Statistical Techniques in Robotics (16-831, F09) Lecture #20 (Oct. 29)

Gaussian Processes (part 1)

Lecturer: Drew Bagnell *Scribe: Alberto Rodriguez*

1 Practical Kalman Filtering (continuation)

One of the problems with the formulation of Kalman Filter is that it only has nice closed form update rules when the dynamics of the system is linear. There are two ways to deal with that limitation:

1. Linearize the dynamics at every step (*Iterative Linearization*).
2. Do sample based estimation of the necessary statistics (*Montecarlo Kalman Filter*).

For the rest of the section we will suppose that we are dealing with the system:

$$x_{t+1} = f(x_t) + \varepsilon, \qquad \text{where } \varepsilon \sim \mathcal{N}(0, Q) \tag{1}$$
$$y_{t+1} = g(x_{t+1}) + \delta, \qquad \text{where } \delta \sim \mathcal{N}(0, R) \tag{2}$$

1.1 Iterative Linearization

As seen in last lecture, the most straightforward way to apply Kalman Filtering whenever the dynamics is not linear is to use first order approximation of both the motion and observation models at every step.

$$x_{t+1} \approx f(\mu_t) + \frac{\partial f}{\partial x}(\mu_t)(x_t - \mu_t) + \varepsilon \tag{3}$$
$$y_{t+1} \approx g(\mu_{t+1}) + \frac{\partial g}{\partial x}(\mu_{t+1})(x_{t+1} - \mu_{t+1}) + \delta \tag{4}$$

In the last lecture we saw how this approximation yields new update rules. In those update rules, matrices A and C are iteratively being estimated by the Jacobian (best first order Taylor approximation of the system) of the of the motion and observation models respectively. The problem with this approach is that the first order Taylor expansion of the dynamic equations of the system is not a robust approximation for most non-linear functions.

An *Statistically Linearized Kalman Filter* tries to overcome that limitation by approximating the Jacobian matrix of the system in a broader region centered in the state of the system.

1.2 Montecarlo Kalman Filter

The second option for dealing with non-linear systems is to replace the update rules by sampled based estimations.

Motion Model

Given the equation of the motion model $x_{t+1} = f(x_t) + \epsilon$, μ_t and Σ_t we need to estimate μ^-_{t+1} and Σ^-_{t+1}. For that we draw samples from the prior distribution x^i_t and pass them through $f(x)$. That way, and with the law of large numbers in hand, we can estimate:

$$\mu^-_{x_{t+1}} = \frac{1}{N}\sum_{i=1}^{N} f(x^i_t) \tag{5}$$

$$\Sigma^-_{x_{t+1}} = \frac{1}{N}\left[\sum \left(f(x^i_t) - \mu^-_{x_{t+1}}\right) \cdot \left(f(x^i_t) - \mu^-_{x_{t+1}}\right)^T\right] + Q \tag{6}$$

NOTE: Q is additive noise, uncorrelated with x_t.

Observation Model

Given the equation of the observation model $y_{t+1} = g(x_{t+1}) + \delta$ the update rules are be the same as before:

$$\mu^+_{x_{t+1}} = \mu^-_{x_{t+1}} + \Sigma_{XY}\Sigma^{-1}_{YY}\left(y_t - \mu_y\right) \tag{7}$$

$$\Sigma^+_{x_{t+1}} = \Sigma^-_{t+1} - \Sigma_{XY}\Sigma^{-1}_{YY}\Sigma_{YX} \tag{8}$$

But now we approximate μ_y, Σ_{YY} and Σ_{XY} as:

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} g(x^i_{t+1}) \tag{9}$$

$$\Sigma_{YY} = \frac{1}{N}\left[\sum_{i=1}^{N}\left(g(x^i_{t+1}) - \mu_y\right) \cdot \left(g(x^i_{t+1} - \mu_y\right)^T\right] + R \tag{10}$$

$$\Sigma_{XY} = \frac{1}{N}\left[\sum_{i=1}^{N}\left(x^i_{t+1} - \mu_{x_{t+1}}\right) \cdot \left(g(x^i_{t+1} - \mu_y\right)^T\right] \tag{11}$$

In comparison with a standard Particle Filter, a Montecarlo Kalman Filter forces some smoothing on the uncertainty what simplifies the process to get a solution. On the other hand it will always be unimodal, while a Kalman Filter can perfectly maintain several modes in the estimation of the distribution.

