

Lecture 7 - Learning in Graphical Models

在第 5 讲中，主要讲授了图模型以及一个简单的推理算法——BP (Belief Propagation)

在第 6 讲中，主要讲授了一些图模型的实际应用，以及如何将 prior 整合进图模型中

在本讲中，我们考虑如何在给定的数据集的情况下，如何估计这些参数的最优取值

7.1 Conditional Random Fields

- Inference VS Learning

- MRF \$\$

$$p(x_1, \dots, x_{100}) = \frac{1}{Z} \exp \left(\sum \psi_i(x_i) + \lambda \sum_{i \sim j} \psi_{ij}(x_i, x_j) \right)$$

- 目前我们已经学了 MRF，它对应的图结构是一个 *factograph* (因子图)，

注意，一般来说我们会将 potential 考虑为负对数的 factor，但是在这里我们写作更一般的形式，并且省略了符号

- Learning ?

- 我们希望能够从大量的数据集中去学习、评估最优的参数 (在这里是正则化强度 λ)

- 对于 MRF，我们将输入的随机变量考虑仅仅考虑为某个特定模型的输入 (例如在图像中，我们将每一个像素关联到一个随机变量，并将这些随机变量的集合作为 \mathcal{X})，而对于待学习的参数，我们希望从更大的数据集中学习，因此我们希望它能够有一个 structured (结构化的) 输出 \$\$

$$f_w : \textcolor{blue}{\mathbf{X}} \rightarrow \textcolor{red}{\mathbf{Y}}$$

- 我们希望构建这样一个函数——它有参数 w ，将输入空间中的 $\mathcal{X} \in \mathbf{X}$ 映射

- Conditional Random Field \$\$

$$p(\mathbf{Y} | \mathbf{X}, w) = \frac{1}{Z} \exp \left(\sum \psi_i(y_i) + \lambda \sum_{i \sim j} \psi_{ij}(y_i, y_j) \right)$$

$$(\mathcal{X}, y_i) + \lambda \sum \lim_{i \sim j} \psi_{ij}(\mathcal{X}, y_i, y_j) \right)$$

—这种情况下，给定成对的输入 \mathcal{X} 和输出 \mathcal{Y} ，我们去预测 w 在什么情况下

- 我们还可以将 CRF 写作更一般的形式
- Conditional Random Field – General Form \$\$

$$p(\mathcal{Y} | \mathcal{X}, w) = \frac{1}{Z(\mathcal{X}, w)} \exp \left\langle \mathcal{X}, w, \psi(\mathcal{X}, \mathcal{Y}) \right\rangle$$

—我们将数据集的所有输入输出对，以及对应的参数 w 级联起来，写为两个并行的等式

这里的 $\langle a, b \rangle$ 表示 a 与 b 的内积；当然也可以写为 $a^T b$ ，但是这种写法更通用

- Feature 函数
 - $\psi(\mathcal{X}, \mathcal{Y}) : \mathbf{X} \times \mathbf{R}^M \rightarrow \mathbf{R}^D$
 - 其中 M 表示输出节点的个数； D 表示特征的维度
 - Feature 函数会将 potential 全部级联起来，指定为若干输入输出对的分解，每一对可能有不同的 potential 定义

$$\psi(\mathcal{X}, \mathcal{Y}) = (\psi_1(\mathcal{X}, \mathcal{Y}_1), \dots, \psi_K(\mathcal{X}, \mathcal{Y}_K))$$
 - Feature 函数描述了输入和输出之间的相互作用，也即由输入的哪些特征可以得到这些输出
- Parameter 向量
 - $w \in \mathbf{R}^D$
 - 用于描述 D 个特征各自的权重（相对于只有单一超参数的模型，允许多维参数也是 CRF 的一个优势）
- Partition 函数（分区函数）
 - $Z(\mathcal{X}, w) = \sum_y \exp \{ \langle \mathbf{w}, \psi(\mathcal{X}, \mathcal{Y}) \rangle \}$
 - 对所有可能的输出 \mathcal{Y} 的指数值求和，用来归一化概率分布 $p(\mathcal{Y} | \mathcal{X}, \mathbf{w})$
- Learning
 - 从数据集 $\mathbf{D} = \{(\mathcal{X}^1, \mathcal{Y}^1), \dots, (\mathcal{X}^N, \mathcal{Y}^N)\}$ 中评估参数 \mathbf{w} 的过程称为学习

7.2 Parameter Estimation

- 我们该如何计算、评估 CRF 的参数呢？

- 从数学上讲，我们的目的是利用最大似然估计 (Maximize likelihood)，找到一个参数 \mathbf{w} ，使得相对于输入 \mathcal{X} ，输出 \mathcal{Y} 的概率最大

- 与机器学习中一样，我们假设数据都是独立同分布的 (Independent and Identically Distributed, IID) \$\$

$$\begin{aligned} \hat{\mathbf{w}}_{ML} &= \underset{\mathbf{w} \in \mathbb{R}^D}{\operatorname{argmax}} \\ p(\mathcal{Y} \mid \mathcal{X}, \mathbf{w}) &\quad \text{quad with quad} \\ p(\mathcal{Y} \mid \mathcal{X}, \mathbf{w}) &= \prod \limits_{n=1}^N p(\mathcal{Y}^n \mid \mathcal{X}^n, \mathbf{w}) \end{aligned}$$

- 换句话说，我们希望找到一个参数向量 $\hat{\mathbf{w}}_{ML}$ ，使得 $p_{model}(\mathcal{Y} \mid \mathcal{X}, \hat{\mathbf{w}}_{ML})$ 的

$$\begin{aligned} \hat{\mathbf{w}}_{ML} &= \underset{\mathbf{w} \in \mathbb{R}^D}{\operatorname{argmin}} \\ \mathcal{L}(\mathbf{w}) &\quad \text{quad with quad} \\ \sum \limits_{n=1}^N \log p(\mathcal{Y}^n \mid \mathcal{X}^n, \mathbf{w}) &= \end{aligned}$$

