Particle Filters

Alex Pan and Alec Kosik

Abstract

Particle Filters allow us to model non-linear and non-Gaussian distributions through Monte Carlo methods. We first discuss Hidden Markov Models, and then discuss a recursive formulation of the distribution of $p(x_t|y_{1:t})$ and how this leads well to an algorithmic approach. Finally, we investigate Kalman Filters, Importance Sampling, and Sequential Importance Resampling. We discuss the motivations for and the relevant problems with each approach.

1 Introduction

Imagine that we have some outside information about a system and that we can generate observations which are dependent on the state of the system. However, the actual state of the system is hidden from us. How can we go about finding the hidden state of the system using our observations? It turns out we can approximate the hidden state through a process called **particle filtering**.

To give a toy example of this, imagine that we have a robot in a room. We know the layout of the room, but we do not know the position of our robot. However, the robot is able to take some sort of measurements with a sensor, for example its distance to walls. Using the sensor data and our existing information about the layout of the room, we use particle filtering to make predictions about where we could be in the room.

We will begin by formally defining the problem setting and then deriving particle filtering algorithms.

2 Hidden Markov Models

A **Hidden Markov Model** (HMM) is defined as a Markov process with states that cannot be observed, but with outputs that are dependent on the hidden state of the system. If we let the random variable X_i represent the i-th state of the system, and the random variable Y_i represent the i-th output of the system, we have:

$$X_1 \sim \mu(x_1) \text{ and } X_t|(X_{t-1} = x_{t-1}) \sim f(x_t|x_{t-1}) \tag{1}$$

$$Y_t|(X_t = x_t) \sim g(y_t|x_t) \tag{2}$$

where X_1 is defined by some initial conditions $\mu(x_1)$. Note how f obeys the Markov property, so the current state x_t is dependent only on the previous state x_{t-1}. Also note that g shows that the current observation y_t is only dependent on the current state x_t.

Let $x_{1:t}$ represent the ordered t-tuple of hidden states $(x_1, \ldots, x_t)$ where x_i is the i-th hidden state. Similarly, let $y_{1:t}$ represent the ordered t-tuple of observations $(y_1, \ldots, y_t)$. There are a few important distributions that follow directly from our HMM setting.

$$\begin{aligned} p(x_{1:t}) &= p(x_1)p(x_2|x_1)p(x_3|x_2, x_1)\ldots p(x_t|x_{t-1}, \ldots, x_1) \\ &= p(x_1)p(x_2|x_1)p(x_3|x_2)\ldots p(x_t|x_{t-1}) \\ &= \mu(x_1)\prod_{i=2}^{t} f(x_i|x_{i-1}) \end{aligned} \tag{3}$$

$$\begin{aligned} p(y_{1:t}|x_{1:t}) &= p(y_1|x_{1:t})p(y_2|x_{1:t}, y_1)(y_3|x_{1:t}, y_2, y_1)\ldots(y_t|x_{1:t}, y_{t-1}, \ldots, y_1) \\ &= p(y_1|x_1)p(y_2|x_2)\ldots(y_t|x_t) \\ &= \prod_{i=1}^{t} g(y_i|x_i) \end{aligned} \tag{4}$$

3 Problem Setup

3.1 General Goals

There are a few different uses of the particle filtering method. We could model the distribution $p(x_{1:t}|y_{1:t})$, the distribution $p(x_t|y_{1:t})$, or even the expected value of functions defined on these distributions. Here, our main goal is to use our observations to determine the hidden state of the system. That is, we want to figure out $p(x_t|y_{1:t})$. This is recognizable as the marginal of $p(x_{1:t}|y_{1:t})$.

