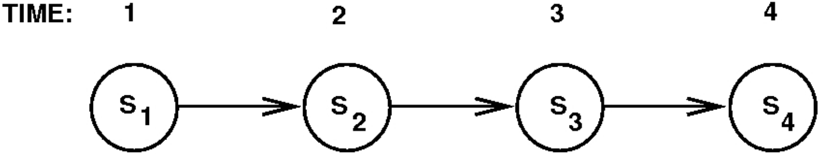
Lecture 8 - Reasoning under Uncertainty (Part II)

Jesse Hoey
School of Computer Science
University of Waterloo

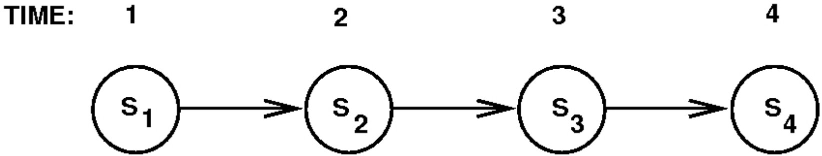
June 13, 2022

Readings: Poole & Mackworth (2nd ed.)Chapt. 8.5 - 8.9

- A node repeats over time
- explicit encoding of time
- chain has length = amount of time you want to model
- event-driven times or clock-driven times
- e.g. Markov chain



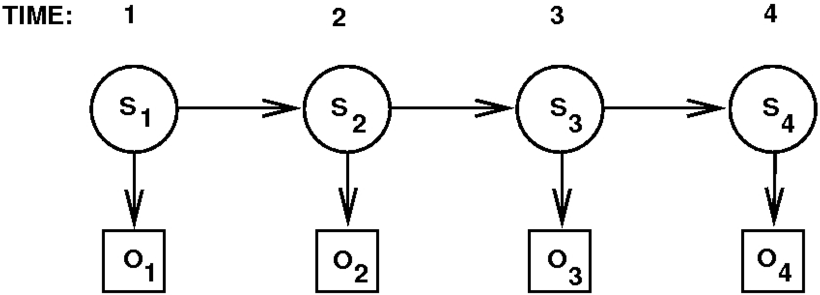
Markov assumption



$$P(S_{t+1}|S_1,\ldots,S_t) = P(S_{t+1}|S_t)$$

This distribution gives the dynamics of the Markov chain

Hidden Markov Models (HMMs)



Add: observations O_t (always observed, so the node is square) and
observation function $P(O_t|S_t)$
Given a sequency of observations $O_1, \ldots, O_t$, can estimate
filtering:

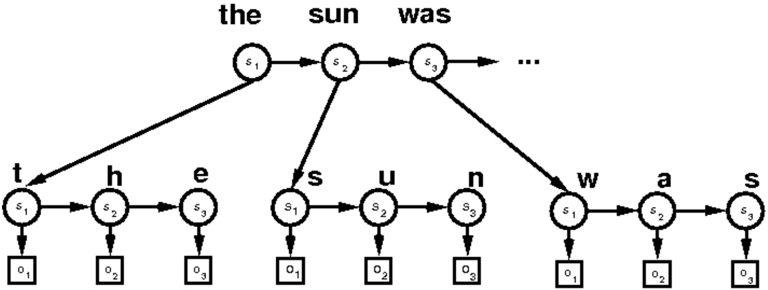
$$P(S_t|O_1, \ldots, O_t)$$

or smoothing, for $k < t$

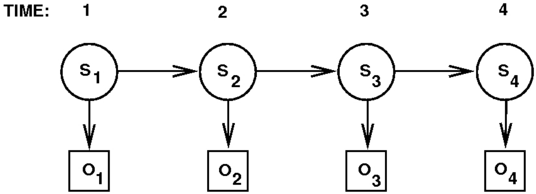
$$P(S_k|O_1, \ldots, O_t)$$

Speech Recognition

- Most well known application of HMMs
- observations : audio features
- states : phonemes
- dynamics : models e.g. co-articulation
- HMMs : words
- Can build hierarchical models (e.g. sentences)



