CS 188: Artificial Intelligence
Dynamic Bayes Nets, Particle Filtering

Instructor: Stuart Russell and Dawn Song (Lecturer: Saagar Sanghavi)

University of California, Berkeley
Filtering, or monitoring, is the task of tracking the distribution $B_t(X) = P_t(X_t | e_1, ..., e_t)$ (the belief state) over time.

- We start with $B_1(X)$ in an initial setting, usually uniform.
- As time passes, or we get observations, we update $B(X)$.
- The Kalman filter was invented in the 60’s and first implemented as a method of trajectory estimation for the Apollo program.
Inference: Find State Given Evidence

- We are given evidence at each time and want to know

\[ B_t(X) = P(X_t | e_{1:t}) \]

- Idea: start with \( P(X_1) \) and derive \( B_t \) in terms of \( B_{t-1} \)
  - equivalently, derive \( B_{t+1} \) in terms of \( B_t \)
Two Steps: Passage of Time + Observation

\[ B(X_t) = P(X_t | e_{1:t}) \]
\[ B'(X_{t+1}) \]

\[ B(X_{t+1}) \]
Inference: Base Cases

\[ P(X_1 | e_1) = \frac{P(X_1, e_1)}{\sum_{x_1} P(x_1, e_1)} \]

\[ P(X_1 | e_1) = \frac{P(e_1 | X_1) P(X_1)}{\sum_{x_1} P(e_1 | x_1) P(x_1)} \]

\[ P(X_2) = \sum_{x_1} P(x_1, X_2) \]

\[ P(X_2) = \sum_{x_1} P(X_2 | x_1) P(x_1) \]
Passage of Time

- Assume we have current belief $P(X \mid \text{evidence to date})$
  
  $B(X_t) = P(X_t \mid e_{1:t})$

- Then, after one time step passes:
  
  $P(X_{t+1} \mid e_{1:t}) = \sum_{x_t} P(X_{t+1}, x_t \mid e_{1:t})$
  
  $= \sum_{x_t} P(X_{t+1} \mid x_t, e_{1:t}) P(x_t \mid e_{1:t})$
  
  $= \sum_{x_t} P(X_{t+1} \mid x_t) P(x_t \mid e_{1:t})$

- Basic idea: beliefs get “pushed” through the transitions
  - With the “B” notation, we have to be careful about what time step $t$ the belief is about, and what evidence it includes

- Or compactly:
  
  $B'(X_{t+1}) = \sum_{x_t} P(X' \mid x_t) B(x_t)$
Example: Passage of Time

- As time passes, uncertainty “accumulates”

(Transition model: ghosts usually go clockwise)
Assume we have current belief \( P(X \mid \text{previous evidence}) \):

\[
B'(X_{t+1}) = P(X_{t+1} \mid e_{1:t})
\]

Then, after evidence comes in:

\[
P(X_{t+1} \mid e_{1:t+1}) = \frac{P(X_{t+1}, e_{t+1} \mid e_{1:t})}{P(e_{t+1} \mid e_{1:t})}
\]

\[
\propto_{X_{t+1}} P(X_{t+1}, e_{t+1} \mid e_{1:t})
\]

\[
= P(e_{t+1} \mid e_{1:t}, X_{t+1})P(X_{t+1} \mid e_{1:t})
\]

\[
= P(e_{t+1} \mid X_{t+1})P(X_{t+1} \mid e_{1:t})
\]

Or, compactly:

\[
B(X_{t+1}) \propto_{X_{t+1}} P(e_{t+1} \mid X_{t+1})B'(X_{t+1})
\]

Basic idea: beliefs “rewighted” by likelihood of evidence

Unlike passage of time, we have to renormalize
Example: Weather HMM

\[ R_t \rightarrow P(R_{t+1} \mid R_t) \]

\[ B(+) = 0.5 \quad B(-) = 0.5 \]

\[ B'(+) = 0.5 \quad B'(-) = 0.5 \]

\[ B(+) = 0.818 \quad B(-) = 0.182 \]

\[ B'(+) = 0.627 \quad B'(-) = 0.373 \]

\[ B(+) = 0.883 \quad B(-) = 0.117 \]

