Temporal Probabilistic Models

Tian-Li Yu

Taiwan Evolutionary Intelligence Laboratory (TEIL)
Department of Electrical Engineering
National Taiwan University
tianliyu@ntu.edu.tw

Outline

Bayesian Inference

- Bayesian learning calculates the probability of each hypothesis h_i, given data D, and makes prediction on that basis:

$$P(h_i|D) = P(D|h_i)\frac{P(h_i)}{P(D)}$$

In our discussion, $P(D)$ is fixed. Therefore,

$$P(h_i|D) \propto P(D|h_i)P(h_i)$$

- $P(h_i|D)$: posterior probability
- $P(D|h_i)$: likelihood
- $P(h_i)$: prior probability (of the hypothesis)

Bayesian Inference

- Assuming observations are i.i.d., $P(D|h_i) = \prod_{d \in D} P(d|h_i)$.
- To predict an unknown instance x:

$$P(x|D) = \sum_i P(x|D, h_i)P(h_i|D) = \sum_i P(x|h_i)P(h_i|D),$$

  where we assume hypothesis solely determine the probability of x.
- The above equation is computationally expensive.
- Common approximation: maximum a posteriori (MAP):

  $P(x|D) \simeq P(x|h_{MAP})$, where $h_{MAP} = \text{argmax}_{h_i} P(h_i|D)$

Maximum A Posteriori and Maximum Likelihood

- In both Bayesian learning and MAP learning, the prior $P(h_i)$ plays an important role.
- A simple assumption is a uniform prior where all $P(h_i)$ are equal.
  - MAP learning reduces to finding an h_i that maximizes $P(D|h_i)$.
  - This is called a maximum-likelihood (ML) hypothesis:

$$h_{ML} = \operatorname*{argmax}_{h_i} P(D|h_i)$$

Bayesian Inference Example

- Toss a 2-sides coin, where the probability of head is θ.
- The hypothesis space $H = \{\ h_\theta \mid 0 \leq \theta \leq 1\}$.
- In experiments (D), n out of N times are head.

$$P(D|h_\theta) = \prod_{j=1}^{N} P(d_j|h_\theta) = \theta^n \cdot (1-\theta)^{N-n}$$

- The log likelihood:

$$L(D|h_\theta) = \log P(D|h_\theta) = n \log \theta + (N-n) \log(1-\theta)$$

- Find the maximum:

$$\frac{d\ L(D|h_\theta)}{d\ \theta} = \frac{n}{\theta} - \frac{N-n}{1-\theta} = 0 \quad \Rightarrow \quad \theta = \frac{n}{N}$$

Maximum Likelihood in Regression = Mean Square Error

- Data $D = \{d_i = (x_i, y_i)\}$. We believe the measure error is of normal distribution: $y_i = c(x_i) + N(0, \sigma^2)$.
- For any hypothesis h, the likelihood

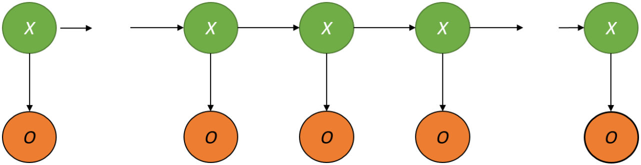
$$P(D|h) = \prod_i P(d_i|h) \propto \prod_i \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{1}{2\sigma^2}(y_i - h(x_i))^2}$$

- Assuming all hypotheses are equally probable:

$$h_{MAP} = h_{ML} = \operatorname*{argmax}_h P(D|h)$$

$$\begin{aligned}
\operatorname*{argmax}_h P(D|h) &= \operatorname*{argmax}_h \log P(D|h) \\
&= \operatorname*{argmax}_h \sum_i \left(\log \frac{1}{\sqrt{2\pi\sigma}} - \frac{1}{2\sigma^2}(y_i - h(x_i))^2 \right) \\
&= \operatorname*{argmin}_h \sum_i (y_i - h(x_i))^2 \qquad \text{(MSE)}
\end{aligned}$$

Time and Uncertainty



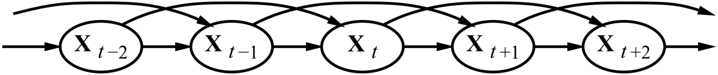
- The world changes; we need to track and predict it.
- X_t: Set of hidden state variables at time t.
  - x_t: hidden state at time t
- O_t: Set of observable state variables at time t.
  - o_t: observation at time t
- Notation: $X_{a:b} = X_a, X_{a+1}, \ldots, X_{b-1}, X_b$

Markov Processes

Assumption: X_t depends on bounded subset of $X_{0:t-1}$.

