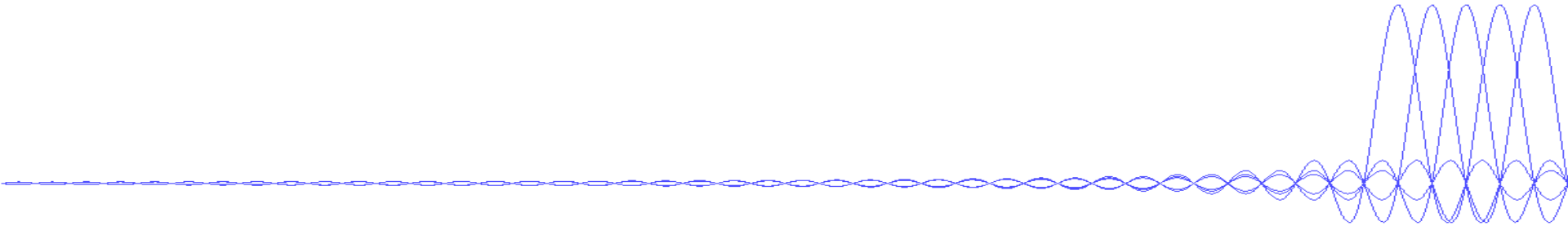
Kyutech
DSP Lab.

The 25th

LSI 2022
Design Contest in Okinawa

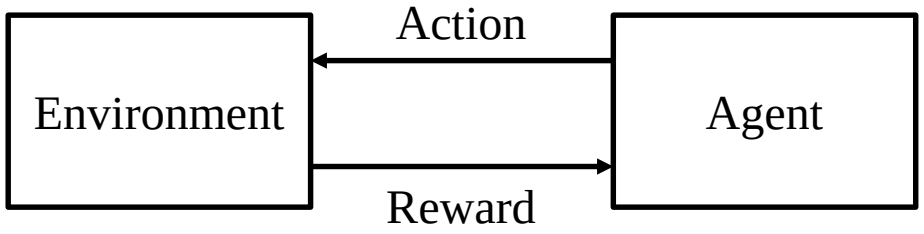
About Deep Q-Network



About DQN(Deep-Q-Network)(1/4)

- DQN(Deep-Q-Network)とは
  - 深層強化学習の一種
    - 強化学習(Q学習)に深層学習を組み合わせた手法
  - 行動指針を決める値(Q学習におけるQ値)を出力する関数をニューラルネットワーク(Neural Network :以降NN)を用いて作成し，そのNNのパラメータを最適化していくことで実際にQ値の近似値を出力させる

- 要素
  - Agent:行動主体
  - State:状態(盤面)
  - Action:行動(コマを置く)
  - Reward:報酬



About DQN(Deep-Q-Network)(2/4)

- 学習の方法
  - ある状態sにおいて行動aをとった時の価値を最適化していく
  - この価値をQ値または状態行動価値と呼び，$Q(s,a)$で表される．
  - Q学習の行動価値関数の更新式は，
    $Q(s_t,a_t) \leftarrow (1-\alpha)Q(s_t,a_t) + \alpha\{r + \gamma \max Q(s_{t+1},a_{t+1})\}$
    であり，上の式を変形すると
    $Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\{r + \gamma \max Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)\}$

    α:学習率
    γ:割引率
    r:報酬

  - $r + \gamma \max Q(s_{t+1},a_{t+1}) - Q(s_t,a_t) \rightarrow 0$　が学習の終了条件

  - つまり，出力される$Q(s_t,a_t)$の値が$r + \gamma \max Q(s_{t+1},a_{t+1})$に近づくように学習させる必要がある

About DQN(Deep-Q-Network)(3/4)

- 学習の方法(続き)
  - 前述した行動価値関数を出力するNNを実現するための目的関数Cを設定する
  - 今回は二乗誤差E_tを使用し，出力値と教師データの差の二乗を誤差として計算する

$$\mathrm{E_t} = [r + \gamma \max Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]^2$$
$$C_t = \Sigma_{t=1}[r + \gamma \max Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]^2$$

