Developmental AI

Rida Asri

October 26, 2024

1 Introduction

Le but de ces travaux pratiques (TPs) est de se familiariser avec la mise en œuvre d'un agent intelligent, en suivant les principes fondamentaux de l'intelligence artificielle développementale. Cette approche vise à développer des agents capables d'apprendre, d'évoluer, et de s'adapter à leur environnement de manière autonome

2 Agent 1

L'objectif de cet agent est de développer sa capacité à prédire les résultats de ses actions tout en ajustant son comportement de manière proactive. Dans un environnement donné, l'agent apprend à anticiper l'issue de chacune de ses actions, en partant de l'hypothèse que chaque action produit systématiquement le même résultat. Ainsi, pour chaque action possible, il enregistre les résultats observés afin d'affiner ses prédictions et d'optimiser son comportement. De plus, afin de favoriser l'exploration et éviter une répétition excessive, l'agent est programmé pour changer d'action après avoir correctement prédit le même résultat quatre fois de suite pour une action donnée. En intégrant cette mémoire des résultats passés et en alternant entre prédiction et exploration, l'agent est capable de mieux appréhender son environnement, développant ainsi une intelligence adaptative au fil des interactions.

2.1 Implementation

L'agent est conçu pour interagir avec un environnement dans le but de prédire correctement le résultat de ses actions, en alternant entre prédiction correcte et exploration.

2.1.1 `__init__(self)`

- Initialise les variables de l'agent.
- `self.states` : Contient une liste de paires `[Action, Prediction]` où chaque paire représente une action (0 ou 1) et une prédiction (0 ou 1). Cela représente toutes les combinaisons possibles d'action et de prédiction au début.
- `self.correct_predictions` : Compteur pour suivre le nombre de prédictions consécutives réussies.
- `self.action` : Stocke l'action choisie (0 ou 1).
- `self.prediction` : Stocke la prédiction actuelle de l'agent (0 ou 1).

Ces dernières (`self.action` et `self.prediction`) sont initialisées avec des choix aléatoires entre 0 et 1.

2.1.2 `remove_state(self, Action_Prediction)`

Supprime de `self.states` une combinaison action-prédiction qui a été tentée et s'est révélée incorrecte, afin de ne plus réessayer cette paire dans le futur.

2.1.3 `Message(self, outcome)`

Affiche un message contenant les informations actuelles sur l'action, la prédiction, l'issue réelle de l'action, et si cette issue satisfait l'agent (c'est-à-dire si la prédiction était correcte).

2.1.4 `step(self, outcome)`

Le cœur de la logique de l'agent, qui s'exécute à chaque étape pour mettre à jour ses variables.

1. Vérifie si la prédiction actuelle (`self.prediction`) correspond à l'issue (`outcome`) produite par l'environnement.

2. Si **incorrecte** :
   - La paire `[action, prediction]` est retirée de `self.states` pour éviter de la réutiliser.
   - L'agent choisit une nouvelle action et une nouvelle prédiction de manière aléatoire, puis vérifie si cette combinaison existe dans `self.states`.
   - Le processus continue jusqu'à ce qu'une combinaison valide soit trouvée.

3. Si **correcte** :
   - Le compteur `self.correct_predictions` est incrémenté.
   - Si ce compteur atteint 4, cela signifie que l'agent a correctement prédit quatre fois de suite. Dans ce cas, l'agent change d'action et de prédiction en inversant leurs valeurs et réinitialise `self.correct_predictions` à 0.

