CS 440/ECE 448 Lecture 27: Exam 3 Review

Outline

- Exam administration
- Exam topics
- Solution methods for posted practice exam

Exam administration

- Where & when: Lincoln Hall Theater, 1:00pm, Mon April 6
- How: Paper exam w/ attached formula sheet and scratch paper
- What to bring:
  - pencils & erasers or pens
  - One 8.5x11 sheet of handwritten notes, front & back
  - ID
- Don’t bring:
  - Calculators or computers
  - Textbook or printed notes

Grading

- Each TA will grade one problem part
- Partial credit will be possible, using a rubric designed by the TA
- Regrades will be possible only if points can be changed while remaining fair to all other students in the class

Recommended approach to studying

- Do every variant of every daily quiz at least once, lectures 12-25 (18-Feb through 30-Mar)
- Make sure you understand the equations behind MP5-10. The exam will not cover coding, but it will cover those equations.
- Then take the practice exam.
  - Treat the quizzes and MPs as your training set
  - Treat the practice exam as your dev set
  - Then you should have a pretty good idea how well you'll do on the test set

Outline

- Exam administration
- Exam topics
- Solution methods for posted practice exam

Topics included on the exam

- Lectures 12-25 (18-Feb through 30-Mar)

Topics included on the exam

- Multilayer networks & Natural language processing
  - Nonlinear regression, softmax
  - Transformer, theorem proving
- Computer Vision & xAI
  - Pinhole camera equations
  - Convolutional neural networks
  - Layer-wise relevance propagation
- Robotics & Search
  - Workspace and configuration space
  - BFS, DFS, UCS, A*
- MDP & RL
  - Bellman’s equation, Policy iteration, Value iteration
  - TD, SARSA, Actor-Critic, REINFORCE

Machine learning: Nonlinear Regression

- Piece-wise constant nonlinear regression:

$$f(\boldsymbol{x}) = \boldsymbol{w}^{(2),T}\sigma\left(\boldsymbol{W}^{(1)}\boldsymbol{x}\right)$$

- Piece-wise linear regression:

$$f(\boldsymbol{x}) = \boldsymbol{w}^{(2),T}\mathrm{ReLU}\left(\boldsymbol{W}^{(1)}\boldsymbol{x}\right)$$

- Back-propagation:

$$\boldsymbol{W}^{(1)} \leftarrow \boldsymbol{W}^{(1)} - \eta\frac{\partial \mathcal{L}_i}{\partial \boldsymbol{W}^{(1)}}$$

$$\boldsymbol{w}^{(2)} \leftarrow \boldsymbol{w}^{(2)} - \eta\frac{\partial \mathcal{L}_i}{\partial \boldsymbol{w}^{(2)}}$$

Machine learning: Softmax

- Sigmoid review

$$f(\boldsymbol{x}) = \sigma(\boldsymbol{x}^T \boldsymbol{w}), \qquad \boldsymbol{w} \leftarrow \boldsymbol{w} + \eta \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \big(y_i - f(\boldsymbol{x})\big)\boldsymbol{x}_i$$

- Multi-class linear classifiers

$$f(\boldsymbol{x}) = \operatorname{argmax} \boldsymbol{W}\boldsymbol{x}$$

- One-hot vectors

$$f_c(\boldsymbol{x}) = \mathbb{I}_{\operatorname{argmax} \boldsymbol{W}\boldsymbol{x}=c}$$

- Softmax nonlinearity

$$f_c(\boldsymbol{x}) = \frac{\exp(\boldsymbol{w}_c^T \boldsymbol{x})}{\sum_{k=1}^{v} \exp(\boldsymbol{w}_k^T \boldsymbol{x})}$$

- Derivative of the log softmax

$$\boldsymbol{w}_c \leftarrow \boldsymbol{w}_c + \eta \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \Big(\mathbb{1}_{y_i=c} - f_c(\boldsymbol{x})\Big)\boldsymbol{x}_i$$

Language processing: Transformer

- Multi-headed dot-product attention: $\boldsymbol{C}_j = \text{softmax}\left(\frac{\boldsymbol{Q}_j \boldsymbol{K}_j^T}{\sqrt{d}}\right)\boldsymbol{V}_j$
- Concatenate and transform: $\boldsymbol{O} = [\boldsymbol{C}_1, \ldots, \boldsymbol{C}_h]\boldsymbol{W}_O$
- Rotary Positional Encoding (RoPE):
$$\boldsymbol{x}_t \mathrel{+}= [\cos(\pi t/T), \sin(\pi t/T), \cos(2\pi t/T), \ldots]^T$$
- LayerNorm:
$$\mu = \frac{1}{nd}\sum_{i=1}^{n}\sum_{k=1}^{d} o_{i,k}\,, \qquad \sigma = \sqrt{\frac{1}{nd}\sum_{i=1}^{n}\sum_{k=1}^{d}\left(o_{i,k} - \mu\right)^2}$$
- Residual Connections: $\boldsymbol{X}^{(l)} = \boldsymbol{X}^{(l-1)} + \boldsymbol{O}^{(l-1)}$

