

## Handling uncertainty over time: predicting, estimating, recognizing, learning

© Chris Atkeson 2004

## Why do we care?

- Speech recognition makes use of dependence of words and phonemes across time.
- Knowing where your robot is makes use of reasoning about processes that unfold over time.
- So does most reasoning, actually.
- Medical diagnosis, politics, stock market, and the choices you make every day, for example.

## Examples

- How tall am I? (continuous)
- What room am I in? (discrete)

## Discrete states and measurements

## What room am I in?

- State: Kitchen vs. Den
- Measurement: Do I smell food?
- Priors:  $p(K) = 0.7$ ,  $p(D) = ? = 1 - p(K)$
- Measurement model:  $p(F|K) = 0.8$ ,  $p(F|D) = 0.1$ ,  $p(\sim F|K) = ?$ ,  $p(\sim F|D) = ?$
- What is  $p(K|F)$ ?  $p(K|\sim F)$ ?

## Bayes' Rule

- $p(A \& B) = p(A|B)p(B) = p(B|A)p(A)$
- So  $p(A|B) = p(B|A)p(A)/p(B)$
- Example:  $p(K|F) = p(F|K)p(K)/p(F)$
- Note that  $p(F) = p(F|K)p(K) + p(F|D)p(D)$

### The Numbers

- $p(F) = p(F|K)p(K) + p(F|D)p(D)$
- $0.59 = 0.8 \cdot 0.7 + 0.1 \cdot 0.3$
- $p(K|F) = p(F|K)p(K)/p(F)$
- $0.95 = 0.8 \cdot 0.7 / 0.59$
- $p(K|\sim F) = p(\sim F|K)p(K)/p(\sim F)$
- $p(\sim F) = 1 - p(F)$
- So  $p(K|\sim F) = 0.34 = 0.2 \cdot 0.7 / 0.41$

### What happens with a second measurement?

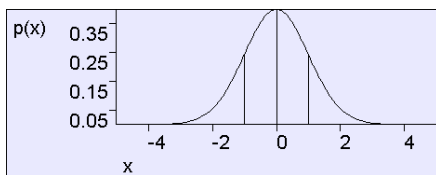
- Same formulas, new prior.
- First measurement: F:  $p(K|F) = 0.95$
- State  $\rightarrow p(F, F|K) = p(F|K)$ , etc.
- $p(F, F|F) = p(F, F|K)p(K|F) + p(F, F|D)p(D|F) = 0.77$
- $p(K|F, F) = p(F, F|K) \cdot p(K|F) / p(F, F|F)$
- $0.99 = 0.8 \cdot 0.95 / 0.77$
- $p(K|F, \sim F) = p(F, \sim F|K) \cdot p(K|F) / p(F, \sim F|F)$
- $0.83 = 0.2 \cdot 0.95 / 0.23$

Draw a picture....

Continuous states and measurements

### Unit variance Gaussian

$$p(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right)$$

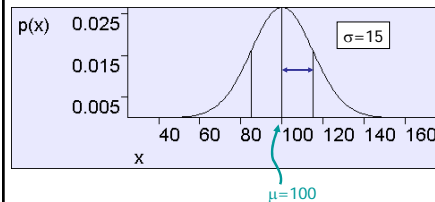


$$E[X] = 0$$

$$\text{Var}[X] = 1$$

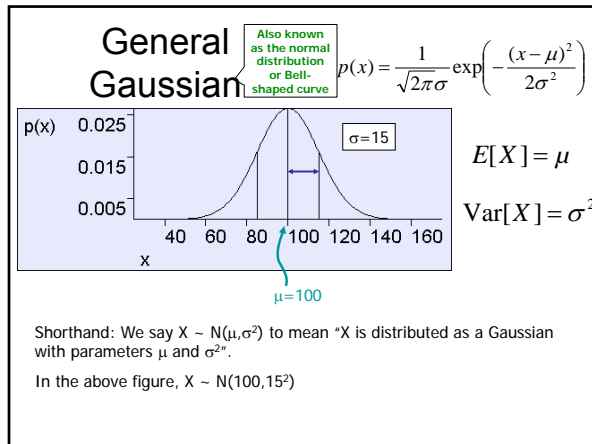
### General Gaussian

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$



$$E[X] = \mu$$

$$\text{Var}[X] = \sigma^2$$



### The Central Limit Theorem

- If  $(X_1, X_2, \dots, X_n)$  are i.i.d. continuous random variables
- Then define  $z = f(x_1, x_2, \dots, x_n) = \frac{1}{n} \sum_{i=1}^n x_i$
- As  $n \rightarrow \infty$ ,  $p(z) \rightarrow$  Gaussian with mean  $E[X_i]$  and variance  $\text{Var}[X_i]$

