

前言

译自“An Introduction to Conditional Random Fields” --Charles Sutton, Andrew McCallum。我也在学习之中，必有错漏之处，希望能依靠大家的力量，共同进步。

目前数学公式显示总出问题，很多更新在gitbook上编译不通过。可以到github上下载pdf和all.md：
https://github.com/cottageLamp/CRFIntroduction_Chinese

总体感觉原文不是很易懂，翻译之后也不好理解。好比一本介绍“降龙十八掌”的入门书，却时不时要求读者参考一下“九阴白骨爪”、“凌波微步”、“易筋经”……，岂不要命？

争取在翻译完成后，写一篇条理清晰的总结在后面，包括一些原文中没有的内容。

摘要

许多任务要对大量的变量进行预测。这些变量相互关联，且依赖于另外的已被观测量。结构化预测方法实质上是分类器与图模型的结合。图模型能够紧凑地对多变量数据建模，而分类器能够利用大规模的输入特征完成预测。本文描述了条件随机场，一种流行的、用于结构化预测的概率方法。CRFs已在广泛的领域中获得大量应用，包括自然语言处理，机器视觉以及生物信息学。我们将描述CRFs的推断方法和训练方法，包括在实现大规模CRFs时的问题。不要求读者具有图模型的知识，希望能对广大的实践者们有用。

1介绍

对很多应用来说，至关重要的是预测互相关多变量的能力。这些应用广泛分布于图片分割及分类、围棋胜负概率的预测、在DNA序列中分离基因组，以及对自然文本进行语法分割。在这些应用中，我们想基于一组观测值 \mathbf{x} ，来预测一个随机输出向量 $\mathbf{y} = y_0, y_1, \dots, y_T$ 。一个相对简单的例子是对自然语言进行词性标注。其中，每个 y_s 对应着 s 位置的单词的词性，而输入 \mathbf{x} 被分解成多个输入特征向量 $\{\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_T\}$ 。每个 \mathbf{x}_s 包含着 s 位置单词的多种信息，如它自身、它的前后缀、它在词典中的身份，以及来自语义数据库的信息（如WordNet）。（专业词汇有问题）

一种办法是为每个位置 s 训练位置无关的分类器 $\mathbf{x} \rightarrow y_s$ ，尤其是当我们要最大化 y_s 的正确率时。然而，困难在于输入变量 y_s 之间存在复杂的依赖性。如在英语中，形容词不常接名词。又如在计算机视觉中，临近区域趋向属于相近的类。另一个难点在于，输出变量常常表现出一种复杂的结构，如语法树。那么，在树的顶端附近选择怎样的语法规则会对整个树有极大的影响。

图模型是一种表达互相关变量的自然的方法。图模型包括：贝叶斯网络，神经网络，因子图，马尔科夫随机场，伊辛模型（Ising model）等等。它们把一个复杂的概率分布分解成许多局部因子(factor)相乘，而这些因子各自对应着变量的一部分。我们有可能描述，按照一组条件独立关系对概率密度进行的分解，能在多大程度上满足着该分布。这种对应关系，使得建模更加容易，因为我们的经验知识常常提供了合理的条件独立假设，而这决定了我们如何进行分解。

关于图模型的工作，特别是自然语言处理相关的，大量地关注了生成模型(generative models)。生成模型显式地建立对所有输入和输出的联合分布 $p(\mathbf{y}, \mathbf{x})$ 。尽管这有一些好处，但存在着重要的局限。不仅是因为输入 \mathbf{x} 的维度可能非常大，还因为输入 \mathbf{x} 内在的复杂的相关性。对它们进行建模是困难的。对输入的相关性进行建模，会导致难以驾驭的模型，而忽略它们却会降低系统的性能。

一种解决办法是**判别**方法，正如在逻辑回归分类器中的做法。这里，我们直接对 $p(\mathbf{y}|\mathbf{x})$ 建模，因为这是完成分类所需的全部。这正是条件随机场（CRFs）所采用的方法。CRFs结合了判别分类器与图模型的优点。一方面能够紧凑地对多变量输出 \mathbf{y} 进行建模，一方面能够应付数量庞大的输入特征 \mathbf{x} ，以用于预测。条件模型的优势在于，它忽略了那些仅仅存在于 \mathbf{x} 内在变量之间的相关性。因此，条件模型要比联合模型具有简单得多的结构。生成模型和CRFs之间的差别，正如朴素贝叶斯分类器与逻辑回归分类器之间的差别。实质上，多元逻辑回归模型可以被看成一种最简单的CRF，因为它只有一个输出。

本文描述了CRFs的建模、推断（前向计算）和参数估计方法。读者不用具有图模型的知识，因而本文希望能对广大的实践者有用。我们从介绍CRFs建模的一些问题开始（第二章），包括线性CRFs通用结构的CRFs，以及包含隐藏变量的隐CRFs（hidden crfs）。我们将说明，为何CRFs既是著名的逻辑回归的扩展，有是判别式的隐马尔科夫模型。

在接下来的两章，我们描述了推断（第4章）和学习（第5章）。**推断**既指计算 $p(\mathbf{y}|\mathbf{x})$ 的边缘分布，也指计算极大似然 $\mathbf{y}^* = \operatorname{argmax}_{\mathbf{y}} p(\mathbf{y}|\mathbf{x})$ 。学习是指参数估计过程，就是找到 $p(\mathbf{y}|\mathbf{x})$ 的参数，使其最大限度地符合一组训练样本 $\{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^N$ 。推断和学习过程往往密切地组合在一起，因为学习过程需要推断做为子过程。

最后，我们讨论了CRFs与其他类模型的关系，包括结构化预测模型，神经网络和最大熵马尔科夫模型（第6章）。

1.1动手方面的细节

本文努力指出动手实现方面的细节，而这常常被学术文献所忽略。例如，我们讨论了特征工程（第2.5节），在推断中避免数值溢出（第4.3节），CRF在一些基准问题上训练时的伸缩性。

因为这是我们关于实现细节的第一个章节，应该提一提可供使用的一些CRFs平台。在写作本文时，一些流行的平台包括：

CRF++	http://crfpp.sourceforge.net/
MALLET	http://mallet.cs.umass.edu/
GRMM	http://mallet.cs.umass.edu/grmm/
CRFSuite	http://www.chokkan.org/software/crfs
FACTORIE	http://www.factorie.cc

除此之外，用于马尔科夫逻辑网络的软件（如Alchemy: <http://alchemy.cs.washington.edu/>）也可用于构建CRF模型。据我们所知，Alchemy, GRMM 和 FACTORIE 是仅有的、能够处理任意的图模型的工具。

2 建模

本章，我们从建模的角度来描述CRFs，阐述了CRF是如何把机构化的输出表示成高维输入向量的分布。可以把CRFs理解成，将逻辑回归分类器扩展到任意的图模型，也可以被理解成生成模型（如隐马尔科夫模型）的判别对应物。

译注：判别和生成模型是两种在理论上等价（可互相推导得到对方），但建模思路相反的模式。

我们从对图模型的简单介绍（第2.1节），以及对NLP中的生成和判别模型的介绍（第2.2节）开始。然后，我们可以给出了CRF的正式定义，包括常用的线性链（linear chains）（第2.3节），以及通用图结构（第2.4节）。因为CRF的准确性严重依赖于所使用的特征，我们也描述了特征工程常用的一些技巧（第2.5节）。最后，我们提供两个CRF应用的例子（第2.6节），以及一个宽泛的、关于CRFs应用领域的报告。

2.1 图模型

图模型是表达和推断多元概率分布的强大框架。它已经在统计模型的许多领域被证明有用，包括编码理论（coding theory），计算机视觉，知识表达（knowledge representation），贝叶斯统计（Bayesian statistics），以及自然语言处理（广告语也太多了吧）。

直接描述包含许多变量的分布，其代价是昂贵的。假如我们用表（table）来描述 n 个二值变量的联合分布，需要 $O(2^n)$ 个浮点数（建议读者理解一下：每个变量有2种可能的取值，而总共有 n 个变量，那么总共有 2^n 种可能的取值。它这里的意思是：给每种取值赋予一个浮点数，表示其概率）。从图模型的角度看，认为一个分布尽管建立在许多变量之上，但常常可以表示成一些局部方程（local functions）的乘积，而这些方程只依赖于少量的变量。这种分解实际上与变量间的某些条件独立性密切相关——两种信息被轻易地用途来概括。实质上，分解、条件独立与图的结构，这三者构成了图模型框架力量的来源：条件独立性视角主要用于设计模型，而分解视角主要用于设计推断算法。

在本节的余下部分，我们从以上两个视角来介绍图模型，关注那些建立在无向图（undirected graphs）之上的模型。关于更详细、更现代的图模型及其推断算法，可参考Koller 和 Friedman 【57】的教材。

2.1.1 无向图

我们考虑随机变量集合 Y 上的概率分布。我们通过整数 $s \in 1, 2, \dots, |Y|$ 来对变量进行索引。每个变量 $Y_s \in Y$ 的取值范围都是集合 \mathcal{Y} 。本文我们只考虑离散的 \mathcal{Y} ，尽管它也可以是连续的。 Y 的一次特定的取值记做 \mathbf{y}_s 。对于 Y 中的特定变量 Y_s ， \mathbf{y}_s 包含了对它的赋值，记做 y_s 。记号 $\mathbf{1}_{\{y=y'\}}$ 表示一个函数，在 $y = y'$ 时取1，而在其他时候取0。我们还需要边缘分布的记号。对于某个固定的取值 y_s ，我们用求和符号 $\sum_{\mathbf{y} \setminus y_s}$ 来表示：在 \mathbf{y} 的全部取值中，那些 $Y_s = y_s$ 的取值的概率的和。

假定，我们相信一个概率分布 p 可以表示成一组因子，记做 $\Psi(\mathbf{y}_a)$ 的连乘。其中， a 是一个整数索引（下标），从1变化到 A ，而 A 就是因子的个数。每个因子 $\Psi(\mathbf{y}_a)$ 只依赖于部分变量 $Y_a \in Y$ 。 $\Psi(\mathbf{y}_a)$ 是一个非负数，可以被看成 \mathbf{y}_a 的自洽性的度量。自洽性高的取值，其发生的概率就高。这种分解让我们更高效地表示分布 p ，因为集合 Y_a 要比完整的集合 Y 小得多。

一个无向图模型是这样一种概率分布，它根据一组给定的因子来分解模型。正式地，给定 Y 的子集 $\{Y_a\}_{a=1}^A$ 的集合，一个无向图模型是所有可以写成下式的分布：

$$p(\mathbf{y}) = \frac{1}{Z} \prod_{a=1}^A \Psi(\mathbf{y}_a) \quad (2.1)$$

其中，对于任意的因子 $\mathcal{F} = \{\Psi(\mathbf{y}_a)\}$ ，及其对应的所有可能的 \mathbf{y}_a ，都有 $\Psi(\mathbf{y}_a) \geq 0$ 。（这些因子又被称作**局部函数**或**自洽性函数**。）我们将用**随机场**来表示由某个无向图定义的特定分布。常数 Z 是一个归一化因子，保证分布 p 的和为1。它定义如下：

$$Z = \sum_{\mathbf{y}} \prod_{a=1}^A \Psi(\mathbf{y}_a). \quad (2.2)$$

Z 的值，考虑成因子集合 \mathcal{F} 的函数的话，也被称作**配分函数（partition function）**。注意，式(2.2)中的求和，需要在爆炸式的 \mathbf{y} 的所有可能取值上进行。因此，计算 Z 通常是不可行的，但是有很多关于估计它的研究（见第4章）。

术语“图模型”的来由，在于式 (2.1) 所表示的因子分解，可以紧凑地表示成一张图。**因子图【58】**提供了一个特别自然的构图方法。一个因子图是一个两两连接图 $G = (V, F, E)$ 。其中，节点的集合 $V = \{1, 2, \dots, |Y|\}$ 索引了模型中的全部随机变量，另一组节点的集合 $F = \{1, 2, \dots, A\}$ 索引了所有的因子。对图的理解是：如果一个变量节点 s 连接到一个因子节点 a ，那么在模型中，变量 Y_s 就是因子 Ψ_a 的一个参数。所以，因子图直接描述了，一个分布是如何被分解成多个局部函数的乘积的。

我们正式地定义——一个因子图是否“描述”了一个分布？记 $N(a)$ 包含了所有连接到因子节点 a 上的变量节点，那么：

定义2.1 仅当存在一组局部方程 $\Psi(y_a)$ ，使得 p 可以写成：

$$p(\mathbf{y}) = Z^{-1} \prod_{a \in F} \Psi(\mathbf{y}_{N(a)}) \quad (2.3)$$

时，一个分布 $p(\mathbf{y})$ 根据因子图 G 分解了。

一组子集描述了无向模型，而一个因子图同样如此。在式 (2.1) 中，取子集为节点的邻居 $\{Y_N(a) | \forall a \in F\}$ 。根据式 (2.1) 定义无向图模型，对应着所有根据 G 进行分解所得的分布。

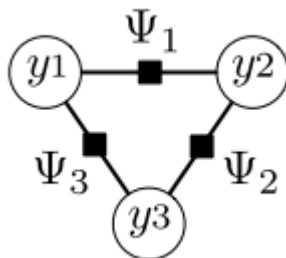


图2.1 带3个变量的因子图

图2.1展示了一个带有3个随机变量的因子图，图中，圆圈是变量节点，而灰色方块是因子节点。我们根据节点的索引进行了标注。这个因子图能够描述所有的带3个变量的分布，前提是对于任意的 $\mathbf{y} = (y_1, y_2, y_3)$ ，该分布能够写成 $p(y_1, y_2, y_3) = \Psi_1(y_1, y_2) \Psi_2(y_2, y_3) \Psi_3(y_1, y_3)$ 的形式。

图模型的因子分解与变量间（在其取值范围里）的条件独立性密切相关。这种联系可通过另一种无向图来理解——马尔科夫网。它直接描述了多元分布的条件独立关系。马尔科夫网只是随机变量的图，不包括因子。现记 G 为整数序列 $V = \{1, 2, \dots, |Y|\}$ 上的无向图，而 V 仍是随机变量的索引。对于某一个索引 s ，记 $N(s)$ 为它的邻居。那么我们称 p 是关于 G 的马尔科夫网，仅当它满足局部的马尔科夫特性：对于任意的两个变量 $Y_s, Y_t \in Y$ ， Y_s 关于它的邻居独立于 Y_t 。

把所有连接到同一个因子的变量都两两连接起来，可将如式(2.1)的分布，变成其对应的马尔科夫网。这很显然，因为由式 (2.1) 而来的条件分布 $p(y_s | \mathbf{y}_{N(s)})$ 仅仅是那些马尔科夫毯中的变量的函数。

从因子分解的角度看，马尔科夫网存在着不好的歧义性。考虑图2.2（左）的3变量马尔科夫网。任何按照 $p(y_1, y_2, y_3) \propto f(y_1, y_2, y_3)$ 分解的分布，都可能与它对应。然而，我们希望使用更严格的参数化—— $p(y_1, y_2, y_3) = f(y_1, y_2)g(y_2, y_3)h(y_1, y_3)$ 。后面这组模型簇是前面的严格子集，且需要更少的数据来获得准确的分布估计译注：参数估计？。然而，马尔科夫网不能区分这两种参数化。相反，因子图无歧义地描述了模型的因子分解。

