INTERNSHIP PROGRESS

Modeling curriculum learning

Table of Contents

WEEKLY OBJECTIVES

	FEBRUARY	MARCH	APRIL	MAY
W1	/	Level 2: accuracy RL Level 2: learning progress RL	Unify the Level 1 models Account for catastrophic interference	Finishing touches Written report
W2	/	Create hypotheses and model ideas for Level 2 & 3	Level 3: integrate both accuracy and learning progress	Written report
W3	Level 1: implement tasks and neural network	Prepare the presentation	Level 3: explore more options (chaining effect? Between-task learning?)	(exams)
W4	Finish 3 models for Level 1 Level 2: accuracy RL learning	**LAB PRESENTATION** Integrate all suggestions	Level 3: adjust and compare model performances	(exams)

Past week

Upcoming week

LAST WEEK'S OBJECTIVES

1

IMPLEMENT ALL 3 TASKS

Easy – Hard – Impossible
Give them the same structure

LEARN TO TRACK MODEL PARAMETERS

Such as accuracy, loss, performance

2

UPLOAD CODE TO GITHUB

Improve understanding of the current model, search for potential improvements.

3

START LEVEL 2: REINFORCEMENT LEARNING

Research teacher level networks, and implement them using accuracy

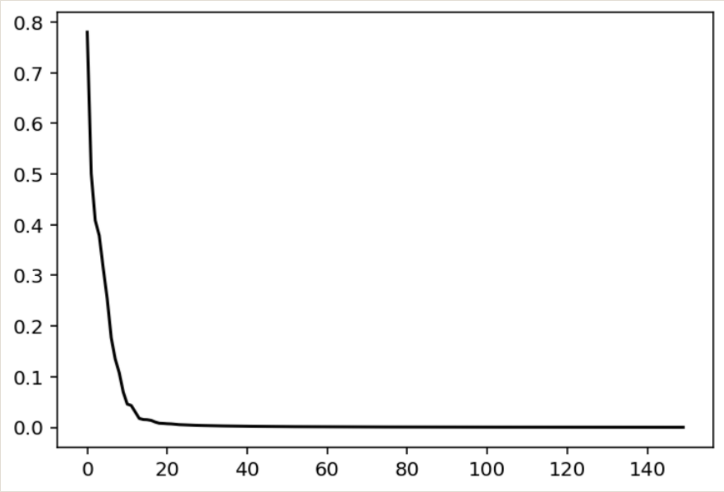
4

SUMMARISE FINDINGS

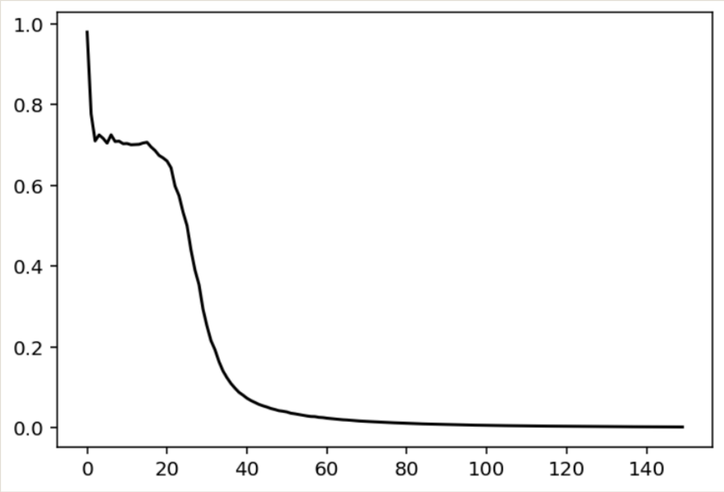
Write a short summary of findings, questions and plan for next week.

