式 (11.16) 可知, $g(y) = f(\phi(y))\phi'(y)$. 如果 f 是均匀的, 则有 $g(y) = \phi'(y)$, 于是 ϕ 可由积分得到, $\phi(y) = \int_b^y g(u)du$, 其中 b 为某个适当的下界. 但是, 我们想要用一个好的伪随机数生成器从 x 中选取点, 再将它们映射到 y. 为此我们需要 ϕ 的反函数 $y = \phi^{-1}(x)$.

例如假设我们想要从 Gauss 分布中抽样. 我们定义累积 Gauss 映射为 $\phi(y) = \int_{-\infty}^y e^{-u^2/2}/\sqrt{2\pi}du$, 并令 $y = \phi^{-1}(x)$. 我们可以生成一个累积高斯函数的反函数查找表, 但这显得很笨拙, 用某些其他方法可能更方便(例如练习 11.10).

我们也可以更一般地将变换方法应用于包含 K 个变量的函数中, 此时 (11.16) 式变为:

$$g(y_1,\ldots,y_K) = f(\phi_1(y_1,\ldots,y_K),\ldots,\phi_K(y_1,\ldots,y_K))|J(\phi)|. \tag{11.17}$$

其中 $J(\phi)$ 为 Jacobi 行列式 (Jacobian), 它的 (i,j) 元为 $\partial\phi_i/\partial y_j$ [Feller 1971].

练习

11.8 证明函数 $g(y) = \alpha^\lambda\lambda y^{\lambda-1}/(\alpha^\lambda + y^\lambda)^2$ 为定义在 $0 \le y < \infty$ 上的密度. 证明从 (0,1) 中均匀地选出 x, 并将 x 映射到 $y = \alpha(\frac{x}{1-x})^{1/\lambda}$ 能够给出对 $g(y)$ 的抽样.

11.9 定义 ϕ 为从变量 (u,w) 到 (x,y) 的映射, 其中 $x=uw, y=(1-u)w$. 证明 $J(\phi)=w$, 其中 J 为 Jacobi 行列式.

11.10 (需要用到微积分!) 假设我们在区间 $[0,1]$ 中选取两个随机数 x 和 y, 然后将 (x,y) 映射到样本点 $\cos(2\pi x)\sqrt{\log(1/y^2)}$. 试证这样可以得到来自 Gauss 分布的正确抽样. 该方法称为 Box-Muller 方法 [Press *et al.* 1992].

通过拒绝从 Dirichlet 中抽样

现在考虑从 Dirichlet 分布中抽样的问题, 由此我们可以说明许多重要原理. 首先假设可以从 gamma 分布 $g(x,\alpha,1)$ 中抽样:

$$g(x,\alpha,1)=e^{-x}x^{\alpha-1}/\Gamma(\alpha)$$

其中 $0<x<\infty$ (见本书第 p. 208 页). 假设我们从参数分别为 α_1 和 α_2 的两个 Gamma 分布中抽样得到 x_1 和 x_2, 则可以定义满足 $u+v=1$ 的变量对 (u,v), 其中令 $u=x_1/(x_1+x_2)$, $v=x_2/(x_1+x_2)$; 相应地我们可以设 $x_1=uw, x_2=(1-u)w$, 然后对 w 求积分. 由式 (11.17) 及练习 11.9 的结果, 我们可以得到变量对 (u,v) 的分布 $D(u,v)$:

$$\begin{aligned}
D(u,v) &= \frac{\int_0^\infty \delta(u+v-1)\mathrm{e}^{-uw}(uw)^{\alpha_1-1}\mathrm{e}^{-vw}(vw)^{\alpha_2-1}w\,dw}{\Gamma(\alpha_1)\Gamma(\alpha_2)} \\
&= \frac{u^{\alpha_1-1}v^{\alpha_2-1}\delta(u+v-1)}{\Gamma(\alpha_1)\Gamma(\alpha_2)}\int_0^\infty \mathrm{e}^{-w}w^{\alpha_1+\alpha_2-1}dw \\
&= u^{\alpha_1-1}v^{\alpha_2-1}\delta(u+v+1)\frac{\Gamma(\alpha_1+\alpha_2)}{\Gamma(\alpha_1)\Gamma(\alpha_2)} \\
&= \mathscr{D}(u,v|\alpha_1,\alpha_2), \qquad (11.18)
\end{aligned}$$

其中 $\mathscr{D}(u,v|\alpha_1,\alpha_2)$ 是参数为 α_1 和 α_2 的 Dirichlet 分布. 也就是说, 要从二变量 Dirichlet 分布 (beta 分布) 中抽样, 我们可以先从两个 gamma 分布中抽样, 其指数分别与该 Dirichlet 分布的两个组分的指数相同, 然后归一化样本给出概率. 这个美妙结论可以推广到含任意变量数目的 Dirichlet 分布 (练习11.11).

因此如果知道如何从 gamma 分布中抽样, 那么我们就可以从 Dirichlet 分布中抽样. 现在可以证明 $g(x,\alpha,1)\le f(x)$ (练习11.12), 其中

$$f(x)=\frac{4e^{-\alpha}\alpha^{\lambda+\alpha}x^{\lambda-1}}{\Gamma(\alpha)(\alpha^\lambda+x^\lambda)^2}, \qquad (11.19)$$

并且 $\lambda=\sqrt{2\alpha-1}$. 于是如果 rand [0,1] 确实能在 0 和 1 之间均匀地抽样, 则 $P(\text{rand}[0,1]<g(x,\alpha,1)/f(x))=g(x,\alpha,1)/f(x)$. 因此如果我们先从分布 f 中抽样, 按概率 $f(x)$ 选取点 x, 然后如果 $\text{rand}[0,1]<g(x,\alpha,1)/f(x)$ 则接受 x, 那么就有:

$$P(x)=f(x)P(\text{rand}[0,1]<g(x,\alpha,1)/f(x))=g(x,\alpha,1).$$

于是这个两阶段过程可以让我们从 gamma 分布中抽样. 剩下的问题就是怎样从 f 中抽样. 然而练习 11.8 已经证明, 先使用 rand [0,1] 从 [0,1] 中选择 u 然后定义 $x=\alpha(u/1-u)^{1/\lambda}$, 等价于从 f 中抽样. 关于这一节内容的更多细节, 以及当 $0<\alpha<1$ 时的合适过程, 请参阅 Law & Kelton [1991]. 图11.2就是利用这种方法生成的.

这是一种**拒绝抽样** (rejection sampling) 过程:分布 g 通过"截取" (trimming down) 分布 f 得到, 而该 f 是解析可解的 (analytically tractable) 而且总大于 g. 这种方法只有当 $f(x)$ 是 $g(x,\alpha,1)$ 的很好近似时才有较好的效果; 否则, 拒绝率会很高. 在两个函数都很大, 即从它们最频繁抽样的范围内抽样时, 函数 $f(x)$ 给出 $g(x,\alpha,1)$ 的一个很好近似. 对这种目的而言, λ 的选择实际上是最优的. 例如当 $\alpha=5$ 并且 $\lambda=\sqrt{2\alpha-1}=3$ 时, 仅有14%的点被拒绝(图11.7的左图), 而当 $\lambda=1$ (图11.7的右图) 时, 有65%被拒绝.

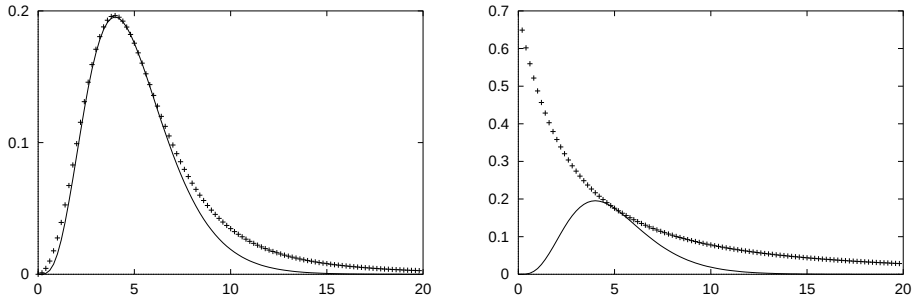


图 11.7: 拒绝抽样: 我们想要从 gamma 分布 $g(x,\alpha,1)$ (实线) 中抽样. 可以从 (11.19) 式给出的函数 f (加号线) 中抽样, 此函数的值总是比 gamma 函数的大. 从 f 中选取一个点 x 后, 我们以某概率接受该点, 此概率值等于该点处 gamma 分布和 f 分布的比值, 即 $g(x,\alpha,1)/f(x)$. 左图中 $\alpha=5,\lambda=3$, 右图中 $\alpha=5,\lambda=1$.

练习

11.11 证明式(11.18) 可以推广到 K 个 gamma 分布的情况, 即从 $g(x,\alpha,1), i = 1,\ldots,K$ 中抽样, 然后求平均, 等价于从Dirichlet 分布 $D(\theta_1,\ldots,\theta_k|\alpha_1,\ldots,\alpha_K)$ 中抽样. (提示: 证明映射 $x_i = u_i w, i \le K-1$ 及 $x_K = (1-\sum u_i)w$ 的 Jacobi 行列式等于 w^{K-1}.)

11.12 证明若 $\alpha > 1$ 且 $1 \le \lambda \le \sqrt{2\alpha-1}$, 则对所有 x 都有 $g(x,\alpha,1) \le f(x)$, 其中 $f(x)$ 由(11.19) 定义. 如果 $\lambda > \sqrt{2\alpha-1}$, 那么又是什么情况?

基于 Metropolis 算法的抽样

通常我们想要从概率论模型中抽样, 此时作为变换方法和拒绝抽样基础的解析方法就都无效. 于是一种可行的方法是使用定义在结果空间 X 上的 Markov 链 [Neal 1996]. 此处我们假设 X 是有限的, 虽然该方法也可以推广到连续的变量与密度.

给定点 x, 一条链 (chain) 指定了到点 y 的转移 $x \to y$ 的概率 $\tau(y|x)$. 如果我们可以从分布 $\tau(y|x)$ 中抽样, 即给定 x 可以按概率 $\tau(y|x)$ 选取 y, 则我们可以生成一条序列 $\{y_i\}$, 其中每个 y_i 可从分布 $\tau(y|y_i-1)$ 中抽样得到.

假设我们现在可以找到某个 τ, 满足:

$$P(x)\tau(y|x) = P(y)\tau(x|y). \tag{11.20}$$

上式称为**细节平衡**(detailed balance) 条件. 可以证明, 细节平衡之下, 对所有点 x 有:

$$\frac{1}{N}\lim_{N\to\infty} C(y_i = x) = P(x), \tag{11.21}$$

其中 $C(y_i = x)$ 是长为 N 的序列中 $y_i = x$ 出现的次数. 因此通过使用 τ 从足够长的序列 $\{y_i\}$ 中抽样, 我们就可以尽可能地近似 P. 上述论断需要添加限制条件: 很显然, 该链必须能够从任何其他点 x 出发达到每一点 y; 换句话说, 对任意 x 和 y, 必须存在一条转移的序列可以从 x 到达 y.

如果我们有一个转移过程 τ 满足 (11.20), 则其生成的序列可以正确地对 P 进行抽样. 但问题是, 我们能否找到这样的过程? Metropolis 算法可以完成这个任务, 它包括两个部分:

(1) 对称的**建议** (proposal) 机制. 给定点 x, 它以概率 $F(y|x)$ 选择一点 y. **对称性** (symmetry) 意味着 $F(y|x) = F(x|y)$.

(2) **接受** (acceptance) 机制. 它以概率 $\min(1, P(y)/P(x))$ 接受建议的 y, 换句话说, 与当前 x 相比具有更大后验概率的 y 总被接受, 而有更小概率的点则以概率 $P(y)/P(x)$ 被随机地接受.

为了证明该方法满足 (11.20), 请注意对 $x \neq y$ 有

$$\begin{aligned} P(x)\tau(y|x) &= P(x)F(y|x)\min(1, P(y)/P(x)) \\ &= F(y|x)\min(P(x), P(y)) \\ &= F(x|y)\min(P(y), P(x)) \\ &= P(y)\tau(x|y). \end{aligned}$$

此处, 我们利用了建议机制的对称性将第二行的 $F(y|x)$ 换成第三行的 $F(x|y)$.

Gibbs 抽样

当概率论模型带有多个变量时, 通常可以从这样的分布中抽样: 该分布由令一个变量变化并保持所有其他变量固定而得到, 即条件分布. Gibbs 抽样所采用的正是这种思想, 它对每个 i 从条件分布 $P(x_i|x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_N)$ 中抽样, 然后令 $i = 1, \ldots, N$ 进行循环.

