勉強したこと

遺伝的アルゴリズムから強化学習へ

マルシェ

8queen問題

4回生　田中　雅也

遺伝的アルゴリズムと強化学習との違い

- 簡単に言えば、遺伝的アルゴリズムはし生物の自然淘汰による進化を取り入れたアルゴリズムで、強化学習は生物の脳を取り入れたアルゴリズムになっている。
- 触ってみて感じたのは、どちらも対応する状況で向き不向きはあるが、汎用性が高く、将来性があるのは強化学習だと感じた。
- ただ、難易度は圧倒的に強化学習の方が上。
- それに加えて、深層強化学習も存在していて、これも難しい。

Q学習の軽い紹介

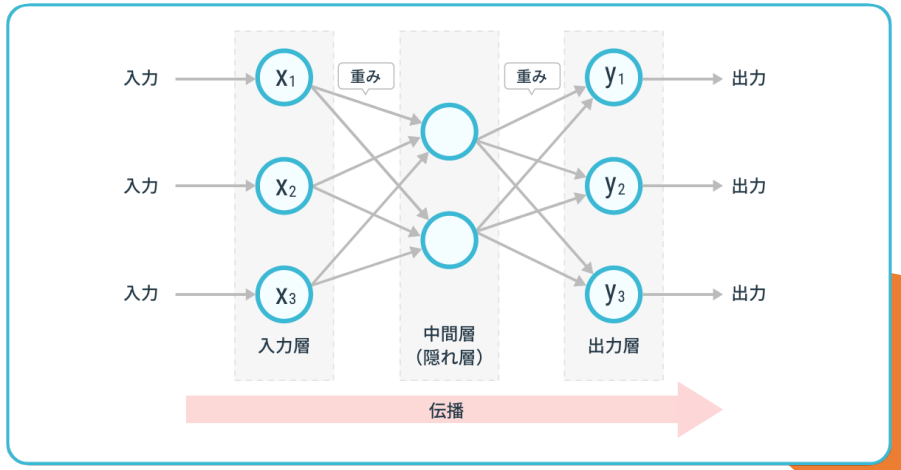
- 例えば、コリドータスクがあるとする。
- コリドータスクは4つのマスがあり、そこにはロボットと宝石がある。ロボットは前に進むか宝石を拾うという行動を起こすことができる。行動を起こすごとに負の報酬(-1)を受け取り。、宝石を拾うことができれば、正の報酬(5)を受け取ることができる。Dただし、壁際で進むや宝石がない場所で拾うという行動を起こしてしまうと、その時点でタスクは終了する。
- Q学習は学習によって正しいQテーブルを導き出すというもの。Qテーブルとは、期待値を状況と行動ごとに求めたもの。
- Qテーブルの更新方法については割愛。

例

- 学習後のロボットはQテーブルの高い方の行動を選ぶ。
- ロボットがこの状況にある時、この場所で拾うという行動を取ってしまうと、報酬は(-1)で終わってしまう。しかし、進むという行動をすると、報酬は(-1)だが、この後に宝石を拾うことができるという未来が存在するため、Qテーブル(期待値)的には進むという行動の方がQテーブルの値が高いということになる。したがって、ロボットは進むという行動を取るようになる。

netQ学習

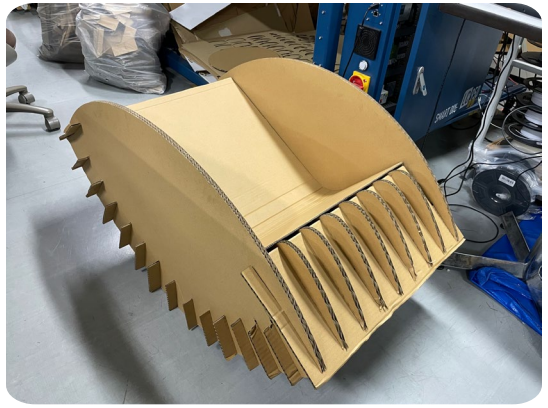
- 今紹介した、Q学習はQテーブルにない状況があれば、数値が設定されてないのだから正しい選択をできない。これを解決するのが、ニューラルネットワーク(いわゆる深層学習)。
- 簡単に説明すると、あらかじめQテーブルを作成するのではなく、観察した状況から行動に応じたQテーブルを求める中間層を学習させるというもの。
- もし正しく学習できていれば、ロボットが知らない状況でもQテーブルが作成され、正しい選択をできるようになる。