5

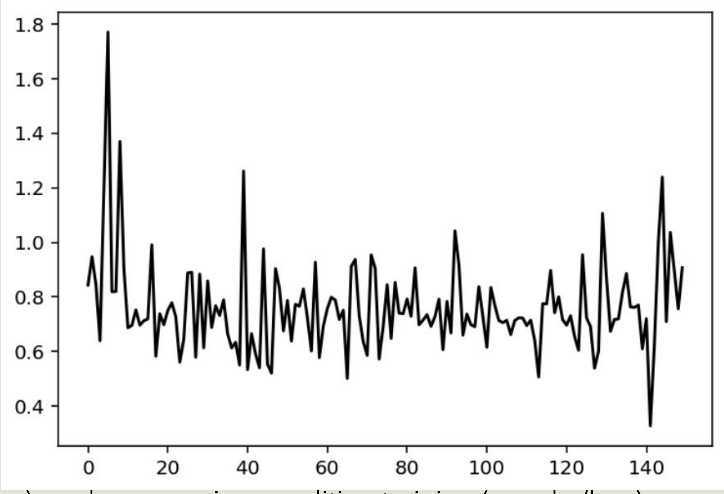
```python
n_input, n_output = train_x.shape[1], 1

model = Sequential()
model.add(Dense(
    units=4, activation='tanh', input_shape=(n_input,)))
model.add(Dense(
    n_output, activation='sigmoid'))
model.build()

model.compile(
    optimizer = tf.keras.optimizers.Adam(learning_rate=0.1),
    loss = tf.keras.losses.BinaryCrossentropy(),
    metrics = ['accuracy'])


##Y_DATA SET + FIT
###AND task
if model_name == 'AND_model':
    train_y = np.array([0, 0, 0, 1])
    train_y = train_y.reshape(4, 1)
    history = model.fit(train_x, train_y, batch_size = 1, epochs=epochs)
    loss_history = history.history["loss"]

###XOR task
elif model_name == 'XOR_model':
   train_y = np.array([0, 1, 1, 0])
   train_y = train_y.reshape(4, 1)
   history = model.fit(train_x, train_y, batch_size = 1, epochs=epochs)
   loss_history = history.history["loss"]

###RM loop
elif model_name == 'RM_model':
   loss_history = []
   accuracy_history = []

   for epoch in range(epochs):
       train_y = np.random.randint(0, 2, size=(4, 1))
       print(f"Epoch {epoch+1}/{epochs}")
       history = model.fit(train_x, train_y, batch_size = 1, epochs=1, verbose=1)
       loss_history.append(history.history["loss"][0])
       accuracy_history.append(history.history["accuracy"][0])
```



a) AND condition training (epochs/loss)



b) XOR condition training (epochs/loss)



c) random mapping condition training (epochs/loss)

Questions

Question 1	▪ Q value network (value-based method) or policy network (policy-based methods)? And other variables…
Question 2	▪ Which learning equation to use?
Notes	▪ I need help to check my understanding of reinforcement learning (see next page)

Several variables affect the rules that shape reinforcement learning.

	Effect on reinforcement learning	**Advantages and disandvages**
Value-based vs Policy-based	**Value-based**: algorithm stores and updates values through a value function **Policy-based**: algorithm learns and updates a policy	**Value-based**: efficient in discrete spaces, but struggles in more complex spaces, harder to adapt. **Policy-based**: complex spaces, stochastic (exploration and uncertainty), flexibility, requires more data.
Model-based vs Model-free	**Model-based**: models the environment through a transition and reward model. **Model-free**: uses estimates and updates based on prediction error.	**Model-based**: faster learning, better for limited data **Model-free**: better for rich data and uncertainty, flexibility
Sampling type	**Bootstrapping**: update estimates based on estimates of future values. **Monte Carlo sampling**: only updates after actual returns (full episodes).	**Bootstrapping**: faster learning but higher bias **Monte Carlo sampling**: slower learning but lower bias.
On-policy vs off-policy	…	

There are two steps in reinforcement learning: action choice and learning

Base: Rescorla-Wagner rule

V_t [option] = V_{t-1} [option] + α (R_{t-1} - V_{t-1} [option])

	LEARNING	ACTION CHOICE
VALUE	**Temporal Difference rule (Q)** Incorporate temporal differences when multiple t exist. $Q_\pi(s,a) = \mathrm{E}\left[\sum_{k=0}^{+\infty} \gamma^\pi R_{t+k+1} \middle| s, a, \pi\right]$ *Π = policy* *Γ = discount*	**Highest Q-value**?
POLICY	**Temporal Difference rule (V)** $V_\pi(s) = \mathrm{E}\left[\sum_{k=0}^{+\infty} \gamma^\pi R_{t+k+1} \middle| s, \pi\right]$	**Softmax function**: calculating values based on expectations of value (probabilities) $P_t[option == A] = \frac{\exp(\beta\ v_t[option==A])}{\sum_i \exp(\beta\ v_t[option==i])}$

Solution: the Bellman equation

$V(s) = max_a\left(R(s,a) + \gamma V(s')\right)$

RL modelling concepts

NEXT WEEK'S OBJECTIVES

1

CREATE A RL MODEL

Using a basic n-bandit task and a Rescorla-Wagner algorithm

INTEGRATE LVL 2

Using accuracy

2

…

…

3

INTEGRATE LVL 2

Using learning progress

4

SUMMARISE FINDINGS

Write a short summary of findings, questions and plan for next week.

5