CS 440/ECE 448 Lecture 35: Exam 5 Review

Outline

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

Exam administration

- Where & when: Lincoln Hall Theater, 8:00am, Tues May12
- How: Paper exam w/ attached formula sheet and scratch paper
- What to bring:
  - pencils & erasers or pens
  - Up to 2 8.5x11 sheets of handwritten notes, front & back
  - ID
- Don’t bring:
  - Calculators or computers
  - Textbook or printed notes

Grading

- Each TA will grade two problem parts
- 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, all lectures, including those not included on exams 1, 2, 3, and 4!
- Make sure you understand the equations behind all MPs. 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

- Decision theory
  - Decision theory, naïve Bayes
  - HMM & Bayesian network
- Machine learning
  - Linear, Logistic, and Nonlinear regression, Softmax
  - Transformer, Theorem proving, Computer vision, CNN, Relevance propagation
- Sequential environments
  - Robotics, Search
  - MDP & RL
- Multi-agent environments
  - Minimax, Alpha-beta, Expectiminimax
  - Game theory, Mixed equilibrium, Adversarial learning

Decision Theory

Bayes Error Rate

$$= \sum_{x} P(x)\left(1 - \max_{y} P(y|x)\right)$$

$$\text{Precision} = P(Y = 1|f(X) = 1)$$
$$\text{Recall} = P(f(X) = 1|Y = 1)$$

$$\text{Specificity} = P(f(X) = 0|Y = 0)$$
$$\text{Sensitivity} = P(f(X) = 1|Y = 1)$$

Confusion Matrix

Correct Label: \ Classified As:	0	1
0	TN	FP
1	FN	TP

Naïve Bayes

- Naïve Bayes classifier:

$$f(x) = \text{argmax}(\log P(Y = y) + \log P(X = x|Y = y))$$

$$\log P(X = x|Y = y) \approx \sum_{i=1}^{n} \log P(W = w_i|Y = y)$$

- Laplace Smoothing w/OOV:

$$P(W = w|Y = y) = \begin{cases} \dfrac{k + \text{Count}(w, y)}{N + k(M + 1)} & \text{in} - \text{vocabulary} \\ \dfrac{k}{N + k(M + 1)} & \text{OOV} \end{cases}$$

- Laplace smoothing w/o OOV:

$$P(W = w|Y = y) = \frac{k + \text{Count}(w, y)}{N + kM}$$

Bayes networks

- Bayesian network: A better way to represent knowledge
  - Reduces space complexity from $\mathcal{O}\{v^n\}$ to $\mathcal{O}\{nv^p\}$ -- huge if $n \gg p$
  - Does not reduce time complexity without extra assumptions
- Key ideas: Independence and Conditional independence
  1. Shared ancestor $\Longrightarrow$ dependent unless ancestor known
  2. Shared descendant $\Longrightarrow$ dependent unless descendant unknown
  3. No shared ancestor or descendent $\Longrightarrow$ independent

HMMs

- Review: Bayesian classifier, Bayesian networks

$$f(x) = \underset{y}{\operatorname{argmax}}\, P(Y = y | X = x)$$

- HMM: Probabilistic reasoning over time

$$\pi_i = P(Y_1 = i)$$
$$a_{i,j} = P(Y_t = j | Y_{t-1} = i)$$
$$b_j(\boldsymbol{x}_t) = P(X_t = \boldsymbol{x}_t | Y_t = j)$$

- Viterbi algorithm

$$\textbf{SCORE:}\ v_t(j) = \max_i v_{t-1}(i) + \log a_{i,j} + \log b_j(\boldsymbol{x}_t)$$

$$\textbf{BACKPOINTER:}\ \psi_t(j) = \underset{i}{\operatorname{argmax}}\, v_{t-1}(i) + \log a_{i,j} + \log b_j(\boldsymbol{x}_t)$$

Learning

- **Learning:** Given $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, find the function $f(X)$ that minimizes some measure of risk.
- **Empirical risk**, a.k.a. training corpus error:

$$\mathcal{R}_{\text{emp}} = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f(x_i))$$

- **True risk,** a.k.a. expected test corpus error:

$$\mathcal{R} = \text{E}[\ell(Y, f(X))] = \mathcal{R}_{\text{emp}} + \mathcal{R}_{\text{generalization}}$$

- **Early Stopping**: Stop when error rate on the dev set reaches a minimum, then test on test set to get an unbiased estimate of true risk

Linear regression

- Definition of linear regression

$$f(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{w}$$

- Mean-squared error

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}_i\,, \qquad \mathcal{L}_i = \frac{1}{2}\epsilon_i^2, \qquad \epsilon_i = f(\boldsymbol{x}_i) - y_i$$

- Gradient descent

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \frac{\partial \mathcal{L}}{\partial \boldsymbol{w}}, \qquad \frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = \frac{1}{n}\sum_{i=1}^{n} \epsilon_i \boldsymbol{x}_i$$

- Stochastic gradient descent

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \frac{\partial \mathcal{L}_i}{\partial \boldsymbol{w}}, \qquad \frac{\partial \mathcal{L}_i}{\partial \boldsymbol{w}} = \epsilon_i \boldsymbol{x}_i$$

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:

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

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

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

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

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: $\boldsymbol{X}^{(l)} = \boldsymbol{X}^{(l-1)} + \boldsymbol{O}^{(l-1)}$

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)

