A Tutorial on Simple Particle Filters

Michael A. Goodrich

October 2, 2006

1 Introduction

Bayes rule is a very powerful tool for doing inference under conditions of uncertainty. In robotics, perhaps the most frequent use of Bayes rule is to translate sensor readings into a map of the environment. Since the sensor readings are noisy, it is desirable to use Bayes rule to accumulate enough noisy measurements to localize obstacles in the world. In this tutorial, I'll try to prepare you to use Bayes rule in the class lab to localize the corners of obstacles.

2 The Bayes Filter

For a robot, it is undesirable to wait until it has gathered a bunch of sensor readings before it applies Bayes rule to give it a clue about the environment. Since robots are usually made to move, it is a good idea to use the information as it comes in to incrementally improve our understanding of the environment. In this section, we'll discuss the general form of doing this.

2.1 A Simple Example of Sequential Estimation

Sequential estimation is the process of incrementally improving our understanding as data is gathered. The Bayes filter is a powerful tool for performing sequential estimation. Before we discuss the details of the Bayes filter, it might be helpful to see an easy-to-understand example of sequential estimation.

Suppose that we know that some physical process is described by a random variable, but that we aren't sure what this random variable looks like. The first piece of information that we want is the mean of this random variable. If we had the pdf for this random variable, $p_X(x)$ we could directly compute the mean as

$$\mu = \int x p_X(x) dx.$$

If we don't have the pdf, we can estimate the mean by collecting samples of the process. Suppose that we have n samples represented by the set

$\{x_1, x_2, \ldots, x_n\}$. The sample mean is given by

$$m = \frac{1}{n}\sum_{i=1}^{n} x_i.$$

The problem with this is that we have to wait until all n samples are collected before we can use the information. The trick is to come up with a difference equation that allows us to incrementally improve the estimate as data becomes available.

Naturally, if we only have one sample, then this is our best guess of the mean. So,

$$m_1 = x_1.$$

We can write m_k as any partial sum

$$m_k = \frac{1}{k}\sum_{i=1}^{k} x_i.$$

Our goal is to write this as some function of x_k and m_{k-1}. We can do this as follows:

$$\begin{aligned}
m_k &= \frac{1}{k}\sum_{i=1}^{k} x_i \\
&= \frac{1}{k}x_k + \frac{1}{k}\sum_{i=1}^{k-1} x_i \\
&= \frac{1}{k}x_k + \frac{1}{k}\frac{k-1}{k-1}\sum_{i=1}^{k-1} x_i \\
&= \frac{1}{k}x_k + \frac{k-1}{k}\Big(\frac{1}{k-1}\sum_{i=1}^{k-1} x_i\Big) \\
&= \frac{1}{k}x_k + \frac{k-1}{k}m_{k-1}.
\end{aligned}$$

This is the sequential estimate of the mean of a random variable.

Problem 1: Generate a MATLAB script that plots the sequential estimate for a series of samples generated from a gaussian distribution and compare it to the (non-sequential) sample mean:

```
x = randn(100,1)
m=zeros(100,1);
m(1,1) = x(1,1);
for (i=2:100)
    m(i,1) = 1/i*x(i,1) + (i-1)/i*m(i-1,1);
end;
plot(m);
set(line([1,100],[mean(x),mean(x)]),'color','m');
```

2.2 State Space

Another form of sequential estimation is to sequentially estimate posterior distributions as actions are taken and as observations are made. If we let $\mathbf{x}_k$ denote the state at time k, a_k denote the action taken at time k, and $\mathbf{z}_k$ denote the observation made at time k, then we can use models for how actions produce consequences and how states are observed to improve our estimates.

2.2.1 State Update Without Noise

For example, consider Newton's laws of motion in a line. Let $\mathbf{x}$ denote the position and velocity of an object moving along this line,

$$\mathbf{x} = \begin{bmatrix} p \\ \dot{p} \end{bmatrix}$$

where p denotes position and where $\dot{p} = \frac{dp}{dt}$ denotes velocity. Consider Newton's laws of motion under constant acceleration.

