UPRIGHT PUBLICATIONS
https://upright.pub/index.php/ijrstp/

Original Contribution

Reward Redistribution as Align-RUDDER: Learning from a Few Demonstrations

Takudzwa Fadziso[1]

Keywords: Reward Redistribution, Align-RUDDER, Learning from Demonstrations

International Journal of Reciprocal Symmetry and Theoretical Physics

Vol. 7, Issue 1, 2020 [Pages 1-8]

Reinforcement to handle difficult tasks with sparse and delayed rewards, learning algorithms demand a large number of samples. Complex tasks are frequently broken down into sub-tasks in a hierarchical manner. A step in the Q-function corresponds to the completion of a sub-task in which the return expectation rises. RUDDER was created to identify these phases and then shift rewards to them, resulting in rapid rewards when sub-tasks are completed. Learning is significantly accelerated since the problem of delayed rewards is alleviated. Current exploration strategies, such as those used in RUDDER, struggle to find episodes with large rewards when dealing with difficult tasks. As a result, we presume that high-reward episodes are presented as demonstrations and do not need to be found through exploration. The number of demonstrations is typically low, and RUDDER's LSTM model does not learn effectively as a deep learning method. As a result, we present Align-RUDDER, which is RUDDER with two major changes. First, Align-RUDDER implies that high-reward episodes are presented as demos, replacing RUDDER's safe exploration and lesson replay buffer. Second, we substitute RUDDER's LSTM model with a profile model derived from multiple demonstration sequence alignment. Bioinformatics has shown that profile models may be built with as little as two demos. Align-RUDDER inherits the concept of reward redistribution, which lowers the time between incentives and hence accelerates learning. On complex artificial tasks with delayed rewards and limited demonstrations, Align-RUDDER surpasses competitors. Align-RUDDER can mine a diamond on the MineCraft obtain Diamond assignment, but only infrequently.

INTRODUCTION

An overview of Align-RUDDER1, our new approach. Complex tasks with sparse and delayed rewards are difficult for reinforcement learning systems to learn (Sutton and Barto, 2018; Rahmandad et al., 2009; Luoma et al., 2017). RUDDER (Arjona-Medina et al., 2019) has demonstrated its ability to learn sparse and delayed rewards. RUDDER requires high-reward episodes in order to store them in its lessons replay buffer. Current exploration strategies, on the other hand, struggle to find episodes with large rewards for complex activities.

Teachers, role models, and prototypes provide humans and animals with high reward events. We suppose that high-reward events are used as demonstrations in this setting. As a result, the demos can take the role of RUDDER's safe exploration and lesson replay buffer. Demonstrating is time-consuming for humans and time-consuming for automated exploration tactics, thus there are usually only a handful available. RUDDER's LSTM, on the other hand, as a deep learning method, necessitates a large number of samples. As a result, we substitute a profile model derived from multiple sequence alignment of the demonstrations for

[1]Institute of Lifelong Learning and Development Studies, Chinhoyi University of Technology, ZIMBABWE
takudzwafadziso@gmail.com

RUDDER's LSTM. In bioinformatics, profile models are used to score novel sequences based on their sequence similarity to the matched sequences (Ahmed, 2021; Ganapathy et al., 2021; Manojkumar et al., 2021; Sharma et al., 2021; Hussain et al., 2021; Raya et al., 2021; Panchal et al., 2021; Bynagari & Ahmed, 2021). The LSTM in RUDDER forecasts an episode's return based on an action-state sub-sequence.

Objectives of the Study

This study is aimed at the LSTM prediction which is substituted with the score of this sub-sequence if it is aligned to the profile model in the new approach (Align-RUDDER) adopted in this study. The LSTM predictions was used for return decomposition and reward redistribution in the RUDDER implementation by exploiting the difference of consecutive forecasts. Align-Rudder uses the difference in alignment scores for subsequent sub-sequences to perform return decomposition and reward redistribution.

