Particle Filter Tutorial 粒子滤波：从推导到应用（一）

粒子滤波贝叶斯滤波

前言：

博主在自主学习粒子滤波的过程中，看了很多文献或博客，不知道是看文献时粗心大意还是悟性太低，看着那么多公式，总是无法把握住粒子滤波的思路，也无法将理论和实践对应起来。比如：理论推导过程中那么多概率公式，概率怎么和系统的状态变量对应上了？状态粒子x_k是怎么一步步采样出来的，为什么程序里面都是直接用状态方程来计算？粒子的权重是怎么来的？经过一段时间的理解，总算理清了它的脉络。同时也觉得，只有对理论的推导心中有数了，才能知道什么样的地方可以用这个算法，以及这个算法有什么不足。因此，本文将结合实际程序给出粒子滤波的详细推导，在推导过程中加入博主自己的理解，如有不妥，请指出，谢谢。

文章架构：

由最基础的贝叶斯估计开始介绍，再引出蒙特卡罗采样，重要性采样，SIS 粒子滤波，重采样，基本粒子滤波 Generic Particle Filter，SIR 粒子滤波，这些概念的引进，都是为了解决上一个概念中出现的问题而环环相扣的。最后给出几个在 matlab 和 python 中的应用，例程包括图像跟踪，滤波，机器人定位。

再往下看之前，也可以看看《卡尔曼滤波:从推导到应用》，好对这种通过状态方程来滤波的思路有所了解。

一、贝叶斯滤波

假设有一个系统，我们知道它的状态方程，和测量方程如下：

$$x_k = f_k\left(x_{k-1}, v_{k-1}\right)$$ 如：

$$x_k = \frac{x_{k-1}}{2} + \frac{25x_{k-1}}{1+x_{k-1}^2} + 8cos\left(1.2\left(k-1\right)\right) + v_k \quad (1)$$

$$y_k = h_k\left(x_k, n_k\right)$$ 如：

$$y_k = \frac{x_k^2}{20} + n_k \quad (2)$$

其中 *x* 为系统状态，*y* 为测量到的数据，*f,h* 是状态转移函数和测量函数，*v,n* 为过程噪声和测量噪声，噪声都是独立同分布的。上面对应的那个例子将会出现在程序中。

从贝叶斯理论的观点来看，状态估计问题（目标跟踪、信号滤波）就是根据之前一系列的已有数据$y_{1:k}$（后验知识）递推的计算出当前状态x_k的可信度。这个可信度就是概率公式$p(x_k|y_{1:k})$，它需要通过预测和更新两个步奏来递推的计算。

预测过程是利用系统模型(状态方程 1)预测状态的先验概率密度，也就是通过已有的先验知识对未来的状态进行猜测，即 p(x(k)|x(k-1))。更新过程则利用最新的测量值对先验概率密度进行修正，得到后验概率密度，也就是对之前的猜测进行修正。

在处理这些问题时，一般都先假设系统的状态转移服从一阶马尔科夫模型，即当前时刻的状态 x(k)只与上一个时刻的状态 x(k-1)有关。这是很自然的一种假设，就像小时候玩飞行棋，下一时刻的飞机跳到的位置只由当前时刻的位置和骰子决定。同时，假设 k 时刻测量到的数据 y(k)只与当前的状态 x(k)有关，如上面的状态方程 2。

为了进行递推，不妨假设已知 k-1 时刻的概率密度函数$p(x_{k-1}|y_{1:k-1})$

预测：由上一时刻的概率密度$p(x_{k-1}|y_{1:k-1})$得到$p(x_k|y_{1:k-1})$，这个公式的含义是既然有了前面 1：k-1 时刻的测量数据，那就可以预测一下状态 x(k)出现的概率。

计算推导如下：

$$\begin{aligned} p(x_k|y_{1:k-1}) &= \int p(x_k, x_{k-1}|y_{1:k-1})\, dx_{k-1} \\ &= \int p(x_k|x_{k-1}, y_{1:k-1})\, p(x_{k-1}|y_{1:k-1})\, dx_{k-1} \\ &= \int p(x_k|x_{k-1})\, p(x_{k-1}|y_{1:k-1})\, dx_{k-1} \end{aligned}$$

等式的第一行到第二行纯粹是贝叶斯公式的应用。第二行得到第三行是由于一阶马尔科夫过程的假设，状态 x(k)只由 x(k-1)决定。

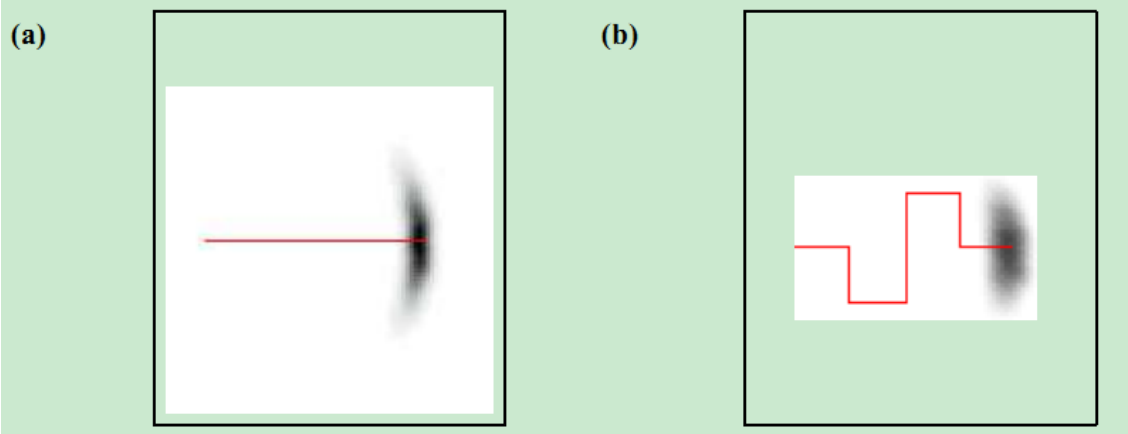
楼主看到这里的时候想到两个问题：

第一：既然 $p(x_{k-1}|x_{k-1}, y_{1:k-1}) = p(x_k|x_{k-1})$，x(k)都只由 x(k-1)决定了，即$p(x_k|x_{k-1})$，在这里弄一个$p(x_k|y_{1:k-1})$是什么意思？

这两个概率公式含义是不一样的，前一个是纯粹根据模型进行预测，x(k)实实在在的由 x(k-1)决定，后一个是既然测到的数据和状态是有关系的，现在已经有了很多测量数据 y 了，那么我可以根据已有的经验对你进行预测，只是猜测 x(k)，而不能决定 x(k)。