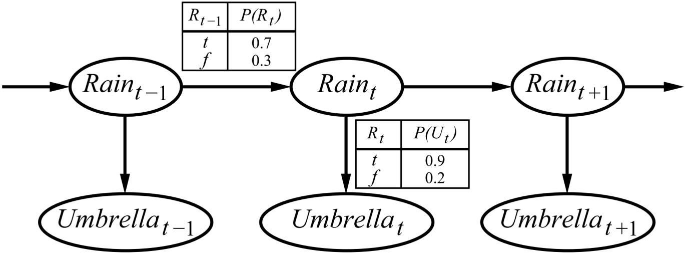
- First-order: $P(X_t|X_{0:t-1}) = P(X_t|X_{t-1})$

- Second-order: $P(X_t|X_{0:t-1}) = P(X_t|X_{t-2}, X_{t-1})$



Sensor Markov assumption: $P(O_t|X_{0:t}, O_{0,t-1}) = P(O_t|X_t)$
Stationary process: transition model $P(X_t|X_{t-1})$ and sensor model $P(O_t|X_t)$ are independent of t.

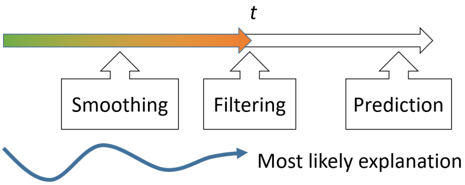
First-order Example



- First-order Markov assumption may not be accurate.
- Possible fixes:
  - Increasing order: $Rain_{t-2}$
  - Increasing state: $Humidity_t$

Temporal Inference Tasks

- Filtering: $P(X_t|o_{0:t})$ – The posterior distribution over the current state given all evidence to date.
- Prediction: $P(X_k|o_{0:t})$ for $k > t$ – Similar to filter, but with less evidence.
- Smoothing: $P(X_k|o_{0:t})$ for $k < t$ – Better estimation given more evidence.
- Most likely explanation: $\text{argmax}_{x_{0:t}} P(x_{0:t}|o_{0:t})$ – The sequence of states that most likely generate these observations.



- Learning: Learn the transition model from observations (EM or statistical learning).

Filtering

Goal: Devise a **recursive** state estimation algorithm.

$$
\begin{aligned}
P(X_{t+1} | o_{0:t+1}) &= P(X_{t+1} | o_{0:t}, o_{t+1}) \\
&= \alpha P(o_{t+1} | X_{t+1}, o_{0:t}) P(X_{t+1} | o_{0:t}) \\
&= \alpha P(o_{t+1} | X_{t+1}) P(X_{t+1} | o_{0:t})
\end{aligned}
$$

Summing out over x_t :

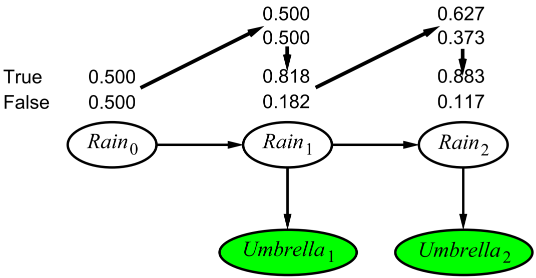
$$
\begin{aligned}
&= \alpha P(o_{t+1} | X_{t+1}) \sum_{x_t} P(X_{t+1} | x_t, o_{0:t}) P(x_t | o_{0:t}) \\
&= \alpha P(o_{t+1} | X_{t+1}) \sum_{x_t} P(X_{t+1} | x_t) P(x_t | o_{0:t})
\end{aligned}
$$

- $f_{0:t+1} = \textsc{Forward}(f_{0:t}, o_{t+1})$ where $f_{0:t} = P(X_t | o_{0:t})$.

Bayesian Derivations

$$
\begin{aligned}
P(A|\ B) &= \sum_{c\in C} P(A, C|\ B) = \sum_{c} P(A, c|\ B) \\
&= \sum_{c} \frac{P(A, B, c)}{P(B)} \\
&= \sum_{c} \frac{P(A, B, c)}{P(B, c)} \cdot \frac{P(B, c)}{P(B)} \\
&= \sum_{c} P(A|\ B, c) \cdot P(c|\ B)
\end{aligned}
$$

- Day 0: $P(R_0) = \langle 0.5, 0.5 \rangle$
- Day 1, umbrella appears:
  $P(R_1) = \sum_{r_0} P(R_1|r_0)P(r_0) = \langle 0.7, 0.3 \rangle \times 0.5 + \langle 0.3, 0.7 \rangle \times 0.5 = \langle 0.5, 0.5 \rangle$
  $P(R_1|u_1) = \alpha P(u_1|R_1)P(R_1) = \alpha \langle 0.9, 0.2 \rangle \cdot \langle 0.5, 0.5 \rangle = \alpha \langle 0.45, 0.1 \rangle \simeq \langle 0.818, 0.182 \rangle$
- Day 2: Umbrella appears:
  $P(R_2|u_1) = \sum_{r_1} P(R_2|r_1)P(r_1|u_1) = \langle 0.7, 0.3 \rangle \times 0.818 + \langle 0.3, 0.7 \rangle \times 0.182 \simeq \langle 0.627, 0.373 \rangle$
  $P(R_2|u_1, u_2) = \alpha P(u_2|R_2)P(R_2|u_1) = \alpha \langle 0.9, 0.2 \rangle \cdot \langle 0.627, 0.373 \rangle = \alpha \langle 0.565, 0.075 \rangle \simeq \langle 0.883, 0.117 \rangle$