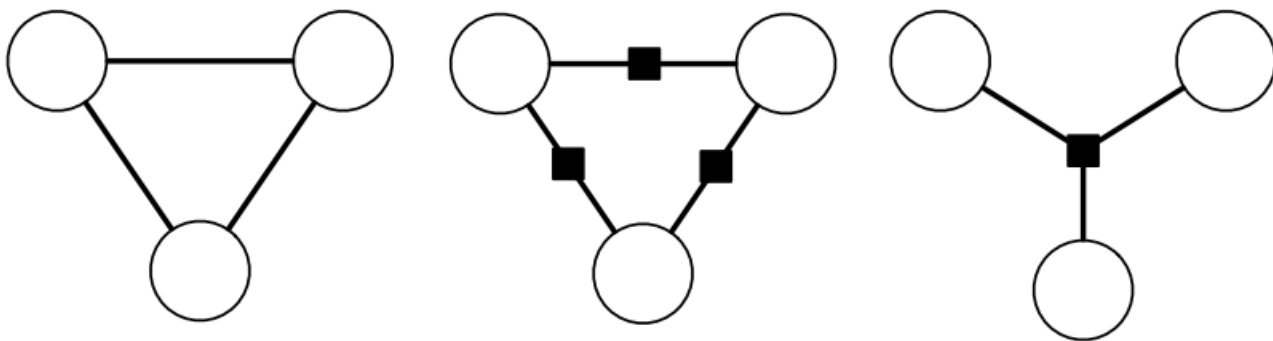


图2.2带有歧义的马尔科夫网（左）。右边的两种分解都有可能与左图对应。

2.1.2 有向图

无向模型中的局部函数无需带有方向性的概率表达，有向图模型却把分布分解成局部的条件概率分布。记 G 为有向无环图， $\pi(s)$ 为 Y_s 的所有父节点的序号集合。一个有向图模型是一簇按照如下分解的分布：

$$p(\mathbf{y}) = \prod_{s=1}^S p(y_s | \mathbf{y}_{\pi(s)}). \quad (2.4)$$

我们称 $p(y_s | \mathbf{y}_{\pi(s)})$ 为**局部条件分布 (local conditional distributions)**。注意，对于没有父节点的变量， $\pi(s)$ 可以是空的。这时， $p(y_s | \mathbf{y}_{\pi(s)})$ 可被理解为 $p(y_s)$ 。可以推断 p 是合理归一化的。可以这样来理解有向模型——其每个因子都在局部完成了特殊的归一化，使得（1）因子相当于局部变量上的条件分布，且（2）归一化常数 $Z = 1$ 。有向模型常常用于生成模型，我们将在第2.2.3节讲述这一点。有向模型的一个例子是贝叶斯模型（2.7），被描述在图2.3（左）了。在这些图中，灰节点表示了某些数据集上观测的变量。贯穿本文，我们都将采用这一习惯。

2.2 生成与判别模型

本节我们探讨几个已被用于自然语言处理的简单图模型。虽然它们已被熟知，但它们一方面可以澄清前文提到的诸多概念，另一方面也可以说明某些今后讨论CRFs时会遇到的议题。我们尤其关注隐马尔科夫模型（HMM），因为它与线性链条件随机场密切相关。

本节的主要目的是对比生成与判别模型。将会提到的模型，包括两个生成模型（朴素贝叶斯和HMM），一个判别模型（逻辑回归模型）。**生成模型**描述了，一个输出向量 \mathbf{y} 以怎样的概率“生成”输入特征 \mathbf{x} 。**判别模型**从相反的方向工作，直接描述了如何利用输入特征 \mathbf{x} 来给输出 \mathbf{y} 赋值。一般来说，这两者可根根据贝叶斯法则互相转化。但在实践中却相去甚远，各自隐藏着一些优点（将在2.2.3节讲述）。

2.2.1 分类

我们首先讨论**分类**问题——根据给定的一个向量 $\mathbf{x} = (x_1, x_2, \dots, x_K)$ ，来预测单一的 \mathbf{y} 变量的离散值（类别标签）。一个简单的方法是，假定当类别标签已知时，所有的特征是独立的。结果是所谓的朴素贝叶斯分类器。它基于如下的联合概率模型：

$$p(\mathbf{y}, \mathbf{x}) = p(\mathbf{y}) \prod_{k=1}^K p(x_k | y). \quad (2.7)$$

这个模型可以描述为图2.3（左）的有向模型。为每个特征 x_k 定义因子 $\Psi(y) = p(y)$ ，以及因子 $\Psi_k(y, x_k) = p(x_k|y)$ ，我们也可以写成因子图。这样的因子图如图2.3（右）所示。

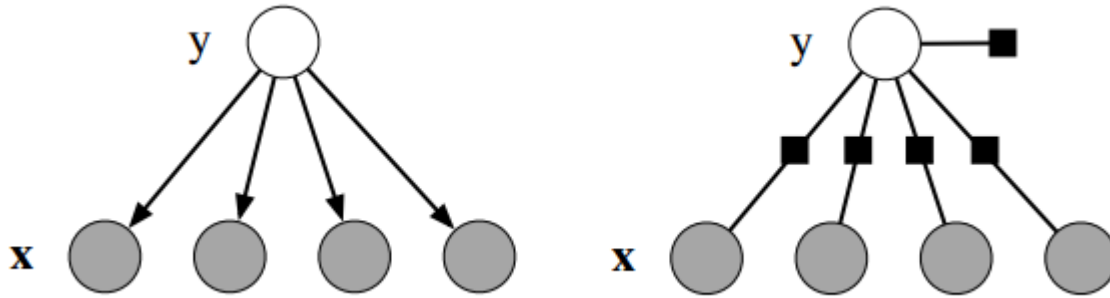


图2.3 朴素贝叶斯分类器，被当成有向模型（左），或因子图（右）

逻辑回归（有时在NLP圈子里叫做**最大熵分类器**）是另一个知名的，且很自然地表达为图模型的分类器。该分类器源于将每个类的逻辑概率， $\log p(y|\mathbf{x})$ ，假设为 \mathbf{x} 的线性函数，以及一个归一化常数。这导致了如下的条件概率：

$$p(y|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\{\theta_y + \sum_{j=1}^K \theta_{y,j} x_j\}, \quad (2.8)$$

其中 $Z(\mathbf{x}) = \sum_y \exp\{\theta_y + \sum_{j=1}^K \theta_{y,j} x_j\}$ ，是归一化常数。而 θ_y 是偏置量，相当于朴素贝叶斯里面的 $\log p(y)$ 。与其像式（2.8）那样为每一个类制定一个权重向量，我们不如采用被所有类共享的一组权重的记号。这一技巧通过定义一组**特征函数(feature functions)**来实现，而这些特征只对某一类时非零。为了达到这个目的，特征权重的特征函数被定义为 $f_{y,j}(y, \mathbf{x}) = \mathbf{1}_{\{y=y\}} x_j$ ，而把偏置权重的特征函数定义为 $f_y(y, \mathbf{x}) = \mathbf{1}_{\{y=y\}}$ 。现在我们可以用 f_k 来遍历每个特征函数 $f_{y,j}$ ，用 θ_k 来索引对应的权重 $\theta_{y,j}$ 。利用这一符号技巧，逻辑回归模型变成了：

$$p(y|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\{\sum_{k=1}^K \theta_k f_k(y, \mathbf{x})\}. \quad (2.9)$$

我们之所以引入这样的记号，是因为它简化了下文介绍CRFs时的记号。译注：（2.8）中的 θ_y 好像丢失了？

2.2.2 序列模型

分类器只对单一变量做预测，但图模型的真正用处在于对大量互相关变量的建模能力。本节，我们讨论了可能是最简单的相关性——图模型中的输出变量被排列成一个序列。为了展示该模型的好处，我们讨论一个自然语言处理中的应用——**命名实体识别 (named-entity recognition, NER)**。NER是在文本中识别并分类命名实体，包括地点（如China），人（如George Bush）和组织（如United Nations）。给定一个句子，命名实体识别任务是把其中的单词切分成几段，每一段对应一个实体，然后对该实体进行分类（类别包括人，组织，地点等等）。该问题的挑战性在于，很多实体的字符串很少见，哪怕在一个很大的训练集上。于是，我们只能根据上下文来识别它们。

一种办法是独立地对每个单词进行分类，看它是一个人、地点、组织或者其他（既不是一个实体）。这种办法的缺点在于：给定输入之后，它假定所有的命名实体标签是独立的。实际上，临近单词的标签是相关的。例如，New York是一个地点，Now York Times却是一个组织。一种缓解这种无关性假设的方法，是把输出变量安排到一个线性链中。这是隐马尔科夫模型（HMM）【111】的方法。一个HMM通过假定一个潜在的状态序列 $\mathbf{Y} = \{y_t\}_{t=1}^T$ ，来对一序列的观测 $\mathbf{X} = \{x_t\}_{t=1}^T$ 建模。记 S 为可能状态的有限集， O 为可能观测的有限集，即是说，对于任何的 t ， $x_t \in O, y_t \in S$ 。译注： S 包含了所有可能的输出值， O 包含了所有可能的输入值。在命名实体例子中， t 位置的单词就是观测 x_t ，而 y_t 是该位置的标签。

为了可行地对联合分布 $p(\mathbf{y}, \mathbf{x})$ 建模，一个HMM做了两个无关性假设。第一，它假设每个状态只依赖于它的前一个状态，即给定 y_{t-1} 之后， y_t 与 y_1, y_2, \dots, y_{t-1} 都无关了。第二，它假定每个观测变量 x_t 只与对应的状态 y_t 有关。基于这些假设，我们可用三个概率分布来指明一个HMM。第一个，初始状态的概率布 $p(y_1)$ ；第二个，转移概率 $p(y_t|y_{t-1})$ ；最后，观测概率 $p(x_t|y_t)$ 。总之，状态序列 \mathbf{y} 于观测序列 \mathbf{x} 的联合分布被分解为：

$$p(\mathbf{y}, \mathbf{x}) = \prod_{t=1}^T p(y_t|y_{t-1})p(x_t|y_t). \quad (2.10)$$

为了简化上式的符号，我们创造了“虚拟”初始状态 y_0 ，它总是0，并是所有状态序列的起点。这让我们把初始状态概率 $p(y_1)$ 写成 $p(y_1|y_0)$ 。

HMMs已在自然语言处理中用于很多序列标注任务，如part-of-speech tagging, 命名实体识别和信息提取。

2.2.3比较

生成模型和判别模型都描述了 (\mathbf{y}, \mathbf{x}) 的分布，却是从不同的方向。生成模型，如朴素贝叶斯分类器和HMM，是一簇按照 $p(\mathbf{y}, \mathbf{x}) = p(\mathbf{y})p(\mathbf{x}|\mathbf{y})$ 进行分解的联合分布。也就是说，它描述了如何根据标签采样或“生成”特征。判别模型，如逻辑回归模型，是一簇条件分布 $p(\mathbf{y}|\mathbf{x})$ 。也就是说，直接对分类规则建模。原理上，利用输入的边缘分布 $p(\mathbf{x})$ ，一个判别模型可以被转化成联合分布 $p(\mathbf{y}, \mathbf{x})$ ，然而很少需要这么做。

判别和生成模型在概念上的主要区别，就是条件分布 $p(\mathbf{y}|\mathbf{x})$ 没有包含 $p(\mathbf{x})$ 的模型，而它对分类并没有用。对 $p(\mathbf{x})$ 建模的困难性在于，它包含了很多高度相关的特征，而这是很难建模的。如在命名实体识别中，朴素的HMM只依赖于单一的特征——单词本身。然而许多单词，特别是专有名词，却从未在训练集中出现过，因而以单词本身作为特征是缺乏足够的信息的。为了对全新单词进行标注，我们想要利用其它的特征，如它的大小写、它的临近单词、它的前后缀、它在预先确定的一组人或地方中的身份（its membership in predetermined lists of people and locations???），等等。

判别模型的主要优势在于它适合包含丰富的、重叠的特征。为了理解这一点，考虑一簇朴素贝叶斯分布(2.7)。这簇联合分布的条件部分均采用了“逻辑回归的形式”（2.9）。然而还有很多其他的联合模型，有些带有 \mathbf{x} 之间的复杂的依赖，而条件分布也采用了（2.9）的形式。为了直接对条件分布建模，我们仍然可以认为 $p(\mathbf{x})$ 是不可知的。判别模型，如CRF，仅对 \mathbf{y} 的条件独立性做假设，以及 \mathbf{y} 如何依赖于 \mathbf{x} ，但是不对 \mathbf{x} 之间的条件独立性做假设。这一点也可以通过图形的方式来理解。假定我们有关于联合分布 $p(\mathbf{y}, \mathbf{x})$ 的因子图，现在要构建条件分布 $p(\mathbf{y}|\mathbf{x})$ 的因子图，那么，所有只与 \mathbf{x} 有关的因子都可以消失了。它们与条件部分无关，因为它们关于 \mathbf{y} 是常数。

为了在生成模型中包含互相关的特征，我们有两个选择。一是增强模型以表达输入间的相关性，如在每个 x_t 之间增加连接。然而很难可操作地这样做。例如，很难想象如何对单词的大小写以及前后缀之间的相关性建模。亦或者，我们也不想去这个件事，因为我们总是看得到输入的句子。

第二个办法是只做一些简单的相关性假设，如朴素贝叶斯假设。例如，带有朴素贝叶斯假设的HMM采用了 $p(\mathbf{x}, \mathbf{y}) = \prod_{t=1}^T p(y_t|y_{t-1}) \prod_{k=1}^K p(x_{tk}|y_t)$ 的形式。这一思路有时很凑效，但也可能很有问题，因为这一独立性假设会影响性能。例如，虽然朴素贝叶斯分类器在文档分类方面表现优秀，它在许多应用中的平均表现要比逻辑回归差【19】。

而且，朴素贝叶斯可以产生差的概率估计。作为说明的例子，想象朴素贝叶斯在一个二分类问题上训练。现在，我们把输入特征向量 $\mathbf{x} = (x_1, x_2, \dots, x_K)$ 重复一下，变换成 $\mathbf{x}' = (x_1, x_1, x_2, x_2, \dots, x_K, x_K)$ ，然后运行朴素贝叶斯分类器。虽然没有任何新的信息被加入到数据中，这一变换却增加了概率估计的信心。就是说，朴素贝叶斯对 $p(\mathbf{y}|\mathbf{x}')$ 的估计，相比于 $p(\mathbf{y}|\mathbf{x})$ ，更倾向远离0.5。

当我们扩展到序列模型的时候，想朴素贝叶斯那样的假设尤其有问题，因为推断过程需要综合模型不同部分的证据。如果序列的每个位置的标签，其概率估计都偏大，那么很难合理地把它们综合起来。

朴素贝叶斯和逻辑回归之间的差别，正是前者是生成的，而后者是判别的。在输入为离散时，这两个分类器在其他方面完全一致。朴素贝叶斯和逻辑回归考虑了相同的假设空间，因为在相同的决策范围里，任何逻辑回归分类器都可以转变成朴素贝叶斯分类器，反之亦然。再者，朴素贝叶斯模型(2.7)与逻辑回归模型 (2.9) 定义了相同的分布簇。我们可以生成式地表示 (2.7) 如下：

$$p(\mathbf{y}, \mathbf{x}) = \frac{\exp\{\sum_k \theta_k f_k(\mathbf{y}, \mathbf{x})\}}{\sum_{\hat{\mathbf{y}}, \hat{\mathbf{x}}} \theta_k f_k(\hat{\mathbf{y}}, \hat{\mathbf{x}})}. \quad (2.11)$$