$$p(x_t|y_{1:t}) = \int p(x_{1:t}|y_{1:t})dx_{1:t-1} \tag{5}$$

Setting our goal as a marginal of $p(x_{1:t}|y_{1:t})$ is helpful because we can refactor the full conditional distribution to see how it comes from our prior distributions derived in (3) and (4). Refactoring and using our initial HMM conditions, we can show:

$$\begin{aligned} p(x_{1:t}|y_{1:t}) &= \frac{p(x_{1:t}, y_{1:t})}{p(y_{1:t})} \\ &= \frac{p(x_{1:t})p(y_{1:t}|x_{1:t})}{p(y_{1:t})} \\ &= \frac{p(x_{1:t})p(y_{1:t}|x_{1:t})}{\int p(x_{1:t}, y_{1:t})dx_{1:t}} \\ &= \frac{\mu(x_1)\prod_{i=2}^{t} f(x_1|x_{i-1})\prod_{i=1}^{t} g(y_i|x_i)}{\int \mu(x_1)\prod_{i=2}^{t} f(x_1|x_{i-1})\prod_{i=1}^{t} g(y_i|x_i)dx_{1:t}} \end{aligned} \tag{6}$$

Thus,

$$p(x_t|y_{1:t}) = \int \frac{\mu(x_1)\prod_{i=2}^{t} f(x_1|x_{i-1})\prod_{i=1}^{t} g(y_i|x_i)}{\int \mu(x_1)\prod_{i=2}^{t} f(x_1|x_{i-1})\prod_{i=1}^{t} g(y_i|x_i)dx_{1:t}} dx_{1:t-1} \tag{7}$$

This looks really messy, but it's helpful to see that this is just integration of functions we already know: f and g. Depending on whether these integrals are analytically tractable, which is not always the case, we can obtain the goal distribution $p(x_t|y_{1:t})$ from (7).

3.2 A Recursive Formulation

In cases where the integrals in (7) are tractable, we can evaluate them. However, it would be really inefficient if we had to evaluate (7) at every time step. Instead, we can formulate this recursively and see how some of the values carry through. This recursiveness is described in two parts: the marginal prediction equation and the marginal update equation.

3.2.1 Marginal Prediction Equation

We now show how to derive the prediction equation:

$$p(x_t|y_{1:t-1}) = \int f(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})dx_{t-1} \tag{8}$$

If we approach this as a marginal by integrating out x_{t-1}:

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

We can make use of the Markov property to see:

$$p(x_t|x_{t-1}, y_{1:t-1}) = p(x_t|x_{t-1}) = f(x_t|x_{t-1})$$

3.2.2 Marginal Update Equation

$$p(x_t|y_{1:t}) = \frac{p(x_t|y_{1:t-1})p(y_t|x_t)}{p(y_t|y_{1:t-1})} \tag{9}$$

The derivation will be broken into two parts. First, we will show that (9) holds iff $p(x_t, y_{1:t}) = p(x_t, y_{1:t-1})p(y_t|x_t)$ holds. Then we will show that $p(x_t, y_{1:t}) = p(x_t, y_{1:t-1})p(y_t|x_t)$. We start by applying Bayes' rule to the left side of (9) and then moving the denominator to the right side:

$$p(x_t, y_{1:t}) = \frac{p(x_t|y_{1:t-1})p(y_t|x_t)}{p(y_t|y_{1:t-1})}p(y_{1:t})$$

Applying Bayes' rule to $p(x_t|y_{1:t-1})$:

$$\begin{aligned}
&= \frac{p(x_t, y_{1:t-1})p(y_t|x_t)}{p(y_t|y_{1:t-1})p(y_{1:t-1})}p(y_{1:t}) \\
&= \frac{p(x_t, y_{1:t})p(y_t|x_t)}{p(y_{1:t})}p(y_{1:t}) \\
&= p(x_t, y_{1:t})p(y_t|x_t)
\end{aligned}$$

Second, we show $p(x_t, y_{1:t}) = p(x_t, y_{1:t-1})p(y_t|x_t)$. Let's start by refactoring the right-side.

$$\begin{aligned}
p(x_t, y_{1:t-1})p(y_t|x_t) &= p(x_t|y_{1:t-1})p(y_{1:t-1})\frac{p(x_t, y_t)}{p(x_t)} \\
&= p(x_t|y_{1:t-1})p(y_{1:t-1})\frac{p(x_t|y_t)p(y_t)}{p(x_t)}
\end{aligned}$$