Smoothing

$$
\begin{aligned}
P(X_k|\, o_{0:t}) &= P(X_k|\, o_{0:k}, o_{k+1:t}) \\
&= \alpha P(X_k|\, o_{0:k}) P(o_{k+1:t}|\, X_k, o_{0:k}) \\
&= \alpha P(X_k|\, o_{0:k}) P(o_{k+1:t}|\, X_k) \\
&= \alpha f_{0:k} \times b_{k+1:t}
\end{aligned}
$$

- Backward message is computed recursively:

$$
\begin{aligned}
P(o_{k+1:t}|X_k) &= \sum_{x_{k+1}} P(o_{k+1:t}|X_k, x_{k+1}) P(x_{k+1}|X_k) \\
&= \sum_{x_{k+1}} P(o_{k+1:t}|x_{k+1}) P(x_{k+1}|X_k) \\
&= \sum_{x_{k+1}} P(o_{k+1}|x_{k+1}) P(o_{k+2:t}|x_{k+1}) P(x_{k+1}|X_k)
\end{aligned}
$$

- $b_{k+1:t} = \textsc{Backward}(b_{k+2:t}, o_{k+1})$ where $b_{k+1:t} = P(o_{k+1:t}|\, X_k)$.

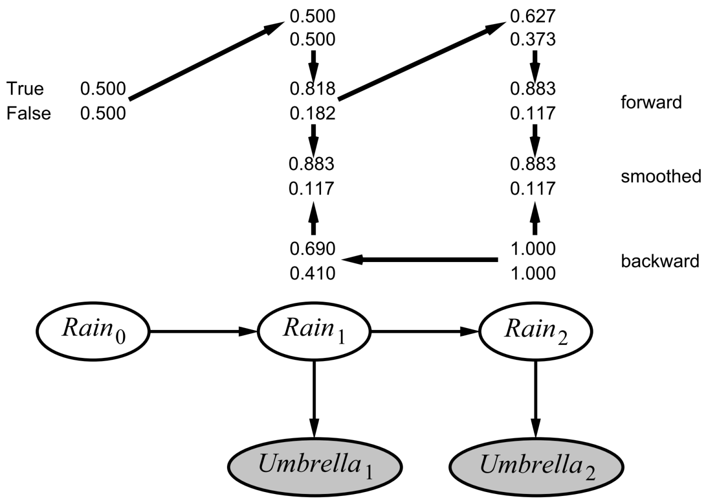
Smoothing Example (1)

- $P(R_1|\ u_1, u_2) = \alpha P(R_1|\ u_1)P(u_2|\ R_1)$

$$\begin{aligned} P(u_2|\ R_1) &= \sum_{r_2} P(u_2|r_2)P(|\ r_2)P(r_2|\ R_1) \\ &= (0.9 \times 1 \times \langle 0.7, 0.3\rangle) + (0.2 \times 1 \times \langle 0.3, 0.7\rangle) \\ &= \langle 0.69, 0.41\rangle \end{aligned}$$

- $P(R_1|\ u_1, u_2) = \alpha\langle 0.818, 0.182\rangle \times \langle 0.69, 0.41\rangle \simeq \langle 0.883, 0.117\rangle$

Smoothing Example (2)



Smoothing Complexity

- Both forward and backward recursion takes a **constant** time per step.
- Smoothing at a particular time k given $o_{0:t}$ takes $O(t)$ in time.
- Smoothing the whole sequence takes $O(t^2)$ in time.
- Down to $O(t)$ with dynamic programming (by reusing the result from previous forward).

FORWARD-BACKWARD

$f[0] \leftarrow$ prior
for $i = 1$ **to** t **do**
    $f[i] \leftarrow \textsc{Forward}(f[i-1], o_i)$
$b \leftarrow 1$
for $i = t$ **to** 1 **do**
    $sv[i] \leftarrow \textsc{Normalize}(f[i] \times b)$
    $b \leftarrow \textsc{Backward}(b, o_i)$

Most Likely Sequence

- Most likely sequence $\neq$ sequence of most likely states!!!
- Most likely path to x_{t+1} = most likely path to x_t + one more step.

$$\max_{x_{0:t}} P(x_{0:t}, X_{t+1} | o_{0:t+1})$$

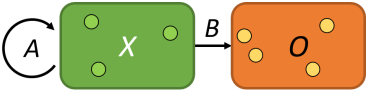
$$= \alpha P(o_{t+1} | X_{t+1}) \max_{x_t} \left(P(X_{t+1} | x_t) \max_{x_{0:t-1}} P(x_{0:t-1}, x_t | o_{0:t}) \right)$$

Define $m_{0:t} = \max_{x_{0:t-1}} P(x_{0:t-1}, x_t | o_{0:t})$,

$$m_{0:t+1} = P(o_{t+1} | X_{t+1}) \max_{x_t} \left(P(X_{t+1} | x_t) m_{0:t} \right)$$

This is the **Viterbi algorithm**.