\$\$

IID 的理解

独立比较好理解：一组随机变量中，改变其中一个随机变量的值，不会影响到别的随机变量的取值（有点像图像编辑工具里面，当你组合两个图形对象时，改变一个图形的长宽比例，另一个图形也会变；可以将独立性想象为“取消组合”的感觉）

同分布的理解：我最开始想的是，似乎对于任何一组数据（有限样本），一定都可以找到一个能够描述它们分布规律的分布函数，那么如何在仅有样本可见的情况下，反推它们是否是同分布的呢？——实际上，我们很难做到，假设同分布只是为了使得最大似然估计、大数定律、中心极限定理等理论可用（也即我们认为所有的样本都通过同一个分布函数生成）；但是实际上，同分布和不同分布的一组数据可能看上去没有任何区别

likelihood VS probability

likelihood (似然) : $\mathcal{L}(\mathbf{w} | \mathcal{D}) = P(\mathcal{D} | \mathbf{w})$, 其中 \mathbf{w} 表示模型的参数, \mathcal{D} 表示观察到的数据 (样本), $P(\mathcal{D} | \mathbf{w})$ 指的是数据在参数 \mathbf{w} 下出现的概率

概率指的是参数固定时, 计算数据的可能性; 似然指的是数据固定时, 计算参数的可能性

- 接下来, 我们用之前讲过的 CRF 的形式替换这里的似然式, 并且推导一下 $\mathcal{L}(\mathbf{w})$

- \$\$

$$\begin{aligned} \mathcal{L}(\mathbf{w}) &= -\sum_{n=1}^N \log p(\mathcal{Y}^n | \mathcal{X}^n, \mathbf{w}) \\ &= -\sum_{n=1}^N \log \left(\frac{1}{Z(\mathcal{X}^n, \mathbf{w})} \exp \left\langle \mathbf{w}, \psi(\mathcal{X}^n, \mathcal{Y}^n) \right\rangle \right) \\ &= -\sum_{n=1}^N \log Z(\mathcal{X}^n, \mathbf{w}) + \log \left(\sum_{\mathcal{Y}} \exp \left\langle \mathbf{w}, \psi(\mathcal{X}^n, \mathcal{Y}) \right\rangle \right) \\ &= -\sum_{n=1}^N \log \left(\sum_{\mathcal{Y}} \exp \left\langle \mathbf{w}, \psi(\mathcal{X}^n, \mathcal{Y}) \right\rangle \right) \end{aligned}$$

- 接下来, 我们考虑如何优化它, 显然它并不是一个二次表达式, 因此我们使

- 梯度下降算法基于此基本原理, 延伸出了很多的变体算法
 - Line search (线搜索法, 图中绿色轨迹)
 - 根据 \mathcal{L} 函数的具体值动态调整步长 η , 而不是固定值
 - 以此动态调整收敛速度, 并且一定程度上防止陷入局部最优解
 - Conjugate gradients (共轭梯度法, 图中红色轨迹)
 - 改进了标准梯度下降的方向选择, 考虑当前梯度和前一步方向的关系, 提升收敛速度
 - 特别适合二次函数目标或大规模问题
- 所有梯度下降算法都基于损失函数 \mathcal{L} 的梯度作为基础, 因此能够正确衡量其梯度是非常重要的
- 下面我们考虑一下之前的负条件对数似然函数的梯度应该如何计算 \$\$

$$\begin{aligned} \mathcal{L}(\mathbf{w}) &= -\sum_{n=1}^N \log \left(\sum_{\mathcal{Y}} \exp \left\langle \mathbf{w}, \psi(\mathcal{X}^n, \mathcal{Y}) \right\rangle \right) \end{aligned}$$

- $$\log \sum_{\mathcal{Y}} \exp \left(\langle w, \psi(\mathcal{X})^n, \mathcal{Y} \rangle \right) - \nabla_w \mathcal{L}(w) = - \sum_{n=1}^N \left[\psi(\mathcal{X})^n, \mathcal{Y} \right]^n$$
- $$\begin{aligned} & \sum_{\mathcal{Y}} \frac{\exp \left(\langle w, \psi(\mathcal{X})^n, \mathcal{Y} \rangle \right)}{\left(\sum_{\mathcal{Y}} \exp \left(\langle w, \psi(\mathcal{X})^n, \mathcal{Y} \rangle \right) \right)^2} \\ & \quad \&= - \sum_{n=1}^N \left[\psi(\mathcal{X})^n, \mathcal{Y} \right]^n \end{aligned}$$
- $$\begin{aligned} & \sum_{\mathcal{Y}} p(\mathcal{Y} | \mathcal{X}^n, w) \psi(\mathcal{X})^n \\ & \quad \&= - \sum_{n=1}^N \left[\psi(\mathcal{X})^n, \mathcal{Y} \right]^n \end{aligned}$$
- $$\mathbb{E}_{\mathcal{Y}} \sim p(\mathcal{Y} | \mathcal{X}^n, w) \psi(\mathcal{X})^n$$

-推导后我们发现，其实损失函数梯度表现为真实特征值与模型预测特征值的期望的

$$\begin{aligned} & \begin{aligned} & \mathcal{L}(w) = - \sum_{n=1}^N \left[\langle w, \psi(\mathcal{X})^n, \mathcal{Y} \rangle \right] \end{aligned} \end{aligned}$$

- $$\log \sum_{\mathcal{Y}} \exp \left(\langle w, \psi(\mathcal{X})^n, \mathcal{Y} \rangle \right) - \nabla_w \mathcal{L}(w) = - \sum_{n=1}^N \left[\psi(\mathcal{X})^n, \mathcal{Y} \right]^n$$
- $$\sum_{\mathcal{Y}} p(\mathcal{Y} | \mathcal{X}^n, w) \psi(\mathcal{X})^n$$