第二：上面公式的最后一行$p\left(x_{k-1}|y_{1:k-1}\right)$是假设已知的，但是$p\left(x_k|x_{k-1}\right)$怎么得到呢？

其实$p\left(x_k|x_{k-1}\right)$它由系统状态方程决定，它的概率分布形状和系统的过程噪声v_{k-1}形状一模一样。如何理解呢?观察状态方程(1)式我们知道，x(k) = Constant(x(k-1)) + v(k-1) 也就是 x(k)由一个通过 x(k-1)计算出的常数叠加一个噪声得到。看下面的图：



如果没有噪声，x(k)完全由 x(k-1)计算得到，也就没由概率分布这个概念了，由于出现了噪声，所以 x(k)不好确定，他的分布就如同图中的阴影部分，实际上形状和噪声是一样的，只是进行了一些平移。理解了这一点，对粒子滤波程序中，状态 x(k)的采样的计算很有帮助，要采样 x(k)，直接采样一个过程噪声，再叠加上 f(x(k-1)) 这个常数就行了。

更新：由$p\left(x_k|y_{1:k-1}\right)$得到后验概率$p\left(x_k|y_{1:k}\right)$。这个后验概率才是真正有用的东西，上一步还只是预测，这里又多了 k 时刻的测量，对上面的预测再进行修正，就是滤波了。这里的后验概率也将是代入到下次的预测，形成递推。

推导：

$$p\left(x_k|y_{1:k}\right) = \frac{p(y_k|x_k,y_{1:k-1})p(x_k|y_{1:k-1})}{p(y_k|y_{1:k-1})}$$

$$= \frac{p(y_k|x_k)p(x_k|y_{1:k-1})}{p(y_k|y_{1:k-1})}$$

其中归一化常数：

$$p(y_k|y_{1:k-1}) = \int p(y_k|x_k)\, p(x_k|y_{1:k-1})\, dx_k$$

等式第一行到第二行是因为测量方程知道, y(k)只与 x(k)有关，$p(y_k|x_k)$也称之为似然函数，由量测方程决定。也和上面的推理一样，$y_k = h(x_k) + n_k$, x(k)部分是常数，$p(y_k|x_k)$也是只和量测噪声 n(k)的概率分布有关，注意这个也将为 SIR 粒子滤波里权重的采样提供编程依据。

贝叶斯滤波到这里就告一段落了。但是，请注意上面的推导过程中需要用到积分，这对于一般的非线性，非高斯系统，很难得到后验概率的解析解。为了解决这个问题，就得引进蒙特卡洛采样。关于它的具体推导请见 下一篇博文。

（转载请注明作者和出处：http://blog.csdn.net/heyijia0327 未经允许请勿用于商业用途）

reference：

1.M. Sanjeev Arulampalam 《A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking》
2.ZHE CHEN 《Bayesian Filtering: From Kalman Filters to Particle Filters, and Beyond》
3.Sebastian THRUN 《Probabilistic Robotics》
3.百度文库 《粒子滤波理论》

Particle Filter Tutorial 粒子滤波：从推导到应用（二）

粒子滤波采样

二、蒙特卡洛采样

假设我们能从一个目标概率分布 p(x)中采样到一系列的样本（粒子）$x_1, \ldots, x_N$，（至于怎么生成服从 p(x)分布的样本，这个问题先放一放），那么就能利用这些样本去估计这个分布的某些函数的期望值。譬如：

$$E(f(x)) = \int_a^b f(x)\, p(x)\, dx$$

$$Var\left(f\left(x\right)\right)=E\left(f\left(x\right)-E\left(f\left(x\right)\right)\right)^{2}=\int_{a}^{b}\left(f\left(x\right)-E\left(f\left(x\right)\right)\right)^{2}p\left(x\right)dx$$

上面的式子其实都是计算期望的问题，只是被积分的函数不同。

蒙特卡洛采样的思想就是用平均值来代替积分，求期望：

$$E\left(f\left(x\right)\right)\approx\frac{f(x_1)+...+f(x_N)}{N}$$

这可以从大数定理的角度去理解它。我们用这种思想去指定不同的 f(x)以便达到估计不同东西的目的。比如：要估计一批同龄人的体重，不分男女，在大样本中男的有 100 个，女的有 20 个，为了少做事，我们按比例抽取 10 个男的，2 个女的，测算这 12 个人的体重求平均就完事了。注意这里的按比例抽取，就可以看成从概率分布 p(x)中进行抽样。

下面再看一个稍微学术一点的例子：

假设有一粒质地均匀的骰子。规定在一次游戏中，连续四次抛掷骰子，至少出现一次 6 个点朝上就算赢。现在来估计赢的概率。我们用$x_k^{(n)}$来表示在第 n 次游戏中，第 k 次投掷的结果，k=1...4。对于分布均匀的骰子，每次投掷服从均匀分布,即：

$$x_k\sim u\left(1,6\right)$$

这里的区间是取整数，1,2,3,4,5,6，代表 6 个面。由于每次投掷都是独立同分布的，所以这里的目标分布 p(x)也是一个均匀分布$\chi=\{1,...6\}^4$。一次游戏就是χ空间中的一个随机点。

为了估计取胜的概率，在第 n 次游戏中定义一个指示函数:

$$f\left(x^{(n)}\right)=\mathbb{I}\left\{0<\sum_{k=1}^{4}\mathbb{I}\left\{x_k^{(n)}=6\right\}\right\}$$

其中，指示函数 I{x }是指，若 x 的条件满足，则结果为 1，不满足结果为 0。回到这个问题，这里函数 f()的意义就是单次游戏中，若四次投掷中只要有一个 6 朝上，f()的结果就会是 1。由此，就可以估计在这样的游戏中取胜的期望，也就是取胜的概率：

$$\theta=E\left(f(x)\right)\approx\frac{1}{N}\sum_{n=1}^{N}f(x^{(n)})$$

当抽样次数 N 足够大的时候，上式就逼近真实取胜概率了，看上面这种估计概率的方法，是通过蒙特卡洛方法的角度去求期望达到估计概率的目的。是不是就跟我们抛硬币的例子一样，抛的次数足够多就可以用来估计正面朝上或反面朝上的概率了。

当然可能有人会问，这样估计的误差有多大，对于这个问题，有兴趣的请去查看我最下面列出的参考文献 2。（啰嗦一句：管的太多太宽，很容易让我们忽略主要问题。博主就是在看文献过程中，这个是啥那个是啥，都去查资料，到头来粒子滤波是干嘛完全不知道了，又重新看资料。个人感觉有问题还是先放一放，主要思路理顺了再关注细节。）