这意味着，如果朴素贝叶斯(2.7)按照极大条件似然来训练，我们会获得与逻辑回归一样的分类器。相反，如果按照生成方法来表示逻辑回归，如 (2.11)，并按照最大化联合似然 $p(\mathbf{y}, \mathbf{x})$ 来训练，我们会得到与朴素贝叶斯同样的分类器。按照Ng和Jordan【98】的说法，朴素贝叶斯和逻辑回归构成了**生成-判别对 (generative-discriminative pair)**。关于最新的生成与判别模型的理论视角，请参考Liang和Jordan【72】。

原理上，我们可能不清楚这两种方案如此不同的原因，毕竟它们之间可通过贝叶斯法则互相转化。如在朴素贝叶斯模型中，是很容易把联合分布 $p(\mathbf{y})p(\mathbf{x}|\mathbf{y})$ 转化成条件分布 $p(\mathbf{y}|\mathbf{x})$ 的。实际上，该条件分布与逻辑回归模型 (2.9) 的形式是一样的。另外如果我们想获得关于数据的“真实”生成模型，即真正把数据产生出来的分布 $p^*(\mathbf{y}, \mathbf{x}) = p^*(\mathbf{y})p^*(\mathbf{x}|\mathbf{y})$ ，那么我们只需简单地计算真实的 $p^*(\mathbf{y}|\mathbf{x})$ ，而这正是判别方法的目标。然而正是因为我们无法准确地获得真实的分布，造成这两种方案在实践中是不同的。先估计 $p(\mathbf{y})p(\mathbf{x}|\mathbf{y})$ ，然后计算 $p(\mathbf{y}|\mathbf{x})$ （生成方案），会产生与直接估计 $p(\mathbf{y}|\mathbf{x})$ 不同的结果。也就是说，生成与判别模型的目标都是估计 $p(\mathbf{y}|\mathbf{x})$ ，却是通过不同的路径达到的。

我们关于生成与判别之间差异的深入观点，来自Minka【93】。假如我们拥有一个生成模型 p_g ，其参数为 θ 。根据定义，其形式为：

$$p_g(\mathbf{y}, \mathbf{x}; \theta) = p_g(\mathbf{y}; \theta)p_g(\mathbf{x}|\mathbf{y}; \theta). \quad (2.12)$$

但是我们可以按照概率的链式法则重写 p_g 如下：

$$p_g(\mathbf{y}, \mathbf{x}; \theta) = p_g(\mathbf{x}; \theta)p_g(\mathbf{y}|\mathbf{x}; \theta), \quad (2.13)$$

其中， $p_g(\mathbf{x}; \theta)$ 和 $p_g(\mathbf{y}|\mathbf{x}; \theta)$ 是通过推断来计算的，即 $p_g(\mathbf{x}; \theta) = \sum_{\mathbf{y}} p_g(\mathbf{y}, \mathbf{x}; \theta)$ 以及 $p_g(\mathbf{y}|\mathbf{x}; \theta) = p_g(\mathbf{y}, \mathbf{x}; \theta)/p_g(\mathbf{x}; \theta)$ 。

现在要在同样的联合分布簇上，把这个生成模型与判别模型做比较。为了这么做，我们定义一个关于输入的先验概率 $p(\mathbf{x})$ ，使得 $p(\mathbf{x})$ 可以从 p_g 的某个参数配置中产生。就是说， $p(\mathbf{x}) = p_c(\mathbf{x}; \theta') = \sum_{\mathbf{y}} p_g(\mathbf{y}, \mathbf{x}; \theta')$ **译注：原文是**
 $p(\mathbf{x}) = p_c(\mathbf{x}; \theta') = \sum_{\mathbf{y}} p_g(\mathbf{y}, \mathbf{x}; \theta')$ ，其中 θ' 往往与 (2.13) 中的 θ 不同。把这与同样从 p_g 中产生的条件分布 $p_c(\mathbf{y}|\mathbf{x}; \theta)$ 组合，即 $p_c(\mathbf{y}|\mathbf{x}; \theta) = p_g(\mathbf{y}, \mathbf{x}; \theta)/p_g(\mathbf{x}; \theta)$ 。那么结果分布是：

$$p_c(\mathbf{y}, \mathbf{x}) = p_c(\mathbf{x}; \theta')p_c(\mathbf{y}|\mathbf{x}; \theta). \quad (2.14)$$

通过比较(2.13)和 (2.14)，可以看到条件方案具有更大的灵活性来拟合数据，因为它不要求 $\theta' = \theta$ 。直观地，因为 (2.13) 中的参数 θ 被同时用于输入的分布和条件部分。那么一组参数需要在两方面都表现良好。潜在地，需要损失我们所关心的 $p(\mathbf{y}|\mathbf{x})$ 的准确性，来弥补我们不怎么关心的 $p(\mathbf{x})$ 的准确性。另一方面，引入了更多的自由度，增加了过拟合的风险，降低了泛化到新数据的能力。

尽管到目前为止我们一直在批判生成模型，它们也有自己的优势。第一，生成模型可以更自然地处理隐藏变量，半标注数据以及未标注数据。在更极端的例子中，当整个数据都未被标注时，生成模型可以按照非监督模式使用。相反，非监督学习在判别模型中不够自然，且扔是一个活跃的研究领域。

第二，在某些例子中生成模型表现得比判别模型好，直观上是因为输入模型 $p(\mathbf{x})$ 对条件分布的影响是光滑的 (smoothing)。Ng和Jordan【98】争辩道，这一作用在小数据机上尤其显著。对于任何特定的数据集，我们不可能知道谁更有优势。总之，要么问题本身需要一个自然的生成模型，要么需要同时预测输入与输出译注：一般应用假定输入为已知，而只需预测输出，都会使生成模型更被青睐。

因为生成模型的形式为 $p(\mathbf{y}, \mathbf{x}) = p(\mathbf{y})p(\mathbf{x}|\mathbf{y})$ ，使得通过有向图来表示它更自然。其中在拓扑意义上，输出 \mathbf{y} 要在输入之前。相似地，我们将会看到，用无向图来表示判别模型更自然。然而，并非总是如此。无向的生成模型，如马尔科夫随机场 (2.32)，以及有向的判别模型，如MEMM (6.2)，有时也会被采用。有时用有向图来表示判别模型也会有用，其中 \mathbf{x} 在 \mathbf{y} 之前。

朴素贝叶斯与逻辑回归之间的关系，正如HMMs和线性链CRFs。正如朴素贝叶斯与逻辑回归是生成-判别对，也存在着HMMs的判别对应物。这一对应物是一种特殊的CRF。我们将在接下来一章中介绍。朴素贝叶斯、逻辑回归、生成模型和CRFs之间的类比，如图2.4所示。

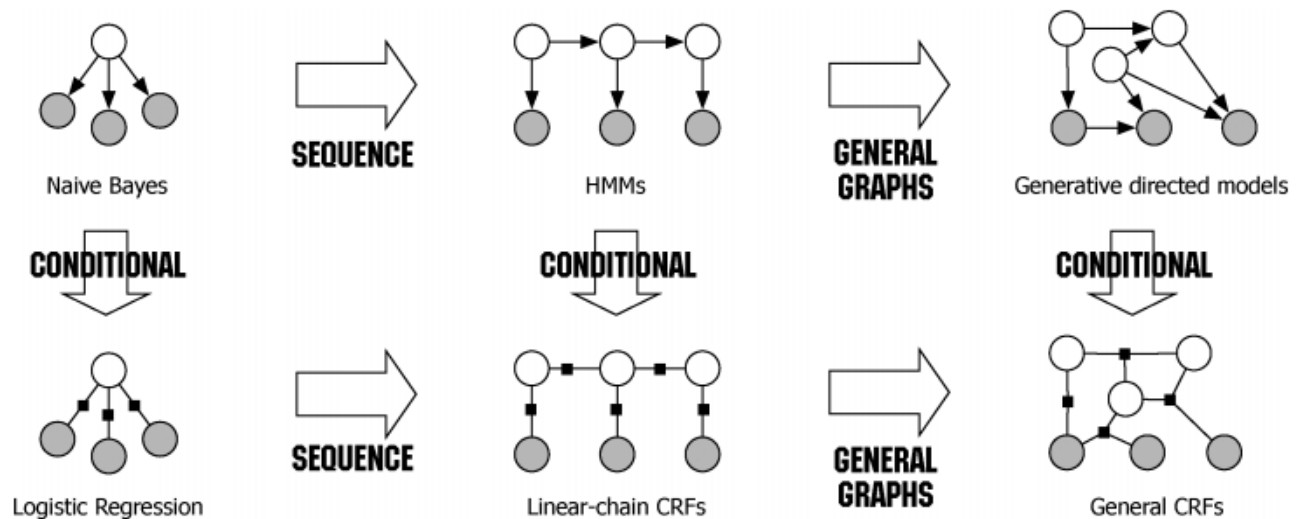


图2.4 朴素贝叶斯、逻辑回归、HMMs、线性链CRFs、生成模型和广义CRFs之间的关系图

2.3 线性链CRFs

为了引出线性链CRFs，我们考虑从HMM的联合分布 $p(\mathbf{y}, \mathbf{x})$ 引出的条件分布 $p(\mathbf{y}|\mathbf{x})$ 。关键点在于，这一条件分布是一种具有特殊的特征方程的CRF。

首先，我们来重写HMM的联合分布(2.10)，使其更利于扩展，即：

$$p(\mathbf{y}, \mathbf{x}) = \frac{1}{Z} \prod_{t=1}^T \exp \left\{ \sum_{i,j \in S} \theta_{ij} \mathbf{1}_{\{y_t=i\}} \mathbf{1}_{\{y_{t-1}=j\}} + \sum_{i \in S} \sum_{o \in O} \mu_{oi} \mathbf{1}_{\{y_t=i\}} \mathbf{1}_{\{x_t=o\}} \right\}, \quad (2.15)$$

其中， $\theta = \{\theta_{ij}, \mu_{oi}\}$ 是分布的实值参数， Z 是归一化常数，能使分布的和为1。如果我们不在 (2.15) 中添加 Z ，那么参数 θ 有可能带来不合理的关于 (\mathbf{y}, \mathbf{x}) 的分布，如当所有参数都是1时。

现在有意思的是，(2.15) (几乎) 确切地描述了 (2.10) 一类的HMMs。每个同类的HMM都可通过如下设置，写成 (2.15) 的形式：

$$\begin{aligned} \theta_{ij} &= \log p(y' = i | y = j) \\ \mu_{oi} &= \log p(x = o | y = i) \\ Z &= 1 \end{aligned}$$

反过来也是正确的，即是说，每个按照 (2.15) 分解的分布都是HMM。（利用4.1节介绍的前向-反向算法，可构造对应的HMM，从而证明这一点）。因而尽管在参数中增加了灵活性，我们却没有扩大分布簇。

通过使用**特征函数feature functions**，我们可以把 (2.15) 弄得更紧凑，正如我们在 (2.9) 的逻辑回归那里一样。每个特征函数都具有形式 $f_k(y_t, y_{t-1}, x_t)$ 。对于 (2.15)，我们需要给每个转移 (i, j) 一个特征 $f_{ij}(y, y', x) = \mathbf{1}_{\{y=i\}} \mathbf{1}_{\{y'=j\}}$ ，以及给每个“状态-特征对” (i, o) 一个特征 $f_{io}(y, y', x) = \mathbf{1}_{\{y=i\}} \mathbf{1}_{\{x=o\}}$ 。我们泛泛地用 f_k 来引用一个特征，其中 f_k 涵盖了全部都的 f_{ij} 和全部的 f_{io} 。于是，我们可以重写HMM如下：

$$p(\mathbf{y}, \mathbf{x}) = \frac{1}{Z} \prod_{t=1}^T \exp \left\{ \sum_{k=1}^K \theta_k f_k(y_t, y_{t-1}, x_t) \right\}. \quad (2.16)$$

再一次，方程 (2.16) 定义了与 (2.15) 完全一样的分布簇，从而也与最初的HMM方程 (2.10) 一样。

最后一步，是把来自HMM (2.16) 的条件分布 $p(\mathbf{y}|\mathbf{x})$ 写出来，即：

$$p(\mathbf{y}|\mathbf{x}) = \frac{p(\mathbf{y}, \mathbf{x})}{\sum_{\mathbf{y}'} p(\mathbf{y}', \mathbf{x})} = \frac{\prod_{t=1}^T \exp \left\{ \sum_{k=1}^K \theta_k f_k(y_t, y_{t-1}, x_t) \right\}}{\sum_{\mathbf{y}'} \prod_{t=1}^T \exp \left\{ \sum_{k=1}^K \theta_k f_k(y'_t, y'_{t-1}, x_t) \right\}}. \quad (2.17)$$

(2.17) 所描述的条件分布，是线性链CRF的一种特例，即那种只包含当前单词作为特征的。然而，很多线性链CRF使用更为丰富的特征，如前后缀等等。幸运的是，将我们现有的记号扩展并非难事。我们只需简单地允许特征函数包含更多的输入。这导致了我们的关于线性链CRFs的一般定义

定义2.2 记 \mathbf{Y}, \mathbf{X} 是随机向量， $\theta = \{\theta_k\} \in \mathcal{R}^K$ 是一个参数向量， $\mathcal{F} = \{f_k(y, y', x_t)\}_{k=1}^K$ 为一组实值特征函数。那么**线性链条件随机场**是如下形式的分布 $p(\mathbf{y}|\mathbf{x})$ ：

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \prod_{t=1}^T \exp \left\{ \sum_{k=1}^K \theta_k f_k(y_t, y_{t-1}, x_t) \right\}, \quad (2.18)$$

其中， $Z(\mathbf{x})$ 是依赖于输入的归一化函数：

$$Z(\mathbf{x}) = \sum_{\mathbf{y}} \prod_{t=1}^T \exp \left\{ \sum_{k=1}^K \theta_k f_k(y_t, y_{t-1}, x_t) \right\}. \quad (2.19)$$

译注：线性链条件随机场，好像是一类随机场，实际是一个随机场——结构是定死的。我觉得这是条件随机场最非常核心的问题，本文却并没有阐明。当然，它对输入的引用还是很灵活的。

注意，线性链CRF可以用 \mathbf{x} 和 \mathbf{y} 上的因子图来描述，即

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^T \Psi_t(y_t, y_{t-1}, x_t) \quad (2.20)$$

其中，局部函数 Ψ_t 具有一种特殊的 log-linear 形式：

$$\Psi_t(y_t, y_{t-1}, x_t) = \exp \left\{ \sum_{k=1}^K \theta_k f_k(y_t, y_{t-1}, x_t) \right\}. \quad (2.21)$$

当我们在下一节进入一般意义CRF的时候，这会很有用。

一般来说，我们将从数据中学得参数 θ 。这将在第5节讲述。

之前我们已看到，如果一个联合分布 $p(\mathbf{y}, \mathbf{x})$ 像HMM一样分解了，那么对应的条件分布 $p(\mathbf{y}|\mathbf{x})$ 是一个线性链CRF。这一很像HMM的CRF如图2.5所示。然而，其他类型的线性链CRFs也是有用的。例如，在一个HMM中，状态 i 到 j 的转移概率与输入无关，都是 $\log p(y_t = j|y_{t-1} = i)$ 。在CRF中，我们可以让转移概率 (i, j) 依赖于当前的观测向量，这只需添加特征 $\mathbf{1}_{\{y_t=j\}} \mathbf{1}_{\{y_{t-1}=i\}} \mathbf{1}_{\{x_t=o\}}$ 。具有这一转移特征的CRF常常被用于文本处理，如图2.6所示。

