CS182 Cheatsheet

Anusha Murali

Search Algorithms

A^* Search

Admissible Heuristic: For every node x, $h(x) \leq h^*(x)$.
Consistent Heuristic: For every 2 nodes x, y, $h(x) \leq c(x,y) + h(y)$, where $c(x,y)$ is the cost of the cheapest path between x and y.

Theorem A^* tree search with an admissible heuristic returns an optimal solution.

Proof. The proof is by induction. Let us assume that A^* tree search with an admissible heuristic results in a suboptimal goal t.

Therefore, the suboptimal goal t was expanded before the optimal goal t^*. Then $\exists$ a node x on the optimal path to t^*. that has been discovered but NOT expanded because t is suboptimal. So:

$$\begin{aligned} f(x) &= g(x) + h(x) && \text{(by definition of } f\text{)} \\ &\leq g(x) + h^*(x) && \text{(by admissibility)} \\ &= g(t^*) < g(t) && \text{(because not yet on the optimal path)} \end{aligned}$$

But $g(t) = f(t)$ because $h(t) = 0$. Therefore, $f(x)$ is strictly smaller than $f(t)$. This means x should have been expanded before t, A^*, contradicting our initial assumption. Therefore, a tree search with an admissible heuristic returns an optimal solution. □

Prove consistency implies admissibility

Proof. The proof is by induction. Let g be the goal state, so we assume that since edge costs are assumed to be non-negative, $h(g) = 0$.

Base case: Let n denote any predecessor of the goal state g. If the heuristic function h is consistent, then by definition,

$$h(n) \leq c(n,g) + h(g) = c(n,g) + 0 = c(n,g) = h^*(n).$$

Therefore, $h(n) \leq h^*(n)$, which shows that h behaves admissibly in the base case.
Induction step: Consider an arbitrary node n. If n' is a successor of n, then our inductive hypothesis is assuming consistency holds between n and n' (i.e: $h(n) \leq c(n,n') + h(n')$), h is admissible on n' (i.e: $h(n') \leq h^*(n')$). We want to prove that $h(n) \leq h^*(n)$.

From the inductive step we have $h(n) \leq c(n,n') + h(n')$ and $h(n') \leq h^*(n')$. Therefore,

$$h(n) \leq c(n,n') + h^*(n') \tag{1}$$

Let $S(n)$ be the set of all the successors of n, and $h^*(n)$ be the optimal cost to reach g from n. So,

$$h^*(n) = \min_{n' \in S(n)} \{c(n,n') + h^*(n')\}. \tag{2}$$

From (1) and (2), we see that $h(n) \leq h^*(n)$ and we are done. □

Prove that admissibility does not necessarily imply consistency.

Proof. This can be proved using a counter example. Consider a graph which consists of a single path with 10 nodes: $(n_0, n_1, n_2, \ldots, n_9)$, where the goal is n_9. Let us assume wlog that all edge costs are equal to 1. Obviously $h^*(n_0) = 9$, and let us make $h(n_0) = 8, h(n_i) = 1, 1 \leq i < 9$ and $h(n_9) = 0$. Clearly, the heuristic function is addmissible because:

1. $h(g) = h(n_9) = 0$
2. $h(n_i) = 1 \leq h^*(n_i) = (9-i), \forall i, 1 \leq i < 9$
3. Finally, $h(n_0) = 8 \leq h^*(n_0) = 9$

However, $h(n)$ is not consistent because,

$$h(n_0) = 8 > c(n_0, n_1) + h(n_1) = 1 + 1 = 2.$$

□

Algorithm	Complete?	Optimal?	Time	Space
BFS	Yes	Not Really	$\Theta(b^d)$	$\Theta(b^d)$
UCS	Sort Of	Yes	$\Theta(b^{\frac{C^*}{\epsilon}})$	$\Theta(b^{\frac{C^*}{\epsilon}})$
DFS	No	No	$\Theta(b^m)$	$\Theta(bm)$
IDS	Yes	No	$\Theta(b^d)$	$\Theta(bd)$

Here, b = the branching factor, d = distance from the start node, C^* = the cost of an optimal solution, ϵ = the cost per action, and m = the maximum depth of the state space.