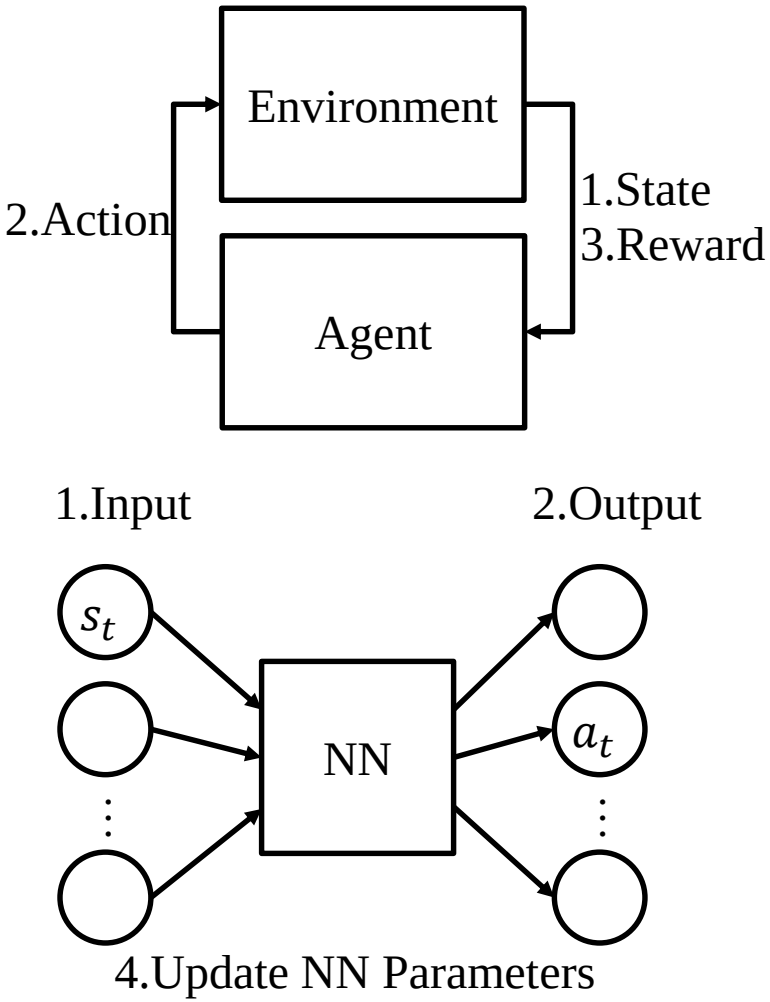
    - 状態s_{t+1}はs_tでa_tを実施して求め，$\max Q(s_{t+1}, a_{t+1})$はNNにs_{t+1}を入力して求める

  - 目的関数Cに対してバックプロパゲーションを用いて，各層での重みとバイアスを計算しNNを更新する

About DQN(Deep-Q-Network)(4/4)

学習の順序

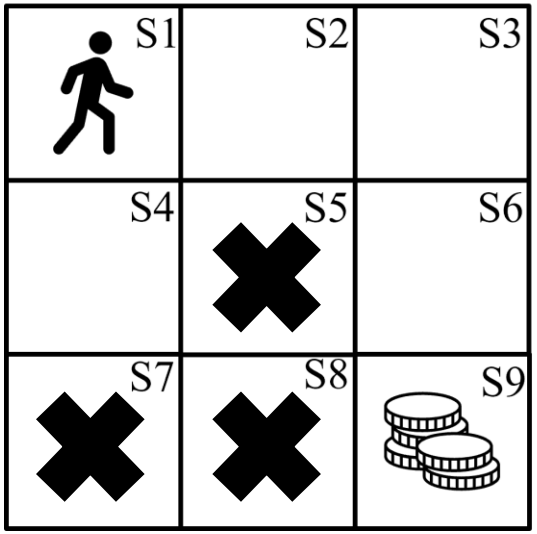
1. タイムステップtごとにAgentは現在の状態s_tを取得し，NNに入力として与える
2. 1で得られた入力におけるNNの出力を参考にAgentは行動a_tを決定する
3. Agentは行動した結果の良し悪しを報酬として受け取る
4. 報酬をもとに誤差関数を作成しNNの重み，バイアスを更新する
5. 上記を繰り返し行う



Shortest route search(1/15)

- 最短経路探索でDQNを考える
- 要素
  - Agent: Person
  - Reward: Money
  - Action: → or ↑ or ← or ↓
  - State: S1~S9(Where the agent is)
  - Start: S1, Goal: S9

Environment



Shortest route search (2/15)

- 学習の順序

1. Agentの場所(State)を取得
2. NNにStateを入力しAgentの動く方向を選択(Q値最大)
3. 報酬を取得して次の状態に遷移
4. 誤差を計算し，NNを更新して1に戻る

- 2においてQ値最大の行動だけを取るとループや局所解に陥る
  →ε-greedy法: εの確率でランダムな行動を取る
  - 1-ε:Q値が最大の行動
  - ε: ランダムな行動

