ヘッブの法則の応用

2017年5月12日

今回の内容

- ヘッブの法則の応用・コード解説。
- 輪読の開始(1-1、1-3)。
- アンケート（３年生で先週いなかった人）

ヘッブの法則の応用

- サンプルコードを基に応用をし、考察を行ってみる。
- その前に、プログラムの解説を行う。

init(hebbクラス内)

- 記憶サイズの定義。
- 記憶(脳内状態の設定)の初期化。
  size×sizeの零行列作成

```
def __init__(self, size):
    self.memory_size = size
    self.memory = reshape([0]*size**2, (size, size))
    print (self.memory)
```

```
h = hebb(3)
```

- サイズ3(神経3つ)の場合は、

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

となる。

learn関数(hebbクラス内)

- 受け取った神経を基に、脳内状態を変化させることで学習を行う。
- 学習の法則は第3回スライド参照。

```
def learn(self, neural_activity):
    if self.memory_size != len(neural_activity):
        return False

    for y in range(self.memory_size):
        for x in range(self.memory_size):
            if x == y:
                self.memory[x,y] = 0
            elif neural_activity[x] == 1 and neural_activity[y] == 1:
                self.memory[x,y] += 1
            elif neural_activity[x] == -1 and neural_activity[y] == -1:
                pass
            else:
                self.memory[x,y] -= 1
    print (self.memory)
    return True
```

learn関数(hebbクラス内)

① 両方の神経が同時に活動したならば、結びつきを強める。

② 両方の神経が同時に活動しなかった（片方が活動、もう片方が活動しない）ならば、結びつきを弱める。

③ 両方の神経がどちらも活動しないならば、変化なし。

④ 自分から自分へシナプスは伸びないので、a_{11}, a_{22} … は常に0とする。

```
def learn(self, neural_activity):
    if self.memory_size != len(neural_activity):
        return False

    for y in range(self.memory_size):
        for x in range(self.memory_size):
            if x == y:
                self.memory[x,y] = 0                                          ・・・④
            elif neural_activity[x] == 1 and neural_activity[y] == 1:         ・・・①
                self.memory[x,y] += 1
            elif neural_activity[x] == -1 and neural_activity[y] == -1:       ・・・③
                pass
            else:                                                             ・・・②
                self.memory[x,y] -= 1
    print (self.memory)
    return True
```

remember関数(hebbクラス内)

- 受け取った神経を基に、思い出しを行う。
- neural_activityの定義の際のreshapeは行列の掛け算を行うために行う。(神経が3の場合、1×3行列を3×1行列にする。)
  例

$$(1,1,-1) \Rightarrow \begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix}$$

```
def remember(self, neural_activity):
    if self.memory_size != len(neural_activity):
        return False

    neural_activity = reshape(neural_activity, (self.memory_size, 1))
    mind = dot(self.memory ,neural_activity)
    return reshape(mind / abs(mind), (1, self.memory_size))[0]
```

remember関数(hebbクラス内)

- mindは、脳内状態と先ほどの変形した神経を掛け合わせたものとなる。
  例

$$\begin{pmatrix} 0 & 3 & -3 \\ 3 & 0 & -3 \\ -3 & -3 & 0 \end{pmatrix} \times \begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix} = \begin{pmatrix} 6 \\ 6 \\ -6 \end{pmatrix}$$

neural_activity

```
def remember(self, neural_activity):
    if self.memory_size != len(neural_activity):
        return False

    neural_activity = reshape(neural_activity, (self.memory_size, 1))
    mind = dot(self.memory ,neural_activity)
    return reshape(mind / abs(mind), (1, self.memory_size))[0]
```

remember関数(hebbクラス内)

- returnするものは、
  神経は1と-1でしか表さないため、計算して現れた数字が、正の数なら1、負の数なら-1とする
  処理を行った上で、適切な形に戻したものとなる。(神経が3の場合、3×1行列を1×3行列にする。)
  例

$$\begin{pmatrix} 6 \\ 6 \\ -6 \end{pmatrix} \simeq \begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix} \Rightarrow (1,1,-1)$$

```
def remember(self, neural_activity):
    if self.memory_size != len(neural_activity):
        return False

    neural_activity = reshape(neural_activity, (self.memory_size, 1))
    mind = dot(self.memory ,neural_activity)
    return reshape(mind / abs(mind), (1, self.memory_size))[0]
```

remember関数(hebbクラス内)の注意

●0÷0が発生する可能性がある。

```
def remember(self, neural_activity):
    if self.memory_size != len(neural_activity):
        return False

    neural_activity = reshape(neural_activity, (self.memory_size, 1))
    mind = dot(self.memory ,neural_activity)
    return reshape(mind / abs(mind), (1, self.memory_size))[0]
```

learn関数

- 受け取った神経をhebbクラスのlearn関数に送り、覚えられたかか覚えられないか表示する。

```
def learn(na):
    if h.learn(na):
        print (str(na)+"を覚えました。")
    else:
        print (str(na)+"は覚えられません。")
```

```
learn([1,1,-1])
learn([1,1,-1])
learn([1,1,-1])
```

remember関数

- 思い出す対象の神経をhebbクラスのremember関数に送り、思い出せたか思い出せなかったかを表示する。

```
def remember(question):
    answer = h.remember(question)
    if any(answer):
        print ("%sから、%sを思い出しました！" % (question, answer))
    else:
        print (str(question)+"を思い出せません。")
```

```
remember([1,1,-1])
remember([1,1,0])
```