接下来，回到我们的主线上，在滤波中蒙特卡洛又是怎么用的呢？

由上面我们知道，它可以用来估计概率，而在上一节中，贝叶斯后验概率的计算里要用到积分，为了解决这个积分难的问题，可以用蒙特卡洛采样来代替计算后验概率。

假设可以从后验概率中采样到 N 个样本，那么后验概率的计算可表示为：

$$\hat{p}(x_n|y_{1:k}) = \frac{1}{N}\sum_{i=1}^{N}\delta(x_n - x_n^{(i)}) \approx p(x_n|y_{1:k})$$

其中，在这个蒙特卡洛方法中，我们定义$f(x) = \delta(x_n - x_n^{(i)})$,是狄拉克函数(dirac delta function)，跟上面的指示函数意思差不多。

看到这里，既然用蒙特卡洛方法能够用来直接估计后验概率，现在估计出了后验概率，那到底怎么用来做图像跟踪或者滤波呢？要做图像跟踪或者滤波，其实就是想知道当前状态的期望值：

$$\begin{aligned} E[f(x_n)] &\approx \int f(x_n)\hat{p}(x_n|y_{1:k})dx_n \\ &= \frac{1}{N}\sum_{i=1}^{N}\int f(x_n)\delta(x_n - x_n^{(i)})dx_n \\ &= \frac{1}{N}\sum_{i=1}^{N} f(x_n^{(i)}) \end{aligned} \quad (1)$$

也就是用这些采样的粒子的状态值直接平均就得到了期望值，也就是滤波后的值，这里的 f(x) 就是每个粒子的状态函数。这就是粒子滤波了，只要从后验概率中采样很多粒子，用它们的状态求平均就得到了滤波结果。

思路看似简单，但是要命的是，后验概率不知道啊，怎么从后验概率分布中采样！所以这样直接去应用是行不通的，这时候得引入重要性采样这个方法来解决这个问题。

三、重要性采样

无法从目标分布中采样，就从一个已知的可以采样的分布里去采样如 q(x|y)，这样

上面的求期望问题就变成了：

$$
\begin{aligned}
E\left[f\left(x_k\right)\right] &= \int f(x_k)\frac{p(x_k|y_{1:k})}{q(x_k|y_{1:k})}q(x_k|y_{1:k})dx_k \\
&= \int f(x_k)\frac{p(y_{1:k}|x_k)p(x_k)}{p(y_{1:k})q(x_k|y_{1:k})}q(x_k|y_{1:k})dx_k \\
&= \int f(x_k)\frac{W_k(x_k)}{p(y_{1:k})}q(x_k|y_{1:k})dx_k
\end{aligned}
$$

(2)式

其中

$$
W_k(x_k) = \frac{p(y_{1:k}|x_k)p(x_k)}{q(x_k|y_{1:k})} \propto \frac{p(x_k|y_{1:k})}{q(x_k|y_{1:k})}
$$

由于：

$$
p(y_{1:k}) = \int p(y_{1:k}|x_k)p(x_k)dx_k
$$

所以(2)式可以进一步写成：

$$
\begin{aligned}
E[f(x_k)] &= \frac{1}{p(y_{1:k})}\int f(x_k)W_k(x_k)q(x_k|y_{1:k})dx_k \\
&= \frac{\int f(x_k)W_k(x_k)q(x_k|y_{1:k})dx_k}{\int p(y_{1:k}|x_k)p(x_k)dx_k} \\
&= \frac{\int f(x_k)W_k(x_k)q(x_k|y_{1:k})dx_k}{\int W_k(x_k)q(x_k|y_{1:k})dx_k} \\
&= \frac{E_{q(x_k|y_{1:k})}[W_k(x_k)f(x_k)]}{E_{q(x_k|y_{1:k})}[W_k(x_k)]}
\end{aligned}
\qquad \text{(3)式}
$$

上面的期望计算都可以通过蒙特卡洛方法来解决它，也就是说，通过采样 N 个样本 $\left\{x_k^{(i)}\right\} \sim q(x_k|y_{1:k})$,用样本的平均来求它们的期望，所以上面的（3）式可以近似为：

$$
E\left[f(x_k)\right] \approx \frac{\frac{1}{N}\sum_{i=1}^{N} W_k(x_k^{(i)})f(x_k^{(i)})}{\frac{1}{N}\sum_{i=1}^{N} W_k(x_k^{(i)})}
$$

$$= \sum_{i=1}^{N} \tilde{W}_k(x_k^{(i)}) f(x_k^{(i)}) \qquad (4)式$$

其中：

$$\tilde{W}_k(x_k^{(i)}) = \frac{W_k(x_k^{(i)})}{\sum_{i=1}^{N} W_k(x_k^{(i)})}$$

这就是归一化以后的权重，而之前在(2)式中的那个权重是没有归一化的。

注意上面的(4)式，它不再是（1）式中所有的粒子状态直接相加求平均了，而是一种加权和的形式。不同的粒子都有它们相应的权重，如果粒子权重大，说明信任该粒子比较多。

到这里已经解决了不能从后验概率直接采样的问题，但是上面这种每个粒子的权重都直接计算的方法，效率低，因为每增加一个采样，p(x(k) |y(1:k))都得重新计算，并且还不好计算这个式子。所以求权重时能否避开计算 p(x(k)|y(1:k))？而最佳的形式是能够以递推的方式去计算权重，这就是所谓的序贯重要性采样（SIS），粒子滤波的原型。

下面开始权重 w 递推形式的推导：

假设重要性概率密度函数$q(x_{0:k} \mid y_{1:k})$，这里 x 的下标是 0:k，也就是说粒子滤波是估计过去所有时刻的状态的后验。假设它可以分解为：

$$q(x_{0:k} \mid y_{1:k}) = q(x_{0:k-1} \mid y_{1:k-1}) q(x_k \mid x_{0:k-1}, y_{1:k})$$

后验概率密度函数的递归形式可以表示为：

$$p(x_{0:k} \mid Y_k) = \frac{p(y_k \mid x_{0:k}, Y_{k-1}) p(x_{0:k} \mid Y_{k-1})}{p(y_k \mid Y_{k-1})}$$

$$= \frac{p(y_k \mid x_{0:k}, Y_{k-1}) p(x_k \mid x_{0:k-1}, Y_{k-1}) p(x_{0:k-1} \mid Y_{k-1})}{p(y_k \mid Y_{k-1})}$$

$$= \frac{p(y_k \mid x_k) p(x_k \mid x_{k-1}) p(x_{0:k-1} \mid Y_{k-1})}{p(y_k \mid Y_{k-1})}$$

$$\propto p(y_k|x_k)p(x_k|x_{k-1})p(x_{0:k-1}|Y_{k-1})$$

其中，为了表示方便，将 y(1:k) 用 Y(k) 来表示，注意 Y 与 y 的区别。同时，上面这个式子和上一节贝叶斯滤波中后验概率的推导是一样的，只是之前的 x(k)变成了这里的 x(0:k)，就是这个不同，导致贝叶斯估计里需要积分，而这里后验概率的分解形式却不用积分。

