1 The Extended Kalman Filter

Previously, we used a Kalman Filter which could only model linear transformations:

\[
\begin{align*}
x_{t+1} &= Ax_t + \varepsilon, \quad \text{where } \varepsilon \sim \mathcal{N}(0, Q) \\
y_{t+1} &= Cx_{t+1} + \delta, \quad \text{where } \delta \sim \mathcal{N}(0, R)
\end{align*}
\]

1.1 Why linear models?

The assumptions of linearity for both the measurement and state transition are essential for the correctness of the Kalman filter. The observation that any linear transformation of a Gaussian random variable yields another Gaussian is important in the derivation of the Kalman filter algorithm, so what happens if state transitions and measurements are no longer linear?

Figure 1 shows the effect of a nonlinear transformation of a Gaussian random variable. The lower right graph shows the random variable distributed about its mean. It is passed through the nonlinear function represented by the upper right graph. The resulting distribution (upper left) is no longer Gaussian in shape, rendering Kalman filtering useless over the domain of the nonlinear function.

![Figure 1: Effect of nonlinear transformation of Gaussian random variable.](image-url)
1.2 The Extended Kalman Filter

Unfortunately, state transitions and measurements are rarely linear in practice. Thus, we would like to be able to model non-linear transformations with our filter. The Extended Kalman Filter or EKF relaxes the linearity assumption by assuming that the state transition and measurement probabilities are governed nonlinear functions \( f \) and \( g \), such that

\[
x_{t+1} = f(x_t) + \varepsilon_{t+1}, \quad \text{where } \varepsilon \sim \mathcal{N}(0, Q) \tag{3}
\]

\[
y_{t+1} = g(x_{t+1}) + \delta_{t+1}, \quad \text{where } \delta \sim \mathcal{N}(0, R) \tag{4}
\]

Notice that the function \( f \) replaces the matrix \( A \) and the function \( g \) replaces \( C \). For arbitrary functions \( f \) and \( g \), the belief is no longer Gaussian. To combat this, the EKF calculates a Gaussian approximation to the true belief. This is shown in figure 2.

![Figure 2: Illustration of EKF](image)

The dashed line on the upper left graph represents the Gaussian approximation (Linearization) to the density of the random variable \( y \). Because it represents belief at some time \( t \) by some mean \( \mu_t \) and covariance \( \Sigma_t \), it still keeps from the original Kalman filter the Gaussian belief distribution, but this belief is no longer approximation. The goal of the EKF is therefore to efficiently estimate the mean and covariance of a posterior.

Linearization approximates the nonlinear function \( f \) and \( g \) at the mean of the Gaussian. Projecting it through the linear function shown in the upper right graph of 2 a linear approximation is shown in red of the upper left graph.
1.3 Linearization via the Taylor Series Expansion

One way to deal with this non-linearity is to use the Taylor series expansion. In this case we
approximate the transformations using the first order Taylor series expansion evaluated at the
mean estimate of $x$ at time $t$, $\mu_t$:

\[
x_{t+1} \approx f(\mu_t) + \frac{\partial f}{\partial x}(\mu_t)(x_t - \mu_t) + \varepsilon
\]
\[y_{t+1} \approx g(\mu_{t+1}) + \frac{\partial g}{\partial x}(\mu_{t+1})(x_{t+1} - \mu_{t+1}) + \delta
\]

1.4 New Update Rules

Using the non-linear transformation above, we get the following new update equations for our mean
and variance estimates for the motion model:

\[
\mu_{t+1} = f(\mu_t^+)
\]
\[
\Sigma_{t+1} = J_f \Sigma_t^+ J_f^T + Q
\]

And for the sensor model:

\[
\mu_{t+1} = \mu_{t+1}^- + \Sigma_{t+1}^- J_g^T(J_g \Sigma_{t+1}^- J_g^T + R)^{-1}[y_{t+1} - g(\mu_{t+1}^-)]
\]
\[
\Sigma_{t+1} = \Sigma_{t+1}^- - \Sigma_{t+1}^- J_g^T(J_g \Sigma_{t+1}^- J_g^T + R)^{-1}J_g \Sigma_{t+1}^-
\]

where $J_f = \frac{\partial f}{\partial x}(\mu_t^+)$ and $J_g = \frac{\partial g}{\partial x}(\mu_{t+1}^-)$. above.

1.5 Problems with Extended Kalman Filtering

- The Taylor series expansion is a poor approximation of most non-linear functions. The accuracy
  of the linearization depends on two factors: the degree of uncertainty and the amount of
  local nonlinearity in the functions being approximated.

- It can be very difficult to implement an EKF correctly: you will most likely get the Jacobian
  wrong on the first try. Even when the Jacobian is wrong, it can be hard to tell whether
  your implementation is incorrect or the poor filtering results are due to the Taylor expansion.
  Furthermore, the function might not be continuous across the range in which the linearization
  is used (e.g. a hinge function).

- EKFs are notoriously bad at handling multiple distinct hypotheses.

Extended Kalman Filters are only as good as their approximation about the mean of the estimate.
It is important to keep the uncertainty of the state estimate small when applying EKFs to avoid
negative effects from the nonlinearities.
1.6 Example: 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 $\mu_w, \Sigma_w$ with our EKF, which we will just refer to as $\mu$ and $\Sigma$ from here on.

