Particle Filtering for Sequential Bayesian Inference

Zhangir Azerbayev

Spring 2020

1 Problem Statement

In many applied domains we wish to do Bayesian inference from noisy measurements sequentially and on-line. The need to accomodate noisy measurements comes from measurement error or from our measurement being a stochastic function of the quantity of interest. We wish to do inference sequentially and on-line so that we can make use of data as it becomes available. For example, we may want to update a financial model according to live data or a robot to update its beliefs as it navigates an environment. Additionally, we may not want to store the entire history of data our model has seen, for example in high-throughput high-dimensional settings such as biomedical data.

We provide a general formulation of our problem as follows. Let $(x_t)_{t\in\mathbb{N}}$ with $x_t \in \mathbb{R}^n$ be a discrete-time continuous-state Markov process with transition kernel $p(x_t|x_{t-1})$ and prior $p(x_{0:t})$. The observations $(y_t)_{t\geq 1}$ are conditionally independent given (x_t) with known marginal distribution $p(y_t|x_t)$. Our goal is to solve the *filtering problem*; that is, to compute $p(x_{0:t}|y_{1:t})$. We are particularly interested in the marginal distribution $p(x_t|y_{1:t})$ and for some function $f_t(x_{0:t})$ the quantity

$$I(f_t) := \mathbb{E}_{p(x_{0:t}|y_{0:t})}[f_t(x_{0:t})].$$

Whenever I play tennis, I must resist the temptation to run up to the ball and smack it into the net. For now, let us make that mistake. Applying Bayes' rule, we see that the posterior distribution is given by

$$p(x_{0:t}|y_{1:t}) = \frac{p(y_{1:t}|x_{0:t})p(x_{0:t})}{\int p(y_{1:t}|x_{0:t})p(x_{0:t})dx_{0:t}}. \tag{1.1}$$

If we wish to estimate the posterior recursively, we get

$$\begin{aligned}
p(x_{0:t+1}|y_{1:t+1}) &= \frac{p(y_{1:t+1}|x_{0:t+1})p(x_{0:t+1})}{p(y_{1:t+1})} \\
&= p(y_{1:t}|x_{1:t})\frac{p(y_{t+1}|x_{t+1})p(x_{0:t+1})}{p(y_{1:t+1})} \\
&= \frac{p(x_{0:t}|p(y_{1:t})}{p(x_{1:t})}\frac{p(y_{t+1}|x_{t+1})p(x_{0:t+1})}{p(y_{1:t+1})} \\
&= p(x_{0:t}|y_{1:t})\frac{p(y_{t+1}|x_{t+1})p(x_{t+1})}{p(y_{t+1}|y_{1:t})} \\
&= p(x_{0:t}|y_{1:t})\frac{p(y_{t+1}|x_{t+1})p(x_{t+1})}{\int p(y_{t+1}|y_{1:t},x_{0:t+1})p(x_{0:t+1})dx_{0:t+1}} \\
&= p(x_{0:t}|y_{1:t})\frac{p(y_{t+1}|x_{t+1})p(x_{t+1})}{\int p(y_{t+1}|x_{t+1})p(x_{0:t+1})dx_{0:t+1}}
\end{aligned}$$

Here we run into a critical obstacle. Either method of computing $p(x_{0:t}|y_{0:t})$ from Bayes rule requires computing a normalizing constant that is conditional on $y_{1:t}$, which in turns requires integrating over $x_{0:t}$. This is a complicated and high-dimensional integral that is guaranteed to become intractable as t grows large. At this point, we may proceed in two ways.

First, we could try to make further assumptions about (x_t) in order to obtain an analytic solution. This leads to the *Kalman filter* [1]. In particular, we require Gaussian errors and linear state transitions. These assumptions are too strict for many applications.

The other approach is to find a way to approximate our high-dimensional integrals with an algorithm that does not scale with t. This leads to the powerful and general but compute-heavy method of *particle filtering* that is the focus of this article.

