Probability Basics

Uncertainty may come from partial observability, noisy sensors, complexity of modeling and lack of (true) world dynamics

Probability summarizes the effects of laziness (don't list all) and ignorance

Bayesian probability relates probability to one's own state of knowledge

The set Ω ("sample space") defines worlds

Events are subsets of Ω, probability is the sum over all possible worlds $\sum_{w \in A} p(w)$

$0 \le P(w) \le 1$ $\quad \sum_{w \in \Omega} P(w) = 1$

Given joint distribution, one can marginalize "sum out" over variables, partial assignment, $P(a|b) = \frac{P(a,b)}{P(b)}$ conditional probabilities allow us to use evidence

Inference by Enumeration

Evidence variables: $E_1 \dots E_k = e_1 \dots e_k$

Query var: Q Hidden vars: $H_1 \dots H_r$

Want: $P(Q|e_1 \dots e_k)$

① select entries consistent with evidence

② Sum out H to get joint of Q and E

③ Normalize → $P(Q|e_1 \dots e_k) = \frac{1}{Z} P(Q, e_1 \dots e_k)$

$Z = \sum_q P(Q, e_1 \dots e_k)$

Worst case time/space: $O(d^n)$

Product Rule: $P(y)\,P(x|y) = P(x,y)$

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

Bayes' Rule: $P(x|y) = \frac{P(y|x)}{P(y)} P(x)$

Bayes Nets and Bayesian Inference

Models describe how world works, accord'd blowup of joint distributions by independence

Independence

Absolute if $\forall x, y \; P(x,y) = P(x)P(y)$

Conditional if $\forall x, y, z \; P(x|y,z) = P(x|z)$ (x and of y|z)

Conditional independence represented by Bayes Nets

Bayesian Networks

Helps factor joint distribution into product of conditional

Reduces table size from $O(d^n)$ to $O(n \cdot d^k)$

Syntax: set of variable nodes X_i, DAG, conditional probability table for each node given parents

Encode joint distribution as $P(x_1, \dots, x_n) = \prod_i P(x_i | \text{Parents}(X_i))$

Variables independent of non-descendants given parents

Markov blanket includes parents, children and children's parents and all variables independent of others given Markov blanket

Inference by Enumeration

"Summing out" hidden variables from joint table → exponentially many products

Variable elimination

Move summation expressions inwards as possible and perform calculation from inside out using factors

① Compute the pointwise product of factors

② Sum out variables from the factor

Pick a query variable, evidence, start w/ initial factors, while H left (pick H, join, sum) and join + normalize

Time and space complexity determined by largest factor

Ordering can affect size of largest factor, not always

Polytrees - DAG with no undirected cycles

Variable elimination linear $O(n)$ if start VE from leaves

Human utilities, Money and the Optimizer's Curse

Given $E[U(L)] = pX + (1-p)Y$, $U(L) = p\,U(\$X) + (1-p)\,U(\$Y)$

Certainty equivalent $CE(L)$ = cash amount such that $CE(L) \sim L$

Insurance premium = $EMV(L) - CE(L)$

Given a set of estimates, we choose one value and with high probability the actual value is much less than V^* (winner's curse)

Assess utility by lottery "best possible prize" vs "worst catastrophe" to price A and adjust p, 1-p

Bayes Nets and Approximate Inference

Sampling allows us to get approximate answer fast with simple model, little memory

Prior Sampling

Generate samples from root onwards and pick relevant

Rejection Sampling

Given a query $P(a|e...)$, reject sample if not consistent otherwise no different from prior sampling

Likelihood Weighting

Fixes problem of unlikely evidence and rejection sampling

Instead, fix evidence, sample rest and weight by $P(e|\text{Parents})$

$\text{Sws}(z,e)\; w(z,e) = \prod_{i=1}^{m} P(z_i | \text{Parents}(z_i)) \prod_{i=1}^{n} P(e_i | \text{Parents}(e_i))$

