

翻译并解释以下内容，翻译不能有遗漏，解释请通俗易懂且详尽，因为我的数学基础不太好，最后总结要点。

## 19 Undirected graphical models (Markov random fields)

### 19.1 Introduction

In Chapter 10, we discussed directed graphical models (DGMs), commonly known as Bayes nets. However, for some domains, being forced to choose a direction for the edges, as required by a DGM, is rather awkward. For example, consider modeling an image. We might suppose that the intensity values of neighboring pixels are correlated. We can create a DAG model with a 2d lattice topology as shown in Figure 19.1(a). This is known as a causal MRF or a Markov mesh (Abend et al. 1965). However, its conditional independence properties are rather unnatural. In particular, the Markov blanket (defined in Section 10.5) of the node X8 in the middle is the other colored nodes (3, 4, 7, 9, 12 and 13) rather than just its 4 nearest neighbors as one might expect.

An alternative is to use an undirected graphical model (UGM), also called a Markov random field (MRF) or Markov network. These do not require us to specify edge orientations, and are much more natural for some problems such as image analysis and spatial statistics. For example, an undirected 2d lattice is shown in Figure 19.1(b); now the Markov blanket of each node is just its nearest neighbors, as we show in Section 19.2.

Roughly speaking, the main advantages of UGMs over DGMs are: (1) they are symmetric and therefore more “natural” for certain domains, such as spatial or relational data; and (2) discriminative UGMs (aka conditional random fields, or CRFs), which define conditional densities of the form  $p(y|x)$ , work better than discriminative DGMs, for reasons we explain in Section 19.6.1. The main disadvantages of UGMs compared to DGMs are: (1) the parameters are less interpretable and less modular, for reasons we explain in Section 19.3; and (2) parameter estimation is computationally more expensive, for reasons we explain in Section 19.5. See (Domke et al. 2008) for an empirical comparison of the two approaches for an image processing task.

## 翻译:

### 19 非有向图模型 (马尔可夫随机场)

#### 19.1 引言

在第10章中，我们讨论了有向图模型 (DGM)，通常称为贝叶斯网络。然而，对于某些领域来说，强制为边缘指定方向（如DGM所要求的）是相当尴尬的。例如，考虑图像建模。我们可以假设相邻像素的强度值是相关的。我们可以创建一个具有二维格子拓扑结构的有向无环图 (DAG) 模型，如图19.1(a)所示。这被称为因果马尔可夫随机场 (MRF) 或马尔可夫网格 (Abend等, 1965)。然而，其条件独立性特性相当不自然。特别是，节点X8的马尔可夫毯（在第10.5节中定义）是其他有色节点（3、4、7、9、12和13），而不仅仅是其4个最近邻节点，正如我们所期望的那样。

另一种选择是使用非有向图模型 (UGM)，也称为马尔可夫随机场 (MRF) 或马尔可夫网络。这些模型不要求我们指定边的方向，并且对于某些问题（如图像分析和空间统计）来说，更加自然。例如，图19.1(b)展示了一个非有向二维格子；现在，每个节点的马尔可夫毯只是其最近邻节点，如我们在19.2节中展示的那样。

大致来说，UGM相对于DGM的主要优点是：（1）它们是对称的，因此对于某些领域（如空间或关系数据）来说更加“自然”；（2）判别式UGM（即条件随机场，CRF），它们定义了形式为 $p(y|x)$ 的条件密度，效果比判别式DGM更好，原因将在19.6.1节中解释。UGM相对于DGM的主要缺点是：（1）参数不易解释，且不如DGM模块化，原因将在19.3节中解释；（2）参数估计计算开销更大，原因将在19.5节中解释。有关两种方法在图像处理任务中比较的实证研究，请参见(Domke等, 2008)。

## 解释:

### 1. 有向图模型 (DGM) 与非有向图模型 (UGM) 的区别:

- **DGM (有向图模型)**：图中每个节点都有一个方向性关系，通常是指从一个节点指向另一个节点。这种关系使得模型能够明确表达变量之间的因果关系。例如，贝叶斯网络就是一种DGM模型。
- **UGM (非有向图模型)**：图中没有明确的边的方向，表示的是变量之间的相关性，而非因果关系。它适用于更自然地描述某些问题，尤其是空间数据或图像数据的建模。

### 2. 为什么在某些情况下使用UGM比DGM更自然:

- 例如在图像建模中，像素之间的强度值往往是相关的，尤其是邻近像素。非有向图模型更能准确地反映这种相关性，因为它不需要强制为图中的边指定方向，能够更好地表示空间上的依赖关系。

### 3. 马尔可夫随机场 (MRF) 及其特性:

- MRF是一种特殊的非有向图模型，常用于图像处理等任务。它的马尔可夫毯（一个节点的马尔可夫毯是该节点与其直接相邻节点的集合）能帮助模型捕捉节点之间的依赖关系。在图19.1(b)中的UGM中，节点的马尔可夫毯是其最近邻节点，而不是更多的节点，这比DGM更符合直觉。

#### 4. 判别式UGM (条件随机场, CRF) 的优势:

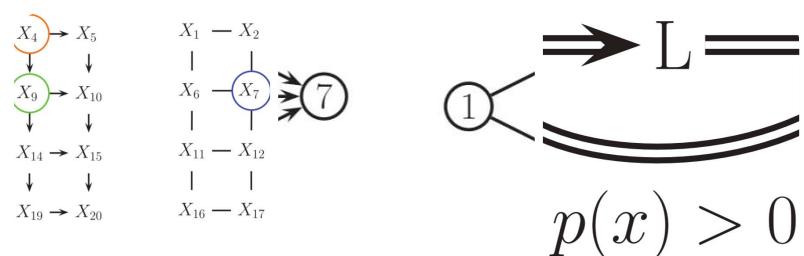
- 判别式UGM如条件随机场 (CRF)，通过定义给定输入条件下的输出概率 (即  $p(y|x)$ )，在许多任务中表现得比有向图模型更好。这是因为条件随机场更加专注于输出条件的建模，而不像贝叶斯网络那样需要同时建模输入和输出的联合概率。

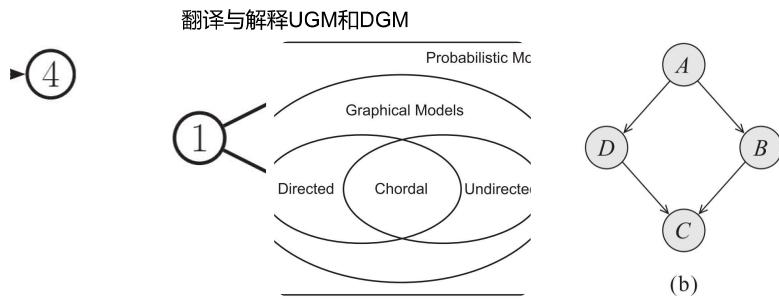
#### 5. UGM的缺点:

- 参数解释性差**: 相比DGM，UGM的参数通常难以直接解释，且模块化较差，这使得它们在某些场景下使用起来不如DGM灵活。
- 计算开销大**: 由于需要处理更复杂的依赖结构，UGM的参数估计过程通常计算量较大。

### 总结要点:

- 有向图模型 (DGM)** : 强制边具有方向，适用于明确的因果关系建模 (如贝叶斯网络)。
- 非有向图模型 (UGM)** : 不需要边的方向，更适合描述如图像、空间数据等领域的相关性。
- 马尔可夫随机场 (MRF)** : 是一种特殊类型的UGM，用于描述节点之间的局部相关性。
- 判别式UGM (条件随机场, CRF)** : 在某些任务中，比DGM更有效，特别是在条件概率建模中。
- UGM的缺点**: 参数解释性差，且计算开销较大。





翻译并解释以下内容，翻译不能有遗漏，解释请通俗易懂且详尽，因为我的数学基础不太好，最后总结要点。

Figure 19.1 (a) A 2d lattice represented as a DAG. The dotted red node X8 is independent of all other nodes (black) given its Markov blanket, which include its parents (blue), children (green) and co-parents (orange). (b) The same model represented as a UGM. The red node X8 is independent of the other black nodes given its neighbors (blue nodes).

Figure 19.2 (a) A DGM. (b) Its moralized version, represented as a UGM.

Figure 19.3 Relationship between Markov properties of UGMs.

Figure 19.4 (a) The ancestral graph induced by the DAG in Figure 19.2(a) wrt  $U = \{2, 4, 5\}$ . (b) The moralized version of (a).

Figure 19.5 DGMs and UGMs can perfectly represent different sets of distributions. Some distributions can be perfectly represented by either DGMs or UGMs; the corresponding graph must be chordal.

Figure 19.6 A UGM and two failed attempts to represent it as a DGM.

## 19.2 Conditional independence properties of UGMs

### 19.2.1 Key properties

UGMs define CI relationships via simple graph separation as follows: for sets of nodes A, B, and C, we say  $xA \perp G xB | xC$  iff C separates A from B in the graph G. This means that, when we remove all the nodes in C, if there are no paths connecting any node in A to any node in B, then the CI property holds. This is called the global Markov property for UGMs. For example, in Figure 19.2(b), we have that  $\{1, 2\} \perp \{6, 7\} | \{3, 4, 5\}$ .

The set of nodes that renders a node t conditionally independent of all the other nodes in the graph is called t's Markov blanket; we will denote this by  $mb(t)$ . Formally,

the Markov blanket satisfies the following property:

$$t \perp\!\!\!\perp V \setminus cl(t) | mb(t) \quad (19.1)$$

where  $cl(t) = mb(t) \cup \{t\}$  is the closure of node  $t$ . One can show that, in a UGM, a node's Markov blanket is its set of immediate neighbors. This is called the undirected local Markov property. For example, in Figure 19.2(b), we have  $mb(5) = \{2, 3, 4, 6\}$ .

From the local Markov property, we can easily see that two nodes are conditionally independent given the rest if there is no direct edge between them. This is called the pairwise Markov property. In symbols, this is written as

$$s \perp\!\!\!\perp t | V \setminus \{s, t\} \iff G_{st} = 0 \quad (19.2)$$

Using the three Markov properties we have discussed, we can derive the following CI properties (amongst others) from the UGM in Figure 19.2(b):

- Pairwise  $1 \perp\!\!\!\perp 7 | \text{rest}$
- Local  $1 \perp\!\!\!\perp \text{rest} | 2, 3$
- Global  $1, 2 \perp\!\!\!\perp 6, 7 | 3, 4, 5$

It is obvious that global Markov implies local Markov which implies pairwise Markov. What is less obvious, but nevertheless true (assuming  $p(x) > 0$  for all  $x$ , i.e., that  $p$  is a positive density), is that pairwise implies global, and hence that all these Markov properties are the same, as illustrated in Figure 19.3 (see e.g., (Koller and Friedman 2009, p119) for a proof).<sup>1</sup> The importance of this result is that it is usually easier to empirically assess pairwise conditional independence; such pairwise CI statements can be used to construct a graph from which global CI statements can be extracted.

### 19.2.2 An undirected alternative to d-separation

We have seen that determining CI relationships in UGMs is much easier than in DGMs, because we do not have to worry about the directionality of the edges. In this section, we show how to determine CI relationships for a DGM using a UGM.

It is tempting to simply convert the DGM to a UGM by dropping the orientation of the edges, but this is clearly incorrect, since a v-structure  $A \rightarrow B \leftarrow C$  has quite different CI properties than the corresponding undirected chain  $A - B - C$ . The latter graph incorrectly states that  $A \perp\!\!\!\perp C | B$ . To avoid such incorrect CI statements, we can add edges between the “unmarried” parents  $A$  and  $C$ , and then drop the arrows from the edges, forming (in this case) a fully

connected undirected graph. This process is called moralization. Figure 19.2(b) gives a larger example of moralization: we interconnect 2 and 3, since they have a common child 5, and we interconnect 4, 5 and 6, since they have a common child 7.

Unfortunately, moralization loses some CI information, and therefore we cannot use the moralized UGM to determine CI properties of the DGM. For example, in Figure 19.2(a), using d-separation, we see that  $4 \perp 5|2$ . Adding a moralization arc  $4 - 5$  would lose this fact (see Figure 19.2(b)). However, notice that the 4-5 moralization edge, due to the common child 7, is not needed if we do not observe 7 or any of its descendants. This suggests the following approach to determining if  $A \perp B|C$ . First we form the ancestral graph of DAG G with respect to  $U = A \cup B \cup C$ . This means we remove all nodes from G that are not in U or are not ancestors of U. We then moralize this ancestral graph, and apply the simple graph separation rules for UGMs. For example, in Figure 19.4(a), we show the ancestral graph for Figure 19.2(a) using  $U = \{2, 4, 5\}$ . In Figure 19.4(b), we show the moralized version of this graph. It is clear that we now correctly conclude that  $4 \perp 5|2$ .

### 19.2.3 Comparing directed and undirected graphical models

Which model has more “expressive power”, a DGM or a UGM? To formalize this question, recall that we say that G is an I-map of a distribution p if  $I(G) \subseteq I(p)$ . Now define G to be perfect map of p if  $I(G) = I(p)$ , in other words, the graph can represent all (and only) the CI properties of the distribution. It turns out that DGMs and UGMs are perfect maps for different sets of distributions (see Figure 19.5). In this sense, neither is more powerful than the other as a representation language.

As an example of some CI relationships that can be perfectly modeled by a DGM but not a UGM, consider a v-structure  $A \rightarrow C \leftarrow B$ . This asserts that  $A \perp B$ , and  $A \perp B|C$ . If we drop the arrows, we get  $A-C-B$ , which asserts  $A \perp B|C$  and  $A \perp B$ , which is incorrect. In fact, there is no UGM that can precisely represent all and only the two CI statements encoded by a vstructure. In general, CI properties in UGMs are monotonic, in the following sense: if  $A \perp B|C$ , then  $A \perp B|(C \cup D)$ . But in DGMs, CI properties

can be non-monotonic, since conditioning on extra variables can eliminate conditional independencies due to explaining away.

As an example of some CI relationships that can be perfectly modeled by a UGM but not a DGM, consider the 4-cycle shown in Figure 19.6(a). One attempt to model this with a DGM is shown in Figure 19.6(b). This correctly asserts that  $A \perp C|B,D$ . However, it incorrectly asserts that  $B \perp D|A$ . Figure 19.6(c) is another incorrect DGM: it correctly encodes  $A \perp C|B,D$ , but incorrectly encodes  $B \perp D$ . In fact there is no DGM that can precisely represent all and only the CI statements encoded by this UGM. Some distributions can be perfectly modeled by either a DGM or a UGM; the resulting graphs are called decomposable or chordal. Roughly speaking, this means the following: if we collapse together all the variables in each maximal clique, to make “mega-variables”, the resulting graph will be a tree. Of course, if the graph is already a tree (which includes chains as a special case), it will be chordal. See Section 20.4.1 for further details.

It seems that you've uploaded several images. How would you like me to assist you with them? Do you want to analyze, explain, or modify any of them? Let me know how I can help!

include analyze, explain any of them

下面把你给的文字与配图逐条完整翻译，再用尽量通俗的方式解释与分析图像，最后给出要点小结。

## 一、逐条翻译（不遗漏）

### 图 19.1

- (a) 用一个 DAG 表示的二维格子。虚线红色节点  $X_8$  在给定其马尔可夫毯时，与所有其他（黑色）节点独立；它的马尔可夫毯包含它的父节点（蓝色）、子节点（绿色）以及共同父节点（橙色）。

(b) 用 UGM 表示相同模型。红色节点  $X_8$  在给定其邻居（蓝色节点）时，与其他黑色节点独立。

### 图 19.2

- (a) 一个 DGM。
- (b) 它的“道德化 (moralized)”版本，用 UGM 表示。

### 图 19.3

UGM 的马尔可夫性质之间的关系。

### 图 19.4

- (a) 由图 19.2(a) 的 DAG 关于  $U = \{2, 4, 5\}$  诱导的祖先图 (ancestral graph)。
- (b) (a) 的“道德化”版本。

### 图 19.5

DGM 与 UGM 能完美表示 (perfectly represent) 不同的分布集合。有些分布既可以被 DGM 完美表示，也可以被 UGM 完美表示；其对应的图必须是弦图 (chordal)。

### 图 19.6

一个 UGM 以及将其表示为 DGM 的两次失败尝试。

## 19.2 UGM 的条件独立性质

### 19.2.1 关键性质

UGM 通过简单的图分离 (graph separation) 来定义条件独立 (CI) 关系：对节点集合  $A, B, C$ ，若并且仅若  $C$  在图  $G$  中把  $A$  与  $B$  分隔开，则称  $x_A \perp_G x_B \mid x_C$ 。这意味着把  $C$  中的节点“移除/遮挡”后，若从  $A$  中任何一个节点到  $B$  中任何一个节点都不存在路径，则该 CI 性质成立。这称为 UGM 的全局马尔可夫性质 (global Markov property)。例如，在图 19.2(b) 中，有  $\{1, 2\} \perp \{6, 7\} \mid \{3, 4, 5\}$ 。

使得一个节点  $t$  与图中其余所有节点条件独立的那一组节点称为  $t$  的马尔可夫毯 (Markov blanket)，记作  $\text{mb}(t)$ 。形式化地，马尔可夫毯满足

$$t \perp V \setminus \text{cl}(t) \mid \text{mb}(t) \quad (19.1)$$

其中  $\text{cl}(t) \triangleq \text{mb}(t) \cup \{t\}$  是  $t$  的闭包。在 UGM 中可以证明：节点的马尔可夫毯就是它的直接邻居集合。这称为无向的局部马尔可夫性质 (undirected local Markov property)。例如，在图 19.2(b) 中， $\text{mb}(5) = \{2, 3, 4, 6\}$ 。

由局部性质可立刻看出：如果两个节点之间没有直接边，则在给定其它所有节点的条件下它们相互独立。这称为**成对马尔可夫性质** (pairwise Markov property)。符号上写作

$$s \perp t \mid V \setminus \{s, t\} \Leftrightarrow G_{st} = 0 \quad (19.2)$$

其中  $G_{st} = 0$  表示  $s$  与  $t$  之间无边。

使用上面三个马尔可夫性质，我们可以从图 19.2(b) 的 UGM 推出（以及其它）如下 CI 性质：

- **成对**:  $1 \perp 7 \mid \text{rest}$  (给定其余所有节点时 1 与 7 独立)
- **局部**:  $1 \perp \text{rest} \mid 2, 3$  (给定 2、3 时，1 与其余节点独立)
- **全局**:  $\{1, 2\} \perp \{6, 7\} \mid \{3, 4, 5\}$

显然，“全局马尔可夫  $\Rightarrow$  局部马尔可夫  $\Rightarrow$  成对马尔可夫”。不那么显然但确实成立的是（假设对所有  $x$  都有  $p(x) > 0$ ，即  $p$  是正的密度）：**成对马尔可夫  $\Rightarrow$  全局马尔可夫**，因而这三种马尔可夫性质是等价的，如图 19.3 所示（证明可参见例如 Koller and Friedman 2009，第 119 页）。该结果的重要性在于：通常更容易在经验数据上检验**成对**的条件独立；这些成对 CI 语句可以用来构建图结构，再从图中读出**全局**的 CI 语句。

### 19.2.2 d-分离的无向替代方法

我们已经看到，在 UGM 中判断 CI 关系比在 DGM 中容易得多，因为不用关心边的方向。本节展示如何用 UGM 来判断一个 DGM 的 CI 关系。

直接把 DGM 的边去掉方向改成 UGM 是错误的——因为  $v$  结构  $A \rightarrow B \leftarrow C$  与对应的无向链  $A - B - C$  的 CI 性质完全不同。后者会错误地给出  $A \perp C \mid B$ 。为避免这种错误，我们可以把“未成婚”的父节点  $A$  与  $C$  补上一条边，再去掉所有箭头；这会得到一个（在此例中是）完全连通的无向图。这个过程称为**道德化** (moralization)。图 19.2(b) 给出了一个更大的道德化例子：因为 2 和 3 共享子节点 5，需要连上 2—3；又因为 4、5、6 共享子节点 7，需要把它们互连。

不幸的是，道德化会丢失某些 CI 信息，因此不能直接用道德化后的 UGM 去判定原 DGM 的 CI。比如在图 19.2(a) 中，用 d-分离可得  $4 \perp 5 \mid 2$ ；若加上道德化的边 4—5（见图 19.2(b))，这一事实就丢了。注意：由共同子节点 7 引入的 4—5 边，在**没有观测到 7 或其后代**时其实并不需要。由此得到如下判定  $A \perp B \mid C$  的方法：

1. 先对 DAG  $G$  构造关于  $U = A \cup B \cup C$  的**祖先图**：删去所有既不在  $U$  中、也不是  $U$  的祖先的节点；
2. 对这个祖先图做**道德化**；
3. 在得到的 UGM 上应用**无向的图分离规则**判断独立性。

例如，图 19.4(a) 给出了对图 19.2(a) 采用  $U = \{2, 4, 5\}$  得到的祖先图；图 19.4(b) 是其道德化版本。由此我们正确得出  $4 \perp 5 \mid 2$ 。

### 19.2.3 有向与无向图模型的比较

“DGM 与 UGM 哪个表达能力更强？”形式化地回忆：若  $I(G) \subseteq I(p)$  则称  $G$  是分布  $p$  的 I-map。若  $I(G) = I(p)$ ，即图能表示恰好该分布的所有且仅有的 CI 性质，则称  $G$  是  $p$  的完美映射 (perfect map)。事实是：DGM 与 UGM 分别是不同分布族的完美映射（见图 19.5）。就“表示语言”而言，二者都不比对方更强。

- DGM 能而 UGM 不能的例子：v 结构  $A \rightarrow C \leftarrow B$  同时断言  $A \perp B$  和  $A \perp B \mid C$ 。若去方向变为  $A - C - B$ ，它断言的是  $A \perp B \mid C$  以及  $A \perp B$ （二者同时），这与 v 结构编码的不一致。事实上，没有任何 UGM 能精确表达 v 结构的仅这两条 CI 语句。一般地，UGM 中的 CI 具有单调性：若  $A \perp B \mid C$ ，则对任意  $D$ ，有  $A \perp B \mid (C \cup D)$ 。而 DGM 中的 CI 可以是非单调的，因为加入额外变量会因“解释互斥/解释消除 (explaining away)”而消除原有的条件独立。
- UGM 能而 DGM 不能的例子：看图 19.6(a) 的四环。图 19.6(b) 的 DGM 试图表示它，确实表达了  $A \perp C \mid B, D$ ，但错误地表达了  $B \perp D \mid A$ 。图 19.6(c) 的另一种 DGM 也失败：它也正确表达  $A \perp C \mid B, D$ ，却错误地表达了  $B \perp D$ 。事实上，没有任何 DGM 能精确表达该 UGM 编码的全部且仅有的 CI 语句。
- 两者都能完美建模的一类图称为可分解/弦图 (decomposable / chordal)。直观地说：如果把每个极大团 (maximal clique) 里的变量“捆绑”成一个超级变量，那么得到的图是树。当然，如果原图就是树（链是其特例），它就是弦图。详见 20.4.1 节。

## 二、配图逐张解析（结合直觉）

### 图 19.1 (格子图)

- (a) 用 DAG 做二维格子时，中间的  $X_8$  的马尔可夫毯很怪：不仅是四个最近邻，还包括“父—子—共同父”一大圈（蓝/绿/橙）。这违背我们对图像“只跟四邻相关”的直觉。
- (b) 换成 UGM， $X_8$  的马尔可夫毯正好就是四个近邻（蓝色）。这符合图像/空间数据“局部相互作用”的自然假设。

### 图 19.2 (DGM 与道德化)

- (a) 一个带 v 结构的 DGM。
- (b) 把每个“未婚父母”连边，再去方向，得到 UGM。它保留了许多依赖关系，但可能引入多余边，从而丢掉部分 CI 信息（例如 4 与 5 在给定 2 时独立这件事）。

### 图 19.3 (马尔可夫性质关系)

- 大箭头  $G \Rightarrow L \Rightarrow P$  表示全局  $\Rightarrow$  局部  $\Rightarrow$  成对总是成立；

- 下面的“回箭头”在标注  $p(x) > 0$  时成立，表示成对  $\Rightarrow$  全局也成立，因此三者等价（这与 Hammersley–Clifford 定理一致）。

#### 图 19.4 (祖先图 + 道德化判 CI)

- 只保留  $U = \{2, 4, 5\}$  及其祖先，减少无关节点干扰；
- 再道德化后用无向分离读结论，可正确得到  $4 \perp 5 \mid 2$ 。

#### 图 19.5 (表达力对比)

- “概率模型”最大圈里包含“图模型”；其内是“有向”“无向”；二者交集的“弦图”是两者都可完美表示的那部分。说明 DGM 与 UGM 各有所长，没有谁全面碾压谁。

#### 图 19.6 (四环与失败的 DGM)

- 左为 UGM 四环，要求“相邻有边，隔一个点条件独立”。
- 两个右图尝试用 DGM 表示：一个多断言了  $B \perp D \mid A$ ，另一个直接断言  $B \perp D$ 。两者都与原 UGM 的 CI 集合不一致，因此失败。

### 三、通俗解释与操作要点

- 三种马尔可夫性质**
  - 成对**：两点没边  $\Rightarrow$  给定其余所有点它俩独立。
  - 局部**：每个点在给定“邻居”（UGM）或“父/子/共同父”（DGM）后，与其余所有点独立。
  - 全局**：删去条件集合节点后，若 A 与 B 之间没有路径，则独立。
  - 在 UGM 且  $p(x) > 0$  时，这三者等价。实践中先检查“有没有边”（最容易），再据此推出复杂的全局独立。
- 用 UGM 判 DGM 的独立性（代替 d-分离）**
  - 取关心的节点  $A, B, C$ ，保留它们和它们的**祖先**（得祖先图）；
  - 把祖先图**道德化**（共同子节点的父母两两连边，再去方向）；
  - 在这个 UGM 上做**图分离**：删去  $C$ ，看  $A, B$  是否断开。  
这样能避免把与问题无关的  $\vee$  结构错误地连边导致信息丢失。
- 表达能力**
  - DGM 擅长**：表示“解释互斥”（ $\vee$  结构）这类**非单调** CI。
  - UGM 擅长**：表示“对称的局部相互作用”（如格子、社交关系）。其 CI **单调**：条件越多，独立性只会增加不会减少。
  - 两者都行**：**弦图**（把每个极大团缩成点后变成树）。树/链天然属于此类。
- 记号速查**

- $\text{mb}(t)$ : 节点  $t$  的马尔可夫毯；UGM 中即“邻居”。
  - $\text{cl}(t) = \text{mb}(t) \cup \{t\}$ : 闭包。
  - $G_{st} = 0$ : 图  $G$  中  $s$  与  $t$  无边。
  - $I(G)$ : 从图中读出的所有 CI 语句集合。
  - I-map / perfect map: 图是否（完全）匹配分布的 CI 集合。
- 

## 四、要点小结

- 在 UGM 中，判断独立性很直观：**删条件，看是否断路**。节点的马尔可夫毯=邻居。
- **成对/局部/全局**三种马尔可夫性质在正分布下等价；实践里可先查“有没有边”。
- 直接把 DGM 去方向会犯错；正确方法：**祖先图 → 道德化 → 无向分离**。
- DGM 能表达 v 结构的“解释互斥”等**非单调** CI；UGM 的 CI 单调、适合**对称本地相互作用**（如图像格子）。
- 有些分布两者都能完美表示——对应图必须是**弦图**（把极大团缩成点后是树）。
- 四环 UGM 是 UGM 的强项：**没有任何** DGM 能准确等价表示其全部 CI。

这一节主要在讲什么？我不太懂

已思考 11s

没关系～这小节的核心其实就三件事：**换一种图（无向图）来描述“谁跟谁相关”，并教你**  
**怎么从图上读出“谁独立于谁”，以及它和有向图的关系与差别。**

---

**用更直白的话说，这一节在讲：**

1. **为什么要用“无向图模型（UGM/MRF）”**
  - 有向图（DGM/贝叶斯网）要给每条边定方向，在图像/空间这类对称场景里很别扭（图 19.1 左）。
  - 无向图不用方向，更符合“只跟邻居相关”的直觉（图 19.1 右）：一个点的**马尔可夫毯**=它的**邻居**。
2. **怎么在 UGM 里判断条件独立（CI）**
  - 规则超简单：**把条件集合 C 的点拿掉，看 A 和 B 之间是否断路**（图分离）。断了就  $A \perp B \mid C$ 。

- 在 UGM 中，三种马尔可夫性质是等价的（图 19.3）：
  - 成对：两点没边  $\Rightarrow$  在给定其它所有点时独立；
  - 局部：给定“邻居”，该点与其余所有点独立；
  - 全局：删掉条件点后，A 与 B 之间无路径。
- 这意味着：**只要能判断“有没有边/有没有路”就能读出所有独立关系。**

### 3. 怎么用 UGM 的方法来判断 DGM 里的独立 (d-分离的替代)

- 直接把 DGM 去掉方向会出错 ( $\vee$  结构会被误解)。
- 正确三步（图 19.4）：
  1. 只保留关心的点  $A, B, C$  和它们的**祖先**（得“祖先图”）；
  2. **道德化**：把每个共同子的父母两两补边，再去掉箭头；
  3. 在这个无向图上用**图分离**判断独立。
- 这样能保留对你这次提问真正相关的 CI 信息。

### 4. DGM vs UGM 的“表达力”对比（图 19.5, 19.6）

- 两者各有擅长，谁也不是绝对更强：
  - DGM 擅长  $\vee$  结构/“**解释互斥**”：条件独立是非单调的（多 conditioning 可能打破独立）。
  - UGM 的**独立是单调的**： $A \perp B \mid C \Rightarrow A \perp B \mid C \cup D$ 。
- 有些分布**两种都能完美表示**，对应的图叫**弦图** (chordal)（把每个极大团缩成点后是一棵树）。
- 典型对比：
  - $\vee$  结构  $A \rightarrow C \leftarrow B$  只能被 DGM 精确表达；UGM 做不到不多不少地表达它的两条 CI。
  - **四环 UGM**（图 19.6 左）没有任何 DGM 能完全等价表示（右两图都是失败例子）。

## 一眼能用的“读图小抄”

- **UGM 判断独立：**
  1. 想看  $A \perp B \mid C$ ；
  2. 在图上删除 C 及其连边；
  3. 若 A 与 B 不连通  $\Rightarrow$  独立；否则相关。
  4. 某点的**马尔可夫毯**=它的**邻居**。
- **DGM 判断独立（用 UGM 辅助）：**

1. 取  $U = A \cup B \cup C$  与其祖先;
  2. 道德化 (共同父母补边, 去箭头) ;
  3. 在该无向图上做上面 1-3 步。
- 

## 图里到底想说明什么?

- 图 19.1: DGM 在格子上给出的马尔可夫毯很“绕”, UGM 则“近邻即毯”, 更自然。
  - 图 19.2–19.4: 展示“道德化 + 祖先图”这套把 DGM 问题变简单的方法。
  - 图 19.3: 在 UGM (且  $p(x) > 0$ ) 里, “成对=局部=全局”。
  - 图 19.5: DGM 与 UGM 的能力像两个交集集合, 交集是弦图。
  - 图 19.6: 四环 UGM 无法被任何 DGM 完美复刻; 反过来  $\vee$  结构也无法被 UGM 完美复刻。
- 

## 本节最重要的结论 (提炼)

- UGM 的独立判断=删点断路; 节点的马尔可夫毯=邻居。
- 在 UGM (正分布) 中, 成对/局部/全局三种马尔可夫性质等价。
- 要在 DGM 中判独立, 可用: 祖先图 → 道德化 → 无向分离。
- DGM 与 UGM 各有拿手:
  - DGM:  $\vee$  结构、解释互斥、非单调 CI;
  - UGM: 对称的局部相互作用 (图像/空间) 、单调 CI。
- 共同可完美表示的就是弦图 (把极大团缩点后是树) 。

如果你愿意, 我可以用一个小例子 (手绘小图或表) 带你跑一遍“删点断路”的判断流程。

翻译并解释以下内容, 翻译不能有遗漏, 解释请通俗易懂且详尽, 因为我的数学基础不太好, 最后总结要点。

### 19.3 Parameterization of MRFs

Although the CI properties of UGM are simpler and more natural than for DGMs, representing the joint distribution for a UGM is less natural than for a DGM, as we see below.

