10 本バンディッド問題

問題

n 本のアームを持つバンディットマシン（スロットマシン）がある。各アーム(a_1~a_n)にはガウス分布(平均:0,分散:1)に従い報酬の期待値$Q^*(a_k)$が設定されている。アームの報酬もまたガウス分布(平均:$Q^*(a_k)$,分散:1)に従い、引いたときに報酬が与えられる。アームをある回数引き続けるとき、合計報酬の期待値を最大にするには、どのようにアームを選択し続ければいいか。

解決策

標本平均手法

各アームに対し、実際に受け取った報酬の平均を算出し、その値を$Q_t(a)$とする。

引用：講義スライド 第 9 回強化学習

> ある行動が選ばれたときに実際に受け取られた報酬を平均化する。
> t回目のプレイにおいて、それまでの間に行動aがk_a回選択されていて、それぞれの回で報酬が$r_1, r_2, \dots, r_{k_a}$得られているとする。

$$Q_t(a) = \frac{r_1 + r_2 + \cdots + r_{k_a}}{k_a}$$

グリーディ手法

標本平均手法の$Q_t(a)$が最大となるアームを選択する。

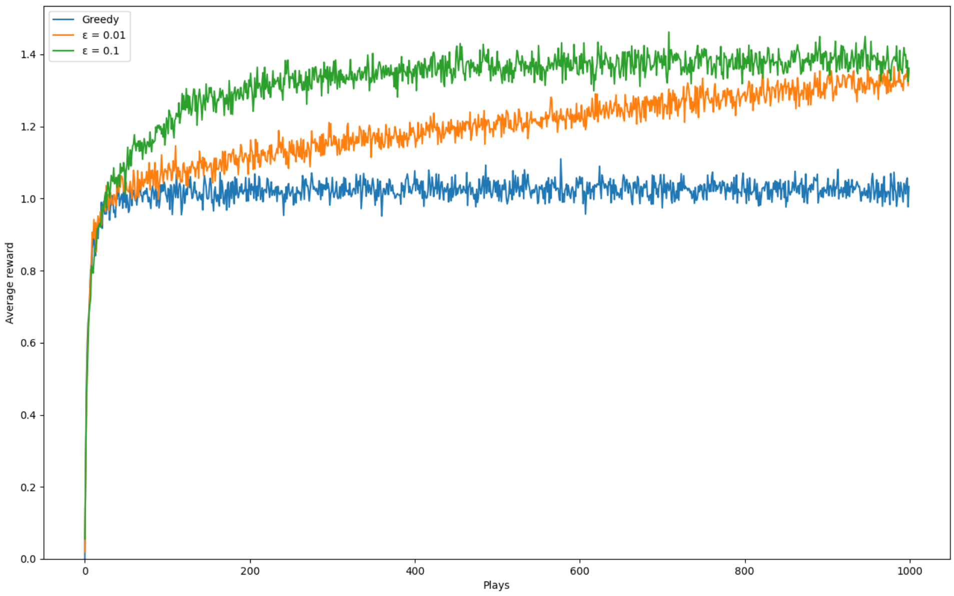
εグリーディ手法

グリーディ手法に加え、たまに(小さい確率εで)$Q_t(a)$とは関係なく無作為にアームを選択する。

実施したシミュレーション

- アームの本数：10 本
- アームを引く回数：1000 回
- 試行回数：2000 回
- 各アームに対する報酬の推定値の初期値は全て 0
- グリーディ手法、εグリーディ手法（ε=0.1, 0.01）の 3 回実施する。

上記の手法に従い 1000 回アームを選択し、その報酬をリストに記録していく。これを 2000 回繰り返し、1~1000 回目の報酬の平均値($\overline{r_1}, \overline{r_2}, \dots, \overline{r_{1000}}$)をグラフにプロットする。下図はその結果である。



この図から得られる合計報酬の大きさは

$$\varepsilon = 0.1 < \varepsilon = 0.01 < \text{グリーディ手法}(\varepsilon = 0)$$

のようになる。

その理由はこのように考えられる。グリーディ手法では、報酬の期待値$Q^*(a_k)$が本来高いアームが、序盤にたまたま低い報酬を与えたとき、「このアームの報酬の期待値は低い」と誤って推測し、それ以降そのアームを選択しなくなるため、合計報酬は最も低くなる。そこでεグリーディ手法では、確率εで無作為にアームを選択することで見過ごす可能性を減らせるため、εが高いほど合計報酬は高くなる。ただし ε=1 とすると必ず無作為にアームを選択し続け、報酬平均は0に収束してしまうため、適度な値を設定する必要がある。

ソースコード

```
import numpy as np
import random
import matplotlib as mpl
import matplotlib.pyplot as plt
mpl.use('tkagg')


PLAY_CNT = 1000     # プレイ回数


def greedy(epsilon):
```

```
    rewardsList = np.empty((0, PLAY_CNT), int)  # 1000回分の報酬*2000回の施行
    for j in range(2000):
        QaActuals = np.random.randn(10)    # Qaの報酬の期待値
        r = [[],[],[],[],[],[],[],[],[],[]]    # アームaを選択したときの報酬リスト
        QaEstimates = [0]*10    # Qaの報酬の期待値の推定値
        rewards = []    # 1000回分の報酬

        for i in range(PLAY_CNT):
            a=0
            if epsilon < random.random():
                QaEstimatesMaxIndexs = [i for i, v in enumerate(QaEstimates) if
v == max(QaEstimates)]
                a = random.choice(QaEstimatesMaxIndexs)    # 推定値が高いアーム
            else:
                a = random.randint(0, 9)    # ランダムに選択したアーム

            li = np.random.normal(QaActuals[a], 1, 1)   # aの報酬
            reward = li[0]
            (r[a]).append(reward)
            rewards.append(reward)
            raLen = len(r[a])
            QaEstimate = (QaEstimates[a]*(raLen-1) + reward)/raLen  # Qa算出（計
算コストの高いsum関数は使わず工夫）
            QaEstimates[a] = QaEstimate

        rewardsList = np.append(rewardsList, np.array([rewards]), axis=0)
        print(j)

    return rewardsList

epsilons = [0, 0.01, 0.1]  # グリーディ手法、εグリーディ手法（ε=0.01, 0.1）
labels = []    # グラフの凡例
for epsilon in epsilons:
    if epsilon == 0:
        labels.append("Greedy")
    else:
```

```
        labels.append("ε = {}".format(epsilon))

for epsilon in epsilons:
    rewardsList = greedy(epsilon)
    rewardsMeans = rewardsList.mean(axis=0) # 2000回分の平均
    x = list(range(len(rewardsMeans)))
    plt.plot(x, rewardsMeans, label = labels.pop(0))

plt.xlabel('Plays')
plt.ylabel('Average reward')
plt.ylim(bottom=0)
plt.legend(loc=0)
plt.show()
```