1.3 Sigma-Point Filter

A Sigma-Point Filter has exactly the same formulation than a Montecarlo Kalman Filter but it draws samples in a deterministic way from interesting locations. Suppose Δ_i are the eigenvectors of the covariance matrix Σ_{x_t}. Then we sample the points as:

$$\mu_{x_t} \pm \lambda_i \Delta_i \quad i = 1 \dots D$$

where D is the dimension of x_t and λ_i is proportional to the eigenvalue correspondent to eigenvector Δ_i. Then, the update rule for the motion model becomes:

$$\mu^-_{x_{t+1}} = \frac{1}{N} w_i \sum_{i=1}^{N} f(x_t^i) \tag{12}$$

$$\Sigma^-_{x_{t+1}} = \frac{1}{N} w_i \left[\sum \left(f(x_t^i) - \mu^-_{x_{t+1}} \right) \cdot \left(f(x_t^i) - \mu^-_{x_{t+1}} \right)^T \right] + Q \tag{13}$$

There are different versions of Sigma-Point filters and they all differ on how weights w_i are selected. All versions chose weights so that the method behaves perfectly for a gaussian model (linear dynamics) and then optimize the weights for with a different criteria. Different versions include Unscented Kalman Filter, Central Difference Kalman Filter, ...

The computational cost of Sigma-Point type filters is $O(d^3)$ for the eigenvector finding (usually implemented by the SVD decomposition) plus $2d + 1$ evaluations of the motion model $f(x)$.

In comparison with a Montecarlo Kalman Filter, a Sigma-Point Filter needs less particles to run and, hence, reduces the number of motion model evaluations, than can be costly. It only needs to be implemented once because all the process is independent of the model (f, g). However, it can perform really bad if facing an adversarial problem, because it samples the space in a deterministic way.

2 Gaussian Processes

[*NOTE: State of the art for non-linear regression.*]

A gaussian process is a random stochastic process where correlation is introduced between neighboring samples (think of a stochastic process as a sequence of random variables). The same way that an instance of a random variable is a single sample, an instance of a stochastic process can be thought as vector of samples:

$$X = [x_1, x_2, x_3 \dots] \tag{14}$$

Gaussian Processes artificially introduce correlation between close samples in that vector in order to enforce some sort of smoothness on the succession of samples. The way that correlation is introduced is by constructing the joint probability distribution of the long vector of samples. Gaussian processes assume that probability distribution to be a multidimensional gaussian:

$$p(X^i) = \frac{1}{z} e^{(X-\mu)\Sigma^{-1}(X-\mu)} \tag{15}$$

The correlation between samples in the succession X^i depends on matrix Σ. In Gaussian Processes the covariance matrix is constructed as the Gram matrix of the samples with some desired kernel $\kappa(\cdot,\cdot)$ as the inner product:

$$\Sigma_{ff} = \begin{pmatrix} \kappa(x_1,x_1) & \kappa(x_1,x_2) & \cdots & \kappa(x_1,x_n) \\ \kappa(x_2,x_1) & \kappa(x_2,x_2) & \cdots & \kappa(x_2,x_n) \\ \vdots & & & \vdots \\ \kappa(x_n,x_1) & \kappa(x_n,x_2) & \dots & \kappa(x_n,x_n) \end{pmatrix} \tag{16}$$

In order to introduce correlation between neighboring samples, the kernel κ is usually designed to have small support and centered around zero (i.e. a triangular or a gaussian). The kernel usually can be described as a function of the distance $x_i - x_j$.

2.1 Gaussian Processes as a prior in function space

A gaussian process can be used as a prior probability distribution over the space of functions when using a Bayesian approach. By doing so, we are imposing that functions where neighboring samples are correlated are more probable. We are enforcing functions to have some sort of smoothness.

Suppose we have some estimates (mean and variance) of the value of a function f(x) in certain locations $F = [f(x_1), f(x_2), f(x_3) \dots f(x_n)]$. Then we can interpret vector F as a gaussian process and give it a specific probability:

$$p(F) = \frac{1}{z} e^{(F-\mu)\Sigma^{-1}(F-\mu)} \tag{17}$$

With Σ constructed the same way as in equation 16 so that we introduce correlation between the values of f at neighboring locations x_i, x_j. This construction will allow us to infer the most probable value of the function in other locations.