#### 19.3.1 The Hammersley-Clifford theorem

Since there is no topological ordering associated with an

undirected graph, we can't use the chain rule to represent  $p(y)$ . So instead of associating CPDs with each node, we associate potential functions or factors with each maximal clique in the graph. We will denote the potential function for clique  $c$  by  $\psi_c(y_c|\theta_c)$ . A potential function can be any non-negative function of its arguments. The joint distribution is then defined to be proportional to the product of clique potentials. Rather surprisingly, one can show that any positive distribution whose CI properties can be represented by a UGM can be represented in this way. We state this result more formally below.

Theorem 19.3.1 (Hammersley-Clifford). A positive distribution  $p(y) > 0$  satisfies the CI properties of an undirected graph  $G$  if and only if it can be represented as a product of factors, one per maximal clique, i.e.,

$$p(y|\theta) = \frac{1}{Z(\theta)} \prod_{c \in C} \psi_c(y_c|\theta_c) \quad (19.3)$$

where  $C$  is the set of all the (maximal) cliques of  $G$ , and  $Z(\theta)$  is the partition function given by

$$Z(\theta) = \sum_{y \in \mathcal{Y}} \prod_{c \in C} \psi_c(y_c|\theta_c) \quad (19.4)$$

Note that the partition function is what ensures the overall distribution sums to 1.

The proof was never published, but can be found in e.g., (Koller and Friedman 2009).

For example, consider the MRF in Figure 10.1(b). If  $p$  satisfies the CI properties of this graph then we can write  $p$  as follows:

$$p(y|\theta) = \frac{1}{Z(\theta)} \psi_{123}(y_1, y_2, y_3) \psi_{234}(y_2, y_3, y_4) \psi_{35}(y_3, y_5) \quad (19.5)$$

where

$$Z = \sum_{y \in \mathcal{Y}} \psi_{123}(y_1, y_2, y_3) \psi_{234}(y_2, y_3, y_4) \psi_{35}(y_3, y_5) \quad (19.6)$$

There is a deep connection between UGMs and statistical physics. In particular, there is a model known as the Gibbs distribution, which can be written as follows:

$$p(y|\theta) = \frac{1}{Z(\theta)} \exp(-\beta E(y|\theta)) \quad (19.7)$$

where  $E(y) > 0$  is the energy associated with the variables in clique  $c$ . We can convert this to a UGM by defining

$$\psi_c(y_c|\theta_c) = \exp(-E(y_c|\theta_c)) \quad (19.8)$$

We see that high probability states correspond to low energy configurations. Models of this form are known as energy based models, and are commonly used in physics and biochemistry, as well as some branches of machine learning (LeCun et al. 2006).

Note that we are free to restrict the parameterization to

the edges of the graph, rather than the maximal cliques.

This is called a pairwise MRF. In Figure 10.1(b), we get

$$p(y|\theta) \propto$$

$$\psi_{12}(y_1, y_2)\psi_{13}(y_1, y_3)\psi_{23}(y_2, y_3)\psi_{24}(y_2, y_4)\psi_{34}(y_3, y_4)\psi_{35}(y_3, y_5) \quad (19.9)$$

$$\propto s \sim t \psi_{st}(y_s, y_t) \quad (19.10)$$

This form is widely used due to its simplicity, although it is not as general.

### 19.3.2 Representing potential functions

If the variables are discrete, we can represent the potential or energy functions as tables of (non-negative) numbers, just as we did with CPTs. However, the potentials are not probabilities. Rather, they represent the relative "compatibility" between the different assignments to the potential. We will see some examples of this below.

A more general approach is to define the log potentials as a linear function of the parameters:

$$\log \psi_c(y_c) = \varphi_c(y_c)^T \theta_c \quad (19.11)$$

where  $\varphi_c(x_c)$  is a feature vector derived from the values of the variables  $y_c$ . The resulting log probability has the form

$$\log p(y|\theta) = c \varphi_c(y_c)^T \theta_c - Z(\theta) \quad (19.12)$$

This is also known as a maximum entropy or a log-linear model.

For example, consider a pairwise MRF, where for each edge, we associate a feature vector of length  $K_2$  as follows:

$$\varphi_{st}(y_s, y_t) = [\dots, I(y_s = j, y_t = k), \dots] \quad (19.13)$$

If we have a weight for each feature, we can convert this into a  $K \times K$  potential function as follows:

$$\psi_{st}(y_s = j, y_t = k) = \exp([\theta^T \varphi_{st}]jk) = \exp(\theta_{st}(j, k)) \quad (19.14)$$

So we see that we can easily represent tabular potentials using a log-linear form. But the log-linear form is more general.

To see why this is useful, suppose we are interested in making a probabilistic model of English spelling. Since certain letter combinations occur together quite frequently (e.g., "ing"), we will need higher order factors to capture this. Suppose we limit ourselves to letter trigrams. A tabular potential still has  $26^3 = 17,576$  parameters in it. However, most of these triples will never occur.

An alternative approach is to define indicator functions that look for certain "special" triples, such as "ing", "qu-", etc. Then we can define the potential on each trigram as

follows:

$$\psi(y_{t-1}, y_t, y_{t+1}) = \exp(k \theta_k \varphi_k(y_{t-1}, y_t, y_{t+1})) \quad (19.15)$$

where  $k$  indexes the different features, corresponding to "ing", "qu-", etc., and  $\varphi_k$  is the corresponding binary feature function. By tying the parameters across locations, we can define the probability of a word of any length using

$$p(y|\theta) \propto \exp(t k \theta_k \varphi_k(y_{t-1}, y_t, y_{t+1})) \quad (19.16)$$

This raises the question of where these feature functions come from. In many applications, they are created by hand to reflect domain knowledge (we will see examples later), but it is also possible to learn them from data, as we discuss in Section 19.5.6.

已思考 14s

下面先**完整翻译**（不遗漏原文任何信息与公式编号），再做**通俗解释**，最后给出**要点总结**。

---

## 一、逐句翻译

### 19.3 马尔可夫随机场 (MRF) 的参数化

尽管 UGM 的条件独立 (CI) 性质比 DGM 更简单、更自然，但如下面所见，用 UGM 来表示一个联合分布要比用 DGM 不那么自然。

#### 19.3.1 Hammersley–Clifford 定理

由于无向图没有与之对应的拓扑排序，我们无法像 DGM 一样用链式法则来表示  $p(\mathbf{y})$ 。因此，我们不是给每个结点关联条件概率分布 (CPD)，而是给图中的每个**极大团**

(maximal clique) **关联一个势函数或因子**。我们用  $\psi_c(\mathbf{y}_c | \theta_c)$  表示团  $c$  的势函数。势函数可以是其自变量上的任意**非负**函数。随后把联合分布定义为各团势的乘积的**比例形式**。

颇为出人意料的是，可以证明：任何其 CI 性质可以由某个 UGM 表示的**正分布**（对所有  $\mathbf{y}$  有  $p(\mathbf{y}) > 0$ ）都能用这种方式表示。我们在下面更形式化地给出该结果。

**定理 19.3.1 (Hammersley–Clifford)**。若  $p(\mathbf{y}) > 0$ ，当且仅当  $p$  能表示为对每个极大团取一个因子并将其相乘时， $p$  满足无向图  $G$  的 CI 性质，即

$$p(\mathbf{y} | \theta) = \frac{1}{Z(\theta)} \prod_{c \in C} \psi_c(\mathbf{y}_c | \theta_c) \quad (19.3)$$

其中  $C$  是  $G$  的 (所有) 极大团的集合， $Z(\theta)$  是**配分函数**，定义为

$$Z(\theta) = \sum_{\mathbf{y}} \prod_{c \in C} \psi_c(\mathbf{y}_c \mid \theta_c) \quad (19.4)$$

注意：配分函数确保整体分布的和为 1。

该定理的证明从未正式发表，但可在例如 (Koller and Friedman, 2009) 中找到。

例如，考虑图 10.1(b) 的 MRF。若  $p$  满足该图的 CI 性质，则我们可以把  $p$  写成：

$$p(\mathbf{y} \mid \theta) = \frac{1}{Z(\theta)} \psi_{123}(y_1, y_2, y_3) \psi_{234}(y_2, y_3, y_4) \psi_{35}(y_3, y_5) \quad (19.5)$$

其中

$$Z = \sum_{\mathbf{y}} \psi_{123}(y_1, y_2, y_3) \psi_{234}(y_2, y_3, y_4) \psi_{35}(y_3, y_5) \quad (19.6)$$

UGM 与统计物理之间有着深刻联系。特别地，有一种称为\*\*吉布斯分布 (Gibbs distribution) \*\*的模型可写为

$$p(\mathbf{y} \mid \theta) = \frac{1}{Z(\theta)} \exp(-\sum_c E(\mathbf{y}_c \mid \theta_c)) \quad (19.7)$$

其中  $E(\mathbf{y}_c) > 0$  是与团  $c$  中变量相关联的能量。我们可以通过定义

$$\psi_c(\mathbf{y}_c \mid \theta_c) = \exp(-E(\mathbf{y}_c \mid \theta_c)) \quad (19.8)$$

把它转化为 UGM。可见，高概率状态对应于低能量的构型。这类形式的模型称为\*\*能量基 (energy-based) \*\*模型，在物理、生物化学以及机器学习的某些分支中都很常用 (LeCun 等, 2006)。

注意，我们也可以只把参数化限制在图的边上，而不是极大团上。这称为成对 MRF (pairwise MRF)。在图 10.1(b) 中，我们得到

$$p(\mathbf{y} \mid \theta) \propto \psi_{12}(y_1, y_2) \psi_{13}(y_1, y_3) \psi_{23}(y_2, y_3) \psi_{24}(y_2, y_4) \psi_{34}(y_3, y_4) \psi_{35}(y_3, y_5) \quad (19.9)$$

$$\propto \prod_{s \sim t} \psi_{st}(y_s, y_t) \quad (19.10)$$

这种形式因其简单而被广泛使用，尽管它不如一般情形那样具有通用性。

### 19.3.2 表示势函数

如果变量是离散的，我们可以像处理 CPT 一样，用（非负）数值的表来表示势或能量函数。不过，势并不是概率；它们表示的是对某个赋值的相对“相容性”。我们随后会看到一些例子。

一种更一般的方法是把\*\*对数势 (log potential) \*\*定义为参数的线性函数:

$$\log \psi_c(\mathbf{y}_c) = \phi_c(\mathbf{y}_c)^\top \theta_c \quad (19.11)$$

其中  $\phi_c(\mathbf{y}_c)$  是由  $\mathbf{y}_c$  的取值构造出的**特征向量**。由此得到的**对数概率**具有如下形式:

$$\log p(\mathbf{y} \mid \theta) = \sum_c \phi_c(\mathbf{y}_c)^\top \theta_c - \log Z(\theta) \quad (19.12)$$

这也称为**最大熵**或\*\*对数线性 (log-linear) \*\*模型。

例如, 考虑一个成对 MRF。对每条边, 我们关联一个长度为  $K^2$  的特征向量, 方式如下:

$$\phi_{st}(y_s, y_t) = [\dots, I(y_s = j, y_t = k), \dots] \quad (19.13)$$

如果我们为每个特征配一个权重, 就可以把它转换成一个  $K \times K$  的势函数:

$$\psi_{st}(y_s = j, y_t = k) = \exp([\theta_{st}^\top \phi_{st}]_{jk}) = \exp(\theta_{st}(j, k)) \quad (19.14)$$

因此我们看到, 用对数线性形式可以很容易地表示表格势; 而且对数线性形式**更一般**。

为说明其用途, 假设我们想建立英文拼写的概率模型。由于某些字母组合 (如 "ing") 经常共现, 我们需要**更高阶的因子**来刻画这种现象。假设我们把自己限制在字母三元组 (trigram)。一个表格势仍然需要  $26^3 = 17,576$  个参数; 然而其中大多数三元组从不会出现。

另一种做法是定义一些只关注“特别”的三元组的**指示函数**, 例如 “ing”、“qu-”等。然后我们可以把每个三元组上的势定义为

$$\psi(y_{t-1}, y_t, y_{t+1}) = \exp(\sum_k \theta_k \phi_k(y_{t-1}, y_t, y_{t+1})) \quad (19.15)$$

其中  $k$  枚举不同的特征 (对应 “ing”、“qu-” 等),  $\phi_k$  是相应的二值特征函数。通过在不同位置**共享参数**, 我们可以定义任意长度单词的概率:

$$p(\mathbf{y} \mid \theta) \propto \exp(\sum_t \sum_k \theta_k \phi_k(y_{t-1}, y_t, y_{t+1})) \quad (19.16)$$

这引出了一个问题: 这些特征函数从哪里来? 在许多应用中, 它们由人工构造以反映领域知识 (后面会举例); 但也可以从**数据中学习**它们, 我们将在 19.5.6 节讨论。

## 二、通俗解释 (把概念讲清)

### 1) 为什么 UGM 要用“势函数 $\times \dots \times / Z$ ”而不是“条件概率相乘”?

- 在 DGM 里按拓扑顺序做链式分解:  $p(\mathbf{y}) = \prod_i p(y_i \mid \text{pa}_i)$ 。
- UGM 没有方向与顺序, 做不了这件事。于是换个思路:
  - 给每个极大团 (图里两两相连、无法再扩大的全连接子图) 放一个**非负打分函数**  $\psi$ ;
  - 把所有团的打分相乘得到总体打分;
  - 再用一个**配分函数**  $Z = \sum_{\mathbf{y}}$  (对所有取值求和) 把它归一化成真正的概率。
- 只要分布是**正的** (任何取值概率不为 0), 且它的 CI 可以由某个 UGM 表示, 就一定能写成这种形式——这就是 Hammersley–Clifford 定理。

直觉:  $\psi$  像“相容性分数”, 分数越大越“合拍”, 乘起来代表所有局部关系的综合;  $Z$  把分数整体缩放到“总和=1”。

## 2) “能量”与“吉布斯分布”

- 把“势”写成  $\psi = \exp(-E)$ , 其中  $E$  是**能量**。
- 于是  $p \propto \exp(-\sum E)$ 。**能量低  $\Rightarrow$  概率高**。
- 这与物理里“系统更愿意处在低能态”一致, 因此叫**能量基模型**。

## 3) 成对 MRF (pairwise) 为何常见?

- 一般而言, 团可能很大 (很多变量一起相互作用), 势函数就复杂;
- **成对 MRF**只在**边上**放势 (两两相互作用), 形式简单、参数少, 工程里最常用;
- 代价: 表达能力比“任意团势”弱 (不是所有 UGM 都能只用边势准确表示)。

## 4) 如何表示“势”?

- **离散变量**: 可以直接用表 (非负数)。但**表不是概率**, 只是兼容度; 最后由  $Z$  统一归一化。
- **更通用: 把对数势设为特征线性**:
 
$$\log \psi = \phi^\top \theta \quad (\text{对数线性/最大熵模型})$$
  - 这让我们能把“是否出现某种模式”用特征  $\phi$  表示, 再由权重  $\theta$  学出偏好。
  - 例: 成对边上, 用  $I(y_s = j, y_t = k)$  组成  $K^2$  维特征, 就得到一个  $K \times K$  的表格势; 通过  $\exp$  可回到正的势 (式 19.14)。

## 5) 为什么要用特征而不是大表? (以英文三元组为例)

- 直接三元组表要  $26^3 = 17,576$  个参数, 数据难以支撑, 而且大多数组合从不出现。
- **特征化**: 只为“重要模式”(比如 “ing”“qu-”) 设特征  $\phi_k$ , 并在不同位置**共享**这些参数, 就能:
  - 大幅减少参数量;

- 允许任意长度的序列（因为特征在  $t$  上平移复用，见式 19.16）。
  - 特征可人工设计（利用先验知识），也可从数据里学出来（后文 19.5.6）。
- 

### 三、关键术语一眼懂

- **极大团** (maximal clique)：子图内任意两点都有边，且再加任何邻点都会破坏“全连通”。
  - **势/因子**  $\psi$ ：非负打分（不是概率）。
  - **配分函数**  $Z$ ：把所有可能赋值的打分求和得到的常数；用于把打分归一化成概率。
  - **正分布**：对任意赋值概率都  $> 0$ 。是 H-C 定理成立的关键前提。
  - **能量**  $E$ ：与  $\psi$  的关系是  $\psi = \exp(-E)$ 。
  - **对数线性/最大熵**： $\log \psi = \phi^\top \theta$ ，把结构放进特征里，把强弱放进权重里。
  - **成对 MRF**：只在边上放势  $\psi_{st}(y_s, y_t)$ 。
- 

### 四、要点总结

- UGM 无法用链式法则分解，因此用团势相乘 / 配分函数归一化来参数化联合分布（H-C 定理保证可行，前提：分布为正）。
- **势是非负“相容度”，不是概率**； $Z$  负责让总概率和为 1。
- **能量基视角**： $\psi = \exp(-E)$ ，**低能量=高概率**（吉布斯分布）。
- **成对 MRF**（只用边势）简单常用，但不如一般团势表达力强。
- **表示势的两种方式**：
  1. **表格**（离散情形直接列数）；
  2. **对数线性（最大熵）**： $\log \psi = \phi^\top \theta$ ，更灵活，可用**特征**表达高阶、稀疏或跨位置共享的规律。
- **三元组例子说明**：与其用庞大稀疏表，不如用**少量有意义的特征**（如 “ing”“qu-”）并**参数共享**，还能扩展到任意长度序列。

如果你愿意，我可以基于一个**二元标签成对 MRF**，手把手写出一个具体的势函数表和它对应的对数线性特征表示，帮助把“势 vs 概率”“ $\exp/Z$ ”这些概念再具体化。

这一节主要在讲什么？我不太懂

已思考 21s

不用慌，这一节（19.3：MRF 的参数化）就在回答一句话：

“在无向图模型里，怎样把联合概率  $p(\mathbf{y})$  写出来？”

因为无向图没有方向，不能像贝叶斯网那样按顺序做“链式相乘”。所以本节给了**三件核心工具**：

---

## 1) 用“势函数 $\times \dots \times +$ 归一化”来写概率

- 给图里每个**极大团**（彼此两两相连、再加就不全连了）放一个**非负打分**  $\psi_c(\mathbf{y}_c)$ （叫**势/因子**）。
- 把所有团的势相乘，再用一个**配分函数**  $Z$  做归一化：

$$p(\mathbf{y}) = \frac{1}{Z} \prod_c \psi_c(\mathbf{y}_c), \quad Z = \sum_{\mathbf{y}} \prod_c \psi_c(\mathbf{y}_c)$$

- **势不是概率**，只是“这个组合有多合拍”的**分数**； $Z$  负责让所有情况的概率加起来等于 1。

### Hammersley–Clifford 定理（本节主角）

只要分布**处处为正**（任何取值概率不为 0），并且它的条件独立性能用某个无向图表示，那么它就**一定**可以写成上面的“**团势相乘/配分归一**”的样子。

含义：UGM 的“断路=独立”的结构信息，和“团势相乘”的数值写法是匹配的。

---

## 2) “能量”视角=物理里的吉布斯分布

把势写成  $\psi_c = \exp(-E_c)$ （**能量**  $E$  越低，势越大）

$$p(\mathbf{y}) \propto \exp\left(-\sum_c E_c(\mathbf{y}_c)\right)$$

这就是**能量基模型**：**低能量  $\Rightarrow$  高概率**。它把物理里的直觉借来用在机器学习上。

---

## 3) 简化版：成对 MRF (pairwise)

- 不是对每个大团放势，而是**只在边上放势**： $\prod_{s \sim t} \psi_{st}(y_s, y_t)$ 。
- **优点**：最常用、实现简单、参数少。
- **代价**：不如“任意团势”通用（有些相互作用需要更高阶团才能表达）。

## 4) 势函数怎么表示?

两条路:

### A. 表格 (离散变量)

直接用非负表格给  $\psi$  赋值。记住它不是概率，只是相容度。

### B. 对数线性/最大熵 (更通用)

设

$$\log \psi_c(\mathbf{y}_c) = \phi_c(\mathbf{y}_c)^\top \theta_c$$

- $\phi$ : 由取值构造的特征 (比如“两个字母是否相同”“是否是 ing”)。
- $\theta$ : 每个特征的权重。
- 这样  $\log p(\mathbf{y}) = \sum_c \phi_c^\top \theta_c - \log Z$ 。
- **好处**: 能把“哪些模式重要”编码进特征里，参数可共享，可做高阶因子而不用巨大表格。

### 小例子 (把抽象变具体)

两个二值节点  $y_1, y_2$ , 只在边上有势。设“相同更合拍”:

- 定个势表:  
 $\psi(0, 0) = 3, \psi(0, 1) = 1, \psi(1, 0) = 1, \psi(1, 1) = 3$
- 配分:  $Z = 3 + 1 + 1 + 3 = 8$
- 概率:  $p(0, 0) = 3/8, p(0, 1) = 1/8, p(1, 0) = 1/8, p(1, 1) = 3/8$ 。
- 用对数线性也能得到同样势: 取特征  $\phi = I(y_1 = y_2)$ , 令  $\theta = \ln 3$ 。则  
 $\log \psi = \theta \cdot I(y_1 = y_2) \Rightarrow \psi = \exp(\theta)$  (相同=3) 或 1 (不同)。

### 语言模型的触发词特征 (为什么要特征而不是大表)

- 直接用三元组 (trigram) 表要  $26^3 = 17,576$  个参数，且多半从不出现。
- 换成特征: 只给“ing”“qu-”… 这类**关键模式**设二值特征  $\phi_k$ , 定义

$$\psi(y_{t-1}, y_t, y_{t+1}) = \exp\left(\sum_k \theta_k \phi_k(\cdot)\right)$$

在不同位置共享同一组  $\theta_k$ , 就能给任意长度单词定义概率，同时参数少、可解释。

---

## 一句话路线图 (做 MRF 的“食谱”)

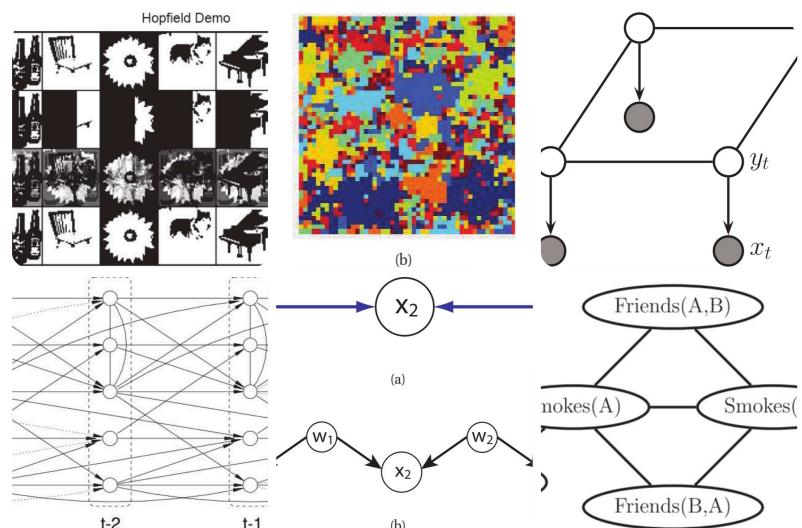
1. 画出 UGM;

2. 选哪些团放势 (常用：只放在边上)；
  3. 选表示法：表格或对数线性特征；
  4. 定义  $p(\mathbf{y}) \propto \prod \psi$ , 并用  $Z$  归一化；
  5. 训练时学习  $\theta$  (后面 19.5 讲)。
- 

## 要点小结

- **核心问题：** UGM 没有方向  $\Rightarrow$  不能链式分解，所以用**团势相乘 / 配分归一**写联合分布。
- **Hammersley–Clifford：** 在**正分布**前提下，这种写法与 UGM 的独立结构**等价**。
- **能量视角：**  $\psi = \exp(-E)$ , 低能量=高概率。
- **成对 MRF：** 只在边上放势，简单常用但不最通用。
- **势的表示：**
  - 表格 (直观)；
  - **对数线性/最大熵** (更灵活，可用特征、可参数共享、能做高阶因子)。
- **特征化的价值：** 少量有意义的特征 (如“ing”) > 巨大稀疏表；还能推广到任意长度序列。

如果你想，我可以用你熟悉的一个小场景 (比如图像二分类、序列标注) 帮你从图到势到  $p(\mathbf{y})$  走一遍完整流程。



翻译并解释以下内容，翻译不能有遗漏，解释请通俗易懂且详尽，因为我的数学基础不太好，最后总结要点。

Figure 19.7 Examples of how an associative memory can

reconstruct images. These are binary images of size  $50 \times 50$  pixels. Top: training images. Row 2: partially visible test images. Row 3: estimate after 5 iterations. Bottom: final state estimate.

Figure 19.8 Visualizing a sample from a 10-state Potts model of size  $128 \times 128$  for different association strengths: (a)  $J = 1.42$ , (b)  $J = 1.44$ , (c)  $J = 1.46$ . The regions are labeled according to size: blue is largest, red is smallest. Used with kind permission of Erik Sudderth.

Figure 19.9 A grid-structured MRF with local evidence nodes.

Figure 19.10 A VAR(2) process represented as a dynamic chain graph.

Figure 19.11 (a) A bi-directed graph. (b) The equivalent DAG. Here the w nodes are latent confounders.

Figure 19.12 An example of a ground Markov logic network represented as a pairwise MRF for 2 people.

## 19.4 Examples of MRFs

In this section, we show how several popular probability models can be conveniently expressed as UGMs.

### 19.4.1 Ising model

The Ising model is an example of an MRF that arose from statistical physics.<sup>3</sup> It was originally used for modeling the behavior of magnets. In particular, let  $y_s \in \{-1, +1\}$  represent the spin of an atom, which can either be spin down or up. In some magnets, called ferro-magnets, neighboring spins tend to line up in the same direction, whereas in other kinds of magnets, called anti-ferromagnets, the spins “want” to be different from their neighbors.

We can model this as an MRF as follows. We create a graph in the form of a 2D or 3D lattice, and connect neighboring variables, as in Figure 19.1(b). We then define the following pairwise clique potential:

$$\psi_{st}(y_s, y_t) = e^{-w_{st}} e^{-w_{st}} e^{w_{st}} (19.17)$$

Here  $w_{st}$  is the coupling strength between nodes s and t. If two nodes are not connected in the graph, we set  $w_{st} = 0$ . We assume that the weight matrix  $W$  is symmetric, so  $w_{st} = w_{ts}$ . Often we assume all edges have the same strength, so  $w_{st} = J$  (assuming  $w_{st} = 0$ ).

If all the weights are positive,  $J > 0$ , then neighboring spins are likely to be in the same state; this can be used to

model ferromagnets, and is an example of an associative Markov network. If the weights are sufficiently strong, the corresponding probability distribution will have two modes, corresponding to the all +1's state and the all -1's state. These are called the ground states of the system.

If all of the weights are negative,  $J < 0$ , then the spins want to be different from their neighbors; this can be used to model an anti-ferromagnet, and results in a frustrated system, in which not all the constraints can be satisfied at the same time. The corresponding probability distribution will have multiple modes. Interestingly, computing the partition function  $Z(J)$  can be done in polynomial time for associative Markov networks, but is NP-hard in general (Cipra 2000).

There is an interesting analogy between Ising models and Gaussian graphical models. First, assuming  $y_t \in \{-1, +1\}$ , we can write the unnormalized log probability of an Ising model as follows:

$$\log \tilde{p}(y) = - \sum_{t=1}^T \sum_{s \neq t} w_{st} y_s y_t = -\frac{1}{2} \sum_{t=1}^T \sum_{s \neq t} y_t W y_s (19.18)$$

(The factor of  $\frac{1}{2}$  arises because we sum each edge twice.)

If  $w_{st} = J > 0$ , we get a low energy (and hence high probability) if neighboring states agree.

Sometimes there is an external field, which is an energy term which is added to each spin. This can be modelled using a local energy term of the form  $-bTy$ , where  $b$  is sometimes called a bias term. The modified distribution is given by

$$\log \tilde{p}(y) = \sum_{t=1}^T \sum_{s \neq t} w_{st} y_s y_t + \sum_{t=1}^T b_t y_t = \frac{1}{2} \sum_{t=1}^T \sum_{s \neq t} y_t W y_s + \sum_{t=1}^T b_t y_t (19.19)$$

where  $\theta = (W, b)$ .

If we define  $\mu = -\frac{1}{2} \sum_{t=1}^T b_t$ ,  $\Sigma = -W$ , and  $c = \frac{1}{2} \sum_{t=1}^T b_t^2$ , we can rewrite this in a form that looks similar to a Gaussian:

$$\tilde{p}(y) \propto \exp(-\frac{1}{2} y^\top \Sigma^{-1} y - \mu^\top y + c) (19.20)$$

One very important difference is that, in the case of Gaussians, the normalization constant,  $Z = |2\pi\Sigma|$ , requires the computation of a matrix determinant, which can be computed in  $O(D^3)$  time, whereas in the case of the Ising model, the normalization constant requires summing over all  $2^D$  bit vectors; this is equivalent to computing the matrix permanent, which is NP-hard in general (Jerrum et al. 2004).

#### 19.4.2 Hopfield networks

A Hopfield network (Hopfield 1982) is a fully connected

Ising model with a symmetric weight matrix,  $W = WT$ .

These weights, plus the bias terms  $b$ , can be learned from training data using (approximate) maximum likelihood, as described in Section 19.5.

The main application of Hopfield networks is as an associative memory or content addressable memory. The idea is this: suppose we train on a set of fully observed bit vectors, corresponding to patterns we want to memorize. Then, at test time, we present a partial pattern to the network. We would like to estimate the missing variables; this is called pattern completion. See Figure 19.7 for an example. This can be thought of as retrieving an example from memory based on a piece of the example itself, hence the term "associative memory".

Since exact inference is intractable in this model, it is standard to use a coordinate descent algorithm known as iterative conditional modes (ICM), which just sets each node to its most likely (lowest energy) state, given all its neighbors. The full conditional can be shown to be

$$p(y_s = 1 | y_{\neg s}, \theta) = \text{sigm}(w^T s, y_{\neg s} + b_s) \quad (19.21)$$

Picking the most probable state amounts to using the rule  $y_s = 1$  if  $w^T s + b_s > 0$  and using  $y_s = 0$  otherwise. (Much better inference algorithms will be discussed later in this book.)

Since inference is deterministic, it is also possible to interpret this model as a recurrent neural network. (This is quite different from the feedforward neural nets studied in Section 16.5; they are univariate conditional density models of the form  $p(y|x, \theta)$  which can only be used for supervised learning.) See Hertz et al. (1991) for further details on Hopfield networks.

A Boltzmann machine generalizes the Hopfield / Ising model by including some hidden nodes, which makes the model representationally more powerful. Inference in such models often uses Gibbs sampling, which is a stochastic version of ICM (see Section 24.2 for details).

However, we could equally well apply Gibbs to a Hopfield net and ICM to a Boltzmann machine: the inference algorithm is not part of the model definition. See Section 27.7 for further details on Boltzmann machines.

