The Bayes Filter

Josiah Hanna

1 Notes on Notation

1. We will write $p(\cdot|y)$ to denote the conditional distribution over some random variable given that we have observed y.

2. We will write $x \sim p(\cdot|y)$ to denote that variable x has its outcome distributed according to the conditional distribution, $p(\cdot|y)$.

2 Robot Interaction Model

We will formalize a robot interacting with its environment as a discrete-time process. At time $t = 0$, the environment is in an initial state, x_0. The robot chooses a control action, u_1, and then the environment transitions to state $x_1 \sim p(\cdot|x_0, u_1)$. The robot then receives an observation, $z_1 \sim g(\cdot|x_1)$. The process then repeats with the robot choosing a new action. We will refer to the full sequence of states up to and including time t as $x_{0:t}$ and similarly for actions, $u_{1:t}$ and observations, $z_{1:t}$. Following common notation (e.g., in the textbook *Probabilistic Robotics*, the state index begins at $t = 0$ and control and observation indices begin at $t = 1$.

3 State Estimation

Robots only perceive observations from their sensors. In general, observations provide incomplete and potentially noisy information about the state of the world. However, robots need to infer the state of the world to make good decisions.

Why does the robot need to estimate state to make good decisions? A single observation is likely missing some critical information. For example, autonomous vehicles cannot observe pedestrians that are currently behind other vehicles even if their presence is known. An alternative approach to state estimation is to attempt to reason based on the full history of observations and controls. However, the number of possible observation/control histories is exponential in the time-step t and so will quickly become intractable to base decision-making on. The state strikes a balance between these extremes by summarizing the useful parts of the observation history into a compact representation containing all aspects that affect the future.

A finals reason that state must be estimated is that it is constantly changing as the robot moves about its environment. The state may evolve if the robot takes action or not. Choosing not to act is another type of action.

Moreover, we want to be probabilistic and infer a belief distribution over states instead of simply guessing a single state as the current one. Doing so allows the robot to represent uncertainty about the state of the world.

4 Probabilistic State Estimation

A naïve approach to state estimation might simply estimate which state is most likely given the robot's observations and actions up to time t. This approach is limited as it chooses a single state with certainty and thus does not allow the robot to represent uncertainty about the state of the world. Instead, we will estimate a *belief distribution*, which is a probability distribution over possible states in the world given everything that has been previously observed. We will write $\texttt{bel}(x_t)$ to denote the probability of state x_t under this belief distribution. Formally, $\texttt{bel}(x_t)$ is the posterior $\Pr(x_t|z_{1:t}, u_{1:t})$ This belief factors in all available information up to and including step t. We will find it useful to also refer to the belief, $\overline{\texttt{bel}}(x_t)$, which we define as $\Pr(x_t|z_{1:t-1}, u_{1:t}$ or the posterior distribution of x_t before the final observation has been received and factored in.

4.1 Naive Belief Computation

Bayes rule gives us a straightforward – but computationally complex – means to compute the robot's belief at time t. Using Bayes Rule, we obtain:

$$\texttt{bel}(x_t) = \Pr(x_t|z_{1:t}, u_{1:t}) = \eta \sum_{x_{1:t-1}} \Pr(x_{1:t}|u_{1:t}) \Pr(z_{1:t}|x_{1:t}),$$

where $\sum_{x_{1:t-1}}$ is a summation over all possible state sequences from time 1 to time $t-1$ and η is a constant that ensures we have a valid probability distribution. With full knowledge of the state and observation probabilities, we can compute $\Pr(x_{1:t}|u_{1:t}) = p(x_1) \prod_{i=2}^{t} p(x_i|x_{i-1}, u_{i-1})$ and $\Pr(z_{1:t}|x_{1:t}) = \prod_{i=1}^{t} g(z_i|x_i)$.

Unfortunately, this approach to belief computation becomes intractable as the interaction sequence grows because the summation above will have an exponential number of terms. It is also memory-intensive as the robot would have to keep around the entire sequence of observations and past control actions in order to update its belief at each moment in time.

5 Recursive Belief Computation

The key idea of the Bayes filter is to update the robot's belief recursively as each new control is taken and a new observation is received. The Bayes filter requires two steps to update the current belief to a belief that incorporates u_t and z_t.

1. **Prediction.** The robot computes a new belief, $\overline{\texttt{bel}}(x_t)$ by predicting the effect of u_t on $\texttt{bel}(x_{t-1})$. In general, the step *increases* the Rob robot's uncertainty about its true state.

2. **Correction.** The robot uses the information in the new observation, z_t, to correct its prediction. This step amounts to applying Bayesian inference with $\overline{\texttt{bel}}(x_t)$ as the prior. In general, this step *decreases* the robot's uncertainty about its true state.