Natural processing: Theorem proving

- 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: $\{\mathcal{V}_P, \mathcal{V}_Q\} \rightarrow \{\mathcal{V}_Q, C\}$ 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 set of propositions whose truth needs to be proven)

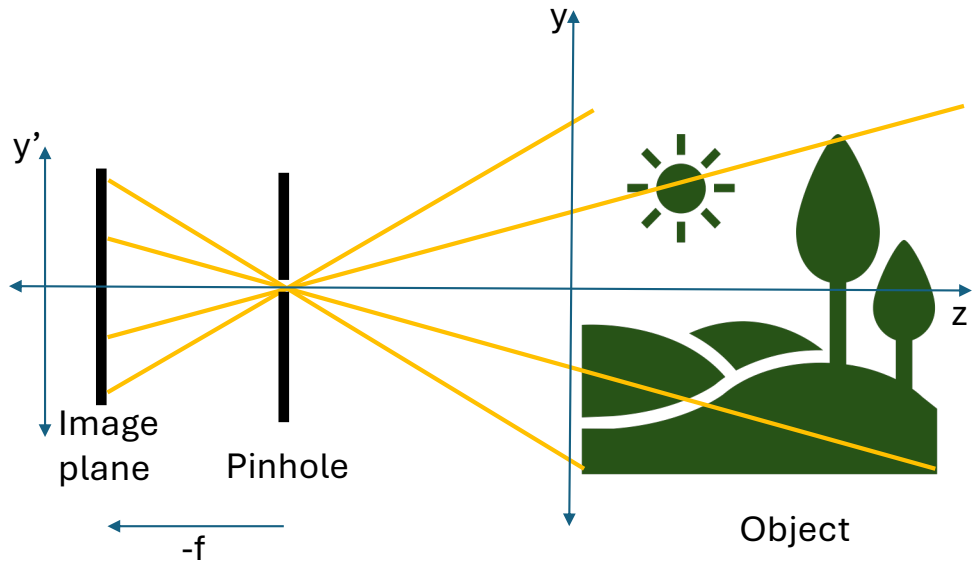
Computer vision: Pinhole camera

- These are similar triangles! So

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

- Solving for (x',y'), we get the pinhole camera equations:

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



Computer vision: Convolution and Max Pooling

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

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

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

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

Explainable AI: Relevance and Bayes nets

- Relevance of input x_d to output z_c:

$$R_{c,d} = \frac{\partial z_c}{\partial x_d} \cdot x_d$$

  - Layer-wise relevance propagation LRP: Normalize so $\sum_d R_{c,d} = R_c$
  - Gradient-weighted class activation mapping Grad-CAM: Keep only positive relevances

- Counter-Factual Reasoning:

$$\max_{F,X} |P(F, X, A = a) - P(F, X, A = \neg a)| > \epsilon ?$$

  - If the answer is yes, then the model says that F depends on A

Robotics

- Workspace vs. Configuration space
  - Forward kinematics: $\boldsymbol{w} = \varphi(\boldsymbol{b}, \boldsymbol{c})$
  - Inverse kinematics: $\mathcal{C}_{\text{obs}} = \{\boldsymbol{c}: \exists \boldsymbol{b}: \varphi(\boldsymbol{b}, \boldsymbol{c}) \in \mathcal{W}_{\text{obs}}\}$
- Path planning: What is the best path?
  - Minimize $\mathcal{C}$-space distance while avoiding obstacles
- Path planning: How do you find the best path?
  - Rectangular discretization
  - Visibility graph
  - Rapid Random Graph

Search: DFS, BFS, UCS

- Depth-first search (DFS)
  - incomplete, inadmissible, non-optimal
  - Space complexity = $\mathcal{O}\{bm\}$, Time 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 $g(n) \leq g^*$

Search: A*

- A* is admissible if you use an admissible heuristic, and optimal (lower computation than any other exact search algorithm!) if you use a consistent heuristic.
- Admissible: $\hat{h}(n) \leq h(n)$
- Consistent: $\hat{h}(n) - \hat{h}(m) \leq d(n, m)$
- $\hat{h}(n) = 0$ is a valid heuristic (Dijkstra's algorithm), 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 or not we want to re-open closed nodes).