$$\begin{aligned} p(T) &= p(0) + T\dot{p}(0) + \frac{1}{2}\ddot{p}(0)T^2 \\ \dot{p}(T) &= \dot{p}(0) + T\ddot{p}(0) \end{aligned}$$

where $\ddot{p} = \frac{d^2p}{dt^2}$ denotes acceleration. Assume that force is the control that we get to apply to this system, and that we hold force constant for T seconds at a time. Then, *force = mass × acceleration* implies that $\ddot{p} = F/m$, and we can write a *state update equation* or *motion model* as

$$\begin{aligned} \mathbf{x}_k &= \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \mathbf{x}_{k-1} + \begin{bmatrix} \frac{T^2}{2m} \\ \frac{T}{m} \end{bmatrix} F_k \\ &= A\mathbf{x}_{k-1} + BF_k, \end{aligned}$$

where A is the corresponding 2×2 matrix and B is the corresponding 2×2 matrix. To get to the form of equation with a_k being the input at time k, we note that the action a_k is force so $a_k = F_k$. (Be careful to not confuse "a" for action with acceleration; I've used the second time derivative of position to denote acceleration.) Thus,

$$\mathbf{x}_k = A\mathbf{x}_{k-1} + Ba_k,$$

Problem 2. In MATLAB, set `F = sin(2*pi/30 * [1:100])`. For 100 steps, compute $\mathbf{x}_k$ given A and B above assuming that $T = 0.1$ and $m = 5$. Plot $\dot{p}$ versus p (which equal the first and second rows of x, respectively). You should see a circle. Why?

```
t=[1:1000];
a = 5*sin(2*pi*t/500);
x=zeros(2,length(t));
x0 = [0;-8];
```

```
T=0.1;m=5;
A = [1 T;0 1];
B = [T*T/(2*m); T/m];
for (i=1:length(t))
     if (i==1) x(:,i) = A*x0 + B*a(1,i);
     else x(:,i) = A*x(:,i-1) + B*a(1,i);
     end;
end;
plot(x(1,:),x(2,:));
```

2.2.2 State Update With Noise

Suppose that we modify the state update equation to include noise.

$$\mathbf{x}_k = A\mathbf{x}_{k-1} + Ba_k + \nu_k, \tag{1}$$

where $\nu_k \sim N(0, \Sigma_x)$, meaning that ν is distributed according to a normal distribution with zero mean and co-variance matrix equal to Σ_x.

Even though we began with a fixed initial state, when we added noise, we found that $\mathbf{x}_k$ can move all over the place. Our goal is to determine what will happen to the distribution of $\mathbf{x}_k$ over time. We'll consider four cases, beginning at the most simple and proceeding to the most complex.

Case 1. In case 1, suppose that we know the initial $\mathbf{x}_0$ exactly and $\Sigma_x = \mathbf{0}$, meaning that there is no noise. As the state evolves according to Equation (1), what does the distribution of $p_X(\mathbf{x}_k)$ look like? Since we know the exact initial state and since there is no noise, we know $\mathbf{x}_k$ exactly which implies that we also know the distribution of $\mathbf{x}_k$ exactly.

Case 2. In case 2, we don't know $\mathbf{x}_0$ exactly. Rather, we just know that $\mathbf{x}_0 \sim p_X(\mathbf{x}_k)$. Suppose again that $\Sigma_x = \mathbf{0}$, meaning that there is no noise. As the state evolves according to Equation (1), what does the distribution of $p_X(\mathbf{x}_k)$ look like? Since the variance in Equation (1) is zero, the shape of $p_X(\mathbf{x}_k)$ is the same as the shape of $p_X(\mathbf{x}_0)$; the only difference is that the mean of this distribution changes.

Case 3. As in case 1, suppose that we know the initial $\mathbf{x}_0$ exactly, but unlike case 1 suppose that $\Sigma_x \neq \mathbf{0}$. What does the distribution of $p_X(\mathbf{x}_k)$ look like? The mean of this distribution is always centered at $\mathbf{x}_k$, but the variance grows to infinity. Why? Because noise keeps getting added in, and this noise accumulates until we really have no idea what the actual value of $\mathbf{x}_k$ is.

Case 4. As in case 2, suppose that we only know the initial distribution of $\mathbf{x}_0$, and as in case 3 suppose that $\Sigma_x \neq \mathbf{0}$. What does the distribution of $p_X(\mathbf{x}_k)$ look like? The mean of this distribution is always centered at some point which is a function of the mean of $p(\mathbf{x}_0)$, but the variance grows to infinity.

PROBLEM 3. In the code segment from problem 2, change
`x(:,i) = A*x(:,i-1) + B*a(1,i)`
to
`x(:,i) = A*x(:,i-1) + B*a(1,i) + .1*randn(2,1)`.

How is `.1*randn(2,1)` related to Σ_x? Run the above code above several times (including noise) and describe what happens. Why does it happen?

2.2.3 Adding Observations

In general, we do not get to directly observe the state $\mathbf{x}$. Instead, we get some partial and corrupted version of the state. For example, suppose that our goal is to estimate the position and velocity of a robot. Then $\mathbf{x}$ is a vector of this position and velocity. Suppose that the only sensor on the robot is a GPS unit. This unit will give a noisy estimate of the position and no information at all about velocity.

