Open in Colab

THE AGENT WHO THRIVED ON GOOD VIBES

Learning objectives

Upon completing this lab, you will be able to implement agents driven by a type of intrinsic motivation called 'interactional motivation.' This refers to the drive to engage in sensorimotor interactions that have a positive valence while avoiding those that have a negative valence.

Setup

Define the Agent class

In [40]:
```
class Agent:
    def __init__(self, _valences):
        """ Creating our agent """
        self._valences = _valences
        self._action = None
        self._predicted_outcome = None

    def action(self, _outcome):
        """ tracing the previous cycle """
        if self._action is not None:
            print(f"Action: {self._action}, Prediction: {self._predicted_outcome
                  f"Prediction: {self._predicted_outcome == _outcome}, Valence:

        """ Computing the next action to enact """
        # TODO: Implement the agent's decision mechanism
        self._action = 0
        # TODO: Implement the agent's anticipation mechanism
        self._predicted_outcome = 0
        return self._action
```

Environment1 class

In [41]:
```
class Environment1:
    """ In Environment 1, action 0 yields outcome 0, action 1 yields outcome 1 "
    def outcome(self, _action):
        # return int(input("entre 0 1 ou 2"))
        if _action == 0:
            return 0
        else:
            return 1
```

Environment2 class

In [42]:
```
class Environment2:
    """ In Environment 2, action 0 yields outcome 1, action 1 yields outcome 0 "
    def outcome(self, _action):
        if _action == 0:
            return 1
        else:
            return 0
```

Define the valence of interactions

In [43]:
```
valences = [[-1, 1],
            [1, -1]]
```

The valence table specifies the valence of each interaction. An interaction is a tuple (action, outcome):

	outcome 0	outcome 1
action 0	-1	1
action 1	1	-1

Instantiate the agent

In [44]:
```
a = Agent(valences)
```

Instantiate the environment

In [45]:
```
e = Environment1()
```

Test run the simulation

In [46]:
```
outcome = 0
for i in range(10):
    action = a.action(outcome)
    outcome = e.outcome(action)
```

```
Action: 0, Prediction: 0, Outcome: 0, Prediction: True, Valence: -1
Action: 0, Prediction: 0, Outcome: 0, Prediction: True, Valence: -1
Action: 0, Prediction: 0, Outcome: 0, Prediction: True, Valence: -1
Action: 0, Prediction: 0, Outcome: 0, Prediction: True, Valence: -1
Action: 0, Prediction: 0, Outcome: 0, Prediction: True, Valence: -1
Action: 0, Prediction: 0, Outcome: 0, Prediction: True, Valence: -1
Action: 0, Prediction: 0, Outcome: 0, Prediction: True, Valence: -1
Action: 0, Prediction: 0, Outcome: 0, Prediction: True, Valence: -1
Action: 0, Prediction: 0, Outcome: 0, Prediction: True, Valence: -1
```

Observe that, on each interaction cycle, the agent is mildly satisfied. On one hand, the agent made correct predictions, on the other hand, it experienced negative valence.

PRELIMINARY EXERCISE

Execute the agent in Environment2. Observed that it obtains a positive valence.

Modify the valence table to give a positive valence when the agent selects action `0` and obtains outcome `0`. Observe that this agent obtains a positive valence in Environment1.

ASSIGNMENT

Implement Agent2 that selects actions that, it predicts, will result in an interaction that have a positive valence.

Only when the agent gets bored does it select an action which it predicts to result in an interaction that have a negative valence.

In the trace, you should see that the agent learns to obtain a positive valence during several interaction cycles. When the agent gest bored, it occasionnaly selects an action that may result in a negative valence.

Create Agent2 by overriding the class Agent

In [47]:
```
class Agent2(Agent):
    def __init__(self, _valences):
        super().__init__(_valences)
        self.action_outcome = {0: None, 1: None}
        self.nb_corrects_in_a_row = 0
        self._action = None
        self._predicted_outcome = None
        self._valences = _valences


    # TODO override the method action(self, _outcome)
    def action(self, _outcome):
        """ On affiche les résultats précédents """
        if self._action is not None:
            print(f"Action: {self._action}, Prediction: {self._predicted_outcome
                  f"Prediction: {self._predicted_outcome == _outcome}, Valence:

        """ On mémorise l'outcome de l'action précédente """
        self.action_outcome[self._action] = _outcome

        # Si on ne sait pas quelle action choisir, on met 0 par défaut
        if self._action is None:
            self._action = 0

        """ Si la prédiction est correcte, on incrémente le compteur de corrects
        if self._predicted_outcome == _outcome or self._valences[self._action][_
            self.nb_corrects_in_a_row += 1
```

```
        else:
            self.nb_corrects_in_a_row = 0

        """ On choisit la prochaine action """
        # Si on a eu 4 corrects d'affilée, on change d'action
        if self.nb_corrects_in_a_row == 4 or self._valences[self._action][_outcc
            # action opposée (action = 1 - action) [1 - 0 => 1, 1 - 1 => 0]
            self._action = 1 - self._action
            self.nb_corrects_in_a_row = 0

        """ Selon l'action choisie, on prédit l'outcome si on peut """
        self._predicted_outcome = self.action_outcome[self._action]
        # Si on ne sait pas, on met une valeur par défaut
        if self._predicted_outcome is None:
            self._predicted_outcome = 0

        return self._action
```

