CS221 Midterm Review Problem Set Solutions

Week 5

Note that this is only a subset of topics covered so far in the course. Make sure you still study the lecture material, previous problem sessions (week 4 is great for MDPs which are not covered here), and use the released back exams to your advantage.

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 means that $g(x+y) = g(x) + g(y)$ and $g(\alpha x) = \alpha g(x)$ for all α.

Solution Linear regression models are linear both in $\phi(x)$ and $\mathbf{w}$, but not in x. A simple example of this is using $\phi(x) = [1, x, x^2, x^3]$, which can be used to fit cubic polynomials.

(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 $\mathbf{w} \cdot \phi(x)y = -1000$ mean? How about $\mathbf{w} \cdot \phi(x)y = 0.1$? You consider using the loss function

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

for gradient descent. Explain why this is a bad idea.

Solution The decision boundary is $w \cdot \phi(x) = 0$. In the first case we are confident ($\mathbf{w} \cdot \phi(x)$ is large, far from decision boundary) but incorrect (sign is wrong). In the second case we are not very confident (near the decision boundary with 0.1) but at least the sign is correct and the point will be correctly classified. This loss has zero gradient almost everywhere!

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

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

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 $\frac{\partial L}{\partial \mathbf{w}}$?

Solution Yes! We use the chain rule:

$$\begin{aligned}\frac{\partial L(x, y; \mathbf{w})}{\partial \mathbf{w}} &= -y \frac{1}{f(x; \mathbf{w})} \frac{\partial f(x; \mathbf{w})}{\partial \mathbf{w}} + (1 - y) \frac{1}{1 - f(x; \mathbf{w})} \frac{\partial f(x; \mathbf{w})}{\partial \mathbf{w}} \\ &= \left(\frac{f(x; \mathbf{w}) - y}{f(x; \mathbf{w})(1 - f(x; \mathbf{w}))} \right) \frac{\partial f(x; \mathbf{w})}{\partial \mathbf{w}}\end{aligned}$$

(d) You can't decide between using the sigmoid function,

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

and a 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 your loss in place of f. Compute the loss from Equation 1 using g for f and then h for f.

Solution To make things easier for ourselves, we can first compute $\frac{\partial g}{\partial \mathbf{w}}$ and $\frac{\partial h}{\partial \mathbf{w}}$ and substitute those into $\frac{\partial f}{\partial \mathbf{w}}$ in our solution from (c). Thus

$$\begin{aligned}
\frac{\partial g(x; \mathbf{w})}{\partial \mathbf{w}} &= -(1 + e^{-\mathbf{w}^T x})^{-2} \frac{\partial}{\partial \mathbf{w}} \left(1 + e^{-\mathbf{w}^T x}\right) \\
&= \frac{x e^{-\mathbf{w}^T x}}{(1 + e^{-\mathbf{w}^T x})^2} \quad \text{(this is a valid answer)} \\
&= x \frac{1}{(1 + e^{-\mathbf{w}^T x})} \frac{e^{-\mathbf{w}^T x}}{(1 + e^{-\mathbf{w}^T x})} \\
&= x g(x; \mathbf{w})(1 - g(x; \mathbf{w}))
\end{aligned}$$

and

$$\begin{aligned}
\frac{\partial h(x; \mathbf{w})}{\partial \mathbf{w}} &= \frac{1}{2} \frac{\partial \tanh(\mathbf{w}^T x)}{\partial \mathbf{w}} \\
&= \frac{1}{2} \frac{\partial}{\partial w} \frac{(e^{\mathbf{w}^T x} - e^{-\mathbf{w}^T x})}{(e^{\mathbf{w}^T x} + e^{-\mathbf{w}^T x})} \\
&= \frac{1}{2} \left[\frac{(e^{\mathbf{w}^T x} + e^{-\mathbf{w}^T x})}{(e^{\mathbf{w}^T x} + e^{-\mathbf{w}^T x})} - \frac{(e^{\mathbf{w}^T x} + e^{-\mathbf{w}^T x})^2}{(e^{\mathbf{w}^T x} + e^{-\mathbf{w}^T x})^2} \right] x \\
&= \frac{1}{2}(1 - \tanh(\mathbf{w}^T x)^2) x
\end{aligned}$$

