Understanding the Behavior of Reinforcement Learning Agents

Jörg Stork, Martin Zaefferer, Thomas Bartz-Beielstein and A. E. Eiben

May 6, 2020

Understanding the Behavior of Reinforcement Learning Agents

Jörg Stork[1], Martin Zaefferer[1], Thomas Bartz-Beielstein[1], and A. E. Eiben[2]

[1] TH Köln `firstname.lastname@th-koeln.de`
[2] Vrije Universiteit Amsterdam `a.e.eiben@vu.nl`

Abstract. Reinforcement Learning (RL) is the process of training agents to solve specific tasks, based on measures of reward. Understanding the behavior of an agent in its environment can be crucial. For instance, if users understand why specific agents fail at a task, they might be able to define better reward functions, to steer the agents' development in the right direction. Understandability also empowers decisions for agent deployment. If we know why the controller of an autonomous car fails or excels in specific traffic situations, we can make better decisions on whether/when to use them in practice. We aim to facilitate the understandability of RL. To that end, we investigate and observe the behavioral space: the set of actions of an agent observed for a set of input states. Consecutively, we develop measures of distance or similarity in that space and analyze how agents compare in their behavior. Moreover, we investigate which states and actions are critical for a task, and determine the correlation between reward and behavior. We utilize two basic RL environments to investigate our measures. The results showcase the high potential of inspecting an agents' behavior and comparing their distance in behavior space.

Keywords: Reinforcement Learning · Behavior · Understandable AI

1 Introduction

In Reinforcement Learning (RL), agents are learning policies to solve a specific task. As an example, we can consider a robot as an agent that has to navigate in a particular environment and react to certain obstacles. A user is in first case interested in the performance of these robots, which is commonly evaluated by their ability to solve the task and further based on a user-defined reward function. Besides this performance assessment, the behavior of the trained robot, e.g., which action it takes for individual states, is the only observable part, as the internals of the policy remains indistinguishable by an external observer. Thus, user desire to analyze and compare the behavior, e.g., to exploit how the robot reacts in certain situations or if it behaves as intended, or not. That is, because even a well-performing robot may have specialized to a particular behavior that was not intended by the user, e.g., he drives only backward.

In this paper, we follow an approach to utilize the behaviors of different agents to span a new space to make comparisons, which is the behavior space.

The primary motivation of this paper is to create a better understanding of this behavior space and develop useful measures for the comparisons of agents without knowing the inner details of their policies. Moreover, these measures could allow to identify how agents in a learning set differ, not concerning their reward, but with regard to their behavior. It is particularly interesting to identify situations (i.e., states), in which an agent behaves differently than expected. As this is a broad topic, we will start by tackling the following research questions:

Q-1 How does the behavior of an agent with high performance compares to similar performing agents or inferior ones?
Q-2 Which input states are important or problematic for the task?
Q-3 Is there a correlation between the reward and behavior of an agent and how do changes in the behavior affect their reward?

Comparing agents in the behavior space has some prerequisites: For most RL environments, individual agents will not visit the entire state space and thus not learn the optimal action for these unobserved states. Unobserved states are particularly present in environments with substantial (i.e., continuous) spaces or exclusive paths. Nevertheless, as we investigate agents that map a policy from state to action space (i.e., neural network policy controllers), we can compute an agent's behavior to any state, even if not observed or even observable by the agent itself. This property allows us to compare two agents in the behavior space on a mutual state set and investigate differences for this set.

However, determining or containing these state sets is difficult, as they usually not initially known or controllable, but based on processing the RL task and discovering them during the learning process of each agent. The contains of the individual state sets for are thus based on the state trajectory each agent follows, e.g., an absolute path for a robot in a maze. Each agent in a learning set will likely perform different trajectories, which renders it in the first case challenging to find a joint input state set to span out a useful behavior space.

Behavior spaces have previously been investigated in the RL literature. Most frequently, they were utilized to measure diversity and enforce explorative search strategies. For instance, Doncieux and Mouret used behavioral similarity measures to encourage the diversity of evolved agents in an evolutionary search [1]. Ollion and Doncieux suggested to measure and enforce exploration in the behavioral space [13]. Meyerson et al. [9] investigated how behavior characterizations can be learned automatically for the purpose of novelty search. Quality diversity algorithms also depend on an effective behavior comparison [14].

