Bayesact I - Background

Jesse Hoey

February 6, 2014

Readings:

- M.Sanjeev Arulampalam, Simon Maskell, Neil Gordon and Tim Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. IEEE Transactions on Signal Processing, Vol 50, No 2, Feb 2002.
- David Silver and Joel Veness Monte Carlo Planning in Large POMDPs. In NIPS 2010.

Background: David Poole and Alan Mackworth. Artificial Intelligence: Foundations of Computational Agents. Cambridge University Press, 2010. Chapters 6 and 9 in particular. See `artint.info`

Uncertainty

Why is uncertainty important?

- Agents (and humans) don't know **everything**,
- but need to make decisions anyways!
- Decisions are made in the absence of information,
- or in the presence of **noisy** information (sensor readings)

The best an agent can do:
know how uncertain it is, and act accordingly

Probability: Frequentist vs. Bayesian

Frequentist view:

probability of heads = # of heads / # of flips

probability of heads **this time** = probability of heads (history)

Uncertainty is **ontological**: pertaining to the world

Bayesian view:

probability of heads **this time** = agent's **belief** about this event

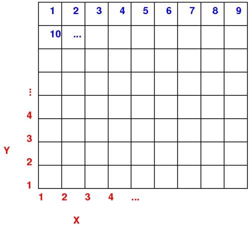
belief of agent A : based on previous experience of agent A

Uncertainty is **epistemological**: pertaining to knowledge

Features

Describe the **world** in terms of a set of **states**: $\{s_1, s_2, ..., s_N\}$

or, as the product of a set of **features**
(also known as **attributes** or **random variables**)

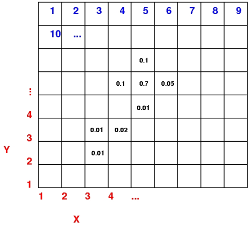


- Number of states $= 2^{number\ \ of\ \ binary\ \ features}$
- Features describe the state space in a **factored** form.
- state $\rightarrow$ factorize $\rightarrow$ feature values
- feature values $\rightarrow$ cross product $\rightarrow$ states

Probability Measure

if X is a random variable (feature, attribute),
it can take on values x, where $x \in Domain(X)$
$Pr(X = x) \equiv Pr(x)$ is the probability that $X = x$
joint probability $Pr(X = x, Y = y) \equiv Pr(x, y)$ is the probability that $X = x$ and $Y = y$ at the same time

Joint probability distribution:



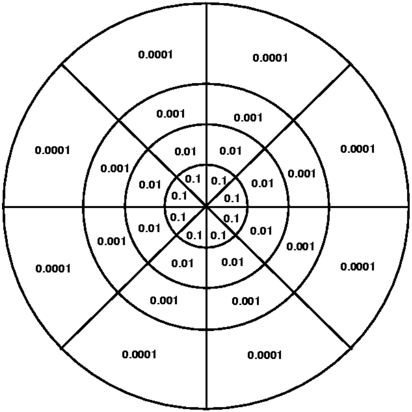
Sum Rule:

$$\sum_x Pr(X = x, Y) = Pr(Y)$$

We call $Pr(Y)$ the marginal distribution over Y

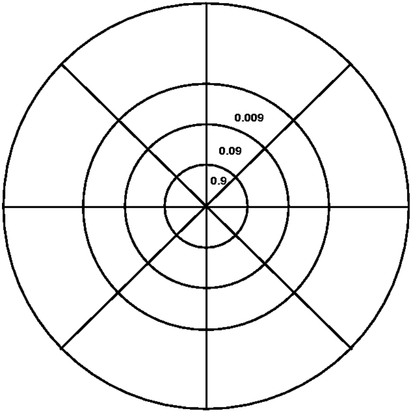
Independence

- describe a system with n features: $2^n - 1$ probabilities
- Use **independence** to reduce number of probabilities
- e.g. radially symmetric dartboard, Pr(hit a sector)
- $Pr(sector) = Pr(r, \theta)$ where $r = 1, \ldots, 4$ and $\theta = 1, \ldots, 8$.
- 32 sectors in total - need to give 31 numbers