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

<table>
<thead>
<tr>
<th></th>
<th>KF</th>
<th>BLR</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>$w$</td>
<td>$A$</td>
<td>$I$</td>
</tr>
<tr>
<td>$C_t$</td>
<td>$x^T$</td>
<td>$y$</td>
<td>$y$</td>
</tr>
<tr>
<td>$Q$</td>
<td>0</td>
<td>$R$</td>
<td>$\gamma^2$</td>
</tr>
</tbody>
</table>

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 $\Sigma_{t+1} = \Sigma_t + \lambda I$.

Alternatively, we could add in a pseudo-motion model to represent growing uncertainty with each time step:

\[
\mu_{t+1} = \mu_t \\
\Sigma_{t+1} = \Sigma_t + \lambda I
\] (11) (12)

Also, since we use the sigmoid function for this example, we use $\gamma^2$ instead of $\sigma^2$ 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.

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

\[
y_t = \sigma(w^T x_t) + \varepsilon, \quad \text{where } \varepsilon \sim \mathcal{N}(0, \gamma^2)
\] (13)

Note that our output in this case is a single value, so our Jacobian will be a vector. We can find the Jacobian $J_\sigma$ 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 $z = w^T x$.

\[
\frac{\partial g}{\partial w} = \frac{\partial g}{\partial z} \frac{\partial z}{\partial w} = \sigma(w^T x)(1 - \sigma(w^T x))x^T \\
J_\sigma = \frac{\partial g}{\partial w}(\mu) = \sigma(\mu^T x)(1 - \sigma(\mu^T x))x^T
\] (14) (15) (16)

This gives update equations:

\[
\mu_{t+1} = \mu_t + \Sigma_t^+ J_\sigma^T (J_\sigma \Sigma_t^+ J_\sigma^T + \gamma^2)^{-1}[y_{t+1} - \sigma(\mu_t^T x)] \\
\Sigma_{t+1} = \Sigma_t - \Sigma_t^+ J_\sigma^T (J_\sigma \Sigma_t^+ J_\sigma^T + \gamma^2)^{-1} J_\sigma \Sigma_t^+
\] (18) (19)
2 Statistically Linearized Kalman Filters

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

\[
\begin{align*}
  x_{t+1} &= f(x_t) + \epsilon_{t+1}, \quad \text{where } \epsilon \sim \mathcal{N}(0, Q) \\
  y_{t+1} &= g(x_{t+1}) + \delta_{t+1}, \quad \text{where } \delta \sim \mathcal{N}(0, R)
\end{align*}
\]  

with non-linear functions \( f \) and \( g \), we can use other approaches besides the EKF. A Statistically Linearized Kalman Filter approximates the Jacobian matrix of the system in a broader region centered at the state of the system. This type of approach offers the benefit 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 Monte Carlo Kalman Filter

One example of such a filter is the Monte Carlo 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 \( x_{t+1} = f(x_t) + \epsilon, \mu_t \) and \( \Sigma_t \) we need to estimate \( \mu_{t+1}^- \) and \( \Sigma_{t+1}^- \). 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:

\[
\begin{align*}
  \mu_{x_{t+1}}^- &= \frac{1}{N} \sum_{i=1}^{N} f(x_i^t) \\
  \Sigma_{x_{t+1}}^- &= \frac{1}{N} \left[ \sum \left( f(x_i^t) - \mu_{x_{t+1}}^- \right) \cdot \left( f(x_i^t) - \mu_{x_{t+1}}^- \right)^T \right] + Q
\end{align*}
\]

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:

\[
\begin{align*}
  \mu_{x_{t+1}}^+ &= \mu_{x_{t+1}}^- + \Sigma_{XY} \Sigma_{YY}^{-1} (y_t - \mu_y) \\
  \Sigma_{x_{t+1}}^+ &= \Sigma_{t+1}^- - \Sigma_{XY} \Sigma_{YY}^{-1} \Sigma_{YX}
\end{align*}
\]
But now we approximate $\mu_y$, $\Sigma_{YY}$ and $\Sigma_{XY}$ as:

$$x_{t+1}^i = f(x_t^i)$$

(26)

$$\mu_y = \frac{1}{N} \sum_{i=1}^{N} g(x_{t+1}^i)$$

(27)

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

(28)

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

(29)

Comparing the Monte Carlo 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 Monte Carlo Kalman Filter but it draws samples in a deterministic way from interesting locations. Suppose $\Delta_i$ are the eigenvectors of the covariance matrix $\Sigma_{x_t}$. Then we sample the points as:

$$\mu_{x_t} \pm \lambda_i \Delta_i \quad i = 1 \ldots d$$

where $d$ is the dimension of $x_t$ and $\lambda_i$ is proportional to the eigenvalue corresponding to eigenvector $\Delta_i$. 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:

$$\mu_{x_{t+1}} = w_i \sum_{i=1}^{N} f(x_t^i)$$

(30)

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

(31)
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 $f(x)$.

**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 $\lambda_i$’s and $w_i$’s. Fig. 3 gives a pictorial representation of the UKF:

**Figure 3:** A comparison of the actual transformation, and the approximations given by the linear approximation and the unscented transformation. (Source: http://cslu.cse.ogi.edu/nssel/ukf/node6.html)

Evaluating the Sigma-Point Filter:
• Good things
  – It needs less particles to run, and therefore 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 poorly 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 performed at every time step, which can be costly. Also, SVD is not always deterministic, so you cannot bound the computation time. This makes it difficult to run online.