強化学習を用いた技術者センスの評価手法の提案

Evaluation of Technical Sense for Engineering using Reinforcement Learning

村上 雄飛

2025年3月1日

東京都立大学法人
東京都立産業技術高等専門学校
情報通信工学コース 山本研究室

研究概要

ロボットの動作制御の開発において, 強化学習に関する研究は数多く進められており, 近年では産業用ロボットや自律移動ロボットなど, 実用面でも優れた成果を挙げている. 一方で, 効果的な学習を実現するためには, アルゴリズムだけでなく, 技術者が適切なパラメータを設定し, 試行錯誤を通じて結果を最適化する能力が極めて重要とされる.[1] この「技術者のスキル」は, 特定の課題を効率的かつ迅速に解決する上で不可欠であり, 特に学習効率や最終的な性能に大きな影響を及ぼす. しかし, 従来はスキルチェックテストやポートフォリオによって判断されてきたため, これらは過去の成果や蓄積された知識の確認に偏りがちであり, 実務で求められる直感力や重要なポイントを見極める洞察力といった, 適応力を直接測定することは困難であった. 特に, 強化学習のように多次元かつ複雑なパラメータ空間を扱う課題では, 技術者の経験や知識が結果に大きく影響するにもかかわらず, これらのスキルを定量的に評価する標準的な指標は存在していない. そこで本研究では, 強化学習の設計プロセスを題材とし, 技術者がどのように問題にアプローチし, 最適化を進めるかを分析することで, 技術者のスキルを客観的に評価する新たな手法を提案する.

Abstract

In recent years, many researches on reinforcement learning have been conducted in the development of robot motion control, and excellent results have been achieved in practical applications, such as industrial robots and autonomous mobile robots. Here, the ability of engineers to optimize is important for effective reinforcement learning. This ability is not only to consider the algorithm, but also to be able to set appropriate parameters and perform trial-and-error.[1] This ability is essential for the efficient solution of tasks and has a big effect on learning efficiency and final results on performance. However, this skill has been conceived through skill check tests and portfolios, which tend to be biased to confirming past results and knowledge. Therefore, it was difficult to directly measure adaptive skills such as intuition and the ability to identify important points in practice. In particular, reinforcement learning deals with multidimensional and complex parameter spaces. Therefore, the experience and knowledge of the engineer have a big effect on the results, but there is no standard indicator to quantitatively assess these skills. In this study, we use the design process of reinforcement learning to analyse how engineers approach and optimise problems. In this way, we suggest a new method for objectively evaluating the skills of engineers.

目 次

1. 序論

1.1. 研究背景

近年, ロボット工学の分野において強化学習（Reinforcement Learning, RL）は優れた成果を上げており, 自律移動ロボットやドローンの動作制御など, 多岐にわたる応用が進められている. 強化学習は, エージェントと呼ばれる自律システムが学習環境との相互作用を通じて最適な行動方策 (行動の選び方) を獲得する手法であり, 未知の環境下で自律的に行動を獲得することが可能であり, 環境に合わせた柔軟な動作制御を実現することができる. しかし, 強化学習の性能を最大化するためには, アルゴリズムの選定だけでなく, ロボットの部位ごとのパラメータや報酬設計, 学習率, 割引率など, 多くのパラメータを適切に設定する必要がある. これらのパラメータ調整は技術者の試行錯誤に依存するため. その調整プロセスは強化学習の成果に大きな影響を与える.

特に, 強化学習の設計においては, 多次元かつ非線形なパラメータ空間を扱う必要があり, 技術者の経験や直感が重要な役割を果たす. 技術者は試行錯誤を繰り返しながら最適なパラメータを見出すが, そのプロセスは技術者の知識や経験に依存するため, 定量的に評価することは極めて難しい. 従来のスキル評価手法では, スキルチェックテストやポートフォリオのような形式的な方法に依存しており, 知識や過去の実績を確認するものが主流である. しかし, 実務で求められる柔軟な問題解決能力や直感的な判断力, 重要な要素を迅速に特定する洞察力といったスキルを, これらの手法で正確に評価することは困難である.

さらに近年の研究では, 強化学習のパフォーマンスはアルゴリズムそのものよりも, 技術者によるハイパーパラメータの調整や環境設計に依存する場合があることが指摘されている.[2] これは, 技術者のスキルが強化学習システム全体の最適化プロセスにおいて重要な役割を果たしていることを示唆している. しかし, このようなスキルを定量的に評価する標準的な指標や評価手法は未だ確立されておらず, 技術者の能力向上やチーム全体のパフォーマンス最適化のための体系的なフィードバックが不足しているのが現状である.

1.2. 研究目的

本研究の目的は, 強化学習の設計過程を利用して技術者のスキルを可視化し, 客観的に評価する新しい手法を定義することである. 従来のスキル評価は, 過去の実績や知識の確認に偏っており, 実務において重要な直感力や問題解決能力を正確に測定することが困難であった.

そこで本研究では, 技術者が強化学習の設計過程においてどのようにパラメータを設定し, 試行錯誤を重ねるかを分析し, 技術者のパラメータ選択の「センス」や「ポイント発見能力」を定量的に評価する手法を提案する. これにより, 強化学習の設計における技術者の能力を客観的に評価し, スキル向上のための体系的なフィードバックを可能とすることを目指す.

2. 理論

2.1. 強化学習の基礎理論

強化学習（Reinforcement Learning, RL）は, 図 1 に示す, エージェント（自律システム）が環境と相互作用しながら, 試行錯誤を重ねて最適な行動を学習する機械学習の一分野である. エージェントは, 行動の結果として環境から報酬（またはペナルティ）を受け取り, その報酬を最大化するような方策（どんな行動をするかの判断基準）を学習する. また、図 2 に強化学習における状態, 行動, 報酬の遷移を示す.

この学習方法は, 教師あり学習 [3] のように正解ラベルを事前に与えるのではなく, エージェントが環境から得られるフィードバックをもとに最適な行動戦略を獲得する点が特徴であり, ロボット制御やゲーム AI, 金融取引最適化など, さまざまな分野で応用されている.

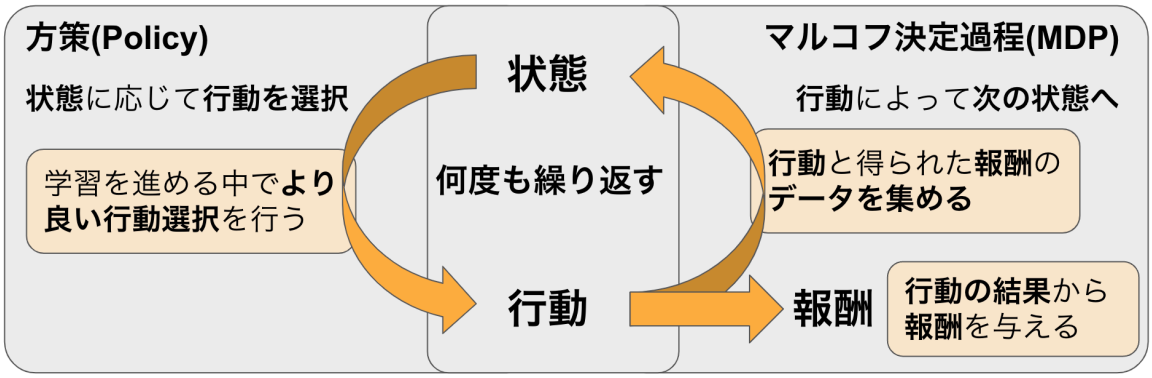

図 1: エージェントと環境との相互作用

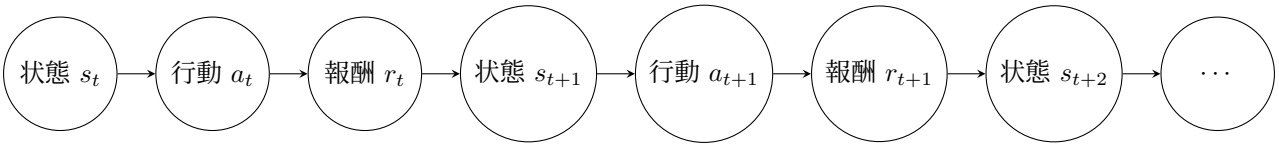

図 2: 強化学習における状態, 行動, 報酬の遷移

2.1.1 用語解説

強化学習を解説する上で重要な用語を表 1 に示す.