Somewhat of a justification for assuming Gaussian noise is common

### Combining Measurements: 1D

- True value  $x$
- Measurements  $m_1, m_2$ :  $E(m_1 - x) = 0$ ,  $\text{Var}(m_1) = \sigma_1^2$ ,  $E(m_2 - x) = 0$ ,  $\text{Var}(m_2) = \sigma_2^2$ , independent
- Linear estimate  $x = k_1 m_1 + k_2 m_2$
- Unbiased estimate means  $k_2 = 1 - k_1$  so  $E(x) = x$
- Minimize  $\text{Var}(x) = k_1^2 \sigma_1^2 + (1 - k_1)^2 \sigma_2^2$
- So  $\partial \text{Var}(x) / \partial k_1 = 0 \rightarrow 2k_1(\sigma_1^2 + \sigma_2^2) - 2\sigma_2^2 = 0$
- So  $k_1 = \sigma_2^2 / (\sigma_1^2 + \sigma_2^2)$ ,  $k_2 = \sigma_1^2 / (\sigma_1^2 + \sigma_2^2)$
- So  $\text{Var}(x) = \sigma_1^2 \sigma_2^2 / (\sigma_1^2 + \sigma_2^2)$
- What happens when  $\sigma_2^2 = 0$ ?  $\sigma_2^2 = \infty$ ?
- BLUE: Best Linear Unbiased Estimator

### Handling Measurements Over Time

- Just keep doing what we did before:  
 $p(\text{state} | \text{new-measurement}) = \dots$

### What is a "state"

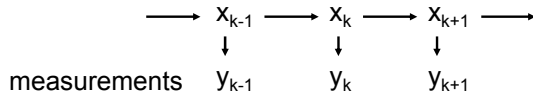
- Everything you need to know to make the best prediction about what happens next.
- Depends how you define the "system" you care about.
- States are called  $x$  or  $s$ . Dependence on time can be indicated by  $x(t)$ .
- States can be discrete or continuous.
- All researchers tend to say "state" when they mean "some features derived from the state". This should be discouraged.
- A "belief state" is your knowledge about the state, which is typically a probability  $p(x)$ .
- Processes with state are called Markov processes.

### Dealing with time

- The concept of state gives us a handy way of thinking about how things evolve over time.
- We will use discrete time, for example 0.001, 0.002, 0.003, ...
- State at time  $t_k$ ,  $x(t_k)$ , will be written  $x_k$  or  $x[k]$ .
- Deterministic state transition function  $x_{k+1} = f(x_k)$
- Stochastic state transition function  $p(x_{k+1} | x_k)$
- Mildly stochastic state transition function  $x_{k+1} = f(x_k) + \varepsilon$ , with  $\varepsilon$  being Gaussian.

### Hidden state

- Sometimes the state is directly measurable/observable.
- Sometimes it isn't. Then you have "hidden state" and a "hidden Markov model" or HMM.
- Examples: Do you have a disease? What am I thinking about? What is wrong with the Mars rover? Where is the Mars rover?



### Measurements

- Measurements (y) are also called evidence (e) and observables (o).
- Measurements can be discrete or continuous.
- Deterministic measurement function  $y_k = g(x_k)$
- Stochastic measurement function  $p(y_k|x_k)$
- Mildly stochastic measurement function  $y_k = g(x_k) + v$ , with  $v$  being Gaussian.

### Standard problems

- Predict the future.
- Estimate the current state (filtering).
- Estimate what happened in the past (smoothing).
- Find the most likely state trajectory (sequence/trajectory (speech) recognition).
- Learn about the process (learn state transition and measurement models).

### Prediction, Case 0

- Deterministic state transition function  $x_{k+1} = f(x_k)$  and known state  $x_k$ : Just apply  $f()$  n times to get  $x_{k+n}$ .
- When we worry about learning the state transition function and the fact that it will always have errors, the question will arise: To predict  $x_{k+n}$ , is it better to learn  $x_{k+1} = f_1(x_k)$  and iterate, or learn  $x_{k+n} = f_n(x_k)$  directly?