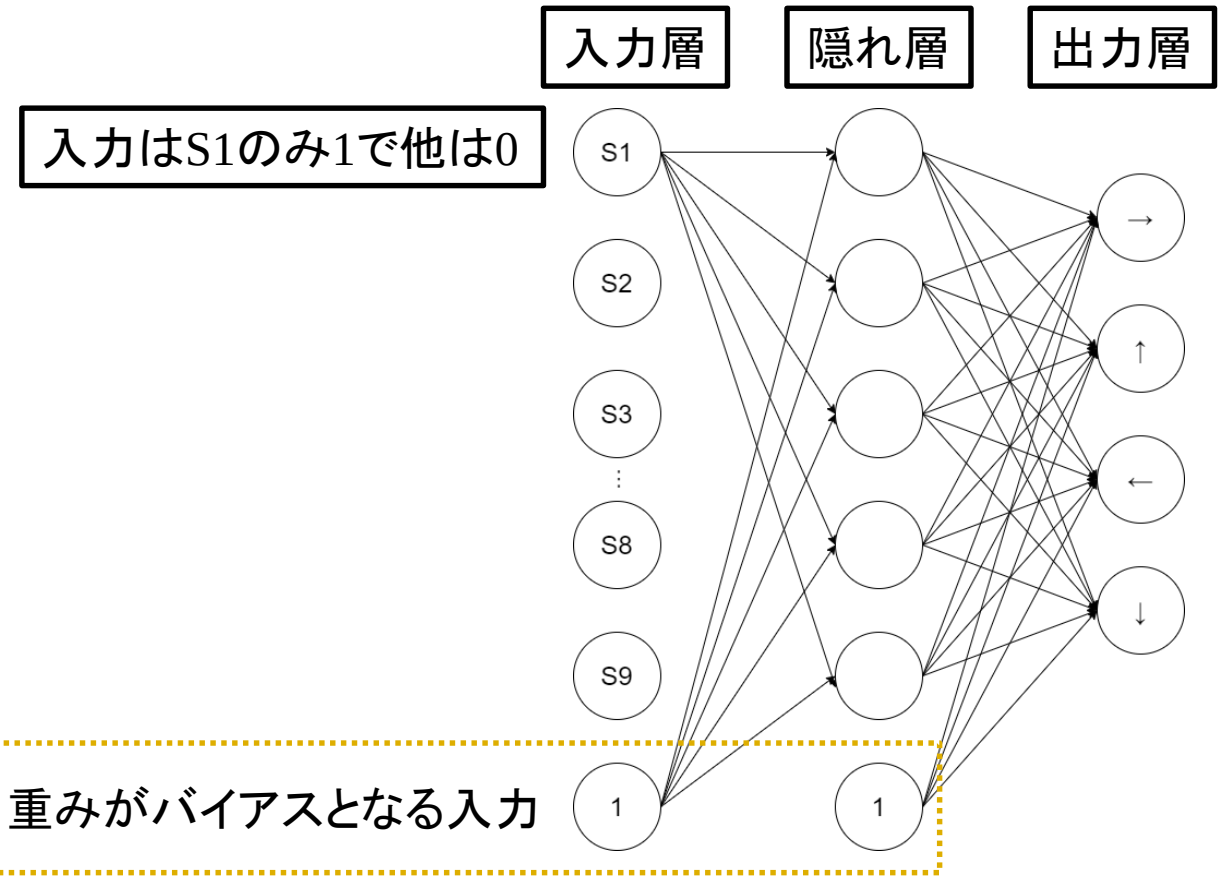
学習が進むにつれてランダムな行動を取る確率を減らす

$\varepsilon = MAXEPISODE - Episode$

Episode: 現在のエピソード数
MAXEPISODE: 最大のエピソード数

Shortest route search (3/15)

例: 初期状態で状態S1をNNに入力した場合の出力を計算する



Shortest route search (4/15)

- 各層の重み，バイアスの初期値は以下のように設定する
(MATLABのプログラムにて乱数シード値1で参照可能)

S1から隠れ層への入力重み	
w_{11}^{2}	0.4170
w_{12}^{2}	0.5388
w_{13}^{2}	0.1404
w_{14}^{2}	0.0391
w_{15}^{2}	0.6865

w_{ij}^{2}: 入力層i番目のノードから隠れ層j番目のノードへの重み
w_{ij}^{3}: 隠れ層i番目のノードから出力層j番目のノードへの重み

バイアスの初期値は全て -1

隠れ層の各ノードから出力層の各ノードへの入力重み									
w_{11}^{3}	0.4479	w_{21}^{3}	0.9086	w_{31}^{3}	0.2936	w_{41}^{3}	0.2878	w_{51}^{3}	0.1300
w_{12}^{3}	0.0194	w_{22}^{3}	0.6788	w_{32}^{3}	0.2116	w_{42}^{3}	0.2655	w_{52}^{3}	0.4916
w_{13}^{3}	0.0534	w_{23}^{3}	0.5741	w_{33}^{3}	0.1467	w_{43}^{3}	0.5893	w_{53}^{3}	0.6998
w_{14}^{3}	0.1023	w_{24}^{3}	0.4141	w_{34}^{3}	0.6944	w_{44}^{3}	0.4142	w_{54}^{3}	0.0500

Shortest route search (5/15)