为了证明这可以从 P 中正确地抽样, 我们只须证明满足细节平衡条件. 即:

$$\begin{aligned} &P(x_1, \ldots, x_n)P(\tilde{x}_i|x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) \\ &\quad = P(x_1, \ldots, x_{i-1}, \tilde{x}_i, x_{i+1}, \ldots, x_n)P(x_i|x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n). \end{aligned}$$

上式可以改写为:

$$\begin{aligned} &P(x_1, \ldots, x_n)P(x_1, \ldots, x_{i-1}, \tilde{x}_i, x_{i+1}, \ldots, x_n)/P(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) \\ &= P(x_1, \ldots, x_{i-1}, \tilde{x}_i, x_{i+1}, \ldots, x_n)P(x_1, \ldots, x_n)/P(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n), \end{aligned}$$

上面的等式显然成立. 假设该过程不会受困于参数空间的某个子集, 即假设它是各态历经 (ergodic) 的, 那么 Gibbs 抽样必定收敛于 P.

Gibbs 抽样可能受困的情况是, 存在两片沿着任意坐标方向都不重叠的密度, 例如在二维的情况下, 有一半密度位于 $[0, 1] \times [0, 1]$ 区域而另一半位于 $[2, 3] \times [2, 3]$ 区域. 注意如果存在小部分的重叠, 例如一半密度在 $[0, 1] \times [0, 1]$ 上为均匀分布而另一半在 $[0.99, 1.99] \times [0.99, 1.99]$ 上为均匀分布, 那么抽样将可以在这两个区域间穿梭, 即使这两区域间的转移并很不频繁.

练习

11.13 问上一个例子中, 在发生区域交叉 (cross-over) 前, 单一区域内样本数的期望是多少?

11.5 从计数估计概率

此前我们使用了掷骰子的例子. 我们需要从数据, 即掷骰子的结果中估计多项分布的参数. 在序列分析中经常会遇到抽象意义上同样的情况, 但是这时结果 i 的掷出次数 n_i 有不同的含义. 例如, n_i 可能表示多序列联配的某一列中氨基酸 i 出现的次数.

假设观测可以表述为结果 i ($i = 1, \ldots, K$) 的计数 n_i, 我们想要为其背后的多项分布估计概率 θ_i. 如果我们有足够多的数据, 那么自然可以使用观测频率 $\theta_i = n_i/N$ 作为概率的估计, 其中 $N = \sum_i n_i$. 这就是最大似然的解 θ_i^{ML}, 证明如下.

对任意 $\theta \neq \theta^{\mathrm{ML}}$, 我们想证明 $P(n|\theta^{\mathrm{ML}}) > P(n|\theta)$. 这等价于证明 $\log[P(n|\theta^{\mathrm{ML}})/P(n|\theta)] > 0$, 如果这里我们只考虑产生非零概率的概率参数. 利用方程 (11.3) 以及 θ^{ML} 的定义, 就有:

$$\begin{aligned} \log \frac{P(n|\theta^{\mathrm{ML}})}{P(n|\theta)} &= \log \frac{\prod_i (\theta_i^{\mathrm{ML}})^{n_i}}{\prod_i \theta_i^{n_i}} \\ &= \sum_i n_i \log \frac{\theta_i^{\mathrm{ML}}}{\theta_i} \\ &= N \sum_i \theta_i^{\mathrm{ML}} \log \frac{\theta_i^{\mathrm{ML}}}{\theta_i} > 0. \end{aligned}$$

最后一个不等式依据如下事实: 相对熵 (11.10) 恒正, 除非两个分布等同. 证毕.

如果数据稀疏, 那么最优估计就不是很清楚. 例如假设我们总共仅有两个计数而且它们都属于同一种残基, 则最大似然估计将给所有其他残基赋予 0 概率. 在这种情况下, 我们将赋予其他残基一定的概率值, 而不完全依赖于如此少的观测值. 因为不存在更多的观测值, 所以这些概率必须从**先验知识** (prior knowledge) 中确定. 这可以通过 Bayes 统计的方法解决, 我们现在将推导 θ 的后验均值估计

我们选择带参数 α 的 Dirichlet 分布 (11.5) 作为先验, 然后为观测 n 的多项分布计算后验 (11.13):

$$P(\theta|n) = \frac{P(n|\theta)\mathscr{D}(\theta|\alpha)}{P(n)}.$$

为了记号方便, 与式 (11.13) 相比, 我们去掉了作为条件的模型 M, 并且考虑了以模型作为隐含条件的所有概率. 我们将多项分布 (11.3) 代入 $P(n|\theta)$, 并将 (11.5) 代入 $\mathscr{D}(\theta|\alpha)$ 可得:

$$P(\theta|n) = \frac{1}{P(n)Z(\alpha)M(n)} \prod_i \theta_i^{n_i+\alpha_i-1} = \frac{Z(n+\alpha)}{P(n)Z(\alpha)M(n)} \mathscr{D}(\theta|n+\alpha).$$

其中最后一步里的 $\prod_i \theta_i^{n_i+\alpha_i-1}$ 可以认为正比于带参数 $n+\alpha$ 的 Dirichlet 分布. 这里 $n+\alpha$ 即为参数 $\{n_i+\alpha_i\}$ 的集合(向量和). 幸运的是, 我们不需要牵涉 gamma 函数就可以完成计算, 因为我们知道 $P(\theta|n)$ 与 $\mathscr{D}(\theta|n+\alpha)$ 都是 θ 的经过适当归一化的概率分布. 这意味着所有的前因子 (prefactor) 必定都可以消去, 并且有:

$$P(\theta|n) = \mathscr{D}(\theta|n+\alpha). \tag{11.22}$$

我们可以看到, 后验本身就是一个类似于先验的 Dirichlet 分布, 当然它们所带的参数是不同的. 可以观察到上式的前因子是 1, 同时这也给出了一个对以后推导有用的推论:

$$P(n) = \frac{Z(n+\alpha)}{Z(\alpha)M(n)}. \tag{11.23}$$

现在, 我们仅需进行积分就可以找到后验均值估计. 由定义 11.15 得:

$$\theta_i^{\mathrm{PME}} = \int \theta_i \mathscr{D}(\theta|n+\alpha)d\theta = Z^{-1}(n+\alpha) \int \theta_i \prod_k \theta_k^{n_k+\alpha_k-1} d\theta. \tag{11.24}$$

我们把 θ_i 放入连乘式, 这样第 i 项就变为 $\theta_i^{n_i+\alpha_i}$. 那么我们可以看到该积分就是 (11.6). 因此我们可以记:

$$\begin{aligned} \theta_i^{\mathrm{PME}} &= \frac{Z(n+\alpha+\delta_i)}{Z(n+\alpha)} \\ &= \frac{n_i+\alpha_i}{N+A}, \end{aligned} \tag{11.25}$$

其中 $A = \sum_i \alpha_i$, 而 δ_i 是第 i 个分量为1其余分量为 0 的向量. 在此, 我们使用了 gamma 函数的性质 (11.7), 即 $\Gamma(x+1) = x\Gamma(x)$; 这就使得我们可以消去除分子中 $n_i+\alpha_i$ 以及分母中 $N+A$ 外的所有项.

应该将这个结果与 ML 估计 θ^{ML} 进行比较. 如果我们将 α 看成是真实值以外的观测值, 那么它恰好就是一个 ML 估计. 这些 α 就像是添加到真实计数中的**伪计数** (pseudocount) . 这使得 Dirichlet 正规化子 (regulariser) 变得十分直观, 并且在某种意义上我们可以舍弃所有的 Bayes 统计而依照伪计数进行思考. 如何使用这些伪计数是很显而易见的: 如果预先知道某个残基, 例如第 i 个很常见, 那么我们就应该给它分配一个较大的伪计数 α_i, 而如果残基 j 通常很罕见, 那么就应给它分配一个较小的伪计数.

值得注意的是, 伪计数的正规化子具有重要的自我调节 (self-regulating) 性质. 如果观测值很多, 即 n 的数目远远大于 α 的数目, 那么这时的估计从本质上讲就等

同于 ML 估计. 另一方面如果观测值非常少, 正规化子将发挥主要作用, 而它给出的估计值也将接近于归一化后的 α 值, 即 $\theta_i \simeq \alpha_i/A$. 因此一般来讲, 我们选择的 α 会等于归一化后的残基总体分布.

混合 Dirichlet 分布

在单一 Dirichlet 分布中表示蛋白质的所有先验知识并不容易; 因此我们自然地想到使用多个 Dirichlet 分布以达到这个目的. 例如我们可能会用一个 Dirichlet 分布描述外露的氨基酸, 用另一个描述内含的氨基酸等等. 用统计学术语来说, 我们可以将其描述为一个**混合** (mixture) 分布. 假设我们有用参数向量 $\alpha^1, \ldots, \alpha^m$ 描述其特征的 m 个 Dirichlet 分布. 混合先验表达如下观念: 任意概率向量 θ 都以概率 q_k 属于混合的某个组分 $\mathscr{D}(\theta|\alpha^k)$. 其形式如下:

$$P(\theta|\alpha^1, \ldots, \alpha^m) = \sum_k q_k \mathscr{D}(\theta|\alpha^k), \tag{11.26}$$

其中 q_k 称为混合系数 (mixture coefficient). 混合系数必须为正且总和为1, 以保证混合分布是适当的概率分布. (这样, 混合分布就可以由任意形式的分布以这种方式构成.) 在前一节中这个概率被称为 $P(\theta)$, 然而这里我们将 α 作为条件加入, 在前面并未明确指出. 这样做将更为方便, 因为下面我们就能使用类似于 $P(\alpha^1|n)$ 的概率. 于是, 我们还可以将 q_k 认定为每个混合系数的**先验**概率 $q_k = P(\alpha^k)$.

对给定的混合, 即对固定的参数 α 以及混合系数, 我们可以使用前一节的结果直接计算后验概率. 由条件概率的定义有:

$$\begin{aligned} P(\theta|n) &= \sum_k P(\theta|\alpha^k, n) P(\alpha^k|n) \\ &= \sum_k P(\alpha^k|n) \mathscr{D}(\theta|n + \alpha^k), \end{aligned}$$

其中我们使用了后验的表达式 (11.22). 为了计算 $P(\alpha^k|n)$, 请注意根据 Bayes 定理并使用 $q_k = P(\alpha^k)$ 有:

$$P(\alpha^k|n) = \frac{q_k P(n|\alpha^k)}{\sum_l q_l P(n|\alpha^l)}.$$

概率 $P(n|\alpha^k)$ 由式 (11.23) 给出 (请回忆前一节中的 $P(n)$ 意味着以 Dirichlet 参数为条件, 因此它就是 $P(n|\alpha^k)$), 于是我们得到:

$$P(\alpha^k|n) = \frac{q_k Z(n + \alpha^k)/Z(\alpha^k)}{\sum_l q_l Z(n + \alpha^l)/Z(\alpha^l)}. \tag{11.27}$$

利用上节的结果 (11.24) 和 (11.25) 我们可以算出 θ_i^{PME} 的最终积分, 于是有:

$$\theta_i^{\mathrm{PME}} = \sum_k P(\alpha^k|n) \frac{n_i + \alpha_i^k}{N + A}. \tag{11.28}$$

使用混合 Dirichlet 的估计类似于使用单一分布的估计: 我们只是基于混合分布的组分对估计进行平均. 但是, 在混合中进行平均时所使用的权重 (11.27) 却是全新的. 虽然在直观上理解这种权重有些困难, 但是它确保了为样本中的高概率混合组分分配较大的权重.

估计先验

关于前一节观点的更详细描述, 请见 Brown *et al.* [1993]以及 Sjölander *et al.* [1996]. 他们利用 Dirichlet 分布混合对列计数的分布进行建模. 他们从一个大数据集, 即大的计数向量集中, 通过估计混合组分与混合系数得到先验.

估计过程如下: 对数据集中每个计数向量 $n^1, \ldots, n^M$, 混合定义了概率

$$P(n^l|\alpha^1, \ldots, \alpha^m; q_1, \ldots, q_m) = \int P(n^l|\theta) P(\theta|\alpha^1, \ldots, \alpha^m; q_1, \ldots, q_m) d\theta. \tag{11.29}$$

如果计数向量相互独立, 那么混合的总似然为:

$$P(\text{data}|\text{mixture}) = \prod_{l=1}^{M} P(n^l|\alpha^1, \ldots, \alpha^m; q_1, \ldots, q_m). \tag{11.30}$$

我们可用梯度下降 (gradient descent) 法或连续最优化的其他方法最大化这个概率.

此时读者也许会问: "为什么使用 ML 估计, 而不是这些刚刚学到的优美的 Bayes 方法?" 如果这样做, 仅需要第一层先验参数上的一个先验. 你可以一直为先验参数加先验. 你必须在某处勉强接受一个你所编造的, 或者由 ML 估计或是其他非 Bayes 方法得到的先验.

11.6 EM 算法

期望最大化 (expectation maximisation , EM) 算法是计算含有"缺失数据" (missing data) 的 ML 估计的一种通用算法 [Dempster, Laird & Rubin 1977]. 估计隐马模型概率的 Baum-Welch 算法是 EM 算法的特例. 对于 HMM 而言, 缺失数据是未知状态, 因为我们只知道观测结果而不知道产生它们的状态序列.