Similar directions have been investigated in the field of surrogate model-based optimization. Here, the term *phenotypic* space has been used, defining a space that encompasses behaviors and outcomes of individuals, rather than their encoding (genotype). Distances in the phenotypic space are used to train surrogate models. This has, for instance, been investigated in the context of tree-coded genetic programming [5, 11, 12, 17]. Similar work focused on graph-coded representations of neural networks. Here, phenotypic spaces and distance measure have been investigated for tasks like classification, reinforcement learning, or for evolving neural network controllers for robotic navigation [15, 4].

Unlike these previous investigations, we aim to look at the behavior space not primarily to improve the performance of optimization or modeling algorithms. Instead, we aim for the understandability of agents' behavior. To do so, we utilize two RL environments, a designed maze with different mutual exclusive paths and the inverted pendulum, with a large real-valued state space.

2 Methods

2.1 Behavior Space in Reinforcement Learning

The *behavior* of an RL agent encompasses its (re-)actions, based on its environment and observed input states. The actions an agent takes for a specific state $s \in S$ is defined by a *policy* $\pi : S \rightarrow B$. The agents get a reward $r \in R$ for each state transition. The methods discussed here apply to a wide range of RL agents, the only prerequisite is their ability to calculate a behavior for states that have not necessarily been observed by those agents themselves. More precisely, we define the behavior as the set of actions for a set of input states. For an agent A, we denote its behavior as B_A, with $B_A = \pi_A(\mathbf{S})$ Here, $\mathbf{S}$ is a set containing n input state vectors, $\pi_A(\mathbf{S})$ is the policy function computing the actions of agent A for all states in $\mathbf{S}$. Consequently, the behavior space $\mathcal{B}$ is the set of all possible behaviors (or the behavior of all possible agents) for a specific RL task, that is, $B_A \in \mathcal{B}$.

2.2 Behavior Comparison

For the comparison of two agents A and A', we can calculate the distance of their behaviors, which can then be defined by $\mathrm{d}(B_A, B_{A'})$ Because the distance depends on the state space, we consider the distance of two behaviors with respect to the same state set $\mathbf{S}$. The employed distance function can be chosen according to the data type of $B_A, B_{A'}$. That is, if they contain continuous values we might use the Euclidean distance. If they are ordinal integers we can choose the Manhattan distance instead, with $\mathrm{d}(B_A, B_{A'}) = \sum_i^n |\pi_A(\mathbf{S_i}) - \pi_{A'}(\mathbf{S_i})|$. The comparison of actions for individual states can provide interesting insights into the specific behavior of an agent.

2.3 Reward Behavior Correlation

To understand the benefits of comparisons in the behavior space, the correlation between changes in the reward to changes in behavior is of interest. Therefore, we investigate a set of m agents $\{A_1, ..., A_m\}$, their behaviors $\{B_{A_1}, ..., B_{A_m}\}$, and their accumulated rewards $\{R_{A_1}, ..., R_{A_m}\}$. We compute the Reward Behavior Correlation (RBC) for all pairwise comparisons

$$\mathrm{RBC_{all}} = \mathrm{cor}\Big(\mathrm{d}\big(\{B_{A_1}, ..., B_{A_m}\}\big), \mathrm{d}\big(\{R_{A_1}, ..., R_{A_m}\}\big)\Big).$$

Here, $\mathrm{d}\big(\{B_{A_1}, ..., B_{A_m}\}\big)$ calculates all pairwise distances of the present agents using the behavioral distance $\mathrm{d}(B_{A_i}, B_{A_j})$. Correspondingly, $\mathrm{d}\big(\{R_{A_1}, ..., R_{A_m}\}\big)$ calculates all pairwise distances of the present agents using a distance of their accumulated rewards $\mathrm{d}(R_{A_i}, R_{A_j})$. The correlation $\mathrm{cor}(.,.)$ may be computed rank-based, if desired, or standard linear correlation (Pearson correlation). Similarly to $\mathrm{RBC}_{\mathrm{all}}$, we can also compare each agent to the optimum agent A_{opt} (the agent with largest reward), instead of performing all pairwise comparisons. We denote this as $\mathrm{RBC}_{\mathrm{opt}}$. A large RBC means that small/large differences in reward coincide with small/large differences in behavior. Hence, a large RBC is a good indicator that the behavior space is easier to traverse for search algorithms, and easier to learn for surrogate models.

This property has a close connection to the Fitness Distance Correlation (FDC) used in evolutionary computation to rate problem difficulty [6]. There, differences in fitness are correlated with distances in the search space. $\mathrm{RBC}_{\mathrm{all}}$ considers all pairwise distances while $\mathrm{RBC}_{\mathrm{opt}}$ and FDC consider distances only between candidate solutions and the global optimum (or best-known solution [7]).