- 入力層→隠れ層での重み付き和を求める
  - 隠れ層のそれぞれのノードについて重み付き和を求める
    - S1以外の入力は0なのでS1についてのみ考える

z_i^2: 隠れ層i番目のノードの重み付き和
w_{ij}^2: 入力層i番目のノードから隠れ層j番目のノードへの重み
b_i^2: 隠れ層i番目のノードでのバイアス

$$z_1^2 = w_{11}^2 \times 1 + b_1^2 = 0.4170 - 1 = -0.5830$$
$$z_2^2 = w_{12}^2 \times 1 + b_2^2 = 0.5388 - 1 = -0.4612$$
$$z_3^2 = w_{13}^2 \times 1 + b_3^2 = 0.1404 - 1 = -0.8596$$
$$z_4^2 = w_{14}^2 \times 1 + b_4^2 = 0.0391 - 1 = -0.9609$$
$$z_5^2 = w_{15}^2 \times 1 + b_5^2 = 0.6865 - 1 = -0.3135$$

Shortest route search (6/15)

- 隠れ層での出力を求める
  - 今回，隠れ層の活性化関数にはSigmoid関数を用いる

a_i^2: 隠れ層i番目のノードの出力

$$a_1^2 = \frac{1}{1+\exp(-z_1^2)} = \frac{1}{1+\exp(0.5830)} = 0.3582$$
$$a_2^2 = \frac{1}{1+\exp(-z_2^2)} = \frac{1}{1+\exp(0.4612)} = 0.3867$$
$$a_3^2 = \frac{1}{1+\exp(-z_3^2)} = \frac{1}{1+\exp(0.8596)} = 0.2974$$
$$a_4^2 = \frac{1}{1+\exp(-z_4^2)} = \frac{1}{1+\exp(0.9609)} = 0.2767$$
$$a_5^2 = \frac{1}{1+\exp(-z_5^2)} = \frac{1}{1+\exp(0.3135)} = 0.4223$$

Shortest route search (7/15)

- 隠れ層→出力層での重み付き和を求める
  - 出力層のそれぞれのノードについて重み付き和を求める

z_i^3: 出力層i番目のノードの重み付き和
w_{ij}^3: 隠れ層i番目のノードから出力層j番目のノードへの重み
b_i^3: 出力層i番目のノードでのバイアス

$$z_1^3 = w_{11}^3 a_1^2 + w_{21}^3 a_2^2 + w_{31}^3 a_3^2 + w_{41}^3 a_4^2 + w_{51}^3 a_5^2 + b_1^3$$
$$= 0.1605 + 0.3514 + 0.0873 + 0.0796 + 0.0549 - 1 = -0.2663$$
$$z_2^3 = w_{12}^3 a_1^2 + w_{22}^3 a_2^2 + w_{32}^3 a_3^2 + w_{42}^3 a_4^2 + w_{52}^3 a_5^2 + b_2^3$$
$$= 0.0069 + 0.2625 + 0.0629 + 0.0735 + 0.2076 - 1 = -0.3866$$
$$z_3^3 = w_{13}^3 a_1^2 + w_{23}^3 a_2^2 + w_{33}^3 a_3^2 + w_{43}^3 a_4^2 + w_{53}^3 a_5^2 + b_3^3$$
$$= 0.0191 + 0.2220 + 0.0436 + 0.1631 + 0.2955 - 1 = -0.2567$$
$$z_4^3 = w_{14}^3 a_1^2 + w_{24}^3 a_2^2 + w_{34}^3 a_3^2 + w_{44}^3 a_4^2 + w_{54}^3 a_5^2 + b_4^3$$
$$= 0.0366 + 0.1601 + 0.2065 + 0.1146 + 0.0211 - 1 = -0.4610$$

Shortest route search (8/15)

- 出力層での出力(Q値)を求める
  - 今回，出力層の活性化関数にはSoftMax関数を用いる

a_i^3: 出力層i番目のノードの出力

$$a_1^3 = \frac{\exp(z_1^3)}{\Sigma_{i=1}^{4} \exp(z_i^3)} = 0.2688$$

$$a_2^3 = \frac{\exp(z_2^3)}{\Sigma_{i=1}^{4} \exp(z_i^3)} = 0.2384$$

$$a_3^3 = \frac{\exp(z_3^3)}{\Sigma_{i=1}^{4} \exp(z_i^3)} = 0.2715$$

$$a_4^3 = \frac{\exp(z_4^3)}{\Sigma_{i=1}^{4} \exp(z_i^3)} = 0.2213$$

Shortest route search (9/15)

- 出力結果からAgentの行動を決定
  - 今回はQ値最大の行動を取ったとする

$$\max Q(s_1, a) = Q(s_1, a_3) = 0.2715$$

  - 左に移動→移動できずその場に留まるため罰則を与える

$$reward = 0, penalty = 5$$

- NNのパラメータを更新する
  - NNに$s_{t+1} = s_1$を入力，出力は先程の出力に等しい

$$\max Q(s_{t+1}, a) = Q(s_1, a_3) = 0.2715$$

  - 誤差を定義する

$$E_3 = [reward - penalty + \gamma \max Q(s_1, a_3) - Q(s_1, a_3)]^2 = 25.2722$$

  - 目的関数の設定

$$C_3 = [reward - penalty + \gamma \max Q(s_1, a_3) - Q(s_1, a_t)]^2 = E_3 = 25.2722$$

