**PEAS Framework**

**Performance Measure**

Correctness – Accuracy

Efficiency – Speed / Resources

**Environment**

Fully / Partially Observable

Deterministic / Stochastic / Strategic

(Fixed / Random / Other Agents Strategic)

Episodic / Sequential

(Independent States / Intertwined States)

Static / Dynamic

(Environment Unchanged / Changing)

Discrete / Continuous

(Distinct Actions / Range of Actions)

Single / Multi-Agent

(Agent Operating Alone / More Agents)

**Actuators**

Outputs from system (e.g. display / sound)

**Sensors**

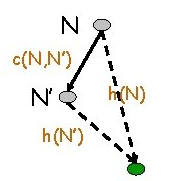
Inputs to system (e.g. keyboard / mouse)

**Heuristics**

**Admissible**

Heuristic always gives estimate lower than or equal to the actual cost of reaching the goal from current node/state n.

**Consistent**



h(N) ≤ c(N,N’) + h(N’)

i.e. always finds lowest possible cost path between every pair of nodes.

(Consistent is **ALWAYS** admissible)

**Dominant**

h1 ≥ h2 for all n and both admissible, h1 dominant

Most dominant one expands least nodes (cheaper)

**Searching**

b: branching factor

d: distance from start node to goal node

m: maximum depth of state space

C: cost of optimal solution

ε: cost per step

**BFS**

Complete: if space is finite

Time: O(b­­d+1) Space: O(bd+1)

Optimal: yes if cost is non-decreasing func of depth

**DFS**

Complete: no (loops)

Time: O(bm) Space: O(bm)

Optimal: no

Alternate: Depth Limited Search

- DFS with depth limit

**Uniform Cost Search**

Pop lowest priority node from PQ, enqueue its children until goal node found

Complete: if step cost ≥ ε

Time: O(bceil(C / ε)) Space: O(bceil(C / ε))

Optimal: yes

**Iterative-Deepening Search**

DFS to increasing depth limit repeatedly

Complete: yes

Time: O(bd) Space: O(d)

Optimal: yes if cost is non-decreasing func of depth

**Best-First Search**

Order frontier nodes: increasing distance from goal

Expand whichever node is closer to goal state

Complete: no (if loops present)

Time: O(bm) Space: O(bm)

Optimal: no

**A\* Search**

Expand next closest node, stop when goal node is next closest node

Complete: yes

Time: O(bd) Space: O(bd)

Optimal: yes

**Hill-Climbing Search**

Improve state by making incremental changes

May get stuck at local maxima if only improving changes are accepted

Allow bad moves but slowly decrease frequency of bad moves to reach global maxima

Time: O(∞) Space: O(b)

Optimal: yes (if convex)

**Beam Search**

Perform multiple instances of Hill-Climbing Search in parallel

**Minimax**

Assume each side makes most optimal next move

Complete: yes (if finite)

Time: O(bm) Space: O(bm)

Optimal: yes (if opponent optimal)

Optimisation: α-β pruning

Keep track of values seen thus far and ignore

paths that will never be chosen

**Machine Learning (Feedback)**

**Supervised**

Correct answers given to learning algorithm

**Unsupervised**

No correct answers given for examples

**Weakly Supervised**

Correct answer given, but not complete/precise

e.g. “there is a face in this picture”

**Reinforcement**

Occasional rewards given if answer closer to right

**Supervised Learning**

We have an idea of relation between input/output

**Regression**

Predict results within continuous output, pass input variables to continuous function

**Classification**

Predict results in discrete output, sort input variables to discrete categories

**Unsupervised Learning**

No idea of relation between input/output

Deriving structure from data without feedback

Cluster data based on relationship among variables

**Linear/Polynomial Regression**

Linear Regression

hθ(x) = θ0 + θ1x1 + … + θnxn for n variables/features

hθ(xi) = θ0 + θixi for specific variable/feature

Mean normalisation:

Cost Function

Each time, vary θ using learning rate α

**Normal Equation**

Y: actual correct output

X: input to be mapped

θ = (XTX)-1XTY

- XTX must be invertible

**Polynomial Regression**

xik instead of just xi

θi is tied to xii+1

========== Midterms End ==========