TP Bandits Manchots

BARREYRE Alexis – FAUCHERY Benoit – GOJARD Romain

# Exercice 1 : Le bandit manchot

1.

class Bandit:

    def \_\_init\_\_(self):

        self.avg = random.gauss(0, 1)

2.

def play(self):

        return random.gauss(self.avg, 1)

3.

from classes import Bandit

import matplotlib.pyplot as plt

bandit1, bandit2, bandit3 = Bandit(), Bandit(), Bandit()

points1, points2, points3 = [], [], []

for i in range(1000):

    value1, value2, value3 = bandit1.play(), bandit2.play(), bandit3.play()

    points1.append(value1) # affichage matplotlib

    points2.append(value2)

    points3.append(value3)

    print(value1, value2, value3)

# Exercice 2 : Le Ban-10

1.

class BanDix:

    def \_\_init\_\_(self):

        self.tab = []

        for i in range(10):

            self.tab.append(Bandit())

2.

def \_\_init\_\_(self):

        self.tab = []

        self.maxAvg = None

        self.maxBanditIndex = 0

        for i in range(10):

            newBandit = Bandit()

            self.tab.append(newBandit)

            if self.maxAvg is None or newBandit.avg > self.maxAvg:

                self.maxAvg = newBandit.avg

                self.maxBanditIndex = i

        self.banditMaxAvg = self.maxBanditIndex

3.

def play(self, arm\_number):

        if arm\_number > 9 or arm\_number < 0:

            raise ValueError("Valeur impossible, erreur")

        return self.tab[arm\_number].play()

# Exercice 3 : Algorithme ε-greedy

1.

class GreedyPlayer:

    def \_\_init\_\_(self, n, eps):

        self.n = n

        self.eps = eps

2.

self.action\_values = [0] \* 10

      self.eval\_count = [0] \* 10

3 & 4.

def get\_action(self):

        explore = random.random() < self.eps

        return explore

5.

def \_greedy\_action(self):

        best\_actions = []

        highest\_value = -float("inf")

        for i in range(len(self.action\_values)):

            if (self.action\_values[i] > highest\_value):

                best\_actions = []

                best\_actions.append(i)

                highest\_value = self.action\_values[i]

            elif (self.action\_values[i] == highest\_value):

                best\_actions.append(i)

        return random.choice(best\_actions)

6.

def \_random\_action(self):

        return random.randint(0, self.n - 1)

7.

if explore < self.eps:

            self.\_random\_action()

        else:

            self.\_greedy\_action()

8.

def reward(self, action, reward):

        self.eval\_count[action] += 1

        self.action\_values[action] += (reward - self.action\_values[action]) / self.eval\_count[action]

9. & 10

ban10 = BanDix()

greedy = GreedyPlayer(0.1)

for i in range(1000):

    action = greedy.get\_action()

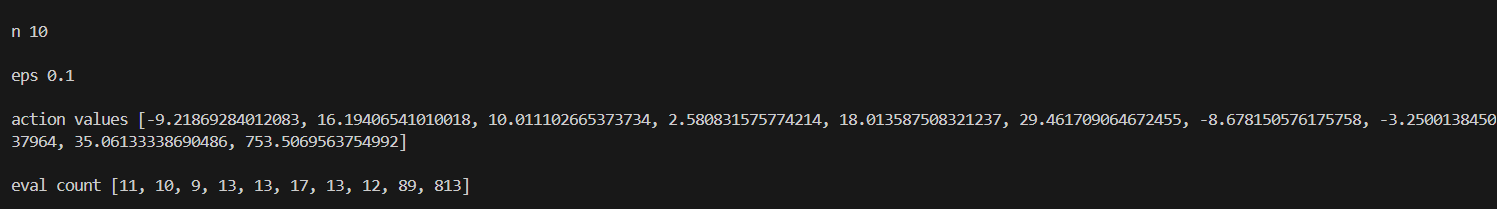
    reward = ban10.play(action)

    greedy.reward(action, reward)

    ban10.\_\_str\_\_()

    greedy.\_\_str\_\_()

11.



# Exercice 4 : Graphiques simples

1.

points = []

for i in range(1000):

    action = greedy.get\_action()

    reward = ban10.play(action)

    points.append(reward)

    greedy.reward(action, reward)

    ban10.\_\_str\_\_()

    greedy.\_\_str\_\_()

# Créer 1 sous-graphique

fig, axs = plt.subplots(1, 1, figsize=(8, 12))

axs.plot(range(1, 1001), points, label='Rewards')

axs.set\_xlabel('i')

axs.set\_ylabel('value')

axs.set\_title('Rewards')

axs.legend()

# Afficher les graphiques

plt.show()

2. & 3.

nbOfGreedy = 2000

for i in range(nbOfGreedy):

    tableBan10.append(BanDix())

    tableGreedyP.append(GreedyPlayer(0.1))

    print([tableBan10[-1].tab[j].avg for j in range(10)])

for i in tqdm(range(1000)):