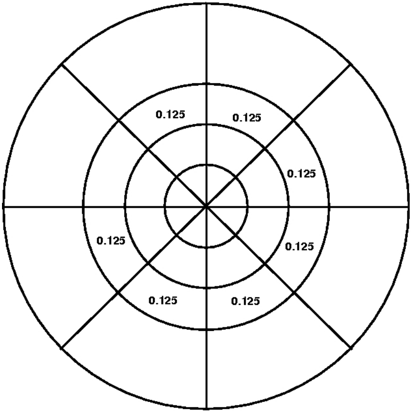
Independence

- describe a system with n features: $2^n - 1$ probabilities
- Use **independence** to reduce number of probabilities
- e.g. radially symmetric dartboard, Pr(hit a sector)
- assume radial independence: $Pr(r, \theta) = Pr(r)Pr(\theta)$
- only need 7+3=10 numbers



Independence

- describe a system with n features: $2^n - 1$ probabilities
- Use **independence** to reduce number of probabilities
- e.g. radially symmetric dartboard, Pr(hit a sector)
- assume radial independence: $Pr(r, \theta) = Pr(r)Pr(\theta)$
- only need 7+3=10 numbers



Conditional Probability

if X and Y are random variables, then

$Pr(x|y)$ is the probability that $X = x$ **given** that $Y = y$.
e.g. $Pr(flies|is\_bird) = ?$

Conditional Independence
$Pr(flies|is\_bird, has\_wings) = Pr(flies|is\_bird)$

so learning *has_wings* doesn't influence beliefs about *flies* if you already know *is_bird*
Product rule (Chain rule):
$Pr(flies, is\_bird) = Pr(flies|is\_bird)Pr(is\_bird)$
$Pr(flies, is\_bird) = Pr(is\_bird|flies)Pr(flies)$
leads to : Bayes' rule
$Pr(is\_bird|flies) = \frac{Pr(flies|is\_bird)Pr(is\_bird)}{Pr(flies)}$

Conditional Probability

if X and Y are random variables, then

$Pr(x|y)$ is the probability that $X = x$ **given** that $Y = y$.
e.g. $Pr(flies|is\_bird) = ?$

Conditional Independence
$Pr(flies|is\_bird, has\_wings) = Pr(flies|is\_bird)$

so learning *has_wings* doesn't influence beliefs about *flies* if you already know *is_bird*

Product rule (Chain rule):

$Pr(flies, is\_bird) = Pr(flies|is\_bird)Pr(is\_bird)$
$Pr(flies, is\_bird) = Pr(is\_bird|flies)Pr(flies)$

leads to : Bayes' rule

$Pr(is\_bird|flies) = \frac{Pr(flies|is\_bird)Pr(is\_bird)}{Pr(flies)}$

Why is Bayes' theorem interesting?

- Often you have causal knowledge:
  $Pr(\mathit{flies} \mid \mathit{is\_bird})$
  $Pr(\mathit{symptom} \mid \mathit{disease})$
  $Pr(\mathit{alarm} \mid \mathit{fire})$
  $Pr($*image looks like*  $\mid$ *a tree is in front of a car*$)$
- and want to do evidential reasoning:
  $Pr(\mathit{is\_bird} \mid \mathit{flies})$
  $Pr(\mathit{disease} \mid \mathit{symptom})$
  $Pr(\mathit{fire} \mid \mathit{alarm})$.
  $Pr($*a tree is in front of a car* $\mid$ *image looks like* $)$

Updating belief: Bayes' Rule

Agent has a **prior belief** in a **hypothesis**, h, $Pr(h)$,

Agent observes some **evidence** e
that has a **likelihood** given the hypothesis: $Pr(e|h)$.

