

Decision-theoretic Planning

Reinforcement Learning and Planning under Uncertainty

ANU COMP6460/4640, Sem 2, 2008

Scott Sanner

NICTA / RSISE

First.Last@nicta.com.au

Directed Graphical Models Crash Course

(a.k.a., Bayesian networks)

Basic Probability

- Joint and conditional distributions:

$$P(a, b) = P(a|b)P(b) = P(b|a)P(a)$$

- Marginalization:

$$P(a) = \sum_{b \in B} P(a, b)$$

Don't memorize!
Derive from first
principles!

- Conditional probability & Bayes rule:

$$P(a|b) = \frac{P(a, b)}{P(b)} = \frac{P(b|a)P(a)}{\sum_{a \in A} P(b|a)P(a)}$$

Discrete Probability Operations I

- Binary Multiplication

$$P(a) \cdot P(b|a) = P(a, b)$$

<i>a</i>	<i>P</i>
0	.7
1	.3

<i>a</i>	<i>b</i>	<i>P</i>
0	0	.1
0	1	.9
1	0	.2
1	1	.8

<i>a</i>	<i>b</i>	<i>P</i>
0	0	.07
0	1	.63
1	0	.06
1	1	.24

- Same principle holds for all binary ops
 - sum, max, etc...

Discrete Probability Operations II

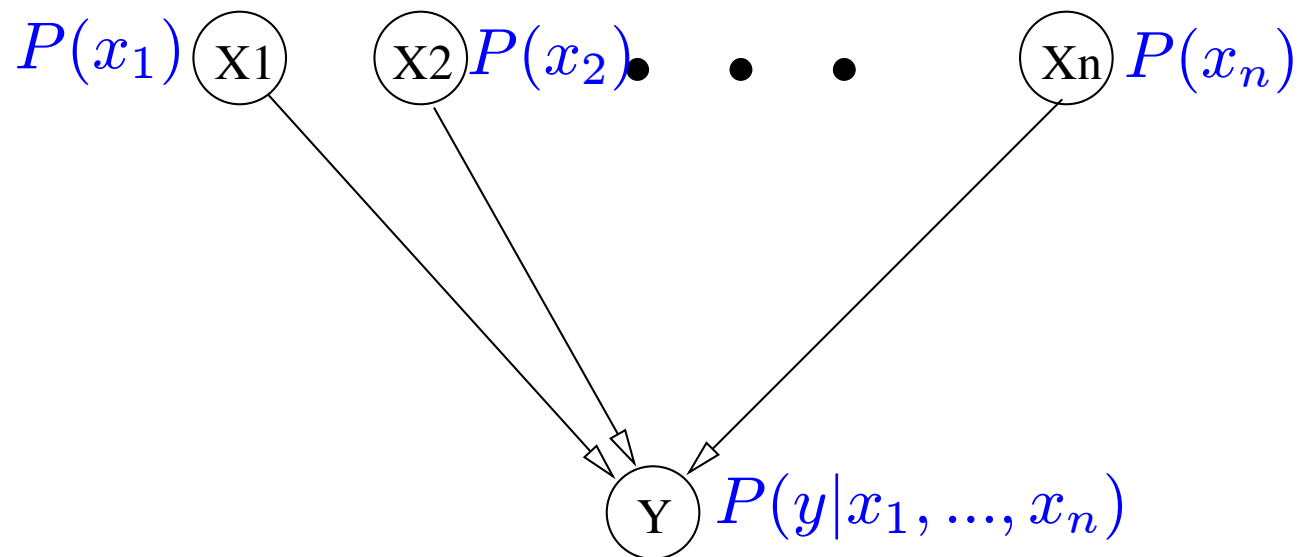
- Marginalization

$$\sum_b P(b, a) = P(a)$$

\sum_b	<table border="1"><thead><tr><th>a</th><th>b</th><th>P</th></tr></thead><tbody><tr><td>0</td><td>0</td><td>.1</td></tr><tr><td>0</td><td>1</td><td>.3</td></tr><tr><td>1</td><td>0</td><td>.2</td></tr><tr><td>1</td><td>1</td><td>.4</td></tr></tbody></table>	a	b	P	0	0	.1	0	1	.3	1	0	.2	1	1	.4	=	<table border="1"><thead><tr><th>a</th><th>P</th></tr></thead><tbody><tr><td>0</td><td>.4</td></tr><tr><td>1</td><td>.6</td></tr></tbody></table>	a	P	0	.4	1	.6
a	b	P																						
0	0	.1																						
0	1	.3																						
1	0	.2																						
1	1	.4																						
a	P																							
0	.4																							
1	.6																							