<table>
<thead>
<tr>
<th>( R_t )</th>
<th>( R_{t+1} )</th>
<th>( P(R_{t+1} \mid R_t) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>+r</td>
<td>+r</td>
<td>0.7</td>
</tr>
<tr>
<td>+r</td>
<td>-r</td>
<td>0.3</td>
</tr>
<tr>
<td>-r</td>
<td>+r</td>
<td>0.3</td>
</tr>
<tr>
<td>-r</td>
<td>-r</td>
<td>0.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( R_t )</th>
<th>( U_t )</th>
<th>( P(U_t \mid R_t) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>+r</td>
<td>+u</td>
<td>0.9</td>
</tr>
<tr>
<td>+r</td>
<td>-u</td>
<td>0.1</td>
</tr>
<tr>
<td>-r</td>
<td>+u</td>
<td>0.2</td>
</tr>
<tr>
<td>-r</td>
<td>-u</td>
<td>0.8</td>
</tr>
</tbody>
</table>
Online Belief Updates

- Every time step, we start with current $P(X | \text{evidence})$

- We update for time:

  $$P(x_t|e_{1:t-1}) = \sum_{x_{t-1}} P(x_{t-1}|e_{1:t-1}) \cdot P(x_t|x_{t-1})$$

- We update for evidence:

  $$P(x_t|e_{1:t}) \propto \prod_{X} P(x_t|e_{1:t-1}) \cdot P(e_t|x_t)$$

- The forward algorithm does both at once (and does not normalize)
The Forward Algorithm

- We are given evidence at each time and want to know

\[ B_t(X) = P(X_t|e_{1:t}) \]

- We can derive the following updates

\[
P(x_t|e_{1:t}) \propto X_t P(x_t, e_{1:t})
\]

\[
= \sum_{x_{t-1}} P(x_{t-1}, x_t, e_{1:t})
\]

\[
= \sum_{x_{t-1}} P(x_{t-1}, e_{1:t-1}) P(x_t|x_{t-1}) P(e_t|x_t)
\]

\[
= P(e_t|x_t) \sum_{x_{t-1}} P(x_t|x_{t-1}) P(x_{t-1}, e_{1:t-1})
\]

We can normalize as we go if we want to have \( P(x|e) \) at each time step, or just once at the end...
Forward / Viterbi Algorithms

**Forward Algorithm (Sum)**

$$f_t[x_t] = P(x_t, e_{1:t})$$

$$= P(e_t|x_t) \sum_{x_{t-1}} P(x_t|x_{t-1}) f_{t-1}[x_{t-1}]$$

**Viterbi Algorithm (Max)**

$$m_t[x_t] = \max_{x_{1:t-1}} P(x_{1:t-1}, x_t, e_{1:t})$$

$$= P(e_t|x_t) \max_{x_{t-1}} P(x_t|x_{t-1}) m_{t-1}[x_{t-1}]$$
Particle Filtering
Approximate Inference on HMMs

- When $|X|$ is more than $10^6$ or so (e.g., 3 ghosts in a 10x20 world), exact inference becomes infeasible.
- Likelihood weighting fails completely – number of samples needed grows exponentially with $T$. 

![Graph showing average absolute error against time step for different methods.](image)
The problem: sample state trajectories go off into low-probability regions, ignoring the evidence; too few “reasonable” samples.

Solution: kill the bad ones, make more of the good ones.

This way the population of samples stays in the high-probability region.

