CS 287: Advanced Robotics
Fall 2009

Lecture 22:
HMMs, Kalman filters

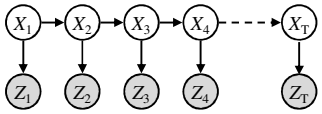
Pieter Abbeel
UC Berkeley EECS

Overview

- Current and next set of lectures
  - *The state is not observed*
  - Instead, we get some sensory information about the state
  - → Challenge: compute a probability distribution over the state which accounts for the sensory information ("evidence") which we have observed.

Hidden Markov Models

- Underlying Markov model over states X_t
  - Assumption 1: X_t independent of X_1, …, X_{t-2} given X_{t-1}
- For each state X_t there is a random variable Z_t which is a sensory measurement of X_t
  - Assumption 2: Z_t is assumed conditionally independent of the other variables *given* X_t
- This gives the following graphical (Bayes net) representation:



Filtering in HMM

- Init $P(x_1)$ [e.g., uniformly]
- Observation update for time 0:

$$P(x_1|z_1) \propto P(z_1|x_1)P(x_1)$$

- For t = 1, 2, …
  - Time update

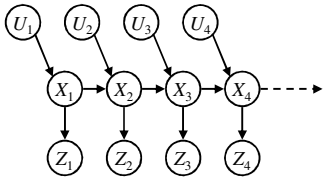
$$P(x_{t+1}|z_{1:t}) = \int_{x_t} P(x_{t+1}|x_t)P(x_t|z_{1:t})dx_t$$

  - Observation update

$$P(x_{t+1}|z_{1:t+1}) \propto P(z_{t+1}|x_{t+1})P(x_{t+1}|z_{1:t})$$

- For discrete state / observation spaces: simply replace integral by summation

With control inputs



- Control inputs known:
  - They can be simply seen as selecting a particular dynamics function
- Control inputs unknown:
  - Assume a distribution over them
- Above drawing assumes open-loop controls. This is rarely the case in practice. [Markov assumption is rarely the case either. Both assumptions seem to have given quite satisfactory results.]

Discrete-time Kalman Filter

Estimates the state x of a discrete-time controlled process that is governed by the linear stochastic difference equation

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t$$

with a measurement

$$z_t = C_t x_t + \delta_t$$

6

Kalman Filter Algorithm

Algorithm **Kalman_filter**($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$):

Prediction:

$$\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$$
$$\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$$

Correction:

$$K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$$
$$\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$$
$$\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$$

Return μ_t, Σ_t

7

Intermezzo: information filter

- From an analytical point of view == Kalman filter
- Difference: keep track of the inverse covariance rather than the covariance matrix

  [matter of some linear algebra manipulations to get into this form]
- Why interesting?
  - Inverse covariance matrix = 0 is easier to work with than covariance matrix = infinity (case of complete uncertainty)
  - Inverse covariance matrix is often sparser than the covariance matrix --- for the "insiders": inverse covariance matrix entry (i,j) = 0 if x_i is conditionally independent of x_j given some set $\{x_k, x_l, \ldots\}$
  - Downside: when extended to non-linear setting, need to solve a linear system to find the mean (around which one can then linearize)
  - See Probabilistic Robotics pp. 78-79 for more in-depth pros/cons

Matlab code data generation example

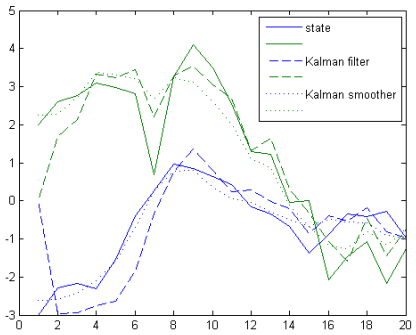
- A = [0.99 0.0074; -0.0136 0.99]; C = [1 1 ; -1 +1];
- x(:,1) = [-3;2];
- Sigma_w = diag([.3 .7]); Sigma_v = [2 .05; .05 1.5];
- w = randn(2,T); w = sqrtm(Sigma_w)*w; v = randn(2,T); v = sqrtm(Sigma_v)*v;
- for t=1:T-1

  x(:,t+1) = A * x(:,t) + w(:,t);

  y(:,t) = C*x(:,t) + v(:,t);

  end
- % now recover the state from the measurements
- P_0 = diag([100 100]); x0 =[0; 0];
- % run Kalman filter and smoother here
- % + plot

Kalman filter/smoother example



Kalman filter property

- If system is observable (=dual of controllable!) then Kalman filter will converge to the true state.
- System is observable iff

  O = [C ; CA ; CA^2 ; … ; CA^{n-1}] is full column rank (1)

Intuition: if no noise, we observe y_0, y_1, … and we have that the unknown initial state x_0 satisfies:

$y_0 = C x_0$

$y_1 = CA x_0$

...