#### 19.4.3 Potts model

It is easy to generalize the Ising model to multiple discrete states,  $y_t \in \{1, 2, \dots, K\}$ . It is common to use a potential

function of the following form:

$$\psi_{st}(y_s, y_t) = \begin{cases} 0 & J > 0 \\ 1 & J \leq 0 \end{cases} (19.22)$$

This is called the Potts model.<sup>5</sup> If  $J > 0$ , then neighboring nodes are encouraged to have the same label. Some samples from this model are shown in Figure 19.8. We see that for  $J > 1.44$ , large clusters occur, for  $J < 1.44$ , many small clusters occur, and at the critical value of  $K = 1.44$ , there is a mix of small and large clusters. This rapid change in behavior as we vary a parameter of the system is called a phase transition, and has been widely studied in the physics community. An analogous phenomenon occurs in the Ising model; see (MacKay 2003, ch 31) for details.

The Potts model can be used as a prior for image segmentation, since it says that neighboring pixels are likely to have the same discrete label and hence belong to the same segment. We can combine this prior with a likelihood term as follows:

$$p(y, x|\theta) = p(y|J) \cdot p(x|y_t, \theta) = \frac{1}{Z(J)} \sum_{s,t} \psi_{st}(y_s, y_t; J) \cdot p(x_t|y_t, \theta) \quad (19.23)$$

where  $p(x_t|y_t = k, \theta)$  is the probability of observing pixel  $x_t$  given that the corresponding segment belongs to class  $k$ . This observation model can be modeled using a Gaussian or a non-parametric density. (Note that we label the hidden nodes  $y_t$  and the observed nodes  $x_t$ , to be compatible with Section 19.6.)

The corresponding graphical model is a mix of undirected and directed edges, as shown in Figure 19.9. The undirected 2d lattice represents the prior  $p(y)$ ; in addition, there are directed edge from each  $y_t$  to its corresponding  $x_t$ , representing the local evidence. Technically speaking, this combination of an undirected and directed graph is called a chain graph. However, since the  $x_t$  nodes are observed, they can be “absorbed” into the model, thus leaving behind an undirected “backbone”.

This model is a 2d analog of an HMM, and could be called a partially observed MRF. As in an HMM, the goal is to perform posterior inference, i.e., to compute (some function of)  $p(y|x, \theta)$ . Unfortunately, the 2d case is provably much harder than the 1d case, and we must resort to approximate methods, as we discuss in later chapters.

Although the Potts prior is adequate for regularizing

supervised learning problems, it is not sufficiently accurate to perform image segmentation in an unsupervised way, since the segments produced by this model do not accurately represent the kinds of segments one sees in natural images (Morris et al. 1996). For the unsupervised case, one needs to use more sophisticated priors, such as the truncated Gaussian process prior of (Sudderth and Jordan 2008).

#### 19.4.4 Gaussian MRFs

An undirected GGM, also called a Gaussian MRF (see e.g., (Rue and Held 2005)), is a pairwise MRF of the following form:

$$p(y|\theta) \propto s \sim t \psi_{st}(ys, yt) t \psi_t(yt) \quad (19.24)$$

$$\psi_{st}(ys, yt) = \exp(-\frac{1}{2} ys \Lambda styt) \quad (19.25)$$

$$\psi_t(yt) = \exp(-\frac{1}{2} \Lambda t y t^2 + \eta t y t) \quad (19.26)$$

(Note that we could easily absorb the node potentials  $\psi_t$  into the edge potentials, but we have kept them separate for clarity.) The joint distribution can be written as follows:

$$p(y|\theta) \propto \exp[\eta T y - \frac{1}{2} y^T \Lambda y] \quad (19.27)$$

We recognize this as a multivariate Gaussian written in information form where  $\Lambda = \Sigma^{-1}$  and  $\eta = \Lambda \mu$ .

If  $\Lambda_{st} = 0$ , then there is no pairwise term connecting  $s$  and  $t$ , so by the factorization theorem (Theorem 2.2.1), we conclude that

$$ys \perp yt | y - (st) \iff \Lambda_{st} = 0 \quad (19.28)$$

The zero entries in  $\Lambda$  are called structural zeros, since they represent the absent edges in the graph. Thus undirected GGMs correspond to sparse precision matrices, a fact which we exploit in Section 26.7.2 to efficiently learn the structure of the graph.

##### 19.4.4.1 Comparing Gaussian DGMs and UGMs

In Section 10.2.5, we saw that directed GGMs correspond to sparse regression matrices, and hence sparse Cholesky factorizations of covariance matrices, whereas undirected GGMs correspond to sparse precision matrices. The advantage of the DAG formulation is that we can make the regression weights  $W$ , and hence  $\Sigma$ , be conditional on covariate information (Pourahmadi 2004), without worrying about positive definite constraints. The disadvantage of the DAG formulation is its dependence on the order, although in certain domains, such as time series, there is already a natural ordering of the variables. It is actually possible to combine both representations,

resulting in a Gaussian chain graph. For example, consider a discrete-time, second-order Markov chain in which the states are continuous,  $y_t \in RD$ . The transition function can be represented as a (vector-valued) linear Gaussian CPD:

$$p(y_t|y_{t-1}, y_{t-2}, \theta) = N(y_t | A_1 y_{t-1} + A_2 y_{t-2}, \Sigma) \quad (19.29)$$

This is called vector auto-regressive or VAR process of order 2. Such models are widely used in econometrics for time-series forecasting.

The time series aspect is most naturally modeled using a DGM. However, if  $\Sigma - 1$  is sparse, then the correlation amongst the components within a time slice is most naturally modeled using a UGM. For example, suppose we have

$$A_1 = ( \begin{array}{ccccccccc} 3 & 5 & 0 & 1 & 5 & 0 & 0 & 0 & 3 & 5 & 0 \\ -1 & 5 & 0 & 2 & 5 & 1 & 3 & 3 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} ), A_2 = ( \begin{array}{ccccccccc} 0 & 0 & -1 & 5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 5 & 0 & 1 & 3 & 0 & 0 & 0 \\ -1 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} ) \quad (19.30)$$

and

$$\Sigma = ( \begin{array}{ccccccccc} 1 & 1 & 2 & 1 & 3 & 0 & 0 & 1 & 2 & 1 \\ -1 & 3 & 0 & 0 & 1 & 3 & -1 & 3 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} ), \Sigma - 1 = ( \begin{array}{ccccccccc} 2.13 & -1.47 & -1.200 & -1.47 \\ 2.13 & 1.20 & 0 & -1.21 \\ 2.13 & 1.20 & 0 & -1.21 \\ 2.13 & 1.20 & 0 & -1.21 \\ 2.13 & 1.20 & 0 & -1.21 \\ 2.13 & 1.20 & 0 & -1.21 \\ 2.13 & 1.20 & 0 & -1.21 \\ 2.13 & 1.20 & 0 & -1.21 \\ 2.13 & 1.20 & 0 & -1.21 \\ 2.13 & 1.20 & 0 & -1.21 \\ 2.13 & 1.20 & 0 & -1.21 \\ 2.13 & 1.20 & 0 & -1.21 \end{array} ) \quad (19.31)$$

The resulting graphical model is illustrated in Figure 19.10. Zeros in the transition matrices  $A_1$  and  $A_2$  correspond to absent directed arcs from  $y_{t-1}$  and  $y_{t-2}$  into  $y_t$ . Zeros in the precision matrix  $\Sigma - 1$  correspond to absent undirected arcs between nodes in  $y_t$ .

Sometimes we have a sparse covariance matrix rather than a sparse precision matrix. This can be represented using a bi-directed graph, where each edge has arrows in both directions, as in Figure 19.11(a). Here nodes that are not connected are unconditionally independent. For example in Figure 19.11(a) we see that  $Y_1 \perp Y_3$ . In the Gaussian case, this means  $\Sigma_{1,3} = \Sigma_{3,1} = 0$ . (A graph representing a sparse covariance matrix is called a covariance graph.) By contrast, if this were an undirected model, we would have that  $Y_1 \perp Y_3 | Y_2$ , and  $\Lambda_{1,3} = \Lambda_{3,1} = 0$ , where  $\Lambda = \Sigma - 1$ .

A bidirected graph can be converted to a DAG with latent variables, where each bidirected edge is replaced with a hidden variable representing a hidden common cause, or confounder, as illustrated in Figure 19.11(b). The relevant CI properties can then be determined using dseparation. We can combine bidirected and directed edges to get a directed mixed graphical model. This is useful for

representing a variety of models, such as ARMA models (Section 18.2.4.4), structural equation models (Section 26.5.5), etc.

#### 19.4.5 Markov logic networks

In Section 10.2.2, we saw how we could “unroll” Markov models and HMMs for an arbitrary number of time steps in order to model variable-length sequences. Similarly, in Section 19.4.1, we saw how we could expand a lattice UGM to model images of any size. What about more complex domains, where we have a variable number of objects and relationships between them? Creating models for such scenarios is often done using first-order logic (see e.g., (Russell and Norvig 2010)). For example, consider the sentences “Smoking causes cancer” and “If two people are friends, and one smokes, then so does the other”. We can write these sentences in first-order logic as follows:

$$\forall x. Sm(x) \Rightarrow Ca(x) \quad (19.32)$$

$$\forall x. \forall y. Fr(x, y) \wedge Sm(x) \Rightarrow Sm(y) \quad (19.33)$$

where  $Sm$  and  $Ca$  are predicates, and  $Fr$  is a relation.

Of course, such rules are not always true. Indeed, this brittleness is the main reason why logical approaches to AI are no longer widely used, at least not in their pure form. There have been a variety of attempts to combine first order logic with probability theory, an area known as statistical relational AI or probabilistic relational modeling (Kersting et al. 2011). One simple approach is to take logical rules and attach weights (known as certainty factors) to them, and then to interpret them as conditional probability distributions. For example, we might say  $p(Ca(x)=1|Sm(x)=1)=0.9$ . Unfortunately, the rule does not say what to predict if  $Sm(x)=0$ . Furthermore, combining CPDs in this way is not guaranteed to define a consistent joint distribution, because the resulting graph may not be a DAG.

An alternative approach is to treat these rules as a way of defining potential functions in an unrolled UGM. The result is known as a Markov logic network (Domingos and Lowd 2009). To specify the network, we first rewrite all the rules in conjunctive normal form (CNF), also known as clausal form. In this case, we get

$$\neg Sm(x) \vee Ca(x) \quad (19.34)$$

$$\neg Fr(x, y) \vee \neg Sm(x) \vee Sm(y) \quad (19.35)$$

The first clause can be read as "Either  $x$  does not smoke or he has cancer", which is logically equivalent to Equation 19.32. (Note that in a clause, any unbound variable, such as  $x$ , is assumed to be universally quantified.)

Inference in first-order logic is only semi-decidable, so it is common to use a restricted subset. A common approach (as used in Prolog) is to restrict the language to Horn clauses, which are clauses that contain at most one positive literal. Essentially this means the model is a series of if-then rules, where the right hand side of the rules (the "then" part, or consequence) has only a single term.

Once we have encoded our knowledge base as a set of clauses, we can attach weights to each one; these weights are the parameter of the model, and they define the clique potentials as follows:

$$\psi_c(xc) = \exp(w_c \varphi_c(xc)) \quad (19.36)$$

where  $\varphi_c(xc)$  is a logical expression which evaluates clause  $c$  applied to the variables  $xc$ , and  $w_c$  is the weight we attach to this clause. Roughly speaking, the weight of a clause specifies the probability of a world in which this clause is satisfied relative to a world in which it is not satisfied.

Now suppose there are two objects (people) in the world, Anna and Bob, which we will denote by constant symbols A and B. We can make a ground network from the above clauses by creating binary random variables  $S_x$ ,  $C_x$ , and  $F_{x,y}$  for  $x, y \in \{A, B\}$ , and then "wiring these up" according to the clauses above. The result is the UGM in Figure 19.12 with 8 binary nodes. Note that we have not encoded the fact that  $Fr$  is a symmetric relation, so  $Fr(A, B)$  and  $Fr(B, A)$  might have different values. Similarly, we have the "degenerate" nodes  $Fr(A, A)$  and  $Fr(B, B)$ , since we did not enforce  $x = y$  in Equation 19.33. (If we add such constraints, then the model compiler, which generates the ground network, could avoid creating redundant nodes.) In summary, we can think of MLNs as a convenient way of specifying a UGM template, that can get unrolled to handle data of arbitrary size. There are several other ways to define relational probabilistic models; see e.g., (Koller and Friedman 2009; Kersting et al. 2011) for details. In some cases, there is uncertainty about the number or existence of objects or relations (the so-called open universe problem). Section 18.6.2 gives a concrete

example in the context of multi-object tracking. See e.g., (Russell and Norvig 2010; Kersting et al. 2011) and references therein for further details.

已思考 41s

下面先把你给的内容完整翻译（包含所有公式与编号），再用通俗语言解释并串讲配图，最后给出要点小结。

---

## 一、逐条翻译（不遗漏）

**图 19.7** 关联记忆 (associative memory) 如何重建图像的一些例子。这些是  $50 \times 50$  像素的二值图像。顶行：训练图像。第2行：部分可见的测试图像。第3行：迭代 5 次后的中间结果。底行：最终状态估计 (恢复后的图像)。

**图 19.8** 从一个  $128 \times 128$  的 10 状态 Potts 模型中采样并可视化，不同的“相互吸引强度”下的样子：(a)  $J = 1.42$ , (b)  $J = 1.44$ , (c)  $J = 1.46$ 。区域按大小着色：蓝色最大，红色最小。经 Erik Sudderth 善意许可使用。

**图 19.9** 一个带局部证据结点的网格结构 MRF。

**图 19.10** 一个将 VAR(2) 过程表示为动态链图 (dynamic chain graph)。

**图 19.11** (a) 一个双向 (bi-directed) 图。 (b) 与之等价的 DAG。其中  $w$  结点是潜在混杂因素 (latent confounders)。

**图 19.12** 一个“落地”的 (ground) 马尔可夫逻辑网络的例子：对 2 个人，用成对 MRF 表示。

---

## 19.4 MRF 的一些例子

在本节里，我们展示若干常见概率模型如何方便地用 UGM 来表示。

### 19.4.1 Ising 模型

Ising 模型是来自统计物理的一个 MRF 示例。它最初用于建模磁体的行为。设  $y_s \in \{-1, +1\}$  表示原子的自旋 (向下或向上)。在某些磁体 (称为**铁磁体**) 中，相邻自旋倾向于**同向**；而在另外一些磁体 (称为**反铁磁体**) 中，自旋“希望”与其邻居**不同**。

我们可以把它建成 MRF：构造一个二维或三维格子图，并像图 19.1(b) 那样连接相邻变量。然后定义如下**成对团势**：

$$\psi_{st}(y_s, y_t) = \begin{bmatrix} e^{w_{st}} & e^{-w_{st}} \\ e^{-w_{st}} & e^{w_{st}} \end{bmatrix} \quad (19.17)$$

这里  $w_{st}$  是结点  $s$  与  $t$  之间的耦合强度。若图中两点不相连，令  $w_{st} = 0$ 。权重矩阵  $W$  设为对称，即  $w_{st} = w_{ts}$ 。常见做法是假设所有边强度相同： $w_{st} = J$ （若不相连则为 0）。

- 若所有权重都正 ( $J > 0$ )，邻居更可能同状态；可用来刻画铁磁体，也是**关联型马尔可夫网络**的例子。若强度足够大，分布会有两个“模态”：全 +1 和全 -1，称为系统的**基态**。
- 若所有权重都负 ( $J < 0$ )，自旋希望与邻居不同；这可刻画反铁磁体，得到一个**挫折系统**（不可能让所有约束同时满足），对应分布会有多个模态。很有意思的是：对关联型马尔可夫网络，配分函数  $Z(J)$  的计算可在多项式时间完成，但**一般情形**是 NP-困难的 (Cipra 2000)。

Ising 与高斯图模型有一个有趣的类比。令  $y_t \in \{-1, +1\}$ ，其未归一化对数概率为

$$\log \tilde{p}(y) = -\sum_{s \sim t} y_s w_{st} y_t = -\frac{1}{2} y^\top W y \quad (19.18)$$

( $\frac{1}{2}$  是因为每条边被算了两次。) 若  $w_{st} = J > 0$ ，相邻状态一致时能量低（概率高）。

有时还会有**外场**，即对每个自旋添加一个能量项。这可用一个**局部能量**  $-b^\top y$  来建模，其中  $b$  常称为**偏置**。修正后的分布为

$$\log \tilde{p}(y) = \sum_{s \sim t} w_{st} y_s y_t + \sum_s b_s y_s = \frac{1}{2} y^\top W y + b^\top y \quad (19.19)$$

其中  $\theta = (W, b)$ 。

若定义  $\mu = -\frac{1}{2}\Sigma^{-1}b$ ,  $\Sigma^{-1} = -W$ ,  $c = \frac{1}{2}\mu^\top \Sigma^{-1}\mu$ , 可改写成类高斯的形式：

$$\tilde{p}(y) \propto \exp(-\frac{1}{2}(y - \mu)^\top \Sigma^{-1}(y - \mu) + c) \quad (19.20)$$

重要差别在于：高斯情形的归一化常数  $Z = |2\pi\Sigma|$  只需算行列式 ( $O(D^3)$ )；而 Ising 的  $Z$  需对所有  $2^D$  个二进制向量求和——等价于计算**矩阵永久行列式**，一般是 NP-困难 (Jerrum 等, 2004)。

## 19.4.2 Hopfield 网络

Hopfield 网络 (Hopfield 1982) 是一个**完全连通**、权重对称  $W = W^\top$  的 Ising 模型。其权重  $W$  与偏置  $b$  可按第 19.5 节所述，用（近似的）极大似然从训练数据学习。

主要应用是**关联记忆 / 内容可寻址存储**：

训练阶段，用想“记住”的**完整**比特向量做训练；测试时给网络一个**缺失**的部分模式；目标

是**补全缺失变量** (pattern completion)。例子见图 19.7：严重遮挡的输入，经过几次迭代后逐步恢复。

由于精确推断不可行，通常用一个**坐标下降算法——迭代条件模式 (ICM)**：给定邻居，把每个结点都设为**其条件下最可能 (最低能量)** 的状态。可得单点条件分布：

$$p(y_s = 1 \mid y_{-s}, \theta) = \text{sigm}(w_{s,:}^\top y_{-s} + b_s) \quad (19.21)$$

选择最可能状态等价于：若  $\sum_t w_{st} y_t > b_s$  就令  $y_s^{*} = 1$ ，否则  $y_s^{*} = 0$ 。（更好的推断算法会在后文介绍。）

因为这个推断是**确定性的**，该模型也可被视为一种**循环神经网络**（不同于第 16.5 节的前馈神经网，那类是  $p(y \mid x, \theta)$  的监督模型）。更多细节见 Hertz 等 (1991)。

**玻尔兹曼机**在 Hopfield/Ising 的基础上引入**隐变量**，使表达能力更强。这类模型常用**Gibbs 采样** (ICM 的随机版，见 24.2 节) 做推断。当然，也可以对 Hopfield 用 Gibbs、对 Boltzmann 用 ICM——**推断算法不是模型定义的一部分** (详见 27.7 节)。

### 19.4.3 Potts 模型

把 Ising 模型推广到**多离散状态**  $y_t \in \{1, 2, \dots, K\}$  很容易。常用的成对势函数为

$$\psi_{st}(y_s, y_t) = \begin{bmatrix} e^J & 0 & 0 \\ 0 & e^J & 0 \\ 0 & 0 & e^J \end{bmatrix} \quad (19.22)$$

这称为**Potts 模型**。若  $J > 0$ ，相邻结点被鼓励取**相同标签**。图 19.8 展示了该模型的若干样本：当  $J > 1.44$  时出现**大团块**；当  $J < 1.44$  时多为**小团块**；在临界值  $J = 1.44$  附近，小与大的团块**混合**。当系统参数变化导致行为**突变**时，称为**相变**。Ising 模型也有类似现象 (详见 MacKay 2003, 第 31 章)。

Potts 模型可作为**图像分割**的先验：它表达“邻近像素标签相同”的偏好。可与似然项结合：

$$p(y, x \mid \theta) = p(y \mid J) \prod_t p(x_t \mid y_t, \theta) = \frac{1}{Z(J)} \prod_{s \sim t} \psi(y_s, y_t; J) \prod_t p(x_t \mid y_t, \theta) \quad (19.23)$$

其中  $p(x_t \mid y_t = k, \theta)$  是在像素属于第  $k$  类时观测到  $x_t$  的概率，可用高斯或非参数密度建模。（为和 19.6 节一致，这里把**隐藏结点**记作  $y_t$ ，**观测结点**记作  $x_t$ 。）

相应的图模型**混合了无向与有向边** (见图 19.9)：无向 2D 格子给出先验  $p(y)$ ；每个  $y_t \rightarrow x_t$  的有向边表示**局部证据**。严格说，这种“有向+无向”的组合称为**链图**。但由于  $x_t$  已观测，可“吸收”进模型，仅留下无向的“主干”。

这个模型是 HMM 的“二维类比”，也可称为**部分观测的 MRF**。目标同 HMM：做后验推断，即计算  $p(y | x, \theta)$ （或其函数）。不幸的是，二维问题比一维**难得多**，通常需要近似方法（见后续章节）。

尽管 Potts 先验足以在**监督学习**里起到“正则化”作用，但若要**无监督地**做分割，它并不够准确（Morris 等，1996）：产生的分割与自然图像中的真实分割差异较大。无监督情形需要更复杂的先验，例如（Sudderth & Jordan 2008）的**截断高斯过程先验**。

---

#### 19.4.4 高斯 MRF (Gaussian MRFs)

无向 GGM（也称**高斯 MRF**；见 Rue & Held 2005）是如下的成对 MRF：

$$p(y | \theta) \propto \prod_{s \sim t} \psi_{st}(y_s, y_t) \prod_t \psi_t(y_t) \quad (19.24)$$

$$\psi_{st}(y_s, y_t) = \exp(-\frac{1}{2} y_s^\top \Lambda_{st} y_t) \quad (19.25)$$

$$\psi_t(y_t) = \exp(-\frac{1}{2} \Lambda_{tt} y_t^\top y_t + \eta_t y_t) \quad (19.26)$$

（当然也可把结点势  $\psi_t$  吸收到边势里，为清楚起见这里分开写。）于是

$$p(y | \theta) \propto \exp[\eta^\top y - \frac{1}{2} y^\top \Lambda y] \quad (19.27)$$

这正是**信息形式** (precision form) 的多元高斯，其中  $\Lambda = \Sigma^{-1}$ ,  $\eta = \Lambda \mu$ 。

若  $\Lambda_{st} = 0$ ，则  $s, t$  之间没有成对项。由**因子分解定理**（定理 2.2.1）可得

$$y_s \perp y_t | y_{-(st)} \Leftrightarrow \Lambda_{st} = 0 \quad (19.28)$$

$\Lambda$  中的 0 称为**结构性零** (structural zeros)，表示图中缺边。因此**无向 GGM  $\Leftrightarrow$  稀疏精度矩阵**。第 26.7.2 节将利用这一点高效学习结构。

##### 19.4.4.1 高斯 DGM 与 UGM 的比较

第 10.2.5 节指出：**有向** GGM 对应**稀疏回归矩阵**（因此是协方差矩阵的稀疏 Cholesky 分解），而**无向** GGM 对应**稀疏精度矩阵**。

- **DAG 的优点**：可让回归权重  $W$ ，从而  $\Sigma$ ，**随协变量改变**（Pourahmadi 2004），且不用担心正定约束。
- **DAG 的缺点**：依赖**变量顺序**；不过在某些领域（如时间序列）顺序是自然存在的。

两种表示法可以**结合成高斯链图**。例：离散时间、二阶马尔可夫链，状态为连续向量  $y_t \in \mathbb{R}^D$ 。转移函数是一个（向量值）**线性-高斯 CPD**：

$$p(y_t | y_{t-1}, y_{t-2}, \theta) = \mathcal{N}(y_t | A_1 y_{t-1} + A_2 y_{t-2}, \Sigma) \quad (19.29)$$

称为二阶**向量自回归** (VAR) 过程。在计量经济学里的时间序列预测广泛使用。

时间依赖更适合用 DGM 表示；若  $\Sigma^{-1}$  稀疏，同一时刻各分量的相关性用 UGM 表示最自然。比如

$$A_1 = \begin{pmatrix} \frac{3}{5} & 0 & \frac{1}{5} & 0 & 0 \\ 0 & \frac{3}{5} & 0 & -\frac{1}{5} & 0 \\ \frac{2}{5} & \frac{1}{3} & \frac{3}{5} & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{2} & \frac{1}{5} \\ 0 & \frac{1}{5} & 0 & \frac{2}{5} & 0 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 0 & 0 & -\frac{1}{5} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \frac{1}{5} & 0 & \frac{1}{3} & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{5} & 0 \end{pmatrix} \quad (19.30)$$

以及

$$\Sigma = \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} & 0 & 0 \\ \frac{1}{2} & 1 & -\frac{1}{3} & 0 & 0 \\ \frac{1}{3} & -\frac{1}{3} & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \quad \Sigma^{-1} = \begin{pmatrix} 2.13 & -1.47 & -1.20 & 0 & 0 \\ -1.47 & 2.13 & 1.20 & 0 & 0 \\ -1.20 & 1.20 & 1.80 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad (19.31)$$

得到的图见图 19.10。矩阵  $A_1, A_2$  中的 0 表示从  $y_{t-1}, y_{t-2}$  指向  $y_t$  的有向边缺失； $\Sigma^{-1}$  中的 0 表示同一时刻结点之间的无向边缺失。

有时我们拥有的是**稀疏协方差**（而不是稀疏精度）。可用**双向图**表示——每条边两端都带箭头（图 19.11(a)）。此时**不相连就表示无条件独立**；例如图 19.11(a) 中  $Y_1 \perp Y_3$ ，在高斯下即  $\Sigma_{1,3} = \Sigma_{3,1} = 0$ 。而若是无向模型，则是**条件独立**  $Y_1 \perp Y_3 \mid Y_2$ ，对应  $\Lambda_{1,3} = \Lambda_{3,1} = 0$  ( $\Lambda = \Sigma^{-1}$ )。

一个**双向图**可转成**带隐变量的 DAG**：把每条双向边替换成一个“共同原因/混杂因素”的隐结点（图 19.11(b)）。之后可用 d-分离判定 CI。

把双向与有向边结合得到**有向混合图模型**，可表示多种模型，如 ARMA (18.2.4.4)、结构方程模型 (26.5.5) 等。

#### 19.4.5 马尔可夫逻辑网络 (MLN)

第 10.2.2 节里，我们把马尔可夫模型与 HMM“展开”任意步数以建模可变长度序列；19.4.1 节中，把格子 UGM“扩展”到任意大小图像。那**对象数量与关系都是可变**的复杂领域呢？常用**一阶逻辑**来建模（如 Russell & Norvig 2010）。例如两句话：“吸烟导致癌症”，“如果两个人是朋友，其中一人抽烟，那么另一人也抽”。它们可写成一阶逻辑：

$$\forall x. \text{Sm}(x) \Rightarrow \text{Ca}(x) \quad (19.32)$$

$$\forall x. \forall y. \text{Fr}(x, y) \wedge \text{Sm}(x) \Rightarrow \text{Sm}(y) \quad (19.33)$$

其中 Sm, Ca 是谓词， Fr 是关系。

当然，这些规则并非总真——这正是一阶逻辑在纯形式上难于直接用于 AI 的原因。把一阶逻辑与概率结合（统计关系 AI/概率关系建模）的方法很多（Kersting 等, 2011）。一种简单法是给逻辑规则加权重（置信因子），并把它们解释为条件概率，如  $p(\text{Ca}(x) = 1 | \text{Sm}(x) = 1) = 0.9$ 。但该规则并没说明  $\text{Sm}(x) = 0$  时预测什么；而且这样拼接 CPD 不保证得到一致的联合分布（图可能不是 DAG）。

另一种办法：把这些规则当作在“展开”的 UGM 中定义势函数的方式——结果就是马尔可夫逻辑网络（MLN）（Domingos & Lowd 2009）。指定网络时，先把所有规则改写成合取范式（CNF，又称子句形式）。此时得到

$$\neg \text{Sm}(x) \vee \text{Ca}(x) \quad (19.34)$$

$$\neg \text{Fr}(x, y) \vee \neg \text{Sm}(x) \vee \text{Sm}(y) \quad (19.35)$$

第一条可读作“要么  $x$  不抽烟，要么他得癌症”，在逻辑上等价于式 (19.32)。（注意子句里未绑定的变量，如  $x$ ，默认全称量化。）

由于一阶逻辑推理只是半可判定，常用受限子集。常见的是限制为 Horn 子句（Prolog 用法）：每个子句最多一个正文字。本质上模型就是一系列 if-then 规则，右边（then，结论）只有一个项。

把知识库编码为子句后，给每个子句加权重；这些权重就是模型参数，它们定义团势：

$$\psi_c(x_c) = \exp(w_c \phi_c(x_c)) \quad (19.36)$$

其中  $\phi_c(x_c)$  是“对子句  $c$  应用到变量  $x_c$  的逻辑表达式”的取值（真/假）， $w_c$  是该子句的权重。粗略地说：子句权重决定“满足该子句的世界”相对于“不满足的世界”的概率大小。

假设世界中有两个人 Anna 与 Bob，记为常量  $A, B$ 。由上述子句生成落地网络：创建二值随机变量  $S_x, C_x, F_{x,y}$  ( $x, y \in \{A, B\}$ )，并按子句连接它们。结果就是图 19.12 中含 8 个二值结点的 UGM。注意我们没有编码“Fr”是对称关系，所以  $\text{Fr}(A, B)$  与  $\text{Fr}(B, A)$  的值可能不同；同样也出现“退化”结点  $\text{Fr}(A, A), \text{Fr}(B, B)$ ，因为在式 (19.33) 中我们没有强制  $x = y$ 。（若加入这些约束，生成落地网的“模型编译器”即可避免冗余结点。）

总之，可把 MLN 看作指定 UGM 模板的一种便捷方式，它能“展开”到任意规模的数据。关于关系概率模型的其它方法，见 Koller & Friedman 2009; Kersting 等 2011 等。在某些场景中，对象或关系的数量/存在性本身也不确定（开放宇宙问题）；18.6.2 节在多目标跟踪中给出具体例子，更多可参考上述文献与其中引用。

## 二、通俗解释与配图导读

- **图 19.7 (Hopfield)**：把几张黑白图“记到”网络里。测试时遮挡 60% 再喂给网络。每次迭代按“邻居投票+阈值”更新像素，5 次后已能看出轮廓，最后恢复到训练图那一类。**关键词：关联记忆、模式补全、ICM（逐点取最可能值）。**
- **Ising 直觉**：像素/原子只有  $-1/+1$  两种状态；邻居“想一样” ( $J > 0$ ) 或“想不一样” ( $J < 0$ )。能量低  $\Leftrightarrow$  概率高。
- **图 19.8 (Potts 相变)**：当“同标偏好”  $J$  从 1.42 增到 1.46，图像从“碎碎的彩块”突然变成“大片同色区域”——这就是**相变**。Potts = 多类别版的 Ising，常做**分割先验**。
- **图 19.9 (链图)**：无向格子表示先验  $p(y)$ ，“像素证据”用有向边  $y_t \rightarrow x_t$  连接。观测了  $x_t$  后，可把它们“吸收”，剩下无向主干。是 HMM 的 2D 类比，但**推断更难**（通常要近似）。
- **高斯 MRF**：把概率写成  $\exp(\eta^\top y - \frac{1}{2}y^\top \Lambda y)$ 。 $\Lambda$  的零元素  $\Leftrightarrow$  图里没边  $\Leftrightarrow$  条件独立。能利用**稀疏性**做高效学习。
- **图 19.10 (VAR 链图)**：时间方向用有向边（回归矩阵  $A_1, A_2$  中的零决定哪条有边），同一时间的相关性用无向边（由  $\Sigma^{-1}$  的零决定）。**DGM+UGM 混合各取所长**。
- **图 19.11 (双向图  $\leftrightarrow$  隐混杂 DAG)**：稀疏协方差可用双向边表示；也可等价成“引入隐藏共同原因”的 DAG，再用 d-分离判独立。
- **图 19.12 (MLN)**：把“抽烟导致癌症”“朋友会互相影响抽烟”这些**一阶逻辑规则**变成**子句+权重**，编译成一个 UGM。对象多了就“展开”更多结点。**MLN=UGM 的模板语言**。

