Statistical Techniques in Robotics (16-831, F12) Lecture#4 (Monday September 10)

Particle Filters and SIR

Lecturer: Drew Bagnell *Scribe:David Fouhey*[1]

1 Introduction

Last time, we discussed normalized importance sampling (NIS). In this lecture we will discuss Particle Filters. In particular, we will explain how they work, and the Good, the Bad, and the Ugly of them: their good aspects, bad aspects as well as fixes, and finally some ugly misconceptions.

1.1 NIS

Recall the normalized importance sampler from last time:

- Sample x_o^i from $p(x_0)$ for $i = 1 \ldots N$.
- Set $w_i = 1/N$.
- For $t = 1 \ldots$
  - For each i, sample x_t^i from $P(x_t \mid x_{t-1}^i)$
  - For each i, $w_i = w_i * p(z_t \mid x_t^i)$
  - Normalize weights
  - Compute expectation.

Unfortunately, as time progresses, most particles become useless since they do not match the observations. The weights then go towards either 0 (most particles) or 1 (the few particles that match the observation). Essentially, we are using particles to keep track of parts of the state space that we know are not likely.

1.2 Solution - Particle Filter (Sampling with Importance Resampling)

The solution is to do NIS, but to resample particles instead of changing the weights. This way, we do not waste particles on unlikely portions of the state space, but instead use our particles more intelligently.

Resampling proceeds as shown in Fig. 1: we select the particles with replacement according to their weight. Intuitively, we can think of this as lining the particles up on a dartbord that is 1 unit long, with width according to weight, and then throwing darts to select which ones to keep (possibly

[1]Some content adapted from previous scribes: Tommy Liu

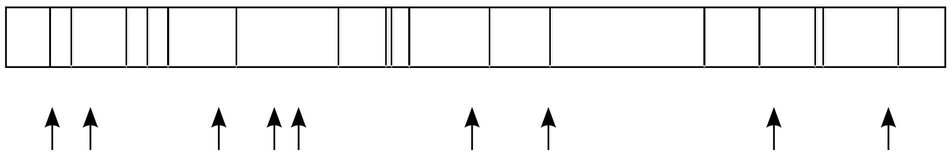


Figure 1: Resampling

taking two copies of one particle). Having selected the particles, we set all their weights to $1/N$, since otherwise we are double counting the weights.

This approach has a variety of names: SIR, Particle Filter, CONDENSATION, etc., and can be applied to a variety of problems. All that a particle filter entails is sampling particles, reweighting them according to a belief, normalizing the weights, and then resampling. Thus, while this lecture primarily deals with localization, particle filters can be applied to a great deal of other problems.

2 The Good

1. Can answer any query (in principle).
2. Will work for any distribution, including multi-modal distributions (unlike Kalman filters).
3. Scale well in computational resources: they are embarassingly parallel, except for the point in the algorithm when you need to do resampling.
4. Easy to implement (in principle).

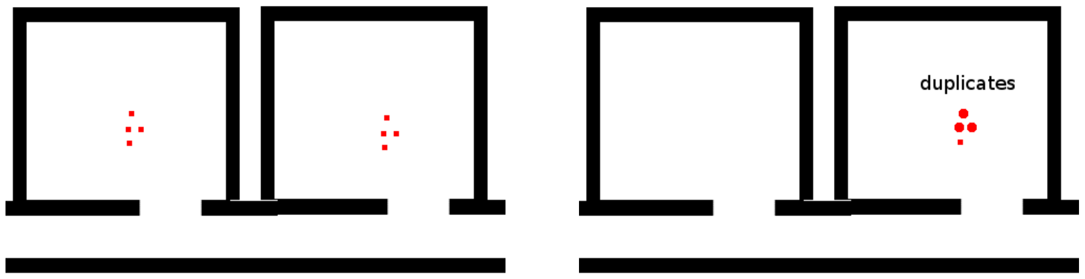
3 The Bad (And Fixes)

In this section, we will discuss some shortcomings of particle filters, particularly in their naïve implementation, and a variety of fixes that are crucial for getting them to work in practice. Many of these fixes are very simple tweaks to the naïve implementation, and should be implemented in practice regardless of whether the associated bad behavior is observed.