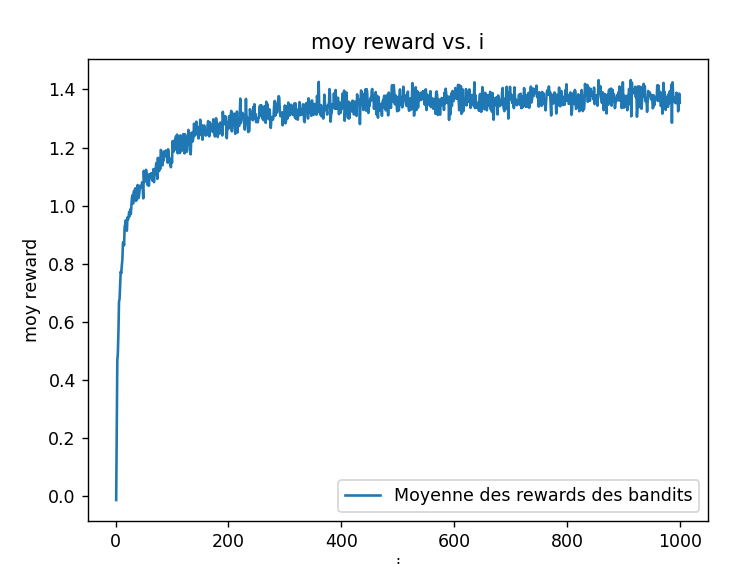
    for j in range(nbOfGreedy):

        action = tableGreedyP[j].get\_action()

        reward = tableBan10[j].play(action)

        tabPointsDesGreedy[i] += reward / nbOfGreedy

        tableGreedyP[j].reward(action, reward)



4.

for i in tqdm(range(1000)):

    for j in range(nbOfGreedy):

        action = tableGreedyP[j].get\_action()

        reward = tableBan10[j].play(action)

        tabPointsDesGreedy[i] += reward / nbOfGreedy

        tableGreedyP[j].reward(action, reward)

        if action == tableBan10[j].maxBanditIndex:

            tabPourcentageGreedy[i] += 1 / 20

5.

nbTabTabGreedy = 3

nbOfGreedy = 2000

tabPointsDesGreedy = [[0] \* 1000 for \_ in range(nbTabTabGreedy)]

tableTableBan10 = [[] for \_ in range(nbTabTabGreedy)]

tableTableGreedy = [[] for \_ in range(nbTabTabGreedy)]

tabTabPourcentageGreedy = [[0] \* 1000 for \_ in range(nbTabTabGreedy)]

for k in range(nbTabTabGreedy):

    for i in range(nbOfGreedy):

        tableTableBan10[k].append(BanDix())

        if k == 0:

            eps = 0

        elif k == 1:

            eps = 0.01

        else:

            eps = 0.1

        tableTableGreedy[k].append(GreedyPlayer(eps))

for i in tqdm(range(1000)):

    for k in range(nbTabTabGreedy):

        for j in range(nbOfGreedy):

            action = tableTableGreedy[k][j].get\_action()

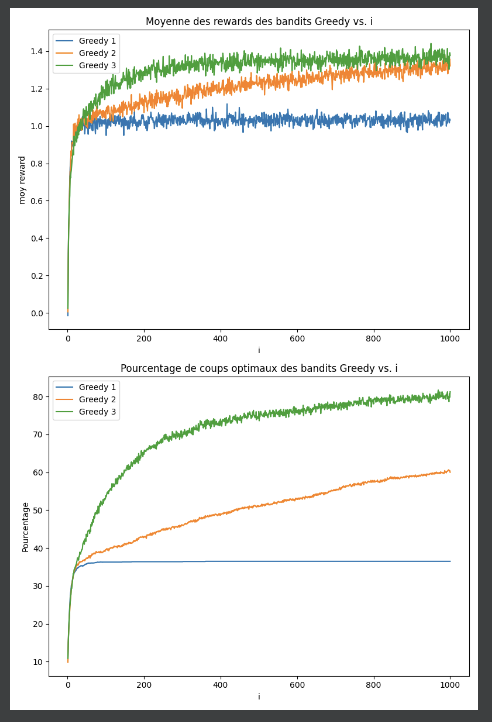
            reward = tableTableBan10[k][j].play(action)

            tabPointsDesGreedy[k][i] += reward / nbOfGreedy

            tableTableGreedy[k][j].reward(action, reward)

            if action == tableTableBan10[k][j].maxBanditIndex:

                tabTabPourcentageGreedy[k][i] += 1 / (nbOfGreedy / 100)



6. On constate que pour la courbe bleue (epsilon = 0), on n’a pas d’exploration donc la courbe stagne autour de 35% de choix optimal. La courbe orange (epsilon = 1) représente une exploration totale, permettant une croissance légère mais bloquant autour de 55%.

# Exercice 5 : Initialisation optimiste

1. & 2.

class OptimistGreedyPlayer:

    def \_\_init\_\_(self, eps, n=10):

        self.n = n

        self.eps = eps

        self.action\_values = [5] \* n

        self.eval\_count = [0] \* n

3.

Une image contenant texte, capture d’écran, ligne, Tracé

Description générée automatiquement

*Courbes de la classe OptimistGreedyPlayer*

5.