MDPs

Rewards w/ discounted utility

$$U([r_0, \dots r_\infty]) = \sum_{t=0}^{\infty} \gamma^t r_t \leq \frac{R_{max}}{1-\gamma}$$

Value Iteration

$V_0(s) = 0$

$$V_{k+1}(s) = \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V_k(s')]$$

repeat until convergence $O(S^2 A)$

Bellman Equations

$V^*(s) = \max_a Q^*(s,a)$ optimal val of a state

$$Q^*(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$$

optimal value of (s,a) is expected val of an agent utility after taking a from s and acting optimally

Policy Evaluation

$V_0^\pi(s) = 0$ $O(S^2)$ per iteration

$$V_{k+1}^\pi(s) = \sum_{s'} T(s,\pi(s),s')[R(s,\pi(s),s') + \gamma V^\pi(s')]$$

Policy Extraction

$$\pi^*(s) = \arg\max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$$

Policy Iteration

$$V_{k+1}^{\pi_i}(s) = \sum_{s'} T(s,\pi_i(s),s')[R(s,\pi_i(s),s') + \gamma V_k^{\pi_i}(s')]$$

improvement: $\pi_{i+1}(s) = \arg\max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^{\pi_i}(s')]$

Probability

$$P(A|B) = \frac{P(A) \cdot P(B|A)}{P(B)} = \frac{P(A \cap B)}{P(B)} =$$

Joint: $P(A,B) = P(B)P(A|B)$

$$P(x_1, x_2, \dots x_n) = \prod_i P(x_i | x_1, \dots x_{i-1})$$

Bayes Net

can calculate any prob using chain rule

Inference

Variable elimination

1. Join (multiply) factors involving X
2. Sum out X

factors are unnormalized

$$P(A|B) = \frac{P(A) * P(B|A)}{P(A) * P(B|A) + P(\bar{A}) * P(B|\bar{A})}$$

Backprop

$h_1 = f_1(x)$ $h_2 = f_2(x)$ $z = h_1 h_2$

$$\frac{\partial z}{\partial x} = h_2 \frac{\partial f_1}{\partial x} + h_1 \frac{\partial f_2}{\partial x}$$

softmax each output is between 0 and 1

↳ total sums to 1 (Prob distribution)

Model-Based Learning

Transitions: $\frac{\#(s,a,s')}{\#(s,a)}$

Reward: $R(s,a,s')$

$${}_nP_k = \frac{n!}{n-k!}$$

$${}_nC_k = \frac{n!}{(n-k)!\,k!}$$

Model Free Learning

Passive Reinforcement

Direct Eval

- fix policy π
- Track utility & count of transitions
- compute est. val w/ utility/visits

TD Learning

$$sample = R(s,\pi(s),s') + \gamma V^\pi(s')$$

Update: $V^\pi(s) = (1-\alpha)V^\pi(s) + (\alpha)sample$

$$V^\pi(s) = V^\pi(s) + \alpha(sample - V^\pi(s))$$

Q-Learning (off-policy learning)

$$Q_{k+1}(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma \max_{a'} Q_k(s',a')]$$

Learn $Q(s,a)$ values as you go

$$sample = R(s,a,s') + \gamma \max_{a'} Q(s',a')$$

$$Q(s,a) = (1-\alpha)Q(s,a) + (\alpha)[sample]$$

Q-val Iteration

$$Q_{k+1}(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma \max_{a'} Q_k(s',a')]$$

Approximate Q-learning

- feature based representation
- use V & Q as linear value functions

$$Q(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \dots + w_n f_n(s,a)$$

Transitions $= (s,a,r,s')$

difference $= [r + \gamma \max_{a'} Q(s',a')] - Q(s,a)$

$Q(s,a) = Q(s,a) + \alpha[\text{difference}]$ Exact Q's

$w_i = w_i + \alpha[\text{difference}] f_i(s,a)$ Approx. Q's

ε-greedy

- prob $0 \leq \varepsilon \leq 1$ act randomly and explore w/ prob ε
- act w/ $(1-\varepsilon)$ w/ curr policy
- large ε is mostly random