Using the fact that: $p(x|yz) = \frac{p(y)p(z)}{p(yz)}\frac{p(x|y)p(x|z)}{p(x)}$ Let: $x = x_t, y = y_{1:t-1}, z = y_t$

We can now see that: $p(x_t|y_{1:t-1}, y_t) = \frac{p(y_{1:t-1})p(y_t)}{p(y_{1:t-1}, y_t)}\frac{p(x_t|y_{1:t-1})p(x_t|y_t)}{p(x_t)}$

Thus,

$$\begin{aligned}
p(x_t|y_{1:t-1})p(y_{1:t-1})\frac{p(x_t|y_t)p(y_t)}{p(x_t)} &= p(x_t|y_{1:t})p(y_{1:t}) \\
&= p(x_t, y_{1:t})
\end{aligned}$$

The equations (8) and (9) give us what we will call the *marginal prediction* and *marginal update* steps respectively. They also make clear an algorithmic approach: by repeating these steps at each time t, we can find $p(x_t|y_{1:t})$ for any t, from our initial state up to our last state. This is also far more computationally efficient because we can save time by using information we already have from the previous iteration, instead of trying to perform a large integration at every single step. Unfortunately, when the integral in the prediction step is not solvable analytically we cannot recursively calculate it. In the Subsection 3.3 we will briefly describe a situation in which we can recursively calculate the marginal prediction and marginal update steps. Then in Subsection 3.4 we will describe the more general particle filtering method for use in cases where the marginal prediction and marginal update equations are not recursively calculable.

3.3 Kalman Filter

Setting different kinds of constraints on the problem leads to different kinds of solutions. It turns out that when $p(x_{t-1}|y_{1:t-1})$ is Gaussian, we can show that $p(x_t|y_{1:t})$ is Gaussian provided that f and g are linear functions on their inputs. In cases when these conditions hold, we can use the **Kalman Filter** to find $p(x_t, y_{1:t})$.

Essentially the Kalman Filter reformulates everything in terms of matrices since f and g are linear. An important thing to note is that the Kalman Filter typically extends our problem setting to include noise. That is to say, our transition and observation distributions both have noise parameters. In a physical setting, this might look something like the wind blowing on the robot while it moves, which might affect its location as it moves, and uncertainty in our sensor data, which is what we would expect in any real life measurement.

We can then reformulate x_t and y_t as a system of matrix equations, and also show that (8) and (9) turn out to be Gaussian distributions. From there, we can derive another set of equations that give us the Kalman Filter, but we won't go through a statement or proof of these equations because some of the linear algebra involved goes outside the scope of this discussion.

3.4 Particle Filtering

When the integral in the marginal prediction step is not analytically calculable, our recursive formulation of the marginal prediction and marginal update steps is intractable. That is, although (8) and (9) look recursive, they are not because the integral in (8) does not decompose. In such cases, we will turn back to equation (6) and simulate that distribution, rewritten as:

$$\begin{aligned} p(x_{1:t}|y_{1:t}) &= \frac{p(x_{1:t})p(y_{1:t}|x_{1:t})}{\int p(x_{1:t}, y_{1:t})dx_{1:t}} \\ &= \frac{\mu(x_1)\prod_{i=2}^{t} f(x_1|x_{i-1})\prod_{i=1}^{t} g(y_i|x_i)}{p(y_{1:t})} \\ &= p(x_{1:t-1}, y_{1:t-1})\frac{f(x_t|x_{t-1})g(y_t|x_t)}{p(y_t|y_{1:t-1})} \\ &\propto p(x_{1:t-1}, y_{1:t-1})f(x_t|x_{t-1})g(y_t|x_t) \end{aligned} \tag{10}$$

