- Calculating Jacobians is expensive

Lecture 6 - Unscented Kalman Filter

- Can see it as an alternative to the EKF when the linearization of EKF (first taylor expansion) works sub-optimally. → when g is highly non-linear near point of linearization

UKF

- rather than linearizing your update func.:
  - compute a set of sigma points (they are also weighted)
  - transform the points using the non-lin. func
  - compute a Gaussian from new weighted points (approximation)
    - ↳ (compute μ & Σ from transformed points)
  - resulting in a new Gaussian out.

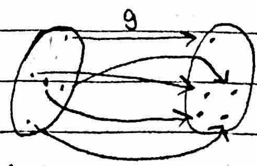
- avoids linearizing around the mean as Taylor expansion (and EKF) does.

Sigma Points

- how do we choose sigma points & weights?
- Select $\mathcal{X}^{(i)}, w^{(i)}$ so that:

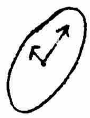
for an identity function

$$\begin{cases} \sum_i w^{(i)} = 1 & \text{(weights)} \\ \mu = \sum_i w^{(i)} \mathcal{X}^{(i)} \\ \Sigma = \sum_i w^{(i)} (\mathcal{X}^{(i)} - \mu)(\mathcal{X}^{(i)} - \mu)^T \end{cases}$$



↳ unscented Transform

- there is no unique soln for $\mathcal{X}^{(i)}, w^{(i)}$



→ eigenvectors go along main axis of ellipse
↳ the eigenvalues give you the ratio between these vectors
↳ 1:1 would be circle

Diagonalize A; $A = PDP^{-1}$

$P = [v_1 \; v_2]$, $D = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$

→ easy exponential ops on diagonal mtx

Choosing Sigma Points

$$X^{[0]} = \mu$$
$$X^{[i]} = \mu + \left(\sqrt{(n+\lambda)\Sigma}\right)_i \quad \text{for } i = 1, \dots, n$$
$$X^{[i]} = \mu - \left(\sqrt{(n+\lambda)\Sigma}\right)_{i-n} \quad i = n+1, \dots, 2n$$

Mtx Sqr root — dimensionality — Scaling Param — Column vector

Matrix Square Root

- defined as S with $\Sigma = SS$
- diagonalize Σ;

$$\Sigma = VDV^{-1}$$
$$= V \begin{bmatrix} d_{11} & 0 \\ 0 & d_{nn} \end{bmatrix} V^{-1}$$
$$= V \begin{bmatrix} \sqrt{d_{11}} & 0 \\ 0 & \sqrt{d_{nn}} \end{bmatrix} \begin{bmatrix} \sqrt{d_{11}} & 0 \\ 0 & \sqrt{d_{nn}} \end{bmatrix} V^{-1}$$

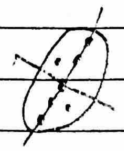
$$\therefore S = V \overbrace{\begin{bmatrix} \sqrt{d_{11}} & 0 \\ 0 & \sqrt{d_{nn}} \end{bmatrix}}^{D^{1/2}} V^{-1} \; ; \; SS = (VD^{1/2}V^{-1})(VD^{1/2}V^{-1}) = VDV^{-1} = \Sigma$$

Cholesky Matrix Square Root

- defined as L with $\Sigma = LL^T$
- result of the cholesky decomp.
- numerically stable solution
  ↳ often used in UKF
- L & Σ have the same eigenvectors

Sigma Points & Eigenvectors

- Can but do not have to lie on main axis of Σ



Sigma Point Weights

$$w_m^{[0]} = \frac{\lambda}{n+\lambda}$$

$$w_c^{[0]} = w_m^{[0]} + (1-\alpha^2+\beta)$$

$$w_m^{[i]} = w_c^{[i]} = \frac{1}{2(n+\lambda)} \quad \text{for } i = 1, \ldots, 2n$$

