Study Notes for AI (CS 440/520)
Lecture 15: Kalman and Particle Filters - Intro to Utility Theory

Corresponding Book Chapters: 15.3-15.4-16.2-16.3-16.4
Note: These notes provide only a short summary and some highlights of the material covered in the corresponding lecture based on notes collected from students. Make sure you check the corresponding chapters. Please report any mistakes in or any other issues with the notes to the instructor.

1 Kalman Filter in the context of Motion Estimation

We can approach Motion Tracking problems in continuous spaces using Gaussian distributions, which can be represented using a small finite set of parameters. Consider all of the variables needed in order to track an object moving through a three dimensional space. For most problems, we can track an object easily enough with the six variables (x, y, z), $(\dot{x}, \dot{y}, \dot{z})$, where the second triple represents a vector for the object's velocities (translational and rotational). For motion tracking problems we still require the prior probability, the transition and observation models in order to be able to apply the equations we have provided for filtering, smoothing, etc. In the context of motion tracking:

- The *transition model* describes the physics of motion.
- The *observation model* describes the measurement process.

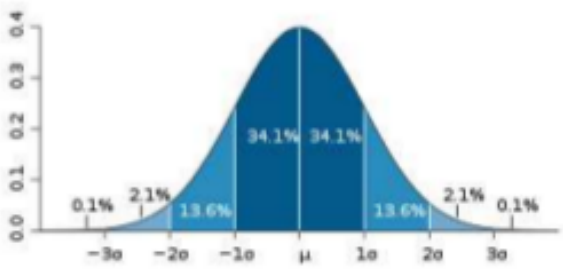


Figure 1: A Gaussian distribution. The darker color is less than one standard deviation from the mean. For the normal distribution, this accounts for about 68% of the set (dark blue), while two standard deviations from the mean (medium and dark blue) account for about 95% and three standard deviations (light, medium, and dark blue) account for about 99.7%.

The Kalman filter approach selects a Gaussian distribution to represent the true belief distribution (see Figure 1). The advantage of a Gaussian distribution is that it can be represented (for multi-dimensional problems) just by

its mean vector μ and its covariance matrix Σ. This means that if the state has n variables: $\{X_1, X_2, \ldots, X_n\}$, then the mean is a vector of n values that stores the mean values for the variables:

$$\mu = \begin{bmatrix}\mu_1 & \mu_2 & \ldots \mu_n\end{bmatrix}^T = \begin{bmatrix}\bar{X}_1 \\ \bar{X}_2 \\ \ldots \\ \bar{X}_n\end{bmatrix}$$

The covariance matrix Σ stores at each location i, j of the matrix the covariance between variables X_i and X_j:

$$cov(X_i, X_j) = E[(X_i - \mu_i)(X_j - \mu_j)]$$

Covariance is a measure of how much two variables change together. If two variables tend to vary together (that is, when one of them is above its expected value, then the other variable tends to be above its expected value, too), then the covariance matrix between the two variables will be positive. Variance is a special case of the covariance, when the two variables are identical. This means that along the diagonal of the covariance matrix, we actually store the variance of the state variables. Overall:

$$\Sigma = \begin{bmatrix} E[(X_1 - \mu_1)^2] & E[(X_1 - \mu_1)(X_2 - \mu_2)] & \ldots & E[(X_1 - \mu_1)(X_n - \mu_n)] \\ E[(X_2 - \mu_2)(X_1 - \mu_1)] & E[(X_2 - \mu_2)^2] & \ldots & E[(X_2 - \mu_2)(X_n - \mu_n)] \\ \ldots & \ldots & \ldots & \ldots \\ E[(X_n - \mu_n)(X_1 - \mu_1)] & E[(X_n - \mu_n)(X_2 - \mu_2)] & \ldots & E[(X_n - \mu_n)^2] \end{bmatrix}$$

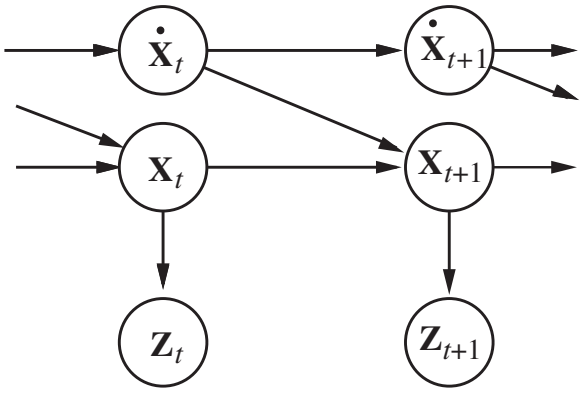


Figure 2: Solving target tracking problems might require estimating the derivatives of the state variables. For example, if X_t is the position of a moving agent in the world, we might have to compute $\dot{X}_t$, the agent's velocity, to properly answer queries. The above dynamic Bayesian network represents the dependencies in the context of target tracking. The variable Z_t corresponds to the evidences.

