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。

译后的感受：

翻译完5.4节之后，已覆盖到了训练的内容。余下的部分就不翻译了。尽管这本教材对我的帮助极大，但我仍认为，它废话太多了。随着翻译的进行，越来越觉得，大概没有多少人愿意去读它。

争取后面自己写一篇，期望能给读者一些简单明了、或多或少的帮助吧。当然，还是要找个时间，排查一下错别字，同时调整一些句子的结构，使更易于理解。

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$\boldsymbol{x}$，来预测一个随机输出向量$\boldsymbol{y} = y_0, y_1, \ldots, y_T$。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$\boldsymbol{x}$被分解成多个输入特征向量$\{\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T\}$。每个$\boldsymbol{x}_s$ 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$\boldsymbol{x} \rightarrow y_s$，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$p(\boldsymbol{y}, \boldsymbol{x})$。尽管这有一些好处，但存在着重要的局限。不仅是因为输入$\boldsymbol{x}$的维度可能非常大，还因为输入$\boldsymbol{x}$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$p(\boldsymbol{y}|\boldsymbol{x})$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$\boldsymbol{y}$进行建模，一方面能够应付数量庞大的输入特征$\boldsymbol{x}$，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$\boldsymbol{x}$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，以及包含潜藏变量的隐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$p(\boldsymbol{y}|\boldsymbol{x})$的边缘分布，也指计算极大似然$\boldsymbol{y}^* = argmax_y p(\boldsymbol{y}|\boldsymbol{x})$。**学习**是指参数估计过程，就是找到$p(\boldsymbol{y}|\boldsymbol{x})$的参数，使其最大限度地符合一组训练样本$\{\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}\}_{i=1}^N$。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型。

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$\boldsymbol{Y}$上的概率分布。我们通过整数$s \in 1, 2, \ldots |\boldsymbol{Y}|$来对变量进行索引。每个变量$\boldsymbol{Y}_s \in \boldsymbol{Y}$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。$\boldsymbol{Y}$的一次特定的取值记做$\boldsymbol{y}_s$。对于$\boldsymbol{Y}$中的特定变量$\boldsymbol{Y}_s$，$\boldsymbol{y}_s$包含了对它的赋值，记做$\boldsymbol{y}_s$。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$\boldsymbol{y}_s$，我们用求和符号$\sum_{\boldsymbol{y} \backslash y_s}$来表示：在$\boldsymbol{y}$的全部取值中，那些$\boldsymbol{Y}_s = \boldsymbol{y}_s$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$\Psi(\boldsymbol{y}_a)$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$\Psi(\boldsymbol{y}_a)$只依赖于部分变量$\boldsymbol{Y}_a \in \boldsymbol{Y}$。$\Psi(\boldsymbol{y}_a)$是一个非负数，可以被看成$\boldsymbol{y}_a$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$\boldsymbol{Y}_a$要比完整的集合$\boldsymbol{Y}$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$\boldsymbol{Y}$的子集$\{\boldsymbol{Y}_a\}_{a=1}^A$的集合，一个无向图模型是所有可以写成下式的分布：

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

其中，对于任意的因子$\mathcal{F} = \{\Psi(\boldsymbol{y}_a)\}$，及其对应的所有可能的$\boldsymbol{y}_a$，都有$\Psi(\boldsymbol{y}_a) \geq 0$。（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：

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

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$\boldsymbol{y}$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，可以建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$V = \{1, 2, \ldots, |Y|\}$索引了模型中的全部随机变量，另一组节点的集合$F = \{1, 2, \ldots, A\}$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一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$\Psi(\boldsymbol{y}_a)$，使得p可以写成：

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

时，一个分布$p(\boldsymbol{y})$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）定义的无向图模型，对应着所有根据G进行分解所得的分布。

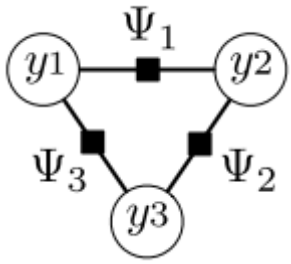

图2.1 带3个变量的因子图

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$\boldsymbol{y} = (y_1, y_2, y_3)$，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$V = \{1, 2, \ldots, |Y|\}$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$p(y_s | \boldsymbol{y}_{N(S)})$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。

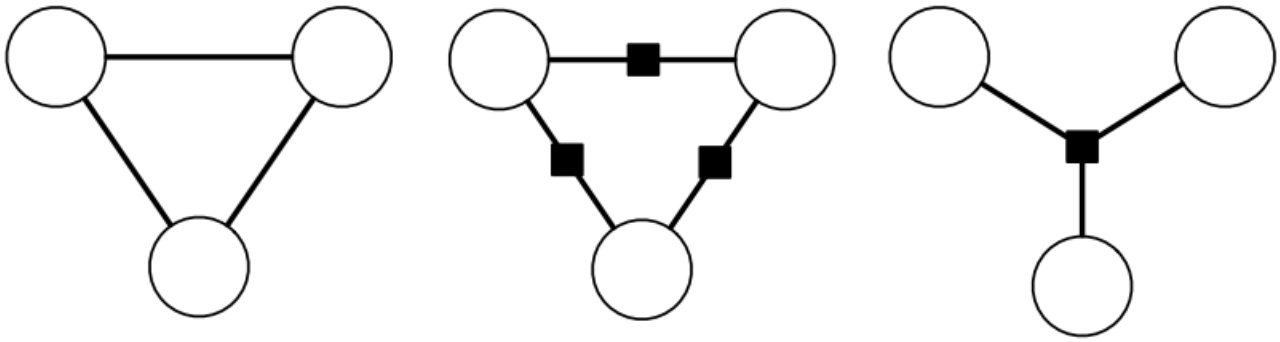

图2.2带有歧义的马尔科夫网（左）。右边的两种分解都有可能与左图对应。

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：

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

我们称$p(y_s|\boldsymbol{y}_{\pi(s)})$为**局部条件分布（localconditionaldistributions）**。注意，对于没有父节点的变量，$\pi(s)$可以是空的。这时，$p(y_s|\boldsymbol{y}_{\pi(s)})$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（朴树贝叶斯和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$\boldsymbol{y}$以怎样的概率"生成"输入特征$\boldsymbol{x}$。**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$\boldsymbol{x}$来给输出$\boldsymbol{y}$赋值。一般来说，这两者可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$\boldsymbol{x} = (x_1, x_2, \ldots, x_K)$，来预测单一的$y$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：

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

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。

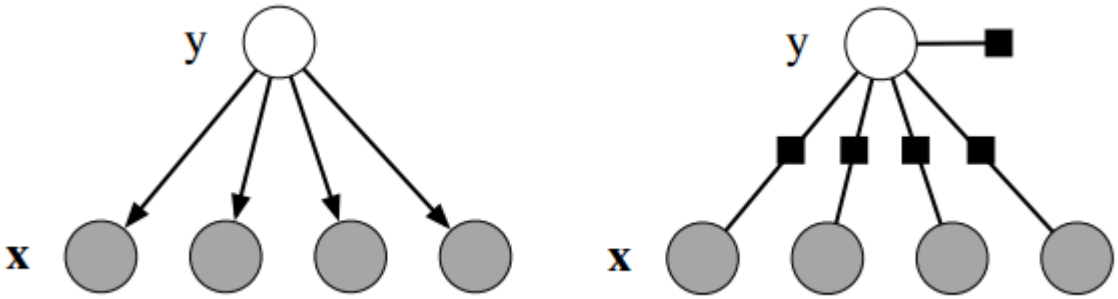

图2.3 朴素贝叶斯分类器，被当成有向模型（左），或因子图（右）

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$p(y|x)$，假设为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：

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

其中$Z(x)=\sum_y exp\{\theta_y+\sum_{j=1}^{K}\theta_{y,j}x_j\}$，是归一化常数。而$\theta_y$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$f_{y',j}(y,x)=1_{\{y'=y\}}x_j$，而把偏置权重的特征函数定义为$f_{y'}(y,x)=1_{\{y'=y\}}$。现在我们可以用$f_k$来遍历每个特征函数$f_{y',j}$，用$\theta_k$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：

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

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$Y=\{y_t\}_{t=1}^{T}$，来对一序列的观测$X=\{x_t\}_{t=1}^{T}$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$p(\boldsymbol{y},\boldsymbol{x})$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 于$y_1,y_2,\ldots,y_{t-1}$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$p(y1)$；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$\boldsymbol{y}$于观测序列$\boldsymbol{x}$的联合分布被分解为：

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

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。这让我们把创始状态概率$p(y_1)$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$(\boldsymbol{y},\boldsymbol{x})$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$p(\boldsymbol{y},\boldsymbol{x})=p(\boldsymbol{y})p(\boldsymbol{x}|\boldsymbol{y})$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$p(\boldsymbol{y}|\boldsymbol{x})$。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$p(\boldsymbol{x})$，一个判别模型可以被转化成联合分布$p(\boldsymbol{y},\boldsymbol{x})$，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$p(\boldsymbol{y}|\boldsymbol{x})$没有包含$p(\boldsymbol{x})$的模型，而它对分类并没有用。对$p(\boldsymbol{x})$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$\boldsymbol{x}$ 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$p(\boldsymbol{x})$是不可知的。判别模型，如CRF，仅对$\boldsymbol{y}$的条件独立性做假设，以及$\boldsymbol{y}$如何依赖于$\boldsymbol{x}$，但是不对$\boldsymbol{x}$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$p(\boldsymbol{y},\boldsymbol{x})$的因子图，现在要构建条件分布$p(\boldsymbol{y}|\boldsymbol{x})$的因子图，那么，所有只与$\boldsymbol{x}$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$\boldsymbol{y}$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$\boldsymbol{x}_t$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，我们也不想去做这个件事，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$p(\boldsymbol{x},\boldsymbol{y})=\prod_{t=1}^{T}p(y_t|y_{t-1})\prod_{k=1}^{K}p(x_{tk}|y_t)$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$\boldsymbol{x}=(x_1,x_2,\ldots,x_K)$重复一下，变换成$\boldsymbol{x}'=(x_1,x_1,x_2,x_2,\ldots,x_K,x_k)$，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$p(y|\boldsymbol{x}')$的估计，相比于$p(y|\boldsymbol{x})$，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：

