漫谈 HMM：Kalman/Particle Filtering

http://freemind.pluskid.org/machine-learning/hmm-kalman-particle-filtering

posted on Free Mind on June 18, 2013
generated with pandoc on December 3, 2015
category: Machine Learning

tags: Algorithm, Probabilistic Graphical Model, Approximation

上次我们讲了 HMM 的 Forward-Backward 算法，得到了关于 α 和 β 的递推公式。不过由于中间需要进行 marginalization，这些式子里有麻烦的积分存在。如果是离散型的随机变量，积分实际上是求和，一般来说就没有什么问题了。但是对于上一次举的火星车移动的例子，实际上随机变量是连续的，比如火星车的位置 z 实际上是一个 $\mathbb{R}^3$（或者其他任何用于火星表面的坐标系）下的连续随机变量，于是我们就面临了求积分的问题。

积分是一件很困难的事情，就算是装备了 Wolfram Mathematica 的 PhD 也不能随便夸海口的，因为有很多积分式本来就没有一个式子可以把结果写下来。当然你可以像 Γ-函数那样自己定义新的符号来隐藏后面算不出来的一大坨式子，不过这对于最终需要做数值计算来说好像是躲得了和尚躲不了庙啊。所以我们就不说这些歪门邪道了，正统的解决方法主要有两种。第一种是把概率分布限制到高斯分布上，由于高斯分布像开了外挂似得，求 marginalization 和 conditioning 之后都还是高斯分布，并且本身只需要用均值和方差两 “个” 量即可描述，所以我们在这里碰到的积分问题都可以解析地计算出来，这里介绍一种代表性的算法叫做 Kalman Filtering；另外一种方法是放弃精确计算，采用近似计算的方法，这里也介绍一种代表性方法叫做 Particle Filtering。

于是一切要从高斯分布说起，高斯分布有几个很好的性质：

- 一个服从多维联合高斯分布的随机变量 X 的线性投影 AX 仍然服从高斯分布
- 一个服从多维联合高斯分布的随机变量 X 在对其中一部分维度进行 marginalization 和 conditioning 的时候，结果仍然是多维联合高斯分布

并且这些结果的高斯分布的参数可以通过 X 本身的分布参数带入公式直接得到，这里就不一个一个写出来了，感兴趣的同学可以自行推导或者参考 Wikipedia。于是，我们回顾一下上次推导得到的 Forward-Backward 算法里关于 α 的递推式

$$\alpha(z_{t+1}) = \int_{z_t} \alpha(z_t) p(z_{t+1}|z_t)\, dz_t \; p(x_{t+1}|z_{t+1})$$

以及 β 的递推式：

$$\beta(z_t) = \int_{z_{t+1}} p(x_{t+1}|z_{t+1}) \beta(z_{t+1}) p(z_{t+1}|z_t)\, dz_{t+1}$$

为了用到高斯分布的性质来求积分，我们需要把积分式里的那些东西都限制为高斯分布，其中主要就是有状态转移 $p(z_{t+1}|z_t)$ 和观察值 $p(x_t|z_t)$ 两种情况。当然使用高斯分布的原因是除了高斯分布之外都没法算，但是这个理由多少有点不够响亮，所以我们得找点其他的 justification：看看用高斯分布来进行建模到底有没有道理。

具体来说，对于火星车的例子，把 $p(x_t|z_t)$ 建模成一个高斯分布实际上是很自然的，因为在给定真实值 z 的情况下，测量误差一般会被建模为一个均值为零，方差为 Σ 的高斯分布，理由当然可以从中心极限定理啊或者各种经验啊之类的扯一大堆，总之这是一个标准做法，这样一来实际上观察值的条件分布就是

$$p(x_t|z_t) = \mathcal{N}(z_t, \Sigma)$$

接下来是状态转移，如果 t 时刻的状态 z_t 已经是服从高斯分布的了（当然从最初的其实状态 z_0 是可以由我们设定的），如何使得 $z_{t+1} = f(z_t)$ 也服从高斯分布呢？根据高斯分布随机变量的性质，如果 f 是一个线性函数，亦即存在矩阵 A 使得 $f(z_t) = Az_t$，那么 z_{t+1} 仍然是一个高斯分布。当然，考虑到之前提过的机械装置本身的操作误差，我们并不能完美地得到想要的结果，所以实际上 f 是这样子的：$f(z_t) = Az_t + \epsilon$，这里 ϵ 是一个独立的误差随机变量，我们再一次将 ϵ 也建模为一个零均值 Σ' 方差的高斯分布，原因和刚才一样：高斯分布一向被用来建模误差，更重要的是，如此一来，z_{t+1} 还是一个高斯分布。并且，在这种情况下：

$$p(z_{t+1}|z_t) = \mathcal{N}(Az_t, \Sigma')$$

值得注意的一点是，这里 f 是线性的这一点非常重要，如果 f 是一个 general 的非线性函数，即使能保证 $z_{t+1}|z_t$ 这个条件分布是高斯的，也没法做到 marginal z_{t+1} 是高斯的，那样的话 Forward-Backward 的递推就没有办法走下去。