2.4 State Importance

The state importance is strongly connected to the task itself. The definition of an important state can be manifold. We looked at several angles:
Important Actions: A state is considered important or problematic, if the correct action for this state is essential for getting a good reward or challenging to learn, e.g., a majority of agents in a learning set fails to learn the correct action.
Unimportant Actions: Depending on the definition of an RL task, specific actions will not affect the reward of an agent, caused either by: First, Degrees of Freedom States (DFS) of an environment, i.e., states with multiple correct actions. Second, Unobserved States (UOS) of an individual agent. Particular environment conditions always lead to the presence of UOS, because they are not reachable by this agent without lowering its reward, e.g., mutual exclusive paths. While the behavior in the UOS is, in the first case, unimportant for the reward of an individual agent, its discovery and analysis could help to design generalizing agents, which learn correct action for all states, instead of only their observed ones. This could be conducted by reshaping the learning environment or the reward function.
Action Reward Rank: For each state, the performed actions can be annotated by the best ranking agent who took this action. This measure could help to identify correct actions and to discover DFS.

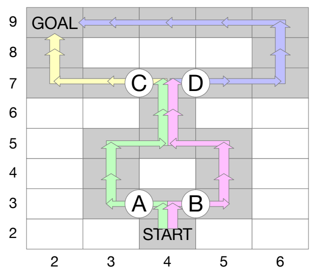
3 Experiments

3.1 Deterministic Maze

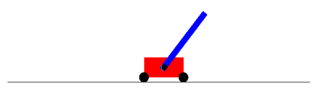
The deterministic maze was designed with *mazelab*[18] as a comprehensible problem where correct actions are known, and behavior is manually rateable. The environment, visualized in Figure 1(a), consists of a 10×7 matrix (shown in figures as 9x6, excluding external walls) with different encoding for accessible ways, walls, the agent and goal. The target is to find the shortest path to the goal. The agent is allowed to take only deterministic actions for each observed agent position in each cardinal direction. Thus they can get stuck against a wall. It gets a small negative reward for each movement, or a higher if running against a wall or moving backward, and a positive reward for reaching the goal. The maximum step size of each agent is fixed to 30, whereas only 11 are needed to follow the shortest path. We designed the maze particular to feature DFS and UOS. It has a total of four paths to the goal and 22 unique agent positions, but these are partly exclusive and successful agent have always UOS. Moreover, the lower fork is a DFS, while the upper one is not. The intention was to construct a problem where agents with the same reward can have different behavior, cause of the forks, and different paths. Moreover, to analyze the effect of different exclusive paths and the UOS on the pairwise behavior comparison.

3.2 Continuous Inverted Pendulum

The inverted pendulum is a time-dependent physics simulator with a continuous input space (Figure 1(b)). The target is to balance the pendulum in the upright position for most time steps, after starting at a random downright position by moving the base car. The environment is evaluated over 500 timesteps but discontinues if the base car moves out of designated limits. The action space was made deterministic for more comprehensible behavior comparisons. The pendulum environment has no exclusive paths, i.e., all states are observable, but agents will have an enormous number of UOS because of the real-valued input space. We also consider the environment to include multiple DFS, e.g., multiple correct behaviors are possible. The environment allows a large number of behaviors, as well as different sized sets of observed states per agent.



(a) Deterministic Maze



(b) Inverted Pendulum

Fig. 1. Environments. For the maze environment, external walls are not displayed. Different ways: A and B are equal in reward, while D is slightly worse than C.

3.3 Generating Reinforcement Learning Agents by Neuroevolution

The RL agents' policies are created and trained by utilizing Neuroevolution by learning ANN policies in an evolutionary process. The underlying algorithm is the graph-based *cartesian genetic programming* `CGP` by A. Turner[3] [8, 16]. For the maze problem, ANNs with 70 inputs and 4 outputs were evolved, where the softmax function computes the resulting action. The pendulum ANN has 6 inputs (5 + 1 bias) and a single output. For an output value of > 0.5, the action is drive left, otherwise, drive right. The ANNs are evolved in connection weights and structure, i.e., the number and placing of connections, nodes, and transfer functions. The maximum number of nodes and connections for each ANN was set for both problems to 100 and 1000, respectively. This leads to a vast amount of different ANN topologies. The inner workings of the ANNs are complex and very difficult to compare[3, 15]. Thus only the reward and the behavior of the agents using these ANNs are considered observable.

