WORKSHEET

U12.4 Policy search on a robot

Learning, and the capstone · University · about 40 min

Name _____ Class _____ Date _____

What this lesson is about

Hill climbing on a real machine: a noisy score, a fair trial, and a budget measured in seconds.

- The setting
- Hill climbing
- The trial is where the rigour is
- Exploration and exploitation
- Why not just write the controller?

Questions

7 marks in all

1. This is the lesson's hill climb on a score whose true best value is 23. What does it print? [1 mark]

```
def score(v):
    return abs(v - 23.0) + 2.0

value, step = 0.0, 20.0
best = score(value)
for i in range(8):
    trial = value + step
    s = score(trial)
    if s < best:
        value, best = trial, s
    else:
        step = -step * 0.6
print(round(value, 2), round(best, 2), round(step, 2))
```

2. Why is a derivative free search the right tool for tuning the lateral offset on the robot? [1 mark]

- ☐ **A** The score comes out of a physical trial, not a formula, so there is no derivative to use
- ☐ **B** Derivative free methods always find the global optimum
- ☐ **C** Derivatives are too slow to compute for one parameter
- ☐ **D** The offset has no effect on the score's slope

3. Each trial starts wherever the previous one ended. What is wrong with the scores? [1 mark]

- ☐ **A** Each score measures the previous policy as much as the current one
- ☐ **B** They are noisier but still unbiased
- ☐ **C** Nothing, as long as the trials are the same length
- ☐ **D** The search will take fewer trials

4. A method needs 10,000 evaluations and each out and back trial takes 1.6 s of robot time. How many hours is that, to 2 decimal places? [1 mark]

5. The noise in one trial's score is about as large as the difference between two neighbouring offsets. What is the search doing? [1 mark]

- ☐ **A** Following coin flips; repeat and average the trials, or make them longer
- ☐ **B** Converging faster, because noise helps it explore
- ☐ **C** Finding the global optimum
- ☐ **D** Nothing different, because the shrinking step handles noise

6. Which statements about exploration and exploitation are right? [1 mark]

Tick every answer that is true.

- ☐ **A** Hill climbing explores with large early steps and exploits as the step shrinks
- ☐ **B** An upper confidence bound rule explores in proportion to how little an option has been tried
- ☐ **C** Exploiting too early polishes a mediocre policy
- ☐ **D** A good enough method makes the trade-off disappear
- ☐ **E** Epsilon-greedy never takes a random action once it has a best option

7. Feeding flow()[0] back into the lateral command would cancel the leak without learning. What does the learned offset offer that feedback does not? [1 mark]

- ☐ **A** It works with no sensor during the run and costs nothing at run time
- ☐ **B** It adapts to changes in the leak that happen after tuning
- ☐ **C** It removes the need for any trials
- ☐ **D** It handles situations the learner never saw

The task: learn the offset that drives it straight

Hill climb the lateral offset using short out and back trials, plot `score` as you go, and then drive the robot at least 90 cm up into the green lane.

```
from bugbot import *
connect()

DT = 0.1
CMD = 60
```

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

Do it on the robot

www.bugbotlab.com/learn/u12-4-policy-search/

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

Challenges

1. Run the same search twice and compare the offsets it lands on. Is the difference smaller than the width of the lane?
2. Score each policy twice and average. How much of your budget did that cost, and did the search end up anywhere better?
3. Search over two numbers, an offset and a gain on `flow()[0]`, by taking turns on each. What breaks first, the budget or the noise?