3.1 Lack of Diversity

Suppose we start out with a distribution of particles as seen in Fig. 2(a): our filter has converged to two identical rooms, and is unsure which room a robot is in. Now, suppose we let the filter run for 100 timesteps, and check back. We observe Fig. 2(b). Our robot has not moved and is getting the same observations; however, over the course of 100 timesteps, the filter has become certain that it is in the right room.

What has happened? The filter is non-deterministic, and so when we pick the particles with equal weight, it is unlikely that we will equal numbers from each room. As time progresses, this selection process only becomes more imbalanced: with observations equally likely, if a state has very few particles in its vicinity, in expection, it will have very few particles after resampling. Further, once a state space loses its particles, there are no ways to regain them without motion happening.



(a) Two rooms with equal probability (b) The same filter, after repeated resampling

Figure 2: A problem with assuming the independence of the observations and using naïve sampling. Without gaining any new information, the particle filter converges on one of the rooms.

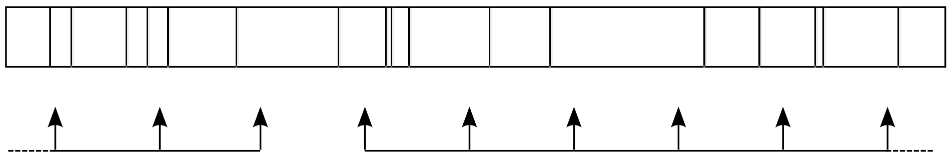


Figure 3: Low Variance Resampling; in practice, this approach is preferred over the naïve approach.

Fixes: There are a number of ways to prevent situations like this from happening in practice.

1. One way is to detect situations in which you do not have enough new information to resample, and only run the resampling when your observation model has sufficiently new observation data to process. One statistic that might be useful for evaluting this is looking at $\max(\{w_i\})/\min(\{w_i\})$, or looking at the entropy of the weights.
2. One easy fix is to run a more sophisticated resampling method, the low-variance resampler. In this approach, which is depicted in Fig. 3, a single number r is drawn, and then particles are selected at every $1/N$ units on the dartboard in our example. Effectively, the same sampler is used, but the random numbers generated are $r+i/N$ for all (positive and negative) i such that $r+i/N \in [0,1]$. This has desirable properties: for instance, every particle with normalized weight over $1/N$ is guaranteed to be selected at least once. Due to these properties and its ease of implementation, it is always recommended that you implement this sampler rather than the naïve approach. Note that this is also known as the "Stochastic Universal Sampler", in the genetic algorithms domain.
3. One easy, but undesirable approach is to just inject uniform samples. This should be treated as a last resort as it is unprincipled.

3.2 Particles are not independent

Unlike in the NIS, the particles are no longer independent. Therefore, while the Law of Large Numbers still holds, we do not have any convenient ways of relating how many particles are used to the accuracy. We therefore do not have concrete ways of determining how many particles to use.

Fix: One fix is to use KLD-Sampling, as described in

D. Fox. *Adapting the Sample Size in Particle Filters Through KLD-Sampling.* In IJRR 22:12, 2003.

Another approach is to use as many particles as one can keep track of without making the particle filter too slow; this is not ideal since it might use tens of thousands of particles to keep track of a tight, unimodal distribution, if the filter converges.

3.3 Problems with Great Observation Model

If the observation is really peaked (i.e., the one shown in Fig. 4, in which the peak is the correct location), then one might have issues in practice. In particular, if you draw samples uniformly along the line (i.e., with a motion model), then it is unlikely that the samples will hit the correct location. This is only one instance of a common problem with importance sampling: importance sampling does not work well when the proposal and importance distributions are diverge significantly.