This is called **resampling** or survival of the fittest.
Particle Filtering

- Represent belief state by a set of samples
  - Samples are called *particles*
  - Time per step is linear in the number of samples
  - But: number needed may be large

- This is how robot localization works in practice
Our representation of $P(X)$ is now a list of $N$ particles

$P(x)$ approximated by number of particles with value $x$

- So, many $x$ may have $P(x) = 0$!
- More particles => more accuracy
- Usually we want a *low-dimensional* marginal

  - E.g., “Where is ghost 1?” rather than “Are ghosts 1,2,3 in [2,6], [5,6], and [8,11]?”
Particle Filtering: Elapse Time

- Each particle is moved by sampling its next position from the transition model

\[ x' = \text{sample}(P(X'|x)) \]

- This is like prior sampling – samples’ frequencies reflect the transition probabilities

- Here, most samples move clockwise, but some move in another direction or stay in place

- This captures the passage of time
  - If enough samples, close to exact values before and after (consistent)
After observing $e_{t+1}$:

- As in likelihood weighting, weight each sample based on the evidence
  - $w^{(j)} = P(e_{t+1} | x_{t+1}^{(j)})$

- Normalize the weights: particles that fit the data better get higher weights, others get lower weights

Particles:

- $(3,2)$ $w=.9$
- $(2,3)$ $w=.2$
- $(3,2)$ $w=.9$
- $(3,1)$ $w=.4$
- $(3,3)$ $w=.4$
- $(3,2)$ $w=.4$
- $(3,2)$ $w=.9$
- $(2,3)$ $w=.2$
- $(1,3)$ $w=.1$
- $(2,3)$ $w=.2$
- $(3,2)$ $w=.2$
- $(2,2)$ $w=.4$
Rather than tracking weighted samples, we **resample**

_N times, we choose from our weighted sample distribution (i.e., draw with replacement)_

Now the update is complete for this time step, continue with the next one (with weights reset to 1)
Summary: Particle Filtering

- **Particles**: track samples of states rather than an explicit distribution

**Prediction**: 
- Particles: 
  - (3,3)  
  - (2,3)  
  - (3,3)  
  - (3,2)  
  - (3,3)  
  - (3,2)  
  - (1,2)  
  - (3,3)  
  - (3,3)  
  - (2,3)

**Update/Weight**: 
- Particles: 
  - (3,2)  \(w=0.9\)  
  - (2,3)  \(w=0.2\)  
  - (3,2)  \(w=0.9\)  
  - (3,1)  \(w=0.4\)  
  - (3,3)  \(w=0.4\)  
  - (3,2)  \(w=0.9\)  
  - (2,3)  \(w=0.2\)  
  - (3,2)  \(w=0.9\)  
  - (2,2)  \(w=0.4\)

**Resample**: 
- (New) Particles: 
  - (3,2)  
  - (2,2)  
  - (2,3)  
  - (3,3)  
  - (3,2)  
  - (1,3)  
  - (2,3)  
  - (3,2)  
  - (3,2)

Consistency: see proof in AIMA Ch. 14
Video of Demo – Moderate Number of Particles
Video of Demo – One Particle
Video of Demo – Huge Number of Particles
In robot localization:

- We know the map, but not the robot’s position
- Observations may be vectors of range finder readings
- State space and readings are typically continuous so we cannot usually represent or compute an exact posterior
- Particle filtering is a main technique
Particle Filter Localization (Sonar)

Global localization with sonar sensors

[Video: global-sonar-uw-annotated.avi]
Robot Mapping

- **SLAM: Simultaneous Localization And Mapping**
  - Robot does not know map or location
  - State $x_t^{(j)}$ consists of position+orientation, map!
  - (Each map usually inferred exactly given sampled position+orientation sequence)
Particle Filter SLAM – Video
Dynamic Bayes Nets
We want to track multiple variables over time, using multiple sources of evidence.

Idea: Repeat a fixed Bayes net structure at each time.