$y_K = CA^K x_0$

This system of equations has a unique solution x_0 iff the matrix [C; CA; … CA^K] has full column rank. B/c any power of a matrix higher than n can be written in terms of lower powers of the same matrix, condition (1) is sufficient to check (i.e., the column rank will not grow anymore after having reached K=n-1).

Simple self-quiz

- The previous slide assumed zero control inputs at all times. Does the same result apply with non-zero control inputs? What changes in the derivation?

Kalman Filter Summary

- Highly efficient: Polynomial in measurement dimensionality *k* and state dimensionality *n*:
  $O(k^{2.376} + n^2)$
- Optimal for linear Gaussian systems!
- Most robotics systems are nonlinear!

→ Extended Kalman filter (EKF)
→ Unscented Kalman filter (UKF)

[And also: particle filter (next lecture)]

13

Announcement: PS2

- I provided a game log of me playing for your convenience.
- However, to ensure you fully understand, I suggest you follow the following procedure:
  - Code up a simple heuristic policy to collect samples from the state space for question 1. Then use these samples as your state samples for ALP and for approximate value iteration.
  - Play the game yourself for question 2 and have it learn to clone your playing style.
- You don't *need* to follow the above procedure, but I strongly suggest to in case you have any doubt about how the algorithms in Q1 and Q2 operate, b/c following the above procedure will force you more blatantly to see the differences (and can only increase you ability to hand in a good PS2).

Announcements

- PS1: will get back to you over the weekend, likely Saturday
- Milestone: ditto
- PS2: don't underestimate it!
- Office hours: canceled today. Feel free to set appointment over email. Also away on Friday actually. Happy to come in on Sat or Sun afternoon by appointment.
- Tuesday 4-5pm 540 Cory: Hadas Kress-Gazit (Cornell)

High-level tasks to correct low-level robot control
In this talk I will present a formal approach to creating robot controllers that ensure the robot satisfies a given high level task. I will describe a framework in which a user specifies a complex and reactive task in Structured English. This task is then automatically translated, using temporal logic and tools from the formal methods world, into a hybrid controller. This controller is guaranteed to control the robot such that its motion and actions satisfy the intended task, in a variety of different environments.

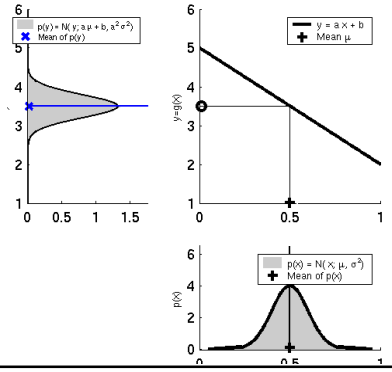
Nonlinear Dynamic Systems

- Most realistic robotic problems involve nonlinear functions

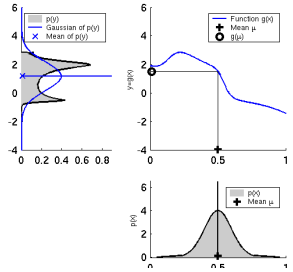
$$x_t = g(u_t, x_{t-1}) + noise$$

$$z_t = h(x_t) + noise$$

Linearity Assumption Revisited

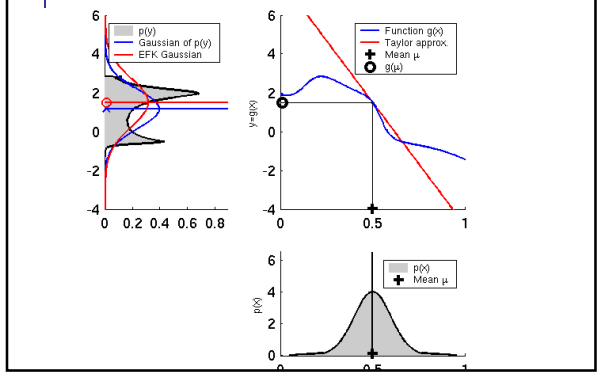


Non-linear Function

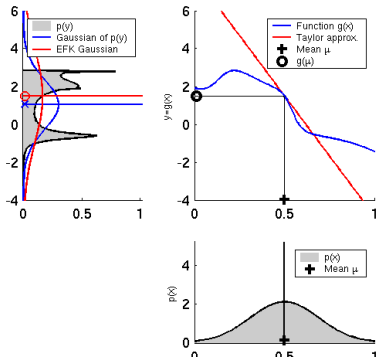


Throughout: "Gaussian of P(y)" is the Gaussian which minimizes the KL-divergence with P(y). It turns out that this means the Gaussian with the same mean and covariance as P(y).