2.2 Résultat

Lorsqu'il interagit avec son environnement, l'agent commence par choisir une action et émet une prédiction concernant le résultat de cette action. La méthode de travail de l'agent repose sur un système de mémoire qui stocke les combinaisons d'actions et de prédictions . À chaque étape, l'agent reçoit un résultat correspondant à son action et vérifie si sa prédiction est correcte. Si la prédiction est incorrecte, l'agent retire la paire action-prédiction de sa mémoire, ce qui lui permet d'éviter de réutiliser des combinaisons qui n'ont pas donné de résultats satisfaisants. En revanche, si la prédiction est correcte, l'agent incrémente un compteur de succès. Lorsque ce compteur atteint quatre, l'agent change d'action et de prédiction en inversant leurs valeurs, ce qui favorise l'exploration de nouvelles stratégies

```
Action: 1, Prediction: 1, Outcome: 1, Satisfaction: True
Action: 1, Prediction: 1, Outcome: 1, Satisfaction: True
Action: 1, Prediction: 1, Outcome: 1, Satisfaction: True
Action: 1, Prediction: 1, Outcome: 1, Satisfaction: True
Action: 0, Prediction: 0, Outcome: 0, Satisfaction: True
Action: 0, Prediction: 0, Outcome: 0, Satisfaction: True
Action: 0, Prediction: 0, Outcome: 0, Satisfaction: True
Action: 0, Prediction: 0, Outcome: 0, Satisfaction: True
Action: 1, Prediction: 1, Outcome: 1, Satisfaction: True
Action: 1, Prediction: 1, Outcome: 1, Satisfaction: True
Action: 1, Prediction: 1, Outcome: 1, Satisfaction: True
Action: 1, Prediction: 1, Outcome: 1, Satisfaction: True
Action: 0, Prediction: 0, Outcome: 0, Satisfaction: True
Action: 0, Prediction: 0, Outcome: 0, Satisfaction: True
Action: 0, Prediction: 0, Outcome: 0, Satisfaction: True
Action: 0, Prediction: 0, Outcome: 0, Satisfaction: True
Action: 1, Prediction: 1, Outcome: 1, Satisfaction: True
Action: 1, Prediction: 1, Outcome: 1, Satisfaction: True
Action: 1, Prediction: 1, Outcome: 1, Satisfaction: True
Action: 1, Prediction: 1, Outcome: 1, Satisfaction: True
```

Figure 1: TRACE ENVIRONNEMENT 1.

```
Action: 0, Prediction: 1, Outcome: 1, Satisfaction: True
Action: 0, Prediction: 1, Outcome: 1, Satisfaction: True
Action: 0, Prediction: 1, Outcome: 1, Satisfaction: True
Action: 0, Prediction: 1, Outcome: 1, Satisfaction: True
Action: 1, Prediction: 0, Outcome: 0, Satisfaction: True
Action: 1, Prediction: 0, Outcome: 0, Satisfaction: True
Action: 1, Prediction: 0, Outcome: 0, Satisfaction: True
Action: 1, Prediction: 0, Outcome: 0, Satisfaction: True
Action: 0, Prediction: 1, Outcome: 1, Satisfaction: True
Action: 0, Prediction: 1, Outcome: 1, Satisfaction: True
Action: 0, Prediction: 1, Outcome: 1, Satisfaction: True
Action: 0, Prediction: 1, Outcome: 1, Satisfaction: True
Action: 1, Prediction: 0, Outcome: 0, Satisfaction: True
Action: 1, Prediction: 0, Outcome: 0, Satisfaction: True
Action: 1, Prediction: 0, Outcome: 0, Satisfaction: True
Action: 1, Prediction: 0, Outcome: 0, Satisfaction: True
Action: 0, Prediction: 1, Outcome: 1, Satisfaction: True
Action: 0, Prediction: 1, Outcome: 1, Satisfaction: True
Action: 0, Prediction: 1, Outcome: 1, Satisfaction: True
Action: 0, Prediction: 1, Outcome: 1, Satisfaction: True
```

Figure 2: TRACE ENVIRONNEMENT 2.

3 Agent 2

L'objectif de l'Agent2 est d'apprendre à choisir des actions en fonction de leur capacité à générer une valence positive lors des interactions avec son environnement. Cet agent est conçu pour prédire les actions qui entraîneront des résultats positifs, optimisant ainsi son comportement au fil des cycles d'interaction. Lorsqu'il éprouve de l'ennui, l'agent est programmé pour sélectionner occasionnellement une action qui pourrait engendrer une valence négative. Ce mécanisme permet à l'agent d'explorer différentes options, tout en continuant à privilégier des actions qui maximisent la valence positive. L'objectif final est de démontrer que l'agent apprend à obtenir une valence positive à travers ses interactions, en maintenant un équilibre entre exploitation des actions réussies et exploration de nouvelles possibilités lorsqu'il s'ennuie.

3.1 Implementation

L'agent est conçu pour interagir avec un environnement en maximisant la valence positive de ses actions tout en gérant l'ennui.