3.4 Experimental Setup for Analyzing the Behavior Measures

Behavior Comparison: First, explorative data analysis is conducted to analyze the behaviors and visualize them in the environment. We analyze the behavior for individual input states particular for the maze problem, as we can manually identify wrong actions and understand their impact on the reward. Further, we use the one-to-one comparison of agents with similar rewards to illustrate their actions to see the influence of DFS and UOS. For the pendulum problem, we analyze and reveal different behavior based on specific inputs and compare the influence of using different state sets as input for behavior computation.
State Importance: The maze environment has certain DFS and UOS, i.e., the forks with different importance and different exclusive paths to reach the goal. The goal of the importance analysis is to discover these states by comparing the behavior of all agents. We take a best-ranked agent as the reference for correct actions and calculate the the percentage difference of actions for each state by one-to-many comparisons, weighted by the difference in rank for these agents, by $\mathrm{d}(B_A, B_{A'}) \times \mathrm{d}(rank_A, rank_{A'})/sum(rank_A, rank_{A'})$. Further, we calculate and visualize the action reward rank (Section 2.4) for each state.
Reward Behavior Correlation: The main challenge in computing the RBC is the selection of a suitable state set to compare the behavior. With the experience from the analysis of the previous experiments, we defined different options to select a suitable state set and analyze which of them leads to the best overall RBC.

4 Results and Discussion

4.1 Neuroevolution

Table 1 summarizes the parameters and outcomes of the experiments. The pendulum agents' rewards were aggregated over 30 different instances for reducing

[3] http://www.cgplibrary.co.uk - accessed: 2018-01-12

the impact of the random start positions; all states and actions from these instances are included in the agents' state sets. The agents of each environment were merged into one data set. Agents with equal state-input sets (i.e., those following precisely the same path) were filtered to acquire a feasible sized data set. Due to the small number of input states for the maze environment, its amount of agents is significantly reduced. Conversely, the majority of trajectories in the pendulum experiment is already unique. The cleaned-up data for each environment consist of all unique agents and input states they observed, the corresponding actions, and their rewards. The agents were ranked, where equal performance leads to a shared rank. The maze problem has two best-ranked (rank 1) agents. In the following, we arbitrarily chose one of these two as a reference agent (denoted as "best agent" where relevant).

4.2 Behavior Comparison

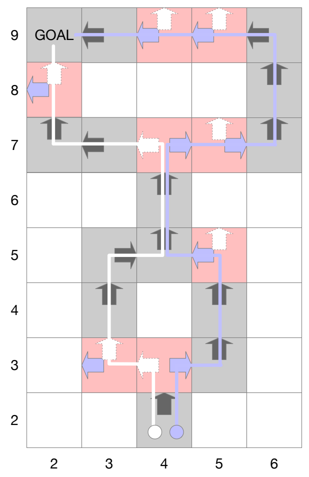
The comparison of the best agent against selected inferior agents for the maze environment on the complete state set is illustrated in Figure 2(a). Interestingly, the best agent does not choose the best action in all states. It only chooses correctly for the states it observed by itself. The agent would run into walls if placed in certain positions (e.g., 5,5 or 4,9). As expected, the comparison of two agents reaching the goal on different trajectories leads to differences in the forking states, cell (4,3), and cell (4,7). Moreover, the behavior distance is amplified by different actions for states that were not observed by both compared agents. Thus, the behavior distance is rather large (d=8).

Another remarkable observation is shown in Figure 2(b), for a comparison between the best agent and a medium-rank agent (rank 10). Despite the significant difference in rank, they have a behavior distance of d=1. Their behavior is nearly identical. Finally, Figure 2(c) shows the distance to an agent with the worst possible reward, which also has a considerable behavior distance (d=16).

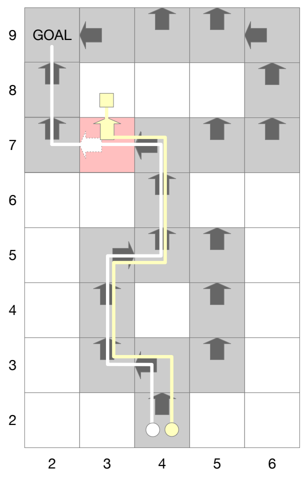
The number of acquired states for the pendulum is enormous and not suitable for complete comparisons as we visualized for the maze. However, we computed behavior differences of smaller state subsets and visualized them using a selected input, the *angular speed* of the pendulum, which is nearly zero if the pendulum is balanced in the upright position. Figure 3(a) shows the behavior of the best agent against the rank 1000 (of 3648) agent by calculating it on best, as well as on rank 2.000 (b) and rank 3500 (c) input sets. We can identify in Figure 3(a), that particular for time-steps 250-300 and 750-800, the rank 1000 agent behaves consequently different. These time-steps illustrate a situation of a falling