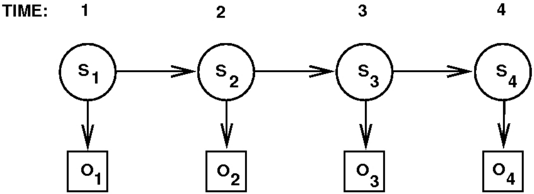
Belief Monitoring in HMMs



filering:

$$
\begin{aligned}
\alpha_i &= P(S_i|o_0 \ldots, o_i) \\
&\propto P(S_i, o_0, \ldots, o_i) \\
&= P(o_i|S_i) \sum_{S_{i-1}} P(S_i, S_{i-1}, o_0, \ldots, o_{i-1}) \\
&= P(o_i|S_i) \sum_{s_{i-1}} P(S_i|S_{i-1}) P(S_{i-1}, o_0, \ldots, o_{i-1}) \\
&\propto P(o_i|S_i) \sum_{s_{i-1}} P(S_i|S_{i-1}) \alpha_{i-1}
\end{aligned}
$$

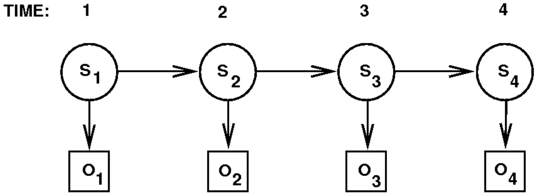
Belief Monitoring in HMMs



smoothing:

$$
\begin{aligned}
\beta_{i+1} &= P(o_{i+1} \ldots, o_T | S_i) \\
&= \sum_{S_{i+1}} P(S_{i+1}, o_{i+1}, \ldots, o_T | S_i) \\
&= \sum_{S_{i+1}} P(o_{i+1} | S_{i+1}, o_{i+2}, \ldots, o_T, S_i) P(S_{i+1}, o_{i+2}, \ldots, o_T | S_i) \\
&= \sum_{S_{i+1}} P(o_{i+1} | S_{i+1}) P(o_{i+2}, \ldots, o_T | S_{i+1}, S_i) P(S_{i+1} | S_i) \\
&= \sum_{S_{i+1}} P(o_{i+1} | S_{i+1}) P(S_{i+1} | S_i) \beta_{i+2}
\end{aligned}
$$

Belief Monitoring in HMMs

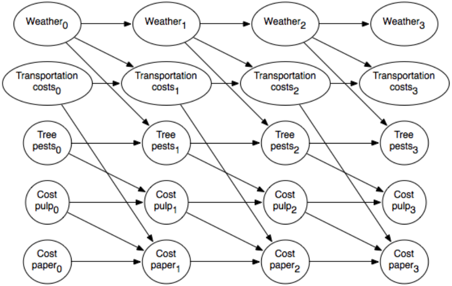


filtering and smoothing together:

$$\alpha_i \beta_{i+1} = P(o_{i+1} \ldots, o_T | S_i) P(S_i | o_0 \ldots, o_i) \propto P(S_i | O)$$

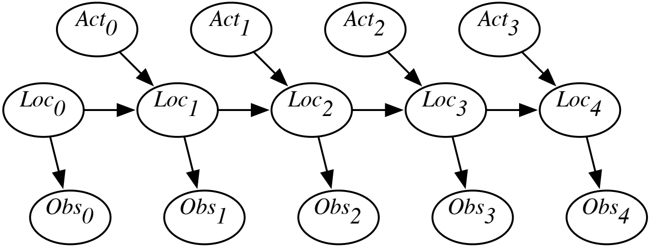
Dynamic Bayesian Networks (DBNs)

- in general, any Bayesian network can repeat over time: DBN
- Many examples can be solved with variable elimination,
- may become too complex with enough variables
- event-driven times or clock-driven times



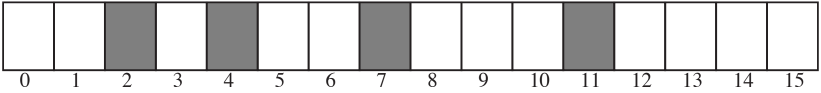
Example: localization

- Suppose a robot wants to determine its location based on its actions and its sensor readings: Localization
- This can be represented by the augmented HMM:



Example localization domain

- Circular corridor, with 16 locations:



- Doors at positions: 2, 4, 7, 11.
- Noisy Sensors
- Stochastic Dynamics
- Robot starts at an unknown location and must determine where it is, known as the kidnapped robot problem.
- see handout `robotloc.pdf`

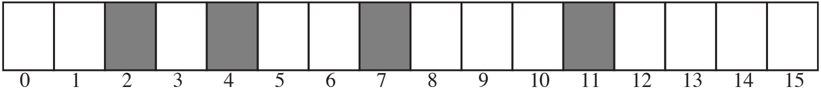
Example Sensor Model

- $P(\mathit{Observe\ Door} \mid \mathit{At\ Door}) = 0.8$
- $P(\mathit{Observe\ Door} \mid \mathit{Not\ At\ Door}) = 0.1$

Example Dynamics Model

- $P(Loc_{t+1} = l | Action_t = goRight \wedge Loc_t = l) = 0.1$
- $P(Loc_{t+1} = l + 1 | Action_t = goRight \wedge Loc_t = l) = 0.8$
- $P(Loc_{t+1} = l + 2 | Action_t = goRight \wedge Loc_t = l) = 0.074$
- $P(Loc_{t+1} = l' | Action_t = goRight \wedge Loc_t = l) = 0.002$ for any other location l'.
  - All location arithmetic is modulo 16.
  - The action *goLeft* works the same but to the left.

Example sequence



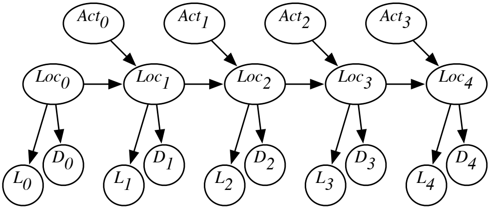
observe door, go right, observe no door, go right, observe door

where is the robot?

$$P(Loc_2 = 4 | O_0 = d, A_0 = r, O_1 = \neg d, A_1 = r, O_2 = d) = 0.42$$

Combining sensor information

- Example: we can combine information from a light sensor and the door sensor Sensor Fusion
- Key Point: Bayesian probability ensures that evidence is integrated proportionally to its precision.
- Sensors are precision weighted

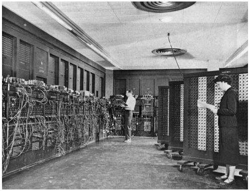


Loc_t robot location at time t
D_t door sensor value at time t
L_t light sensor value at time t

Probability Distribution and Monte Carlo

John von Neumann
1903 - 1957

Stanlislaw Ulam
1909-1984



ENIAC
1949

Monte Carlo
1949

Stochastic Simulation

- Idea: probabilities $\leftrightarrow$ samples
- Get probabilities from samples:

X	*count*
x_1	n_1
$\vdots$	$\vdots$
x_k	n_k
total	m

$\leftrightarrow$

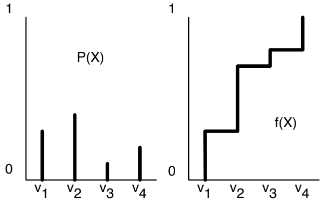
X	*probability*
x_1	n_1/m
$\vdots$	$\vdots$
x_k	n_k/m

- If we could sample from a variable's (posterior) probability, we could estimate its (posterior) probability.

Generating samples from a distribution

For a variable X with a discrete domain or a (one-dimensional) real domain:

- Totally order the values of the domain of X.
- Generate the cumulative probability distribution: $f(x) = P(X \leq x)$.
- Select a value y uniformly in the range $[0, 1]$.
- Select the x such that $f(x) = y$.



