Tracking and Filtering

Petteri Nurmi
Questions

- What are state space models? Why are they relevant in position tracking?
- What is a Bayesian optimal filter? Which two steps form the filter?
- What is the Kalman filter? What are its main advantages / disadvantages?
- What are particle filters? What are their main advantages / disadvantages?
Tracking

- Tracking refers to monitoring of location related information over time (see Lecture II)
  - Position tracking: estimating the position of an entity
  - Trajectory tracking: estimating the path that an entity is taking
  - Buddy tracking: estimating when a friend enters/leaves close proximity
- Focus of this lecture is on so-called *filtering* techniques for position tracking
Preliminaries – Bayes’ Theorem

- A theorem that expresses how subjective belief should rationally change to account for evidence

\[ P(\theta | D) = \frac{P(D | \theta) P(\theta)}{P(D)} \]

- Common use in parameter estimation:
  - \( \Theta \) corresponds to parameters of interest
  - \( D \) corresponds to measurements
Example:

Suppose a drug test is 99.2% accurate (positive and negative cases) and that 0.75% of people use the drug.

What is the probability that an individual testing positive is a user?

\[
\]

\[
= \frac{0.992 \cdot 0.0075}{0.992 \cdot 0.0075 + 0.008 \cdot 0.9925}
\]

\[
\approx 48.37\%
\]
Preliminaries – Monte Carlo Integration

• Method for evaluating integrals using repeated random sampling
  • Can be used to approximate expected value of a function of random variables

• Estimating expected values:
  • Let \( X \) be a random variable with probability density \( f(x) \)
  • Let \( g(\cdot) \) denote an arbitrary function
  • The expectation of \( g \) with respect to \( f(x) \) is given by:
    \[
    E(g(x)) = \int g(x) f(x) \, dx
    \]
  • Monte Carlo estimate: \( E(g(x)) = \frac{1}{n} \sum_i g(x_i) \)
Monte Carlo Integration – Example

• Assume we know that user’s orientation is given by:
  \[ \theta \sim N(\theta | \pi/6, \pi/12) \]
• What is \( E(\sin(\theta)) \)?
• Generate samples \( x_i \sim N(\theta | \pi/6, \pi/12) \)
• Calculate \( \frac{1}{n} \sum x_i \sin(x_i) \)
• Example (Matlab):
  – \( s = \text{normrnd(pi/6,pi/12,1000,1)}; \)
  – \( \text{sum(sin(s))}/ 1000 = 0.477558 \)
  – \( \sin(pi/6) = 0.50 \)
Preliminaries – Importance Sampling

- Statistical technique for estimating properties of a distribution using samples from another distribution
  - Can be used in the context of Monte Carlo integration when desired distribution too complex

- Basic idea:
  - Choose a function \( g \)
  - Generate samples from \( g \)
  - Estimate \( w(x) = p(x) / g(x) \)
  - Return \( \mu = \sum w(x) f(x) / n \)
    - Note: \( g(x) \) must have same support as \( p(x) \)
State Space Models

- Mathematical model of a physical system as a set of input, output and state variables
- Relationships between variables follow differential and/or algebraic equations
- State variables: smallest set of system variables that can be used to represent the entire state of the system at a given time
- Common form:
  \[ x_k = Ax_{k-1} + v \]
  \[ y_k = Ux_k + w \]
  State equation
  Measurement equation
Hidden Markov Models

- Probabilistic **state space** model
  - Measurements \( y_k \sim p(y_k | x_k) \) (e.g., GPS coord.)
  - System state \( x_k \sim p(x_k | x_{k-1}) \) (true location)
- The measurements \( y_k \) are observed by the system
- The true state \( x_k \) is unknown
- Example: Gaussian random walk
  - State evolves according a Gaussian model with noise: \( x_k \sim x_{k-1} + N(0,p) \)
  - Measurements correspond to the system state with added noise: \( y_k \sim x_k + N(0,q) \)
Hidden Markov Models - Example
Hidden Markov Models - Assumptions

- The dynamic model $p(x_k | x_{k-1})$ governs the evolution of the system state
  - Assumption I: Future $x_k$ is independent of the past given the present $x_{k-1}$
    - $p(x_k | x_{1:k-1}, y_{1:k-1}) = p(x_k | x_{k-1})$
  - Assumption II: Past $x_{k-1}$ is independent of the future given the present $x_k$
    - $p(x_{k-1} | x_{k:T}, y_{k:T}) = p(x_{k-1} | x_k)$
- The measurements $y_k$ are conditionally independent given current system state $x_k$
  - $p(y_k | x_{1:k}, y_{1:k-1}) = p(y_k | x_k)$
Filtering, Prediction, Smoothing