Upstream vals not influenced by downstream evidence ∴

Gibbs Sampling

Markov chain Monte Carlo, random walk w/ sampling and averaging

Fix evidence variables, pick variable each time and sample for a value conditioned on all others

Eventually you will sample from true posterior

Markov Models

Introduce notion of time by connecting state nodes

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

Stationarity assumption - transition P does not change

Markov assumption: X_t indp. of $X_0, \dots, X_{t-2}$ given X_{t-1}

is 1st order, kth order have k previous

Joint distribution → $P(X_0, \dots, X_T) = P(X_0) \prod_t P(X_t | X_{t-1})$

Forward Algorithm

At time t, $P(X_t) = \sum_{x_{t-1}} P(X_t | X_{t-1} = x_{t-1}) P(X_{t-1} = x_{t-1})$

Assuming stationary, can also multiply by transition matrix and $t \to t+x$

Stationary distribution can be reached, $P_\infty = P_{\infty+1} = TP_\infty$

Hidden Markov Models

Markov chain over states X, observe E at each time-step

Define by initial dist $(P(X_0))$, t's model $P(X_t|X_{t-1})$, sensor model $P(E_t|X_t)$

Joint distribution given by $P(X_0, \dots, X_T, E_T) = P(X_0) \prod_{t=1}^{T} P(X_t|X_{t-1}) P(E_t|X_t)$

Inference Tasks

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

Start with uniform to and use a predict/update cycle to keep track of current belief state

Filtering algorithm

We want a recursive alg of form $P(X_{t+1}|e_{1:t+1}) = g(e_{t+1}, P(X_t|e_{1:t}))$

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

Cost per time step: $C(|X|^2)$, time and space $O(1)$

Linear algebra: $f_{1:t+1} = \alpha O_{t+1} T^T f_{1:t}$, O has E_t of obs on diagonal

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

State trellis: states + transitions

Each arc is $x_{t-1} \to x_t$ with weight $P(x_t|x_{t-1})P(e_t|x_t)$

Viterbi algorithm

Find max probability of any path to a state t

$m_{1:t+1} = \text{Viterbi}(m_{1:t}, e_{t+1}) = P(e_{t+1}|X_{t+1}) \max_{x_t} (P(X_{t+1}|x_t)\, m_{1:t})$

Time: $O(|X|^2 T)$, Space: $O(|X|T)$, # paths: $O(|X|^T)$

Dynamic Bayes Nets

HMM with multiple vars

Sparse dependencies, exp fewer parameters

Exact Inference

Offline: Variable elimination $P(X_T|e_{1:T})$, unroll network for T steps

Online: Eliminate variable from prev time, store factors for current but largest factor contains all variables for current time

Particle Filtering

Likelihood weighting needs exponentially more samples with T, so initialize particles and resample the fittest

Start w/ N particles, $N \ll |X|$, $P(x)$ approximated by # particles

Propagate forward: $x_{t+1} \sim P(X_{t+1}|x_t)$, sample from prev t.m.

Observe: $w = P(e_t|x_{t+1})$, weight samples based on evidence

Resample: sample N times from weighted sample dist.

Decision Theory

A rational agent maximizes expected utility given knowledge

Worst-case minimax does not care about exact terminal value scale, optimal decision invariant under monotone trans.

Magnitudes are important for average-case expectimax

Utilities are functions from real outcomes to real numbers

Axioms of Rationality

Orderability: $(A \succ B) \vee (B \succ A) \vee (A \sim B)$

Transitivity: $(A \succ B) \wedge (B \succ C) \Rightarrow (A \succ C)$

Continuity: $A \succ B \succ C \Rightarrow \exists p\, [p, A; 1-p, C] \sim B$

Substitutability: $A \sim B \Rightarrow [p, A; 1-p, C] \sim [p, B; 1-p, C]$

Monotonicity: $A \succ B \Rightarrow (p \ge q \iff [p, A; 1-p, B] \succeq [q, A; 1-q, B])$