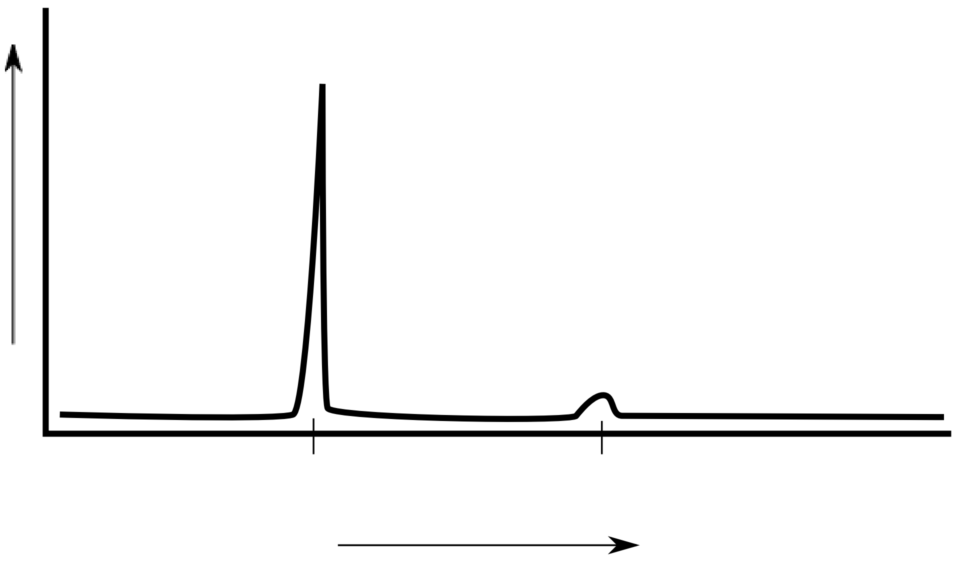


Figure 4: A very good observation model. Counterintuitively, this reliabiilty may pose problems for a vanilla implementation of a particle filter.

Fixes: There are a number of fixes for this problem, including more principled approaches and unprincipled and less effective hacks.

1. **Sample from a better model than the motion model:** the underlying problem is that the the proposal distribution (motion model) does not match the distribution that we want to sample from (observation model). The best, and most principled solution is to change the distribution. For instance, if one can sample directly from the observation model, one should do so. This is generally known as a dual particle filter.
2. **Rao-Blackwellization:** try to sample fewer dimensions of the distribution, and do the rest analytically. Basically, rely on sampling as little as possible.
3. **Squash the distribution:** take all of the probabilities to some power $1/p$ less than 1. When this is done, p observations count as 1 observation.

4. **More particles:** Sample more particles to ensure that that the peak is sampled (although this is unsatisfactory).

5. **Pretend your observation model is worse:** convolve the observation model with a Gaussian, and pretend that the resulting much less confident observation model is the actual model. This is a last resort, as it throws away a lot of valuable data.

3.4 Computationally expensive

Even though particle filters are parallel, good performance requires a lot of particles, and parallelization can only do so much.

Fix: If the distribution is unimodal, it makes more sense to use a Kalman Filter.

3.5 Non-deterministic

People are resistant to non-determinstic algorithms because they produce a distribution of behaviors given the same input rather than produce consistent output. This makes them difficult to predict and debug.

Fix: Use a fixed random seed, which will produce repeatable results.

3.6 Hard to Measure Performance

The particle filter does not provide a confidence along with its predictions. It can wander off and converge to a mode without recognizing that the mode is wrong.

4 The Ugly

Here are a few common misconceptions about particle filters.

1. **Particle Filter = Sample from Motion Model, Weight By Observation Model.** One can actually sample from any distribution, and frequently in practice will no sample from the motion model. In fact, one might sample directly from the observation model.

2. **Particle Filters are for localization.** In reality, can apply to any state estimation problem.

3. **Particle Filters are anything with samples.** This is wrong: NIS uses samples, but does not do resampling.