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

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$p(y,\boldsymbol{x})$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$p(\boldsymbol{y})p(\boldsymbol{x}|\boldsymbol{y})$转化成条件分布$p(\boldsymbol{y}|\boldsymbol{x})$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$p^*(\boldsymbol{y},\boldsymbol{x}) = p^*(\boldsymbol{y})p^*(\boldsymbol{x}|\boldsymbol{y})$，那么我们只需简单地计算真实的$p^*(\boldsymbol{y}|\boldsymbol{x})$，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$p(\boldsymbol{y})p(\boldsymbol{x}|\boldsymbol{y})$，然后计算$p(\boldsymbol{y}|\boldsymbol{x})$（生成方案），会产生与直接估计$p(\boldsymbol{y}|\boldsymbol{x})$不同的结果。也就是说，生成与判别模型的目标都是估计$p(\boldsymbol{y}|\boldsymbol{x})$，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：

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

但是我们也可以按照概率的链式法则重写p_g如下：

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

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

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$p(\boldsymbol{x})$，使得$p(\boldsymbol{x})$可以从p_g的某个参数配置中产生。就是说，$p(\boldsymbol{x}) = p_c(\boldsymbol{x};\theta') = \sum_{\boldsymbol{y}} p_g(\boldsymbol{y},\boldsymbol{x};\theta')$译注：原文是$p(\boldsymbol{x}) = p_c(\boldsymbol{x};\theta') = \sum_{\boldsymbol{y}} p_g(\boldsymbol{y},\boldsymbol{x}|\theta')$，其中$\theta'$往往与（2.13）中的$\theta$不同。把这与同样从$p_g$中产生的条件分布$p_c(\boldsymbol{y}|\boldsymbol{x};\theta)$组合，即$p_c(\boldsymbol{y}|\boldsymbol{x};\theta) = p_g(\boldsymbol{y},\boldsymbol{x};\theta)/p_g(\boldsymbol{x};\theta)$。那么结果分布是：

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

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$p(\boldsymbol{y}|\boldsymbol{x})$的准确性，来弥补我们不怎么关心的$p(\boldsymbol{x})$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$p(\boldsymbol{x})$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$p(\boldsymbol{y},\boldsymbol{x})=p(\boldsymbol{y})p(\boldsymbol{x}|\boldsymbol{y})$，使得通过有向图来表示它更自然。其中在拓扑意义上，输出$\boldsymbol{y}$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$\boldsymbol{x}$在$\boldsymbol{y}$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。

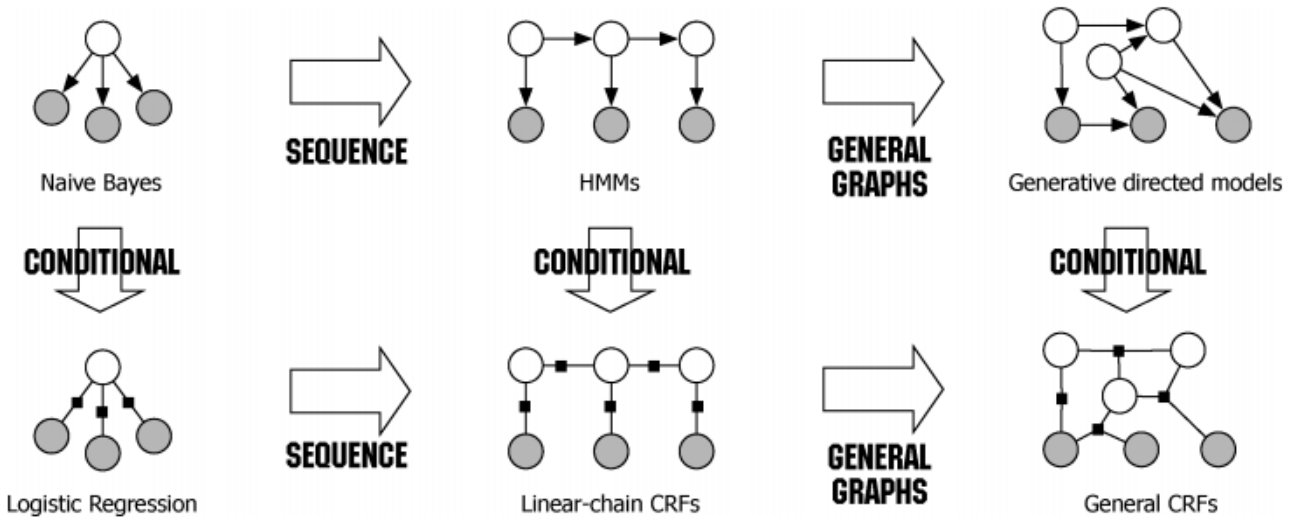

图2.4 朴素贝叶斯、逻辑回归、HMMS、线性链CRFs、生成模型和广义CRFs之间的关系图

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$p(\boldsymbol{y},\boldsymbol{x})$引出的条件分布$p(\boldsymbol{y}|\boldsymbol{x})$。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：

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

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$(\boldsymbol{y},\boldsymbol{x})$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：

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

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$p(\boldsymbol{y}|\boldsymbol{x})$写出来，即：

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

（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。这导致了我们关于线性链CRFs的一般定义

定义2.2 记Y, X是随机向量，$\theta = \{\theta_k\} \in \mathcal{R}^K$是一个参数向量，$\mathcal{F} = \{f_k(y, y', \boldsymbol{x}_t)\}_{k=1}^{K}$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$p(\boldsymbol{y}|\boldsymbol{x})$：

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

其中，$Z(\boldsymbol{x})$是依赖于输入的归一化函数：

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

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$\boldsymbol{x}$和$\boldsymbol{y}$上的因子图来描述，即

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

其中，局部函数Ψ_t具有一种特殊的 log-linear形式：

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

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$\boldsymbol{\theta}$。这将在第5节讲述。

之前我们已看到，如果一个联合分布$p(\boldsymbol{y},\boldsymbol{x})$像HMM一样分解了，那么对应的条件分布$p(\boldsymbol{y}|\boldsymbol{x})$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$\mathbf{1}_{\{y_t=j\}}\mathbf{1}_{\{y_{t-1}=j\}}\mathbf{1}_{\{x_t=o\}}$。 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$\boldsymbol{x}_1,\cdots,\boldsymbol{x}_T$之间的关系，我们可以让因子$\Psi_t$依赖于所有的输入$\boldsymbol{x}$。这不会大破线性图的结构——允许我们把$\boldsymbol{x}$当成单一的整体变量。结果，特征函数可以写成$f_k(y_t,y_{t-1},\boldsymbol{x})$，从而可以把全部的输入变量$\boldsymbol{x}$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$\boldsymbol{x} = (\boldsymbol{x}_1,\cdots,\boldsymbol{x}_T)$画成一个巨大的观测节点，冰杯所有的因子依赖，而不是把$\boldsymbol{x}_1,\cdot,\boldsymbol{x}_T$画成独立的节点。

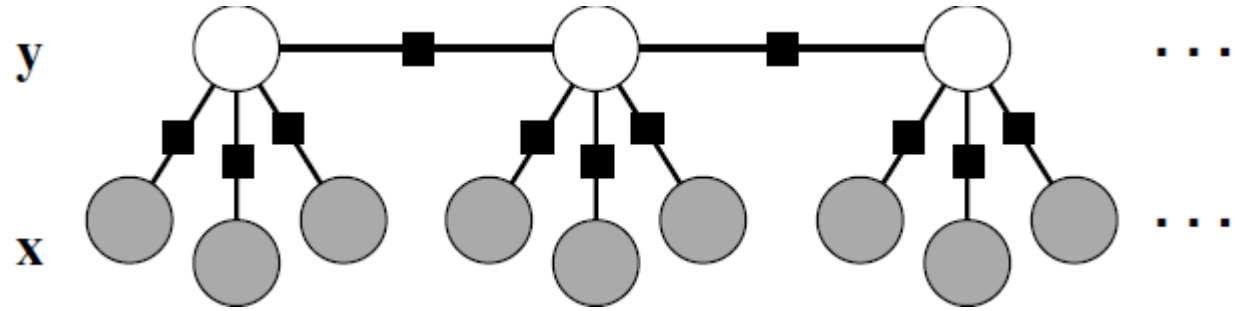

图2.5 来自式（2.17）的类HMM的线性链CRF

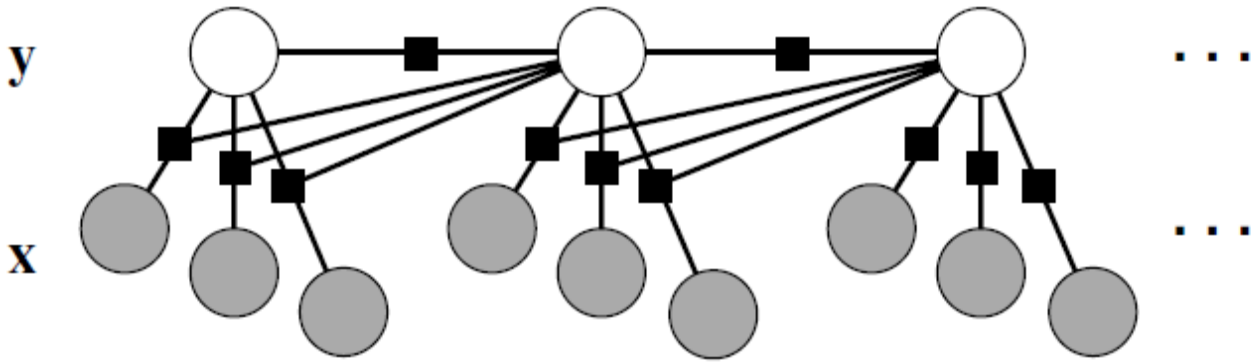

图2.6 转移因子依赖于当前输入的线性链CRF

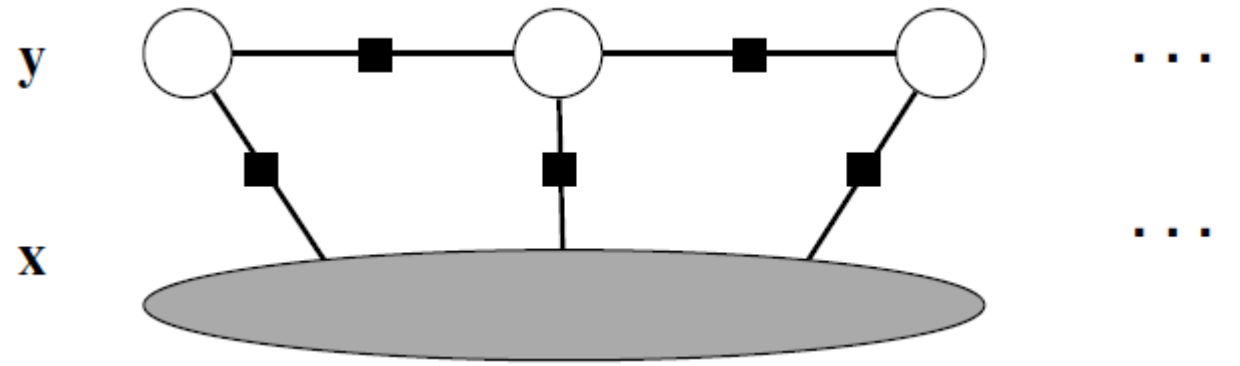

图2.7 转移因子依赖于全部输入的线性链CRF

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$\boldsymbol{x}_t$。$\boldsymbol{x}_t$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$\boldsymbol{x}_{t+1}$，那么$\boldsymbol{x}_t$应当包含了$\boldsymbol{x}_{t+1}$。

