CS440/ECE448 Lecture 28: Exam 2 Review

Mark Hasegawa-Johnson

Outline

- How to take the exam
- Topics
    - Configuration space, Search, A*, Theorem proving
    - MDP, Model-based RL, Q-learning, Policy learning, Reward hacking
    - Computer vision, Convolution
- Sample problems

How to take the exam

- Reserve a time at https://us.prairietest.com/
- Show up at the appointed time to take the exam
- There will be 8 multiple choice questions

Configuration space

- Workspace (e.g., $\boldsymbol{w} = [x, y]^T$) vs. Configuration space (e.g., $\boldsymbol{q} = [\theta_1, \theta_2]^T$)
- Path planning: shortest path in configuration space
  - First, map obstacles from workspace into configuration space
  - Visibility graph: states=vertices of obstacles in configuration space
  - Rapid Random Trees (RRT): states=random, resampled near the best path after every iteration

Search

- Depth-first search (DFS)
  - incomplete, inadmissible, non-optimal
  - Time complexity = $\mathcal{O}\{bm\}$, Space complexity = $\mathcal{O}\{b^m\}$
- Breadth-first search (BFS)
  - complete, inadmissible (unless each edge has cost 1), non-optimal
  - Time complexity = Space complexity = $\mathcal{O}\{b^d\}$
- Uniform-cost search (UCS)
  - complete, admissible, non-optimal
  - Time complexity = Space complexity = # nodes with $g(n) \leq g^*$

A* search

- Admissible heuristic: $\hat{h}(n) \leq h(n)$
- Consistent heuristic: $\hat{h}(n) - \hat{h}(m) \leq h(n,m)$
- $\hat{h}(n) = 0$ is a valid heuristic (equal to UCS), but usually we want to invent an $\hat{h}(n)$ as large as we can, subject to one of the two constraints above (depending on whether we want to re-open closed nodes).

Theorem proving

- Logic:
  - $\neg$ (not) , $\wedge$ (and) , $\vee$ (or), $\Longrightarrow$ (implies), $\Longleftrightarrow$ (equivalent)
  - First-Order Logic: $\exists x: F(x)$ (there exists), $\forall x: F(x)$
- Proving "there exists" theorems: find an x that satisfies the statement
- Variable normalization: each rule uses a different set of variable names
- Unification: Find a substitution $S: \left\{\mathcal{V}_P, \mathcal{V}_Q\right\} \rightarrow \{\mathcal{V}_Q, C\}$ such that $S(P) = S(Q) = U$, or prove that no such substitution exists
- Forward-chaining: Search problem in which each action is a unification, and the state is the set of all known true propositions
- Backward-chaining: Search problem in which each action is a unification, and the state is the goal (the proposition whose truth needs to be proven)

Markov Decision Processes

- Bellman equation:

$$u(s) = r(s) + \gamma \max_a \sum_{s'} P(S_{t+1} = s' | S_t = s, a) u(s')$$

- Value iteration:

$$u_i(s) = r(s) + \gamma \max_a \sum_{s'} P(S_{t+1} = s' | S_t = s, a) u_{i-1}(s')$$

- Policy iteration:

$$u_i(s) = r(s) + \gamma \sum_{s'} P(S_{t+1} = s' | S_t = s, \pi_i(s)) u_i(s')$$

$$\pi_{i+1}(s) = \operatorname*{argmax}_a \sum_{s'} P(S_{t+1} = s' | S_t = s, a) u_i(s')$$

Model-based reinforcement learning

- Model-based learning

$$P(s_{t+1}|s_t, a_t) = \frac{N(s_t, a_t, s_{t+1}) + k}{\sum_{s' \in \mathcal{S}} N(s_t, a_t, s') + k|\mathcal{S}|}$$

- Exploration vs. Exploitation
  - Epsilon-first: explore every action at least ϵ times
  - Epsilon-greedy: explore at random with probability ϵ

