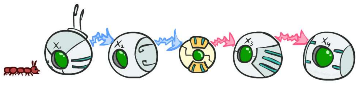
CSE 473: Artificial Intelligence

Hidden Markov Models

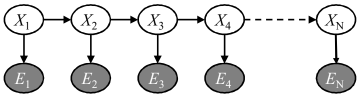


Steve Tanimoto --- University of Washington

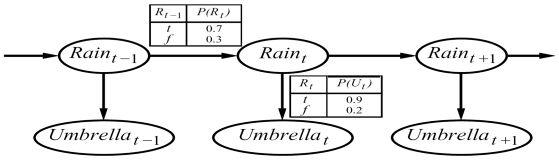
[Most slides were created by Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley. All CS188 materials are available at http://ai.berkeley.edu.]

Hidden Markov Models

- Markov chains not so useful for most agents
  - Eventually you don't know anything anymore
  - Need observations to update your beliefs
- Hidden Markov models (HMMs)
  - Underlying Markov chain over states S
  - You observe outputs (effects) at each time step
  - As a Bayes' net:

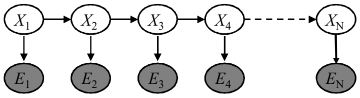


Example



- An HMM is defined by:
  - Initial distribution: $P(X_1)$
  - Transitions: $P(X_t|X_{t-1})$
  - Emissions: $P(E|X)$

Hidden Markov Models



- Defines a joint probability distribution:

$$P(X_1, \ldots, X_n, E_1, \ldots, E_n) =$$
$$P(X_{1:n}, E_{1:n}) =$$
$$P(X_1)P(E_1|X_1)\prod_{t=2}^{N} P(X_t|X_{t-1})P(E_t|X_t)$$

Ghostbusters HMM

- $P(X_1)$ = uniform
- $P(X'|X)$ = ghosts usually move clockwise, but sometimes move in a random direction or stay put
- $P(E|X)$ = same sensor model as before: red means close, green means far away.

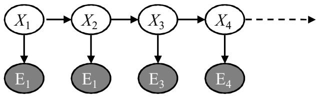
1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

$P(X_1)$

1/6	1/6	1/2
0	1/6	0
0	0	0

$P(X'|X=<1,2>)$

Etc…



P(E|X)

P(red | 3)	P(orange | 3)	P(yellow | 3)	P(green | 3)
0.05	0.15	0.5	0.3

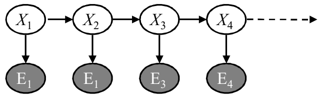
Etc… (must specify for other distances)

HMM Computations

- Given
  - parameters
  - evidence $E_{1:n} = e_{1:n}$
- Inference problems include:
  - Filtering, find $P(X_t|e_{1:t})$ for all t
  - Smoothing, find $P(X_t|e_{1:n})$ for all t
  - Most probable explanation, find $x^*_{1:n} = \text{argmax}_{x_{1:n}} P(x_{1:n}|e_{1:n})$

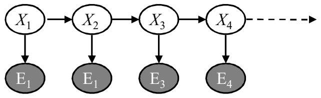
Real HMM Examples

- Speech recognition HMMs:
  - Observations are acoustic signals (continuous valued)
  - States are specific positions in specific words (so, tens of thousands)



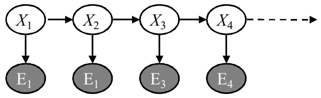
Real HMM Examples

- Machine translation HMMs:
  - Observations are words (tens of thousands)
  - States are translation options



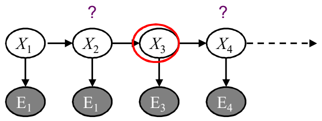
Real HMM Examples

- Robot tracking:
  - Observations are range readings (continuous)
  - States are positions on a map (continuous)



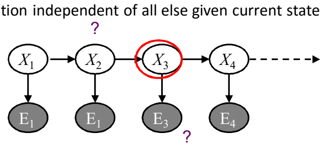
Conditional Independence

- HMMs have two important independence properties:
  - Markov hidden process, future depends on past via the present



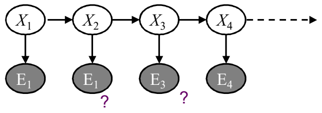
Conditional Independence

- HMMs have two important independence properties:
  - Markov hidden process, future depends on past via the present
  - Current observation independent of all else given current state