Test your Agent2 in Environment1

In [48]:

```
a = Agent2(valences)
e = Environment1()
outcome = 0
for i in range(20):
    action = a.action(outcome)
    outcome = e.outcome(action)
```

```
Action: 1, Prediction: 0, Outcome: 1, Prediction: False, Valence: -1
Action: 0, Prediction: 0, Outcome: 0, Prediction: True, Valence: -1
Action: 1, Prediction: 1, Outcome: 1, Prediction: True, Valence: -1
Action: 0, Prediction: 0, Outcome: 0, Prediction: True, Valence: -1
Action: 1, Prediction: 1, Outcome: 1, Prediction: True, Valence: -1
Action: 0, Prediction: 0, Outcome: 0, Prediction: True, Valence: -1
Action: 1, Prediction: 1, Outcome: 1, Prediction: True, Valence: -1
Action: 0, Prediction: 0, Outcome: 0, Prediction: True, Valence: -1
Action: 1, Prediction: 1, Outcome: 1, Prediction: True, Valence: -1
Action: 0, Prediction: 0, Outcome: 0, Prediction: True, Valence: -1
Action: 1, Prediction: 1, Outcome: 1, Prediction: True, Valence: -1
Action: 0, Prediction: 0, Outcome: 0, Prediction: True, Valence: -1
Action: 1, Prediction: 1, Outcome: 1, Prediction: True, Valence: -1
Action: 0, Prediction: 0, Outcome: 0, Prediction: True, Valence: -1
Action: 1, Prediction: 1, Outcome: 1, Prediction: True, Valence: -1
Action: 0, Prediction: 0, Outcome: 0, Prediction: True, Valence: -1
Action: 1, Prediction: 1, Outcome: 1, Prediction: True, Valence: -1
Action: 0, Prediction: 0, Outcome: 0, Prediction: True, Valence: -1
Action: 1, Prediction: 1, Outcome: 1, Prediction: True, Valence: -1
```

Test your Agent2 in Environment2

In [49]:

```
a = Agent2(valences)
e = Environment2()
outcome = 0
for i in range(20):
    action = a.action(outcome)
    outcome = e.outcome(action)
```

```
Action: 1, Prediction: 0, Outcome: 0, Prediction: True, Valence: 1
Action: 1, Prediction: 0, Outcome: 0, Prediction: True, Valence: 1
Action: 1, Prediction: 0, Outcome: 0, Prediction: True, Valence: 1
Action: 1, Prediction: 0, Outcome: 0, Prediction: True, Valence: 1
Action: 0, Prediction: 0, Outcome: 1, Prediction: False, Valence: 1
Action: 0, Prediction: 1, Outcome: 1, Prediction: True, Valence: 1
Action: 0, Prediction: 1, Outcome: 1, Prediction: True, Valence: 1
Action: 0, Prediction: 1, Outcome: 1, Prediction: True, Valence: 1
Action: 1, Prediction: 0, Outcome: 0, Prediction: True, Valence: 1
Action: 1, Prediction: 0, Outcome: 0, Prediction: True, Valence: 1
Action: 1, Prediction: 0, Outcome: 0, Prediction: True, Valence: 1
Action: 1, Prediction: 0, Outcome: 0, Prediction: True, Valence: 1
Action: 0, Prediction: 1, Outcome: 1, Prediction: True, Valence: 1
Action: 0, Prediction: 1, Outcome: 1, Prediction: True, Valence: 1
Action: 0, Prediction: 1, Outcome: 1, Prediction: True, Valence: 1
Action: 0, Prediction: 1, Outcome: 1, Prediction: True, Valence: 1
Action: 1, Prediction: 0, Outcome: 0, Prediction: True, Valence: 1
Action: 1, Prediction: 0, Outcome: 0, Prediction: True, Valence: 1
Action: 1, Prediction: 0, Outcome: 0, Prediction: True, Valence: 1
```

Test your agent with a different valence table

Note that, depending on the valence that you define, it may be impossible for the agent to obtain a positive valence in some environments.