We can model such a situation as follows:

$$\begin{aligned} \mathbf{z}_k &= \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \mathbf{x}_k + \eta_k \\ &= C\mathbf{x}_k + \eta_k \end{aligned}$$

where $\eta_k \sim N(0, \Sigma_z)$. The matrix C is our observation matrix. We've given it for the specific case of the GPS sensor, but it is easy to generalize it to more complicated sets of sensors.

If we know $\mathbf{x}_k$ exactly, how is $\mathbf{z}_k$ distributed? It is distributed $N(C\mathbf{x}_k, \Sigma_z)$ – the mean is simply the transformed value of $\mathbf{x}$.

In a later lecture, we'll present an algorithm that uses Bayes rule to combine noisy observations and noisy state updates to allow us to determine how $\mathbf{x}_k$ is distributed given that we know the initial distribution of $\mathbf{x}_0$ and that η and ν are gaussian. The remainder of this tutorial is to present an algorithm that allows us to estimate this distribution using a "digitized" version of the probability density functions. We'll begin by showing how we can apply Bayes rule to the problem.

2.3 Bayes Filter

Recall that Bayes rule gives us a way of estimating the posterior estimate of a state given a likelihood and a prior estimate. In general form, Bayes rule is given as

$$p(\mathbf{x}_k|\mathbf{z}_k) = \frac{p(\mathbf{z}_k|\mathbf{x}_k) \times prior}{p(\mathbf{z}_k)} \tag{2}$$

It quickly gets to be a pain to track the fraction on the right of the equation. Since we can always normalize after our computations to make $p(\mathbf{x}_k|\mathbf{z}_k)$ into a pdf, we will drop this normalizer and write

$$p(\mathbf{x}_k|\mathbf{z}_k) = \alpha p(\mathbf{z}_k|\mathbf{x}_k) \times prior$$

where α is the normalizer constant. The equation states that the posterior estimate of our state vector $\mathbf{x}_k$ after we have observe $\mathbf{z}_k$ is equal to the likelihood that a particular state generated the observation times the prior.

Recall that a sequential estimation problem is one where we begin with an initial guess of $\mathbf{x}_0$ and track it through a series of transitions $\mathbf{x}_k \rightarrow \mathbf{x}_{k+1}$ and a series of observations $\mathbf{z}_k$. Where do we get the prior estimate in a sequential estimation problem? The term "prior" in this context means our best guess of the distribution of $\mathbf{x}_k$ before we get the observation $\mathbf{z}_k$. Our best guess should use all previous observations and our knowledge of what control actions we previously chose. Thus, our prior is given by

$$prior = p(\mathbf{x}_k|\mathbf{z}_{k-1}, \mathbf{z}_{k-2}, \ldots, \mathbf{z}_1, a_k, a_{k-1}, \ldots, a_0).$$

Note how this equation takes all previous observations (up to time $k-1$ and all applied actions to generate a guess what $\mathbf{x}_k$ is without including the latest observation $\mathbf{z}_k$.

We now introduce some new terminology which we borrow from [1]. Let $\overline{bel}(\mathbf{x}_k)$ denote the prior distribution of the state. That is,

$$\overline{bel}(\mathbf{x}_k) = p(\mathbf{x}_k|\mathbf{z}_{k-1}, \mathbf{z}_{k-2}, \ldots, \mathbf{z}_1, a_k, a_{k-1}, \ldots, a_0).$$

Let $bel(\mathbf{x}_k)$ denote the posterior estimate of our state. Given this terminology, we can rewrite Bayes rule in Equation (2) as

$$bel(\mathbf{x}_k) = \alpha p(\mathbf{z}_k|\mathbf{x}_k)\overline{bel}(\mathbf{x}_k).$$

If we can obtain a good formula for $\overline{bel}(\mathbf{x}_k)$, then we can solve the above equation. This equation is known as the *Measurement* update equation – it represents how observing $\mathbf{z}_k$ changes our posterior into our prior.