Hidden Markov Models (HMM)



5-tuple: (X, O, Π, A, B)

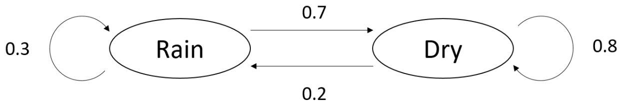
- Set of hidden states: $X = \{x_i\}$
- Set of observable states: $O = \{o_i\}$
- Initial probabilities: $\Pi = [\pi_i] = [P(x_i)]$
- Transition probabilities: $A = [a_{ij}] = [P(x_i | x_j)]$
- Observation probabilities: $B = [b_{ij}] = [P(o_i | x_j)]$

Markov Models

3-tuple: (S, Π, A)

- Set of states: $S = \{s_1, s_2, \ldots, s_N\}$
  - Sequence of states during the process: $s_{i_1}, s_{i_2}, \ldots, s_{i_k}, \ldots$
  - Markov chain property (first-order): probability of each subsequent state depends only on the previous state:
    $P(s_{i_k} | s_{i_1}, s_{i_2}, \ldots) = P(s_{i_k} | s_{i_{k-1}})$
- Initial probabilities: $\Pi = [\pi_i] = [P(s_i)]$
- Transition probabilities: $A = [a_{ij}] = [P(s_i | s_j)]$

Example of 2-State Markov Model



$$\Pi = \begin{bmatrix} P(Rain) \\ P(Dry) \end{bmatrix} = \begin{bmatrix} 0.4 \\ 0.6 \end{bmatrix} \qquad A = \begin{bmatrix} P(R|R) & P(R|D) \\ P(D|R) & P(D|D) \end{bmatrix} = \begin{bmatrix} 0.3 & 0.2 \\ 0.7 & 0.8 \end{bmatrix}$$

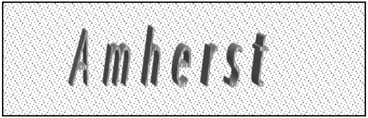
- $Q_0 = \Pi$; $Q_1 = A\,\Pi = \begin{bmatrix} 0.3 & 0.2 \\ 0.7 & 0.8 \end{bmatrix} \begin{bmatrix} 0.4 \\ 0.6 \end{bmatrix} = \begin{bmatrix} 0.24 \\ 0.76 \end{bmatrix}$; $Q_2 = A^2\,\Pi$
- $P(D, D, R, R) = P(R|\ R)P(R|\ D)P(D|\ D)P(D) = 0.3 \cdot 0.2 \cdot 0.8 \cdot 0.6$

Model Inference Tasks

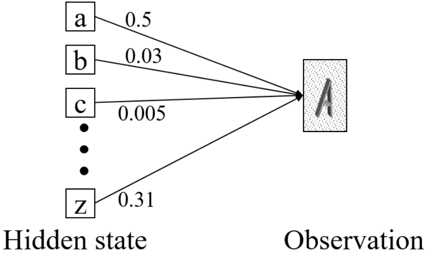
- Evaluation problem: $P(o_{0:t}|\ X, O, \Pi, A, B)$
  Calculate the probability that model M has generated sequence $o_{0:t}$.
- Decoding problem (most likely explanation): $\text{argmax}_{x_{0:t}} P(x_{0:t}|o_{0:t})$
  The sequence of states that most likely generate these observations.
- Learning problem: $\text{argmax}_{\Pi,A,B} P(o_{0:t}|\ X, O, \Pi, A, B)$
  Given some training observation sequences $o_{0:t}$, X and O, determine HMM parameters Π, A, B that best fit the training data.

Word Recognition Example (1)

- Typed word recognition, assume all characters are separated.

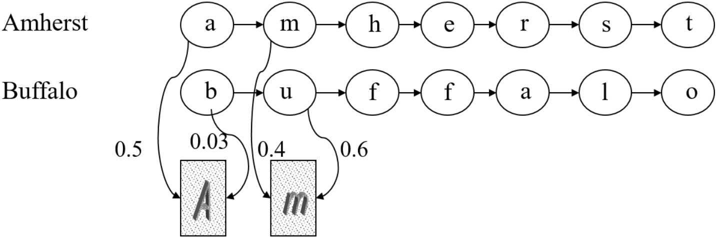


- Character recognizer outputs probability of the image being particular character, $P(\text{image} \mid \text{character})$. This can be handled by deep learning or HMM that we will talk later.



Word Recognition Example (2)

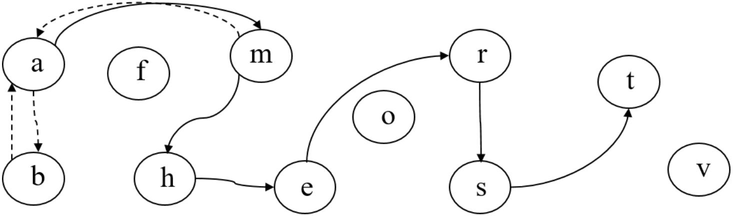
- If lexicon is given, we can construct separate HMM for each word.



- Recognition of words is equivalent to the problem of evaluating HMM models — the evaluation problem.