表 1: 強化学習の用語解説

用語	説明
エージェント（Agent）	環境と相互作用し, 行動を選択して学習を行う自律システム
環境（Environment）	エージェントの行動に対して報酬を与えるシステム
状態（State）	環境の現在の状況を表す情報. エージェントが観測する
行動（Action）	エージェントが環境に対して行う操作. エージェントは状態に基づいて行動を選択する
報酬（Reward）	エージェントの行動に対する環境からのフィードバック. エージェントの行動を評価する尺度
方策（Policy）	エージェントが各状態においてどの行動を選択するかを決定する戦略. $\pi(a\|s)$ で表され, 状態 s における行動 a の選択確率を示す
価値関数（Value Function）	特定の状態または状態-行動ペアに対する累積的な報酬の期待値. 状態価値関数 $V(s)$ や行動価値関数 $Q(s,a)$ 等の種類がある

2.1.2 学習手法

表 2 に示す等り, 強化学習には, 主に方策ベースと価値ベースの 2 つの学習手法 [4] があり, 方策ベースの学習では, エージェントは最適な方策 $\pi(a|s)$ を学習する. 具体的には, 方策勾配法（Policy Gradient Methods）[5] 等のアルゴリズムを使い, 各状態 s において最適な行動 a を選択する確率を学習する. また, 価値ベースの学習では, エージェントは各状態 s または状態-行動ペア s, a に対する価値（Q 値）を学習し, この価値を元にして最適な行動を選択する. 代表的なアルゴリズムに Q 学習（Q-learning）[6] や SARSA[7] がある.

表 2: 強化学習アルゴリズムの分類

学習手法	アルゴリズム例
方策ベース	REINFORCE, PPO
価値ベース	Q 学習, SARSA

2.2. 強化学習の基礎理論

2.2.1 マルコフ決定過程

マルコフ決定過程（MDP: Markov Decision Process）[8] は, エージェントが不確実な環境の中で, 行動を選択して報酬を最大化する問題をモデル化するためのフレームワークである. エージェントが環境と相互作用しながら行動を選択し, 報酬を得ることで目標を達成する問題を定式化するための枠組みであり,MDP は, 特に状態遷移がマルコフ性（Markov Property）を持つ環境において適用される.

マルコフ性とは, ある変数が 1 ステップ前の変数のみに影響を受けて確率的に変化する性質である. つまり, システムの現在の状態が与えられたとき, 将来の状態の確率分布は過去の状態の影響を受けず, 現在の状態のみに依存することを意味している. これを数式で表すと, 式 (1) が得られる.

$$P(S_{t+1}|S_t, S_{t-1}, \ldots, S_0) = P(S_{t+1}|S_t) \tag{1}$$

この性質を持つ状態遷移モデルをマルコフモデルと呼び, マルコフモデルに基づいて生成される状態遷移の系列はマルコフ過程（Markov Process）またはマルコフ連鎖（Markov Chain）と呼ばれる.

2.2.2 ベルマン方程式

ベルマン方程式は, 動的計画法 [9] のすでに計算した結果を保存するメモ化と呼ばれる手法を用いることで計算量を大幅に減らし, 状態価値関数（State Value Function）と行動価値関数（Action Value Function）を再帰的に表現する数式である, この方程式を用いることで, 最適な方策 π^* を求めるための価値関数を計算することができる.

状態価値関数 $V^\pi(s)$ は, エージェントが状態 s において方策 π に従って行動し続けたときに得られる期待報酬の総和である. これを再帰的に表現すると, 式 (2) のベルマン方程式が得られる. この式は, 現在の状態価値が, 即時報酬 $R(s,a)$ と, 次の状態の価値 $V^\pi(s')$ の割引値 γ の合計として表されることを意味する.

$$v_\pi(s) = \sum_{a,s'} \pi(a|s)p(s'|s,a)\left\{r(s,a,s') + \gamma v_\pi(s')\right\} \tag{2}$$

- s：現在の状態
- a：現在の行動
- s'：次の状態
- $R(s,a)$：報酬関数
- γ：割引率（$0 \leq \gamma \leq 1$）

2.2.3 モンテカルロ法

モンテカルロ法 [10] とは, ランダムな試行（シミュレーション）を多数回行うことで, 期待値や確率分布を推定する数値計算手法の一つである. モンテカルロ法に基づく強化学習では, 環境内でエージェントが多数のエピソードを経験し, それらの履歴から状態価値関数 $V(s)$ または行動価値関数 $Q(s,a)$ を推定する.

この推定方法は「逐次平均法（sample mean）」に基づいており, 回数が増えるほど推定値の精度が向上するため, マルコフ決定過程のように, 環境の遷移確率 $P(s'|s,a)$ や報酬関数 $R(s,a)$ を知らなくても適用可能である. また, 表 3 に示す通り, モンテカルロ法は, ランダムな試行を通じて価値関数を推定するため, 環境のモデルを必要とせず, 逐次的に学習を行うことができる.

表 3: モンテカルロ法の特徴

特徴	説明
ランダム試行	多数のランダムな試行を行うことで, 期待値や確率分布を推定
逐次平均法	試行回数が増えるほど推定値の精度が向上
環境モデル不要	環境の遷移確率や報酬関数を知らなくても適用可能

2.2.4 TD 法

TD 法（Temporal Difference, 時間差分）[11] は, 強化学習における価値関数の更新手法の一つであり, モンテカルロ法（Monte Carlo）と動的計画法（Dynamic Programming）の中間的なアプローチとして位置付けられる.TD 法は, 環境の完全なモデルが分からない場合でも, 実際の経験（サンプル）から学習できる特長を持つため, 現実世界のように何が起きるかわからない環境でも強化学習を適用することができる. TD 法では, 式 (3) に示す更新式を用いて状態価値関数を更新する.

$$V(s_t) \leftarrow V(s_t) + \alpha \left(r_t + \gamma V(s_{t+1}) - V(s_t) \right) \tag{3}$$

- α ：学習率（Learning rate, $0 < \alpha \leq 1$）
- γ ：割引率（Discount factor, $0 \leq \gamma \leq 1$）
- r_t ：時刻 t で得られた報酬
- $V(s_t)$ ：現在の状態 s_t における状態価値関数
- $V(s_{t+1})$ ：次の状態 s_{t+1} における状態価値関数

この式は,「現在の状態価値 $V(s_t)$ を, 実際に得られた報酬 r_t と次の状態の価値 $V(s_{t+1})$ を用いた推定値で補正する」という考え方に基づく.

また, 表 4 に示す通り,TD 法は, モンテカルロ法と比較して, 各ステップで逐次的に学習を行うため, オンライン学習に適しているという特長がある.

表 4: TD 法の特徴

特徴	説明
サンプルベース	実際の経験（サンプル）から学習する
モデル不要	環境の完全なモデルが不要
オンライン学習	各ステップで逐次的に学習を行う
バイアスと分散のトレードオフ	モンテカルロ法と動的計画法の中間的な特性を持つ

2.2.5 Q 学習

強化学習の具体的な学習方法の例として, 価値ベースの代表的な学習手法である Q 学習（Q-learning）[6] を解説する. Q 学習は,TD 法の枠組みを利用し, 状態と行動の組み合わせに対する価値を学習することで, 最適な方策を導出する. 強化学習においてエージェントが環境との相互作用を通じて最適な行動方策を学習するための代表的な手法であり, この手法では, 状態 s において行動 a をとったときの価値を表す行動価値関数 $Q(s,a)$ を学習し, 逐次的に更新することで最適方策を求める.

Q 学習では, 式 (4) の更新式を用いて Q 値を学習する.

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha \left[R(s_t,a_t) + \gamma \max_{a'} Q(s_{t+1},a') - Q(s_t,a_t) \right] \tag{4}$$

- s_t：時刻 t の状態
- a_t：時刻 t における行動
- $R(s_t,a_t)$：時刻 t での報酬
- α：学習率（$0 < \alpha \leq 1$）
- γ：割引率（$0 \leq \gamma \leq 1$）
- $\max_{a'} Q(s_{t+1},a')$：次の状態 s_{t+1} での最大 Q 値

この更新式は, 現在の Q 値を, 新たに得た報酬と将来の最大 Q 値の和を考慮して段階的に, 更新する.

Q 学習は以下の手順で進められる

1. Q 値の初期化: すべての $Q(s,a)$ を適当な値（例:0）に設定する.
2. 行動の選択: 現在の状態 s において,ϵ-greedy 法などを用いて行動 a を選択する.
3. 環境との相互作用: 行動 a を実行し, 報酬 $R(s,a)$ を観測し, 新しい状態 s' に遷移する.
4. Q 値の更新: 更新式に基づいて $Q(s,a)$ を修正する.
5. 終了条件の確認: エピソード終了条件を満たしていない場合,2 に戻る.

