CS3243 Tutorial 2

Eric Han

Feburary 3, 2022

Annoucements

Important admin:

1. Attendance Marking via CAS:
   - 2120/CS3243/T14 03-02-2022 13:00-14:00
   - 2120/CS3243/T15 03-02-2022 15:00-16:00
   - 2120/CS3243/T16 03-02-2022 16:00-17:00
2. Assignment 1 scores and comments are out on turnitin, do check!

Questions from tutorials

1. What is the type of proof for Q4?
   - See wiki https://en.wikipedia.org/wiki/Mathematical_proof link to see the different math proofs. I would consider Q4 to be proof by construction.

Introduction

Eric Han

4th Year PhD Student

I am interested in scaling machine learning towards higher dimensions in Bayesian Optimization, Gaussian Processes, Convex and non-convex optimization and Reinforcement Learning.

Feel free to telegram me `@Eric_Vader`/on our group:

1. regarding the module,
2. research,
3. grad sch,
4. NUS
5. etc. . .

Expectations

Expectations of you

1. Fill the seats from the front.
2. Come prepared to the Tutorial.
   2.1 You will be asked at random to discuss your answer.
3. Refrain from taking pictures of the slides.
   3.1 Learn to take good notes.
   3.2 The slides will *not* be distributed.
4. Try to be on time.

Any comments or suggestions for the lessons going forward?

Question 1

a. Provide a counter-example to show that the tree-based implementation for the **Greedy Best-First Search** algorithm is incomplete.
b. Briefly explain why the limited-graph-based implementation of the **Greedy Best-First Search** algorithm is complete.
c. Provide a counter-example to show that neither the tree-based nor the limited-graph-based implementations of the **Greedy Best-First Search** algorithm are optimal.

Recap

- What is Greedy-Best-First Search (GBFS) algorithm?
- What does is - incomplete?
  - Incomplete describes the algorithm being stuck in a loop where $h(.)$ are lowest.

Question 1a - Answer

s	$S0$	$S1$	$S2$	G
$h(s)$	3	4	5	0

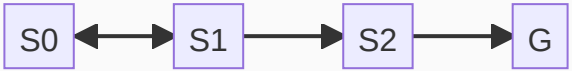


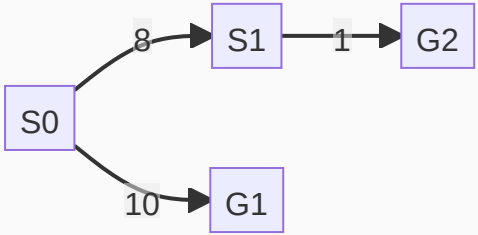
Figure 1: GBFS Infinite Loop (Incomplete) Example

1. $S0$ is explored, $S1$ is added to front of frontier.
2. $S1$ is explored, $S0$ is added to the front of frontier.
3. etc...

Question 1b - Answer

- Limited-graph-based implementation of GBFS will not explore redundant paths.
- Assuming a finite search space, there are finite paths.
- Hence, limited-graph-based implementation of GBFS will eventually visit all states within the search space and find a goal state.

Question 1c - Answer



s	$S0$	$S1$	$G1$	$G2$
$h(s)$	9	1	0	0

Figure 2: Non-Optimal Example

For both implementations:

1. $S0$ is explored, $G1$ is added to front of frontier - $f(.) = h(.)$
2. Non-optimal path $S0 \rightarrow G1$ returned.

Question 2

a. Prove that the tree-based variant of the **A* Search** algorithm is optimal when an **admissible heuristic** is utilised.
b. Prove that the limited-graph-based variant of the **A* Search** algorithm is optimal when a **consistent heuristic** is utilised.

Recap

- What is the intuition behind A* Search?
- What is an admissible heuristic?
- What is a consistent heuristic?

Recap

- What is the intuition behind A* Search?
- What is an admissible heuristic?
- What is a consistent heuristic?