Decision Networks

Bayes Net + ▭ action nodes + ◇ utility nodes

Fix evidence e, for each action a: fix action node to a, compute Posterior $P(w|e,a)$ for wot U, compute $\sum_w P(w|e,a)\,U(a,w)$, return a w/ highest utility

Value of Information

Expected improvement in decision quality from observation of var

$VPI(E_i|e) = [\sum_{e_i} P(e_i|e) \max_a EU(a|e,e_i)] - \max_a EU(a|e)$

Non-negative: $VPI(E_i|e) \ge 0$, Not (usually) additive: $VPI(E_j, E_k|e) \ne VPI(E_j|e) + VPI(E_k|e)$

Order-independent: $VPI(E_j, E_k|e) = VPI(E_j|e) + VPI(E_k|e, E_j)$

Markov Decision Process

Definition: set of states $s \in S$, set of actions $a \in A$, transition model $T(s,a,s')$, reward fcn $R(s,a,s')$ or just $R(s)$

$P(s'|s,a)$

Start state, terminal state with zero rewards

Fully observable but probabilistic search problem

Policy $\pi^*: s \to A$ maximizes expected utility for all states

Utility $U[r_0, r_1, \dots r_\infty] = r_0 + \gamma r_1 + \gamma^2 r_2 \dots$ where reward bounded by $\pm R_{max}/(1-\gamma)$

Allows infinite streams or if terminal state reached w.p. 1 (geom. series)

$\pi^*(s)$ - optimal action, state s $\quad V^*(s) = V^{\pi^*}(s)$ - value of state (optimality from s)

$Q^*(s,a)$ - expected util of (s,a) optimal where $V^*(s) = \max_a Q^*(s,a)$

Bellman Update Equations: $V_{t+1}(s) \leftarrow \max_a \sum_{s'} P(s'|a,s)[R(s,a,s') + \gamma V_t(s')]$

Repeat until convergence or ε, B is a contraction by γ → exp fast conv.

Policy extraction: $\pi\, v(s) = \text{argmax}_a \sum_{s'} P(s'|a,s)[R(s,a,s') + \gamma V(s')]$

Policy iteration: $\pi_{i+1}(s) = \text{argmax}_a \sum_{s'} P(s'|a,s)[R(s,a,s') + \gamma V^{\pi_i}(s')]$

Reinforcement Learning

Maximize expected rewards based on observed sample transitions

Model-based Learning

① Learn empirical MDP by (counting) for s,a, namely $P(s'|s,a)$ and $R(s,a,s')$

② Solve learned MDP by value/policy iteration

Makes good use of experience but not scalable to large $|S|$

Direct-Utility Estimation

Use sum of discounted rewards to S and average over trials

No P/R required, optimal, but ignores state connections + longer to learn

Temporal Difference Learning

TD rule: $V^\pi(s) \leftarrow V^\pi(s)(1-\alpha) + \alpha[R(s,\pi(s),s') + \gamma V^\pi(s')]$

Approximate Bellman update, can't use value/improve policy w/ trans. model

Q-Learning

Q-learning gives us both optimal policy/value

$Q(s,a) \leftarrow (1-\alpha)\, Q(s,a) + \alpha[R(s,a,s') + \gamma \max_{a'} Q(s,a')]$

Converges as long as we explore enough and decay α appropriately: $\sum_t \alpha_t = \infty$ and $\sum_t \alpha_t^2 < \infty$ like $\alpha_t = O(1/t)$

Exploration vs Exploitation

Pure exploitation often gets stuck in not very optimal policy

ε-greedy: flip coin to try random action, make sure to decay ε

Exploration function: e.g. $f(u,n) = u + k/\sqrt{n}$ encourage exploration at 1st

Modified $Q \to Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha[R(s,a,s') + \gamma \max_{a'} f(Q(s',a'), n(s',a'))]$

Regret measures total cost of youthful errors, minimizing also encompasses optimal learning being optimal

Approximate Q-Learning

Describe state using vector of features for generality

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

update: $w_i \leftarrow w_i + \alpha[R(s,a,s') + \gamma \max_{a'} Q(s',a') - Q(s,a)] f_i(s,a)$ $\frac{\partial Q_w(s,a)}{\partial w_i}$