## 三、要点小结

- **Ising/Potts**：用局部相互作用（同/不同）刻画全局结构； $J > 0$  鼓励同标，出现**团块/相变**； $J < 0$  产生多模态与**挫折**。
- **Hopfield**：完全连通的 Ising，可做**关联记忆/模式补全**；常用 ICM 或 Gibbs 做近似推断。
- **Potts + 观测**：做分割时， $p(y)$  (Potts 先验)  $\times \prod p(x_t | y_t)$  (像素似然)；形成**链图**，但 2D 推断困难。
- **高斯 MRF**：联合分布在**信息形式**下是二次型； $\Lambda_{st} = 0 \Leftrightarrow$  条件独立；适合用**稀疏精度矩阵**表示结构。
- **DGM vs UGM (高斯)**：DGM 适合时间顺序与回归（可随协变量变化），UGM 适合同步相关（稀疏精度）；可合成**链图**。
- **双向图与混合模型**：稀疏协方差  $\rightarrow$  双向图；等价于“带隐混杂的 DAG”。

- **MLN**: 用逻辑子句 + 权重定义势函数，生成能处理任意数量对象与关系的 UGM。

想把其中任意一个模型再用一个**具体小例子**（含势表、能量、Z、ICM 更新）走一遍，我可以马上写出来帮你“落地”。

这一节主要在讲什么？我不太懂

已思考 9s

没事~这小节（19.4: MRF 的实例）其实就是用一堆有名的模型来示范：怎样把它们写成无向图模型（或和无向图结合的模型）。核心想让你获得直觉：**局部相互作用 + 势函数  $\Rightarrow$  全局概率模型**。

## 一句话版

把“邻居之间想相同/想不同”的简单规则，做成势函数放在图上，就能得到从磁体、记忆、图像分割到高斯时间序列、逻辑规则等各种模型。

## 关键模型速览（通俗直觉）

- **Ising 模型 ( $\pm 1$ )**：每个点只有两种状态；边权  $J > 0$  鼓励邻居**相同**（铁磁）， $J < 0$  鼓励**不同**（反铁磁，常多模态/“挫折”）。配分函数一般很难算。
- **Hopfield 网络**：Ising 的“全连通版”，可当**关联记忆**：给残缺图像，靠邻域“投票”补全（常用 ICM 或 Gibbs）。图 19.7 展示从遮挡到恢复。
- **Potts 模型 (K 类)**：Ising 的多类别版； $J > 0$  让相邻像素倾向同类，适合做**图像分割先验**。随  $J$  变化会出现**相变**（图 19.8：小块→大块）。和像素似然结合后形成**链图**（图 19.9），2D 推断比 1D 难，通常要近似。
- **高斯 MRF (GGM)**：联合分布写成信息形式  $\exp(\eta^\top y - \frac{1}{2}y^\top \Lambda y)$ 。**精度矩阵  $\Lambda$  的零元  $\Leftrightarrow$  图里没边  $\Leftrightarrow$  条件独立**，便于做**稀疏学习**。
  - 和**有向高斯的对比**：有向图对应**稀疏回归/Cholesky**（顺序明确如时间序列时很自然）；无向图对应**稀疏精度**（同一时刻的相关性）。两者可**混合成链图**（图 19.10 的 VAR(2)）。
  - **双向图**：表示**稀疏协方差**（不连通=无条件独立）；等价于“加一个隐共同原因”的 DAG（图 19.11）。
- **马尔可夫逻辑网络 (MLN)**：把**一阶逻辑规则**（如“抽烟→癌症”，“朋友会互相影响抽烟”）变成**子句 + 权重**，当作势函数，生成能对**任意数量对象/关系**展开的 UGM 模板（图 19.12）。

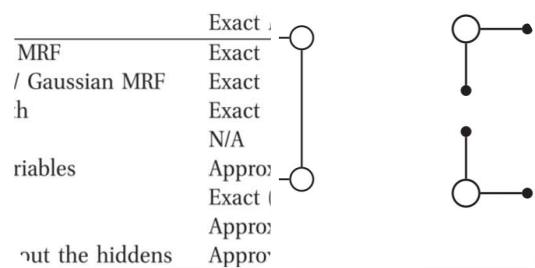
## 图里想表达的点

- 19.7: Hopfield 用邻域一致性把严重遮挡的图像补回来。
- 19.8: Potts 的相变现象;  $J$  大时形成大簇。
- 19.9: 分割模型=无向格子先验  $p(y) = \text{每个像素的证据 } y_t \rightarrow x_t$  (链图)。
- 19.10: 时间用有向 (回归矩阵), 同一时刻相关用无向 (精度矩阵)。
- 19.11: 稀疏协方差 $\leftrightarrow$ 双向边 $\leftrightarrow$ 带隐混杂的有向图。
- 19.12: 逻辑子句加权落地成一个成对 MRF。

## 为什么重要 (带走什么)

- 很多看似不同的模型, 其实都能用MRF/UGM 的语言表达。
- $J > 0$  的“同类偏好”是图像分割、去噪、记忆补全等任务的通用先验。
- 高斯情形里, “学结构”就是学精度矩阵的稀疏模式。
- 逻辑 + 概率 (MLN) 让我们能处理可变对象/关系的复杂世界。

如果你想, 我可以选其中一个 (比如 Potts+分割 或 Hopfield) 用一个小数值例子手把手演一下“势函数 $\rightarrow$ 更新 $\rightarrow$ 结果”的全过程。



翻译并解释以下内容, 翻译不能有遗漏, 解释请通俗易懂且详尽, 因为我的数学基础不太好, 最后总结要点。

Table 19.1 Some methods that can be used to compute approximate ML/ MAP parameter estimates for MRFs/ CRFs. Low tree-width means that, in order for the method to be efficient, the graph must “tree-like”;

Figure 19.13 (a) A small 2d lattice. (b) The representation used by pseudo likelihood. Solid nodes are observed neighbors.

### 19.5 Learning

In this section, we discuss how to perform ML and MAP parameter estimation for MRFs. We will see that this is

quite computationally expensive. For this reason, it is rare to perform Bayesian inference for the parameters of MRFs (although see (Qi et al. 2005)).

### 19.5.1 Training maxent models using gradient methods

Consider an MRF in log-linear form:

$$p(y|\theta) = \frac{1}{Z(\theta)} \exp c^T \theta \prod_{c \in \mathcal{C}} \varphi_c(y) \quad (19.37)$$

where  $c$  indexes the cliques. The scaled log-likelihood is given by

$$l(\theta) = \sum_i N_i \log p(y_i|\theta) = \sum_i N_i c^T \theta - \log Z(\theta) \quad (19.38)$$

Since MRFs are in the exponential family, we know that this function is convex in  $\theta$  (see Section 9.2.3), so it has a unique global maximum which we can find using gradient-based optimizers. In particular, the derivative for the weights of a particular clique,  $c$ , is given by

$$\partial \partial \theta c = \sum_i \varphi_c(y_i) - \partial \partial \theta c \log Z(\theta) \quad (19.39)$$

Exercise 19.1 asks you to show that the derivative of the log partition function wrt  $\theta_c$  is the expectation of the  $c$ 'th feature under the model, i.e.,

$$\partial \log Z(\theta) \partial \theta c = E[\varphi_c(y)|\theta] = \sum_i \varphi_c(y_i)p(y_i|\theta) \quad (19.40)$$

Hence the gradient of the log likelihood is

$$\partial \partial \theta c = \sum_i \varphi_c(y_i) - E[\varphi_c(y)] \quad (19.41)$$

In the first term, we fix  $y$  to its observed values; this is sometimes called the clamped term. In the second term,  $y$  is free; this is sometimes called the unclamped term or contrastive term. Note that computing the unclamped term requires inference in the model, and this must be done once per gradient step. This makes UGM training much slower than DGM training.

The gradient of the log likelihood can be rewritten as the expected feature vector according to the empirical distribution minus the model's expectation of the feature vector:

$$\partial \partial \theta c = E_{\text{emp}}[\varphi_c(y)] - E_p(\cdot|\theta)[\varphi_c(y)] \quad (19.42)$$

At the optimum, the gradient will be zero, so the empirical distribution of the features will match the model's predictions:

$$E_{\text{emp}}[\varphi_c(y)] = E_p(\cdot|\theta)[\varphi_c(y)] \quad (19.43)$$

This is called moment matching. This observation motivates a different optimization algorithm which we discuss in Section 19.5.7.

### 19.5.2 Training partially observed maxent models

Suppose we have missing data and/or hidden variables in

our model. In general, we can represent such models as follows:

$$p(y, h|\theta) = \frac{1}{Z(\theta)} \exp(c \theta^T c \varphi_c(h, y)) \quad (19.44)$$

The log likelihood has the form

$$l(\theta) = \sum_i \log h_i p(y_i, h_i|\theta) = \sum_i \log \frac{1}{Z(\theta)} h_i p^\sim(y_i, h_i|\theta) \quad (19.45)$$

where

$$p^\sim(y, h|\theta) = \exp(c \theta^T c \varphi_c(h, y)) \quad (19.46)$$

is the unnormalized distribution. The term  $h_i p^\sim(y_i, h_i|\theta)$  is the same as the partition function for the whole model, except that  $y$  is fixed at  $y_i$ . Hence the gradient is just the expected features where we clamp  $y_i$ , but average over  $h$ :

$$\partial \partial \theta c \log h_i p^\sim(y_i, h_i|\theta) = E[\varphi_c(h, y_i)|\theta] \quad (19.47)$$

So the overall gradient is given by

$$\partial \partial \theta c = \sum_i \{E[\varphi_c(h, y_i)|\theta] - E[\varphi_c(h, y)|\theta]\} \quad (19.48)$$

The first set of expectations are computed by “clamping” the visible nodes to their observed values, and the second set are computed by letting the visible nodes be free. In both cases, we marginalize over  $h_i$ .

An alternative approach is to use generalized EM, where we use gradient methods in the M step. See (Koller and Friedman 2009, p956) for details.

### 19.5.3 Approximate methods for computing the MLEs of MRFs

When fitting a UGM there is (in general) no closed form solution for the ML or the MAP estimate of the parameters, so we need to use gradient-based optimizers. This gradient requires inference. In models where inference is intractable, learning also becomes intractable. This has motivated various computationally faster alternatives to ML/MAP estimation, which we list in Table 19.1. We discuss some of these alternatives below, and defer others to later sections.

### 19.5.4 Pseudo likelihood

One alternative to MLE is to maximize the pseudo likelihood (Besag 1975), defined as follows:

$$IPL(\theta) = \prod_{d=1}^D \prod_{i=1}^{N_d} \log p(y_{id}|y_{-d}, \theta) \quad (19.49)$$

That is, we optimize the product of the full conditionals, also known as the composite likelihood (Lindsay 1988). Compare this to the objective for maximum likelihood:

$$ML(\theta) = \sum_i \log p(y_i|\theta) = \sum_i \log p(y_i|\theta) \quad (19.50)$$

In the case of Gaussian MRFs, PL is equivalent to ML

(Besag 1975), but this is not true in general (Liang and Jordan 2008).

The PL approach is illustrated in Figure 19.13 for a 2d grid. We learn to predict each node, given all of its neighbors. This objective is generally fast to compute since each full conditional  $p(yid|y_i, -d, \theta)$  only requires summing over the states of a single node,  $yid$ , in order to compute the local normalization constant. The PL approach is similar to fitting each full conditional separately (which is the method used to train dependency networks, discussed in Section 26.2.2), except that the parameters are tied between adjacent nodes.

One problem with PL is that it is hard to apply to models with hidden variables (Parise and Welling 2005). Another more subtle problem is that each node assumes that its neighbors have known values. If node  $t \in nbr(s)$  is a perfect predictor for node  $s$ , then  $s$  will learn to rely completely on node  $t$ , even at the expense of ignoring other potentially useful information, such as its local evidence.

However, experiments in (Parise and Welling 2005; Hoefling and Tibshirani 2009) suggest that PL works as well as exact ML for fully observed Ising models, and of course PL is much faster.

### 19.5.5 Stochastic maximum likelihood

Recall that the gradient of the log-likelihood for a fully observed MRF is given by

$$\nabla\theta(\theta) = 1/N \sum_i [\varphi(y_i) - E[\varphi(y)]] \quad (19.51)$$

The gradient for a partially observed MRF is similar. In both cases, we can approximate the model expectations using Monte Carlo sampling. We can combine this with stochastic gradient descent (Section 8.5.2), which takes samples from the empirical distribution.

Typically we use MCMC to generate the samples. Of course, running MCMC to convergence at each step of the inner loop would be extremely slow. Fortunately, it was shown in (Younes 1989) that we can start the MCMC chain at its previous value, and just take a few steps. In other words, we sample  $y_{s,k}$  by initializing the MCMC chain at  $y_{s,k-1}$ , and then run for a few iterations. This is valid since  $p(y|\theta_k)$  is likely to be close to  $p(y|\theta_{k-1})$ , since we only changed the parameters a small amount. We call this algorithm stochastic maximum likelihood or SML.

(There is a closely related algorithm called persistent contrastive divergence which we discuss in Section 27.7.2.5.)

#### 19.5.6 Feature induction for maxent models

MRFs require a good set of features. One unsupervised way to learn such features, known as feature induction, is to start with a base set of features, and then to continually create new feature combinations out of old ones, greedily adding the best ones to the model. This approach was first proposed in (Pietra et al. 1997; Zhu et al. 1997), and was later extended to the CRF case in (McCallum 2003). To illustrate the basic idea, we present an example from (Pietra et al. 1997), which described how to build unconditional probabilistic models to represent English spelling. Initially the model has no features, which represents the uniform distribution. The algorithm starts by choosing to add the feature

$$\varphi_1(y) = t \mathbf{1}(y_t \in \{a, \dots, z\}) \quad (19.52)$$

which checks if any letter is lower case or not. After the feature is added, the parameters are (re)-fit by maximum likelihood. For this feature, it turns out that  $\hat{\theta}_1 = 1.944$ , which means that a word with a lowercase letter in any position is about  $e^{1.944} \approx 7$  times more likely than the same word without a lowercase letter in that position.

Some samples from this model, generated using (annealed) Gibbs sampling (Section 24.2), are shown below.

m, r, xeko, iijiir, b, to, jz, gsr, wq, vf, x, ga, msmGh, pcp, d, oziVlal,  
 hzag, yzop, io, advzmxnv, ijv\_bolft, x, emx, kayerf, mlj,  
 rawzyb, jp, ag,  
 ctdnnnbg, wgdw, t, kguv, cy, spxcq, uzflbbf, dxtkkn, cxwx,  
 jpd, ztzh, lv,  
 zhpkvnu, l^, r, qee, nynrx, atze4n, ik, se, w, lrh, hp+,  
 yrqyka'h, zcngotcnx,  
 igcump, zjcjs, lqpWiqu, cefmfhc, o, lb, fdcY, tzby, yopxmvk,  
 by, fz, t, govyccm,  
 ijjiduwfzo, 6xr, duh, ejv, pk, pjw, l, fl, w

The second feature added by the algorithm checks if two adjacent characters are lower case:

$$\varphi_2(y) = s \sim t \mathbf{1}(y_s \in \{a, \dots, z\}, y_t \in \{a, \dots, z\}) \quad (19.53)$$

Now the model has the form

$$p(y) = 1 Z \exp(\theta_1 \varphi_1(y) + \theta_2 \varphi_2(y)) \quad (19.54)$$

Continuing in this way, the algorithm adds features for the strings  $s >$  and  $ing >$ , where  $>$  represents the end of word, and for various regular expressions such as [0-9], etc. Some samples from the model with 1000 features, generated using (annealed) Gibbs sampling, are shown below.

was, reaser, in, there, to, will, „ was, by, homes, thing, be,  
reloverated,  
ther, which, conists, at, fores, anditing, with, Mr., proveral,  
the, „ \*\*\*,  
on't, prolling, prothere, „ mento, at, yaou, 1, chestraing,  
for, have, to,  
intrally, of, qut, .. best, compers, \*\*\*, cluseliment, uster, of,  
is, deveral,  
this, thise, of, offset, inatever, thifer, constrained, stater,  
vill, in, thase,  
in, youse, menttering, and, ., of, in, verate, of, to

This approach of feature learning can be thought of as a form of graphical model structure learning (Chapter 26), except it is more fine-grained: we add features that are useful, regardless of the resulting graph structure.

However, the resulting graphs can become densely connected, which makes inference (and hence parameter estimation) intractable.

#### 19.5.7 Iterative proportional fitting (IPF)

Consider a pairwise MRF where the potentials are represented as tables, with one parameter per variable setting. We can represent this in log-linear form using

$$\psi_{st}(ys, yt) = \exp \left( \theta^T \mathbf{st} \right) \quad (19.55)$$

and similarly for  $\psi_t(yt)$ . Thus the feature vectors are just indicator functions.

From Equation 19.43, we have that, at the maximum of the likelihood, the empirical expectation of the features equals the model's expectation:

$$E_{\text{emp}} [\mathbf{I}(ys = j, yt = k)] = E_p(\cdot | \theta) [\mathbf{I}(ys = j, yt = k)] \quad (19.56)$$

$$p_{\text{emp}}(ys = j, yt = k) = p(ys = j, yt = k | \theta) \quad (19.57)$$

where  $p_{\text{emp}}$  is the empirical probability:

$$p_{\text{emp}}(ys = j, yt = k) = \frac{N_{st,jk}}{N} \quad (19.58)$$

For a general graph, the condition that must hold at the optimum is

$$p_{\text{emp}}(yc) = p(yc | \theta) \quad (19.59)$$

For a special family of graphs known as decomposable graphs (defined in Section 20.4.1), one can show that  $p(\mathbf{yc}|\theta) = \psi c(\mathbf{yc})$ . However, even if the graph is not decomposable, we can imagine trying to enforce this condition. This suggests an iterative coordinate ascent scheme where at each step we compute

$$\psi^{t+1} c(\mathbf{yc}) = \psi^t c(\mathbf{yc}) \times p_{\text{emp}}(\mathbf{yc}) p(\mathbf{yc}|\psi^t) \quad (19.60)$$

where the multiplication is elementwise. This is known as iterative proportional fitting or IPF (Fienberg 1970; Bishop et al. 1975).

### 19.5.7.1 Example

Let us consider a simple example from [http://en.wikipedia.org/wiki/Iterative\\_proportional\\_fitting](http://en.wikipedia.org/wiki/Iterative_proportional_fitting). We have two binary variables,  $Y_1$  and  $Y_2$ , where  $Y_{n1} = 1$  if man  $n$  is left handed, and  $Y_{n1} = 0$  otherwise; similarly,  $Y_{n2} = 1$  if woman  $n$  is left handed, and  $Y_{n2} = 0$  otherwise. We can summarize the data using the following  $2 \times 2$  contingency table:

	right-handed	left-handed	Total
male	43	9	52
female	44	4	48
Total	87	13	100

Suppose we want to fit a disconnected graphical model containing nodes  $Y_1$  and  $Y_2$  but with no edge between them. That is, we want to find vectors  $\psi_1$  and  $\psi_2$  such that  $M = \psi_1 \psi_2^T \approx C$ , where  $M$  are the model's expected counts, and  $C$  are the empirical counts. By moment matching, we find that the row and column sums of the model must exactly match the row and column sums of the data. One possible solution is to use  $\psi_1 = [0.5200, 0.4800]$  and  $\psi_2 = [87, 13]$ . Below we show the model's predictions,  $M = \psi_1 \psi_2^T$ .

	right-handed	left-handed	Total
male	45.24	6.76	52
female	41.76	6.24	48
Total	87	13	100

It is easy to see that this matches the required constraints. See `IPFdemo2x2` for some Matlab code that computes these numbers. This method is easily generalized to arbitrary graphs.

### 19.5.7.2 Speed of IPF

IPF is a fixed point algorithm for enforcing the moment matching constraints and is guaranteed to converge to the global optimum (Bishop et al. 1975). The number of iterations depends on the form of the model. If the graph is decomposable, then IPF converges in a single iteration, but in general, IPF may require many iterations.

It is clear that the dominant cost of IPF is computing the required marginals under the model. Efficient methods, such as the junction tree algorithm (Section 20.4), can be used, resulting in something called efficient IPF (Jirousek and Preucil 1995).

Nevertheless, coordinate descent can be slow. An alternative method is to update all the parameters at once, by simply following the gradient of the likelihood. This gradient approach has the further significant advantage that it works for models in which the clique potentials may not be fully parameterized, i.e., the features may not consist of all possible indicators for each clique, but instead can be arbitrary. Although it is possible to adapt IPF to this setting of general features, resulting in a method known as iterative scaling, in practice the gradient method is much faster (Malouf 2002; Minka 2003).

#### 19.5.7.3 Generalizations of IPF

We can use IPF to fit Gaussian graphical models: instead of working with empirical counts, we work with empirical means and covariances (Speed and Kiiveri 1986). It is also possible to create a Bayesian IPF algorithm for sampling from the posterior of the model's parameters (see e.g., (Dobra and Massam 2010)).

#### 19.5.7.4 IPF for decomposable graphical models

There is a special family of undirected graphical models known as decomposable graphical models. This is formally defined in Section 20.4.1, but the basic idea is that it contains graphs which are "tree-like". Such graphs can be represented by UGMs or DGMs without any loss of information.

In the case of decomposable graphical models, IPF converges in one iteration. In fact, the MLE has a closed form solution (Lauritzen 1996). In particular, for tabular potentials we have

$$\hat{\psi}^c(y_c = k) = N \sum_{i=1}^N I(y_i, c = k) / N \quad (19.61)$$

and for Gaussian potentials, we have

$$\hat{\mu}^c = \bar{y}_c \quad (19.62)$$

By using conjugate priors, we can also easily compute the full posterior over the model parameters in the decomposable case, just as we did in the DGM case. See (Lauritzen 1996) for details.

已思考 16s

下面先**完整翻译**（含全部公式与编号、图表说明），再给出**通俗解释与要点总结**。

---

## 一、逐条翻译（不遗漏）

**表 19.1** 计算 MRF/CRF 的近似极大似然 (ML) / 极大后验 (MAP) 参数估计的若干方法。  
低树宽表示：要想方法高效，图结构必须“像树”一样。

- **Closed form (闭式解)** | 限制：仅限弦图/可分解的 MRF | 精确？：是 | 见：第 19.5.7.4 节
- **IPF (迭代比例拟合)** | 限制：仅表格/高斯 MRF | 精确？：是 | 见：第 19.5.7 节
- **基于梯度的优化** | 限制：树宽低 (图近似树状) | 精确？：是 | 见：第 19.5.1 节
- **最大间隔训练** | 限制：仅 CRFs | 精确？：不适用 (N/A) | 见：第 19.7 节
- **伪似然** | 限制：无隐变量 | 精确？：近似 | 见：第 19.5.4 节
- **随机最大似然 (SML)** | 限制：— | 精确？：精确 (受蒙特卡洛误差影响) | 见：第 19.5.5 节
- **对比散度 (CD)** | 限制：— | 精确？：近似 | 见：第 27.7.2.4 节
- **最小概率流 (MPF)** | 限制：能把隐变量积分掉 | 精确？：近似 | 见：Sohl-Dickstein 等 (2011)

**图 19.13** (a) 一个很小的二维格子。 (b) 伪似然使用的表示法。实心节点是已观测的邻居。

---

## 19.5 学习

本节讨论如何对 MRF 做 ML 与 MAP 参数估计。我们将看到这在计算上很昂贵。因此很少对 MRF 的参数做**完整的贝叶斯推断**（但参见 Qi 等, 2005）。

### 19.5.1 用梯度法训练最大熵模型

考虑对数线性形式的 MRF：

$$p(y \mid \theta) = \frac{1}{Z(\theta)} \exp \left( \sum_c \theta_c^\top \phi_c(y) \right) \quad (19.37)$$

其中  $c$  为团索引。缩放后的对数似然为

$$\ell(\theta) = \frac{1}{N} \sum_i \log p(y_i \mid \theta) = \frac{1}{N} \sum_i \sum_c \theta_c^\top \phi_c(y_i) - \log Z(\theta) \quad (19.38)$$

MRF 属于指数族，因此  $\ell(\theta)$  对  $\theta$  凸（见 9.2.3 节），存在唯一全局最大，可用**基于梯度的优化器求解**。对某个团  $c$  的权重，梯度为

$$\frac{\partial \ell}{\partial \theta_c} = \frac{1}{N} \sum_i \phi_c(y_i) - \frac{\partial}{\partial \theta_c} \log Z(\theta) \quad (19.39)$$

**练习 19.1** 要你证明： $\log Z$  对  $\theta_c$  的导数是该团特征的**模型期望**：

$$\frac{\partial \log Z(\theta)}{\partial \theta_c} = E[\phi_c(y) | \theta] = \sum_y \phi_c(y) p(y | \theta) \quad (19.40)$$

因此对数似然的梯度为

$$\frac{\partial \ell}{\partial \theta_c} = \frac{1}{N} \sum_i \phi_c(y_i) - E[\phi_c(y)] \quad (19.41)$$

第一项把  $y$  钳住在观测值 (clamped)；第二项让  $y$  自由 (unclamped/对比项)。计算自由项需要在模型里做推断，而且每一步梯度都要做一次——这让 UGM 的训练远慢于 DGM。

也可写成“经验分布的期望 – 模型的期望”：

$$\frac{\partial \ell}{\partial \theta_c} = E_{p_{\text{emp}}}[\phi_c(y)] - E_{p(\cdot|\theta)}[\phi_c(y)] \quad (19.42)$$

在最优点，梯度为 0，于是**矩匹配**成立：

$$E_{p_{\text{emp}}}[\phi_c(y)] = E_{p(\cdot|\theta)}[\phi_c(y)] \quad (19.43)$$

这启发了另一种优化算法（见 19.5.7 节）。

## 19.5.2 训练部分观测的最大熵模型

若有缺失数据/隐变量，模型可写为

$$p(y, h | \theta) = \frac{1}{Z(\theta)} \exp \left( \sum_c \theta_c^\top \phi_c(h, y) \right) \quad (19.44)$$

对数似然为

$$\ell(\theta) = \frac{1}{N} \sum_i \log \sum_{h_i} p(y_i, h_i | \theta) = \frac{1}{N} \sum_i \log \frac{1}{Z(\theta)} \sum_{h_i} \tilde{p}(y_i, h_i | \theta) \quad (19.45)$$

其中

$$\tilde{p}(y, h | \theta) = \exp \left( \sum_c \theta_c^\top \phi_c(h, y) \right) \quad (19.46)$$

是未归一化分布。项  $\sum_{h_i} \tilde{p}(y_i, h_i | \theta)$  与全模型的配分函数类似，只是把  $y$  固定为  $y_i$ 。故其梯度等于在\*\*钳住  $y_i$ \*\*并对  $h$  求期望的特征：

$$\frac{\partial}{\partial \theta_c} \log \sum_{h_i} \tilde{p}(y_i, h_i | \theta) = E[\phi_c(h, y_i) | \theta] \quad (19.47)$$

于是总体梯度为

$$\frac{\partial \ell}{\partial \theta_c} = \frac{1}{N} \sum_i \{E[\phi_c(h, y_i) | \theta] - E[\phi_c(h, y) | \theta]\} \quad (19.48)$$

两项都要对  $h_i$  积分；第一项钳住可见结点，第二项放开。

另一条路是广义 EM：M 步用梯度法（见 Koller & Friedman 2009，第 956 页）。

### 19.5.3 近似 MLE

一般地，UGM 的 ML / MAP 没有闭式解，需梯度优化；而梯度又需推断。推断不可 tractable 时，学习也不可 tractable。于是出现多种更快的替代法（表 19.1），本节挑几种介绍，其余放到后文。

### 19.5.4 伪似然 (PL)

MLE 的一个替代是最大化伪似然 (Besag 1975)：

$$\ell_{PL}(\theta) = \sum_y p_{emp}(y) \sum_{d=1}^D \log p(y_d | y_{-d}) = \frac{1}{N} \sum_{i=1}^N \sum_{d=1}^D \log p(y_{id} | y_{i,-d}, \theta) \quad (19.49)$$

即优化各全条件分布的乘积（也叫组合似然，Lindsay 1988）。对照 MLE 目标：

$$ML(\theta) = \sum_y p_{emp}(y) \log p(y | \theta) = \sum_{i=1}^N \log p(y_i | \theta) \quad (19.50)$$

在高斯 MRF 中，PL 等价于 ML (Besag 1975)；一般情况不等价 (Liang & Jordan 2008)。

图 19.13 说明：在 2D 网格上，我们学习“给定所有邻居，预测该结点”。每个条件  $p(y_{id} | y_{i,-d}, \theta)$  只需对单个结点的取值求和来做局部归一化，因此计算快。它类似“分别拟合每个全条件”（依赖网络的训练法，见 26.2.2），只不过相邻结点间参数共享。

PL 的问题：对含隐变量的模型不易用 (Parise & Welling 2005)。更隐蔽的问题：每个结点假设邻居值确定。如果某邻居是完美预测器，该结点可能过度依赖它而忽视其它信息（如局部证据）。

但实验证明 (Parise & Welling 2005; Hoefling & Tibshirani 2009)：对完全可观测的 Ising，PL 与精确 ML 表现相当，且更快。

### 19.5.5 随机最大似然 (SML)

回忆完全观测 MRF 的梯度：

$$\nabla_{\theta} \ell(\theta) = \frac{1}{N} \sum_i [\phi(y_i) - E[\phi(y)]] \quad (19.51)$$

部分观测情形类似。两者都可用蒙特卡洛估计模型期望，并与随机梯度下降结合 (8.5.2)。

通常用 MCMC 取样。若每一步都把 MCMC 跑到收敛会非常慢。幸运的是，Younes (1989) 证明：可以把马尔可夫链从**上一步的样本**开始，只跑**少量步**即可。即用上次的  $y_{s,k-1}$  初始化，迭代几步得到  $y_{s,k}$ 。这是合理的，因为参数变化小， $p(y | \theta_k)$  近似  $p(y | \theta_{k-1})$ 。该算法称 SML（与之密切相关的是**持久对比散度**，见 27.7.2.5）。

### 19.5.6 最大熵模型的特征归纳 (Feature Induction)

MRF 需要一组好的特征。无监督的学法之一是**特征归纳**：从一组基础特征出发，不断把已有特征**组合**出新特征，**贪心地加入效果最好的**。最早见于 Pietra 等 (1997)；Zhu 等 (1997)，CRF 中的扩展见 McCallum (2003)。