所以，总结起来，把一切都建模成高斯分布的做法实际上在各个方面都是相当自然和有道理的，唯一的一个缺陷就是状态转移这里被限制成了只能是线性函数。这也是为什么 Kalman Filter 被称为是线性方法。这在有些时候是一个非常大的限制，所以不得不再探索其他的方法和扩展，其中有一个扩展叫做 Extended Kalman Filter，是将非线性的状态转移进行线性化近似，据说这个扩展算法被用在了阿波罗登月计划里。不过就最土的 Kalman Filter 本身已经在非常多的问题中得到了广泛和成功的应用了。

说了半天，其实还没有讲 Kalman Filter 到底是什么，其实 Kalman Filter 就是在我们刚才的描述下将所有的分布全部取成高斯的情况下的连续随机变量 HMM 的 Forward-Backward 算法……的变种。之所以说是一种变种，是因为它并不是计算 α 和 β，而是计算了 α 和一个叫做 γ 的东西。其中 α 的定义和之前一样的，而 γ 其实也不陌生：

$$\gamma(z_t) \triangleq p(z_t|x_0^T) = \frac{\alpha(z_t)\beta(z_t)}{p(x_0^T)}$$

也可以推导出一个像 β 一样的从后往前的递推公式。于是整个算法还是 Forward 和 Backward 两轮，Forward 的时候是计算 $\alpha(z_t) = p(z_t|x_0^t)$，也就是根据到目前为止的观测数据所能得到的对于状态值 z_t 的分布估计，其中每一步迭代又被分为两步完成，第一步计算 $p(z_t|x_0^{t-1})$ 叫做 “Prediction”，也就是直接根据状态转移预测从 $t-1$ 时刻到 t 时刻之后可能所处的状态了，接下来会计算 $p(z_t|x_0^t)$，也就是加入 t 时刻的测量值 x_t，这一步叫做 “Update”。整个正向的迭代合在一起叫做 “Filtering”。而 Backward 的迭代则是根据所有时刻 $0, \dots, T$ 的观察值来对 z_t 时刻的状态分布的计算进行修正，也就是计算 $p(z_t|x_0^T)$，这个步骤也有个名字，叫做 “Smoothing”。

那么，如果是任意的连续的随机变量，到底应该怎么做呢？既然精确计算不行了，那么久只有求助于近似计算。首先面临的一个问题是如何去表示一个分布，最直接的近似表达一个分布的方法莫过于所谓的 empirical distribution 了：如果 $x_1, \dots, x_N$ 是服从于 P_x 的 IID 样本[1]的话，那么

$$\hat{P}_x(\cdot) = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}_{x_i}(\cdot)$$

就是关于原来分布的一个近似（如果是概率密度函数的话则要使用狄拉克 δ 函数），特别地，关于原来分布的期望可以通过这个 empirical distribution 近似为

$$\mathbb{E}_x[f(x)] \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

根据强大数定理，当 $N \to \infty$ 的时候右边会以概率 1 收敛到左边。也就是说，一个分布可以由一堆 sample 来近似表示，不过这里我们要用一个改良的表示方法，除了 sample 之外，每个 sample x_i 还被赋予一个权值 w_i。具体来说，这个东西是来自于 Importance Sampling——本身也是一个非常重要的 sampling 方法，它可以在不知道 partition function 的情况下对一个分布进行采用。

具体来说，如果一个分布

$$p(x) = \frac{p^*(x)}{Z_p}$$

其中 Z_p 是 partition function（或者叫做 normalization factor），在概率图模型中经常都会碰到 Z_p 很难算的情况，这样采样也会变得很困难，

[1] 突然想起之前有过一次有趣的关于 “样本” 是什么的讨论，结论是，所谓 x 是分布 P_x 的一个样本，这句话在数学上的意义其实就是在说 x 是个随机变量并且它的分布是 P_x。

不过 Importance Sampling 可以解决这个问题。具体的办法是先找另一个比较容易采用的分布 q（比如是高斯分布或者平均分布之类的），并从 q 那里采 N 个样本 $x_1, \ldots, x_N$，然后对每个样本定义权值

$$w_i = \frac{p^*(x_i)}{q(x_i)}$$

在对期望进行近似的时候，不再使用简单的平均而是加权平均：

$$\frac{\frac{1}{N}\sum_{i=1}^N w_i f(x_i)}{\frac{1}{N}\sum_{i=1}^N w_i} \to \mathbb{E}_p[f(x)] \ a.s., \quad \text{as } N \to \infty$$