粒子权值的递归形式可以表示为:

$$w_k^{(i)} \propto \frac{p(x_{0:k}^{(i)}|Y_k)}{q(x_{0:k}^{(i)}|Y_k)}$$

$$= \frac{p(y_k|x_k^{(i)})p(x_k^{(i)}|x_{k-1}^{(i)})p(x_{0:k-1}^{(i)}|Y_{k-1})}{q(x_k^{(i)}|x_{0:k-1}^{(i)},Y_k)q(x_{0:k-1}^{(i)}|Y_{k-1})}$$

$$= w_{k-1}^{(i)} \frac{p(y_k|x_k^{(i)})p(x_k^{(i)}|x_{k-1}^{(i)})}{q(x_k^{(i)}|x_{0:k-1}^{(i)},Y_k)} \quad (5)式$$

注意，这种权重递推形式的推导是在前面（2）式的形式下进行推导的，也就是没有归一化。而在进行状态估计的公式为 $\sum_{i=1}^{N} \tilde{W}_k(x_k^{(i)}) f(x_k^{(i)})$ 这个公式中的的权重是归一化以后的，所以在实际应用中，递推计算出 w(k)后,要进行归一化，才能够代入(4)式中去计算期望。同时，上面(5)式中的分子是不是很熟悉，在上一节贝叶斯滤波中我们都已经做了介绍，p(y|x),p(x(k)|x(k-1))的形状实际上和状态方程中噪声的概率分布形状是一样的，只是均值不同了。因此这个递推的(5)式和前面的非递推形式相比，公式里的概率都是已知的，权重的计算可以说没有编程方面的难度了。权重也有了以后，只要进行稍微的总结就可以得到 SIS Filter。

四、Sequential Importance Sampling (SIS) Filter

在实际应用中我们可以假设重要性分布 q()满足：

$$q(x_k|x_{0:k-1}, y_{1:k}) = q(x_k|x_{k-1}, y_k)$$

这个假设说明重要性分布只和前一时刻的状态 x(k-1)以及测量 y(k)有关了，那么(5)式就可以转化为：

$$w_k^{(i)} \propto w_{k-1}^{(i)} \frac{p(y_k|x_k^{(i)})p(x_k^{(i)}|x_{k-1}^{(i)})}{q(x_k^{(i)}|x_{k-1}^{(i)}, y_k)}$$

在做了这么多假设和为了解决一个个问题以后，终于有了一个像样的粒子滤波算法了，他就是序贯重要性采样滤波。
下面用伪代码的形式给出这个算法：

----------pseudo code----------

$$[\{x_k^{(i)}, w_k^{(i)}\}_{i=1}^N] = SIS(\{x_{k-1}^{(i)}, w_{k-1}^{(i)}\}_{i=1}^N, Y_k)$$

For i=1:N

(1)采样：$x_k^{(i)} \sim q(x_k^{(i)}|x_{k-1}^{(i)}, y_k)$;

(2)根据 $w_k^{(i)} \propto w_{k-1}^{(i)} \frac{p(y_k|x_k^{(i)})p(x_k^{(i)}|x_{k-1}^{(i)})}{q(x_k^{(i)}|x_{k-1}^{(i)}, y_k)}$ 递推计算各个粒子的权重；

End For

粒子权值归一化。粒子有了，粒子的权重有了，就可以由(4)式,对每个粒子的状态进行加权去估计目标的状态了。

----------end ----------

这个算法就是粒子滤波的前身了。只是在实际应用中，又发现了很多问题，如粒子权重退化的问题，因此就有了重采样(resample)，就有了基本的粒子滤波算法。还有就是重要性概率密度 q()的选择问题，等等。都留到下一章 去解决。

在这一章中，我们是用的重要性采样这种方法去解决的后验概率无法采样的问题。实际上，关于如何从后验概率采样，也就是如何生成特定概率密度的样本，有很多经典的方法（如拒绝采样，Markov Chain Monte Carlo，Metropolis-Hastings 算法，

Gibbs 采样），这里面可以单独作为一个课题去学习了，有兴趣的可以去看看《统计之都 的一篇博文》，强烈推荐，参考文献里的前几个也都不错。

（转载请注明作者和出处：http://blog.csdn.net/heyijia0327 未经允许请勿用于商业用途）

reference：

1. Gabriel A. Terejanu 《Tutorial on Monte Carlo Techniques》

2. Taylan Cemgil 《A Tutorial Introduction to Monte Carlo methods, Markov Chain Monte Carlo and Particle Filtering》

3. M. Sanjeev Arulampalam 《A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking》

4. ZHE CHEN 《Bayesian Filtering: From Kalman Filters to Particle Filters, and Beyond》

5.百度文库《粒子滤波理论》

6. Haykin 《Neural Networks and learning Machines 》Chapter 14

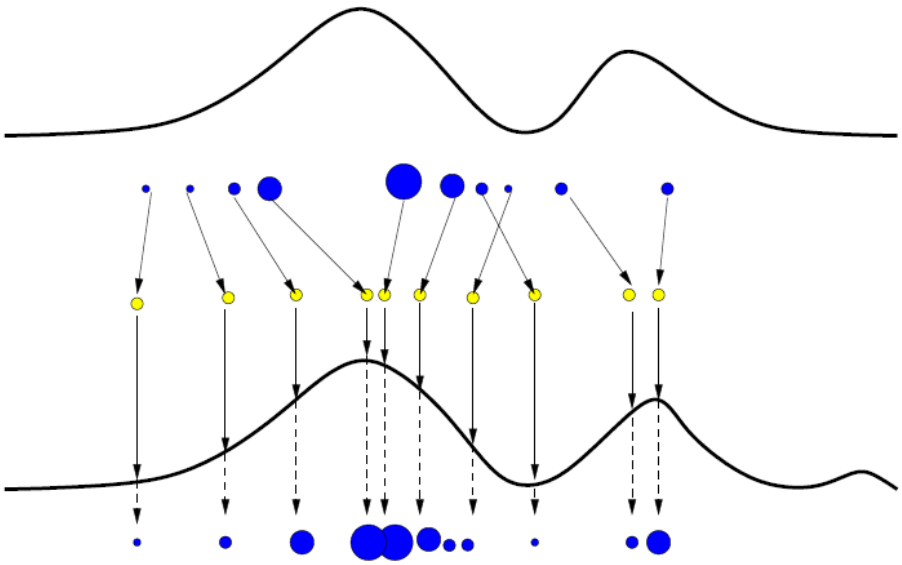
7. 统计之都 <LDA-math-MCMC 和 Gibbs Sampling>