Conditional Independence

- HMMs have two important independence properties:
  - Markov hidden process, future depends on past via the present
  - Current observation independent of all else given current state



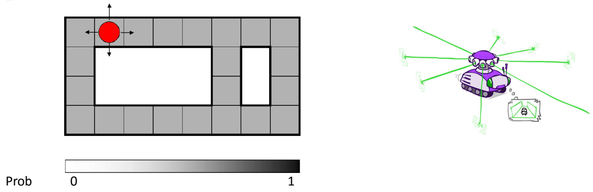
- Quiz: does this mean that observations are independent given no evidence?
  - [No, correlated by the hidden state]

Filtering / Monitoring

- Filtering, or monitoring, is the task of tracking the distribution B(X) (the belief state) over time
- We start with B(X) in an initial setting, usually uniform
- As time passes, or we get observations, we update B(X)
- The Kalman filter (one method – Real valued values)
  - invented in the 60's as a method of trajectory estimation for the Apollo program

Example: Robot Localization

Example from Michael Pfeiffer

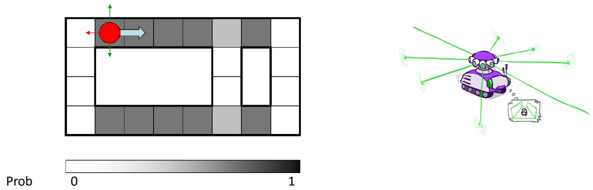


t=0

Sensor model: can read in which directions there is a wall, never more than 1 mistake

Motion model: may not execute action with small prob.

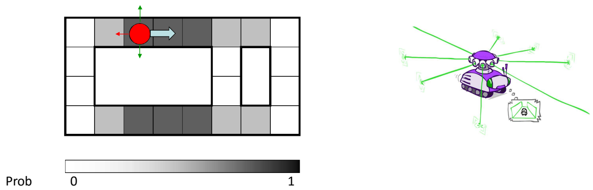
Example: Robot Localization



t=1

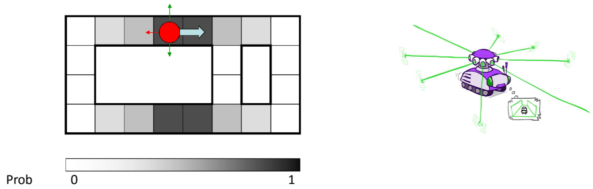
Lighter grey: was possible to get the reading, but less likely b/c required 1 mistake

Example: Robot Localization



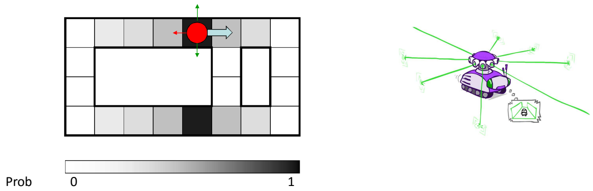
t=2

Example: Robot Localization



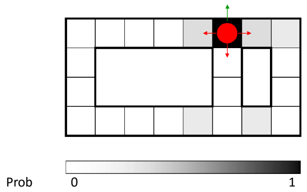
t=3

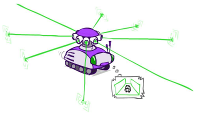
Example: Robot Localization



t=4

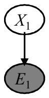
Example: Robot Localization





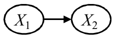
t=5

Inference Recap: Simple Cases



$$P(X_1|e_1)$$

$$P(x_1|e_1) = P(x_1,e_1)/P(e_1)$$
$$\propto_{X_1} P(x_1,e_1)$$
$$= P(x_1)P(e_1|x_1)$$

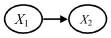


$$P(X_2)$$

$$P(x_2) = \sum_{x_1} P(x_1,x_2)$$
$$= \sum_{x_1} P(x_1)P(x_2|x_1)$$

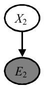
Online Belief Updates

- Every time step, we start with current P(X | 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:



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

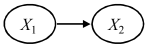
- The forward algorithm does both at once (and doesn't normalize)
- Problem: space is |X| and time is |X|^2 per time step

Passage of Time

- Assume we have current belief P(X | evidence to date)

$$B(X_t) = P(X_t|e_{1:t})$$

- Then, after one time step passes:



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

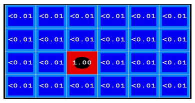
- Or, compactly:

$$B'(X') = \sum_{x} P(X'|x)B(x)$$

- 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

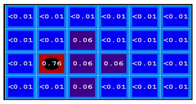
Example: Passage of Time

- As time passes, uncertainty "accumulates"