Table 1. Parameters and Results of the Neuroevolution Run for both Environments

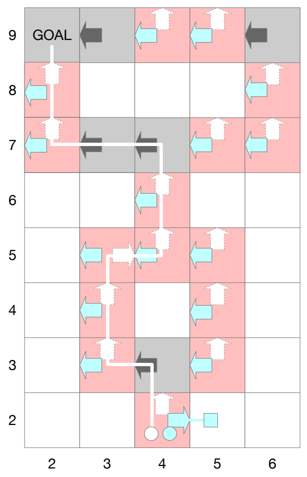
	maze	pendulum		maze	pendulum
repeated runs	12	1	total agents	48e3	4020
evaluations per run	4020	4020 × 30	unique agents	43	3648
observed states per agent	30	max 15e3	unique states	22	30e6



(a) Rank 1 vs Rank 3, d=8



(b) Rank 1 vs Rank 10, d=1



(c) Rank 1 vs Worst, d=16

Fig. 2. One-to-one comparison of the best agent against other agents of different rank ranked agents. Grey cells: agents act the same. Red cells: agents act differently. White lines and arrows: best agent trajectory and actions. Yellow/Blue lines and arrows= comparison trajectory and actions. a) The 3rd rank agent reaches the goal but uses a different trajectory than the best agent b) The agent of rank ten runs against a wall at cell (3,7), but has the same actions as the best agent otherwise. c) The worst agent runs against a wall immediately at cell (4,2) but would go left otherwise.

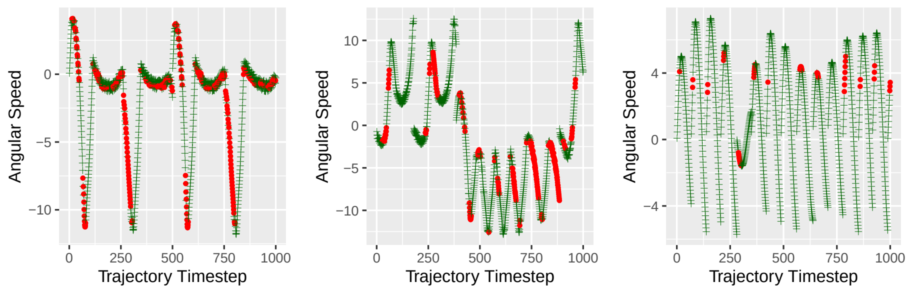
pendulum, shortly after it was balanced. While the best agent countersteers this movement, the rank 1000 agent accelerates it. Consequently, we were able to identify a situation in which the lower-ranked agent fails to learn the correct actions. However, as the actions are based on all inputs and the angular speed is just one of them, finding these situations manually remains challenging. Figure 3(c) shows what happens if the behavior of two agents is compared on the input set of a distant ranked agent. The behavior of the rank 3500 is hugely different: Each recorded trajectory is only some time steps long, presumably caused by the agent fast driving the base car against the horizontal limit, which leads to termination. For such extreme situations, both compared agents (best vs. rank 1000) seem to behave similarly. Conversely, their difference in reward seems to be related to smaller differences in critical situations. We can summarize these observations to identify some properties of the behavior space:

I) Small changes in the behavior can lead to large changes in the reward.

II) Agents with the same reward can have a considerable behavior distance, mainly caused by DFS and UOS present in the compared state set.

III) The input state set has a huge impact on the behavioral distance comparison.

These observations reflect a central challenge of behavioral comparisons: Finding important states and a suitable state set for conducting behavior comparisons. We argue that comparing the behavior on sets with DFS and UOS can



(a) Trajectory from best, total difference is 33%

(b) Trajectory rank 2000, total difference is 24%

(c) Trajectory rank 3500, total difference is 4%

Fig. 3. Behavior comparison for the pendulum. The behavior difference from best versus rank 1000 is shown (Green cross= same action, red dot= different action) for the *angular speed* value over the first 1000 states of state sets from best, rank 2000, and rank 3500. As visible, the behavior difference is influenced by the state sets. Particularly, the dissimilarity in (c) is smaller.

help to distinguish between agents of similar reward, but is presumably overestimating their behavior distance or leads to significant variances due to random actions. Moreover, comparing agents on state sets of other agents, even without considering the influence of UOS, might not reveal useful behavior distances, as these states represent situations not suitable for telling apart good behavior.

