Filtering algorithm

- **Aim**: devise a **recursive filtering** algorithm of the form
  \[ P(X_{t+1} | e_{1:t+1}) = g(e_{t+1}, P(X_t | e_{1:t})) \]

- \( P(X_{t+1} | e_{1:t+1}) = P(X_{t+1} | e_{1:t}, e_{t+1}) \)

- \( = \alpha P(e_{t+1} | X_{t+1}, e_{1:t}) P(X_{t+1} | e_{1:t}) \)

- \( = \alpha P(e_{t+1} | X_{t+1}) P(X_{t+1} | e_{1:t}) \)

- \( = \alpha P(e_{t+1} | X_{t+1}) \sum_{x_t} P(x_t | e_{1:t}) P(X_{t+1} | x_t, e_{1:t}) \)

- \( = \alpha P(e_{t+1} | X_{t+1}) \sum_{x_t} P(x_t | e_{1:t}) P(X_{t+1} | x_t) \)

*Given by HMM*  *Pre-computed*  *Given by HMM*
Filtering algorithm

- **Aim:** devise a *recursive filtering* algorithm of the form
  - \( P(X_{t+1} | e_{1:t+1}) = g(e_{t+1}, P(X_t | e_{1:t}) ) \)

- \( P(X_{t+1} | e_{1:t+1}) = P(X_{t+1} | e_{1:t}, e_{t+1}) \)
  - \( = \alpha P(e_{t+1} | X_{t+1}, e_{1:t}) P(X_{t+1} | e_{1:t}) \)

**LHS:** \( P(X_{t+1}, e_{1:t}, e_{t+1})/P(e_{1:t}, e_{t+1}) \)
**RHS:** \( \alpha P(e_{t+1}, X_{t+1}, e_{1:t})/P(X_{t+1}, e_{1:t}) \) * \( P(X_{t+1}, e_{1:t})/ P(e_{1:t}) \)

**RHS:** \( \alpha P(e_{t+1}, X_{t+1}, e_{1:t}) / P(e_{1:t}) \)
- \( \alpha = P(e_{1:t}) / P(e_{1:t}, e_{t+1}) \) which is the same for all \( x_{t+1} \)
Filtering algorithm

- Aim: devise a *recursive filtering* algorithm of the form
  - \( P(X_{t+1} | e_{1:t+1}) = g(e_{t+1}, P(X_t | e_{1:t})) \)

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

Why does \( P(e_{t+1} | X_{t+1}, e_{1:t}) = P(e_{t+1} | X_{t+1}) \)?

Variables are independent of non-descendants given parents
If I know \( X_4 \), nothing else will help be better predict \( e_4 \)
Filtering algorithm

- **Aim:** devise a *recursive filtering* algorithm of the form
  \[ P(X_{t+1} | e_{1:t+1}) = g(e_{t+1}, P(X_t | e_{1:t})) \]

- \[ P(X_{t+1} | e_{1:t+1}) = P(X_{t+1} | e_{1:t}, e_{t+1}) \]
- \[ = \alpha P(e_{t+1} | X_{t+1}, e_{1:t}) \ P(X_{t+1} | e_{1:t}) \]
- \[ = \alpha P(e_{t+1} | X_{t+1}) \ P(X_{t+1} | e_{1:t}) \]
- \[ = \alpha P(e_{t+1} | X_{t+1}) \ \sum_{x_t} P(x_t | e_{1:t}) \ P(X_{t+1} | x_t, e_{1:t}) \]

\[ \sum_{x_t} P(X_{t+1} | x_t, e_{1:t}) \ P(x_t | e_{1:t}) = \sum_{x_t} P(X_{t+1}, x_t | e_{1:t}) = P(X_{t+1} | e_{1:t}) \]
Filtering algorithm

- **Aim:** devise a **recursive filtering** algorithm of the form
  \[ P(X_{t+1} | e_{1:t+1}) = g(e_{t+1}, P(X_t | e_{1:t})) \]

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

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

  \[ = \alpha P(e_{t+1} | X_{t+1}) \sum_{x_t} P(x_t | e_{1:t}) P(X_{t+1} | x_t, e_{1:t}) \]

  \[ = \alpha P(e_{t+1} | X_{t+1}) \sum_{x_t} P(x_t | e_{1:t}) P(X_{t+1} | x_t) \]