Completeness: An algorithm is complete if it is guaranteed to find a solution whenever one exists.

Convex Optimization

Convex Sets

- The intersection of any collection convex sets is a convex set
- The union of convex sets is not necessarily convex
- A set is convex if and only if its intersection with any line is convex

Example: Prove that the following sets are convex, or show why they are not.

1. All $x \in \mathbb{R}^n$ such that $Ax \leq b$ and $Cx = d$.
   Proof: For some x, y such that $Ax \leq b$ and $Ay \leq b$, we have that $A(\theta x) \leq \theta b$ and $A(1-\theta)x \leq (1-\theta)b$. Then $A(\theta x) + A(1-\theta)y \leq \theta b + (1-\theta)b = b$. We can pull out the A from the left side to get $A(\theta x + (1-\theta)y \leq b$. Using the same logic, we can show that if $Cx = d$ and $Cy = d$, $C(\theta x + (1-\theta)y) = d$. So by the definition of convexity, both conditions are satisfied. ■
2. The two dimensional ball defined by $x^2 + y^2 \leq 4$ for $(x,y) \in \mathbb{R}^2$.
   Proof: Let (x,y) and (u,v) be two points in the ball. Then $x^2+y^2 \leq 4$ and $u^2+v^2 \leq 4$. Then, $\theta(x,y) = (\theta x, \theta y)$ and $(1-\theta)(u,v) = ((1-\theta)u, (1-\theta)v)$. Because $\theta, (1-\theta) \leq 1$, $\theta^2x^2+\theta^2y^2 \leq 4\theta$ and $(1-\theta)^2u^2+(1-\theta^2)v^2 \leq 4(1-\theta)$. So $\theta^2x^2+\theta^2y^2+(1-\theta)^2u^2+(1-\theta^2)v^2 \leq 4\theta + 4(1-\theta) = 4$. ■
3. The portion of the ball defined by $x^2 + y^2 \leq 4$ and $x^2 + y^2 \geq 1$.
   We can see that this is not a convex set through a simple geometric argument: the area of this ball is basically the shape of a donut. You cannot draw a line connecting two points directly across the donut hole from each other

Convex Functions

Definition: A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if and only if for any $x, y \in \mathbb{R}^n$ and $\theta \in [0,1]$,

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y).$$

For functions f that are twice differentiable, $f''(x) \geq 0$ for all $x \in \mathbb{R}$.

Convex Optimization Problem

Definition: A *convex optimization problem* is a specialization of a general optimizatio problem $\min_{\mathbf{x}} f(\mathbf{x})$ such that $\mathbf{x} \in \mathcal{F}$ where the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is a *conve function*, and the feasible region $\mathcal{F}$ is a *convex set.*

Example: (Linear Programming) The linear programming problem can be formulated as finding

$$\min_{\mathrm{x}} \mathrm{c}^T\mathrm{x} \quad \text{such that } A\mathrm{x} = \mathrm{a} \quad \text{and } B\mathrm{x} \leq \mathrm{b},$$

where $\mathrm{x} \in \mathbb{R}^n$ is the optimization variable, and $\mathrm{c} \in \mathbb{R}^n, A \in \mathbb{R}^{m \times n}, \mathrm{a} \in \mathbb{R}^m, B \in \mathbb{R}^{k \times n}, \mathrm{b} \in \mathbb{R}^k$ are the problem data.

Verbally, we are trying to minimize a (convex) linear objective function subject to linear constraints (which gives us a convex set). Taking the dot product of x with c gives us our objective function evaluated at an input of x, the pairing A and a encode a set of equality constraints and the pairing B and b encode a set of inequality constraints.
Example: (Linear Programming) Consider the example of a manufacturer of animal feed who is producing feed mix for dairy cattle. In our simple example the feed mix contains two active ingredients and a filler to provide bulk. One kg of feed mix must contain a minimum quantity of each of four nutrients as below:

Nutrient	A	B	C	D
gram	90	50	20	2

The ingredients have the following nutrient values and cost:

	A	B	C	D	Cost/kg
Ingredient 1 (gram/kg)	100	80	40	10	40
Ingredient 2 (gram/kg)	200	150	20	0	60

What should be the amounts of active ingredients and filler in one kg of feed mix? Model this as an LP.