举 Pietra 等 (1997) 的例子：为**英文拼写**建无条件概率模型。起始模型无特征（均匀分布）。算法首先加入

$$\phi_1(y) = \sum_t I(y_t \in \{a, \dots, z\}) \quad (19.52)$$

检测是否有小写字母。随后用 ML 重新估计参数；该特征得到  $\hat{\theta}_1 = 1.944$ ，意味着**只要某位置是小写**，该词的概率约为  $e^{1.944} \approx 7$  倍。用（退火的）Gibbs 采样得到的样本示例略（原文列出一串“词样本”）。

第二个加入的特征检测**相邻两个字符均为小写**：

$$\phi_2(y) = \sum_{s \sim t} I(y_s \in a..z, y_t \in a..z) \quad (19.53)$$

此时

$$p(y) = \frac{1}{Z} \exp (\theta_1 \phi_1(y) + \theta_2 \phi_2(y)) \quad (19.54)$$

继续下去，算法会为字符串“s>”“ing>”（> 为词尾）及正则如 “[0-9]” 等加入特征。用 1000 个特征训练后采样，得到更像“英语”的词串。

这种**特征学习**可看作一种**图模型结构学习**（第 26 章），但更细粒度：添加“有用”的特征而

不管图会变成什么形状。缺点是最终图可能很稠密，推断（进而参数估计）就可能不可 tractable。

### 19.5.7 迭代比例拟合 (IPF)

考虑成对 MRF，其势用表表示、**每个取值一个参数**。可用对数线性写成

$$\psi_{st}(y_s, y_t) = \exp\{\theta_{st}^\top [I(y_s=1, y_t=1), \dots, I(y_s=K, y_t=K)]\} \quad (19.55)$$

$\psi_t(y_t)$  类似。故特征就是**指示函数**。

由式 (19.43)，在似然最大时：

$$E_{p_{\text{emp}}}[I(y_s=j, y_t=k)] = E_{p(\cdot|\theta)}[I(y_s=j, y_t=k)] \quad (19.56)$$

即

$$p_{\text{emp}}(y_s=j, y_t=k) = p(y_s=j, y_t=k | \theta) \quad (19.57)$$

其中经验概率

$$p_{\text{emp}}(y_s=j, y_t=k) = \frac{N_{st,jk}}{N} = \frac{1}{N} \sum_{n=1}^N I(y_s^n=j, y_t^n=k) \quad (19.58)$$

一般图的最优条件是

$$p_{\text{emp}}(y_c) = p(y_c | \theta) \quad (19.59)$$

对一类称为**可分解图** (20.4.1 节定义)，可得  $p(y_c | \theta) = \psi_c(y_c)$ 。即便不是可分解，我们也可以强制这个条件，得到**坐标上升**的迭代：

$$\psi_c^{(t+1)}(y_c) = \psi_c^{(t)}(y_c) \times \frac{p_{\text{emp}}(y_c)}{p(y_c | \psi^{(t)})} \quad (\text{逐元素相乘}) \quad (19.60)$$

这就是 IPF (Fienberg 1970; Bishop 等 1975)。

#### 19.5.7.1 例子

来自维基百科的小例子：两个二值变量  $Y_1, Y_2$ 。 $Y_1^n = 1$  表示第  $n$  个**男性**左撇； $Y_2^n = 1$  表示第  $n$  个**女性**左撇。数据列联表：

	右撇	左撇	合计
男性	43	9	52
女性	44	4	48
合计	87	13	100

要拟合一个**不相连的图** ( $Y_1$  与  $Y_2$  无边)。即找  $\psi_1, \psi_2$  使得模型计数  $M = \psi_1 \psi_2^\top \approx C$  (经验计数)。矩匹配要求模型的**行和、列和**与数据完全一致。一个解:  $\psi_1 = [0.5200, 0.4800]$ ,  $\psi_2 = [87, 13]$ 。则

- 男: 右 45.24、左 6.76 (合 52)
- 女: 右 41.76、左 6.24 (合 48)
- 合计: 右 87、左 13 (合 100)

满足所需约束。该法可推广到任意图。

### 19.5.7.2 IPF 的速度

IPF 是一个**不动点算法**, 能收敛到全局最优 (Bishop 等 1975)。所需迭代次数取决于模型: **可分解图时一轮即收敛**; 一般情况可能需要很多轮。

主要成本在于**计算模型边缘分布**。用**结点树 (junction tree)** 等高效推断法可得到“**高效 IPF**” (Jirousek & Preucil 1995)。

**尽管如此, 坐标下降仍可能慢。另一法: 一次性更新全部参数, 即走似然梯度。**梯度法的重大优势: **适用于任意特征** (不要求每个团用完整指示特征); 即使把 IPF 推广到一般特征 (叫“**迭代缩放**”), 实践里**梯度法更快** (Malouf 2002; Minka 2003)。

### 19.5.7.3 推广

IPF 可用于**高斯图模型**: 把经验计数换成**经验均值与协方差** (Speed & Kiiveri 1986)。也可以构造**贝叶斯 IPF**来从参数后验采样 (如 Dobra & Massam 2010)。

### 19.5.7.4 可分解图的 IPF

有一类无向图叫**可分解图** (20.4.1 节正式定义), 直观上“像树”。这类图用 UGM 或 DGM 表示都不丢信息。

对可分解图, IPF 一轮收敛, MLE 甚至有**闭式解** (Lauritzen 1996)。

- **表格势:**

$$\hat{\psi}_c(y_c = k) = \frac{1}{N} \sum_{i=1}^N I(y_{i,c} = k) \quad (19.61)$$

- **高斯势:**

$$\hat{\mu}_c = \frac{1}{N} \sum_{i=1}^N y_{ic}, \quad \hat{\Sigma}_c = \frac{1}{N} \sum_i (y_{ic} - \hat{\mu}_c)(y_{ic} - \hat{\mu}_c)^\top \quad (19.62)$$

配合共轭先验, 还能像 DGM 一样**直接给出后验** (见 Lauritzen 1996)。

## 二、通俗解释 (把难点讲直白)

- **为什么训练 MRF 难?**

因为对数似然的梯度 = 【数据里的特征均值】 - 【模型里的特征均值】。第二项要算模型的期望 $\Rightarrow$ 要做**推断**，而且每次更新参数都要再推断一次；若图有环/大团，推断本身就难。

- **两种“期望”的心法**

- **clamped (夹住)**：把观测  $y$  固定，容易算。
- **unclamped (放开)**：按当前参数对所有  $y$  求期望，最难。  
训练就是让两者**相等 (矩匹配)**。

- **应对之道**

1. **梯度 + 近似推断**：Junction tree/Loopy BP/采样；
2. **伪似然**：把联合拆成很多条件，每次只归一个结点——快，但有偏；
3. SML/CD：用**持久的** MCMC近似模型期望；
4. IPF：直接“把模型边缘拉到等于数据边缘”，一条团一条团**按比例缩放势**；可分解图时一轮收敛；
5. **特征归纳**：边训练边“造特征”，让模型表达力够用，但要小心图变得太稠密。

- **什么时候能有闭式解？**

\*\*可分解图（树/弦图）\*\*就能：表格势=经验频率，高斯势=样本均值/协方差。

---

### 三、要点小结

- **MRF 训练的核心**： $\nabla \ell = \text{数据期望} - \text{模型期望}$ 。难点在**模型期望需要推断与配分函数**。
- **完全观测**：用梯度法；**部分观测**：再对隐变量做期望（或用广义 EM）。
- **近似学习方法族**（各有优缺）：
  - **伪似然 (PL)**：快、可扩展；完全观测 Ising 很好用；对隐变量与过度依赖邻居要谨慎。
  - SML/CD：用（持久）MCMC 近似模型期望，结合随机梯度。
  - IPF：按**边缘比例**迭代缩放势，满足**矩匹配**；可分解图一轮到位；一般图需迭代且要能算边缘。
  - **闭式解**：仅在**可分解图**存在（表格=频率，高斯=样本均值/协方差）。
  - **最小概率流**等其它法：不走配分函数梯度，进一步加速但近似。
- **工程建议**：
  - 图“树状/树宽低” $\rightarrow$ **梯度+确切推断**；
  - 完全观测的格子/Ising  $\rightarrow$  PL 或 SML；

- 高斯图 → 直接在**精度矩阵/边缘上**做 (可用 IPF/稀疏学习) ;
- 需要强表达力 → **特征归纳**, 但注意推断复杂度。

如果你想, 我可以根据一个迷你 Ising 网格 (如  $4 \times 4$ ) , 演示一次PL 与 SML的训练步骤 (包含梯度计算与一次 MCMC 近似) , 把这些概念进一步落到代码/数字上。

这一节主要在讲什么? 我不太懂

已思考 13s

这小节 (19.5: Learning) 说的就是一件事——**怎么给 MRF/CRF 学参数**。核心结论: **很难、很贵**, 因为每一步都要做推断 (算配分函数/模型期望) 。

## 一句话版

训练 MRF/CRF = 让“数据里的特征均值”和“模型预测的特征均值”一致 (矩匹配) 。  
但模型期望要在图上做推断, 所以常用各种近似: 伪似然、采样+随机梯度、IPF 等。

## 关键点 (通俗解释)

### 1. 最大熵/对数线性训练 (标准做法)

- 把模型写成  $p(y) \propto \exp(\sum_c \theta_c^\top \phi_c(y))$ 。
- 对数似然的梯度 = **数据期望** - **模型期望**。
  - clamped (夹住) : 用观测的  $y$  直接算特征 (容易) 。
  - unclamped (放开) : 按当前参数对所有  $y$  求期望 (难, 要推断) 。
- 最优时两者相等 ⇒ **矩匹配**。
- **有隐变量时**: clamped 项要对隐变量再取期望, 难度更大 (可用广义 EM) 。

### 2. 为什么难?

- 每做一步梯度更新, 都要再做一次推断 (算模型期望/配分函数) 。图有环、团大时非常慢。

### 3. 一堆更快的替代法 (表 19.1)

- **闭式解**: 只在**可分解/弦图**才有。
- **IPF (迭代比例拟合)** : 对**表格/高斯MRF**精确; 按“数据边缘 ÷ 模型边缘”的比例**缩放势**, 强制矩匹配。可分解图一轮收敛。

- 梯度法 + 精确推断：图树宽低时可行。
- 最大间隔：只针对 CRF (判别式)。
- 伪似然 (PL)：最大化  $\prod_d p(y_d | y_{-d})$ ；快，适合完全观测的格子/Ising；但对隐变量不友好，易过度依赖邻居 (图 19.13)。
- 随机最大似然 (SML) /持久链：用 MCMC 近似模型期望，每步从上一次样本热启动，只跑几步就更新。
- 对比散度 (CD)、最小概率流 (MPF)：进一步避开精确配分函数，属于近似。

#### 4. 特征归纳 (Feature induction)

- MRF 很吃特征。可以从简单特征出发，贪心地组合/添加能提升似然的特征 (如检测“ing”、相邻小写等)，但图可能变很稠密，推断会变难。

#### 5. IPF 要点

- 目标：让每个团的**经验边缘** = **模型边缘** (矩匹配)。
- 迭代式： $\psi_c^{(t+1)} \leftarrow \psi_c^{(t)} \times \frac{p_{\text{emp}}(y_c)}{p(y_c | \psi^{(t)})}$  (逐元素)。
- 主要花费在算边缘分布；可配合**结点树**提速。
- **可分解图**：表格势=经验频率，高斯势=样本均值/协方差，甚至有**闭式** MLE。

### 图表在说什么？

- 表 19.1：把常见训练法按“适用条件/是否精确/在哪一节”做了路线图。
- 图 19.13：伪似然训练时，每个空心节点都把周围实心邻居当已知，只学它的**条件分布**。

### 什么时候用哪个法？（实用清单）

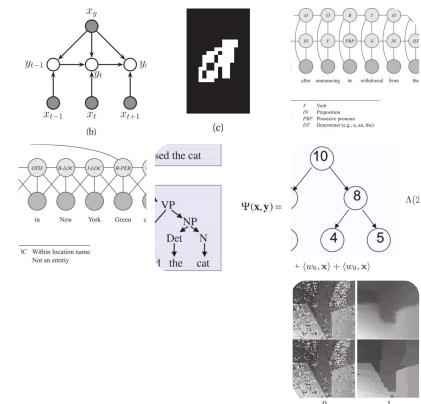
- 图接近树/树宽低 → 梯度 + 精确推断。
- 完全观测的格子/Ising → 伪似然或SML (快且效果好)。
- 需要精确边缘且是可分解图 → IPF/闭式。
- 有隐变量 → SML/CD (采样近似) 或广义 EM。
- 特征很多/想自动造特征 → 特征归纳 (注意推断复杂度)。

### 记住这几句

- 训练 MRF 的难点=模型期望 (要推断)。
- 矩匹配是核心：让数据特征  $\approx$  模型特征。

- PL/SML/IPF 是三条常用捷径：

- PL 快；
- SML 用持久采样近似；
- IPF 直接把边缘“拧到位”，可分解图一轮到底。



翻译并解释以下内容，翻译不能有遗漏，解释请通俗易懂且详尽，因为我的数学基础不太好，最后总结要点。

Figure 19.14 Various models for sequential data. (a) A generative directed HMM. (b) A discriminative directed MEMM. (c) A discriminative undirected CRF.

Figure 19.15 Example of handwritten letter recognition. In the word 'brace', the 'r' and the 'c' look very similar, but can be disambiguated using context.

Figure 19.16 A CRF for joint POS tagging and NP segmentation.

Figure 19.17 A skip-chain CRF for named entity recognition.

Figure 19.18 Illustration of a simple parse tree based on a context free grammar in Chomsky normal form. The feature vector  $\varphi(x, y) = \Psi(x, y)$  counts the number of times each production rule was used.

Figure 19.19 Illustration of a simple label taxonomy, and how it can be used to compute a distributed representation for the label for class 2. In this figure,  $\varphi(x) = x$ ,  $\varphi(y=2) = \Lambda(2)$ ,  $\varphi(x,y)$  is denoted by  $\Psi(x, 2)$ , and  $wT\varphi(x,y)$  is denoted by  $w$ ,  $\Psi(x, 2)$ .

Figure 19.20 Illustration of belief propagation for stereo depth estimation. Left column: image and true disparities. Remaining columns: initial estimate, estimate after 1

iteration, and estimate at convergence. Top row: Gaussian edge potentials. Bottom row: robust edge potentials.

## 19.6 Conditional random fields (CRFs)

A conditional random field or CRF (Lafferty et al. 2001), sometimes a discriminative random field (Kumar and Hebert 2003), is just a version of an MRF where all the clique potentials are conditioned on input features:

$$p(y|x,w) = \frac{1}{Z(x,w)} \exp(\sum_c \psi_c(y_c|x,w)) \quad (19.63)$$

A CRF can be thought of as a structured output extension of logistic regression. We will usually assume a log-linear representation of the potentials:

$$\psi_c(y_c|x,w) = \exp(w^T \phi(x, y_c)) \quad (19.64)$$

where  $\phi(x, y_c)$  is a feature vector derived from the global inputs  $x$  and the local set of labels  $y_c$ . We will give some examples below which will make this notation clearer.

The advantage of a CRF over an MRF is analogous to the advantage of a discriminative classifier over a generative classifier (see Section 8.6), namely, we don't need to "waste resources" modeling things that we always observe. Instead we can focus our attention on modeling what we care about, namely the distribution of labels given the data.

Another important advantage of CRFs is that we can make the potentials (or factors) of the model be data-dependent. For example, in image processing applications, we may "turn off" the label smoothing between two neighboring nodes  $s$  and  $t$  if there is an observed discontinuity in the image intensity between pixels  $s$  and  $t$ . Similarly, in natural language processing problems, we can make the latent labels depend on global properties of the sentence, such as which language it is written in. It is hard to incorporate global features into generative models.

The disadvantage of CRFs over MRFs is that they require labeled training data, and they are slower to train, as we explain in Section 19.6.3. This is analogous to the strengths and weaknesses of logistic regression vs naive Bayes, discussed in Section 8.6.

### 19.6.1 Chain-structured CRFs, MEMMs and the label-bias problem

The most widely used kind of CRF uses a chain-structured graph to model correlation amongst neighboring labels.

Such models are useful for a variety of sequence labeling tasks (see Section 19.6.2).

Traditionally, HMMs (discussed in detail in Chapter 17) have been used for such tasks. These are joint density models of the form

$$p(x, y|w) = \prod_{t=1}^T p(y_t|y_{t-1}, w)p(x_t|y_t, w) \quad (19.65)$$

where we have dropped the initial  $p(y_1)$  term for simplicity. See Figure 19.14(a). If we observe both  $x_t$  and  $y_t$  for all  $t$ , it is very easy to train such models, using techniques described in Section 17.5.1.

An HMM requires specifying a generative observation model,  $p(x_t|y_t, w)$ , which can be difficult. Furthermore, each  $x_t$  is required to be local, since it is hard to define a generative model for the whole stream of observations,  $x = x_1:T$ .

An obvious way to make a discriminative version of an HMM is to “reverse the arrows” from  $y_t$  to  $x_t$ , as in Figure 19.14(b). This defines a directed discriminative model of the form

$$p(y|x, w) = \prod_{t=1}^T p(y_t|y_{t-1}, x, w) \quad (19.66)$$

where  $x = (x_1:T, x_g)$ ,  $x_g$  are global features, and  $x_t$  are features specific to node  $t$ . (This partition into local and global is not necessary, but helps when comparing to HMMs.) This is called a maximum entropy Markov model or MEMM (McCallum et al. 2000; Kakade et al. 2002).

An MEMM is simply a Markov chain in which the state transition probabilities are conditioned on the input features. (It is therefore a special case of an input-output HMM, discussed in Section 17.6.3.) This seems like the natural generalization of logistic regression to the structured output setting, but it suffers from a subtle problem known (rather obscurely) as the label bias problem (Lafferty et al. 2001). The problem is that local features at time  $t$  do not influence states prior to time  $t$ . This follows by examining the DAG, which shows that  $x_t$  is d-separated from  $y_{t-1}$  (and all earlier time points) by the v-structure at  $y_t$ , which is a hidden child, thus blocking the information flow.

To understand what this means in practice, consider the part of speech (POS) tagging task. Suppose we see the word “banks”; this could be a verb (as in “he banks at BoA”), or a noun (as in “the river banks were overflowing”). Locally the POS tag for the word is

ambiguous. However, suppose that later in the sentence, we see the word “fishing”; this gives us enough context to infer that the sense of “banks” is “river banks”. However, in an MEMM (unlike in an HMM and CRF), the “fishing” evidence will not flow backwards, so we will not be able to disambiguate “banks”.

Now consider a chain-structured CRF. This model has the form

$$p(y|x,w) = \frac{1}{Z(x,w)} \prod_{t=1}^T \psi(y_t|x,w) \quad (19.67)$$

From the graph in Figure 19.14(c), we see that the label bias problem no longer exists, since  $y_t$  does not block the information from  $x_t$  from reaching other  $y_t$  nodes.

The label bias problem in MEMMs occurs because directed models are locally normalized, meaning each CPD sums to 1. By contrast, MRFs and CRFs are globally normalized, which means that local factors do not need to sum to 1, since the partition function  $Z$ , which sums over all joint configurations, will ensure the model defines a valid distribution. However, this solution comes at a price: we do not get a valid probability distribution over  $y$  until we have seen the whole sentence, since only then can we normalize over all configurations. Consequently, CRFs are not as useful as DGMs (whether discriminative or generative) for online or real-time inference. Furthermore, the fact that  $Z$  depends on all the nodes, and hence all their parameters, makes CRFs much slower to train than DGMs, as we will see in Section 19.6.3.

### 19.6.2 Applications of CRFs

CRFs have been applied to many interesting problems; we give a representative sample below. These applications illustrate several useful modeling tricks, and will also provide motivation for some of the inference techniques we will discuss in Chapter 20.

#### 19.6.2.1 Handwriting recognition

A natural application of CRFs is to classify hand-written digit strings, as illustrated in Figure 19.15. The key observation is that locally a letter may be ambiguous, but by depending on the (unknown) labels of one’s neighbors, it is possible to use context to reduce the error rate. Note that the node potential,  $\psi_t(y_t|x_t)$ , is often taken to be a probabilistic discriminative classifier, such as a neural network or RVM, that is trained on isolated letters, and

the edge potentials,  $\psi_{st}(ys, yt)$ , are often taken to be a language bigram model. Later we will discuss how to train all the potentials jointly.

#### 19.6.2.2 Noun phrase chunking

One common NLP task is noun phrase chunking, which refers to the task of segmenting a sentence into its distinct noun phrases (NPs). This is a simple example of a technique known as shallow parsing.

In more detail, we tag each word in the sentence with B (meaning beginning of a new NP), I (meaning inside a NP), or O (meaning outside an NP). This is called BIO notation. For example, in the following sentence, the NPs are marked with brackets:

B I O O OB IO B I I

(British Airways) rose after announcing (its withdrawal)  
from (the UAI deal)

(We need the B symbol so that we can distinguish II, meaning two words within a single NP, from BB, meaning two separate NPs.)

A standard approach to this problem would first convert the string of words into a string of POS tags, and then convert the POS tags to a string of BIOS. However, such a pipeline method can propagate errors. A more robust approach is to build a joint probabilistic model of the form  $p(\text{NP1:T, POS1:T}|\text{words1:T})$ . One way to do this is to use the CRF in Figure 19.16. The connections between adjacent labels encode the probability of transitioning between the B, I and O states, and can enforce constraints such as the fact that B must precede I. The features are usually hand engineered and include things like: does this word begin with a capital letter, is this word followed by a full stop, is this word a noun, etc. Typically there are  $\sim 1,000\text{--}10,000$  features per node.

The number of features has minimal impact on the inference time, since the features are observed and do not need to be summed over. (There is a small increase in the cost of evaluating potential functions with many features, but this is usually negligible; if not, one can use  $\ell_1$  regularization to prune out irrelevant features.) However, the graph structure can have a dramatic effect on inference time. The model in Figure 19.16 is tractable, since it is essentially a “fat chain”, so we can use the forwards-backwards algorithm (Section 17.4.3) for exact

inference in  $O(T|POS|^2|NP|^2)$  time, where  $|POS|$  is the number of POS tags, and  $|NP|$  is the number of NP tags. However, the seemingly similar graph in Figure 19.17, to be explained below, is computationally intractable.

#### 19.6.2.3 Named entity recognition

A task that is related to NP chunking is named entity extraction. Instead of just segmenting out noun phrases, we can segment out phrases to do with people and locations. Similar techniques are used to automatically populate your calendar from your email messages; this is called information extraction.

A simple approach to this is to use a chain-structured CRF, but to expand the state space from BIO to B-Per, I-Per, B-Loc, I-Loc, and Other. However, sometimes it is ambiguous whether a word is a person, location, or something else. (Proper nouns are particularly difficult to deal with because they belong to an open class, that is, there is an unbounded number of possible names, unlike the set of nouns and verbs, which is large but essentially fixed.) We can get better performance by considering long-range correlations between words. For example, we might add a link between all occurrences of the same word, and force the word to have the same tag in each occurrence. (The same technique can also be helpful for resolving the identity of pronouns.) This is known as a skip-chain CRF. See Figure 19.17 for an illustration.

We see that the graph structure itself changes depending on the input, which is an additional advantage of CRFs over generative models. Unfortunately, inference in this model is generally more expensive than in a simple chain with local connections, for reasons explained in Section 20.5.

#### 19.6.2.4 Natural language parsing

A generalization of chain-structured models for language is to use probabilistic grammars. In particular, a probabilistic context free grammar or PCFG is a set of rewrite or production rules of the form  $\sigma \rightarrow \sigma_1 \dots \sigma_n$  or  $\sigma \rightarrow x$ , where  $\sigma, \sigma_1, \dots, \sigma_n \in \Sigma$  are non-terminals (analogous to parts of speech), and  $x \in X$  are terminals, i.e., words. See Figure 19.18 for an example. Each such rule has an associated probability. The resulting model defines a probability distribution over sequences of words. We can compute the probability of observing a particular

sequence  $x = x_1 \dots x_T$  by summing over all trees that generate it. This can be done in  $O(T^3)$  time using the inside-outside algorithm; see e.g., (Jurafsky and Martin 2008; Manning and Schuetze 1999) for details.

PCFGs are generative models. It is possible to make discriminative versions which encode the probability of a labeled tree,  $y$ , given a sequence of words,  $x$ , by using a CRF of the form  $p(y|x) \propto \exp(wT\varphi(x, y))$ . For example, we might define  $\varphi(x, y)$  to count the number of times each production rule was used (which is analogous to the number of state transitions in a chain-structured model). See e.g., (Taskar et al. 2004) for details.

#### 19.6.2.5 Hierarchical classification

Suppose we are performing multi-class classification, where we have a label taxonomy, which groups the classes into a hierarchy. We can encode the position of  $y$  within this hierarchy by defining a binary vector  $\varphi(y)$ , where we turn on the bit for component  $y$  and for all its children. This can be combined with input features  $\varphi(x)$  using a tensor product,  $\varphi(x,y) = \varphi(x) \otimes \varphi(y)$ . See Figure 19.19 for an example.

This method is widely used for text classification, where manually constructed taxonomies (such as the Open Directory Project at [www.dmoz.org](http://www.dmoz.org)) are quite common. The benefit is that information can be shared between the parameters for nearby categories, enabling generalization across classes.

#### 19.6.2.6 Protein side-chain prediction

An interesting analog to the skip-chain model arises in the problem of predicting the structure of protein side chains. Each residue in the side chain has 4 dihedral angles, which are usually discretized into 3 values called rotamers. The goal is to predict this discrete sequence of angles,  $y$ , from the discrete sequence of amino acids,  $x$ .

We can define an energy function  $E(x, y)$ , where we include various pairwise interaction terms between nearby residues (elements of the  $y$  vector). This energy is usually defined as a weighted sum of individual energy terms,  $E(x, y|w) = \sum_{j=1}^J \theta_j E_j(x, y)$ , where the  $E_j$  are energy contribution due to various electrostatic charges, hydrogen bonding potentials, etc, and  $w$  are the parameters of the model. See (Yanover et al. 2007) for details.

Given the model, we can compute the most probable side

chain configuration using  $y^* = \operatorname{argmin}_y E(x, y|w)$ . In general, this problem is NP-hard, depending on the nature of the graph induced by the  $E_j$  terms, due to long-range connections between the variables. Nevertheless, some special cases can be efficiently handled, using methods discussed in Section 22.6.

#### 19.6.2.7 Stereo vision

Low-level vision problems are problems where the input is an image (or set of images), and the output is a processed version of the image. In such cases, it is common to use 2d latticestructured models; the models are similar to Figure 19.9, except that the features can be global, and are not generated by the model. We will assume a pairwise CRF.

A classic low-level vision problem is dense stereo reconstruction, where the goal is to estimate the depth of every pixel given two images taken from slightly different angles. In this section (based on (Sudderth and Freeman 2008)), we give a sketch of how a simple CRF can be used to solve this task. See e.g., (Sun et al. 2003) for a more sophisticated model.

By using some standard preprocessing techniques, one can convert depth estimation into a problem of estimating the disparity  $y_s$  between the pixel at location  $(is, js)$  in the left image and the corresponding pixel at location  $(is + ys, js)$  in the right image. We typically assume that corresponding pixels have similar intensity, so we define a local node potential of the form

$$\psi_s(y_s|x) \propto \exp - \frac{1}{2\sigma^2} (x_L(is, js) - x_R(is + ys, js))^2 \quad (19.68)$$

where  $x_L$  is the left image and  $x_R$  is the right image. This equation can be generalized to model the intensity of small windows around each location. In highly textured regions, it is usually possible to find the corresponding patch using cross correlation, but in regions of low texture, there will be considerable ambiguity about the correct value of  $y_s$ .

We can easily add a Gaussian prior on the edges of the MRF that encodes the assumption that neighboring disparities  $y_s, y_t$  should be similar, as follows:

$$\psi_{st}(y_s, y_t) \propto \exp - \frac{1}{2\gamma^2} (y_s - y_t)^2 \quad (19.69)$$

The resulting model is a Gaussian CRF.

However, using Gaussian edge-potentials will oversmooth the estimate, since this prior fails to account for the

occasional large changes in disparity that occur between neighboring pixels which are on different sides of an occlusion boundary. One gets much better results using a truncated Gaussian potential of the form

$$\psi_{st}(ys, yt) \propto \exp - 1/2\gamma^2 \min(ys - yt)^2, \delta_0 \geq 0 \quad (19.70)$$

where  $\gamma$  encodes the expected smoothness, and  $\delta_0$  encodes the maximum penalty that will be imposed if disparities are significantly different. This is called a discontinuity preserving potential; note that such penalties are not convex. The local evidence potential can be made robust in a similar way, in order to handle outliers due to specularities, occlusions, etc.

Figure 19.20 illustrates the difference between these two forms of prior. On the top left is an image from the standard Middlebury stereo benchmark dataset (Scharstein and Szeliski 2002). On the bottom left is the corresponding true disparity values. The remaining columns represent the estimated disparity after 0, 1 and an “infinite” number of rounds of loopy belief propagation (see Section 22.2), where by “infinite” we mean the results at convergence. The top row shows the results using a Gaussian edge potential, and the bottom row shows the results using the truncated potential. The latter is clearly better.

Unfortunately, performing inference with real-valued variables is computationally difficult, unless the model is jointly Gaussian. Consequently, it is common to discretize the variables. (For example, Figure 19.20(bottom) used 50 states.) The edge potentials still have the form given in Equation 19.69. The resulting model is called a metric CRF, since the potentials form a metric. Inference in metric CRFs is more efficient than in CRFs where the discrete labels have no natural ordering, as we explain in Section 22.6.3.3. See Section 22.6.4 for a comparison of various approximate inference methods applied to low-level CRFs, and see (Blake et al. 2011; Prince 2012) for more details on probabilistic models for computer vision.

### 19.6.3 CRF training

We can modify the gradient based optimization of MRFs described in Section 19.5.1 to the CRF case in a straightforward way. In particular, the scaled log-likelihood becomes

$$l(w) = \frac{1}{N} \sum_i \log p(y_i|x_i, w) = \frac{1}{N} \sum_i c_w^T c \varphi_c(y_i, x_i) - \log Z(w),$$

$x_i$ ) (19.71)

and the gradient becomes

$$\partial \partial w_c = 1 N i \varphi_c(y_i, x_i) - \partial \partial w_c \log Z(w, x_i) \quad (19.72)$$

$$= 1 N i [\varphi_c(y_i, x_i) - E[\varphi_c(y, x_i)]] \quad (19.73)$$

Note that we now have to perform inference for every single training case inside each gradient step, which is  $O(N)$  times slower than the MRF case. This is because the partition function depends on the inputs  $x_i$ .

In most applications of CRFs (and some applications of MRFs), the size of the graph structure can vary. Hence we need to use parameter tying to ensure we can define a distribution of arbitrary size. In the pairwise case, we can write the model as follows:

$$p(y|x,w) = 1 Z(w, x) \exp w^T \varphi(y, x) \quad (19.74)$$

where  $w = [w_n, w_e]$  are the node and edge parameters, and  $\varphi(y, x) = [t \varphi_t(y_t, x), s \sim t \varphi_{st}(y_s, y_t, x)]$  (19.75)

are the summed node and edge features (these are the sufficient statistics). The gradient expression is easily modified to handle this case. In practice, it is important to use a prior/ regularization to prevent overfitting. If we use a Gaussian prior, the new objective becomes

$$l(w) = 1 N i \log p(y_i|x_i, w) - \lambda \|w\|_2^2 \quad (19.76)$$

It is simple to modify the gradient expression.