-目前的计算复杂度： $O(NC^MD)$ – N ：数据集中的样本数量 (≈ 100 to $1,000$)

$$\begin{aligned} \psi(\mathcal{X}, \mathcal{Y}) &= (\psi_1(\mathcal{X}), \psi_2(\mathcal{X}), \dots, \psi_K(\mathcal{X})) \\ (\mathcal{X}, \mathcal{Y}_K) &\qquad \text{quad } w = (w_1, \dots, w_K) \end{aligned}$$

-也即我们可以将特征分解为 K 个 *potential*, 特征权重分解为 K 个权重向量 – 因此,

```

\begin{aligned}
& \sum \{\mathcal{Y}\} \exp \left( \langle \mathbf{w}, \psi(\mathcal{X})^n, \mathcal{Y} \rangle \right) \\
&= \sum \{\mathcal{Y}\} \exp \left( \sum_k \langle \mathbf{w}_k, \psi_k(\mathcal{X})^n, \mathcal{Y} \rangle \right) \\
&= \sum \{\mathcal{Y}\} \prod_k \underbrace{\exp \left( \langle \mathbf{w}_k, \psi_k(\mathcal{X})^n, \mathcal{Y} \rangle \right)}_{\text{k'th factor}} \\
\end{aligned}

```

—事实上，我们只是将其分解为了若干单个*potential*和对应权重向量的和的形式，即

```

\begin{aligned}
& \sum \{\mathcal{Y}\} p(\mathcal{Y} | \mathcal{X}^n, \mathbf{w}) \\
& \psi(\mathcal{X}^n, \mathcal{Y}) \\
&= \mathbb{E}[\mathcal{Y} | \sim p(\mathcal{Y} | \mathcal{X}^n, \mathbf{w})] \\
& [\psi(\mathcal{X}^n, \mathcal{Y})] \\
&= \left( \sum_k \psi_k(\mathcal{X}^n, \mathcal{Y}) \right) \\
& \left[ \psi_k(\mathcal{X}^n, \mathcal{Y}) \right]_{k \in \{1, \dots, K\}} \\
& \\
&= \left( \sum_k p(\mathcal{Y}_k | \mathcal{X}^n, \mathbf{w}) \right) \\
& \psi_k(\mathcal{X}^n, \mathcal{Y})_{k \in \{1, \dots, K\}} \\
\end{aligned}

```

—将整体预测特征期望形式拆解为每个局部特征 $\psi_k(\mathcal{X}^n, \mathcal{Y}_k)$ 的形式，然后利用边缘

```

\begin{aligned}
\mathcal{L}(\mathbf{w}) &= \textcolor{red}{\log \sum_{n=1}^N \bigg[ \langle \mathbf{w}, \psi(\mathcal{X}^n, \mathcal{Y}^n) \rangle \bigg]} \\
& \\
\bullet & \textcolor{red}{\log \sum \{\mathcal{Y}\} \exp \big[ \langle \mathbf{w}, \psi(\mathcal{X})^n, \mathcal{Y} \rangle \big]} \\
& \textcolor{red}{\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}) = - \sum_{n=1}^N \bigg[ \psi(\mathcal{X}^n, \mathcal{Y}^n) \bigg]} \\
& \\
\bullet & \textcolor{red}{\sum \{\mathcal{Y}\} p(\mathcal{Y} | \mathcal{X}^n, \mathbf{w}) \psi(\mathcal{X}^n, \mathcal{Y})} \\
\end{aligned}

```

\$—简化后的时间复杂度： $O(N \textcolor{red}{C^M D}) \rightarrow O(N \textcolor{red}{KC^F D}) - N$

- : 数据集中的样本数量 (≈ 100 to $1,000,000$)
 - M : 输出节点的数量 (≈ 100 to $1,000,000$)
 - C : 每个输出节点的最大标签数量 (≈ 2 to 100)
 - D : 特征空间 ψ 的维度
 - K : factor 的个数
 - F : 最大 factor 的阶数 (≈ 2 to 3)
-

- 另一个导致计算复杂度很高的地方在于数据集的大小—— N
 - 在每一次更新梯度的时候完整地遍历 N 个样本会导致更新速度很慢
 - 另外，通常不是所有的数据都可以加载进内存中
- 这种情况下，我们如何评估参数？
 1. 简化模型结构来计算计算量——通常会让结果变得更糟
 2. 在完整数据集上抽样一个子集进行训练——容易丢失原始数据中地信息，导致模型对全局数据学习不充分
 3. 在多个 GPU 上并行化——并没有减少总体计算量，并且同步和通信地开销成为了瓶颈
 4. Stochastic gradient descent (随机梯度下降)

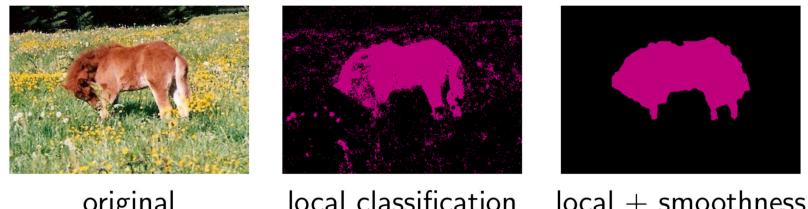
- Stochastic Gradient Descent (SGD)

- 在每一次梯度迭代中
- 构造一个随机的子集 $D' \subset D$ (一般来说, $D' \leq 256$)
- 使用一个近似的梯度 $\nabla_{\mathbf{w}}$

$$\begin{aligned} & \nabla_{\mathbf{w}} \approx - \sum_{\mathcal{D}' \in \mathcal{D}} (\mathcal{X}^n, \mathcal{Y}^n) \\ & \left[\psi(\mathcal{X}^n, \mathcal{Y}^n) - E_{\mathcal{Y}}[\psi(\mathcal{X}^n, \mathbf{w})] \right] \end{aligned}$$

