プログラミング演習
～総合演習 1～

1　目的

自分の”考え”を”プログラム”として実装する．

1.1　バンディットというゲーム

これまでに扱ってきたバンディットは「ゲーム」である．ゲームにはプレイヤーがおり，勝ち負けが存在する．これまでの演習では，人間がプレイヤーであり，バンディットの腕の選択を行っていた．バンディットというゲームにおける勝ち負けは，沢山報酬を得たものが勝者であり，報酬を得られなかったものが敗者である．このように，バンディットというゲームがあり，プレイヤーがおり，このゲームを「解く」とはより高い報酬を得るための腕を選択することである．

1.2　ゲームを解くプログラムの作成

これまでにバンディットというゲームを対象に，for 文やポインタ，関数など C 言語の練習をしてきた．ここでは，バンディットというゲームを解くプログラムを作成することを目的とする．そのための方法として，あるバンディットを対象に (例えば bandit00.o)

1. 自分でバンディットを解いてみる
   - これまでの演習で作成した「10 回プレイするバンディット」を使って高い報酬を得られるように自分でプレイしてみる
2. どのように腕を選べば高い報酬を得られるのか言葉で説明してみる
3. 自分で説明した言葉をプログラムとして作ってみる

というように，まず自分で解いてみて，得られた「自分の考える解き方」をプログラムとして実装することである．

2　事始め:人間によるプレイ

こでは，まず自分でプレイし，バンディットを解いてみる．

2.1 準備

bandit00.o を対象に，(これまでに演習で作成している)「10 回プレイするバンディット」を用意する．

2.2 bandit00 を解く

bandit00 をプレイし，高い報酬を得られるように腕を選択する．どのように選べば高い報酬を得られるか考え，言葉にする．

2.3 bandit01,02,... を解く

bandit00 以外のバンディットもプレイして，どのように腕を選べば高い報酬を得られるか考え，言葉にする．

2.4 注意:「正解」について

バンディットはゲームである．そのため，「正解」を教えない．考えた腕の選択の仕方に，「正解」はなく，正しいかどうかも分からない．何を拠り所にするかというと，「他人との比較」を用いる．つまり，自分の考えた方法でバンディットをプレイした場合に得た総報酬量が，

- 他の人の総報酬量より多ければ自分の考えた方法はより良い
- 他の人の総報酬量より少なければ自分の考えた方法は改善の余地がある

と考える．
正解がない (絶対評価がない) ので，他者との比較 (相対評価) によって自分の評価を表す．

2.5 スコア計算法

他者との比較の時に大事なのは，同じ様に比較することである．例えば，

- 自分は 10 回の合計である総報酬量を提示し，相手が 100 回の合計である総報酬量を提示して比較してもよいか？
- 自分は 10 回の平均である報酬値を提示し，相手は 10 回のうちの最大報酬値を提示して比較してもよいか？
- 自分は最初の 10 回の総報酬量を提示し，相手は 100 回行ったうちの最後の 10 回分の総報酬量を提示して比較してもよいか？

など，異なるルールの基に得た値を比較すると，相手よりも良いのか悪いのか分からない．
本演習では，比較できる報酬の値 (スコア) の計算方法を以下のようにする (詳しくは付録 付録 A を参照)．

- プレイヤープログラムは 10 万回バンディットをプレイする
- 連続した 1 万回のプレイで得た報酬の合計を「部分報酬量」とする
- 10 万回の中で連続した 1 万回の選び方は沢山あるが，全ての場合を考え，最も高い「部分報酬量」をス

コアとする

これは，最初から最後まで色々試しながらプレイする仕方でも，前半でデータを集め，後半で勝負をするプログラムでも，同じ様に比較できるように考え, 全プレイの中の「一部分」の値で比較することとした計算方法である．

Web ページ上の「スコア (演習の成績とは関係なし)」には，過去の夜間主コースの学生や TA の学生のスコアも記載しているので，参考にすること．

3 製作対象 バンディットプレイヤープログラム

ここでは，2.2 で考えた bandit00 用のプレイヤープログラムを作成する．プレイヤープログラムは，自分で考えた「腕をどのように選ぶのか」をプログラムにしたものである．そのひな形を 3.2 に用意した．

ひな型のうち，大部分はスコアを計算する部分である．特に 19 行目から始まる for 文で 10 万回のバンディットプレイを行っており，32 行目-43 行目で連続した 1 万回の部分報酬量を計算している．最終的に最も高い部分報酬量を 46 行目で出力している．これらの部分はスコアの計算であり，改変せずに使用する．

今回製作するプログラムは，自分で考えた「腕をどのように選ぶのか」である．プログラム中では 25 行目の

- select_arm =

の変数に番号をいれることで，選択する腕をきめる．

本演習では，この select_arm にどのような値を入れ，どのように腕を選択するかプログラムする．基本的には select_arm 前後にプログラムを挿入し，「自分の考えた腕の選択法」をプログラム化することを本演習の目的とする．

3.1 準備

ディレクトリ ”programming13”を作成する．今回の演習では，プログラムの作成や必要ファイルのダウンロードは，programming13 ディレクトリで行う．

3.2 プログラム

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include "bandit.h"
4
5  #define MAX_TRIAL 100000
6
7  int main(){
8    /* 変数定義・初期化 */
9    int i,j,select_arm=0;
10   double reward=0.0,score[10000], max_score=0.0, tmp_score;
11   for(i=0 ; i<10000 ; i++){
```