假设某统计模型由参数 θ 决定. 观测值称为 x, 而 x 的概率由某缺失数据 y 决定. 对于接下来我们要处理的 HMM, θ 是所有模型参数 a 和 e 的集合, 而 y 代表贯穿整个模型的路径. 我们的目的是找到使如下对数似然达到最大的模型:

$$\log P(x|\theta) = \log \sum_y P(x, y|\theta).$$

此处及以后, x 表示所有的观测, 可以是一条或多条序列. 为了能再次使用明确表示所有序列的记号, 我们需要在下列所有方程中对序列进行额外的求和.

现在假设已有一个有效的模型 θ^t. 我们想要估计更好的新模型 θ^{t+1}. 利用$P(x, y|\theta) = P(y|x, \theta)P(x|\theta)$, 我们可将对数似然写为:

$$\log P(x|\theta) = \log P(x, y|\theta) - \log P(y|x, \theta).$$

将其乘以 $P(y|x, \theta^t)$, 并对 y 求和得:

$$\log P(x|\theta) = \sum_y P(y|x, \theta^t) \log P(x, y|\theta) - \sum_y P(y|x, \theta^t) \log P(y|x, \theta).$$

我们称上式右边的第一项为 $Q(\theta|\theta^t)$, 即

$$Q(\theta|\theta^t) = \sum_y P(y|x, \theta^t) \log P(x, y|\theta). \tag{11.31}$$

我们希望 $\log P(x|\theta)$ 能比 $\log P(x|\theta^t)$ 大, 于是它们的差为正. 利用上面的两个方程, 我们可以将两者之差写为:

$$\log P(x|\theta) - \log P(x|\theta^t) = \\ Q(\theta|\theta^t) - Q(\theta^t|\theta^t) + \sum_y P(y|x, \theta^t) \log \frac{P(y|x, \theta^t)}{P(y|x, \theta)}.$$

上式的最后一项是 $P(y|x, \theta^t)$ 之于 $P(y|x, \theta)$ 的相对熵 (11.10), 因此它总是非负的, 于是有:

$$\log P(x|\theta) - \log P(x|\theta^t) \geq Q(\theta|\theta^t) - Q(\theta^t|\theta^t) \tag{11.32}$$

仅当 $\theta = \theta^t$, 或对某些 $\theta \neq \theta^t$ 有 $P(y|x, \theta^t) = P(y|x, \theta)$ 时等号成立. 选择

$$\theta^{t+1} = \underset{\theta}{\text{argmax}}\, Q(\theta|\theta^t). \tag{11.33}$$

能够保证差为正, 因此新模型的似然总大于 θ^t 的似然. 当然, 若已经达到最大值, 即 $\theta^{t+1} = \theta^t$, 则似然将不再改变.

式 (11.31) 中的函数 Q 是 $\log P(x,y|\theta)$ 在 y 分布上的一个平均, 其中的 y 分布从当前参数集 θ^t 得到. 它通常解析地表示为 θ 的函数, 此函数中的常数为旧模型中的期望值. 这在即将逐步推导 HMM 时会更具体化. 形式上, EM 算法通常简述为:

算法: 期望最大化

E 步骤: 计算 Q 函数 (11.31).

M 步骤: 相对于 θ, 最大化 $Q(\theta|\theta^t)$. ◁

从上面我们可以看到, 似然在每次迭代过程中都递增, 所以当 $t \to \infty$ 时, 这个过程将总是渐进地达到局部(也许是全局)最大值. 对于许多模型而言, 例如 HMM , 这两个步骤都可以解析地进行下去. 如果第二步不能准确地运行, 那么我们就可以使用某种数值优化技巧最大化 Q. 实际上并没有必要最大化 Q; 只须令 $Q(\theta^{t+1}|\theta^t)$ 大于 $Q(\theta^t|\theta^t)$ 即可. 这种递增 Q 而未必使其最大化的算法称为广义 EM 算法 (generalised EM , GEM) [Dempster, Laird & Rubin 1977]. 关于 EM 思想的其他推广见 Meng & Rubin [1992]和 Neal & Hinton [1993].

Baum-Welch 算法的EM 解释

对于 HMM, 我们现在将简单地描述形成第3章中 (本书第 p. 44 页) Baum-Welch 算法的 EM 步骤推导. 此处我们想要最大化似然:

$$\log P(x|\theta) = \sum_{\pi} \log P(x,\pi|\theta),$$

所以"缺失数据" 是状态路径 π. 于是 (11.31) 的 Q 由下式给出:

$$Q(\theta|\theta^t) = \sum_{\pi} P(\pi|x,\theta^t) \log P(x,\pi|\theta). \tag{11.34}$$

对于一条给定的路径, 模型中的每个参数都会在由乘积式 (3.6) 给出的 $P(x,\pi|\theta)$ 中出现多次. 如果它是一个转移概率, 那么我们就称这个次数为 $A_{kl}(\pi)$, 对于发射概率, 那么就称为 $E_k(b,\pi)$, 即 $E_k(b,\pi)$ 表示路径 π 中的状态 k 里出现字符 b 的次数(它依赖于观测序列, 然而此处我们并没有明确地表示出来). 于是我们可将式 (3.6) 写为:

$$P(x,\pi|\theta) = \prod_{k=1}^{M} \prod_{b} [e_k(b)]^{E_k(b,\pi)} \prod_{k=0}^{M} \prod_{l=1}^{M} a_{kl}^{A_{kl}(\pi)},$$

其中第一个连乘号遍及字母表中的所有字符 b. 取对数后, (11.34) 可以写为:

$$Q(\theta|\theta^t) = \sum_{\pi} P(\pi|x,\theta^t) \times \left[\sum_{k=1}^{M} \sum_{b} E_k(b,\pi) \log e_k(b) + \sum_{k=0}^{M} \sum_{l=1}^{M} A_{kl}(\pi) \log a_{kl} \right]. \tag{11.35}$$

我们可以看到, (本书第 p. 44 页的) Baum-Welch 算法中 (3.20) 和 (3.21) 所定义的期望值 A_{kl} 和 $E_k(b)$ 分别可以写成 $A_{kl}(\pi)$ 和 $E_k(b,\pi)$ 关于 $P(\pi|x,\theta^t)$ 的期望:

$$E_k(b) = \sum_{\pi} P(\pi|x,\theta^t) E_k(b,\pi) \quad \text{及} \quad A_{kl} = \sum_{\pi} P(\pi|x,\theta^t) A_{kl}(\pi).$$

因此在式 (11.35) 中先对 π 求和有:

$$Q(\theta|\theta^t) = \sum_{k=1}^{M} \sum_{b} E_k(b) \log e_k(b) + \sum_{k=0}^{M} \sum_{l=1}^{M} A_{kl} \log a_{kl}. \tag{11.36}$$

最后, 我们需要证明式 (3.18) 使得 (11.36) 达到最大值. 让我们首先考虑 A 项. 当 $a_{ij}^0 = \frac{A_{ij}}{\sum_k A_{ik}}$ 时的该项与其他任意 a_{ij} 时的该项间之差为:

$$\sum_{k=0}^{M} \sum_{l=1}^{M} A_{kl} \log \frac{a_{kl}^0}{a_{kl}} = \sum_{k=0}^{M} \left(\sum_{l'} A_{kl'} \right) \sum_{l=1}^{M} a_{kl}^0 \log \frac{a_{kl}^0}{a_{kl}}.$$

最后一个表达式是相对熵 (11.10), 因此除非 $a_{kl} = a^0_{kl}$, 该式都大于 0. 这就证明了最大值出现在 a^0_{kl} 处. 同样的证明过程可用于 E 项.

对 HMM 而言, EM 算法中的 E 步骤包含了计算期望 $E_k(b)$ 和 A_{kl}. 如第 3 章所述, 此步骤由前向-后向 (forward-backward) 过程完成. 这就完全确定了函数 Q, 而且最大值可由这些数直接表示. 因此, M 步骤仅是将 $E_k(b)$ 和 A_{kl} 代入到由 (3.18) 所给出的用于重估计 $e_k(b)$ 和 a_{kl} 的公式中.

Abrahams, J. P., van den Berg, M., van Batenburg, E. and Pleij, C. 1990. Prediction of RNA secondary structure, including pseudoknotting, by computer simulation. *Nucleic Acids Research* 18:3035–3044.

Allison, L. and Wallace, C. S. 1993. The posterior probability distribution of alignments and its application to parameter estimation of evolutionary trees and to optimisation of muliple alignments. Technical Report TR 93/188, Monash University Computer Science.

Allison, L., Wallace, C. S. and Yee, C. N. 1992a. Finite-state models in the alignment of macromolecules. *Journal of Molecular Evolution* 35:77–89.

Allison, L., Wallace, C. S. and Yee, C. N. 1992b. Minimum message length encoding, evolutionary trees and multiple alignment. In *Hawaii International Conference on System Sciences*, volume 1, 663–674.

Altman, R., Brutlag, D., Karp, P., Lathrop, R. and Searls, D., eds. 1994. *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology*. AAAI Press.

Altschul, S. F. 1989. Gap costs for multiple sequence alignment. *Journal of Theoretical Biology* 138:297–309.

Altschul, S. F. 1991. Amino acid substitution matrices from an information theoretic perspective. *Journal of Molecular Biology* 219:555–565.

Altschul, S. F. and Erickson, B. W. 1986. Optimal sequence alignment using affine gap costs. *Bulletin of Mathematical Biology* 48:603–616.

Altschul, S. F. and Gish, W. 1996. Local alignment statistics. *Methods in Enzymology* 266:460–480.

Altschul, S. F. and Lipman, D. J. 1989. Trees, stars, and multiple biological sequence alignment. *SIAM Journal of Applied Mathematics* 49:197–209.

Altschul, S. F., Carroll, R. J. and Lipman, D. J. 1989. Weights for data related by a tree. *Journal of Molecular Biology* 207:647–653.

Altschul, S. F., Gish, W., Miller, W., Myers, E. W. and Lipman, D. J. 1990. Basic local alignment search tool. *Journal of Molecular Biology* 215:403–410.

Altschul, S. F., Madden, T. L., Schaffer, A. A., Zhang, J., Zhang, Z., Miller, W. and Lipman, D. J. 1997. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research* 25:3389–3402.

Asai, K., Hayamizu, S. and Handa, K. 1993. Prediction of protein secondary structure by the hidden Markov model. *Computer Applications in the Biosciences* 9:141–146.

Asmussen, S. 1987. *Applied Probability and Queues*. Wiley.

Atteson, K. 1997. The performance of the neighbor-joining method of phylogeny reconstruction. In Mirkin, B., McMorris, F., Roberts, F. and Rzhetsky, A., eds., *Mathematical Hierarchies and Biology*. American Mathematical Society. 133–148.

Bahl, L. R., Brown, P. F., de Souza, P. V. and Mercer, R. L. 1986. Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In *Proceedings of ICASSP '86*, 49–52.

Bailey, T. L. and Elkan, C. 1994. Fitting a mixture model by expectation maximization to discover motifs in biopolymers. In Altman, R., Brutlag, D., Karp, P., Lathrop, R. and Searls, D., eds., *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology*, 28–36. AAAI Press.

Bailey, T. L. and Elkan, C. 1995. The value of prior knowledge in discovering motifs with MEME. In Rawlings, C., Clark, D., Altman, R., Hunter, L., Lengauer, T. and Wodak, S., eds., *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology*, 21–29. AAAI Press.

Bairoch, A. and Apweiler, R. 1997. The SWISS-PROT protein sequence data bank and its supplement TrEMBL. *Nucleic Acids Research* 25:31–36.

Bairoch, A., Bucher, P. and Hofmann, K. 1997. The PROSITE database, its status in 1997. *Nucleic Acids Research* 25:217–221.

Baldi, P. and Brunak, S. 1998. *Bioinformatics – The Machine Learning Approach*. MIT Press.

Baldi, P. and Chauvin, Y. 1994. Smooth on-line learning algorithms for hidden Markov models. *Neural Computation* 6:307–318.

Baldi, P. and Chauvin, Y. 1995. Protein modeling with hybrid hidden Markov model/neural network architectures. In Rawlings, C., Clark, D., Altman, R., Hunter, L., Lengauer, T. and Wodak, S., eds., *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology*, 39–47. AAAI Press.

Baldi, P., Brunak, S., Chauvin, Y. and Krogh, A. 1996. Naturally occurring nucleosome positioning signals in human exons. *Journal of Molecular Biology* 263:503–510.

Baldi, P., Chauvin, Y., Hunkapiller, T. and McClure, M. A. 1994. Hidden Markov models of biological primary sequence information. *Proceedings of the National Academy of Sciences of the USA* 91:1059–1063.

Bandelt, H.-J. and Dress, A. W. M. 1992. Split decomposition: a new and useful approach to phylogenetic analysis of distance data. *Molecular Phylogenetics and Evolution* 1:242–252.

Barton, G. J. 1993. An efficient algorithm to locate all locally optimal alignments between two sequences allowing for gaps. *Computer Applications in the Biosciences* 9:729–734.

Barton, G. J. and Sternberg, M. J. E. 1987. A strategy for the rapid multiple alignment of protein sequences. *Journal of Molecular Biology* 198:327–337.

Baserga, S. J. and Steitz, J. A. 1993. The diverse world of small ribonucleoproteins. In Gesteland, R. F. and Atkins, J. F., eds., *The RNA World*. Cold Spring Harbor Press. pp. 359–381.