Shortest route search (10/15)

- 設定した目的関数に対しバックプロパゲーションを行う
  - 出力層の入力重みについて計算する
  - 目的関数を重みで偏微分

$$\frac{\partial C_3}{\partial w_{i3}^3} = \frac{\partial C_3}{\partial z_3^3}\frac{\partial z_3^3}{\partial w_{i3}^3}$$

$\delta_3^3 = \frac{\partial C_3}{\partial z_3^3}$とおくと

$$\frac{\partial C_3}{\partial w_{i3}^3} = \delta_3^3 a_i^2$$

$$\leftarrow \frac{\partial z_3^3}{\partial w_{i3}^3} = a_i^2$$

さらに,

$$\delta_3^3 = \frac{\partial C_3}{\partial z_3^3} = \frac{\partial C_3}{\partial a_3^3}\frac{\partial a_3^3}{\partial z_3^3}$$

Shortest route search (11/15)

■ 続き

$$\frac{\partial C_3}{\partial a_3^3} = reward - penalty + \gamma \max Q(s_1, a_3) - Q(s_1, a_3) = -5.0271$$

$$\frac{\partial a_3^3}{\partial z_3^3} = (1 - a_3^3)a_3^3 = 0.1978$$

$$\delta_3^3 = (-5.0271) \times 0.1978 = -0.9942$$

以上より，更新後の出力層の3番目のノードの入力重みは

$$w_{13}^3 = w_{13}^3 + \alpha[a_1^2 \times (-0.9942)] = 0.0534 + 0.001\big(0.3582 \times (-0.9942)\big) = 0.0530$$
$$w_{23}^3 = w_{23}^3 + \alpha[a_2^2 \times (-0.9942)] = 0.5741 + 0.001\big(0.3867 \times (-0.9942)\big) = 0.5737$$
$$w_{33}^3 = w_{33}^3 + \alpha[a_3^2 \times (-0.9942)] = 0.1467 + 0.001\big(0.2974 \times (-0.9942)\big) = 0.1464$$
$$w_{43}^3 = w_{43}^3 + \alpha[a_4^2 \times (-0.9942)] = 0.5893 + 0.001\big(0.2767 \times (-0.9942)\big) = 0.5890$$
$$w_{53}^3 = w_{53}^3 + \alpha\big[a_5^2 \times (-0.9942)\big] = 0.6998 + 0.001\big(0.4223 \times (-0.9942)\big) = 0.6994$$

となる

Shortest route search (12/15)

- 続き
  - 出力層の入力バイアスについて計算する
  - 目的関数をバイアスで偏微分

$$\frac{\partial C_3}{\partial b_3^3} = \frac{\partial C_3}{\partial z_3^3}\frac{\partial z_3^3}{\partial b_3^3}$$

$\frac{\partial C_3}{\partial z_3^3} = \delta_3^3$は既知

$$\frac{\partial z_3^3}{\partial b_3^3} = 1$$

したがって更新後の出力層の3番目のノードのバイアスは

$$b_3^3 = b_3^3 + \alpha\delta_3^3 = -1 + 0.001 \times (-0.9942) = -1.0010$$

となり，重みもバイアスも数値計算で求められる

Shortest route search (13/15)

- 続き
  - 隠れ層の入力重みについて計算する
  - 目的関数を重みで偏微分

$$\frac{\partial C_3}{\partial w_{1j}^2} = \frac{\partial C_3}{\partial z_j^2}\frac{\partial z_j^2}{\partial w_{1j}^2}$$

$$\frac{\partial z_j^2}{\partial w_{1j}^2} = 1$$

$\delta_j^2 = \frac{\partial C_3}{\partial z_j^2} = \frac{\partial C_3}{\partial a_j^2}\frac{\partial a_j^2}{\partial z_j^2}$とおく

$$\frac{\partial C_3}{\partial a_j^2} = \frac{\partial C_3}{\partial z_3^3}\frac{\partial z_3^3}{\partial a_j^2} = \delta_3^3 w_{j3}^3$$

$$\frac{\partial a_j^2}{\partial z_j^2} = \left(1 - a_j^2\right)a_j^2$$

Shortest route search (14/15)

■ 続き

$$\delta_3^3 w_{j3}^3 = (-0.0531, -0.5708, -0.1458, -0.5859, -0.6957)$$
$$\left(1 - a_j^2\right) a_j^2 = (0.2299, 0.2372, 0.2090, 0.2001, 0.2440)$$
$$\delta_j^2 = (-0.0122, -0.1354, -0.0305, -0.1172, -0.1698)$$