实际上，因为CRFs不在乎输入变量 $\mathbf{x}_1, \dots, \mathbf{x}_T$ 之间的关系，我们可以让因子 Ψ_t 依赖于所有的输入 \mathbf{x} 。这不会大破线性图的结构——允许我们把 \mathbf{x} 当成单一的整体变量。结果，特征函数可以写成 $f_k(y_t, y_{t-1}, \mathbf{x})$ ，从而可以把全部的输入变量 \mathbf{x} 一块考虑。这一事实对CRFs都适用，而不只是对线性链。具有这一结构的线性链如图2.7所示。途中，我们把 $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_T)$ 画成一个巨大的观测节点，冰杯所有的因子依赖，而不是把 $\mathbf{x}_1, \dots, \mathbf{x}_T$ 画成独立的节点。

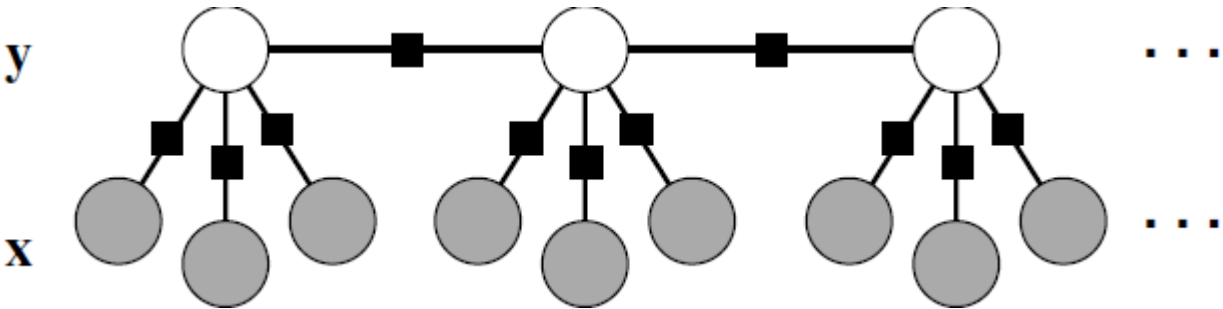


图2.5 来自式 (2.17) 的类HMM的线性链CRF

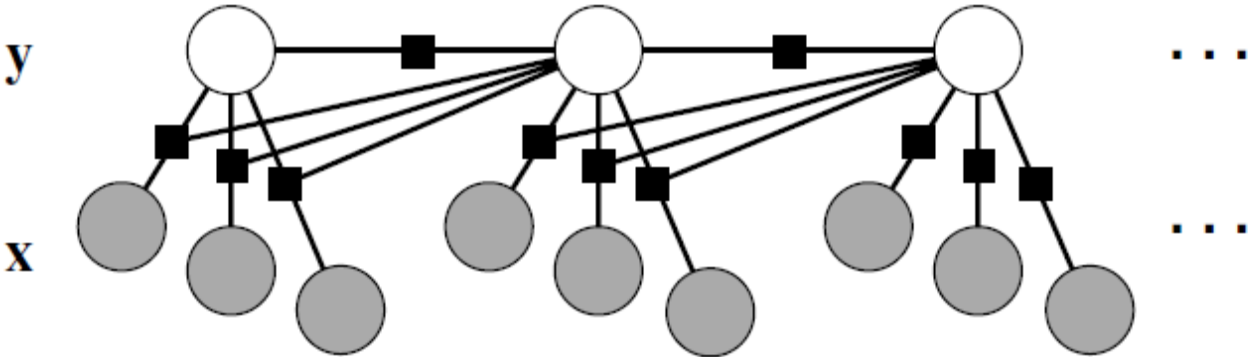


图2.6 转移因子依赖于当前输入的线性链CRF

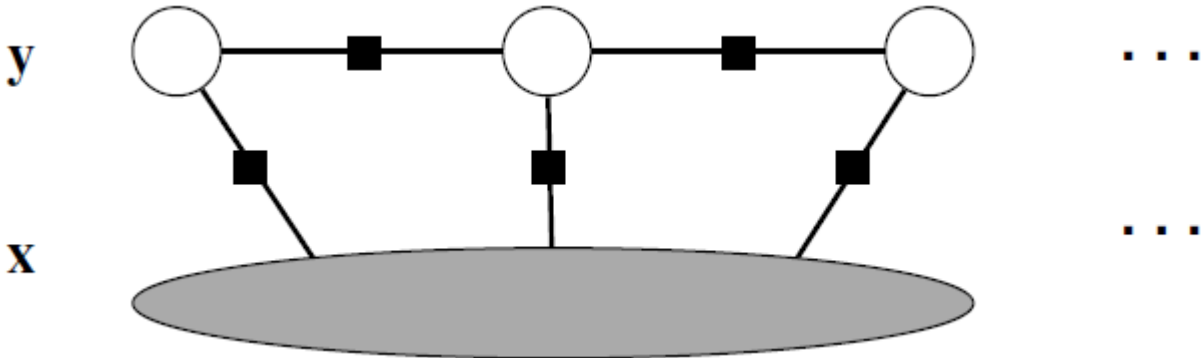


图2.7 转移因子依赖于全部输入的线性链CRF

需支出，在我们关于线性链CRF的定义中，特征函数可以从任意时刻依赖于输入，把 f_k 关于输入的参数写成了 \mathbf{x}_t 。 \mathbf{x}_t 应当被理解成——计算 t 时刻特征所需的全部输入译注：而不是 t 时刻的输入。例如，如果CRF需要下一时刻的单词 \mathbf{x}_{t+1} ，那么 \mathbf{x}_t 应当包含了 \mathbf{x}_{t+1} 。

最后，归一化常数 $Z(\mathbf{x})$ 需要在全部可能的输出序列上求和，包含有爆炸式的大量的项。然而，它可以被前向-反向算法有效地解，正如我们在第4.1节所揭示的。

2.4 通用CRFs

现在，我们将刚刚探讨的线性链扩展到通用图，以与Lafferty在【63】中对CFR的定义相匹配。概念上，这一扩展是显而易见的。我们只需简单地把线性链因子图变成通用因子图。

定义2.3 记 G 是在 X, Y 上的因子图。如果对于 X 中任意的值 \mathbf{x} ，分布 $p(\mathbf{y}|\mathbf{x})$ 是根据 G 来分解的，那么 (X, Y) 是一个**条件随机场conditional random field**。

那么，每个条件分布 $p(\mathbf{y}|\mathbf{x})$ 都是某些因子图的CRF，包括是平凡的。如果 $F \in \{\Psi_a\}$ 是 G 中的因子的集合，那么一个CRF的条件分布为：

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{a=1}^A \Psi_a(\mathbf{y}_a, \mathbf{x}_a). \quad (2.22)$$

本定义相比一般无向图的定义（2.1），差别在于归一化常数 $Z(\mathbf{x})$ 现在变成了关于输入 \mathbf{x} 的函数。因为条件性趋向于简化图模型， $Z(\mathbf{x})$ 有可能被计算，而 Z 却不是。

正如我们在HMMs和线性链CRFs中的做法，让 Ψ_a 是一组特征的线性函数是有用的，即：

$$\Psi_a(\mathbf{y}_a, \mathbf{x}_a) = \exp \left\{ \sum_{k=1}^{K(A)} \theta_{ak} f_{ak}(\mathbf{y}_a, \mathbf{x}_a) \right\}, \quad (2.23)$$

其中特征函数 f_{ak} 和权重 θ_{ak} 都使用了因子的下标 a ，这是为了强调每个因子都有自己的权重集。一般来说，每个因子也可以拥有自己的特征函数。注意，如果 \mathbf{x} 和 \mathbf{y} 是离散的，那么（2.23）中的log-线性假设并没有带来额外的局限，因为我们可以给 $(\mathbf{y}_a, \mathbf{x}_a)$ 的每一个值安排一个指示函数 f_{ak} ，类似于我们把HMMs转变成线性链CRF时的做法。

综合（2.22）和（2.23），可以把log-线性因子CRF的条件分布写成

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{\Psi_A \in F} \exp \left\{ \sum_{k=1}^{K(A)} \theta_{ak} f_{ak}(\mathbf{y}_a, \mathbf{x}_a) \right\}. \quad (2.24)$$

另外，许多应用模型常常需要参数绑定。以线性链为例，每一时刻的因子 $\Psi_t(\mathbf{y}_t, \mathbf{y}_{t-1}, \mathbf{x}_t)$ 常常使用相同的权重。为了表示这一情况，我们把 G 的因子划分成 $\mathcal{C} = \{C_1, C_2, \dots, C_P\}$ ，其中每个 C_P 是一个**团模板clique template**，是一组共享了特征函数 $\{f_{pk}(\mathbf{x}_c, \mathbf{y}_c)\}_{k=1}^{K(p)}$ 和参数 $\theta_p \in \mathcal{R}^{K(p)}$ 的因子。一个使用了团模板的CRF可以写成

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{C_p \in \mathcal{C}} \prod_{\Psi_c \in C_p} \Psi_c(\mathbf{x}_c, \mathbf{y}_c; \theta_p). \quad (2.27)$$

其中每个模板因子是这样参数化的

$$\Psi_c(\mathbf{x}_c, \mathbf{y}_c; \theta_p) = \exp \left\{ \sum_{k=1}^{K(p)} \theta_{pk} f_{pk}(\mathbf{x}_c, \mathbf{y}_c) \right\}, \quad (2.26)$$

而归一化函数为

$$Z(\mathbf{x}) = \sum_{\mathbf{y}} \prod_{C_p \in \mathcal{C}} \prod_{\Psi_c \in C_p} \Psi_c(\mathbf{x}_c, \mathbf{y}_c). \quad (2.27)$$

这一团模板的记号方法即指明了结构重复，也指明了参数绑定。以线性链CRF为例，典型的团模板

$C_0 = \{\Psi_t(\mathbf{y}_t, \mathbf{y}_{t-1}, \mathbf{x}_t)\}_{t=1}^T$ 倍整个网络使用，因而 $\mathcal{C} = \{C_0\}$ 是元素单一的集合。如果相反地，我们希望给每个因子 Ψ_t 分配独立的参数，就像非齐次HMM，那么需要 T 个模板，即 $\mathcal{C} = \{C_t\}_{t=1}^T, C_t = \{\Psi_t(\mathbf{y}_t, \mathbf{y}_{t-1}, \mathbf{x}_t)\}$ 。

定义通用CRF时，如何给出重复的结构以及参数绑定，是属于最需要考虑的问题。人们推荐了一系列的规范，用于指定团模板，而我们仅仅在这里简单的罗列一下。例如，**动态条件随机场dynamic conditional random field** [140] 是一些序列模型，允许在每个时刻拥有多个标签 **译注：不是指有多个类别，而是有多个变量**，而不只是单一的标签，很像动态贝叶斯网络。第二，**关系马尔科夫网relational Markov networks** [142]，是一种用类SQL的语法来指明图结构和参数绑定的通用CRF。**马尔科夫逻辑网Markov logic networks** [113, 128] 用逻辑式子 (logic formulae) 来给出无向图的局部函数的分数。实质上，知识库中的每条一阶规则都存在一组参数。MLN的逻辑部分，本质上，可以被看成一种编码惯例，用来指明无向图中的重复结构以及参数绑定。Imperatively define factor graphs [87] 使用了完整表达的Turing-complete函数来定义团模板，即给出了模型的结构，也给出了充分统计量 f_{pk} 。这些函数灵活地采用了先进的编程思想，包括递归、任意搜索 (arbitrary search)、惰性计算以及记忆化。本文采用的团模板的记号，来自于Taskar et al. [142], Sutton et al. [140], Richardson 和 Domingos [113], 以及McCallum et al. [87]

2.5特征工程

不知道怎么翻译这里的**专业名词**

这一节，我们讲述一些特征工程中的技巧。虽然主要用于语言处理，它们还是很通用的。最主要的权衡很典型——大的特征集可以提高预测的精度，因为决策便捷更加灵活，但却需要更大的内存来保存参数，且可能因为过拟合而降低预测精度。

标签-观测特征?Label-observation features. 首先，当标签是离散变量，那么团模板 C_p 的特征 f_{pk} 常常采用如下的特定形式：

$$f_{pk}(\mathbf{y}_c, \mathbf{x}_c) = \mathbf{1}_{\{y_c = \tilde{y}_c\}} q_{pk}(\mathbf{x}_c). \quad (2.28)$$

也就是说，一个特征只在输出正好为 \tilde{y}_c 时才非零，而一旦如此，便只与输入有关。我们把具有这种形式的特征称为**标签-观测特征**。本质上可以这么来理解：特征只依赖于输入 \mathbf{x}_c ，但每一种输出都有自己的一组权重。这一特征表示法的计算效率也很高，因为计算每个 q_{pk} 都可能涉及文本或图片处理，而只需要处理一次，就可用于每一个用到它的特征。为了避免混淆，我们把函数 $q_{pk}(\mathbf{x}_c)$ 叫做观测函数，而不是特征。观测函数的例子有“单词 \mathbf{x}_t 是大写的”或“单词 \mathbf{x}_t 以ing结尾”。

Unsupported Features. 使用标签-观测特征可能会带来数量庞大的参数。例如在CRFs的第一个大规模应用中，Sha和Pereira [125] 在他们的最佳模型中，使用了3百8十万个参数。其中的很多特城从未在训练数据中出现过——它们总是0。原因在于，许多观测函数只与一小部分的标签相对应。例如在命名实体识别任务中，“单词 \mathbf{x}_t 是with，而标签 \mathbf{y}_t 是CITY-NAME”，似乎永远不可能在训练集中为真。我们把它们称为unsupported features。可能很意外，这些特征也可能有用，因为可以给它们赋予负的权重，从而防止给错的标签以高的概率。（降低那些从未出现过的标签序列的分数，将会增加那些出现过的标签序列的概率，所以在后文我们描述的参数估计方法中，会给这些特征以负的权重）。包含unsupported features常常带来精度的少量提升，并以巨大的参数数量为代价。

我们曾利用一个特别的技术，来选择unsupported features的一小部分。这可以看成是使用更少内存来利用的unsupported feature的一次简单探索，可以被称为“unsuported features trick”。它认为许多unsupported features是无用的，因为模型不太可能因为它们的激活而犯错。例如，那个“with”特征不太可能有用，因为with是一个常见的单词，且总是属于OTHER标签（即它不是一个名词）。为了减少参数的数量，我们只保留那些有可能剔除错误的unsupported features。一个简单的方法是：首先训练一个不带unsupported feature的CRF，并在几次

迭代后就停下来，使得模型并没有完全训练好。然后考虑那些模型未能给正确答案以高概率的团，给它们增加 unsupported features。在上面这个例子中，如果我们发现训练集中有一个样本 i ，其 t 位置的序列 $\mathbf{x}_t^{(i)}$ 是 with，而 $y_t^{(i)}$ 不是“CITY-NAME”原文是 $y_t^{(i)}$ is not CITY-NAME。我认为应去掉 not。译文则保留了 this not，并且 $p(y_t = \text{CITY} - \text{NAME} | \mathbf{x}_t^{(i)}) > \epsilon$ 时 (ϵ 是一个阈值)，我们增加“with”这一特征。

连线-观测特征和节点-观测特征? Edge-Observation and Node-Observation Features. 为了减少模型中的特征数量，我们可以只在某些团使用标签-观测特征，而不是全部。最常见的两种标签-观测特征是 **连线-观测特征** 和 **节点-观测特征**。考虑一个具有 M 个观测函数 $\{q_m(\mathbf{x})\}, m \in \{1, 2, \dots, M\}$ 的线性链 CRF。如果使用了连线-观测特征，那么每个局部函数可以依赖于全部的观测函数。那么，我们可以使用这样的特征：单词 \mathbf{x}_t 是 New， y_t 是 LOCATION 且 y_{t-1} 也是 LOCATION。这会导致模型拥有大量的参数，带来内存消耗和过拟合的缺点。一种解决办法是采用节点-观测特征。使用这一类型的特征，转移因子 **就是局部函数吧?** 不在依赖于观测函数。于是我们可以使用类似“ y_t 是 LOCATION，且 y_{t-1} 是 LOCATION”，以及“ \mathbf{x}_t 是 NEW，且 y_t 是 LOCATION”的特征，而不能使用那种一次把 $\mathbf{x}_t, y_t, y_{t-1}$ 都依赖上的特征。连线-观测特征和节点特征都正式地在表 2.1 中给出了。一般来说，以上两种特征的选择，需要根据具体的问题来定，如需要考虑观测函数的数量，以及数据集的大小。

Table 2.1. Edge-observation features versus node-observation features.