```
12        score[i]=0.0;
13      }
14
15      init_bandit(); /* バンディットの初期化 */
16
17      /* MAX_TRIAL 回まで自動実行 */
18      /* 連続した 10000 回のうち最大のスコアを自動計算・更新 */
19      for(i=0 ; i<MAX_TRIAL ; i++){
20
21        /* 意思決定・それによるバンディットの実行*/
22        /* ここから自作のプログラム                */
23        /* 選択する腕を決めて select_arm に代入して*/
24
25        select_arm =
26
27        /* ここまで自作のプログラム                */
28
29        reward = bandit(select_arm);
30        if(reward < 0.0) reward = 0.0;
31
32        /* 連続した 10000 回の最大スコアの確認 */
33        tmp_score=0.0;
34        for(j=0 ; j<10000 ; j++) tmp_score += score[j];
35        if(tmp_score > max_score) max_score = tmp_score;
36
37        /* 連続した 10000 回のスコアを更新            */
38        /* score[0] 〜score[9999] に対し,           */
39        /*   最も古いもの score[9999] を消し,       */
40        /*   一個ずつずらし (score[j] = score[j-1]) */
41        /*   最も新しいものを score[0] に入れる      */
42        for(j=9999 ; j> 0 ; j--) score[j] = score[j-1];
43        score[0] = reward;
44      }
45
46      printf("最大総獲得報酬: %lf\n", max_score);
47      return 0;
48    }
```

3.2.1 注意

以下のことに注意してプログラムの作成を行うこと．

- 自分で作成するプログラムの中で，バンディット関数・bandit() を使用してはいけない
  - バンディット関数・bandit() を一回実行することは，バンディットのゲームを一回プレイすることである．select_arm を決めるための「自分で考えた腕の選択法」の場所でバンディット関数・bandit() を使用するということは，「実際にバンディットをプレイした結果どうなるか考え，本番のバンディットプレイに望む」ことになり，スコア計算法の「連続したプレイの結果の部分報酬量」の計算が成り立たなくなってしまう．
- srand 関数をしようしてはいけない
  - srand 関数はプログラムの中で「一回だけ実行」するように出来ている．srand 関数は init_bandit の中で実行しているので，再び使用するとプログラムが期待通り動かなくなる可能性がある．

3.3 コンパイル

上記のプログラムを作成し，ファイル名 player00.c として保存してコンパイルを行う．コンパイル後の名前を gameplay00 とする．

> gcc -o gameplay00 player00.c bandit00.o

3.4 動作実験

作成したプログラムを実行させ，スコアを出力させる．またより良いスコアを出せるようにプレイヤープログラムを改良する．

3.5 やってみよう

bandit01 に対してもやってみよう

付録 A

プレーヤーが N 本腕バンディットのレバーを一回選択する行為を”1 試行”という．この時，連続した 10000 試行の結果得た報酬の合計をプレーヤーのスコアとする．ただし，第一回目の試行から計算する必要はなく，試行途中の好きな時点からの試行でよい．

基本的には，一回目の試行から計算を行い，全試行を通して最大の報酬をスコアとする．全試行数 10，連続した 3 試行の結果得た報酬の合計をスコアとした場合の例を以下に示す．

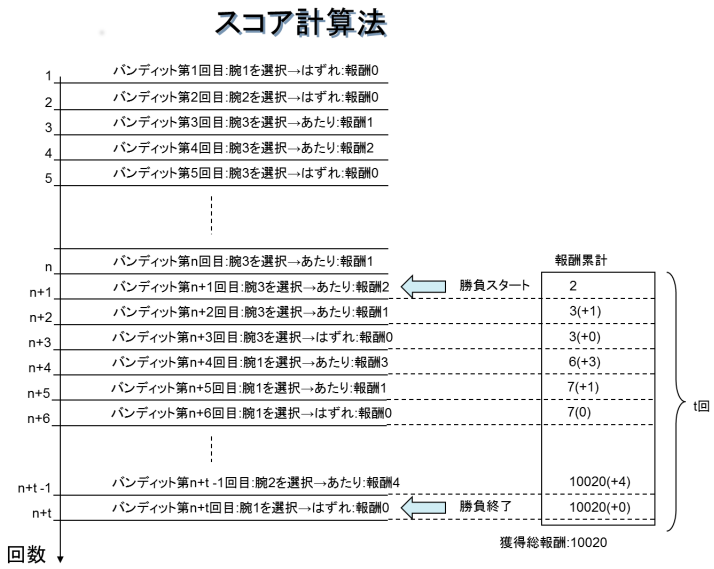

図 1　スコア計算法

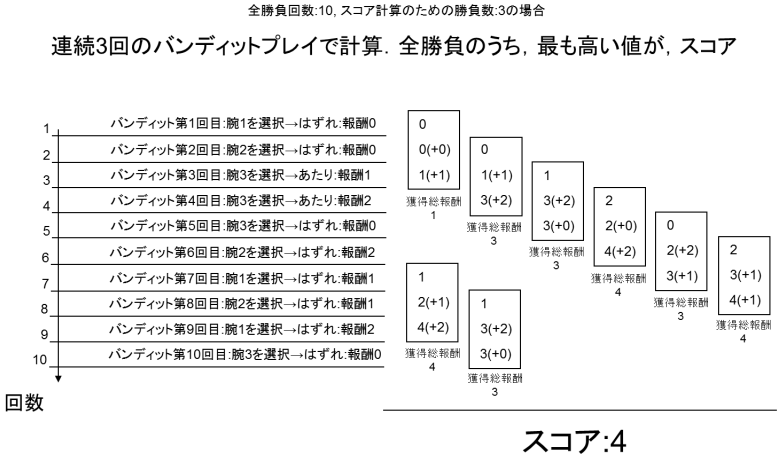

図 2　スコア計算法の例