Interpreting Graphical Models I

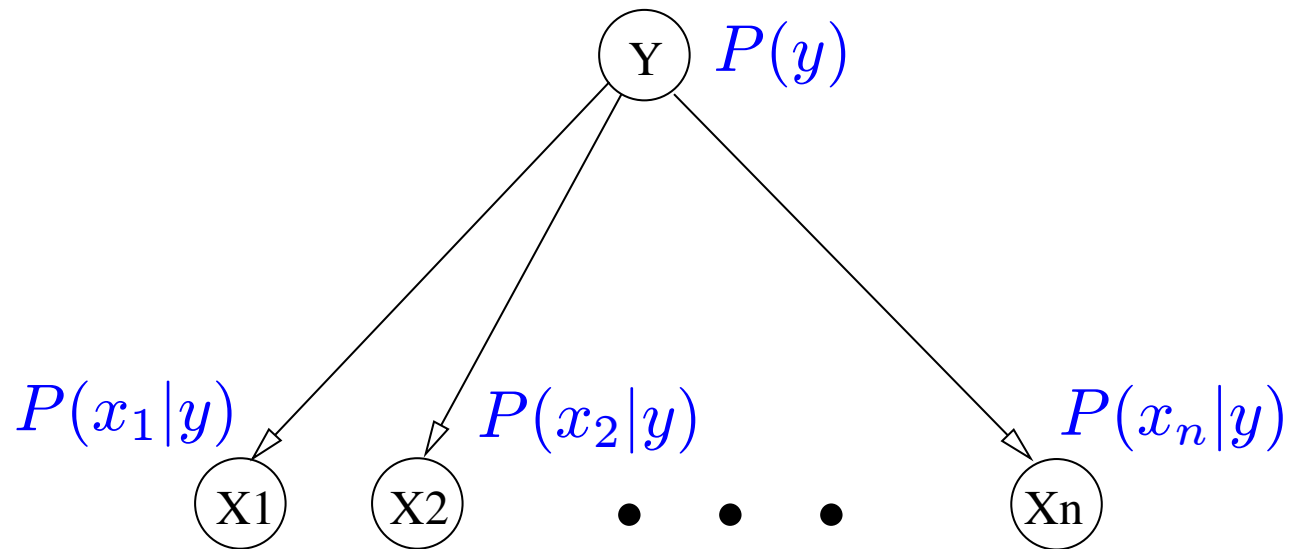
- Each edge corresponds to a conditional probability
- Root nodes correspond to prior probabilities
- Joint probability is just product of all edges and priors



$$P(y, x_1, \dots, x_n) = P(y|x_1, \dots, x_n)P(x_1) \cdots P(x_n)$$

Interpreting Graphical Models II

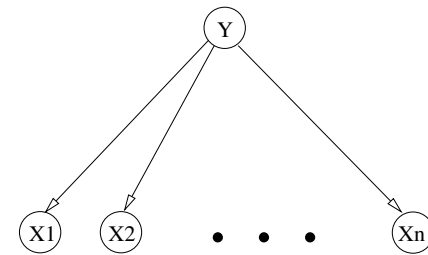
- Each edge corresponds to a conditional probability
- Root nodes correspond to prior probabilities
- Joint probability is just product of all edges and priors



$$P(y, x_1, \dots, x_n) = P(x_1|y) \cdots P(x_n|y)P(y)$$

Why Graphical Models?

- Represent a joint distribution over 40 RVs
 - How many parameters?
 - How efficient is inference?



- Need a compact representation
 - Exploit dependences (given by graph structure)
 - Fewer parameters (probabilities)
 - Easier to specify by humans
 - If learning, more data per parameter \Rightarrow lower variance
 - Can exploit graphical structure in inference
 - Variable elimination...

Variable Elimination I

- When marginalizing over x , factor out all probabilities independent of x :

$$\begin{aligned} P(y) &= \sum_{x_1, \dots, x_n} P(y|x_1, \dots, x_n)P(x_1) \cdots P(x_n) \\ &= \sum_{x_1, \dots, x_n} \underbrace{P(y|x_1, \dots, x_n)P(x_1) \cdots P(x_n)}_{O(2^{n+1})} \end{aligned}$$

– Curly braces show number of FLOPS

Variable Elimination II

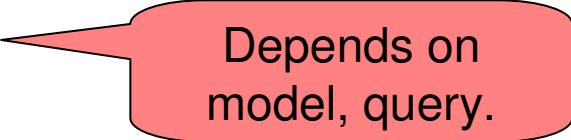
- When marginalizing over x , factor out all probabilities independent of x :

$$\begin{aligned} P(x_1) &= \sum_{y, x_2, \dots, x_n} P(y|x_1, \dots, x_n) P(x_1) \cdots P(x_n) \\ &= \sum_y \underbrace{P(y|x_1, \dots, x_n) P(x_1)}_{=1} \underbrace{P(x_2) \cdots P(x_n)}_{O(2^1)} \end{aligned}$$

– Curly braces show number of FLOPS

Graphical Models Recap

- This was a pragmatic introduction
 - Laws of probabilities
 - Discrete representation and operations
 - Use graphical models to structure joint dist.
 - Edges represent conditional dependences
 - Variable elimination inference to exploit structure
- Benefits
 - Compact representation
 - Easier specification / robust learning of params
 - Can often do efficient inference



Depends on
model, query.

What about Continuous RVs?