This equation is easily calculable because we always know f and g and we know $p(x_{1:t-1}, y_{1:t-1})$ recursively. We do not know $p(y_t|y_{1:t-1})$, but it can be ignored since we only care about the probability of a given path of states $x_{1:t}$ relative to the probability of other states—hence the $\propto$. In this scenario we have the following prediction and update steps[1]:

$$\text{Prediction step: } p(x_{1:t}|y_{1:t-1}) = f(x_t|x_{t-1})p(x_{1:t-1}|y_{1:t-1}) \tag{11}$$

$$\text{Update step: } p(x_{1:t}|y_{1:t}) \propto g(y_t|x_t)p(x_{1:t}|y_{1:t-1}) \tag{12}$$

Let's assess our current situation. We want to find the distribution for $p(x_t|y_{1:t})$. We found a calculable equation for $p(x_{1:t}|y_{1:t})$ and noticed that our goal is just a marginal of this. We tried integrating $p(x_{1:t}|y_{1:t})$ over $x_{1:t-1}$ to get our marginal but our best expression of this integral is only recursively calculable under certain constraints. When $p(x_{1:t}|y_{1:t})$ does not abide by these constraints we have no analytical solution and thus turn to a *simulation* method called particle filtering. In the next section we will explicate a Monte Carlo method that will help us model $p(x_{1:t}|y_{1:t})$ by recursively simulating the distributions of (11) and (12) at each time step.

[1]These prediction and update steps are more common in the particle filtering literature since the marginal prediction and marginal update steps are computationally expensive in non-linear, non-gaussian contexts

4 Monte Carlo Methods

In general, *not just in the particle filtering case*, we can model any distribution $\pi(x_{1:n})$ by sampling from π so that $x_{1:n}^{(i)} \sim \pi(x_{1:n})$ for $i = 1, \ldots, N$, and then assigning each sample a probability of $1/N$. For example, if $x_{1:n}^{(i)} = x_{1:n}^{(j)} = (x_1, \ldots, x_n)$ then the probability that $X_{1:n} = (x_1, \ldots, x_n)$ is $2/N$. We will read $x_{1:n}^{(i)}$ as the i-th sample, on the path $x_{1:n}$. Similarly, we would read $x_n^{(i)}$ as the i-th sample at the state x_n. Accordingly, we can represent our simulated model of π by:

$$\widehat{\pi}(x_{1:n}) = \begin{cases} 1/N & \text{at } x_{1:n}^{(1)} \\ \vdots & \\ 1/N & \text{at } x_{1:n}^{(N)} \end{cases} \tag{13}$$

We can then move towards an approximation of (11) and (12) by sampling from our transition distribution at each time step $x_t^{(i)} \sim f(x_t|x_{t-1})$ for $i = 1, \ldots, N$. Using these samples, we build an approximate distribution for the transition distribution where each particle i has probability equal to $1/N$. For now, we are only concerned with the probability of x_t based on the last state x_{t-1}.

$$\widehat{f}(x_t|x_{t-1}) = \begin{cases} 1/N & \text{at } x_t^{(1)} \\ \vdots & \\ 1/N & \text{at } x_t^{(N)} \end{cases} \tag{14}$$

As is, this method does not seem very helpful since we are modelling a distribution that we can sample from, so presumably, we already know the distribution analytically. In the next section we'll see why this simulation technique is important.

5 Importance Sampling & Sequential Importance Sampling

In this section we will develop the basic but not widely used method of particle filtering. In doing so, we deal with the full posterior $p(x_{1:t}|y_{1:t})$. However, simulating this distribution is the same as simulating our goal distribution, the marginal of the full posterior, $p(x_t|y_{1:t})$. The only difference is that the full posterior will be a t-dimensional model while our goal will just be a model of the t-th dimension.