Notice that the matrix is symmetric and actually it is also positive-definite, which simplifies a lot the computations involved in the Kalman filter operations.

Why is, however, the Kalman filter a useful approach for temporal probabilistic estimation? The Kalman filter is closed under the standard Bayesian network operations, such as filtering. In particular the following properties hold for the Kalman filter:

1. If the current distribution $P(X_t|e_{1:t})$ is Gaussian and the transition model $P(X_{t+1}|X_t)$ is linear Gaussian then the one-step prediction:

$$\underbrace{P(X_{t+1}|e_{1:t})}_{\text{is also Gaussian}} = \int_{X_t} P(X_{t+1}|X_t)P(X_t|e_{1:t})dt$$

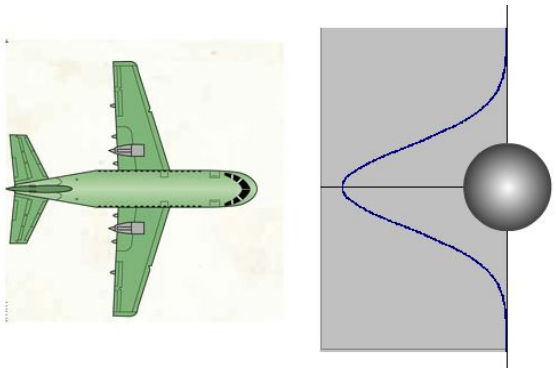


Figure 3: A plane flying towards an obstacle.

2. If the predicted distribution $P(X_{t+1}|e_{1:t})$ is Gaussian and the observation model $P(e_{t+1}|X_{t+1})$ is linear Gaussian then the updated distribution:

$$\underbrace{P(X_{t+1}|e_{1:t+1})}_{\text{is also Gaussian}} = \alpha \cdot P(e_{t+1}|X_{t+1}) \cdot P(X_{t+1}|e_{1:t})$$

Given the above properties, the Kalman filter can be applied for an infinite number of steps to problems that involve only linear Gaussian transition and observation models, because under these assumptions the true probability distribution function is always a Gaussian distribution.

Why is this property so important? Except very few cases, as under the above assumptions, filtering with continuous networks generates state distributions whose representation grows without bound over time. This means that in the general case we need an infinite number of variables to represent belief distribution over continuous state variables.

On top of this advantage, the actual equations of the Kalman filter (not described here) are the optimum solution to the filtering problem in continuous space under the above assumptions.

Thus, the question arises: Are there any drawbacks in the Kalman filter approach? The answer is yes. Unfortunately, the assumptions required for the Kalman filter to be closed under Bayesian operations and optimal are very strong (linear Gaussian transition and observation models). There are two cases under which the Kalman filter fails:

1. The underlying processes are **non-linear**. There are, however, approximate versions of the Kalman filter that can be applied to non-linear processes. In particular, the *Extended Kalman filter* approach often overcomes issues with non-linearities in the system. The idea is that if we have a non-linear Gaussian process: $y = A \cdot x + B + e$, where the matrix A or vector B have non-linear terms (e.g., in target tracking the $sin()$ and $cos()$ functions often appear - which result in non-linear systems) we can apply Taylor approximation and "linearize" these non-linear parameters. This means that the functions A and B are approximated by linear functions locally for the given value of x. The Extended Kalman filter works very well for smooth, well-behaved systems and allows the tracker to maintain and update a Gaussian state distribution as an approximation of the true underlying distribution.
2. What are "non-smooth" or "poorly behaved" systems? Consider the example in Figure 3. A plane is flying and in front of it there is an obstacle. The most probable locations for the plane after it passes the obstacle are either above it or below it. Thus the true distribution has two separated peaks, or two modes as they called. Such distributions are referred to as **multi-modal distributions**. If we try to approximate the true multi-modal distribution with a single Gaussian distribution, then we might get a result similar to the one