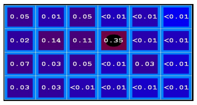
<0.01	<0.01	<0.01	<0.01	<0.01	<0.01
<0.01	<0.01	<0.01	<0.01	<0.01	<0.01
<0.01	<0.01	1.00	<0.01	<0.01	<0.01
<0.01	<0.01	<0.01	<0.01	<0.01	<0.01

T = 1



<0.01	<0.01	<0.01	<0.01	<0.01	<0.01
<0.01	<0.01	0.06	<0.01	<0.01	<0.01
<0.01	0.76	0.06	0.06	<0.01	<0.01
<0.01	<0.01	0.06	<0.01	<0.01	<0.01

T = 2



0.05	0.01	0.05	<0.01	<0.01	<0.01
0.02	0.14	0.11	0.35	<0.01	<0.01
0.07	0.03	0.05	<0.01	0.03	<0.01
0.03	0.03	<0.01	<0.01	<0.01	<0.01

T = 5

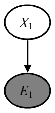
$$B'(X') = \sum_{x} P(X'|x)B(x)$$

Transition model: ghosts usually go clockwise

Observation

- Assume we have current belief P(X | previous evidence):

$$B'(X_{t+1}) = P(X_{t+1}|e_{1:t})$$



- Then:

$$P(X_{t+1}|e_{1:t+1}) \propto P(e_{t+1}|X_{t+1})P(X_{t+1}|e_{1:t})$$

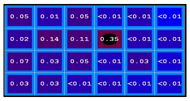
- Or:

$$B(X_{t+1}) \propto P(e|X)B'(X_{t+1})$$

- Basic idea: beliefs reweighted by likelihood of evidence
- Unlike passage of time, we have to renormalize

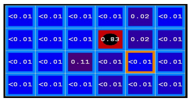
Example: Observation

- As we get observations, beliefs get reweighted, uncertainty "decreases"



0.05	0.01	0.05	<0.01	<0.01	<0.01
0.02	0.14	0.11	0.35	<0.01	<0.01
0.07	0.03	0.05	<0.01	0.03	<0.01
0.03	0.03	<0.01	<0.01	<0.01	<0.01

Before observation



<0.01	<0.01	<0.01	<0.01	0.02	<0.01
<0.01	<0.01	<0.01	0.83	0.02	<0.01
<0.01	<0.01	0.11	<0.01	<0.01	<0.01
<0.01	<0.01	<0.01	<0.01	<0.01	<0.01

After observation

$$B(X) \propto P(e|X)B'(X)$$

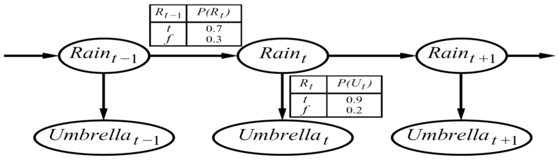
The Forward Algorithm

- We want to know: $B_t(X) = P(X_t|e_{1:t})$
- We can derive the following updates

$$\begin{aligned} P(x_t|e_{1:t}) &\propto_X P(x_t, e_{1:t}) \\ &= \sum_{x_{t-1}} P(x_{t-1}, x_t, e_{1:t}) \\ &= \sum_{x_{t-1}} P(x_{t-1}, e_{1:t-1}) P(x_t|x_{t-1}) P(e_t|x_t) \\ &= P(e_t|x_t) \sum_{x_{t-1}} P(x_t|x_{t-1}) P(x_{t-1}, e_{1:t-1}) \end{aligned}$$

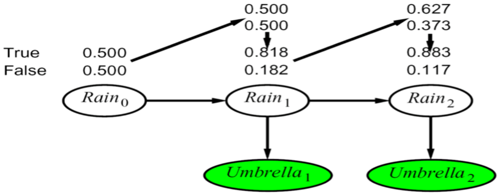
- To get $B_t(X)$, compute each entry and normalize

Example: Run the Filter

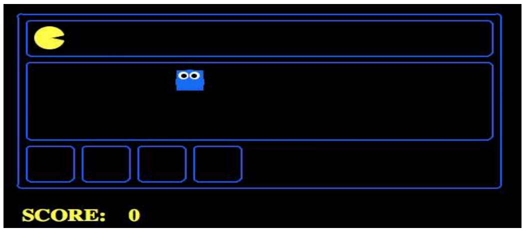


- An HMM is defined by:
  - Initial distribution: $P(X_1)$
  - Transitions: $P(X_t|X_{t-1})$
  - Emissions: $P(E|X)$