ε -greedy 法 [12] とは, 強化学習における探索（exploration）と活用（exploitation）のバランスを取るためのシンプルで効果的な手法である. このアルゴリズムは, エージェントが環境で行動を選択する際に, 確率的に最適な行動や, ランダムな行動を選択することを目的としている. 表 5 のようにランダムな行動 (探索) と最良の行動 (活用) を乱数を用いて行動を選択する.

表 5: ε -greedy 法の特徴

特徴	説明
探索（Exploration）	確率 ϵ でランダムな行動を選ぶ
活用（Exploitation）	確率 $1-\epsilon$ で現在の Q テーブルに従った最良の行動を選ぶ

例：$\epsilon = 0.01$ の場合, 1% の確率でランダムに行動を選び, 99% の確率で今までの学習で最も最良の行動を選択する.

2.3. 深層強化学習について

深層強化学習（Deep Reinforcement Learning, DRL）[13] は, 強化学習の枠組みにニューラルネットワーク技術を組み合わせたアルゴリズムであり, 特に大規模で複雑な状態空間や行動空間における問題を解決するために使用される. 従来の強化学習アルゴリズムである Q 学習や SARSA などは,Q 値の更新に必要なテーブルのサイズが膨大になり, 学習が非効率的になってしまう. これに対し, 深層強化学習では, ニューラルネットワークを用いた Q 値や方策関数を近似する手法により, 大規模な問題を扱うことができる. これによって従来の強化学習と比べて学習精度が大幅に向上した.

2.3.1 強化学習の問題点

強化学習の代表的なアルゴリズムである Q 学習では, 状態と行動のペアごとに Q 値を格納するテーブルを用いて学習を行う. そのため,Q 学習は状態空間が大きくなった場合, 状態と行動のペア数が指数的に増加するため, テーブルによる格納が不可能となり, 膨大な計算量により学習が困難となる.

さらに, 状態空間が連続的である場合,Q 値を格納するテーブルのサイズを無限に近い値としなければならない. この問題を解決するため, 深層強化学習はニューラルネットワークを用いて Q 関数を近似し, 状態空間の次元を縮小しつつ, 効率的に学習を進める.

2.3.2 ニューラルネットワーク

ニューラルネットワーク（NN: Neural Network）は, 人間の脳の神経回路を模倣した数理モデルであり, 機械学習や深層学習（Deep Learning）の基盤となる技術の一つである. ニューラルネットワークの基本単位であるニューロンは, 式 (5) のような数式で表される.

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{5}$$

- x_i は入力
- w_i は重み（学習パラメータ）
- b はバイアス（学習パラメータ）
- $f(\cdot)$ は活性化関数（Activation Function）
- y はニューロンの出力

このモデルは, 入力層（Input Layer）, 隠れ層（Hidden Layer）, 出力層（Output Layer）の 3 つの層から構成される. 図 3 に, シンプルな 3 層ニューラルネットワークの模式図を示す.

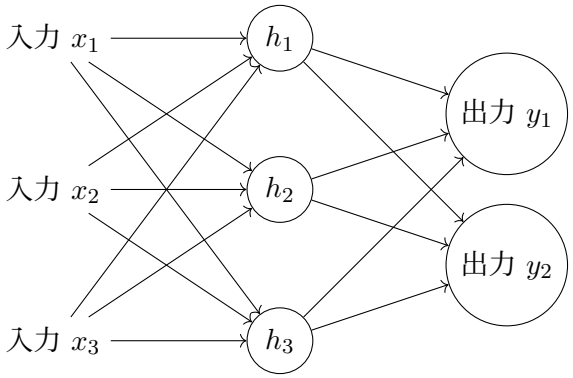

図 3: ニューラルネットワークの模式図

2.3.3 活性化関数

活性化関数（Activation Function）[14] は, 入力の線形結合を非線形変換する役割を持つ. ニューラルネットワークの各層で活性化関数を用いることで, モデルが複雑な関数を表現できるようになる. ニューラルネットワークの代表的な活性化関数として, シグモイド関数,[15]ReLU,[16] ソフトマックス関数 [17] がある.

1. シグモイド関数（Sigmoid Function）:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{6}$$

2. ReLU（Rectified Linear Unit）:

$$f(x) = \max(0, x) \tag{7}$$

3. ソフトマックス関数（Softmax Function）:

$$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{8}$$

2.3.4 損失関数

損失関数（Loss Function）[18] とは, 機械学習モデルがどれだけ誤差を含んでいるかを数値的に評価する指標であり, モデルの学習において最小化する目的関数の一つである. 損失関数の値が小さいほど, モデルの予測精度が高いと言える. 代表的な損失関数には平均二乗誤差（Mean Squared Error: MSE）[19] やクロスエントロピー損失（Cross-Entropy Loss）[20] があり, 式 (9) , 式 (10) の数式を用いて誤差を計算する.

平均二乗誤差

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{9}$$

クロスエントロピー損失

$$\text{Cross-Entropy} = -\sum_{i=1}^{N} \sum_{c=1}^{C} y_{i,c} \log \hat{y}_{i,c} \tag{10}$$

2.3.5　誤差逆伝播法

誤差逆伝播法（Backpropagation, BP）[21] とは, ニューラルネットワークの学習において, 損失関数の勾配を効率的に計算し, ネットワークの重みを更新する手法である. この手法により, ニューラルネットワークが入力データと正解ラベルを基に適切な重みを学習し, 予測精度を向上させることができる. また, この手法は, 順伝播（Forward Propagation）と逆伝播（Backward Propagation）の 2 つのステップから成る.

まず, 順伝播では, 入力から出力までの計算を行い, 損失を求める. 各層の出力は, 前の層の出力と重み W, バイアス b を用いて 式 (11) のように計算される.

$$z = Wa + b \tag{11}$$

活性化関数 f を適用し, 次の層の入力 a' を得る.

$$a' = f(z) \tag{12}$$

最後の出力層で予測値 $\hat{y}$ を求め, 損失関数 L を計算する.

次に, 逆伝播では, 損失 L の勾配を計算し, 重みとバイアスを更新する. 各層の誤差 δ は 式 (13) のように求められる.

$$\delta = \frac{\partial L}{\partial a} \cdot f'(z) \tag{13}$$

この誤差を利用し, 勾配降下法に基づいて重み W とバイアス b を更新する.

$$W := W - \eta \frac{\partial L}{\partial W} \tag{14}$$

$$b := b - \eta \frac{\partial L}{\partial b} \tag{15}$$

ここで,η は学習率であり, 適切に調整することで学習の速度と精度が変化する. また, プロセスを繰り返すことで, ニューラルネットワークは誤差を最小化し, より正確な予測が可能となる.

2.3.6　ニューラルネットワークでの学習

ニューラルネットワークは, 損失関数（Loss Function）を最小化するように学習する.

学習は以下の手順で行われる.

1. 順伝播（Forward Propagation）: - 入力をネットワークに通し, 出力を計算する.
2. 誤差の計算 : - 損失関数を用いて出力の誤差を求める.
3. 誤差逆伝播（Backpropagation）: - 誤差を各ニューロンに遡って伝播し, 重み w を更新する.
4. 最適化（Optimization）: - 勾配降下法を用いて重みを更新する.

2.3.7 DQN

DQN（Deep Q-Network）[22] は,Q 学習をニューラルネットワークと組み合わせた価値ベースのアルゴリズムであり,Q 関数をニューラルネットワークで近似することにより,Q 学習の問題を解決する. 具体的には,Q 値の推定器として深層ニューラルネットワークを利用する. Q 関数の近似は 式 (16) のように表される.

$$Q_\theta(s,a) \approx Q^*(s,a) \tag{16}$$

ここで,θ はニューラルネットワークのパラメータであり,Q 値の近似を行う. また,Q ネットワークの更新は, 次の損失関数を最小化することによって行い, 式 (17) のように表される.

$$L(\theta) = \mathbb{E}\left[\left(r_t + \gamma \max_{a'} Q_{\theta'}(s_{t+1},a') - Q_\theta(s_t,a_t)\right)^2\right] \tag{17}$$

ここで,θ' はターゲットネットワークのパラメータであり,$Q_{\theta'}$ はターゲット Q 値を表す.DQN では, ターゲットネットワークを定期的に更新することで, 学習の安定性を向上させる. また, 経験リプレイ（Experience Replay）を使用して, 過去の経験をランダムにサンプリングし, 学習に利用することで, 相関のあるデータに対するバイアスを減少させる.