Exploration fn

Modified Q-update: $Q(s,a) = \alpha R(s,a,s') + \gamma \max_{a'} f(Q(s',a'), N(s',a'))$

$f(u,n) = u + \frac{k}{n}$ encouraged to explore states less visited

Search Problems

- state space = all possible states
- successor fn: takes in state, action and computes cost and successor state
- start state: initial state
- goal test: test if state is goal

Manhattan distance $= |x_1 - x_2| + |y_1 - y_2|$

Admissibility: $\forall n, 0 \le h(n) \le h^*(n)$

$g(n)$: total backward cost comp. by UCS (arc cost)

$h(n)$: estimated forward cost

$g(n)+h(n)$: total cost used by arc

consistency: $\forall A, C,\ h(A) - h(C) \le cost(A,C)$ — along a path f value never decreases

$H(G)=0$ to be admissible & consistent

consistency ⇒ admissibility $h(A) \le cost(A,C) + h(C)$

Dominance: $h_a \ge h_c$ if $\forall n,\ h_a(n) \ge h_c(n)$

Search:

DFS: explores deepest node, LIFO stack

complete: no, can get caught in cycles   Time: worst case $O(b^m)$ (b: branching factor, m: max depth)

optimal: no   space: $O(bm)$

BFS: selects shallowest node. FIFO queue

complete: yes, if sol'n exists   Time $O(b^s)$, solution at depth s

optimal: no, doesn't account for cost.   space: $O(b^s)$

optimal if all edges = 1

UCS: selects lowest cost. Heap based pq (cumulative cost)

complete: yes, if sol'n exists   Time: $O(b^{C^*/\varepsilon})$

optimal: yes, as long as positive edges   Space: $O(b^{C^*/\varepsilon})$

C^* optimal path cost

ε arc cost of C^*

Informed Search

Greedy: selects lowest heuristic node. same as UCS, but forward cost instead of backward

- complete: no
- optimal: no, bad heuristic

A*: selects lowest estimated total cost to goal. PQ using back & forward cost for priority

complete: yes   optimal: yes (if consistent & admissible)

For A* optimal: tree search admissible, graph || consistent.

Tree Search: will visit a state of the underlying problem graph multiple times

Graph Search

add nodes to set, never expand a state twice

Games

deterministic: actions determined

stochastic: actions random

minimax: $O(b^m)$

Alpha-Beta Pruning

Time Comp: $O(b^{m/2})$

$\alpha: -\infty$ max $\beta: \infty$ min

Forward chaining proof

i. From (9) {w/0, y/B} infer that...

From (1) & ... infer that...

Backward Chaining proof

1. Goal $H \le B+T$. From (8), two new subgoals

a) Goal: $H \le y$

b) Goal: $H + 0 \le B+T$

CSP.

consists of variables, domains, constraints

N var. domain size d → $O(d^N)$ assignment

unary: only on 1 variable

binary: constraint b/t two var.

Backtracking search:

- fix var order & assign vals
- don't assign previously assigned values
- if no value, backtrack

Forward checking

- after assigning a value to a var, prune the domains of unassigned variables that share constraint w/ it.

arc consistency

$X \to Y$ is consistent iff for every X in the tail there's some Y which could be assigned w/o violating a constraint.

If X loses a value, neighbors of X need to be rechecked

- detects failure earlier than forward checking

$O(n^2 d^3)$ can be reduced to $O(n^2 d^2)$

MRV: chose variable w/ fewest valid remaining values

LCV: select value that prunes the least values from domains of unassigned vars

k-consistency

- guarantees for any set of k nodes in CSP, an assignment to any k-1 nodes guarantees k^{th} node will have one consistent value

$k=2 \equiv$ arc consistency

higher k → harder to compute

Tree-structured

- can be solved in $O(nd^2)$ from $O(d^n)$
- backward pass arc consistency
- forward pass assignment w/o backtrack

cutset conditioning

find smallest subset of vars in constraint graph w/ removal makes a tree.