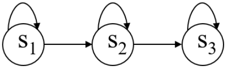
Word Recognition Example (3)

- Without lexicon info, we can construct one single HMM for all words.
- Observation probabilities are as before.
- Transition probabilities and initial probabilities are calculated from the language model (or dictionary),



- Recognition of words is equivalent to determining the best sequence of hidden states — the decoding problem.

Character Recognition with HMM Example (1)

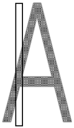
- Structure of hidden states: 

- Observation: number of islands in the vertical slice (assuming 1∼3).

•HMM for character 'A' :

Transition probabilities: $\{a_{ij}\} = \begin{pmatrix} .8 & .2 & 0 \\ 0 & .8 & .2 \\ 0 & 0 & 1 \end{pmatrix}$

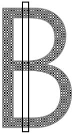
Observation probabilities: $\{b_{jk}\} = \begin{pmatrix} .9 & .1 & 0 \\ .1 & .8 & .1 \\ .9 & .1 & 0 \end{pmatrix}$



•HMM for character 'B' :

Transition probabilities: $\{a_{ij}\} = \begin{pmatrix} .8 & .2 & 0 \\ 0 & .8 & .2 \\ 0 & 0 & 1 \end{pmatrix}$

Observation probabilities: $\{b_{jk}\} = \begin{pmatrix} .9 & .1 & 0 \\ 0 & .2 & .8 \\ .6 & .4 & 0 \end{pmatrix}$



Character Recognition with HMM Example (2)

- 4 slices from left to right with the number of islands: $(1, 3, 2, 1)$.

For 'A'

Hidden state sequence	Transition prob.	Observation probab.	
$s_1 \to s_1 \to s_2 \to s_3$:	$0.8 \times 0.2 \times 0.2$	$\times 0.9 \times 0.0 \times 0.8 \times 0.9$	$= 0$
$s_1 \to s_2 \to s_2 \to s_3$:	$0.2 \times 0.8 \times 0.2$	$\times 0.9 \times 0.1 \times 0.8 \times 0.9$	$\simeq 0.00207$
$s_1 \to s_2 \to s_3 \to s_3$:	$0.2 \times 0.2 \times 1.0$	$\times 0.9 \times 0.1 \times 0.1 \times 0.9$	$\simeq 0.00032$
		Total:	$\simeq 0.00239$

For 'B'

Hidden state sequence	Transition prob.	Observation probab.	
$s_1 \to s_1 \to s_2 \to s_3$:	$0.8 \times 0.2 \times 0.2$	$\times 0.9 \times 0.0 \times 0.2 \times 0.6$	$= 0$
$s_1 \to s_2 \to s_2 \to s_3$:	$0.2 \times 0.8 \times 0.2$	$\times 0.9 \times 0.8 \times 0.2 \times 0.6$	$\simeq 0.00276$
$s_1 \to s_2 \to s_3 \to s_3$:	$0.2 \times 0.2 \times 1.0$	$\times 0.9 \times 0.8 \times 0.4 \times 0.6$	$\simeq 0.00691$
		Total:	$\simeq 0.00967$

Evaluation Problem

$$P(o_{0:t} | X, O, \Pi, A, B)$$

- $P(o_{0:t} | X, O, \Pi, A, B) = \sum_{x_{0:t} \in X_{0:t}} P(o_{0:t} | x_{0:t}, O, \Pi, A, B)$
  Exponential computational time!
- Use **Forward-Backward Algorithm** for again efficient computation.
  - Recall that $f_{0:t} = P(X_t | o_{0:t})$ can be calculated by forward recursion.
  - $P(o_{0:t}) = \sum_{x_t \in X_t} P(o_{0:t} | X_t) P(X_t = x_t)$

Kalman Filters

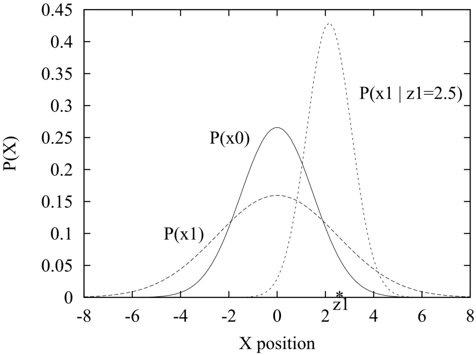
- Modeling systems by a set of continuous variables.
- Gaussian prior, linear Gaussian transition model and observation model.
- Prediction step: if $P(X_t | o_{0:t})$ is Gaussian, then the prediction $P(X_{t+1} | o_{0:t}) = \int_{x_t} P(X_{t+1} | x_t)P(x_t | o_{0:t})dx_t$ is Gaussian.
- Also, the updated distribution $P(X_{t+1} | o_{0:t+1}) = \alpha P(o_{t+1} | X_{t+1})P(X_{t+1} | o_{0:t})$ is Gaussian.

- Does not apply if the transition model is nonlinear. Consider the extended Kalman filter.

Kalman Filter Example

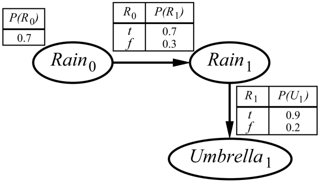
- Gaussian random walk on 1D with stdev σ_x; sensor stdev σ_z.