2.4. PPO アルゴリズム

PPO（Proximal Policy Optimization）[23] は, 強化学習における方策最適化のためのアルゴリズムであり,TRPO（Trust Region Policy Optimization）アルゴリズムの改善版として提案された.PPO は, 安定性と効率性を向上させるために, 方策の更新を制約付きで行うことで, 大きな変更を避けつつ, 最適な方策を見つけることを目指す. 本節では, 方策勾配法を基にしたアルゴリズムの概要と,PPO の特性について解説する.

2.4.1 方策勾配法の基礎

強化学習における方策勾配法（Policy Gradient Methods）は, 方策 $\pi_\theta(a|s)$ をパラメータ θ で表現し, 期待累積報酬を最大化することを目的としている. 方策勾配法では, 方策のパラメータ θ を 式 (18) ように更新するための勾配を計算する.

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s) \cdot Q^\pi(s,a)\right] \tag{18}$$

ここで,$J(\theta)$ は期待累積報酬で,$Q^\pi(s,a)$ は状態 s と行動 a に対する行動価値関数である. 勾配 $\nabla_\theta \log \pi_\theta(a|s)$ は方策の対数微分で,$Q^\pi(s,a)$ はその行動を取ったときの報酬の期待値を示し, 式 (18) は, 方策 π_θ のパラメータを更新するために必要な勾配を計算する方法である.

2.4.2 TRPO アルゴリズムについて

TRPO（Trust Region Policy Optimization）[24] は, 方策の更新が大きすぎることを防ぐために, 方策更新に対する制約を設けたアルゴリズムである.TRPO では, 方策の更新量を制限するために, 式 (19) の制約を導入する.

$$\mathbb{E}_s \left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)} \right] \leq 1 + \delta \tag{19}$$

ここで,$\pi_{\theta_{\text{old}}}(a|s)$ は以前の方策で,$\pi_\theta(a|s)$ は現在の方策である. δ は許容される更新量の上限を定めるハイパーパラメータであり, この制約によって方策の急激な変化を防ぐことができる. これにより,TRPO は安定性が高く, 過度な方策の変更を防ぐことがでるが,TRPO には計算コストが高いという欠点がある.

2.4.3 PPO アルゴリズムの学習方法

PPO アルゴリズムは,TRPO の概念を簡素化し, 計算の効率性を高めるために提案された.PPO は, 式 (20) のような方策更新を行う.

$$L(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right] \tag{20}$$

ここで,$r_t(\theta)$ は 式 (21) の比率である.

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \tag{21}$$

$\hat{A}_t$ は, 状態 s_t におけるアドバンテージ関数であり, 報酬の推定誤差を示す. このアドバンテージ関数 $\hat{A}_t$ は, 現在の方策 π_θ と過去の方策 $\pi_{\theta_{\text{old}}}$ の違いを評価し, 方策の更新が最適に行われるように調整できる.

PPO では, 方策の更新量を制限するために, 比率 $r_t(\theta)$ を $[1-\epsilon, 1+\epsilon]$ でクリッピングし, このクリッピングによって, 方策の更新が急激に変わることを防ぎ, 安定した学習を促進する.ϵ は, 方策の変更を許容する範囲を設定するハイパーパラメータである.

2.4.4 PPO の特徴と利点

PPO は,TRPO の優れた安定性を維持しつつ, より計算効率が高く, 実装が簡単であるため, 多くの強化学習の実問題において優れたパフォーマンスを発揮する. 表 6 に PPO と TRPO の比較を示す.

表 6: TRPO と PPO の比較

特徴	TRPO	PPO
計算効率	低い	高い
安定性	高い	高い
実装の簡便さ	複雑	簡単

3. 提案手法

3.1. 提案手法の概要

これまでの多くの研究では, 強化学習のアルゴリズムやパラメータ最適化の手法については多くの議論がなされてきたが, 技術者がどのように適切なパラメータを選択し, それを改善していくのかというプロセスに着目した研究は少ない. また, 技術者のパラメータ選択の「センス」や試行錯誤の過程における「ポイント発見能力」を客観的に評価する手法は, 明確に定義されていない.

そこで, 本研究では, 強化学習の設計過程に着目し, 限られたリソースの中で効率的に業務を行うために求められる技術者センスの評価手法を提案する. 特に, ロボットの各部位に関するパラメータ調整に着目し, 技術者が試行錯誤を通じて選択した結果がロボット工学を学んだ技術者と専門技術を学んだことな無い人ではどのような違いがあるかを評価した. これにより, 今まで定量的に評価をすることのできなかった強化学習における技術者のスキルを客観的に定義し, 最適なパラメータ設計のためのフィードバックを提供する.

3.2. 具体的な提案手法

単純化した 4 足歩行ロボットのパラメータを部位ごとに選択してもらい, 平均報酬, 損失関数の値,KL-Divergence の値を比較した. この時, ロボット工学を学んだ技術者はセンスがあると仮定し, 技術者と専門技術を学んだことのない人とではどのような特徴が現れるかを分析した.

4. 実施内容

4.1. 実験概要

4.1.1 実験の進め方

本実験では，強化学習を用いたロボットのパラメータ選択に関する技術者センスを評価するため, 図 4 に示す単純化した 4 足歩行ロボットを用いて，一定時間内にロボットをより速く移動させることを目的とした課題を設定した.

被験者には 4.1.2 ロボットの部位ごとのパラメータ に示す 4 足歩行ロボットの主要素となるパラメータの選択を依頼した. 初回設定での計算ののち，シミュレーション結果と報酬グラフの遷移を被験者に提示し, スコアの向上を目指して前回の設定したパラメータを参考に, 再度パラメータを選択してもらった. このプロセスを 4 回繰り返し, 計 5 回のパラメータ選択を行ってもらった.

最終的な成果や改善の過程を平均報酬, 損失関数,KL-Divergence を通じて技術者としてのセンスや課題解決のためのセンスやポイント発見能力を評価・検討した．なお，表 7 に示す通り，実験に参加した被験者 A はロボット工学を学んだ教員, 被験者 B,C,D はそれぞれロボット工学, 情報通信工学, 医療福祉工学を学んでいる学生, 被験者 E,F はそれらの専門技術を学んだことのない人とし, 合計 6 名の被験者で実験を行った.

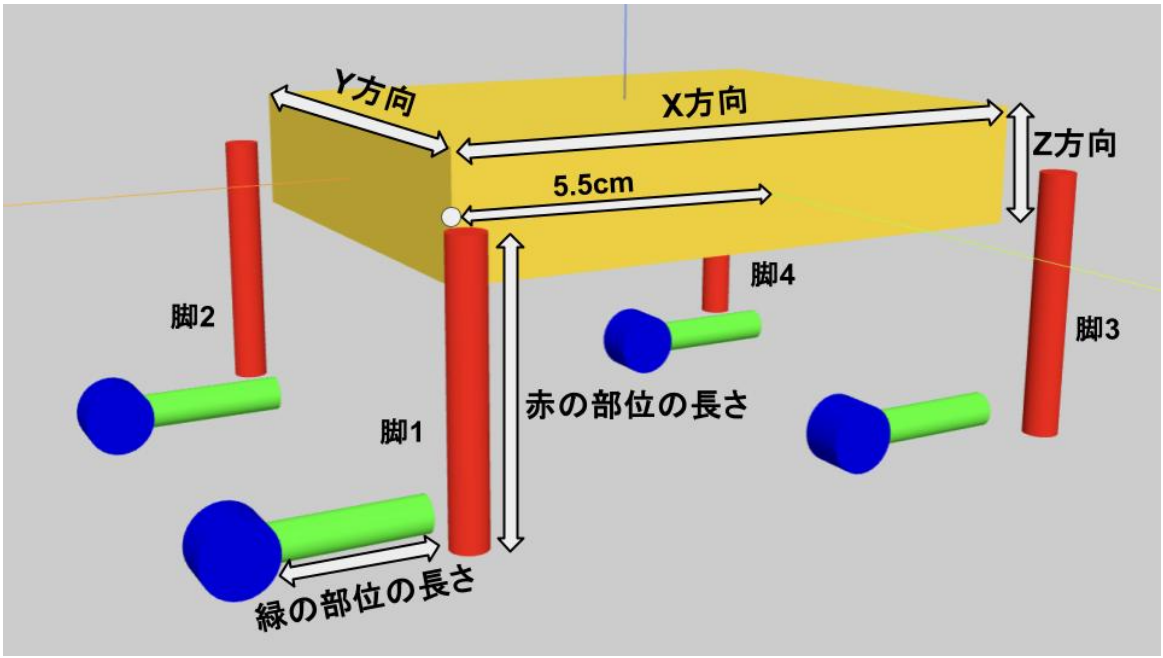

図 4: 単純化した 4 足歩行ロボットのモデル

表 7: 実験に参加した被験者の一覧

被験者	専門分野
A	ロボット工学（教員）
B	ロボット工学（学生）
C	情報通信工学（学生）
D	医療福祉工学（学生）
E	専門技術なし（素人）
F	専門技術なし（素人）

4.1.2 ロボットの部位ごとのパラメータ

表 8 に本実験において被験者が選択するロボットの部位ごとのパラメータを示す. 被験者には, 初回のパラメータ選択時に, デフォルトとして, 設定されたロボットのパラメータ例を提示し, そのパラメータを用いた際の強化学習の進行過程を動画で示した. これにより, 被験者はロボットの学習の仕組みを理解し, より速く前進できるロボットを設計するためにどのようなパラメータが適切であるかを考える基盤とした. また, ロボット技術に触れたことのない被験者がパラメータを考えて選択できるように, パラメータの設定は決められた選択肢から行えるよう工夫した.