*Given by HMM*  *Pre-computed*  *Given by HMM*

*Variables are independent of non-descendants given parents*
“Forward” algorithm

\[ P(X_{t+1} | e_{1:t+1}) = \alpha \ P(e_{t+1} | X_{t+1}) \sum_{x_t} P(x_t | e_{1:t}) P(X_{t+1} | x_t) \]

- \( f_{1:t+1} = \text{FORWARD}(f_{1:t}, e_{t+1}) ; \) \( f_{1:t} \) is \( P(X_t | e_{1:t}) \) *for t=0, note \( e_{1:0} \) is empty
- Cost per time step: \( O(|X|^2) \) where \( |X| \) is the number of states
- Time and space costs are constant, independent of \( t \)
- \( O(|X|^2) \) is infeasible for models with many state variables
- We get to invent really cool approximate filtering algorithms
And the same thing in linear algebra

- Transition matrix $T$, observation matrix $O_t$
  - Observation matrix has state likelihoods for $E_t$ along diagonal
  - E.g., for $U_1 = \text{true}$, $O_1 = \begin{pmatrix} 0.2 & 0 \\ 0 & 0.9 \end{pmatrix}$

- Filtering algorithm becomes
  - $f_{1:t+1} = \alpha \ O_{t+1} \ T^T f_{1:t}$

| $X_{t-1}$ | $P(X_t|X_{t-1})$ |
|-----------|------------------|
| sun       | 0.9              |
| rain      | 0.1              |

| $W_t$  | $P(U_t|W_t)$ |
|--------|--------------|
| true   |              |
| sun    | 0.2          |
| rain   | 0.9          |
Example: Weather HMM

\[ f(\text{sun}) = 0.5 \]
\[ f(\text{rain}) = 0.5 \]

\[ f(\text{sun}) = 0.25 \]
\[ f(\text{rain}) = 0.75 \]

\[ f(\text{sun}) = 0.154 \]
\[ f(\text{rain}) = 0.846 \]

\[ P(W_{t-1} | W_t) \]

| \( W_{t-1} \) | \( P(W_t | W_{t-1}) \) |
|-----------------|-----------------|
| sun             | 0.9             | 0.1             |
| rain            | 0.3             | 0.7             |

\[ P(U_t | W_t) \]

| \( W_t \) | \( P(U_t | W_t) \) |
|------------|-----------------|
| true       | 0.2             | 0.8             |
| false      | 0.9             | 0.1             |
Pacman – Hunting Invisible Ghosts with Sonar

[Demo: Pacman – Sonar – No Beliefs(L14D1)]
Video of Demo Pacman – Sonar
Most Likely Explanation
Inference tasks

- **Filtering**: $P(X_t | e_{1:t})$
  - *belief state*—input to the decision process of a rational agent

- **Prediction**: $P(X_{t+k} | e_{1:t})$ for $k > 0$
  - evaluation of possible action sequences; like filtering without the evidence

- **Smoothing**: $P(X_k | e_{1:t})$ for $0 \leq k < t$
  - better estimate of past states, essential for learning

- **Most likely explanation**: $\arg \max_{x_{1:t}} P(x_{1:t} | e_{1:t})$
  - speech recognition, decoding with a noisy channel
Most likely explanation = most probable path

- **State trellis**: graph of states and transitions over time

\[ X_0 \quad X_1 \quad \ldots \quad X_T \]

- \( \arg \max_{x_{1:t}} P(x_{1:t} \mid e_{1:t}) \)
- \( = \arg \max_{x_{1:t}} \alpha P(x_{1:t}, e_{1:t}) \)
- \( = \arg \max_{x_{1:t}} P(x_{1:t}, e_{1:t}) \)
- \( = \arg \max_{x_{1:t}} P(x_0) \prod_t P(x_t \mid x_{t-1}) P(e_t \mid x_t) \)
- \( = \arg \max_{x_{1:t}} \log [ P(x_0) \prod_t P(x_t \mid x_{t-1}) P(e_t \mid x_t) ] \)
- \( = \arg \min_{x_{1:t}} -\log P(x_0) + \sum_t -\log P(x_t \mid x_{t-1}) + -\log P(e_t \mid x_t) \)