Linear fcns approx TD converge, Q diverge but nonlinear may work better using neural network instead of linear function

Machine Learning

Supervised Learning (correct answers) $f(x_i) = y_i$

Learn unknown target function f given training set of labeled (x_i, y_i) and output h (classification, regression → discrete, real-valued)

Tradeoff between degrees of fit and complexity

Decision Trees

Partitions input space, assign label to each group

Increasing hypothesis language expressiveness → more consist. hyp, bad at h/m

Choose attributes by # parents/ & entropy for info about classification

$B(p) = H(<p, 1-p>) = \sum_i p_i \log p_i$, # bits to classify new example

Information gain: $B(p/(p+n)) - \sum_k ((p_k+n_k)/(p+n)) B(p_k/(p_k+n_k))$

Generate hyp on training set, tie by validation, accuracy on test set

Accuracy: fraction correct, learning curve: accuracy vs training size

Neural Networks

Input = $\sum_i w_{i,j} a_i$ by a_j, $w_{i,j}$ (broadcasts) output: $a_j = g(in_j) = g(\sum_i w_{i,j} a_i) = g(w_j \cdot a)$

Activation functions g: threshold$(x) = 1$, sigmoid$(x) = 1/(1+e^{-x})$

Threshold perceptron as linear separator, $g(w \cdot x) = 1$ if $w \cdot x > 0$

Perceptron learning rule: $w_i \leftarrow w_i + \alpha(y - h_w(x))\,x_i$

Linearly separable if $\exists$ hyperplane, converge to min-error if α ↓ well

Multilayer perception - feed forward neural network $\exists \ge 1$ hidden, sigmoid

Loss = $\sum_i (y_i - h_w(x_i))^2$, $w \leftarrow w - \alpha \nabla_w$ Loss by backpropagation (output → hidden)

Using enough hidden nodes can represent any function

Statistical Learning

Choices: hypothesis space H, degree of fit, degree of fcn complexity, finding good h: knowing if h will predict well

Classical stats/ML uses linear feature combs, loss functions, regularization and complexity penalties, optimization/discrete search, testing

Maximum Likelihood Estimation

$P(y|x,h)$, Data: $(x_1, y_1), \dots (x_N, y_N)$, $\text{argmax}_h P(h)\, \pi_i P(y_i|x_i,h)$, x_i/y_i

optimization/discrete search, empirical process theory → pitfall

Evidence: $x = x_1, \dots, x_N$ Likelihood $P(x_1, \dots, x_N|\theta)$, Max likelihood by solving for $\partial L = \log P(x|\theta)/\partial\theta = 0$ to get θ_{ML}

Laplace smoothing with α: $\theta_{Lap} = (N_{ML} + \alpha)/(N + k\alpha)$

Naive Bayes

Polytree for $P(y|x) = \alpha P(y) \pi_i P(x_i|y)$, $P(x_i|y) = \sum_k P(x_{ik}|y) P(x_{ik}|x)$

Linear separator

Estimate $P(c_i)$ by ML, $P(w_i|c)$ w/ some Laplace smoothing

Agents and Rationality

Agent acts, operates autonomously, perceives environment, persist, adapts
Rational agent acts to increase (expected) outcome
Agents perceive environment by sensors and acts by actuators
Agent function is abstract mathematical concept; program implements (concrete)
Performance measure determines rational behavior, should design with desired environment in mind (designer)
Agent should learn, autonomy measures reliance on prior knowledge

Environments

Task environment: performance measure, environment, actuators, sensors
Fully/partially observable - sensors give full/partial knowledge of env
Single/multi-agent - other agent may be object → agents/objects
Deterministic/stochastic - next state probabilities may (not) be important, distinguish between part chos; deterministic vs stochastic
Episodic/sequential - current decision does (not) affect later decision
Static/Dynamic - env changing while agent deliberates, semi-past
Discrete/continuous - time handling; chess vs. real-time taxi
known/unknown - known if outcome/prob known, else unknown