### Prediction, Case 1

- Stochastic state transition function  $p(x_{k+1}|x_k)$ , discrete states, belief state  $p(x_k)$
- Use tables to represent  $p(x_k)$
- Propagate belief state:  $p(x_{k+1}) = \sum p(x_{k+1}|x_k)p(x_k)$

Matrix notation:

Vector  $p_k$ , Transition matrix  $M$ ,  $M_{ij} = p(x_i|x_j)$ ;  $i, j$ , components, not time.

Propagate belief state:  $p_{k+1} = Mp_k$

Stationary distribution  $M^\infty = \lim(n \rightarrow \infty) M^n$

Mixing time:  $n$  for which  $M^n \approx M^\infty$

### Prediction, Case 2

- Stochastic state transition function  $p(x_{k+1}|x_k)$ , continuous states, belief state  $p(x_k)$
- Propagate belief state analytically if possible  $p(x_{k+1}) = \int p(x_{k+1}|x_k)p(x_k)dx_k$
- Particle filtering (actually many ways to implement).
- Sample  $p(x_k)$ .
- For each sample, sample  $p(x_{k+1}|x_k)$ .
- Normalize/resample resulting samples to get  $p(x_{k+1})$ .
- Iterate to get  $p(x_{k+n})$

### Prediction, Case 3

- Mildly stochastic state transition function with  $p(x_k)$  being  $N(\mu, \Sigma_x)$ ,  $x_{k+1} = f(x_k) + \varepsilon$ , with  $\varepsilon$  being  $N(0, \Sigma_\varepsilon)$ ,  $\varepsilon$  independent of process.
- $E(x_{k+1}) \approx f(\mu)$
- $A = \partial f / \partial x$
- $\text{Var}(x_{k+1}) \approx A \Sigma_x A^T + \Sigma_\varepsilon$
- $p(x_{k+1})$  is  $N(E(x_{k+1}), \text{Var}(x_{k+1}))$ .
- Exact if  $f()$  linear.
- Iterate to get  $p(x_{k+n})$ .
- Much simpler than particle filtering.

### Filtering, in general

- Start with  $p(x_{k-1}^+)$
- Predict  $p(x_k^-)$
- Apply measurement using Bayes' Rule to get  $p(x_k^+) = p(x_k | y_k)$
- $p(x_k | y_k) = p(y_k | x_k) p(x_k) / p(y_k)$
- Sometimes we ignore  $p(y_k)$  and just renormalize as necessary, so all we have to do is  $p(x_k | y_k) = \alpha p(y_k | x_k) p(x_k)$

### Filtering, Case 1

- Stochastic state transition function  $p(x_{k+1} | x_k)$ , discrete states, belief state  $p(x_k)$ ,  $p(y_k | x_k)$
- Use tables to represent  $p(x_k)$
- Propagate belief state:

$$p(x_{k+1}^-) = \sum p(x_{k+1} | x_k) p(x_k^+)$$

- Weight each entry by  $p(y_k | x_k)$ :

$$p(x_{k+1}^+) \propto p(y_k | x_k) p(x_{k+1}^-)$$

- Normalize so sum of  $p() = 1$
- This is called a Discrete Bayes Filter

### Filtering, Case 2

- Stochastic state transition function  $p(x_{k+1} | x_k)$ , continuous states, belief state  $p(x_k)$
- Particle filtering (actually many ways to implement).
- Sample  $p(x_k)$ .
- For each sample, sample  $p(x_{k+1} | x_k)$ .
- Weight each sample by  $p(y_k | x_k)$ .
- Normalize/resample resulting samples to get  $p(x_{k+1})$ .
- Iterate to get  $p(x_{k+n})$

### EEE 581 Lecture 16

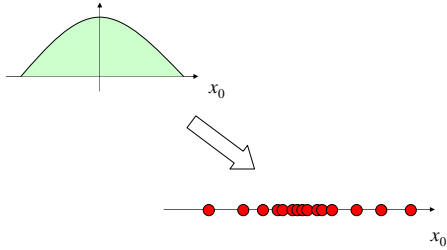
#### Particle Filters: a Gentle Introduction

<http://www.fulton.asu.edu/~morrell/581/>

### Particle Filter Algorithm

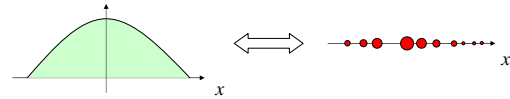
- Create particles as samples from the initial state distribution  $p(x_0)$ .
- For  $k$  going from 1 to  $K$ 
  - Update each particle using the state update equation.
  - Compute weights for each particle using the observation value.
  - (Optionally) resample particles.