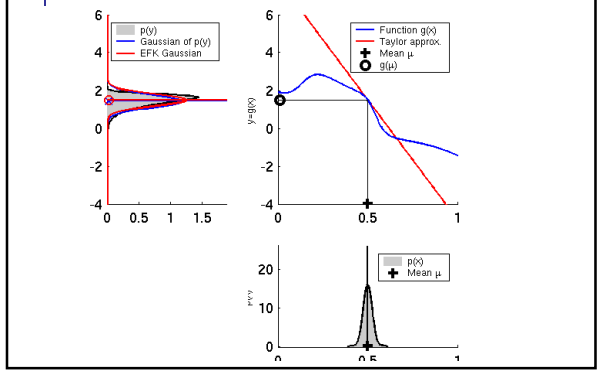
EKF Linearization (1)



EKF Linearization (2)



EKF Linearization (3)



EKF Linearization: First Order Taylor Series Expansion

- Prediction:

$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + \frac{\partial g(u_t, \mu_{t-1})}{\partial x_{t-1}} (x_{t-1} - \mu_{t-1})$$

$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + G_t (x_{t-1} - \mu_{t-1})$$

- Correction:

$$h(x_t) \approx h(\bar{\mu}_t) + \frac{\partial h(\bar{\mu}_t)}{\partial x_t} (x_t - \bar{\mu}_t)$$

$$h(x_t) \approx h(\bar{\mu}_t) + H_t (x_t - \bar{\mu}_t)$$

EKF Algorithm

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

Prediction:

$$\bar{\mu}_t = g(u_t, \mu_{t-1}) \quad \longleftarrow \quad \bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$$

$$\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t \quad \longleftarrow \quad \bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$$

Correction:

$$K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1} \quad \longleftarrow \quad K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$$

$$\mu_t = \bar{\mu}_t + K_t (z_t - h(\bar{\mu}_t)) \quad \longleftarrow \quad \mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$$

$$\Sigma_t = (I - K_t H_t) \bar{\Sigma}_t \quad \longleftarrow \quad \Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$$

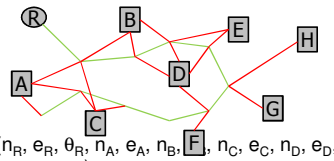
Return μ_t, Σ_t

$$H_t = \frac{\partial h(\bar{\mu}_t)}{\partial x_t} \qquad G_t = \frac{\partial g(u_t, \mu_{t-1})}{\partial x_{t-1}}$$

Simultaneous Localization and Mapping (SLAM)

- Robot navigating in unknown environment
- Perception
  - Environment: typically through laser-range finders, sonars, cameras
  - Odometry (its own motion): inertial sensing, wheel encoders, (if outdoors and clear sky-view: gps)

Simultaneous Localization and Mapping (SLAM)



- State: (n_R, e_R, θ_R, n_A, e_A, n_B, e_B, n_C, e_C, n_D, e_D, n_E, e_E, n_F, e_F, n_G, e_G, n_H, e_H)
- Transition model:
  - Robot motion model; Landmarks stay in place

Simultaneous Localization and Mapping (SLAM)

- In practice: robot is not aware of all landmarks from the beginning
- Moreover: no use in keeping track of landmarks the robot has not received any measurements about

→ Incrementally grow the state when new landmarks get encountered.

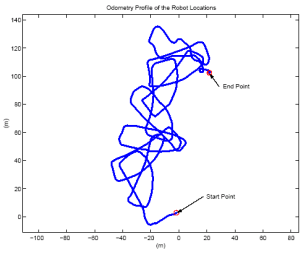
Simultaneous Localization and Mapping (SLAM)

- Landmark measurement model: robot measures [x_k; y_k], the position of landmark k expressed in coordinate frame attached to the robot:
  - $h(n_R, e_R, \theta_R, n_k, e_k) = [x_k; y_k] = R(\theta)\ (\ [n_k; e_k] - [n_R; e_R]\ )$
- Often also some odometry measurements
  - E.g., wheel encoders
  - As they measure the control input being applied, they are often incorporated directly as control inputs (why?)

