Techniques of Artificial Intelligence

Exercises – Search Space & Concept Learning

Dipankar Sengupta
Dipankar.Sengupta@vub.ac.be

Roxana Rădulescu
Roxana.Radulescu@vub.ac.be

February 29, 2016

Fill in your contact information to receive announcements and materials for the course in the form: `http://goo.gl/forms/uNlWXFL8dT`

4. **The Missionaries and Cannibals Puzzle**
   There are three missionaries and three cannibals on the west bank of a river. There is a boat on the west bank that can hold no more than two people. The missionaries wish to cross to the east bank. But they have a problem: If on either bank the cannibals ever outnumber the missionaries, the outnumbered missionaries will be eaten. Is there a way for the missionaries to get to the east bank without losing anyone?

   (a) Represent this puzzle as a search problem by defining the states and actions.

   (b) Investigate your initial representation of the puzzle as you did in part (a): is there any redundant information in your representation? Try to remove any redundant information from it.

   (c) How many different states are there?

   (d) Try to solve the puzzle yourself.

   Answer:

   (a) A straightforward approach would be to represent the state by an array containing the following:

   - number of missionaries at the left
   - number of cannibals at the left
   - number of missionaries at the right
   - number of cannibals at the right
   - number of missionaries in the boat
   - number of cannibals in the boat
   - position of the boat

   (b) This representation is overcomplete: if you know how many missionaries and cannibals we have at the one bank and in the boat, we know how many there are at the other bank, as we always have 3 missionaries and 3 cannibals. So, a first improvement might be the following representation:

   - number of missionaries at the left
   - number of cannibals at the left
   - number of missionaries in the boat
   - number of cannibals in the boat
   - position of the boat

However, as the boat can contain at most 1 missionary and 1 cannibal, the cannibals can never outnumber the missionaries in the boat, so it is of no use to represent the number of each in the boat. So, we could simply store:

- number of missionaries at the left
- number of cannibals at the left
- position of the boat

For instance $< 3, 3, 1 >$ represent the initial state, with 3 missionaries and 3 cannibals at the left bank, with the boat at the left bank. The state$< 3, 1, 0 >$ can then be reached from this state, by the action“transfer two cannibals to the right bank”. As a result, we have 3 missionaries at the left bank, one cannibal at the left bank and the boat at the right bank.

(c) A state is represented by $< x, y, z >$, with x varying between 0 and 3, y varying between 0 and 3 and z being either 0 or 1. So, we have $4 \times 4 \times 2 = 32$ states at most.

For the boat being at left, we can for instance list all 16 possible states: 3 3 1 / 3 2 1 / 3 1 1 / 3 0 1 / 2 3 1 / 2 2 1 / 2 1 1 / 2 0 1 / 1 3 1 / 1 2 1 / 1 1 1 / 1 0 1 / 0 3 1 / 0 2 1/ 0 1 1 / 0 0 1.

From this list, we can eliminate the states 2 3 1 / 2 1 1 / 2 0 1 / 1 3 1 / 1 2 1 / 1 0 1 because we have more cannibals than missionaries at one bank. The same holds of course for the boat being at the other side. So we have $3266 = 20$ states.

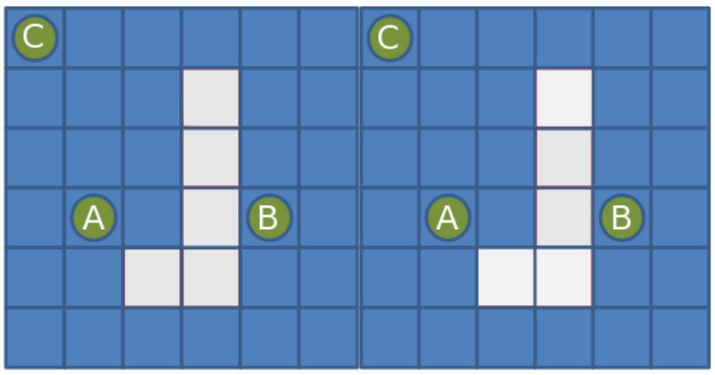
The states $< 0, 0, 1 >$ and $< 3, 3, 0 >$ are not valid (because it implies that you move an empty boat), the states $< 3, 0, 1 >$ and $< 0, 3, 0 >$ are not reachable, because all preceding states are invalid: $< 3, 0, 1 >$ can only be reached from $< 2, 0, 0 >$ or $< 1, 0, 0 >$. However, in both states the cannibals do outnumber the missionaries. In total we have hence 16 states.