演習

① 神経の数を増やしてみる。
② まずAを3回覚えさせた後、Bを何回覚えさせるとBが常習化するか、神経の数を増やしつつ、回数の変化を調べる。

①のヒント・参考

- サンプルでは神経の数を指定している部分がある。
- また、その数に対応して学習させる内容の次元も変化することになる。

```
h = hebb(3)
learn([1,1,-1])
```

②のヒント・参考(1/5)

- 神経3つの時は、A[1,1,-1]を3回覚えてから、B[1,-1,1]を2回覚えたとしてもBを正しく思い出せなかった。Aの方が常習化している状態。

```
[1,  1,  -1]を覚えました。
[1,  1,  -1]を覚えました。
[1,  1,  -1]を覚えました。
[1,  1,  -1]から、[  1   1  -1]を思い出しました！
[1,  1,   0]から、[  1   1  -1]を思い出しました！
[1,  -1,  1]を覚えました。
[1,  -1,  1]を覚えました。
[1,  1,  -1]から、[  1   1  -1]を思い出しました！
[1,  -1,  1]から、[-1  -1   1]を思い出しました！
```

②のヒント・参考(2/5)

- A[1,1,-1]を3回覚えてから、B[1,-1,1]を3回覚えると、警告文が出る。

```
[1, 1, -1]を覚えました。
[1, 1, -1]を覚えました。
[1, 1, -1]を覚えました。
[1, 1, -1]から、[ 1  1 -1]を思い出しました！
[1, 1, 0]から、[ 1  1 -1]を思い出しました！
[1, -1, 1]を覚えました。
[1, -1, 1]を覚えました。
[1, -1, 1]を覚えました。
C:\Users\admin\workspace\p1\p1\hebb.py:37: RuntimeWarning: divide by zero encountered in divide
  return reshape(mind / abs(mind), (1, self.memory_size))[0]
[1, 1, -1]から、[ 0  1 -1]を思い出しました！
[1, -1, 1]から、[ 0 -1  1]を思い出しました！
```

②のヒント・参考(3/5)

- 計算結果を返す部分で、その結果に0が存在すると、0÷0という計算が発生してしまうために警告文が出る。

```
def remember(self, neural_activity):
    if self.memory_size != len(neural_activity):
        return False

    neural_activity = reshape(neural_activity, (self.memory_size, 1))
    mind = dot(self.memory ,neural_activity) #掛け合わせる
    return reshape(mind / abs(mind), (1, self.memory_size))[0]
```

②のヒント・参考(4/5)

●結果の値としては、AもBも思い出せそうで思い出せない、どっちつかずな状況になる。

```
[1, 1, -1]から、[ 0  1 -1]を思い出しました！
[1, -1, 1]から、[ 0 -1  1]を思い出しました！
```

②のヒント・参考(5/5)

- A[1,1,-1]を3回覚えてから、B[1,-1,1]を4回覚えると、Bを正しく思い出せる。Bの方が常習化したということになる。

```
[1, 1, -1]を覚えました。
[1, 1, -1]を覚えました。
[1, 1, -1]を覚えました。
[1, 1, -1]から、[ 1  1 -1]を思い出しました！
[1, 1, 0]から、[ 1  1 -1]を思い出しました！
[1, -1, 1]を覚えました。
[1, -1, 1]を覚えました。
[1, -1, 1]を覚えました。
[1, -1, 1]を覚えました。
[1, 1, -1]から、[-1  1 -1]を思い出しました！
[1, -1, 1]から、[ 1 -1  1]を思い出しました！
```

3年生輪読予定表

担当箇所	名前	発表予定日
1-1	保科	5/12
1-3	猪狩	5/12
2-1	村田 西澤	5/19
2-2 2-3	西ヶ谷 坂中	5/19
3-1	米倉 田之井	5/26
3-2	松田 清水	5/26

3年生輪読予定表

担当箇所	名前	発表予定日
3-3	宿野 前原 村田	6/2
4章	門木 山下 上野 數見	6/9 6/16 6/23
5章	下窪 黒川 関根 坂本	6/30 7/7

次回

- ホップフィールドネットワーク。
- 輪読(2-1、2-2、2-3)。