Probabilistic Graphical Model

http://freemind.pluskid.org/machine-learning/probabilistic-graphical-model

posted on Free Mind on December 24, 2012
generated with pandoc on December 3, 2015
category: Machine Learning

tags: Probabilistic Graphical Model, Learning Models

这个学期上了一门叫做 6.438: *Algorithms for Inference* 的课，里面大部分的内容都是关于 Probabilistic Graphical Model (PGM) 的，以前对 PGM 的了解几乎处于空白状态，通过这个课多少有了一些了解，于是趁我自己还没有忘记之前在这里做一下简单的总结。不过这个东西其实牵涉到的内容非常非常多，光看 Daphne Koller 她们那本《Probabilistic Graphical Models: Principles and Techniques》的页数（1280）就知道了，而且下学期还有一门叫做 6.437 的姊妹篇课程（虽然不确定会不会去上），可想而知我们在 6.438 里学到的东西其实只是其中的一小部分啦，况且这个 field 目前也仍然还有许多新的研究在发展。所以，即使是采用蜻蜓点水的方式，也不可能在一篇 blog 里 cover 到许多的内容。好了，“即使写得很烂也不能怪我” 的借口找完之后，我们就直接进入正题吧！ :D

一般讨论 PGM 似乎都会从三个方面来讲：

1. Modeling
2. Inference
3. Learning

由于篇幅有限我就不从头介绍 Conditional Independence、Bayesian Network、Markov Random Field、Factor Graph 这些基本概念了，我们直接跳入 Inference——也就是图模型的基本用法。所谓 Inference，当然一方面可以说成名侦探柯南那样子的玄乎，说白了其实就是：在给定一个多变量的联合概率分布（Joint Distribution）的情况下，计算某些变量的边际概率分布（Marginal Distribution）。注意这里还没有涉及到 Learning 之类的东西，整个模型都是已知的。

比如，考虑一个非常 trivial 的例子，两个变量 X 和 Y 的 Joint Distribution 为 $P_{XY}(x,y)$，我想要求 X 的 Marginal[1]

$$P_X(x) = \sum_y P_{XY}(x,y)$$

这就是一个最简单的 Inference 的例子。看起来好像没有什么复杂的，就是一个求和而已。接下来我们来看一个稍微具体一点的例子。如图 1 所示的一个 Hidden Markov Model (HMM)。

假设 X_i 代表第 i 天老板的心情，而 Y_i 代表第 i 天老板给你回邮件的详细程度（详细、简略、不回）。图中的箭头代表因果关系（Causality），我们假设老板在第 i 天回邮件的情况完全取决于他那天的心情，比如，如果心情不好的话，不回邮件的概率是 0.7，简略回邮件的概率是 0.2，

[1] 简单起见大部分情况下我们都假设离散型的随机变量，因而使用求和而不是积分。不过一般涉及到具体的计算之前，直接把求和符号替换成积分符号对于大部分结论也都还是成立的。

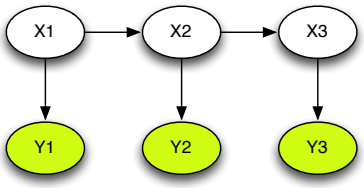

Figure 1: An example of Hidden Markov Model PGM.

详细回邮件的概率是 0.1。类似地，我们假设老板在第 i 天的心情只受到前一天心情的影响，而和更早的时间无关。并且我们假设整个模型 Joint Distribution 是已知的。

在这样的情况下，比如，我们想要知道老板在第 3 天的心情，只要算出 X_3 的 Marginal Distribution 即可。这和刚才的 trivial 例子一样，也是通过求和即可得到。通常情况下我们的 Inference 并不是凭空进行的，而是需要先得到一些 Observation。比如这里，我们用带颜色的节点表示我们可以观察到的变量 Y_i，也就是老板每天回邮件的情况。于是，“根据这两天老板回我邮件的情况，我想推测一下老板某天[2]的心情如何” 这样的问题就看起来更像我们现实生活中的 Inference 了：对比一下，已知犯罪现场、证人证词、嫌疑人相关资料等等，想要推理出谁是犯人。`:p`

[2] 可以是今天，过去某天，甚至将来某天。当然准确性完全要取决于你的模型好坏咯。

加入了 Observation（也叫做 Evidence）之后的 Inference 并没有变得很复杂，因为

$$P_{X_1|Y_1^3}(X_1|\hat{y}_1^3) = \frac{P_{X_1,Y_1^3}(X_1,\hat{y}_1^3)}{P_{Y_1^3}(\hat{y}_1^3)}$$