Let x_1 and x_2 be the number of kilograms of ingredients 1 and 2 respectively that we include in the feed. Then, assuming that we want to minimize cost, the appropriate LP for this situation is $\min_{x_1, x_2} 40x_1 + 60x_2$ such that:

$$\begin{aligned} 100x_1 + 200x_2 &\geq 90 \\ 80x_1 + 150x_2 &\geq 50 \\ 40x_1 + 20x_2 &\geq 20 \\ 10x_1 &\geq 2 \\ x_1 + x_2 &\leq 1 \\ x_1, x_2 &\geq 0 \end{aligned}$$

Game Theory

Normal Form Game

The normal form of a game is defined as:

- Set of players $N = \{1, \ldots, n\}$
- Strategy set S: The strategies that the players can take
- Utility function of player I : $u_I : S^n \to R$,, the utility player i will have when each $j \in N$ plays the strategy $s_j \in S$.

Dominance: Refers to a strategy $s = (s_1, \ldots, s_n)$ where each player i chooses a strategy s_i that maximizes their utility u_i regardless of the strategy s_j used by any other player $j \in N \ \{i\}$.

Nash Equilibrium

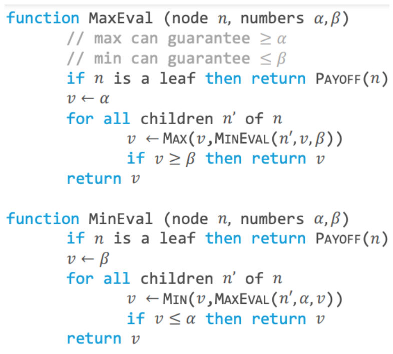
Definition 5 *A vector of strategies $\boldsymbol{s} = (s_1, \ldots, s_n) \in S^n$ such that for all $i \in N$ and for all $s'_i \in S$ the following is true: $u_i(\boldsymbol{s}) \geq u_i(s_1, \ldots, s_{i-1}, s'_i, s_{i+1}, \ldots, s_n)$*

In other words, in a Nash equilibrium, no player wants to unilaterally deviate from their current strategy.

Mixed Strategies

A mixed strategy is a probability distribution over (pure) strategies.

Minimax and α-β Pruning



```
function MaxEval (node $n$, numbers $\alpha,\beta$)
        // max can guarantee $\geq \alpha$
        // min can guarantee $\leq \beta$
        if $n$ is a leaf then return PAYOFF($n$)
        $v \leftarrow \alpha$
        for all children $n'$ of $n$
                $v \leftarrow$ MAX($v$,MINEVAL($n', v, \beta$))
                if $v \geq \beta$ then return $v$
        return $v$

function MinEval (node $n$, numbers $\alpha,\beta$)
        if $n$ is a leaf then return PAYOFF($n$)
        $v \leftarrow \beta$
        for all children $n'$ of $n$
                $v \leftarrow$ MIN($v$,MAXEVAL($n', \alpha, v$))
                if $v \leq \alpha$ then return $v$
        return $v$
```

The α-β pruning is an improvement over the regular minimax algorithm. Here, we have 2 additional arguments, α, β. Note that the statement `if` $v \geq \beta$ `then return` v is an additional one in `MaxEval` for alpha-beta pruning. Similarly, the statement `if` $v \leq$ `then return` v is an additional one in `MinEval` for alpha-beta pruning.

Stackelberg Security Games

Unlike a Normal form game, Stackelberg is a form where the leader must commit to the strategy and the follower can observe before committing to a strategy. During computation, we can always break ties in favor of the leader. This means that if the utility for the follower is the same, we can choose the follower's strategy that would result in a higher utility for the leader. The game will play out according to the following steps:

1. The leader will commit to some strategy
2. The follower will observe the leader's selected strategy (not necessarily knowing the move the leader makes if he plays with a mixed strategy)
3. Given the observations they make in step 2, the follower will commit to a strategy

Example: Consider the following Stackelberg game where the row player is the leader:

$(1,1)$	$(3,0)$
$(0,0)$	$(2,1)$

We can find the maximum utility for the leader when they commit to the strategy (0.5, 0.5). Given that we break ties in favor of the leader and our follower has an equal probability of reaching their maximum utility from choosing either column, our follower will commit to the right column. Given this commitment, we can find the total expected utility for the leader by summing up our utility options multiplied by the probability that each option occurs:

$$(.5)(3) + (.5)(2) = 1.5 + 1 = 2.5$$

which is greater than the utility for the leader at our Nash equilibrium point of (1, 1).
Let the leader of our Stackelberg game play with a mixed strategy called x_1. We can define the set of best response strategies, or the set of pure strategies that player 2 can play that maximize the utility of the follower, that the follower can use as $B_2(x_1)$.

$$B_2(x_1) = \text{argmax}_{S_2 \in S} u_2(x_1, s_2)$$

Where s_2 represents the selected follower strategy, S represents the set of strategies, S_2 represents the set of pure strategies that player 2 can play, and $u_2(x_1, s_2)$ represents the utility for the follower given that the leader uses strategy x_1 and the follower uses the strategy s_2.
Example 2: Consider the following game:

$(3,1)$	$(5,0)$
$(2,0)$	$(4,2)$

Find the nash equilibrium for this game and find the strategy that the leader can play to maximize their utility in a stackelberg game where the row player is the leader.
We first find the nash equilibrium for this game. For our row player, we see that the top row is a dominant strategy, meaning that regardless of what the column player will do, the row player will always be betterr off by choosing the top row. Given this information, our column player can maximize his utility by choosing the column who's top row value is greater. $1 > 0$, meaning that the column player will choose the left column. Therefore, our nash equilibrium exists at (3, 1).

Now, we want to find the strategy that the leader can play to maximize their utility in a stackelberg game. Looking at our game space, we know that we can maximize the payoff for the leader if we can convince the column player to play the right column as often as possible while still allowing the (5,0) option to be a possibility. If the leader plays the strategy $(\frac{2}{3}, \frac{1}{3})$, our column player will respond by choosing the right column every time because we break ties in favor of the leader, as the expected payoff for the follower will be $1 \times \frac{2}{3}$ for the left column and $2 \times \frac{1}{3}$ for the right column. Given this setup, the leader will earn a utility of $5\frac{2}{3}$ of the time and a utility of $4\frac{1}{3}$ of the time, giving the leader an expected utility of $\frac{10}{3} + \frac{4}{3} = 4.67$. Therefore, the leader can maximize their utility in a stackelberg game with strategy $(\frac{2}{3}, \frac{1}{3})$.

Behavioral Game Theory

Example: You're participating in a game show, and after completing several challenges, you're presented with three final options for your prize:

1. Option A: A guaranteed prize of \$10,000.
2. Option B: A 95% chance of receiving \$10,500 and a 5% chance of receiving \$0.
3. Option C: A 2% chance of receiving \$500,000 and a 98% chance of receiving \$0.

Calculate the expected value of each option and then determine which option an individual might prefer for 1. A risk-neutral profile, 2. a risk-seeking profile, and 3. a risk-averse profile.

1. The expected value (EV) of Option A: E(A) = $\$10,000$
2. The expected value (EV) of Option B: E(B) = $0.95 \cdot \$10,500 = \$9,975$
3. The expected value (EV) of Option C: E(C) = $0.02 \cdot \$500,000 = \$10,000$

Given the choices: (1) Option A guarantees \$10,000. (2) Option B has an expected value of \$9,975. (3) Option C has an expected value of \$10,000.
A risk-neutral person, might be indifferent between Options A and C since they have the same expected value. A risk-averse person would unquestionably choose Option A since there is no uncertainty. A risk-seeking person would pick Option C. Despite having the same expected value as Option A, utility of potentially winning \$500,000 - even with a very slim chance. For review, remember the graphs of utility for these profiles. Risk-neutral tends to increase linearly with the true value. Risk-seeking tends to increase faster than linear with respect to true value. And Risk-adverse tends to level off below the risk-neutral line.