这里的收敛性同样是根据大数定理得到的，对分子分母分别求期望就可以很容易得到想要的结果[2]。而整个过程有效地避开了对 Z_p 的计算。当然 Importance Sampling 也并不是随便抓一个平均分布就可以解决世间一切问题的银弹，对采样分布 q 的选取有时候会对近似的好坏收敛速率等产生非常大的影响。

[2] 注意这里必须在强大数定理的那种 “以概率 1 收敛” 意义下才能分子分母分别求期望之后再相除，只用弱大数定理似乎不足以得到这样的结论。感谢 XH 同学提供的参考。

而把这里的权重和样本组成的对叫做 Particle，就成了 Particle Filter 算法，其整体结构和 Kalman Filter 如出一辙，其中 Prediction 是从 $p(z_t|x_0^t)$ 计算 $p(z_{t+1}|x_0^t)$，换句话说，我们已经有了 $p(z_t|x_0^t)$ 分布的 particle $\{z_i^t, w_i^t\}_{i=1}^N$，现在要通过这些 particle 来生成 $p(z_{t+1}|x_0^t)$ 的 particle。由于 HMM 结构的特殊性，我们可以令

- 每个 z_i^{t+1} 是由 $p(z_{t+1}|z_i^t)$ 中采样出来的一个 sample。我们假设这个步骤是容易实现的，否则可以再用一次 Importance Sampling 或者用其他的 Metropolis-Hastings 之类的近似采样方法。
- 而每个样本的权重则保持不变 $\tilde{w}_i^{t+1} = w_i^t$。

假设原来的 z_i^t 本身是采样自分布 q，而权重递归地等于 $w_i^t = p(z_i^t|x_0^t)/q(z_i^t)$。这里上下标有点混乱了.....-.-bb t 是用于表示时刻的量，而 i 则用于对 particle 进行计数。于是我们有

$$\begin{aligned}
\mathbb{E}[\tilde{w}_i^{t+1}] &= \int \tilde{w}_i^{t+1} q(z_i^t)\, dz_i^t \\
&= \int p(z_i^t|x_0^t)\, dz_i^t \\
&= 1 \\
\mathbb{E}[\tilde{w}_i^{t+1} f(z_i^{t+1})] &= \mathbb{E}\left[\mathbb{E}[\tilde{w}_i^{t+1} f(z_i^{t+1})|z_i^t]\right] \\
&= \int\int \tilde{w}_i^{t+1} f(z_i^{t+1}) p(z_i^{t+1}|z_i^t) q(z_i^t)\, dz_i^{t+1} dz_i^t \\
&= \int f(z_i^{t+1}) \int p(z_i^t|x_0^t) p(z_i^{t+1}|z_i^t)\, dz_i^t\, dz_i^{t+1} \\
&= \int f(z_i^{t+1}) p(z_i^{t+1}|x_0^t)\, dz_i^{t+1} \\
&= \mathbb{E}[f(z_i^{t+1})|x_0^t]
\end{aligned}$$

也就是说 particle $z_i^{t+1}, \tilde{w}_i^{t+1}$ 确实是在对分布 $p(z_{t+1}|x_0^t)$ 在做近似。接下来是 Update step，这次的 particle 是保持 z_i^{t+1} 不变，而修改权重

$$w_i^{t+1} = \tilde{w}_i^{t+1} p(x_{i+1}|z_i^{t+1}) = w_i^t p(x_{i+1}|z_i^{t+1})$$

和刚才一样我们可以证明对这样的 particle 加权平均其实是在对 $p(z_i^{t+1}|x_0^{t+1})$ 在做近似，具体推导就不专门写出来了。总而言之这样不断地迭代就可以一直得到关于 $p(z_t|x_0^t)$ 的一个近似表示，而又不受限于状态转移的具体形式——当然，必须也要使得对 $p(z_{t+1}|z_i^t)$ 的工作可以完成才行。

结束之前再提一句的是，随着迭代的进行有些 particle 的权重可能会变得非常非常小，所以有时候需要进行一下 resampling。而用 particle 对分布进行近似这个思想当然不止适用于 HMM，而是可以用在 general 的 graphical model 的 inference 上，但是对于 general 的 graph，就没有 HMM 那么良好的结构有上面的巧妙的算法可以直接得到迭代下一步的 particle 了，而是需要显式地把几个分布相乘再 marginalize 然后采样新的 particle，其中每一步都并不是 trivial 的，比如 “两个由 particle 表示的分布相乘之后究竟应该会是什么样的”。还有就是 particle filter 属于 Sequential Monte Carlo 方法的一种。

至于为什么这些算法都叫做什么什么 Filter 嘛……其实我也不知道，估计是因为是一步一步迭代的，并且每次输入 x_t 会得到 $p(z_t|x_0^t)$ 所以看起来像一个 filter 一样的？