Hoeffding Bound

p is true probability, s is sample average, n is number of samples

- $P(|s - p| > \epsilon) \leq 2e^{-2n\epsilon^2}$
- if we want an error greater than ϵ in less than a fraction δ of the cases, solve for n:

$$2e^{-2n\epsilon^2} < \delta$$

$$n > \frac{-ln\frac{\delta}{2}}{2\epsilon^2}$$

- we have

ϵ error	cases with error $> \epsilon$	samples needed
0.1	1/20	184
0.01	1/20	18,445
0.1	1/100	265

Forward sampling in a belief network

- Sample the variables one at a time;
- sample parents of X before you sample X.
- Given values for the parents of X, sample from the probability of X given its parents.
- for samples $s_i, i = 1 \dots N$:

$$P(X = x_i) \propto \sum_{s_i} \delta(x_i) = N_{X=x_i}$$

where

$$\delta(x_i) = \begin{cases} 1 & \text{if } X = x_i \text{ in } s_i \\ 0 & \text{otherwise} \end{cases}$$

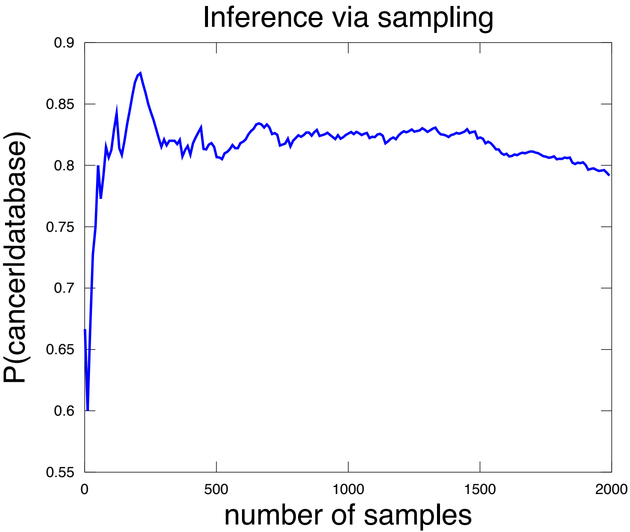
Sampling for a belief network: inference

Sample	Malfnction	Cancer	TestB	TestA	Report	Database
s_1	false	false	true	true	false	false
s_2	false	true	true	true	true	true
s_3	false	true	true	true	true	true
s_4	false	false	false	true	false	false
s_5	true	true	true	true	false	false
s_6	false	true	false	true	false	false
s_7	false	false	false	true	false	true
			...			
s_{1000}	false	false	false	true	false	false

To get $P(H = h_i|E = e_i)$ simply

- count the number of samples that have $H = h_i$ and $E = e_i$, $N(h_i, e_i)$
- divide by the number of samples that have $E = e_i$, $N(e_i)$
- $P(H = h_i|E = e_i) = \frac{P(H=h_i \wedge E=e_i)}{P(E=e_i)} = \frac{N(h_i,e_i)}{N(e_i)}$
- $P(C = True|Database = True)$ based on first 7 samples?

Forward Sampling

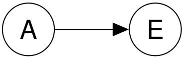


Rejection Sampling

- To estimate a posterior probability given evidence $Y_1 = v_1 \wedge \ldots \wedge Y_j = v_j$:
- If, for any i, a sample assigns Y_i to any value other than v_i reject that sample.
- The non-rejected samples are distributed according to the posterior probability.
- in the Hoeffding bound, n is the number of non-rejected samples

Example Network

$P(A = true)$
0.4



A	$P(E = true)$
true	0.1
false	0.3

If we draw N samples $s_{i=1\ldots N}$ by

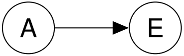
- sampling A: $a_{i=1\ldots N}$
- sampling from E given A: $e_{i=1\ldots N}$

then

- $\approx N_t = 0.4N$ of them will have $A = true$, and of these $\approx 10\%$ will have $E = true$
- $\approx N_f = 0.6N$ of them will have $A = false$, and of these $\approx 30\%$ will have $E = true$

Example Network

$P(A = true)$
0.4



A	$P(E = true)$
true	0.1
false	0.3

so we have

A	E	N_{AE}
true	false	$N_{tf} = 0.4 \times 0.9 \times N$
true	true	$N_{tt} = 0.4 \times 0.1 \times N$
false	false	$N_{ff} = 0.6 \times 0.7 \times N$
false	true	$N_{ft} = 0.6 \times 0.3 \times N$

