Graphical Models

- Acyclic directed graph tell us how to write the joint probability in terms of conditional variables.

\[
p(B, E, A, J, M) = p(B)p(E)p(A|B, E)p(J|A)p(M|A)
\]
Graphical models

- If we don’t know A, B and E are independent.
  \[ P(B, E) = P(B)P(E) \]
- But if we know A, then B and E are dependent.
  \[ P(B, E|A) \neq P(B|A)P(E|A) \]

- If we don’t know A, J and M are dependent
  \[ P(J, M) \neq P(J)P(M) \]
- If we know A, J and M are independent
  \[ P(J, M|A) = P(J|A)P(M|A) \]

State Estimation

- State Vector:
  - Nx1 column vector of quantities we care about.
  \[ x = \begin{bmatrix} x \\ y \\ \theta \\ s \end{bmatrix} \]
  - Which quantities to include is an engineering choice
  - Could also estimate acceleration, angular velocity.
  - Could also include information about the world (e.g. landmarks)

- State Estimation:
  - The probabilistic estimation of the state vector.
State Estimation

- Our graphical model

![Graphical model diagram]

- Many ways of representing probability distribution
  - We’ll use multivariate Gaussians

State Estimation: Overview

- Suppose at time step 1, we have an estimate of our state vector (our prior):
  \[ p(x_1) \]

- Two basic operations:
  - Propagation
    - Account for passage of time
  - Observation
    - Incorporate information from sensors
Propagating

Suppose that time $\Delta t$ passes. How does our state evolve?
- Some function of our state $x$ and noise $w$:

$$
\begin{bmatrix}
x' \\
y' \\
\theta' \\
s'
\end{bmatrix}
= f(x, w) =
\begin{bmatrix}
x + s\Delta t \cos(\theta) + w_1 \\
y + s\Delta t \sin(\theta) + w_2 \\
\theta + w_3 \\
s + w_4
\end{bmatrix}
$$

How do we propagate our mean and covariance?
- (This is what we did last lecture!)

Propagating

- Propagate mean?
  - Just plug in current state value.
  - Usually, $E(w) = 0$

- Propagate covariance?
  - It’s non-linear, so linearize.
  - But propagation is function of state and $w$…
    - Linearize WRT both!
Propagation

- Linearize:
  \[
  \begin{bmatrix}
  x' \\
  y' \\
  \theta' \\
  s'
  \end{bmatrix} = f(x, w) = \begin{bmatrix}
  x + s \Delta t \cos(\theta) + w_1 \\
  y + s \Delta t \sin(\theta) + w_2 \\
  \theta + w_3 \\
  s + w_4
  \end{bmatrix}
  \]
  
  \[f(x, w) \approx J_x^f (x - u_x) + J_w^f (w - u_w) + f(u_x, u_w)\]

- Our Jacobians:
  \[
  J_x^f = \begin{bmatrix}
  1 & 0 & -s \Delta t \sin(\theta) & \Delta t \cos(\theta) \\
  0 & 1 & s \Delta t \cos(\theta) & \Delta t \sin(\theta) \\
  0 & 0 & 1 & 0 \\
  0 & 0 & 0 & 1
  \end{bmatrix}
  \quad J_w^f = \begin{bmatrix}
  1 & 0 & 0 & 0 \\
  0 & 1 & 0 & 0 \\
  0 & 0 & 1 & 0 \\
  0 & 0 & 0 & 1
  \end{bmatrix}
  \]

Propagation (summary)

- Write down propagation equation in terms of previous state and noise:
  \[x' = f(x, w)\]

- Linearize by computing Jacobians \(J_x^f, J_w^f\)

- Propagate:
  \[
  u_x' = f(u_x, u_w) = f(u_x, 0)
  \]
  \[
  \Sigma_x' = J_x \Sigma_x J_x^T + J_w \Sigma_w J_w^T
  \]
Observation

- Suppose we get a sensor observation:

- The observation tells us something about our state. What distribution do we want now?
  - We want the state given all data (observations)!

  \[ p(x_2 | z_2) \]

  What is this in terms of quantities that we know???

---

Observation

- We want: \( p(x_2 | z_2) \)

- Apply Bayes’ rule:

  \[
  p(x_2 | z_2) = \frac{p(z_2 | x_2)p(x_2)}{p(z_2)}
  \]

  - sensor model prior from propagation step
  - normalization constant

- We know all of these quantities!
Sensor Model

- Perhaps we have a compass that observes the heading, contaminated by white noise $w_1$

$$z(x) = \theta + w_1$$

- If we know how $w_1$ is distributed, we can compute the distribution $p(z|x)$
  - Mean and covariance projection again!