shown in the Figure. The actual peak of the distribution is in a location where there is no probability of the plane being there. One possible solution to this problem is to use a "switching" Kalman filter or a Gaussian Sum filter. The idea behind these approaches is to use multiple Kalman filters in parallel to track the state of the agent. In the above example, some of those Kalman filters will be propagated above the obstacle and some of the them will be propagated below the obstacle. Eventually those filters that do not agree with the data will be dropped by the algorithm. Obviously, this is a more computationally expensive approach and it also involves more user parameters (e.g., number of Kalman filters, when to drop an estimate, when to split an estimate etc.).

The alternative approach to the Kalman filter, the particle filter, is actually able to directly deal both with non-linear processes and multi-modal distributions at the expense of computational power.

2 Particle Filter

A particle filter follows the same principle like some of the approximate methods for inference in Bayesian networks (likelihood weighting) but is applied to temporal state estimation problems. A particle filter maintains a population of N samples. Each sample - or particle, as it is called in this case - corresponds to an estimate about the system's state. Each particle also stores a weight of how probable this estimate is given the currently available observations.

The particle filter algorithm operates as follows:

- A population of N samples is constructed by sampling from the prior distribution $P(x_0)$
- Then the following update cycle is repeated for each time step:
  1. Each sample if propagated forward by sampling the next state value x_{t+1} given the current value x_t using the transition model $P(x_{t+1}|x_t)$.
  2. Each sample is weighted by the likelihood it assigns to new evidence according to the observation model $P(e_{t+1}|x_{t+1})$.
  3. The population is resampled to generate a new population of N samples. The probability that a sample is selected, is proportional to its weight. The new samples produced are assigned equal weights.

Why does this approach work? Suppose at time t the samples correctly represent the belief distribution. This means that if we have N samples total then:

$$P(x_t|e_{1:t}) = \frac{N(x_t|e_{1:t})}{N} \quad \text{for } N \to \infty \tag{1}$$

where $N(x_t|e_{1:t})$ is the number of samples that have $X_t = x_t$ (the variable X_t has value x_t).

How many samples are then reaching the state x_{t+1}? The answer is that we just apply the transition model to get the new state for each sample. Consequently:

$$N(x_{t+1}|e_{1:t}) = \sum_{x_t} P(x_{t+1}|x_t) \cdot N(x_t|e_{1:t}) \tag{2}$$

Then we reweight each particle according to the $P(e_{t+1}|x_{t+1})$. Therefore the total weight of the samples in x_{t+1} after seeing e_{t+1} is:

$$W(x_{t+1}|e_{1:t+1}) = P(e_{t+1}|x_{t+1}) \cdot N(x_{t+1}|e_{1:t}) \tag{3}$$

Then the resampling step has the following effect:

Therefore, the sample population after one update cycle correctly represents the true distribution at time $t+1$. The approach has been experimentally shown in many application areas (e.g., robot localization) to be also computationally efficient and effective.

$$
\begin{aligned}
\frac{N(x_{t+1}|e_{1:t+1})}{N} &= \alpha \cdot W(x_{t+1}|e_{1:t+1}) && \text{(Equation 3)} \\
&= \alpha \cdot P(e_{t+1}|x_{t+1}) \cdot N(x_{t+1}|e_{1:t}) && \text{(Equation 2)} \\
&= \alpha \cdot P(e_{t+1}|x_{t+1}) \cdot \textstyle\sum_{x_t} P(x_{t+1}|X_t) \cdot N(x_t|e_{1:t}) && \text{(Equation 1)} \\
&= \alpha \cdot N \cdot P(e_{t+1}|x_{t+1}) \cdot \textstyle\sum_{x_t} P(x_{t+1}|X_t) \cdot P(x_t|e_{1:t}) \\
&= \alpha' P(e_{t+1}|x_{t+1}) \cdot \textstyle\sum_{x_t} P(x_{t+1}|X_t) \cdot P(x_t|e_{1:t}) \\
&= P(x_{t+1}|e_{1:t+1})
\end{aligned}
$$

The last few lectures correspond to state estimation problems under uncertainty. The ultimate goal of this course, however, is to present ways for agents to make intelligent decisions. Consequently, the following material will focus on how intelligent agents make decisions in uncertain environments. The approaches we have discussed for state estimation are still relevant in the following discussion.