$O((n-c)d^2)$ (remove c arcs)

$O(d^c (n-c) d^2)$ total time

Local search (iterative improvement)

- random assign vars
- selects var that violates most constraints and reset to value that violates the least constraints
- not optimal or complete

Logic

$\alpha \vDash \beta$ iff in every world where α is true, β is also true

α β

To prove: model checking

Theorem proving
- search for a sequence of proof steps leading from α to β

$\alpha \Rightarrow \beta \equiv \neg\beta \Rightarrow \neg\alpha$

$\alpha \Rightarrow \beta \equiv \neg\alpha \vee \beta$

$\alpha \Leftrightarrow \beta \equiv \alpha \Rightarrow \beta \wedge \beta \Rightarrow \alpha$

$\neg(\alpha \wedge \beta) \equiv \neg\alpha \vee \neg\beta$

$\neg(\alpha \vee \beta) \equiv \neg\alpha \wedge \neg\beta$

$\alpha \wedge (\beta \vee \gamma) \equiv (\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$

$\alpha \vee (\beta \wedge \gamma) \equiv (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$

DPLL

1. Early Termination: return as soon as satisfied
2. Pure symbols: only shows up positive or neg. can be assigned immediately
3. Unit clause: clause w/ just one literal or a disjunction w/ one literal and many falses.

If KB contains A and $A \Rightarrow B$, we can infer B

If KB contains $A \wedge B$ we can infer A or B

If KB contains A and B we can infer $A \wedge B$

Probability

Chain: $P(A|B) = \frac{P(A)P(B|A)}{P(B)}$

Joint: $P(A,B) = P(A)P(B|A)$

Query var Evidence

$P(Q_1 \ldots Q_{12} | e_1, \ldots e_{12})$ Hidden vars

Inference by enumeration (compute everything)

Bayes Net

- acyclic graph of nodes   node Parents
- cond. dist. for each node $P(X|A_1, \ldots A_n)$
- can calculate any prob using chain rule

$P(X_1, x_2, \ldots X_n) = \prod_{i=1}^{n} P(x_i | parents(x_i))$

Inference

Variable elimination

1. Join factors involving X
2. Sum out X

factors are unnormalized

Sampling

Rejection Sampling

reject samples that don't match evidence

Likelihood weighting

set evidence var and weight = 1

multiply weight by sampled variables that affect evidence variables

- $P(T|+c,+e)$, set $w_j = 1$
  - sample t
  - $w_j = w_j \cdot P(+c|t) \ldots$ Probability of seeing the evidence

evidence w/ k leaf nodes, weights $O(2^{-k})$

Gibbs

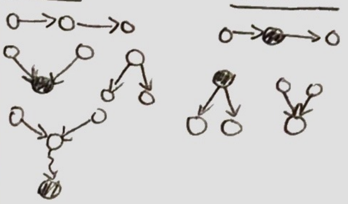
- set evidence, then random assign rest
- pick one, clear and resample w/ assigned vars
  - resample $P(-a|+c,+b)$

X from $P(X|markov\_blanket(x))$

D-separation

1. shade observed
2. Enumerate path X–Y
3. If all triples on path active, active
4. If no active path, X, Y are cond. indep.