Algorithm 1 provides pseudocode for the complete belief update of the Bayes filter. When used sequentially on robot data, $u_{1:t}$ and $z_{1:t}$, the Bayes filter algorithm will compute the same posterior $\texttt{bel}(x_t)$ as the direct use of Bayes rule on all of the data at once. The proof of this fact is left as an exercise to the reader or office hours.

Algorithm 1 Bayes Filter

Input: Previous belief $\texttt{bel}(x_{t-1})$, control input u_t, measurement z_t
Output: Updated belief $\texttt{bel}(x_t)$
procedure BayesFilter($\texttt{bel}(x_{t-1}), u_t, z_t$)
    // Prediction step
    for all x_t **do**
        $\overline{\texttt{bel}}(x_t) \leftarrow \sum_{x_{t-1}} p(x_t \mid u_t, x_{t-1}) \cdot \texttt{bel}(x_{t-1})$
    end for
    // Correction step
    for all x_t **do**
        $\texttt{bel}(x_t) \leftarrow \eta \cdot g(z_t \mid x_t) \cdot \overline{\texttt{bel}}(x_t)$ // η is the normalizing constant.
    end for
    return $\texttt{bel}(x_t)$
end procedure

6 Limitations of the Bayes Filter

There are several limitations of the Bayes filter and it is often more of theoretical interest than real application. In the following weeks, we will introduce the Kalman filter family and Particle filters which are significantly more applicable to real robots than Bayes filters.

In particular, the Bayes filter has the following limitations:

1. The computational complexity of exact summation. The prediction step often involves summing over the entire state space or integrating if the state space is continuous. Doing so is computationally infeasible for high-dimensional problems. The same limitation applies to computing the normalization constant, η, in the update step. The high computational complexity limits scalability and may make the filter too slow to use for real-time belief updates.

2. Markov assumption. The validity of the Bayes filter relies upon the Markov assumption, i.e., that $p(x_t|x_{t-1}, u_t) = p(x_t|x_{0:t-1}, u_{1:t})$. This assumption is unlikely to hold in practice.

3. Assumption of known models. The Bayes filter requires accurate knowledge of the robot's state transition model, p, and sensor model, g. Errors or approximations in these models can lead to poor performance.

4. Scalability. The Bayes filter is difficult to implement with large or high dimensional state spaces. Approximations can be used but these may make it more likely the Markov assumption is violated.

7 Bayes Smoother

In some applications, we are not only interested in $\texttt{bel}(x_t)$ for the current time t but in $\texttt{bel}(x_t)$ for all t from $t = 0$ to some $T = T$. The Bayes filter can almost solve this problem as it produces a belief for each step $\texttt{bel}(x_t)$. However, it is not the optimal approach as it neglects to use information available in $z_{t:T}$ to refine the belief $\texttt{bel}(x_t)$. To see why observations after t can help infer x_t, consider listening to a person talking in a noisy room. You are unsure if they just said "The price will rise" or "The prize will rise" due to the similar sound of "price" and "prize." As they continue talking, you hear "due to inflation" and so you infer that the more likely utterance is "The price will rise due to inflation." In this case, you have used the most recent information to refine a belief about something that happened in the past.

The Bayes filter only considers information available up to time t when inferring $\texttt{bel}(x_t)$. To incorporate later observations, we will turn to the Bayes smoother. The Bayes smoother is a post-processing procedure that is ran over the sequence of beliefs computed by the Bayes filter. Pseudocode is provided in Algorithm 2.

Algorithm 2 Bayes Smoother

Input: Beliefs $\texttt{bel}(x_t)$ for $t = 1 \ldots T$, motion model $p(x_{t+1} \mid x_t, u_{t+1})$
Output: Smoothed beliefs $\texttt{bel}'(x_t)$ for $t = 1 \ldots T$
procedure BayesSmoother($\texttt{bel}(x_t), u_t$)
    // Initialization: Start with the final belief
    $\texttt{bel}'(x_T) \leftarrow \texttt{bel}(x_T)$
    // Backward smoothing step
    for $t = T - 1$ **to** 1 **do**
        for all x_t **do**
            $\texttt{bel}'(x_t) \leftarrow \texttt{bel}(x_t) \cdot \sum_{x_{t+1}} \frac{p(x_{t+1}|x_t,u_{t+1}) \cdot \texttt{bel}'(x_{t+1})}{\texttt{bel}(x_{t+1})}$
        end for
    end for
    return $\texttt{bel}'(x_t)$ for $t = 1 \ldots T$
end procedure