and we can plug these into our solution from (c). For sigmoid g:

$$\begin{aligned}
\frac{\partial L(x, y; \mathbf{w})}{\partial \mathbf{w}} &= \left(\frac{g(x; \mathbf{w}) - y}{g(x; \mathbf{w})(1 - g(x; \mathbf{w}))} \right) \frac{\partial g(x; \mathbf{w})}{\partial \mathbf{w}} \\
&= \left(\frac{g(x; \mathbf{w}) - y}{g(x; \mathbf{w})(1 - g(x; \mathbf{w}))} \right) g(x; \mathbf{w})(1 - g(x; \mathbf{w})) x \\
&= x(g(x; \mathbf{w}) - y)
\end{aligned}$$

which looks surprisingly similar to the gradient of squared error loss. For tanh h:

$$\begin{aligned}
\frac{\partial L(x,y;\mathbf{w})}{\partial \mathbf{w}} &= \left(\frac{h(x;\mathbf{w})-y}{h(x;\mathbf{w})(1-h(x;\mathbf{w}))}\right)\frac{\partial h(x;\mathbf{w})}{\partial \mathbf{w}} \\
&= \left(\frac{h(x;\mathbf{w})-y}{\frac{1}{2}(\tanh(\mathbf{w}^T x)+1)(1-\frac{1}{2}(\tanh(\mathbf{w}^T x)+1))}\right)\frac{1}{2}(1-\tanh(\mathbf{w}^T x)^2)x \\
&= \left(\frac{h(x;\mathbf{w})-y}{(\tanh(\mathbf{w}^T x)+1)\frac{1}{2}(1-\tanh(\mathbf{w}^T x))}\right)(1-\tanh(\mathbf{w}^T x)^2)x \\
&= 2x(h(x;\mathbf{w})-y)
\end{aligned}$$

isn't that neat!

(e) Assume 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. (if you couldn't get it like this, don't worry, it's just some algebra). Explain why this loss function's gradient is very convenient for backpropagation. Hint: think about what quantities we compute when evaluating $L(x,y;\mathbf{w})$.

Solution This gradient is convenient because we compute all components of it during the forward pass (evaluation of $L(x,y;\mathbf{w})$). The gradient just uses x (our data), y (our label), and $f(x;\mathbf{w})$ which we have to compute anyways when we compute the loss. Thus there is no extra computation when backpropagating other than putting the pieces together.

(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 since you've already differentiated them). Thus rather than plug 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

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

And the loss is now:

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

You figure you need to do some more differentiating, but you think you can reuse some things from earlier problems. Can we we reuse our result from (d) for $\frac{\partial L}{\partial w}$?

Solution Yes, we can reuse our result for $\frac{\partial L}{\partial \mathbf{w}}$, with just a slight change. We need to replace x with $z=N(x;A,B)$ whenever it appears. Remember we were differentiating with respect to $\mathbf{w}$ so we treated x as a constant. By replacing it with $z=N(x;A,B)$ we are just redefining a *constant*, which won't change the actual differentiation process.

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:

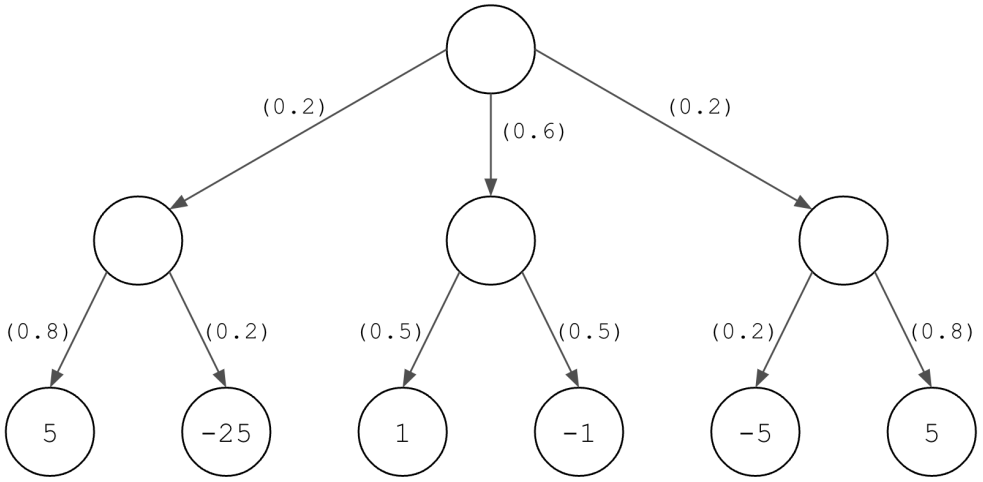


Figure 1

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.

Solution Bottom row left to right: $-1, 0, 3$. Overall value is 0.4. Under expectimax they would choose the right subtree and get a value of 3.

(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.

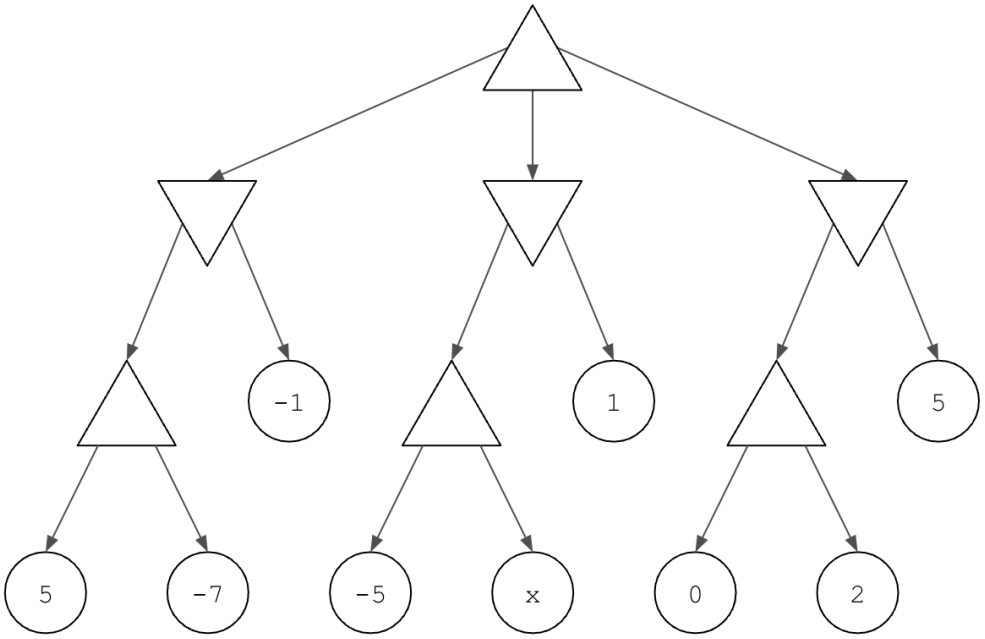


Figure 2

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

Solution Bottom row of maximizing triangles, left to right: 5,-5,2
Second row of minimizing triangles, left to right: -1,-5,2
Top (value of the game): 2.
No, we cannot max the maximizing player lose by changing x. If $x < -5$ it is ignored by its parent maximizer, and if $-5 \leq x < 1$ it is chosen by the minimizer but ignored by the right subtree having a minimax value of 2. If $x \geq 1$ then the 1 in the middle subtree will be chosen by the minimizer.

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

Solution No! Proved in the slides. If we could improve by deviating then it wouldn't be minimax by definition.

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

Solution Bottom row of minimizing triangles, left to right: -20, 50, -5, 5, 0, 0
Expected value of middle row of circles, left to right: -6, 0, 0
Expected value of the game (maximizer at the root): 0

