深層強化学習における報酬関数の自動生成及び自動修正に関する研究

TR24008 鈴木 佳三　　指導教授：藤吉 弘亘

1. はじめに

強化学習 (RL) は，エージェントが環境との相互作用を通じて方策を学習する機械学習手法の一種であり，ロボット制御やゲーム攻略などの分野で応用が進んでいる．RL では，環境から与えられる報酬をもとに行動を評価し，報酬を最大化するように方策を更新する．そのため，報酬関数の設計はエージェントの学習や性能を左右する重要な要素である．一方で複雑なタスクでは報酬の設計が難しく，専門知識や試行錯誤への依存が大きな課題となっている．この問題に対し，大規模言語モデル (LLM) を用いて報酬関数を自動生成・修正する Text2Reward (T2R) [1] が提案されている．これにより，自動で報酬関数を生成できる一方で，生成された報酬関数が必ずしも実行可能であるとは限らず，環境の仕様と不整合なコードや未定義変数を含む場合がある．また，報酬関数の修正において人間による評価やフィードバックを前提としており，設計者の主観や負担に依存する点が課題として残されている．

本研究では，これらの課題に対処するため，報酬関数の自動生成および自動修正を安定して実現するフレームワークを提案する．提案手法では，LLM が生成した報酬関数の実行可能性を担保する Auto Debug Module と，自動的に RL 結果を分析する Feedback LLM を導入することで，人間に依存しない報酬関数の生成および改善を目指す．

2. RL における報酬設計と従来法

RL において，報酬関数はエージェントの行動を数値的に評価し，学習の方向性を決定づける重要な要素である．エージェントは報酬を最大化するように方策を更新するため，報酬設計は最終的な方策の性質や学習性能に大きな影響を与える．

2.1 人手による報酬設計

一般的な RL では，タスクの目的を人間が解釈し，目標状態への到達や制約条件の遵守などを評価基準として，状態や行動に応じた報酬を定義する．報酬は単一の項目で与えられる場合もあるが，多くの場合は複数の評価項目を組み合わせた加算形式で表現される．この際，各評価項目に対する重み付けや，疎報酬か密報酬かといった報酬形状を人手で設定する．これらの設定は探索の容易さや学習の安定性に強く影響するため，学習曲線やエージェントの行動例を観察しながら報酬関数を反復的に修正する試行錯誤が必要となる．専門家はタスク構造や RL アルゴリズムの特性を踏まえた調整が可能である一方，非専門家にとっては，どの評価項目をどの程度強調すべきかを判断することが難しい．その結果，報酬関数の品質に差が生じ，意図しない行動の誘発や学習の停滞が生じることがある．

2.2 従来手法：Text2Reward

T2R は，LLM を用いて報酬関数を自動生成し，人間のフィードバックを通じて修正する手法である．図 1 に示すように，まず LLM に対して RL タスクの概要や環境情報を与え，報酬関数のコードを生成させる．生成された報酬関数を用いて RL を実施し，その学習結果やエージェントの振る舞いをもとに，人間がフィードバックを与えることで報酬関数を更新する．

T2R は，従来人手に依存していた報酬設計を自動化する可能性を示している．一方で，実際の RL タスクに適用する際には，生成された報酬関数の実行可能性や，人間によるフィードバックの与え方に起因する問題がある．そこで，T2R を用いた事前調査を行い，これらの問題点を明らかにする．

2.3 事前調査 1：生成関数の実行可能性

本調査では，LLM が生成する報酬関数の実行可能性について検証する．LLM に対して，ManiSkill2 の PushChair-v1 を対象とした報酬関数の生成を依頼し，生成された関数を実際の RL 環境上で実行する．実行時にエラーが発生した場合は，当該関数を実行不可能と判定し，新たに報酬関数の生成を行う．この操作を繰り返し，10 個の実行可能な報酬関数が得られるまで試行する．

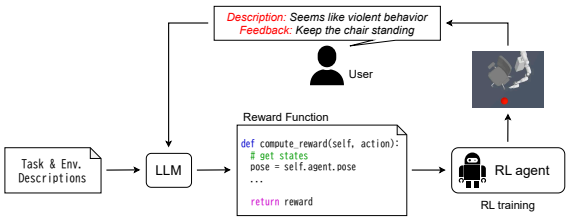

図 1: Text2Reward

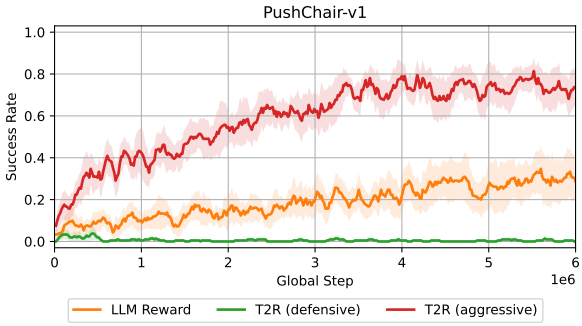

図 2: フィードバック方針の違いによる RL への影響

結果を表 1 に示す．この調査により，LLM が生成した報酬関数には，未定義の変数や環境に存在しない属性を参照する記述が含まれる場合が多く，実行可能な関数を得るためには複数回の生成が必要であることがわかった．このことから，T2R では報酬関数生成の段階で実行可能性が十分に担保されておらず，LLM による生成に不安定性が存在することが確認できる．

表 1: T2R における報酬関数の実行可能性

LLM	総生成回数	実行可能率 [%]
GPT-4o	36	27.8
GPT-5	38	26.3

2.4 事前調査 2：修正方針の違いによる RL への影響

本調査では，報酬関数の修正時に用いるフィードバックの違いが与える RL への影響について調査する．GPT-5 が生成した報酬関数 (LLM Reward) を異なる 2 つの方針で修正し，それぞれで RL を実施する．一つは安全性や必要最低限の動作を重視する保守的な修正方針 (defensive) であり，もう一方はタスクの達成速度や効率を重視する積極的な修正方針 (aggressive) である．これらのフィードバックに基づいて報酬関数を修正し，同一条件下で RL を実行した．RL タスクには，PushChair-v1 を用いる．

結果を図 2 に示す．aggressive は最高で 80%程度のタスク成功率を示したが，defensive は終始 0%付近で停滞している．この調査により，フィードバック方針の違いによって学習の進行や最終的な性能に大きな差が生じることが確認された．同じ初期報酬関数を用いた場合であっても，人間の判断による修正内容の違いがタスク成功率やエージェントの行動に大きく影響する．この結果は，T2R における報酬修正が人間の熟練度や価値観に強く依存していることを示しており，非専門家にとって安定した性能向上を達成することが困難であるという課題を示唆している．

3. 提案手法：Auto-Text2Reward

本研究では，図 3 に示すように，報酬関数の自動生成および自動修正を安定して実現するためのフレームワークを提案する．本手法では，Code Generate LLM，Auto

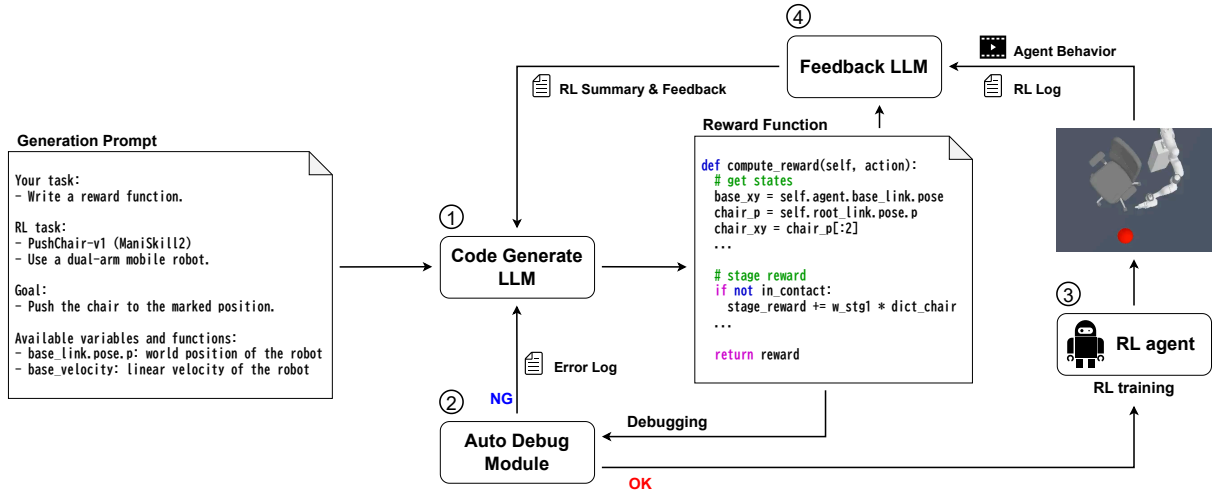

図 3: 提案手法 (Auto-Text2Reward)