All given by HMM

Alternative form
Most likely explanation = most probable path

- **State trellis**: graph of states and transitions over time

  ![Trellis Diagram]

  - Each arc represents some transition $x_{t-1} \rightarrow x_t$
  - Each arc has weight $P(x_t \mid x_{t-1}) P(e_t \mid x_t)$ (arcs to initial states have weight $P(x_0)$)
  - The **product** of weights on a path is proportional to that state sequence’s probability
  - Forward algorithm computes sums of paths, **Viterbi algorithm** computes best paths
Forward / Viterbi algorithms

Forward Algorithm (sum)
For each state at time $t$, keep track of the total probability of all paths to it

$$f_{1:t+1} = \text{FORWARD}(f_{1:t}, e_{t+1})$$
$$= \alpha P(e_{t+1}|X_{t+1}) \sum_{x_t} P(X_{t+1}|x_t) f_{1:t}$$

Viterbi Algorithm (max)
For each state at time $t$, keep track of the maximum probability of any path to it

$$m_{1:t+1} = \text{VITERBI}(m_{1:t}, e_{t+1})$$
$$= P(e_{t+1}|X_{t+1}) \max_{x_t} P(X_{t+1}|x_t) m_{1:t}$$
Viterbi algorithm contd.

Time complexity? \( \mathcal{O}(|X|^2 T) \)

Space complexity? \( \mathcal{O}(|X| T) \)

Number of paths? \( \mathcal{O}(|X|^T) \)
Viterbi in negative log space

argmax of product of probabilities
= argmin of sum of negative log probabilities
= minimum-cost path

Viterbi is essentially breadth-first graph search
What about A*?
CS 188: Artificial Intelligence
Dynamic Bayes Nets and Particle Filters

Slides from Stuart Russell
University of California, Berkeley
Dynamic Bayes Nets
Dynamic Bayes Nets (DBNs)

- 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 $t$ or $t-1$
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

- Sparse dependencies => exponentially fewer parameters in DBN
  - E.g., 20 Boolean state variables, 3 parents each;
    - DBN has $20 \times 2^3 = 160$ parameters, HMM has $2^{20} \times 2^{20} = \sim 10^{12}$ 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 $P(X_T|e_{1:T})$
- Online: eliminate all variables from the previous time step; store factors for current time only
- Problem: largest factor contains all variables for current time (plus a few more)
Application: ICU monitoring

- **State**: variables describing physiological state of patient
- **Evidence**: values obtained from monitoring devices
- **Transition model**: physiological dynamics, sensor dynamics
- **Query variables**: pathophysiological conditions (a.k.a. bad things)
Toy DBN: heart rate monitoring
The enhanced heart-rate DBN's inferences on data from a healthy 40-year-old male patient.
ICU data: 22 variables, 1min ave

