ANSWERS

U8.3 Log odds

Mapping · University · about 30 min

What this lesson is about

Why the cells hold a log odds and not a probability, and what the clamp is for.

- The problem with probabilities
- The odds, and their logarithm
- Getting back to a probability
- The two constants
- The clamp, which is not optional
- What a grid cannot represent

Questions

8 marks in all

1. What does this program print? [1 mark]

```
import math
p = 0.7
l = math.log(p / (1 - p))
print(round(l, 2))
print(round(1.0 / (1.0 + math.exp(-l)), 2))
```

Answer:

```
0.85
0.7
```

The odds are 0.7 / 0.3 = 7/3, and log(7/3) = 0.85, which is L_OCC. The logistic function turns it back into 0.7.

2. What does this program print? [1 mark]

```
import math
L_OCC, L_FREE, L_MAX = 0.85, -0.4, 10.0
l = 0.0
for update in (L_OCC, L_OCC, L_FREE, L_OCC, L_FREE):
    l = max(-L_MAX, min(L_MAX, l + update))
print(round(l, 2), round(1.0 / (1.0 + math.exp(-l)), 3))
```

Answer:

```
1.75 0.852
```

Three hits and two pass-throughs add to 3 x 0.85 - 2 x 0.4 = 1.75, and 1 / (1 + exp(-1.75)) = 0.852. No multiplication and no normaliser.

3. Why do occupancy grids store log odds rather than probabilities? [1 mark]

- ☐ **A** Bayes' rule becomes addition, with no normaliser, no underflow, and no cell stuck for ever at 0 or 1
- ☐ **B** Log odds use less memory than a probability
- ☐ **C** Log odds make neighbouring cells independent
- ☐ **D** Log odds are always between 0 and 1, so they are easier to draw

> **Answer:** A. The range is the whole real line, an unknown cell is exactly 0, and evidence is symmetric: +0.85 then -0.85 returns a cell to where it started.

4. Why is |L_OCC| larger than |L_FREE| in almost every implementation? [1 mark]

- ☐ **A** A hit is stronger evidence: things that reflect are there, but a beam can pass through a cell and miss a thin object in it
- ☐ **B** Occupied cells are rarer, so they need a larger update to be seen
- ☐ **C** It stops free space growing faster than walls
- ☐ **D** It corrects for the half-cell bias of ray casting

> **Answer:** A. L_OCC = 0.85 means a hit is right with probability 0.7, while L_FREE = -0.4 is a pass-through right about 0.6 of the time.

5. A mapper steps along each beam in half cells and adds L_FREE = -0.4 at every step. What is one beam passing through a cell really worth? [1 mark]

- ☐ **A** About -0.8, nearly as much as a hit, because the steps land in most cells twice
- ☐ **B** -0.4, as intended
- ☐ **C** -0.2, because each step covers only half a cell
- ☐ **D** Nothing, because the second step cancels the first

> **Answer:** A. Two steps in the same cell add L_FREE twice. Collect the cells a beam crosses in a set and add L_FREE to each once, or a pass stops being weaker evidence than a hit.

6. What is the main purpose of clamping each cell's log odds to plus or minus 10? [1 mark]

- ☐ **A** It caps how confident a cell can become, and so how long it takes to change its mind when the world changes
- ☐ **B** It prevents floating point overflow
- ☐ **C** It keeps the probability exactly between 0.1 and 0.9
- ☐ **D** It stops free cells becoming frontiers

> **Answer:** A. Without it, a wall stared at for a minute reaches several hundred, and when the wall moves the robot plans around it for more than two minutes.

7. A cell starts unknown at log odds 0 and receives only hits of +0.85, with a clamp at 10. How many hits does it take to reach the clamp? [1 mark]

> **Answer:** 12. After 11 hits it is 9.35, below the clamp. The 12th would take it to 10.2, so the clamp holds it at 10.

8. The function log(p / (1 - p)) has another standard name, the inverse of the logistic function. What is it? [1 mark]

> **Answer:** logit. It is the logit, the same quantity a neural network's final layer produces before a sigmoid.

The task: adding up the evidence

Print `p one:`, the probability a cell is occupied after one hit starting from an empty map. Then take 25 readings, updating a grid in log odds with a clamp at 10 and each cell changed at most once per reading, and print `wall l:` and `wall p:` for the cell the wall is in.

```
from bugbot import *
import math
connect()

CELL = 5.0
L_OCC, L_FREE, L_MAX = 0.85, -0.4, 10.0
X0, Y0 = 100.0, 50.0
```

The hint students can ask for: Each hit adds 0.85 to the cell's log odds and each pass-through subtracts 0.4, and the total is held between -10 and +10. Turning a log odds back into a probability is the logistic function; one hit from an empty map is that function of 0.85.

A solution

```
from bugbot import *
import math
connect()

CELL = 5.0
L_OCC, L_FREE, L_MAX = 0.85, -0.4, 10.0
X0, Y0 = 100.0, 50.0

def p_of(l):
    return 1.0 / (1.0 + math.exp(-l))

print("p one:", round(p_of(L_OCC), 4))

grid = {}

def bump(key, amount):
    grid[key] = max(-L_MAX, min(L_MAX, grid.get(key, 0.0) + amount))

for i in range(25):
    d = distance()
    free = set()                    # each cell the ray crosses, once
    r = 0.0
    while r < d - CELL:
        free.add((int(X0 / CELL), int((Y0 + r) / CELL)))
        r += CELL / 2
    for key in free:
        bump(key, L_FREE)
    bump((int(X0 / CELL), int((Y0 + d) / CELL)), L_OCC)
    wait(0.1)

wall = grid[(int(X0 / CELL), 30)]
print("wall l:", round(wall, 2))
print("wall p:", round(p_of(wall), 5))
```

Any program that meets the task's checks is marked correct in the simulator; this is one way, not the only way.