Edge-observation features:		
$f(y_t, y_{t-1}, \mathbf{x}_t) = q_m(\mathbf{x}_t) \mathbf{1}_{\{y_t=y\}} \mathbf{1}_{\{y_{t-1}=y'\}}$	$\forall y, y' \in \mathcal{Y}, \forall m$	
$f(y_t, \mathbf{x}_t) = q_m(\mathbf{x}_t) \mathbf{1}_{\{y_t=y\}}$	$\forall y \in \mathcal{Y}, \forall m$	
Node-observation features:		
$f(y_t, y_{t-1}, \mathbf{x}_t) = \mathbf{1}_{\{y_t=y\}} \mathbf{1}_{\{y_{t-1}=y'\}}$	$\forall y, y' \in \mathcal{Y}$	
$f(y_t, \mathbf{x}_t) = q_m(\mathbf{x}_t) \mathbf{1}_{\{y_t=y\}}$	$\forall y \in \mathcal{Y}, \forall m$	

Boundary Labels. 最后一个问题是如何在边缘上取标签，例如一个序列的开始和结尾，或一张画的边缘。有时，边缘上的标签与其他标签不同。例如，大写字母在一个句子的中间意味着是专有名词，但如果是在句子的开始却没有这样的意味。一个简单的办法，是在标签序列的前面加一个特殊的标签——START。这允许模型学习得到关于边缘的特性。例如，如果连线-观测特征也被使用了，那么像“ $y_{t-1} = \text{START}$ 且 $y_t = \text{PERSON}$ 且 \mathbf{x}_t 大写”这样的特征，可以表示，大写这一特征在句子的开始时并不是有效的。

特征归纳? Feature Induction 上文介绍的“unsupported features trick”是“feature induction”的简化版。

McCallum [83] 提供了 CRFs 特征归纳的更有条理的方法。其中，模型一开始只有一些基本特征，而训练过程会增加这些特征的连接。另外一个选择是特征选择。一个现代的特征选择方法是 L_1 规则化。我们将在第 5.1.1 介绍它。Lavergne et al. [65] 发现，在最好的时候， L_1 可以找到一种模型。它只有 1% 的参数是非零的，却获得与稠密特征集相当的性能。他们还发现，利用 L_2 规则化目标函数，来对 L_1 规则化所得的非零特征进行微调，也是有用的。

Categorical Features 类属特征 (非数值特征)。如果观测是类属的，而不是有序的，就是说，它们是离散而没有内在的顺序性，那么将它们转化成二值化特征是重要的。例如，很合理将特征 $f_k(y, \mathbf{x}_t)$ 定义为“如果 \mathbf{x}_t 是单词 dog 时， $f_k = 1$ ，否则为 0”。相反，把 f_k 定义为单词 \mathbf{x}_t 在文本词典中的序号是不合理的。因而在文本处理中，CRF 特征常常是二值化的；而在其他诸如视觉和语音识别中，特征常常是数值的。对于数值特征，标准的做法是通过归一化，使其均值为 0 而标准差为 1，或者把它们二值化，使其变成类属特征。

Features from Different Time Steps. 我们对于特征 $f_k(y_t, y_{t-1}, \mathbf{x}_t)$ 的关注可能遮掩了一点，即通常需要让特征的依赖范围，从最近邻扩展到附近的标签。一个这种特征的例子是“单词 \mathbf{x}_{t+2} 是 Times，而标签 y_t 是 ORGANIZATION”。这有利于识别名词“New York Times”报纸。同样，也临近特征的组合也是有用的，例如“单词 \mathbf{x}_{t+1} 和 \mathbf{x}_{t+2} 是 York Times”。

Features as Backoff回退特征? 在语言处理中,有时需要在模型中包含冗余因子。例如在线性链CRF中,有人会使用连接因子 $\Psi_t(y_t, y_{t-1}, x_t)$ 的同时,还使用变量因子 $\Psi_t(y_t, x_t)$ 。虽然只使用连接因子也可以定义同样的分布簇,然而当数据量小于特征的数量时,冗余节点因子却像回退语言模型那样有用。(当拥有百万级的特征时,很多数据是很小的!)当使用冗余特征时,规则化(5.1.1节)是很必要的,因为惩罚大的权重会让权重分布到重叠的特征上。

Features as Model Combination.另一种有意思的特征可以是相同任务的更简单方法的结果。例如,如果已经拥有了任务的简单规则库simple rule-base系统(例如这样的规则“1900和2100中间的数字字符串表示一个年份”),那么该系统的输出可被用做CRF的观测函数。另一个例子是名录特征gazetteer features,即其观测函数建立在一个预先建立的列表上,如“如果 x_t 出现在了Wikipedia提供的某个城市名单列表中,那么 $q(x_t) = 1$ ”。

更复杂的例子是把生成模型的输出当做判别模型的输出来用。例如人们可以使用 $f_t(y, x_t) = p_{HMM}(y_t = y|x)$ 作为特征,其中 p_{HMM} 表示某个HMM(在相近数据集训练所得的)所给出的 $y_t = y$ 的边缘概率。让HMM和CRF-with-HMM-feature在同一个数据上训练通常不是一个好的想法,因为HMM需要在它自己的数据集上表现极好,而这会让CRF过分依赖与HMM。这一技术可用于提高某个早前的、同一任务的系统的性能。Bernal et al【7】是这一概念的、在DNA序列中识别基因的一个好例子。

相关的想法是对输入 x_t 进行聚类,用任何方法对语料库中的单词进行聚类,然后用类别标签来作为单词 x_t 的附加特征。这种特征在Miller et al.[90]那里取得了好的效果。

Input-Dependent Structure.在通用CRF中,有时需要让 $p(y|x)$ 的图结构随着输入 x 变化。关于此的一个简单例子是关于文本处理的“skip-chain CRF”【37,117,133】。其背后的思想是,一旦某个单词在句子中出现了两次,我们希望它们属于相同的标签。于是我们在这两个单词中间增加一条连接特征。这让 y 之上的图结构依赖于输入 x 。

2.6 例子

这一节,我们提供两个CRF是应用的细节。第一个是自然语言文本的线性链CRF,而第二个是计算机视觉的网状CRF。

2.6.1命名实体识别

暂略

2.6.2图片分割Image Labeling

许多不同的CRF拓扑结构被用于计算机视觉。作为一个例子,我们希望根据前景和背景来对图片的区域分类。亦或按照人工构造物和非人工构造物【61,62】;天空、水域和菜地等来分类【49】。

正式地,记 $x = (x_1, x_2, \dots, x_T)$ 为一个向量,表示一张 $\sqrt{T} \times \sqrt{T}$ 的图片。就是说, $x_{1:\sqrt{T}}$ 代表第一行, $x_{\sqrt{T}+1:\sqrt{T}+2}$ 表示第二行,依次类推。每个 x_i 表示某个像素的值。简单起见,只考虑黑白图片,那么每个 x_i 都是0~255的一个实值,表示位置 i 的像素的亮度。(这可以轻易地扩展到彩色图)。目的是推断一个向量 $y = (y_1, y_2, \dots, y_T)$,其中每个 y_i 是位置 i 的标签,如+1表示人工构造物,而-1表示其他。

已有大量的计算机视觉文献,贡献了大量的图像特征。例如,给定一个像素位置 i ,我们可以计算其 5×5 的窗口内的亮度直方图,然后把每个柱体里的像素个数作为特征。通常会使用更复杂的特征,如图像的梯度特征,texture特征【127】以及SIFT特征【77】。重要的是,这些特征不只依赖于像素 x_i 自身,而是一个领域或全图的像素。

图片有一个基本的特征,就是临近的像素趋向属于相同的类别。把这一想法融入模型的办法是引入一个先验的 y 的分布,增加“光滑”分割的概率。计算机视觉中最常见的先验分布是网状的马尔科夫图模型,叫做**马尔科夫随机场 Markov random field**【10】。MRF是拥有两种因子的无向模型:一种因子把标签 y_i 与对应的像素 x_i 联系起来,另一种鼓励邻近的标签 y_i 和 y_j 相一致。

正式地，用 \mathcal{N} 定义像素间的邻居关系，即当 \mathbf{x}_i 和 \mathbf{x}_j 属于邻居时， $(i, j) \in \mathcal{N}$ 。一般来说， \mathcal{N} 的定义需使能构成一个 $\sqrt{T} \times \sqrt{T}$ 。一个MRF是一个生成模型：

$$p(\mathbf{y}) = \frac{1}{Z} \prod_{(i,j) \in \mathcal{N}} \Psi(y_i, y_j)$$

$$p(\mathbf{y}, \mathbf{x}) = p(\mathbf{y}) \prod_{i=1}^T p(\mathbf{x}_i | y_i). \quad (2.32)$$

其中， Ψ 是鼓励平滑性的因子。通常在 $y_i = y_j$ 时，让 $\Psi(y_i, y_j) = 1$ ，而其他时候为 α ，而 $\alpha < 1$ 是从数据中学到的参数。其背后的想法是，当 $\alpha < 1$ 时，存在快速的推断算法用来最大化 $\log p(\mathbf{y}, \mathbf{x})$ 。 $p(\mathbf{x}_i | y_i)$ 是像素值关于类别的条件分布。例如， \mathbf{x}_i 上的混合高斯。

MRF的缺点在于，很难使用一个区域上的特征。否则， $p(\mathbf{x} | \mathbf{y})$ 会变拥有很复杂的结构。条件模型提供了一个解决之道。

关于本任务，我们描述的CRF与MRF很像，但却允许因子依赖于单个或连接的像素的任何特征。记 $q(\mathbf{x}_i)$ 为在 \mathbf{x}_i 附近的区域上提取的特征，例如颜色直方图或图像梯度。进一步，我们定义 \mathbf{x}_i 和 \mathbf{x}_j 之间的特征向量 $v(\mathbf{x}_i, \mathbf{x}_j)$ ，以使模型能够处理 \mathbf{x}_i 与 \mathbf{y}_i 之间的相似与不同。一种办法是把 $v(\mathbf{x}_i, \mathbf{x}_j)$ 定义为 $q(\mathbf{x}_i)$ 和 $q(\mathbf{x}_j)$ 的叉乘，就是说，先计算矩阵 $q(\mathbf{x}_i)q(\mathbf{x}_j)^T$ ，然后展平成一个向量。

我们一直把 q 和 v 称为特征，这是计算机视觉领域的惯用名。然而本文所说的特征需要同时依赖输入 \mathbf{x} 和标签 \mathbf{y} 。所以，我们把 q 和 v 称为观测函数，并用于定义CRF的label-observation特征：

$$f_m(\mathbf{y}_i, \mathbf{x}_i) = \mathbf{1}_{\{y_i=m\}} \forall m \in \{0, 1\}$$

$$g_{m,m'}(\mathbf{y}_i, \mathbf{y}_j, \mathbf{x}_i, \mathbf{x}_j) = \mathbf{1}_{\{y_i=m\}} \mathbf{1}_{\{y_j=m'\}} v(\mathbf{x}_i, \mathbf{x}_j) \forall m, m' \in \{0, 1\}$$

$$f(\mathbf{y}_i, \mathbf{x}_i) = \begin{pmatrix} f_0(\mathbf{y}_i, \mathbf{x}_i) \\ f_1(\mathbf{y}_i, \mathbf{x}_i) \end{pmatrix}$$

$$g(\mathbf{y}_i, \mathbf{y}_j, \mathbf{x}_i, \mathbf{x}_j) = \begin{pmatrix} g_{00}(\mathbf{y}_i, \mathbf{y}_j, \mathbf{x}_i, \mathbf{x}_j) \\ g_{01}(\mathbf{y}_i, \mathbf{y}_j, \mathbf{x}_i, \mathbf{x}_j) \\ g_{10}(\mathbf{y}_i, \mathbf{y}_j, \mathbf{x}_i, \mathbf{x}_j) \\ g_{11}(\mathbf{y}_i, \mathbf{y}_j, \mathbf{x}_i, \mathbf{x}_j) \end{pmatrix}$$

使用label-observation特征，可允许每个标签拥有自己独立的权重集。

为了让本例子更具体，这里提供一个已被一些杰出的应用【14,119】所采用的 g 和 v 。前文用的是 q ，估计是笔误。考虑(2.32) MRF中的因子 $\Psi(\mathbf{y}_i, \mathbf{y}_j)$ 。虽然 Ψ 鼓励了一致性，但缺乏灵活性。如果 \mathbf{x}_i 和 \mathbf{x}_j 具有不同的标签，我们期望他们具有不同的灰度，因为不同的物体倾向于拥有不同的色度。因而，当类别分界线的两边具有明显不同的亮度时，我们不会那么惊讶（相比于完全相同的亮度）。遗憾的是， Ψ 对这两种情况使用了相同的差异惩罚，因为特征（potential?）与像素值无关。为了解决这个问题，推荐使用下面的特征：

$$v(\mathbf{x}_i, \mathbf{x}_j) = \exp\{-\beta(\mathbf{x}_i - \mathbf{x}_j)^2\}$$

$$g(\mathbf{y}_i, \mathbf{y}_j, \mathbf{x}_i, \mathbf{x}_j) = \mathbf{1}_{\{y_i \neq y_j\}} v(\mathbf{x}_i, \mathbf{x}_j). \quad (2.33)$$

综合起来，CRF模型是：

$$p(\mathbf{y} | \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp \left\{ \sum_{i=1}^T \theta^T f(\mathbf{y}_i, \mathbf{x}_i) + \sum_{(i,j) \in \mathcal{N}} \lambda^T g(\mathbf{y}_i, \mathbf{y}_j, \mathbf{x}_i, \mathbf{x}_j) \right\}. \quad (2.34)$$

其中, $\alpha \in \mathcal{R}, \theta \in \mathcal{R}^K, \lambda \in \mathcal{R}^{K^2}$, 是模型的参数。前两项与MRF中的两种因子相类似。第一项表示在 \mathbf{x}_i 附近所得到的关于其标签 y_i 的信息。使用(2.23)所描述的 g , 第二项鼓励近邻的标签相同, 但要看他们亮度的差异。