3.1.1 `__init__(self)`

- **valence** : Une matrice qui représente la valence associée à chaque action et à chaque issue. La valence est utilisée pour évaluer la qualité des actions de l'agent.
- **Max_Ennui** : Le seuil d'ennui qui, une fois atteint, oblige l'agent à changer d'action.
- **self.action** : Représente l'action actuelle choisie par l'agent (initialisée à 0).
- **self.pred_outcome** : Stocke l'issue prédite pour l'action en cours.
- **self.memoire** : Un dictionnaire qui enregistre les issues observées pour chaque action afin de mémoriser le comportement passé.
- **self.listePrefere** : Une liste qui stocke l'action préférée et l'issue correspondante. Initialisée à [0, 0], elle indique que l'agent commence sans préférence.
- **self.Numbre_action_Prefere** : Compteur pour suivre le nombre d'actions préférées consécutives.
- **self.actionfinal** : Stocke l'action finale à effectuer.

3.1.2 `Action(self, outcome)`

- **Paramètre outcome** : L'issue obtenue à partir de l'interaction avec l'environnement.
- **Calcul de la préférence** : Compare la valence actuelle (pour l'action préférée) avec celle de l'action actuelle. Si la nouvelle action (avec l'issue observée) a une valence plus élevée, elle devient l'action préférée.
- **Changement d'action en cas d'ennui** : Si le compteur d'actions préférées atteint le maximum d'ennui, l'agent change d'action en choisissant l'action opposée.
- **Mise à jour de la mémoire** : Appelle la méthode `Update_Memoire` pour enregistrer l'issue de l'action actuelle.
- **Incrémentation du compteur** : Si l'action actuelle correspond à l'action préférée, incrémente le compteur des actions préférées consécutives.
- **Retourne l'action à exécuter** : La méthode retourne l'action choisie.

3.1.3 `Update_Memoire(self, outcome)`

Met à jour la mémoire de l'agent en fonction de l'issue obtenue. Si l'action n'est pas encore enregistrée, elle est ajoutée avec l'issue correspondante. Si l'action existe déjà mais que l'issue change, la mémoire est mise à jour pour refléter le nouvel outcome.

3.1.4 `Message(self, outcome)`

Affiche les informations actuelles sur l'action, la prédiction, l'issue obtenue, la validité de la prédiction, et la valence associée à l'action.

3.2 Résultat

L'Agent2 est conçu pour interagir avec son environnement en optimisant la valence positive de ses actions. Au début de chaque interaction, l'agent évalue la valence associée à ses actions potentielles à l'aide d'une matrice de valence, qui fournit une évaluation de la qualité des résultats anticipés. L'agent sélectionne d'abord l'action qui, selon ses prédictions, générera une interaction avec une valence positive. Toutefois, lorsque l'agent ressent de l'ennui, c'est-à-dire lorsque le compteur des actions préférées consécutives atteint un seuil prédéfini, il change de stratégie et opte pour une action qui pourrait entraîner une valence négative. Cette stratégie permet à l'agent de diversifier ses interactions et d'éviter de se retrouver dans un cycle d'actions répétitives. Au fil des cycles d'interaction, l'Agent2 apprend à maximiser sa valence positive en adaptant ses choix en fonction des résultats précédemment observés, et sa mémoire joue un rôle essentiel dans ce processus d'apprentissage, lui permettant d'ajuster ses actions en fonction des issues passées.

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

Figure 3: TRACE ENVIRONNEMENT 1 avec valence = [[-1, 1], [-1, 1]].

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

Figure 4: TRACE ENVIRONNEMENT 2 avec valence = [[-1, 1], [-1, 1]].

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

Figure 5: TRACE ENVIRONNEMENT 1 avec valence = [[1, -1], [1, -1]].

4 Agent 3

L'objectif principal de cet agent est de développer un comportement exploratoire dans un environnement simulé, en maximisant la valence positive de ses interactions. L'agent est capable de sélectionner des actions appropriées, en utilisant une matrice de valences qui attribue une valeur à chaque action et à son issue. En apprenant à évaluer les résultats de ses actions, l'agent doit éviter de se retrouver bloqué dans un coin de l'environnement. Ainsi, il doit adopter des stratégies qui lui permettent de réagir de manière appropriée aux différentes situations rencontrées, notamment en tenant compte des collisions avec les bords de la fenêtre. À travers cette expérience, les participants apprendront à concevoir un agent capable d'explorer efficacement son environnement tout en prenant en compte les interactions passées pour améliorer ses décisions futures.