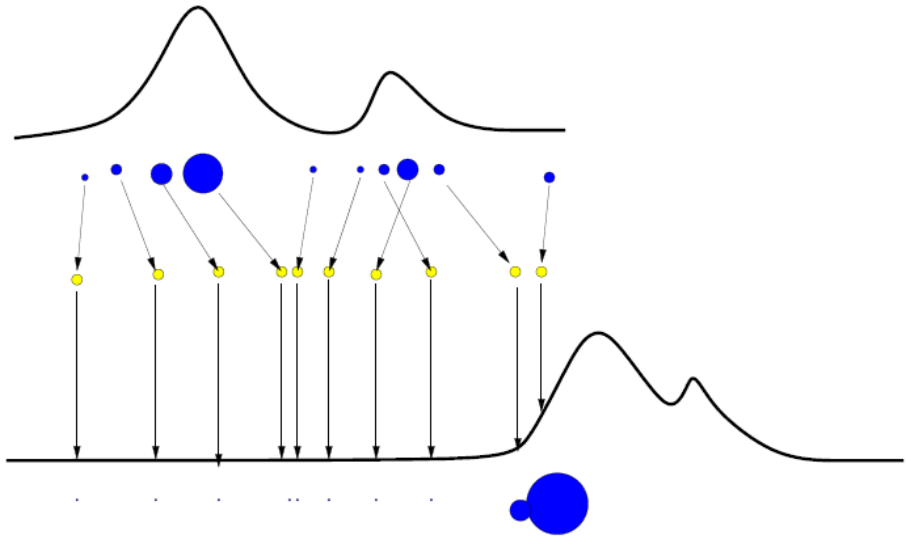
Particle Filter Tutorial 粒子滤波：从推导到应用（三）

粒子滤波重采样

五、重采样

在应用 SIS 滤波的过程中，存在一个退化的问题。就是经过几次迭代以后，很多粒子的权重都变得很小，可以忽略了，只有少数粒子的权重比较大。并且粒子权值的方差随着时间增大，状态空间中的有效粒子数较少。随着无效采样粒子数目的增加，使得大量的计算浪费在对估计后验滤波概率分布几乎不起作用的粒子上，使得估计性能下降，如图所示。





通常采用有效粒子数来衡量粒子权值的退化程度，即

$$N_{eff} = N/(1+\mathrm{var}(w_k^{*(i)}))$$

$$w_k^{*(i)} = \frac{p(x_k^{(i)} \mid y_{1:k})}{q(x_k^{(i)} \mid x_{k-1}^{(i)}, y_{1:k})}$$

这个式子的含义是，有效粒子数越小，即权重的方差越大，也就是说权重大的和权重小的之间差距大，表明权值退化越严重。在实际计算中，有效粒子数可以近似为：

$$\hat{N}_{eff} \approx \frac{1}{\sum_{i=1}^{N}(w_k^{(i)})^2}$$

在进行序贯重要性采样时，若上式小于事先设定的某一阈值，则应当采取一些措施加以控制。克服序贯重要性采样算法权值退化现象最直接的方法是增加粒子数，而这会造成计算量的相应增加，影响计算的实时性。因此，一般采用以下两种途径：(1)选择合适的重要性概率密度函数；(2)在序贯重要性采样之后，采用重采样方法。

对于第一种方法：选取重要性概率密度函数的一个标准就是使得粒子权值的方差最小。关于这部分内容，还是推荐百度文库的那篇文章《粒子滤波理论》，他在这里也引申出来了几种不同的粒子滤波方法。

这里着重讲第二种方法，重采样。

重采样的思路是：既然那些权重小的不起作用了，那就不要了。要保持粒子数目不变，得用一些新的粒子来取代它们。找新粒子最简单的方法就是将权重大的粒子多复制几个出来，至于复制几个？那就在权重大的粒子里面让它们根据自己权重所占的比例去分配，也就是老大分身分得最多，老二分得次多，以此类推。下面以数学的形式来进行说明。

前面已经说明了求某种期望问题变成了这种加权和的形式：

$$p(x_k|y_{1:k}) = \sum_{i=1}^{N} w_k^{(i)} \delta(x_k - x_k^{(i)}) \quad (1)$$

通过重采样以后，希望表示成：

$$\tilde{p}(x_k|y_{1:k}) = \sum_{j=1}^{N} \frac{1}{N} \delta(x_k - x_k^{(j)}) = \sum_{i=1}^{N} \frac{n_i}{N} \delta(x_k - x_k^{(i)}) \quad (2)$$

，注意对比(1)和(2)。$x_k^{(i)}$ 是第 k 时刻的粒子。$x_k^{(j)}$ 是 k 时刻重采样以后的粒子。其中 n(i) 是指粒子$x_k^{(i)}$ 在产生新的粒子集$x_k^{(j)}$ 时被复制的次数。（2）式中第一个等号说明重采样以后，所有的粒子权重一样，都是 1/N，只是有的粒子多出现了 n(i)次。

思路有了，就看具体的操作方法了。在《On resampling algorithms for particle fi lters》 这篇 paper 里讲了四种重采样的方法。这四种方法大同小异。如果你接触过遗传算法的话，理解起来就很容易，就是遗传算法中那种轮盘赌的思想。