Active Inactive



Laplace Smoothing

$P_{MLE}(x) = \frac{count(x)}{N}$

$P_{LAP,k}(x) = \frac{count(x)+k}{N+k|X|}$

$P_{LAP,k}(x|y) = \frac{count(x,y)+k}{count(y)+k|X|}$

$P_{LAP,\infty}(x) = \frac{1}{|X|}$

Decision Networks

$EU(a|e) = \sum_{x_1,\ldots x_n} P(X_1, \ldots X_n|e)\, U(a, X_1, \ldots, X_n)$

highest utility is MEU

Ex: $EU(leave|bad) = \sum_w P(w|bad)\, U(leave, w)$

$EU(take|bad) = \sum_w (w|bad)\, U(take, w)$

$MEU(F=bad) = \max_a EU(a|bad)$

VPI → how much max expected utility gained from info

$VPI(E'|e) = MEU(e,E') - MEU(e)$

$= \sum_{e'} P(e'|e)\, MEU(e,e')$

$VPI(F) = MEU(F) - MEU(\emptyset)$

Properties of VPI

nonnegativity: $\forall E', e \;\; VPI(E'|e) \geq 0$

nonadditivity: $VPI(E_j, E_k|e) \neq VPI(E_j|e) + VPI(E_k|e)$

If we know both gives more info than just one

$VPI(B,C) > VPI(B) + VPI(C)$

If B=C then $VPI(B,C) < VPI(B) + VPI(C)$

one gives just as much info as other

Order-Independence

$VPI(E_j, E_k|e) = VPI(E_j|e) + VPI(E_k|e, E_j) = VPI(E_k|e) + VPI(E_j|e, E_k)$

HMM

Mini Forward algorithm

$Pr(W_{i+1}) = \sum_{w_i} Pr(W_i, W_{i+1})$

$= \sum_{w_i} Pr(w_{i+1}|w_i)\, Pr(w_i)$

Stationary (same as mini forward)

$P(w_{t+1}) = P(w_t)$

$= \sum_{w_t} P(w_{t+1}|w_t)\, P(w_t)$

$P(W_{\infty+1}) = P(W_\infty)$

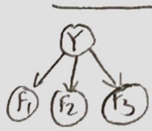
Vertibi Alg

$m_t[x_t] = \max_{x_{1:t-1}} P(x_{1:t}, e_{1:t})$

$= \max_{x_{1:t-1}} P(e_t|x_t) P(x_t|x_{t-1}) \max_{x_{1:t-2}} P(x_{1:t-1}, e_{1:t-1})$

$= P(e_t|x_t) \max_{x_{t-1}} P(x_t|x_{t-1})\, m_{t-1}[x_{t-1}]$

Naive Bayes



$prediction(f_1, \ldots, f_n) = \arg\max_y P(Y=y|F_1=f_1, \ldots F_N=f_n)$

$= \arg\max_y P(Y=y) \prod_{i=1}^{n} P(F_i=f_i|Y=y)$

$P(Y, F_1=f_1, \ldots, F_N=f_n) = \begin{bmatrix} P(Y=y_1, F_1=f_1, \ldots F_N=f_n) \\ \vdots \\ P(Y=y_n, F_1=f_1, \ldots F_N=f_n) \end{bmatrix}$

cond. indep. evidence

Likelihood (i.i.d)

$\mathcal{L}(\theta) = P_\theta(X_1, \ldots X_n)$

$\mathcal{L}(\theta) = \prod_{i=1}^{N} P_\theta(X_i)$

$\frac{d}{d\theta} \mathcal{L}(\theta) = 0$ (gradient)

$\log(\mathcal{L}(\theta)) = \log\left(\prod_{i=1}^{N} P_\theta(X_i)\right)$

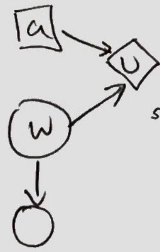
if no theta, prob just a factor so can be taken out of derivative

Prior Sampling

$S_{PS}(x_1, \ldots x_n) = \prod_i P(x_i | Parents(X_i))$

$= P(X_1, \ldots, X_n)$

sample procedure consistent



similar to Bayes Net

Markov Model

- Transition model $P(X_t|X_{t-1})$
- Stationary Assumption = transition probs are same
- X_{t+1} is indep. $X_0, \ldots, X_{t-1}$ given X_t
- $P(X_0, \ldots, X_T) = P(X_0) \prod_t P(X_t|X_{t-1})$

Markov Blanket: consist of a variable's parents, children, children other parents

Belief dist.