• Assume at time $k$ we have the measurements sequence $y_{1:k-1}$ available
  • Prediction: Estimating future values of the system
  • Filtering: Estimate current value of the system in a way that it is consistent with the measurements
  • Smoothing: Estimate past values of the system based on a measurement sequence
Bayesian Optimal Filter - Principle

• Given a sequence of measurements $y_{1:k}$, determine the current state of the system $x_k$
• Bayesian optimal filter returns a distribution over the current state given the measurements $p(x_k | y_{1:k})$
  • Requires prior distribution $p(x_0)$
  • State space model:
    - $x_k \sim p(x_k | x_{k-1})$
    - $y_k \sim p(y_k | x_k)$
  • Measurement sequence $y_{1:k} = y_1, ..., y_k$
Bayesian Optimal Filter – Prediction and Update Steps

• We focus on **sequential** filtering
  • Observed values change over time

• A sequential Bayesian filter consists of two steps:
  • Prediction: estimating the most likely next value of the system state given the measurements up to that point
    – Given $y_{1:k-1}$ estimate the probability distribution $p(x_{k}, | y_{1:k-1})$
  • Updating: given new sensor measurement $y_{k}$, refine the distribution of $x_{k}$ to be consistent with $y_{k}$
    – Corresponds to determining the distribution $p(x_{k}, | y_{1:k})$
Bayesian Optimal Filter –
Prediction Step

• Assume the posterior of the previous time is known
  \[ p(x_{k-1} | y_{1:k-1}) \]

• Given a new measurement \( x_k \), we have:
  \[
  p(x_k, x_{k-1} | y_{1:k-1}) = p(x_k | x_{k-1}, y_{1:k-1}) \ p(x_{k-1} | y_{1:k-1}) \\
  = p(x_k | x_{k-1}) \ p(x_{k-1} | y_{1:k-1})
  \]

• Probability distribution of the current state can be determined using marginalization:
  \[
  p(x_k | y_{1:k-1}) = \int p(x_k, x_{k-1} | y_{1:k-1}) \ dx_{k-1} \\
  = \int p(x_k | x_{k-1}) \ p(x_{k-1} | y_{1:k-1}) \ dx_{k-1}
  \]

• This is known as the Chapman-Kolmogorov equation
Bayesian Optimal Filter – Update Step

- Given a new sensor measurement $y_k$, update the distribution of $x_k$ to be consistent with $y_k$

$$p(x_k, \mid y_{1:k}) = Z_k^{-1} p(y_k \mid x_k, y_{1:k-1}) p(x_k, \mid y_{1:k-1})$$

$$= Z_k^{-1} p(y_k \mid x_k) p(x_k, \mid y_{1:k-1})$$

- Normalizer
- Measurement likelihood
- Prior: follows from the prediction step (Chapman-Kolmogorov Equation)
Bayesian Optimal Filter – Summary

• Initialization
  • Initial state given by the prior distribution \( p(x_0) \)

• Prediction
  • Given past measurements, estimate the probability over the current state
    \[
    p(x_k, | y_{1:k-1}) = \int p(x_k | x_{k-1}) p(x_{k-1} | y_{1:k-1}) \, dx_{k-1}
    \]

• Update
  • Given new measurement, update probability of current state to be consistent with the measurement
    \[
    p(x_k, | y_{1:k}) = Z_k^{-1} p(y_k | x_k) p(x_k, | y_{1:k-1})
    \]

• Normalization constant
  • \( Z_k = \int p(y_k | x_k) p(x_k, | y_{1:k-1}) \, dx_k \)
A closed form solution for the Bayesian optimal filter for linear and Gaussian equations.