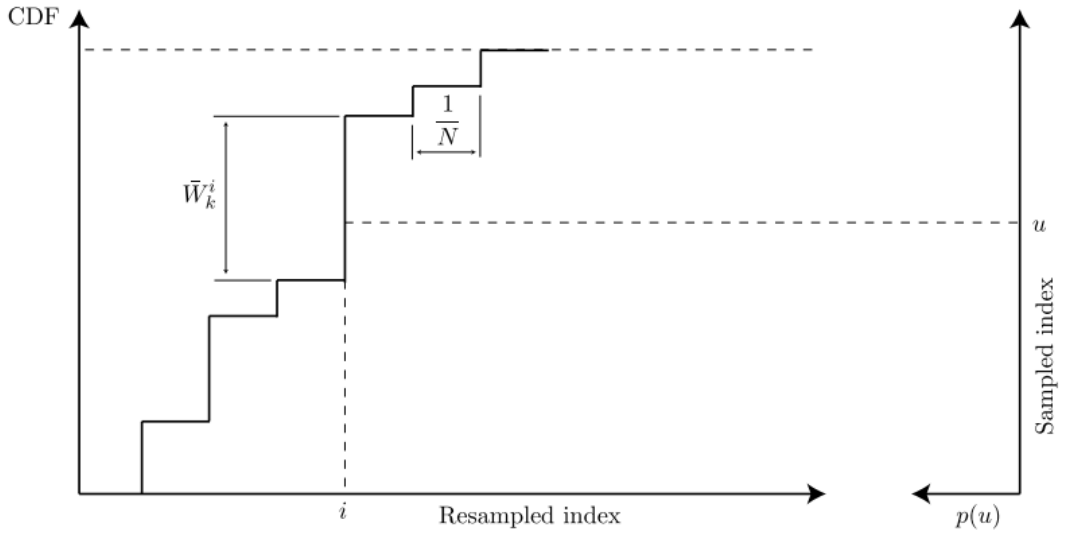
在这里用个简单的例子来说明：

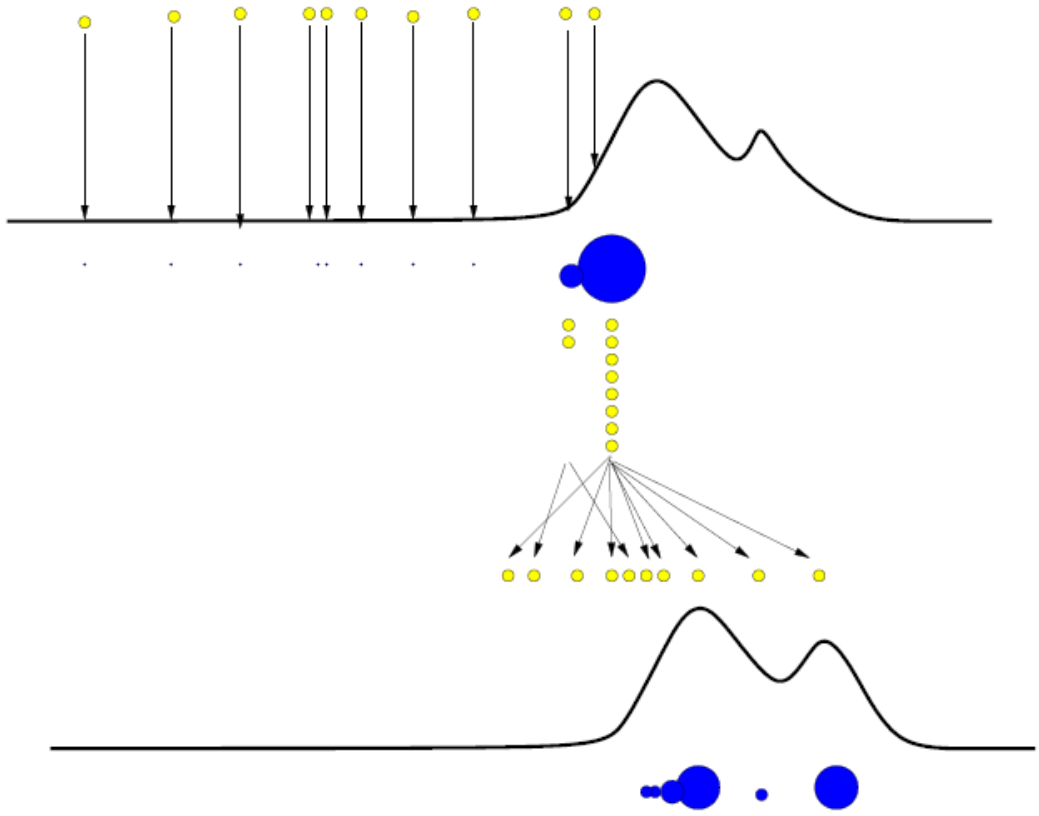
假设有 3 个粒子，在第 k 时刻的时候，他们的权重分别是 0.1, 0.1 ,0.8, 然后计算他们的概率累计和(matlab 中为 cumsum())得到： [0.1, 0.2, 1]。接着，我们用服从[0,1]之间的均匀分布随机采样 3 个值，假设为 0.15 , 0.38 和 0.54。也就是说，第二个粒子复制一次，第三个粒子复制两次。

在 MATLAB 中一句命令就可以方便的实现这个过程：

[~, j] = histc(rand(N,1), [0 cumsum(w')]); 关于 histc 的用法可以点击 histc 用法。

对于上面的过程，还可以对着下面的图加深理解：





将重采样的方法放入之前的 SIS 算法中，便形成了基本粒子滤波算法。

标准的粒子滤波算法流程为：

(1) 粒子集初始化，$k=0$：

对于$i=1,2,\cdots,N$，由先验$p(x_0)$生成采样粒子$\{x_0^{(i)}\}_{i=1}^N$

(2) 对于$k=1,2,\cdots$，循环执行以下步骤：

① 重要性采样：对于$i=1,2,\cdots,N$，从重要性概率密度中生成采样粒子$\{\tilde{x}_k^{(i)}\}_{i=1}^N$，计

算粒子权值$\tilde{w}_k^{(i)}$，并进行归一化；

② 重采样：对粒子集$\{\tilde{x}_k^{(i)},\tilde{w}_k^{(i)}\}$进行重采样，重采样后的粒子集为$\{x_k^{(i)},1/N\}$；

③ 输出：计算k时刻的状态估计值：$\hat{x}_k=\sum_{i=1}^{N}\tilde{x}_k^{(i)}\tilde{w}_k^{(i)}$。

重采样的思路很简单，但是当你仔细分析权重的计算公式时：

$$W_k(x_k)=\frac{p(y_{1:k}|x_k)p(x_k)}{q(x_k|y_{1:k})}\propto\frac{p(x_k|y_{1:k})}{q(x_k|y_{1:k})}$$

会有疑问，权重大的就多复制几次，这一定是正确的吗？权重大，如果是分子大，即后验概率大，那说明确实应该在后验概率大的地方多放几个粒子。但权重大也有可能是分母小造成的，这时候的分子也可能小，也就是实际的后验概率也可能小，这时候的大权重可能就没那么优秀了。何况，这种简单的重采样会使得粒子的多样性丢失，到最后可能都变成了只剩一种粒子的分身。在遗传算法中好歹还引入了变异来解决多样性的问题。当然，粒子滤波里也有专门的方法：正则粒子滤波，有兴趣的可以查阅相关资料。

至此，整个粒子滤波的流程已经清晰明朗了，在实际应用中还有一些不确定的就是重要性概率密度的选择。在下一章中，首先引出 SIR 粒子滤波，接着用 SIR 滤波来进行实践应用。

reference:

1. N. J. Gordon 《Beyond the Kalman Filter:Particle filters for tracking applications》
2. 百度文库《粒子滤波理论》
3. Jeroen D. Hol《On resampling algorithms for particle fi lters》
4. Particle filters: How to do resampling?
5. Gabriel A. Terejanu 《Tutorial on Monte Carlo Techniques》

.