Bashford, D., Chothia, C. and Lesk, A. M. 1987. Determinants of a protein fold: unique features of the globin amino acid sequence. *Journal of Molecular Biology* 196:199–216.

Baum, L. E. 1972. An equality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. *Inequalities* 3:1–8.

Bengio, Y., De Mori, R., Flammia, G. and Kompe, R. 1992. Global optimization of a neural network–hidden Markov model hybrid. *IEEE Transactions on Neural Networks* 3:252–259.

Berger, J. O. 1985. *Statistical Decision Theory and Bayesian Analysis*. Springer-Verlag.

Berger, M. P. and Munson, P. J. 1991. A novel randomized iterative strategy for aligning multiple protein sequences. *Computer Applications in the Biosciences* 7:479–484.

Binder, K. and Heerman, D. W. 1988. *Monte Carlo Simulation in Statistical Mechanics*. Springer-Verlag.

Bird, A. 1987. CpG islands as gene markers in the vertebrate nucleus. *Trends in Genetics* 3:342–347.

Birney, E. and Durbin, R. 1997. Dynamite: a flexible code generating language for dynamic programming methods used in sequence comparison. In Gaasterland, T., Karp, P., Karplus, K., Ouzounis, C., Sander, C. and Valencia, A., eds., *Proceedings of the Fifth International Conference on Intelligent Systems for Molecular Biology*, 56–64. AAAI Press.

Bishop, M. J. and Thompson, E. A. 1986. Maximum likelihood alignment of DNA sequences. *Journal of Molecular Biology* 190:159–165.

Borodovsky, M. and McIninch, J. 1993. GENMARK: parallel gene recognition for both DNA strands. *Computers and Chemistry* 17:123–133.

Borodovsky, M. Y., Sprizhitsky, Y. A., Golovanov, E. I. and Alexandrov, A. A. 1986a. Statistical patterns in the primary structure of the functional regions of the Escherichia coli genome. I. Frequency characteristics. *Molecularnaya Biologia* 20:826–833. (English translation).

Borodovsky, M. Y., Sprizhitsky, Y. A., Golovanov, E. I. and Alexandrov, A. A. 1986b. Statistical patterns in the primary structure of the functional regions of the Escherichia Coli genome. II. Nonuniform Markov models. *Molecularnaya Biologia* 20:833–840. (English translation).

Borodovsky, M. Y., Sprizhitsky, Y. A., Golovanov, E. I. and Alexandrov, A. A. 1986c. Statistical patterns in the primary structure of the functional regions of the Escherichia Coli genome. III. Computer recognition of coding regions. *Molecularnaya Biologia* 20:1144–1150. (English translation).

Bowie, J. U., Luthy, R. and Eisenberg, D. 1991. A method to identify protein sequences that fold into a known three-dimensional structure. *Science* 253:164–170.

Box, G. E. P. and Tiao, G. C. 1992. *Bayesian Inference in Statistical Analysis*. Wiley-Interscience.

Branden, C. and Tooze, J. 1991. *Introduction to Protein Structure*. Garland.

Brendel, V., Beckmann, J. S. and Trifonov, E. N. 1986. Linguistics of nucleotide sequences: morphology and comparison of vocabularies. *Journal of Biomolecular Structure and Dynamics* 4:11–20.

Brooks, D. R. and McLennan, D. A. 1991. *Phylogeny, Ecology and Behaviour*. University of Chicago Press.

Brown, M. and Wilson, C. 1995. RNA pseudoknot modeling using intersections of stochastic context-free grammars with applications to database search. Unpublished manuscript available from http://www.cse.ucsc.edu/research/compbio/pseudoknot.html.

Brown, M., Hughey, R., Krogh, A., Mian, I. S., Sjölander, K. and Haussler, D. 1993. Using Dirichlet mixture priors to derive hidden Markov models for protein families. In Hunter, L., Searls, D. B. and Shavlik, J., eds., *Proceedings of the First International Conference on Intelligent Systems for Molecular Biology*, 47–55. AAAI Press.

Brunak, S., Engelbrecht, J. and Knudsen, S. 1991. Prediction of human mRNA donor and acceptor sites from the DNA sequence. *Journal of Molecular Biology* 220:49–65.

Bucher, P. and Hofmann, K. 1996. A sequence similarity search algorithm based on a probabilistic interpretation of an alignment scoring system. In States, D. J., Agarwal, P., Gaasterland, T., Hunter, L. and Smith, R. F., eds., *Proceedings of the Fourth International Conference on Intelligent Systems for Molecular Biology*, 44–51. AAAI Press.

Bucher, P., Karplus, K., Moeri, N. and Hofmann, K. 1996. A flexible motif search technique based on generalized profiles. *Computers and Chemistry* 20:3–24.

Buneman, P. 1971. The recovery of trees from measures of dissimilarity. In Hodson, F. R., Kendall, D. G. and Tautu, P., eds., *Mathematics in the Archaeological and Historical Sciences*. Edinburgh University Press. pp. 387–395.

Burge, C. and Karlin, S. 1997. Prediction of complete gene structures in human genomic DNA. *Journal of Molecular Biology* 268:78–94.

Camin, J. H. and Sokal, R. R. 1965. A method for deducing branching sequences in phylogeny. *Evolution* 19:311–327.

Cardon, L. R. and Stormo, G. D. 1992. Expectation maximization algorithm for identifying protein-binding sites with variable lengths from unaligned DNA fragments. *Journal of Molecular Biology* 223:159–170.

Carrillo, H. and Lipman, D. 1988. The multiple sequence alignment problem in biology. *SIAM Journal of Applied Mathematics* 48:1073–1082.

Cary, R. B. and Stormo, G. D. 1995. Graph-theoretic approach to RNA modeling using comparative data. In Rawlings, C., Clark, D., Altman, R., Hunter, L., Lengauer, T. and Wodak, S., eds., *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology*, 75–80. AAAI Press.

Casella, G. and Berger, R. L. 1990. *Statistical Inference*. Duxbury Press.

Cavender, J. A. 1978. Taxonomy with confidence. *Mathematical Biosciences* 40:271–280.

Cech, T. R. and Bass, B. L. 1986. Biological catalysis by RNA. *Annual Review of Biochemistry* 55:599–629.

Chan, S. C., Wong, A. K. C. and Chiu, D. K. Y. 1992. A survey of multiple sequence comparison methods. *Bulletin of Mathematical Biology* 54:563–598.

Chang, W. I. and Lawler, E. L. 1990. Approximate string matching in sublinear expected time. In *Proceedings of the 31st Annual IEEE Symposium on Foundations Computer Science*, 116–124. IEEE.

Chao, K. M., Hardison, R. C. and Miller, W. 1994. Recent developments in linear-space alignment methods: a survey. *Journal of Computational Biology* 1:271–291.

Chao, K. M., Pearson, W. R. and Miller, W. 1992. Aligning two sequences within a specified diagonal band. *Computer Applications in the Biosciences* 8:481–487.

Chiu, D. K. Y. and Kolodziejczak, T. 1991. Inferring consensus structure from nucleic acid sequences. *Computer Applications in the Biosciences* 7:347–352.

Chomsky, N. 1956. Three models for the description of language. *IRE Transactions Information Theory* 2:113–124.

Chomsky, N. 1959. On certain formal properties of grammars. *Information and Control* 2:137–167.

Chothia, C. and Lesk, A. M. 1986. The relation between the divergence of sequence and structure in proteins. *EMBO Journal* 5:823–826.

Churchill, G. A. 1989. Stochastic models for heterogeneous DNA sequences. *Bulletin of Mathematical Biology* 51:79–94.

Churchill, G. A. 1992. Hidden markov chains and the analysis of genome structure. *Computers and Chemistry* 16:107–115.

Claverie, J.-M. 1994. Some useful statistical properties of position-weight matrices. *Computers and Chemistry* 18:287–294.

Collado-Vides, J. 1989. A transformational-grammar approach to the study of the regulation of gene expression. *Journal of Theoretical Biology* 136:403–425.

Collado-Vides, J. 1991. A syntactic representation of units of genetic information – a syntax of units of genetic information. *Journal of Theoretical Biology* 148:401–429.

Corpet, F. and Michot, B. 1994. RNAlign program: alignment of RNA sequences using both primary and secondary structures. *Computer Applications in the Biosciences* 10:389–399.

Cover, T. M. and Thomas, J. A. 1991. *Elements of Information Theory.* John Wiley & Sons, Inc.

Cox, D. R. 1962. Further results on tests of separate families of hypotheses. *Journal of the Royal Statistical Society, B* 24:406–424.

Cox, D. R. and Miller, H. D. 1965. *The Theory of Stochastic Processes.* Chapman & Hall.

Dandekar, T. and Hentze, M. W. 1995. Finding the hairpin in the haystack: searching for RNA motifs. *Trends in Genetics* 11:45–50.

Dayhoff, M. O., Eck, R. V. and Park, C. M. 1972. In Dayhoff, M. O., ed., *Atlas of Protein Sequence and Structure*, volume 5. National Biomedical Research Foundation, Washington D.C. pp. 89–99.

Dayhoff, M. O., Schwartz, R. M. and Orcutt, B. C. 1978. A model of evolutionary change in proteins. In Dayhoff, M. O., ed., *Atlas of Protein Sequence and Structure*, volume 5, supplement 3. National Biomedical Research Foundation, Washington D.C. pp. 345–352.

Dembo, A. and Karlin, S. 1991. Strong limit theorems of empirical functionals for large exceedances of partial sums of i.i.d. variables. *Annals of Probability* 19:1737–1755.

Dempster, A. P., Laird, N. M. and Rubin, D. B. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society* B 39:1–38.

Dong, S. and Searls, D. B. 1994. Gene structure prediction by linguistic methods. *Genomics* 23:540–551.

Doolittle, R. F., Feng, D.-F., Tsang, S., Cho, G. and Little, E. 1996. Determining divergence times of the major kingdoms of living organisms with a protein clock. *Science* 271:470–477.

Eck, R. V. and Dayhoff, M. O. 1966. *Atlas of Protein Sequence and Structure.* National Biomedical Research Foundation.

Eddy, S. R. 1995. Multiple alignment using hidden Markov models. In Rawlings, C., Clark, D., Altman, R., Hunter, L., Lengauer, T. and Wodak, S., eds., *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology*, 114–120. AAAI Press.

Eddy, S. R. 1996. Hidden Markov models. *Current Opinion in Structural Biology* 6:361–365.

Eddy, S. R. and Durbin, R. 1994. RNA sequence analysis using covariance models. *Nucleic Acids Research* 22:2079–2088.

Eddy, S. R., Mitchison, G. and Durbin, R. 1995. Maximum discrimination hidden Markov models of sequence consensus. *Journal of Computational Biology* 2:9–23.

Edwards, A. W. F. 1970. Estimation of the branch points of a branching diffusion process. *Journal of the Royal Statistical Society, B* 32:155–174.

Edwards, A. W. F. 1992. *Likelihood*. Johns Hopkins Universty Press.

Edwards, A. W. F. 1996. The origin and early development of the method of minimum evolution for the reconstruction of phylogenetic trees. *Systematic Biology* 45:179–191.

Edwards, A. W. F. and Cavalli-Sforza, L. 1963. The reconstruction of evolution. *Annals of Human Genetics* 27:105.

Edwards, A. W. F. and Cavalli-Sforza, L. 1964. Reconstruction of evolutionary trees. In Heywood, V. H. and McNeill, J., eds., *Phenetic and Phylogenetic Classification*. Systematics Association Publication No. 6. pp. 67–76.

Efron, B. and Tibshirani, R. J. 1993. *An Introduction to the Bootstrap*. Chapman and Hall.

Efron, B., Halloran, E. and Holmes, S. 1996. Bootstrap confidence levels for phylogenetic trees. *Proceedings of the National Academy of Sciences of the USA* 93:13429–13434.

Feller, W. 1971. *An Introduction to Probability Theory and its Applications, Vol II.* John Wiley and Sons.

Felsenstein, J. 1973. Maximum-likelihood estimation of evolutionary trees from continuous characters. *American Journal of Human Genetics* 25:471–492.

Felsenstein, J. 1978a. Cases in which parsimony or compatibility methods will be positively misleading. *Systematic Zoology* 27:401–410.

Felsenstein, J. 1978b. The number of evolutionary trees. *Systematic Zoology* 27:27–33.

Felsenstein, J. 1981a. Evolutionary trees from DNA sequences: a maximum likelihood approach. *Journal of Molecular Evolution* 17:368–376.

Felsenstein, J. 1981b. A likelihood approach to character weighting and what it tells us about parsimony and compatibility. *Biological Journal of the Linnean Society* 16:183–196.

Felsenstein, J. 1985. Confidence limits on phylogenies: an approach using the bootstrap. *Evolution* 39:783–791.

Felsenstein, J. 1996. Inferring phylogenies from protein sequences by parsimony, distance, and likelihood methods. *Methods in Enzymology* 266:418–427.

Felsenstein, J. and Churchill, G. A. 1996. A hidden Markov model approach to variation among sites in rate of evolution. *Molecular Biology and Evolution* 13:93–104.