Assumes system can be modeled using the linear Gauss-Markov model:

Transition matrix of the system dynamics:

\[ x_k = A_{k-1} x_{k-1} + q_k \]

White process noise \( N(0, Q) \)

Measurement model matrix:

\[ y_k = H_k x_k + r_k \]

White measurement noise \( N(0, R) \)

\[
p(x_k | x_{k-1}) = N(x_k | A_{k-1} x_{k-1}, Q)
p(y_k | x_k) = N(y_k | H_k x_k, R)
\]
Kalman Filter – Prediction Step

• Prediction equation
  • Assume posterior of previous step is Gaussian
    \[ p(x_{k-1} \mid y_{1:k-1}) = N(x_{k-1} \mid m_{k-1}, P_{k-1}) \]
  • From the Chapman-Kolmogorov equation we get
    \[ p(x_k, y_{1:k-1}) = \int p(x_k \mid x_{k-1}) p(x_{k-1} \mid y_{1:k-1}) \, dx_{k-1} \]
    \[ = \int N(x_k \mid A_{k-1} x_{k-1}, Q) N(x_{k-1} \mid m_{k-1}, P_{k-1}) \, dx_{k-1} \]
    \[ = N(x_k \mid A_{k-1} m_{k-1}, A_{k-1} P_{k-1} A_{k-1}^T + Q) \]
Kalman Filter – Prediction Step

- The probability distribution $p(x_k|y_{1:k-1})$ is represented by two variables:
  - Estimated mean of the state variable
  - Estimate covariance matrix of the state variable
- The prediction step calculates a priori estimates of these variables
  - *A priori* estimate of the mean value of state:
    $$m^*_k \sim A_{k-1} m_{k-1}$$
  - *A priori* estimate of covariance matrix of current state:
    $$P^*_k \sim A_{k-1} P_{k-1} A^{T}_{k-1} + Q$$
Kalman Filter – Update Step

- The joint distribution of $x_k$ and $y_k$ is given by

$$p(x_k, y_k \mid y_{1:k-1}) = p(y_k \mid x_k) p(x_k \mid y_{1:k-1})$$

$$= N(y_k \mid H_k x_k, R) \ast N(x_k \mid A_{k-1} m_{k-1}, A_{k-1} P_{k-1} A_{k-1}^T + Q)$$

$$p(x_k, y_k \mid y_{1:k-1}) \sim N \left( \begin{bmatrix} x_k \\ y_k \end{bmatrix} \bigg| \begin{bmatrix} m_k^* \\ H_k m_k^* \end{bmatrix} \right) \begin{pmatrix} P_k^* & P_k^* H_k^T \\ H_k P_k^* & H_k P_k^* H_k^T + R \end{pmatrix}$$
Kalman Filter – Update Step

- We have
  \[ p(x_k, | y_{1:k}) = p(x_k, | y_k, y_{1:k-1}) \]
- Following the identities given in the Appendix, we can derive the following estimates:

\[
\begin{align*}
  m_k &\sim m^*_k + P^*_k H^T_k \left( H_k P^*_k H^T_k + R_k \right)^{-1} (y_k - H_k m^*_k) \\
  P_k &\sim P^*_k - P^*_k H^T_k \left( H_k P^*_k H^T_k + R_k \right)^{-1} P^*_k H^T_k 
\end{align*}
\]
Kalman Filter – Update Step

\[ m_k \sim m_k^* + P_k^* H_k^T \left( H_k P_k^* H_k^T + R_k \right)^{-1} (y_k - H_k m_k^*) \]

\[ P_k \sim P_k^* - P_k^* H_k^T \left( H_k P_k^* H_k^T + R_k \right)^{-1} P_k^* H_k^T \]
Kalman Filter – Summary

- Prediction step
  - A priori mean and covariance estimated by propagating the values of the previous state using system dynamics

- Update step
  - Mean given by the combination of a priori state estimate, Kalman gain and measurement residual
  - Covariance given by the combination of a priori covariance estimate, Kalman gain and a term that corresponds to the Kalman gain multiplied by covariance residual
Kalman Filter – Example

- Assume we are tracking the 2D position of an object, its velocity and orientation of motion
- State model $x_k = A_{k-1} x_{k-1} + Q$

$$A = \begin{pmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad x_k = \begin{bmatrix} x \\ y \\ v \cos \alpha \\ v \sin \alpha \end{bmatrix}$$

$Q \sim \mathcal{N}(0, I\sigma)$
Kalman Filter – Example

• Measurement model \( y_k = H_k x_k + R \) given by:

\[
y_k = \begin{bmatrix}
x^*
y^*
v^* \cos \alpha 
v^* \sin \alpha
\end{bmatrix}
\]

\( R \sim \mathcal{N}(0, \mathbf{I} \varepsilon) \)

\( H = \mathbf{I} \)
Kalman Filter – Example
Kalman Filter – Properties

- Only applicable to linear Gaussian models. For non-linear cases need other techniques
  - Extended Kalman filter
  - Particle filtering
- If model is time-invariant, Kalman gain converges to a constant and filter becomes stationary
- Measurements affect only mean estimates, sequence of Kalman gains can be calculated offline and stored for later use
Particle Filtering