这里我们用一个简化的记法 Y_1^3 表示 Y_1, Y_2, Y_3。$\hat{y}_i$ 表示变量 Y_i 所 Observe 到的值。可以看到分子分母其实都各自是一个 Marginal，可以分别用求和的办法计算得到。更进一步地注意到，由于 Y 是已知的，所以分母其实只是一个常数而已，它的作用其实就是对整个式子做一下 normalization，使得最后得到的结果是一个合法的概率分布（求和等于一）。所以实际计算的时候我们可以不用专门去计算分母，而在最后一步的时候再做一下 normalization 就可以了。所以，带 Evidence 的 Inference 仍然和我们最开始讲的 trivial 的例子一样，只是一个求和而已。

全文完……啊，不，哦，好像 PGM 还没有出场是吧？`^_^bbb` 所以呢，故事其实是这样的，虽然 Inference 变成了求和的问题，但是其实求和也是可以很困难的。比如说，我下个月想出去玩，所以我想预测一下 30 天后老板的心情如何，于是我求 $P_{X_{30}}$ 这个 Marginal[3]：

[3] 注意由于未来的日子里老板如何回我邮件我是还没有观察到的，因此我也采用求和的方式把他们 marginalize 掉。

$$P_{X_{30}}(x_{30}) = \sum_{x_1^{29}, y_1^{30}} P_{X_1^{30},Y_1^{30}}(x_1^{30}, y_1^{30})$$

虽然 X 只取两种值，Y 取三种值，但是估计一下这整个和式的计算复杂度的话，需要进行的求和操作大概 10^{23} 这个数量级上。恐怖之处就在于，这东西是指数级别增长的。解决办法是通过利用 Joint Distribution 所具有的特殊**结构**来进行简化计算。

根据我们前面的描述，整个 Joint Distribution 是可以分解成这样的样子的：

$$P_{X_1^{30},Y_1^{30}}(x_1^{30},y_1^{30}) = P_{X_1}(x_1)P_{Y_1|X_1}(y_1|x_1)\prod_{i=2}^{30}P_{X_i|X_{i-1}}(x_i|x_{i-1})P_{Y_i|X_i}(y_i|x_i)$$

这样子的分解也正好可以用图 1 表示出来。分解成这样之后计算就可以简化了，为了避免记号太长了，我们这里只考虑两天的情况：

$$\begin{aligned}
P_{X_2}(x_2) &= \sum_{x_1,y_1,y_2} P_{X_1X_2Y_1Y_2}(x_1,x_2,y_1,y_2) \\
&= \sum_{x_1}\sum_{y_1}\sum_{y_2} P_{X_1}(x_1)P_{Y_1|X_1}(y_1|x_1)P_{X_2|X_1}(x_2|x_1)P_{Y_2|X_2}(y_2|x_2) \\
&= \sum_{y_2} P_{Y_2|X_2}(y_2|x_2)\sum_{x_1} P_{X_1}(x_1)P_{X_2|X_1}(x_2|x_1)\sum_{y_1} P_{Y_1|X_1}(y_1|x_1)
\end{aligned}$$

可以看到我们把整体的求和利用 Joint Distribution 的结构分解成了不同颜色部分的局部求和。计算复杂度也从原来的指数级别变成了现在的平方级别。

我们可以把红色部分记为 $M_{Y_2\to X_2}(x_2)$，蓝色部分记为 $M_{Y_1\to X_1}(x_1)$，于是上面的式子变成

$$P_{X_2}(x_2) = M_{Y_2\to X_2}(x_2)\sum_{x_1} P_{X_1}(x_1)P_{X_2|X_1}(x_2|x_1)M_{Y_1\to X_1}(x_1)$$

整个计算过程可以从 HMM 的 Graphical Model 上比较清楚地看出来，如图 2 所示，我们这里所进行的分解其实就是在每个变量的 “局部” 进行求和计算然后逐渐地将求和的结果通过类似于 Message 的方式 Propogate 到最终节点（X_2）那里。而 $M_{Y_1\to X_1}(x_1)$ 这样的记号正是表示了从 Y_1 到 X_1 的消息。中间的节点，比如 X_1，会将自己收到的消息（$M_{Y_1\to X_1}(x_1)$）进行整合，得到新的消息，例如上面式子中的橙色部分，就是 X_1 自己发往 X_2 的消息 $M_{X_1\to X_2}(x_2)$。

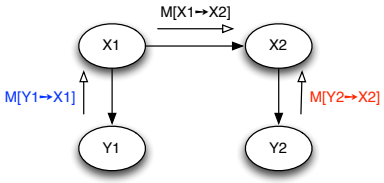

Figure 2: Message passing example on a simple HMM.