Feng, D.-F. and Doolittle, R. F. 1987. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *Journal of Molecular Evolution* 25:351–360.

Feng, D.-F. and Doolittle, R. F. 1996. Progressive alignment of amino acid sequences and construction of phylogenetic trees from them. *Methods in Enzymology* 266:368–382.

Fichant, G. A. and Burks, C. 1991. Identifying potential tRNA genes in genomic DNA sequences. *Journal of Molecular Biology* 220:659–671.

Fields, D. S. and Gutell, R. R. 1996. An analysis of large rRNA sequences folded by a thermodynamic method. *Folding and Design* 1:419–430.

Fitch, W. M. 1971. Toward defining the course of evolution: minimum change for a specifed tree topology. *Systematic Zoology* 20:406–416.

Fitch, W. M. and Margoliash, E. 1967a. Construction of phylogenetic trees. *Science* 155:279–284.

Fitch, W. M. and Margoliash, E. 1967b. A method for estimating the number of invariant amino acid coding positions in a gene using cytochrome c as a model case. *Biochemical Genetics* 1:65–71.

Frasconi, P. and Bengio, Y. 1994. An EM approach to grammatical inference: input/output HMMs. In *Proceedings of the 12th IAPR International Conference on Pattern Recognition*, volume 2, 289–294. IEEE Comput. Soc. Press.

Freier, S. M., Kierzek, R., Jaeger, J. A., Sugimoto, N., Caruthers, M. H., Neilson, T. and Turner, D. H. 1986. Improved free-energy parameters for predictions of RNA duplex stability. *Proceedings of the National Academy of Sciences of the USA* 83:9373–9377.

Gaasterland, T., Karp, P., Karplus, K., Ouzounis, C., Sander, C. and Valencia, A., eds. 1997. *Proceedings of the Fifth International Conference on Intelligent Systems for Molecular Biology*. AAAI Press.

Gautheret, D., Major, F. and Cedergren, R. 1990. Pattern searching/alignment with RNA primary and secondary structures: an effective descriptor for tRNA. *Computer Applications in the Biosciences* 6:325–331.

Gerstein, M. and Levitt, M. 1996. Using iterative dynamic programming to obtain accurate pairwise and multiple alignments of protein structures. In States, D. J., Agarwal, P., Gaasterland, T., Hunter, L. and Smith, R. F., eds., *Proceedings of the Fourth International Conference on Intelligent Systems for Molecular Biology*, 59–67. AAAI Press.

Gerstein, M., Sonnhammer, E. L. L. and Chothia, C. 1994. Volume changes in protein evolution. *Journal of Molecular Biology* 236:1067–1078.

Gersting, J. L. 1993. *Mathematical Structures for Computer Science*. W. H. Freeman.

Gesteland, R. F. and Atkins, J. F., eds. 1993. *The RNA World*. Cold Spring Harbor Laboratory Press.

Gilbert, W. 1986. The RNA world. *Nature* 319:618.

Gold, L., Polisky, B., Uhlenbeck, O. and Yarus, M. 1995. Diversity of oligonucleotide functions. *Annual Review of Biochemistry* 64:763–797.

Goldman, N. 1993. Statistical tests of models of DNA substitution. *Journal of Molecular Evolution* 36:182–198.

Goldman, N. and Yang, Z. 1994. A codon-based model of nucleotide substitution for protein-coding DNA sequences. *Molecular Biology and Evolution* 11:725–735.

Gonnet, G. H., Cohen, M. A. and Benner, S. A. 1992. Exhaustive matching of the entire protein sequence database. *Science* 256:1443–1445.

Gotoh, O. 1982. An improved algorithm for matching biological sequences. *Journal of Molecular Biology* 162:705–708.

Gotoh, O. 1993. Optimal alignment between groups of sequences and its application to multiple sequence alignment. *Computer Applications in the Biosciences* 9:361–370.

Gotoh, O. 1996. Significant improvement in accuracy of multiple protein alignments by iterative refinement as assessed by reference to structural alignments. *Journal of Molecular Biology* 264:823–838.

Grate, L. 1995. Automatic RNA secondary structure determination with stochastic context-free grammars. In Rawlings, C., Clark, D., Altman, R., Hunter, L., Lengauer, T. and Wodak, S., eds., *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology*, 136–144. AAAI Press.

Gribskov, M. and Veretnik, S. 1996. Identification of sequence patterns with profile analysis. *Methods in Enzymology* 266:198–212.

Gribskov, M., Lüthy, R. and Eisenberg, D. 1990. Profile analysis. *Methods in Enzymology* 183:146–159.

Gribskov, M., McLachlan, A. D. and Eisenberg, D. 1987. Profile analysis: detection of distantly related proteins. *Proceedings of the National Academy of Sciences of the USA* 84:4355–4358.

Gultyaev, A. P. 1991. The computer simulation of RNA folding involving pseudoknot formation. *Nucleic Acids Research* 19:2489–2494.

Gumbel, E. J. 1958. *Statistics of Extremes*. Columbia University Press.

Gupta, S. K., Kececioglu, J. D. and Schaffer, A. A. 1995. Improving the practical space and time efficiency of the shortest-paths approach to sum-of-pairs multiple sequence alignment. *Journal of Computational Biology* 2:459–472.

Gutell, R. R. 1993. Collection of small subunit (16S and 16S-like) ribosomal RNA structures. *Nucleic Acids Research* 21:3051–3054.

Gutell, R. R., Power, A., Hertz, G. Z., Putz, E. J. and Stormo, G. D. 1992. Identifying constraints on the higher-order structure of RNA: continued development and application of comparative sequence analysis methods. *Nucleic Acids Research* 20:5785–5795.

Hannenhalli, S., Chappey, C., Koonin, E. V. and Pevsner, P. A. 1995. Genome sequence comparison and scenarios for gene rearrangements: a test case. *Genomics* 30:299–311.

Harpaz, Y. and Chothia, C. 1994. Many of the immunoglobulin superfamily domains in cell adhesion molecules and surface receptors belong to a new structural set which is close to that containing variable domains. *Journal of Molecular Biology* 238:528–539.

Harrison, M. A. 1978. *Introduction to Formal Language Theory*. Addison-Wesley.

Hasegawa, M., Kishino, H. and Yano, T. 1985. Dating the human-ape splitting by a molecular clock of mitochondrial DNA. *Journal of Molecular Evolution* 22:160–174.

Haussler, D., Krogh, A., Mian, I. S. and Sjölander, K. 1993. Protein modeling using hidden Markov models: analysis of globins. In Mudge, T. N., Milutinovic, V. and Hunter, L., eds., *Proceedings of the Twenty-Sixth Annual Hawaii International Conference on System Sciences*, volume 1, 792–802. IEEE Computer Society Press.

Hebsgaard, S. M., Korning, P. G., Tolstrup, N., Engelbrecht, J., Rouzé, P. and Brunak, S. 1996. Splice site prediction in Arabidopsis thaliana pre-mRNA by combining local and global sequence information. *Nucleic Acids Research* 24:3439–3452.

Hein, J. 1989a. A new method that simultaneously aligns and reconstructs ancestral sequences for any number of homologous sequences, when the phylogeny is given. *Molecular Biology and Evolution* 6:649–668.

Hein, J. 1989b. A tree reconstruction method that is economical in the number of pairwise comparisons used. *Molecular Biology and Evolution* 6:669–684.

Hein, J. 1993. A heuristic method to reconstruct the history of sequences subject to recombination. *Journal of Molecular Evolution* 36:396–405.

Henderson, J., Salzberg, S. and Fasman, K. H. 1997. Finding genes in DNA with a hidden Markov model. *Journal of Computational Biology* 4:127–141.

Hendy, M. D. and Penny, D. 1989. A framework for the quantitative study of evolutionary trees. *Systematic Zoology* 38:297–309.

Henikoff, J. G. and Henikoff, S. 1996. Using substitution probabilities to improve position-specific scoring matrices. *Computer Applications in the Biosciences* 12:135–143.

Henikoff, S. and Henikoff, J. G. 1991. Automated assembly of protein blocks for database searching. *Nucleic Acids Research* 19:6565–6572.

Henikoff, S. and Henikoff, J. G. 1992. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences of the USA* 89:10915–10919.

Henikoff, S. and Henikoff, J. G. 1994. Position-based sequence weights. *Journal of Molecular Biology* 243:574–578.

Hertz, G. Z., Hartzell III, G. W. and Stormo, G. D. 1990. Identification of consensus patterns in unaligned DNA sequences known to be functionally related. *Computer Applications in the Biosciences* 6:81–92.

Higgins, D. G. and Sharp, P. M. 1989. Fast and sensitive multiple sequence alignments on a microcomputer. *Computer Applications in the Biosciences* 5:151–153.

Higgins, D. G., Bleasby, A. J. and Fuchs, R. 1992. CLUSTAL V: improved software for multiple sequence alignment. *Computer Applications in the Biosciences* 8:189–191.

Hillis, D. M. and Bull, J. J. 1993. An empirical test of bootstrapping as a method for assessing confidence in phylogenetic analysis. *Systematic Biology* 42:182–192.

Hillis, D. M., Bull, J. J., White, M. E., Badgett, M. R. and Molineux, I. J. 1992. Experimental phylogenetics: generation of a known phylogeny. *Science* 255:589–592.

Hirosawa, M., Hoshida, M., Ishikawa, M. and Toya, T. 1993. MASCOT: multiple alignment system for protein sequences based on three-way dynamic programming. *Computer Applications in the Biosciences* 9:161–167.

Hirschberg, D. S. 1975. A linear space algorithm for computing maximal common subsequences. *Communications of the ACM* 18:341–343.

Hogeweg, P. and Hesper, B. 1984. The alignment of sets of sequences and the construction of phyletic trees: an integrated method. *Journal of Molecular Evolution* 20:175–186.

Holm, L. and Sander, C. 1993. Protein structure comparison by alignment of distance matrices. *Journal of Molecular Biology* 233:123–138.

Hopcroft, J. E. and Ullman, J. D. 1979. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley.

Huang, X. and Zhang, J. 1996. Methods for comparing a DNA sequence with a protein sequence. *Computer Applications in the Biosciences* 12:497–506.

Hudson, R. R. 1990. Gene genealogies and the coalescent process. In Futuyma, D. and Antonovics, J., eds., *Gene Genealogies and the Coalescent Process*. Oxford University Press. pp. 1–44.

Huelsenbeck, J. P. and Rannala, B. 1997. Phylogenetic methods come of age: testing hypotheses in an evolutionary context. *Science* 276:227–232.

Hughey, R. and Krogh, A. 1996. Hidden Markov models for sequence analysis: extension and analysis of the basic method. *Computer Applications in the Biosciences* 12:95–107.

Jacob, F. 1977. Evolution and tinkering. *Science* 196:1161–1166.

Jefferys, W. H. and Berger, J. O. 1992. Ockham's razor and Bayesian analysis. *American Scientist* 80:64–72.

Juang, B. H. and Rabiner, L. R. 1991. Hidden Markov models for speech recognition. *Technometrics* 33:251–272.

Jukes, T. H. and Cantor, C. 1969. Evolution of protein molecules. In *Mammalian Protein Metabolism*. Academic Press. pp. 21–132.

Karlin, S. and Altschul, S. F. 1990. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proceedings of the National Academy of Sciences of the USA* 87:2264–2268.

Karlin, S. and Altschul, S. F. 1993. Applications and statistics for multiple high-scoring segments in molecular sequences. *Proceedings of the National Academy of Sciences of the USA* 90:5873–5877.

Karplus, K. 1995. Evaluating regularizers for estimating distributions of amino acids. In Rawlings, C., Clark, D., Altman, R., Hunter, L., Lengauer, T. and Wodak, S., eds., *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology*, 188–196. AAAI Press.

Keeping, E. S. 1995. *Introduction to Statistical Inference*. Dover Publications.

Kim, J. and Pramanik, S. 1994. An efficient method for multiple sequence alignment. In Altman, R., Brutlag, D., Karp, P., Lathrop, R. and Searls, D., eds., *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology*, 212–218. AAAI Press.

Kim, J., Pramanik, S. and Chung, M. J. 1994. Multiple sequence alignment using simulated annealing. *Computer Applications in the Biosciences* 10:419–426.

Kimura, M. 1980. A simple method for estimating evolutionary rates of base substitutions through comparative studies of necleotide sequences. *Journal of Molecular Evolution* 16:111–120.

Kimura, M. 1983. *The Neutral Theory of Molecular Evolution*. Cambridge University Press.

Kingman, J. F. C. 1982a. The coalescent. *Stochastic Processes and their Applications* 13:235–248.

Kingman, J. F. C. 1982b. On the genealogy of large populations. *Journal of Applied Probability* 19A:27–43.

Kirkpatrick, S., Gelatt, Jr., C. D. and Vecchi, M. P. 1983. Optimization by simulated annealing. *Science* 220:671–680.

Kishino, H., Miyata, T. and Hasegawa, M. 1990. Maximum likelihood inference of protein phylogeny and the origin of chloroplasts. *Journal of Molecular Evolution* 31:151–160.

