1   Non-Linear Regression

In this example we want to use an EKF for a non-linear regression problem. In this case our state vector is a set of weights w and we wish to estimate the mean and variance μ_w, Σ_w with our EKF, which we will just refer to as μ and Σ from here on.

Table 1: Comparing a general Kalman Filter with a Bayes Linear Regression Example

KF	BLR	Difference
x	w	
A	I	No process model for how weights are changing over time
C_t	x^T	Different linearlization at every time step.
Q	0	Growing uncertainty with time.
Y	y	$y = w^T x + \epsilon$
Σ_0	Σ_w	prior
μ_0	μ_w	prior

For our observation model we want to use the non-linear sigmoid function $\sigma(x) = \frac{1}{1+\exp(x)}$:

$$y_t \;=\; \sigma(w^T x_t) + \varepsilon, \qquad \text{where } \; \varepsilon \sim \mathcal{N}(0, \gamma^2) \tag{1}$$

Note that our output in this case is a single value, so our Jacobian will be a vector. We can find the Jacobian J_σ using the chain rule and the fact that $\frac{\partial \sigma}{\partial x} = \sigma(x)(1 - \sigma(x))$. Here we actually want to find the derivative of $g(w) = \sigma(w^T x)$, so we use the chain rule with $z = w^T x$.

$$\frac{\partial g}{\partial w} \;=\; \frac{\partial g}{\partial z}\frac{\partial z}{\partial w} \tag{2}$$

$$= \; \sigma(w^T x)(1 - \sigma(w^T x))x^T \tag{3}$$

$$J_\sigma \;=\; \frac{\partial g}{\partial w}(\mu) \tag{4}$$

$$= \; \sigma(\mu^T x)(1 - \sigma(\mu^T x))x^T \tag{5}$$

This gives update equations:

$$\mu_{t+1} \;=\; \mu_t^+ + \Sigma_t^+ J_\sigma^T (J_\sigma \Sigma_t^+ J_\sigma^T + \gamma^2)^{-1}[y_{t+1} - \sigma(\mu_t^{+T} x)] \tag{6}$$

$$\Sigma_{t+1} \;=\; \Sigma_t^+ - \Sigma_t^+ J_\sigma^T (J_\sigma \Sigma_t^+ J_\sigma^T + \gamma^2)^{-1} J_\sigma \Sigma_t^+ \tag{7}$$

---

[1]Some content adapted from previous scribes: Jackie Libby, Hyunggi Cho, and Alberto Rodriguez

For this non-linear regression example, there is no motion step, so the update step just takes the output of the update step from the last round. In other words, $\mu_{t+1}^- = \mu_t^+$ and $\Sigma_{t+1}^- = \Sigma_t^+$. Alternatively, we could add in a pseudo motion model to represent growing uncertainty with each time step:

$$\mu_{t+1}^- = \mu_t \tag{8}$$
$$\Sigma_{t+1}^- = \Sigma_t + \lambda I \tag{9}$$

where λ is forgetting factor and I is an identity matrix.

Also, since we use the sigmoid function for this example, we use γ^2 instead of σ^2 to denote the variance, to avoid confusion. More generally, this would be the covariance, R, but in this example the sigmoid function outputs a scalar.

2 Statistically Linearized Kalman Filters

Assuming we have the general nonlinear system given by the following equations:

$$x_{t+1} = f(x_t) + \varepsilon_{t+1}, \qquad \text{where } \varepsilon \sim \mathcal{N}(0, Q) \tag{10}$$
$$y_{t+1} = g(x_{t+1}) + \delta_{t+1}, \qquad \text{where } \delta \sim \mathcal{N}(0, R) \tag{11}$$

with non-linear functions f and g, we can use other approaches besides the EKF. A *Statistically Linearized Kalman Filter* tries to overcome that limitation by approximating the Jacobian matrix of the system in a broader region centered at the state of the system. This type of approach also offers the benefit that it that it does not require continuity or differentiability of the motion and measurement models. Since it is not necessary to compute Jacobian matrices, these methods can offer benefits in terms of computational efficiency as well. However, a downside is that they require the non-linear functions f and g to be provided in closed form.

2.1 Montecarlo Kalman Filter

One example of such a filter is the Mantecarlo Kalman Filter (MCKF). This isn't actually a real filter used in practice, but it is presented here for the sake of demonstration, as a precursor to the discussion of the Unscented Kalman Filter next.

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_t^i and pass them through $f(x)$. That way, and with the law of large numbers in hand, we can estimate:

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

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

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 the same as before:

$$\mu_{x_{t+1}}^{+} = \mu_{x_{t+1}}^{-} + \Sigma_{XY}\Sigma_{YY}^{-1}(y_t - \mu_y) \tag{14}$$

$$\Sigma_{x_{t+1}}^{+} = \Sigma_{t+1}^{-} - \Sigma_{XY}\Sigma_{YY}^{-1}\Sigma_{YX} \tag{15}$$

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

$$\mu_y = \frac{1}{N}\sum_{i=1}^{N} g(x_t^i) \tag{16}$$

$$\Sigma_{YY} = \frac{1}{N}\left[\sum_{i=1}^{N}\left(g(x_t^i) - \mu_y\right) \cdot \left(g(x_t^i) - \mu_y\right)^T\right] + R \tag{17}$$

$$\Sigma_{XY} = \frac{1}{N}\left[\sum_{i=1}^{N}\left(x_t^i - \mu_{x_t}\right) \cdot \left(g(x_t^i) - \mu_y\right)^T\right] \tag{18}$$

In the above equations, we can choose to replace $\mu_{x_{t+1}}^{-}$ with $f(\mu_{x_t})$ and μ_y with $g(\mu_{x_{t+1}}^{-})$. In other words, we take the mean of our samples and pass them through the nonlinear functions f and g, instead of passing every sample through the functions. If the samples were perfectly gaussian, these two methods would produce the same result.

