Lecture Notes: Particle Filter

Putting together all the theory from recursive Bayesian estimation, Monte Carlo approximation, and sequential importance sampling, we can now describe the particle filter.

As with all other filters, the first step is to define the model that will be used in the problem. This model includes:

- X_t, a set of state variables
- A_t, the set of dynamic noises
- f(), the state transition equation
- Y_t, the set of measurements
- N_t, the set of measurement noises
- g(), the observation equation

The dynamic noise and measurement noise can be non-Gaussian, but they must be tractable. That means you must be able to write an equation specifying its distribution. There are variants of the basic particle filter algorithm that do not require this, but for the version presented here, we make that assumption.

The state transition equation and observation equation must also be tractable, but they can be non-linear, non-differentiable, piecewise, or just about anything so long as the equations can be written.

Since the particle filter is a Monte Carlo approximation, the distribution $p(x|y)$ is represented using a number of samples. In the context of the particle filter, the samples are usually called particles. They are denoted as:

$$\chi = \{x^{(m)}, w^{(m)}\}_{m=1}^{M} \tag{1}$$

where $x^{(m)}$ represents the state of particle m and $w^{(m)}$ represents the weight of particle m. A fidelity M must be chosen, in this case the number of particles used to represent the unknown distribution $p(x|y)$. A typical value for simple state spaces is $M = 100$ to $M = 1000$. The value chosen will depend upon the number of state variables and the complexity of $p(x|y)$; in other words, M should be larger as the dimensionality and shape complexity of $p(x|y)$ increases.

The particle filter can be initialized like all other filters. The state of each particle can be initialized to reasonable values using the first measurement, or some a priori knowledge, or zero if nothing else is known. It is important to remember that the state of a particle refers to a vector of state variables. Thus, the values of the state variables for a single particle may be initialized to different values, or all set to zero. The weight of each particle should be initialized to $\frac{1}{M}$. It is important to remember that the weight of a particle is a single value; there is not a weight associated with each state variable of a particle.

Like other filters, the particle filter algorithm follows a predict-update cycle, as follows:

1. Each particle m is propagated through the state transition equation:

$$\{x_t^{(m)} = f(x_{t-1}^{(m)}, a_t^{(m)})\}_{m=1}^{M} \tag{2}$$

   The value $a_t^{(m)}$ represents the dynamic noise from $t-1$ to t, and is randomly and independently calculated for each particle m. It may be envisioned as each particle taking a different "guess" at the dynamic noise undertaken for the current iteration.

2. Using the new measurement vector y_t, the weight for each particle is updated:

$$\tilde{w}_t^{(m)} = w_{t-1}^{(m)} \cdot p(y_t | x_t^{(m)}) \tag{3}$$

   This weight update equation is based upon selecting the importance distribution as the prior importance function. Other choices for the importance distribution lead to different formulations for the weight update equation. If this is not clear, please see the previous lecture notes on importance sampling for further information.

3. Normalize the updated weights, so they sum to 1:

$$w_t^{(m)} = \frac{\tilde{w}_t^{(m)}}{\sum_{m=1}^{M} \tilde{w}_t^{(m)}} \tag{4}$$

4. Compute the desired output, such as the expected value (mean):

$$E[x_t] \approx \sum_{m=1}^{M} x_t^{(m)} \cdot w_t^{(m)} \tag{5}$$

   Recall that the expected value may not be the desired output, but one can use similar methods to calculate local maxima or other values of interest.

5. Check if sampling is necessary, and if so, resample. This step is explained more below.

6. Let $t = t + 1$; iterate.

The net effect of this algorithm is that at each time step, the particles "swarm" to a distribution of possible new states. It is unknown where the system has actually gone, but the hope is that some of the particles have transitioned in a similar direction. An observation is then taken, and the weight of each particle is updated according to how well its predicted transition matches against the observation. Finally, the weights are renormalized to keep their sum equal to 1, so that they properly represent a probability distribution.

This algorithm forms the basic operation of the particle filter. However, a problem can occur. Some particles may wander away to the point that their weight approaches zero.