最后，归一化常数$Z(\boldsymbol{x})$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$\boldsymbol{X},\boldsymbol{Y}$上的因子图。如果对于$\boldsymbol{X}$中任意的值$\boldsymbol{x}$，分布$\boldsymbol{p}(\boldsymbol{y}|\boldsymbol{x})$是根据G来分解的，那么$(\boldsymbol{X},\boldsymbol{Y})$是一个**条件随机场conditional random field**。

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

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

本定义相比一般无向图的定义（2.1），差别在于归一化常数$Z(\boldsymbol{x})$现在变成了关于输入$\boldsymbol{x}$的函数。因为条件性趋向于简化图模型，$Z(\boldsymbol{x})$ 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$\boldsymbol{\Psi}_a$是一组特征的线性函数是有用的，即：

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

其中特征函数$\boldsymbol{f}_{ak}$和权重$\boldsymbol{\theta}_{ak}$都使用了因子的下标$\boldsymbol{a}$，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$\boldsymbol{x}$和$\boldsymbol{y}$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$(\boldsymbol{y}_a,\boldsymbol{x}_a)$的每一个值安排一个指示函数$\boldsymbol{f}_{ak}$，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

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

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$\boldsymbol{\Psi}_t(y_t,y_{t-1},\boldsymbol{x}_t)$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$\boldsymbol{\mathcal{C}}=\{C_1,C_2,\cdots,C_P\}$，其中每个$C_P$是一个**团模板clique template**，是一组共享了特征函数$\{\boldsymbol{f}_{pk}(\boldsymbol{x}_c,\boldsymbol{y}_c)\}_{k=1}^{K(p)}$和参数$\boldsymbol{\theta}_p\in\mathcal{R}^{K(p)}$的因子。一个使用了团模板的CRF可以写成

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

其中每个模板因子是这样参数化的

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

而归一化函数为

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

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$C_0 = \{\Psi_t(y_t, y_{t-1}, \boldsymbol{x}_t)\}_{t=1}^T$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$\Psi_t$分配独立的参数，就像非齐次HMM，那么需要T个模板，即$\mathcal{C} = \{C_t\}_{t=1}^T, C_t = \{\Psi_t(y_t, y_{t-1}, \boldsymbol{x}_t)\}$。

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$\boldsymbol{f}_{pk}$。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$\mathcal{C}_p$的特征$\boldsymbol{f}_{pk}$常常采用如下的特定形式：

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

也就是说，一个特征只在输出正好为$\tilde{\boldsymbol{y}}_c$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$\boldsymbol{x}_c$，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$q_{pk}(\boldsymbol{x}_c)$叫做观测函数，而不是特征。观测函数的例子有“单词$\boldsymbol{x}_t$是大写的”或“单词$\boldsymbol{x}_t$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$\boldsymbol{x}_t$是with，而标签$\boldsymbol{y}_t$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$x_t^{(i)}$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。译文则保留了这个not，并且$p(y_t = CITY - NAME|x_T^{(i)}) > \epsilon$时（$\epsilon$时一个阈值），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$\{q_m(\boldsymbol{x})\}, m \in \{1, 2, \cdots, M\}$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"x_t是NEW，且y_t是LOCATION"的特征，而不能使用那种一次把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:

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

$$f(y_t, \mathbf{x}_t) = q_m(\mathbf{x}_t)\mathbf{1}_{\{\mathbf{y}_t=y\}} \qquad \forall y \in \mathcal{Y}, \forall m$$

Node-observation features:

$$f(y_t, y_{t-1}, \mathbf{x}_t) = \mathbf{1}_{\{y_t=y\}}\mathbf{1}_{\{y_{t-1}=y'\}} \qquad \forall y, y' \in \mathcal{Y}$$

$$f(y_t, \mathbf{x}_t) = q_m(\mathbf{x}_t)\mathbf{1}_{\{y_t=y\}} \qquad \forall y \in \mathcal{Y}, \forall m$$

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"$y_{t-1} = START$且$y_t = PERSON$且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CRFsf 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$f_k(y, x_t)$定义为"如果x_t是单词dog时，$f_k = 1$，否则为0"。相反，把f_k定义为单词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$f_k(y_t, y_{t-1}, x_t)$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x_{t+1}和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$\Psi_t(y_t, y_{t-1}, \boldsymbol{x}_t)$的同时，还使用变量因子$\Psi_t(y_t, \boldsymbol{x}_t)$。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simpl'e 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$\boldsymbol{x}_t$出现在了Wikipedia提供的某个城市名单列表中，那么$q(\boldsymbol{x}_t) = 1$"。

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$f_t(y, \boldsymbol{x}_t) = p_{HMM}(y_t = y|\boldsymbol{x})$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$\boldsymbol{x}_t$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$\boldsymbol{x}_t$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$p(\boldsymbol{y}|\boldsymbol{x})$d 图结构随着输入$\boldsymbol{x}$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$\boldsymbol{y}$之上的图结构依赖于输入$\boldsymbol{x}$。

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$\boldsymbol{x} = (x_1, x_2, \cdots, x_T)$为一个向量，表示一张$\sqrt{T} \times \sqrt{T}$的图片。就是说，$\boldsymbol{x}_{1:\sqrt{T}}$代表第一行，$\boldsymbol{x}_{\sqrt{T}+1:\sqrt{T}+2}$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$\boldsymbol{y} = (y_1, y_2, \cdots, y_T)$，其中每个$y_i$是位子$i$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，texton特征【127】以及SIFT特征【77】。重要的是，这些特征不只依赖于像素$\boldsymbol{x}_i$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五香图模型，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$\boldsymbol{x}_i$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x_i和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：

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

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

其中，Ψ是鼓励光滑性的因子。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$\log p(\boldsymbol{y}, \boldsymbol{x})$。$p(x_i | y_i)$是像素值关于类别的条件分布。例如，$x_i$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，$p(\boldsymbol{x}|\boldsymbol{y})$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$q(x_i)$为在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x_i和x_j之间的特征向量$v(x_i, x_j)$，以使模型能够处理x_i与y_i之间的相似与不同。一种办法是把$v(x_i, x_j)$定义为$q(x_i)$和$q(x_j)$的叉乘，就是说，线计算矩阵$q(x_i)q(x_j)^T$，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$\boldsymbol{x}$和标签$\boldsymbol{y}$。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：

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

$$g_{m,m'}(y_i, y_j, x_i, x_j) = \mathbf{1}_{\{y_i = m\}} \mathbf{1}_{\{y_j = m\}} v(x_i, x_j)) \forall m, m' \in \{0, 1\}$$

$$f(y_i, x_i) = \begin{pmatrix} f_0(y_i, x_i) \\ f_1(y_i, x_i) \end{pmatrix}$$

$$g(y_i, y_j, x_i, x_j) = \begin{pmatrix} g_{00}(y_i, y_j, x_i, x_j) \\ g_{01}(y_i, y_j, x_i, x_j) \\ g_{10}(y_i, y_j, x_i, x_j) \\ g_{11}(y_i, y_j, x_i, x_j) \end{pmatrix}$$

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$\Psi(y_i, y_j)$。虽然Ψ鼓励了一致性，但缺乏灵活性。如果x_i和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：

$$v(x_i, x_j) = exp\left\{-\beta(x_i - x_j)^2\right\}$$

$$g(y_i, y_j, x_i, x_j) = \mathbf{1}_{\{y_i \neq y_j\}} v(x_i, x_j). \quad (2.33)$$

综合起来，CRF模型是：

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

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$p(\boldsymbol{y}|\boldsymbol{x})$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

3.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$p(\boldsymbol{y}|\boldsymbol{x}, \theta)$与一组输入输出均已知的训练样本$D = \{\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}\}_{i=1}^N$相匹配。我们希望，给定任何一个输入样本$\boldsymbol{x}^{(i)}$，从模型推断出的关于输出的分布$p(\boldsymbol{y}|\boldsymbol{x}^{(i)}, \theta)$，能“像是”从训练数据中得来的真实的输出$\boldsymbol{y}^{(i)}$。

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$\boldsymbol{x}$，然后从$p(\boldsymbol{y}|\boldsymbol{x}, \theta)$中采样$\boldsymbol{y}$，触发特征$\boldsymbol{f_k}(y_t, y_{t-1}, \boldsymbol{x}_t)$的概率，能与训练数据中$\boldsymbol{f_k}$发生的概率相等。正式地，要求$\boldsymbol{f_k}$满足：

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

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

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

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$p(y_t, y_{t-1}|\boldsymbol{x}, \theta)$，而是通用图模型中一个因子a上的所有变量$\boldsymbol{Y}_a$的边缘分布$p(\boldsymbol{y}_a|\boldsymbol{x}, \theta)$。

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$\boldsymbol{x}$和参数$\boldsymbol{\theta}$的条件下，计算关于输出$\boldsymbol{y}$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$\boldsymbol{Y}_a$上的边缘分布$p(\boldsymbol{y}|\boldsymbol{x}, \theta)$。 $\boldsymbol{Y}_a$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$Z(\boldsymbol{x})$的副产品来计算。

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

边缘分布$p(\boldsymbol{y}_a|\boldsymbol{x}, \theta)$与归一化函数$Z(\boldsymbol{x})$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$\boldsymbol{y}^*$，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。

4.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$\boldsymbol{x}$后，要预测最可能的输出$\boldsymbol{y}^* = \arg\max_{\boldsymbol{y}} p(\boldsymbol{y}|\boldsymbol{x})$。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$p(y_t|\boldsymbol{x})$和连接的$p(y_t, y_{t-1}|\boldsymbol{x})$。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，因而我们希望能在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，也因为它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$p(\boldsymbol{y},\boldsymbol{x})=\prod_t \Psi_t(y_t,y_{t-1},x_t)$，而因子被定义为

$$\Psi_t(j,t,x) \stackrel{def}{=} p(y_t=j|y_{t-1}=i)p(x_t=x|y_t=j). \quad (4.1)$$