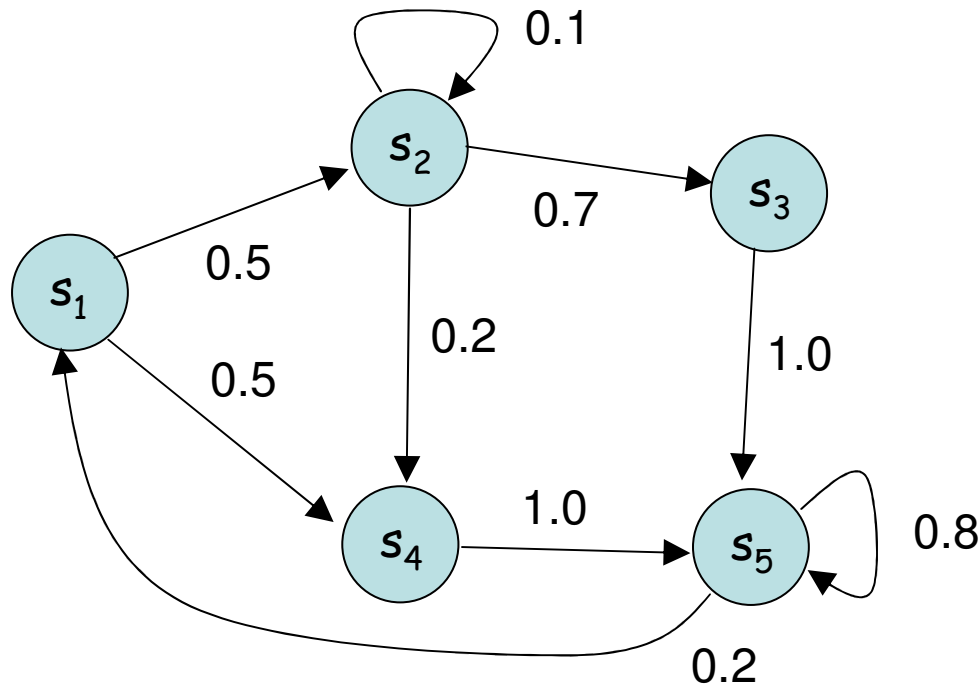
- For continuous case
 - Gaussians
 - Specification and inference work out cleanly
 - Many nice properties for algorithms
 - Loopy belief propagation
 - Non-gaussians
 - Specification often requires symbolic function representation
 - Inference requires algebraic manipulation
 - Some useful approximation algorithms
 - Expectation propagation
- We only focus on discrete probabilities in course
 - But many ideas generalize

Markov Models Crash Course

(a.k.a., Markov Chains or
Markov Processes)

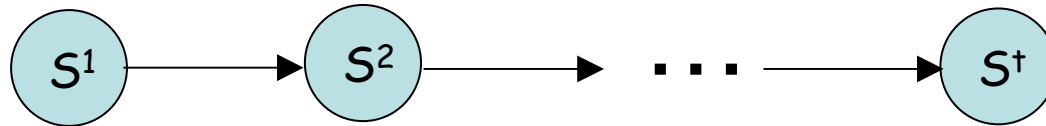
Markov Models (or Markov Chains)

- At each time step, probabilistically transition from current state to next state ($S = \{s_1, s_2, \dots, s_n\}$)
- Finite State Machine (FSM) view for $n=5$:



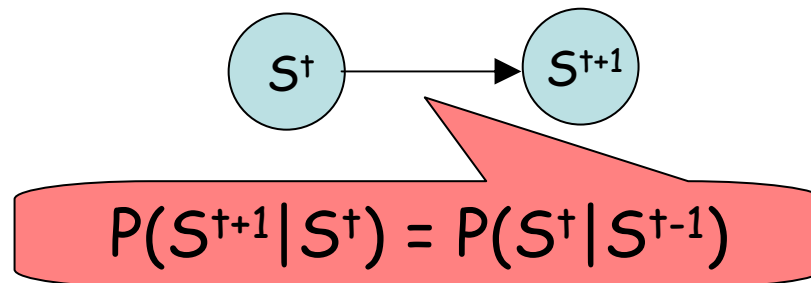
Markov Models

- The graphical model view for t steps:



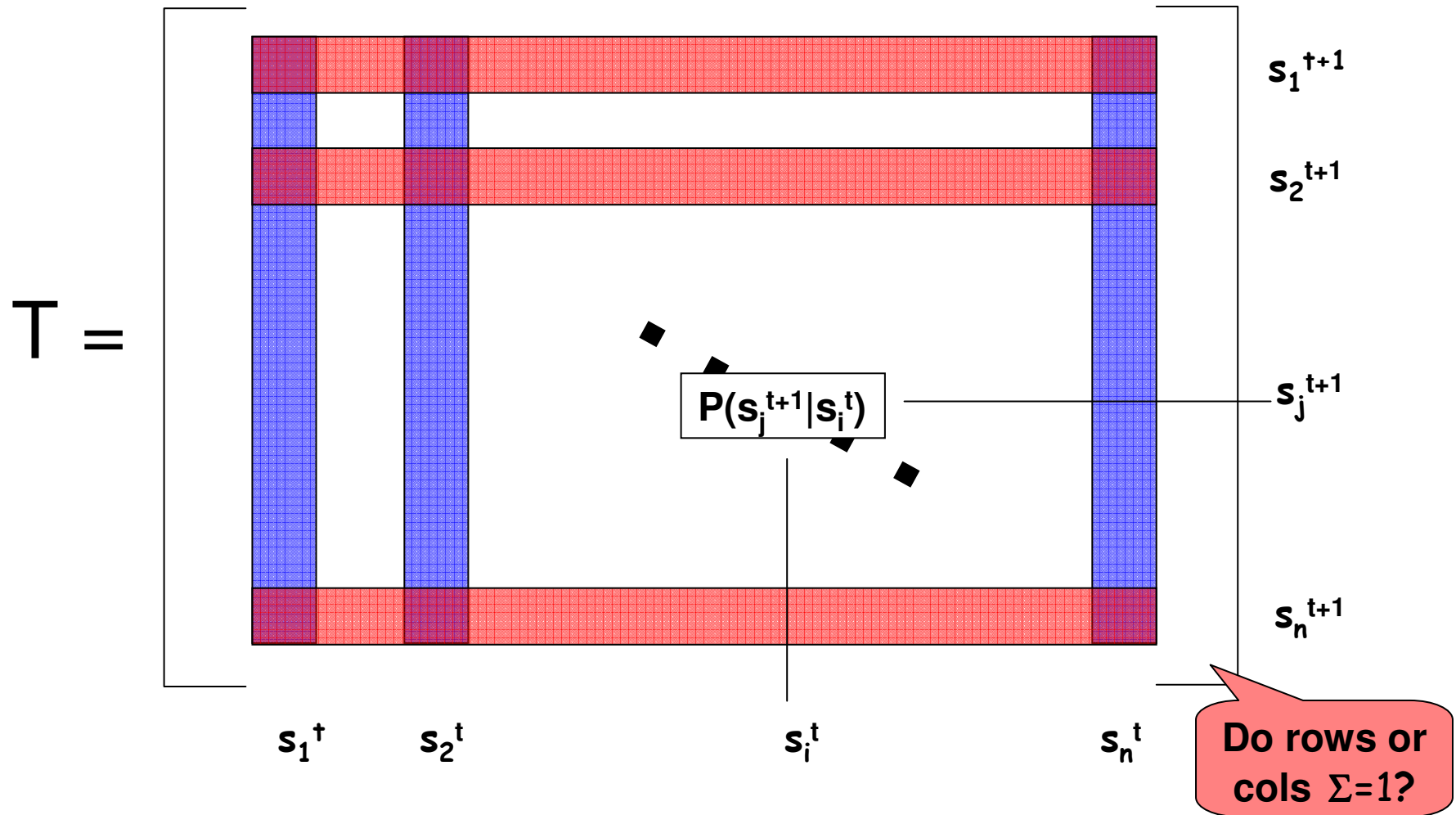
– Note: for $t = \infty$, an infinite graphical model!

- Or assuming transition stationarity, just:



Transition Matrix

- Represent $P(s^{t+1}|s^t)$ as transition matrix:



Transition Probabilities

- Formally

- Define state set $S^t = \{s_1, s_2, \dots, s_n\} ; \forall t$
- Define transition matrix $T_{ij}^t = P(S_i^{t+1} | S_j^t) ; \forall t$

- Properties of T_{ij}

- *Stationary*: $T_{ij}^t = T_{ij}^{t-1}$ OR $P(S^{t+1} | S^t) = P(S^t | S^{t-1}) ; \forall t$
- *Irreducible*: Possible to get from any s_i to s_j eventually
- *Aperiodic*: Time to return has periodicity = 1
- *Transient*: Positive probability of not returning to state
- *Recurrent*: Not transient
- *Ergodic*: Aperiodic and (positive) recurrent

Examples
of each?

Distribution at Time t

- Given $P(s^0)$, what is $P(s^t)$?
- Use var. elim. to marginalize over intermediate time steps
 - $P(s^t) = \sum_{s^i=s_1, \dots, s_{t-1}} P(s^0) \prod_{i=0 \dots t-1} P(s^{i+1} | s^i)$

If no evidence after time t, all factors for t+1 and after marginalize out

- Or let $P s^0$ & $P s^t$ be column vectors...
 - Then simply: $P s^t = (T^t) P s^0$
 - Note: Intimate connection between matrix ops and var. elim.
 - When $P(s^{i+1} | s^i)$ factors as a DBN...
capture many efficiencies of var. elim. via sparse matrix ops

Stationary Distribution

- Stationary Distribution π at $t=\infty$
 - $\pi = (T^{\infty}) P_S^0$
 - If T ergodic & irreducible, P_S^0 irrelevant
 - Reaches *unique* steady-state distribution: $\pi = T\pi$
 - So $\pi =$ any column of T^{∞}
 - Can solve via eigenvector analysis (note: $\lambda=1$)
 - Related to (Krylov) iterated eigenvector computation
 - Or use fixed point to solve linear system
 - $T\pi - \pi = 0 \rightarrow \pi' T' - \pi' = 0 \rightarrow \pi' (T' - I) = 0$
s.t. constraints on π
 - Can solve linear system via matrix inversion
 - » $(T' - I)$ guaranteed full rank \therefore invertible