(d) One solution is the following:

$< 3, 3, 1 >$, $< 3, 1, 0 >$, $< 3, 2, 1 >$, $< 3, 0, 0 >$, $< 3, 1, 1 >$, $< 1, 1, 0 >$, $< 2, 2, 1 >$, $< 0, 2, 0 >$, $< 0, 3, 1 >$, $< 0, 1, 0 >$, $< 0, 2, 1 >$, $< 0, 0, 0 >$

5. **Path planning**

Imagine a simple blocks world as shown below. A block can be moved a single cell at a time in each of the four directions. We have to find the shortest path from configuration A to configuration B.



(a) Discuss whether depth first search and breadth first search will always find a path from A to B. Will they find the shortest path?

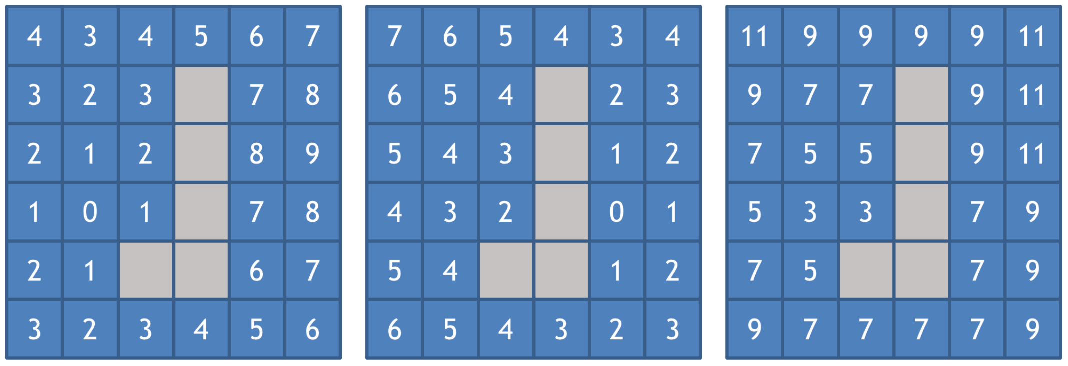
(b) Suppose that we want to use informed search in order to guide our search. We will use the Manhattan distance as a search heuristic. Discuss whether best first search will find the shortest path, use the left drawing to add f, g and h costs. Will A* do so? Use the right drawing to add f, g and h costs.

(c) Now suppose that there is a direct underground connection from C to B. Hence, the shortest path from A to B now is over C (it is only five steps). Argue whether A* will find this shortest path.

Answer:

(a) If it is ensured that the algorithms do not get stuck in loops, then they will find a solution. Depth first search will return the first path to the goal state it happens to encounter, which is not guaranteed to be the shortest one. Breadth first search, will find the shortest path.

(b) In this particular case, we can easily see that there are actually two possible ways of going from A to B in an efficient manner, either travel above the obstacle or travel below the obstacle. If you calculate the f-cost of every cell in the grid, it is observed that both paths have exactly the same f-cost when using the best first search. However, the lower path is shorter. So, with best first search, it is not guaranteed to find the shortest path. With A* the shortest path is found, because the heuristic is admissible.



4	3	4	5	6	7
3	2	3	▨	7	8
2	1	2	▨	8	9
1	0	1	▨	7	8
2	1	▨	▨	6	7
3	2	3	4	5	6

g(n) (for shortest path to n)

7	6	5	4	3	4
6	5	4	▨	2	3
5	4	3	▨	1	2
4	3	2	▨	0	1
5	4	▨	▨	1	2
6	5	4	3	2	3

h(n)

11	9	9	9	9	11
9	7	7	▨	9	11
7	5	5	▨	9	11
5	3	3	▨	7	9
7	5	▨	▨	7	9
9	7	7	7	7	9

f(n)

(c) The shortcut through the underground connection is not reflected in the A* heuristic. At position C, the f-cost will be 11 (g = 4, h = 7). So, our heuristic overestimates the cost to the goal, thus the heuristic is not admissible. A* will prefer the solution found in B.