如果把这个HMM看成带权重的有限状态机，那么$\Psi_t(j,i,x)$就是当观测为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$p(\boldsymbol{x})$的。前向后向算法背后的思想是，首先把$p(\boldsymbol{x})=\sum_{\boldsymbol{y}} p(\boldsymbol{x},\boldsymbol{y})$的求和运算，按照如下的方式重写

$$\begin{aligned} p(\boldsymbol{x}) &= \sum_{\boldsymbol{y}} \prod_{t=1}^{T} \Psi_t(y_t,y_{t-1},x_t) \\ &= \sum_{y_T}\sum_{y_{T-1}} \Psi_T(y_T,y_{T-1},x_T) \sum_{y_{T-2}} \Psi_{T-1}(Y_{T-1},y_{T-2},x_{T-1}) \sum_{y_{T-2}} \cdots \quad (4.3) \end{aligned}$$

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：

$$\begin{aligned} \alpha_t(j) &\stackrel{def}{=} p(\boldsymbol{x}_{<1\cdots t>}, y_t=j) \\ &= \sum_{\boldsymbol{y}_{<1\cdots t-1>}} \Psi_t(j,y_{t-1},x_t) \prod_{t'=1}^{t-1} \Psi_{t'}(y_{t'},y_{t'-1},x_{t'}), (4.5) \end{aligned}$$

其中，求和运算的下标$\boldsymbol{y}_{1\cdots t-1}$，表示要覆盖$y_1,y_2,\cdots,y_{t-1}$的所有可能值。这一alpha可以通过递归的方式计算

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

而初始变量为$\alpha_1(j)=\Psi_1(j,y_0,x_1)$。（回忆（2.10），知道$y_0$是HMM的固定的初始值）。反复地递归（4.6）式，可知$p(\boldsymbol{x})=\sum_{y_T}\alpha_T(y_T)$。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

$$\begin{aligned} \beta_t(i) &\stackrel{def}{=} p(\boldsymbol{x}_{<t+1\cdots T>}|y_t=i) \\ &= \sum_{\boldsymbol{y}_{<t+1\cdots T>}} \prod_{t'=t+1}^{T} \Psi_{t'}(y_{t'},y_{t'-1},x_{t'}), (4.8) \end{aligned}$$

而其递归为

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

其中，初始量为$\beta_T(i)=1$。与前向类似，我们也可以用后向变量来计算
$p(\boldsymbol{x})=\beta_0(y_0)\stackrel{def}{=}\sum_{y_1}\Psi_1(y_1,y_0,x_1)\beta_1(y_1)$。

要计算边缘分布$p(y_{t-1}, y_t|\boldsymbol{x})$（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

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

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，$\boldsymbol{x}_{<1\cdots t-1>}$与$\boldsymbol{x}_{<t+1\cdots T>}$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

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

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：

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

而$1/p(\boldsymbol{x})$相当于分布的归一化常数。我们通过$p(\boldsymbol{x}) = \beta_0(y_0)$或$p(\boldsymbol{x}) = \sum_{i\in S}\alpha_T(i)$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$\boldsymbol{y}^* = \arg\max_{\boldsymbol{y}} p(\boldsymbol{y}|\boldsymbol{x})$，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$\boldsymbol{\alpha}$像类似的变量为

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

而这可以通过类似的递归来计算:

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

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:

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

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

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

其中

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

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）可不经过修改就用于线性链CRFs，只是其含义有所改变。在CRF中，$\alpha_t(j) = p(\boldsymbol{x}_{<1\cdots t>}, y_t = j)$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$Z(\boldsymbol{x})$，而不是$p(\boldsymbol{x})$，且$Z(\boldsymbol{x}) = \beta_0(y_0), Z(\boldsymbol{x}) = \sum_{i \in S} \alpha_T(i)$。

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

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

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

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$p(\boldsymbol{y}|\boldsymbol{x})$中采样$\boldsymbol{y}$，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$\arg\max_{\boldsymbol{y}} p(\boldsymbol{y}|\boldsymbol{x})$，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$S \subset [1, 2, \cdots, T]$（不一定是连接在一起的）的边缘概率$p(\boldsymbol{y_S}|\boldsymbol{x})$。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。最槽糕的时候，它们需要指数级别的耗时，但在工程实践中扔不失为有效的方法。最流行的精确算法是联和树算法。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，使它在最槽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$p(\boldsymbol{y})$，还是像CRFs的条件分布$p(\boldsymbol{y}|\boldsymbol{x})$。为了强调这一点，也为了简化记号，我们去掉了对$\boldsymbol{x}$的依赖，而只讨论联合分布$p(\boldsymbol{y})$的推断。该分布从某些因子图$G = (V, F)$而来，即

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

若想将这里的讨论用于CRFs，只需用$\Psi_a(\boldsymbol{y}_a, \boldsymbol{x}_a)$替换上面的$\Psi_a(\boldsymbol{y}_a)$，同时将$Z$换成$p(\boldsymbol{y})$。这样就可以对$\boldsymbol{x}$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$p(\boldsymbol{y})$，还是CRF 的$p(\boldsymbol{y}|\boldsymbol{x})$，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$p(y_s)$，而是从联合分布$p(\boldsymbol{y})$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$\boldsymbol{Y}$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$p(y_s)$。假如函数$f(\boldsymbol{y})$服从分布$p(\boldsymbol{y})$，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$\boldsymbol{y}^1, \boldsymbol{y}^2, \cdots, \boldsymbol{y}^M$，我们可以通过下式来估计这个期望

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

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$\boldsymbol{y}^j$，那么需要产生下一个样本$\boldsymbol{y}^{j+1}$ (1)取$\boldsymbol{y}^{j+1} = \boldsymbol{y}^j$。 (2)对每个$s \in S$，重采样$\boldsymbol{Y}_s$。从分布$p(y_s|\boldsymbol{y}_{\setminus s}, \boldsymbol{x})$中采样$y_s^{j+1}$。

(3)返回$\boldsymbol{y}^{j+1}$作为结果。

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

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

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$p(\boldsymbol{y}|\boldsymbol{x})$容易计算得多，因为它只需遍历$y_s$的所有可能取值，而不是整个$\boldsymbol{y}$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$p(\boldsymbol{y})$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$p(\boldsymbol{y})$。

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$V_a \backslash Y_s$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

$$\begin{aligned} p(y_s) &\propto \sum_{\boldsymbol{y} \backslash y_s} \prod_{a \in F} \Psi_a(\boldsymbol{y}_a) (4.21) \\ &= \sum_{\boldsymbol{y} \backslash y_s} \prod_{a \in N(s)} \prod_{\Psi_b \in F_a} \Psi_b(\boldsymbol{y}_b) (4.24) \\ &= \prod_{a \in N(s)} \sum_{\boldsymbol{y}_{V_a} \backslash y_s} \prod_{\Psi_b \in F_a} \Psi_b(\boldsymbol{y}_b). (4.25) \end{aligned}$$

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。

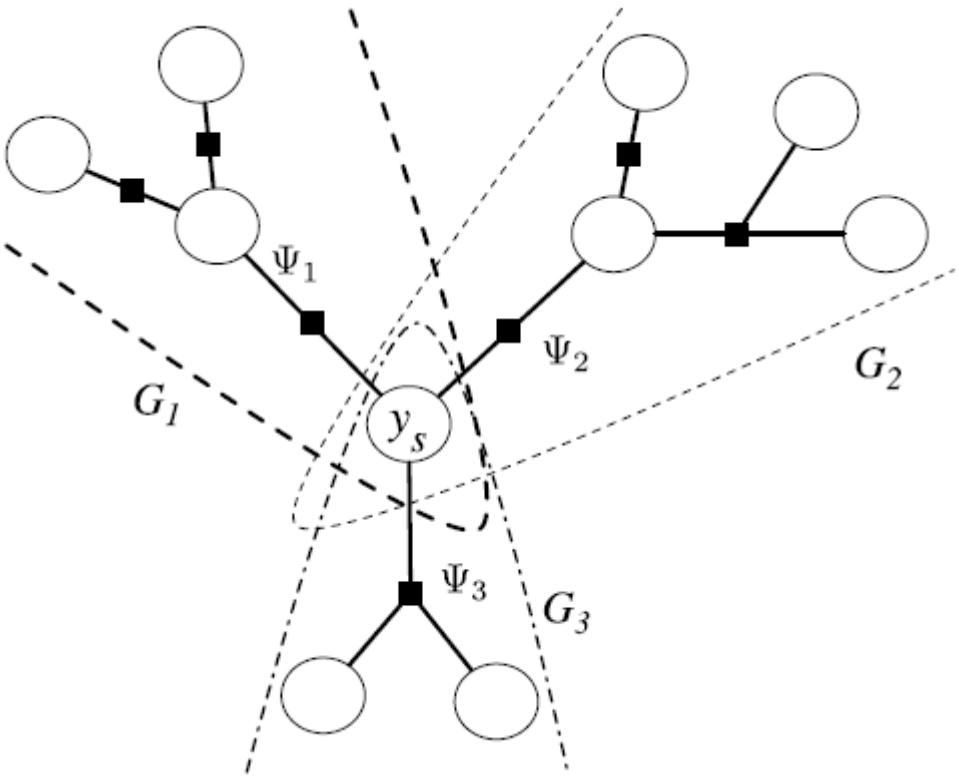

图4.1 树形图的边缘分布是如何被划分的。这一划分被置信传播算法（4.2.2）所用。

把上式的每个因子记为m_{as}，那么

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

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:

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

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：

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

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：

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

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$p(\boldsymbol{y}_a)$和联合概率$p(\boldsymbol{y})$。（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

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

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$p(\boldsymbol{y})$，可以发现联合分布总是按照下面的方式分解的：

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

例如，对于线性链，这变成

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

通过消元、移项等操作，上式不过是我们熟悉的$p(\boldsymbol{y}) = \prod_t p(y_t, y_{t-1})$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$p(\boldsymbol{y})$。也可得到$Z = p(\boldsymbol{y})^{-1} \prod_{a \in F} \Psi_a(y_a)$。

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$alpha_2$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$p(\boldsymbol{y}_a)$的解构是线性链（4.14）式的一般化。

