WORKSHEET

8.8 Project: situations

BugBotLab

Learning · Robot club · about 30 min

Name _____ Class _____ Date _____

What this lesson is about

A classifier decides what the robot does next.

- The idea
- Decide once
- The geometry

Questions

6 marks in all

1. The classifier says `wall-ahead`. What does the plan do? [1 mark]

- ☐ **A** Slide sideways along the wall to find its end
- ☐ **B** Drive forward
- ☐ **C** Stop and wait
- ☐ **D** Turn round and go home

2. Near the end of the wall the label flickers between `gap-left` and `gap-right`. How do you stop the robot sliding back and forth? [1 mark]

- ☐ **A** Remember the first gap label in a variable and follow that plan from then on
- ☐ **B** Act on whichever label came last
- ☐ **C** Stop until the labels agree for ten seconds
- ☐ **D** Ignore the classifier and drive straight

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

```
plan = None
for label in ["open", "wall-ahead", "gap-left", "gap-right", "gap-left"]:
    if plan is None and label.startswith("gap"):
        plan = label
print("plan:", plan)
```

4. Which labels mean the wall ends here? [1 mark]

Tick every answer that is true.

- ☐ **A** gap-left
- ☐ **B** gap-right
- ☐ **C** open
- ☐ **D** wall-ahead

5. The wall is 60 cm ahead of the start and the goal zone is centred 81 cm ahead. How many cm beyond the wall is the goal's centre? [1 mark]

---

6. Why keep printing `model says: <label>` as the robot goes? [1 mark]

- ☐ **A** It shows you afterwards what the model was thinking at each moment
- ☐ **B** The classifier only works if you print
- ☐ **C** It slows the robot down safely
- ☐ **D** It sends the label to other robots

The task: situations

Get from the start to the goal zone without touching anything, using the classifier to decide what to do, printing `model says: <label>` at least five times on the way.

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()
def level_rows():
    # rows 2 and 3: the 16 readings that look straight ahead
    return tof_grid()[16:32]

def nearest(sample, data):
    # the label of the recorded sample most like this one (smallest sum of squared
differences)
    best_label, best_d = None, 1e18
    for feats, label in data:
        d = sum((a - b) ** 2 for a, b in zip(feats, sample))
        if d < best_d:
            best_label, best_d = label, d
    return best_label
DATA = [
    ([54, 52, 51, 51, 51, 51, 52, 54, 54, 52, 51, 51, 51, 51, 52, 54], 'open'),
    ([16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16], 'wall-ahead'),
    ([36, 47, 46, 46, 46, 16, 16, 16, 36, 47, 46, 46, 46, 16, 16, 16], 'gap-left'),
    ([16, 16, 16, 46, 46, 46, 47, 36, 16, 16, 16, 46, 46, 46, 47, 36], 'gap-right'),
    ([400, 400, 61, 61, 61, 61, 400, 400, 96, 62, 61, 61, 61, 61, 62, 64], 'open'),
    ([52, 50, 49, 49, 49, 49, 50, 52, 51, 50, 49, 49, 49, 49, 50, 51], 'open'),
    ([39, 38, 37, 37, 37, 37, 38, 39, 39, 38, 37, 37, 37, 37, 38, 39], 'open'),
    ([400, 400, 61, 61, 61, 61, 400, 400, 64, 62, 61, 61, 61, 61, 62, 64], 'open'),
    ([52, 50, 49, 49, 49, 49, 50, 52, 51, 50, 49, 49, 49, 49, 50, 51], 'open'),
    ([39, 38, 37, 37, 37, 37, 38, 39, 39, 38, 37, 37, 37, 37, 38, 39], 'open'),
    ([400, 400, 61, 61, 61, 61, 400, 400, 64, 62, 61, 61, 61, 61, 62, 96], 'open'),
    ([52, 50, 49, 49, 49, 49, 50, 52, 51, 50, 49, 49, 49, 49, 50, 51], 'open'),
    ([39, 38, 37, 37, 37, 37, 38, 39, 39, 38, 37, 37, 37, 37, 38, 39], 'open'),
    ([26, 25, 25, 25, 25, 25, 25, 26, 26, 25, 25, 25, 25, 25, 25, 26], 'wall-ahead'),
    ([18, 17, 17, 17, 17, 17, 17, 18, 17, 17, 17, 17, 17, 17, 17, 17], 'wall-ahead'),
    ([9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9], 'wall-ahead'),
    ([26, 25, 25, 25, 25, 25, 25, 26, 26, 25, 25, 25, 25, 25, 25, 26], 'wall-ahead'),
    ([18, 17, 17, 17, 17, 17, 17, 18, 17, 17, 17, 17, 17, 17, 17, 17], 'wall-ahead'),
    ([9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9], 'wall-ahead'),
    ([26, 25, 25, 25, 25, 25, 25, 26, 26, 25, 25, 25, 25, 25, 25, 26], 'wall-ahead'),
    ([18, 17, 17, 17, 17, 17, 17, 18, 17, 17, 17, 17, 17, 17, 17, 17], 'wall-ahead'),
    ([9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9], 'wall-ahead'),
    ([24, 34, 55, 59, 59, 59, 29, 30, 24, 33, 55, 59, 59, 59, 29, 30], 'gap-left'),
    ([24, 34, 51, 51, 51, 51, 25, 22, 24, 33, 51, 51, 51, 51, 25, 22], 'gap-left'),
    ([24, 34, 43, 43, 43, 43, 44, 18, 24, 33, 43, 43, 43, 43, 44, 18], 'gap-left'),
    ([42, 58, 59, 59, 29, 29, 29, 30, 42, 58, 59, 59, 29, 29, 29, 30], 'gap-left'),
    ([42, 52, 51, 51, 21, 21, 21, 22, 42, 52, 51, 51, 21, 21, 21, 22], 'gap-left'),
    ([42, 44, 43, 43, 13, 13, 13, 13, 42, 44, 43, 43, 13, 13, 13, 13], 'gap-left'),
    ([30, 29, 29, 29, 59, 59, 58, 42, 30, 29, 29, 29, 59, 59, 58, 42], 'gap-right'),
    ([22, 21, 21, 21, 51, 51, 52, 42, 22, 21, 21, 21, 51, 51, 52, 42], 'gap-right'),
    ([13, 13, 13, 13, 43, 43, 44, 42, 13, 13, 13, 13, 43, 43, 44, 42], 'gap-right'),
    ([30, 29, 59, 59, 59, 55, 34, 24, 30, 29, 59, 59, 59, 55, 33, 24], 'gap-right'),
    ([22, 25, 51, 51, 51, 51, 34, 24, 22, 25, 51, 51, 51, 51, 33, 24], 'gap-right'),
    ([18, 44, 43, 43, 43, 43, 34, 24, 18, 44, 43, 43, 43, 43, 33, 24], 'gap-right'),
]
# do this 50 times (tick counts from 0)
for tick in range(50):
```

```
    print('model says:', nearest(level_rows(), DATA))
    # drive forward at 60 (keeps going until the next command)
    forward(60)
    # pause 0.1 s (the robot keeps doing what it was told)
    wait(0.1)
# all motors off
stop()
```

Plan your program here, then type it in and press Run.

Do it on the robot

www.bugbotlab.com/learn/8-8-project-situations/

The simulator checks it and tells you when it passes. Nothing to install, no account.

Challenges

1. Go through the right-hand gap instead.
2. Use the network from lesson 8.4 as the classifier instead of `nearest`.
3. Start the robot facing the wall from the other side (edit the scene in free play) and get back.