LITERATURE REVIEW

Temporal difference and Monte Carlo versus Align-RUDDER

As demos, we anticipate having high reward episodes. Through alignment approaches, Align-RUDDER leverages these episodes to discover state-actions indicative of high rewards. Align-RUDDER then redistributes incentives to these state-actions, allowing a policy to be adjusted so that these state-actions are achieved more frequently. As a result, the return on investment is higher, and relevant episodes are sampled more frequently. For delayed rewards and model-free reinforcement learning, (I) temporal difference (TD) suffers from vanishing information, even with eligibility traces (Arjona-Medina et al., 2019); (II) Monte Carlo (MC) must average over all possible futures, resulting in high variance (Arjona-Medina et al., 2019). (Arjona-Medina et al., 2019). Model-based approaches such as Monte-Carlo Tree Search (MCTS) can handle delayed and unusual rewards if models are available, as they are for GO and chess (Silver et al., 2016; 2017).

Q-functions (action-value functions) are step functions, as a basic understanding

A hierarchical structure made up of sub-tasks or sub-goals distinguishes complex tasks (see first row in left panel of Figure 1). An optimal policy's Q-function resembles a step function. A change in return expectation, or the expected quantity of the return or the chance of obtaining the return, is a step in the Q-function. Achievements, failures, completing sub-tasks, reaching a sub-goal, or changes in the environment are all indicated by steps. Identifying large steps is critical for significantly speeding up learning: understanding the large steps in the Q-function helps you to boost the return and sample more relevant episodes.

Every state-action pair's expected return must be predicted by a Q-function. As a result, it's likely to produce a prediction error at some time, which could obstruct learning (see second row in left panel of Figure 1). It is not essential to anticipate the expected return for each state-action pair because the Q-function is largely constant. It's enough to find important state-actions throughout the episode and utilize them to forecast the predicted return (see third row in left panel of Figure 1). The LSTM network (Hochreiter and Schmidhuber, 1995; 1997a, b) is ideal for storing only relevant state-actions in its memory cells. It only changes them when a new relevant state-action combination arises, thus its output remains constant until the memory cells are refreshed. The core concept that Q-functions are step functions motivates return decomposition and reward redistribution to detect these phases and speed up learning.

Redistribution of Rewards: Decomposition of Ideas and Returns

Given an episodic Markov decision process, we consider reward redistributions produced using return decomposition (MDP). It is assumed that the Q-function is a step function (blue curve in first row in right panel of Figure 1). The steps of the Q-function are identified using return decomposition (green arrows in right panel Figure 1). Given the state-action sub-sequence, a function predicts the predicted return (large red arrow in first row in right panel of Figure 1) (LSTM in RUDDER, alignment model in Align-RUDDER). The prediction is broken down into individual Q-function steps (green arrows in Figure 1). The steps are removed by the redistributed rewards (little red arrows in the second and third rows of the right panel of Figure 1). As a result, the anticipated future reward is zero (blue curve at zero in last row in right panel of Figure 1). Because delayed incentives are no longer existent, understanding the Q-values simplifies calculating the expected immediate rewards (little red arrows in right panel of Figure 1) because future rewards are zero.

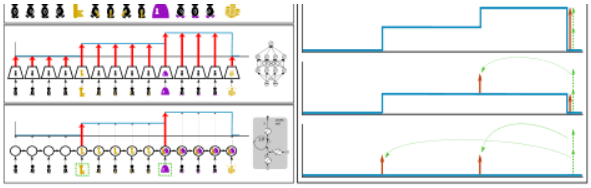

Figure 1: Basic insight of RUDDER (left panel) and reward redistribution (right panel)