Types of Agents

Simple Reflex Agents
Base action off current percept (no history), ↓ table (pressure Partly thus)
Infinite loops are possible; can randomize actions
Model-based Agents
Maintain internal state based on percept history
Use model, update state from perception to "best guess" world state
Goal-based Agents
Make decision based on state as well as goal
More flexible than reflex-based, goal-oriented decisions
Utility-based Agents
Generalization of goal-based agents for optimality
Rational can make tradeoffs to maximize expected utility

Search Problems

Environment is observable, discrete, known and deterministic
"Formulate, search, execute" - offline search ↓ to obs in execution
Formulation: ① Initial state ② Actions ③ Transition model $\{(s,a,s')\}$ state space
Can form graph ↓ states connected by links = actions
④ Goal Test (check s=goal state) ⑤ Path cost function $c(s,a,s')$
Optimal solutions entail lowest path cost, Tree-search + explored → graph search
In each node in search tree, remember n.state, n.parent, n.action, n.path-cost
Completeness measures find solution if exists

Uninformed Search Strategies

Breadth-first Search (queue as fringe)
Goal test applied when node generated, finds shallowest node d
Time/Space $\in O(b^d)$, dominated by frontier size
Uniform-cost Search (priority queue as fringe)
Expands node ↓ lowest path cost $g(n)$, optimal path when expand
Complete if every step > small, use ε Time/space $\in O(b^{1+\lfloor C^*/\varepsilon \rfloor})$
Depth-first Search (stack as fringe)
Graph search in finite state space complete while tree search not
Time/Space $\in O(b^m)$, $O(bm)$ → good for memory (BFS-like)
Depth-Limited Search - fix depth limit, Iterative Deepening - keep expand
Iterative Lengthening ↑ path costs, Bidirectional Search (BFS/UCS) → $b^{d/2}+b^{d/2} < b^d$

Informed (Heuristic) Search

Make use of combination of path cost $g(n)$ and est rem cost $h(n)$
Greedy-Best-First Search
$f(n) = h(n)$, tree search infinite loop, graph-search is complete
A* Search
$f(n) = g(n) + h(n)$, complete and optimal if $h(n)$ admissible
for tree search and if consistent for graph search
Node expanded ⇒ optimal path found, consistent ≡ $h(n) \le c(n,a,n') + h(n')$
Expands nodes along goal contour
Iterative Deepening using $f(n)$, Reuse best-first backtrack to makes state
Heuristic Functions

Local Search

Evaluate and modify current state; path cost is not relevant
Use small amounts of memory; can usually find reasonable sol in large/infinite spaces
complete - always finds solution, optimal - always finds global min/max
Hill-Climbing Search
Steepest-ascent/greedy local search; complete state formulation
② local maxima - stuck ③ ridges - seq of local maxima ④ plateau - flat area, shoulder
Stochastic - random from uphill moves with weighted probability
Simulated Annealing
Replace state by picking random move, and take it better
otherwise accept ↓ probability $\propto T$ which you gradually decrease
to shake the system out of local min/max
Local-beam search
Start ↓ k random states and generate all successors, keep top k
Parallel search threads can pass info around
Stochastic beam search - choose k successors at random & how "good"
Genetic Algorithm
Similar to stochastic beam search but generate successors by combining two parent states
Use objective function as fitness, cross over point and mutation

Agent Program Representations

Atomic "Black box"
Each state of world indivisible and has no internal structure
Factored Representation
Split state into variables and attributes, each has a value
Can use booleans and continuous to represent uncertainty
Structured Representation
Model how parts of the world are connected to each other

Learning Agents

Learning element makes improvements → performance element
Performance element makes decisions, learning uses critic feedback
Problem generator suggests to performance element → informative exp

Adversarial Search

Search problem: S_0 - Initial state, player(s) - which player has move
actions(s) - return set of legal moves, terminal test, Utility(s,p):
objective/payoff function → zero-sum - payoff same in all games