The more this happens, the smaller the number of particles there are that contribute to the approximation of the distribution. In order to determine if resampling is needed, the coefficient of variation statistic can be calculated:

$$\mathrm{CV} = \frac{\mathrm{VAR}(w^{(m)})}{E^2[w^{(m)}]} = \frac{\frac{1}{M}\sum_{m=1}^{M}\left(w^{(m)} - \frac{1}{M}\sum_{m=1}^{M} w^{(m)}\right)^2}{\left(\frac{1}{M}\sum_{m=1}^{M} w^{(m)}\right)^2} = \frac{1}{M}\sum_{m=1}^{M}(M \cdot w^{(m)} - 1)^2 \tag{6}$$

Note that although the time subscript has been omitted, it may be assumed that the equation refers to the weights after updating. The effective sample size can then be calculated as:

$$\mathrm{ESS} = \frac{M}{1 + \mathrm{CV}} \tag{7}$$

The effective sample size describes how many particles have an appreciable weight. In order to check if resampling is necessary, the effective sample size can be tested against the number of particles:

```
if (ESS < 0.5 M)
  resample
```

In this example, the threshold is set to 50% of the particles. For some problems, resampling every time step may be appropriate. However, as the number of particles grows, a lot of computations can be saved by resampling only when the ESS grows too small.

Resampling can be accomplished by a variety of methods. The most common resampling method is called **select with replacement**. The idea is to kill off particles with negligible weights, and replace them with copies of particles that have large weights. It works as follows:

```
Assume particle states in P[1...M], weights in W[1...M].

Q=cumsum(W);          calculate the running totals
t=rand(M+1);          t is an array of M+1 uniform random numbers 0 to 1
T=sort(t);            sort them smallest to largest
T[M+1]=1.0;           boundary condition for cumulative hist
i=j=1;                arrays start at 1
while (i<=M)
  if (T[i] < Q[j])
    Index[i]=j;
    i=i+1;
  else
    j=j+1;
  end if
end while
```

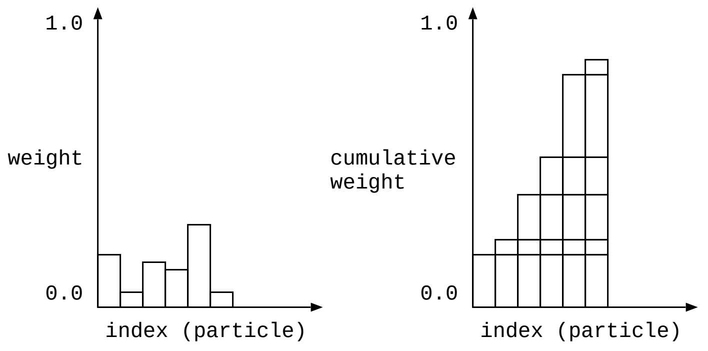

Figure 1: Using a cumulative weight distribution to select new particles.

```
loop (i=1; i<=M; i=i+1)
  NewP[i]=P[index[i]];
  NewW[i]=1/M;
end loop
```

This algorithm computes a list of indices of particles. The list may include 1 or more copies of the same index (particle). It may also skip over 1 or more indices (particles). After computing the list, it creates a new list of particles of equal weights.

In creating the list, particles with lower weights are less likely to get copied, and particles with higher weights are more likely to get multiple copies. This is accomplished by searching randomly in the sorted cumulative weight total. Figure 1 illustrates the process. Imagine picking a number between 0.0 and 1.0. Find that number on the Y-axis of the cumulative weight graph. Draw a horizontal line across the graph to where it touches a bar. The index of that bar is the index of the selected particle. Clearly, this process is more likely to pick indices of particles that had larger weights.

In summary, the particle filter is not one specific set of equations. It has many options that need to be specified, besides the usual selection of model variables and equations. These include:

- proposal distribution $q()$, which determines the weight update equation
- number of particles M (fidelity of approximation)
- when to resample
- resampling method

Next lecture, an example of the particle filter in operation will be demonstrated.