4.1 Implementation

L'agent 3 est une version améliorée de l'agent 2, conçue spécifiquement pour l'environnement ColabTurtle. Cette nouvelle itération intègre plusieurs modifications clés qui renforcent son comportement exploratoire et adaptatif. Voici les principales améliorations :

4.1.1 Gestion de l'ennui améliorée

L'agent 3 introduit un mécanisme sophistiqué de gestion de l'ennui. Grâce à un compteur `Numbre_action_Prefere`, il suit le nombre d'actions préférées consécutives. Lorsque ce compteur atteint un seuil défini par `Max_Ennui`, l'agent choisit aléatoirement une action parmi celles non préférées (0, 1 ou 2). Ce processus empêche l'agent de rester bloqué dans un cycle d'actions répétitives, ce qui améliore sa capacité à explorer de nouvelles options dans l'environnement de ColabTurtle.

4.1.2 Mise à jour de la mémoire

L'agent 3 inclut une méthode `Update_Memoire`, qui actualise sa mémoire en fonction des résultats observés. Cette méthode enregistre les résultats des actions précédentes et ne met à jour l'issue que si celle-ci change. En gardant une trace de ses expériences passées, l'agent adapte son comportement pour maximiser ses chances de succès dans l'environnement dynamique de ColabTurtle.

4.1.3 Mécanisme d'action

La méthode `Action` a été repensée pour intégrer une logique de sélection basée sur les valences associées. L'agent évalue maintenant la valence de l'action en cours par rapport à celle préférée et ajuste ses préférences en conséquence. Cette approche permet à l'agent de choisir des actions de manière plus réfléchie, tenant compte des effets de ses décisions passées sur son comportement futur.

4.2 Résultat

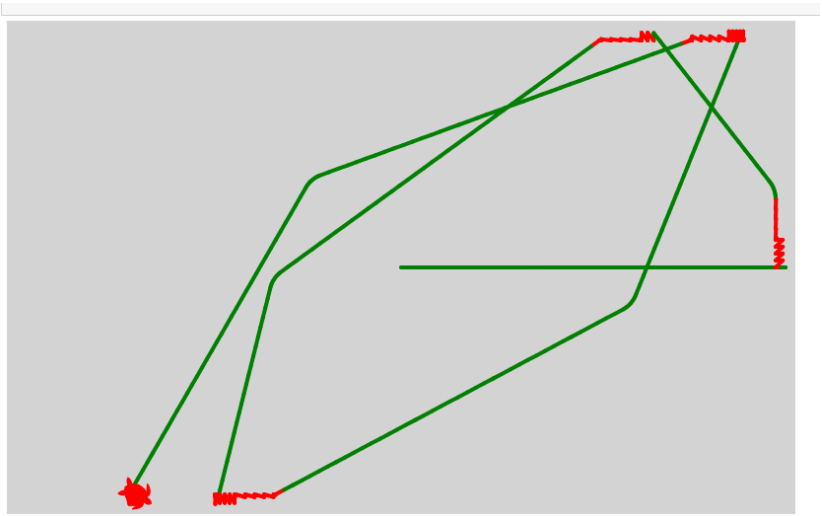

Figure 6: TRACE Agent 2 avec ColabTurtleEnvironment.

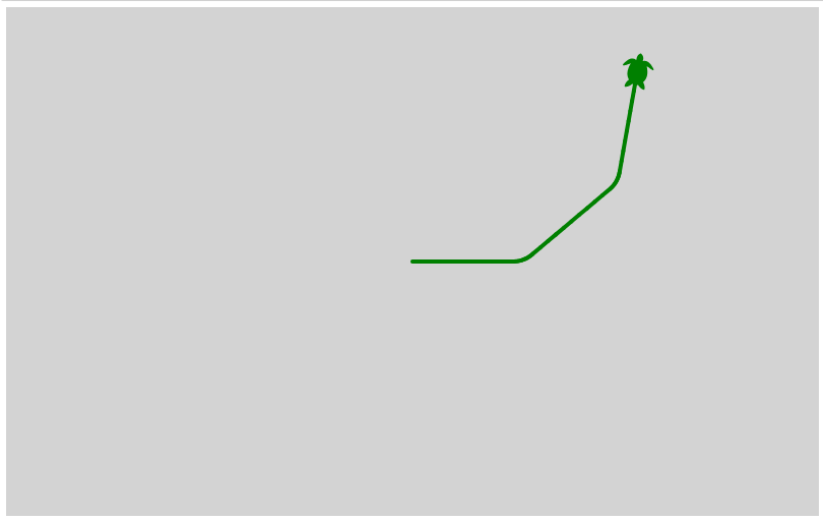

Figure 7: TRACE Agent 3 avec ColabTurtleEnvironment.