Markov decision process

- Bellman equation: n nonlinear equations in n unknowns

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

- Policy iteration: converges quickly if your initial guess $\pi_1(s)$ is good

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

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

- Value iteration: converges exponentially quickly after at least one path reaches each reward

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

RL: Model-based 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

RL: 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

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

$$q(s_t, a_t) \leftarrow q(s_t, a_t) + \eta\big(q_{local}(s_t, a_t, r_t, s_{t+1}) - q(s_t, a_t)\big)$$

SARSA = on-policy Q-learning

$$q_{\text{local}}(s_t, a_t, r_t, s_{t+1}, a_{t+1}) = r_t + \gamma q_t(s_{t+1}, a_{t+1})$$

$$q(s_t, a_t) \leftarrow q(s_t, a_t) + \eta\big(q_{local}(s_t, a_t, r_t, s_{t+1}, a_{t+1}) - q(s_t, a_t)\big)$$

RL: 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 $\{(s_1, a_1), \cdots, (s_T, a_T), r\}$,

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

Outline

- Exam administration
- Exam topics
- Solution methods for posted practice exam

Question 1 ***(28 points)***

Consider a neural network with the following architecture. The input is a scalar x, and there is a weight vector $\mathbf{w} = [w_1, \ldots, w_n]^T$. The vector outputs $\mathbf{f}$, $\mathbf{g}$ and $\mathbf{h}$ are defined as

$$\begin{aligned} \mathbf{f} &= x\mathbf{w}, \\ \mathbf{g} &= \exp(\mathbf{f}), \\ \mathbf{h} &= \frac{\mathbf{g}}{\sum_{i=1}^{n} g_i}, \end{aligned}$$

where $\exp(\cdot)$ is applied element-wise to its argument.

(a) (14 points) Suppose that you know $\frac{\partial \mathscr{L}}{\partial h_j} = 0$ for all j except $j = 2$, and suppose that $\frac{\partial \mathscr{L}}{\partial h_2}$ is known. In terms of $\frac{\partial \mathscr{L}}{\partial h_2}$ and any of the elements of $\mathbf{f}$, $\mathbf{g}$, $\mathbf{h}$, $\mathbf{w}$, find $\frac{\partial \mathscr{L}}{\partial g_2}$.

(b) (14 points) Suppose that you know $\frac{\partial \mathscr{L}}{\partial g_j}$ for all j. In terms of $\frac{\partial \mathscr{L}}{\partial g_j}$, and in terms of x and/or any of the elements of $\mathbf{f}$, $\mathbf{g}$, $\mathbf{h}$, $\mathbf{w}$, find $\frac{\partial \mathscr{L}}{\partial x}$.

Solution method: Chain rule!

$$\frac{\partial \mathscr{L}}{\partial g_2} = \left(\frac{1}{\sum_{i=1}^{n} g_i} - \frac{g_2}{\left(\sum_{i=1}^{n} g_i\right)^2} \right) \frac{\partial \mathscr{L}}{\partial h_2}$$

$$\frac{\partial \mathscr{L}}{\partial x} = \sum_i \sum_j \frac{\partial \mathscr{L}}{\partial g_i} \frac{\partial g_i}{\partial f_j} \frac{\partial f_j}{\partial x} = \sum_j \frac{\partial \mathscr{L}}{\partial g_j} g_j w_j$$

Question 2 ***(15 points)***

Consider an unusual convolutional neural network with input image $x[i,j]$, weights $w[i]$, and output image $f[i,j]$ related as:

$$f[k,l] = \sum_i x[k-i, l-i]w[i]$$

In terms of the elements of x, w, and/or f, what is $\frac{\partial f[k,l]}{\partial x[i,j]}$?

Solution method: Variable substitution

$$i \leftarrow k-i, \qquad l-i \leftarrow l-k+i$$

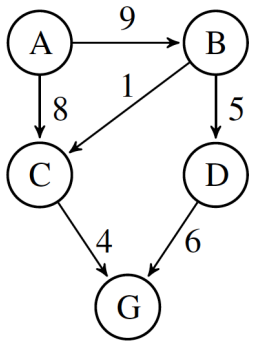
...so...

$$\sum x[k-i, l-i]w[i] = \sum x[i, l-k+i]w[k-i]$$

...so...

$$\frac{\partial w[k,l]}{\partial x[i,j]} = \begin{cases} w[k-i] & k-i = l-j \\ 0 & \text{otherwise} \end{cases}$$

Question 3 *(28 points)*

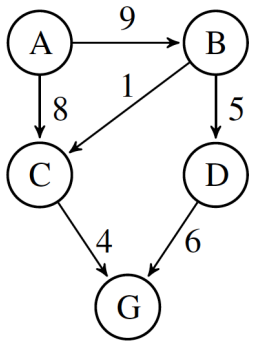


(a) (14 points) The search graph above starts at node A, and ends at node G. Create a table showing, in the first column, the node that is expanded by uniform cost search at each step of the search process (starting with A, ending with G), and in the second column, the set of nodes that are in the frontier after the node in the first column has been expanded. Optionally, you may list a priority next to each node in the frontier if you wish. Assume that, if a node has already been expanded, it need not be placed back in the frontier. Ties are broken in alphabetical order.