六、Sampling Importance Resampling Filter (SIR)

SIR 滤波器很容易由前面的基本粒子滤波推导出来，只要对粒子的重要性概率密度函数做出特定的选择即可。在 SIR 中，选取：

$$q(x_k^{(i)}|x_{k-1}^{(i)}, y_k) = p(x_k^{(i)}|x_{k-1}^{(i)})$$

p(x(k)|x(k-1))这是先验概率，在第一章贝叶斯滤波预测部分已经说过怎么用状态方程来得到它。将这个式子代入到第二章 SIS 推导出的权重公式中：

$$w_k^{(i)} \propto w_{k-1}^{(i)} \frac{p(y_k|x_k^{(i)})p(x_k^{(i)}|x_{k-1}^{(i)})}{q(x_k^{(i)}|x_{k-1}^{(i)}, y_k)}$$

得到：

$$w_k^{(i)} \propto w_{k-1}^{(i)} p(y_k|x_k^{(i)}) \qquad \text{(1)式}$$

由之前的重采样我们知道，实际上每次重采样以后，有$w_{k-1}^{(i)} = \frac{1}{N}$。

所以(1)式可以进一步简化成：

$$w_k^{(i)} \propto p\left(y_k|x_k^{(i)}\right)$$

这里又出来一个概率采样$p\left(y_k|x_k^i\right)$，实际怎么得到这个概率，程序里面又怎么去采样呢？

先搞清这个概率$p\left(y_k|x_k^i\right)$的含义，它表示在状态 x 出现的条件下，测量 y 出现的概率。在机器人定位里面就是，在机器人处于位姿 x 时，此时传感器数据 y 出现的概率。更简单的例子是，我要找到一个年龄是 14 岁的男孩（状态 x），身高为 170（测量 y）的概率。要知道 y 出现的概率，需要知道此时 y 的分布。这里以第一篇文章的状态方程为例，由系统状态方程可知，测量是在真实值附近添加了一个高斯噪声。因此，y 的分布就是以真实测量值为均值，以噪声方差为方差的一个高斯分布。因此，权重的采样过程就是：当粒子处于 x 状态时，能够得到该粒子的测量 y。要知道这个测量 y 出现的概率，就只要把它放到以真实值为均值，噪声方差为方差的高斯分布里去计算就行了：

到这里，就可以看成 SIR 只和系统状态方程有关了，不用自己另外去设计概率密度函数，所以在很多程序中都是用的这种方法。

下面以伪代码的形式给出 SIR 滤波器：

----------SIR Particle Filter pseudo code----------

$$\left[\left\{x_k^{(i)}, w_k^{(i)}\right\}_{i=1}^{N}\right] = SIR\left[\left\{x_{k-1}^{(i)}, w_{k-1}^{(i)}\right\}_{i=1}^{N}, y_k\right]$$

- FOR i = 1:N

(1)采样粒子：$x_k^{(i)} \sim p\left(x_k | x_{k-1}^{(i)}\right)$

(2)计算粒子的权重：$w_k^{(i)} = p\left(y_k | x_k^{(i)}\right)$

- END FOR
- 计算粒子权重和，t=sum(w)
- 对每个粒子，用上面的权重和进行归一化,w = w/t
- 粒子有了，每个粒子权重有了，进行状态估计 $\sum_{i=1}^{N} \tilde{W}_k(x_k^{(i)}) f(x_k^{(i)})$
- 重采样

----------end ----------

在上面算法中，每进行一次，都必须重采样一次，这是由于权重的计算方式决定的。

分析上面算法中的采样，发现它并没有加入测量 y(k)。只是凭先验知识 p(x(k)|x(k-1))进行的采样，而不是用的修正了的后验概率。所以这种算法存在效率不高和对奇异点(outliers)敏感的问题。但不管怎样，SIR 确实简单易用。

七、粒子滤波的应用

在这里主要以第一章的状态方程作为例子进行演示。

$$x_k = \frac{x_{k-1}}{2} + \frac{25x_{k-1}}{1+x_{k-1}^2} + 8cos\left(1.2\left(k-1\right)\right) + v_k$$