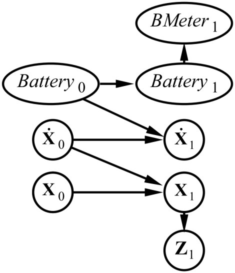
$$\mu_{t+1} = \frac{(\sigma_t^2 + \sigma_x^2)z_{t+1} + \sigma_z^2 \mu_t}{\sigma_t^2 + \sigma_x^2 + \sigma_z^2} \qquad \sigma_{t+1}^2 = \frac{(\sigma_t^2 + \sigma_x^2)\sigma_z^2}{\sigma_t^2 + \sigma_x^2 + \sigma_z^2}$$



Dynamic Bayesian Networks (DBNs)

- X_t, O_t contain arbitrarily many variables in a replicated Bayesian net.

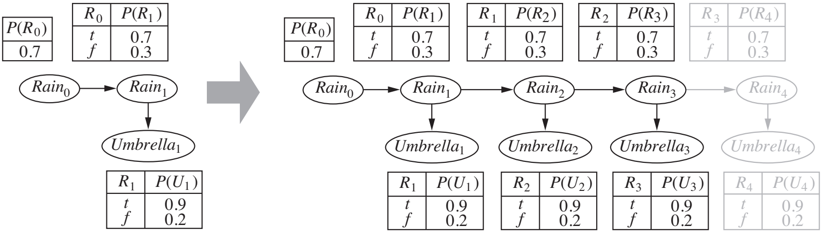




- Every HMM is a signle-variable DBN; every discrete DBN is an HMM. Key: Sparsity.

- Every Kalman filter is a DBN; but few DBNs are KFs.

Dynamic Bayesian Networks (DBNs)



- Naive inference: unroll to be a Bayesian net.

Bayesian Networks

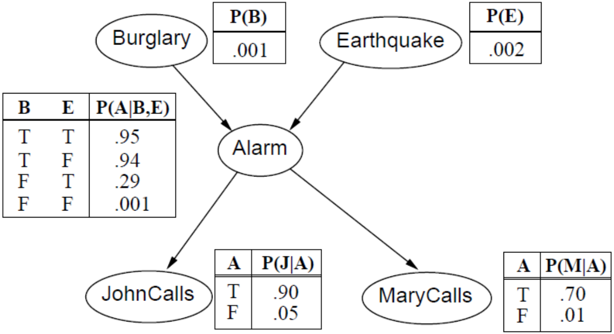
- A simple, graphical notation for conditional independence assertions and hence for compact specification of full joint distributions
- Syntax:
  - a set of nodes, one per variable
  - a directed acyclic graph (DAG) (link $\approx$ directly influences)
  - a conditional distribution for each node given its parents:
    $P(X_i|Parents(X_i))$
- In the simplest case, conditional distribution represented as a conditional probability table (CPT) giving the distribution over X_i for each combination of parent values

Alarm Example

- Variables: *Burglar*, *Earthquake*, *Alarm*, *JohnCalls*, *MaryCalls*
- Network topology reflects causal knowledge:
  - A burglar may set the alarm off.
  - An earthquake may set the alarm off.
  - The alarm may cause Mary to call.
  - The alarm may cause John to call.

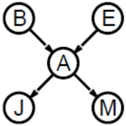
Alarm Example

- I'm at work, neighbor John calls to say my alarm is ringing, but neighbor Mary doesn't call. Sometimes it's set off by minor earthquakes. Is there a burglar?

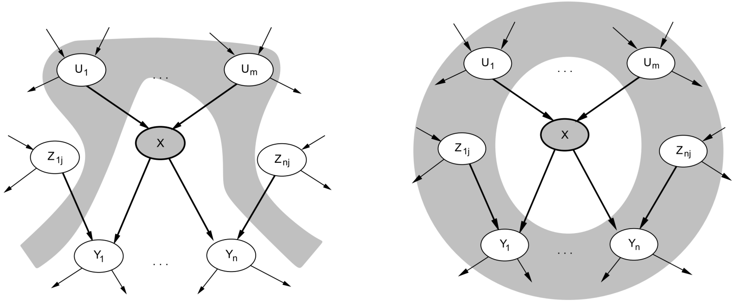


Compactness

- A CPT for Boolean X_i with k Boolean parents has 2^k rows for the combinations of parent values.
- Each row requires one number p for $X_i = true$ (the number for $X_i = false$ is just $1 - p$)
- If each variable has no more than k parents, the complete network requires $O(n \cdot 2^k)$ numbers.
- $O(n)$, vs. $O(2^n)$ for the full joint distribution.
- For the alarm net, $1 + 1 + 4 + 2 + 2 = 10$ numbers (vs. $2^5 - 1 = 31$)



Conditional Independence

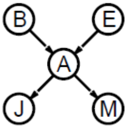


- Each node is conditionally independent of its **nondescendants** given its parents.
- Each node is conditionally independent of all others given its **Markov blanket**: parents + children + children's parents.