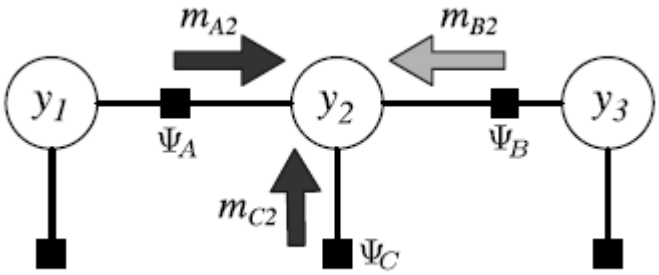

图4.2 前向后向算法与置信传播算法在线性链图中的一致性。具体请参考正文

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$q(\boldsymbol{y}_s)$。

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，而更多的细节请供参考文章【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$\boldsymbol{y}$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$\boldsymbol{y}$的所有可能的分布的集合，并取目标函数为：

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

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$q(\boldsymbol{y}) = \prod_s q_s(\boldsymbol{y}_s)$，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

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

其中d_i是i的阶，表示连接到$\boldsymbol{y}_i$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：

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

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）$\boldsymbol{q}$，其元素为$q_{a;\boldsymbol{y}_a}$（对应一个因子a以及相关变量$\boldsymbol{y}_a$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 consitent），就是说，

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

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$q(\boldsymbol{y})$拥有这些边缘概率（that there exists a single joint $q(\boldsymbol{y})$ that has those marginals我没能理解这句话）。因此，分布$q_a(\boldsymbol{y}_a)$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见解，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$\log Z$的。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$\Psi_c(\boldsymbol{x}_c, \boldsymbol{y}_c)$需要计算 参数向量$\theta_p$和特征向量$\boldsymbol{f}_c\{f_{pk}(\boldsymbol{y}_c, \boldsymbol{x}_c)|\forall p, \forall k\}$的内积。一般来说，向量$\boldsymbol{f}_c$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（trainsitions）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$Z(\boldsymbol{x})$的计算，因为可以按照$Z(\boldsymbol{x}) = p(\boldsymbol{y}'|\boldsymbol{x})^{-1} \prod_t (\Psi_t(y'_t, y'_{t+1}, \boldsymbol{x}_t))$来计算，其中$p(\boldsymbol{y}'|\boldsymbol{x})^{-1}$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：

$$\log \alpha_t(j) = \bigoplus_{i \in S} \left(\log \Psi_t(j, i, x_t) + \log \alpha_{t-1}(i) \right), (4.38)$$

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：

$$a \bigoplus b = a + \log(1 + e^{b-a}) = b + \log(1 + e^{a-b}), (4, 39)$$

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$\mathbf{log}$和$\mathbf{exp}$运算。这对HMMs是对的，但不是CRFs。因为CRFs总归是要调用$\mathbf{exp}$来计算$\Psi_t(y_t, y_{t+1}, \boldsymbol{x}_t)$，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。

5.参数估计

这一节，我们讲述如何估计CRF的参数$\theta = \{\theta_k\}$。在最典型、最简单的情况下，数据是完全标注的，但也有研究是关于半监督CRF、带隐藏变量的CRF和关系学习的CRF。

极大似然是一种训练CRF的方法，就是说，要选择参数，使训练数据在模型意义下具有最高的概率。原理上，它与逻辑回归的做法很像。考虑我们在第2节所讲述的这些模型之间的联系，这一点应该不让人意外。主要的区别点在于计算方面：CRF倾向拥有更多参数、更复杂的结构，导致了更高的训练成本。

在树形CRF中，极大似然可基于数值优化过程，以第4.1节江苏的推断算法为子过程。推断算法同时计算了似然和它的梯度。一般来说，似然是关于参数的凸函数，意味着有效的优化过程是现成的，且一定收敛到最优点。

我们从讲述极大似然开始，包含有线性链（第5.1.1节）和通用图结构（第5.1.2节），还包括隐藏变量的情况。我们也将讲述两种加快参数训练的方法：随机梯度下降法（挖掘数据中的 iid 结构，第5.2节）和多线程训练（第5.3节）。

对于通用CRF，精确的极大似然训练是不存在的，因而需要近似过程。泛泛地说，有两种解决问题的策略。一是使用易于计算的函数来近似该似然，叫做**代理似然surrogate likelihood**，再数值地优化该代理函数。第二种方法是边缘概率近似。它在极大似然训练需要精确计算的时候，嵌入一个近似的推断算法来计算边缘分布。这里需要小心，因为近似推断和学习之间存在着微妙而复杂的作用。我们在第5.4节讨论这些。

5.1极大似然

5.1.1线性链CRF

线性链CRF的极大似然参数可以用数值优化的方法确定。我们拥有iid训练数据$\mathcal{D}=\{\boldsymbol{x}^{(i)},\boldsymbol{y}^{(i)}\}_{i=1}^{N}$，其中$\boldsymbol{x}^{(i)}=\{x_1^{(i)},x_2^{(i)},\cdots,x_T^{(i)}\}$是一系列输入，而$\boldsymbol{y}^{(i)}=\{y_1^{(i)},y_2^{(i)},\cdots,y_t^{(i)}\}$是期望的预测结果。为了简化符号，我们假设每个训练序列$\boldsymbol{x}^{(i)}$的长度都是T。一般来说，每个序列的长度不必相同——也就是说，T依赖于i。下面的讨论可直接扩展以覆盖这种情况。

参数估计一般通过带惩罚项的极大似然来完成。因为我们对条件分布建模了，那么如下的log似然，有时也叫**条件log似然**，正合适：

$$\ell(\theta)=\sum_{i=1}^{N}\log p(\boldsymbol{y}^{(i)}|\boldsymbol{x}^{(i)};\theta).\ (5.1)$$

计算极大似然估计，实际是最大化$\ell(\theta)$。就是说，所求的估计为$\hat{\theta}_{ML}=\sup_{\theta}\ell(\theta)$。

一种理解$p(\boldsymbol{y}^{(i)}|\boldsymbol{x}^{(i)};\theta)$的办法，是想象它与某各任意的先验概率$p(\boldsymbol{x};\theta)$结合以构成联合分布概率$p(\boldsymbol{y},\boldsymbol{x})$。然后我们的联合log似然为

$$\log p(\boldsymbol{y}|\boldsymbol{x};\theta)=\log p(\boldsymbol{y}|\boldsymbol{x};\theta)+\log p(\boldsymbol{x};\theta'),\ (5.2)$$

注意，项$p(\boldsymbol{x};\theta')$与条件分布的 参数$\theta$无关。如果我们不用估计$p(\boldsymbol{x})$，那么当计算$\theta$的极大似然估计时，那么可以直接去掉（5.2）中的第二项。结果正是（5.1）。

将CRF的模型（2.18）带入（5.1），我们得到：

$$\ell(\theta)=\sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{k=1}^{K}\theta_k f_k(y_t^{(i)},y_{t-1}^{(i)},\boldsymbol{x}_t^{(i)})-\sum_{i=1}^{N}\log Z(\boldsymbol{x}^{(i)}),\ (5.3)$$

通常，我们拥有数量庞大的参数，如几十万个。为了避免过拟合，我们使用**规则化regularization**，就是对权重向量过大的模进行惩罚。常见的惩罚项是θ的欧几里得范数，以及**规则化参数$1/2\sigma^2$来定义惩罚强度。规则化的log似然为

$$\ell=\sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{k=1}^{K}\theta_k f_k(y_t^{(i)},y_{t-1}^{(i)},\boldsymbol{x}_t^{(i)})-\sum_{i=1}^{N}\log Z(\boldsymbol{x}^{(i)})-\sum_{k=1}^{K}\frac{\theta_k^2}{2\sigma^2}.\ (5.4)$$

σ^2是一个自由参数，用来决定对大权重的惩罚力度。其直观想法是避免少数特征就支配了预测结果。根据规则化的写法，规则化可以被理解成最大化了一个后验(MAP)估计，就像θ被赋予了均值为0方差为$\sigma^2 I$的高斯后验分布一样 不是特别清楚这里的意思，但也无关紧要。我在实践中用的是等式约束。。确定最佳的规则化参数需要 高计算量的参数扫描。幸运的是，所得模型的精度并不敏感于σ^2（如，10倍以内不会带来大的影响）。最佳的σ^2与训练数据集的大小有关。对于地5.5节江苏的训练集来说，我们通常去$\sigma^2=10$。

也可以用L_1范数来作为规则化，而这相当于对参数做了double exponential prior假设【44】。这得到如下的带惩罚的似然

$$\ell'(\theta)=\sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{k=1}^{K}\theta_k f_k(y_t^{(i)},y_{t-1}^{(i)},\boldsymbol{x}_t^{(i)})-\sum_{i=1}^{N}\log Z(\boldsymbol{x}^{(i)})-\alpha\sum_{k=1}^{K}|\theta_k|.\ (5.5)$$

其中，α是需要调节的规则化参数，就像L_2范数的σ^2。这种规则化鼓励稀疏的参数，即大多数θ_k为0。这对特征选择很有用，以及另外理论上的优势【97】。实践中，用L_1规则化训练的模型更稀疏，但精度方面与L_2规则化大致相当【57】。L_1范数的缺点是它在0处不可导，某种程度上让数值优化变得复杂【3,44,160】。

一般来说，$\ell(\theta)$的极大值没有封闭解，因而需要数值优化。（5.4）的偏微分是

$$\frac{\partial \ell}{\partial \theta_k} = \sum_{i=1}^{N}\sum_{t=1}^{T} f_k(y_t^{(i)}, y_{t-1}^{(i)}, \boldsymbol{x}_t^{(i)}) - \sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{y,y'} f_k(y, y', \boldsymbol{x}_t^{(i)}) p(y, y' | \boldsymbol{x}_t^{(i)}) - \frac{\theta_k}{\sigma^2}. \quad (5.6)$$

可以把第一项看成f_k在经验分布$\tilde{p}$下的期望。这一分布的定义是

$$\tilde{p}(\boldsymbol{y}, \boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}_{\{\boldsymbol{y}=\boldsymbol{y}^{(i)}\}} \mathbf{1}_{\{\boldsymbol{x}=\boldsymbol{x}^{(i)}\}}. \quad (5.7)$$

第二项来自于$\log Z(\boldsymbol{x})$的偏导数，相当于f_k在模型分布$p(\boldsymbol{y}|\boldsymbol{x};\theta)\tilde{p}(\boldsymbol{x})$下的期望。因此，当规则化极大似然的达到它的解时，梯度为0，意味着这两个期望相等。这一优点是指数家族中极大似然估计的标准结果。

