強化学習による環境別のロボットの基本行為の獲得について

奈良女子大学理学部情報科学科 4 回生 新出研究室 11251502 江川鈴菜

平成 27 年 2 月 14 日

1 はじめに

我々は実世界において目的達成のための最適な行動をとることのできるロボットの研究課題に取り組んでいる。これを実現するにあたり、ロボットが自身で現実世界の動的な環境に対応した基本行為をとることが求められる。

本研究ではロボットとして教育用 LEGO Mindstorms NXT(以下 NXT と記載) を使用した。従来の研究 [1] では、NXT を高レベルの記述で制御できるライブラリ、NXT Python+が構築された。しかし、基本行為の一つである 90 度回転などのパワーの調整は随時人の手で各 NXT に対応したパラメータを入力する必要があった。パラメータは床の摩擦や NXT の重量によって異なり、その調整に手間がかかる問題があった。

よって本研究では NXT 自身が指定した回転角度に適したパワーパラメータの調整を行う学習機構を NXT Python+上に実現することを目指した。なお本研究は、奈良女子大学理学部 4 回生樽井との共同研究である。本論文ではプログラムの実装と実験結果について述べる。

2 研究背景

2.1 LEGO Mindstorms NXT

本研究では小型ロボットとして比較的安価で動かしやすい NXT を使用した。これは LEGO 社から販売されているロボットで、ブロックを組み合わせ自由にロボット製作を行う事が可能である。

NXT には 3 つの出力ポートと 4 つの入力ポートがある。出力ポートから繋がれたモーターは回転数を数値として調整することが可能である。入力ポートには光センサ、衝撃センサ、コンパスセンサが繋げることができる。本研究ではライントレースに 2 台の光センサを用い、強化学習の報酬設定においてコンパスセンサで回転角度の計測を行った。また、NXT を動かすには、PC から USB による通信か、Bluetooth による通信で制御プログラムを転送し、実行する。実験では、Bluetooth による通信で NXT を動作させた。

2.2 NXT Python+

従来の研究 [1] では、GUI による入力操作を行えるようにするため、GUI の設計が容易である Python を NXT を制御するプログラミング言語として選択した。そして、パワーを直接指定する

下位レベルの命令を基にしたライブラリである NXT Python を基に、一定期間の前進や 90 度回転など、より上位レベルでの基本行為の記述が可能になるよう、NXT Python を基にして作られたライブラリが NXT Python+である。

2.3 強化学習プログラム

NXT Python+において、前身や 90 度の回転など基本行為を行う際、NXT の重量や電池の残量、環境の変化などに合わせてモーターの回転数のパワーのパラメータを人間が GUI による入力で調節する必要があった（図 1)。この場合、動的環境下において連続して実験を行う際に、環境が変化する度にパラメータの調整を行わなければならず、時間がかかってしまうという問題があった。

その過程を簡易化し実験時間を短縮するために、今回ライブラリを拡張して、NXT が自身で環境に対応したパラメータ調整を行うことができる強化学習プログラムを構築した。

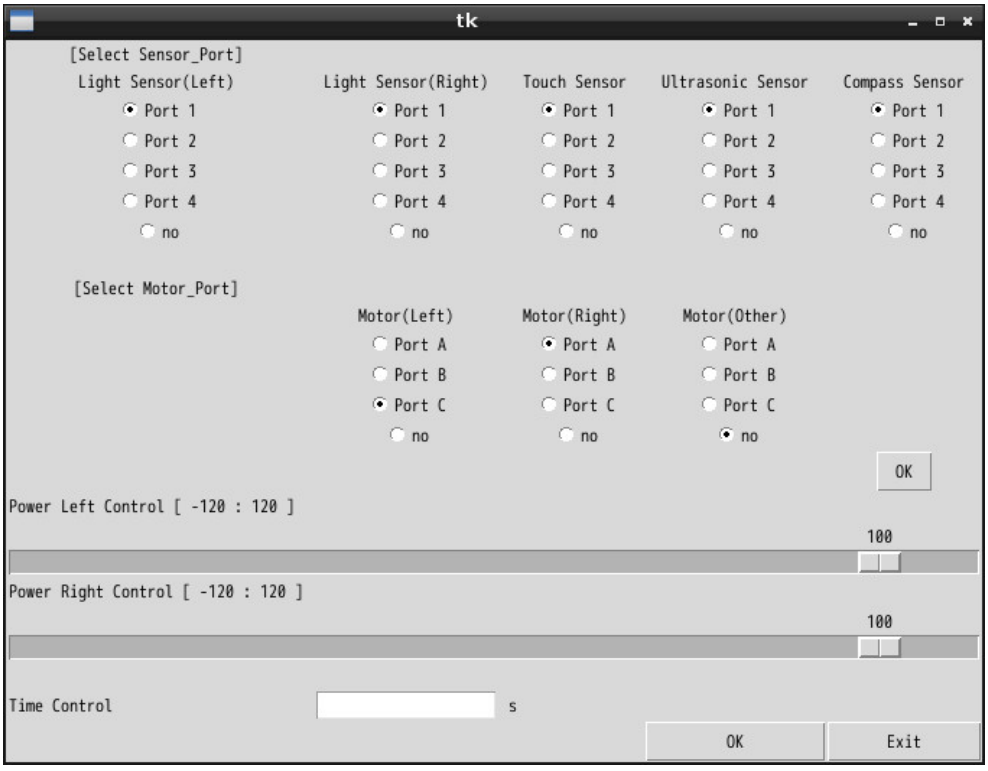


図 1: NXT Python+の GUI 画面

3 実装

3.1 実装方針

今回、NXT Python+内に Learn という学習機構プログラムのクラスを追加した。Learn クラスには「ε-greedy 法」での強化学習を行うためのメソッド群と、強化学習の選択肢の自動生成を行うメソッド群、そして実際に NXT を動かして強化学習の試行を行うメインメソッドがある。

以下に、強化学習を行うためのメソッド群を示す。

- __init__
  パラメータの選択肢のリストを生成する__init__メソッド（コンストラクタ）。リストにはパラメーター値、平均報酬の推定値、選択肢が選択された回数が入る。

- greedy_select(param)
  平均報酬の推定値が最も良い選択肢の番号を返すメソッド。param は選択肢のリスト。

- random_select(size)
  ランダムに選んだ選択肢の番号を返すメソッド。size は選択肢の個数。

- random_LR()
  学習時に回転する方向をランダムに選び、右か左の値を返すメソッド。

- trial(param, selection, LR)
  学習時、選択されたパラメータを 1 回試行し、報酬を得て平均報酬の推定値と選んだ回数を更新するメソッド。selection は選んだ選択肢の番号を表し、LR は選んだ左右の値を表す。

- comp_reward(be)
  学習の報酬の評価をコンパスセンサーの値で行い、報酬を返すメソッド。be は移動前の方位。

本研究の実験では繰り返しプログラムを動かすことで、NXT が適切なパラメータを得ることを想定した。よって、パラメータの選択肢は実験の一度目の初期値は固定してあるが、二度目以降の選択肢を自動生成することでより適切なパラメータを得ることが出来るようにした。このためのメソッドとして以下の 3 つを作成した。

- make_para(param,cut)
  前回の学習でデータファイルに保存された最適値のパラメータを基に任意の数のパラメータのリストを生成するメソッド。cut は刻み幅の値。

- cut_set()
  保存された前回の最適値の推定平均報酬によって、パラメータのリストの刻み幅を設定するメソッド。

- ep_cut(ep,cut)
  学習のランダム性を調整する ε の値を刻み幅によって変化させ、設定するメソッド。ep は ε の値、cut は刻み幅の値

3.2 強化学習

強化学習には ε -greedy 法を使用した。

報酬を得るための回転の評価にはコンパスセンサを使用した。交差点で NXT を一時停止させ、その地点から NXT が 90 度回転する事を「1」、全く回転を行わなかった場合を「0」として、回転した角度によって評価を行った。(図 2) 報酬は 90 度に近いほど高い報酬が得られるようになっている。このように、評価をコンパスセンサで行ったため、人手で判定を行うよりも円滑に実験を行うことが出来るようになった。

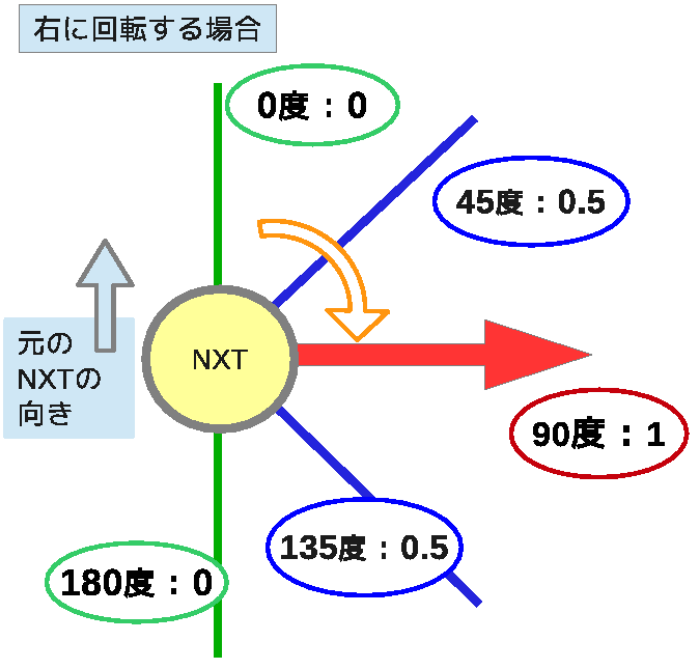


図 2: コンパスセンサでの評価

また、強化学習では一般に多数回の実験を要するが、実世界で多数回の実験を行うことは時間も手間もかかるため難しい。そのため、より効率の良い学習方法を取ることが重要となった。そこで、一定回数の試行を一度の実験とし、パラメータの選択肢リストを一度の実験ごとに更新することでより適切なパラメータを得ることができ、効率のよい学習を行うことができるようにした。実験では、一度目の初期値を一定にし、二度目以降のパラメータの選択肢リストは前回の学習結果を踏まえてリストを自動生成する。

実験の学習結果は、最終的にテキストファイルへ保存される。ここで保存されるのは最適値のパラメータとその推定平均報酬値である。次の学習を行う際に、保存されたデータを読み込み、それを基にパラメータのリストを生成する。前回の最適値を基準値としてそれぞれのパラメータの刻み幅を設定してリストは生成される。

刻み幅の設定を cut_set メソッドが行い、前回の最適値の推定平均報酬値を基に刻み幅の設定を行う。これは推定平均報酬値が低いと前回の最適値が実際の適切なパラメータとは離れている場合を考慮し、より適切なパラメータを見つけるためにパラメータ同士の値をあえて離れさせるようにしたためである。逆に推定平均報酬値が高い場合はより適切なパラメータを検証するために細かい刻み幅にするようにした。刻み幅の設定の詳細は表 1 のようになっている。

表 1: 刻み幅の設定

推定平均報酬値	0.0 以上 0.3 以下	0.5 以下	0.7 以下	0.85 以下	1.0 以下
刻み幅	9	7	5	3	2

そして、make_para メソッドで前回の最適値を基準値として設定された刻み幅によって任意の数の選択肢が生成され、新しい選択肢を使用して再度学習を行う。

また、強化学習 ε -greedy 法の戦略として、 ε でランダムな選択肢を選ぶ確率を調整することができる。本研究では実世界での実験を行うため、一度の実験の試行回数を多くすることは難しい。そのため、 ε を小さくしすぎると一度も選ばれない選択肢が出てしまうという問題があった。また、大きすぎても報酬の総計の期待値が減ってしまう。今回、刻み幅を変化させるという手法をとったため、刻み値によってランダム性を変化させることで効率の良い学習を目指した。刻み値が大きな場合はランダム性を高めて平均報酬の推定値を得るためのデータを多く集めるようにした。 ε の設定の詳細は表 2 のようにした。

表 2: ε の設定

刻み幅	9,7	5	3,2
ε	0.3	0.2	0.1

4 実験

本研究で構築した学習機構により、NXT が自身で環境に適した基本行為の獲得が可能となった。その有用性を検証するために実験を行った。

4.1 実験手法

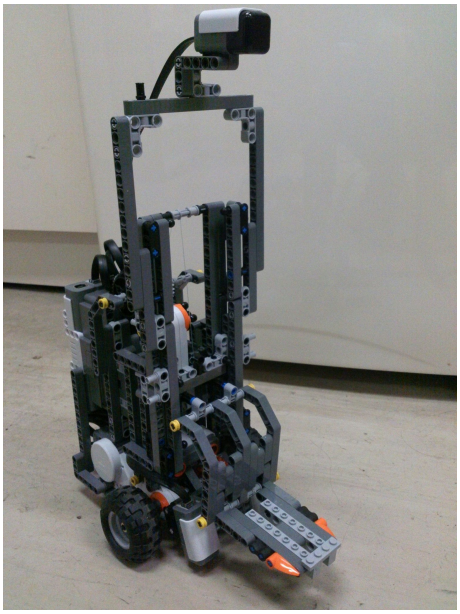
今回の実験では 2 台の構造の異なる NXT(図 3) を使用し、強化学習プログラムを使うことによって NXT 自身で 90 度回転するためのパラメータの調整を行えるようにした。重量の違う NXT を用いることで今回のプログラムがそれぞれの NXT に対応したパラメータを学習することが可能であることを検証した。

なお、実験ではプログラムによる学習を 3 度行い、それぞれの試行回数は 70 回とした。NXT の左右のモーターは片方のみ動かすこととし、左折もしくは右折のみのパラメータ調整を行うこととする。今回、パラメータの選択肢は 3 つとし、初期値は一度目の学習では（65, 74, 83）と設定した。この場合、刻み幅は最大値の 9 である。また、パラメータのリストは二度目以降は前回の実験の最適値とその推定平均報酬値を基に自動生成される。

実験結果はテキストファイルに保存され、学習した結果の最適なパラメータとその推定平均報酬値を保存する。今回、学習の経過を検証するために 10 回の試行ごとに学習の現状の出力と保存を行った。

実際に実験に使用したコースは図 4 である。これは、視覚的にも 90 度の回転がわかりやすいコースを設定した。NXT はコース上で、線上をライントレースで進む。この際、ライントレースのプログラムは NXT Python+のものを用いた。NXT は交差点で停止し、回転角度の学習の試行を行う。今回の実験では、NXT が線上から外れ、ライントレースを行うことが出来ない角度で回転した場合にもコース外へ外れることがないように、2 重の円のコースを設定した。また、コースから外れる可能性のある場所では学習を行わず、人の手による進路選択の指示も可能である。

[1]NXT 1



[2]NXT 2

図 3: 実験用 NXT

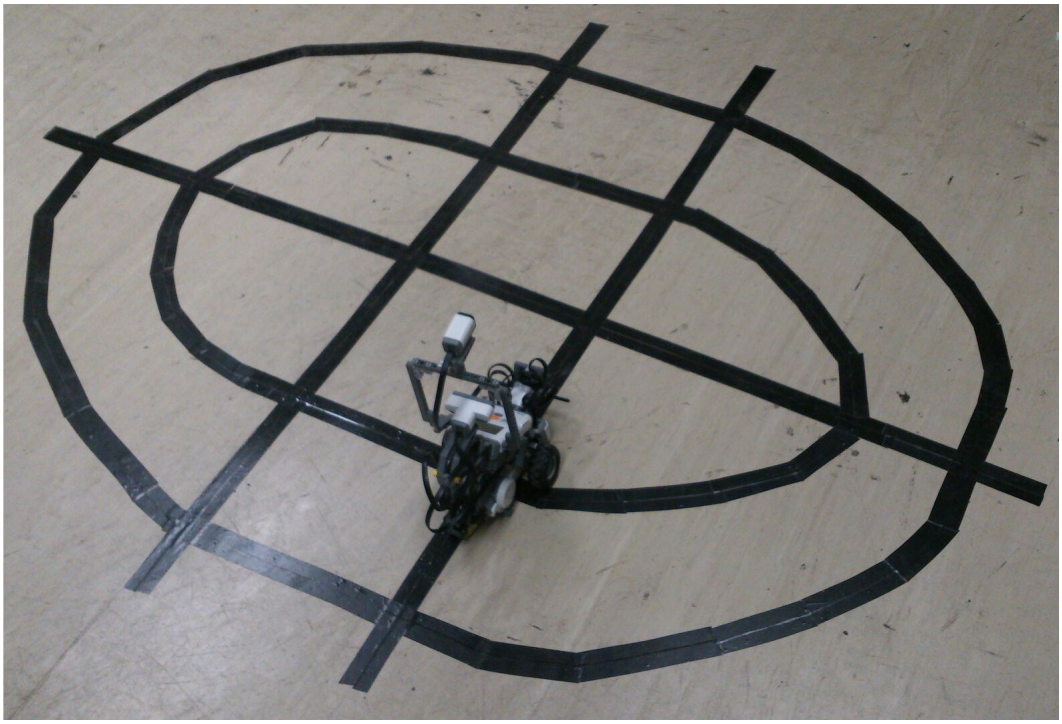


図 4: 実験コース

4.2 実験結果

ここでは 2 台の NXT をそれぞれ NXT1,NXT2(図 3) と呼ぶ。NXT1 でプログラムを動かしてみると、一度目の試行の出力結果は以下 (表 3) のようになった。

表 3: 出力結果:一度目

10 回試行
選択肢 0:　65　3 選択　推定平均報酬 0.28
選択肢 1:　74　7 選択　推定平均報酬 0.79
選択肢 2:　83　0 選択　推定平均報酬 0.00
平均報酬　0.63

20 回試行
選択肢 0:　65　4 選択　推定平均報酬 0.31
選択肢 1:　74　9 選択　推定平均報酬 0.76
選択肢 2:　83　7 選択　推定平均報酬 0.90
平均報酬　0.72

このように最初は選択肢 1 の選択回数が多いが、20 回試行終了の時点で推定平均報酬の一番高い選択肢 2 の選択回数が増えるようになる。そして、60 回,70 回試行終了時点での試行結果は以下 (表 4) のようになった。

表 4: 出力結果:一度目 (2)

60 回試行
選択肢 0:　65　6 選択　推定平均報酬 0.31
選択肢 1:　74　14 選択　推定平均報酬 0.77
選択肢 2:　83　40 選択　推定平均報酬 0.81
平均報酬　0.75

70 回試行
選択肢 0:　65　6 選択　推定平均報酬 0.31
選択肢 1:　74　14 選択　推定平均報酬 0.77
選択肢 2:　83　50 選択　推定平均報酬 0.82
平均報酬　0.76

以上のように推定平均報酬の高い選択肢 2 の選択回数が最終的に最も多くなる。このことから NXT1 の実験の一度目は選択肢 2 のパラメータ 83 が最適値として保存される。

実験の二度目はパラメータの選択肢が変化する。前回の最適値の 83 を基準値とし、その推定平均報酬の 0.82 が 0.7 以上 0.85 以下なので、刻み幅は 3 となる。そのため、二度目の選択肢は (80,83,86) となった。プログラムの出力結果の一部を以下 (表 5) に示す。

以上の結果より選択肢 0 のパラメータ 80 が 2 度目の最適値として保存された。この最適値を用いて三度目の試行の選択肢を生成した結果、(78,80,82) が三度目の選択肢となった。プログラムの出力結果の一部を以下 (表 6) に示す。

表 5: 出力結果:二度目

30 回試行
選択肢 0:　80　28 選択　推定平均報酬 0.87
選択肢 1:　83　0 選択　推定平均報酬 0.00
選択肢 2:　86　2 選択　推定平均報酬 0.54
平均報酬　0.85

70 回試行
選択肢 0:　80　60 選択　推定平均報酬 0.86
選択肢 1:　83　5 選択　推定平均報酬 0.83
選択肢 2:　86　5 選択　推定平均報酬 0.59
平均報酬　0.84

表 6: 出力結果:三度目

30 回試行
選択肢 0:　78　26 選択　推定平均報酬 0.87
選択肢 1:　80　1 選択　推定平均報酬 0.80
選択肢 2:　82　3 選択　推定平均報酬 0.74
平均報酬　0.85