- SGD VS GD - 在 SGD 中 Line search 不再可行——因此我们需要引入步长

-
- 最后一个问题，同样可能影响计算复杂度的地方——特征空间的维度 D
 - 下面给出一些具体应用中的例子来说明 D 对整体计算复杂度的影响
 - Semantic Segmentation



- ○ 局部特征

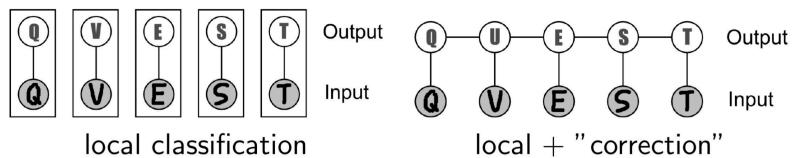
- 在语义分割任务中，局部特征 $\psi_i(\mathcal{X}, y_i)$ 可能会非常长，例如它可能取自 R^{10000} ，它描述了某个像素或超像素（有时我们会取一个大于原始像素大小 batch 作为基本单位考虑，称为超像素）的特征——这个像素属于一匹马的可能性为 0.1，属于草地的可能性为 0.7；这些特征通常采用特征描述符（如 HoG 特征，或深度特征 deep features）

- 几何约束

- 同时，还引入 pairwise（也可能 highwise）的约束—— $\psi_{ij}(y_i, y_j) = \text{fit}(y_i, y_j) \in R^1$ 描述第 i, j 关键点位置之间的几何关系，用于惩罚不合理的几何关系

- 通过 CRF 模型整合局部和整体约束，得到一个最优的估计结果

- Handwriting Recognition



- ○ 局部特征

- 类似的，对于手写字体识别任务中，我们也有着非常长的局部特征，同样可以利用 HoG 或 deep features 来描述其局部特征

- 字序约束

- $\psi_{ij}(y_i, y_j) = e_{y_i}^T, e_{y_j}^T \in R^{26 \times 26}$ ，我们可能根据该字母前面或后面出现的字母在一个单词中出现的概率，从而做一个约束——我们可能从一个很大的文本数据集中学习这一约束结果

- 通过 CRF 模型来结合局部与全局特征

- Pose Estimation



- ○ 局部特征
 - 类似的，对于姿态估计任务中，我们也有着非常长的局部特征，利用 HoG 或者 deep features 来描述其局部特征
 - 几何约束
 - 类似语义分割任务中，我们可能引入一些 pose 的先验知识，来约束两个或多个像素之间的关系
 - 通过 CRF 模型结合局部与全局特征

HoG 特征描述符

HoG (Histogram of Oriented Gradients, 方向梯度直方图) 是一种经典的图像特征提取方法，其核心思想是通过计算图像局部区域的梯度方向分布（直方图），形成一个固定维度的特征向量，从而捕捉物体的形状信息。

- 总结一下，在 CV 中应用 CRF 时的一些典型的特征函数类型
 - Unary 项 $\psi_i(\mathcal{X}, y_i)$
 - 描述局部特征，通常是高维的，可以考虑为 local classifier
 - Pairwise 项 $\psi_{ij}(y_i, y_j)$
 - 编码先验知识，通常是低位的，惩罚不连续性
 - 关联项 $\psi_{ij}(\mathcal{X}, y_i, y_j)$
 - 有时 pairwise 项也取决于输入 \mathcal{X}
- 一般来说，我们利用 pair-wise 项作为 local 结果的 clean-up 结果——即，利用平滑/连续的惩罚，来去除只有局部项结果产生的一些噪声
- Piece-wise Training
 - 有时，将整个模型训练一次不容易
 - 如果上面说的这些 factor 项是以非线性方式依赖参数的
 - 如果特征是高维的，学习速度会很慢
 - 由于这些问题，直接优化整个 CRF 模型可能会付出很高的代价——因此，我们使用一种更简单、更快速的替代方法： Piece-wise

Training

- 我们引入一个预训练分类器：

1. 为每个局部标签 y_i 预训练一个分类器 $p(y_i | \mathcal{X})$, 表示在给定输入 \mathcal{X} 的情况下, 局部标签的概率; 这些分类器可以是任意的强分类器, 如非线性 SVM 或 CNN
2. 然后我们将局部分类器的对数概率用作特征函数, 也即 $\psi_i(\mathcal{X}, y_i) = \log p(y_i | \mathcal{X})$, 至此, 这些特征函数的维度就从原来的高维降低为 1 维标量
3. 为每个局部分类器学习一个 1 维权重 w_i , 作为其对应特征函数的权重

- 优点

- 由于特征向量的维度很低, 所以训练和推理的速度都会快很多
- 局部分类器本身可以定义为很强的模型, 例如非线性 SVM、CNN 等

- 缺点

- 如果局部分类器性能较差, 那么CRF训练也没有办法解决这一点

- Summary

- 给出:

- 训练集 $\mathcal{D} = \{(\mathcal{X}^1, \mathcal{Y}^1), \dots, (\mathcal{X}^N, \mathcal{Y}^N)\}$ with $(\mathcal{X}^n, \mathcal{Y}^n) \stackrel{IID}{\approx} p_{data}(\mathcal{X}, \mathcal{Y})$

- 特征函数 $\psi(\mathcal{X}, \mathcal{Y}) : \mathbb{X} \times \mathbb{R}^M \rightarrow \mathbb{R}^D$

- 目标:

- 找到一个参数向量 $\hat{\mathbf{w}}_{ML}$ 使得

$$p_{model}(\mathcal{Y}|\mathcal{X}, \hat{\mathbf{w}}_{ML}) = \frac{1}{Z(\mathcal{X}, \hat{\mathbf{w}}_{ML})} \exp \left\{ \langle \hat{\mathbf{w}}_{ML}, \psi(\mathcal{X}, \mathcal{Y}) \rangle \right\} \approx p_{data}(\mathcal{Y}|\mathcal{X})$$

- 最小化负条件对数似然函数 \$\$

$$\begin{aligned} \mathcal{L}(\mathbf{w}) = & - \sum_{n=1}^N \left[\langle \mathbf{w}, \psi(\mathcal{X}^n, \mathcal{Y}^n) \rangle - \log \sum_{y \in \mathcal{Y}} \exp \left(\langle \mathbf{w}, \psi(\mathcal{X}^n, y) \rangle \right) \right] \\ & \left. \right] \end{aligned}$$

- 这是一个凸函数——因此GD算法可以得到全局的最优解 – 训练需要重复迭代

$$\begin{aligned} \mathcal{L}(\mathbf{w}) = & -\textcolor{red}{\sum_{n=1}^N} \left[\langle \psi(\mathcal{X}^n, \mathcal{Y}^n), \mathbf{w} \rangle - \log \textcolor{red}{\sum} \{\psi(\mathcal{Y}^n) \mid \mathcal{Y}^n \in \mathcal{Y}\} \right] \\ & \exp \left[\langle \mathbf{w}, \textcolor{red}{\sum} \{\psi(\mathcal{X}^n, \mathcal{Y}^n) \mid \mathcal{Y}^n \in \mathcal{Y}\} \rangle \right] \end{aligned}$$

| Problem | Solution | Method |
|--------------------------|-------------------|-----------------------------|
| $ \mathbb{Y} $ too large | exploit structure | belief propagation |
| N too large | mini-batches | stochastic gradient descent |
| D too large | trained ψ | piece-wise training |

7.3 Deep Structured Models

- Log-Linear Models

- \$\$

$$p(\mathcal{Y} \mid \mathcal{X}, \mathbf{w}) = \frac{1}{Z(\mathcal{X}, \mathbf{w})} \exp \left[\langle \mathbf{w}, \psi(\mathcal{X}, \mathcal{Y}) \rangle \right]$$

—参数 \mathbf{w} 和特征 $\psi(\mathcal{X}, \mathcal{Y})$ 通过内积运算结合，也即模型是参数的线性函数—

- Deep Structured Models

- \$\$

$$p(\mathcal{Y} \mid \mathcal{X}, \mathbf{w}) = \frac{1}{Z(\mathcal{X}, \mathbf{w})} \exp \left[\langle \psi(\mathcal{X}, \mathcal{Y}), \mathbf{w} \rangle \right]$$

—*potential*函数直接通过参数 \mathbf{w} 进行参数化（例如 ψ 可以是神经网络，具备

- 类似于负条件对数似然函数，我们也可以写出 deep structure 的似然函数及其梯度 \$\$

$\begin{aligned} \mathcal{L}(\mathbf{w}) &= -\sum_{n=1}^N \left[\psi(\mathcal{X}^n, \mathcal{Y}^n, \mathbf{w}) - \log \sum \{\psi(\mathcal{Y}^n) \mid \mathcal{Y}^n \in \mathcal{Y}\} \right] \end{aligned}$

$\begin{aligned} \mathcal{L}(\mathbf{w}) &= -\sum_{n=1}^N \left[\psi(\mathcal{X}^n, \mathcal{Y}^n, \mathbf{w}) - \log \sum \{\psi(\mathcal{Y}^n) \mid \mathcal{Y}^n \in \mathcal{Y}\} \right] \end{aligned}$

```

\log \sum{\mathcal{Y}} \exp \left( \psi(\mathcal{X}^n, \mathcal{Y}),
\textcolor{red}{\mathbf{w}}) \right)
\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}) &= - \sum{n=1}^N \left[ \right.
\textcolor{red}{\nabla_{\mathbf{w}} \nabla_{\mathbf{w}} \psi(\mathcal{X}^n, \mathcal{Y})} \\
\left. \sum{\mathcal{Y}} p(\mathcal{Y} : \mathcal{X}^n, \textcolor{red}{\mathbf{w}}) \right] \\
\textcolor{red}{\nabla_{\mathbf{w}} \nabla_{\mathbf{w}} \psi(\mathcal{X}^n, \mathcal{Y})} \\
\left. \right].
\end{aligned}

```

> 其中红色部分表示与负条件对数似然的区别

- 同样的，我们也可以用分解后求和的方式拆解

$$\psi(\mathcal{X}, \mathcal{Y}, \mathbf{w}) = (\psi_1(\mathcal{X}, \mathcal{Y}_1, \mathbf{w}), \dots, \psi_K(\mathcal{X}, \mathcal{Y}_K, \mathbf{w}))$$

- Deep Structured Models

- 算法过程

- 前向传播计算 $\psi_l(\mathcal{X}, \mathcal{Y}_k, \mathbf{w})$
- 反向传播计算 $\nabla_{\mathbf{w}} \psi(\mathcal{X}^n, \mathcal{Y}, \mathbf{w})$
- 利用 message passing 计算边缘分布概率
- 更新参数 \mathbf{w}

