CS221 Midterm Review

Week 5

Midterm Prep

This review only covers the **general machine learning** and **games** topics, which are of course only a subset of the topics covered so far in the course. The problems in this handout aim to (i) gather all the general machine learning topics that we covered at the start of the quarter into one problem and (ii) go over the most recent topic, games, which was just introduced this week.

If time permits during the problem session, then we may also go over Problem 2 from Week 3's problem session on **search** (see the handout for that week's Friday session on the website).

Besides **general machine learning**, **search**, and **games**, make sure you also refresh yourself on the **k-means clustering algorithm** as well as **Markov decision processes (MDPs)**. For MDPs, we recommend looking back at the second half of Week 4's problem session, during which we went over some comprehensive problems regarding MDPs and when each MDP algorithm is useful (the handout and slides for that week's Friday session are also on the website).

For more practice problems, see the Files on this class's Canvas, which should include three exams with solutions from previous years.

Practice Problems

1) ***"The fear of loss [functions] is a path to the dark side."*** **- Yoda**

(a) We have a trained linear regression model $f_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x)$. In your own words, explain why we call this model "linear". Is it linear in x? Linear in $\phi(x)$? Linear in $\mathbf{w}$? Note that linearity for some generic function g means that $g(x + y) = g(x) + g(y)$ and $g(\alpha x) = \alpha g(x)$ for all parameters α.

(b) We are working with a classification model $f_{\mathbf{w}}(x) = \text{sign}(\mathbf{w} \cdot \phi(x))$. What is the decision boundary? What does $\mathbf{w} \cdot \phi(x)y = -1000$ imply about how well our model classified the point (x, y)? What does $\mathbf{w} \cdot \phi(x)y = 0.1$ imply about how well our model classified the point (x, y)?

Additionally, you consider using the following loss function

$$\mathbb{1}[(\mathbf{w} \cdot \phi(x))y \leq 0]$$

for gradient descent. Explain why using this loss function is a bad idea.

(c) After solving the prior problem, you realize the zero-one loss function is a bad idea and instead decide to use the logistic loss function. Your data is $y \in \{0, +1\}$, so you define the logistic loss as follows

$$L(x, y; \mathbf{w}) = -y \log(f(x; \mathbf{w})) - (1 - y) \log(1 - f(x; \mathbf{w})) \tag{1}$$

where f has a range of $[0, 1]$. Before picking f, you'd like to differentiate L with respect to $\mathbf{w}$. Is this possible, and if so, what is $\frac{\partial L}{\partial \mathbf{w}}$? (Food for thought: how would the derivative change if it were over a summation?)

(d) For your function f in the above loss function, you can't decide between using the sigmoid function,

$$g(x; \mathbf{w}) = \frac{1}{1 + e^{-\mathbf{w}^T x}}$$

or the shifted tanh function,

$$h(x; \mathbf{w}) = \frac{1}{2} \tanh(\mathbf{w}^T x) + \frac{1}{2} \quad \text{with} \quad \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

in place of f. How would the derivative from Part (c) look like with function g above in place of f, and with function h above in place of f?

(e) Assume that you were able to format $\frac{\partial L(x,y;\mathbf{w})}{\partial \mathbf{w}}$ as $cx(f(x; \mathbf{w}) - y)$ in the previous problem, where c is a constant and f is the corresponding sigmoid g or tanh h functions. Explain why writing the derivative of the loss function in the form of $cx(f(x; \mathbf{w}) - y)$ is very convenient for backpropagation.

Hint: Draw out the backpropagation tree and think about what quantities we would need to compute when evaluating $L(x, y; \mathbf{w})$.

(f) Unfortunately your model has poor performance for both sigmoid and tanh. You think that it's because your model isn't expressive enough. You decide to make your model a neural network to hopefully fix that. You decide to keep the loss function and still use either sigmoid or tanh as your final activation (mostly because you've already done the work in differentiating them).

Now, rather than pluging x directly into your choice for f, you plug x into a neural network $N(x; A, B) = z$, and then take $f(z; \mathbf{w})$. Let

$$N(x; A, B) = B \max\{Ax, 0\} = z$$

(Self check: Do you remember the derivative of the max function?)

The loss function is now:

$$L(x, y; A, B, \mathbf{w}) = -y \log(f(N(x; A, B); \mathbf{w})) - (1 - y) \log(1 - f(N(x; A, B); \mathbf{w}))$$

It looks you need to find the derivative again for this new loss function, but you think you can reuse your work from the earlier parts of this problem. Can we we reuse our result from (d) for $\frac{\partial L}{\partial w}$?

(Food for thought: suppose we figure that our model's poor performance was due to overfitting instead. Why might L_2 regularization help, and how would it change our loss function from Part (c)?)