表 8: 設定してもらうパラメータ

被験者が選択するパラメータ	選択肢
胴体のサイズ	8.5cm × 8cm × 1cm ～ 15.5cm × 8cm × 3.0cm の 8 段階
胴体の重さ (g)	56g ～ 166g の 11 段階
脚 1～4(赤) の長さ (cm)	1cm ～ 5cm の 9 段階
脚 1～4(緑) の長さ (cm)	1cm ～ 5cm の 9 段階
脚 1～4(青) の直径 (cm)	0.2cm ～ 1cm の 5 段階
脚 1～4(青) の長さ (cm)	0.4cm ～ 1.2cm の 5 段階
脚 1～4(青) の剛性値	1000N/m ～ 3000N/m の 3 段階
脚 1～4(青) の摩擦係数	0.5 ～ 1.5 の 3 段階

4.1.3 実験環境

本実験では, 強化学習のアルゴリズムとして Proximal Policy Optimization（PPO）を使用した. 報酬設計, 学習率, 割引率などのハイパーパラメータは, デフォルトのロボットモデルに適した値を設定し, 物理シミュレーションには PyBullet を用い, ロボットの構成要素を URDF 形式で定義した. また, シミュレーション環境は凹凸のない平坦な地形とし, ロボットの移動性能に関するパラメータの選択が影響を与えるかを評価した. 本実験の目的は, 被験者が設定された環境で適切なロボットの部位のパラメータを選択できるかを評価することである. そのため, トルクや速度に関する制約は設けず, 被験者が自由にパラメータを調整できる環境とした. 実験環境の概要を表 9 に示す. また, 図 5 に pybullet のシミュレーション環境, 図 6 に本実験で被験者のパラメータ選択に使用した GoogleForm を示す.

表 9: 実験環境の概要

項目	詳細
強化学習アルゴリズム	PPO（Proximal Policy Optimization）
シミュレーションエンジン	PyBullet
ロボットモデル形式	URDF（Unified Robot Description Format）
地形	凹凸のない平坦な地形
物理パラメータ制約	トルク・速度の制約なし

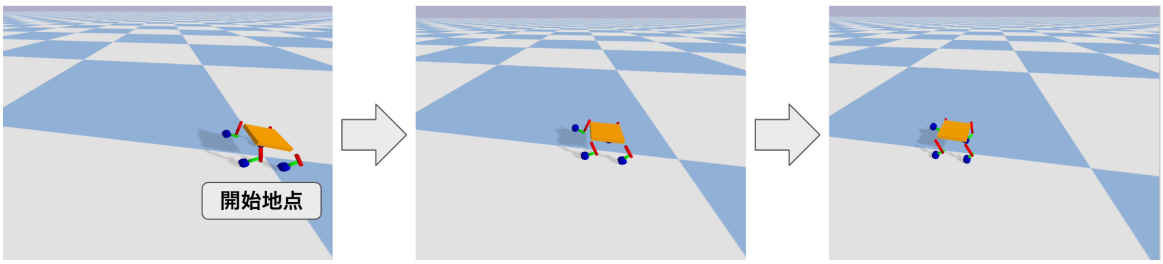

図 5: エージェントと環境との相互作用

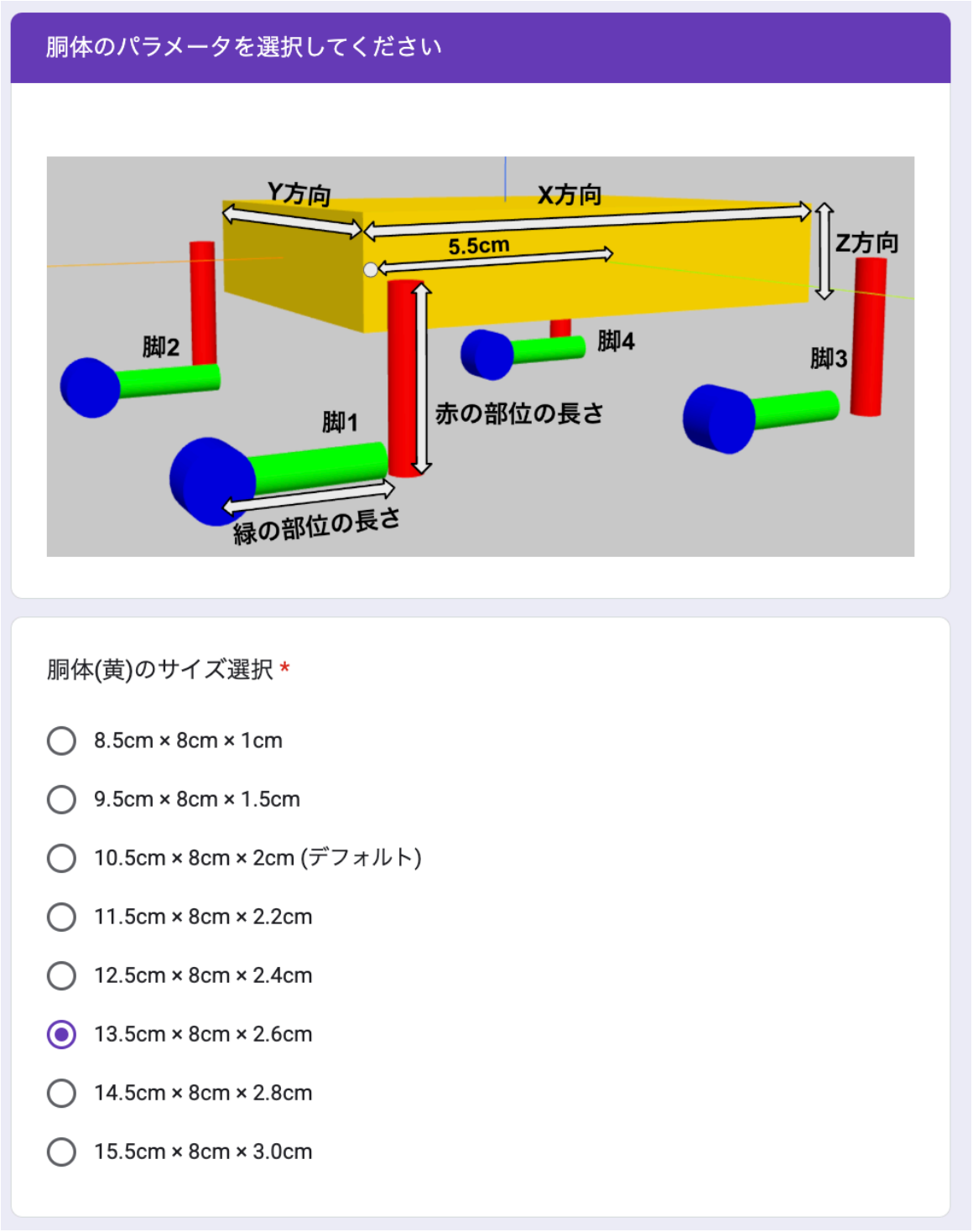

図 6: 実験で使用した GoogleForm

4.2. 比較したデータ

本実験では被験者のパラメータ選択のセンスを評価するため, 以下のデータについて解析した.

4.2.1 平均報酬

平均報酬は, エージェントが環境から得たエピソードごとの報酬の平均値であり, これは被験者が選択したパラメータでのロボットの行動の自由度を表している.

高い平均報酬は, 収束が早いが安定せず、即時報酬を優先していることを示している. 逆に, 低い平均報酬は, 長期的な報酬を優先しているため, 収束が遅いが安定していることを示している.