Alternatively, we can use  $l_1$  regularization. For example, we could use  $l_1$  for the edge weights  $w_e$  to learn a sparse graph structure, and  $l_2$  for the node weights  $w_n$ , as in (Schmidt et al. 2008). In other words, the objective becomes

$$l(w) = 1 N i \log p(y_i|x_i, w) - \lambda_1 \|w_e\|_1 - \lambda_2 \|w_n\|_2^2 \quad (19.77)$$

Unfortunately, the optimization algorithms are more complicated when we use  $l_1$  (see Section 13.4), although the problem is still convex. To handle large datasets, we can use stochastic gradient descent (SGD), as described in Section 8.5.2. It is possible (and useful) to define CRFs with hidden variables, for example to allow for an unknown alignment between the visible features and the hidden labels (see e.g., (Schnitzspan et al. 2010)). In this case, the objective function is no longer convex. Nevertheless, we can find a locally optimal ML or MAP parameter estimate using EM and/ or gradient methods.

下面先逐段完整翻译（含图注与所有公式编号），再给出通俗解释与要点总结。我会把式子原样保留，便于对照学习。

---

## 图注翻译

**图 19.14** 序列数据的多种模型。(a) 生成式的有向 HMM。(b) 判别式的有向 MEMM。(c) 判别式的无向 CRF。

**图 19.15** 手写字母识别示例。在单词 “brace” 中，字母 r 和 c 在局部非常相似，但利用上下文可以消歧。

**图 19.16** 一个用于同时做词性标注 (POS) 和名词短语 (NP) 分块的 CRF。

**图 19.17** 用于命名实体识别的跳链 (skip-chain) CRF。

**图 19.18** 依据乔姆斯基基形 (CNF) 的一个简单句法树示意。特征向量  $\phi(x, y) = \Psi(x, y)$  统计了每条产生式被使用的次数。

**图 19.19** 一个简单的标签层次 (taxonomy) 示意，以及如何用它给类别 2 的标签计算一个分布式表示。图中， $\phi(x) = x$ ， $\phi(y = 2) = \Lambda(2)$ ， $\phi(x, y)$  记作  $\Psi(x, 2)$ ，而  $w^\top \phi(x, y)$  记作  $\langle w, \Psi(x, 2) \rangle$ 。

**图 19.20** 用置信传播做立体匹配 (深度估计) 的示意。左列：原图与真实视差。其余列：初始估计、迭代 1 次后的估计、以及收敛时 (“ $\infty$ ”) 的估计。上行：高斯边势；下行：\*\*鲁棒 (截断) \*\*边势。

---

## 19.6 条件随机场 (CRFs)

### 定义与动机

条件随机场 (CRF; Lafferty 等, 2001)，有时也称判别式随机场 (Kumar 与 Hebert, 2003)，就是一种**所有团势都条件化在输入特征上的 MRF 版本**：

$$p(y \mid x, w) = \frac{1}{Z(x, w)} \prod_c \psi_c(y_c \mid x, w) \quad (19.63)$$

CRF 可以看作逻辑回归在结构化输出上的推广。我们通常把势函数写成对数线性形式：

$$\psi_c(y_c \mid x, w) = \exp(w_c^\top \phi(x, y_c)) \quad (19.64)$$

其中  $\phi(x, y_c)$  是由全局输入  $x$  与局部标签集合  $y_c$  提取出的特征向量。下面会给出例子帮助理解。

相较于 MRF，CRF 的优势类似于判别式分类器相较于生成式分类器（见 §8.6）：**不必“浪费精力”去建模那些总是可见的东西**，而是把注意力集中在我们关心的目标——**给定数据的**

标签分布  $p(y \mid x)$ 。

CRF 的另一个重要优势是：**势（因子）可以依赖数据**。例如在图像处理中，若相邻像素  $s, t$  的灰度出现明显跳变，我们可以**关闭**这条边的“平滑”作用；在 NLP 中，隐含标签可以依赖整句的全局属性（如语言种类）。把这种全局特征纳入**生成式模型**通常很困难。

CRF 的劣势：需要**带标签的训练数据**，并且训练更慢（见 §19.6.3）。这与**逻辑回归 vs 朴素贝叶斯**的优劣势非常相似（§8.6）。

---

### 19.6.1 链式 CRF、MEMM 与标签偏置问题

使用**链式图**去描述相邻标签的相关性是最常见的 CRF 形式，适用于多种序列标注任务（§19.6.2）。

传统上，这类任务常用 HMM（第 17 章）：

$$p(x, y \mid w) = \prod_{t=1}^T p(y_t \mid y_{t-1}, w) p(x_t \mid y_t, w) \quad (19.65)$$

（为简洁起见省略初始项  $p(y_1)$ ；见图 19.14(a)）。若每个时刻都观测到  $x_t, y_t$ ，训练非常容易（§17.5.1）。

HMM 要求我们给出**生成式观测模型**  $p(x_t \mid y_t, w)$ ，这往往不易；同时它要求  $x_t$  **局部化**，因为要为整个观测序列  $x = x_{1:T}$  指定生成式模型很困难。

一个显而易见的判别式改造是把箭头从  $y_t \rightarrow x_t$  **反过来**（图 19.14(b)），得到

$$p(y \mid x, w) = \prod_t p(y_t \mid y_{t-1}, x, w) \quad (19.66)$$

其中  $x = (x_{1:T}, x_g)$ ， $x_g$  是**全局特征**， $x_t$  是结点  $t$  的局部特征（这种“局部/全局”的划分不是必须，只是便于与 HMM 比较）。这叫**最大熵马尔可夫模型**（MEMM；McCallum 等，2000；Kakade 等，2002）。

MEMM 就是**状态转移概率受输入特征调制的马尔可夫链**（因此是 §17.6.3 的输入-输出 HMM 的特例）。它看似是把逻辑回归自然推广到结构化情形，但有一个微妙的缺陷——**标签偏置（label bias）**（Lafferty 等，2001）：**时刻 t 的局部特征不会影响 t 之前的状态**。从 DAG 可见， $x_t$  与  $y_{t-1}$ （以及更早时刻）被  $y_t$  处的 **v 结构** d-分离了，信息被“堵住”。

实际含义：做 POS 标注时，单词 “banks” 可能是动词也可能是名词。后文若出现 “fishing”，就暗示 “banks” 指“河岸”。在 MEMM 中（不同于 HMM 与 CRF），这个后文证

据不能向后传，导致无法消歧。

而链式 CRF 为

$$p(y \mid x, w) = \frac{1}{Z(x, w)} \left( \prod_{t=1}^T \psi(y_t \mid x, w) \right) \left( \prod_{t=1}^{T-1} \psi(y_t, y_{t+1} \mid x, w) \right) \quad (19.67)$$

从图 19.14(c) 可见，**标签偏置不再存在**： $y_t$  不会阻断来自  $x_t$  的信息向其它  $y$  传播。

MEMM 的偏置来自**局部归一化**（每个条件分布都要和为 1）。MRF/CRF 是**全局归一化**：局部因子不用各自归一，配分函数  $Z$  对所有联合配置求和，保证整体是合法分布。代价是：**要看到整句话才能得到合法的  $p(y \mid x)$** （因为只有那时才能做全局归一），因此 CRF 不如 DGM 适合**在线/实时推断**；并且  $Z$  依赖所有结点及其参数，使得 CRF 训练比 DGM 更慢（§19.6.3）。

---

## 19.6.2 CRF 的应用（代表性示例）

这些应用展示了若干建模技巧，也为第 20 章的推断方法提供动机。

### 19.6.2.1 手写识别

识别手写串（图 19.15）是 CRF 的天然应用：**单个字母可能含糊**，但利用**邻居标签**的上下文可以降错。常见做法：节点势  $\psi_i(y_t \mid x_t)$  用一个判别式分类器（NN/RVM）在**单字母**上训练；边势  $\psi_{st}(y_s, y_t)$  用**语言双元模型**。后文会讲如何**联合训练**所有势。

### 19.6.2.2 名词短语分块（NP chunking）

任务：把句子切成若干 NP。用 BIO 标注：B=开头，I=内部，O=外部。**用流水线**（word→POS→BIO）会**传播误差**；更稳健的是**联合建模**  $p(\text{NP}_{1:T}, \text{POS}_{1:T} \mid \text{words}_{1:T})$ 。图 19.16 的 CRF 就是做这个：相邻标签间的连接编码 B/I/O 的转移并可**施加约束**（如 B 必须在 I 之前）。特征多为手工设计（是否大写、是否句号、是否名词等），**每个结点通常有 1k–10k 个特征**。

**特征数量对推断影响小**（观测量不要求和）；真正决定推断复杂度的是**图结构**。图 19.16 属于“胖链”，可用前向-后向做**精确推断**，时间  $O(T|\text{POS}|^2|\text{NP}|^2)$ 。而看似相近的图 19.17（见下节）却不可 tractable。

### 19.6.2.3 命名实体识别（NER）

用链式 CRF，把状态拓展成 B-PER、I-PER、B-LOC、I-LOC、OTH 等即可。但很多词对“人/地/其它”**本就含糊**。更好办法是考虑**远距离相关**：例如把所有**相同词**连起来，强制各处的标签一致（对代词指代消解也有帮助）。这叫**跳链 CRF**（图 19.17）。其**图结构随输入改变**，这正是 CRF 相对生成式模型的又一优势；但推断比纯链式**更贵**（§20.5 解释原因）。

### 19.6.2.4 自然语言句法解析

链式模型的推广是**概率文法**。PCFG 用产生式  $\sigma \rightarrow \sigma' \sigma''$  或  $\sigma \rightarrow x$  ( $\sigma$  为非终结符,  $x$  为词) 及其概率来定义对词序列的分布 (图 19.18)。序列  $x$  的概率可对所有生成它的树求和, 用 inside-outside 算法  $O(T^3)$  完成。

也可以做**判别式版本**: 用 CRF 建模  $p(y | x) \propto \exp(w^\top \phi(x, y))$ , 其中  $\phi$  统计各产生式使用次数 (类似链上“状态转移次数”)。见 Taskar 等 (2004)。

### 19.6.2.5 分层分类 (Hierarchical classification)

若类别有**层次结构** (taxonomy), 可用二进制向量  $\phi(y)$  表示  $y$  在层次中的位置: 把  $y$  及其所有**祖先/路径**位置置 1。与输入特征  $\phi(x)$  用张量积合成  $\phi(x, y) = \phi(x) \otimes \phi(y)$  (图 19.19)。这在**文本分类**中常见 (如 DMOZ 目录), 好处是**相近类别共享参数**, 提升跨类泛化。

### 19.6.2.6 蛋白质侧链预测

与“跳链”相似的问题: 要从氨基酸序列  $x$  预测离散的**二面角**序列  $y$  (每个残基 4 个角, 通常离散成 3 个“转子”)。定义能量

$$E(x, y | w) = \sum_{j=1}^D \theta_j E_j(x, y)$$

各  $E_j$  是静电、氢键等贡献项。求最可能构型等价于

$$y^{*} = \arg \min_y E(x, y | w).$$

一般是 NP 难 (因为存在长程相互作用), 但有一些**特殊情形**可高效求解 (§22.6)。

### 19.6.2.7 立体视觉 (Stereo)

低层视觉中常用**2D 格**的成对 CRF。目标是从一对视图估计每个像素的**视差**  $y_s$ 。假设对应像素强度相近, 节点势:

$$\psi_s(y_s | x) \propto \exp\left[-\frac{1}{2\sigma^2}(x_L(i_s, j_s) - x_R(i_s + y_s, j_s))^2\right] \quad (19.68)$$

边势可用高斯平滑先验:

$$\psi_{st}(y_s, y_t) \propto \exp\left[-\frac{1}{2\gamma^2}(y_s - y_t)^2\right] \quad (19.69)$$

这得到**高斯 CRF**。但纯高斯会过度平滑, 无法保留遮挡边界的大跳变。更好的是**截断高斯** (鲁棒) **\*\*势**:

$$\psi_{st}(y_s, y_t) \propto \exp\left[-\frac{1}{2\gamma^2} \min\{(y_s - y_t)^2, \delta_0^2\}\right] \quad (19.70)$$

$\gamma$ : 预期平滑度;  $\delta_0$ : 当差异很大时的**最大惩罚**。这是**保边缘的**（非凸）势，本地证据也可做类似鲁棒化。图 19.20 展示：鲁棒势明显更好。

用实数变量做推断很难（除非整体高斯），故常把变量**离散化**（图 19.20 下图用了 50 个状态），边势仍按式 (19.69) 近似。此模型称**metric CRF**（势满足度量性质）；推断比无序标签的 CRF 更高效（§22.6.3.3）。更多近似推断比较见 §22.6.4；视觉概率模型综述参见 Blake 等 (2011)、Prince (2012)。

---

### 19.6.3 CRF 的训练

把 §19.5.1 的 MRF 梯度法**直接改到** CRF 即可。缩放对数似然：

$$\ell(w) = \frac{1}{N} \sum_i \log p(y_i | x_i, w) = \frac{1}{N} \sum_i \left( \sum_c w_c^\top \phi_c(y_i, x_i) - \log Z(w, x_i) \right) \quad (19.71)$$

梯度：

$$\begin{aligned} \frac{\partial \ell}{\partial w_c} &= \frac{1}{N} \sum_i \phi_c(y_i, x_i) - \frac{1}{Z(w, x_i)} \mathbb{E}[\phi_c(y_i, x_i)] \end{aligned} \quad (19.72)$$

$$\mathbb{E}[\phi_c(y_i, x_i)] \quad (19.73)$$

注意：**每个梯度步都要对每个训练样本** 做一次推断（因为  $Z$  依赖  $x_i$ ），因此比 MRF 慢  $O(N)$  倍。

许多 CRF（也有些 MRF）**图大小会变**，需用**参数共享**来应对任意规模。成对情形下写成

$$p(y | x, w) = \frac{1}{Z(w, x)} \exp(w^\top \phi(y, x)) \quad (19.74)$$

其中  $w = [w_n, w_e]$  为**点/边参数**，

$$\phi(y, x) = [\sum_t \phi_t(y_t, x), \sum_{s \sim t} \phi_{st}(y_s, y_t, x)] \quad (19.75)$$

是充分统计量。训练时要**正则化**防止过拟合。若用高斯先验，目标变为

$$\ell(w) = \frac{1}{N} \sum_i \log p(y_i | x_i, w) - \lambda \|w\|_2^2 \quad (19.76)$$

也可用  $l_1$  正则。例如对边权用  $l_1$ （学稀疏图），对点权用  $l_2$  (Schmidt 等, 2008)：

$$\ell(w) = \frac{1}{N} \sum_i \log p(y_i | x_i, w) - \lambda_1 \|w_e\|_1 - \lambda_2 \|w_n\|_2^2 \quad (19.77)$$

用了  $l_1$  后优化更复杂 (§13.4)，但仍是凸问题。大数据可用 SGD (§8.5.2)。

CRF 也可以含隐变量 (例如未知的对齐；Schnitzspan 等, 2010)，这时目标不再凸，但可用 EM/梯度求到局部最优的 ML/MAP 估计。

---

## 通俗解释 (把难点说白)

- **CRF 是啥？**

你可以把它理解为“逻辑回归 + 图结构”：每个结点/边上的打分函数  $\psi = \exp(w^\top \phi)$ ，全局概率靠一个配分函数统一归一化。

- **为啥 CRF 比 HMM/MEMM 更香？**

- 不需要写“ $x$  怎么生成”的模型，直接学  $y | x$ 。
- 特征可以是全局的、数据依赖的（比如图像边缘处关闭平滑）。
- 解决了 MEMM 的标签偏置：后面的证据能往回传。

- **代价是什么？**

- **训练慢**：每步要对每个样本做推断，因为  $Z$  依赖样本的  $x$ 。
- **在线应用不方便**：得等到看到整句才能全局归一。

- **在哪些任务好用？**

- **手写/字符串**：上下文消歧。
- **NLP**：NP 分块、NER（跳链可连远距离相同词）、句法解析（把产生式计数当特征）。
- **层次分类**：让相近类共享参数。
- **蛋白质结构**：复杂相互作用的能量最小化。
- **立体视觉**：节点是匹配代价，边是平滑先验；截断惩罚能保边缘，效果优于纯高斯。

- **训练怎么做？**

梯度 = 【数据特征】 - 【模型特征】（要做推断）；参数共享要靠把点/边特征求和形成充分统计量；用  $l_2/l_1$  正则；大数据用 SGD；有隐变量就用 EM+梯度。

---

## 要点总结 (记住这几条)

1. CRF = 条件化的 MRF： $\psi_c(y_c | x) = \exp(w_c^\top \phi(x, y_c))$ ，全局归一靠  $Z(x, w)$ 。

## 2. 优于 MEMM/HMM 的两点：

- 聚焦  $p(y | x)$ , 可用任意 (甚至全局) 特征;
- 无标签偏置, 信息可在图上双向流动。

3. 代价：训练慢（每样本要推断），不适合在线实时归一。

4. 应用面广：字符识别、NP/NER、解析、层次分类、蛋白质侧链、立体匹配等。

5. 训练公式： $\ell(w) = \frac{1}{N} \sum_i [w^\top \phi(y_i, x_i) - \log Z(w, x_i)],$

$\nabla \ell = \frac{1}{N} \sum_i [\phi(y_i, x_i) - E(\phi(y, x_i))]$ ; 常配合参数共享与正则化。

## 6. 建模技巧：

- 图像/视觉中用鲁棒（截断）边势保边界；
- 文本中用跳链连接远距离一致性；
- 解析用产生式计数作特征；
- 层次分类用  $\phi(x) \otimes \phi(y)$  共享信息。

如果你想，我可以选一个任务（比如链式 CRF 做 NER 或 立体匹配的鲁棒势），给你画出具体的特征函数  $\phi$  和一步步的推断/训练流程小例子。

这一节主要在讲什么？我不太懂

已思考 11s

下面是“19.6 条件随机场 (CRFs) ”这节在讲的核心，用不带公式的方式梳理：

## 这节到底讲啥？

一句话：CRF 是把“逻辑回归”升级到结构化输出（整条序列、整张图像），用图上的平滑/一致性约束把各个局部分类器拧成一个整体来预测  $y | x$ 。

## 为什么要用 CRF？

- 只建模我们关心的东西：直接学  $p(y | x)$ ，不用像 HMM 那样还得写“ $x$  是怎么生成的”。
- 能用任意特征：局部的、全局的、数据相关的都能放进来（比如图像边缘处“关闭”平滑）。
- 解决 MEMM 的“标签偏置”：后面出现的证据可以向前影响前面的标签，不会被有向模型的局部归一化“卡住”。

## 它长啥样、怎么用？

- 最常见是**链式 CRF**：做序列标注（词性、分块、命名实体、手写字符）——每个位置一个“局部分类器”，相邻位置用“边”鼓励前后一致/合法转移。
- 还能做：
  - 跳链 CRF**：把句子里同一个词的多个位置连起来，利用远距离一致性（做 NER 很有用）。
  - 文法/解析**：把“使用了哪些产生式”当特征，做判别式解析。
  - 层次分类**：用类别树的编码当标签特征，让近邻类别共享信息。
  - 视觉**：2D 网格上的 CRF 做立体匹配/分割；边势用**截断惩罚**可保留物体边界。

**直觉**：把每个结点看成一个“逻辑回归打分器”，边上再加“相邻要相似/满足语法”的约束，然后全局一起归一化，输出整条序列/整幅图的联合预测。

---

## 代价与训练

- 训练慢**：每一步都要做一次“全局推断”（因为归一化常数依赖整体输入），比有向模型慢不少。
  - 不是在线友好**：通常要看到**整句/整图**才能给出合法概率。
  - 实战里用**参数共享**（点参数、边参数在各位置复用），并配**L2/L1 正则**；大数据用 SGD。
  - 若引入**隐变量/对齐**，目标不再凸，但可以用 EM + 梯度找到可用解。
- 

## 什么时候该想起 CRF？

- 结果是一串**标签**或一张**标签图**，而且**上下文/邻域信息**很重要；
  - 你手里有丰富的、甚至全局的**输入特征**；
  - 不要求严格的**在线实时概率**；
  - 例子：词性/分块/命名实体、手写串识别、图像分割与立体深度、层次化多分类、蛋白质侧链等。
- 

## 要点小结

- CRF = 条件化的 MRF = 结构化的逻辑回归。**
- 相比 HMM/MEMM：更能用特征、无标签偏置，但**训练更慢**。

- 典型形状：链式（可加跳链）与2D网格；视觉里常用鲁棒（截断）边势保边界。
- 训练梯度本质是“数据特征 – 模型特征”，因而必须反复做推断；配合参数共享与正则化。

如果你想，我可以挑一个具体任务（比如“链式 CRF 做命名实体”或“立体匹配的截断边势”）画出特征设计与推断/训练的步骤小抄。

翻译并解释以下内容，翻译不能有遗漏，解释请通俗易懂且详尽，因为我的数学基础不太好，最后总结要点。

## 19.7 Structural SVMs

We have seen that training a CRF requires inference, in order to compute the expected sufficient statistics needed to evaluate the gradient. For certain models, computing a joint MAP estimate of the states is provably simpler than computing marginals, as we discuss in Section 22.6. In this section, we discuss a way to train structured output classifiers that leverage the existence of fast MAP solvers. (To avoid confusion with MAP estimation of parameters, we will often refer to MAP estimation of states as decoding.) These methods are known as structural support vector machines or SSVMs (Tsochantaridis et al. 2005). (There is also a very similar class of methods known as max margin Markov networks or M3nets (Taskar et al. 2003); see Section 19.7.2 for a discussion of the differences.)

### 19.7.1 SSVMs: a probabilistic view

In this book, we have mostly concentrated on fitting models using MAP parameter estimation, i.e., by minimizing functions of the form

$$\text{RMAP}(w) = -\log p(w) - N \sum_{i=1}^N \log p(y_i|x_i, w) \quad (19.78)$$

However, at test time, we pick the label so as to minimize the posterior expected loss (defined in Section 5.7):

$$\hat{y}(x|w) = \operatorname{argmin}_{\hat{y}} \hat{y} L(\hat{y}, y)p(y|x, w) \quad (19.79)$$

where  $L(y^*, \hat{y})$  is the loss we incur when we estimate  $\hat{y}$  but the truth is  $y^*$ . It therefore seems reasonable to take the loss function into account when performing parameter estimation.<sup>10</sup> So, following (Yuille and He 2011), let us instead minimize the posterior expected

loss on the training set:

$$REL(w) = -\log p(w) + \sum_{i=1}^N \log y_i L(y_i, y_i) p(y_i | x_i, w) \quad (19.80)$$

In the special case of 0-1 loss,  $L(y_i, y_i) = 1 - \delta_{y_i, y_i}$ , this reduces to RMAP. We will assume that we can write our model in the following form:

$$p(y|x, w) = \exp(w^T \varphi(x, y)) Z(x, w) \quad (19.81)$$

$$p(w) = \exp(-E(w)) Z \quad (19.82)$$

where  $Z(x, w) = \sum_y \exp(w^T \varphi(x, y))$ . Also, let us define  $L(y_i, y) = \exp(-L(y_i, y))$ . With this, we can rewrite our objective as follows:

$$REL(w) = -\log p(w) + \sum_i \log y_i \exp(-L(y_i, y_i)) \exp(w^T \varphi(x_i, y_i)) Z(x, w) \quad (19.83)$$

$$= E(w) + \sum_i -\log Z(x_i, w) + \log y_i \exp(-L(y_i, y_i)) + w^T \varphi(x_i, y_i) \# \quad (19.84)$$

We will now consider various bounds in order to simplify this objective. First note that for any function  $f(y)$  we have

$$\max_{y \in Y} f(y) \leq \log \sum_{y \in Y} \exp[f(y)] \leq \log |Y| \exp \max_{y \in Y} f(y)$$

$$= \log |Y| + \max_{y \in Y} f(y) \quad (19.85)$$

For example, suppose  $Y = \{0, 1, 2\}$  and  $f(y) = y$ . Then we have

$$2 = \log[\exp(2)] \leq \log[\exp(0) + \exp(1) + \exp(2)] \leq \log[3 \times \exp(2)] = \log(3) + 2 \quad (19.86)$$

We can ignore the  $\log |Y|$  term, which is independent of  $y$ , and treat  $\max_{y \in Y} f(y)$  as both a lower and upper bound.

Hence we see that

$$REL(w) \approx E(w) + \sum_i \max_{y \in Y} y & L(y_i, y_i) + w^T \varphi(x_i, y_i) - \max_{y \in Y} y w^T \varphi(x_i, y_i) \quad (19.87)$$

where  $x \sim y$  means  $c_1 + x \leq y + c_2$  for some constants  $c_1, c_2$ . Unfortunately, this objective is not convex in  $w$ .

However, we can devise a convex upper bound by exploiting the following looser lower bound on the log-sum-exp function:

$$f(y) \leq \log \sum_{y' \in Y} \exp[f(y')] \quad (19.88)$$

for any  $y \in Y$ . Applying this equation to our earlier example, for  $f(y) = y$  and  $Y = \{0, 1, 2\}$ , we get  $1 \leq \log[\exp(1)] \leq \log[\exp(0) + \exp(1) + \exp(2)]$ . And applying this bound to  $REL$  we get

$$REL(w) \leq E(w) + \sum_i \max_{y \in Y} y & L(y_i, y_i) + w^T \varphi(x_i, y_i) - w^T \varphi(x_i, y_i) \quad (19.89)$$

If we set  $E(w) = -\frac{1}{2} C \|w\|^2$  (corresponding to a spherical Gaussian prior), we get

$$RSSVM(w) = \frac{1}{2} \|w\|^2 + C \sum_i \max_{y \in Y} y & L(y_i, y_i) + w^T \varphi(x_i, y_i) - w^T \varphi(x_i, y_i) \quad (19.90)$$

This is the same objective as used in the SSVM approach of (Tsochantaridis et al. 2005).

In the special case that  $Y = \{-1, +1\}$   $L(y^*, y) = 1 - \delta y, y^*$ , and  $\varphi(x, y) = 1/2 yx$ , this criterion reduces to the following (by considering the two cases that  $y = y_i$  and  $y = -y_i$ ):

$RSVM(w) = 1/2 \|w\|^2 + C \sum_{i=1}^N \max\{0, 1 - y_i w^T x_i\}$  (19.91)  
which is the standard binary SVM objective (see Equation 14.57).

So we see that the SSVM criterion can be seen as optimizing an upper bound on the Bayesian objective, a result first shown in (Yuille and He 2011). This bound will be tight (and hence the approximation will be a good one) when  $\|w\|$  is large, since in that case,  $p(y|x, w)$  will concentrate its mass on  $\text{argmax}_y p(y|x, w)$ . Unfortunately, a large  $\|w\|$  corresponds to a model that is likely to overfit, so it is unlikely that we will be working in this regime (because we will tune the strength of the regularizer to avoid this situation). An alternative justification for the SVM criterion is that it focusses effort on fitting parameters that affect the decision boundary. This is a better use of computational resources than fitting the full distribution, especially when the model is wrong.

### 19.7.2 SSVMs: a non-probabilistic view

We now present SSVMs in a more traditional (non-probabilistic) way, following (Tsochantaridis et al. 2005). The resulting objective will be the same as the one above. However, this derivation will set the stage for the algorithms we discuss below. Let  $f(x; w) = \text{argmax}_{y \in Y} w^T \varphi(x, y)$  be the prediction function. We can obtain zero loss on the training set using this predictor if

$$\forall i. \max_{y \in Y \setminus y_i} w^T \varphi(x_i, y) \leq w^T \varphi(x_i, y_i) \quad (19.92)$$

Each one of these nonlinear inequalities can be equivalently replaced by  $|Y| - 1$  linear inequalities, resulting in a total of  $N|Y| - N$  linear constraints of the following form:

$$\forall i. \forall y \in Y \setminus y_i. w^T \varphi(x_i, y) - w^T \varphi(x_i, y_i) \geq 0 \quad (19.93)$$

For brevity, we introduce the notation

$$\delta_i(y) = \varphi(x_i, y_i) - \varphi(x_i, y) \quad (19.94)$$

so we can rewrite these constraints as  $w^T \delta_i(y) \geq 0$ . If we can achieve zero loss, there will typically be multiple solution vectors  $w$ . We pick the one that maximizes the margin, defined as

$$\gamma = \min_i f(x_i, y_i; w) - \max_{y \in Y \setminus y_i} f(x_i, y; w) \quad (19.95)$$

Since the margin can be made arbitrarily large by rescaling  $w$ , we fix its norm to be 1, resulting in the optimization problem

$$\max_{\gamma, w} \gamma, w : \|w\|=1 \text{ s.t. } \forall i. \forall y \in Y \setminus y_i. w^T \delta_i(y) \geq \gamma \quad (19.96)$$

Equivalently, we can write

$$\min w \ 1/2 \|w\|^2 \text{ s.t. } \forall i. \forall y \in Y \setminus y_i. w^T \delta_i(y) \geq 1 \quad (19.97)$$

To allow for the case where zero loss cannot be achieved (equivalent to the data being inseparable in the case of binary classification), we relax the constraints by introducing slack terms  $\xi_i$ , one per data case. This yields

$$\min w, \xi \ 1/2 \|w\|^2 + C \sum_{i=1}^N \xi_i \text{ s.t. } \forall i. \forall y \in Y \setminus y_i. w^T \delta_i(y) \geq 1 - \xi_i, \xi_i \geq 0 \quad (19.98)$$

In the case of structured outputs, we don't want to treat all constraint violations equally. For example, in a segmentation problem, getting one position wrong should be punished less than getting many positions wrong. One way to achieve this is to divide the slack variable by the size of the loss (this is called slack re-scaling). This yields

$$\min w, \xi \ 1/2 \|w\|^2 + C \sum_{i=1}^N \xi_i \text{ s.t. } \forall i. \forall y \in Y \setminus y_i. w^T \delta_i(y) \geq 1 - \xi_i / L(y_i, y), \xi_i \geq 0 \quad (19.99)$$

Alternatively, we can define the margin to be proportional to the loss (this is called margin re-rescaling). This yields

$$\min w, \xi \ 1/2 \|w\|^2 + C \sum_{i=1}^N \xi_i \text{ s.t. } \forall i. \forall y \in Y \setminus y_i. w^T \delta_i(y) \geq L(y_i, y) - \xi_i, \xi_i \geq 0 \quad (19.100)$$

(In fact, we can write  $\forall y \in Y$  instead of  $\forall y \in Y \setminus y_i$ , since if  $y = y_i$ , then  $w^T \delta_i(y) = 0$  and  $\xi_i = 0$ . By using the simpler notation, which doesn't exclude  $y_i$ , we add an extra but redundant constraint.) This latter approach is used in M3nets.

For future reference, note that we can solve for the  $\xi_i$  terms as follows:

$$\xi_i = \max\{0, \max y (L(y_i, y) - w^T \delta_i)\} = \max y (L(y_i, y) - w^T \delta_i) \quad (19.101)$$

Substituting in, and dropping the constraints, we get the following equivalent problem:

$$\min w \ 1/2 \|w\|^2 + C \sum_i \max y (L(y_i, y) + w^T \varphi(x_i, y)) - w^T \varphi(x_i, y) \quad (19.102)$$

### 19.7.2.1 Empirical risk minimization

Let us pause and consider whether the above objective is reasonable. Recall that in the frequentist approach to machine learning (Section 6.5), the goal is to minimize the regularized empirical risk, defined by

$$R(w) + C \sum_{i=1}^N L(y_i, f(x_i, w)) \quad (19.103)$$

where  $R(w)$  is the regularizer, and  $f(x_i, w) = \arg\max_y w^T \varphi(x_i, y)$  is the prediction. Since this objective is hard to optimize, because the loss is not differentiable, we will construct a convex upper bound instead. We can show that

$$R(w) + C \sum_{i=1}^N \max_y L(y_i, y) - w^T \delta_i \quad (19.104)$$

is such a convex upper bound. To see this, note that

$$L(y_i, f(x_i, w)) \leq L(y_i, f(x_i, w)) - w^T \varphi(x_i, y_i) + w^T \varphi(x_i, \hat{y}_i) \quad (19.105)$$

$$\leq \max_y L(y_i, y) - w^T \varphi(x_i, y_i) + w^T \varphi(x_i, y) \quad (19.106)$$

Using this bound and  $R(w) = \frac{1}{2} \|w\|^2$  yields Equation 19.102.