Variables at time $t$ can have parents at time $t-1$. 

\[
\begin{align*}
G_1^a & \quad G_1^b \\
& \quad E_1^a \\
G_2^a & \quad G_2^b \\
& \quad E_2^a \\
G_3^a & \quad G_3^b \\
& \quad E_3^a
\end{align*}
\]
DBNs and HMMs

- Every HMM is a single-variable DBN
- Every discrete DBN is an HMM
  - HMM state is Cartesian product of DBN state variables

  \[ X_t \xrightarrow{} X_{t+1} \]
  \[ Y_t \xrightarrow{} Y_{t+1} \]
  \[ Z_t \xrightarrow{} Z_{t+1} \]

- Sparse dependencies => exponentially fewer parameters in DBN
  - E.g., 20 state variables, 3 parents each;
    
    DBN has \(20 \times 2^3 = 160\) parameters, HMM has \(2^{20} \times 2^{20} \approx 10^{12}\) parameters
Exact Inference in DBNs

- Variable elimination applies to dynamic Bayes nets
- Offline: “unroll” the network for $T$ time steps, then eliminate variables to find $P(X_T | e_{1:T})$

\[ \begin{align*}
G_1^a & \quad E_1^a \\
G_1^b & \quad E_1^b \\
G_2^a & \quad E_2^a \\
G_2^b & \quad E_2^b \\
G_3^a & \quad E_3^a \\
G_3^b & \quad E_3^b
\end{align*} \]

- Online: eliminate all variables from the previous time step; store factors for current time only
- When in doubt, Bayes Rule it out!
- Problem: largest factor contains all variables for current time (plus a few more)
A particle is a complete sample for a time step

**Initialize**: Generate prior samples for the t=1 Bayes net
- Example particle: $G_1^a = (3,3) \quad G_1^b = (5,3)$

**Elapse time**: Sample a successor for each particle
- Example successor: $G_2^a = (2,3) \quad G_2^b = (6,3)$

**Observe**: Weight each *entire* sample by the likelihood of the evidence conditioned on the sample
- Likelihood: $P(E_1^a \mid G_1^a) \times P(E_1^b \mid G_1^b)$

**Resample**: Select prior samples (tuples of values) in proportion to their likelihood
Toy DBN: heart rate monitoring
CS 188: Artificial Intelligence

Rational Decisions

Instructor: Stuart Russell and Dawn Song (Lecturer: Saagar Sanghavi)

University of California, Berkeley
Utilities
Maximum Expected Utility

- **Principle of maximum expected utility:**
  - A rational agent should choose the action that *maximizes its expected utility, given its knowledge*

- **Questions:**
  - Where do utilities come from?
  - How do we know such utilities even exist?
  - How do we know that averaging even makes sense?
  - What if our behavior (preferences) can’t be described by utilities?
The need for numbers

- For worst-case minimax reasoning, terminal value scale doesn’t matter
  - We just want better states to have higher evaluations (get the ordering right)
  - The optimal decision is invariant under any *monotonic transformation*

- For average-case expectimax reasoning, we need *magnitudes* to be meaningful
Utilities

- Utilities are functions from outcomes (states of the world) to real numbers that describe an agent’s preferences.

- Where do utilities come from?
  - In a game, may be simple (+1/-1)
  - Utilities summarize the agent’s goals
  - Theorem: any “rational” preferences can be summarized as a utility function

- We hard-wire utilities and let behaviors emerge
  - Why don’t we let agents pick utilities?
  - Why don’t we prescribe behaviors?
Utilities: Uncertain Outcomes

Getting ice cream

Get Single

Get Double

Oops

Whew!
Preferences

- An agent must have preferences among:
  - Prizes: $A, B$, etc.
  - Lotteries: situations with uncertain prizes
    $L = [p, A; (1-p), B]$

- Notation:
  - Preference: $A > B$
  - Indifference: $A \sim B$
We want some constraints on preferences before we call them rational, such as:

Axiom of Transitivity: $(A > B) \land (B > C) \implies (A > C)$

For example: an agent with intransitive preferences can be induced to give away all of its money

- If $B > C$, then an agent with $C$ would pay (say) 1 cent to get $B$
- If $A > B$, then an agent with $B$ would pay (say) 1 cent to get $A$
- If $C > A$, then an agent with $A$ would pay (say) 1 cent to get $C$
Rational Preferences