要计算似然$\ell(\theta)$和它的微分，需要我们在第4节介绍的推断技术。首先在似然的计算中，推断被用来计算归一化函数$Z(\boldsymbol{x}^{(i)})$，是需要遍历所有可能的标签值的。其次是在微分的计算中，推断被用来计算边缘分布$p(y, y'|\boldsymbol{x}_t^{(i)})$。因为这两个量都依赖于$\boldsymbol{x}^{(i)}$，所以为每个样本计算似然时都要运行一次推断。这是与生成模型（如第2.1.1节介绍的无向生成模型）的一个不同点。极大似然也可以用于无向生成模型的训练，但此时Z只与参数有关，而与输入无关。一次似然的计算就要进行N次推断，而这激发了随机梯度上升方法（5.2节）。

现在我们讨论如何优化$\ell(\theta)$。函数$\ell(\theta)$是凹的，这是由于形式如$g(\boldsymbol{x}) = \log \sum_i \exp x_i$的函数的凸性。凸性对参数估计很有帮助，因为它意味着每个局部极值点就是全局最优点。另外，如果使用了一个严格凹的规则化，如$L2$规则化，那么目标函数也变得严格地凹，这暗示它拥有唯一的全局最优点。

或许沿着梯度（5.6）最速上升是优化ℓ的最简单方法，但这在实践中需要太多次迭代了。牛顿方法的收敛要快得多，因为它利用了似然的曲率，但是需要计算Hessian——二阶微分矩阵。Hessian矩阵的大小是参数个数的平方。既然工程中的应用常常使用几万甚至几百万个参数，简单地保存整个Hessian矩阵是不切实际的。

相反，（5.4）的优化需要对二阶信息做近似。特别成功的方法是准牛顿方法，如BFGS【8】。它从目标函数的一阶微分中计算Hessian矩阵的近似。完整的对Hessian矩阵的$K \times K$近似仍然需要二次方的尺寸，所以需要有限内存版本的BFGS——来自Byrd等【17】。共轭梯度是另一个对二阶信息做近似的优化技术，并在CRF中获得了成功。关于有限内存BFGS和共轭梯度的优秀介绍，请参考Nocedal和Wright【100】。也可想成是一个黑箱优化程序 that is a drop-in replacement for vanilla gradient ascent。当使用了这些二阶方法之后，基于梯度的优化方法比Lafferty等【63】所介绍的那种原始的iterative scaling方法快得多。这被一些作者【80,92,125,153】实践表明了。最后，置信域方法最近在多项逻辑回归上表现良好【74】，因而或许也对CRF表现良好。

这些优化算法——最速下降，牛顿法，准牛顿法，共轭梯度法和置信域法——是非线性函数的标准数值优化技术。我们把它们看成CRF规则化极大似然的现成方法。这些算法通常需要能够计算目标函数的值和梯度。在我们这里，目标函数值是（5.4）式，而一阶导数在（5.6）式中给出了。这也是我们在第4节，除了边缘概率，还讲述了如何计算归一化函数$Z(\boldsymbol{x})$的原因。

最后，我们讨论一下训练线性链模型的计算代价。正如我们将在第4.1节看到的?，单个训练样本的似然和梯度可在$O(TM^2)$次计算内，通过前向后向算法获得。其中M是标签的个数而T是训练样本的长度。因为我们需要为每个训练样本运行一次前向后向算法，所以一次似然和梯度的计算量为$O(TM^2N)$。所以总的训练代价为$O(TM^2NG)$。其中G是优化过程计算梯度的次数应该是迭代次数。可惜，G依赖于数据集且难以提前估计。对于线性链的batch L-BFGS，这个数一般（但不总）是100。对很多数据集来说，这一计算量是合理的。但如果变量的个数M十分巨大，或者训练样本的个数N十分庞大时，可能会让训练变得昂贵。根据标签的数量，训练CRF可能需要几分钟或是几天——可参考5.5节的例子。

5.1.2 通用CRF

通用CRF的参数估计在本质上与线性链一样，只是对模型期望的计算需要更通用的推断算法。首先，我们讨论观测完整的情况，其中训练数据和测试数据相互独立。这时，条件log似然，使用2.4节的记法，就是

$$\ell(\theta) = \sum_{C_p=\mathcal{C}} \sum_{\Psi_c \in C_p} \sum_{k=1}^{K(p)} \theta_{pk} f_{pk}(\boldsymbol{x}_c, \boldsymbol{y}_c) - \log Z(\boldsymbol{x}). \quad (5.8)$$

本节的方程都不显式地遍历训练样本集，因为如果某个应用刚好拥有 idd 训练样本 idd 是啥？，那么可以用图G中断开的多个子块表示。

log似然关于团模板C_p的参数θ_{pk}的偏导数为：

$$\frac{\partial \ell}{\partial \theta_{pk}} = \sum_{\Psi_c \in C_p} f_{pk}(\boldsymbol{x_c}, \boldsymbol{y_c}) - \sum_{\Psi_c \in C_p} \sum_{\boldsymbol{y'_c}} f_{pk}(\boldsymbol{x_c}, \boldsymbol{y'_c}) p(\boldsymbol{y'_c}|\boldsymbol{x}). \quad (5.9)$$

上面的函数$\ell(\theta)$与线性链的具有许多共同的特点。首先，0梯度条件可以解释为"要求充分统计$F_{pk}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{\Psi_c} f_{pk}(\boldsymbol{x_c}, \boldsymbol{y_c})$在经验分布和模型分布下具有相同的期望。其次，函数$\ell(\theta)$是凹的，因而可以使用像共轭梯度或L-BFGS这样的二阶最大化技术来解。最后，规则化的方法与线性链如出一辙。

到目前为止的所有讨论，都假设训练数据包含完整的输出变量的值。**潜变量Latent variables**是一些在训练和测试时都未知的变量。在Quatoni等的【109,110】中，拥有潜变量的CRF被称为**隐状态hidden-state**CRF(HCRFs)。关于其他早期的HCRFs，请参考【84,138】。训练带潜变量的CRF要更困难，因为需要计算潜变量的边缘概率来完成推断。

假如我们的CRF的输入为$\boldsymbol{x}$，$\boldsymbol{y}$为训练数据集中可观测的变量，而$\boldsymbol{w}$为另外的潜变量。从而，CRF具有如下形式

$$p(\boldsymbol{y}, \boldsymbol{w}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \prod_{C_p \in \mathcal{C}} \prod_{\Psi_c \in C_p} \Psi_c(\boldsymbol{x_c}, \boldsymbol{w_c}, \boldsymbol{y_c}; \boldsymbol{\theta_p}). \quad (5.10)$$

一种在训练时用来最大化的目标函数是如下的边缘似然：

$$\ell(\theta) = \log p(\boldsymbol{y}|\boldsymbol{x}) = \log \sum_{\boldsymbol{w}} p(\boldsymbol{y}, \boldsymbol{w}|\boldsymbol{x}). \quad (5.11)$$

第一个问题是如何计算边缘似然$\ell(\theta)$，因为如果有许多变量$\boldsymbol{w}$，那么求和就不能被直接计算。关键是要意识到，我们只需要遍历训练集中出现过的$\boldsymbol{y}$，而不是所有可能的$\boldsymbol{y}$，来计算$\log \sum_{\boldsymbol{w}} p(\boldsymbol{y}, \boldsymbol{w}|\boldsymbol{x})$。于是，利用原始的CRF（5.10），并让变量$\boldsymbol{Y}$都取它们在训练集中的值，得到$\boldsymbol{w}$的分布：

$$p(\boldsymbol{w}|\boldsymbol{y}, \boldsymbol{x}) = \frac{1}{Z(\boldsymbol{y}, \boldsymbol{x})} \prod_{C_p \in \mathcal{C}} \prod_{\Psi_c \in C_p} \Psi_c(\boldsymbol{x_c}, \boldsymbol{w_c}, \boldsymbol{y_c}; \boldsymbol{\theta_p}), \quad (5.12)$$

其中，归一化因子为

$$Z(\boldsymbol{y}, \boldsymbol{x}) = \sum_{\boldsymbol{w}} \prod_{C_p \in \mathcal{C}} \prod_{\Psi_c \in C_p} \Psi_c(\boldsymbol{x_c}, \boldsymbol{w_c}, \boldsymbol{y_c}; \boldsymbol{\theta_p}). \quad (5.13)$$

可以像$Z(\boldsymbol{x})$那样去计算这一新的归一化常量$Z(\boldsymbol{y}, \boldsymbol{x})$。实际上，$Z(\boldsymbol{y}, \boldsymbol{x})$更易于计算，因为它只需遍历$\boldsymbol{w}$来求和，而$Z(\boldsymbol{x})$需要遍历$\boldsymbol{w}$和$\boldsymbol{y}$。从图的角度，这相当于说，在图$G$中固定了$\boldsymbol{y}$的值后，可以让只剩下$\boldsymbol{w}$的结构得到简化。

一旦计算了$Z(\boldsymbol{y}, \boldsymbol{x})$，边缘似然可以按照如下方式计算

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \sum_{\boldsymbol{w}} \prod_{C_p \in \mathcal{C}} \prod_{\Psi_c \in C_p} \Psi_c(\boldsymbol{x}_c, \boldsymbol{w}_c, \boldsymbol{y}_c; \boldsymbol{\theta}_p) = \frac{Z(\boldsymbol{y}, \boldsymbol{x})}{Z(\boldsymbol{x})}. \quad (5.14)$$

现在我们已经拥有了计算ℓ的方法，那么来讨论如何关于θ来最大化它。因为ℓ一般不再是凸的(log - sum-exp是凸的，但两个log-sum-exp的差却不一定了），所以求它的最大值是困难的。所以，优化过程一般只能保证得到局部极大点。不管使用了什么优化技术，都要特别小心地初始化模型的初始值，以达到全局最大值。

我们讨论最大化ℓ的两种方法：像Quattoni等【109】那样直接使用梯度；像McCallum等【84】那样使用EM。（另外，这里也很适合使用随机梯度下降，第5.2节）要直接最大化ℓ，我们需要计算梯度。利用下面的事实是最简单的方法。对于任何的函数$f(\theta)$，我们有

$$\frac{df}{d\theta} = f(\theta) \frac{d \log f}{d\theta}, \quad (5.15)$$

这里，我们对$\log f$使用了链式法则，并对公式做了调整。将这一点应用到边缘似然$\ell(\theta) = \log \sum_{\boldsymbol{w}} p(\boldsymbol{y}, \boldsymbol{w}|\boldsymbol{x})$得到

$$\begin{aligned} \frac{\partial \ell}{\partial \theta_{pk}} &= \frac{1}{\sum_{\boldsymbol{w}} p(\boldsymbol{y}, \boldsymbol{w}|\boldsymbol{x})} \sum_{\boldsymbol{w}} \frac{\partial}{\partial \theta_{pk}} [p(\boldsymbol{y}, \boldsymbol{w}|\boldsymbol{x})] \quad (5.16) \\ &= \sum_{\boldsymbol{w}} p(\boldsymbol{w}|\boldsymbol{y}, \boldsymbol{x}) \frac{\partial}{\partial \theta_{pk}} [\log p(\boldsymbol{y}, \boldsymbol{w}|\boldsymbol{x})]. \quad (5.17) \end{aligned}$$