5.1 Importance Sampling

Importance Sampling (IS) allows us to sample from a proposal distribution $q(x_{1:n})$, one we know e.g. a uniform or gaussian, and then weight samples from this proposal distribution to match the target distribution $t(x_{1:n})$ that we want to sample from. At first, this may sound like a trick—if we know the weight (difference) between the target and proposal, how could we not know the target—but if we consider our weight to be something like our function g, IS begins to make sense. Assuming we can approximate the prediction step by assigning each sample $x_{1:t}^{(i)}$ a probability $w_{1:t-1}^{(i)}$, which we will show in the next subsection, we can then approximate the update step as follows:

$$\widehat{p}(x_{1:t}|y_{1:t}) = \begin{cases} w_{1:t-1}^{(1)} \frac{g(y_t|x_t^{(1)})}{\sum_{j=1}^{N} w_{1:t-1}^{(j)} g(y_t|x_t^{(j)})} & \text{at } x_{1:t}^{(1)} \\ \vdots & \\ w_{1:t-1}^{(N)} \frac{g(y_t|x_t^{(N)})}{\sum_{j=1}^{N} w_{1:t-1}^{(j)} g(y_t|x_t^{(j)})} & \text{at } x_{1:t}^{(N)} \end{cases} \tag{15}$$

5.2 Sequential Importance Sampling

Our first particle filtering algorithm follows easily from the last two sections. We need to approximate the prediction and update distributions; we have already shown the prediction step. For each time step t we estimate the update distribution:

$$\widehat{p}(x_{1:t}|y_{1:t-1}) = \begin{cases} w_{1:t-1}^{(1)} & \text{at } x_{1:t}^{(1)} \\ \vdots & \\ w_{1:t-1}^{(N)} & \text{at } x_{1:t}^{(N)} \end{cases} \tag{16}$$

where $w_{1:t-1}^{(i)}$ are the normalized weights up to time t defined as:

$$w_{1:t-1}^{(i)} = \frac{w_{1:t-2}^{(i)} g(y_t|x_t^{(i)})}{\sum_{j=1}^{N} w_{1:t-2}^{(j)} g(y_t|x_t^{(j)})} \tag{17}$$

5.2.1 SIS Algorithm

Simulating a particle i up to time t, we are given a previous path $x_{1:t-1}^{(i)}$, weighted according to $w_{1:t-1}^{(i)}$. We then draw a current state $x_t^{(i)}$ from the transition distribution and add it to the current path $x_{1:t} = (x_{1:t-1}^{(i)}, x_t^{(i)})$. This completes the prediction step. To update, we multiply the probability of $x_{1:t} = x_{1:t}^{(i)}$, written $p(x_{1:t}|y_{1:t-1}) = w_{1:t-1}^{(i)}$, by $\frac{g(y_t|x_t)}{\sum_{i=1}^{N} g(y_k|x_k^{(i)})}$.

1. *Initialization:*

   For $i = 1, \ldots N$:

   Sample $x_0^{(i)} \sim \mu(x_0)$

   Assign weights $\widetilde{w}_0^{(i)} = g(y_0|x_0^{(i)})$

   Normalize weights $w_0^{(i)} = \frac{\widetilde{w}_0^{(i)}}{\sum_{i=1}^{n} \widetilde{w}_0^{(i)}}$

2. *Importance Sampling:*

   For $t = 1, \ldots, T$:

   Sample $x_t^{(i)} \sim f(x_t|x_{t-1}^i)$

   Assign weights $\widetilde{w}_t^{(i)} = g(y_t|x_t^{(i)})$

   Normalize weights $w_t^{(i)} = \frac{\widetilde{w}_t^{(i)}}{\sum_{i=1}^{n} \widetilde{w}_t^{(i)}}$

3. Return $\{x_T^{(i)}, w_T^{(i)}\}_{i=1}^{N}$

SIS is almost never used because of something called the "degeneracy problem". It does however, lay the groundwork for a widely used algorithm called SIR which we will discuss in the next section.

5.3 Degeneracy Problem

When we are sampling from our reweighted distribution, it is possible that certain points are assigned very low weight. In future iterations, this is likely to compound, making their weights essentially negligible. This manifests as a handful of points having nearly all the weight while the rest of our points have "degenerated". This is a problem because our distribution essentially is now defined by just a few of the original points we started with, which is not enough to get an accurate representation of our target distribution.

6 Bootstrap (SIR)

We can effectively solve the degeneracy problem by resampling. After each iteration $t > 0$ we will resample all N particles from the weighted distribution defined in the previous time step. Thus particles with low-to-negligible weights at t will have a low probability of persisting to $t + 1$. See Figure 1 for a visualization of this process.