Konings, D. A. M. and Gutell, R. R. 1995. A comparison of thermodynamic foldings with comparatively derived structures of 16S and 16S-like rRNAs. *RNA* 1:559–574.

Konings, D. A. M. and Hogeweg, P. 1989. Pattern analysis of RNA secondary structure: similarity and consensus of minimal-energy folding. *Journal of Molecular Biology* 207:597–614.

Krogh, A. 1994. Hidden Markov models for labeled sequences. In *Proceedings of the 12th IAPR International Conference on Pattern Recognition*, 140–144. IEEE Computer Society Press.

Krogh, A. 1997a. Gene finding: putting the parts together. In Bishop, M., ed., *Guide to Human Genome Computing*. Academic Press, 2nd edition. To appear.

Krogh, A. 1997b. Two methods for improving performance of a HMM and their application for gene finding. In Gaasterland, T., Karp, P., Karplus, K., Ouzounis, C., Sander, C. and Valencia, A., eds., *Proceedings of the Fifth International Conference on Intelligent Systems for Molecular Biology*, 179–186. AAAI Press.

Krogh, A. 1998. An introduction to hidden Markov models for biological sequences. In Salzberg, S., Searls, D. and Kasif, S., eds., *Computational Biology: Pattern Analysis and Machine Learning Methods*. Elsevier. Chapter 4. In press.

Krogh, A. and Mitchison, G. 1995. Maximum entropy weighting of aligned sequences of proteins or DNA. In Rawlings, C., Clark, D., Altman, R., Hunter, L., Lengauer, T. and Wodak, S., eds., *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology*, 215–221. AAAI Press.

Krogh, A., Mian, I. S. and Haussler, D. 1994. A hidden Markov model that finds genes in *E. coli* DNA. *Nucleic Acids Research* 22:4768–4778.

Krogh, A., Brown, M., Mian, I. S., Sjölander, K. and Haussler, D. 1994. Hidden Markov models in computational biology: applications to protein modeling. *Journal of Molecular Biology* 235:1501–1531.

Kuhner, M. K., Yamato, J. and Felsenstein, J. 1995. Estimating effective population size and mutation rate from sequence data using Metropolis–Hastings sampling. *Genetics* 140:1421–1430.

Kulp, D., Haussler, D., Reese, M. G. and Eeckman, F. H. 1996. A generalized hidden Markov model for the recognition of human genes in DNA. In States, D. J., Agarwal, P., Gaasterland, T., Hunter, L. and Smith, R. F., eds., *Proceedings of the Fourth International Conference on Intelligent Systems for Molecular Biology*, 134–142. AAAI Press.

Langley, C. H. and Fitch, W. M. 1974. An examination of the constancy of the rate of molecular evolution. *Journal of Molecular Evolution* 3:161–177.

Lari, K. and Young, S. J. 1990. The estimation of stochastic context-free grammars using the inside–outside algorithm. *Computer Speech and Language* 4:35–56.

Lari, K. and Young, S. J. 1991. Applications of stochastic context-free grammars using the inside–outside algorithm. *Computer Speech and Language* 5:237–257.

Larsen, N. and Zwieb, C. 1993. The signal recognition particle database (SRPDB). *Nucleic Acids Research* 21:3019–3020.

Law, A. M. and Kelton, W. D. 1991. *Simulation Modelling and Analysis*. McGraw-Hill.

Lawrence, C. E. and Reilly, A. A. 1990. An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. *Proteins* 7:41–51.

Lawrence, C. E., Altschul, S. F., Boguski, M. S., Liu, J. S., Neuwald, A. F. and Wootton, J. C. 1993. Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science* 262:208–214.

Lefebvre, F. 1995. An optimized parsing algorithm well suited to RNA folding. In Rawlings, C., Clark, D., Altman, R., Hunter, L., Lengauer, T. and Wodak, S., eds., *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology*, 222–230. AAAI Press.

Lefebvre, F. 1996. A grammar-based unification of several alignment and folding algorithms. In States, D. J., Agarwal, P., Gaasterland, T., Hunter, L. and Smith, R. F., eds., *Proceedings of the Fourth International Conference on Intelligent Systems for Molecular Biology*, 143–154. AAAI Press.

Lindenmayer, A. 1968. Mathematical models for cellular interactions in development I. filaments with one-sided inputs. *Journal of Theoretical Biology* 18:280–299.

Lipman, D. J., Altschul, S. F. and Kececioglu, J. D. 1989. A tool for multiple sequence alignment. *Proceedings of the National Academy of Sciences of the USA* 86:4412–4415.

Lisacek, F., Diaz, Y. and Michel, F. 1994. Automatic identification of group I intron cores in genomic DNA sequences. *Journal of Molecular Biology* 235:1206–1217.

Lowe, T. M. and Eddy, S. R. 1997. tRNAscan-SE: a program for improved detection of transfer RNA genes in genomic sequence. *Nucleic Acids Research* 25:955–964.

Lukashin, A. V., Engelbrecht, J. and Brunak, S. 1992. Multiple alignment using simulated annealing: branch point definition in human mRNA splicing. *Nucleic Acids Research* 20:2511–2516.

Luthy, R., McLachlan, A. D. and Eisenberg, D. 1991. Secondary structure-based profiles: use of structure-conserving scoring tables in searching protein sequence databases for structural similarities. *Proteins* 10:229–239.

Luthy, R., Xenarios, I. and Bucher, P. 1994. Improving the sensitivity of the sequence profile method. *Protein Science* 3:139–146.

MacKay, D. J. C. 1992. Bayesian interpolation. *Neural Computation* 4:415–447.

MacKay, D. J. C. and Peto, L. 1995. A hierarchical Dirichlet language model. *Natural Language Engineering* 1:1–19.

Margalit, H., Shapiro, B. A., Oppenheim, A. B. and Maizel, J. V. 1989. Detection of common motifs in RNA secondary structures. *Nucleic Acids Research* 17:4829–4845.

Mau, B., Newton, M. A. and Larget, B. 1996. Bayesian phylogenetic inference via Markov chain Monte Carlo methods. Technical Report 961, Statistics Department, University of Wisconsin-Madison.

Maxwell, E. S. and Fournier, M. J. 1995. The small nucleolar RNAs. *Annual Review of Biochemistry* 64:897–934.

McCaskill, J. S. 1990. The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers* 29:1105–1119.

McClure, M. A., Vasi, T. K. and Fitch, W. M. 1994. Comparative analysis of multiple protein-sequence alignment methods. *Journal of Molecular Evolution* 11:571–592.

McKeown, M. 1992. Alternative mRNA splicing. *Annual Review of Cell Biology* 8:133–155.

Melefors, O. and Hentze, M. W. 1993. Translational regulation by mRNA/protein interactions in eukaryotic cells: ferritin and beyond. *BioEssays* 15:85–90.

Meng, X.-L. and Rubin, D. B. 1992. Recent extensions to the EM algorithm. *Bayesian Statistics* 4:307–320.

Mevissen, H. T. and Vingron, M. 1996. Quantifying the local reliability of a sequence alignment. *Protein Engineering* 9:127–132.

Miller, W. and Myers, E. W. 1988. Sequence comparison with concave weighting functions. *Bulletin of Mathematical Biology* 50:97–120.

Mitchison, G. 1998. Probabilistic modelling of phylogeny and alignment. *Molecular Biology and Evolution* submitted.

Mitchison, G. and Durbin, R. 1995. Tree-based maximal likelihood substitution matrices and hidden Markov models. *Journal of Molecular Evolution* 41:1139–1151.

Miyazawa, S. 1994. A reliable sequence alignment method based on probabilities of residue correspondence. *Protein Engineering* 8:999–1009.

Mott, R. 1992. Maximum likelihood estimation of the statistical distribution of Smith–Waterman local sequence similarity scores. *Bulletin of Mathematical Biology* 54:59–75.

Myers, E. W. 1994. A sublinear algorithm for approximate keyword searching. *Algorithmica* 12:345–374.

Myers, G. 1995. Approximately matching context-free languages. *Information Processing Letters* 54:85–92.

Neal, R. M. 1996. *Bayesian Learning in Neural Networks.* Springer (Lecture Notes in Statistics).

Neal, R. M. and Hinton, G. E. 1993. A new view of the EM algorithm that justifies incremental and other variants. Preprint, Dept. of Computer Science, Univ. of Toronto, available from ftp://archive.cis.ohio-state.edu/pub/neuroprose/neal.em.ps.Z.

Needleman, S. B. and Wunsch, C. D. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology* 48:443–453.

Noller, H. F., Hoffarth, V. and Zimniak, L. 1992. Unusual resistance of peptidyl transferase to protein extraction procedures. *Science* 256:1416–1419.

Normandin, Y. and Morgera, S. D. 1991. An improved MMIE training algorithm for speaker-independent, small vocabulary, continuous speech recognition. In *Proceedings of ICASSP '91*, 537–540.

Nussinov, R., Pieczenik, G., Griggs, J. R. and Kleitman, D. J. 1978. Algorithms for loop matchings. *SIAM Journal of Applied Mathematics* 35:68–82.

Pavesi, A., Conterlo, F., Bolchi, A., Dieci, G. and Ottonello, S. 1994. Identification of new eukaryotic tRNA genes in genomic DNA databases by a multistep weight matrix analysis of transcriptional control regions. *Nucleic Acids Research* 22:1247–1256.

Pearson, W. R. 1995. Comparison of methods for searching protein sequence databases. *Protein Science* 4:1145–1160.

Pearson, W. R. 1996. Effective protein sequence comparison. *Methods in Enzymology* 266:227–258.

Pearson, W. R. and Lipman, D. J. 1988. Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences of the USA* 4:2444–2448.

Pearson, W. R. and Miller, W. 1992. Dynamic programming algorithms for biological sequence comparison. *Methods in Enzymology* 210:575–601.

Pedersen, A. G., Baldi, P., Brunak, S. and Chauvin, Y. 1996. Characterization of prokaryotic and eukaryotic promoters using hidden Markov models. In States, D. J., Agarwal, P., Gaasterland, T., Hunter, L. and Smith, R. F., eds., *Proceedings of the Fourth International Conference on Intelligent Systems for Molecular Biology*, 182–191. AAAI Press.

Peltz, S. W. and Jacobson, A. 1992. mRNA stability: in trans-it. *Current Opinion in Cell Biology* 4:979–983.

Pesole, G., Attimonelli, M. and Saccone, C. 1994. Linguistic approaches to the analysis of sequence information. *Trends in Biotechnology* 12:401–408.

Pietrokovski, S., Hirshon, J. and Trifonov, E. N. 1990. Linguistic measure of taxonomic and functional relatedness of nucleotide sequences. *Journal of Biomolecular Structure and Dynamics* 7:1251–1268.

Preparata, F. P. and Shamos, M. I. 1985. *Computational Geometry*. Springer-Verlag.

Press, W. H., Teukolsky, S. A., Vetterling, W. T. and Flannery, B. P. 1992. *Numerical Recipes in C*. Cambridge University Press.

Rabiner, L. R. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE* 77:257–286.

Rabiner, L. R. and Juang, B. H. 1986. An introduction to hidden Markov models. *IEEE ASSP Magazine* 3:4–16.

Rabiner, L. R. and Juang, B. H. 1993. *Fundamentals of Speech Recognition*. Prentice-Hall.

Rannala, B. and Yang, Z. 1996. Probability distribution of molecular evolutionary trees: a new method of phylogenetic inference. *Journal of Molecular Evolution* 43:304–311.

Rawlings, C., Clark, D., Altman, R., Hunter, L., Lengauer, T. and Wodak, S., eds. 1995. *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology*. AAAI Press.

Reese, M. G., Eeckman, F. H., Kulp, D. and Haussler, D. 1997. Improved splice site detection in Genie. *Journal of Computational Biology* 4:311–323.

Renals, S., Morgan, N., Bourlard, H., Cohen, M. and Franco, H. 1994. Connectionist probability estimators in hmm speech recognition. *IEEE Transactions on Speech and Audio Processing* 2:161–174.

Riis, S. K. and Krogh, A. 1997. Hidden neural networks: a framework for HMM/NN hybrids. In *Proceedings of ICASSP '97*, 3233–3236. IEEE.

Ripley, B. D. 1996. *Pattern Recognition and Neural Networks*. Cambridge University Press.

Rosenblueth, D. A., Thieffry, D., Huerta, A. M., Salgado, H. and Collado-Vides, J. 1996. Syntactic recognition of regulatory regions in *Escherichia coli*. *Computer Applications in the Biosciences* 12:415–422.

Russell, R. B. and Barton, G. J. 1992. Multiple protein sequence alignment from tertiary structure comparison: assignment of global and residue confidence levels. *Proteins* 14:309–323.

Saitou, N. 1996. Reconstruction of gene trees from sequence data. *Methods in Enzymology* 266:427–448.

Saitou, N. and Nei, M. 1987. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution* 4:406–425.

Sakakibara, Y., Brown, M., Hughey, R., Mian, I. S., Sjölander, K., Underwood, R. C. and Haussler, D. 1994. Stochastic context-free grammars for tRNA modeling. *Nucleic Acids Research* 22:5112–5120.