したがってS1から隠れ層への更新後の入力重みは

$$w_{11}^2 = w_{11}^2 + \alpha\delta_1^2 = 0.4170 + 0.001 \times (-0.0122) = 0.4170$$
$$w_{12}^2 = w_{12}^2 + \alpha\delta_2^2 = 0.5388 + 0.001 \times (-0.1354) = 0.5387$$
$$w_{13}^2 = w_{13}^2 + \alpha\delta_3^2 = 0.1404 + 0.001 \times (-0.0305) = 0.1404$$
$$w_{14}^2 = w_{14}^2 + \alpha\delta_4^2 = 0.0391 + 0.001 \times (-0.1172) = 0.0390$$
$$w_{15}^2 = w_{15}^2 + \alpha\delta_5^2 = 0.6865 + 0.001 \times (-0.1698) = 0.6863$$

となる

Shortest route search (15/15)

- 続き
  - 隠れ層の入力バイアスについて計算する
  - 目的関数をバイアスで偏微分

$$\frac{\partial C_3}{\partial b_j^2} = \frac{\partial C_3}{\partial z_j^2}\frac{\partial z_j^2}{\partial b_j^2}$$

$\frac{\partial C_3}{\partial z_j^2} = \delta_j^2$ は既知

$$\frac{\partial z_j^2}{\partial b_j^2} = 1$$

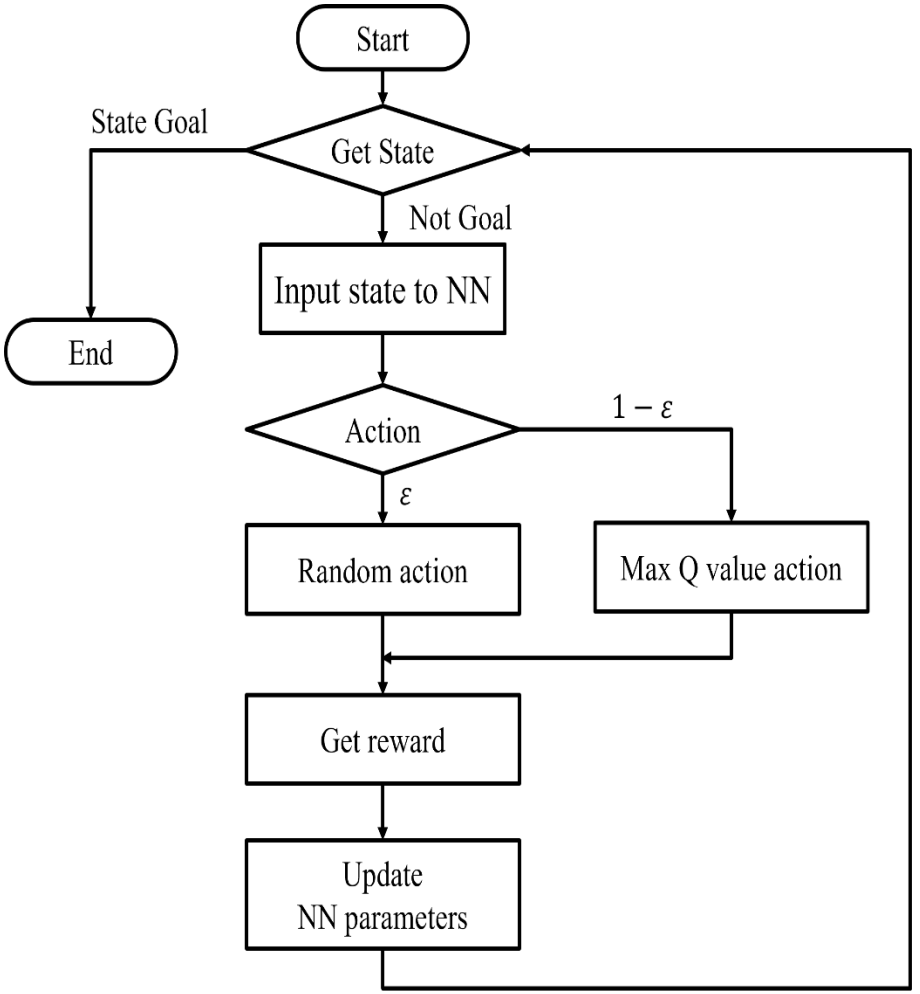
したがって更新後の隠れ層のバイアスは

$$b_1^2 = b_1^2 + \alpha\delta_1^2 = -1 + 0.001 \times (-0.0122) = -1.0000$$
$$b_2^2 = b_2^2 + \alpha\delta_2^2 = -1 + 0.001 \times (-0.1354) = -1.0001$$
$$b_3^2 = b_3^2 + \alpha\delta_3^2 = -1 + 0.001 \times (-0.0305) = -1.0000$$
$$b_4^2 = b_4^2 + \alpha\delta_4^2 = -1 + 0.001 \times (-0.1172) = -1.0001$$
$$b_5^2 = b_5^2 + \alpha\delta_5^2 = -1 + 0.001 \times (-0.1698) = -1.0002$$