6. The optimality condition of A* is expressed as follows: “If the heuristic h(n) is admissible, then the solution returned by A* is optimal”. In order to prove this, we assume that the optimal solution n* has an optimal cost f(n*) = C*.

   (a) Suppose that there is a solution G2 which is not optimal in the agenda. What does this learn about f(G2)?

   (b) Suppose that there is a node n in the agenda which is on the path to the optimal solution G. What does this learn about f(n)?

   (c) Proof that if h(n) is admissible, then the solution returned by A* is optimal.

   Answer:

   A heuristic is admissible if it never overestimates the cost to the goal.

   (a) Suppose A* returns a solution which is not optimal. This means that at some point a non optimal solution G2 must be in the agenda. If G2 is not optimal, then $f(G2) > f(n*) = C*$

   (b) If h(n) is admissible, then h does not overestimate the cost to the goal, so we can put $f(n) < C*$

(c) Combining a and b results in the inequality: $f(n) < C* < f(G2)$. Hence, every node on the path towards the optimal solution has a cost less than the cost of the non optimal solution G2. Hence, G2 will never be expanded from the priority queue. So, G2 cannot be the solution of the A* algorithm.

7. A new game comes with a board which contains 7 cells (as shown below). There are 6 pivots, each containing a number. Each pivot has to be in one cell and each cell can contain at most one pivot. Hence, there is always one empty cell. A pivot can be moved one cell to the left or one cell to the right (if it does not break the rules given above) with a cost of 1. A pivot can also jump over another pivot, with cost 2 (if it does not break the rules given above). The goal of the game is to have the empty cell in the middle, such that the sum of the blocks at the left of the empty cell is equal to the sum of the blocks at the right of the empty cell.

8	1	5		2	3	7

(a) Model this as a state space search problem.

(b) Which of the search algorithms studied in class is best suited for solving this puzzle? Motivate (max 5 lines).

(c) Give a sequence of states leading to the goal state.

Answer:

(a) The state can be modelled by taking the vector of the board, that encodes the position of each element and of the white space, along. From this encoding we can extract the position of the empty cell (to check if it is in the middle) and also if the state is a goal state or not (by computing the two sums). For example, for the starting position the state could be $< 8, 1, 5, -, 2, 3, 7 >$. At each step there are 4 available actions: swapping the empty cell with the first (cost 1) or the second (cost 2) cell on the left and the same for the right direction. If we consider the graph search version, the branching factor is at most 3, except for the expansion of the initial state, which is 4.

(b) There are multiple goal states for the problem, as we can split the elements in any combination that outputs the same sum for both sides, while the order of the elements inside the groups does not matter: $< 1, 7, 5, -, 8, 3, 2 >, < 7, 1, 5, -, 8, 2, 3 >, ...$ A simple uninformed search approach for the problem can be UCS, as we do not have a uniform cost across our search space. It is not straightforward to apply informed search approaches for this case: designing a heuristic (admissible for A*) proves to be hard as there is not clear connection between the performed actions and their effect on the sum balance. However a nice intuition can be given by the difference between the left and right sums. If $left - right$ is positive that means we have an excess of elements in the left side so for sure the empty cell should move to the left. This could be used to prune the search tree during the search process.

(c) 815-237 (0), 81-5237 (1), 8125-37 (3), 81253-7 (4), 812-357 (6), 8-21357 (8), 82-1357 (9), 821-357 (10), 8213-57 (11), 82-3157 (13), 823-157 (14) - 10 steps, cost 14

8. **Calculating the size of the hypothesis space** (based on exercise 2.1 from the course book)
Suppose there are m attributes in a learning task and that every attribute i can take k_i possible values. What will be the size of the hypothesis space?

Answer:

Syntactically different: $\prod_{i=1}^{m}(k_i + 2)$

Semantically different: $1 + \prod_{i=1}^{m}(k_i + 1)$

9. **Order of training instances**
In candidate elimination, suppose you have n training instances $T_1 \ldots T_n$. After the n_{th} training instance, candidate elimination learned the boundaries S and G. Will S and G differ or not when providing the training instances in reverse order: $T_n \ldots T_1$? Explain why (not).