Sankoff, D. 1975. Minimal mutation trees of sequences. *SIAM Journal of Applied Mathematics* 28:35–42.

Sankoff, D. and Cedergren, R. J. 1983. Simultaneous comparison of three or more sequences related by a tree. In Sankoff, D. and Kruskal, J. B., eds., *Time Warps, String Edits, and Macromolecules: the Theory and Practice of Sequence Comparison*. Addison-Wesley. Chapter 9, pp. 253–264.

Sankoff, D. and Kruskal, J. B. 1983. *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Addison-Wesley.

Sankoff, D., Morel, C. and Cedergren, R. J. 1973. Evolution of 5S RNA and the nonrandomness of base replacement. *Nature New Biology* 245:232–234.

Schneider, T. D. and Stephens, R. M. 1990. Sequence logos: a new way to display consensus sequences. *Nucleic Acids Research* 18:6097–6100.

Schuster, P. 1995. How to search for RNA structures. Theoretical concepts in evolutionary biotechnology. *Journal of Biotechnology* 41:239–257.

Schuster, P., Fontana, W., Stadler, P. F. and Hofacker, I. L. 1994. From sequences to shapes and back: a case study in RNA secondary structures. *Proceedings of the Royal Society: Biological Sciences, Series B* 255:279–284.

Schwartz, R. and Chow, Y.-L. 1990. The N-best algorithm: an efficient and exact procedure for finding the n most likely hypotheses. In *Proceedings of ICASSP'90*, 81–84.

Searls, D. B. 1992. The linguistics of DNA. *American Scientist* 80:579–591.

Searls, D. B. and Murphy, K. P. 1995. Automata-theoretic models of mutation and alignment. In Rawlings, C., Clark, D., Altman, R., Hunter, L., Lengauer, T. and Wodak, S., eds., *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology*, 341–349. AAAI Press.

Shapiro, B. A. and Wu, J. C. 1996. An annealing mutation operator in the genetic algorithms for RNA folding. *Computer Applications in the Biosciences* 12:171–180.

Shapiro, B. A. and Zhang, K. 1990. Comparing multiple RNA secondary structures using tree comparisons. *Computer Applications in the Biosciences* 6:309–318.

Shimamura, M., Yasue, H., Ohshima, K., Abe, H., Kato, H., Kishiro, T., Goto, M., Munechika, I. and Okada, N. 1997. Molecular evidence from retroposons that whales form a clade within even-toed ungulates. *Nature* 388:666–670.

Shpaer, E. G., Robinson, M., Yee, D., Candlin, J. D., Mines, R. and Hunkapiller, T. 1996. Sensitivity and selectivity in protein similarity searches: a comparison of Smith–Waterman in hardware to BLAST and FASTA. *Genomics* 38:179–191.

Sibbald, P. R. and Argos, P. 1990. Weighting aligned protein or nucleic acid sequences to correct for unequal representation. *Journal of Molecular Biology* 216:813–818.

Sjölander, K., Karplus, K., Brown, M., Hughey, R., Krogh, A., Mian, I. S. and Haussler, D. 1996. Dirichlet mixtures: a method for improved detection of weak but significant protein sequence homology. *Computer Applications in the Biosciences* 12:327–345.

Smith, T. F. and Waterman, M. S. 1981. Identification of common molecular subsequences. *Journal of Molecular Biology* 147:195–197.

Sokal, R. R. and Michener, C. D. 1958. A statistical method for evaluating systematic relationships. *University of Kansas Scientific Bulletin* 28:1409–1438.

Sonnhammer, E. L. L., Eddy, S. R. and Durbin, R. 1997. Pfam: a comprehensive database of protein domain families based on seed alignments. *Proteins* 28:405–420.

Staden, R. 1988. Methods to define and locate patterns of motifs in sequences. *Computer Applications in the Biosciences* 4:53–60.

States, D. J., Agarwal, P., Gaasterland, T., Hunter, L. and Smith, R. F., eds. 1996. *Proceedings of the Fourth International Conference on Intelligent Systems for Molecular Biology*. AAAI Press.

Steinberg, S., Misch, A. and Sprinzl, M. 1993. Compilation of tRNA sequences and sequences of tRNA genes. *Nucleic Acids Research* 21:3011–3015.

Stolcke, A. and Omohundro, S. M. 1993. Hidden Markov model induction by Bayesian model merging. In Hanson, S. J., Cowan, J. D. and Giles, C. L., eds., *Advances in Neural Information Processing Systems 5*, volume 5, 11–18. Morgan Kaufmann Publishers, Inc.

Stormo, G. D. 1990. Consensus patterns in DNA. *Methods in Enzymology* 183:211–221.

Stormo, G. D. and Hartzell III, G. W. 1989. Identifying protein–binding sites from unaligned DNA fragments. *Proceedings of the National Academy of Sciences of the USA* 86:1183–1187.

Stormo, G. D. and Haussler, D. 1994. Optimally parsing a sequence into different classes based on multiple types of evidence. In Altman, R., Brutlag, D., Karp, P., Lathrop, R. and Searls, D., eds., *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology*, 369–375. AAAI Press.

Studier, J. A. and Keppler, K. J. 1988. A note on the neighbour-joining algorithm of Saitou and Nei. *Molecular Biology and Evolution* 5:729–731.

Swofford, D. L. and Olsen, G. J. 1996. Phylogeny reconstruction. In Hillis, D. M. and Moritz, C., eds., *Molecular Systematics*. Sinauer Associates. pp. 407–511.

Tatusov, R. L., Altschul, S. F. and Koonin, E. V. 1994. Detection of conserved segments in proteins: iterative scanning of sequence databases with alignment blocks. *Proceedings of the National Academy of Sciences of the USA* 91:12091–12095.

Taylor, W. R. 1987. Multiple sequence alignment by a pairwise algorithm. *Computer Applications in the Biosciences* 3:81–87.

Thompson, E. A. 1975. *Human Evolutionary Trees*. Cambridge University Press.

Thompson, J. D., Higgins, D. G. and Gibson, T. J. 1994a. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position specific gap penalties and weight matrix choice. *Nucleic Acids Research* 22:4673–4680.

Thompson, J. D., Higgins, D. G. and Gibson, T. J. 1994b. Improved sensitivity of profile searches through the use of sequence weights and gap excision. *Computer Applications in the Biosciences* 10:19–29.

Thorne, J. L., Kishino, H. and Felsenstein, J. 1992. Inching toward reality: an improved likelihood model of sequence evolution. *Methods in Enzymology* 34:3–16.

Tolstrup, N., Rouzé, P. and Brunak, S. 1997. A branch point consensus from Arabidopsis found by non-circular analysis allows for better prediction of acceptor sites. *Nucleic Acids Research* 25:3159–3164.

Tuerk, C., MacDougal, S. and Gold, L. 1992. RNA pesudoknots that inhibit human immunodeficiency virus type 1 reverse transcriptase. *Proceedings of the National Academy of Sciences of the USA* 89:6988–6992.

Turner, D. H., Sugimoto, N., Jaeger, J. A., Longfellow, C. E., Freier, S. M. and Kierzek, R. 1987. Improved parameters for prediction of RNA structure. *Cold Spring Harbor Symposia Quantitative Biology* 52:123–133.

van Batenburg, F. H. D., Gultyaev, A. P. and Pleij, C. W. A. 1995. An APL-programmed genetic algorithm for the prediction of RNA secondary structure. *Journal of Theoretical Biology* 174:269–280.

Vingron, M. 1996. Near-optimal sequence alignment. *Current Opinion in Structural Biology* 6:346–352.

Vingron, M. and Waterman, M. S. 1994. Sequence alignment and penalty choice: review of concepts, case studies and implications. *Journal of Molecular Biology* 235:1–12.

Waterman, M. S. 1995. *Introduction to Computational Biology*. Chapman & Hall.

Waterman, M. S. and Eggert, M. 1987. A new algorithm for best subsequence alignments with application to tRNA–rRNA comparisons. *Journal of Molecular Biology* 197:723–725.

Waterman, M. S. and Perlwitz, M. D. 1984. Line geometries for sequence comparisons. *Bulletin of Mathematical Biology* 46:567–577.

Watson, J. D., Hopkins, N. H., Roberts, J. W., Steitz, J. A. and Weiner, A. M. 1987. *Molecular Biology of the Gene*. Benjamin/Cummings.

Wilmanns, M. and Eisenberg, D. 1993. Three-dimensional profiles from residue-pair preferences: identification of sequences with beta/alpha-barrel fold. *Proceedings of the National Academy of Sciences of the USA* 90:1379–1383.

Witherell, G. W., Gott, J. M. and Uhlenbeck, O. C. 1991. Specific interaction between RNA phage coat proteins and RNA. *Progress in Nucleic Acid Research and Molecular Biology* 40:185–220.

Woese, C. R. and Pace, N. R. 1993. Probing RNA structure, function, and history by comparative analysis. In Gesteland, R. F. and Atkins, J. F., eds., *The RNA World*. Cold Spring Harbor Laboratory Press. pp. 91–117.

Wray, G. A., Levinto, J. S. and Shapiro, L. H. 1996. Molecular evidence for deep precambrian divergences among metazoan phyla. *Science* 274:568–573.

Wu, S. and Manber, U. 1992. Fast text searching allowing errors. *Communications of the ACM* 35:83–90.

Yada, T. and Hirosawa, M. 1996. Detection of short protein coding regions within the Cyanobacterium genome: application of the hidden Markov model. *DNA Research* 3:355–361.

Yada, T., Sazuka, T. and Hirosawa, M. 1997. Analysis of sequence patterns surrounding the translation initiation sites on Cyanobacterium genome using the hidden Markov model. *DNA Research* 4:1–7.

Yang, Z. 1993. Maximum-likelihood estimation of phylogeny from DNA sequences when substitution rates differ over sites. *Molecular Biology and Evolution* 10:1396–1401.

Yang, Z. 1994. Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: approximate methods. *Journal of Molecular Evolution* 39:306–314.

Zuckerkandel, E. and Pauling, L. 1962. Molecular disease, evolution and genetic heterogeneity. In Marsha, M. and Pullman, B., eds., *Horizons in Biochemistry*. Academic Press. pp. 189–225.

Zuker, M. 1989a. Computer prediction of RNA structure. *Methods in Enzymology* 180:262–288.

Zuker, M. 1989b. On finding all suboptimal foldings of an RNA molecule. *Science* 244:48–52.

Zuker, M. 1991. Suboptimal sequence alignment in molecular biology: alignment with error analysis. *Journal of Molecular Biology* 221:403–420.

Zuker, M. and Stiegler, P. 1981. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Research* 9:133–148.

部分术语汉英对照

Γ 函数 gamma function
β 分布 beta distribution
β 折叠 beta sheet
Gibbs采样 Gibbs sampling
Jacobi行列式 Jacobian
Markov链 Markov Chain
Occam剃刀 Occam's razor
X染色体 X chromosome
氨基酸 amino acid
胞嘧啶 cytosine
贝叶斯法则 Bayes' rule
本征值 eigenvalue
比特 bit
边际概率 marginal probability
标准差 standard deviation
参数估计 parameter estimation
侧翼模型 flanking model
插入删除 indel
插入状态 insert state
产生式 production
超度规条件 ultrametric condition
抽样 sampling
串联拷贝 tandem copy
大O记号 Big-O notation
待查序列 query sequence
等高线图 contour plot
等同状态 identical state
颠换 transversion
迭代精调方法 Iterative refinement method
动态规划 dynamic programming
豆血红蛋白 leghaemoglobin
端素蛋白结构 telokin structure
堆积 stacking
对称建议机制 symmetric proposal mechanism
对称性 symmetry
对角化 diagonalise
对数变换 log transformation
对数几率比 log odds-ratio
对数似然 log likelihood
多分支环 multi-branched loops
多项分布 multinomial distribution
多序列联配 multiple alignment
二分支的 bifurcated
二级结构 secondary structure
二项分布 binomial distribution
二序列联配 pairwise alignment
发夹环 hairpin loop
发射概率 emission probability
范式 normal form
方差 variance
仿射分值 affine score
仿射空位罚分 affine gap penalty
非齐次Markov链 inhomogeneous Markov chain
非确定性的 nondeterministic
非限制性文法 unrestricted grammars
非终端字符 nonterminal
分而治之 divide and conquer
分歧 divergence
分析树 parse tree
分支定界 branch and bound
分支节点 branch node
分支型 branch pattern
分值矩阵 score matrix
分子钟 molecular clock
负二项分布 negative binomial distribution
改写规则 rewriting rule
概率分布 probability distribution
概率密度 probability density
高斯分布 Gaussian distribution
根节点 root node
共轭分布 conjugate distributions
共轭梯度 conjugate gradient
广义EM generalised EM
归一化 normalise
恒等矩阵 identity matrix
后序遍历 post-order traversal
后验分布 a posteriori distribution
后验概率 posterior probability
后验概率分布 posterior probability distribution
后验混合系数 posterior mixture coefficient

