WANN におけるシナプス荷重の調整と活性化関数の慎重な選択

(指導教員 山口 智 教授)

山口研究室 2031133 増田 瑞樹

1. 既存手法

WANN では，ネットワーク構造を変更することで精度を向上させており，変更のうちのひとつとして活性化関数の変更を行っている．
変更するノードは隠れ層のからランダムに一つ選ばれ，10 種類ある活性化関数の中から現在採用されていない 9 種類のどれかに等しい確率で変更される．

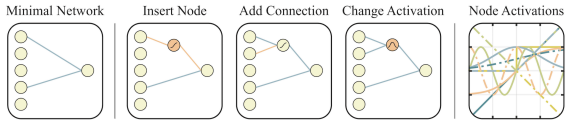

図 1: ネットワークの変更方法

WANN はその性質上ネットワークの深さが一定ではないため，ノードが隠れ層のものかどうかを判断する必要がある．
また，ノードはそれぞれ 10 種類のうちどれかの活性化関数を持っている．

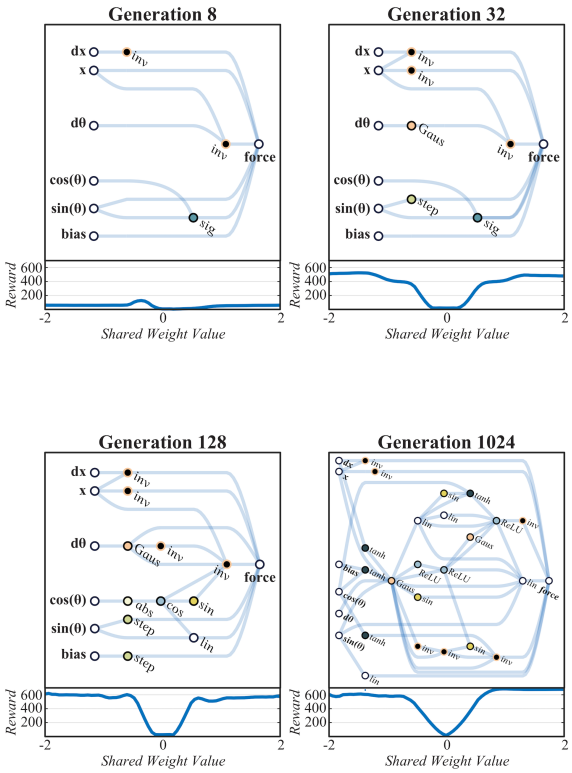

図 2: ネットワークの成長過程

既存手法の活性化関数変更アルゴリズム

ノード情報を取得

ノード情報の中からランダムな隠れ層のノード ID を取得

現在の活性化関数以外の関数へ活性化 ID を変更

return node

2. 提案手法

活性化関数の変更確率の決定方法については，ある領域内 $-n, n$ での関数同士 f_x と g_x の差の二条の合計 (1)，シナプスの入力 x に対応する出力 f_x と g_x の差の合計 (2) を実装し，これらと既存手法を比較しようと考えている．

$$\int_{-n}^{n}(f(x)-g(x))^2 \tag{1}$$

$$(f(\sum_{k=1}^{n}(n)-b)-g(\sum_{k=1}^{n}(n)-b))^2 \tag{2}$$

上で求めた値 f_n の逆数を取り，自身の活性化関数は 0 とし，行の合計が 1 になるよう正規化したものを変更先の確率 P_n として採用する

$$P_n = \frac{Q_n}{\sum_i (Q_i)} \tag{3}$$

$$Q_i = \begin{cases} \dfrac{1}{f_x} & (x \neq t) \\ 0 & (x = t) \end{cases}$$

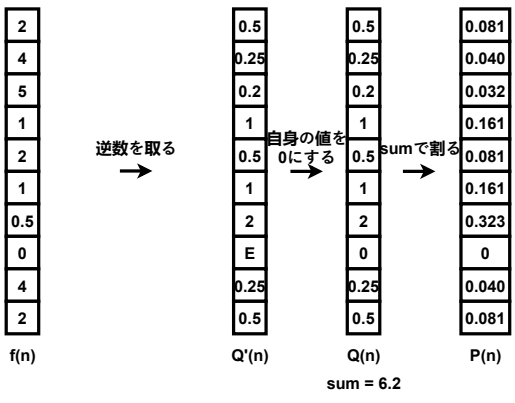

図 3: 変更確率の求め方

3. 進捗

(1) は終了している，活性化関数のレパートリーは最初から決まっている上，プログラムを走らせるたびに計算するコストが惜しいので最初から計算結果をテーブルに保存することで実装した．