   Answer: The concept of version space aims at invariance to instance order, keeping not a single concept description but a set of possible descriptions that evolves as new instances are presented.

10. What is the version space while tracing the **candidate elimination algorithm** with the following examples?

    $Architecture \in \{Gothic, Romanesque\}$

    $Size \in \{Small, Large\}$

    $Steeples \in \{Zero, One, Two\}$

    Example 1: Arch = G, Sz = S, St = 2 → classified building
    Example 2: Arch = R, Sz = S, St = 2 → non-classified building
    Example 3: Arch = G, Sz = L, St = 2 → classified building
    Example 4: Arch = G, Sz = S, St = 0 → non-classified building
    Example 5: Arch = R, Sz = L, St = 2 → classified building

    Answer:

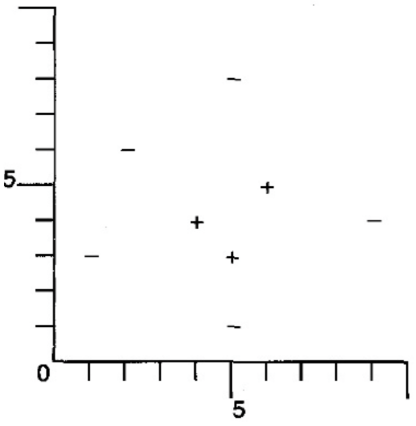
    $S0 = \{\emptyset, \emptyset, \emptyset\}$ and $G0 = \{?, ?, ?\}$
    $S1 = \{G, S, 2\}$ and $G1 = \{?, ?, ?\}$
    $S2 = \{G, S, 2\}$ and $G2 = \{G, ?, ?\}$
    $S3 = \{G, ?, 2\}$ and $G3 = \{G, ?, ?\}$
    $S4 = \{G, ?, 2\}$ and $S4 = \{G, ?, 2\}$
    $S5 = G5 = \emptyset$

11. **Rectangular version spaces and candidate elimination** (exercise 2.4 from the course book)
Consider the instance space consisting of integer points in the x, y plane and the set of hypotheses H consisting of rectangles. More precisely, hypotheses are of the form $a \leqslant x \leqslant b$, $c \leqslant y \leqslant d$, where a, b, c and d can be any integers. Consider the version space with respect to the set of positive (+) and negative (-) training examples:

$$\begin{array}{ll} -(1,3) & -(2,6) \\ +(6,5) & +(5,3) \\ -(9,4) & -(5,1) \\ +(4,4) & -(5,8) \end{array}$$

    (a) What is the S boundary of the version space in this case? Write a diagram with the training data and the S boundary.
    (b) What is the G boundary of this version space? Draw that in the diagram as well.
    (c) Suppose the learner may suggest a new x, y instance and ask the trainer for its classification. Suggest a query guaranteed to reduce the size of the version space, regardless how the trainer classifies it. Suggest one that will not.
    (d) Now assume you are the teacher, attempting to reach a particular target concept, $3 \leqslant x \leqslant 5, 2 \leqslant y \leqslant 9$. What is the smallest number of training examples you can provide so that the Candidate- Elimination algorithm will perfectly learn the concept?

    Answer:



(a) $S = \{< 4, 6, 3, 5 >\}$

(b) $G = \{< 3, 8, 2, 7 >\}$

(c) To reduce the VS: $(4, 6)$ or $(7, 3)$. Instances with no impact on the VS: $(5, 4)$ or $(3, 9)$.

(d) 6 points: $+(3, 9), +(5, 2), -(2, 5), -(4, 1), -(6, 5), -(4, 10)$

12. **Finding a maximally specific consistent hypothesis** (exercise 2.7 from the course book)
Consider a concept learning problem in which each instance is a real number and in which each hypothesis is an interval over the reals. More precisely, each hypothesis in H is of the form: $a < x < b$ as in $4.5 < x < 6.1$, meaning that all real numbers between 4.5 and 6.1 are classified as positive examples and all others are classified as negative examples.

(a) Explain informally why there cannot be a maximally specific consistent hypothesis for any set of positive training examples.

(b) Suggest a modification to the hypothesis representation so this will not happen.

Answer:

(a) $a < x < b$ is not a well defined hypothesis representation

(b) $a \leq x \leq b$