4.2.2 損失関数

損失関数は, エージェントの方策や価値関数の更新における誤差を示し, 強化学習ではこの損失関数を最小化することで, エージェントの学習を進める. 本実験では損失関数を組み合わせ, 式 (22) のように定義することで, 初期段階ではランダムな行動を行い, ある程度の探索が行われると報酬が最大化するように収束させた.

高い損失関数は予想通りの選択ができておらず, 低い損失関数は, 予想通りに選択が適切であることを示している. そのため, 損失関数は,p らメータが適切か否かを判断することができる.

$$L_{\text{total}} = L_{\text{policy}} + c_1 L_{\text{value}} - c_2 L_{\text{entropy}}. \tag{22}$$

- L_{policy} : 方策損失
- L_{value} : 状態価値損失
- L_{entropy} : エントロピー損失
- c_1 : 価値損失の重み
- c_2 : エントロピー損失の重み

4.2.3 KL-Divergence

KL-Divergence は, 2 つの確率分布間の差異を測定する指標で,PPO アルゴリズムでは, 現在の方策と以前の方策の間の KL-Divergence を監視することで, 方策の更新が適切に行われているかを確認する. 具体的には, 式 (23) のように計算される.

KL-Divergence の値が 1 よりも大きい場合, 方策の更新が急激すぎる可能性があり, 学習の安定性が損なわれることがある. また, グラフからパラメータ選択の方針をどこで変えているのかを判断することができる. そのため,KL-Divergence は, パラメータ選択の方針が正しいかを評価することができる.

$$D_{KL}(\pi_{\theta_{\text{old}}} || \pi_\theta) = \mathbb{E}_{s\sim\pi_{\theta_{\text{old}}}} \left[\sum_a \pi_{\theta_{\text{old}}}(a|s) \log \frac{\pi_{\theta_{\text{old}}}(a|s)}{\pi_\theta(a|s)} \right] \tag{23}$$

5. 実施結果

5.1. 被験者が選択したパラメータ

図 7 に, 被験者が選択したパラメータを適用した 4 足歩行ロボットのモデルの一例を示す. また, 表 10 , 表 11 に各被験者が選択したパラメータの一覧を示す.

各被験者の選択結果から, 脚の長さに関するパラメータの調整を行う傾向あり, 特に緑の脚部分の変更は全ての被験者が調整を行っている. また, 4 足全ての脚のパラメータを選択できるように実験を行ったが全ての被験者が前脚と後脚でそれぞれ設定を行う場合と全ての脚を同じ設定をする場合の 2 種類のみしかおらず, 全ての脚をそれぞれ異なる設定にする被験者はいなかった.

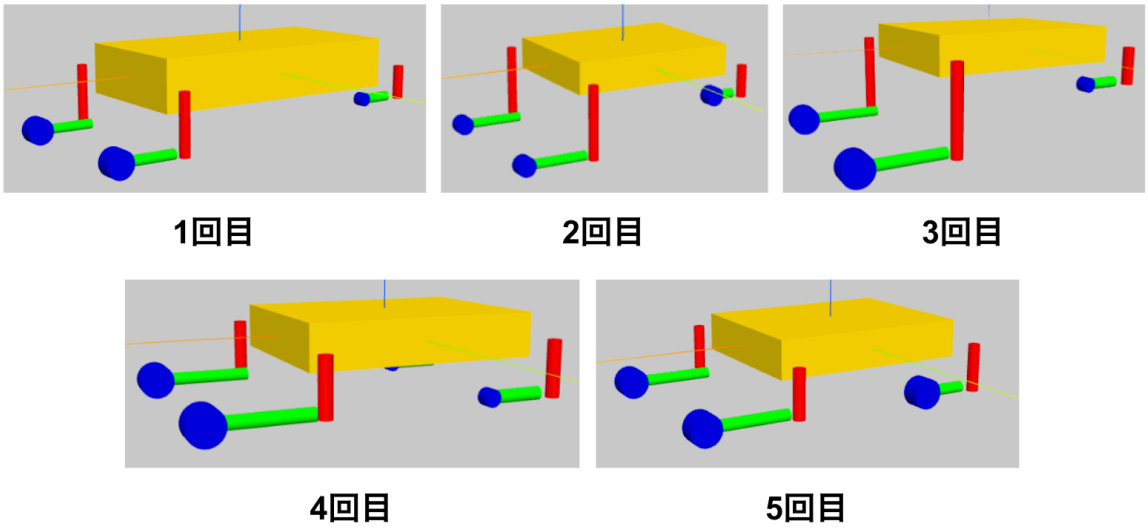

図 7: 被験者 C が選択したパラメータを適用した 4 足歩行ロボットのモデル

表 10: 被験者が選択したパラメータ（1/2）

回数	胴体 (黄) の大きさ	胴体 (黄) の重さ	脚 1,2(赤) の長さ	脚 1,2(緑) の長さ	脚 3,4(赤) の長さ
1 回目	15.5cm × 8cm × 3.0cm	89g	3.5cm	3.5cm	2cm
2 回目	10.5cm × 8cm × 2cm	67g	4.5cm	4cm	2cm
3 回目	10.5cm × 8cm × 2cm	89g	4.5cm	4.5cm	2cm
4 回目	10.5cm × 8cm × 2cm	56g	2.5cm	4.5cm	2.5cm
5 回目	10.5cm × 8cm × 2cm	89g	2.5cm	4.5cm	2.5cm

表 11: 被験者が選択したパラメータ（2/2）

回数	脚 3,4(緑) の長さ	脚 1,2 の円柱 (青) の直径と長さ	脚 1,2 の円柱 (青) の接地具合	脚 3,4 の円柱 (青) の直径と長さ	脚 3,4 の円柱 (青) の接地具合
1 回目	2cm	直径：0.8cm 長さ：1.0cm	剛性値：2000 N/m 摩擦の強さ：0.5	直径：0.4cm 長さ：0.6cm	剛性値：2000 N/m 摩擦の強さ：0.5
2 回目	1.5cm	直径：0.6cm 長さ：0.8cm	剛性値：1000 N/m 摩擦の強さ：1.0	直径：0.6cm 長さ：0.8cm	剛性値：3000 N/m 摩擦の強さ：1.5
3 回目	2cm	直径：0.8cm 長さ：1.0cm	剛性値：2000 N/m 摩擦の強さ：0.5	直径：0.4cm 長さ：0.6cm	剛性値：2000 N/m 摩擦の強さ：0.5
4 回目	2.5cm	直径：0.8cm 長さ：1.0cm	剛性値：2000 N/m 摩擦の強さ：0.5	直径：0.4cm 長さ：0.6cm	剛性値：2000 N/m 摩擦の強さ：0.5
5 回目	2.5cm	直径：0.8cm 長さ：1.0cm	剛性値：3000 N/m 摩擦の強さ：1.5	直径：0.8cm 長さ：1.0cm	剛性値：3000 N/m 摩擦の強さ：1.5

5.2. 被験者の学習結果

以下に被験者が選択したロボットの部位ごとのパラメータによる平均報酬, 損失関数,KL-Divergence の学習遷移と累積の学習結果を示す.

5.2.1 平均報酬

図 8 に被験者の 5 回の選択での平均報酬の推移の一例を示し, 表 12 に各被験者の平均報酬の累積を示す. 図 8 から被験者 C は, 2 回目, 3 回目の改善過程で平均報酬の最大値が減少していることが読み取れる. これは被験者 C が改善過程でのパラメータの調整時に, 適切なパラメータを選択できなかったことを示している.