- In real world applications noise is often non-Gaussian and the state space is non-linear
  \[ x_k = z(x_{k-1}) + q_k \]
  \[ y_k = h(x_k) + r_k \]

- Particle filtering uses Monte Carlo integration recursively to approximate the filtering distribution

- As shown earlier, for function \( g(\cdot) \) and distribution \( f(x) \), we have:
  \[ E(g(x)) = \int g(x) f(x) \, dx \]

- When \( f(x) \) equals the filtering density, we get:
  \[ E(g(x)) = \int g(x) \, p(x_k, | y_{1:k}) \, dx \approx \frac{1}{K} \sum_j g(x_{k,j}) \]
Particle Filtering – Sequential Importance Sampling

• Most particle filters variants of the sequential importance sampling algorithm
  • Recursive Monte Carlo implementation of importance sampling (see slides 8-11)
• Approximates filtering distribution by a weighted set of particles
  • \( \int g(x) \ p(x_k, y_{1:k}) \ dx \approx \sum_j w^j g(x_k^j) \)
  • \( w^j \) is the **importance weight** of particle
  • Require that \( \sum_j w^j = 1 \)
Sequential Importance Sampling

Overview

- **Initialization:**
  - Draw $K$ particles according to prior distribution
  - Set $w^j = 1/ K$ for all particles
- **Estimation step:**
  - Draw $K$ samples from proposal distribution:
    \[ x_k^j \sim \pi(x_k \mid x_k^j, y_{1:k}) \]
  - Update importance weight of particle $j$
    \[ w_k^j = w_{k-1}^j p(y_k \mid x_k^j) p(x_k^j \mid x_{k-1}^j) / \pi(x_k^j, x_{1:k-1}^j, y_{1:k}) \]
  - Normalize weights so that $\sum_j w_k^j = 1$
  - State of filter can be estimated using $\sum_j w_k^j x_k^j$
Particle Filtering – Sequential Importance Sampling

- Performance depends on the choice of proposal distribution $\pi(x_k \mid x_{1:k-1}, y_{1:k})$
- Optimal proposal distribution:
  \[
  \pi(x_k, \mid x_{1:k-1}, y_{1:k}) = p(x_k \mid x_{k-1}, y_k)
  \]
- Transition prior:
  \[
  \pi(x_k, \mid y_{1:k}) = p(x_k \mid x_{k-1}) \Rightarrow \\
  w_k^j = w_{k-1}^j p(y_k \mid x_k^j)
  \]
Particle Filtering – Resampling

- The filter is **degenerate** when all but one importance weight are close to zero
- Resampling used to overcome degeneracy
  - Replicate particles in proportion to their weights
  - Performed using random sampling, e.g., Bootstrapping
- Need for resampling can be determined by estimating the **effective** number of particles
  - \( N_{\text{EFF}} = \frac{1}{\sum_j (w_k^j)^2} \)
Resampling – Example

<table>
<thead>
<tr>
<th>Particle index</th>
<th>Weights before resampling</th>
<th>Index of particles selected through resampling</th>
<th>Weights after resampling</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0016</td>
<td>2</td>
<td>0.1</td>
</tr>
<tr>
<td>2</td>
<td>0.7507</td>
<td>2</td>
<td>0.1</td>
</tr>
<tr>
<td>3</td>
<td>0.0000</td>
<td>2</td>
<td>0.1</td>
</tr>
<tr>
<td>4</td>
<td>0.1028</td>
<td>7</td>
<td>0.1</td>
</tr>
<tr>
<td>5</td>
<td>0.0632</td>
<td>2</td>
<td>0.1</td>
</tr>
<tr>
<td>6</td>
<td>0.0000</td>
<td>7</td>
<td>0.1</td>
</tr>
<tr>
<td>7</td>
<td>0.0701</td>
<td>2</td>
<td>0.1</td>
</tr>
<tr>
<td>8</td>
<td>0.0000</td>
<td>2</td>
<td>0.1</td>
</tr>
<tr>
<td>9</td>
<td>0.0116</td>
<td>5</td>
<td>0.1</td>
</tr>
<tr>
<td>10</td>
<td>0.0000</td>
<td>2</td>
<td>0.1</td>
</tr>
</tbody>
</table>