Example HMM



Example Pac-man



Summary: Filtering

- Filtering is the inference process of finding a distribution over X_T given e_1 through e_T : $P(X_T | e_{1:t})$
- We first compute $P(X_1 | e_1)$:
- For each t from 2 to T, we have $P(X_{t-1} | e_{1:t-1})$
- **Elapse time:** compute $P(X_t | e_{1:t-1})$

$$P(x_1|e_1) \propto P(x_1) \cdot P(e_1|x_1)$$

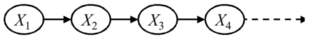
$$P(x_t|e_{1:t-1}) = \sum_{x_{t-1}} P(x_{t-1}|e_{1:t-1}) \cdot P(x_t|x_{t-1})$$

- **Observe:** compute $P(X_t | e_{1:t-1}, e_t) = P(X_t | e_{1:t})$

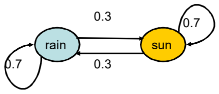
$$P(x_t|e_{1:t}) \propto P(x_t|e_{1:t-1}) \cdot P(e_t|x_t)$$

Recap: Reasoning Over Time

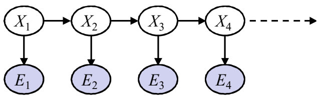
- Stationary Markov models



$P(X_1)$ $P(X|X_{-1})$



- Hidden Markov models



$P(E|X)$

X	E	P
rain	umbrella	0.9
rain	no umbrella	0.1
sun	umbrella	0.2
sun	no umbrella	0.8

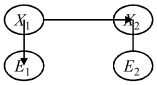
Recap: Filtering

Elapse time: compute P(X_t | $e_{1:t-1}$)

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

Observe: compute P(X_t | $e_{1:t}$)

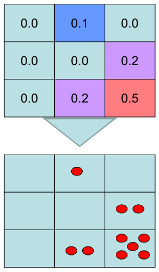
$$P(x_t|e_{1:t}) \propto P(x_t|e_{1:t-1}) \cdot P(e_t|x_t)$$



	Belief: <P(rain), P(sun)>	
$P(X_1)$	<0.5, 0.5>	*Prior on X_1*
$P(X_1 \mid E_1 = umbrella)$	<0.82, 0.18>	*Observe*
$P(X_2 \mid E_1 = umbrella)$	<0.63, 0.37>	*Elapse time*
$P(X_2 \mid E_1 = umb, E_2 = umb)$	<0.88, 0.12>	*Observe*

Particle Filtering

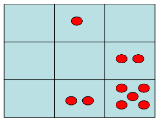
- Sometimes |X| is too big to use exact inference
  - |X| may be too big to even store B(X)
  - E.g. X is continuous
  - $|X|^2$ may be too big to do updates
- Solution: approximate inference
  - Track samples of X, not all values
  - Samples are called particles
  - Time per step is linear in the number of samples
  - But: number needed may be large
  - In memory: list of particles, not states
- This is how robot localization works in practice



0.0	0.1	0.0
0.0	0.0	0.2
0.0	0.2	0.5

Representation: Particles

- Our representation of P(X) is now a list of N particles (samples)
  - Generally, N << |X|
  - Storing map from X to counts would defeat the point
- P(x) approximated by number of particles with value x
  - So, many x will have P(x) = 0!
  - More particles, more accuracy
- For now, all particles have a weight of 1



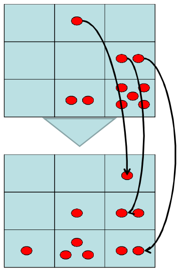
Particles:
(3,3)
(2,3)
(3,3)
(3,2)
(3,3)
(3,2)
(2,1)
(3,3)
(3,3)
(2,1)

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 probs
  - Here, most samples move clockwise, but some move in another direction or stay in place
- This captures the passage of time
  - If we have enough samples, close to the exact values before and after (consistent)



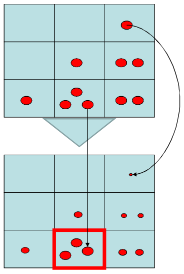
Particle Filtering: Observe

- Slightly trickier:
  - Don't do rejection sampling (why not?)
  - We don't sample the observation, we fix it
  - This is similar to likelihood weighting, so we downweight our samples based on the evidence

$$w(x) = P(e|x)$$

$$B(X) \propto P(e|X)B'(X)$$

  - Note that, as before, the probabilities don't sum to one, since most have been downweighted (in fact they sum to an approximation of P(e))