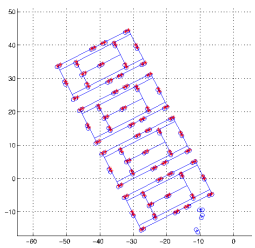
EKF SLAM Application

[courtesy by J. Leonard] 33

EKF SLAM Application



odometry



estimated trajectory

[courtesy by John Leonard] 34

EKF-SLAM: practical challenges

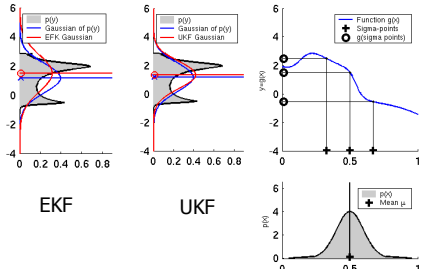
- Defining landmarks
  - Laser range finder: Distinct geometric features (e.g. use RANSAC to find lines, then use corners as features)
  - Camera: "interest point detectors", textures, color, …
- Often need to track multiple hypotheses
  - Data association/Correspondence problem: when seeing features that constitute a landmark --- Which landmark is it?
  - Closing the loop problem: how to know you are closing a loop?
  - → Can split off multiple EKFs whenever there is ambiguity;
  - → Keep track of the likelihood score of each EKF and discard the ones with low likelihood score
- Computational complexity with large numbers of landmarks.

EKF Summary

- Highly efficient: Polynomial in measurement dimensionality *k* and state dimensionality *n*:
  $O(k^{2.376} + n^2)$
- Not optimal!
- Can diverge if nonlinearities are large!
- Works surprisingly well even when all assumptions are violated!
- Note duality with linearizing a non-linear system and then running LQR back-ups to obtain the optimal linear controller!

36

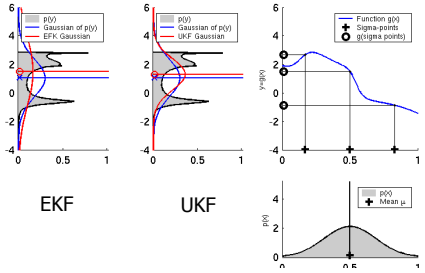
Linearization via Unscented Transform



EKF UKF

37

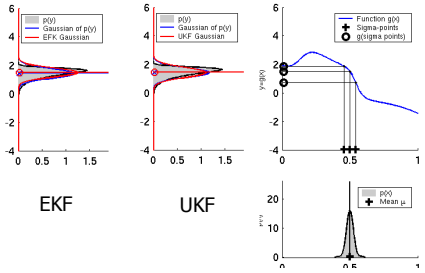
UKF Sigma-Point Estimate (2)



EKF UKF

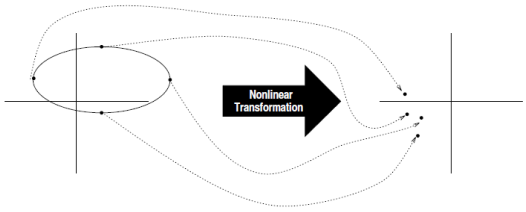
38

UKF Sigma-Point Estimate (3)



EKF UKF

UKF Sigma-Point Estimate (4)



UKF intuition why it can perform better

- Assume we know the distribution over X and it has a mean \bar{x}
- Y = f(X)

$$\mathbf{f}[\mathbf{x}] = \mathbf{f}[\bar{\mathbf{x}} + \delta\mathbf{x}]$$
$$= \mathbf{f}[\bar{\mathbf{x}}] + \nabla\mathbf{f}\delta\mathbf{x} + \frac{1}{2}\nabla^2\mathbf{f}\delta\mathbf{x}^2 + \frac{1}{3!}\nabla^3\mathbf{f}\delta\mathbf{x}^3 + \frac{1}{4!}\nabla^4\mathbf{f}\delta\mathbf{x}^4 + \cdots$$

$$\bar{\mathbf{y}} = \mathbf{f}[\bar{\mathbf{x}}] + \frac{1}{2}\nabla^2\mathbf{f}\,\mathbf{P}_{xx} + \frac{1}{2}\nabla^4\mathbf{f}\,\mathrm{E}[\delta\mathbf{x}^4] + \cdots$$

$$\mathbf{P}_{yy} = \nabla\mathbf{f}\,\mathbf{P}_{xx}(\nabla\mathbf{f})^T + \frac{1}{2\times 4!}\nabla^2\mathbf{f}\left(\mathrm{E}[\delta\mathbf{x}^4] - \mathrm{E}[\delta\mathbf{x}^2\mathbf{P}_{xx}] - \mathrm{E}[\mathbf{P}_{xx}\delta\mathbf{x}^2] + \mathbf{P}_{xx}^2\right)(\nabla^2\mathbf{f})^T + \frac{1}{3!}\nabla^3\mathbf{f}\,\mathrm{E}[\delta\mathbf{x}^4](\nabla\mathbf{f})^T + \cdots.$$