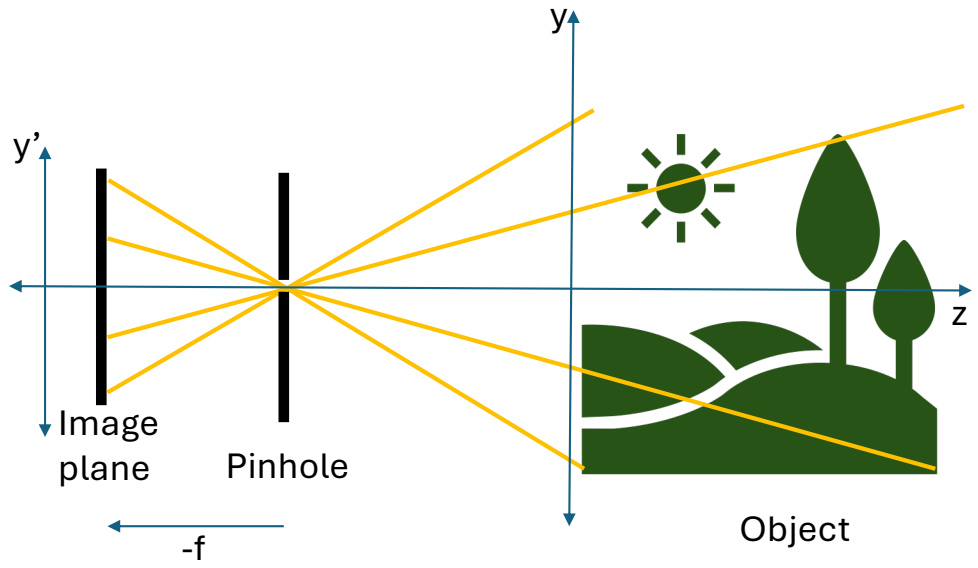
Computer vision: Pinhole camera

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

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

The alpha-beta algorithm

- Max inherits α, β from parents, sets $v = -\infty$, then for each child:
  - Set $v = \max(v, \text{child's } v)$
  - Set $\alpha = \max(\alpha, \text{child's } v)$
  - If $\alpha \geq \beta$, prune all remaining children

- Min inherits α, β from parents, sets $v = \infty$, then for each child:
  - Set $v = \min(v, \text{child's } v)$
  - Set $\beta = \min(\beta, \text{child's } v)$
  - If $\alpha \geq \beta$, prune all remaining children

Expectiminimax

Remember that the solution to a Markov decision process is the policy that maximizes the Bellman equation, which we could call the expectimax equation:

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

Expectiminimax maximizes a generalized Bellman equation:

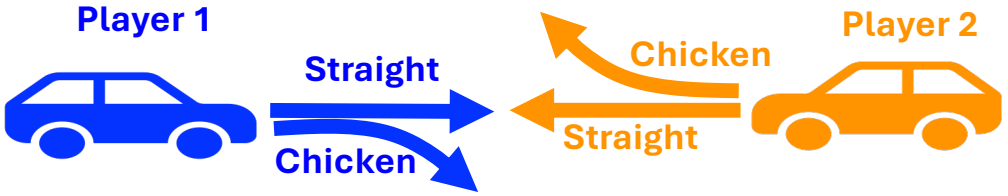
$$v(s) = \begin{cases} \max_a \sum_{s'} P(S_{t+1} = s' | S_t = s, a) v(s') & s \text{ is a max state} \\ \min_a \sum_{s'} P(S_{t+1} = s' | S_t = s, a) v(s') & s \text{ is a min state} \end{cases}$$

These are the same operation. They are both NP-complete and have similar approximate solution methods, e.g., value iteration, policy iteration, etc.

Game Theory

- Dominant strategy
  - A strategy that's optimal for one player, regardless of what the other player does
  - Not all games have dominant strategies
- Nash equilibrium
  - An outcome (one action by each player) such that, knowing the other player's action, each player has no reason to change their own action
  - Every game with a finite set of actions has at least one Nash equilibrium, though it might be a mixed-strategy equilibrium.
- Pareto optimal
  - An outcome such that neither player would be able to win more without simultaneously forcing the other player to lose more
  - Every game has at least one Pareto optimal outcome. Usually there are many, representing different tradeoffs between the two players.

Mixed Equilibrium & Adversarial Learning



Bob \ Alice	Defect	Cooperate
Defect	-10, -10	-1, 2
Cooperate	2, -1	1, 1

- The Game of Chicken
- Mixed equilibrium: Randomness is rational behavior because

$$\begin{bmatrix}-1\\1\end{bmatrix}^T \boldsymbol{A\beta} = \mathbf{0}. \text{ and. } \boldsymbol{\alpha}^T \boldsymbol{B} \begin{bmatrix}-1\\1\end{bmatrix} = 0$$

- Adversarial learning (doesn't always converge)

$$a = a + \eta \frac{\partial E[r_A]}{\partial a}. \text{ and. } b = b + \eta \frac{\partial E[r_B]}{\partial b}$$

- Generative adversarial networks

Mechanism Design & Repeated Games

- Mechanism design
  - Design rewards so that rational behavior benefits the game designer
- Second-price auction: Auctioneer sacrifices profit for information

$$b_1^* = \underset{b_1}{\operatorname{argmax}}\, E[\text{earn}] = \underset{b_1}{\operatorname{argmax}}\, P(b_1)(v_1 - \boldsymbol{b_0})$$

- Repeated games (max = find a Nash equilibrium):

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

Outline

- Exam administration
- Exam topics
- Solution methods for posted practice exam: see exam and solution online