We want to compute

$P(a|e) = P(A = true|E = true) \propto \sum_{s_i} \delta(a_i = true)\delta(e_i = true)$

$$P(a|e) = \frac{P(a \wedge e)}{P(e)} \approx \frac{N_{tt}}{N_{tt} + N_{ft}}$$

$$= \frac{0.1 \times 0.4 \times N}{0.1 \times 0.4 \times N + 0.3 \times 0.6 \times N} \qquad = 0.182$$

Importance weights

$P(A = true)$
0.4

$$A \longrightarrow E$$

A	$P(E = true)$
true	0.1
false	0.3

- we can do better since we can weight the samples
- weights = prob. that the evidence is observed
- N_t samples with $A = true$ have weight of $w_t = 0.1$
  this is $P(E = true|A = true)$
- N_f samples with $A = false$ have weight of $w_f = 0.3$
  this is $P(E = true|A = false)$
- can do better because we don't need to generate the 90% of samples (when $A = true$) that don't agree with the evidence - we simply assign all samples a weight of 0.1
- thus, we are mixing exact inference (the 0.1) with sampling.

Importance weights

$P(A = true)$
0.4

A	$P(E = true)$
true	0.1
false	0.3

- Compute sum of all weights of the samples with $A = true$

$$W_t = \sum_i w_t \delta(a_i = true) = N_t \times 0.1$$

- Compute sum of all weights of the samples with $A = false$

$$W_f = \sum_i w_f \delta(a_i = false) = N_f \times 0.3$$

- finally, compute

$$P(a|e) = \frac{W_t}{W_t + W_f} = \frac{0.1 \times 0.4 \times N}{0.1 \times 0.4 \times N + 0.3 \times 0.6 \times N}$$

Importance weights

$P(A = true)$
0.4

A	$P(E = true)$
true	0.1
false	0.3

- In fact, the As don't need to even be sampled from $P(A)$
- Can be sampled from some $q(A)$, say $q(A = true) = 0.5$
- and each sample will have a new weight $P(a)/q(a)$
- $q(A)$ is a proposal distribution.

Importance weights

$P(A = true)$
0.4

$$A \longrightarrow E$$

A	$P(E = true)$
true	0.1
false	0.3

- helps when it is hard to sample from $P(A)$, but we can evaluate $P^*(A) \propto P(A)$ given a sample
- rejection sampling uses $q = P$
- rejection sampling uses all variables including observed ones, and all weights on samples are set to 1.0

Importance weights

$P(A = true)$
0.4

A	$P(E = true)$
true	0.1
false	0.3

- $N'_t = q(a)N$ samples with $A = true$ have weight of $0.1 \times \frac{P^*(a)}{q(a)} = 0.1 \times \frac{\alpha 0.4}{0.5}$
- $N'_f = q(\bar{a})N$ samples with $A = false$ have weight of $0.3 \times \frac{P^*(\bar{a})}{q(\bar{a})} = 0.3 \times \frac{\alpha 0.6}{0.5}$

Importance weights

$P(A = true)$
0.4

$A \longrightarrow E$

A	$P(E = true)$
true	0.1
false	0.3

- total weight of all samples with $A = true$

$$W'_t = \sum_i w_i \delta(a_i = true) = N'_t \times 0.1 \times \frac{\alpha 0.4}{0.5}$$
$$= 0.5N \times 0.1 \times \frac{\alpha 0.4}{0.5} = 0.1 \times \alpha \times 0.4 \times N$$

- total weight of all samples with $A = false$

$$W'_f = \sum_i w_i \delta(a_i = true) = N'_f \times 0.3 \times \frac{\alpha 0.6}{0.5}$$
$$= 0.5N \times 0.3 \times \frac{\alpha 0.6}{0.5} = 0.3 \times \alpha \times 0.6 \times N$$