Observation: Putting things together

- We want our posterior distribution (conditioned on evidence):

$$p(x_2|z_2) \propto p(z_2|x_2)p(x_2)$$

- We’re representing each of these probabilities as Gaussian random variables, so we can write:

$$p(x_2|z_2) \propto Ke^{-\frac{1}{2}(z_2-z(x))^T\Sigma_z^{-1}(z_2-z(x))}e^{-\frac{1}{2}(x-u_x)^T\Sigma_x^{-1}(x-u_x)}$$
Observation: Putting things together

\[ p(x_2|z_2) \propto Ke^{-\frac{1}{2}(z_2-z(x))^T\Sigma_z^{-1}(z_2-z(x))}e^{-\frac{1}{2}(x-u_x)^T\Sigma_x^{-1}(x-u_x)} \]

\[ z - z(x) = z - Hd - z_0 = r - Hd \]

- Substitute and take logarithm:

\[ \chi^2 = (r - Hd)^T \Sigma_z^{-1}(r - Hd) + d^T \Sigma_x^{-1}d \]

A little math…

\[ \chi^2 = (r - Hd)^T \Sigma_z^{-1}(r - Hd) + d^T \Sigma_x^{-1}d \]

- Expand…

\[ \chi^2 = r^T \Sigma_z^{-1}r - 2d^T H^T \Sigma_z^{-1}r + d^T H^T \Sigma_z^{-1}Hd + d^T \Sigma_x^{-1}d \]

- Minimize by differentiating WRT d:

\[ \frac{\partial \chi^2}{\partial d} = -2H^T \Sigma_z^{-1}r + 2H^T \Sigma_z^{-1}Hd + 2\Sigma_x^{-1}d = 0 \]

\[ d = (H^T \Sigma_z^{-1}H + \Sigma_x^{-1})^{-1}H^T \Sigma_z^{-1}r \]
A solution, at last!

\[ d = (H^T \Sigma_z^{-1} H + \Sigma_x^{-1})^{-1} H^T \Sigma_z^{-1} r \]

- **Computational complexity?**
  - Matrix inversion is \(O(N^3)\) and is the dimension of the whole state vector!

- **Memory requirements?**
  - We’re going to have to store the covariance matrix, which is \(O(N^2)\)

---

Improving the method

- **Matrix inversion lemma (for invertible \(C\)):**

\[
(A + B C D)^{-1} B C = A^{-1} B (C^{-1} + D A^{-1} B)^{-1}
\]


\[ d = (H^T \Sigma_z^{-1} H + \Sigma_x^{-1})^{-1} H^T \Sigma_z^{-1} r \]

\[ d = \Sigma_x H^T (\Sigma_z + H \Sigma_x H^T)^{-1} r \]

- **Computational complexity now?**
Extended Kalman Filter

- This method called EKF
  - We've glossed over covariance updates for observation… they're ugly.

- Slightly more general/standard form:

\[ K = \Sigma_x^{-1} J_x^T (J_x \Sigma_x^{-1} J_x^T + J_w \Sigma_w J_w^T)^{-1} \]
\[ x = x^- + K(z - f(x^-, 0)) \]
\[ \Sigma_x = \Sigma_x^- - K J_x \Sigma_x^- \]

EKF: Intuition

- It's a low pass filter

\[ x = x^- + K(z - f(x^-, 0)) \]

  - How much does the observation disagree with our prior? "innovation"
  - How much do we trust this measurement, and should we adjust our state? "Kalman gain"

- Compare to IIR filter: \( y[n] = y[n-1] + ax[n] \)
  - EKF: we adjust gain \( a \) at every iteration
EKF: Intuition (Cartoon version!)

\[ K = \Sigma_x J_x^T (J_x \Sigma_x J_x^T + J_w \Sigma_w J_w^T)^{-1} \]
\[ x = x^- + K(z - f(x^-, 0)) \]

(Pretend that J is invertible)

\[ x = x^- + J_x^{-1} \frac{J_x \Sigma_x J_x^T}{J_x \Sigma_x J_x^T + J_w \Sigma_w J_w^T} (z - f(x^-, 0)) \]

Project from observation space to state space

What fraction of the uncertainty was our prior responsible for? (larger \(\rightarrow\) trust observation more)

How much does the observation disagree with our prior?

Why “Extended” Kalman Filter?

- The “Kalman Filter” was originally derived for purely linear systems
- Applicability is thus limited to linear problems
- Extended Kalman Filter is generalization to non-linear systems
  - But inexact!
EKF: Linearization Error

- Observations are “incorporated” only once
  - State and covariance are updated based on linearization point *at that point in time*

- If state estimate is inaccurate, linearization point will be inaccurate.
  - Introduces error into state estimate
  - Covariance is decreased as though there was no error introduced.
  - Filter becomes over-confident.

Properties

- Does our distribution become more or less certain?
  - After a propagation step
  - After an observation
Demo: No observations

Demo: With Observations
Summary

- Graphical models
  - Encode conditional dependence of variables
  - Shows how to write joint distribution

- State Estimation
  - Propagation: Simple mean/covariance projection
  - Observation: Posterior can be written as product of prior and observation model
    - Gaussian case: quadratic loss

- Extended Kalman Filter
  - Efficient (sort of) method of recursively computing the posterior