- Measurements of heart rate, blood pressure
- Measurements of blood O\textsubscript{2} (different places), intracranial pressure
- Measurements from ventilator (lung stiffness, CO\textsubscript{2} concentration, ...)
- Artifacts with other causes
- Coughs
<table>
<thead>
<tr>
<th>Drug</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>propofol (88)</td>
<td></td>
</tr>
<tr>
<td>acetaminophen (650)</td>
<td></td>
</tr>
<tr>
<td>fentanyl (43)</td>
<td></td>
</tr>
<tr>
<td>indomethacin (25)</td>
<td></td>
</tr>
<tr>
<td>cefepime (2)</td>
<td></td>
</tr>
<tr>
<td>ceftriaxone (1000)</td>
<td></td>
</tr>
<tr>
<td>metronidazole (Flagyl) (500)</td>
<td></td>
</tr>
<tr>
<td>vancomycin (1000)</td>
<td></td>
</tr>
<tr>
<td>Zosyn (piperacillin-tazobactam) (4)</td>
<td></td>
</tr>
<tr>
<td>phenylephrine (167)</td>
<td></td>
</tr>
<tr>
<td>furosemide (20)</td>
<td></td>
</tr>
<tr>
<td>dextrose 50% (50)</td>
<td></td>
</tr>
<tr>
<td>regular insulin (4; 14)</td>
<td></td>
</tr>
<tr>
<td>phenytoin (143)</td>
<td></td>
</tr>
<tr>
<td>ascorbic acid (Vitamin C) (500)</td>
<td></td>
</tr>
<tr>
<td>bisacodyl (10)</td>
<td></td>
</tr>
<tr>
<td>docusate sodium (250)</td>
<td></td>
</tr>
<tr>
<td>Gastrografin (400)</td>
<td></td>
</tr>
<tr>
<td>KCl (12)</td>
<td></td>
</tr>
<tr>
<td>KPO4 (18)</td>
<td></td>
</tr>
<tr>
<td>magnesium sulfate (2)</td>
<td></td>
</tr>
<tr>
<td>NaCl (3)</td>
<td></td>
</tr>
<tr>
<td>NaPO4 (2)</td>
<td></td>
</tr>
<tr>
<td>ProMod (3)</td>
<td></td>
</tr>
<tr>
<td>retinol (Vitamin A) (25000)</td>
<td></td>
</tr>
<tr>
<td>sodium phosphate (22)</td>
<td></td>
</tr>
<tr>
<td>vecuronium (10)</td>
<td></td>
</tr>
<tr>
<td>ZnSO4 (220)</td>
<td></td>
</tr>
<tr>
<td>farnotidine (20)</td>
<td></td>
</tr>
<tr>
<td>heparin (5000)</td>
<td></td>
</tr>
</tbody>
</table>
Blood pressure measurement
One-second vs one-minute data

"Bag" artifacts at 1-second resolution

Zeroing artifact at 1-second resolution

"Bag" artifacts as observed in one-minute average data

Zeroing artifact as observed in 1-minute average data
Detection of “bag” events

ROC curve for hypertension detection (SBP>160mmHg)
ALARM
Particle Filtering
We need a new algorithm!

- 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 \textit{exponentially} with $T$
We need a new idea!

- 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

- A particle is a possible world state
  - (i.e. a possible assignment of values for each variable at a single timestep)

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

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

- So, many $x$ may have $P(x) = 0$!
- More particles => more accuracy (cf. frequency histograms)
- 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]?”
- Estimates of low-dimensional marginals more accurate
  - (in log space; equally accurate in absolute terms)
Particle Filtering: Prediction step

- Particle $j$ in state $x_t^{(j)}$ samples a new state directly from the transition model:
  - $x_{t+1}^{(j)} \sim P(X_{t+1} \mid x_t^{(j)})$
- Here, most samples move clockwise, but some move in another direction or stay in place
After observing $e_{t+1}$:

- As in likelihood weighting, weight each sample based on the evidence
  - $w^{(j)} = P(e_{t+1} | x_{t+1}^{(j)})$
- Particles that fit the data better get higher weights, others get lower weights
- Normalize the weights across all particles
Particle Filtering: Resample

- 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/N)
Particles: track samples of states rather than an explicit distribution

Prediction

Update/Weight

Resample

Particles:
(3,3)
(2,3)
(3,3)
(3,2)
(3,3)
(1,2)
(3,3)
(2,3)
(2,3)

Particles:
(3,2)
(2,3)
(3,2)
(3,1)
(3,3)
(1,3)
(2,3)
(3,2)
(2,2)

Particles:
(3,2) w=.9
(2,3) w=.2
(3,2) w=.9
(3,1) w=.4
(3,3) w=.4
(3,2) w=.9
(2,3) w=.2
(3,2) w=.9
(2,2) w=.4

(New) Particles:
(3,2)
(2,2)
(3,2)
(2,3)
(3,3)
(3,2)
(1,3)
(2,3)
(3,2)

Consistency: see proof in AIMA Ch. 14 (requires DBN probabilities to be bounded away from 0)
Particle filtering on umbrella model

![Graph showing average absolute error over time steps for different models.](image)
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: RBPF)

DP-SLAM, Ron Parr
Particle Filter SLAM – Video

[Demo: PARTICLES-SLAM-fastslam.avi]