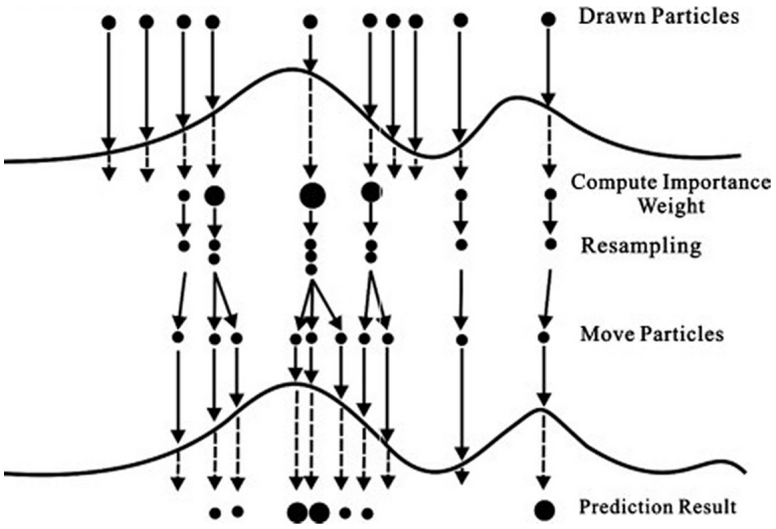

Figure 1: The SIR algorithm, adopted from [1].

6.1 Bootstrap Algorithm

1. *Initialization:*

   For $i = 1, \dots N$:

   Sample $x_0^{(i)} \sim \mu(x_0)$

   Assign weights $\widetilde{w}_0^{(i)} = g(y_0|x_0^{(i)})$

   Normalize weights $w_0^{(i)} = \frac{\widetilde{w}_0^{(i)}}{\sum_{i=1}^n \widetilde{w}_0^{(i)}}$

2. *Importance Sampling:*

   For $t = 1, \dots, T$:

   Sample $x_t^{(i)} \sim f(x_t|\widetilde{x}_{t-1}^i)$

   Assign weights $\widetilde{w}_t^{(i)} = g(y_t|x_t^{(i)})$

   Normalize weights $w_t^{(i)} = \frac{\widetilde{w}_t^{(i)}}{\sum_{i=1}^n \widetilde{w}_t^{(i)}}$

   Resample $\widetilde{x}_t^{(i)}$ from $x_t^{(i)}$ according to the weight distribution with replacement.

3. Return $\{x_T^{(i)}, w_T^{(i)}\}_{i=1}^N$

6.2 Sampling Impoverishment Problem

While this addresses the degeneracy problem, it turns out that resampling has it's own issue which we call the "sampling impoverishment problem". This comes about because our higher weighted points are more likely to be drawn multiple times. Instead of having many low-weight points spread out across the state-space—in the case of SIS and the degeneracy problem—we will have lots of evenly-weighted points concentrated around areas where high-weight points originally occurred. Intuitively, the degeneracy problem leaves you with a diverse scattering of point masses with a few ill-defined peaks while the sampling impoverishment problem leaves you with *only* well defined peaks. There are methods that address both of these issues but they require fancy sounding things like the "Epanechnikov Kernel", which our outside the scope of our discussion. It turns out that SIR is generally good enough and is the algorithm that people generally use.

7 Conclusion

References

[1] I. Steinruecken, Christian, "Advanced sampling." http://www.inference.phy.cam.ac.uk/tcs27/talks/sampling.html.

[2] G. Doucet, Arnaud, de Freitas, Nando, *Sequential Monte Carlo Methods in Practice*. Springer-Verlag New York, 2001.

[3] J. Doucet, Arnaud, "A tutorial on particle filtering and smoothing: Fifteen years later." http://www.stats.ox.ac.uk/~doucet/doucet_johansen_tutorialPF2011.pdf, 2008.

[4] E. Orhan, "Particle filtering." http://www.cns.nyu.edu/~eorhan/notes/particle-filtering.pdf, 2012.