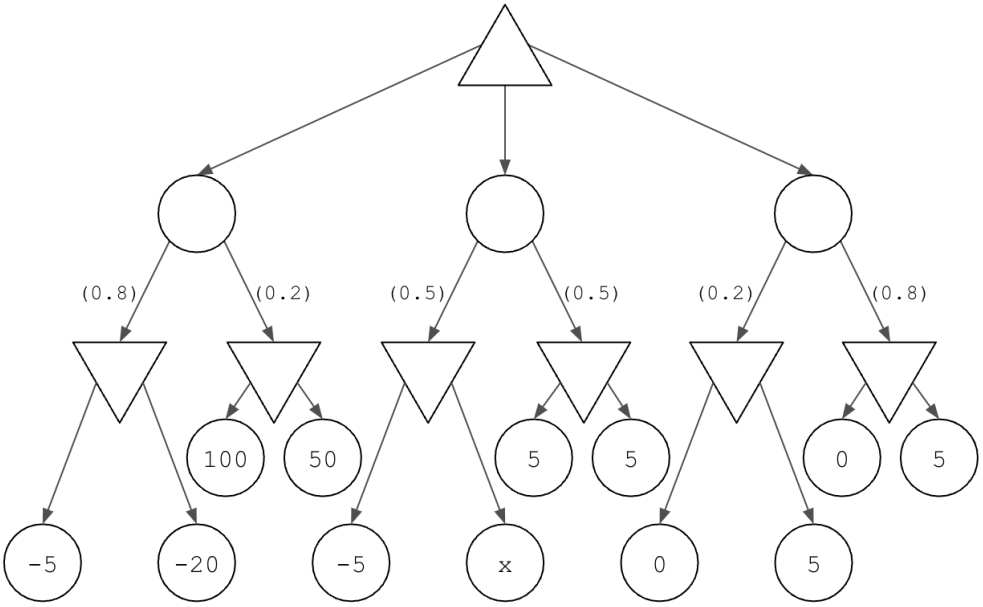


Figure 3

I like to think that the third player playing the randomness in the circles is my lucky penny I lost in third grade that has been out to get me ever since. (The point is that the circles follow a known stochastic policy, they aren't 'playing' the game the same was as the two players are).

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

Solution No, making $x < -5$ would result in decreasing the expected value of the middle subtree, but the right subtree has an expected value of 0. If we take $x > -5$ it'll be ignored by its parent minimizer.

(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?

Solution With $x = -5$ both the middle subtree and rightmost subtree have expected value of 0. However in the right subtree the expected value has zero variance, the game will always draw since both minimizing nodes have value 0. The middle subtree is averaging -5 and 5 with equal probability, which is an expected value of 0 as well, but this time with non-zero variance. However, it isn't necessarily better to break the tie towards larger variance. We go from guaranteeing a draw to losing half the time and winning half the time if we take

the middle subtree, and one isn't necessarily 'better' than the other given the defined utilities of the game.