5 Agent 4

L'objectif de cet agent est de concevoir un agent évolutif motivé par l'interaction, capable d'adapter sa prochaine action en fonction du contexte des interactions précédentes. À l'issue de ce laboratoire, vous serez en mesure d'implémenter un agent qui :

- **Adapte son comportement** : L'agent doit être capable de mémoriser les interactions passées et d'utiliser ces informations pour influencer ses décisions futures.
- **Prédit les résultats** : L'agent doit être en mesure de prédire l'issue de ses actions en fonction du contexte des interactions antérieures, afin de maximiser la valence positive lors de ses actions.
- **Optimise ses actions** : En analysant les interactions précédentes, l'agent doit choisir des actions qui lui permettront d'obtenir la plus haute valence possible, que ce soit dans les environnements définis (Environment1, Environment2 ou Environment3).

5.1 Implementation

5.1.1 self._interactions

- **Type** : Dictionnaire
- **Description** : Ce dictionnaire stocke toutes les interactions possibles, où chaque interaction est indexée par sa clé (générée par la méthode `key()` de la classe `Interaction`). Cela permet un accès rapide à chaque interaction par son identifiant.

5.1.2 self._intended_interaction

- **Type** : Instance de `Interaction`
- **Description** : Cet attribut représente l'interaction prévue (c'est-à-dire l'action et le résultat anticipé) par l'agent. Il est initialisé avec l'interaction correspondant à la clé "00".

5.1.3 __init__(self, _interactions)

- **Paramètre** : `_interactions` (liste d'interactions)
- **Fonctionnalité** :
  - Initialise le dictionnaire des interactions en itérant sur la liste fournie. Chaque interaction est ajoutée au dictionnaire avec sa clé comme index.
  - Définit l'interaction prévue initiale à "00", qui est généralement un état de départ.

5.1.4 action(self, _outcome)

- **Paramètre** : `_outcome` (résultat d'une interaction précédente)
- **Fonctionnalité** :
  - Récupère l'interaction précédente basée sur l'action prévue et le résultat (_outcome). Cela permet de garder une trace de ce qui s'est passé dans le cycle précédent.
  - Affiche les détails de l'action (action actuelle, résultat prévu, résultat réel, et valence de l'interaction précédente) pour le débogage.
  - Cherche l'interaction avec la meilleure valence parmi toutes les interactions disponibles. Cela implique :
    - Initialiser `best_valence` à une valeur très basse (négative) pour s'assurer que toute valence positive soit prise en compte.
    - Parcourir chaque interaction dans `_interactions` et mettre à jour `best_valence` et `best_interaction` si une interaction avec une valence plus élevée est trouvée.

* Met à jour l'interaction prévue (`self._intended_interaction`) avec l'interaction ayant la meilleure valence.
* Retourne l'action correspondante à l'interaction prévue pour le prochain cycle.

5.2 Résultat

```
a = Agent4(interactions)
e = Environment1()
outcome = 0
for i in range(20):
    action = a.action(outcome)
    outcome = e.outcome(action)
```

```
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: 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: 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: 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: 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)
```

Figure 8: TRACE Agent 4 avec Environment 2.

```
a = Agent4(interactions)
e = Environment3()
outcome = 0
for i in range(20):
    action = a.action(outcome)
    outcome = e.outcome(action)
```

```
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: 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: 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: 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: 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)
```

Figure 9: TRACE Agent 4 avec Environment 3.

```
a = Agent4(interactions)
e = Environment1()
outcome = 0
for i in range(20):
    action = a.action(outcome)
    outcome = e.outcome(action)
```

```
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: 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: 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: 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: 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)
```

Figure 10: TRACE Agent 4 avec Environment 1.