4.3 State Importance

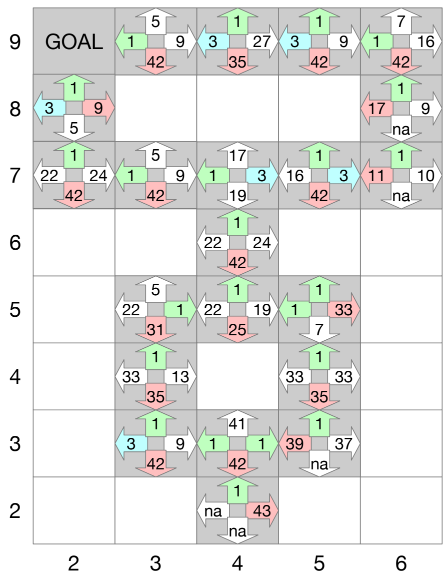
For the state importance, we illustrate the percentage of agents with behavior differing from the best agent for each state, weighted by their differences in rank. Figure 4(a) shows this statistic only for agents reaching the goal, while Figure 4(b) concerns all agents. The states with a high value are considered to be more important, as the majority of agents behave dissimilarly to the presumed ‘correct‘ action with a high impact on the reward rank. For the comparison between the best agents, many states show no importance, i.e., similar behavior in this set. The maze design was designed, so the importance of the DFS fork in (4,3), should be less than of the fork in (4,7), which is no DFS. This is represented by our importance measure, as (4,7) has double as high value, visible in both Figure 4(a) and 4(b). However, the importance measure also provides other states with a high importance value, particularly visible in the upper part of the maze. This can be explained by the type of behavior comparison (all states) and the influence of UOS for each agent. Agents can have ‘wrong’ behavior for these states, even if they can solve the environment. This is observable in Figure 4(c), where for each state, the best agent choosing a certain action is shown. While for the state (4,3) and also for (4,7), we see a correct identification of different ways, (5,5), (4,9) and (5,9) give the wrong idea of correct actions, as the supposedly best-outlined action is surprisingly to run against a wall. This effect of UOS is amplified if

	2	3	4	5	6
9	GOAL	0	100	100	0
8	100				0
7	0	0	100	100	0
6			0		
5		0	0	100	
4		0		0	
3		50	50	0	
2			0		

(a) State Importance Rank 1-4

	2	3	4	5	6
9	GOAL	85	33	63	59
8	75				44
7	39	85	81	74	61
6			47		
5		59	47	68	
4		37		20	
3		58	56	14	
2			3		

(b) State Importance all Ranks



(c) Ranked Actions per State

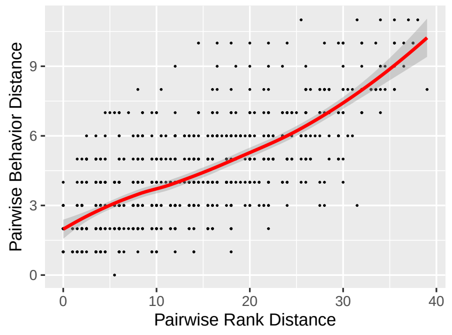
Fig. 4. (a) State importance calculated using either the best agents or (b) all agents. Higher values depict higher importance, colored by value quarters for easier comparison. (c) Action reward rank. Shows the best rank choosing each action for each state. Green=rank 1, blue=rank 3, red= worst action rank. The two rank 1 agents choose different actions in (4,3) and (5,5). (4,3) is DFS, and (5,5) an UOS for the best agent.

all agents are considered. For example, the lowest-ranked agent runs directly against a wall. However, we compute and compare its behavior (intensified by its low rank) on all states. We assume that if we compare a large set of agents, the UOS, with their presumably random actions, do not affect the importance as strongly. The shown importance is thus presumably higher in the states of the upper part of the maze, as only a minority of agents reach this part of the maze. For the rest, we are just comparing their behavior on UOS. Thus, our importance measures could also help to identify states/parts of an environment which are rarely reached by any agent in a set.

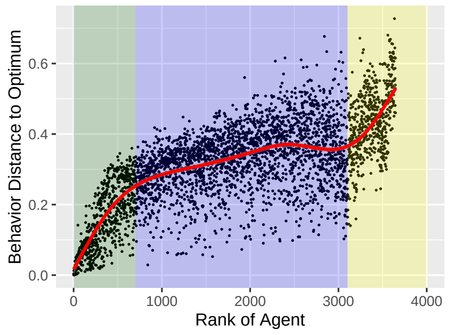
4.4 Reward Behavior Correlation

For the analysis of the RBC, the previous results have shown that it is essential to choose a suitable state set for each pairwise comparison. We thus compared several state sets to conduct each behavior comparisons between two agents:

- Input set A: Random states sampled from all known states of all agents.
- Input set B: All known states of an environment.
- Input set C: The observed states of the best agent.
- Input set D: The observed states of both compared agents.