2) "I am the Lorax who speaks for the [game] trees, which you seem to be [alpha-beta pruning] as fast as you please!" - The Lorax

(a) Evaluate the following game (Figure 1) where the edges are probabilities:

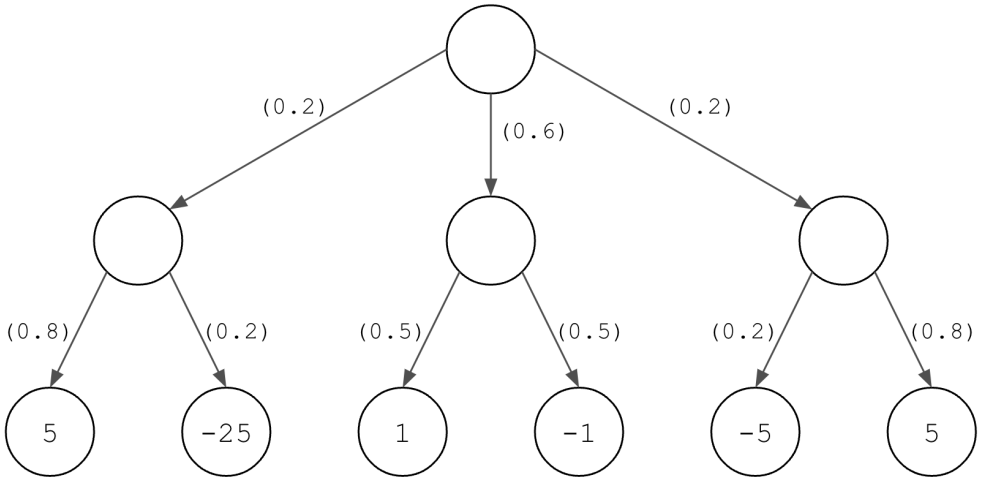

Figure 1

Pretend the top node is now a maximizing player. Under expectimax, which action should they take (left, center, or right) and what is the value of the game?

(b) Evaluate the game in Figure 2 using the minimax strategies for both players, with $x = -5$. Recall that upwards pointing triangles is the maximizing player and downwards pointing is the minimizing player.

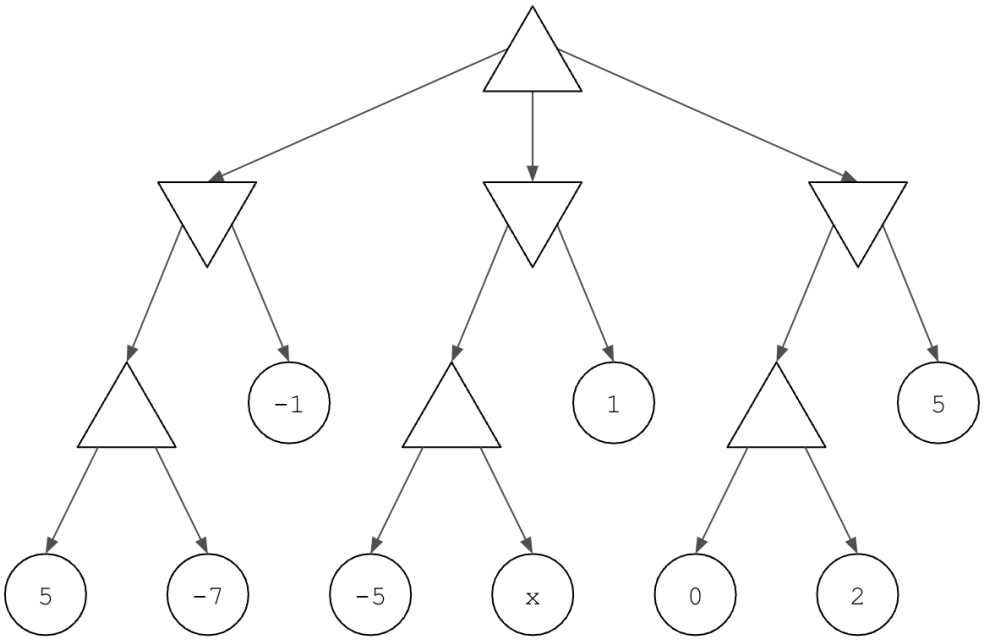

Figure 2

Can we pick x so that the maximizing player loses? Why or why not.

(c) Can either player do better by deviating from minimax assuming the other stays?

(d) Evaluate the game in Figure 3 under the expectiminimax strategy, using $x = -5$. Write down a funny answer for who the third player is represented by the circles.