注意, 这是(2.25)所示的通用CRF的一个例子。这里, 我们有3个团模板, 每个对应于(2.34)的一项。

(2.34)与(2.32)之间的不同, 类似于图2.6和2.5的线性链CRF模型的不同: “像素对”之上的特征现在不只与标签有关, 还与图像上反映的特征有关。顺带说明一下, 从(2.32) MRF模型所得到的分布 $p(\mathbf{y}|\mathbf{x})$ 是CRF的一个特例, 即 $\lambda = 0$ 。

有很多方法可以改进这一简单的CRF。第一, 特征函数 q 和 v 可以更加复杂, 如将形状和纹理考虑进来【127】, 或者依赖于全图(而不只是局部的领域)。更进一步, 我们可以使用比网格更复杂的图结构。例如, 可以让因子建立在标签的领域上【49,56】。关于计算机视觉中更深入的CRF及其图结构, 可以参考Nowozin和Lampert【101】

2.7 CRFs的应用

略

2.8关于术语的说明

略

算法总览

接下来的两节中, 我们将讨论CRFs的推断和参数估计。**参数估计Parameter estimation**是要找到一组参数 θ , 使得分布 $p(\mathbf{y}|\mathbf{x}, \theta)$ 与一组输入输出均已知的训练样本 $D = \{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^N$ 相匹配。我们希望, 给定任何一个输入样本 $\mathbf{x}^{(i)}$, 从模型推断出的关于输出的分布 $p(\mathbf{y}|\mathbf{x}^{(i)}, \theta)$, 能“像是”从训练数据中得来的真实的输出 $\mathbf{y}^{(i)}$ 。

要量化地来理解这一点, 可考虑模型中定义的特征函数。考虑线性链CRF。我们希望, 随机地从模型中选择一个输入序列 \mathbf{x} , 然后从 $p(\mathbf{y}|\mathbf{x}, \theta)$ 中采样 \mathbf{y} , 触发特征 $f_k(y_t, y_{t-1}, \mathbf{x}_t)$ 的概率, 能与训练数据中 f_k 发生的概率相等。正式地, 要求 f_k 满足:

$$\sum_{i=1}^T \sum_{t=1}^T f_k(y_t^{(i)}) = \sum_{i=1}^N \sum_{t=1}^T \sum_{y, y'} f_k(y, y', \mathbf{x}_t^{(i)}) p(y_t = y, y_{t-1} = y' | \mathbf{x}^{(i)}).$$

重要的是, 这一方程组可被看成某个关于参数的目标函数的梯度。这一点是很重要的, 因为当我们拥有这一目标函数之后, 可以用标准的数值方法来优化它。拥有这一特性的目标函数是如下的似然

$$l(\theta) = p(\mathbf{y}^{(i)} | \mathbf{x}^{(i)}, \theta) = \sum_{i=1}^N \sum_{t=1}^T \sum_{k=1}^K \theta_k f_k(y_t^{(i)}, y_{t-1}^{(i)}, \mathbf{x}_t^{(i)}) - \sum_{i=1}^N \log Z(\mathbf{x}^{(i)}),$$

这是训练样本在模型意义下的概率, 是关于参数的函数。

训练CRFs的标准方法是最大化似然, 即寻找 $\hat{\theta}_{ML} = \sup_{\theta} l(\theta)$ 。就是说, $\hat{\theta}_{ML}$ 是最有可能产生训练数据的参数。可将似然对其参数求偏导数并置为0, 即得前面讨论的方程组。这恰恰会产生我们刚说的, 关于特征的期望。

尽管我们只讨论了线性链CRF的极大似然，同样的想法也适用于通用CRFs。在通用CRFs中，不用链条邻域间的变量的边缘分布 $p(y_t, y_{t-1} | \mathbf{x}, \theta)$ ，而是通用图模型中一个因子 a 上的所有变量 Y_a 的边缘分布 $p(y_a | \mathbf{x}, \theta)$ 。

参数估计需要计算上述的边缘分布，在计算上是个大挑战。这是**概率推断probabilistic inference**的任务。一般来说，所谓推断，是要在给定输入 \mathbf{x} 和参数 θ 的条件下，计算关于输出 \mathbf{y} 的预测。我们需要关注关于推断的两个特别重要的任务：

- 计算输出变量的子集 Y_a 上的边缘分布 $p(\mathbf{y} | \mathbf{x}, \theta)$ 。 Y_a 一般要么包含单一的变量，要么是与某个因子连接的所有变量。关于这一问题，一般是作为计算归一化函数 $Z(\mathbf{x})$ 的副产品来计算。

- 计算输出 $\mathbf{y}^* = \arg \max_{\mathbf{y}} p(\mathbf{y} | \mathbf{x}, \theta)$ ，即关于输入 \mathbf{x} 的最可能的输出。

边缘分布 $p(y_a | \mathbf{x}, \theta)$ 与归一化函数 $Z(\mathbf{x})$ 通常用于参数估计。有些参数估计方法，如用limited memory BFGS来优化极大似然时，同时需要边缘分布和归一化函数。也有参数方法，如随机梯度下降法，只需要边缘分布。所谓的Viterbi \mathbf{y}^* ，用于给某个未在训练中出现的输入赋予一组标签。

这一推断任务，可使用标准的图模型方法解决。对于树图模型，可精确地计算这些量，而对于一般的图却只能做近似估计。

接下来的两节将讨论推断和参数估计，包括线性链和通用CRFs。在第4节，我们讨论推断方法，包括对树图的精确方法，以及对一般模型的近似估计方法。从某种角度看，因为CRF是一种无向图模型，使得标准的图模型方法也能适用，但我们关注那些最常用于CRF的近似方法。在第5节，我们讨论参数估计。虽然极大似然理解起来很简单，但计算量却非常大。我们不仅描述以近似推断为基础的极大似然方法，还包括一些其他的近似训练方法，用于增加样本量和模型复杂度的伸缩性。

推断

推断的效率对CRFs至关重要，无论是对训练还是预测。有两个关于推断的任务。一是给定新的输入 \mathbf{x} 后，要预测最可能的输出 $\mathbf{y}^* = \arg \max_{\mathbf{y}} p(\mathbf{y} | \mathbf{x})$ 。二是，正如第5节所述，参数估计时所需的边缘分布，如单个节点的 $p(y_t | \mathbf{x})$ 和连接的 $p(y_t, y_{t-1} | \mathbf{x})$ 。这两个任务可被看成two different smirings下的同一操作。就是说，把边缘概率问题改成求最大值问题，我们只需简单地把求和运算变成求最大运算。

对于离散的情况，可通过穷举的办法计算边缘概率，然而所需的计算时间会因 Y 的尺寸而指数爆炸。实际上，对于通用图来说，关于推断的两个问题都是困难的，因为任何命题可满足性问题propositional satisfiability problem都可以轻易地用因子图来表示。

可快速而精确地解线性链CRFs，其方法是HMMs的动态规划算法的变体。在第4.1节，我们从计算边缘分布的forward-backward算法以及计算最可能赋值的Viterbi算法开始。这些算法那是通用的置信传播算法belief propagation algorithm在树图模型（4.2.2）上的特例。对于更复杂的模型，需要近似的推断算法。

可以说，CRF的推断问题与一般的图模型并无差别，因而一般图模型的推断算法也都适用，如一些教材【57,59】所言。然而关于CRFs，我们需要时刻注意两点。第一，在参数估计（5.1.1）时需要反复执行推断任务，非常耗时，因而我们希望能计算效率和准确性之间做些权衡。第二，若采用了近似推断，那可能会带来推断过程与训练过程之间复杂的相互作用。我们把这一议题延后至第5节，因为我们将在那里讨论参数估计，然而有必要在这里提出这个问题，因为它严重影响着对推断算法的选择。

4.1线性链CRFs

这一节，我们简要介绍HMMs的标准推断算法——前向后向以及Viterbi算法，以及如何将它们应用在线性链CRFs上。Rabiner的【111】是一份关于这些算法在HMM上的研究。所有这些算法都只是第4.2.2节将要描述的置信传播算法的特例。然而，我们仍将详细讨论这一在线性链上的特例，因为它能让后面的讨论更具体，也因为自身在工程上就很有用。

首先，我们引入一些记号，能简化接下来的前向后向递归（forward backward recursion）。一个HMM可以写成 $Z=1$ 的因子图 $p(\mathbf{y}, \mathbf{x}) = \prod_t \Psi_t(\mathbf{y}_t, \mathbf{y}_{t-1}, \mathbf{x}_t)$ ，而因子被定义为

$$\Psi_t(j, t, \mathbf{x}) \stackrel{\text{def}}{=} p(\mathbf{y}_t = j | \mathbf{y}_{t-1} = i) p(\mathbf{x}_t = \mathbf{x} | \mathbf{y}_t = j). \quad (4.1)$$

如果把这个HMM看成带权重的有限状态机，那么 $\Psi_t(j, i, \mathbf{x})$ 就是当观测为 \mathbf{x} 时，从状态 i 变成 j 的权重。

现在我们来研究HMM的前向算法，这是用来计算观测值的概率 $p(\mathbf{x})$ 的。前向后向算法背后的思想是，首先把 $p(\mathbf{x}) = \sum_{\mathbf{y}} p(\mathbf{x}, \mathbf{y})$ 的求和运算，按照如下的方式重写

$$\begin{aligned} p(\mathbf{x}) &= \sum_{\mathbf{y}} \prod_{t=1}^T \Psi_t(\mathbf{y}_t, \mathbf{y}_{t-1}, \mathbf{x}_t) \\ &= \sum_{\mathbf{y}_T} \sum_{\mathbf{y}_{T-1}} \Psi_T(\mathbf{y}_T, \mathbf{y}_{T-1}, \mathbf{x}_T) \sum_{\mathbf{y}_{T-2}} \Psi_{T-1}(\mathbf{y}_{T-1}, \mathbf{y}_{T-2}, \mathbf{x}_{T-1}) \sum_{\mathbf{y}_{T-3}} \cdots \end{aligned} \quad (4.3)$$

现在我们可以看到，在进行外部的求和运算时，其内部的求和运算要被反复调用。因此，我们可以把里面的保存起来，从而爆炸式地减少了计算量。

这导致了所谓的前向变量 α_t ，是大小为 M 的向量（ M 是状态的数量），用来保存求和的中间结果。其定义为：

$$\begin{aligned} \alpha_t(j) &\stackrel{\text{def}}{=} p(\mathbf{x}_{\langle 1 \dots t \rangle}, \mathbf{y}_t = j) \\ &= \sum_{\mathbf{y}_{\langle 1 \dots t-1 \rangle}} \Psi_t(j, \mathbf{y}_{t-1}, \mathbf{x}_t) \prod_{t'=1}^{t-1} \Psi_{t'}(\mathbf{y}_{t'}, \mathbf{y}_{t'-1}, \mathbf{x}_{t'}), \end{aligned} \quad (4.5)$$

其中，求和运算的下标 $\mathbf{y}_{1 \dots t-1}$ ，表示要覆盖 $\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_{t-1}$ 的所有可能值。这一 α 可以通过递归的方式计算

$$\alpha_t(j) = \sum_{i \in S} \Psi_t(j, i, \mathbf{x}_t) \alpha_{t-1}(i), \quad (4.6)$$

而初始变量为 $\alpha_1(j) = \Psi_1(j, \mathbf{y}_0, \mathbf{x}_1)$ 。（回忆（2.10），知道 \mathbf{y}_0 是HMM的固定的初始值）。反复地递归（4.6）式，可知 $p(\mathbf{x}) = \sum_{\mathbf{y}_T} \alpha_T(\mathbf{y}_T)$ 。正式地证明应该需要数学归纳法。

后向递归于此相同，除了在（4.3）中把求和的顺序颠倒过来。所得的定义为

$$\begin{aligned} \beta_t(i) &\stackrel{\text{def}}{=} p(\mathbf{x}_{\langle t+1 \dots T \rangle} | \mathbf{y}_t = i) \\ &= \sum_{\mathbf{y}_{\langle t+1 \dots T \rangle}} \prod_{t'=t+1}^T \Psi_{t'}(\mathbf{y}_{t'}, \mathbf{y}_{t'-1}, \mathbf{x}_{t'}), \end{aligned} \quad (4.8)$$

而其递归为

$$\beta_t(i) = \sum_{j \in S} \Psi_{t+1}(j, i, \mathbf{x}_{t+1}) \beta_{t+1}(j), \quad (4.9)$$

其中，初始量为 $\beta_T(i) = 1$ 。与前向类似，我们也可以使用后向变量来计算

$$p(\mathbf{x}) = \beta_0(\mathbf{y}_0) \stackrel{\text{def}}{=} \sum_{\mathbf{y}_1} \Psi_1(\mathbf{y}_1, \mathbf{y}_0, \mathbf{x}_1) \beta_1(\mathbf{y}_1).$$

要计算边缘分布 $p(y_{t-1}, y_t | \mathbf{x})$ (在参数估计时需要), 我们要把前向和后向的结果综合起来。这可以从概率或因子分解的角度来看。首先, 从概率的角度来看, 我们写成

$$\begin{aligned} p(y_{t-1}, y_t | \mathbf{x}) &= \frac{p(\mathbf{x} | y_{t-1}, y_t) p(y_{t-1}, y_t)}{p(\mathbf{x})} \\ &= \frac{p(\mathbf{x}_{<1 \dots t-1>}, y_{t-1}) p(y_t | y_{t-1}) p(x_t | y_t) p(\mathbf{x}_{<t+1 \dots T>} | y_t)}{p(\mathbf{x})} \\ &= \frac{1}{p(\mathbf{x})} \alpha_{t-1} \Psi_t(y_t, y_{t-1}, x_t) \beta_t(y_t), \quad (4.12) \end{aligned}$$

在上式的第二行中, 我们基于如下的事实: 给定 y_t, y_{t-1} 后, $\mathbf{x}_{<1 \dots t-1>}$ 与 $\mathbf{x}_{<t+1 \dots T>}$ 以及 x_t 无关。同样的, 从因子分解的角度看, 我们利用分配率得

$$\begin{aligned} p(y_{t-1}, y_t | \mathbf{x}) &= \frac{1}{p(\mathbf{x})} \Psi_t(y_t, y_{t-1}, x_t) \\ &\quad \times \left(\sum_{\mathbf{y}_{<1 \dots t-2>}} \prod_{t'=1}^{t-1} \Psi_{t'}(y_{t'}, y_{t'-1}, x_{t'}) \right) \\ &\quad \times \left(\sum_{\mathbf{y}_{<t+1 \dots T>}} \prod_{t'=t+1}^T \Psi_{t'}(y_{t'}, y_{t'-1}, x_{t'}) \right) \end{aligned}$$

然后, 通过带入 α 与 β 的定义, 我们得到与前文一样的结果, 即:

$$p(y_{t-1}, y_t | \mathbf{x}) = \frac{1}{p(\mathbf{x})} \alpha_{t-1}(y_{t-1}) \Psi_t(y_t, y_{t-1}, x_t) \beta_t(y_t). \quad (4.14)$$

而 $1/p(\mathbf{x})$ 相当于分布的归一化常数。我们通过 $p(\mathbf{x}) = \beta_0(y_0)$ 或 $p(\mathbf{x}) = \sum_{i \in S} \alpha_T(i)$ 来计算它。