后验解码 posterior decodeing
后验均值估计 posterior mean esitmator(PME)
互信息 mutual information
环 loop
回溯 traceback
回溯指针 back-pointer
回文语言 palindrome language
混合先验 mixture prior
激酶 kinase
极值分布 extreme value distribution
极值分布 extreme value distribution(EVD)
假扭结 pseudoknot
简约法 parsimony
剪枝算法 pruning algorithm
渐进联配 progressive alignment
渐近分布 asymptotic distribution
建议分布 proposal distribution
交迭 overlap
交迭性匹配 overlap match
交联 crossing-over
接受机制 acceptance mechanism
茎 stem
茎环 stem loop
局部联配 local alignment
局部最优 local maxima
聚类算法 clustering algorithm
拒绝采样 rejection sampling
均布先验 uniform prior
均匀密度 uniform density
均匀性 uniformity
均值 mean
开放阅读框 open reading frame(ORF)
拷贝语言 copy language
可加性 additivity
可逆性 reversibility
空边 dummy edge
空间复杂度 memory complexity
空位 gap
空位罚分 gap penalty
跨膜的 transmembrane
累积高斯映射 cumulative Gaussian map
联合概率 joint probability
列型HMM profile HMM
列型 profile
邻接聚类算法 neighbour-joining clustering algorithm
零模型 null model
灵敏度 sensitivity
门 phylum/phyla
模拟退火 simulated annealing
模体 motif
木村距离 Kimura distance
内部节点 internal node
内部外部训练 inside-outside training
内环 interior loop
能量最小化折叠算法 energy minimisation folding algorithm
判别 discrimination
判别性估计 discriminative estimation
旁系同源 paralogue
配对和 sum of pairs(SP)
配分函数 partition function
匹配状态 match state
平稳性 stationarity
期望最大化 expectation maximisation(EM)
启发式规则 heuristic rule
启发式联配算法 heuristic alignment algorithm
前向算法 forward algorithm
权重矩阵 weight matrices
全局标度因子 global scaling factor
全局联配 global alignment
全局匹配 global match
缺失数据 missing data
确定性的 deterministic
弱大数定理 weak law of large numbers
三级结构 tertiary structure
三联密码子 coden triplet
删除状态 deletion
上下文无关文法 context-free grammars
上下文有关文法 context-sensitive grammar
生成文法 generative grammars
失配 mismatch
时间复杂度 time complexity
疏水的 hydrophobic
水平转移 horizontal transfer
斯特林公式 Stirling's formula
四点条件 four-point condition
溯祖 coalesce
随机模型 random model
随机上下文无关文法 stochastic context-free grammars(SCFG)
随机突变 random mutation
随机正则文法 stochastic regular grammars
梯度下降 gradient descent
梯度下降法 gradient descent method
替代概率 substitution probability
替换矩阵 subsitution matrix
条件概率 conditional probability
条件最大似然 conditional maximum likelihood(CML)
同源性 homology
凸(环) bulge(loop)

图灵自动机 Turing automaton
推断 infer,inference
外群 outgroup
伪计数 pseudocount
伪随机数生成器 pseudo-random number generator
未分支的 unbranched
位置特异分值矩阵 position specific score matrix
无根树 unrooted tree
系统发育树 phylogenetic tree
细胞内 intracellular
细胞色素C cytochrome C
细胞外 extracellular
细致平衡 detailed balance
下推栈 push-down stack
下推自动机 push-down automaton
先验分布 prior distribution
先验概率 prior probability
限速步骤 limiting step
线性分值 linear score
线性空间联配 linear space alignment
线性有界自动机 linear bounded automaton
相对似然性 relatvie likelihood
协方差模型 covariance model(CM)
信息量 information content
信息论 information theory
形式语言理论 formal language theory
序列等同度 sequence identity
序列空间 sequence space
序列图 sequence graph
叶节点 leaf
一致结构预测 consensus structure prediction
遗传漂变 genetic drift
隐马模型 hidden Markov model
有限状态自动机 finite state automaton(FSA)
原核基因 prokaryotic gene
正规化子 regulariser
帧 frame
枝长 edge length
直系同源 orthologue
指导树 guide tree
置换 transition
置信估计 confidence estimate
置信区间 confidence interval
终端字符 terminal
重复性匹配 repeated match
重组事件 recombinational event
转换文法 transformational grammars
状态转移 state transition
子树 subtree
子序列 subsequence
自动机 automaton
自然选择 natural selection
自循环模型 self-looping model
自展法 bootstrap
字母表 alphabet
组合爆炸 combinatorial explosion
最大后验 maximum a posteriori(MAP)
最大互信息 maximum mutual information (MMI)
最大似然 maximum likelihood
最大似然估计 maximum likelihood estimate
最大熵 maximum entropy
最小二乘匹配 least-square match
最小描述长度 minimum description length
最优联配 optimal alignment
坐标变换 coordinate transformation
熵 entropy
蝾螈 axolotl

部分术语英汉对照

Baum-Welch re-estimation equation
Baum-Welch重估计方程
Bayes' rule 贝叶斯法则
Bayesian model 贝叶斯模型
Big-O notation 大O记号
Chomsky hierarchy Chomsky层次结构
CpG island CpG岛
Dirichlet distribution Dirichlet分布
Dirichlet mixture Dirichlet混合分布
Dirichlet prior distribution Dirichlet先验分布
EM algorithm EM算法
Gaussian distribution 高斯分布
Gibbs sampling Gibbs采样
HMM 隐马模型
Iterative refinement method 迭代精调方法
Jacobian Jacobi行列式
Kimura distance 木村距离
Monte Carlo method Monte Carlo方法
Occam's razor Occam剃刀
Shannon entropy Shannon熵
Stirling's formula 斯特林公式
Turing automaton 图灵自动机
UPGMA 非加权对组算术平均法
X chromosome X染色体
a posteriori distribution 后验分布
acceptance mechanism 接受机制
additivity 可加性
affine gap penalty 仿射空位罚分
affine score 仿射分值
alpha helices α 螺旋
alphabet 字母表
amino acid 氨基酸
architecture 架构
asymptotic distribution 渐近分布
automaton 自动机
axolotl 蝾螈
beta distribution β 分布
beta sheet β 折叠
bifurcated 二分支的
binomial distribution 二项分布
bit 比特
block 区块
bootstrap 自展法
branch and bound 分支定界
bulge(loop) 凸(环)
clustering algorithm 聚类算法
coalesce 溯祖
coden triplet 三联密码子
combinatorial explosion 组合爆炸
conditional maximum likelihood(CML) 条件最大似然
conditional probability 条件概率
confidence estimate 置信估计
confidence interval 置信区间
conjugate distributions 共轭分布
conjugate gradient 共轭梯度
consensus sequence 代表序列
consensus structure prediction 一致结构预测
context-free grammars 上下文无关文法
context-sensitive grammar 上下文有关文法
contour plot 等高线图
coordinate transformation 坐标变换
copy language 拷贝语言
correction factor 修正因子
covariance model(CM) 协方差模型
crossing-over 交联
cumulative Gaussian map 累积高斯映射
cytochrome C 细胞色素C
cytosine 胞嘧啶
deletion 删除
deletion 删除状态
detailed balance 细致平衡
deterministic 确定性的
diagonalise 对角化
discriminative estimation 判别性估计
divide and conquer 分而治之
dummy edge 空边
duration modelling 时长建模
dynamic programming 动态规划
edit distance 编辑距离
eigenvalue 本征值
emission probability 发射概率

energy minimisation folding algorithm 能量最小化折叠算法
entropy 熵
eukaryotic promoter 真核启动子
expectation maximisation(EM) 期望最大化
extracellular 细胞外
extreme value distribution(EVD) 极值分布
extreme value distribution 极值分布
finite state automaton(FSA) 有限状态自动机
flanking model 侧翼模型
formal language theory 形式语言理论
forward algorithm 前向算法
four-point condition 四点条件
frame 帧
gamma function Γ 函数
gap penalty 空位罚分
gap-extension 空位延伸
gap-open 空位开启
gap 空位
generative grammars 生成文法
genetic drift 遗传漂变
global alignment 全局联配
global match 全局匹配
global scaling factor 全局标度因子
golbin family 球蛋白家族
gradient descent method 梯度下降法
gradient descent 梯度下降
guide tree 指导树
hairpin loop 发夹环
heuristic alignment algorithm 启发式联配算法
heuristic rule 启发式规则
hit extension 击点拓展
homology 同源性
horizontal transfer 水平转移
hybrid match condition 混合匹配条件
hydrophobic 疏水的
identical state 等同状态
identity matrix 恒等矩阵
indel 插入删除
information content 信息量
information theory 信息论
inhomogeneous Markov chain 非齐次Markov链
insert state 插入状态
insertion 插入
inside-outside training 内部外部训练
interior loop 内环
internal node 内部节点
intracellular 细胞内
joint probability 联合概率
kinase 激酶
least-square match 最小二乘匹配
leghaemoglobin 豆血红蛋白
limiting step 限速步骤
linear bounded automaton 线性有界自动机
linear score 线性分值
linear space alignment 线性空间联配
local alignment 局部联配
local maxima 局部最优
log likelihood 对数似然
log odds-ratio 对数几率比
logistic function 逻辑斯蒂函数
long-range correlation 长程关联
loop region 弯折区域
loop 环
marginal probability 边际概率
match state 匹配状态
maximum a posteriori(MAP) 最大后验
maximum entropy 最大熵
maximum likelihood estimate 最大似然估计
maximum likelihood 最大似然
maximum mutual information (MMI) 最大互信息
mean 均值
memory complexity 空间复杂度
methyl-C 甲基化胞嘧啶
minimum description length 最小描述长度
minimum entropy 最小熵
mismatch 失配
missing data 缺失数据
mixture prior 混合先验
molecular clock 分子钟
motif 模体
multi-branched loops 多分支环
multinomial distribution 多项分布
multiple alignment 多序列联配
multiplicative factor 倍乘因子
mutual information 互信息
natural selection 自然选择
negative binomial distribution 负二项分布
neighbour-joining clustering algorithm 邻接聚类算法
neighbour-joining 邻接法
nondeterministic 非确定性的
nonterminal 非终端字符
normal form 范式
normalise 归一化
null model 零模型
null state 空状态
numerical stability 数值稳定性

optimal alignment 最优联配
orthologue 直系同源
outgroup 外群
overlap match 交迭性匹配
overlap 交迭
pairwise alignment 二序列联配
palindrome language 回文语言
paralogue 旁系同源
parameter estimation 参数估计
parse tree 分析树
parse 分析
parsimony 简约法
partition function 配分函数
phylogenetic tree 系统发育树
phylum/phyla 门
point accepted mutation 可接受点突变
position specific score matrix 位置特异分值矩阵
position-specific score 位置特异分值
post-order traversal 后序遍历
posterior decodeing 后验解码
posterior mean esitmator(PME) 后验均值估计
posterior mixture coefficient 后验混合系数
posterior probability distribution 后验概率分布
posterior probability 后验概率
primary sequence model 一级序列模型
prior distribution 先验分布
prior probability 先验概率
probability density 概率密度
probability distribution 概率分布
production 产生式
profile HMM 列型HMM
profile HMM 列型隐马模型
profile 列型
progressive alignment 渐进联配
prokaryotic gene 原核基因
proposal distribution 建议分布
pruning algorithm 剪枝算法
pseudo-random number generator 伪随机数生成器
pseudocount 伪计数
pseudoknot 假扭结
push-down automaton 下推自动机
push-down stack 下推栈
query sequence 待查序列
random mutation 随机突变
recombinational event 重组事件
regulariser 正规化子
rejection sampling 拒绝采样
relatvie likelihood 相对似然性
repeated match 重复性匹配
reversibility 可逆性
rewriting rule 改写规则
root ancestral sequence 根祖先序列
root node 根节点
sampling 抽样
scale paramter 标度参数
score matrix 分值矩阵
scoring model 计分模型
secondary structure 二级结构
self-looping model 自循环模型
sensitivity 灵敏度
sequence graph 序列图
sequence identity 序列等同度
sequence space 序列空间
significance of score 分值的显著性
silent state 沉默状态
simulated annealing 模拟退火
stacking 堆积
standard deviation 标准差
state transition 状态转移
stationarity 平稳性
stem loop 茎环
stem 茎
stochastic context-free grammars(SCFG) 随机上下文无关文法
stochastic regular grammars 随机正则文法
subsequence 子序列
subsitution matrix 替换矩阵
substitution probability 替代概率
substitution 替换
subtree 子树
switch point 切换点
symmetric proposal mechanism 对称建议机制
symmetry 对称性
tandem copy 串联拷贝
telokin structure 端素蛋白结构
terminal 终端字符
tertiary structure 三级结构
time complexity 时间复杂度
traceback 回溯
transformational grammars 转换文法
transition 置换
transmembrane 跨膜的
transversion 颠换
ultrametric condition 超度规条件
unbranched 未分支的
underflow error 下溢错误
uniform density 均匀密度
uniform prior 均布先验
uniformity 均匀性
unrestricted grammars 非限制性文法
variance 方差
weak law of large numbers 弱大数定理
weight matrices 权重矩阵

作者索引

正体数字标注正文页码，斜体标注参考文献页码

主题索引