Joint Distribution

- Global semantics defines the full joint distribution as the product of the local conditional distributions:

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | parents(X_i))$$



- e.g., $P(j \wedge m \wedge a \wedge \neg b \wedge \neg e)$
  $= P(j|a)P(m|a)P(a|\neg b, \neg e)P(\neg b)P(\neg e)$
  $= 0.9 \cdot 0.7 \cdot 0.001 \cdot 0.999 \cdot 0.998$
  ≈ 0.00628

Inference by Stochastic Simulation

- Basic ideas:
  1. Draw samples from a sampling distribution S.
  2. Compute an approximation posterior probability $\hat{P}$.
  3. Show this converges to the true probability P.
- Outline
  - Direct sampling
  - Reject sampling
  - Likelihood weighting
  - Markov chain Monte Carlo (MCMC)

Direct and Reject Sampling

- Direct sampling:
  - Simply sample from root(s) to generate samples of the joint distribution.
  - Works for acyclic networks; use Gibbs sampling for cyclic ones.
- Reject sampling
  - Use direct sampling to generate samples, but only record those consistent with the evidence.
  - $\hat{P}(X|o) \simeq P(X,o)/P(o)$
  - Problem: expensive if $P(o)$ is small (drops exponentially with $|e|$).

$\textsc{Reject-Sampling}(bn, o)$

for $i = 1$ to N
    $x = \textsc{Direct-Sample}(bn)$
    if x is consistent with e
        $W[v] = W[v] + 1$, where v is the value of X in x.
return $\textsc{Normalize}(W)$

Likelihood Weighting

- Fix evidence, sample only non-evidence variables and weight each sample by the likelihood.

LIKELIHOOD-WEIGHTING(bn, o)

for $i = 1$ to N
  $x, w =$ WEIGHTED-SAMPLE(bn)
  $W[v] = W[v] + w$, where v is the value of X in x.
return NORMALIZE(W)

WEIGHTED-SAMPLE(bn, o)

$x =$ an event with n elements; $w = 1$
for $i = 1$ to n
  if X_i has a value x_i in o **then** $w = w \times P(X_i = x_i \mid parents(X_i))$
  else $x_i =$ a random sample from $P(X_i \mid parents(X_i))$
return x, w

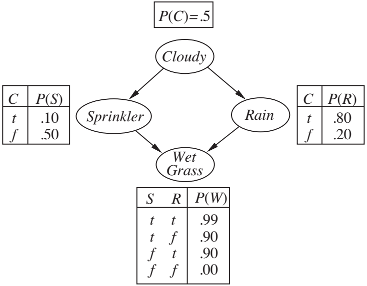
Likelihood Weighting Example

- Query: $P(R|C=T, W=T)$.

1. Topological order: C, S, R, W
2. $w = 1$.
3. $w = w \times P(C=T) = 0.5$.
4. $P(S|C=T) = \langle 0.1, 0.9 \rangle$.

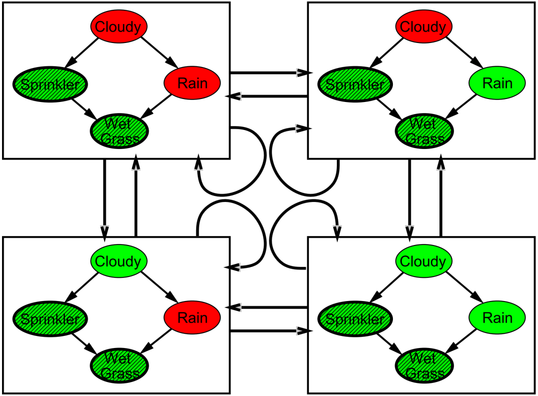
   Sample $S = F$.
5. $P(R|C=T) = \langle 0.8, 0.2 \rangle$.

   Sample $R = T$.
6. $w = w \times P(W=T|S=F, R=T) = 0.45$.
7. Return $\langle T, F, T, T \rangle$ with weight 0.45.



Markov Chain Monte Carlo (MCMC)

- Fix evidence variables.
- Sample non-evidence variables given Markov blanket.



Sampling Given Markov Blanket (Gibbs)

- Gibbs sampling: $P(x_i|\bar{x}_i, e) = P(x_i|mb(X_i))$

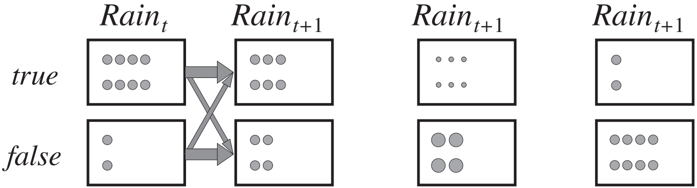
Given Markov blanket:

$$P(x_i|mb(X_i)) = P(x_i|parents(X_i)) \cdot \prod_{Z_j \in children(X_i)} P(z_j|Parents(Z_j)))$$

- Markov blanket of C is S and R.
- Markov blanket of R is C, S, and W.

Particle Filtering

- Ensure that the population of particles tracks the high-likelihood regions of the state-space.
- Replicate particles proportional to likelihood for the current evidence.