总的来说, 前向后向算法就是: 首先用(4.6)计算每个 α_t , 然后用(4.9)计算每个 β_t , 然后用(4.14)计算边缘分布。

最后, 如要计算最可能的输出 $\mathbf{y}^* = \arg \max_{\mathbf{y}} p(\mathbf{y} | \mathbf{x})$, 我们发现之前在 (4.3) 中使用的技巧仍然有效。这带来了Viterbi算法。与前向变量 α 像类似的变量为

$$\delta_t(j) \stackrel{def}{=} \max_{\mathbf{y}_{<1 \dots t-1>}} \Psi_t(j, y_{t-1}, x_t) \prod_{t'=1}^{t-1} \Psi_{t'}(y_{t'}, y_{t'-1}, x_{t'}). \quad (4.15)$$

而这可以通过类似的递归来计算:

$$\delta_t(j) = \max_{i \in S} \Psi_t(j, i, x_t) \delta_{t-1}(i). \quad (4.16)$$

δ 被计算出来之后, 最可能的输出可通过如下的后向递归计算:

$$\begin{aligned} y_T^* &= \arg \max_{i \in S} \delta_T(i) \\ y_t^* &= \arg \max_{i \in S} \Psi_t(y_{t+1}^*, i, x_{t+1}) \delta_t(i) \text{ for } t < T \end{aligned}$$

对 δ_t 和 y_t^* 的递归, 构成了Viterbi算法。

现在, 我们已经讲述了HMMs的前向后向和Viterbi算法。将其扩展到线性链CRFs是直接的。线性链CRFs的前向后向算法与HMMs的一样, 只是转移权重 $\Psi_t(j, i, x_t)$ 的定义需要改变。我们注意到, (2.18) 的CRF模型可以重写为

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^T \Psi_t(y_t, y_{t-1}, \mathbf{x}_t), \quad (4.17)$$

其中

$$\Psi_t(y_t, y_{t-1}, \mathbf{x}_t) = \exp \left\{ \sum_k \theta_k f_k(y_t, y_{t-1}, \mathbf{x}_t) \right\}. \quad (4.18)$$

使用这里的定义，前向递归（4.6）、后向递归（4.9）以及Viterbi递归（4.16）可不经修改就用于线性链CRFs，只是其含义有所改变。在CRF中， $\alpha_t(j) = p(\mathbf{x}_{<1..t>}, y_t = j)$ 不再具有概率的含义，而需从因子分解的角度来理解。就是说，我们需根据（4.5）定义 α ，（4.8）定义 β ，（4.15）定义 δ 。同样地，前向后向递归的结果现在变成了 $Z(\mathbf{x})$ ，而不是 $p(\mathbf{x})$ ，且 $Z(\mathbf{x}) = \beta_0(y_0)$ ， $Z(\mathbf{x}) = \sum_{i \in S} \alpha_T(i)$ 。

关于边缘分布，方程（4.14）仍然有效，只是需用 $Z(\mathbf{x})$ 替换 $p(\mathbf{x})$ ，即

$$p(y_{t-1}, y_t | \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \alpha_{t-1}(y_{t-1}) \Psi_t(y_t, y_{t-1}, \mathbf{x}_t) \beta_t(y_t). \quad (4.19)$$

$$p(y_t | \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \alpha_t(y_t) \beta_t(y_t). \quad (4.20)$$

我们补充三个可被上面的算法直接解的推断任务。一，如果我们想从后验概率 $p(\mathbf{y}|\mathbf{x})$ 中采样 \mathbf{y} ，可使用前向算法+后向采样过程，就如在HMMs中的做法一样。二，如果不想只找到唯一的最可能输出 $\arg \max_{\mathbf{y}} p(\mathbf{y}|\mathbf{x})$ ，而是前k个可能输出，我们可以使用HMMs的标准算法[129]。最后，有时候我们要计算一组节点 $S \subset [1, 2, \dots, T]$ （不一定是连接在一起的）的边缘概率 $p(\mathbf{y}_S | \mathbf{x})$ 。例如，这可用于评估模型在部分输入上的性能。这一边缘概率可使用Culotta和McCallum【30】描述的带约束前向后向算法来解。

4.2图模型的推断

通用图模型的精确推断算法也有不少。最糟糕的时候，它们需要指数级别的耗时，但在工程实践中仍不失为有效的方法。最流行的精确算法是联合树算法。它连续地把变量组合起来，使整个图变成一棵树。一旦这棵等效树被建立了起来，我们就可以用已有的、针对树的精确推断算法了。然而对于某些复杂的图，联合树算法需要做规模巨大的聚类，使它在最糟糕时仍需要指数级别的计算时间。关于精确算法的更多细节，请参考Koller和Friedman【57】。

由于精确推断的复杂性，大量的努力朝向了近似推断算法。有两类近似算法获得了最多的关注：蒙特卡洛算法和变分法。蒙特卡洛算法是统计算法，尝试从分布中近似地产生样本。变分法把推断问题转变成最优化问题，然后尝试找到边缘概率的最近的估计。一般来说，只要给与足够的时间，蒙特卡洛算法总能无偏地从分布中进行采样，但在实践中一般无法知道何时做到了这一点。变分法非常快速，但倾向于偏差，就是说，它天生具有一些误差，且哪怕拥有足够的计算时间也不能消除。尽管如此，变分法对CRFs是有用的。因为参数估计需要多次执行推断，所以快速的推断对于高效地训练十分关键。关于蒙特卡洛算法，可参考Robert和Casella【116】。对于变分方法，可参考Wainwright和Jordan【150】。

从两个方面，本节的内容特别针对CRFs，但也适用于从某些因子图得来的任何分布，不管它是联合分布 $p(\mathbf{y})$ ，还是像CRFs的条件分布 $p(\mathbf{y}|\mathbf{x})$ 。为了强调这一点，也为了简化记号，我们去掉了 \mathbf{x} 的依赖，而只讨论联合分布 $p(\mathbf{y})$ 的推断。该分布从某些因子图 $G = (V, F)$ 而来，即

$$p(\mathbf{y}) = Z^{-1} \prod_{a \in F} \Psi_a(\mathbf{y}_a).$$

若想将这里的讨论用于CRFs，只需用 $\Psi_a(\mathbf{y}_a, \mathbf{x}_a)$ 替换上面的 $\Psi_a(\mathbf{y}_a)$ ，同时将 Z 换成 $p(\mathbf{y})$ 。这样就可以对 \mathbf{x} 依赖了。这不仅是记号的问题，还会影响到具体的实践：推断算法可实现成适用于一般的因子图，而无需知道它是无向的联合分布 $p(\mathbf{y})$ ，还是CRF的 $p(\mathbf{y}|\mathbf{x})$ ，或甚至是有向的图模型。

在本节的剩余部分，我们扼要地介绍近似推断算法的两个例子，分别来自两个大类。我们不能在这里包含所有的近似推断算法。相反地，我们的目标是想强调近似推断算法给CRF的训练带来的一般性问题。本节中，我们关注推断算法本身，而在第5节介绍它们在CRFs中的应用。

4.2.1 马尔科夫链蒙特卡洛

当前最流行的复杂模型的蒙特卡洛方法是马尔科夫链蒙特卡洛（MCMC）【116】。它不去直接估计边缘概率 $p(\mathbf{y}_s)$ ，而是从联合分布 $p(\mathbf{y})$ 中产生估计样本。MCMC方法通过构造一个马尔科夫链，使其状态空间与 \mathbf{Y} 相同，小心地用该链做仿真足够长的时间，使得链的状态分布接近 $p(\mathbf{y}_s)$ 。假如函数 $f(\mathbf{y})$ 服从分布 $p(\mathbf{y})$ ，而我们想估计它的期望。给定MCMC方法的马尔科夫链的一组样本 $\mathbf{y}^1, \mathbf{y}^2, \dots, \mathbf{y}^M$ ，我们可以通过下式来估计这个期望

$$\sum_{\mathbf{y}} p(\mathbf{y}) f(\mathbf{y}) \approx \frac{1}{M} \sum_{j=1}^M f(\mathbf{y}^j). \quad (4.21)$$

下一节我们将发现，CRF的训练需要这一形式的期望。MCMC方法的一个简单例子是Gibbs采样。在Gibbs算法的每个迭代周期里，每个变量独立地被重采样而保持其他变量不变。假如我们已经在第 j 次采样获得了样本 \mathbf{y}^j ，那么需要产生下一个样本 \mathbf{y}^{j+1} (1)取 $\mathbf{y}^{j+1} = \mathbf{y}^j$ 。(2)对每个 $s \in S$ ，重采样 \mathbf{y}_s 。从分布 $p(\mathbf{y}_s | \mathbf{y}_{\setminus s}, \mathbf{x})$ 中采样 \mathbf{y}_s^{j+1} 。

(3)返回 \mathbf{y}^{j+1} 作为结果。

回忆一下地2.1.1节， $\sum_{\mathbf{y} \setminus \mathbf{y}_s}$ 表示求和运算要遍历 \mathbf{y} 的所有可能取值，除了 \mathbf{y}_s 取值 \mathbf{y}_s 。

上面的过程定义了一个马尔科夫链，可用于近似式（4.21）的期望。对于通用因子图，条件概率可按照下式计算

$$p(\mathbf{y}_s | \mathbf{y}_{\setminus s}) = \kappa \prod_{a \in F} \Psi_a(\mathbf{y}_a), \quad (4.22)$$

其中， κ 是归一化常量。（下文中， κ 是一般意义的归一化常量，不要求在不同的式子中取相同的值）。式（4.22）的 κ 要比联合概率 $p(\mathbf{y}|\mathbf{x})$ 容易计算得多，因为它只需遍历 \mathbf{y}_s 的所有可能取值，而不是整个 \mathbf{y} 向量。

Gibbs的一个主要优点是它易于实现。实际上，像BUGS这种软件包允许以图模型作为输入，自动编译一个Gibbs取样器用于近似【78】。Gibbs的主要缺点是，当 $p(\mathbf{y})$ 存在强相关时，Gibbs性能较差，而这在序列形式的数据中很常见。关于性能较差，我们的意思是，它需要很多次迭代才能让马尔科夫链的样本接近于所需的分布 $p(\mathbf{y})$ 。

有大量的关于MCMC算法的文献。Robert和Casella的教材【116】提供了一个综述。然而，CRFs领域却不常用MCMC算法。原因可能正如我们说过的，极大似然方法做参数估计时，需要计算边缘概率很多次。不考虑复杂的策略，那么每次梯度下降时，都要为每一个参数集的每一个训练样本运行一次MCMC链。而MCMC链自身就需要千把次迭代才能收敛，使得这种方法在计算上难以实行。读者可能想到了一些解决办法，比如不等马尔科夫链收敛就返回（参考地5.4.3节）。

4.2.2 置信传播

置信传播belief propagation (BP) 是一种重要的变分推断算法variational inference algorithm 翻译成变分推断算法似乎不合适。这里似乎只是“变体”的含义，把推断问题变成优化问题。我们将在本节解释它。BP同时还是线性链CRFs的精确推断算法的一般化。

假如因子图 $G = (V, F)$ 是一棵树，而我们希望计算变量 Y_s 的边缘概率。BP 背后的思想在于，它认为每个与 Y_s 相连的因子按照乘法来提供边缘概率，我们称它们为**信息message**。因为图是一棵树，所以每个信息可被单独计算。更正式地，对每个因子 $a \in N(s)$ ，记 $G_a = (V_a, F_a)$ 为包含 Y_s 、 Ψ_a 以其 Ψ_a 全部"上游"的 G 的子图。所谓的"上游"，我们指 V_a 包含了所有被 Ψ_a 隔开，从而与 Y_s 分离的变量，以及从而与 F_a 分离的因子。参见图4.1。对于每个 $a \in N(s)$ ，每个 $V_a \setminus Y_s$ 相互独立，因为 G 是一棵树。对 F_a 亦如此。这意味着，我们可以把边缘概率所需的求和运算划分成多个独立的子问题相乘，即

$$p(y_s) \propto \sum_{\mathbf{y} \setminus y_s} \prod_{a \in F} \Psi_a(\mathbf{y}_a) \quad (4.21)$$

$$= \sum_{\mathbf{y} \setminus y_s} \prod_{a \in N(s)} \prod_{\Psi_b \in F_a} \Psi_b(\mathbf{y}_b) \quad (4.24)$$

$$= \prod_{a \in N(s)} \sum_{\mathbf{y}_{V_a \setminus y_s}} \prod_{\Psi_b \in F_a} \Psi_b(\mathbf{y}_b). \quad (4.25)$$

虽然上面的记号不够明显，但需注意变量 y_s 包含在每个 y_a 中，所以它在 (4.25) 的两边都出现了。

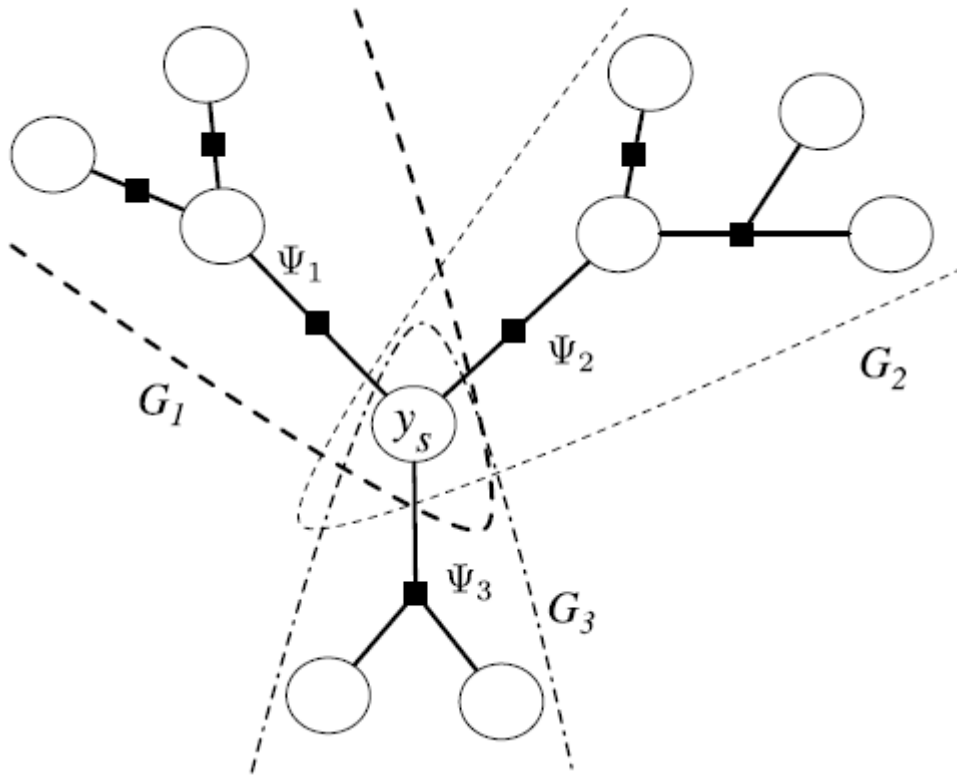


图4.1 树形图的边缘分布是如何被划分的。这一划分被置信传播算法 (4.2.2) 所用。

把上式的每个因子记为 m_{as} ，那么

$$m_{as}(y_s) = \sum_{\mathbf{y}_{V_a \setminus y_s}} \prod_{\Psi_b \in F_a} \Psi_b(\mathbf{y}_b). \quad (4.26)$$