这是完整CRF的梯度的期望，是关于$\boldsymbol{w}$的。这个表达式可简化为

$$\frac{\partial \ell}{\partial \theta_{pk}} = \sum_{\Psi_c \in C_p} \sum_{\boldsymbol{w}'_c} p(\boldsymbol{w}'_c|\boldsymbol{y}, \boldsymbol{x}) f_k(\boldsymbol{y}_c, \boldsymbol{x}_c, \boldsymbol{w}'_c) - \sum_{\Psi_c \in C_p} \sum_{\boldsymbol{w}'_c, \boldsymbol{y}'_c} p(\boldsymbol{w}'_c, \boldsymbol{y}'_c|\boldsymbol{x}_c) f_k(\boldsymbol{y}'_c, \boldsymbol{x}_c, \boldsymbol{w}'_c). \quad (5.18)$$

这一梯度要求计算两类边缘概率。第一项包含了边缘概率$p(\boldsymbol{w}'_c|\boldsymbol{y}, \boldsymbol{x})$，正是（5.12）式的收缩CRF的边缘分布。第二项包含了一种不同的边缘概率$p(\boldsymbol{w}'_c, \boldsymbol{y}'_c|\boldsymbol{x}_c)$，正是完整CRF所需的边缘概率。只要我们计算了这一梯度，就可以用标准的（像共轭梯度算法）技术来最大化ℓ。对于BFGS，我们的经验是，依赖记忆（memory based）对Hessian矩阵的近似变得有歧义，这是因为凸性被破坏了，就像潜变量CRF这里的情况。有一种实践中的小技巧，当上面的情况发生时，重置对Hessian矩阵的近似。

除此之外，也可以通过期望最大化（EM）来优化ℓ。在EM算法的每次迭代里，当前的 参数向量$\theta^{(j)}$通过如下方式更新。首先在E步（E-step）中，按照$q(\boldsymbol{w}) = p(\boldsymbol{w}|\boldsymbol{y}, \boldsymbol{x}; \theta^{(j)})$计算得到一个松弛函数。其次在M步（M-step）中，一个新的参数向量$\theta^{(j+1)}$计算为

$$\theta^{(j+1)} = \arg\max_{\theta'} \sum_{\boldsymbol{w}'} q(\boldsymbol{w}') \log p(\boldsymbol{y}, \boldsymbol{w}'|\boldsymbol{x}; \theta'). \quad (5.19)$$

直接最大化算法与EM算法是显著地相似的。通过吧q带入(5.19)并求导可看到这一点。所得的梯度与直接梯度（5.18）式几乎一样。所不同的是，EM中的分布$p(\boldsymbol{w}|\boldsymbol{y}, \boldsymbol{x})$是从预先确定的参数获得的，而不是从最大化过程中提取。我们目前尚缺乏两者在潜变量CRF中的对比的经验。

5.2 随机梯度方法

到目前为止，我们讨论过的优化方法适用于整批处理（batch setting）。就是说，它们先扫描完整个训练集后，才能更新模型的参数。如果训练数据的idd样本数量极大，这看起来很浪费。我还有一个经验——会造成梯度极大。我们猜测，训练数据中的大量不同样本会提供近似的关于模型参数的信息，因而有可能不用扫描所有的样本，而是只读取少量样本后，就更新模型的参数。

随机梯度下降Stochastic gradient descent(SGD)是一种简单的优化方法，用来利用这一洞见。基本的想法是，在每次迭代里，随机地选择一个训练样本，然后采用这一样本带来的梯度，并使用一个小的步长。在整批处理的方案里，梯度下降通常不是有话的好方法，因为局部的最速下降方向（就是负的梯度）可能指向与最值点完全不同的地方。所以随机梯度方法含有一个有趣的权衡：L-BFGS的单步的迭代方向比SGD的好很多，但SGD方向的计算要快得多。

为了简化符号，我们只给出线性链的SGD。然而，它可以很容易用于任意的图结构——只要训练数据是iid 的。单个训练样本$(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})$的似然的梯度为：

$$\frac{\partial \ell_i}{\partial \theta_k} = \sum_{t=1}^{T} f_k(y_t^{(i)}, y_{t-1}^{(i)}, \boldsymbol{x}_t^{(i)}) - \sum_{t=1}^{T} \sum_{\boldsymbol{y},\boldsymbol{y}'} f_k(y, y', \boldsymbol{x}_t^{(i)}) p(y, y' | \boldsymbol{x}^{(i)}) - \frac{\theta_k}{N\sigma^2}. \quad (5.20)$$

这与完整的梯度（5.6）一样，只是有两点不同：遍历所有样本的求和被去掉了，以及在规则化项中多出来的因子$1/N$。这保证整批的梯度等于单样本梯度的和，即$\bigtriangledown \ell = \sum_{i=1}^{N} \bigtriangledown \ell_i$，其中我们用$\bigtriangledown \ell_i$来表示单个样本$i$的梯度。

在SGD的每次迭代里，我们随机地选择一个样本$(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})$。然后从原有的向量$\theta^{(m-1)}$中计算新的参数向量如下

$$\theta^{(m)} = \theta^{(m-1)} + \alpha_m \bigtriangledown \ell_i(\theta^{(m-1)}), \quad (5.21)$$

其中，$\alpha_m > 0$，是步长参数，用于控制参数们在多大程度上演着梯度方向更新。如果步长过大，参数会在每次迭代里，沿着所选的样本的方向摆动过远。如果α_m太小，则训练过程会非常慢，甚至在极端的例子中，从数值角度看参数已经收敛了，但其实离最小点还很远呢。

我们希望α_m随着m的增大而减小，以让优化算法收敛到某个唯一的答案。随机近似过程【54,115】提供了收敛性的经典结果，即至少要求$\sum_m \alpha_m = \infty$以及$\sum_m \alpha_m^2 < \infty$。即是说，$\alpha$应该收敛到0，但是不能过快。采用类似$\alpha_m \sim \frac{1}{m}$或$\alpha_m \sim \frac{1}{\sqrt{m}}$的步长是最常用的方法，能满足上面的要求。然而，简单地采用$\alpha_m = 1/m$常常不好，因为第一个步长太大了。实际上，常见的技巧是如下的安排：

$$\alpha_m = \frac{1}{\sigma^2(m_0 + m)}, \quad (5.22)$$

其中，m_0是一个自由参数。对于这一参数，有一个推荐的方法。Leon Bottou【13】的软件包$crfsgd$一次采样一个小的训练数据子集，然后在这一子集上使用各种固定步长α运行SGD。选择α^*，使得该子集的数据在一次运算后的似然是最大的，然后通过让$\alpha_0 = \alpha^*$来确定m_0。

随机梯度下降算法在神经网络文献中又被称为反向传播。过去的许多年中，发展了大量的调节这一算法的技巧【66】。最近，先进在线优化方法【27,43,126,149】重新引发了关注。它也在线地更新参数，但比简单SGD更复杂。Vishwanathan等【149】是第一个在CRF中使用随机梯度方法的应用。

随机梯度方法的主要缺点是它们需要 tuning，这不同于现成的求解器如共轭梯度和L-BFGS。随机梯度方法也不能用于训练数据不是idd 的relational settings，也不能用于小的数据集。对于合适的数据集，共轭梯度方法可以带来可观的加速。

5.3并行

随机梯度下降通过只计算少量的样本来加速计算过程。另一种加快计算的方法是并行地计算不同样本的梯度。因为梯度（5.6）是在所有的样本上求和，可以轻易地把计算划分成多个线程，其中每个线程只计算训练样本的一个子集的梯度。如果CRF在多核机器上运行，那么多个线程将并行运行，极大地加快了梯度计算。这一特点为许多常见的机器学习算法所共有，如Chu等【22】指出的。

原理上，人们也可以吧梯度计算分布到多个机器上，而不是同一台机器的多个核。然而在网络中传递大量的参数可能会成为问题。一种潜在的解决发办法是异步地更新参数。这一想法的最近一个例子是把并行计算融入随机梯度方法的【64】。

5.4近似训练

我们所描述过的训练方法，包括随机和并行梯度方法，都假定易处理的图结构。就是说，我们可以有效地计算归一化函数$Z(\boldsymbol{x})$和边缘分布$p(\boldsymbol{y}_c|\boldsymbol{x})$。这对线性链和树形结构CRF是有效的。早期的CRF关注于它们，即是因为它们的推断易于完成，也因为它们很自然地适合一些任务，如NLP中的序列标注任务。

当图的结构更复杂时，边缘分布和归一化函数都不是易于计算的，使我们必须凭借近似手段。如地4节所说的，存在着大量的近似推断算法。在讨论CRF时，却有一个需要进一步考虑的关键问题——近似推断被嵌入到更大的参数优化过程里。

有两个通用的近似训练CRF的策略【139】：**代理似然surrogate likelihood**策略（通过修改目标函数）和**近似边缘概率approximate marginals**策略（通过近似梯度）。第一种策略要寻找$\ell(\theta)$的替代者（就像BP近似（5.27）），被称作代理似然。它需是易于计算的同时，仍偏爱着好的参数。然后用基于梯度的方法来优化这一代理似然，就像真实似然一样。近似边缘策略使用通用的推断算法来计算边缘$p(\boldsymbol{y}_c|\boldsymbol{x})$的近似，然后在式（5.9）中用近似概率替代精确概率，再用这一近似梯度来完成梯度下降过程。

虽然代理似然和近似边缘概率方法紧密相关，但却是不同的。通常一个代理似然方法直接产生一个近似概率方法，因为就像$\log Z(\boldsymbol{x})$的导数给出了真实的边缘概率，近似的$\log Z(\boldsymbol{x})$也可以看成是边缘概率的近似。这些近似的边缘概率有时被称为**pseudomarginals**【151】。然而，反方向却不总是成立：例如可以证明，存在着这样的边缘概率近似过程，任何的似然函数的导数都不与它对应【131,139】。

代理似然的主要优点是，拥有一个目标函数可以让该方法的特性易于理解——包括对人和对优化过程。先进的优化引擎，如共轭梯度和BFGS，需要一个目标函数来运行。另一方面，近似概率的优势则是更灵活。它易于与任何的推断算法相配合，包括一些技巧，如提前终止BP和MCMC。另外，近似概率方法与随机梯度框架匹配良好。

