ANSWERS

8.8 Project: situations

Learning · Robot club · about 30 min

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

> **Answer:** A. `open` drives forward, `wall-ahead` slides along, a gap label lines up and goes through.

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

> **Answer:** A. Decide once. A robot that changes its plan every tenth of a second goes nowhere.

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)
```

> **Answer:**
>
> ```
> plan: gap-left
> ```
>
> The first gap label is gap-left. After that `plan` is no longer None, so the later labels do not change it.

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

> **Answer:** A, B. Either gap label says the end of the wall is in view. Then line up with it and go through.

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]

> **Answer:** 21. 81 - 60 = 21 cm, so once through the gap there is a short drive to the goal.

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

> **Answer:** A. When the robot does something odd, those lines tell you whether the model or the plan was to blame.

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()
```

The hint students can ask for: Get past the wall to the green zone using the classifier to decide what to do: open means go, wall-ahead means slide sideways, a gap means go through it. Print `model says: <label>` as you go.

A solution

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

def wrapped(h):
    return (h + 180) % 360 - 180

def go_to(x, y, speed=60):
    px, py = position()
    a = math.radians(wrapped(math.degrees(math.atan2(x - px, y - py)) - heading()))
    drive(speed * math.cos(a), speed * math.sin(a), wrapped(0 - heading()) * 3)

def near(x, y, cm=4):
    px, py = position()
    return math.hypot(x - px, y - py) < cm

def drive_to(x, y):
    while not near(x, y):
        go_to(x, y)
        wait(0.1)
    stop()
    wait(0.4)
def level_rows():
    return tof_grid()[16:32]              # rows 2 and 3: the 16 readings that look
straight ahead

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'),
]
plan = None
for tick in range(590):
    x, y = position()
    label = nearest(level_rows(), DATA)
    if tick % 10 == 0:
        print("model says:", label)
    if y > 74:                                  # past the wall: head for the goal
        if near(0, 82, cm=5):
            break
        go_to(0, 82)
    elif plan == "left":                        # a gap was seen on the left: line up with
it, then through
        left(50) if x > -42 else go_to(-42, 80)
    elif plan == "right":
        right(50) if x < 42 else go_to(42, 80)
    elif label == "open":
        forward(60)
    elif label == "wall-ahead":
        left(60)                                # slide along the wall to find its end
    else:
        plan = "left" if label == "gap-left" else "right"   # decide once, then stick to it
    wait(0.1)
stop()
```

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