Markov Decision Process

MDP Construction: Markov Decision Processes (MDP) construction:

- S: a set of states
- s_0: initial state
- $A(s)$: A set of actions for each state $s \in S$
- $P(s'|s,a)$: A transition model that gives the probability of reaching state s' if action a is taken from state s
- $R(s)$: A reward function that gives us the reward for state s

Example: Angela is a investor, and she's in a casino with a goal to win \$50. She currently has \$10 in her pocket. Angela can place bets in multiples of \$5, and the casino allows bets from \$5 to \$25. The game works as follows:

- If Angela places a bet and wins, she earns double the amount. So if she bets \$5 and wins, she has \$20 in her pocket.
- If Angela places a bet and loses, she loses the amount she bet. So if she bets \$5 and loses, she has \$5 in her pocket. Assume Angela cannot have negative money.
- Angela wins a bet with probability 3/5.

Formulate this problem as an MDP.

- State: There is a state representing each possible amount of money Angela has (i.e. multiples of five between \$0 and \$50).
- Action: The amount of money Angela bets, from \$5 to \$25, in increments of five.
- Transition Function: $P(s'|s,a) = 3/5$ if double the bet amount associated with a increases Angela's money from s to s'. $P(s'|s,a) = 2/5$ if the bet amount associated with a decreases Angela's money from s to s'. All other transition probabilities are zero.
- Reward: Since the goal is to win \$50, we can initialize the state associated with \$50 to have a reward of 1 and all other states to have a reward of 0.

Bellman Equations

For any state s, we can find the utility $U(s)$ given by the optimal policy using the Bellman equations:

$$U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a) * U(s')$$

Once we have found $U(s)$ for all $s \in S$, we can find the optimal policy for any state S:

$$\pi^*(s) \in \arg\max_{a \in A(s)} \sum_{s'} P(s'|s,a) * U(s')$$

Value Iteration: In the Bellman equations above, we see that the optimal utility $U(s)$ depend on the optimal utility for other states $U(s')$. How can we compute the optimal utility and find our optimal policy? To do so, we start with an initial function U and iteratively update our estimates for the optima utility until we reach a convergence condition:

$$U_{i+1}(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a) * U_i(s')$$

Given some small $\epsilon > 0$, we define the stopping condition to be:

$$\max_{s \in S} |U_{i+1}(s) - U_i(s)| < \frac{\epsilon(1-\gamma)}{\gamma}$$

Policy Iteration: Beginning with initial policy π_0:
Step 1: Policy evaluation (until convergence) Given policy π_I we can calculate its utility U_i as:

$$U_i(s) = R(s) + \gamma \sum_{s'} P(s'|s, \pi_i(s)) \times U_i(s')$$

Step 2: Policy improvement (one step) Using this U_i, calculate an updated policy π_{I+1}:

$$\pi_{i+1}(s) \in \arg\max_{a \in A(s)} \sum_{s'} P(s'|s,a) \times U_i(s')$$

We repeat the above steps until there is no change in utilities.

Question 1: What are the similarities and differences between policy iteration and value iteration?
Both are based off of the Bellman Equations and have a convergence condition. Value iteration finds the optimal utility, while Policy Iteration computes the optimal policy directly.
Question 2: Explain why Policy Iteration is guaranteed to converge. Intuitively, policy iteration is guaranteed to converge because at every iteration (before convergence) your policy can only improve by definition. This means that we will never encounter the same policy twice. The total number of candidate policies is $(\#\text{actions})^{\#\text{states}}$. After this many iterations, you are guaranteed to have converged.
Question 3: Prove that policy iteration converges at the optimal policy and value function
Let π_k be the policy that we have after convergence. Then, by definition, $\pi_k(s) = \pi_{k+1}(s), \forall s \in S$. In addition, if we were to evaluate our policy π_k, our utility U_k would be:

$$U_k(s) = R(s) + \gamma \sum_{s'} P(s'|s, \pi_k(s)) * U_k(s')$$

Because policy iteration has converged, the action, a, that π_k will take at state s is the one that maximizes $\sum_{s'} P(s'|s,a) \times U_k(s')$. Otherwise, policy improvement would have found this action.

Restless Bandits

In the restless bandits model, there are N arms, each of which is an individual MDP. We choose K arms to pull, and the actions influence the state of each MDP per its transition model.

- Total state space: $|S|^N$ where each arm has $|S|$ states
- Total action space: $\binom{N}{K}$.