$$y_k = \frac{x_k^2}{20} + n_k$$

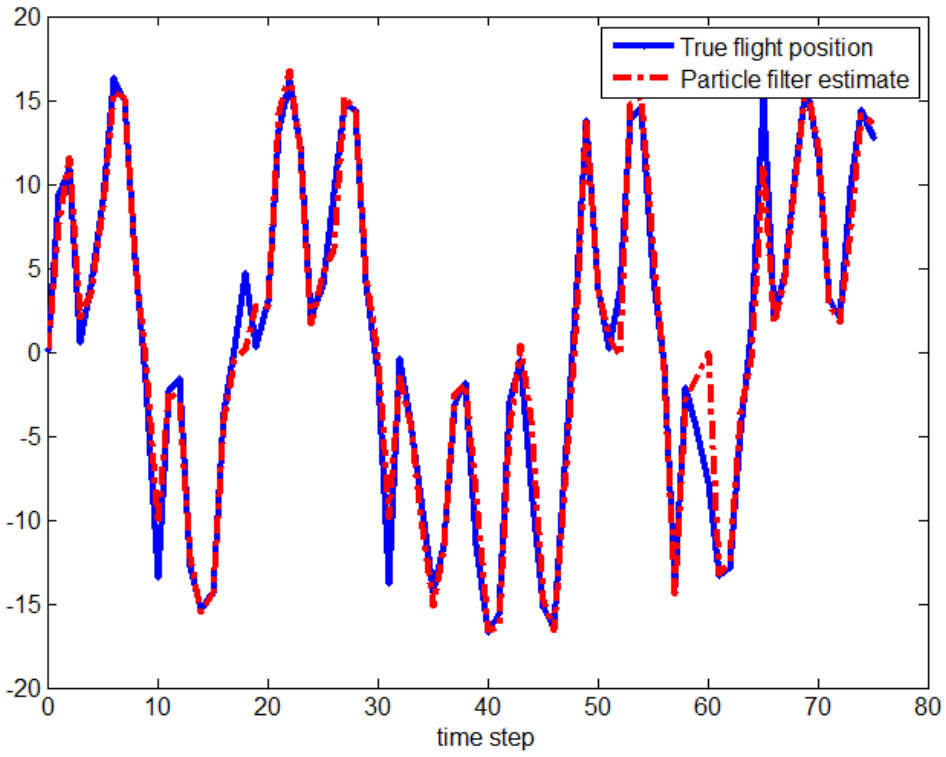
在这个存在过程噪声和量测噪声的系统中，估计状态 x(k)。

[python] view plaincopy

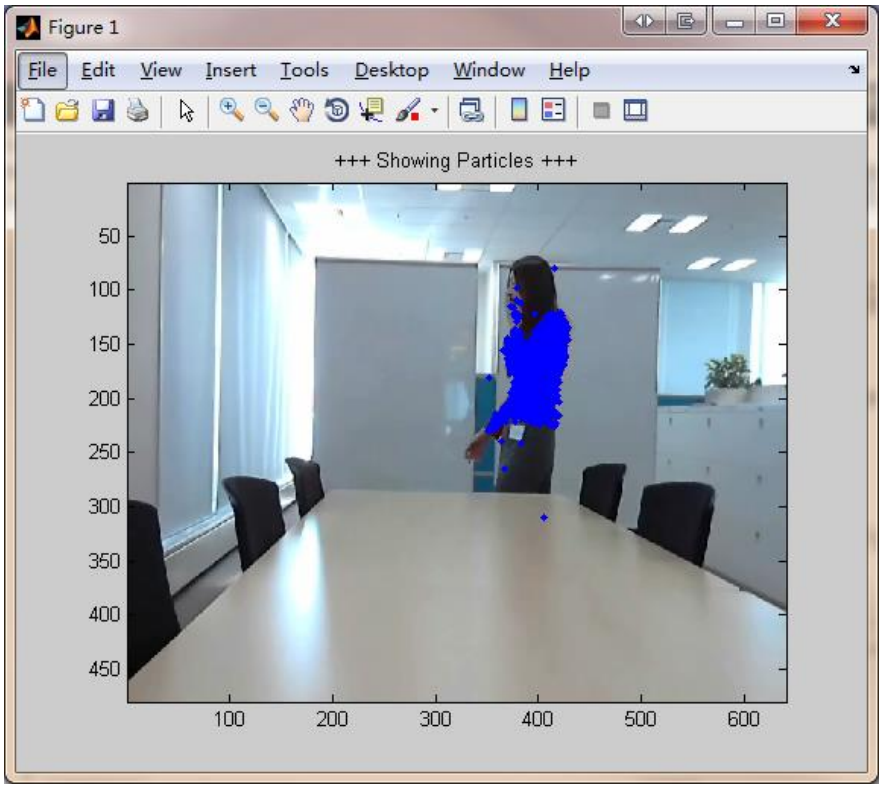
```
%% SIR 粒子滤波的应用，算法流程参见博客
http://blog.csdn.net/heyijia0327/article/details/40899819
clear all
close all
clc
%% initialize the variables
x = 0.1; % initial actual state
x_N = 1; % 系统过程噪声的协方差 (由于是一维的，这里就是方差)
x_R = 1; % 测量的协方差
T = 75;  % 共进行 75 次
N = 100; % 粒子数，越大效果越好，计算量也越大

%initilize our initial, prior particle distribution as a gaussian around
%the true initial value

V = 2; %初始分布的方差
x_P = []; % 粒子
% 用一个高斯分布随机的产生初始的粒子
for i = 1:N
    x_P(i) = x + sqrt(V) * randn;
end

z_out = [x^2 / 20 + sqrt(x_R) * randn];  %实际测量值
x_out = [x];  %the actual output vector for measurement values.
x_est = [x]; % time by time output of the particle filters estimate
x_est_out = [x_est]; % the vector of particle filter estimates.

for t = 1:T
    x = 0.5*x + 25*x/(1 + x^2) + 8*cos(1.2*(t-1)) +  sqrt(x_N)*randn;
    z = x^2/20 + sqrt(x_R)*randn;
    for i = 1:N
        %从先验 p(x(k)|x(k-1))中采样
        x_P_update(i) = 0.5*x_P(i) + 25*x_P(i)/(1 + x_P(i)^2) + 8*cos(1.2*(t-1)) + sqrt(x_N)*randn;
        %计算采样粒子的值，为后面根据似然去计算权重做铺垫
        z_update(i) = x_P_update(i)^2/20;
        %对每个粒子计算其权重，这里假设量测噪声是高斯分布。所以 w = p(y|x)对应下面的计算公式
        P_w(i) = (1/sqrt(2*pi*x_R)) * exp(-(z - z_update(i))^2/(2*x_R));
    end
    % 归一化.
    P_w = P_w./sum(P_w);

    %% Resampling 这里没有用博客里之前说的 histc 函数，不过目的和效果是一样的
```

```
    for i = 1 : N
        x_P(i) = x_P_update(find(rand <= cumsum(P_w),1));    % 粒子权重大的将多得到后代
    end                                                      % find( ,1) 返回第一个 符合前面条件的数的 下标

    %状态估计，重采样以后，每个粒子的权重都变成了 1/N
    x_est = mean(x_P);

    % Save data in arrays for later plotting
    x_out = [x_out x];
    z_out = [z_out z];
    x_est_out = [x_est_out x_est];

end

t = 0:T;
figure(1);
clf
plot(t, x_out, '.-b', t, x_est_out, '-.r','linewidth',3);
set(gca,'FontSize',12); set(gcf,'Color','White');
xlabel('time step'); ylabel('flight position');
legend('True flight position', 'Particle filter estimate');
```

滤波后的结果如下：



这是粒子滤波的一个应用，还有一个目标跟踪(matlab)，机器人定位(python)的例子，我一并放入压缩文件，供大家下载，[下载请点击]。(下载需要 1 个积分，下载完评论资源你就可以赚回那 1 积分，相当于没损失)。请原谅博主的这一点点自私。两个程序得效果如下：



粒子滤波从推导到应用这个系列到这里就结束了。结合前面几章的问题起来看，基本的粒子滤波里可改进的地方很多，正由于此才诞生了很多优化了的算法，而这篇博客只理顺了基本算法的思路，希望有帮到大家。

另外，个人感觉粒子滤波和遗传算法真是像极了。同时，如果你觉得这种用很多粒子来计算的方式效率低，在工程应用中不好接受，推荐看看无味卡尔曼滤波（UKF）,他是有选择的产生粒子，而不是盲目的随机产生。

ps:今年上半年的时候就很想写这篇博客了，但是每次看到那么多公式，实在有点不愿意去敲。到最近又用到，才下了决心，希望以后自己改了这个毛病，扎扎实实的做好每一步，不眼高手低。

reference：

1.M. Sanjeev Arulampalam 《A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking》