エージェント工学 最終レポート課題

澤 祐里（21–1–037–0801）

2021 年 7 月 15 日

目次

1　目的

リサイクルロボットについての強化学習サンプルプログラムを用いて、パラメータを変更することなどで強化学習について考察する。

1.1　課題 1

リサイクルロボットの MDP を 3 種類作成する。

1.2　課題 2

ε Greedy 手法と Softmax 手法の概要を説明する。

1.3　課題 3

softmax 手法で、学習率と割引率を変化させて、更新回数と方策の比較を行う。

2 本論

2.1 課題 1

2.1.1 手順

recycleRobotMDP.csv の表 1 を参考にして遷移確率や報酬を変更した case2, case3 を作成する。作成した報酬の与え方がエージェントの目標になっていることを示す。

表 1 元の MDP(case1)

状態	行動	次状態	遷移確率	報酬
START	N	high	1	0
high	search	high	0.8	12
high	search	low	0.2	4
low	search	GOAL	0.1	-3
low	search	low	0.9	2
high	wait	high	1	1
high	wait	low	0	1
low	wait	GOAL	0	1
low	wait	low	1	1
low	recharge	GOAL	1	0
low	recharge	low	0	0
GOAL	N	high	1	0

case2,case3 をそれぞれ以下のように作成した。

case2 表 2 のように、遷移確率は変更せず報酬だけを変更した。報酬の決め方は、以下の点を意識した。

- 行動「recharge」以外で状態「GOAL」になるということは、他の存在に助けてもらう必要がある。そのため、リサイクルロボット単体で完結するように、行動「recharge」以外で次状態が「GOAL」になる全ての行動の報酬を「-7」にした。
- 上記の理由で、行動「recharge」の報酬を「7」にした。
- リサイクルロボットに積極的に「search」してもらいたいため、行動「wait」の報酬を「-1」にした。

case3 表 3 のように、遷移確率を変更して、「0.1」などの小さな確率でリサイクルロボットのバッテリーが減るようにした。報酬の決め方は、以下の点を意識した。

- リサイクルロボットがバッテリーを無駄なく使えるように、バッテリーが減る場合の報酬を行動ごとにそれぞれ減らした。
- 行動「recharge」以外で状態「GOAL」になるということは、他の存在に助けてもらう必要がある。そのため、リサイクルロボット単体で完結するように、行動「recharge」以外で次状態が「GOAL」になる全ての行動の報酬を「-7」にした。
- 上記の理由で、行動「recharge」の報酬を「7」にした。
- リサイクルロボットに積極的に「search」してもらいたいため、行動「wait」の報酬を「-1」にした。

表 2　作成した MDP(case2)

状態	行動	次状態	遷移確率	報酬
START	N	high	1	0
high	search	high	0.8	10
high	search	low	0.2	4
low	search	GOAL	0.1	-7
low	search	low	0.9	2
high	wait	high	1	-1
high	wait	low	0	0
low	wait	GOAL	0	0
low	wait	low	1	-1
low	recharge	GOAL	1	7
low	recharge	low	0	0
GOAL	N	high	1	0

表 3　作成した MDP(case3)

状態	行動	次状態	遷移確率	報酬
START	N	high	1	0
high	search	high	0.8	10
high	search	low	0.2	4
low	search	GOAL	0.1	-7
low	search	low	0.9	2
high	wait	high	0.9	-1
high	wait	low	0.1	-4
low	wait	GOAL	0.1	-7
low	wait	low	0.9	-1
low	recharge	GOAL	0.8	7
low	recharge	low	0.2	-1
GOAL	N	high	1	0

2.1.2　結果

以下に、引数「softmax 0.1 0.9 1000 recycleRobotMDP.csv S-RRlog.csv S-RRPolicy.csv S-RRQValue.csv」での、case2 と case3 の結果を示す。

case2 表 4 のような結果となった。

- 状態「high」の時、行動「search」の Q 値が行動「wait」よりも 2 倍以上高い。これは、行動「wait」の報酬を「-1」にしたことが影響して、エージェントに積極的に行動「search」を行なわせていることを示している。
- 状態「low」の時、行動「recharge」の q 値が一番高くなっている。これは、行動「recharge」以外で次状態が「GOAL」になる全ての行動の報酬を「-7」にしたことに加え、行動「recharge」の報酬を「7」にしたことが影響して、エージェントに積極的に行動「recharge」を行なわせていることを示している。

case3 表 5 のような結果となった。

- 状態「high」の時、行動「search」の Q 値が行動「wait」よりも 3 倍近く高い。これは、バッテリーが減る場合の報酬を行動ごとにそれぞれ減らしたことによる影響で、より行動「search」が選ばれやすくなったことを示している。
- 状態「low」の時、行動「recharge」の q 値が一番高くなっている。これは、行動「recharge」以外で次状態が「GOAL」になる全ての行動の報酬を「-7」にしたことに加え、行動「recharge」の報酬を「7」にしたことが影響して、エージェントに積極的に行動「recharge」を行なわせていることを示している。
- 小さい確率でバッテリーが減るように遷移確率を変更した影響で、状態「low」の時の行動「wait」の Q 値が case2 に比べて減っている。

表 4　case2 の結果

状態	行動	Q 値	Q 値更新数
GOAL	N	0	0
START	N	0	0
high	search	11.194368	54
high	wait	4.146994	54
low	recharge	4.310171	94
low	search	1.989767	94
low	wait	1.20334	94

表 5　case3 の結果

状態	行動	Q 値	Q 値更新数
GOAL	N	0	0
START	N	0	0
high	search	9.995437	58
high	wait	3.695292	58
low	recharge	3.895574	109
low	search	1.728605	109
low	wait	0.750725	109

2.1.3　考察

- case2 の結果と case3 の結果を見比べると、case3 では、状態「low」の時の行動「wait」の Q 値が半分近くまで減っている。他の行動の Q 値も少し減ってはいるものの、行動「wait」ほどは減ってはいない。このことから、相対的に行動「recharge」と行動「search」の価値が上がっていることが分かる。これは、case3 でバッテリーが時間経過でも減るという概念を追加した影響で、「wait」の Q 値が減少した。また、状態「low」になる頻度が増えたことで行動「recharge」の価値が上がった。しかし、case3 では、行動「recharge」にも失敗の概念を追加しているため、行動「recharge」の価値の上昇が抑えられたことで、行動「search」も選ばれやすくなり、その結果として、行動「recharge」と行動「search」の価値が共に上がったと考えられる。
- MDP 上で、case2 と case3 での状態「high」だけを見比べると、その違いは case3 での行動「wait」に報酬が「-4」の遷移先が増えただけであるため、case3 の行動「wait」の価値が下がることが予想できる。しかし、case2 の結果と case3 の結果を見比べると、case3 では、状態「high」の時の行動の Q 値が全体的に下がっている上に、行動「search」の方が行動「wait」よりも Q 値が多く下がってしまっている。なぜこのような結果となったのかを考えると、まず、状態「high」の行動「search」と「wait」では、「search」の方が、次状態が「low」になる確率が高い。また、case3 では case2 に比べて、状態「low」の場合に Q 値が下がってしまう確率の合計が増えている。以上のことから、状態「high」の時の行動「search」の Q 値が下がってしまったと考えられる。

2.2 課題 2

Greedy 選択 ある状態 s において、様々な行動の選択肢が存在したとする。その時の、ある行動 a の Q 値を状態 s における全ての行動の Q 値の合計で割って平均値を求める。求めた値を状態 s での行動 a の価値とする。同様に、他の行動 b の Q 値も状態 s における全ての行動の Q 値の合計で割って平均値を求める。求めた値を状態 s での行動 b の価値とする。このようにして、状態 s における全ての行動の価値を求めて、それらの行動の中で価値が 1 番高い行動を選択する。この手法では、常に Q 値で行動が決まるため、ランダム性は存在しない。[1]

eGreedy 選択 上記の Greedy 選択にランダム性を持たせたもので、確率「1- ε」で Greedy 選択を行う。残りの確率「 ε」では、ランダムに行動を選択する。 ε の値は 0 から 1 で任意に決める。ランダム性があるため、学習中に主に使われる。[1]

ボルツマン手法 ある状態 s での様々な行動の価値を Greedy 選択と同様の方法で求める。時間の経過によって、選択パターンが変わっていく。始めの方は、様々な行動を選択するが、時間が経過するほど結果が収束していき、価値が高い行動ばかりを選ぶようになる。[1]

ルーレット選択 ある状態 s での様々な行動の価値を Greedy 選択と同様の方法で求める。行動はランダムで決められるが、価値の高さの割合でそれぞれの行動の選ばれやすさの確率が決まるため、価値の高い行動ほど選ばれやすい。[1]

2.3　課題 3

2.3.1　結果

case3 で softmax 手法を用いて、学習値と割引率をそれぞれ変更したところ、以下の表 6 のようになった。

表 6　case3 における方策値と Q 値更新回数の比較

学習率「0.1」, 割引率「0.9」			
状態	行動	方策値	Q 値更新数
high	search	0.893663	59
high	wait	0.106337	59
low	recharge	0.763399	157
low	search	0.154658	157
low	wait	0.081943	157

学習率「0.2」, 割引率「0.9」			
状態	行動	方策値	Q 値更新数
high	search	0.97054	80
high	wait	0.02946	80
low	recharge	0.782385	170
low	search	0.162611	170
low	wait	0.055004	170

学習率「0.3」, 割引率「0.9」			
状態	行動	方策値	Q 値更新数
high	search	0.976938	88
high	wait	0.023062	88
low	recharge	0.787632	150
low	search	0.166601	150
low	wait	0.045767	150

学習率「0.4」, 割引率「0.9」			
状態	行動	方策値	Q 値更新数
high	search	0.978408	95
high	wait	0.021592	95
low	recharge	0.78079	164
low	search	0.17033	164
low	wait	0.04888	164

学習率「0.5」, 割引率「0.9」			
状態	行動	方策値	Q 値更新数
high	search	0.975403	121
high	wait	0.024597	121
low	recharge	0.755647	159
low	search	0.175716	159
low	wait	0.068637	159

学習率「0.1」, 割引率「0.8」			
状態	行動	方策値	Q 値更新数
high	search	0.853609	59
high	wait	0.146391	59
low	recharge	0.72808	119
low	search	0.172476	119
low	wait	0.099444	119

学習率「0.1」, 割引率「0.7」			
状態	行動	方策値	Q 値更新数
high	search	0.939968	80
high	wait	0.060032	80
low	recharge	0.741986	152
low	search	0.17956	152
low	wait	0.078454	152

学習率「0.1」, 割引率「0.6」			
状態	行動	方策値	Q 値更新数
high	search	0.942567	80
high	wait	0.057433	80
low	recharge	0.787558	129
low	search	0.132545	129
low	wait	0.079897	129

学習率「0.1」, 割引率「0.5」			
状態	行動	方策値	Q 値更新数
high	search	0.941697	82
high	wait	0.058303	82
low	recharge	0.735046	129
low	search	0.209096	129
low	wait	0.055858	129

2.3.2 考察

- 結果を見比べると、学習率が「0.1」の時は、状態「high」の行動「search」の方策値が「0.893663」となっているが、学習率「0.2〜0.5」では、方策値が「約 0.97」となっている。このことから、学習率を高くするとより価値の高い方法ばかりを選びやすくなっていると考えられる。
- 結果を見比べると、割引率が「0.9」の時は、状態「high」の行動「search」の方策値が「0.893663」となっており、割引率「0.2」では、方策値が「0.853609」と下がっている。しかし、割引率「0.7〜0.5」では、方策値が「約 0.94」と上がっていることから、割引率ごとに方策値が変動するものの、割引率がある一定のラインを超えると方策値が収束していくと考えられる。
- Q 値更新数と学習率の関係を見ると、学習率が上がるたびに Q 値更新数が増えていくのが分かる。このことから、学習率を上げると、それに伴って学習の時間も増えていくことが考えられる。
- Q 値更新数と割引率の関係を見ると、学習率とは違い、Q 値更新数の上がり方もばらばらであるため、特定の割引率によっては、学習の時間が増加したり、逆に減少したりもすることがあると考えられる。

参考文献

[1] https://drive.google.com/file/d/1pqyeJ20TixzuxkGSjkr2cp0MEKh6jj6w/view?usp=drive_web&authuser=0