## Initial State Distribution: Samples Only



## Samples and Weights

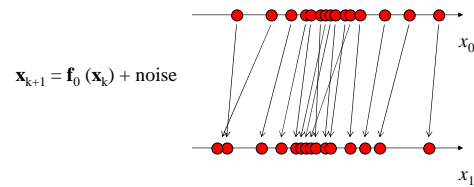
- Each particle has a value and a weight



## Importance Sampling

- Ideally, the particles would represent samples drawn from the distribution  $p(x)$ .
  - In practice, we usually cannot get  $p(x)$  in closed form; in any case, it would usually be difficult to draw samples from  $p(x)$ .
- We use importance sampling:
  - Particles are drawn from an *importance distribution*.
  - Particles are weighted by *importance weights*.

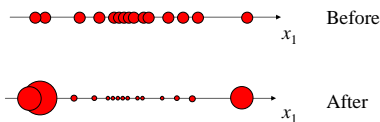
## State Update



Things are more complicated if we have multimodal  $p(x_{k+1}|x_k)$

## Compute Weights

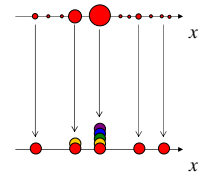
Use  $p(y|x)$  to alter weights



Can also draw samples with replacement using  $p(y|x) \cdot \text{weight}$  as  $p(\text{selection})$

## Resampling

- In inference problems, most weights tend to zero except a few (from particles that closely match observations), which become large.
- We resample to concentrate particles in regions where  $p(x|y)$  is larger.



## Advantages of Particle Filters

- Under general conditions, the particle filter estimate becomes asymptotically optimal as the number of particles goes to infinity.
- Non-linear, non-Gaussian state update and observation equations can be used.
- Multi-modal distributions are not a problem.
- Particle filter solutions to inference problems are often easy to formulate.

## Disadvantages of Particle Filters

- Naïve formulations of problems usually result in significant computation times.
- It is hard to tell if you have enough particles.
- The best importance distribution and/or resampling methods may be very problem specific.

## Conclusions

Particle filters (and other Monte Carlo methods) are a powerful tool to solve difficult inference problems.

- Formulating a filter is now a tractable exercise for many previously difficult or impossible problems.
- Implementing a filter effectively may require significant creativity and expertise to keep the computational requirements tractable.

## Particle Filtering Comments

- Reinvented many times in many fields: sequential Monte Carlo, condensation, bootstrap filtering, interacting particle approximations, survival of the fittest, ...
- Do you need  $R^d$  samples to cover space?  $R$  is crude measure of linear resolution,  $d$  is dimensionality.
- You maintain a belief state  $p(x)$ . How do you answer the question “Where is the robot now?” mean, best sample, robust mean, max likelihood, ... What happens if  $p(x)$  really is multimodal?

## Return to our regularly scheduled programming ...

- Filtering ...

## Filtering, Case 3

- Mildly stochastic state transition function with  $p(x_k)$  being  $N(\mu, \Sigma_x)$ ,  $x_{k+1} = f(x_k) + \varepsilon$ , with  $\varepsilon$  being  $N(0, \Sigma_\varepsilon)$  and independent of process.
- Mildly stochastic measurement function  $y_k = g(x_k) + v$ , with  $v$  being  $N(0, \Sigma_v)$  and independent of everything else.
- This will lead to Kalman Filtering
- Nonlinear  $f()$  or  $g()$  means you are doing Extended Kalman Filtering (EKF).

### Filtering, Case 3 Prediction Step

- $E(x_{k+1}^-) \approx f(\mu)$
- $A = \partial f / \partial x$
- $\text{Var}(x_{k+1}^-) \approx A \Sigma_x A^T + \Sigma_\epsilon$
- $p(x_{k+1}^-)$  is  $N(E(x_{k+1}^-), \text{Var}(x_{k+1}^-))$

### Filtering, Case 3 Measurement Update Step

- $E(x_k^+) \approx E(x_k^-) + K_k(y_k - g(E(x_k^-)))$
- $C = \partial g / \partial x$
- $\Sigma_k^- = \text{Var}(x_k^-)$
- $\text{Var}(x_k^+) \approx \Sigma_k^- - K_k C \Sigma_k^-$
- $S_k = C \Sigma_k^- C^T + \Sigma_v$
- $K_k = \Sigma_k^- C^T S_k^{-1}$
- $p(x_k^+)$  is  $N(E(x_k^+), \text{Var}(x_k^+))$
- This all comes from Gaussian lecture ...