マルシェ

- 試作1号
- 課題点:内側から外側への力が加わり、裂けた。座り心地が悪い。

- 試作2号
- 改善点:横材を上から出なくしたから入れた。縦材の枚数を2倍にした。座りやすくなった。



先生からのお題

非線形・離散的・パターンマッチング

８クイーン問題から発案

この後に紹介する配置よりも良い評価値だったら何かしらの一杯を！

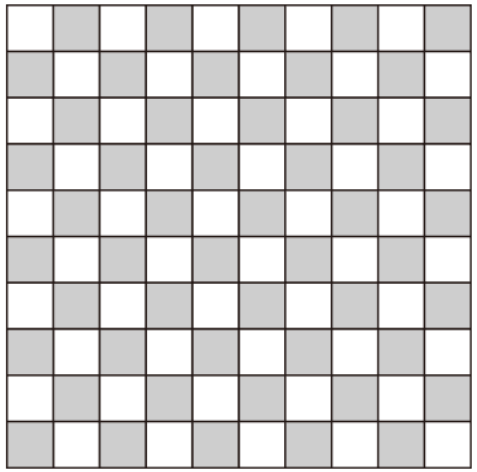
GO Poll 08_b(manaba)　適合する配置計画を立案しなさい。

1. 問題：10マス✖10セルの正方形にユニット空間を配置する。

2. 制約条件
    1. ユニット空間の開口部の直線上にいずれのユニットも配置しない。

3. 設定
    1. ユニットは下記の4種類。開口部はいずれも両側にある。
        1. A：行方向（X軸方向）に窓がある。
        2. B：列方向（Y軸方向）に窓がある。
        3. Cは左上及び右下の対角方向に窓がある。（比例・相関）
        4. Dは左下及び右上の対角方向に窓がある。（半比例・逆相関）
    2. 視認距離は無限である。開口の壁上の位置や大きさに意味はない。

4. 評価(関数)
    1. 成立するユニットの総数を高く
    2. ユニット毎の数の最大値と最小値の差を小さく
    3. 1.と2.の重みは同じ



所感

- 遺伝的アルゴリズムでは、解きにくい問題。
- 遺伝子の表示方法がわからない。gridでのunitの場所を遺伝子情報にした場合、どのように交叉させればよいのかがわからない。あるいは、unitの入れる順番を遺伝子情報にした場合も考えたが、その場合、最適化の目的にunitの数を増やすというものがあるため、遺伝子の数が増えたり減ったりするという問題がある。この問題を解決したとしても、順番がこの問題を解く手掛かりになるとは思えない。

貪欲法

- その場ででの最善と思われる行動を繰り返していく行動のこと。ランダムに配置していって、入れれない場所を消していくという方法を取った。
- gridを作り、対応するアルファベットを配置、入れれない場所には１を配置していく。0がなくなるか、終了子ウントが一定の値になるまで繰り返す。65:A,66:B,67:C,68:D

```
[[ 0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  1]
 [ 0  0  0  0  0  0  0  0  1  0]
 [ 0  0  0  0  0  0  0 68  0  0]
 [ 0  0  0  0  0  0  1  0  0  0]]
```

```
[[ 0  0  0 67  0  0  0  0 66  1]
 [ 1  1  1  1  1  1  1 65  1  1]
 [67  0  0  0  0  1  0  1  1  0]
 [67  1  0  0  0  0  1  1  1  0]
 [ 0  1  1  0  0  1  1  1  1  0]
 [ 0  0  1  1 68  1  0  0  1  0]
 [ 0  0  0  1  1  0  0  0  1  1]
 [ 0  0  1 68  1  1  0  0  1  0]
 [ 0  1  1  0  0  1  1 68  1  0]
 [ 1  1  1  1  1  1  1  1  1 65]]
```

```
[[ 0 66 66 67 68 67  1  1 66  1]
 [ 1  1  1  1  1  1  1 65  1  1]
 [67  1  1  0  0  1 66  1  1 68]
 [67  1  1  0  1  0  1  1  1  0]
 [ 1  1  1  1 65  1  1  1  1  1]
 [67  1  1  1 68  1  1  0  1  1]
 [ 1  1  1  1  1  1  1 65  1  1]
 [68  1  1 68  1  1  1 68  1 68]
 [67  1  1  1  0  1  1 68  1 68]
 [ 1  1  1  1  1  1  1  1  1 65]]
```

貪欲法

- 1000回ほど回して、top10を選出したらまぁまぁいい配置が見つかった。
- (6,5,7,7)=25-2=23
- (5,6,6,6)=23-1=22
- バックトラックでは、どこでその選択が違うのかを判定するタイミングなどが分からなかった。コンピュータのゴリ押しになっちゃった。