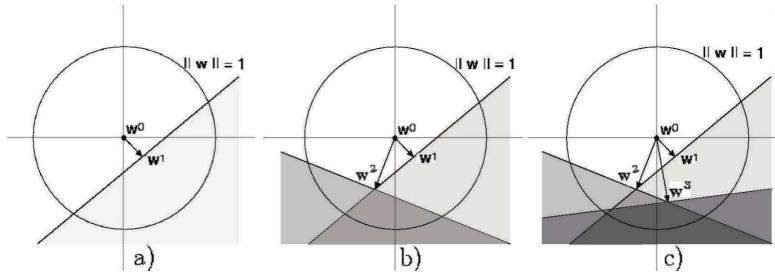
### 19.7.2.2 Computational issues

Although the above objectives are simple quadratic programs (QP), they have  $O(N|Y|)$  constraints. This is intractable, since  $Y$  is usually exponentially large. In the case of the margin rescaling formulation, it is possible to reduce the exponential number of constraints to a polynomial number, provided the loss function and the feature vector decompose according to a graphical model. This is the approach used in M3nets (Taskar et al. 2003). An alternative approach is to work directly with the exponentially sized QP. This allows for the use of more general loss functions. There are several possible methods to make this feasible. One is to use cutting plane methods. Another is to use stochastic subgradient methods. We discuss both of these below.

### 19.7.3 Cutting plane methods for fitting SSVMs

In this section, we discuss an efficient algorithm for fitting SSVMs due to (Joachims et al. 2009). This method can handle general loss functions, and is implemented in the popular SVMstruct package<sup>11</sup>. The method is based on the cutting plane method from convex optimization (Kelley 1960). The basic idea is as follows. We start with an initial guess  $w$  and no constraints. At each iteration, we then do the following: for each example  $i$ , we find the “most violated” constraint involving  $x_i$  and  $\hat{y}_i$ . If the loss-augmented margin violation exceeds the current value of  $\xi_i$  by more than  $\epsilon$ , we add  $\hat{y}_i$  to the working set of constraints for this training case,  $W_i$ , and then solve the resulting new QP to find the new  $w$ ,  $\xi$ . See Figure 19.21 for a sketch, and Algorithm 11 for the pseudo code. (Since at each step we only add one new constraint, we can warm-

start the QP solver.) We can easily modify the algorithm to optimize the slack rescaling version by replacing the expression  $L(y_i, \hat{y}) - w^T \delta_i(\hat{y}_i)$  with  $L(y_i, \hat{y})(1 - w^T \delta_i(\hat{y}_i))$ . The key to the efficiency of this method is that only polynomially many constraints need to be added, and as soon as they are, the exponential number of other constraints are guaranteed to also be satisfied to within a tolerance of (see (Tschantaridis et al. 2005) for the proof).



翻译并解释以下内容，翻译不能有遗漏，解释请通俗易懂且详尽，因为我的数学基础不太好，最后总结要点。

Figure 19.21 Illustration of the cutting plane algorithm in 2d. We start with the estimate  $w = w_0 = 0$ . (a) We add the first constraint; the shaded region is the new feasible set. The new minimum norm solution is  $w_1$ . (b) We add another constraint; the dark shaded region is the new feasible set. (c) We add a third constraint.

## 19.7 Structural SVMs

We have seen that training a CRF requires inference, in order to compute the expected sufficient statistics needed to evaluate the gradient. For certain models, computing a joint MAP estimate of the states is provably simpler than computing marginals, as we discuss in Section 22.6. In this section, we discuss a way to train structured output classifiers that leverages the existence of fast MAP solvers. (To avoid confusion with MAP estimation of parameters, we will often refer to MAP estimation of states as decoding.) These methods are known as structural support vector machines or SSVMs (Tschantaridis et al. 2005). (There is also a very similar class of methods known as max margin Markov networks

or M3nets (Taskar et al. 2003); see Section 19.7.2 for a discussion of the differences.)

### 19.7.1 SSVMs: a probabilistic view

In this book, we have mostly concentrated on fitting models using MAP parameter estimation, i.e., by minimizing functions of the form

$$RMAP(w) = -\log p(w) - N \sum_{i=1}^N \log p(y_i|x_i, w) \quad (19.78)$$

However, at test time, we pick the label so as to minimize the posterior expected loss (defined in Section 5.7):

$$\hat{y}(x|w) = \operatorname{argmin}_y \hat{y} L(\hat{y}, y) p(y|x, w) \quad (19.79)$$

where  $L(y^*, \hat{y})$  is the loss we incur when we estimate  $\hat{y}$  but the truth is  $y^*$ . It therefore seems reasonable to take the loss function into account when performing parameter estimation.<sup>10</sup> So, following (Yuille and He 2011), let us instead minimize the posterior expected loss on the training set:

$$REL(w) = -\log p(w) + N \sum_{i=1}^N \log y L(y_i, y) p(y|x_i, w) \quad (19.80)$$

In the special case of 0-1 loss,  $L(y_i, y) = 1 - \delta_{y_i, y}$ , this reduces to RMAP. We will assume that we can write our model in the following form:

$$p(y|x, w) = \exp(w^T \varphi(x, y)) Z(x, w) \quad (19.81)$$

$$p(w) = \exp(-E(w)) Z \quad (19.82)$$

where  $Z(x, w) = \int y \exp(w^T \varphi(x, y)) dy$ . Also, let us define  $L(y_i, y) = \exp(-L(y_i, y))$ . With this, we can rewrite our objective as follows:

$$REL(w) = -\log p(w) + \sum_i \log y \exp(-L(y_i, y)) \exp(w^T \varphi(x_i, y)) Z(x, w) \quad (19.83)$$

$$= E(w) + \sum_i -\log Z(x_i, w) + \log y \exp(-L(y_i, y)) + w^T \varphi(x_i, y) \# \quad (19.84)$$

We will now consider various bounds in order to simplify this objective. First note that for any function  $f(y)$  we have

$$\max_{y \in Y} f(y) \leq \log \sum_{y \in Y} \exp[f(y)] \leq \log |Y| \exp \max_{y \in Y} f(y) \\ = \log |Y| + \max_{y \in Y} f(y) \quad (19.85)$$

For example, suppose  $Y = \{0, 1, 2\}$  and  $f(y) = y$ . Then we have

$$2 = \log[\exp(2)] \leq \log[\exp(0) + \exp(1) + \exp(2)] \leq \log[3 \times \exp(2)] = \log(3) + 2 \quad (19.86)$$

We can ignore the  $\log |Y|$  term, which is independent of  $y$ , and treat  $\max_{y \in Y} f(y)$  as both a lower and upper bound.

Hence we see that

$$REL(w) \approx E(w) + N \sum_{i=1}^N \max_y f(y) & L(y_i, y) + w^T \varphi(x_i, y) - \max_y w^T \varphi(x_i, y) \quad (19.87)$$

where  $x \sim y$  means  $c_1 + x \leq y + c_2$  for some constants

c1,c2. Unfortunately, this objective is not convex in w.

However, we can devise a convex upper bound by exploiting the following looser lower bound on the log-sum-exp function:

$$f(y) \leq \log y \exp[f(y)] \quad (19.88)$$

for any  $y \in Y$ . Applying this equation to our earlier example, for  $f(y) = y$  and  $y = 1$ , we get  $1 = \log[\exp(1)] \leq \log[\exp(0) + \exp(1) + \exp(2)]$ . And applying this bound to REL we get

$$\text{REL}(w) \leq E(w) + N \sum_{i=1}^N \max\{y_i & L^*(y_i, y) + w^T \varphi(x_i, y) \\ - w^T \varphi(x_i, y_i)\} \quad (19.89)$$

If we set  $E(w) = -\frac{1}{2} C \|w\|^2$  (corresponding to a spherical Gaussian prior), we get

$$\text{RSSVM}(w) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^N \max\{y_i & L^*(y_i, y) + w^T \varphi(x_i, y) \\ - w^T \varphi(x_i, y_i)\} \quad (19.90)$$

This is the same objective as used in the SSVM approach of (Tsochantaridis et al. 2005). In the special case that  $Y = \{-1, +1\}$ ,  $L(y^*, y) = 1 - \delta_{y,y^*}$ , and  $\varphi(x, y) = 1/2 yx$ , this criterion reduces to the following (by considering the two cases that  $y = y_i$  and  $y = -y_i$ ):

$\text{RSVM}(w) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^N \max\{0, 1 - y_i w^T x_i\} \quad (19.91)$

which is the standard binary SVM objective (see Equation 14.57). So we see that the SSVM criterion can be seen as optimizing an upper bound on the Bayesian objective, a result first shown in (Yuille and He 2011). This bound will be tight (and hence the approximation will be a good one) when  $\|w\|$  is large, since in that case,  $p(y|x, w)$  will concentrate its mass on  $\arg\max_y p(y|x, w)$ . Unfortunately, a large  $\|w\|$  corresponds to a model that is likely to overfit, so it is unlikely that we will be working in this regime (because we will tune the strength of the regularizer to avoid this situation). An alternative justification for the SVM criterion is that it focusses effort on fitting parameters that affect the decision boundary. This is a better use of computational resources than fitting the full distribution, especially when the model is wrong.

### 19.7.2 SSVMs: a non-probabilistic view

We now present SSVMs in a more traditional (non-probabilistic) way, following (Tsochantaridis et al. 2005). The resulting objective will be the same as the one above. However, this derivation will set the stage for the algorithms we discuss below. Let  $f(x; w) = \arg\max_{y \in Y} w^T \varphi(x, y)$  be the prediction function. We can obtain zero

loss on the training set using this predictor if

$$\forall i. \max_{y \in Y \setminus y_i} w^T \varphi(x_i, y) \leq w^T \varphi(x_i, y_i) \quad (19.92)$$

Each one of these nonlinear inequalities can be equivalently replaced by  $|Y| - 1$  linear inequalities, resulting in a total of  $N|Y| - N$  linear constraints of the following form:

$$\forall i. \forall y \in Y \setminus y_i. w^T \varphi(x_i, y) - w^T \varphi(x_i, y_i) \geq 0 \quad (19.93)$$

For brevity, we introduce the notation

$$\delta_i(y) = \varphi(x_i, y) - \varphi(x_i, y_i) \quad (19.94)$$

so we can rewrite these constraints as  $w^T \delta_i(y) \geq 0$ . If we can achieve zero loss, there will typically be multiple solution vectors  $w$ . We pick the one that maximizes the margin, defined as

$$\gamma = \min_i f(x_i, y_i; w) - \max_{y \in Y \setminus y_i} f(x_i, y; w) \quad (19.95)$$

Since the margin can be made arbitrarily large by rescaling  $w$ , we fix its norm to be 1, resulting in the optimization problem

$$\max_{\gamma, w: \|w\|=1} \text{s.t. } \forall i. \forall y \in Y \setminus y_i. w^T \delta_i(y) \geq \gamma \quad (19.96)$$

Equivalently, we can write

$$\min_w \frac{1}{2} \|w\|^2 \text{s.t. } \forall i. \forall y \in Y \setminus y_i. w^T \delta_i(y) \geq 1 \quad (19.97)$$

To allow for the case where zero loss cannot be achieved (equivalent to the data being inseparable in the case of binary classification), we relax the constraints by introducing slack terms  $\xi_i$ , one per data case. This yields

$$\min_{w, \xi} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^N \xi_i \text{s.t. } \forall i. \forall y \in Y \setminus y_i. w^T \delta_i(y) \geq 1 - \xi_i, \xi_i \geq 0 \quad (19.98)$$

In the case of structured outputs, we don't want to treat all constraint violations equally. For example, in a segmentation problem, getting one position wrong should be punished less than getting many positions wrong. One way to achieve this is to divide the slack variable by the size of the loss (this is called slack re-scaling). This yields

$$\min_{w, \xi} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^N \xi_i \text{s.t. } \forall i. \forall y \in Y \setminus y_i. w^T \delta_i(y) \geq 1 - \xi_i L(y_i, y), \xi_i \geq 0 \quad (19.99)$$

Alternatively, we can define the margin to be proportional to the loss (this is called margin re-rescaling). This yields

$$\min_{w, \xi} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^N \xi_i \text{s.t. } \forall i. \forall y \in Y \setminus y_i. w^T \delta_i(y) \geq L(y_i, y) - \xi_i, \xi_i \geq 0 \quad (19.100)$$

(In fact, we can write  $\forall y \in Y$  instead of  $\forall y \in Y \setminus y_i$ , since if  $y = y_i$ , then  $w^T \delta_i(y) = 0$  and  $\xi_i = 0$ . By using the simpler notation, which doesn't exclude  $y_i$ , we add an extra but redundant constraint.) This latter approach is used in

### M3nets.

For future reference, note that we can solve for the  $\xi^*$  terms as follows:

$$\xi^* i(w) = \max\{0, \max y (L(y_i, y) - w^T \delta_i)\} = \max y (L(y_i, y) - w^T \delta_i) \quad (19.101)$$

Substituting in, and dropping the constraints, we get the following equivalent problem:

$$\min w \frac{1}{2} \|w\|^2 + C \sum_i \max y (L(y_i, y) + w^T \varphi(x_i, y)) - w^T \varphi(x_i, y) \quad (19.102)$$

#### 19.7.2.1 Empirical risk minimization

Let us pause and consider whether the above objective is reasonable. Recall that in the frequentist approach to machine learning (Section 6.5), the goal is to minimize the regularized empirical risk, defined by

$$R(w) + C \sum_i L(y_i, f(x_i, w)) \quad (19.103)$$

where  $R(w)$  is the regularizer, and  $f(x_i, w) = \arg\max_y w^T \varphi(x_i, y) = \hat{y}_i$  is the prediction. Since this objective is hard to optimize, because the loss is not differentiable, we will construct a convex upper bound instead. We can show that

$$R(w) + C \sum_i \max y (L(y_i, y) - w^T \delta_i) \quad (19.104)$$

is such a convex upper bound. To see this, note that

$$L(y_i, f(x_i, w)) \leq L(y_i, f(x_i, w)) - w^T \varphi(x_i, y_i) + w^T \varphi(x_i, \hat{y}_i) \quad (19.105)$$

$$\leq \max y L(y_i, y) - w^T \varphi(x_i, y_i) + w^T \varphi(x_i, y) \quad (19.106)$$

Using this bound and  $R(w) = \frac{1}{2} \|w\|^2$  yields Equation 19.102.

#### 19.7.2.2 Computational issues

Although the above objectives are simple quadratic programs (QP), they have  $O(N|Y|)$  constraints. This is intractable, since  $Y$  is usually exponentially large. In the case of the margin rescaling formulation, it is possible to reduce the exponential number of constraints to a polynomial number, provided the loss function and the feature vector decompose according to a graphical model. This is the approach used in M3nets (Taskar et al. 2003).

An alternative approach is to work directly with the exponentially sized QP. This allows for the use of more general loss functions. There are several possible methods to make this feasible. One is to use cutting plane methods. Another is to use stochastic subgradient methods. We discuss both of these below.

#### 19.7.3 Cutting plane methods for fitting SSVMs

section, we discuss an efficient algorithm for fitting SSVMs due to (Joachims et al. 2009). This method can handle general loss functions, and is implemented in the popular SVMstruct package. The method is based on the cutting plane method from convex optimization (Kelley 1960).

The basic idea is as follows. We start with an initial guess  $w$  and no constraints. At each iteration, we then do the following: for each example  $i$ , we find the “most violated” constraint involving  $x_i$  and  $\hat{y}_i$ . If the loss-augmented margin violation exceeds the current value of  $\xi_i$  by more than  $\epsilon$ , we add  $\hat{y}_i$  to the working set of constraints for this training case,  $W_i$ , and then solve the resulting new QP to find the new  $w$ ,  $\xi$ . See Figure 19.21 for a sketch. (Since at each step we only add one new constraint, we can warm-start the QP solver.) We can easily modify the algorithm to optimize the slack rescaling version by replacing the expression  $L(y_i, y) - w^T \delta_i(\hat{y}_i)$  with  $L(y_i, y)(1 - w^T \delta_i(\hat{y}_i))$ .

The key to the efficiency of this method is that only polynomially many constraints need to be added, and as soon as they are, the exponential number of other constraints are guaranteed to also be satisfied to within a tolerance of  $\epsilon$ .

#### 19.7.3.1 Loss-augmented decoding

The other key to efficiency is the ability to find the most violated constraint in line 5 of the algorithm, i.e., to compute

$$\operatorname{argmax}_{y \in Y} L(y_i, y) - w^T \delta_i(y) = \operatorname{argmax}_{y \in Y} L(y_i, y) + w^T \varphi(x_i, y) \quad (19.107)$$

We call this process loss-augmented decoding. (In (Joachims et al. 2009), this procedure is called the separation oracle.) If the loss function has an additive decomposition of the same form as the features, then we can fold the loss into the weight vector, i.e., we can find a new set of parameters  $w$  such that  $(w - )^T \delta_i(y) = w^T \delta_i(y)$ . We can then use a standard decoding algorithm, such as Viterbi, on the model  $p(y|x, w)$ . In the special case of 0-1 loss, the optimum will either be the best solution,  $\operatorname{argmax}_y w^T \varphi(x_i, y)$ , with a value of  $0 - w^T \delta_i(\hat{y})$ , or it will be the second best solution, i.e.,

$$\tilde{y} = \operatorname{argmax}_y = \hat{y} w^T \varphi(x_i, y) \quad (19.108)$$

which achieves an overall value of  $1 - w^T \delta_i(\tilde{y})$ . For chain structured CRFs, we can use the Viterbi algorithm to do

decoding; the second best path will differ from the best path in a single position, which can be obtained by changing the variable whose max marginal is closest to its decision boundary to its second best value. We can generalize this (with a bit more work) to find the N-best list (Schwarz and Chow 1990; Nilsson and Goldberger 2001).

For Hamming loss,  $L(y^*, y) = \sum_i I(y^*_i \neq y_i)$ , and for the F1 score (defined in Section 5.7.2.3), we can devise a dynamic programming algorithm to compute Equation 19.107. See (Altun et al. 2006) for details. Other models and loss function combinations will require different methods.

### 19.7.3.2 A linear time algorithm

Although the above algorithm takes polynomial time, we can do better, and devise an algorithm that runs in linear time, assuming we use a linear kernel (i.e., we work with the original features  $\phi(x, y)$  and do not apply the kernel trick). The basic idea, as explained in (Joachims et al. 2009), is to have a single slack variable,  $\xi$ , instead of  $N$ , but to use  $|Y|N$  constraints, instead of just  $N|Y|$ . Specifically, we optimize the following (assuming the margin rescaling formulation):

$$\begin{aligned} & \min_{w, \xi \geq 0} \frac{1}{2} \|w\|^2 + C\xi \\ & \text{s.t. } \forall (y_1, \dots, y_N) \in Y^N : \sum_{i=1}^N w^T \delta_i(y_i) \geq \sum_{i=1}^N L(y_i, y_i) - \xi \end{aligned} \quad (19.109)$$

Compare this to the original version, which was

$$\begin{aligned} & \min_{w, \xi \geq 0} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^N \xi_i \\ & \text{s.t. } \forall i = 1: N, \forall y \in Y : w^T \delta_i(y) \geq L(y_i, y_i) - \xi_i \end{aligned} \quad (19.110)$$

One can show that any solution  $w^*$  of Equation 19.109 is also a solution of Equation 19.110 and vice versa, with  $\xi^* = \sum_{i=1}^N \xi_i$ .

We can optimize Equation 19.109 using the cutting plane algorithm in Algorithm 10. (This is what is implemented in SVMstruct.) The inner QP in line 4 can be solved in  $O(N)$  time using the method of (Joachims 2006). In line 7 we make  $N$  calls to the loss-augmented decoder. Finally, it can be shown that the number of iterations is a constant independent on  $N$ . Thus the overall running time is linear.

### 19.7.4 Online algorithms for fitting SSVMs

Although the cutting plane algorithm can be made to run in time linear in the number of data points, that can still be slow if we have a large dataset. In such cases, it is preferable to use online learning. We briefly mention a

few possible algorithms below.

#### 19.7.4.1 The structured perceptron algorithm

A very simple algorithm for fitting SSVMs is the structured perceptron algorithm (Collins 2002). This method is an extension of the regular perceptron algorithm of Section 8.5.4. At each step, we compute  $\hat{y} = \operatorname{argmax} p(y|x)$  (e.g., using the Viterbi algorithm) for the current training sample  $x$ . If  $\hat{y} = y$ , we do nothing, otherwise we update the weight vector using

$$w_{k+1} = w_k + \varphi(y, x) - \varphi(\hat{y}, x) \quad (19.111)$$

To get good performance, it is necessary to average the parameters over the last few updates (see Section 8.5.2 for details), rather than using the most recent value.

#### 19.7.4.2 Stochastic subgradient descent

The disadvantage of the structured perceptron algorithm is that it implicitly assumes 0-1 loss, and it does not enforce any kind of margin. An alternative approach is to perform stochastic subgradient descent. A specific instance of this is the Pegasos algorithm (Shalev-Shwartz et al. 2007), which stands for “primal estimated sub-gradient solver for SVM”. Pegasos was designed for binary SVMs, but can be extended to SSVMs. Let us start by considering the objective function:

$$f(w) = N \sum_{i=1}^N \max(y_i, \hat{y}_i) + w^T \varphi(x_i, \hat{y}_i) - w^T \varphi(x_i, y_i) + \lambda \|w\|^2 \quad (19.112)$$

Letting  $\hat{y}_i$  be the argmax of this max. Then the subgradient of this objective function is

$$g(w) = N \sum_{i=1}^N \varphi(x_i, \hat{y}_i) - \varphi(x_i, y_i) + 2\lambda w \quad (19.113)$$

In stochastic subgradient descent, we approximate this gradient with a single term,  $i$ , and then perform an update:

$$w_{k+1} = w_k - \eta_k g_i(w_k) = w_k - \eta_k [\varphi(x_i, \hat{y}_i) - \varphi(x_i, y_i) + (2/N)\lambda w] \quad (19.114)$$

where  $\eta_k$  is the step size parameter, which should satisfy the Robbins-Monro conditions (Section 8.5.2.1). (Notice that the perceptron algorithm is just a special case where  $\lambda = 0$  and  $\eta_k = 1$ .) To ensure that  $w$  has unit norm, we can project it onto the  $L_2$  ball after each update.

#### 19.7.5 Latent structural SVMs

In many applications of interest, we have latent or hidden variables  $h$ . For example, in object detection problems, we may be told that the image contains an object, so  $y = 1$ , but we may not know where it is. The location of the

object, or its pose, can be considered a hidden variable. Or in machine translation, we may know the source text  $x$  (say English) and the target text  $y$  (say French), but we typically do not know the alignment between the words. We will extend our model as follows, to get a latent CRF:

$$p(y, h|x, w) = \exp(w^T \varphi(x, y, h)) Z(x, w) \quad (19.115)$$

$$Z(x, w) = \sum_h \exp(w^T \varphi(x, y, h)) \quad (19.116)$$

In addition, we introduce the loss function  $L(y^*, y, h)$ ; this measures the loss when the “action” that we take is to predict  $y$  using latent variables  $h$ . We could just use  $L(y^*, y)$  as before, since  $h$  is usually a nuisance variable and not of direct interest. However,  $h$  can sometimes play a useful role in defining a loss function.<sup>12</sup> Given the loss function, we define our objective as

$$REL(w) = -\log p(w) + \sum_i \log \left[ \sum_h \exp(-L(y_i, y, h)) \exp(w^T \varphi(x_i, y, h)) \right] / Z(x, w) \quad (19.117)$$

Using the same loose lower bound as before, we get

$$REL(w) \leq E(w) + N \sum_i \max_{y, h} L^*(y_i, y, h) + w^T \varphi(x_i, y, h) - N \sum_i \max_h w^T \varphi(x_i, y_i, h) \quad (19.118)$$

If we set  $E(w) = -\frac{1}{2} C \|w\|^2$ , we get the same objective as is optimized in latent SVMs (Yu and Joachims 2009).

Unfortunately, this objective is no longer convex.

However, it is a difference of convex functions, and hence can be solved efficiently using the CCCP or concave-convex procedure (Yuille and Rangarajan 2003). This is a method for minimizing functions of the form  $f(w) - g(w)$ , where  $f$  and  $g$  are convex. The method alternates between finding a linear upper bound  $u$  on  $-g$ , and then minimizing the convex function  $f(w) + u(w)$ ; see Algorithm 6 for the pseudocode. CCCP is guaranteed to decrease the objective at every iteration, and to converge to a local minimum or a saddle point.

When applied to latent SSVMs, CCCP is very similar to (hard) EM. In the “E step”, we compute the linear upper bound by setting  $v_t = -C N \sum_i \varphi(x_i, y_i, h^*_t)$ , where  $h^*_t = \arg \max_h w^T \varphi(x_i, y_i, h)$  (19.119)

In the “M step”, we estimate  $w$  using techniques for solving fully visible SSVMs. Specifically, we minimize

$$\frac{1}{2} \|w\|^2 + C N \sum_i \max_{y, h} (L(y_i, y, h) + w^T \varphi(x_i, y, h)) - C N \sum_i w^T \varphi(x_i, y_i, h^*_t) \quad (19.120)$$

下面先给出逐字不漏的中文翻译（含公式编号保持不变），随后是通俗解释与要点总结。

---

## 图 19.21 说明（翻译）

图 19.21 用二维情形说明割平面（cutting plane）算法。我们从估计值  $w = w^0 = 0$  开始。

- (a) 加入第一个约束；阴影区域是新的可行域。新的最小范数解是  $w^1$ 。
  - (b) 再加入一个约束；深色阴影区域是新的可行域。
  - (c) 加入第三个约束。
- 

## 19.7 结构化 SVM (Structural SVMs) (翻译)

我们已经看到，训练 CRF 需要做推断，以便计算梯度所需的“期望充分统计量”。对某些模型来说，如第 22.6 节所述，**对状态做联合的 MAP 估计**要比计算边缘分布简单得多。本节讨论一种**利用快速 MAP 求解器**来训练结构化输出分类器的方法。（为避免与参数的 MAP 估计混淆，我们常把对**状态**做的 MAP 估计称为**解码 decoding**。）这些方法称为**结构化支持向量机** (SSVM, Tsochantaridis et al., 2005)。（还有一类非常相近的方法叫**最大间隔马尔可夫网络**，M3nets, Taskar et al., 2003；关于两者差别见 19.7.2 节。）

### 19.7.1 SSVM：概率视角（翻译）

本书大部分时候都用 MAP **参数估计** 来拟合模型，即最小化下式：

$$R_{\text{MAP}}(w) = -\log p(w) - \sum_{i=1}^N \log p(y_i | x_i, w) \quad (19.78)$$

然而在测试时，我们选择标签是为了**最小化后验期望损失**（第 5.7 节定义）：

$$\hat{y}(x | w) = \arg \min_{\hat{y}} \sum_y L(\hat{y}, y) p(y | x, w) \quad (19.79)$$

其中  $L(y^{*}, \hat{y})$  是当真实值为  $y^{*}$  而我们估计为  $\hat{y}$  时的损失。因此，在做参数估计时把损失函数考虑进去似乎是合理的<sup>10</sup>。于是，参照 (Yuille and He, 2011)，我们改为最小化**训练集上的后验期望损失**：

$$R_{\text{EL}}(w) = -\log p(w) + \sum_{i=1}^N \log \sum_y L(y_i, y) p(y | x_i, w) \quad (19.80)$$

在 0-1 损失的特例下， $L(y_i, y) = 1 - \delta_{y,y_i}$ ，该式化为  $R_{\text{MAP}}$ 。

我们假设模型可写为

$$p(y | x, w) = \frac{\exp(w^\top \phi(x, y))}{Z(x, w)} \quad (19.81)$$

$$p(w) = \frac{\exp(-E(w))}{Z} \quad (19.82)$$

其中  $Z(x, w) = \sum_y \exp(w^\top \phi(x, y))$ 。并记  $L(y_i, y) = \exp(\tilde{L}(y_i, y))$ 。据此可把目标改写为

$$R_{\text{EL}}(w) = -\log p(w) + \sum_i \log \frac{\sum_y \exp(\tilde{L}(y_i, y)) \exp(w^\top \phi(x_i, y))}{Z(x_i, w)} \quad (19.83)$$

$$= E(w) + \sum_i [-\log Z(x_i, w) + \log \sum_y \exp(\tilde{L}(y_i, y) + w^\top \phi(x_i, y))] \quad (19.84)$$

我们现在通过若干上/下界来简化该目标。首先对任意函数  $f(y)$  都有

$$\max_{y \in Y} f(y) \leq \log \sum_{y \in Y} \exp[f(y)] \leq \log(|Y| \exp(\max_y f(y))) = \log |Y| + \max_y f(y) \quad (19.85)$$

例如令  $Y = \{0, 1, 2\}$ 、 $f(y) = y$ , 则

$$2 = \log[\exp(2)] \leq \log[\exp(0) + \exp(1) + \exp(2)] \leq \log[3 \times \exp(2)] = \log 3 + \log 2 \quad (19.86)$$

我们可以忽略与  $y$  无关的  $\log |Y|$  项, 把  $\max_y f(y)$  同时当作下界与上界。于是得到

$$R_{\text{EL}}(w) \sim E(w) + \sum_{i=1}^N [\max_y \{\tilde{L}(y_i, y) + w^\top \phi(x_i, y)\} - \max_y w^\top \phi(x_i, y)] \quad (19.87)$$

其中  $x \sim y$  表示存在常数  $c_1, c_2$  使  $c_1 + x \leq y + c_2$ 。遗憾的是, 该目标对  $w$  不是凸的。

不过我们可利用 log-sum-exp 的一个更松的下界构造一个凸上界:

$$f(\bar{y}) \leq \log \sum_y \exp[f(y)] \quad (19.88)$$

对任意  $\bar{y} \in Y$  成立。把它用在前述例子中, 取  $f(y) = y$ ,  $\bar{y} = 1$ , 得  $1 = \log[\exp(1)] \leq \log[\exp(0) + \exp(1) + \exp(2)]$ 。把该界用于  $R_{\text{EL}}$ :

$$R_{\text{EL}}(w) \leq E(w) + \sum_{i=1}^N \max_y \{\tilde{L}(y_i, y) + w^\top \phi(x_i, y)\} - w^\top \phi(x_i, y_i) \quad (19.89)$$

若取  $E(w) = -\frac{1}{2C} \|w\|_2^2$  (等价于球形高斯先验), 得到

$$R_{\text{SSVM}}(w) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^N [\max_y \{\tilde{L}(y_i, y) + w^\top \phi(x_i, y)\} - w^\top \phi(x_i, y_i)] \quad (19.90)$$

这与 (Tsochantaridis et al., 2005) 的 SSVM 目标相同。

当  $Y = \{-1, +1\}$ 、 $L(y^*, y) = 1 - \delta_{y,y^*}$ 、 $\phi(x, y) = \frac{1}{2}yx$  时，通过分别考虑  $y = y_i$  与  $y \neq y_i$  两种情况，上式退化为

$$R_{\text{SVM}}(w) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^N \max\{0, 1 - y_i w^\top x_i\} \quad (19.91)$$