Minimax Search
Alternate min and max plies, accomplished by DFS to terminal
Multiplayer games may use vector $\langle a,b,c \rangle$ by policy

Alpha-Beta Pruning
As we search, keep values α (best max), β (best min value)
and cut off exploration if value worse than β
May use transposition table of known states to cut down
Cut off search ↓ evaluation to make imperfect real-time decision
Can modify α-β to use eval beyond certain point
Good for quiescent positions, horizon is trickier

Searching with Nondeterministic Actions

Stochastic games → chance, expectiminimax
Percepts help in partially observable or nondeterministic environment
Contingency plan (strategy) - by all possible seq of percepts
Solutions look like trees (if-then-else), rather than sequences
AND-OR search trees alternate OR (you) and AND (env) options
and solution must be able to handle ≥1 OR and all (1+N) AND
some nondeterministic environments may require optimal sol

Searching with Partial Observations

Belief state is set of all possible physical states you may be in
No observation
Belief states: Up to 2^N unique sets of belief states ≡ belief-state space
Initial state: set of all states in P initially
Actions: If illegal actions have no effect, then
$\bigcup_{s \in b}$ Actions$_P(s)$, else $\bigcap_{s \in b}$ Actions$_P(s)$ ⇒ safer
Transition Model: deterministic ⇒ $b' = \text{Result}(b,a) = \{s' : s' = \text{result}_P(s,a) \text{ and } s \in b\}$
Goal Test: Check that all belief states satisfy else $\bigcup_{s \in b}$ Results$_P(s,a)$
Path cost may be different between states
with Observation
Prediction: $\hat{b} = \text{Predict}(b,a)$
Observation prediction: determine set of possible percepts observable
Update: Partitions and returns set of belief states that
correspond to intake percept $o \Rightarrow b_o = \text{update}(\hat{b}, o)$
Nondeterminism just enlarges the number of belief states

Constraint Satisfaction Problems

Utilize a factored representation of variable assignment to
simultaneously satisfy all constraints
X - set of variables $\{X_1, \ldots, X_n\}$, D - set of domains, one for each var
C - set of constraints ⇒ allowable combinations of values
constraint graphs - to visualize systems of binary constraints
global constraint - arbitrary # of variables
node consistency ≡ enforce unary, arc enforce binary
path consistency ≡ tuples $\{X_i, X_j\}$ ↓ X_m if $\{X_i, X_m\}, \{X_m, X_j\}$
solutions are commutative
Backtracking Search (DFS one var at time, assign, backtrack if fail) (flexible)
MRV - minimum remaining value
Degree - pressume items ↓ higher degree
LCV - least constraining value
Forward Checking - choose variable, establish arc consistency
from neighbors and continue, backtrack if no domain
MAC (Maintaining Arc Consistency) - call AC3 instead of assigning
from neighbors
Chronological backtracking + backjumping
min conflicts - pick random variable and min conflicts, repeat
Cutset Conditioning
Choose subset of graph to form a tree
Loop through cutset and run tree CSP solver $\in O(d^{2n})$
which ensures backward consistency then runs forward checking

Logical Agents

knowledge base ≡ set of sentences in knowledge representation language that describes the world
TELL and ASK operations both involve inference to derive new sentences from old

Logic

Entailment implies β follows logically from α, all worlds where α are true require β to be true
sound - can derive entailed sentences
complete - can derive any sentence that is entailed

Propositional Logic
$\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$
Truth tables allow you to express truth/result sentences
Model-Checking, Theorem-Proving
Valid - True in all models, satisfiable - true in some model
Prove β from α by unsatisfiability of $\alpha \wedge \neg\beta$ (contradiction)
Initial state: knowledge base
Actions: All inference rules applied to sentences matching up top half of inference rule
Result: Add sentence in bottom half
Goal: State we are trying to prove
Conjunctive normal form - $\wedge$'s of $\vee$'s
Horn form - at most one positive literal in $\vee$
DPLL - pure symbol (same sign), unit clause → early termination