Debug Module，Feedback LLM を組み合わせ，T2R における生成段階および修正段階の不安定性の低減を図る．

3.1 報酬関数の自動生成

存在しない変数の参照や記述ミスを抑制するために，以下の手順に従い報酬関数を生成する．まず，①の Code Generate LLM が，RL 環境で直接使用可能な変数や関数を明示したプロンプトを基に，報酬関数を生成する．次に，②の Auto Debug Module に生成した報酬関数を渡し，環境において正常に実行可能であるか検証する．ここでエラーが発生した場合，Auto Debug Module はエラーログを LLM に共有し，再生成を行うことで実行可能な報酬関数のみを選別する．正常に実行可能な報酬関数が得られたら，③の実際の RL 環境に渡り，RL を実施する．

3.2 報酬関数の自動修正

人間の熟練度や判断基準に依存しないように，RL の結果を自動的に分析する Feedback LLM を導入する．RL の実施後，④の Feedback LLM を用いて RL の分析を行う．ここでは，学習中のタスク成功率や獲得報酬の推移などの RL ログデータ，エージェントの振る舞いを記録した画像列を入力として受け取り，学習状況やエージェントの振る舞いを分析する．その後，使用した報酬関数と分析結果を踏まえ，報酬関数の改善案であるフィードバックを生成する．これらの分析結果とフィードバックに基づき，①の Code Generate LLM が再び報酬関数を生成する．

4. 評価実験

提案手法の有効性を検証するため，報酬関数の実行可能性と，報酬設計の品質に関する評価実験を行う．

4.1 報酬関数の実行可能性

生成した報酬関数の実行可能性を検証するため，ManiSkill2 のタスクを対象として報酬関数を生成する．事前調査 1 と同様に，LLM には GPT-5 を用いて，10 個の実行可能な報酬関数を生成するまでに要した試行回数を計測する．

結果を表 2 に示す．提案手法は，いずれのタスクにおいても従来手法より少ない試行回数で実行可能な報酬関数を生成しており，プロンプト改善および Auto Debug Module が報酬関数の実行可能性を向上させていることが確認できる．

表 2: 提案手法における報酬関数の実行可能性

タスク	手法	総生成回数	実行可能率 [%]
PushChair-v1	T2R	38	26.3
	Ours	14	**71.4**
OpenCabinetDoor-v1	T2R	72	13.9
	Ours	11	**90.9**
OpenCabinetDrawer-v1	T2R	75	13.3
	Ours	14	**71.4**

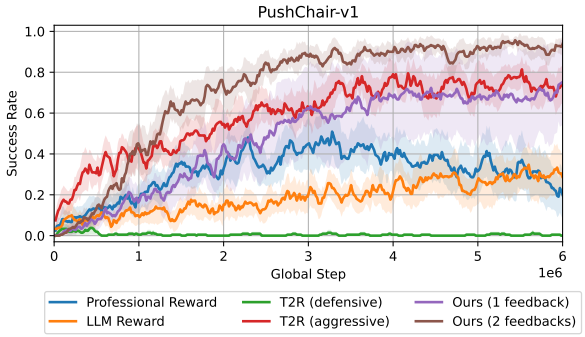

図 4: タスク成功率の推移

4.2 報酬設計の品質

生成した報酬関数の品質を検証するため，PushChair-v1 を用いて RL を実施する．比較対象として，専門家設計の Professional Reward，GPT-5 が生成した LLM Reward，T2R によって LLM Reward を保守的および積極的な方針で修正した T2R (defensive/aggressive)，提案手法によって n 回の自動修正を行った Ours (n feedback) を用いる．

各エージェントの学習過程におけるタスク成功率を図 4 に示す．提案手法は，いずれの報酬関数よりも高いタスク成功率を達成した．また，提案手法による修正を繰り返すことで，修正前の報酬関数を用いた学習より性能が向上している．以上より，提案手法は人間を介せず，報酬関数の自動生成および自動修正を実現したといえる．

5. おわりに

本研究では，RL における報酬設計の困難さに着目し，報酬関数の自動生成および自動修正を安定して実現するフレームワークを提案した．T2R の課題に対し，提案手法では，生成された報酬関数の実行可能性を担保する Auto Debug Module と，RL の結果を分析してフィードバックを生成する Feedback LLM を導入した．評価実験より，提案手法は専門家が設計した報酬関数や従来手法により得られた報酬関数と比較して，エージェントの性能を改善させることが可能であることを確認した．

今後は，RL 分析サマリーのさらなる最適化や，動画情報の導入方法，フィードバック生成頻度の制御，あるいは軽量な言語モデルとの併用といった観点から，計算効率と分析精度の両立を検討する．

参考文献

[1] T. Xie, *et al.*, “Text2Reward: Reward Shaping with Language Models for Reinforcement Learning”, ICLR, 2024.

研究実績

[1] 鈴木佳三 等, ”MaskDP による事前学習のマルチドメイン拡張”, 日本ロボット学会学術講演会, 2024.