Solution method: Expand nodes in order of increasing cost g(n)

Expand	Frontier
A	B:9, C:8
C	B:9, G:12
B	G:12, D:14
G	D:14

Question 3 *(28 points)*



(b) (14 points) Suppose you're given the following heuristic, and asked to search the graph using an A* search: $\hat{h}(A) = 5, \hat{h}(B) = 5, \hat{h}(C) = 2, \hat{h}(D) = 2, \hat{h}(G) = 0$. Create a table showing, in the first column, the node that is expanded by A* search at each step of the search process, and in the second column, the set of nodes that are in the frontier after the node in the first column has been expanded. Optionally, you may list a priority next to each node in the frontier if you wish. Note that, since this heuristic is not consistent, you may need to expand a node more than once. Ties are broken in alphabetical order.

Solution method: List $\hat{g}(n)$ and $\hat{h}(n)$, expand in order of increasing sum

Expand	Frontier
A	B(9,5), C(8,2)
C	B(9,5), G(12,0)
G	B(9,5)

Question 4 ***(29 points)***

Consider an MDP with two states, $s \in \{0,1\}$, and two actions, $a \in \{0,1\}$. The states have rewards $r(0) = 9$ and $r(1) = 5$, and the transition probabilities are:

s,a	$P(s'=0\|s,a)$	$P(s'=1\|s,a)$
0,0	0.8	0.2
0,1	0.3	0.7
1,0	0.4	0.6
1,1	0.1	0.9

(a) (15 points) Suppose you start out with an initial policy that always tries action $a = 0$, i.e., $\pi_1(s) = 0$ for $s \in \{0,1\}$. The policy-dependent utility vector $\mathbf{u}_1 = [u_1(0), u_1(1)]^T$ can be computed by solving a linear equation of the form $\mathbf{Au} = \mathbf{b}$. Find the numerical values of the matrix $\mathbf{A}$ and the vector $\mathbf{b}$ assuming the discount factor $\gamma = \frac{1}{2}$.

Solution method: Look up the equation for policy evaluation on the formula sheet!

$$u_1(s) = r(s) + \gamma \sum_{s'} P(s'|s,\pi(s)) u_1(s')$$

$$u_1(0) = 9 + \frac{1}{2}(0.8u_1(0) + 0.2u_1(1))$$

$$u_1(1) = 5 + \frac{1}{2}(0.4u_1(0) + 0.6u_1(1))$$

$$0.6u_1(0) - 0.1u_1(1) = 9$$

$$-0.2u_1(0) + 0.7u_1(1) = 5$$

$$\mathbf{A} = \begin{bmatrix} 0.6 & -0.1 \\ -0.2 & 0.7 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 9 \\ 5 \end{bmatrix}$$

(b) (14 points) The actual state utilities in this problem are approximately $u(0) = 17$, $u(1) = 12$. Assuming $\gamma = \frac{1}{2}$, what is the q-function, $q(s,a)$, that corresponds to these utilities?

Solution method: Remember the definition of $q(s,a)$. It's the same as utility, but assuming that, instead of doing the best action, you do action a. With that definition, we can re-write Bellman's equation as:

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

...plugging in the provided numerical values, we get:

$$q(0,0) = 9 + \frac{1}{2}(0.8 \times 17 + 0.2 \times 12)$$

$$q(0,1) = 9 + \frac{1}{2}(0.3 \times 17 + 0.7 \times 12)$$

$$q(1,0) = 5 + \frac{1}{2}(0.4 \times 17 + 0.6 \times 12)$$

$$q(1,1) = 5 + \frac{1}{2}(0.1 \times 17 + 0.9 \times 12)$$

Exam 3 summary

- Where & when: Lincoln Hall Theater, 1:00pm, Mon April 6
- How: Paper exam w/ attached formula sheet and scratch paper
- What to bring:
  - pencils & erasers or pens
  - One 8.5x11 sheet of handwritten notes, front & back
  - ID
- Don’t bring:
  - Calculators or computers
  - Textbook or printed notes
- Recommended study method:
  - Lectures 12-25 (18-Feb through 30-Mar)
  - MP 5-10 (equations, not code)
  - Practice exam