In [50]:

```
# Choose different valences
valences = [[1, -1],
            [1, -1]]
# Run the agent
a = Agent2(valences)
e = Environment2()
outcome = 0
for i in range(20):
    action = a.action(outcome)
    outcome = e.outcome(action)
```

```
Action: 0, Prediction: 0, Outcome: 1, Prediction: False, Valence: -1
Action: 1, Prediction: 0, Outcome: 0, Prediction: True, Valence: 1
Action: 1, Prediction: 0, Outcome: 0, Prediction: True, Valence: 1
Action: 1, Prediction: 0, Outcome: 0, Prediction: True, Valence: 1
Action: 1, Prediction: 0, Outcome: 0, Prediction: True, Valence: 1
Action: 0, Prediction: 1, Outcome: 1, Prediction: True, Valence: -1
Action: 1, Prediction: 0, Outcome: 0, Prediction: True, Valence: 1
Action: 1, Prediction: 0, Outcome: 0, Prediction: True, Valence: 1
Action: 1, Prediction: 0, Outcome: 0, Prediction: True, Valence: 1
Action: 1, Prediction: 0, Outcome: 0, Prediction: True, Valence: 1
Action: 0, Prediction: 1, Outcome: 1, Prediction: True, Valence: -1
Action: 1, Prediction: 0, Outcome: 0, Prediction: True, Valence: 1
Action: 1, Prediction: 0, Outcome: 0, Prediction: True, Valence: 1
Action: 1, Prediction: 0, Outcome: 0, Prediction: True, Valence: 1
Action: 1, Prediction: 0, Outcome: 0, Prediction: True, Valence: 1
Action: 0, Prediction: 1, Outcome: 1, Prediction: True, Valence: -1
Action: 1, Prediction: 0, Outcome: 0, Prediction: True, Valence: 1
Action: 1, Prediction: 0, Outcome: 0, Prediction: True, Valence: 1
Action: 1, Prediction: 0, Outcome: 0, Prediction: True, Valence: 1
```

Report

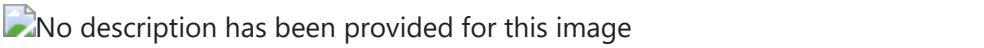
Explain what you programmed and what results you observed. Export this document as PDF including your code, the traces you obtained, and your explanations below (no more than a few paragraphs):

Pour l'Agent2, nous avons développé un agent qui en plus de l'Agent1 prend en compte des interaction. Une interaction est un tuple (action, outcome) qui a une certaine valeur. Ici, les valeurs des interactions sont soit négative (-1), soit positive (1). L'agent2 essaie toujours de prédire la sortie de l'action qu'il a choisit mais regarde et prends en compte la valence l'interaction qui en découle. L'agent2 va préférer une valence positive à une valence négative. De ce fait, si il reçoit une valence négative il va changer d'action pour choisir une action qui lui donne un valence positive, tout en faisant attention que l'agent ne s'ennuie pas et que si il commence à s'ennuyer, alors il va changer d'action même si cela implique d'avoir une valence négative.

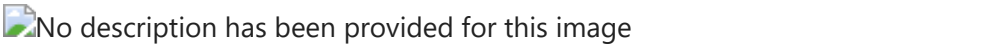
Tableau de valence utilisé pour les 2 premières traces:

	outcome 0	**outcome 1**
action 0	-1	1
action 1	1	-1

Dans l'image ci-dessous l'agent évoluant dans l'environnemen1 ne reçoit que des valences négatives. Cela est dû au tableau de valence qui fait que peut importe l'action que choisit l'agent il recevra toujours une valence négative. Par conséquent il va à chaque fois changer l'action


No description has been provided for this image

A l'inverse, toujours à cause du tableau de valence choisit, dans l'environnemnt2, l'agent ne va recevoir que des valences positives ce qui implique qu'il ne va changer d'action uniquelent lorsqu'il va s'ennuyer


No description has been provided for this image

Pour avoir quelque chose de moins binaire (recevoir que des valences positives ou que négatives), nous avons défini un nouveau tableau de valencesqui en fonction de l'action choisi revoit soit une valence positive soit une valence négative.

Tableau de valence choisit:

	outcome 0	**outcome 1**
action 0	1	-1
action 1	1	-1

Ci-dessous la trace de l'agent2 dans l'environnement2 avec le nouveau tableau de valence. Avec ce tableau de valence et dans cet environnement, lorque l'agent choisit l'action 1 il valence recevoir une valence positive et une valence négative pour l'action 0. L'agent va donc effectuer l'action 1 jusqu'à ce qu'il s'ennuie pouis l'action 0 une fois et revenir à l'action 1.

 No description has been provided for this image