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

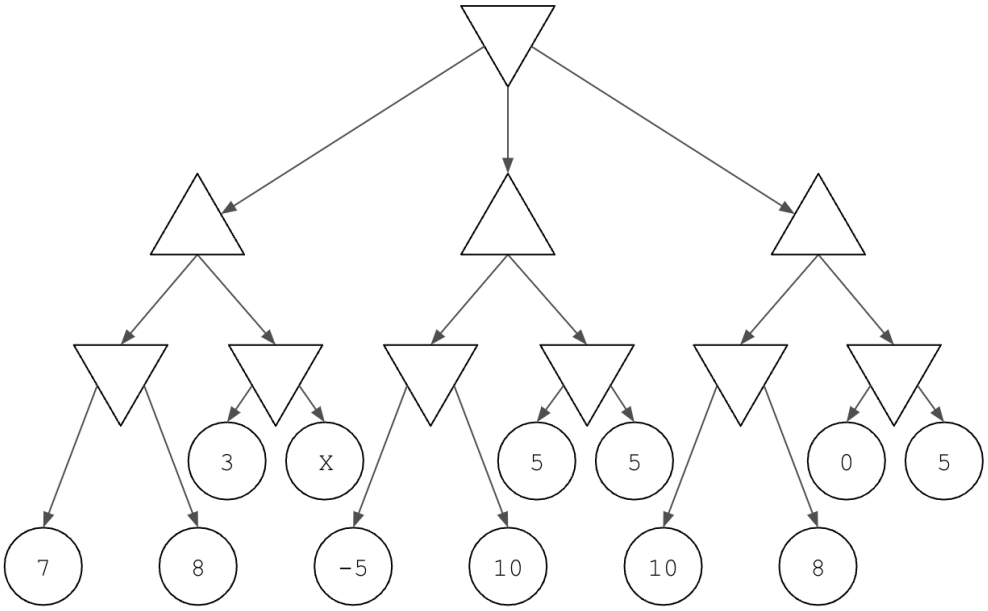


Figure 4

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

Solution The bottom left minimizing node will choose 7. Then the parent maximizer to that minimizer will choose at least 7. Going to the second leftmost minimizer, the first value is a 3, meaning the minimizer will choose at most 3. Since the maximizer already has access to a 7, it isn't going to care if the second minimizing node offers a 3 or something smaller. Thus we don't care about the value of x since it won't change our decision in the left subtree.

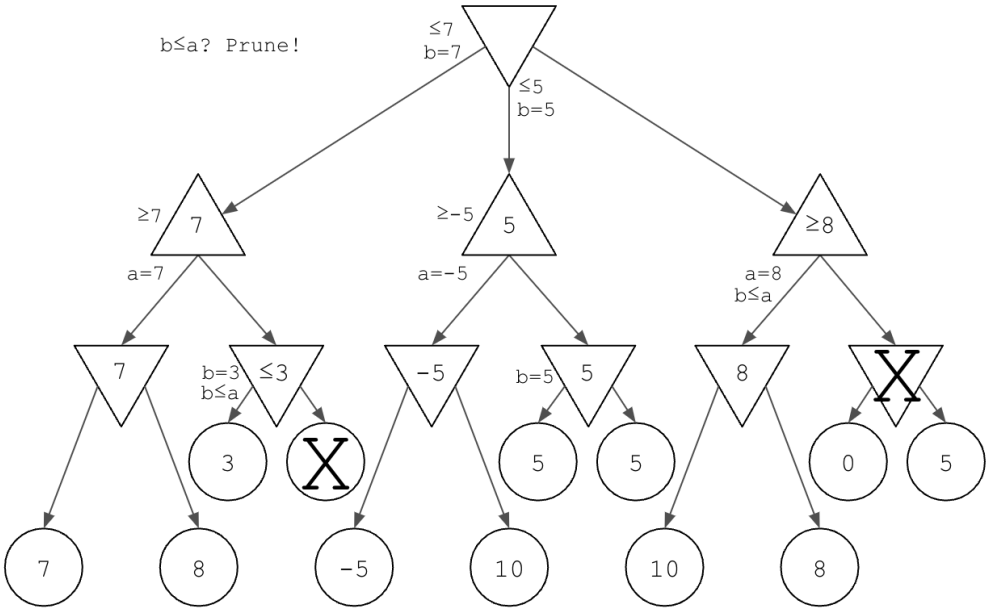
(h) To run alpha-beta pruning we:

- 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.



Solution Solution:

See we can confirm that we cannot prune anything in the middle subtree since we would be pruning the optimal value of 5.

(i) You recall learning about Nash Equilibrium and the Prisoner's Dilemma in lecture, but you can't remember exactly what the numbers were. You remember there were three conditions:

- If both testify, then both are sentenced to a years in jail.
- If both refuse, then both are sentenced to b years in jail.
- If only one testifies, they get 0 years and the other gets c years.

And you remember that a Nash Equilibrium is when no player has incentive to change their strategy. First, write down the payoff matrix for this game in terms of a, b, c, and 0. The matrix should look symmetric. Then, write down the conditions (or just values for a, b, and c) so that both testify is a Nash Equilibrium.

Solution

$P_1 \backslash P_2$	Testify	Refuse
Testify	$a \backslash a$	$0 \backslash c$
Refuse	$c \backslash 0$	$b \backslash b$

Both testify is just regular Prisoner's Dilemma, so $c < a < b < 0$. You can verify this by trying both strategies for one player, and seeing which is in the other player's best interest. If P_2 testifies, then P_1 can testify for a or refuse for c, so $a > c$ implies testify. If P_2 refuses, then P_1 can testify for 0 or refuse for b, and if $b < 0$ they will testify.