- 这个方法的问题是什么？

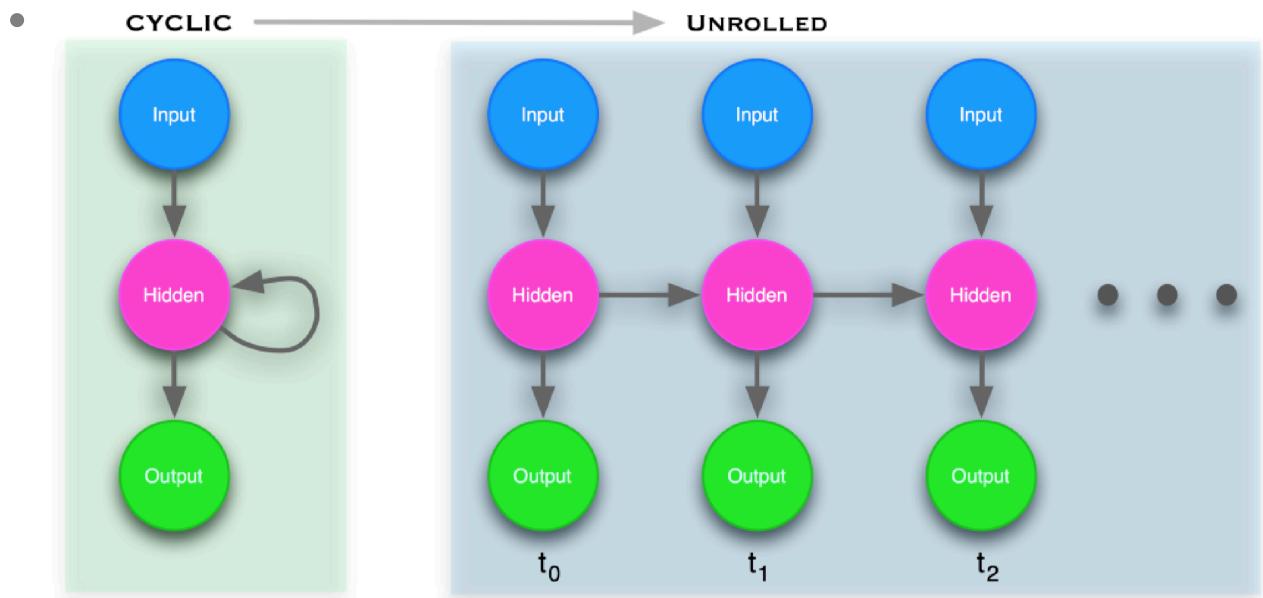
- 计算速度慢——每一次梯度更新都需要执行前向传播、反向传播、边缘分布计算
 - 而且每一步都依赖于图模型的推断，导致训练速度很慢

- 替代方案：

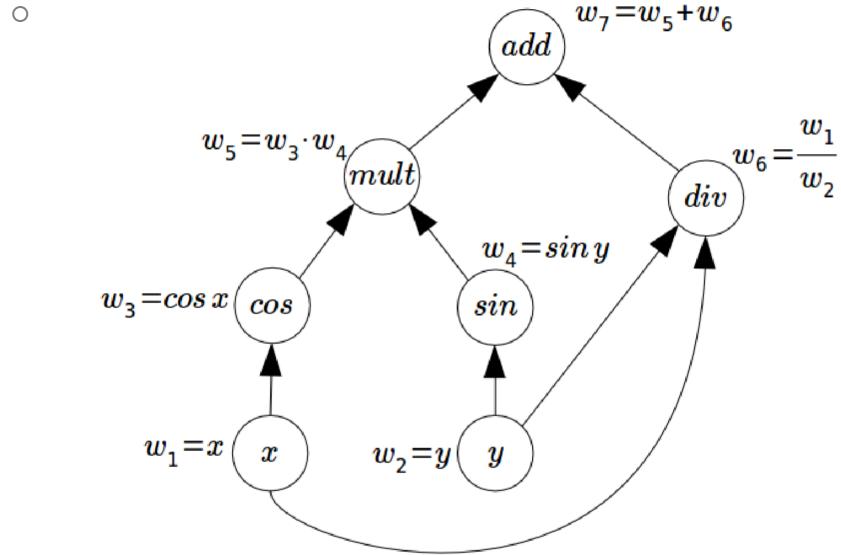
- [Interleave learning and inference, Chen et al., ICML 2015](#)，也即在训练过程中将学习和推理交替进行，而不是每次梯度更新都完整执行推理——尽管有所优化，但仍然较慢
 - Unrolled Inference——方法更简单，不过失去了模型的概率解释，导致优化过程可能偏离概率模型的原始目标

7.3.1 Inference Unrolling

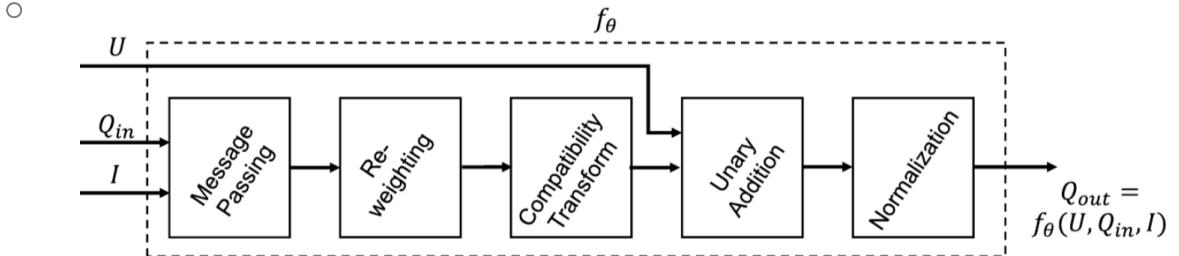
- 想法
 - 将推理过程（例如图模型的 BP 推理）考虑为一系列的小的计算过程
 - “展开” 固定数量的迭代，而不是在模型收敛时停止（与 RNN 中的时间步展开类似）
 - 使用自动微分计算梯度（深度学习框架内置的工具）
- 问题
 - 将问题简化为了经验风险最小化（Empirical risk minimization）问题，它是一种确定性的目标函数，效率足够高，但是放弃了概率推断的严谨性
 - 可以将其看作一种正则化形式，固定数量的迭代引入了一种硬约束，避免模型过度拟合——但是相应的，固定的迭代次数很可能导致模型不足以达到收敛，从而降低准确性