We will obtain this formula using as an intermediate step the joint distribution of $\mathbf{x}_k$ and $\mathbf{x}_{k-1}$ conditioned on all previous observations and previous controls. Recall that the marginal distribution can be obtained from the joint distribution by "integrating out" unneeded variables. Thus,

$$\begin{aligned}
\overline{bel}(\mathbf{x}_k) &= p(\mathbf{x}_k|\mathbf{z}_{k-1}, \mathbf{z}_{k-2}, \ldots, \mathbf{z}_1, a_k, a_{k-1}, \ldots, a_0) \\
&= \int p(\mathbf{x}_k, \mathbf{x}_{k-1}|\mathbf{z}_{k-1}, \mathbf{z}_{k-2}, \ldots, \mathbf{z}_1, a_k, a_{k-1}, \ldots, a_0) d\mathbf{x}_{k-1}.
\end{aligned}$$

We'll now factor this using the observation that $p(A, B|C) = p(A|B, C)p(B|C)$ to get

$$\begin{aligned}
\overline{bel}(\mathbf{x}_k) &= \int p(\mathbf{x}_k, \mathbf{x}_{k-1}|\mathbf{z}_{k-1}, \mathbf{z}_{k-2}, \ldots, \mathbf{z}_1, a_k, a_{k-1}, \ldots, a_0) d\mathbf{x}_{k-1} \\
&= \int p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{z}_{k-1}, \mathbf{z}_{k-2}, \ldots, \mathbf{z}_1, a_k, a_{k-1}, \ldots, a_0) \times \\
&\qquad p(\mathbf{x}_{k-1}|\mathbf{z}_{k-1}, \mathbf{z}_{k-2}, \ldots, \mathbf{z}_1, a_k, a_{k-1}, \ldots, a_0) d\mathbf{x}_{k-1}.
\end{aligned}$$

Fortunately, we can simply this hideous equation. Consider the first term on the right hand side. It is the conditional distribution of the current state $\mathbf{x}_k$ given the previous state $\mathbf{x}_{k-1}$, all previous observations, and all controls. If we knew the previous state, than all previous observations are irrelevant (see Equation (1));

the conditional distribution of the current state is conditionally independent of previous observations given the previous state. Additionally, knowing the previous state makes all but the current control a_k irrelevant. (Recall that the current state depends on a_k – see Equation (1).) Thus, we can simplify

$$p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{z}_{k-1}, \mathbf{z}_{k-2}, \ldots, \mathbf{z}_1, a_k, a_{k-1}, \ldots, a_0) = p(\mathbf{x}_k|\mathbf{x}_{k-1}, a_k).$$

Similarly, $p(\mathbf{x}_{k-1}|\mathbf{z}_{k-1}, \mathbf{z}_{k-2}, \ldots, \mathbf{z}_1, a_k, a_{k-1}, \ldots, a_0)$ does not depend on a_{k-1} which means that

$$p(\mathbf{x}_{k-1}|\mathbf{z}_{k-1}, \mathbf{z}_{k-2}, \ldots, \mathbf{z}_1, a_k, a_{k-1}, \ldots, a_0) = p(\mathbf{x}_{k-1}|\mathbf{z}_{k-1}, \mathbf{z}_{k-2}, \ldots, \mathbf{z}_1, a_{k-1}, \ldots, a_0).$$

But this ugly term on the right is simply $bel(\mathbf{x}_{k-1}$. Thus,

$$\overline{bel}(\mathbf{x}_k) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1}, a_k) bel(\mathbf{x}_{k-1}) d\mathbf{x}_{k-1}.$$

We call this equation the *Prediction* update; it represents what we think the current state will be given our estimate of the previous state and what action we apply.

To summarize, the Bayes filter combines the *Prediction* update with the *Measurement* update

$$\overline{bel}(\mathbf{x}_k) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1}, a_k) bel(\mathbf{x}_{k-1}) d\mathbf{x}_{k-1} \quad (3)$$

$$bel(\mathbf{x}_k) = \alpha p(\mathbf{z}_k|\mathbf{x}_k) \overline{bel}(\mathbf{x}_k). \quad (4)$$

3 Particle Filters

If the relationship between state updates and observations is linear (i.e., uses the matrices A, B, and C), and if the noises are gaussian, then Equations (3)-(4) can be solved in closed form. We'll derive some of this later in the semester. Before we do so, we want to solve a harder problem using an approximate technique. The harder problem assumes that the noise is not gaussian and that the state update and observation are not linear. Thus, the problem we are trying to solve is

$$\begin{aligned} \mathbf{x}_k &= f(\mathbf{x}_{k-1}, a_{k-1}) + \nu_k \\ \mathbf{z}_k &= g(\mathbf{x}_k) + \eta_k, \end{aligned}$$

where neither ν nor η are gaussian.

I've run out or time. I'll finish this tutorial later. To compensate, I'll distribute section 4.3 from [1] in class. I'll also distribute a MATLAB example of using the particle filter.

3.1 Representation

3.2 Prediction

3.3 Measurement

3.4 Resampling

References

[1] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, Cambridge, Massachusetts, USA, 2005.