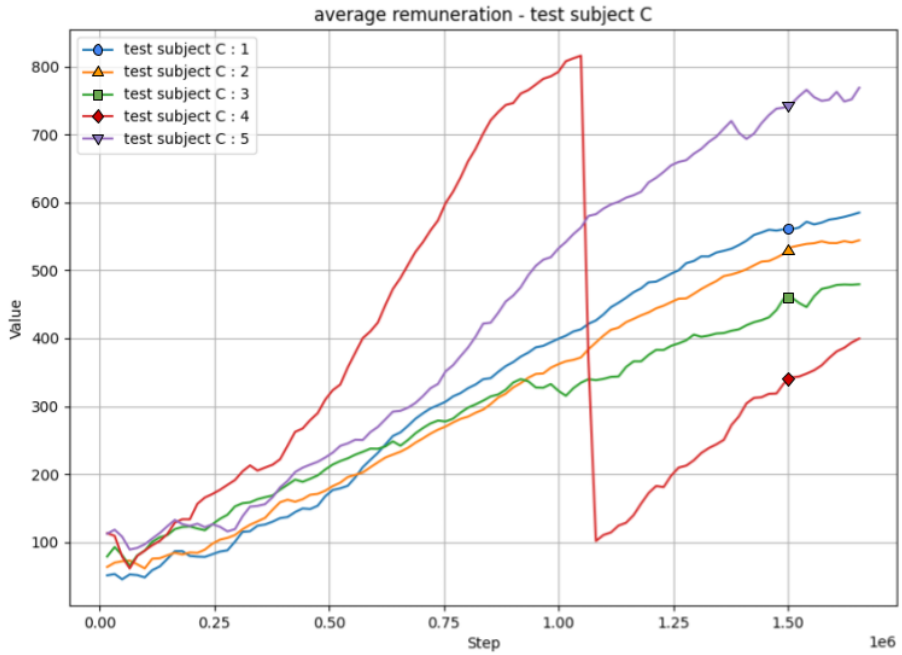

図 8: 被験者 C の平均報酬の推移

表 12: T 各被験者の平均報酬の累積

各被験者	平均報酬の累積
A	1210132.23
B	1485115.57
C	2573922.71
D	3320420.86
E	3795977.39
F	3478294.79

5.2.2 損失関数

図 9 に被験者の 5 回の選択での損失関数の値の推移の一例を示し, 表 13 に各被験者の損失関数の値の累積を示す. 図 9 から学習の後半で損失が増加していることがよみとれる. これは, 学習が収束したため, ランダムな行動をとったことを示している.

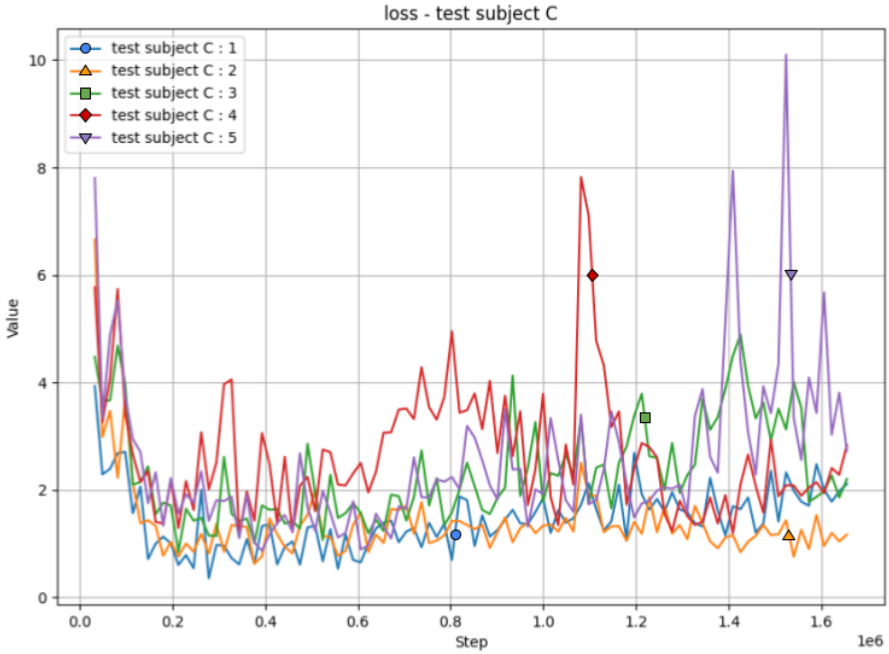

図 9: 被験者 C の損失関数の値の推移

表 13: 各被験者の損失関数の値の累積

各被験者	損失関数の値の累積
A	7297.38
B	9413.64
C	15674.04
D	53023.70
E	32181.58
F	31722.02

5.2.3 KL-Divergence

図 10 に被験者の 5 回の選択での KL-Divergence の値の推移の一例を示し, 表 14 に各被験者の KL-Divergence の値の累積を示す. 図 10 から被験者の 4 回目,5 回目の KL-Divergence の値が大きくなっていることが読み取れる. これは, 方策の更新が急激すぎる可能性があることを示している.

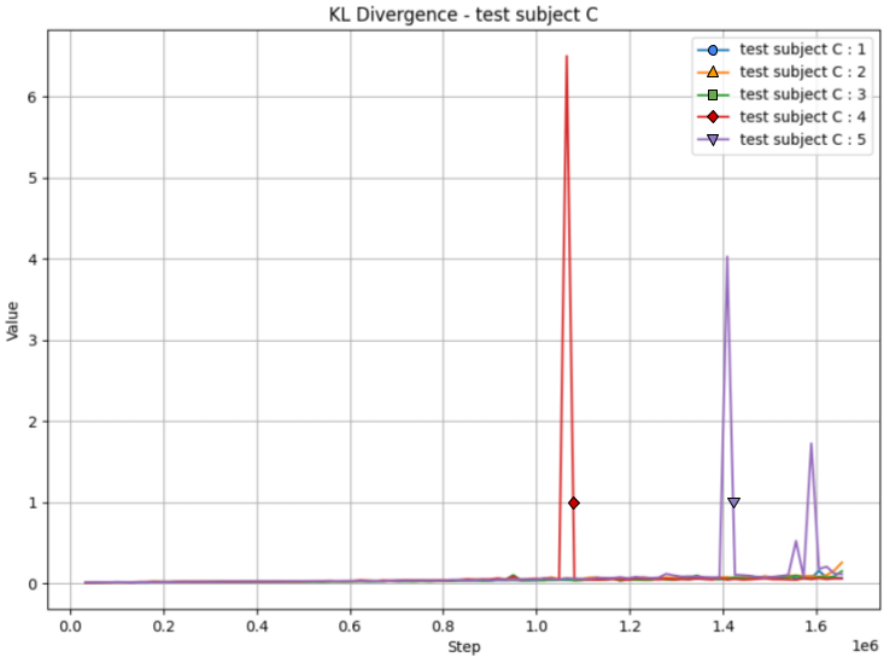

図 10: 被験者 C の KL-Divergence の推移

表 14: 各被験者の KL-Divergence の累積

各被験者	KL-Divergence の累積
A	250.55
B	256.10
C	528.11
D	2559.80
E	4699.97
F	1053.47

6. 考察

ロボットの各部位のパラメータ選択における技術者のセンスを評価するため, ロボット工学を学んだ被験者 A,B をセンスがある技術者と仮定し, 階層クラスタリングによる分類を行った. 特徴量として平均報酬, 損失関数,KL ダイバージェンスの累積値を用いた. 図 11 , 図 12 , 図 13 のデンドログラムから被験者 A,B が近い特徴を持つことが確認された. また, 被験者 E,F も近い特徴を示しており, それぞれは専門技術を学んでいないため, 強化学習の設計過程で技術者のセンスを評価することができたと考えられる.