可以看到用 Graph 来分析这个计算过程使得问题变得更清楚了。另外，我们还需要注意到的是，如果现在我想计算的不是 X_2 的 Marginal，反过来要计算 X_1 的 Marginal 的话，我们新的计算过程中有一部分 Message 是和原来一模一样的，例如 $M_{Y_1\to X_1}(x_1)$ 和 $M_{Y_2\to X_2}(x_2)$ 都是不变的。利用这一点，通过动态规划的方法，我们可以得到和求一个节点的 Marginal 同阶的复杂度但是同时求得所有节点的 Marginal 的算法，也就是传说中的 Sum-Product 算法，或者叫做 Belief Propagation 算法。

小结一下就是：Inference 问题的计算复杂度比较高，但是如果概率模型本身有比较好的结构的话，我们可以通过有效地利用其结构来降低计算复杂度。而 Graphical Model 是刻画概率分布的结构的一种直观又

强大的工具，并且图论里的各种工具和结论也可以被用来对问题进行分析。另一方面，用 Graphical Model 来刻画模型还使得对许多实际问题的建模变得非常方便，因为很多问题都有非常直观的图模型的表示，例如 directed graph 的典型例子有 HMM、Topic Modeling 之类的；undirected graph 的例子也有很多，例如在做 Image Processing 的时候经常用的那种像素 grid 的模型（我不知道有没有专门的名字……-.-bb）等等。

再回到算法本身，我们刚才（非常粗略地）介绍的 Sum-Product 算法，只有在 Graph 是一个 Tree 的时候才能用（HMM 是一个很典型的 Tree 的例子）。但是大部分时候实际问题中得到的模型往往都不是一个树形结构。这个时候其中的一个选择就是用 Junction Tree 算法。需要对原来的图模型进行 Moralize、然后转化为 Chordal Graph，找出其中的 Maximal Cliques，然后构造 Junction Tree，最后就可以在得到的 Tree 上跑 (Generalized) Sum-Product 算法。最后得到的 Junction Tree 的节点的大小刻画了整个计算的复杂度，值得注意的是构造 Junction Tree 的过程几乎全都是图论上的操作，如果不借助这些图论的工具，直接对一个任意的复杂的概率分布分析其 Inference 的复杂性将会是非常繁琐的吧。

这里有必要稍微提一下我们刚才一直在说的所谓的概率模型的 “结构” 是什么：其实就是概率分布中所蕴含的 (Conditional) Independence。直观来讲，一个分布中所蕴含的独立性越多，进行 Inference 的复杂度就越小，因为如果变量之间都互相独立的话，那么 Marginalization 等操作都可以在局部小范围地进行而不牵涉到 “远处” 的其他变量，因而复杂度也就低了。而 Graph 对独立性的刻画则体现在其中节点的连通性上，这一点在有向图中并不是特别直观，但是在无向图中就完全是直观意义上的，如果 X 和 Y 在图上是不连通的，那么他们就是相互独立的。如果 X 和 Y 之间的路径被 Z 节点完全阻断，那么 X 和 Y 在观察到 Z 的时候是独立的——也就是所谓的条件独立性。于是，粗略地来看，一个 graph 上的边越稀疏，它所蕴含的（条件）独立性也就越多，因而计算复杂性也越小。这样的复杂性刻画是很符合我们的 intuition 的，因为稀疏的 Graph 看起来也比较 “简单” 嘛。

不过有时候 intuition 也并不是那么准确，因为要能真正 “看” 出复杂度的话，是需要转化为 Junction Tree 之后才行的，而在实际中有一些本来看起来蛮简单的 Graph 实际上却很复杂。比如图 3 所示的 Grid 形状的 Undirected Graphical Model，在图像分析（例如 Segmentation）中经常用到。每个节点只和周围的四个节点直接相关，好像是很简单的模型，但是如果我们把它转化为 Chordal Graph（这是构造 Junction Tree 中需要的步骤）的话，如图 3 所示的红色的边是新加入使得该图成为 Chordal Graph，这还只是 9 个节点的情况，本来准备用更具混乱性的 16 个节点的情况，但是我发现很难画清楚。而实际中对图像进行分析，如果每个像素代表一个节点的话，那么至少也是几百几千个节点，最后 Chordal 化之后将会变得一团糟，边变得稠密起来，对应的 Junction Tree 也会具有非常大的节点，最终的计算复杂度也会很高很高。