となり，重みもバイアスも数値計算で求められる

Program flow

- プログラムのフローチャート
  - NNの重みの初期値はランダムに決定
  - NNのバイアスの初期値は-1
  - Goalに到着した場合はその世代の学習を終了する
  - Actionはε-greedy法より，εの確率でランダムな行動を取る



Explain source code (MATLAB)

- 今回使用しているプログラム
  - sw_move_state.m
  - sw_update_Q.m
  - print_weight.m
  - print_route.m
  - sw_Q_Learning.m

sw_move_state.m

- Agentが行動を決定する関数
  - 引数: 重み・バイアス，隠れ層・出力層の活性化関数，epsilon，現在の状態，迷路盤面の行列
  - 返り値: 次の状態，行動，Q値

1. one hot入力(1つの要素のみ1で他が0である入力)を作成
2. 隠れ層での重み付き和を計算
3. 活性化関数を通して隠れ層の出力を作成
4. 出力層の重み付き和を計算
5. 活性化関数を通して出力層の出力を作成
6. 出力をQ値として保存
7. ε-greedy法に基づいて行動を決定

Kyutech DSP Lab.

sw_update_Q.m

- 行動結果をもとにNNのパラメータを更新する関数
  - 引数: 現在の状態，次の状態，重み・バイアス，隠れ層・出力層の活性化関数，行動，報酬・罰則，学習率，割引率，迷路盤面の行列
  - 返り値: 重み・バイアス

1. 次の状態のQ値の計算(sw_move_state.mの2~6と同様)
2. 現在のQ値を計算(sw_move_state.mの2~6と同様)
3. 二乗誤差を定義し目的関数を作成
4. バックプロパゲーションを用いてNNのパラメータを更新

Kyutech DSP Lab.

print_weight.m

- NNの重み，バイアス，行動決定に用いる値を表示する関数
  - 引数: 重み・バイアス
  - 返り値: 無し

1. 隠れ層における重み，バイアスをそれぞれ表示
2. 出力層における重み，バイアスをそれぞれ表示
3. 行動決定に用いる値(Q値)を算出して表示

Kyutech DSP Lab.

print_route.m

- NNの出力を参考に移動経路を出力する関数
  - 引数:重み・バイアス，隠れ層・出力層の活性化関数
  - 返り値: 無し

1. S1(Start)からQ値を算出する
2. Q値最大の行動を取り状態遷移を表示する
   1. S9(Goal)の場合はGoalと表示
   2. 周りの壁に当たった場合はエラー表示
   3. 最長ルート(maze_i*maze_j)でもゴールできない場合はエラー表示
3. 次の状態に遷移した場合は再度Q値を算出し，2の処理を行う

Kyutech DSP Lab.

sw_Q_Learning.m

- 学習を行うためのメインプログラム
  - 学習条件を設定する
    - 最大のエピソード数，最大の行動回数，隠れ層のノード数
    - ハイパーパラメータ，epsilon，NNのパラメータの初期値
    - 迷路盤面(reward_table)の設定
    - 活性化関数の設定
  - 各関数を呼び出して学習を行う

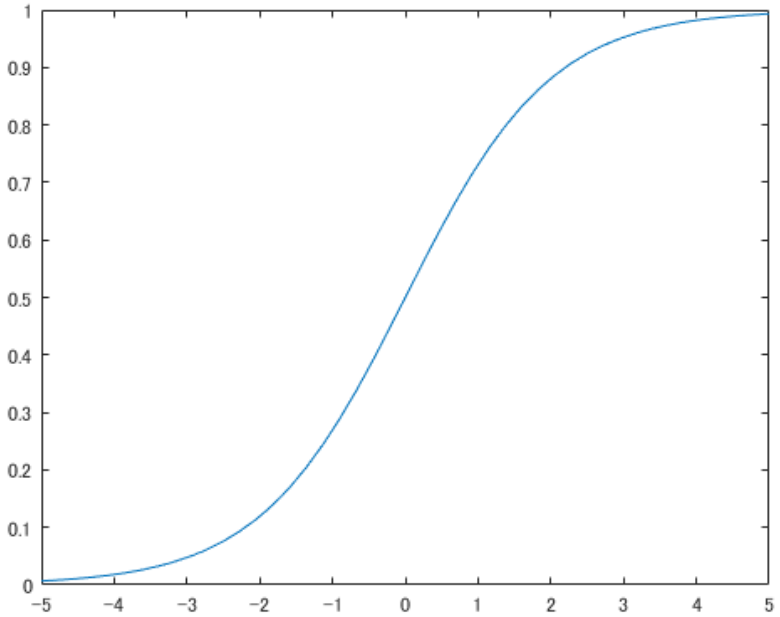
Kyutech DSP Lab.