3 Utility Theory

In artificial intelligence, we design agents that make intelligent decisions. This section focuses on decision-making under uncertainty. In this context, agents pick states based on their utilities.

A utility function assigns a single number to express the desirability of a state. $U : S \Rightarrow R$ is used to denote the utility of a state, where S is the state space of a problem and R is the set of real numbers. These utilities are used in combination with probabilities of outcomes to get expected utilities for every action. Agents choose the actions that maximize their expected utility.

For example, consider a non-deterministic action A, which has possible outcome states $Result_i(A)$. Index i spans over the different outcomes. Each outcome has a probability assigned to it by the agent before the action is performed:

$$P(Result_i(A)|Do(A), E)$$

E corresponds to the evidence variables. Now we want to maximize the expected utility. If $U(Result_i(A))$ is the utility of state $(Result_i(A))$, then the expected utility is:

$$EU(A|E) = \sum_i P(Result_i(A)|Do(A), E)U(Result_i(A))$$

Maximum Expected Utility (MEU) principle: A rational agent will choose the action that maximizes the expected utility. To find this maximum, we would have to enumerate all actions, which is not practical for long sequences. Instead we must find a different way to handle these problems, starting with simple decisions.

3.1 Constraints on Rational Preferences

First we will define some notation for the constraints on preferences that an agent should have.

$A \succ B$ (A is preferred to B)
$A \sim B$ (A is indifferent to B)
$A \succeq B$ (The agent prefers A to B or is indifferent to them)

A and B here are lotteries. A Lottery is a probability distribution over a set of outcomes. These outcomes can be called prizes of the lottery. A lottery can be defined according to the equation below, where $C_1, ..., C_n$ are possible outcomes and $p_1, ..., p_n$ are the probabilities of these outcomes.

$$L = [p_1, C_1; p_2, C_2; ...p_n, C_n]$$

The outcome of a lottery is either an atomic state or another lottery.
There are also constraints on lotteries, using the notation introduced above:

- **Orderability:**
$$\forall A, B : (A \succ B) \vee (B \succ A) \vee (A \sim B)$$
A rational agent must prefer one lottery or the other, or else the agent considers the two lotteries to be equivalent.
- **Transitivity:**
$$(A \succ B) \wedge (B \succ C) \Rightarrow (A \succ C)$$
If an agent prefers A over B, and prefers B over C, then the agent prefers A over C.
- **Continuity:**
$$A \succ B \succ C \Rightarrow \exists p : [p, A; (1-p), C] \sim B$$
If B is between A and C as a preference, then there is a probability that makes the selection of B equivalent to the selection of the lottery that yields A with a probability p and C with probability 1-p.
For example, consider a case where there are three lotteries A, B, and C. Winning the A lottery will yield \$1,000,000; winning the B lottery will yield \$10,000; and winning the C lottery will yield \$0. The agent has the following choice: Either select B or gamble between A and C with probability p. For a certain probability p of winning A, these two choices will be the same for the agent.
$$p * 1,000,000 = 10,000 \quad \Rightarrow \quad p = 0.01$$
When the probability is 1% that the agent can win lottery A, the two choices are considered equivalent.
- **Substitutability:**
$$A \sim B \Rightarrow [p, A; (1-p), C] \sim [p, B; (1-p), C]$$
If an agent is indifferent between two lotteries, A and B, then the agent is indifferent between two more complex lotteries where the only difference is that B is substituted for A. This is true no matter what the probability p is and no matter what the other outcomes are in the lotteries.
- **Monotonicity:**
$$A \succ B \Rightarrow (p \geq q \Leftrightarrow [q, A; (1-p), B] \prec [p, A; (1-p), B]$$
If A is a preferred outcome over B, then a lottery that selects with higher probability A over B is preferred over one with a lower probability for A.
- **Decomposability:**
$$[p, A; (1-p), [q, B; (1-p), C]] \Rightarrow [p, A; (1-p)q, B; (1-p)(1-q), C]]$$
Compound lotteries can be simplified using the laws of probability. This also means that two consecutive lotteries can be combined into one equivalent lottery. This is also known as the "no fun in gambling" rule. Figure 4 provides an illustration of this constraint.