即标准的二分类 SVM 目标 (见式 14.57)。

因此，SSVM 目标可以看作是对贝叶斯目标的一个上界的优化 (Yuille and He, 2011 首次指出)。当  $\|w\|$  很大时该上界很紧，因为此时  $p(y | x, w)$  会把概率质量集中到  $\arg \max_y p(y | x, w)$ 。不幸的是， $\|w\|$  大通常意味着过拟合，实际调参会避免处于该区间。另一种对 SVM 准则的解释是：它把精力集中在影响决策边界的参数上——当模型并不完全正确时，这比去拟合完整分布更划算。

### 19.7.2 SSVM：非概率视角（翻译）

下面按 (Tsochantaridis et al., 2005) 用更传统 (非概率) 的方式给出 SSVM。所得目标与上节相同，但该推导为后续算法做铺垫。令

$f(x; w) = \arg \max_{y \in Y} w^\top \phi(x, y)$  为预测函数。若用该预测器能在训练集上得到零损失，则必须有

$$\forall i. \max_{y \in Y \setminus \{y_i\}} w^\top \phi(x_i, y) \leq w^\top \phi(x_i, y_i) \quad (19.92)$$

每个这样的**非线性不等式**都可等价地替换为  $|Y| - 1$  个**线性不等式**，总计得到  $N|Y| - N$  个线性约束：

$$\forall i. \forall y \in Y \setminus \{y_i\}. w^\top \phi(x_i, y_i) - w^\top \phi(x_i, y) \geq 0 \quad (19.93)$$

为简洁，记

$$\delta_i(y) = \phi(x_i, y_i) - \phi(x_i, y) \quad (19.94)$$

则约束可写为  $w^\top \delta_i(y) \geq 0$ 。若能达到零损失，解向量  $w$  通常不唯一。我们选间隔最大的那个，间隔定义为

$$\gamma = \min_i f(x_i, y_i; w) - \max_{y \in Y \setminus \{y_i\}} f(x_i, y; w) \quad (19.95)$$

因把  $w$  按比例放大可使  $\gamma$  任意大，我们把  $\|w\| = 1$  固定住，于是得到

$$\max_{\gamma, w: \|w\|=1} \text{s.t. } \forall i, \forall y \neq y_i, w^\top \delta_i(y) \geq \gamma \quad (19.96)$$

等价地也可写成

$$\min_w \frac{1}{2} \|w\|^2 \quad \text{s.t. } \forall i, \forall y \in y_i, w^\top \delta_i(y) \geq 1 \quad (19.97)$$

为处理**不能零损失**（类似二分类中数据不可分）的情况，我们为每个样本引入一个**松弛变量**  $\xi_i$ ：

$$\min_{w, \xi} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^N \xi_i \quad \text{s.t. } \forall i, \forall y \in y_i, w^\top \delta_i(y) \geq 1 - \xi_i, \xi_i \geq 0 \quad (19.98)$$

在**结构化输出**中，不应把所有违约等同对待。例如在分割中，错一个位置的惩罚应小于错很多位置。做法之一是**按损失缩放松弛** (slack re-scaling)：

$$\min_{w, \xi} \frac{1}{2} \|w\|^2 + C \sum_i \xi_i \quad \text{s.t. } \forall i, \forall y \in y_i, w^\top \delta_i(y) \geq 1 - \frac{\xi_i}{L(y_i, y)}, \xi_i \geq 0 \quad (19.99)$$

或者把**间隔按损失缩放** (margin re-scaling)：

$$\min_{w, \xi} \frac{1}{2} \|w\|^2 + C \sum_i \xi_i \quad \text{s.t. } \forall i, \forall y \in y_i, w^\top \delta_i(y) \geq L(y_i, y) - \xi_i, \xi_i \geq 0 \quad (19.100)$$

（事实上可写成  $\forall y \in Y$ ；若  $y = y_i$  则  $w^\top \delta_i(y) = 0$  且  $\xi_i = 0$ 。使用不排除  $y_i$  的更简单记法，只是多了一个冗余约束。）后一种（间隔缩放）用于 M3nets。

后面会用到： $\xi_i^{*}$  的解为

$$\xi_i^{*}(w) = \max \{0, \max_y (L(y_i, y) - w^\top \delta_i(y))\} = \max_y (L(y_i, y) - w^\top \delta_i(y)) \quad (19.101)$$

代回并去掉显式约束，得到等价问题

$$\min_w \frac{1}{2} \|w\|^2 + C \sum_i [\max_y (L(y_i, y) + w^\top \phi(x_i, y)) - w^\top \phi(x_i, y_i)] \quad (19.102)$$

### 19.7.2.1 经验风险最小化（翻译）

停一下，看看上述目标是否合理。频率学派的目标（第 6.5 节）是最小化**正则化的经验风险**：

$$R(w) + \frac{C}{N} \sum_{i=1}^N L(y_i, f(x_i, w)) \quad (19.103)$$

其中  $R(w)$  为正则项， $f(x_i, w) = \arg \max_y w^\top \phi(x_i, y) = \hat{y}_i$  是预测。由于该目标难以优化（损失不可导），我们构造一个**凸上界**。可以证明

$$R(w) + \frac{C}{N} \sum_i \max_y (L(y_i, y) - w^\top \delta_i(y)) \quad (19.104)$$

是这样的凸上界。理由是

$$L(y_i, f(x_i, w)) \leq L(y_i, f(x_i, w)) - w^\top \phi(x_i, y_i) + w^\top \phi(x_i, \hat{y}_i) \quad (19.105)$$

$$\leq \max_y (L(y_i, y) - w^\top \phi(x_i, y_i) + w^\top \phi(x_i, y)) \quad (19.106)$$

取  $R(w) = \frac{1}{2} \|w\|^2$  就得到式 (19.102)。

### 19.7.2.2 计算问题 (翻译)

虽然上述目标是**二次规划** (QP)，但它有  $O(N|Y|)$  个约束。由于  $|Y|$  通常呈指数级，这不可行。对**间隔缩放**的形式，如果损失函数与特征向量可按某个**图模型**分解，便能把指数个约束降到多项式个——这是 M3nets (Taskar et al., 2003) 的做法。另一种办法是不降维，**直接处理指数规模的 QP**，从而允许更一般的损失函数。使之可行的若干方法包括：**割平面法与随机次梯度法**。下面分别介绍。

### 19.7.3 用割平面法拟合 SSVM (翻译)

本节介绍 (Joachims et al., 2009) 提出的高效 SSVM 拟合算法。该方法可处理一般的损失函数，并在广泛使用的 SVMstruct<sup>11</sup> 包中实现。它基于凸优化中的**割平面法** (Kelley, 1960)。

基本思想如下：从初始  $w$  (且无约束) 开始。每次迭代中，对每个样本  $i$ ，找到涉及  $(x_i, \hat{y}_i)$  的“最违反”**约束**；若损失增强后的间隔违背**超过当前  $\zeta_i$  多于  $\varepsilon$** ，就把该  $\hat{y}_i$  加入**训练样本  $i$  的工作集** $W_i$ ，然后解新的 QP 得到新的  $w, \zeta$ 。见图 19.21。（由于每步只新增一个约束，可对 QP 做**热启动**。）把表达式  $L(y_i, y) - w^\top \delta_i(\hat{y}_i)$  换为  $L(y_i, y)(1 - w^\top \delta_i(\hat{y}_i))$ ，即可优化**松弛缩放**版本。

该方法高效的关键在于：只需添加**多项式数量**的约束，一旦这些约束被满足，其余**指数多个**约束也必然在容差  $\varepsilon$  内被满足（证明见 Tsochantaridis et al., 2005）。

#### 19.7.3.1 损失增强解码 (翻译)

效率的另一关键是能在算法第 5 行找到**最违反约束**，即计算

$$\arg \max_{y \in Y} (L(y_i, y) - w^\top \delta_i(y)) = \arg \max_{y \in Y} (L(y_i, y) + w^\top \phi(x_i, y)) \quad (19.107)$$

我们称之为**损失增强解码** (loss-augmented decoding；在 Joachims et al., 2009 中也叫“分离预言机”)。若损失函数与特征具有相同的**可加分解**，即可把损失“吸收”进权向量：找到新的参数  $w'$ ，使  $(w')^\top \delta_i(y) = w^\top \delta_i(y)$ 。此时可在模型  $p(y | x, w')$  上用标准解码算法

(如 Viterbi) 求解。对 0-1 损失, 最优要么是最好的解  $\hat{y} = \arg \max_y w^\top \phi(x_i, y)$  (其值  $0 - w^\top \delta_i(\hat{y})$ ) , 要么是次优解

$$\tilde{y} = \arg \max_{y \in \hat{y}} w^\top \phi(x_i, y) \quad (19.108)$$

其整体值  $1 - w^\top \delta_i(\tilde{y})$ 。对链式 CRF, 可用 Viterbi 解码; 次优路径与最优路径只在一个位置不同——把离决策边界最近的那个变量改成它的第二好取值即可。进一步还可求 N-best 列表 (Schwarz & Chow, 1990; Nilsson & Goldberger, 2001)。

对 Hamming 损失  $L(y^{*}, y) = \sum_t \mathbf{1}(y_t^{*} \neq y_t)$  与 F1 分数 (第 5.7.2.3 节), 可以设计动态规划来计算式 (19.107) (见 Altun et al., 2006)。其他模型与损失组合需用对应方法。

### 19.7.3.2 线性时间算法 (翻译)

上面的算法虽是多项式时间, 但还能更快: 在使用线性核 (直接用原特征  $\phi(x, y)$ , 不使用核技巧) 时, 可把算法做到线性时间。如 (Joachims et al., 2009) 所述, 思路是把多个人的  $\zeta_i$  换成单个松弛变量  $\zeta$ , 但把约束从每人  $|Y|$  个换成整批  $|Y|^N$  个。具体地 (以间隔缩放为例) 优化:

$$\min_{w, \zeta \geq 0} \frac{1}{2} \|w\|_2^2 + C\zeta \quad \text{s.t. } \forall (y_1, \dots, y_N) \in Y^N : \frac{1}{N} w^\top \sum_{i=1}^N \delta_i(y_i) \geq \frac{1}{N} \sum_{i=1}^N L(y_i, y_i) \theta \quad (19.109)$$

与原始版本比较:

$$\min_{w, \zeta \geq 0} \frac{1}{2} \|w\|_2^2 + \frac{C}{N} \sum_i \zeta_i \quad \text{s.t. } \forall i, \forall y \in Y : w^\top \delta_i(y) \geq L(y_i, y) - \zeta_i \quad (19.110)$$

可证明, 式 (19.109) 的任意解  $w^{*}$  也是式 (19.110) 的解, 反之亦然, 且  $\zeta^{*} = \frac{1}{N} \sum_i \zeta_i^{*}$ 。

式 (19.109) 可用割平面算法 (算法 10) 优化 (SVMstruct 的实现)。第 4 行的内层 QP 可用 (Joachims, 2006) 的方法在  $O(N)$  时间内求解; 第 7 行做  $N$  次损失增强解码; 且迭代次数可证明与  $N$  无关, 因此总时间为线性。

### 19.7.4 在线拟合 SSVM 的算法 (翻译)

虽然割平面算法能做到随样本数线性增长, 但当数据极大时仍可能偏慢, 此时更适合用在线学习。下面简述几种算法。

#### 19.7.4.1 结构化感知机 (Structured Perceptron) (翻译)

一种非常简单的 SSVM 拟合方法是结构化感知机 (Collins, 2002), 它扩展了第 8.5.4 节的普通感知机。每步对当前样本  $x$  计算

$\hat{y} = \arg \max p(y | x)$  (如用 Viterbi)。若  $\hat{y} = y$  则不更新, 否则

$$w_{k+1} = w_k + \phi(y, x) - \phi(\hat{y}, x) \quad (19.111)$$

要取得好效果，需要对最近若干次更新的参数做**平均**（见 8.5.2 节），而不直接用最新一次的值。

#### 19.7.4.2 随机次梯度下降（翻译）

结构化感知机的缺点是**隐含 0-1 损失**且不强制间隔。另一种方法是**随机次梯度下降**。一个具体实例是 **Pegasos** (Shalev-Shwartz et al., 2007, “SVM 的原始估计次梯度求解器”）。

**Pegasos** 为二分类 SVM 设计，但可扩展到 SSVM。先看目标：

$$f(w) = \sum_{i=1}^N \max_{\hat{y}_i} (L(y_i, \hat{y}_i) + w^\top \phi(x_i, \hat{y}_i) - w^\top \phi(x_i, y_i)) + \lambda \|w\|^2 \quad (19.112)$$

令  $\hat{y}_i$  是上式中  $\max$  的取值，则其次梯度为

$$g(w) = \sum_{i=1}^N (\phi(x_i, \hat{y}_i) - \phi(x_i, y_i)) + 2\lambda w \quad (19.113)$$

随机次梯度下降用单个样本的项近似该梯度并更新：

$$w_{k+1} = w_k - \eta_k g_i(w_k) = w_k - \eta_k [\phi(x_i, \hat{y}_i) - \phi(x_i, y_i) + (2/N)\lambda w] \quad (19.114)$$

步长  $\eta_k$  需满足 **Robbins–Monro 条件** (8.5.2.1 节)。（注意：感知机是其特例， $\lambda = 0$ ,  $\eta_k = 1$ 。）为保证  $\|w\| = 1$ ，每次更新后可把  $w$  投影到  $l_2$  球上。

#### 19.7.5 潜变量结构化 SVM（翻译）

很多应用中含有**潜在/隐藏变量**  $h$ 。例如目标检测里我们知道图像里有目标 ( $y = 1$ )，但**不知道位置/姿态**；位置/姿态可视为隐藏变量。又如机器翻译中，给定源句  $x$  (如英文) 与目标句  $y$  (如法文)，但通常**不知道词对齐**。我们把模型扩展为**潜变量 CRF**：

$$p(y, h | x, w) = \frac{\exp(w^\top \phi(x, y, h))}{Z(x, w)} \quad (19.115)$$

$$Z(x, w) = \sum_{y, h} \exp(w^\top \phi(x, y, h)) \quad (19.116)$$

此外定义损失函数  $L(y^{*}, y, h)$ ：它度量当我们用潜变量  $h$  来预测  $y$  时的损失。也可以像以前那样用  $L(y^{*}, y)$ ，因为  $h$  往往只是**干扰变量**；但有时  $h$  有助于定义损失<sup>12</sup>。给定损失，我们的目标是

$$R_{\text{EL}}(w) = -\log p(w) + \sum_i \log \left[ \frac{\sum_{y, h} \exp(\tilde{L}(y_i, y, h)) \exp(w^\top \phi(x_i, y, h))}{Z(x, w)} \right] \quad (19.117)$$

用之前同样的松下界，可得

$$R_{\text{EL}}(w) \leq E(w) + \sum_{i=1}^N \max_{y,h} \{\tilde{L}(y_i, y, h) + w^\top \phi(x_i, y, h)\} - \sum_{i=1}^N \max_h w^\top \phi(x_i, y_i, h)$$

若取  $E(w) = -\frac{1}{2C} \|w\|^2$ , 就得到**潜变量 SVM** (Yu & Joachims, 2009) 所优化的同一目标。

但该目标**不再凸**。不过它是“**凸函数之差**” (DC), 可用 **CCCP (凹-凸程序)** (Yuille & Rangarajan, 2003) 高效求解。CCCP 用于最小化  $f(w) - g(w)$  ( $f, g$  均凸), 交替进行: 先对  $-g$  求一个线性上界  $u$ , 再最小化  $f(w) + u(w)$  (伪代码见算法 6)。CCCP 保证每次迭代都降低目标, 并收敛到一个**局部极小/鞍点**。

应用到**潜变量 SSVM**时, CCCP 与**硬 EM**很像。**“E 步”**计算线性上界, 令

$$v_t = -C \sum_{i=1}^N \phi(x_i, y_i, h_i^{*}), \quad h_i^{*} = \arg \max_h w_t^\top \phi(x_i, y_i, h) \quad (19.119)$$

**“M 步”**用“完全可见”的 SSVM 技术估计  $w$ , 即**最小化**

$$\frac{1}{2} \|w\|^2 + C \sum_{i=1}^N \max_{y,h} (L(y_i, y, h) + w^\top \phi(x_i, y, h)) - C \sum_{i=1}^N w^\top \phi(x_i, y_i, h_i^{*}) \quad (19.120)$$

## 通俗解释

### SSVM 要解决的问题

训练 CRF 要反复做“求期望”的推断, 往往很慢; 而很多结构化任务里, **只要能快速求“整条标签的最优解” (MAP/解码)**, 模型就能用得上。SSVM 就是: **不用学完整的概率分布**, 而是**直接学一个打分函数**  $w^\top \phi(x, y)$ , 让“正确结构的得分比任何错误结构高出至少与损失成比例的间隔”。训练时大量使用**解码器** (找最高分的  $y$ ), 所以当解码有快速算法时 (如链/树结构的 Viterbi、图割、图匹配等), **训练也能快很多**。

### 两种推导方式

- **概率版**: 把“最小化期望损失”的贝叶斯目标做上界近似, 得到一个**凸目标** (19.90), 这就是 SSVM。它本质是**只关心决策边界的参数学习**。
- **几何/非概率版**: 用 SVM 思路: 让正确结构与错误结构**至少差一个间隔**, 不行就用**松弛变量**; 错误越严重 (损失越大), 就要求**更大的间隔** (间隔/松弛缩放)。

### 如何高效求解

- 直接写成 QP 会有**指数多的约束**。解决法:

1. **割平面**: 每次只找**当前最违反**的一条约束 (靠一次“损失增强解码”获得) , 把它加入, 反复迭代直到没有严重违反的约束。图 19.21 展示了“不断加切平面、缩小可行域”的直观过程。
2. **线性时间版**: 合并松弛变量+巧妙的 QP 解法, 使总时间**与样本数线性** (在使用线性核时) 。
3. **在线/大规模**: 可以用**结构化感知机** (非常简单、等价 0-1 损失、无间隔保障) 或**随机次梯度** (Pegasos 思路, 带正则与投影) , 边到样本边更新。

## 损失增强解码

要找“最违反”的约束, 本质是解

$$\arg \max_y [L(y_i, y) + w^\top \phi(x_i, y)].$$

如果损失与特征的分解方式一致, **就能把损失“并入”权重**, 用常规解码器 (如 Viterbi、图割) 解决。0-1 损失时, 候选通常只需比较**最优与次优**。

## 带潜变量的 SSVM

有隐藏位姿、对齐等时, 用**潜变量 SSVM**。目标变成“正确  $y$  的**最好隐藏配置**要比任何错误  $y$  的**任意隐藏配置**高出与损失相关的间隔”。该目标非凸, 但可用 \*\*CCCP (类似硬 EM) \*\*稳定下降到一个好解:

- E: 固定  $w$ , 给每个样本选一个“最优隐藏项”;
- M: 固定隐藏, 用普通 SSVM 学  $w$ 。

## 要点小结

- **SSVM = 结构化输出的最大间隔学习**: 只学打分  $w^\top \phi(x, y)$ , 不学完整  $p(y | x)$ 。
- **与 CRF 的取舍**: CRF 需做求期望推断 (慢), SSVM 只需要**解码** (快、可用现成 MAP 求解器) 。
- **目标函数**:  $\frac{1}{2} \|w\|^2 + C \sum_i [\max_y \{L(y_i, y) + w^\top \phi(x_i, y)\} - w^\top \phi(x_i, y_i)]$ 。
- **损失进入训练**: 通过**间隔/松弛缩放**让“错得多罚得重”。
- **训练算法**: 割平面 (核心在**损失增强解码**) 、线性时间版本、在线方法 (结构化感知机、随机次梯度/ Pegasos) 。
- **潜变量 SSVM**: 用 CCCP / 硬 EM 交替优化隐藏与参数。
- **何时选 SSVM**: 当解码快、关心**预测结构质量**而非概率校准, 或模型有失配时, SSVM 往往更高效、更鲁棒。

这一节主要在讲什么？我不太懂

已思考 10s

这段“19.7 结构化 SVM (SSVM) ”主要讲——

当我们做**结构化预测**（整条序列、整幅分割图、整棵解析树等）时，如何不用学整套概率分布（像 CRF 那样要算期望/边缘，训练很慢），而是**只学一个打分函数并用解码器 (MAP) 就能高效训练。**

## 核心想法 (通俗版)

- 定义一个**打分函数**:  $F(x, y) = w^\top \phi(x, y)$ 。  
预测时只取分最高的结构:  $\hat{y} = \arg \max_y F(x, y)$ 。
- 训练时让“**正确结构**  $y_i$ ”的分数，**至少比任何错误结构高出一个与损失成比例的间隔**:

$$w^\top \phi(x_i, y_i) \geq w^\top \phi(x_i, y) + L(y_i, y) \quad (\forall y)$$

这就是把**损失**直接放进训练目标里，“错得越多 (L 越大) 必须被甩开得越远”。

- 这样得到的目标是个**凸的二次规划**（和普通 SVM 很像）：

$$\frac{1}{2} \|w\|^2 + C \sum_i [\max_y \{L(y_i, y) + w^\top \phi(x_i, y)\} - w^\top \phi(x_i, y_i)]$$

它可以看成对“最小化期望损失”的**贝叶斯目标的上界**进行优化，但只关注**决策边界**，训练更省事。

## 为什么比 CRF 快？

- CRF 训练每步要做“**求期望**”（算边缘/分区函数），很贵。
- SSVM 训练只需要反复做“**解码**”（求  $\arg \max_y$ ），而对很多图结构（链、树、某些网格）**解码快**（如 Viterbi、图割、匹配等）。
- 于是把训练问题转成“能否快速解码”的问题：**能解码  $\Rightarrow$  能高效训练。**

## 关键算法

- **割平面 (cutting plane)**：一开始没约束，每次找**当前最违反的约束**（称**损失增强解码**： $\arg \max_y L(y_i, y) + w^\top \phi(x_i, y)$ ），加进去并重解小的 QP；重复直到没有“严重违反”。图 19.21 画的就是不断“加切面、缩小可行域”的过程。
- **线性时间版本**：合并松弛变量 + 特定 QP 解法，使运行时间对样本数近似**线性**（线性核时）。
- **在线大规模**：

- **结构化感知机**: 错了就  $w \leftarrow w + \phi(x, y) - \phi(x, \hat{y})$  (相当于 0-1 损失、无显式间隔) ;
- **随机次梯度 (Pegasos 风格)** : 对上面的凸目标做 SGD, 并投影到  $l_2$  球。
- **潜变量 SSVM**: 标签里还有隐藏量  $h$  (如目标位置/姿态、词对齐) 时, 用 \*\*CCCP (凹凸程序, 类似硬 EM) \*\*交替: 先用当前  $w$  选每个样本的最优  $h$ , 再在“已填好  $h$ ”的样本上做一次普通 SSVM。

## 什么时候用 SSVM?

- 你重视预测质量而非概率校准;
- 有快速 MAP/解码器可用;
- CRF 训练太慢、模型可能失配时, SSVM 往往更高效、鲁棒。

## 一句话总结

SSVM 用“最大间隔 + 损失驱动”的方式学习结构化打分函数, 训练过程中只需要会“解码”, 因此在能快速 MAP 的任务上, 比需要算边缘的 CRF 训练更快更实用。

基于本对话里我贴过的整章原文 (19章), 输出: ①知识点总结 (分点+关键词+用途) ②子章节关联 (依赖/对比/递进+关系句, 附Mermaid可选) ③本章总结。

好的, 基于你提供的第19章内容, 这里是对本章的知识点总结、子章节的关联以及本章的总结。

### ① 知识点总结 (分点+关键词+用途)

#### 1. 条件随机场 (CRF)

- **关键词**: 结构化输出、概率图模型、全局归一化
- **用途**: 用于处理具有依赖结构的序列数据, 特别是用于序列标注、分割任务。通过条件化输入, CRF减少了冗余, 专注于建模标签之间的关系。

#### 2. 训练 CRF 模型的挑战

- **关键词**: 推理、梯度、计算复杂性、梯度下降
- **用途**: 训练 CRF 需要计算期望统计量, 这涉及到计算边缘概率或分区函数, 推理通常是计算瓶颈。常见的训练方法包括基于梯度的优化。

### 3. 最大边际 Markov 网络 (M3Nets)

- **关键词:** 最大边际、Markov 网络、近似
- **用途:** M3Nets 是一种与 CRF 相似的模型，关注最大化标签间的间隔，适用于需要最大化决策边界的任务。

### 4. 结构化支持向量机 (SSVM)

- **关键词:** MAP解码、边界优化、切割平面法、损失增强解码
- **用途:** 适用于高效训练结构化输出模型。与 CRF 相比，SSVM 训练时不需要计算边缘概率，而是通过解码（MAP估计）快速找到最优标签序列，并通过最大化间隔优化模型。

### 5. 潜在变量支持向量机 (Latent SSVM)

- **关键词:** 潜变量、隐藏层、EM算法、CCCP
- **用途:** 扩展 SSVM 以处理潜在变量的情况。潜在变量在某些任务中起到重要作用，比如图像目标检测中的位置、姿态等。这需要用到CCCP算法或硬EM进行优化。

### 6. 结构化感知机与随机梯度下降

- **关键词:** 感知机、在线学习、结构化预测
- **用途:** 用于在线训练 SSVM，可以处理动态数据集，感知机是最基础的结构化预测模型，随机梯度下降（Pegasos）算法则用于大规模训练。

---

## ② 子章节关联 (依赖/对比/递进+关系句, 附Mermaid可选)

### 1. CRF vs SSVM

- **依赖关系:** CRF 是条件随机场的核心内容，而 SSVM 是基于优化决策边界进行的结构化学习。两者都涉及结构化预测，但 CRF 更多依赖于计算边缘概率和分区函数，而 SSVM 依赖于解码的快速 MAP 求解。
- **对比:**
  - CRF: **需要推理**来计算边缘和期望统计量。
  - SSVM: **利用解码**（MAP估计）直接优化，避免了推理过程。

### 2. CRF vs M3Nets

- **递进关系:** CRF 和 M3Nets 具有相似性，但 M3Nets 更关注最大化标签间的间隔（最大边际），而 CRF 更专注于**全局归一化**来建模标签间的依赖。
- **对比:**
  - CRF: 适用于需要考虑全局依赖的场景。
  - M3Nets: 适用于需要优化决策边界的场景。

### 3. SSVM的优化目标与训练方法

- **依赖关系**: SSVM 的目标函数与标准的二次规划 (SVM) 类似, 使用最大间隔原理进行优化, 但引入了**损失函数和结构化标签的处理**。
- **递进关系**: SSVM 通过割平面法逐步优化解空间, 而**在线学习方法** (如结构化感知机和随机次梯度下降) 为大规模数据集的训练提供了解决方案。

### 4. 潜在变量SSVM与标准SSVM

- **递进关系**: 潜在变量 SSVM 是标准 SSVM 的扩展, 处理隐藏变量 (如目标位置、词对齐) 的问题。需要通过更复杂的优化方法 (如 CCCP) 来优化模型。

## ③ 本章总结

本章介绍了**结构化支持向量机 (SSVM)**, 这一方法通过引入最大间隔的理念, 解决了结构化预测中的决策边界问题。与传统的 CRF 相比, SSVM 通过将解码 (MAP估计) 作为训练的一部分, 减少了计算上的复杂性, 并通过优化最大间隔来进行训练。这种方法特别适用于那些**具有明确决策边界**的任务, 比如序列标注、图像分割等。

具体内容包括:

1. **SSVM的概率视角**: 通过对目标函数引入损失函数 (例如0-1损失), 训练时不仅仅关注标签本身, 还考虑了错误标签对模型的影响。
2. **SSVM的非概率视角**: 通过引入线性约束和最大间隔的思想, 优化过程中通过优化决策边界, 提高预测精度。
3. **切割平面法**: 介绍了如何通过逐步添加约束来解决SSVM的优化问题, 避免了计算复杂度的爆炸。
4. **潜在变量SSVM**: 扩展SSVM处理隐藏变量 (如目标位置、词对齐等) 的问题, 这需要使用像CCCP这样的特殊算法来优化。

总的来说, SSVM为结构化输出问题提供了高效的训练方法, 特别是在解码器能够快速求解MAP估计的场景下。

请用通俗易懂的话总结本章, 使我能够理解

好的, 我来用通俗易懂的语言总结一下本章的内容, 帮助你理解。

## 本章核心：结构化支持向量机 (SSVM)

本章的重点是介绍一种称为“**结构化支持向量机 (SSVM)**”的机器学习方法。这个方法主要用于解决复杂的预测问题，其中输出不仅仅是一个简单的标签，而是一个具有结构的复杂输出（比如一句话的每个词的标签，或者一张图片中的每个像素的分类）。

---

### 1. 传统的CRF与SSVM的区别

首先，我们了解了**条件随机场 (CRF)**，它通常用于处理具有依赖关系的数据（比如标注一段话中的每个词，或者标注图片中的每个区域）。然而，训练CRF时，需要通过推理来计算一些概率，这样会非常耗时和复杂。

SSVM的方法则不同，它通过一种叫做**最大边际**的技巧来训练模型。最大边际的意思是**尽量将决策边界 (不同类别的区分线) 拉得更远**，让模型能够做出更清晰的决策。

**SSVM与CRF的区别在于**，SSVM更注重优化决策边界，不需要像CRF那样每次都计算复杂的概率。

---

### 2. SSVM的工作原理

SSVM的目标是找到一条**决策边界**，这个边界会**尽量把不同类别的样本分开**。在训练过程中，SSVM会通过调整参数（像是SVM中的权重）来达到最佳效果。

- 在SSVM的训练中，我们通过**最大化间隔**来确保模型能更好地区分不同类别的样本。
- 如果有些样本被误分类了（也就是分类错误），SSVM会通过引入**损失函数**来惩罚这种错误，并在训练时优化这个损失。

这就像是在绘制一条分割线，分隔开不同类别的样本，确保分类的间隔尽量大，这样就能提高分类的准确性。

---

### 3. 切割平面法 (Cutting Plane Method)

为了高效训练SSVM，作者介绍了**切割平面法**。这个方法就像在每次迭代中通过添加“约束”来逐步优化我们的模型。通过这种方法，SSVM的训练变得更加高效，特别是当模型有很多约束时，这种方法能够大大减少计算量。

---

### 4. 潜在变量SSVM

在很多实际问题中，我们的标签并不完全可见，可能有一些**潜在变量**。例如，在物体检测中，我们知道图像中有一个物体，但我们并不知道它的位置，这个位置就是一个潜在变

量。

**潜在变量SSVM**就是一种可以处理这种潜在变量的扩展方法。它通过引入这些隐藏的信息来帮助我们做出更好的预测，并且通过**EM算法**（一种常用的优化方法）来处理这些潜在变量。

---

## 5. 总结

这章的核心是介绍了**结构化支持向量机 (SSVM)**，它通过最大化决策边界的间隔来进行训练，比传统的CRF方法更加高效。SSVM不需要计算复杂的边缘概率，而是通过解码快速优化模型的输出。切割平面法和潜在变量SSVM进一步提高了模型的效率和适用范围。

**SSVM的优势：**

- 适用于复杂的结构化输出问题（比如多标签序列或图片分割）。
- 相比CRF，SSVM训练时不需要复杂的概率推理，更加高效。
- 能处理潜在变量（如目标的位置、词语对齐等）。

这使得SSVM成为一种非常强大的机器学习工具，尤其是在需要处理结构化数据和输出的场景下。