簡素な遺伝子アルゴリズム

2019 年 3 月 17 日

目次

1 初めに-遺伝子アルゴリズムと AI 1

1.1 Script の実行と結果 1

1.2 簡単な遺伝子アルゴリズム 2

2 genetic_main の組み上げと実行 4

3 分析 5

1 初めに-遺伝子アルゴリズムと AI

統計学に大数の法則すなわち「数多くの試行を重ねることにより事象の出現回数が理論上の値に近づく定理」というものがある。

AI のエンジンにニューロ、遺伝子アルゴリズム、ディープラーニングなどがあるがいずれも最適解を得るのではなく、数多くの試行から得られた結果が最適解の周辺に近づくことで、この大数の法則と同様の感覚を持っている。

日経ソフトウエア 2019 年 3 月号の特集に「人工知能の強力手法ー遺伝子アルゴリズム」が載っている。著者は本田啓一郎。大学院で遺伝子アルゴリズムを研究したと紹介されている。

最初の記事は遺伝子アルゴリズムの原理を知る「OneMax] 問題で Python のパッケージを利用しているが記事を参照しながら最初から J で作成してみる。

1.1 Script の実行と結果

- ある試行：12 ビットで５０回試行した結果
- 内 35 回は 12 ビットまで達しない
- 最良は 6 回の GA で 12bit 達成（2 回）

```
|: genetic_analysis 50 12
```

```
+----+----+----+-----+-----+-----+-----+-----+NB. GA 実行回数
|6 12|8 12|9 12|10 12|11 12|12 10|12 11|12 12|NB. クリアしたビット数
+----+----+----+-----+-----+-----+-----+-----+
|2   |2   |2   |5    |1    |5    |30   |3    |NB. 同じ結果の個数
+----+----+----+-----+-----+-----+-----+-----+
```

1.2 簡単な遺伝子アルゴリズム

記事では 10 ビットの遺伝子で構成された染色体をもつ個体としている。遺伝子といえば ACGT で基底 4 だが、何が優劣かもわからないので単純に 0/1 の構成とし、12 ビットで 1 個体とした。

最初は 12 ビットモデルで作成し、フリービットモデルへの拡張は後に行う。

1. 一個体の発生。12 の因子を持ち、0/1 を半分ずつ持つ。

```
   gen12=: 3 : ' ;(6#2)?2'


   gen12 ''
1 0 1 0 1 0 1 0 0 1 1 0
```

2. 次で 6 個体を生成。 >: で 1 をオリジンとする。
   setup ” で 12 個体を生成する

```
   gen12 ^:(>:i.6) ''
0 1 0 1 1 0 1 0 1 0 1 0
1 0 1 0 1 0 0 1 1 0 0 1
0 1 1 0 1 0 0 1 0 1 0 1
1 0 0 1 1 0 0 1 1 0 0 1
0 1 0 1 0 1 1 0 0 1 0 1
0 1 1 0 0 1 1 0 1 0 0 1
```

3. カップリング。2 組づつ乱数で組合わせる
   `coupling=: 3 : ' (12 ? 12) { y'`
4. 2 組に切り分ける

```
   ,. cut12 setup ''
+----------------------+
|1 0 1 0 0 1 0 1 0 1 1 0|
```

```
|0 1 1 0 1 0 1 0 1 0 0 1|
+----------------------+
|1 0 0 1 0 1 0 1 0 1 0 1|
|1 0 0 1 0 1 1 0 0 1 0 1|
+----------------------+
|1 0 0 1 0 1 0 1 1 0 0 1|
|0 1 0 1 1 0 0 1 0 1 1 0|
+----------------------+
........
```

5. 交叉箇所の決定　XY の遺伝子の交叉箇所を選ぶ

```
   crossindex ''
0 1 0 1 0 1 1 0 0 1 0 1
1 0 1 0 1 0 0 1 1 0 1 0

crossindex=: 3 : '  tmp,: -. tmp=. gen12 y'
```

6. 交叉　次世代を 6 個体生成

```
   cross setup''
1 0 0 1 0 1 1 0 0 1 1 1
1 1 0 0 1 0 0 1 1 0 0 1
0 1 1 1 0 1 1 1 0 0 1 0
1 0 1 1 1 1 0 1 0 1 0 0
1 1 1 1 1 0 1 0 1 1 1 1
0 1 1 1 0 0 0 1 0 0 1 1

cross=: 3 : 0
NB. cross XY
NB. Usage: cross setup ''
CutG12=: {@|: L:0 cut12@coupling y NB. {@|: make vertical 2 pair
Ind=: {@>gen12 ''                  NB.
 >>(<Ind) { (L:0) CutG12           NB. using both dualbox
)
```

7. 突然変異 交叉で誕生した 6 の個体に夫々乱数で選んだ 1 要素を反転する

```
   Rand  NB. 各個体の変異の箇所
+-+-+--+-+-+-+
|0|2|11|1|7|6|
+-+-+--+-+-+-+


   tmp1  NB. 反転後のビット。これを書き戻す
+-+-+-+-+-+-+
|0|0|1|1|0|0|
+-+-+-+-+-+-+
```

8. 最初の 12 個体と生成した 6 個体をソートし上位 12 個体を選抜する単純再生産型

2 genetic_main の組み上げと実行

- genetic の　核。（生成）クロス、突然変異、選択のプロセスを一回行う

```
genetic0=: 3 : 'select  y,mutation@cross y'

   genetic0 setup ''
0 1 1 1 0 0 0 1 1 1 0 1
1 1 0 0 1 0 1 1 0 1 1 0
1 1 1 0 1 1 1 0 0 0 0 1
0 1 1 0 1 1 0 0 0 1 1 1
1 0 0 1 0 1 1 0 1 0 0 1
0 1 1 0 0 1 0 1 1 0 1 0
0 1 1 0 1 0 0 1 0 1 1 0
0 1 0 1 1 0 0 1 0 1 0 1
0 1 0 1 1 0 0 1 0 1 1 0
0 1 1 0 0 1 1 0 1 0 1 0
1 0 0 1 1 0 1 0 0 1 1 0
0 1 1 0 1 0 1 0 1 0 0 1
```

- バッチ。指定回数の反復。11（0 オリジン）に達したら停止する

```
genetic_main=: 3 : 0
NB. main batch
NB.usage: genetic_main 12
```

```
Repeat =.  y  NB. 50
T0=. setup ''
for_ctr. i. Repeat do.
 T0=.genetic0 T0
 sum=. ctr, {. +/ "1 T0
 if. 12 = {: sum do.  goto_end.  end.
end.
label_end.
sum

)
```

- 実行結果 (世代（max=12）ビット合計（max=12）

```
   genetic_main 12
3 12
   genetic_main 12
9 12
   genetic_main 12
11 12
   genetic_main 12
11 10
```

3　分析

1. 指定回数のうちどのステップで打ち止めかを分析する
2. 次の例は 12 世代までの遺伝子アルゴリズムを 50 回試行した結果。
   - 12 世代で 12 ビット達成は 15 回
   - 最速は 4 世代で 12 ビットに達した
   - 36 回 (1+5+6) は 12 ビット未達成
   - 最少は 9 ビットまでで終わった

```
     genetic_analysis 50 12
世代:達成ビット:出現回数
+-----+--+
|4 12 |1 |
+-----+--+
|5 12 |1 |
```

```
+-----+--+
|7 12 |1 |
+-----+--+
|8 12 |1 |
+-----+--+
|9 12 |1 |
+-----+--+
|10 12|3 |
+-----+--+
|11 12|2 |
+-----+--+
|12 9 |1 |
+-----+--+
|12 10|5 |
+-----+--+
|12 11|30|
+-----+--+
|12 12|4 |
+-----+--+
```

3. Script

```
count=: 3 : 0
NB. usage: count ANS
cutind=. ~: tmp0=./:~ y
tmp1=.cutind <;.1 tmp0
({. L:0 tmp1),.# L:0 tmp1
)

genetic_analy0=: 3 : 0
NB. Usage: genetic_anal  50 12
'times gen'=. y
ANS=: <0 0
for_ctr. i.times do.
temp=.genetic_main gen
ANS=. ANS,<temp
end.
```

```
>}.ANS
)

genetic_analysis=: count@genetic_analy0
```

Referecces

本田啓一郎「人工知能の強力手法ー遺伝子アルゴリズム」日経ソフトウエア 2019 年 3 月号
Script は http://japla.sakura.ne.jp の Workshop 2019/03 から DL できる