(a) Propagate (b) Weight (c) Resample

Particle Filtering

- Assume particle consistent at t: $\frac{N(x_t|o_{0:t})}{N} = P(x_t|o_{0:t})$
- Propagate:

$$N(x_{t+1}|o_{0:t}) = \sum_{x_t} P(x_{t+1}|x_t)N(x_t|o_{0:t})$$

- Weight by likelihood for o_{t+1}:

$$W(x_{t+1}|o_{0:t+1}) = P(o_{t+1}|x_{t+1})N(x_{t+1}|o_{0:t})$$

- Resample: $\frac{N(x_{t+1}|o_{0:t+1})}{N}$

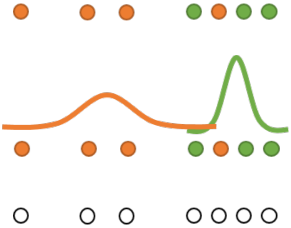
$$= \alpha W(x_{t+1}|o_{0:t+1}) = \alpha P(o_{t+1}|x_{t+1})N(x_{t+1}|o_{0:t})$$

$$= \alpha P(o_{t+1}|x_{t+1}) \sum_{x_t} P(x_{t+1}|x_t)N(x_t|o_{0:t})$$

$$= \alpha' P(o_{t+1}|x_{t+1}) \sum_{x_t} P(x_{t+1}|x_t)P(x_t|o_{0:t}) = P(x_{t+1}|o_{0:t+1})$$

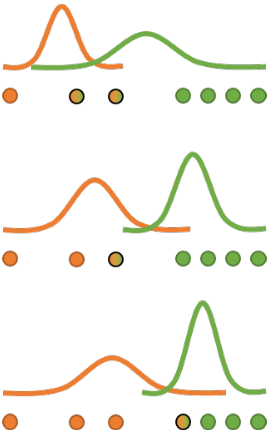
Expectation-Maximization Algorithm Example (1)

- Consider 1D two-Gaussian problem.
- If labels are given, no problem!
- What if labels are hidden?



EM Algorithm Example (2)

1. Randomly initialize hidden parameters (or according to prior).
2. **E-step:** Computer the expectation (the belongness of data).
3. **M-step:** Update hidden parameters by maximizing likelihood.
4. Repeat steps 2∼3 until termination.



Convex Optimization for EM

- For real values, $f(x)$ is **convex** iff $f''(x) \geq 0, \forall x \in \mathbb{R}$.
- For real vectors, $f(\vec{x})$ is **convex** iff $\det(H) \geq 0, \forall \vec{x} \in \mathbb{R}^d$, where $H = \left[\frac{\partial^2 f}{\partial x_i \partial x_j}\right]$.
- **Jensen's inequality:** If f is convex, $E[f(X)] \geq f(E[X])$.

EM derivations (1)

- Consider independent samples o_i with model parameters θ to learn,

  Likelihood: $\prod_i P(o_i|\,\theta)$ Log likelihood: $L(\theta) = \sum_i \ln P(o_i|\,\theta)$
- Introduce hidden states and sum out:

  $L(\theta) = \sum_i \ln \left(\sum_{x_j} P(o_i, x_j|\,\theta) \right)$
- Consider probability density function of the hidden state $P_X(x_j)$:

  $L(\theta) = \sum_i \ln \left(\sum_{x_j} P_X(x_j) \frac{P(o_i, x_j|\,\theta)}{P_X(x_j)} \right)$
- Let $Z_i = \frac{P(o_i, X|\,\theta)}{P_X(X)}$,

  $L(\theta) = \sum_i \ln E_{z \sim P_X}[Z_i]$

EM derivations (2)

- Since ln is concave,

$$\begin{aligned} L(\theta) &= \textstyle\sum_i \ln E_{z\sim P_X}[Z_i] \geq \sum_i E_{z\sim P_X}[\ln Z_i] \\ &= \textstyle\sum_i \sum_{x_j} P_x(x_j) \ln\left(\frac{P(o_i, x_j|\,\theta)}{P_x(x_j)}\right) \\ &\equiv LB(X;\theta) \end{aligned}$$

- Maximizing the lower bound
  - Equality holds in Jensen's inequality when the random variable is actually a constant.

$$\frac{P(o_i, x_j|\,\theta)}{P_x(x_j)} = c \quad \Rightarrow \quad P_x(x_j) = \frac{P(o_i, x_j|\,\theta)}{c}$$

$$P_x(x_j) = \frac{P(o_i, x_j|\,\theta)}{\sum_{x_j} P(o_i, x_j|\,\theta)} = \frac{P(o_i, x_j|\,\theta)}{P(o_i|\,\theta)} = P(o_i|\,x_j, \theta)$$

EM Algorithm

- **E-Step:**

$$\text{For given } \theta,\ P_x(x_j) = P(o_i|\, x_j, \theta)$$

- **M-Step:**

$$\text{Update parameters: } \theta \leftarrow \text{argmax}_\theta\, LB(X;\theta)$$

- Actual calculation depends on models. Some commonly used models include linear and Gaussian mixture model.