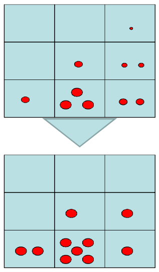


Particle Filtering: Resample

- Rather than tracking weighted samples, we resample
- N times, we choose from our weighted sample distribution (i.e. draw with replacement)
- This is equivalent to renormalizing the distribution
- Now the update is complete for this time step, continue with the next one

Old Particles:
(3,3) w=0.1
(2,1) w=0.9
(2,1) w=0.9
(3,1) w=0.4
(3,2) w=0.3
(2,2) w=0.4
(1,1) w=0.4
(3,1) w=0.4
(2,1) w=0.9
(3,2) w=0.3

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



Recap: Particle Filtering

At each time step t, we have a set of N particles / samples

- Initialization: Sample from prior, reweight and resample
- Three step procedure, to move to time t+1:

1. Sample transitions: for each each particle x, sample next state

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

2. Reweight: for each particle, compute its weight given the actual observation e

- Resample$w(x) = P(e|x)$hts, and sample N new particles from the resulting distribution over states

Particle Filtering Summary

- Represent current belief P(X | evidence to date) as set of *n* samples (actual assignments X=x)
- For each new observation e:

1. Sample transition, once for each current particle x

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

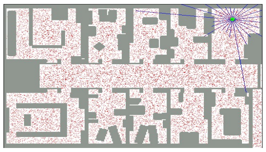
2. For each new sample x', compute importance weights for the new evidence e:

$$w(x') = P(e|x')$$

3. Finally, normalize the importance weights and resample N new particles

Robot Localization

- 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 (works basically like a very fine grid) and so we cannot store B(X)
  - Particle filtering is a main technique

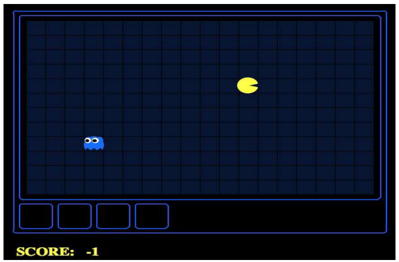


Robot Localization

QuickTime™ and a
GIF decompressor
are needed to see this picture.

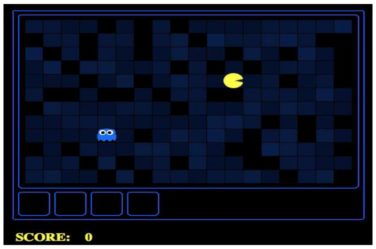
Which Algorithm?

Exact filter, uniform initial beliefs



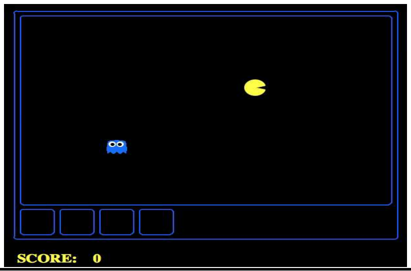
Which Algorithm?

Particle filter, uniform initial beliefs, 300 particles



Which Algorithm?

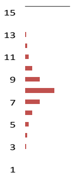
Particle filter, uniform initial beliefs, 25 particles



P4: Ghostbusters

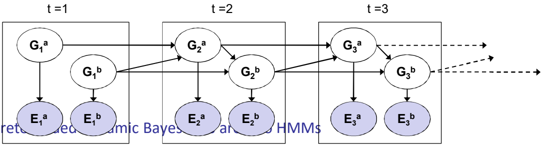
- **Plot:** Pacman's grandfather, Grandpac, learned to hunt ghosts for sport.
- He was blinded by his power, but could hear the ghosts' banging and clanging.
- **Transition Model:** All ghosts move randomly, but are sometimes biased
- **Emission Model:** Pacman knows a "noisy" distance to each ghost

Noisy distance prob
True distance = 8



Dynamic Bayes Nets (DBNs)

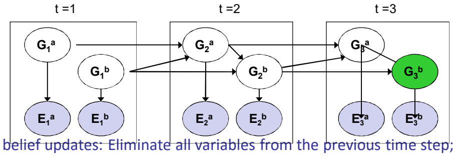
- We want to track multiple variables over time, using multiple sources of evidence
- Idea: Repeat a fixed Bayes net structure at each time
- Variables from time *t* can condition on those from *t-1*