The agent's **posterior belief** about h after observing e, $Pr(h|e)$,

is given by **Bayes' Rule:**

$$Pr(h|e) = \frac{Pr(e|h)Pr(h)}{Pr(e)} = \frac{Pr(e|h)p(h)}{\sum_h Pr(e|h)Pr(h)}$$

Expected Values

expected value of a function on X, $V(X)$:

$\mathbb{E}_{Pr(x)}(V) = \sum_{x \in Dom(X)} Pr(x)V(x)$

where $Pr(x)$ is the probability that $X = x$.

This is useful in decision making, where $V(X)$ is the *utility* of situation X.

Bayesian decision making is then

$$\begin{aligned}&\arg\max_{decision} \mathbb{E}_{Pr(outcome|decision)}(V(\mathtt{decision}))\\&\quad = \arg\max_{decision} \sum_{outcome} Pr(outcome|decision)V(outcome)\end{aligned}$$

Value of Independence

- complete independence reduces both *representation* and *inference* from $O(2^n)$ to $O(n)$
- Unfortunately, complete mutual independence is rare
- Fortunately, most domains do exhibit a fair amount of *conditional independence*
- **Bayesian Networks** or **Belief Networks** (BNs) encode this information

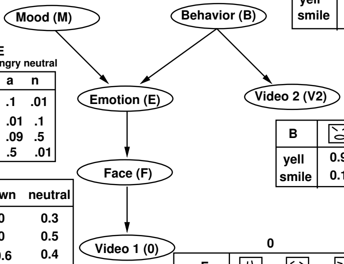
Bayesian Networks

A Bayesian Network (Belief Network, Probabilistic Network) or BN over variables $\{X_1, X_2, \ldots, X_N\}$ consists of:

- a DAG whose nodes are the variables
- a set of Conditional Probability tables (CPTs) giving $Pr(X_i|Parents(X_i))$ for each X_i

M	Pr(M)
good	0.8
bad	0.2

B	Pr(B)
yell	0.1
smile	0.9



E: happy sad angry neutral

M	B	h	s	a	n
good	yell	.01	.88	.1	.01
good	smile	.88	.01	.01	.1
bad	yell	.01	.4	.09	.5
bad	smile	.4	.09	.5	.01

v2

B		
yell	0.9	0.1
smile	0.1	0.9

F

E	smile	cry	frown	neutral
happy	0.7	0	0	0.3
sad	0	0.5	0	0.5
angry	0	0	0.6	0.4
neutral	0	0	0	1.0

0

F				
smile	0.85	0.05	0.05	0.05
cry	0.05	0.85	0.05	0.05
frown	0.05	0.05	0.85	0.05
neutral	0.05	0.05	0.05	0.85

Stochastic Simulation

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

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

$\leftrightarrow$

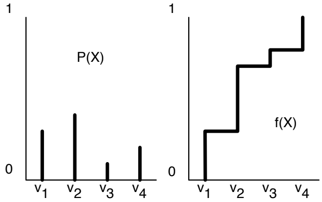
X	$probability$
x_1	n_1/m
$\vdots$	$\vdots$
x_k	n_k/m

- 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: $f(x) = Pr(X \leq x)$.
- 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

- $Pr(|s - p| > \epsilon) \leq 2e^{-2n\epsilon^2}$
- 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.