二条誤差の活性化関数変更アルゴリズム

```
ランダムな隠れ層のノード ID を取得
ノード ID に対応する活性化関数を元に二条誤差をテーブルから参照する
活性化関数を重み付きランダム選択
return node
```

(2) については実装途中の段階で，あるノードの出力を得るためには一つ手前の層から接続されちえるノードの出力を得る必要があり，最終的に入力層の出力が必要になる．入力ノードの出力を得るプログラムは未完成．

入力を考慮した活性化関数変更アルゴリズム

```
focus = ランダムな隠れ層のノード ID
for i:
    table[i] = output(conn, node, focus, act)

def output(conn, node, focus, act)
    out, weight, able
    list = conn から目的地が focus だけを抽出
    for i:
        list[i] の元が入力ノードなら:
            return 入力ノードの出力
        list[i] の元が隠れ層ノードなら:
            new_focus = list[i] の元ノード ID
            new_act = list[i] の元ノードの活性化関数
            val = output(conn, node, new_focus, new_act)
            out.append(val)
            weight.append(list[i] の重み)
            able.append(list[i] の有効値)

    sum = 0
    for i:
        sum += out[i] * weight[i]

    return node
```

4. 今後の課題

Biped タスクの場合，1 回のテストで入力されるデータが 1 回ではないので，いつのデータをどのようにして入力として扱うかを考えたい．

また，活性化関数変更アルゴリズムが完成したらシナプス荷重の提案手法実装も行う．

また，二条誤差をもとに変更確率を求めたが，その方法を変更したい．自身と自身の活性化関数の距離は当然 0 であるし，同じ活性化関数に乗り換えることは乗り換えていないことと同じなので無視できるが，自身と違う活性化関数との距離が 0 や 0 にとても近い数値になると，変更確率を独占してしまうため，逆数を取る前にすべての距離に同じ数を加えることでこれを防ぐ．

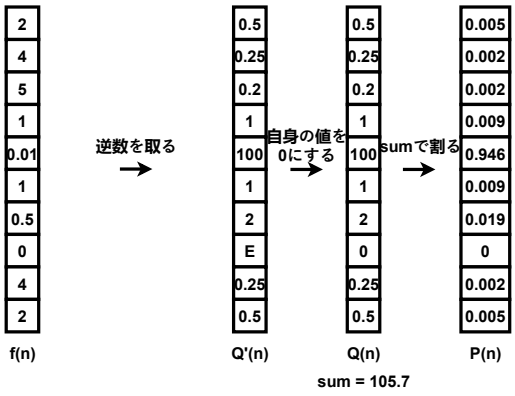

図 4: 現在の手法

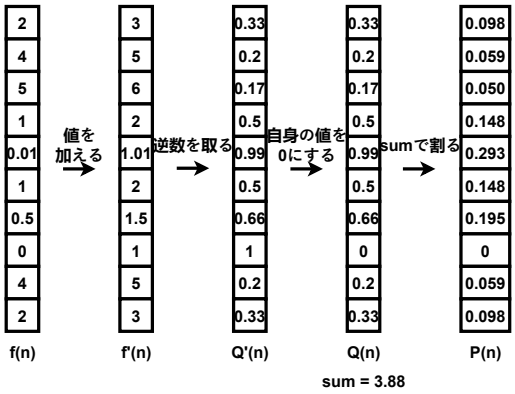

図 5: 改良後の手法

5. 付録

ネットワークのデータ構造

```
.conn - (np_array) - connection genes
    [5 X nUniqueGenes]
    [0,:] == Innovation Number (unique Id)
    [1,:] == Source Node Id
    [2,:] == Destination Node Id
    [3,:] == Weight Value
    [4,:] == Enabled?
.node - (np_array) - node genes
    [3 X nUniqueGenes]
    [0,:] == Node Id
    [1,:] == Type
    [2,:] == Activation function (as int)
```