Importance weights

$P(A = true)$
0.4

A	$P(E = true)$
true	0.1
false	0.3

- finally, compute

$$P(a|e) = \frac{W'_t}{W'_t + W'_f} = \frac{0.1 \times \alpha \times 0.4 \times N}{0.1 \times \alpha \times 0.4 \times N + 0.3 \times \alpha \times 0.6 \times N}$$

Example Proposal Distributions

- Sometimes we may want to choose a proposal distribution that is different than the actual probability distribution
- We may want to skew the proposal - because we may have some additional knowlege about the data, for example
- or, we can generate proposals from the data itself using some procedural knowledge that is not directly encoded in the BN
- Can be important in multiple/many dimensions,

Stochastic sampling

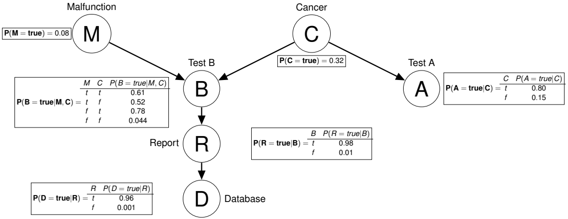
Recall variable elimination: To compute $P(Z, Y_1 = v_1, \ldots, Y_j = v_j)$, we sum out the other variables, $Z_1, \ldots, Z_k = \{X_1, \ldots, X_n\} - \{Z\} - \{Y_1, \ldots, Y_j\}$.

$$P(Z, Y_1 = v_1, \ldots, Y_j = v_j)$$
$$= \sum_{Z_k} \cdots \sum_{Z_1} \prod_{i=1}^{n} P(X_i | parents(X_i))_{Y_1 = v_1, \ldots, Y_j = v_j}$$

Now, we sample $Z_{l+1}, \ldots, Z_k$ and sum $Z_1, \ldots, Z_l$,

$$= \sum_{s_i = \{z_{l+1,i}, \ldots, z_{k,i}\}} \left[\sum_{Z_1 \ldots Z_l} \prod_{i=1}^{l} P(Z_i | parents(Z_i))_{Y_1 = v_1, \ldots, Y_j = v_j} \right] \frac{P(Z_{l+1,i}, \ldots, Z_{k,i})}{q(z_{l+1,i}, \ldots, z_{k,i})}$$

Importance Sampling example



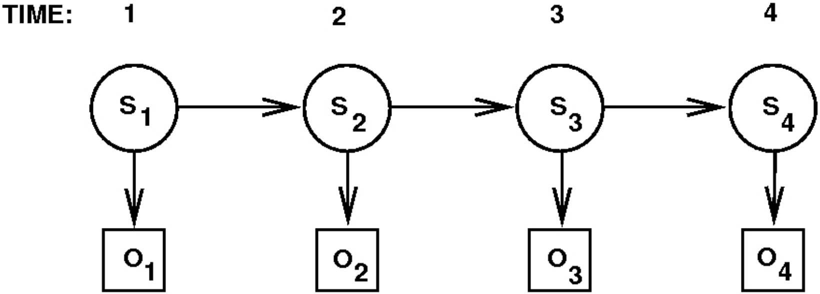
Compute $P(B|D = true, A = false)$ by sampling C and M.

- use $q(C = true) = P(C = true) = 0.32$
  and $q(M = true) = P(M = true) = 0.08$
- use $q(C = true) = 0.5$
  and $q(M = true) = P(M = true) = 0.08$
- use $q(C = true) = q(M = true) = 0.5$

$$P(B|D = true, A = false) \propto \sum_{s_i=\{c_i,m_i\}} P(B, D = true, A = false|c_i, m_i)$$

see `sampling-inference.pdf`

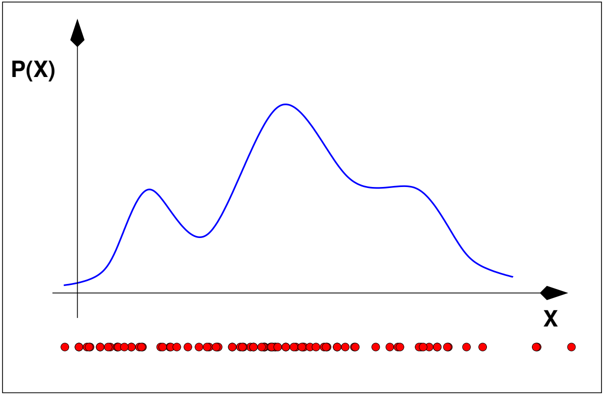
Stochastic Sampling for HMMs (and other DBNS)