Multiple sequence alignment is used to redistribute rewards

For reward redistribution via return decomposition, RUDDER employs an LSTM model. The difference between two successive LSTM model predictions is the reward redistribution. If a state-action pair improves the return forecast, it is rewarded right away. The redistributed reward is $R_{t+1} = g((s; a)_{0:t}) g((s; a)_{0:t-1})$ using state-action sub-sequences $(s; a)_{0:t} = (s_0; a_0; : : : ; s_t; a_t)$, where g is an LSTM model that anticipates the return of the episode. Because the greatest steps of the Q-function lower the prediction error the most, the LSTM model learns to mimic them first. As a result, the LSTM model retrieves the relevant state-action pairs first (events). We now use sequence alignment approaches to replace the LSTM model with a profile model. The profile model is the outcome of several demonstration sequence alignment and allows fresh sequences to be aligned to it. Both the $(s; a)_{0:t-1}$ and $(s; a)_{0:t}$ sub-sequences are translated to event sequences and then aligned to the profile model. As a result, both sequences are assigned a score S that is proportional to the function g. $R_{t+1} = g((s; a)_{0:t}) g((s; a)_{0:t-1})$ is the redistributed reward. Figure 2 shows a biological sequence alignment on the left panel and a demonstrative alignment on the right panel.

Reward redistribution

RUDDER (Arjona-Medina et al., 2019) introduced the notions of reward redistribution and return decomposition, which are also essential principles in our Align-RUDDER. Return decomposition-based reward redistribution removes – or at least reduces – award delays while maintaining the same optimal rules. When employing multiple sequence alignment to develop a reward redistribution model, Align-RUDDER is justified by the theories of return decomposition and reward redistribution. The principles and theory of return decomposition and reward redistribution are briefly review.

We consider a finite MDP defined by the 5-tuple $(\mathcal{P} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \gamma)$ where $\mathcal{S}$ and $\mathcal{A}$ are sets of finite states and actions, respectively, and $\mathcal{R}$ is the set of bounded rewards *r*. The associated random variables for a given time step *t* are S_t, A_t, and R_{t+1}. $\mathcal{P}$ also has transition-reward distributions $p(S_{t+1} = s', R_{t+1} = r \,|S_t = s, A_t = a)$, as well as a discount factor $\gamma \epsilon (0; 1]$, which we set to $\gamma = 1$. A Markov policy $\pi(a|s)$ is the probability of taking an action in response to a state s. MDPs having a finite temporal horizon or MDPs with an absorbing state are considered. $G_t = \sum_{k=0}^{T-t} \gamma^k R_{t+k+1}$ is the discounted return of a sequence of length T at time t. The Q-function for a given policy is $q^\pi(s, a) = E_\pi [G_t \,|S_t = s, A_t = a]$, as is customary. The expectation of x is $E_\pi [x \,|s, a]$, where the random variable is a sequence of states, actions, and rewards created using the transition-reward distribution p, policy π , and starting at (s; a). At t = 0, the goal is to design an optimal policy $\pi^* = argmax_\pi E_\pi [G_o]$ that maximizes the expected return. In order to provide stationary optimal policies, we assume that the states s are time-aware (time t can be derived from each state). A deterministic optimal policy π^* exists, according to Proposition 4.4.3 in (Puterman, 2005).

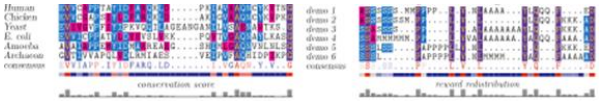

Figure 2: Left panel: Alignment of biological sequences (triosephosphate isomerase) giving a conservation score. Right panel: Alignment of demonstrations using the conservation score for reward redistribution

A sequence-Markov decision process (SDP) is a decision process with Markov transition probabilities but no requirement for a Markov reward probability. Return-equivalent SDPs $\check{\mathcal{P}}$ *and* $\mathcal{P}$ with varying reward probabilities have the same expected return at t = 0 for each policy д, while strictly return-equivalent SDPs have the same expected return for every episode. Return-equivalent SDPs have the same optimal policies since the expected return at t = 0 is the same. A sequence-Markov decision process (SDP) is a decision process with Markov transition probabilities but no requirement for a Markov reward probability. Return-equivalent SDPs $\check{\mathcal{P}}$ *and* $\mathcal{P}$ with varying reward probabilities have the same expected return at t = 0 for each policy, while strictly return-equivalent SDPs have the same expected return for every episode. Return-equivalent SDPs have the same optimal policies since the expected return at t = 0 is the same.