Summary

- A* Search: $f(N) = g(N) + h(N)$
- Admissible heuristic: $h(N) \leq h^*(N)$
- Consistent heuristic: $h(N) \leq c(N, N') + h(N')$

Where,

- $f(.)$ - Evaluation Function.
- $g(.)$ - Cost Function from start to current node.
- $h(.)$ - Estimated cost from current node to goal.
- $c(N, N')$ - Action cost from N to N'.

Question 2a - Answer

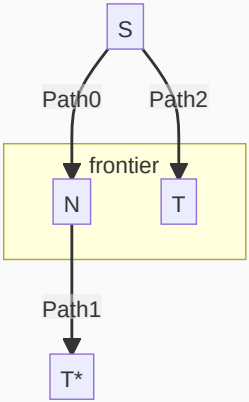


Figure 3: Illustration

Where:

- S - Inital State
- N - Intermediate State along optimal Path0, Path1
- T - Sub Optimal Goal state along sub-optimal Path2
- T^* - Goal state along optimal Path0, Path1

Proof by Contradiction:

1. Assume sub-optimal T is found instead of the T^*
2. $\implies$ T is found before $N \rightarrow f(T) < f(N)$
3. Since T^* is optimal, $f(T) > f(T^*)$
   - 3.1 $f(T) > f(T^*)$
   - 3.2 $f(T) > g(T^*) + h(T^*) \implies f(T) > g(T^*)$
   - 3.3 $f(T) > g(N) + h^*(N) \implies f(T) > g(N) + h(N)$
   - 3.4 $\implies f(T) > f(N)$
4. Since Tree-Search would have explored $T^*, \perp f(T) < f(N)$

Question 2b - Answer

Intuition: Show that when a node is popped from frontier, optimal path to it is found.

Let the optimal path from start node s_0 to any node s_g be $P_{0\to g} = s_0, s_1, \cdots, s_g$. We show that when s_g is popped, $f(s_g) = g(s_g) + h(s_g) = g^*(s_g) + h(s_g)$ where $g^*(s_g)$ is the optimal path cost over $P_{0\to g}$ using induction.

- **Base**: $f(s_0) = g(s_0) + h(s_0) = h(s_0)$.
- **Inductive**: Assume for $P_{0\to k}$, $\implies g(s_k) = g^*(s_k)$
  - $g^*(s_{k+1})$ is the minimum path cost, $\implies g(s_{k+1}) \geq g^*(s_{k+1})$
  - Since consistent, $h(s_k) \leq c(s_k, s_{k+1}) + h(s_{k+1}) \implies s_k$ is popped.
  - $f(s_{k+1}) \leq g(s_k) + c(s_k, s_{k+1}) + h(s_{k+1})$
  - $\implies g(s_{k+1}) + h(s_{k+1}) \leq g(s_k) + c(s_k, s_{k+1}) + h(s_{k+1})$
  - $\implies g(s_{k+1}) \leq g(s_k) + c(s_k, s_{k+1}) = g^*(s_k) + c(s_k, s_{k+1}) = g^*(s_{k+1})$
  - $\implies g(s_{k+1}) \leq g^*(s_{k+1})$
  - $g(s_{k+1}) \leq g^*(s_{k+1}) \wedge g(s_{k+1}) \geq g^*(s_{k+1}) \implies g(s_{k+1}) = g^*(s_{k+1})$

Question 3

a. Given that a heuristic h is such that $h(t) = 0$, where t is any goal state, prove that if h is consistent, then it must be admissible.
b. Give an example of an admissible heuristic that is not consistent.

Recap

- Admissible heuristic: $h(N) \leq h^*(N)$
- Consistent heuristic: $h(N) \leq c(N, N') + h(N')$

Question 3a - Answer

Intuition: Show on the number of action required to reach the goal from n to goal t.