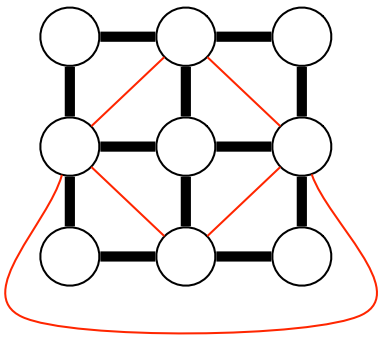

Figure 3: An example of a grid turned into chordal graph (by adding red edges).

当然这完全不能怪 Graphical Model，因为它只是如实地给出你问题的

结构的一个刻画而已，而这里之所以发生这种情况并不是这个刻画不够好，而是问题本身就具有这样的复杂性。然而，如同我们刚才所说的，这样的例子在实际问题中又是随处可见的，所以我们也不能就此撒手不管了：解决办法是，用 Approximation 吧。

Approximate Inference 试图找到一种计算复杂度比较地，但是又能给出近似正确的解的方法。具体的算法有许多，最直接的一种叫做 Loopy Belief Propagation (LBP)。这得回到我们最初的 Belief Propagation 算法，我们刚才说那个算法只能用在树上，因为那个算法就是在以 Graph 是一棵树的前提下推导出来的。但是 LBP 完全不管对应的 Graph 是不是真的是一棵树，有没有 loop 什么的，就直接把树上推导的 Message 计算公式拿过来开始算，不停地迭代直到收敛为止。怎么看怎么不靠谱啊是吧？`-.-bb` 我自己对于 LBP 的印象也就只有其中一次作业用到了而已，它确实收敛了，而且也给出了看起来还比较满意的结果，但是总的来说还是相当 tricky 的，因为动一些看起来不太相关的因素（比如 parallel update vs. in-place update）就可以导致它完全不收敛了。不过 LBP 也并不是如想象的那么不靠谱，在理论上也还是有 justification 的，这个稍后再介绍。

另外还有两类很重要的 Approximate Inference 算法：Variational Inference 和 Sampling Method。Variational Inference 的基本想法是把 Inference 问题转化为一个等价的优化问题，当然，由于等价性，如果原来的问题很难的话，得到的优化问题也会很难，不过这个时候我们可以对优化问题做一些修改，比如说把优化问题的可行域进行限制，使得优化问题限制到一个非常简化的定义域上，从而使得问题变得简单。当然，因此也无法得到正确的解了，而取决于你对优化问题的修改方法，可能会得到一个 Lower Bound 或者 Upper Bound。当然，一旦进入了 Optimization 的地盘，就有很多 Optimization 相关的技术和工具可以用了，可以说有“增加了无限可能”啊，比如说在 MAP Inference 中还有 Lagrange Dual 等方式来求 Upper Bound 的方法。这里看起来好像就是纯优化问题，和 Graphical Model 没有什么关系的样子，不过其实在具体的优化问题求解的时候往往里面会有一些子问题会需要求某个局部的期望啊之类的，仍然是需要用到 Inference 的，所以也脱不开图模型的关系。

刚才我们说 LBP 的 justification，其实也是来自于 Variational Inference。LBP 可以看成是一种叫做 Bethe Approximation 的 Variational Inference 算法的具体计算方法。具体来讲，我们可以证明如下的结论：

1. LBP 至少存在一个不动点（根据 Brouwer 不动点定理）
2. LBP 的所有不动点都是对应的 Bethe Approximation 的局部极值点（不过不知道是极小还是极大……`-.-bb`）

不过，虽然 LBP 是存在不动点的，并且不动点也都是某合理的优化问题的局部极值点，但是我们完全不知道 LBP 的迭代步骤到底会不会收敛到某个不动点——关于收敛性这一点就比较复杂了，似乎需要具体问题来进行分析，可以用 Computation Tree 之类的工具来分析。

Variational Inference 的问题在于近似是通过对模型的修改来实现的，也就是说，即使修改后的优化问题完全收敛到精确的解了，也无法得到原始问题的精确解。而 Sampling 的方法则不一样，它们可以保证 Asymptotically 得到原始问题的精确解，换句话说，当 Sample 的个数趋向于无穷多个的时候……好吧，我承认，好像也是半斤八两啊。`^_^bb`