每个 m_{as} 正是从子图 G_a 过来的关于变量的 y_s 的边缘分布。 y_s 在全图上的边缘分布，正是每个子图上的边缘分布的乘积。这就好比 $m_{as}(y_s)$ 是因子 a 传给 Y_s 的**信息message**，而这一信息汇合 a 上游的全部作用。同样地，我们可定义从变量到因子的信息：

$$m_{sa}(y_s) = \sum_{\mathbf{y}_{V_a}} \prod_{\Psi_b \in F_s} \Psi_b(\mathbf{y}_b). \quad (4.27)$$

然后考虑 (4.25) 式，我们知道边缘概率 $p(y_s)$ 与所有到达 Y_s 的消息的乘积成比例。同样地，因子的边缘概率可计算为：

$$p(y_a) \propto \Psi_a(y_a) \prod_{s \in N(a)} m_{sa}(y_a). \quad (4.28)$$

直接按照 (4.26) 式计算消息还不行，因为需要遍历 y_{V_a} 的所有可能取值来进行求和运算，而有时 V_a 是很大的集合。幸运的是，消息也可被写成递归的形式，从而只需局部的求和。递归形式是：

$$\begin{aligned} m_{as}(y_s) &= \sum_{y_a \setminus y_s} \Psi_a(y_a) \prod_{t \in a \setminus s} m_{ta}(y_t) \\ m_{sa}(y_s) &= \prod_{b \in N(s) \setminus a} m_{bs}(y_s). \end{aligned} \quad (4.29)$$

通过依次带入，可知这一递归形式符合 m 的定义，也可通过数学归纳法证明。对于树形图，有可能通过合理的安排，使得每个消息被发送之前已收到它所依赖的上游消息，比如首先从根开始发送消息。这就是置信传播算法【103】。

除了计算单个变量的边缘概率，我们也希望计算银子的概率 $p(y_a)$ 和联合概率 $p(y)$ 。（回忆一下，后面这个任务是困难的，因为要计算归一化函数 $\log Z$ ）。首先，我们可以像单个变量那样解构，从而计算因子的边缘概率，得到

$$p(y_a) = \kappa \Psi_a(y_a) \prod_{s \in N(a)} m_{sa}(y_s). \quad (4.30)$$

其中， κ 是归一化常量。实际上，这一想法适用于所有相连接的变量集——无须属于同一个因子——虽然当这一集合很大时，计算 κ 仍是不实际的。

BP也可用来计算归一化常数 Z 。可被传播算法直接计算得到，就像4.1节的前向后向算法。除此之外，也可在算法的末尾求得近似的边缘概率时去计算 Z 。对于树形结构的分布 $p(y)$ ，可以发现联合分布总是按照下面的方式分解的：

$$p(y) = \prod_{s \in V} p(y_s) \prod_a \frac{p(y_a)}{\prod_{t \in a} p(y_t)}. \quad (4.31)$$

例如，对于线性链，这变成

$$p(y) = \prod_{t=1}^T p(y_t) \prod_{t=1}^T \frac{p(y_t, y_{t-1})}{p(y_t)p(y_{t-1})}, \quad (4.32)$$

通过消元、移项等操作，上式不过是我们熟悉的 $p(y) = \prod_t p(y_t, y_{t-1})$ 的另一种写法而已。利用这一点，我们可以利用每个变量和因子的边缘概率计算 $p(y)$ 。也可得到 $Z = p(y)^{-1} \prod_{a \in F} \Psi_a(y_a)$ 。

如果 G 是一棵树，置信传播算法精确地计算得到边缘分布。实际上，如果 G 是线性链，BP退化成前向后向算法（4.1节）。为了说明这一点，请参考图4.2。该图展示了带3个节点的线性链，附带有我们刚描述的BP消息。为了与前向后向对应起来，我们在第4.1节记做 α_2 的前向信息对应于消息 m_{A2} 于 m_{C2} 的乘积（图中深灰色箭头）。反向消息 β_2 与消息 m_{B2} 相对应（图中浅灰色箭头）。实际上，式 (4.30) 对 $p(y_a)$ 的解构是线性链 (4.14) 式的一般化。

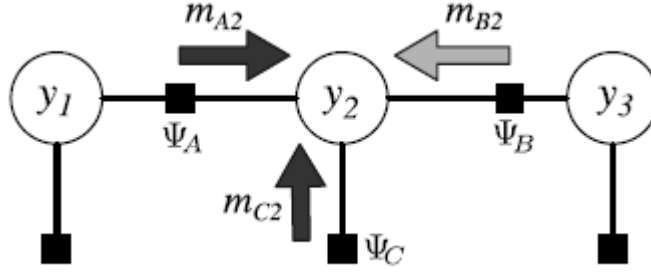


图4.2 前向后向算法与置信传播算法在线性链图中的一致性。具体请参考正文

如果 G 不是一棵树，(4.29)式对消息的更新不一定返回精确的边缘概率，也不能保证收敛性，但我们仍可迭代更新以求某个稳定点。这一过程叫做**循环置信传播loopy belief propagation**。为了强调这是求近似的过程，我们把循环BP得到的边缘概率称为置信 (beliefs)，而不是边缘概率，并记做 $q(y_s)$ 。

现在，仍需要确定更新消息的顺序。在树形结构中，任何传播顺序都会收敛于正确的边缘分布，对循环传播却不行。甚至，消息更新的顺序不仅会影响最终的结果，还会影响算法是否收敛。实践中表现良好的一个简单选择是随机地更新消息。例如，随机地将因子排序，然后对每个因子依次按照(4.29)发送再接收消息。然而，更复杂的策略【35,135,152】也可以是有效的。

奇怪的是，循环BP也可被看成推算的变分法。就是说，存在置信的目标函数可被BP过程近似地最小化。我们在下文给出这一论点的综述，而更多的细节请参考文章【150,158】。

一个变分算法背后的一般思想是：

(1) 定义一组可控的近似 \mathcal{Q} ，以及对于 $q \in \mathcal{Q}$ 定义一个目标函数 $\mathcal{O}(q)$ 。每个 q 可以是边缘概率易于计算的分布，也可以直接是一组边缘分布的近似。如果是后者，那么近似的边缘概率常被称为**伪边缘概率 pseudomarginals**，因为它们不必是 \mathbf{y} 的联合分布的任何边缘概率。函数 \mathcal{O} 必须设计成是对 $q \in \mathcal{Q}$ 与 p 的近似程度的测量。

(2) 找到“最近”的近似 $q^* = \min_{q \in \mathcal{Q}} \mathcal{O}(q)$ 。

(3) 用 q^* 来近似 p 的边缘概率。

例如，我们取 \mathcal{Q} 为 \mathbf{y} 的所有可能的分布的集合，并取目标函数为：

$$\begin{aligned} \mathcal{O}(q) &= KL(q||p) - \log Z \quad (4.33) \\ &= -H(q) - \sum_a \sum_{\mathbf{y}_a} q(\mathbf{y}_a) \log \Psi_a(\mathbf{y}_a), \quad (4.34) \end{aligned}$$

一旦通过优化获得了 q^* ，我们可以用 q^* 的边缘概率来近似 p 的。实际上，这个问题的解是 $q^* = p$ 且 $\mathcal{O}(q^*) = -\log Z$ 。因此，解这个变分问题就相当于精确地推断。可以通过改变集合 \mathcal{Q} 来设计推断方法——例如让 q 充分地分解 (fully factorized) 或使用别的目标函数 \mathcal{O} 。例如，平均场方法 (mean field method) 是通过要求 q 充分地分解而提出的，即选择某些 q_s 满足 $q(\mathbf{y}) = \prod_s q_s(y_s)$ ，并找到使式(4.34)的 $\mathcal{O}(q)$ 最大化的 q 。

有了上面的变分法的背景知识，让我们看看如何将置信传播算法放入这个框架。我们做两个近似。首先，我们近似了式(4.34)的难以计算的熵 $H(q)$ 。如果 q 是一棵树，那么气上可精确地写为

$$H_{BETHE}(q) = - \sum_a \sum_{\mathbf{y}} q(\mathbf{y}_a) \log q(\mathbf{y}_a) + \sum_i \sum_{y_i} (d_i - 1) q(y_i) \log q(y_i), \quad (4.35)$$

其中 d_i 是 i 的阶，表示连接到 y_i 上的因子的数量。这是通过把联合分布的树形因子分解式(4.31)带入熵的定义得到的。如果 q 不是一棵树，我们仍然可以把 H_{BETHE} 看成 H 的近似，用于计算精确的变分目标函数 \mathcal{O} 。这带来了Bethe free熵：

$$\mathcal{O}(q) = -H_{BETHE}(q) - \sum_a \sum_{\mathbf{y}_a} q(\mathbf{y}_a) \log \Psi_a(\mathbf{y}_a) \quad (4.36)$$

目标函数 \mathcal{O}_{BETHE} 只通过它的边缘概率依赖于 q ，因而与其在所有可能的分布 q 上寻优，不如在所有的边缘概率向量所构成的空间里寻优。特别低，每个分布 q 有一个配套的置信向量（belief vector） \mathbf{q} ，其元素为 $q_{a;\mathbf{y}_a}$ （对应一个因子 a 以及相关变量 \mathbf{y}_a 的取值）和 $q_{i;y_i}$ （对应每个变量 i 及其取值）。所有可能的置信向量组成的空间，又被称为 marginal polytope 【150】。然而对于棘手的模型，其marginal polytope的结构可能及其复杂。

这给我们带来了第二种变分近似——循环BP。其中，目标函数 \mathcal{O} 是在松弛的marginal polytope上最小化的。松弛是因为它只要求置信（beliefs）在局部一致（locally consistent），就是说，

$$\sum_{\mathbf{y}_a \setminus y_i} q_a(\mathbf{y}_a) = q_i(y_i) \quad \forall a, i \in a. \quad (4.27)$$

从技术的角度讲，如果一组推定的边缘分布满足（4.27）式，并不意味着他们在整体上一致（globally consistent）。即是说，存在一个唯一的联合概率 $q(\mathbf{y})$ 拥有这些边缘概率（that there exists a single joint $q(\mathbf{y})$ that has those marginals 我没能理解这句话）。因此，分布 $q_a(\mathbf{y}_a)$ 又被称为伪边缘概率（pseudomarginal）。

Yedidia 等【157】证明了，在约束（4.37）下， \mathcal{O} 的驻点是循环BP的固定点。所以，我们可以把 \mathcal{O} 看成是，循环BP固定点运算所尝试优化的，目标函数。

这一变分视角让我们对该方法有了新的深入理解，而这是不能单单从信息传递视角所想到的。一个最重要的见解是关于如何用循环BP来近似 $\log Z$ 。因为我们用 $\min_q \mathcal{O}_{BETHE}(q)$ 来近似 $\min_q \mathcal{O}(q)$ ，而 $\min_q \mathcal{O}(q) = \log Z$ ，因而用 $\log Z_{BETHE} = \min_q \mathcal{O}_{BETHE}(q)$ 来近似 $\log Z$ 是合理的。当我们在5.4.2节讨论CRF的参数估计时，这一点就很重要了。

4.3 实现方面的注意点

这一节，我们讲述一些在CRFs推断的实践中尤其重要的技术：稀疏性以及防止数值溢出。

首先，利用模型的稀疏性常常能够加快推断。有两类相关的稀疏性：因子值的稀疏性和特征的稀疏性。首先是关于因子值，记得在线性链时，每次前向更新（4.6）和后向更新（4.9）要被执行 $O(M^2)$ 次。就是说，与标签的数量 M 的二次方有关。相似地在通用CRFs中，如果因子是连接着成对的两个变量，那么一次循环BP的更新也需要 $O(M^2)$ 次。然而在某些模型中可更高效地实现推断，因为存在先验知识，知道不是所有的因子的取值 y_t, y_{t-1} 都是可能的。就是说，对于许多的取值 y_t, y_{t-1} ，因子 $\Psi_t(y_t, y_{t-1})$ 总是0。这时，把消息传递迭代变成稀疏矩阵运算可以节省计算量。

另一种有用的稀疏性是特征向量的稀疏性。回忆一下（2.26），计算一个因子 $\Psi_c(\mathbf{x}_c, \mathbf{y}_c)$ 需要计算参数向量 θ_p 和特征向量 $\mathbf{f}_c\{\mathbf{f}_{pk}(\mathbf{y}_c, \mathbf{x}_c) | \forall p, \forall k\}$ 的内积。一般来说，向量 \mathbf{f}_c 的许多元素是0。例如自然语言处理常常包含单词是否出现作为特征。这时，使用稀疏向量方式可以节省大量的计算因子 Ψ_c 的时间。类似地，我们可以用稀疏性来减少似然梯度的计算时间，如第5节所讨论的。

还有一个可以加快前向后向算法的技巧，就是将某些参数与某些转移（transitions）绑定起来【24】。这减少了模型的转移矩阵的大小，减轻计算量与标签数量的二次方关系。

第二个实现推断时需注意的是如何避免数值溢出。前向后向算法和置信传播的概率值，如 α_t 和 m_{sa} ，通常比数值的精度还小（小于浮点数的数值精度）（例如HMM中的 α_t ，随着 t 以指数的方式趋向于0）。有两个标准的方法来解决这一常见问题。一种方式是将每个 α_t 和 β_t 归一化，从而剔除小的值。这一缩放不会影响对 $Z(\mathbf{x})$ 的计算，因为可以按照 $Z(\mathbf{x}) = p(\mathbf{y}'|\mathbf{x})^{-1} \prod_t (\Psi_t(y'_t, y'_{t+1}, \mathbf{x}_t))$ 来计算，其中 $p(\mathbf{y}'|\mathbf{x})^{-1}$ 是从（4.31）的边缘概率计算来的。然而实际上，【111】描述了更有效的方法，其中的缩放技巧可用于前向后向算法以及循环BP。不管怎么样，它不影响最后的置信值（values of the beliefs）。

防止数值溢出的第二个方法是在对数域完成计算。即是说，前向递归 (4.6) 变成：

$$\log \alpha_t(j) = \bigoplus_{i \in S} (\log \Psi_t(j, i, \mathbf{x}_t) + \log \alpha_{t-1}(i)), \quad (4.38)$$

其中， \bigoplus 表示 $a \bigoplus b = \log(e^a + e^b)$ 。一开始，这似乎不能改进什么，因为数值精度在计算 e^a 和 e^b 时有所损失。然而， \bigoplus 可以计算成：

$$a \bigoplus b = a + \log(1 + e^{b-a}) = b + \log(1 + e^{a-b}), \quad (4.39)$$

当我们选择小一点的指数时，这一运算的数值稳定性要好很多。

初一看，我们喜欢归一化方法胜过对数域方法，因为对数域方法需要 $O(TM^2)$ 次调用耗时的 \log 和 \exp 运算。这对HMMs是对的，但不是CRFs。因为CRFs总归是要调用 \exp 来计算 $\Psi_t(\mathbf{y}_t, \mathbf{y}_{t+1}, \mathbf{x}_t)$ ，哪怕在归一化方法中。因此在CRFs中，调用这些运算不可避免。在最坏的时候，有 TM^2 个这样的 Ψ_t ，因而归一化方法需要调用这些特殊的函数 TM^2 次，与指数域方法一样。然而，有一些特殊的情况，归一化方法可以更快，如当转移特征不依赖于观测时，那么只有 M^2 个不同的 Ψ_t 。