- Automatic Differentiation (自动微分)
 - 通过逐步计算函数的导数来实现高效梯度计算的方法
 - 关键思想
 - 给定函数 f 可以表示为一系列简单函数的复合 $f = f_0 \circ f_1 \circ \dots \circ f_n$ ，每一个简单的函数 f_k 都有一个简单的导数，然后我们利用链式法则 $\frac{\partial f}{\partial x} = \frac{\partial f_0}{\partial f_1} \cdot \frac{\partial f_1}{\partial f_2} \cdot \dots \cdot \frac{\partial f_n}{\partial x}$ ，从而可以计算出给定函数的梯度
 - ○ 计算图（例如 $f(x, y) = \cos(x) \sin(y) + \frac{x}{y}$ ）



- Examples
- Conditional Random Fields as Recurrent Neural Networks
 - $E(\mathbf{x}) = \sum_i \psi_u(x_i) + \sum_{i < j} \psi_p(x_i, x_j)$
 - 给出一个由 unary 和 pairwise 项组成的能力函数
 - $\psi_p(x_i, x_j) = \mu(x_i, x_j) \sum_{m=1}^M w^{(m)} k_G^{(m)}(\mathbf{f}_i, \mathbf{f}_j)$
 - 利用高斯核的定义，定义了特征函数



Algorithm 1 Mean-field in dense CRFs [29], broken down to common CNN operations.

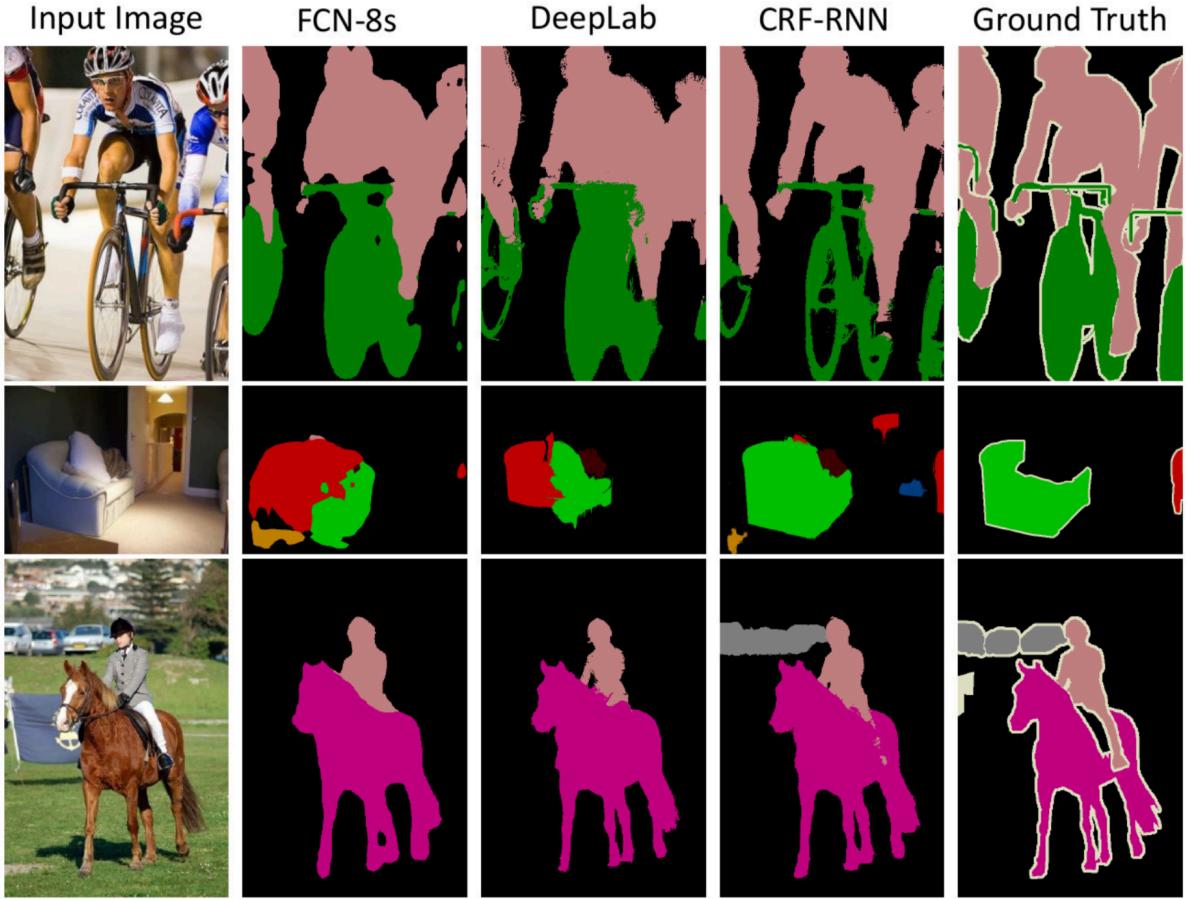
```

 $Q_i(l) \leftarrow \frac{1}{Z_i} \exp(U_i(l)) \text{ for all } i$                                 ▷ Initialization
while not converged do
     $\tilde{Q}_i^{(m)}(l) \leftarrow \sum_{j \neq i} k^{(m)}(\mathbf{f}_i, \mathbf{f}_j) Q_j(l)$  for all  $m$           ▷ Message Passing
     $\check{Q}_i(l) \leftarrow \sum_m w^{(m)} \tilde{Q}_i^{(m)}(l)$                                          ▷ Weighting Filter Outputs
     $\hat{Q}_i(l) \leftarrow \sum_{l' \in \mathcal{L}} \mu(l, l') \check{Q}_i(l')$                                ▷ Compatibility Transform
     $\check{Q}_i(l) \leftarrow U_i(l) - \hat{Q}_i(l)$                                          ▷ Adding Unary Potentials
     $Q_i \leftarrow \frac{1}{Z_i} \exp(\check{Q}_i(l))$                                          ▷ Normalizing
end while