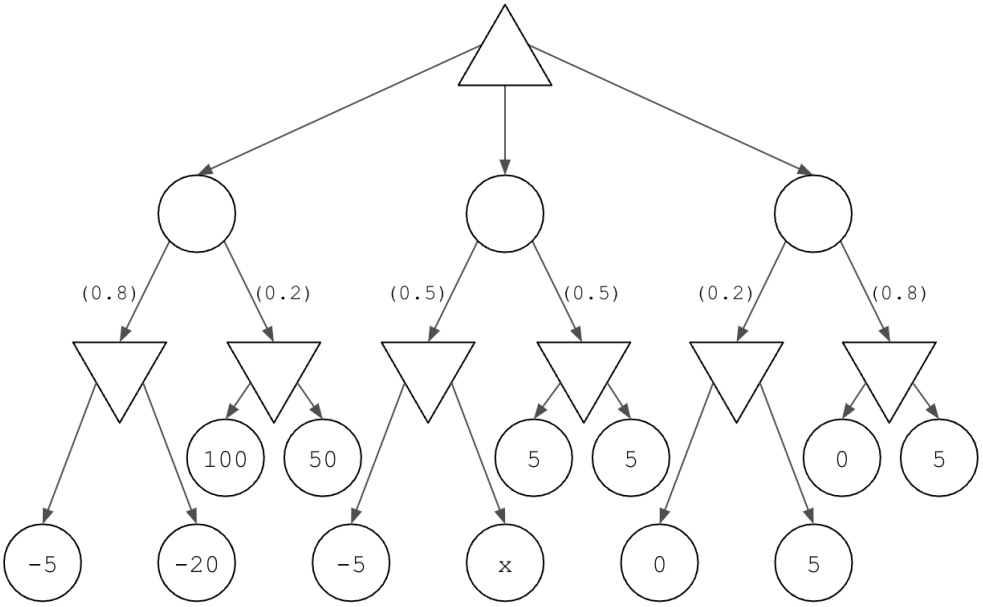

Figure 3

(e) In the previous problem, is there a value of x we can choose so that the game does not end in a draw?

(f) Assume that in the case of a tie in the value of multiple options, the maximizing player chooses the rightmost tied-value action. Still referring to (d) and Figure 3 with $x = -5$, explain, in your own words, why expectiminimax always chooses to draw the game given this choice of tie-breaking. Is there a better way of breaking ties?

(g) Let's develop some intuition for alpha-beta pruning. Look at the minimax tree in Figure 4, specifically the left subtree.

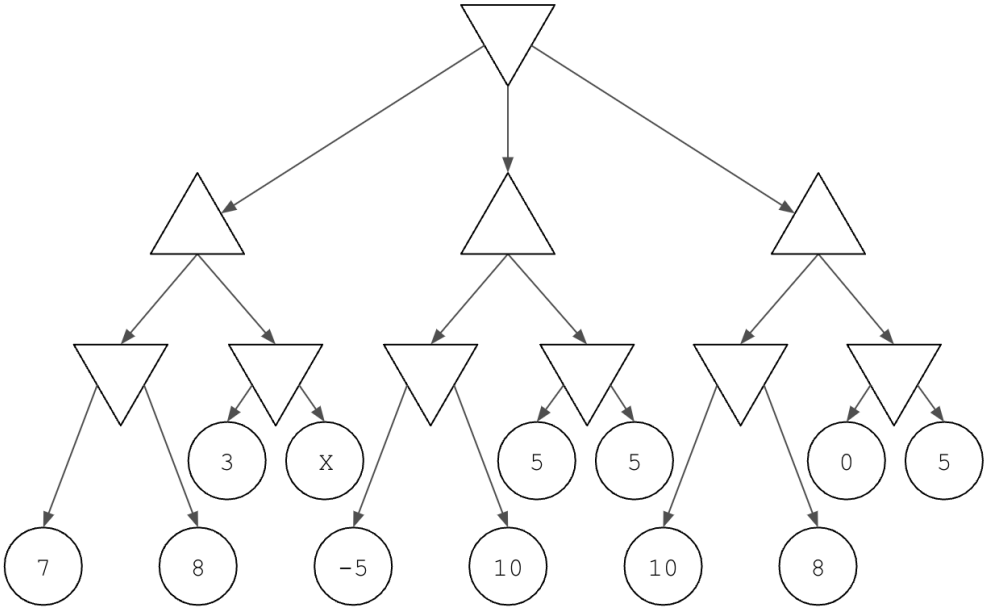

Figure 4

Explain why we don't care about the value of x.

(h) To run alpha-beta pruning we:

- Find a_s, the lower bound on value at max node s.
- Find b_s, the upper bound on value at min node s.
- To prune, calculate the following (where $s' \leq s$ indicates ancestors):

$$\alpha_s = \max_{s' \leq s} a_{s'} \quad \text{and} \quad \beta_s = \min_{s' \leq s} b_{s'}$$

noting that we are maximizing over lower bounds and minimizing over upper bounds.

Run alpha-beta pruning on the tree from the previous question, Figure 4.