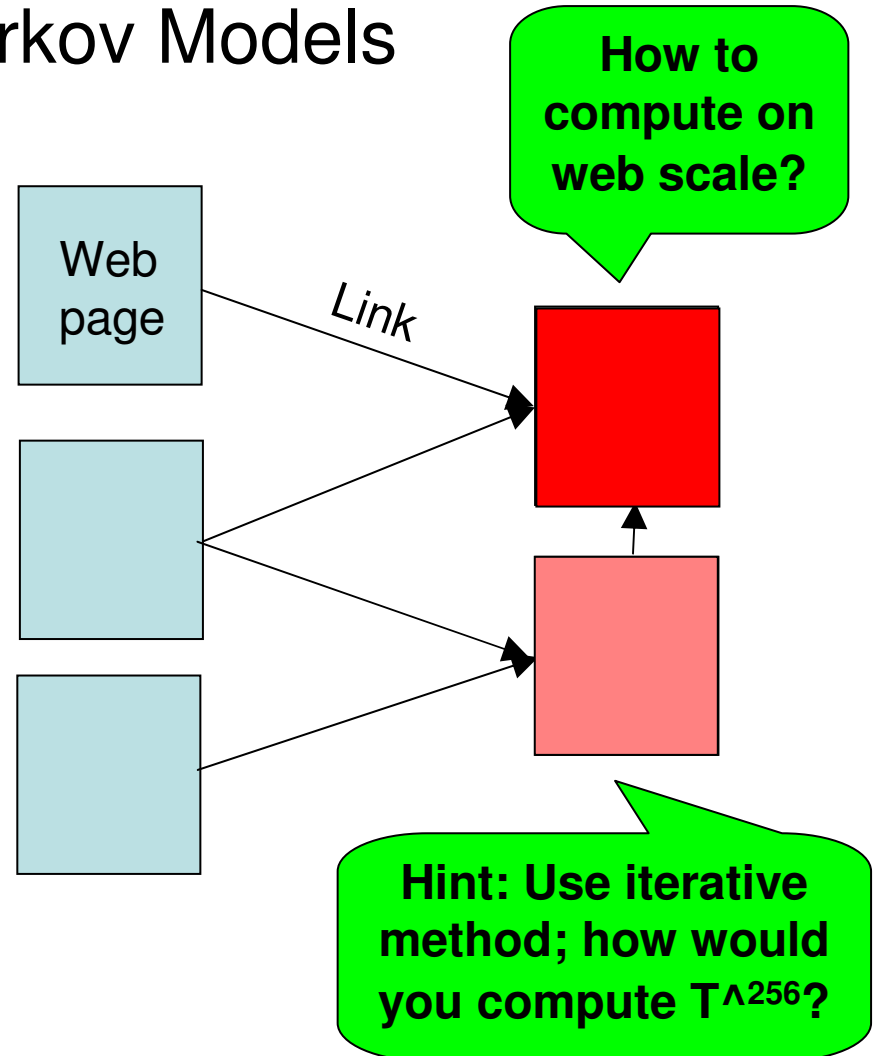
Why? What are they?

Markov Model Applications

- Simple theory, ingenious applications:
 - n^{th} -order Markov models
 - Relax Markovian assumption to previous n states
 - Used in text and speech processing
 - N-grams for predicting next word occurrence
 - Colocation identification
 - [Dasher](#) for text input, try it in your [web browser](#)
 - More generally
 - Physics (states of systems)
 - Queuing theory (random entries and exits)
 - Economics, Biology, Chemistry, etc...
 - Google!

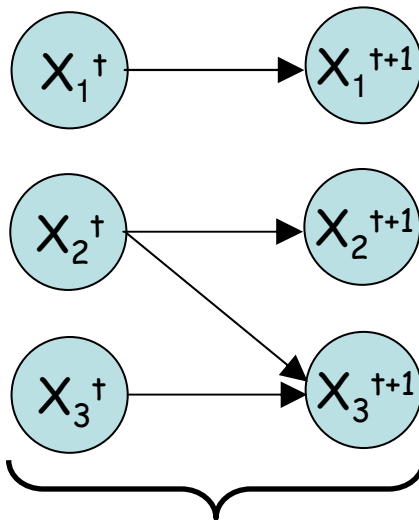
Google PageRank Example

- Very beautiful use of Markov Models
- Model of web browsing:
 - Probabilistically take link with $\sim 1/k$ chance if k links
 - Small chance of random transition
- Stationary distribution π gives PageRank!
 - Measure of “authority”



Factored Markov Models

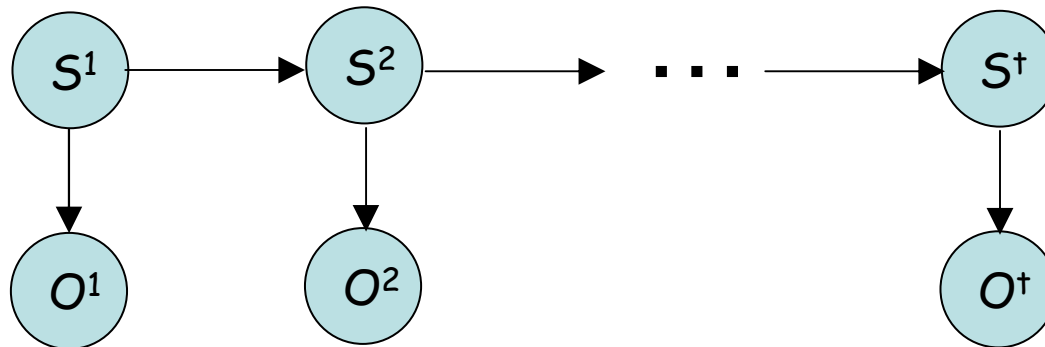
- The Dynamic Bayes Net (DBN) view:
 - State factors into variables: X_1, X_2, \dots, X_k
 - Capture transition independences



$$P(x_1^{t+1}, x_2^{t+1}, \dots, x_k^{t+1} \mid x_1^t, x_2^t, \dots, x_k^t) = \Pi \dots$$