Let the no. of actions $k(n)$ to be required to reach from n to t; Node n_k is k steps away from t, ie. $k = 3 \implies P_{n_3 \to t} : n_3 \to n_2 \to n_1 \to t$.

- **Base**: 1 action; i.e. node n_1 is one step away from t.
  - Since consistent, $h(n_1) \leq c(n_1, t) + h(t)$ and $h(t) = 0$
  - $\implies h(n_1) \leq c(n_1, t) = h^*(n_1)$
- **Inductive**: Assume for $k-1$ actions, path $P_{n_{k-1} \to t}, h(n_{k-1}) \leq h^*(n_{k-1})$.
  - Since consistent, $h(n_k) \leq c(n_k, n_{k-1}) + h(n_{k-1})$
  - $\implies h(n_k) \leq c(n_k, n_{k-1}) + h^*(n_{k-1}) = h^*(n_k)$.

Question 3b - Answer

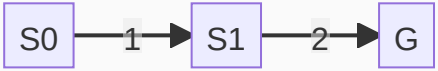


Figure 4: Example

s	$S0$	$S1$	G
$h(s)$	3	1	0
$h^*(s)$	3	2	0
$h(s) \leq h^*(s)$	T	T	T

Heuristic h is

- Admissible - $\forall s : h(s) \leq h^*(s)$
- Not consistent - $3 = h(s_0) > c(s_0, s_1) + h(s_1) = 2$

Question 4

We have seen various search strategies in class, and analyzed their worst-case running time. Prove that any deterministic search algorithm will, in the worst case, search the entire state space. More formally, prove the following theorem

Theorem 1. Let $\mathcal{A}$ be some complete, deterministic search algorithm. Then for any search problem defined by a finite connected graph $G = \langle V, E \rangle$ (where V is the set of possible states and E are the transition edges between them), there exists a choice of start node s_0 and goal node g so that $\mathcal{A}$ searches through the entire graph G.

Question 4 - Answer

Intuition: We try to find a goal node g such that $\mathcal{A}$ searches all of G.

In the t-th iteration (Count only iterations where $\mathcal{A}$ explores a new node),

- Let g_t to be the goal node chosen only in t, to be $U_{t-1} \setminus U_t$
- Let U_t be the set of unsearched nodes, $V = U_0$.

1. Since there is finite $|V|$ and $\mathcal{A}$ is complete, then $U_0 \subset U_1 \subset U_2 \subset \cdots \subset U_{|V|} = \emptyset$
2. $\mathcal{A}$ terminates on the k-th iteration with search sequence S_k, finding goal node g_k.
3. Since $\mathcal{A}$ is deterministic, when we reset the goal node to be any in U_{k+1} (worse case to be g_{k+1}), it will take the same search route $S_k + [g_{k+1}]$ to reach the new goal.

Hence, we can always choose $g_{|V|}$ to be the goal node, such that $\mathcal{A}$ searches all of G.

Question 5

a. In the search problem below, we have listed 5 heuristics. Indicate whether each heuristic is admissible and/or consistent in the table below.

b. Write out the order of the nodes that are explored by the **A* Search** algorithm. Assume a limited-graph-based implementation that utilises heuristic h_4.

c. Which heuristic would you use? Explain why.

d. Prove or disprove the following statement:

   The heuristic $h(n) = max\{h3(n), h5(n)\}$ is admissible.

Question 5 - Answer

Question 5a

s	S	A	B	G	Admissible	Consistent
$h_1(s)$	0	0	0	0	T	T
$h_2(s)$	8	1	1	0	T	F
$h_3(s)$	9	3	2	0	T	T
$h_4(s)$	6	3	1	0	T	F
$h_5(s)$	8	4	2	0	F	F
$\max\{h_3(s), h_5(s)\}$	9	4	2	0	F	F

Question 5b

$S - A - B - G$

Question 5c

h_3, as $h_3 = h^*$ is the optimal heuristic.

Question 5d

$4 = h(A) > h^*(A) = 3$