(a) Pairwise RBC_{all} for maze, Correlation=0.72, Input set E



(b) Pairwise RBC_{opt} for pendulum, Correlation=0.70, Input set D

Fig. 5. (a) $\mathrm{RBC}_{\mathrm{all}}$ plot for the maze environment (b) $\mathrm{RBC}_{\mathrm{opt}}$ plot for the pendulum problem. Computed on input set E and D, respectively. A high correlation is visible.

– Input set E: The observed states of one compared agent.

The results are displayed in Figure 5 and Table 2. For the pendulum problem, we calculated the $\mathrm{RBC}_{\mathrm{all}}$ on a subset of agents (equidistant sampled) to significantly reduce the otherwise infeasible computation time. Figure 5(a) and 5(b) shows the resulting $\mathrm{RBC}_{\mathrm{all}}$ and $\mathrm{RBC}_{\mathrm{opt}}$ values, respectively. For both, the overall correlation is strongly positive. In 5(b) it differs between good agents (rank 1-800), medium agents (800-3100), and poor agents (3100-3600). The other input sets (A, B, C, and E) lead to an inferior $\mathrm{RBC}_{\mathrm{opt}}$ for both problems. Moreover, set D and E lead to the highest RBC_{all}, with a very significant difference for the maze problem. We assume, that the best correlation would be achieved if agents are compared in their mutual behavior space. The presumed cause for the higher RBC_{all} is the reduction of the influence of unobserved states in the comparison. Particular, the maze environment agents have less UOS where they likely act random, if set D or E are considered. Including UOS in a comparison does thus not lead to a more detailed behavior distance, but one with higher overall variance, thus leading to an inferior RBC. This is visible for the pendulum, which for all sets has a large amount of UOS due to the continuous state space, which leads to smaller difference in the variants to compute the $\mathrm{RBC}_{\mathrm{all}}$. The overall positive $\mathrm{RBC}_{\mathrm{opt}}$ and $\mathrm{RBC}_{\mathrm{all}}$ outline the high potential of agent comparisons in the behavior space to improve the search for good ranking agents.

Table 2. $\mathrm{RBC}_{\mathrm{all}}$ of all agents for different input sets

environment	A) random	B) all	C) reference	D) both observed	E) one observed
maze	0.27	0.29	0.28	0.62	**0.72**
pendulum	0.36	na	0.34	**0.45**	0.36

5 Conclusion

In this work, we investigated properties of the behavior space of RL agents and how this space can help to compare agents in learning sets to gain valuable insights. Regarding our research questions, we can conclude for Q-1, that even small changes in the behavior can have considerable effects on the reward. At the same time, agents achieving the same reward can show quite different behavior. We believe that focusing only on the reward of an agent might not be the optimal choice. Rather, the agents' behavior can give valuable insights on how agents achieve that reward. This can reveal agents with surprising behavior or help to improve the learning process. For instance, reward functions can be designed to enforce or suppress specific behavior.

The analysis of Q-2 has shown that accessing the variable importance is challenging and highly dependent on the underlying set of agents and the environment. These challenges are mainly caused by comparing an agent on states, which were not observed by it, or are even not observable by this agent due to environment restrictions, e.g., mutually exclusive paths. For these cases, the behavior of any agent can be random, even for the ones with the best reward. A comparison of behavior on these states might deliver misleading results. Only if states were observed by multiple agents, we can access their true importance.

This finding is further stressed when considering Q-3. The RBC is highest if we consider pairwise behavior comparisons on those states that have been observed by both compared agents. The high positive RBC shows that the behavior space is a promising concept. We suggest that searching in that space may be beneficial. For future work, we aim to take a close look at how the understanding of behavioral spaces can be exploited:

Reward measures: Ideally, reward measures help to steer the search into desirable areas of the search space. Understanding which states are critical to receiving a good match between behavior and reward may help to design better reward measures. The importance of developing good reward measures for RL is stressed in a review by Doncieux and Mouret [2].

Search in behavior space: The usage of the behavior distance of agents as an additional search criterion seems very attractive. It can be used to preserve diversity in evolutionary search procedures [1]. Further, the search for a specific behavior may be of interest, independent or in addition to reward-driven search, e.g., by modeling the reward to behavior space with surrogate models. An example application would be *inverse reinforcement learning* [10]. The search in behavior space allows the use of completely different agent topologies or even comparing agents trained by different algorithms.