$w_c \longrightarrow$ weights for computing covariance

$w_m \longrightarrow$ weights for computing mean

Recover the Gaussian

$$\mu' = \sum_{i=0}^{2n} w_m^{[i]} g(\mathcal{X}^{[i]})$$

$$\Sigma' = \sum_{i=0}^{2n} w_c^{[i]} (g(\mathcal{X}^{[i]}) - \mu')(g(\mathcal{X}^{[i]}) - \mu')^T$$

- Note; if sigma points are close to mean, then this is not much better than EKF. Also if way too far can become worse than EKF. Usually better than EKF if you choose the right parameters

UT Parameters

- free params since there isn't a unique soln
- Scaled unscented transform suggests

$\kappa \geq 0$ → influences how far sigma pts away from mean

$\alpha \in (0, 1]$

$\lambda = \alpha^2(n+\kappa) - n$

$\beta = 2$ → optimal choice for gaussians

UKF - Algorithm

1 Unscented_Kalman_Filter$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$:

2. $\mathcal{X}_{t-1} = (\mu_{t-1} \quad \mu_{t-1} + \sqrt{(n+\lambda)\Sigma_{t-1}} \quad \mu_{t-1} - \sqrt{(n+\lambda)\Sigma_{t-1}})$

$$\bar{\mu} = \sum_{i=0}^{2n} w_m^{[i]} g(u_t, \mathcal{X}_{t-1})^{[i]}$$

$$\bar{\Sigma} = \sum_{i=0}^{2n} w_c^{[i]} (g(u_t, \mathcal{X}_{t-1})^{[i]} - \bar{\mu}_t)(g(u_t, \mathcal{X}_{t-1})^{[i]} - \bar{\mu}_t)^T + R_t$$

→ motion noise

$$\bar{\mathcal{X}}_t = (\bar{\mu}_t \quad \bar{\mu}_t + \sqrt{(n+\lambda)\bar{\Sigma}_t} \quad \bar{\mu}_t - \sqrt{(n+\lambda)\bar{\Sigma}_t}$$

$$\hat{z}_t = \sum_{i=0}^{2n} w_m^{[i]} h(\bar{\mathcal{X}}_t)^{[i]}$$

$$S_t = \sum_{i=0}^{2n} w_c^{[i]} (h(\bar{\mathcal{X}}_t)^{[i]} - \hat{z}_t)(h(\bar{\mathcal{X}}_t)^{[i]} - \hat{z}_t)^T + Q_t$$

$$\bar{\Sigma}_t^{x,z} = \sum_{i=0}^{2n} w_c^{[i]} (\bar{\mathcal{X}}_t^{[i]} - \bar{\mu}_t)(h(\bar{\mathcal{X}}_t)^{[i]} - \hat{z}_t)^T$$

$$K_t = \bar{\Sigma}_t^{x,z} S_t^{-1}$$

$$\mu_t = \bar{\mu}_t + K_t(z_t - \hat{z}_t)$$

$$\Sigma_t = \bar{\Sigma}_t - K_t S_t K_t^T$$

return μ_t, Σ_t

$$K_t = \underbrace{\bar{\Sigma}_t H_t^T}_{\bar{\Sigma}_t^{x,z}} \underbrace{(H_t \bar{\Sigma}_t H_t^T + Q_t)}_{S_t}{}^{-1}$$

↳ (from EKF)

- rather than Jacobian, we calculate new gaussian

$$\left.\begin{aligned} \Sigma_t &= (I - K_t H_t)\bar{\Sigma}_t \\ &= \bar{\Sigma}_t - K_t S_t K_t^T \end{aligned}\right\} \text{Proof in slides}$$

UKF vs. EKF

- Same result for linear models
- UKF better for non-lin models
- Differences are "Somewhat small"
- no Jacobians needed for UKF
- Same Complexity class (UKF little slower in practice)
- Still restricted to Gaussians.