The Axioms of Rationality

Orderability:
\[(A > B) \lor (B > A) \lor (A \sim B)\]

Transitivity:
\[(A > B) \land (B > C) \Rightarrow (A > C)\]

Continuity:
\[(A > B > C) \Rightarrow \exists p \ [p, A; 1-p, C] \sim B\]

Substitutability:
\[(A \sim B) \Rightarrow [p, A; 1-p, C] \sim [p, B; 1-p, C]\]

Monotonicity:
\[(A > B) \Rightarrow (p \geq q) \iff [p, A; 1-p, B] \geq [q, A; 1-q, B]\]

Theorem: Rational preferences imply behavior describable as maximization of expected utility
Theorem [Ramsey, 1931; von Neumann & Morgenstern, 1944]

Given any preferences satisfying these constraints, there exists a real-valued function $U$ such that:

$U(A) \geq U(B) \iff A \succeq B$

$U([p_1,S_1; \ldots ; p_n,S_n]) = p_1 U(S_1) + \ldots + p_n U(S_n)$

- i.e. values assigned by $U$ preserve preferences of both prizes and lotteries!
- Optimal policy invariant under positive affine transformation $U' = aU + b, a > 0$

Maximum expected utility (MEU) principle:

- Choose the action that maximizes expected utility
- Note: rationality does not require representing or manipulating utilities and probabilities
  - E.g., a lookup table for perfect tic-tac-toe
Human Utilities
Utilities map states to real numbers. Which numbers?

Standard approach to assessment (elicitation) of human utilities:

- Compare a prize $A$ to a standard lottery $L_p$ between
  - “best possible prize” $u_T$ with probability $p$
  - “worst possible catastrophe” $u_\perp$ with probability $1-p$
- Adjust lottery probability $p$ until indifference: $A \sim L_p$
- Resulting $p$ is a utility in $[0,1]$

