WORKSHEET

8.4 A tiny network

Learning · Robot club · about 30 min

Name _____ Class _____ Date _____

What this lesson is about

Weights, a loss, and watching it fall: training in the browser.

- Numbers in, numbers out
- Untrained
- Training
- Why the centring matters

Questions

7 marks in all

1. What does a trained network keep, so the data can be thrown away? [1 mark]

- ☐ **A** Its weights
- ☐ **B** Every sample it was shown
- ☐ **C** Just the list of labels
- ☐ **D** The loss from the last epoch

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

```
labels = sorted(["open", "wall-ahead", "gap-left", "gap-right"])
row = [0] * len(labels)
row[labels.index("open")] = 1
print(labels)
print(row)
```

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

```
for reading in [400, 30, 10]:
    print((reading - 30) / 20)
```

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

```
inputs = [1.0, -0.5]
weights = [2.0, 4.0]
bias = 0.5
total = sum(i * w for i, w in zip(inputs, weights)) + bias
print(total)
```

5. Put one epoch of training in order. [1 mark]

Number the lines 1 to 4 to put them in the right order.

___ `Step: move every weight a little`

___ `Measure how wrong they are: the loss`

___ `Backward: work out the gradients`

___ `Forward: work out the outputs from the inputs`

6. The learning rate `lr` is set far too big. What do you see? [1 mark]

- ☐ **A** The loss bounces about and never settles
- ☐ **B** The loss creeps down very slowly
- ☐ **C** The loss drops to zero at once
- ☐ **D** Training is skipped

7. Scaling the inputs as `X / 100` instead of `(X - 30) / 20` leaves the network stuck at about 88%. Why? [1 mark]

- ☐ **A** The inputs are no longer centred near zero, which gives the network worse numbers to work with
- ☐ **B** Dividing by 100 deletes some samples
- ☐ **C** The labels stop being one-hot
- ☐ **D** The network needs more hidden numbers

The task: a tiny network

Train the network on `DATA`, printing `epoch <n>: loss <value>` every 50 epochs, then print `accuracy: <n>%` and get it above 90. Do not drive.

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()
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'),
]
import numpy as np
X = (np.array([f for f, _ in DATA], dtype=float) - 30) / 20
print('samples:', X.shape)
```

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

Do it on the robot

www.bugbotlab.com/learn/8-4-a-tiny-network/

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

Challenges

1. Try 4 hidden numbers, then 16. Does it still reach 100%? How fast?
2. Try `lr` of 0.05 and of 3. Print the loss every 10 epochs to see what each does.
3. Classify the current view: put `tof_grid()[16:32]` (the 16 readings that look straight ahead) through the same scaling and the trained weights, and print the label with the biggest output.