70 回試行
選択肢 0:　78　62 選択　推定平均報酬 0.86
選択肢 1:　83　3 選択　推定平均報酬 0.82
選択肢 2:　86　5 選択　推定平均報酬 0.81
平均報酬　0.84

NXT2 でも同様に実験を行い、それぞれの NXT の実験結果を表にまとめた。表 7,8 では 3 度実験を行ったうちの最終的な結果をまとめている。

表 7,8 の結果を見ると、1 度の実験ごとに刻み幅を変えることによって初期値から確実に適切な数値へと近づくことができている。また、それぞれで最良の推定平均報酬である選択肢を選択できていることがわかる。

今回の実験では実験時間は NXT1 では 31 分 04 秒、NXT2 では 27 分 03 秒と、どちらも 30 分前後で実験を終えることができた。本研究の初期段階では共に 1 時間ほどかかっていたので、約半分の時間短縮となり、学習の効率化を行えたことが示せた。

4.3　考察

複数回の実験を前提とし、実験ごとに選択肢を変化させることで初期値に影響されることなくそれぞれの NXT が適切なパラメータを獲得することが出来た。この選択肢の生成には刻み幅の変更という工夫をすることで前回の学習を利用した、より効率の良い学習方法を取ることができた。

また、コンパスセンサを利用した評価による報酬の獲得によって、より人手を必要としない実験を行うことが出来た。NXT のコンパスセンサは磁場に影響されやすく、正確な方位の計測には不

表 7: NXT1 の実験結果

	選択肢	選択回数	推定平均報酬	平均報酬	実験時間
1 回目	65	6 回	0.31	0.76	10:29
	74	14 回	0.77		
	83	50 回	0.82		
2 回目	80	60 回	0.86	0.84	10:29
	83	5 回	0.83		
	86	5 回	0.59		
3 回目	78	62 回	0.86	0.86	10:16
	80	3 回	0.82		
	82	5 回	0.81		

表 8: NXT2 の実験結果

	選択肢	選択回数	推定平均報酬	平均報酬	実験時間
1 回目	65	13 回	0.29	0.71	9:39
	74	10 回	0.64		
	83	47 回	0.84		
2 回目	80	13 回	0.82	0.83	7:49
	83	1 回	0.71		
	86	56 回	0.83		
3 回目	83	6 回	0.72	0.83	9:35
	86	62 回	0.84		
	89	2 回	0.78		

向きであったため、これまでの研究では実用化が困難だった。しかし、コンパスセンサをロボットの向きの決定に直接用いるのではなく、行動の評価にのみ用いる今回の使用方法においては人間の視覚での評価との差は見られず、十分に活用することが可能だった。コンパスセンサでの評価により人手の介入を減らせたことで大幅な実験時間の短縮に繋がり、NXT が最適な行動を獲得することに寄与した。

5　まとめと今後の課題

今回、実世界における小型ロボット NXT の強化学習が可能となった。本研究によって、NXT が自身で環境に適した最適な基本行為を獲得できる学習機構プログラムが実現した。

今回の実験では異なる重量の NXT を同環境下で実験を行い、それぞれに適したパラメータを獲得するための学習結果を検証したが、今後の研究では異なる環境下においても最適なパラメータを獲得できるかを検証していく必要がある。

また、今後の課題として、人手を必要としない学習方法の発案、それに対応したコース環境の設定が必要である。そして少ない回数、実験時間でよりよい学習が行える工夫をしていきたい。また、実世界の動的な環境に対応するために、回転の角度の変更に対応する機能やライントレースな

しでの学習を目指したい。以上のように、より効率の良い学習を NXT が行えるように改善することが今後求められる。

6　謝辞

本研究を遂行するにあたり、丁寧にご指導して下さった指導教官の新出尚之准教授に深く感謝し、厚く御礼申し上げます。また鴨浩靖准教授、更に新出、鴨研究室の皆様に感謝の意を表します。ありがとうございました。

参考文献

[1] 小島侑子. 小型ロボット制御のための汎用ライブラリの構築. 修士論文, 奈良女子大学大学院人間文化研究科博士前期課程情報科学専攻, 2012.