Hidden Markov Models

- Formally
 - Define state set $S^t = \{s_1, s_2, \dots, s_n\} ; \forall t$
 - Define observation set $O^t = \{o_1, o_2, \dots, o_m\} ; \forall t$
 - Define observation prob $P(O^t | S^t); \forall t$
 - Define transition prob $P(S^{t+1} | S^t) ; \forall t$
- Graphical Model view:



Sequential Decision Theory

- We've looked at sequential *prediction*
 - *But* what if we can choose actions that affect model?
 - And differing utilities for different states?
- Fully observable case (MDP):
 - Markov Decision Process (MDP) = Markov Model + Actions
- Partially observable case (POMDP):
 - Partially Observable MDP = HMM + Actions
- Model-based reinforcement Learning (RL):
 - Learning *structure and parameters* from experience

Decision Diagrams

(A very useful data structure for representation and inference)

Function Representation (Tables)

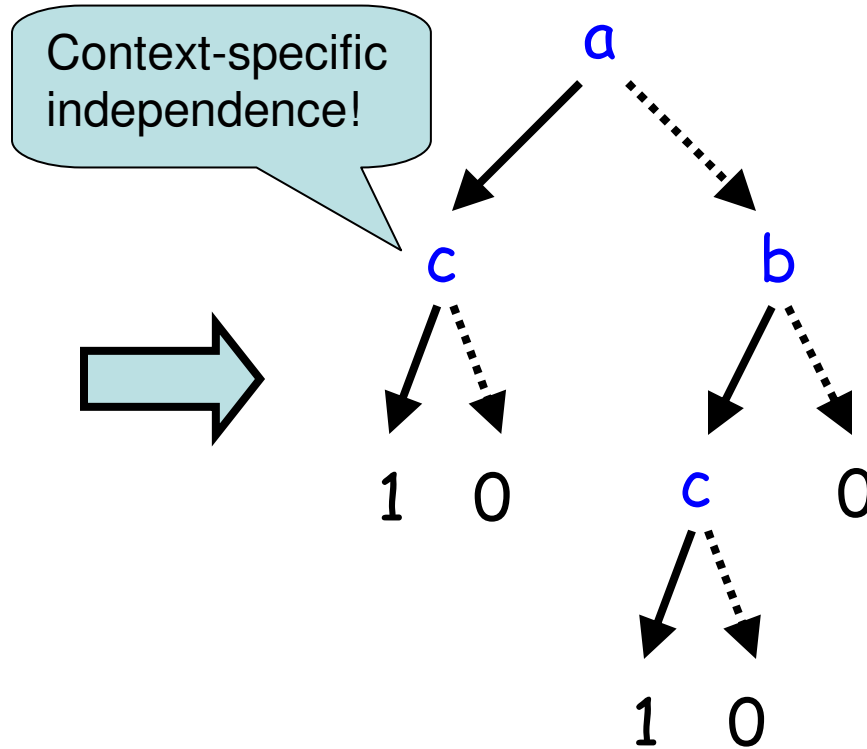
- How do we represent a function from $B^n \rightarrow \mathcal{R}$?
- How about a fully enumerated table...
- ...OK, but can we be more compact?

a	b	c	F(a,b,c)
0	0	0	0.00
0	0	1	0.00
0	1	0	0.00
0	1	1	1.00
1	0	0	0.00
1	0	1	1.00
1	1	0	0.00
1	1	1	1.00

Function Representation (Trees)

- How about a tree? Sure, now can simplify.

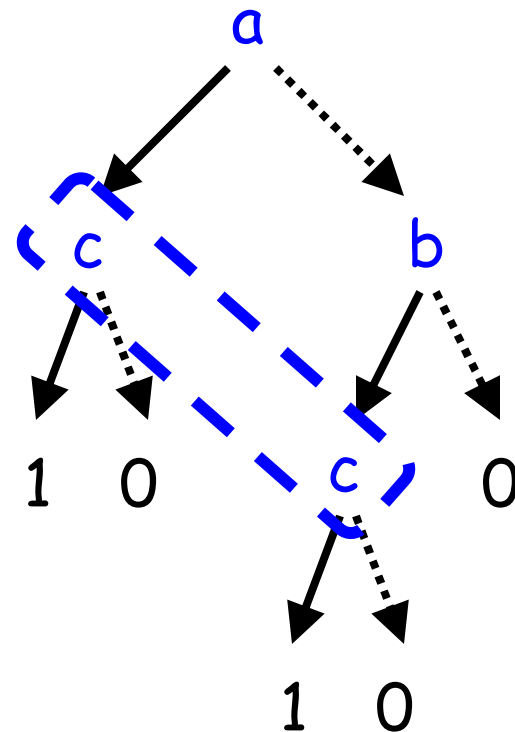
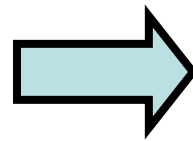
a	b	c	F(a,b,c)
0	0	0	0.00
0	0	1	0.00
0	1	0	0.00
0	1	1	1.00
1	0	0	0.00
1	0	1	1.00
1	1	0	0.00
1	1	1	1.00