### Unscented Filter

- Numerically find best fit Gaussian instead of analytical computation.
- Good if  $f()$  or  $g()$  strongly nonlinear.

### What I would like to see

- Combine particle system and Kalman filter, so each particle maintains a simple distribution, instead of just a point estimate.

### Smoothing, in general

- Have  $y_{1:N}$ , want  $p(x_k | y_{1:N})$
- Know how to compute  $p(x_k | y_{1:k})$  from filtering slides
- $p(x_k | y_{1:N}) = p(x_k | y_{1:k}, y_{k+1:N})$
- $p(x_k | y_{1:N}) \propto p(x_k | y_{1:k}) p(y_{k+1:N} | y_{1:k}, x_k)$
- $p(x_k | y_{1:N}) \propto p(x_k | y_{1:k}) p(y_{k+1:N} | x_k)$
- $p(y_{k+1:N} | x_k) = \int p(y_{k+1:N} | x_k, x_{k+1}) p(x_{k+1} | x_k) dx_{k+1}$
- $= \int p(y_{k+1:N} | x_{k+1}) p(x_{k+1} | x_k) dx_{k+1}$
- $= \int p(y_{k+1} | x_{k+1}) p(y_{k+2:N} | x_{k+1}) p(x_{k+1} | x_k) dx_{k+1}$
- Note recursion implied by  $p(y_{k+i+1:N} | x_{k+i})$

### Smoothing, general comments

- Need to maintain distributional information at all time steps from forward filter.
- Case 1: discrete states: forward/backward algorithm.
- Case 2: continuous states, nasty dynamics or noise: particle smoothing (expensive).
- Case 3: continuous states, Gaussian noise: Kalman smoother.



## Finding most likely state trajectory

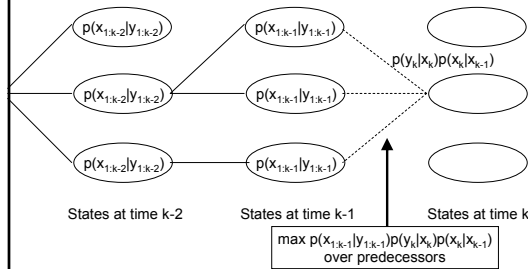
- Goal in speech recognition
- $p(x_1, x_2, \dots, x_N | y_{1:N}) \neq p(x_1 | y_{1:N}) p(x_2 | y_{1:N}) \dots p(x_N | y_{1:N})$
- Are we screwed? Computing joint probability is hard!

## Viterbi Algorithm

- $\max p(x_{1:k} | y_{1:k})$
- $= \max p(y_{1:k} | x_{1:k}) p(x_{1:k})$
- $= \max p(y_{1:k-1}, y_k | x_{1:k}) p(x_{1:k})$
- $= \max p(y_{1:k-1} | x_{1:k-1}) p(y_k | x_k) p(x_{1:k})$
- $= \max p(y_{1:k-1} | x_{1:k-1}) p(y_k | x_k) p(x_k | x_{1:k-1}) p(x_{1:k-1})$
- $= \max [p(y_{1:k-1} | x_{1:k-1}) p(x_{1:k-1})] p(y_k | x_k) p(x_k | x_{1:k-1})$
- $= \max p(x_{1:k-1} | y_{1:k-1}) p(y_k | x_k) p(x_k | x_{k-1})$
- Note recursion
- Do we evaluate this over all possible sequences?

## Viterbi Algorithm (2)

- Use dynamic programming



## Viterbi Algorithm (3)

- Well, this still only really works for discrete states.
- Continuous states have too many possible states at each step.
- D dimensions, R resolution in each dimension implies  $R^D$  states at each time step.
- Ask me about local sequence maximization.

## Learning

- Given data, want to learn dynamic/transition and sensor models.
- Smooth, choose most likely state at each time, learn models, iterate.
- This is known as the EM algorithm.
- Discrete case: Baum-Welch Algorithm