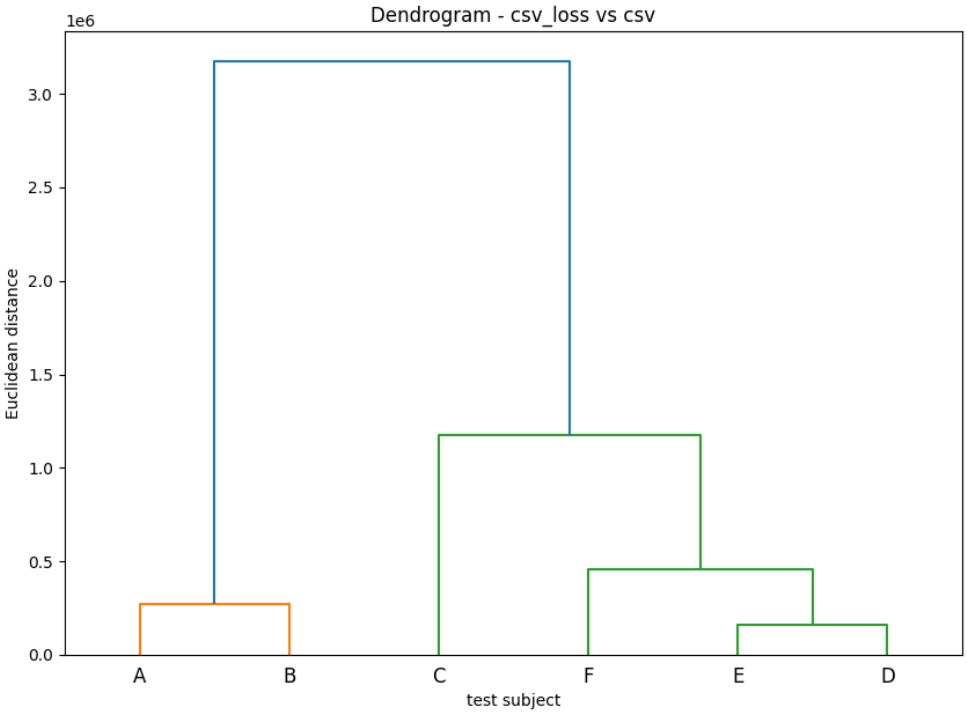

図 11: 平均報酬と損失関数の累積値によるデンドログラム

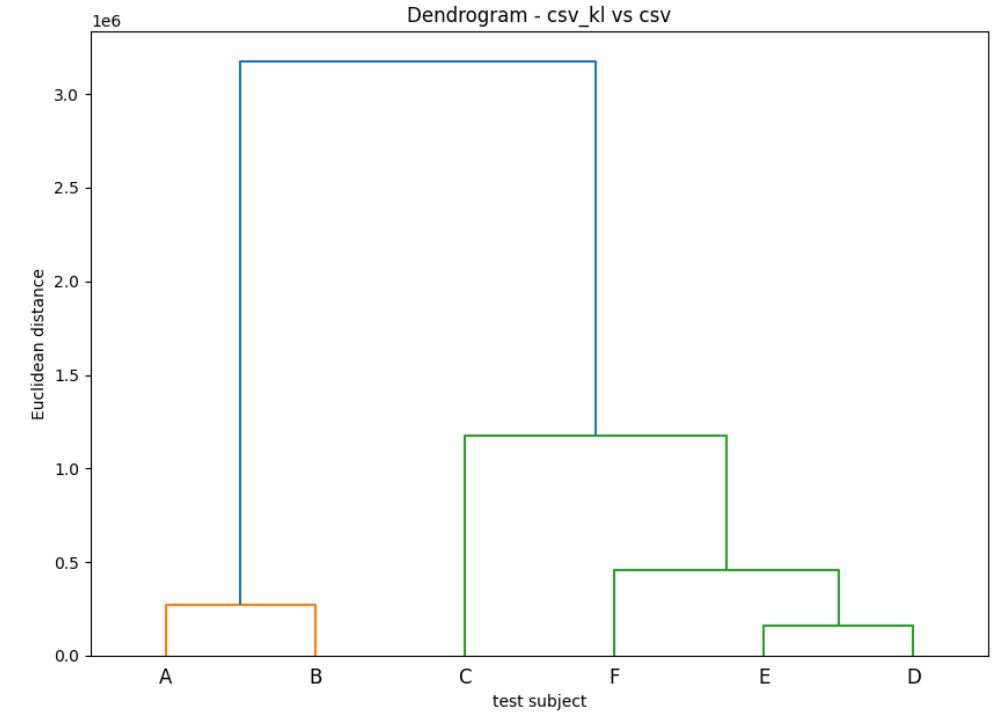

図 12: 平均報酬と KL-Divergence の累積値によるデンドログラム

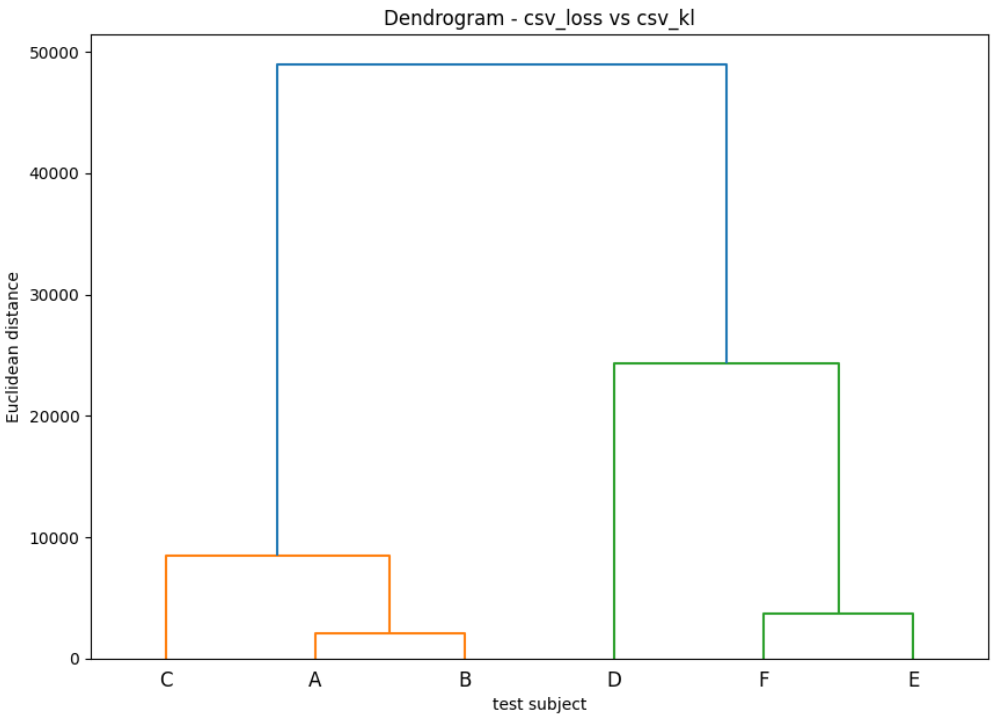

図 13: KL-Divergence と損失関数の累積値によるデンドログラム

7. まとめ

本研究では, 強化学習の設計過程, 特にロボットの部位ごとのパラメータ選択から技術者センスの評価手法を提案した. 平均報酬, 損失関数,KL-Divergence の累積値を通じて解析を行い, 技術者センスが高いと仮定したロボット工学を学んだ教員と学生は共に近い特徴を示し, それらの専門技術を学んだことのない被験者と分離することができた. この結果から技術者センスを平均報酬, 損失関数,KL-Divergence の累積値から技術者と非技術者を分離することが可能である事を示すことができた.

謝辞

本研究を遂行するにあたり, 多大なるご指導を賜りました山本昇志教授に深く感謝いたします. また, 研究内容について適切な助言をくださった専攻科の先輩方にも感謝申し上げます. さらに, 高専 5 年間にわたり, 学業や生活面で温かくご指導・ご鞭撻を賜りました情報通信工学コースの先生方に心より御礼申し上げるとともに, 共に学び, 支え合った学友たちにも深く感謝いたします.

参考文献

[1] 尾崎嘉彦, 野村将寛, and 大西正輝. 機械学習におけるハイパパラメータ最適化手法：概要と特徴. **信学論** *Vol.J103-D*, No.9:615–631, 2020.

[2] 今井翔太. **強化学習の基礎と深層強化学習**. 機械学習プロフェッショナルシリーズ. 講談社, 2020.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 25, pages 1097–1105, 2012.

[4] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1st edition, 1998.

[5] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992.

[6] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.

[7] G. A. Rummery and M. Niranjan. On-line q-learning using connectionist systems. *Technical Report CUED/F-INFENG/TR 166, Cambridge University*, 1994.

[8] Martin L Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994.

[9] Richard Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[10] Nicolas Metropolis and S. Ulam. The monte carlo method. *Journal of the American Statistical Association*, 44(247):335–341, 1949.

[11] Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, 1988.

[12] M. I. Jordan P. R. Kaelbling S. P. Singh, T. Jaakkola. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning*, 38(3):287–308, 2000.

[13] David Silver et al. Volodymyr Mnih, Koray Kavukcuoglu. Playing atari with deep reinforcement learning. *arXiv:1312.5602*, 2013.

[14] Genevieve B. Orr Klaus-Robert Müller Yann LeCun, Leon Bottou. Efficient backprop. *Neural Networks: Tricks of the Trade*, pages 9–50, 1998.

[15] John D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science and Engineering*, 9(3):90–95, 2007.

[16] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. *Proceedings of the 27th International Conference on Machine Learning (ICML)*, 2010.

[17] J. S. Bridle. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. *Neurocomputing*, pages 227–236, 1990.

[18] Yoshua Bengio Patrick Haffner Yann LeCun, Léon Bottou. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[19] Carl Friedrich Gauss. *Theoria Motus Corporum Coelestium in Sectionibus Conicis Solem Ambientium*. Friedrich Perthes und I.H. Besser, 1809.

[20] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, 1948.

[21] Ronald J. Williams David E. Rumelhart, Geoffrey E. Hinton. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[22] David Silver et al. Volodymyr Mnih, Koray Kavukcuoglu. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[23] Prafulla Dhariwal Alec Radford Oleg Klimov John Schulman, Filip Wolski. Proximal policy optimization algorithms. *arXiv:1707.06347*, 2017.

[24] Pieter Abbeel Michael Jordan Philipp Moritz John Schulman, Sergey Levine. Trust region policy optimization. *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 1889–1897, 2015.