[Julier and Uhlmann, 1997]

UKF intuition why it can perform better

- Assume
  - 1. We represent our distribution over x by a set of sample points.
  - 2. We propagate the points directly through the function f.
- Then:
  - We don't have any errors in f !!
  - The accuracy we obtain can be related to how well the first, second, third, … moments of the samples correspond to the first, second, third, … moments of the true distribution over x.

Self-quiz

- When would the UKF significantly outperform the EKF?

Original unscented transform

- Picks a minimal set of sample points that match 1st, 2nd and 3rd moments of a Gaussian:

$$\begin{aligned} \boldsymbol{\mathcal{X}}_0 &= \bar{\mathbf{x}} & W_0 &= \kappa/(n+\kappa) \\ \boldsymbol{\mathcal{X}}_i &= \mathbf{x} + \left(\sqrt{(n+\kappa)\mathbf{P}_{xx}}\right)_i & W_i &= 1/2(n+\kappa) \\ \boldsymbol{\mathcal{X}}_{i+n} &= \bar{\mathbf{x}} - \left(\sqrt{(n+\kappa)\mathbf{P}_{xx}}\right)_i & W_{i+n} &= 1/2(n+\kappa) \end{aligned}$$

- \bar{x} = mean, P_{xx} = covariance, i → i'th row, $x \in \Re^n$
- \kappa : extra degree of freedom to fine-tune the higher order moments of the approximation; when x is Gaussian, n+\kappa = 3 is a suggested heuristic

[Julier and Uhlmann, 1997]

Unscented Kalman filter

- Dynamics update:
  - Can simply use unscented transform and estimate the mean and variance at the next time from the sample points
- Observation update:
  - Use sigmapoints from unscented transform to compute the covariance matrix between x_t and z_t. Then can do the standard update.

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

1. $\mathcal{X}_{t-1} = (\mu_{t-1} \quad \mu_{t-1} + \gamma\sqrt{\Sigma_{t-1}} \quad \mu_{t-1} - \gamma\sqrt{\Sigma_{t-1}})$
2. $\bar{\mathcal{X}}_t^* = g(\mu_t, \mathcal{X}_{t-1})$
3. $\bar{\mu}_t = \sum_{i=0}^{2n} w_m^{[i]} \bar{\mathcal{X}}_t^{*[i]}$
4. $\bar{\Sigma}_t = \sum_{i=0}^{2n} w_c^{[i]} (\bar{\mathcal{X}}_t^{*[i]} - \bar{\mu}_t)(\bar{\mathcal{X}}_t^{*[i]} - \bar{\mu}_t)^\top + R_t$
5. $\bar{\mathcal{X}}_t = (\bar{\mu}_t \quad \bar{\mu}_t + \gamma\sqrt{\bar{\Sigma}_t} \quad \bar{\mu}_t - \gamma\sqrt{\bar{\Sigma}_t})$
6. $\bar{\mathcal{Z}}_t = h(\bar{\mathcal{X}}_t)$
7. $\hat{z}_t = \sum_{i=0}^{2n} w_m^{[i]} \bar{\mathcal{Z}}_t^{[i]}$
8. $S_t = \sum_{i=0}^{2n} w_c^{[i]} \left(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t\right) \left(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t\right)^\top + Q_t$
9. $\bar{\Sigma}_t^{x,z} = \sum_{i=0}^{2n} w_c^{[i]} \left(\bar{\mathcal{X}}_t^{[i]} - \bar{\mu}_t\right) \left(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t\right)^\top$
10. $K_t = \bar{\Sigma}_t^{x,z} S_t^{-1}$
11. $\mu_t = \bar{\mu}_t + K_t(z_t - \hat{z}_t)$
12. $\Sigma_t = \bar{\Sigma}_t - K_t S_t K_t^\top$
13. **return** μ_t, Σ_t

[Table 3.4 in Probabilistic Robotics]

UKF Summary

- Highly efficient: Same complexity as EKF, with a constant factor slower in typical practical applications
- Better linearization than EKF: Accurate in first two terms of Taylor expansion (EKF only first term)
- Derivative-free: No Jacobians needed
- Still not optimal!