Sampling for a belief network: inference

Sample	Mood	Behavior	Emotion	V2	Face	0
s_1	good	smile	happy	smile	smile	smile
s_2	good	yell	sad	yell	smile	smile
s_3	bad	smile	sad	smile	cry	cry
s_4	bad	yell	angry	yell	frown	frown
s_5	good	smile	happy	smile	smile	smile
s_6	bad	smile	sad	smile	smile	smile
			...			
s_{1000}	bad	yell	angry	yell	frown	frown

To get $Pr(H = h_i | Ev = ev_i)$ simply

- count the number of samples that have $H = h_i$ and $Ev = ev_i$, $N(h_i, ev_i)$
- divide by the number of samples that have $E = e_i$, $N(ev_i)$
- $Pr(H = h_i | Ev = ev_i) = \frac{Pr(H=h_i \wedge Ev=ev_i)}{Pr(Ev=ev_i)} = \frac{N(h_i, ev_i)}{N(ev_i)}$

Only need those samples that have $Ev = ev_i$: Rejection sampling

Importance Sampling

- If we can compute $Pr(evidence|sample)$ we can weight the (partial) sample by this value.
- To get the posterior probability, we do a weighted sum over the samples; weighting each sample by its probability.
- We don't need to sample all of the variables as long as we weight each sample appropriately.
- We thus mix exact inference with sampling.
- Don't even have to draw from any true distribution, $Pr(B)$
- Draw from *proposal* $q(B) \rightarrow b_i$,
- additionally weight by $Pr(b_i)/q(b_i)$

Importance Sampling

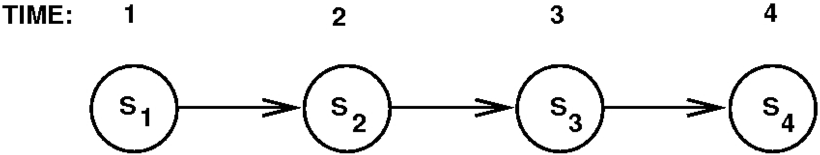
e.g. given evidence v_2, o, m, we can draw i^{th} sample:

1. draw from $q(B) \to b_i$
2. draw from $Pr(E|m, b_i) \to e_i$
3. weight by $Pr(v_2|b_i)Pr(o|e_i)Pr(b_i)/q(b_i)$, where $Pr(o|e_i) = \sum_f Pr(o|f)Pr(f|e_i)$

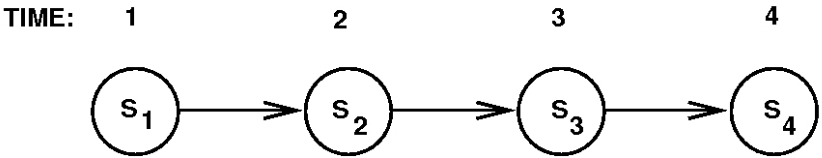
W_e: sum of all weights for all samples with $e_i = e$
W : sum of all weights
$Pr(E = e|v_2, o) = W_e/W$

Probability and Time

- 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



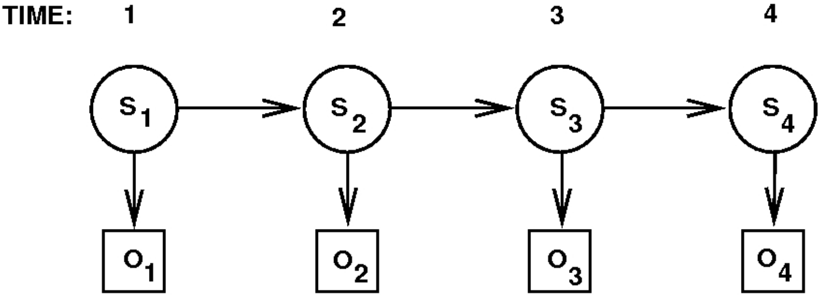
Markov assumption



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

This distribution gives the **dynamics** of the Markov chain

Hidden Markov Models (HMMs)



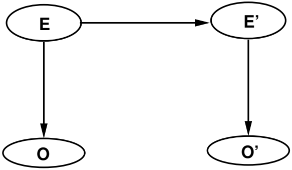
Add: observations O_t and
observation function $Pr(O_t|S_t)$
Given a sequency of observations $O_1, \ldots, O_t$, can estimate
filtering:

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

or **smoothing**, for $k < t$

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

Emotions over time



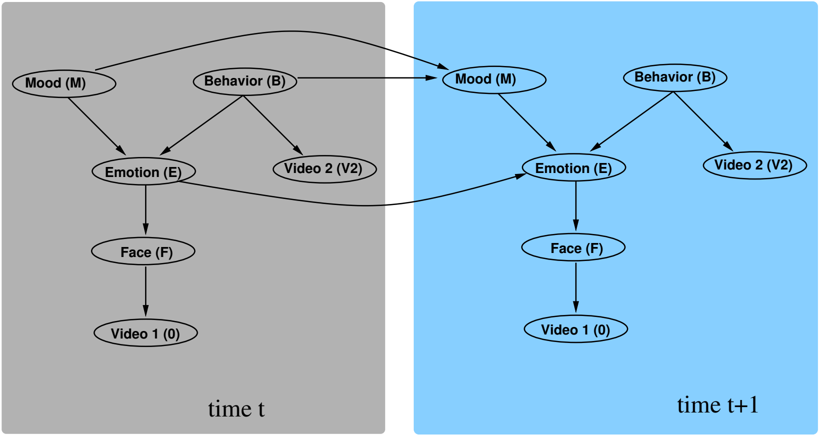
E'

E	happy	sad	angry	neutral
happy	0.9	0.04	0.01	0.05
sad	0.01	0.5	0.2	0.28
angry	0.01	0.2	0.5	0.28
neutral	0.5	0.1	0.1	0.3

0

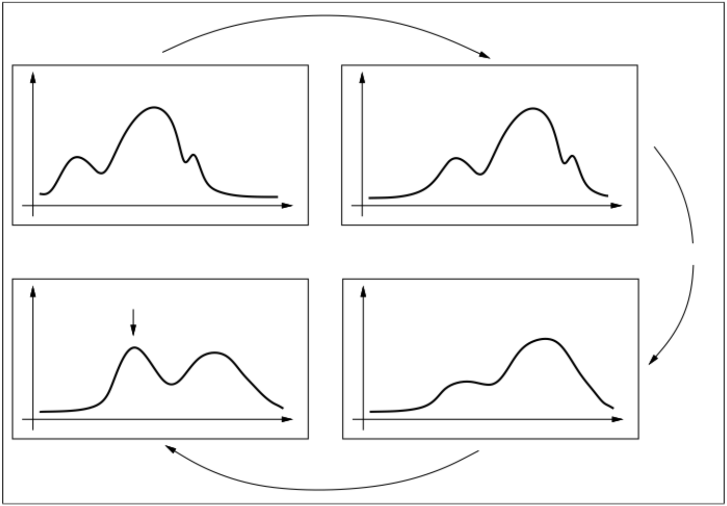
E				
happy	0.85	0.05	0.05	0.05
sad	0.05	0.85	0.05	0.05
angry	0.05	0.05	0.85	0.05
neutral	0.05	0.05	0.05	0.85

Dynamics Bayesian Networks (DBNs)

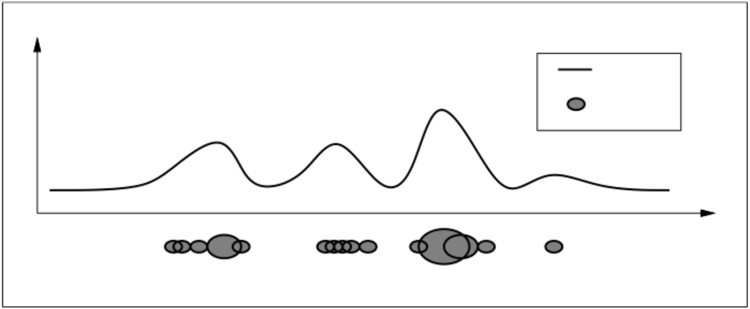


Particle Filtering

Evidence arrives over time:

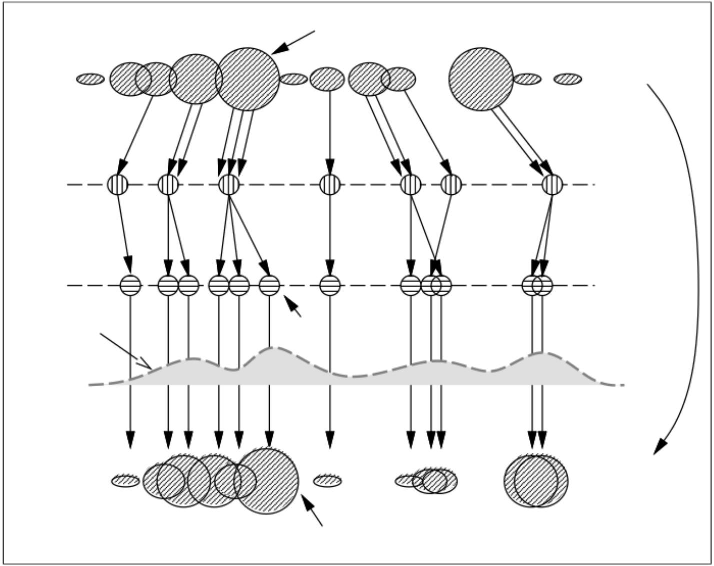


Represent distributions with samples:

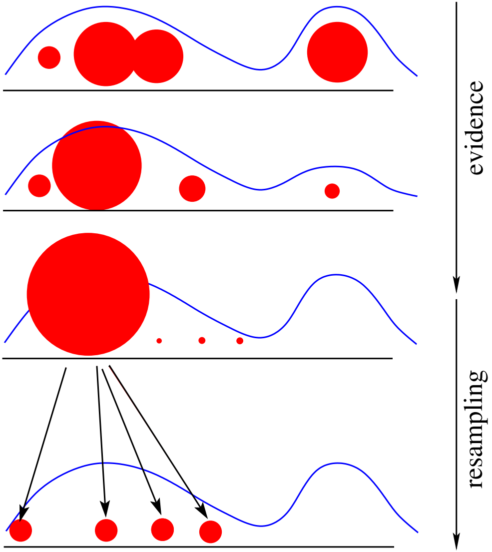


Particle Filtering

Update samples using particle filter:



Resampling

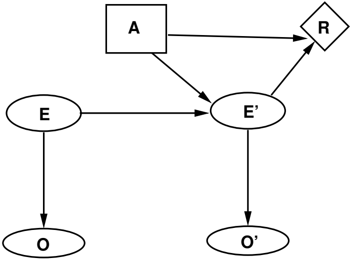


- avoids degeneracies in the samples
- all importance weights $\to 0$ except one
- performance of the algorithm depends on the resampling method.

POMDPs

Partially Observable Markov Decision Process:

A: smile, yell, sympathise, do_nothing



A: do_nothing

E	happy	sad	angry	neutral
	E'			
happy	0.9	0.04	0.01	0.05
sad	0.01	0.5	0.2	0.28
angry	0.01	0.2	0.5	0.28
neutral	0.5	0.1	0.1	0.3

E	R(E)
happy	1.0
sad	–0.1
angry	–1.0
neutral	0.0

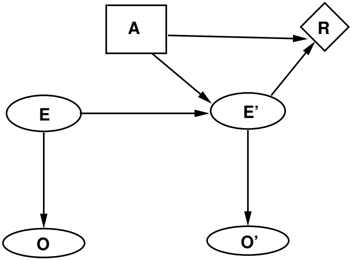
0

E				
happy	0.85	0.05	0.05	0.05
sad	0.05	0.85	0.05	0.05
angry	0.05	0.05	0.85	0.05
neutral	0.05	0.05	0.05	0.85

POMDPs

Partially Observable Markov Decision Process:

A: smile, yell, sympathise, do_nothing



A: smile_at

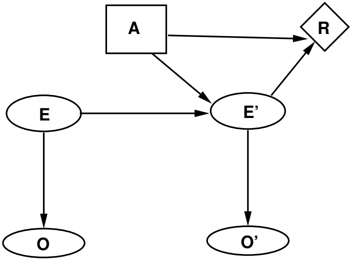
	E’			
E	**happy**	**sad**	**angry**	**neutral**
happy	0.99	0.0	0.0	0.01
sad	0.01	0.9	0.01	0.08
angry	0.01	0.2	0.3	0.48
neutral	0.7	0.1	0.1	0.1

E	R(E)
happy	1.0
sad	–0.1
angry	–1.0
neutral	0.0

	0			
E				
happy	0.85	0.05	0.05	0.05
sad	0.05	0.85	0.05	0.05
angry	0.05	0.05	0.85	0.05
neutral	0.05	0.05	0.05	0.85

Partially Observable Markov Decision Process:

A: smile, yell, sympathise, do_nothing



A: sympathise_with

E	E' happy	sad	angry	neutral
happy	0.8	0.14	0.01	0.05
sad	0.2	0.3	0.1	0.4
angry	0.01	0.1	0.7	0.18
neutral	0.3	0.1	0.1	0.5

E	R(E)
happy	1.0
sad	−0.1
angry	−1.0
neutral	0.0

0

E				
happy	0.85	0.05	0.05	0.05
sad	0.05	0.85	0.05	0.05
angry	0.05	0.05	0.85	0.05
neutral	0.05	0.05	0.05	0.85

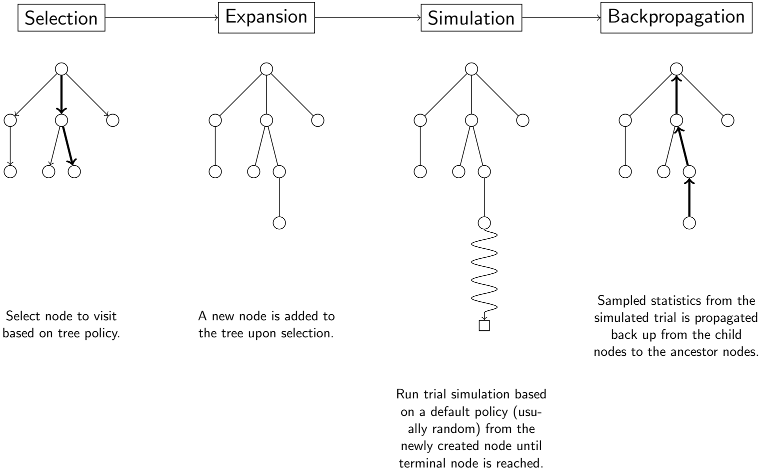
Policies

Policy: maps beliefs states into actions $\pi(b(s)) \rightarrow a$

Two ways to compute a policy

1. Backwards search
   - Dynamic programming (Variable Elimination)
   - in MDP:
     $Q_t(s,a) = R(s,a) + \gamma \sum_{s'} Pr(s'|s,a) \max_{a'} Q_{t-1}(s',a')$
   - in POMDP: $Q_t(b(s),a)$
2. Forwards search : Monte Carlo Tree Search (MCTS)
   - Expand the search tree
   - Expand more deeply in promising directions
   - Ensure exploration using e.g. UCB

MCTS


Selection
Expansion
Simulation
Backpropagation
Select node to visit based on tree policy.
A new node is added to the tree upon selection.
Run trial simulation based on a default policy (usually random) from the newly created node until terminal node is reached.
Sampled statistics from the simulated trial is propagated back up from the child nodes to the ancestor nodes.

Next:

- Bayesian Affect Control Theory (I)
- Bayesian Affect Control Theory (II)