Function Representation (ADDs)

- Why not a directed acyclic graph (DAG)?

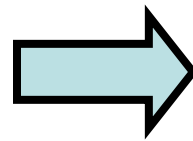
a	b	c	F(a,b,c)
0	0	0	0.00
0	0	1	0.00
0	1	0	0.00
0	1	1	1.00
1	0	0	0.00
1	0	1	1.00
1	1	0	0.00
1	1	1	1.00



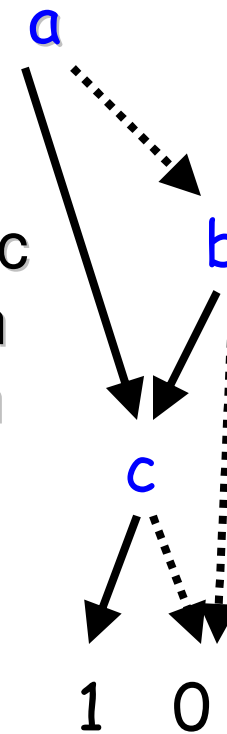
Function Representation (ADDs)

- Why not a directed acyclic graph (DAG)?

a	b	c	F(a,b,c)
0	0	0	0.00
0	0	1	0.00
0	1	0	0.00
0	1	1	1.00
1	0	0	0.00
1	0	1	1.00
1	1	0	0.00
1	1	1	1.00

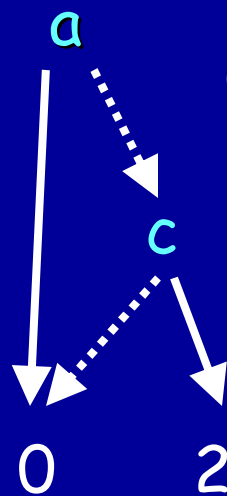
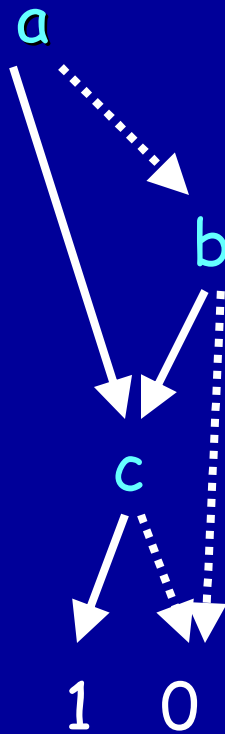


Algebraic
Decision
Diagram
(ADD)



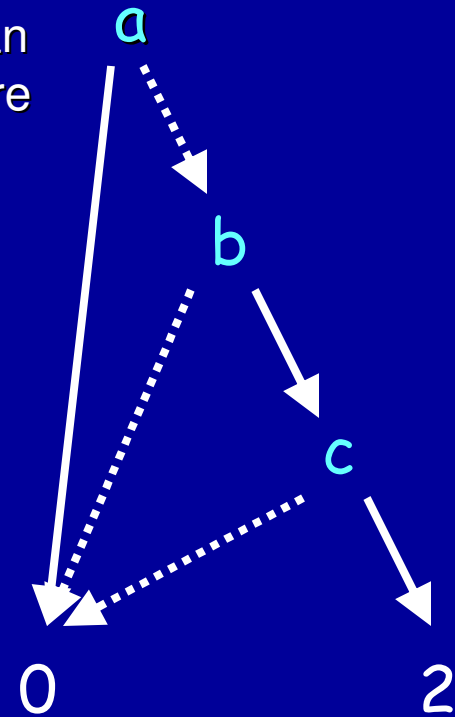
Binary Operations (ADDs)

- Why do we order the variables?
- This enables us to do efficient binary operations...




Result: ADD operations can be **much** more efficient than using tables

=



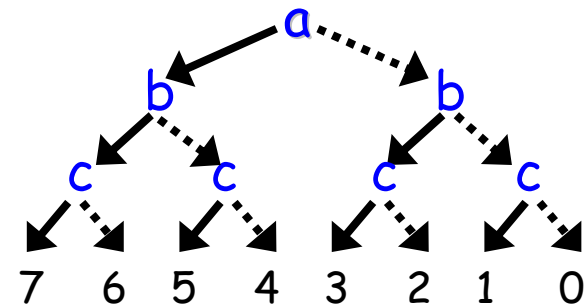
Why are we covering ADDs?