Sequential Monte Carlo or Particle Filter

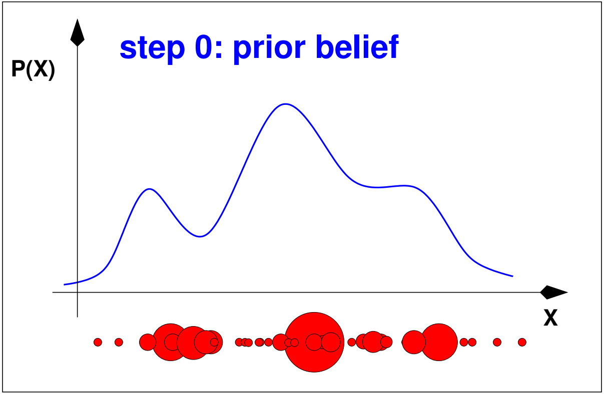
- sequential stochastic sampling
- keep track of $P(S_t)$ at the current time t
- represent $P(S_t)$ with a set of samples
- update as new observations o_{t+1} arrive
  1. predict $P(S_{t+1}) \propto P(S_{t+1}|S_t)$
  2. compute weights as $P(o_{t+1}|S_{t+1})$
  3. resample according to weights

Particle Filtering



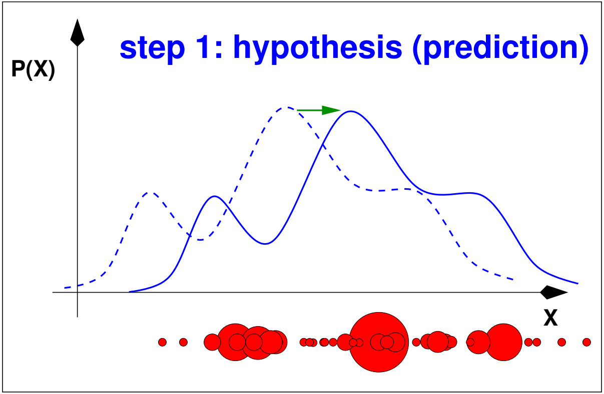
sample i: $\{x_i\}$

Particle Filtering



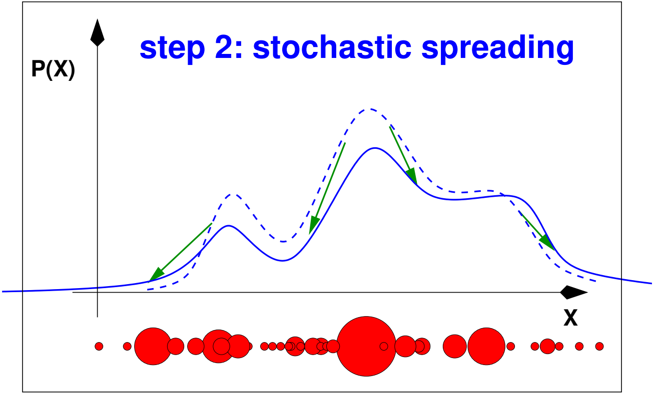
sample i: $\{x_i, w_i\}$

Particle Filtering



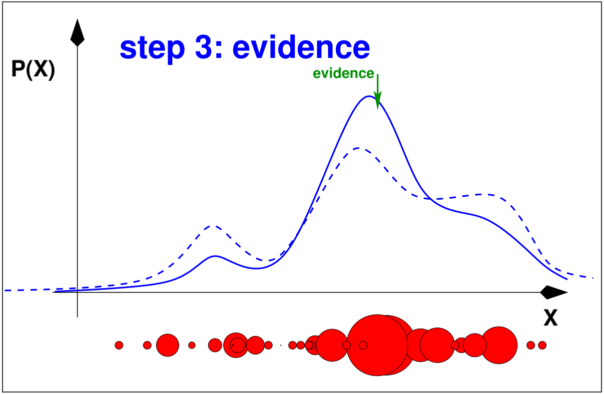
