Intelligent Agents

Agent: anything that can perceive its environment and act upon it
Percept sequences: the history of perception of an agent
Agent function: what maps percept sequences into actions
Tabulating: putting inputs and outputs of an agent into a table

Types of Searches

Uninformed Search

Depth First: Keep going until cannot, then swap children
Breadth First: Expand all children of the closest node
Best-first: Expand the node closest to the result
Uniform-cost: Expand closest node accounting for node distance
Depth limited: Depth first up to a limit
Bidirectional: Start from beginning and end and meet in the middle

Informed Search

Greedy Best-first: Expand the closest node
A*: Expand the smallest heuristic + distance to start
Weighted A*: Multiply heuristic by a constant to make A *more greedy*
Beam*: best first, but only keep the top k nodes*
IDA**: A* but only add nodes to frontier if below the threshold
SMA*: A* but keep top nodes until memory runs out, then prune

CSP

Constraints

Node Consistency: all unary constraints are satisfied
Arc Consistency: all binary constraints can be satisfied with current possible values
Path Consistency: all binary constraints can be satisfied upon triplets of nodes

Searches

Backtracking

Backtracking: Depth first for CSPs, when fail, backtrack to okay state
MRV: Pick the first variable to have the smallest state space, to reduce the tree quickly
Least constraining value: Choose values that restrain the least to improve chances of finding a solution
Forward checking: remove all arc-inconsistent values after any assignment

Local

1. Start with random assignments
2. Pick a variable, modify its value to reduce collisions

Tabu: keep list of failed states to avoid revisitation
Weighting: weight certain constraints more so they don't de-solve

Adversarial Search

Minimax

Minimax: picks a move that has the best worst possible outcome assuming an optimal opponent
Alpha-Beta pruning: removing nodes that have less than optimal or more than possible heuristics assuming both players play optimally
Horizon effect: imminent loss is continually delayed as algorithm cannot see so far in advance

Monte Carlo TS

MCTS: Selects, expands, simulates, then back propagates nodes to choose a move
Selection Policy: A function to create moves that explore or refine
Playout Policy: A function to deterministically choose moves for both sides to simulate the end game of a node
UCBI: Selection Policy that ranks moves based on certainty and win-rate

Uncertainty

Bayes Rule:
$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$
$\text{Posterior} = \frac{\text{Likelihood}\cdot\text{Prior}}{\text{Marginal}}$
Naive Bayes Rule: $P(X_1, \ldots, X_n) = P(X_1) \prod_{i=2}^{n} P(X_i|X_1)$ (Assuming independence across variables)
Bayes net state probability: $P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i|\text{parents}(X_i))$
False Negative: Heuristic falsely reports false

Clustering and Neural Networks

k-means update Algorithm: $\frac{\sum_{n=1}^{N} r_{n,k}\vec{x}_n}{\sum_{n=1}^{N} r_{n,k}} = \vec{\mu}_k$
MSE: $\sum_{n=1}^{N} (\vec{o}_n - \vec{c}_n)^2$
NN Forward Pass: $f(\vec{b} + W@\vec{x})$
NN Backpropogation: $f'(\vec{b} + W@\vec{x}) \cdot \vec{x}$
Sigmoid: $\frac{1}{1+e^{-x}}$
Sigmoid': $\sigma(1-\sigma)$

Sequential Data

Where X is the state and E is the effect
Transition: $P(X_{t+1}|X_t)$
Effect: $P(E_t|X_t)$

Filtering: $P(X_t|e_{1:t})$
Prediction: $P(X_{t+k}|e_{1:k})$
Smoothing: $P(X_k|e_{1:t})$
Most likely Solution: $\text{argmax}_{X_1:t} P(X_{1:t}|e_{1:t}))$