Summary: Q-learning

Q-learning:

$$q(s,a) = r(s) + \gamma \sum_{s'} P(s'|s,a)u(s')$$
$$u(s) = \max_{a \in \mathcal{A}} q(s,a)$$

TD-learning = Q-learning with smoothing

$$q_{\text{local}}(s_t, a_t, r_t, s_{t+1}) = r_t + \gamma \max_{a \in \mathcal{A}} q_t(s_{t+1}, a)$$
$$q(s_t, a_t) \leftarrow q(s_t, a_t) + \eta\left(q_{local}(s_t, a_t, r_t, s_{t+1}) - q(s_t, a_t)\right)$$

SARSA = on-policy Q-learning

$$q_{\text{local}}(s_t, a_t, r_t, s_{t+1}, a_{t+1}) = r_t + \gamma q_t(s_{t+1}, a_{t+1})$$
$$q(s_t, a_t) \leftarrow q(s_t, a_t) + \eta\left(q_{local}(s_t, a_t, r_t, s_{t+1}, a_{t+1}) - q(s_t, a_t)\right)$$

Policy learning

- Policy learning

$$\pi_a(s) = P(a \text{ is the action the agent will perform} | \text{state } s)$$

- Imitation learning: Given a database of teacher actions,

$$\mathcal{L} = -\log P(\mathcal{D}) = -\sum_{t=1}^{n} \log \pi_{a_t}(s_t)$$

- Actor-Critic:

$$\mathcal{L}_{critic} = \frac{1}{2}\left(q(s,a) - q_{local}\,(s,a)\,\right)^2, \qquad \mathcal{L}_{actor} = -\sum_a \pi_a(s) q(s,a)$$

- REINFORCE: Given a stored episode $\{(s_1, a_1), \cdots, (s_T, a_T), r\}$,

$$\boldsymbol{W} \leftarrow \boldsymbol{W} + \Delta \boldsymbol{W}, \qquad \Delta w_{ij} = \eta(r-b) \sum_{t=1}^{T} \frac{\partial \log \pi_{a_t}(s_t)}{\partial w_{ij}}$$

Image formation & processing

- Pinhole camera equations:

$$\frac{x'}{f} = -\frac{x}{z}, \qquad \frac{y'}{f} = -\frac{y}{z}$$

- Vanishing point = parallel lines:

$$x_1 = az + c_1, \qquad y_1 = bz + d_1$$
$$x_2 = az + c_2, \qquad y_2 = bz + d_2$$

- Edge detection using difference-of-Gaussians:

$$h(m,n) = \frac{1}{2\pi\sigma^2} e^{-\left(\left(\frac{m}{\sigma}\right)^2 + \left(\frac{n}{\sigma}\right)^2\right)}$$

$$h_x{}'(x', y') = \frac{(h(x'+1, y') - h(x'-1, y'))}{2}$$

$$h_y{}'(x', y') = \frac{(h(x', y'+1) - h(x', y'-1))}{2}$$

Convolution and Max Pooling

$$y[k,l] = w[k,l] * x[k,l] = \sum_i \sum_j x[k-i, l-j] w[i,j]$$

$$\frac{d\mathcal{L}}{dw[i,j]} = \sum_k \sum_l \frac{d\mathcal{L}}{dy[k,l]} \frac{dy[k,l]}{dw[i,j]}$$

$$z[m,n] = \max_{\substack{(m-1)p+1 \le k \le mp, \\ (n-1)p+1 \le l \le np}} y[k,l]$$

$$\frac{d\mathcal{L}}{dy[k,l]} = \begin{cases} \dfrac{d\mathcal{L}}{dz[m,n]} & \text{if } y[k,l] = \max\limits_{\substack{(m-1)p+1 \le i \le mp, \\ (n-1)p+1 \le j \le np}} y[i,j] \\ 0 & \text{otherwise} \end{cases}$$