- Instead of representing discrete functions as tables...
- Represent as ADDs
 - Can multiply, add, max ADDs
 - Can even marginalize  How?
 - No worse than table space / time

ADD Inefficiency

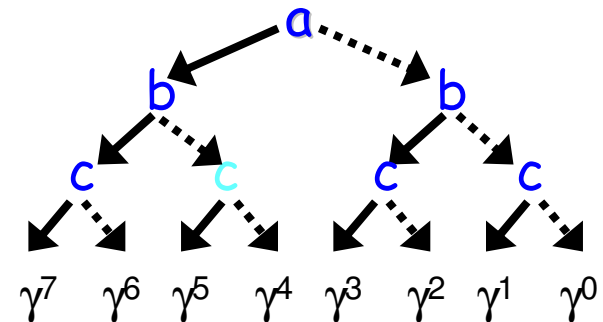
- Is context-specific independence enough?
- Or do we need more compactness?
- Example 1: Additive reward/utility functions

$$\begin{aligned} - R(a,b,c) &= R(a) + R(b) + R(c) \\ &= 4a + 2b + c \end{aligned}$$



- Example 2: Multiplicative value functions

$$\begin{aligned} - V(a,b,c) &= V(a) \cdot V(b) \cdot V(c) \\ &= \gamma^{(4a + 2b + c)} \end{aligned}$$



Factored MDPs

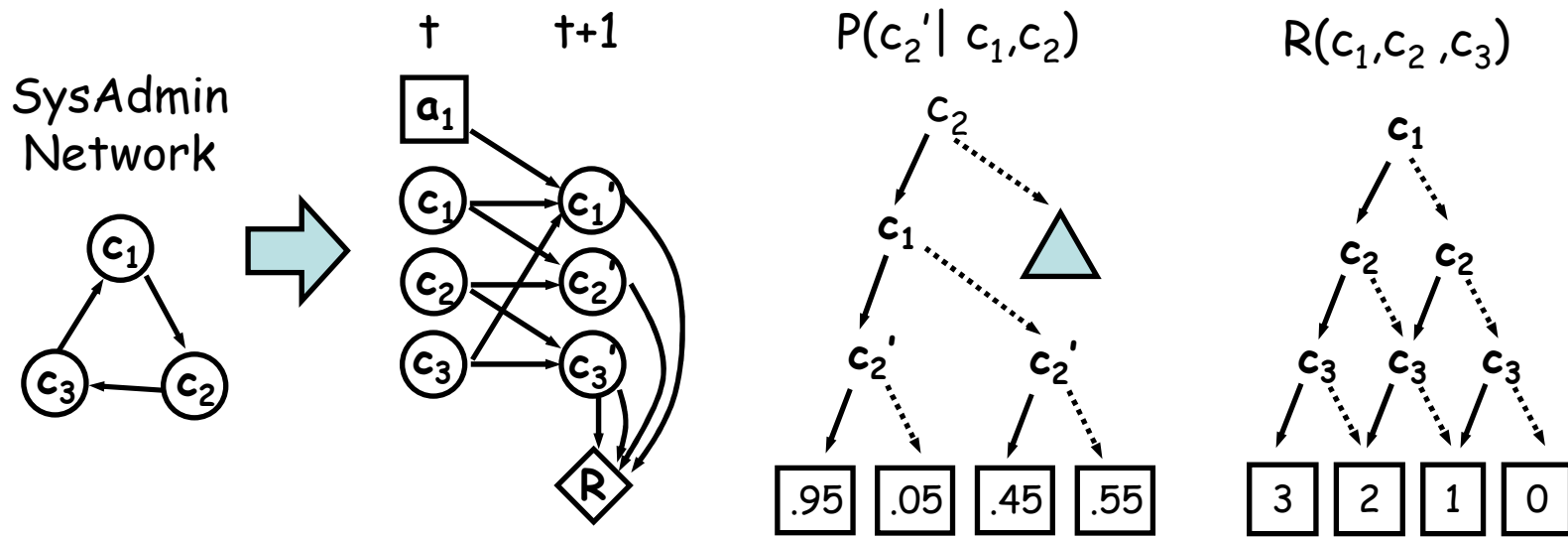
(Compactly specifying
and solving MDPs)

Factored MDPs

- State
 - Just state variables: (x_1, \dots, x_n)
 - How many distinct states?
- Actions
 - A discrete set $A = \{a_1, \dots, a_2\}$
 - As usual
- Transition and reward?

Factored MDPs and Value Iteration

- DBNs and ADDs to specify T & R in MDP:



- SPUDD (HSHB, 1999): factored value iteration

$$V^{t+1}(\vec{x}) = \max_{a \in A} \left\{ R(\vec{x}, a) + \gamma \sum_{\vec{x}'} \left[\prod_{i=1}^n P(x'_i | Par(x'_i), a) V^t(\vec{x}') \right] \right\}$$

Recap

- Bayes nets and inference
 - Dynamic Bayes nets
- Markov chains
- Decision diagrams
 - Often more compact than tables
 - Operations often more efficient
- Specifying factored MDPs with above
 - And solving using efficient repr. / inference

Next Week

- Approximation in factored MDPs:
 - APRICODD
 - Approximation with SPUDD
 - Linear-value Approximation
 - Uses neat trick for solving LPs with constraint generation