2 Particle Filtering

Particle filtering is a conceptually simple and powerful method for approaching the general filtering problem if one is willing to implement a compute-heavy Monte-Carlo simulation. The basic idea of particle filtering is as follows. The intractable integrals in section (1) become easy to approximate if we are able to take many samples from the posterior. In order to sample from the posterior, we introduce the technique of *sequential importance sampling* (SIS). In practice, SIS tends to converge to degenerate solutions, so we introduce a resampling trick that is enough to make our algorithm operationally effective.

2.1 Sequential Importance Sampling

Let us introduce a proposal function $\pi(x_{0:t}|y_{1:t})$, conceptually similar to the proposal function used in the Metropolis-Hastings algorithm. We define our proposal function based on our prior.

$$\pi(x_{0:t}|y_{1:t}) := p(x_{0:t}).$$

. Define the *importance weight* $w(x_{0:t})$ by

$$w(x_{0:t}) = \frac{p(x_{0:t}|y_{1:t})}{\pi(x_{0:t}|y_{1:t})}.$$

Then we have that

$$I(f_t) = \frac{\int f_t(x_{0:t})w(x_{0:t})\pi(x_{0:t}|y_{1:t})dx_{0:t}}{\int w(x_{0:t})\pi(x_{0:t}|y_{1:t})dx_{0:t}}.$$

Notice that

$$\int f_t(x_{0:t})w(x_{0:t})\pi(x_{0:t}|y_{1:t})dx_{0:t} = \mathbb{E}_{x_{0:t}\sim\pi(x_{0:t}|y_{1:t})}[f_t(x_{0:t})w(x_{0:t})]. \tag{2.1}$$

And similarly that

$$\int w(x_{0:t})\pi(x_{0:t}|y_{1:t})dx_{0:t} = \mathbb{E}_{x_{0:t}\sim\pi(x_{0:t}|y_{1:t})}[w(x_{0:t})]. \tag{2.2}$$

Now suppose we draw N iid samples $x_{0:t}^{(i)}$, called *particles*, from $\pi(x_{0:t}|y_{1:t})$. Thus, by the law of large numbers we can approximate 2.1 as

$$\frac{1}{N}\sum_{i=1}^{N} f_t\left(x_{0:t}^{(i)}\right) w\left(x_{0:t}^{(i)}\right).$$

and 2.2 as

$$\frac{1}{N}\sum_{i=1}^{N} w\left(x_{0:t}^{(i)}\right).$$

Thus we can obtain an estimate of $I(f_t)$ by

$$\hat{I}(f_t) = \frac{\frac{1}{N}\sum_{i=1}^{N} f_t(x_{0:t}^{(i)})w(x_{0:t}^{(i)})}{\frac{1}{N}\sum_{i=1}^{N} w(x_{0:t}^{(i)})} = \sum_{i=1}^{N} f_t\left(x_{0:t}^{(i)}\right)\tilde{w}_t^{(i)},$$

where $\tilde{w}_t^{(i)}$ are normalized importance weights. It remains to show how we update our importance weights. We do not know $p(x_{0:t}|y_{1:t})$, however from the definition of our importance weights we can see that

$$\tilde{w}_t^{(i)} \propto \tilde{w}_{t-1}^{(i)} \frac{p(y_t|x_t^{(i)})p(x_t^{(i)}|x_{t-1}^{(i)})}{\pi(x_t^{(i)}|x_{0:t-1}^{(i)}, y_{1:t})} = \tilde{w}_{t-1}^{(i)} p\left(y_t|x_t^{(i)}\right). \tag{2.3}$$

2.2 Resampling: The Bootstrap Filter

The SIS algorithm bypasses calculating high-dimensional integrals, however, it still scales poorly with respect to t. To gain an intuition for why, think of approximating $p(x_{0:t}|y_{1:t})$ as consisting of two steps:

1. Pick a good set of samples $\{x_{0:t}^{(i)}\}$.
2. Calculate their likelihoods $p(x_{0:t}^{(i)}|y_{1:t})$.

SIS allows to do (2), but does not help at all with (1) since we are still choosing our particles by sampling from the prior. Thus, for any fixed N, there is a large enough t such that $\{x_{0:t}^{(i)}\}_{i=1}^{N}$ is a sparse sample of the state space and is not guaranteed to include any high-likelihood points. But if we fix t and make N large, we still run into problems. Notice that if $p(x_{0:t}^{(i)}|y_{1:t}) \ll p(x_{0:t}^{(j)}|y_{1:t})$, then we should expect that $p(x_{0:k}^{(i)}|y_{1:k}) \ll p(x_{0:k}^{(j)}|y_{1:k})$ for all $t < k$ due to the recursive nature of our weight updates. In practice, when we run the SIS algorithms all but one of the importance weights rapidly converge to 0 (see accompanying code). In order to avoid this problem we need some way of resampling our particles so that low importance particles are discarded and replaced with high-likelihood particles.

There are many schemes for performing this resampling. We will consider the simplest: the *bootstrap filter*. In SIS, our approximation of $p(x_{0:t}|y_{1:t})$ is given by

$$p(x_{0:t}|y_{1:t}) = \sum_{i=1}^{N} \tilde{w}_i \delta_{x_{0:t}^{(i)}}(x_{0:t}),$$

where δ is the Dirac delta function. In the bootstrap filter, we will do away with importance weights as a part of our model (even though they are still used in intermediate computations). Instead, we approximate $p(x_{0:t}|y_{1:t})$ as

$$p(x_{0:t}|y_{1:t}) = \frac{1}{N} \sum_{i=1}^{N} \delta_{x_{0:t}^{(i)}}(x_{0:t}),$$

where the $\{x_{0:t}^{(i)}\}$ are now sampled from $p(x_{0:t}|y_{1:t})$ instead of the prior. To compute such $\{x_{0:t}^{(i)}\}$, we modify the SIS algorithm with a *selection step*, The selection step resamples particles so that those inconsistent with $y_{1:t}$ are discarded and those consistent with $y_{1:t}$ are multiplied. In precise terms, The bootstrap filter's Bayesian update at time t is as follows:

1. *Importance sampling step*
   - For each $i = 1, \ldots, N$, sample $\tilde{x}_t^{(i)} \sim p(x_t|x_{t-1})$ and let $\tilde{x}_{0:t}^{(i)} = (x_{0:t}^{(i)}, \tilde{x}_t^{(i)})$.
   - Compute normalized importance weights (cf. equation 2.3)

$$\tilde{w}_t^{(i)} \propto p(y_t|\tilde{x}_t^{(i)}).$$

2. *Selection step*

   - Resample with replacement N particles $x_{0:t}^{(i)}$ from $\{\tilde{x}_{0:t}^{(i)}\}$ according to their importance weights.

In the accompanying Python code, we see that this method is effective in drawing diverse samples from $p(x_{0:t}|y_{1:t})$.

3 Further Reading

This expository article roughly follows chapter 1 of [2]. The full text is a canonical reference for particle filtering and sequential Monte Carlo. See section II for a number of convergence theorems concerning algorithms of the kind we discuss. See section III for a discussion of more advanced particle filtering algorithms, particularly the resample-move algorithm.

Probabilistic programming is a programming paradigm designed to make expressing complex statistical inference easy and natural. For an implementation of particle filtering in a probabilistic programming language, see [3]. For an example of how particle filtering is used in modern large-scale statistical inference problems, see [4].

The bearings-only tracking problem discussed in the accompanying code is described in [5].

References

[1] B. Anderson and J. Moore, *Optimal Filtering*. Prentice-Hall, 1979.

[2] A. Doucet, N. Freitas, K. Murphy, and S. Russell, "Sequential monte carlo methods in practice," 01 2013.

[3] "Tutorial: Particle filtering in gen." `https://www.gen.dev/tutorials/particle-filtering-in-gen/Particle\%20Filtering\%20in\%20Gen`.

[4] M. Berke, M. Belledonne, Z. Azerbayev, and J. Jara-Ettinger, "Learning a metacognition for object detection," 2021.

[5] W. R. Gilks and C. Berzuini, "Following a moving target—monte carlo inference for dynamic bayesian models," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 1, pp. 127–146, 2001.