Appendix

- NNに用いられる活性化関数
  - Sigmoid関数
  - tanh関数
  - SoftMax関数

NNに用いられる活性化関数(1/3)

- Sigmoid関数



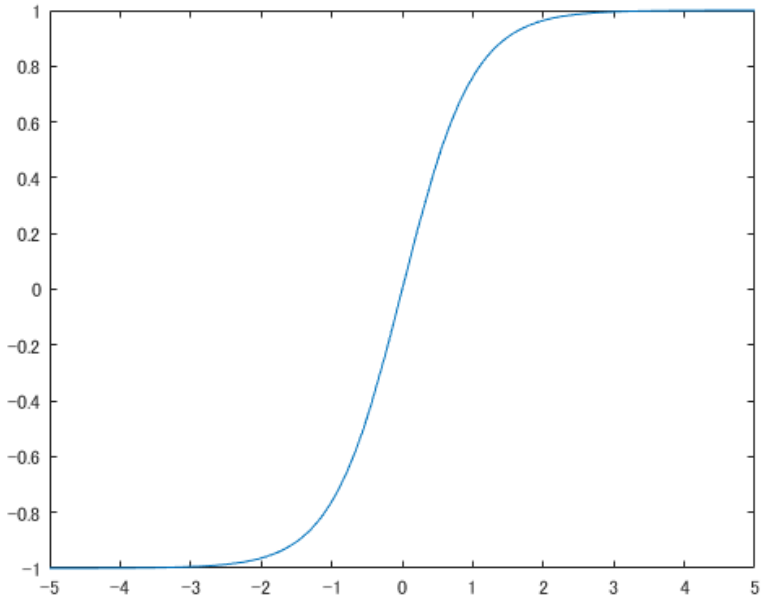
・微分した値を出力から求める事ができる
・勾配消失などの問題が起きやすい

$$y = \frac{1}{1 + e^{-x}}$$

$$\frac{\partial y}{\partial x} = \left(1 - \frac{1}{1 + e^{-x}}\right)\frac{1}{1 + e^{-x}} = (1 - y)y$$

NNに用いられる活性化関数(2/3)

- tanh関数



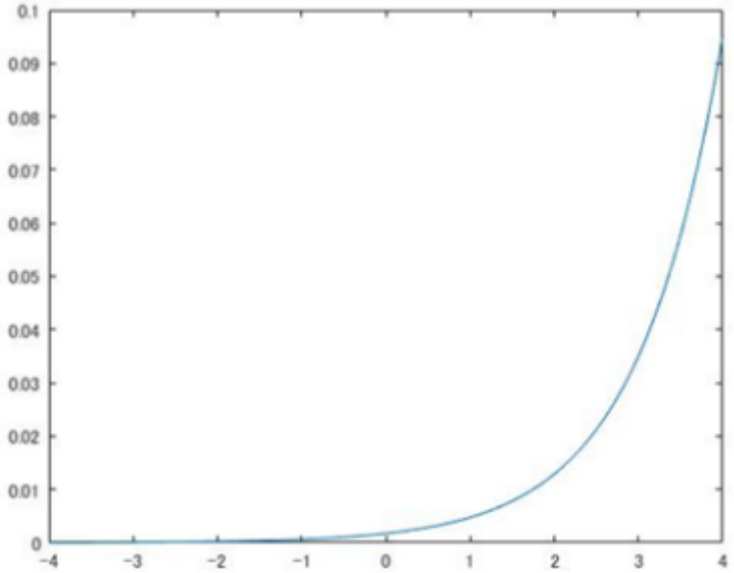
・出力として負の値を許容できる
・勾配消失が起きにくい

$$y = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

$$\frac{\partial y}{\partial x} = \frac{4}{(e^{x} + e^{-x})^{2}} = \frac{1}{\cosh^{2} x}$$

NNに用いられる活性化関数(3/3)

- SoftMax関数



・主に出力層で用いられる
・$\Sigma_{i=1}^{n} y_i = 1$となり出力を確率としてみなせる

$$y_i = \frac{e^{x_i}}{\Sigma_{i=1}^{n} e^{x_i}}$$

$$\frac{\partial y_i}{\partial x_j} = \begin{cases} y_i(1 - y_i) & (i = j) \\ -y_i y_j & (i \neq j) \end{cases}$$

Kyutech DSP Lab.

参考文献

書籍

斎藤康毅.『ゼロから作るDeep Learning—Pythonで学ぶディープラーニングの理論と実装』. オライリー・ジャパン. 2016年

小川雄太郎.『つくりながら学ぶ！深層強化学習—PyTorchによる実践プログラミング』. マイナビ出版. 2018年