- Discrete valued dynamic Bayes nets are also HMMs

Exact Inference in DBNs

- Variable elimination applies to dynamic Bayes nets
- Procedure: "unroll" the network for T time steps, then eliminate variables until $P(X_T|e_{1:T})$ is computed



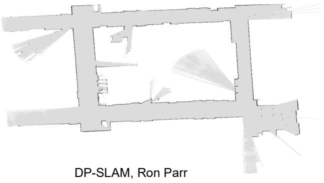
- Online belief updates: Eliminate all variables from the previous time step; store factors for current time only

DBN Particle Filters

- A particle is a complete sample for a time step
- **Initialize**: Generate prior samples for the t=1 Bayes net
  - Example particle: $\mathbf{G_1^a}$ = (3,3) $\mathbf{G_1^b}$ = (5,3)
- **Elapse time**: Sample a successor for each particle
  - Example successor: $\mathbf{G_2^a}$ = (2,3) $\mathbf{G_2^b}$ = (6,3)
- **Observe**: Weight each entire sample by the likelihood of the evidence conditioned on the sample
  - Likelihood: $P(\mathbf{E_1^a} \mid \mathbf{G_1^a}) * P(\mathbf{E_1^b} \mid \mathbf{G_1^b})$
- **Resample:** Select prior samples (tuples of values) in proportion to their likelihood

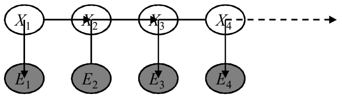
SLAM

- SLAM = Simultaneous Localization And Mapping
  - We do not know the map or our location
  - Our belief state is over maps and positions!
  - Main techniques: Kalman filtering (Gaussian HMMs) and particle methods
- [DEMOS]



DP-SLAM, Ron Parr

Best Explanation Queries

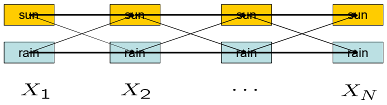


- Query: most likely seq:

$$\arg\max_{x_{1:t}} P(x_{1:t}|e_{1:t})$$

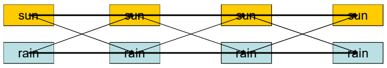
State Path Trellis

- State trellis: graph of states and transitions over time



- Each arc represents some transition $x_{t-1} \rightarrow x_t$
- Each arc has weight $P(x_t|x_{t-1})P(e_t|x_t)$
- Each path is a sequence of states
- The product of weights on a path is the seq's probability
- Can think of the Forward (and now Viterbi) algorithms as computing sums of all paths (best paths) in this graph

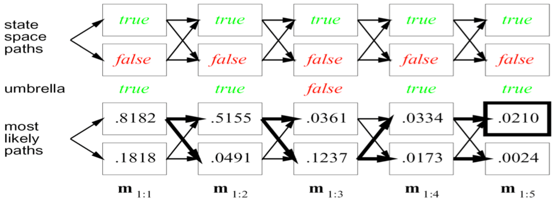
Viterbi Algorithm



$$x^*_{1:T} = \arg\max_{x_{1:T}} P(x_{1:T}|e_{1:T}) = \arg\max_{x_{1:T}} P(x_{1:T}, e_{1:T})$$

$$\begin{aligned} m_t[x_t] &= \max_{x_{1:t-1}} P(x_{1:t-1}, x_t, e_{1:t}) \\ &= \max_{x_{1:t-1}} P(x_{1:t-1}, e_{1:t-1})P(x_t|x_{t-1})P(e_t|x_t) \\ &= P(e_t|x_t) \max_{x_{t-1}} P(x_t|x_{t-1}) \max_{x_{1:t-2}} P(x_{1:t-1}, e_{1:t-1}) \\ &= P(e_t|x_t) \max_{x_{t-1}} P(x_t|x_{t-1}) m_{t-1}[x_{t-1}] \end{aligned}$$

Example


$Rain_1$
$Rain_2$
$Rain_3$
$Rain_4$
$Rain_5$
state space paths
true
true
true
true
true
false
false
false
false
false
umbrella
true
true
false
true
true
most likely paths
.8182
.5155
.0361
.0334
.0210
.1818
.0491
.1237
.0173
.0024
$\mathbf{m}_{1:1}$
$\mathbf{m}_{1:2}$
$\mathbf{m}_{1:3}$
$\mathbf{m}_{1:4}$
$\mathbf{m}_{1:5}$