近似推断与参数估计的相互作用存在这一些无法完全理解的方面。Kulesza和Pereira【60】展示了一个例子，其中感知机算法与max-product置信传播之间有病态的相互作用。相反，代理似然方法不表现出这种病态，正如Wainwright【151】针对凸代理似然而指出的。

为了让讨论具体化，在本节的余下部分，我们讨论几个代理似然和近似概率方法的例子。我们基于伪似然（pseudolikelihood）（5.4.1节）和置信传播（5.4.2)来讨论代理似然方法，而基于置信传播（5.4.2）和MCMC（5.4.3）来讨论近似梯度方法。

5.4.1伪似然

最早之一的代理似然是伪似然【9】。伪似然的想法是让训练目标只依赖于单一变量的条件分布。因为这些分布的归一化常数只依赖于单一变量，它们的计算变得十分有效率。在CRF这里，伪似然是

$$\ell_{PL} = \sum_{s \in V} \log p(y_s|\boldsymbol{y}_{N(s)}, \boldsymbol{x}\theta). \quad (5.23)$$

这里，求和运算要遍历图中所有的输出节点，而$\boldsymbol{y}_{N(s)}$是$\boldsymbol{Y}_s$邻域内的变量$\boldsymbol{N}(s)$的值。（就像在式（5.8）中，我们不显式包含覆盖所有训练样本的求和）

直观地来理解伪似然，它试图让模型分布的局部条件分布$p(y_s|\boldsymbol{y}_{N(s)}, \boldsymbol{x}; \theta)$匹配到训练数据上。又因为模型的条件独立假设，局部的条件分布就足以指明联合分布了。（这与Gibbs采样器背后的动因相似）

通过最大化伪似然来估计参数，即$\hat{\theta}_{PL} = \max_{\theta} \ell_{PL}(\theta)$。通常，最大化过程要使用像limited-memory BFGS的二阶方法，但在原理上，并行计算或随机梯度也可以像真似然那样用于伪似然。此外，也可以像极大似然那样使用规则化。

没有任何意味说明伪似然ℓ_{PL}是一个对真似然的闭逼近（close approximation）。相反，是要让极大伪似然估计$\hat{\theta}_{PL}$逼近极大似然估计$\hat{\theta}_{ML}$。就是说，两个函数的值并不趋同，但它们的极值点趋同。在一些特定的条件下，尤其当模型空间正确时，可以表明，伪似然是渐进地正确的。就是说，当拥有无限多数据时，它将给出真实的参数。

伪似然背后的动因是计算效率。伪似然的计算和优化不用计算$Z(\boldsymbol{x})$和边缘分布。虽然伪似然有时在NLP上被证明有效【146】，但更多时候伪似然的性能槽糕【134】。直观地相当于，Gibbs采样器在序列模型中逐渐混淆。在视觉问题中，对伪似然常见的一个批评是说，“过于重视edge potentials了”【149】。一种直观的解释是说，用伪似然训练的模型学会了利用邻近变量的真值，但在测试时它们却不存在了。

可以采用块blockwise"版本的伪似然来提升性能。其中，局部项包含有模型中更大区域的条件概率。以线性链为例，人们可以考虑一种 per-edge 的伪似然：

$$\ell_{EPL}(\theta) = \sum_{t=1}^{T-1} \log p(y_t, y_{t+1} | y_{t-1}, y_{t+2}, \theta). \quad (5.24)$$

(这里假定，我们的序列被虚标签y_0和y_{T+1}包围，已让上面的表达式正确)

块版本的伪似然是组合似然【34,75】的一个特例。在组合似然中，似然中的每一项不只预测了单一变量或成对变量，而是用户选择的任何尺寸的一块变量。组合似然扩展了标准的伪似然和块伪似然。关于组合似然估计的渐进一致性和正规性有一些理论结果。通常来说，大的块导致更好的参数估计——不管是理论上还是实践中。这带来了训练时间和结果参数质量之间的权衡。

最后，Sutton和McCallum的分段训练方法【134,137】与组合似然有关，但将其看作置信传播算法更易于理解，因而我们将它放到下一节。

5.4.2置信传播

循环置信传播算法（4.2.2节）可用于CRF的近似训练。这既可从伪似然的角度，也可从近似梯度的角度来完成。

在近似梯度算法中，在训练的每次迭代里，我们在训练输入$\boldsymbol{x}$上运行循环BP，为模型的每个团产生一组近似的边缘概率$q(\boldsymbol{y}_c)$。然后，我们把BP边缘概率带入，以近似真实梯度（5.9）。结果是近似的偏微分

$$\frac{\partial \tilde{\ell}}{\partial \theta_{pk}} = \sum_{\Psi_c \in C_p} f_{pk}(\boldsymbol{x}_c, \boldsymbol{y}_c) - \sum_{\Psi_c \in C_p} \sum_{\boldsymbol{y}'} f_{pk}(\boldsymbol{x}_c, \boldsymbol{y}'_c) q(\boldsymbol{y}'_c). \quad (5.25)$$

这可以用来更新当前的参数：

$$\theta_{pk}^{(t+1)} = \theta_{pk}^{(t)} + \alpha \frac{\partial \tilde{\ell}}{\partial \theta_{pk}}, \quad (5.26)$$

其中，$\alpha > 0$，是步长参数。这种方式的优点是它极其简单，尤其对一个外部的随机梯度近似法有用。

更有趣的是，在代理似然框架里也有使用循环BP的可能。我们需要给出真实似然（5.8）的一些代理函数，只要代理似然的梯度与近似BP梯度（5.26）相等。这看起来要求过高，然而幸运的是可以使用4.2.2节描述的Bethe free energy。

前面说过，循环置信传播可被看成一种优化算法。当关于所有的局部一致性置信向量上，最小化了目标函数$\mathcal{O}_{BETHE}(q)$，那么最小值$\min_q \mathcal{O}(q)$可被用来做归一化函数（partition function）的近似。把这一近似带入真似然（5.8），得到针对某个固定的置信向量q的近似似然：

$$\ell_{BETHE}(\theta,q)=\sum_{C_p\in\mathcal{C}}\sum_{\Psi_c\in C_p}\log\Psi_c(\boldsymbol{x}_c,\boldsymbol{y}_c)-\sum_{C_p\in\mathcal{C}}\sum_{\Psi_c\in C_p}q(\boldsymbol{y}_c)\log\frac{q(\boldsymbol{y}_c)}{\Psi_c(\boldsymbol{x}_c,\boldsymbol{y}_c)}+\sum_{s\in Y}(1-d_i)q(y_s)\log q(y_s). \quad (5.27)$$

于是，近似训练可以被看成优化问题$\max_\theta \min_q \ell_{BETHE}(\theta,q)$。这是一个**鞍点问题saddlepoint problem**，关于一个变量最大化（去找最佳的参数）并关于其他最小化（去解近似推断问题）。一种解鞍点问题的方法是协调上升——交替地"固定θ后关于q最小化ℓ_{BETHE}"和"固定b原文如此后关于θ使用一个梯度更新来最大化ℓ_{BETHE}。第一步就是运行循环BP算法。关键在于第二步，（5.27）关于权重θ_k的偏导数正是（5.26）。这正是我们所需要的。

除此之外，也可以使用一个不同的代理似然，即

$$\hat{\ell}(\theta;q)=\log\left[\frac{\prod_{C_p\in\mathcal{C}}\prod_{\Psi_c\in C_p}q(\boldsymbol{y}_c)}{\prod_{s\in Y}q(y_s)^{d_s-1}}\right], \quad (5.28)$$

就是说，与其使用真的联合似然，不如把每个团的近似置信相乘，然后除以节点的置信to avoid overcounting。这样有一个好处——它是树结构模型的真似然的直接扩展。这可通过比较（5.28）和（4.32）看出。可以用我们在【91,94】上给出的Bethe energy的对偶版本来证明这一代理似然的正当性。当BP收敛，对于所得的置信向量q，可以看出$\ell_{BETHE}(\theta,q)=\hat{\ell}(\theta,q)$。这一等式并不总是成立，如当BP不收敛时。

另一种与BP相关的代理似然方法是**分片估计piecewise estimator**【137】。模型的因子被划分成易于操作的子图，并各自独立地训练。当局部特征的信息足够丰富时，这一方法意外地优秀（比 伪似然更好）。Sutton和Minka【139】讨论了分片训练与提前结束置信传播之间的紧密联系。

5.4.3 马尔科夫链蒙特卡洛

马尔科夫链蒙特卡洛（MCMC）推断方法（4.2.1）节可在近似边缘概率框架下用于CRF训练。一旦我们选择了一个马尔科夫链，其稳定分布为$p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})$，那么算法就是，运行多次迭代，然后用所得的近似边缘概率$\hat{p}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})$来近似梯度（5.9）式的真边缘概率。

实践中，MCMC方法却不怎么常用于CRF，因为MCMC方法通常需要多次迭代才收敛，而我们已经指出，训练过程需要反复运行推断任务。

解决这一难点的方法之一是对照散度（contrastive divergence ?）（CD）【50】。其中，式（5,9)中的真边缘概率$p(\boldsymbol{y}_c|\boldsymbol{x})$被只少量迭代的MCMC近似，而马尔科夫链的初始状态（就是$\boldsymbol{y}$的值）被设置成训练数据中的值。CD主要用于潜变量模型，如受限波尔茨曼机。尽管在原理上CD也适用于CRF，但我们没有看到关于此的多少工作。

另一种可能是一种更加新的方法，叫做SampleRanke【155】，whose objective is that the learned parameters score paris of $\boldsymbol{y}_s$ such that their sorted ranking obeys a given supervised ranking（通常被指定为某个固定的、将$\boldsymbol{y}$与真实目标$\boldsymbol{y}$进行比较的打分函数）。可通过MCMC采样器的序列状态 对 来计算 梯度的近似。就像CD，SampleRank在每个MCMC步中执行参数更新，而不去等马尔科夫链收敛。实验表明，SampleRank训练的模型的精度比CD有质的提升【155】。

与近似边缘概率框架不同，将MCMC推断用在代理似然框架则十分困难，因为从所周知地，很难从MCMC方法的样本中获得$\log Z(\boldsymbol{x})$的好的估计。

5.5 Implementation Concerns

不翻译余下的部分了。尽管这本教材对我的帮助极大，但我仍认为它废话太多了。争取后面自己写一篇，期望能给读者一些简单明了、或多或少的帮助吧。