$B(W_i) = P(W_i|f_1, \ldots, f_i)$

belief ≡ Prob W_i given evidence

$= P(W_i|f_{1:i})$

$B'(W_i) = P(W_i|f_1, \ldots, f_{i-1})$ evidence up to $i-1$

$= P(W_i|f_{1:i-1})$

Forward Alg

$B'(W_{i+1}) = \sum_{w_i} P(W_{i+1}|W_i) B(W_i)$

$B(W_{i+1}) \propto Pr(f_{i+1}|W_{i+1}) B'(W_{i+1})$

$B(W_{t+1}) \propto Pr(f_{i+1}|W_{i+1}) \sum_{w_i} Pr(W_{i+1}|W_i) B(W_i)$

Time elapse: determine $B'(W_{i+1})$ from $B(W_i)$

Obs update: $B(W_{i+1})$ from $B'(W_{i+1})$

HMM

Initial distribution $P(X_0)$

Transition model: $P(X_t|X_{t-1})$

Sensor model: $P(E_t|X_t)$

Joint = $P(X_0, X_1, \ldots, X_T, E_T)$

$= P(X_0) \prod_{t=1:T} P(X_t|X_{t-1}) P(E_t|X_t)$

$F_1 \perp\!\!\!\perp W_0 | W_1$

$\forall i = 2, \ldots, n \;\; W_i \perp\!\!\!\perp \{W_0, \ldots, W_{i-2}, F_1, \ldots F_{i-1}\} | W_{i-1}$

$\forall i = 2, \ldots, n; \; F_i \perp\!\!\!\perp \{W_0, \ldots W_{i-1}, F_1, \ldots, F_{i-1}\} | W_i$

F_i are observations

Max likelihood geom dist

$P(F_i = 1 | Y = ham)$ corresponds to counting the # of ham emails in which word i appears and dividing it by the total # of ham emails

Particle simulation

observe weights

resample: sample weights by using weights to create distributions for samples.

Assign particle per new distribution

Time-elapse update: based off current particle assignment, use prob of next timestep as distribution and assign based off a new sample

Observation update: 1. calculate weight of each particle given new evidence

2. calc total weight for each state (if needed)
3. if sums to 0, reinitialize
4. else normalize and resample particles as above

Each particle is moved by sampling its next position from the transition model

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

Fix observation, similar to likelihood weighting

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

$f(x) \propto P(e|X) f'(X)$

Resample: rather than tracking weighted samples, we resample N times, choose from our weighted sample distribution equivalent to renormalizing the distribution

Perceptron

activation $w(x) = \sum w_i f_i(x) = w^T \cdot f(x)$

$$\text{classify}(x): \begin{cases} + & \text{if activation}(x) > 0 \\ - & \text{if activation}(x) < 0 \end{cases}$$

Binary Perceptron Alg

1. Init all weights to 0
2. For each train sample, $f(x)$ and true class label $y^* \in \{-1, +1\}$ do:

$$y = \text{classify}(x) = \begin{cases} +1 & \text{if act.} = w^T f(x) > 0 \\ -1 & \text{if act.} = w^T f(x) < 0 \end{cases}$$

compare predicted label to true label

if $y = y^*$, do nothing

else update

$w \leftarrow w + y^* \cdot f(x)$

Multi-class Perceptron

$W_0 = [-2\ 2\ 1]$ $f(x) = [-2\ 3\ 1]$

$W_1 = [0\ 3\ 4]$

$W_2 = [1\ .4\ -2]$ $s_0 = 11$ $s_1 = 13$ $s_2 = 8$

would predict s_1

- if s_1 not correct, subtract feature from s_1
- add feature to correct weight

POMDPS

State S, Actions A, Transition func $P(s'|s,a)$, Rewards R, Observations O, obs func $P(o|s)$ (or $O(s,o)$)

Gradient Ascent

Init w α = learning rate

for iter = 1, 2, ...

$w \leftarrow w + \alpha * \nabla g(w)$

- for descent

Sigmoid func

$$g(z) = \frac{1}{1+e^{-z}}$$

$$g'(z) = g(z)(1 - g(z))$$

Hyperbolic Tangent

$$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

$$g'(z) = 1 - g(z)^2$$

ReLU

$$g(z) = \max(0, z)$$

$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

$f(x) = g(h(x))$ $f'(x) = g'(h(x))\, h'(x)$

Decision Tree

Split on node w/ most info gained

↳ all positive or all negative