\[ N_{\text{EFF}} = 1.7 \]
Particle Filtering – Example

- Estimate position of object from a known starting point given estimates of velocity and orientation
- For simplicity, we assume Gaussian noise
  \[ v_k = v_{k-1} + \mu \quad \mu \sim \mathcal{N}(0, 0.25) \]
  \[ \alpha_k = \alpha_{k-1} + \epsilon \quad \epsilon \sim \mathcal{N}(0, 12.5) \]
- Particles maintain estimate of velocity and angle
  \[ x_k^j = \{ v_k^j, \alpha_k^j \} \]
- Assume velocity and orientation independent
  \[ v_k, \alpha_k | v_k^j, \alpha_k^j \sim \mathcal{N} \left( \begin{bmatrix} v_k \\ \alpha_k \end{bmatrix}, \begin{bmatrix} v_k^j \\ \alpha_k^j \end{bmatrix} \begin{pmatrix} 0.25 & 0 \\ 0 & 12.5 \end{pmatrix} \right) \]
Particle Filtering – Example

SIS

True value

SIR
Particle Filtering – Extensions

• Auxiliary particle filters
  • Sampling utilizes an auxiliary variable, can be used to reduce variance of estimated states

• Rao-Blackwellization
  • Estimate components with linear dynamics analytically (Kalman filter) and apply particle filters only for components with non-linear parts

• KLD sampling
  • Particle filter that adapts the number of particles based on the uncertainty in the current state
Particle Filtering – Summary

• Technique that uses Monte Carlo simulations to implement sequential Bayesian optimal filtering
  • SIS: Sequential Importance Sampling
    – Represent filtering distribution using a weighted set of particles
    – Proposal distribution used to sample state of particles
  • SIR: Sequential Importance Resampling
    – Overcome particle degeneracy by resampling particles when effective number of particles small
• Trade-off between efficiency and number of particles
  • Higher number of particles provides more accurate approximation, but requires more computational resources
Summary

• State space
  • Mathematical model of a physical system
  • Consists of input, output and control variables

• Probabilistic state space model
  • State space model where measurements are random variables and estimates are probabilistic

• Bayesian optimal filter
  • Probabilistic recursive estimator
  • Provides a probabilistic distribution over the state of a dynamical system
Summary

• Kalman filter
  • Closed form solution of the Bayesian optimal filter for linear and Gaussian state space models

• Particle filters
  • Sequential Monte Carlo technique that approximate Bayesian optimal filter
  • Enable to handle non-linear relationships and non-Gaussian noise
  • Resampling essential to avoid degeneracy
Literature


Literature


• Fox, D., Adapting the sample size in particle filters through KLD-sampling *International journal of robotics research*, 2003, 22, 985-1003

• Gilks, W.; Spiegelhalter, D. & Richardson, S., Markov Chain Monte Carlo in Practice *Chapman & Hall/CRC*, 1996

Appendix –
Multivariate Gaussian Distribution

- Multivariate Gaussian probability density:
\[ \mathcal{N}(x|m, P) = \frac{1}{(2\pi)^{n/2}|P|^{1/2}} \exp\left(-\frac{1}{2}(x-m)^T P^{-1} (x-m)\right) \]

- Let \( x \) and \( y \) have Gaussian distributions
\[ p(x) = \mathcal{N}(x|m, P) \quad p(y|x) = \mathcal{N}(y|Hx, R) \]

- Then the joint and marginal distributions are
\[ \begin{pmatrix} x \\ y \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} m \\ Hm \end{pmatrix}, \begin{pmatrix} P & PH^T \\ HP & HPH^T + R \end{pmatrix}\right) \]
\[ y \sim \mathcal{N}(Hm, HPH^T + R) \]
Appendix –
Multivariate Gaussian Distribution

• If random variables $x$ and $y$ have the joint Gaussian probability density

$$
\begin{pmatrix} x \\ y \end{pmatrix} \sim \mathcal{N}\left( \begin{pmatrix} a \\ b \end{pmatrix}, \begin{pmatrix} A & C \\ C^T & B \end{pmatrix} \right)
$$

• Then the marginal and conditional densities of $x$ and $y$ are given as follows:

$$
\begin{align*}
x & \sim \mathcal{N}(a, A) \\
y & \sim \mathcal{N}(b, B) \\
x | y & \sim \mathcal{N}(a + CB^{-1}(y - b), A - CB^{-1}C^T) \\
y | x & \sim \mathcal{N}(b + C^TA^{-1}(x - a), B - C^TA^{-1}C)
\end{align*}
$$