sample i: $\{x_i, w_i\}$

Particle Filtering



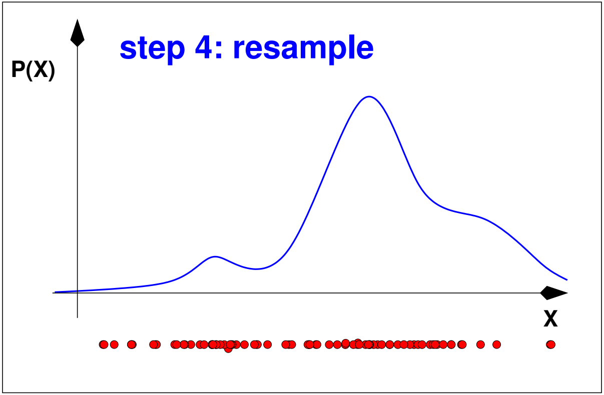
sample i: $\{x_i, w_i\}$

Particle Filtering



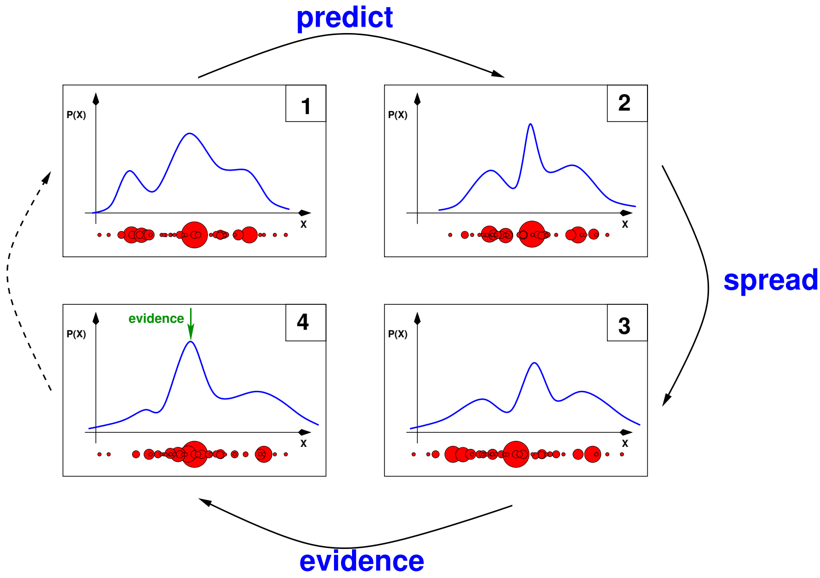
sample i: $\{x_i, w_i\}$

Particle Filtering

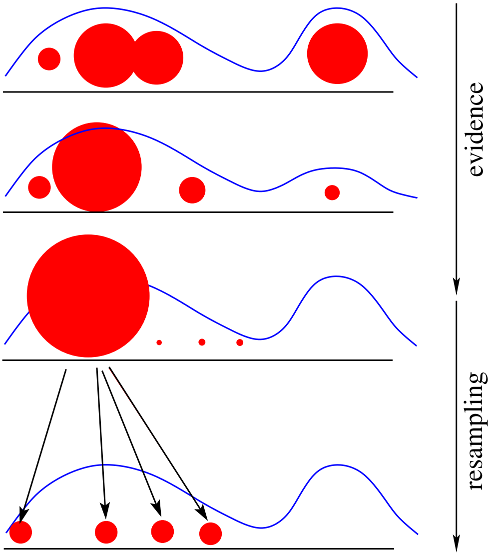


sample i: $\{x_i, w_i\}$

Bayesian Sequential Updates



Resampling



- avoids degeneracies in the samples
- all importance weights $\rightarrow$ 0 except one
- performance of the algorithm depends on the resampling method

Next:

- Supervised Learning under Uncertainty (Poole & Mackworth (2nd ed.)10.1,10.4)