3) Ikea is a BFS (Big Furniture Store)

Oh no! You're trapped in an Ikea! Let's use search to get out. You're located at $(0, 0)$ and the exit is at $(1, 1)$. Between you and the exit are a significantly large number of boxes (denoted $b_i \in B$), situated somewhere in the square defined by corners $(0, 0) - (1, 0) - (1, 1) - (0, 1)$, blocking your path. You can remove boxes that are in your way, but you'd like to minimize your effort.

For this problem, consider the algorithms: Backtracking, BFS, DFS, DFS-ID, Dynamic Programming, UCS, and A^*.

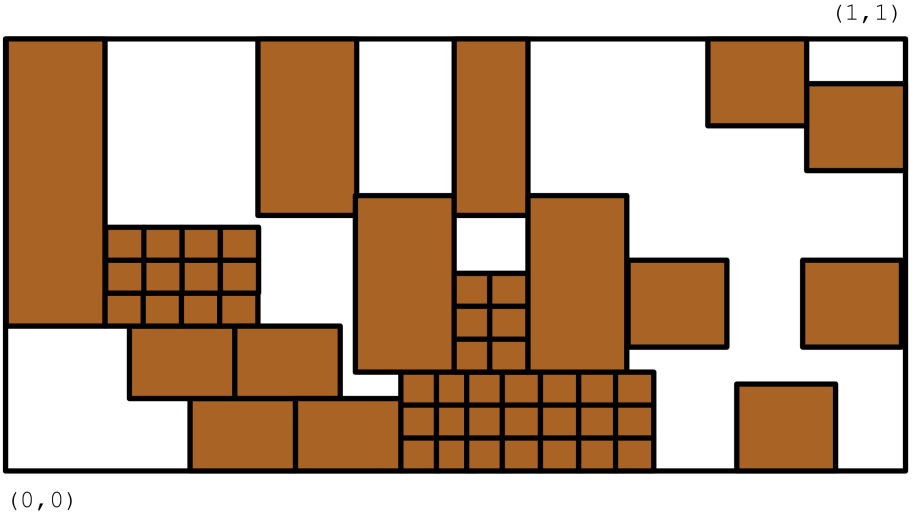


Figure 5: Example scenario, with start at bottom left and end at top right.

(a) Let's formalize this as a search problem:

- $s_{\text{start}} =$

  Solution s_0, which just represents our starting position $(0, 0)$.
- IsEnd$(s) =$

  Solution s_f, representing our final position $(1, 1)$.
- States $S =$

  Solution $\{s_0 \cup B \cup s_f\}$
- Actions$(s) =$

  Solution All states reachable from s, meaning $\{s_i\}$ such that that you can get to s_i from s without going through any boxes.
- Succ$(s, a) =$

Solution Since a at a state s are just potential successive actions, $\text{Succ}(s, a) = a$.

- $\text{Cost}(s, a) =$ unknown for now

(b) Consider that the boxes are all empty and you don't care how many you have to remove. Write a cost function for this scenario. What algorithm would you use to find a path to the exit if you only care about *how long* your algorithm takes to run? What is the exact worst case length of your path (number of boxes traversed)? Is it possible for DFS to return the shortest path?

Solution $\text{Cost}(s, a) = 0$ for all state/action pairs. DFS would just go down the tree until it reaches the end state. Although it won't necessarily be the shortest path (it is possible but unlikely), it won't backtrack through the tree at all since every path eventually leads to the end state after going through at most $|B|$ boxes. Note backtracking search wouldn't necessarily be as fast (in this case) since it explores all paths and DFS would just follow one path.

(c) After throwing hundreds of tiny little boxes out of your way, you realize you do in fact care how many boxes you have to remove. Write a cost function for this scenario. What algorithm would you use to find the shortest path to the exit? If there are $O(b)$ boxes reachable from each box, and the shortest path is length d, what is the complexity of this algorithm? What is the memory cost?

Solution $\text{Cost}(s, a) = c$ where c is any positive constant. BFS is one option, with memory cost $O(b^d)$ time and memory cost. DFS with iterative deepening is another option with $O(d)$ memory cost and $O(b^d)$ time complexity. Backtracking technically works but is much slower with $O(b^{|B|})$ time complexity and $O(|B|)$ memory cost.