Sampling 的基本思想是，如果我们可以得到分布 $P(x)$ 的一些 sample $x_1, \ldots, x_n$，那么很多相关的量都可以近似地计算出来，也就是用 Empirical Frequency Distribution 来近似原始的 Distribution，根据大数定理，当 sample 数趋向于无穷的时候，我们也会趋向于正确的结果。典型的方法有 Gibbs Sampling 以及推广的 Metropolis–Hastings 算法，是通过构造 Markov Chain 的方法来进行采样的。另外如果需要计算的只是某些变量的 Marginal，那么也可以用一些特殊的办法，不用对所有变量进行采样。例如在 non-discrete non-Gaussian 的 HMM 中，有种叫做 Particle Filtering 的方法，就是使用 Importance Sampling 来对局部的条件分布进行采样做近似计算的。同样的，采样的过程中也会有一些子问题需要用到 Inference，所以仍然需要 Graphical Model 的帮助的。

这样 Inference 就暂时告一段落了，接下来再简单总结一下 PGM 的 Learning 问题。Learning 有好几种情况，一种比较常见的是，已知图模型的结构，但是不知道具体的参数。比如你通过某种方法搞到了老板过去一个月的每日心情，然后结合过去一个月他回你邮件的情况，试图估计出他每天的心情转换概率分布的参数。在 Directed Graphical Model 的情况下做 Maximum Likelihood Learning 异常简单：只要对数据进行计数然后每个局部的分布用对应的 Empirical Frequency Distribution 代替就是最优的了；如果需要加 regularization 或者 prior，选择 conjugate prior 的话（一般来说也没有什么选择），也可以得到很简单的 closed-form 解。Undirected Tree 则要麻烦一点，如果是 Chordal Graph 的话，也是可以得到 closed-form 解的，但是 general 的 graph 就必须要用迭代法了。更加麻烦的情况是数据不完整的情况，比如某些变量没有观察到之类的，这个时候可以用 EM 算法来解决。

另一种情况是连 Graph 的 structure 都不知道，也就是不知道哪些边是连起来的，希望从数据中学习出来。典型的例子是医疗中某种病可能和许多不同的因素相关，现在把这些因素搜集起来，从过往的病人的数据中，希望得到一个这些因素之间的相互关系的图模型结构出来。典型的结构学习方法是定义一个 structure score，然后选出 score 最高的那个 graph structure。

一个最直接的 score 就是 likelihood score，定义为该 Graph Structure 下的 Maximum Likelihood Estimation 的模型参数所得到的 Likelihood 值。这个 score 的问题在于通常越复杂的模型分数越高。另外有一个叫做 Bayesian Score 的东西，加入了两个先验分布：一个 $P(\mathcal{G})$ 是结构先验，一个 $P(\theta_{\mathcal{G}}|\mathcal{G})$ 是在给定某个结构时模型参数的先验，整个 score 定义为：

$$\ell_B(\mathcal{G}, D) = \log P(\mathcal{G}, D) = \log P(D|\mathcal{G}) + \log P(\mathcal{G})$$

其中

$$P(D|\mathcal{G}) = \int P(D|\mathcal{G}, \theta_{\mathcal{G}}) p(\theta_{\mathcal{G}}|\mathcal{G}) d\theta_{\mathcal{G}}$$

一看就知道这玩意很难算，对于数据点个数 n 特别大的情况，Bayesian Score 可以由如下红色部分的 Bayesian Information Criterion (BIC) 来近似：

$$\ell_B(\mathcal{G}, D) \approx \hat{\ell}(\mathcal{G}, D) - \frac{\log n}{2} \dim \mathcal{G} + O(1)$$

这里 $\hat{\ell}$ 是之前提到的 Likelihood Score，而 $\dim \mathcal{G}$ 则是模型 $\mathcal{G}$ 的自由度，也就是参数个数。可以看到 BIC 其实就是在 Likelihood Score 上减去了一个代表模型复杂性的量，从而从一定程度上减缓了 Likelihood Score 一味选择复杂模型的趋势。

然后还要提一下的就是，Learning 的过程当然不是枚举所有可能的图结构然后选择分数最高的那个，而是会用一些贪心呀、迭代呀，动态规划之类的方法来尝试解决。

最后的最后，就是圣诞快乐啦！`:D` 果然用这样的篇幅来介绍最后就几乎成为名词罗列了……