Note that last year's lectures used x_{t+1}^i instead of x_t^i in these update equations. I believe it is more correct to use x_t^i. There can be two cases: case 1 is that we're doing an update step on the same timestep as a motion step, (which is what the overall notation is suggesting with the +'s and -'s). In this case, we would not have x_{t+1}^i's to sample from, because we are still at time step t. Case 2 is that we're doing an update step at the next timestep, and that we are treating the motion and update steps as asynchronous events that happen independently from each other. In this case, we would look at the update step separately from the motion step, and we could leave the time subscript out altogether because an update step happens at one moment in time.

Comparing the Montecarlo Kalman Filter (MCKF) with the standard Particle Filter (PF):

- Good things about the MCKF

- The MCKF forces some smoothing on the uncertainty that simplifies the process to get a solution.
- The MCKF doesn't need as many samples as the PF. The number of samples is on the order of $O(d^2)$. To estimate n parameters, we need $O(n)$ particles. Here we have d^2 parameters.

- Bad things about the MCKF

  - The MCKF will always be unimodal, while a the PF can perfectly maintain several modes in the estimation of the distribution.

2.2 Sigma-Point Filter

A Sigma-Point Filter has exactly the same formulation as 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 corresponding to eigenvector Δ_i. In other words, we pick one point along each eigenvector direction. We also sometimes pick the last point as the mean itself, so the number of points is $2d + 1$. Then, the update rule for the motion model becomes:

$$\mu^-_{x_{t+1}} = w_i \sum_{i=1}^{N} f(x_t^i) \tag{19}$$

$$\Sigma^-_{x_{t+1}} = w_i \left[\sum \left(f(x_t^i) - \mu^-_{x_{t+1}} \right) \cdot \left(f(x_t^i) - \mu^-_{x_{t+1}} \right)^T \right] + Q \tag{20}$$

This assumes the weights sum to 1. There are different versions of Sigma-Point filters and they all differ on how weights w_i are selected. All versions choose weights so that the method behaves perfectly for a gaussian model (linear dynamics) and then optimize the weights for 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 finding the eigenvectors (usually implemented by the SVD decomposition) plus $2d + 1$ evaluations of the motion model $f(x)$.

Evaluating the Sigma-Point Filter:

- Good things

  - It needs less particles to run, and hence reduces the number of motion model evaluations, which can be costly.
  - It only needs to be implemented once because the algorithm is generic, for any choice of your model, (f, g).

– You don't have pretend like you know calculus, because there are no Jacobians.

- Bad things

  – It can perform really bad if facing an adversarial problem, because it samples the space in a deterministic way.

  – As the dimension goes up, the center weight can become negative. (One fix is to leave the center value out for the variance calculations.)

  – The eigenvalue decomposition need to be done at every time step, which can be costly. Also, SVD is not always deterministic, so you count bound the computation time. This makes it difficult to run online.

Summary of Sigma-Point Filter Approach

- Compute sigma points and weights

- Transform sigma points through system dynamics

- Reconstruct random variable statistics using weighted sample mean and covariance

- Perform Kalman measurement update

The Unscented Kalman Filter (UKF) is a type of Sigma-Point Filter, for specific choices of λ_i's and w_i's. Fig. 1 gives a pictorial representation of the UKF:
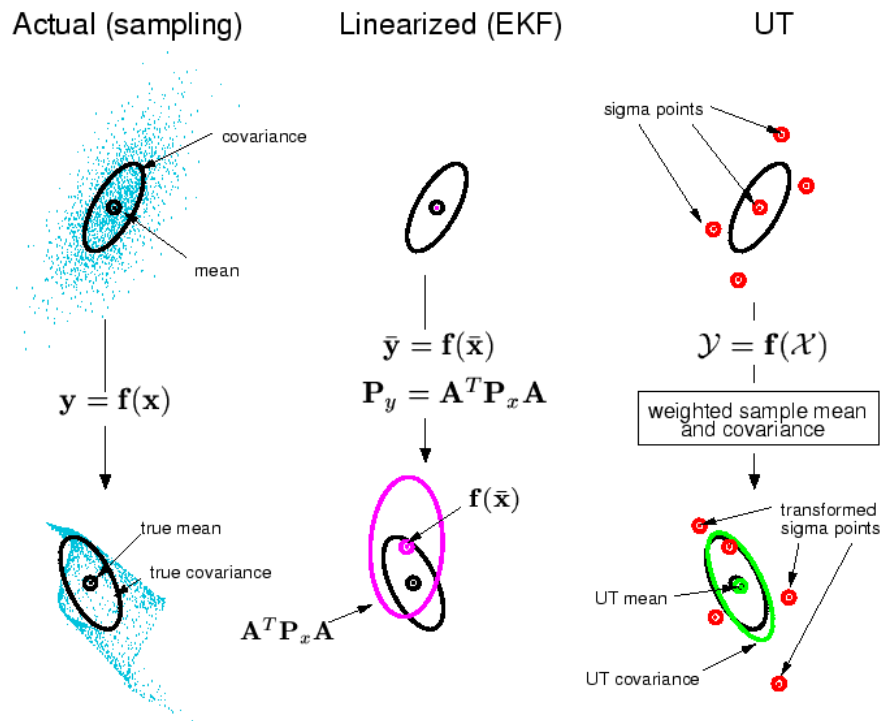
Figure 1: A comparison of the actual transformation, and the approximations given by the linear approximation and the unscented transformation.
(Source: http://cslu.cse.ogi.edu/nsel/ukf/node6.html)