Search operators: Finally, a good understanding of the latent, behavioral space may help to define better search operators. For instance, search operators could be designed to search directly in the behavior space, rather than the policy or topology space.

References

1. S. Doncieux and J. Mouret. Behavioral diversity measures for evolutionary robotics. In *IEEE Congress on Evolutionary Computation*, pages 1–8, 2010.
2. S. Doncieux and J.-B. Mouret. Beyond black-box optimization: a review of selective pressures for evolutionary robotics. *Evolutionary Intelligence*, 7(2):71–93, jul 2014.
3. A. Gaier, A. Asteroth, and J.-B. Mouret. Data-efficient neuroevolution with kernel-based surrogate models. In *Genetic and Evolutionary Computation Conference (GECCO)*, 2018.
4. A. Hagg, M. Zaefferer, J. Stork, and A. Gaier. Prediction of neural network performance by phenotypic modeling. In M. López-Ibáñez, editor, *Proceedings of the Genetic and Evolutionary Computation Conference Companion - GECCO'19*, GECCO '19, pages 1576–1582, Prague, Czech Republic, 2019. ACM.
5. T. Hildebrandt and J. Branke. On using surrogates with genetic programming. *Evolutionary Computation*, 23(3):343–367, June 2015.
6. T. Jones and S. Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In L. J. Eshelman, editor, *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 184–192, Pittsburgh, PA, USA, July 1995. Morgan Kaufmann.
7. L. Kallel and M. Schoenauer. Fitness distance correlation for variable length representations. Technical Report 363, CMAP, Ecole Polytechnique, 1996.
8. M. M. Khan, G. M. Khan, and J. F. Miller. Evolution of neural networks using cartesian genetic programming. In *IEEE Congress on Evolutionary Computation*, pages 1–8, July 2010.
9. E. Meyerson, J. Lehman, and R. Miikkulainen. Learning behavior characterizations for novelty search. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, GECCO '16, page 149–156, New York, NY, USA, 2016. Association for Computing Machinery.
10. A. Y. Ng, S. J. Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, pages 663–670, 2000.
11. S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan. Selection schemes in surrogate-assisted genetic programming for job shop scheduling. In G. Dick, W. N. Browne, P. Whigham, M. Zhang, L. T. Bui, H. Ishibuchi, Y. Jin, X. Li, Y. Shi, P. Singh, K. C. Tan, and K. Tang, editors, *Simulated Evolution and Learning: 10th International Conference, SEAL 2014*, volume 8886 of *Lecture Notes in Computer Science*, pages 656–667, Dunedin, New Zealand, 2014. Springer.
12. S. Nguyen, M. Zhang, and K. C. Tan. Surrogate-assisted genetic programming with simplified models for automated design of dispatching rules. *IEEE Transactions on Cybernetics*, 47(9):2951–2965, May 2016.
13. C. Ollion and S. Doncieux. Why and how to measure exploration in behavioral space. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO '11, page 267–274, New York, NY, USA, 2011. Association for Computing Machinery.
14. J. K. Pugh, L. B. Soros, and K. O. Stanley. Searching for quality diversity when diversity is unaligned with quality. In *International Conference on Parallel Problem Solving from Nature*, pages 880–889. Springer, 2016.
15. J. Stork, M. Zaefferer, T. Bartz-Beielstein, and A. E. Eiben. Surrogate models for enhancing the efficiency of neuroevolution in reinforcement learning. In M. López-Ibáñez, editor, *Proceedings of the Genetic and Evolutionary Computation Conference - GECCO'19*, GECCO '19, pages 934–942, Prague, Czech Republic, 2019. ACM.

16. A. J. Turner and J. F. Miller. Cartesian genetic programming encoded artificial neural networks: a comparison using three benchmarks. In *Proc. GECCO'13*, pages 1005–1012. ACM, 2013.
17. M. Zaefferer, J. Stork, O. Flasch, and T. Bartz-Beielstein. Linear combination of distance measures for surrogate models in genetic programming. In A. Auger, C. M. Fonseca, N. Lourenço, P. Machado, L. Paquete, and D. Whitley, editors, *Parallel Problem Solving from Nature – PPSN XV: 15th International Conference*, volume 11102 of *Lecture Notes in Computer Science*, pages 220–231, Coimbra, Portugal, Sept. 2018. Springer.
18. X. Zuo. mazelab: A customizable framework to create maze and gridworld environments. https://github.com/zuoxingdong/mazelab, 2018.