If the above axioms hold, then $\exists$ a real-valued function U that operates on states so that:
$U(A) > U(B)$ iff $A \succ B$ and
$U(A) = U(B)$ iff $A \sim B$
then the utility of a lottery is $U([p_1, S_1; ...; p_n, S_n]) = \sum_i p_i * U(S_i)$ **[Maximum Expected Utility Principle]**

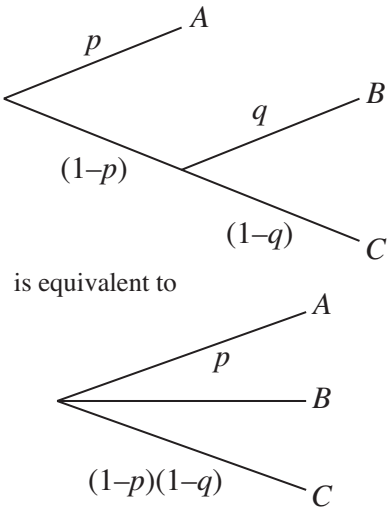


Figure 4: The "no fun in gambling" rule of utility theory.

3.2 Utility Functions

Defining utilities is useful in that we can design utility functions that can change the behavior of the agent to a desired behavior.

3.2.1 Utility of Money

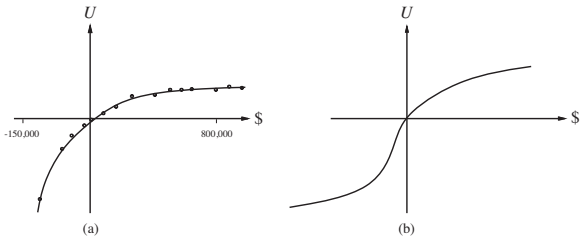
One example that everyone can relate to in some form is the utility of money. An agent would exhibit monotonic preference which would mean that the agent would always prefer more money to less. But this is not always exactly the case. Consider the case where you can either take \$1,000,000 or gamble between winning \$0 and \$3,000,000 by flipping a coin. So our expected monetary value of the gamble looks like this:

$$0.5 \cdot (\$0) + 0.5 \cdot (\$3,000,000) = \$1,500,000$$

And the expected monetary value for the first case is \$1,000,000. This does not necessarily mean that gambling is the better deal. Some people would rather take \$1,000,000 if it is worth a lot to them, while people with billions of dollars may gamble because the \$1,000,000 will probably not make a huge difference to them. Studies have shown that the utility of money is proportional to the logarithm of the amount, shown in the first graph below. After a certain amount of money, risk-averse agents prefer a sure payoff that is less than the expected monetary value of the gamble. However, if someone is \$10,000,000 in debt, they may consider gambling with a 50/50 chance of winning \$10,000,000 or losing \$20,000,000 because of desperation. This type of agent is risk seeking and the curve of this behavior can be seen in the second graph below.

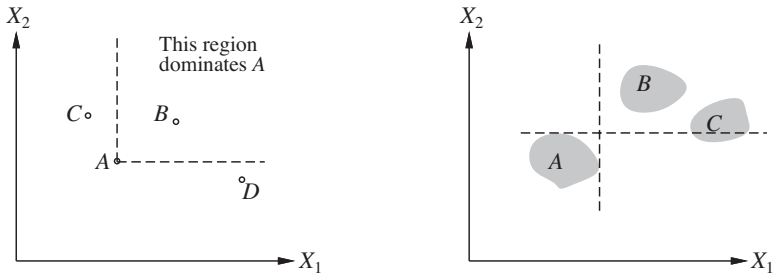
3.3 Multi-attribute Utility Functions

Multi-attribute problems are problems in which there is more than one attribute. If an airport needs to be built, then cost of land, distance from centers of population, noise levels, and safety issues are all attributes associated with this problem. There are cases, however, where a multi-attribute problem can be equivalent to a single-attribute problem.



3.3.1 Dominance

Suppose that an airport site S_1 costs less, generates less noise, and is safer than an airport site S_2. Since S_2 is worse on all attributes, it does not even need to be considered. S_1 has a strict dominance over S_2 in this case. Strict dominance can be useful in narrowing choices.



In the figure above, section (a) shows a deterministic case and section (b) shows an uncertain case. In section (a), Option A is strictly dominated by B, but not by C or D. In section (b) of the figure, A is strictly dominated by B, but not by C. Here Option B would be S_2 in our airport example.