(d) Inside one of the boxes you've found a nifty little drone that can fly up, look at the boxes, and perform a sort of DFS up to a given depth. This means that given a depth $\tilde{d}$, the drone reports whether there is a path of length at most $\tilde{d}$ or not (there is no information about a shorter path if one is found). Let's say you know there exists a path of length d_0 from part (b). You don't know how long the battery in your drone is going to last, so to conserve it you'd like to have it run the algorithm for as few $\tilde{d}$ as possible. What should you do (can be a rough description of your method) and how many runs will you need at most?

Solution Given that the algorithm only tells us yes/no if a path of depth $\tilde{d}$ exists, we can use binary search on the interval $[1, d_0 - 1]$, starting with $\lfloor \frac{d_0 - 1}{2} \rfloor$. If the algorithm finds a path of that length we recurse at to the middle of the lower interval, if not we go to the middle of the higher interval and recurse. This leads to $\lceil \log_2 \frac{d_0}{} \rceil$ runs of our algorithm.

(e) You've come to realize that not all boxes are equally easy to move, and you'd prefer to move small boxes over large boxes. For simplicity, let's assume that all

boxes are equally dense, so the effort to move them is **nonzero** and proportional to their area. What is our new cost function? Can we use dynamic programming to minimize the the effort along our path to the exit? What about UCS? If $|B| \to \infty$ will UCS work with some restrictions on the problem? If so what are the restrictions?

Solution The new cost function is $\text{Cost}(s, a) = c_a$, where c_a is the cost of the box a. Dynamic programming cannot be used, as the graph contains cycles. UCS will work, even for infinite boxes, but only if the number of actions is finite.

(f) UCS seems to be taking too long so you decide to give A^* a try. Your friend suggests you use distance (either Euclidean/ℓ_2 or Manhattan/ℓ_1) to the goal as a heuristic. You can use the center of each box as the 'location' of the box. Prove or disprove that your friend's heuristic is consistent.

Solution For either distance metric we have one part of the definition of consistency, that $h(s_f) = 0$. For the other part we need to show that

$$\text{Cost}'(s, a) = \text{Cost}(s, a) + h(\text{Succ}(s, a)) - h(s) \geq 0$$

Before we move any further, it's important to recognize *what* this formula means. Our actual cost of action a at s plus the estimated future cost at the next state needs to be at least the estimated future cost at s. This is just the triangle inequality. However, let's say we are at a box with location $(1, 0.8)$ (call this state s). We can go directly to a box at location $(0.5, 1)$ incurring cost 0.1. We have then that $h(s) = 0.2$ and $h(\text{Succ}(s, a)) = 0.5$ (under either distance metric) with $\text{Cost}(s, a) = 0.1$. This gives a negative adjusted cost! Thus the heuristic is not consistent. This is because there is no penalty/cost for movement in space, only moving boxes.

(g) Since distance didn't work you need a new heuristic. Propose one and prove that it works.

Solution Let's try using the shortest length path, ignoring cost, from s to s_f as our heuristic, ignoring the cost at each box. To check if this consistent, we see that if a brings us closer to the goal (in terms of path length) then $h(\text{Succ}(s, a)) = h(s) - 1$. The action could bring us to a state equal length path to the goal, so $h(\text{Succ}(s, a)) = h(s)$. Lastly, the action could bring us away from the goal in terms of path, so we'd need to go back through s on the shortest path, so $h(\text{Succ}(s, a)) = h(s) + 1$. This means $h(\text{Succ}(s, a)) - h(s)$ is either $-1, 0$, or 1. This is a problem when $h(\text{Succ}(s, a)) - h(s) = -1$, since the heuristic would only be consistent when $\text{Cost}(s, a) > 1$. To fix this, we can use the fact that $\text{Cost}(s, a) > 0$ and let $c_{\min}$ be the smallest cost box to move. Then we can multiply h by $c_{\min}$ and get:

$$\text{Cost}(s, a) + h(\text{Succ}(s, a)) - h(s) \geq \text{Cost}(s, a) - c_{\min} \geq 0$$

And so the heuristic of using the future cost relaxation where we assume all boxes are the same effort as the smallest box is consistent.