```

- 利用称为 Mean-field 的图模型推算法

- 结果展示



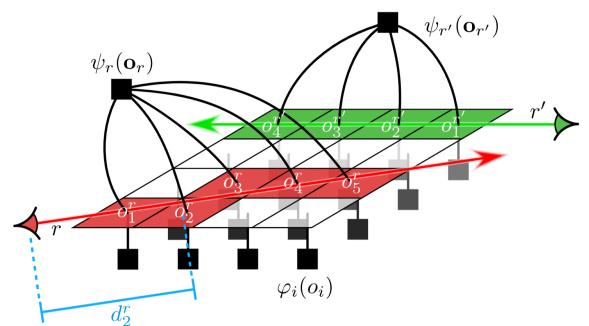
[Zheng et al.: Conditional Random Fields as Recurrent Neural Networks. ICCV, 2015.](#)

- RayNet: Learning Volumetric 3D Reconstruction

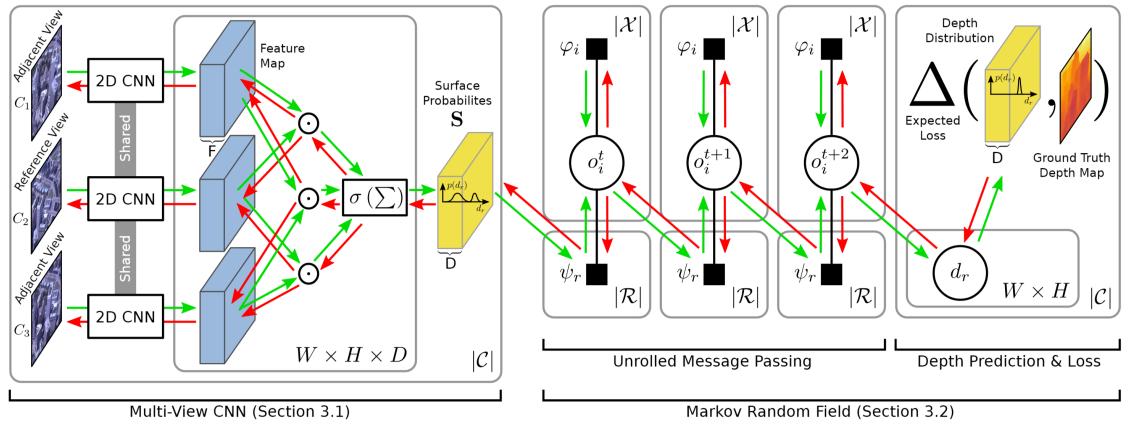
Distribution over voxel occupancies:

Corresponding factor graph:

$$\begin{aligned}
 p(\mathbf{o}) &= \frac{1}{Z} \prod_{i \in \mathcal{X}} \underbrace{\varphi_i(o_i)}_{\text{unary}} \prod_{r \in \mathcal{R}} \underbrace{\psi_r(\mathbf{o}_r)}_{\text{ray}}
 \\
 \varphi_i(o_i) &= \gamma^{o_i} (1 - \gamma)^{1-o_i}
 \\
 \psi_r(\mathbf{o}_r) &= \sum_{i=1}^{N_r} o_i^r \prod_{j < i} (1 - o_j^r) s_i^r
 \end{aligned}$$

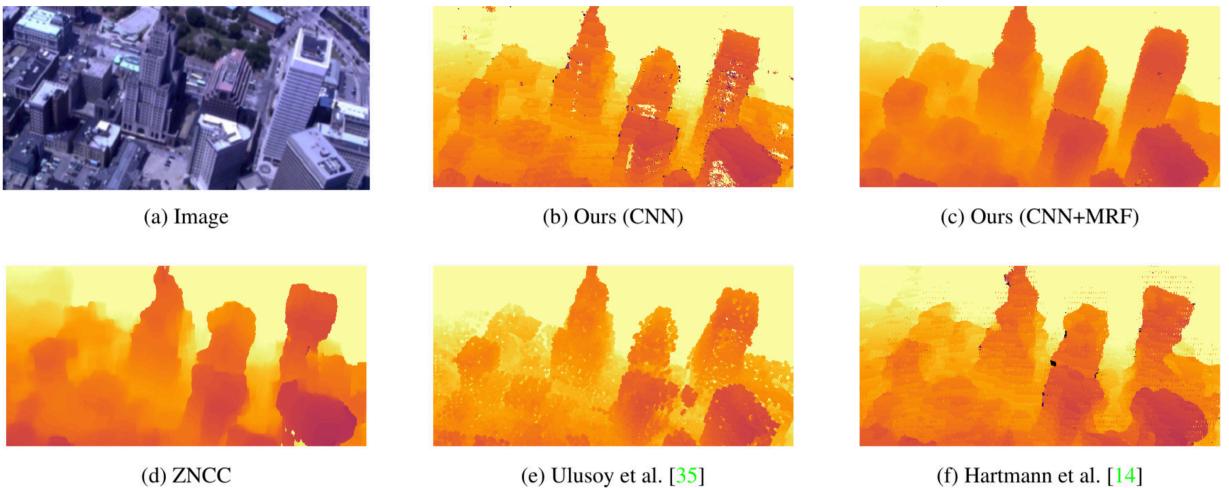


- 在每一个体素上都有一个小型的神经网络，利用该体素的占有率为，估计该体素的 depth 值，然后在传递过程中对体素的 depth 值编码



- 利用多视角 CNN 提取图像特征，然后基于 MRF，进行深度预测和表面概率的计算，最后输出深度图并计算损失，用于监督学习

- 结果展示



[Paschalidou, Ulusoy, Schmitt, van Gool and Geiger: RayNet: Learning Volumetric 3D Reconstruction with Ray Potentials. CVPR, 2018.](#)