Pay $50

```
<table>
<thead>
<tr>
<th></th>
<th>No change</th>
<th>Instant death</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_T$</td>
<td>0.9999999</td>
<td>0.0000001</td>
</tr>
<tr>
<td>$u_\perp$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Money

- Money **does not** behave as a utility function, but we can talk about the utility of having money (or being in debt)
- Given a lottery $L = [p, \$X; (1-p), \$Y]$
  - The **expected monetary value** $\text{EMV}(L) = pX + (1-p)Y$
  - The utility is $U(L) = pU(\$X) + (1-p)U(\$Y)$
  - Typically, $U(L) < U(\text{EMV}(L))$
  - In this sense, people are **risk-averse**
- E.g., how much would you pay for a lottery ticket $L=[0.5, \$10,000; 0.5, \$0]$?
  - The **certainty equivalent** of a lottery $\text{CE}(L)$ is the cash amount such that $\text{CE}(L) \sim L$
  - The **insurance premium** is $\text{EMV}(L) - \text{CE}(L)$
  - If people were risk-neutral, this would be zero!
Decision Networks
In its most general form, a decision network represents information about
- Its current state
- Its possible actions
- The state that will result from its actions
- The utility of that state

Decision network = Bayes net + Actions + Utilities

- **Action nodes** (rectangles, cannot have parents, will have value fixed by algorithm)
- **Utility nodes** (diamond, depends on action and chance nodes)
Decision Networks

Umbrella

Weather

Forecast
Example: Take an umbrella?

<table>
<thead>
<tr>
<th>A</th>
<th>W</th>
<th>U(A,W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>leave</td>
<td>sun</td>
<td>100</td>
</tr>
<tr>
<td>leave</td>
<td>rain</td>
<td>0</td>
</tr>
<tr>
<td>take</td>
<td>sun</td>
<td>20</td>
</tr>
<tr>
<td>take</td>
<td>rain</td>
<td>70</td>
</tr>
</tbody>
</table>

### Weather

<table>
<thead>
<tr>
<th>W</th>
<th>P(W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>sun</td>
<td>0.7</td>
</tr>
</tbody>
</table>

### Umbrella

### Forecast

| W  | P(F=bad|W) |
|----|--------|
| sun| 0.17   |
| rain| 0.77  |
Decision Networks

- Decision network = Bayes net + Actions + Utilities
  - Action nodes (rectangles, cannot have parents, will have value fixed by algorithm)
  - Utility nodes (diamond, depends on action and chance nodes)

- Decision algorithm:
  - Fix evidence $e$
  - For each possible action $a$
    - Fix action node to $a$
    - Compute posterior $P(W|e,a)$ for parents $W$ of $U$
    - Compute expected utility $\sum_w P(w|e,a) \cdot U(a,w)$
  - Return action with highest expected utility
Example: Take an umbrella?

- **Decision algorithm:**
  - Fix evidence \( e \)
  - For each possible action \( a \)
    - Fix action node to \( a \)
    - Compute posterior \( P(W|e,a) \) for parents \( W \) of \( U \)
    - Compute expected utility of action \( a: \sum_w P(w|e,a) \cdot U(a,w) \)
  - Return action with highest expected utility

Umbrella = leave

\[
EU(\text{leave}|F=\text{bad}) = \sum_w P(w|F=\text{bad}) \cdot U(\text{leave},w)
\]

\[
= 0.34 \times 100 + 0.66 \times 0 = 34
\]

Umbrella = take

\[
EU(\text{take}|F=\text{bad}) = \sum_w P(w|F=\text{bad}) \cdot U(\text{take},w)
\]

\[
= 0.34 \times 20 + 0.66 \times 70 = 53
\]

Optimal decision = take!
Decision network with utilities on outcome states

Here, $U$ is a true utility.

With an action node as parent, it is sometimes called a $Q$-value.
Value of Information
Value of information

- Suppose you haven’t yet seen the forecast
  - $\text{EU}(\text{leave} \mid \ ) = 0.7 \times 100 + 0.3 \times 0 = 70$
  - $\text{EU}(\text{take} \mid \ ) = 0.7 \times 20 + 0.3 \times 70 = 35$

- **What if you look at the forecast?**
- If Forecast=good
  - $\text{EU}(\text{leave} \mid \text{F}=\text{good}) = 0.89 \times 100 + 0.11 \times 0 = 89$
  - $\text{EU}(\text{take} \mid \text{F}=\text{good}) = 0.89 \times 20 + 0.11 \times 70 = 25$
- If Forecast=bad
  - $\text{EU}(\text{leave} \mid \text{F}=\text{bad}) = 0.34 \times 100 + 0.66 \times 0 = 34$
  - $\text{EU}(\text{take} \mid \text{F}=\text{bad}) = 0.34 \times 20 + 0.66 \times 70 = 53$
- $\text{P}(\text{Forecast}) = <0.65,0.35>$
- Expected utility given forecast
  - $= 0.65 \times 89 + 0.35 \times 53 = 76.4$
- **Value of information** $= 76.4 - 70 = 6.4$
Value of information contd.

- General idea: value of information = *expected improvement in decision quality* from observing value of a variable
  - E.g., oil company deciding on seismic exploration and test drilling
  - E.g., doctor deciding whether to order a blood test
  - E.g., person deciding on whether to look before crossing the road
- Key point: decision network contains everything needed to compute it!
- \[
  \text{VPI}(E_i \mid e) = \left[ \sum_{e_i} P(e_i \mid e) \max_a \text{EU}(a \mid e_i, e) \right] - \max_a \text{EU}(a \mid e)
\]
VPI Properties

VPI is non-negative! $VPI(E_i | e) \geq 0$

VPI is not (usually) additive: $VPI(E_i, E_j | e) \neq VPI(E_i | e) + VPI(E_j | e)$

VPI is order-independent: $VPI(E_i, E_j | e) = VPI(E_j, E_i | e)$