Sequence Alignment Rewards Redistribution

Sequence alignment is a technique used in bioinformatics to find similarities between biological sequences and establish their evolutionary relationship (Needleman and Wunsch, 1970; Smith and Waterman, 1981; Bynagari, 2017; Bynagari, 2018; Bynagari, 2019; Bynagari & Amin, 2019; Bynagari & Fadziso, 2018; Manavalan, 2016; Manavalan, 2018; Manavalan, 2019a; Manavalan, 2019b; Manavalan & Bynagari, 2015; Manavalan & Chisty, 2019; Manavalan & Donepudi, 2016). Such alignment strategies are used by Align-RUDDER to align two or more demos with a high return. We think that the demonstrations all follow the same basic technique, thus they're similar and can be grouped together. Stormo et al. (1982) found that the

alignment produces a profile model in the form of a consensus sequence (the strategy), a frequency matrix, or a Position-Specific Scoring Matrix (PSSM). The difference between the scores of consecutive sub-sequences when aligned to the profile model is the redistributed reward of a new sequence. The new reward redistribution strategy consists of five steps as indicated in Figure 3: (I) Define events to turn state-action sequence episodes into event sequences. (II) Use Equation: $s_{i,j} = In\left(\frac{q_{ij}}{P_i P_j}\right) / \lambda^*$ to create an alignment score mechanism that aligns relevant event with one another. (III) Align all of the demonstrations in multiple sequences. (IV) Calculate the profile model and PSSM according to Equation $\sum_{i=1,j=1}^{n,n} P_i P_j \exp(\lambda s_{i,j})$. (V) Rebalance the reward: each sub-sequence τt of a new episode τ is aligned to the profile. The redistributed reward R_{t+1} is proportional to be difference of scores S based on the PSSM given in $\sum_{i=1,j=1}^{n,n} P_i P_j \exp(\lambda s_{i,j})$, that is $R_{t+1} \propto S(\tau_t) - S(\tau_{t-1})$

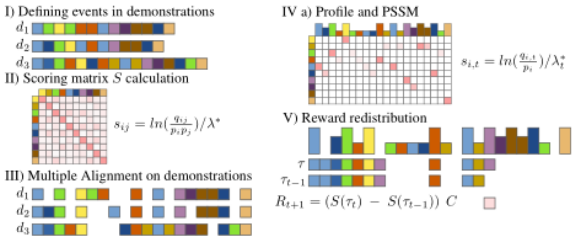

Figure 3: The five steps of Align-RUDDER's reward redistribution

Reward Redistribution

The profile model is used to redistribute rewards. A sequence, $\tau = e_{0:T}$ (e_t is the event at position t) can be aligned to the profile, yielding the score $S(\tau) = \sum_{t=0}^{L} s_{xi,t}$, where $s_{i,t}$ is the alignment score for event i at location t, x_t is the event at position t in the alignment, and L is the profile length. Because there are gaps in the alignment, L ≥ T and $x_t \neq e_t$. If the prefix sequence of τ of length t + 1 is $\tau_t = e_{0:t}$, then the payoff is

$$c = (S(\tau_t) - S(\tau_{t-1}) C = g((s; a)0:t) - g((s; a)0:t-1);$$

$$R_{T+2} = \overline{G_O} \sum_{t=0}^{T} Rt + 1, C = \frac{E_{demo}[\overline{G_O}]}{E_{demo}[\sum_{t=0}^{T} S(\tau_t) - S(\tau_{t-1})]}$$

where $R_{T+2} = \overline{G_O} \sum_{t=0}^{T} Rt + 1$ represents the sequence's original return, and $S(\tau_{t-1}) = 0$. C scales R_{t+1} to the range of $\overline{G_O}$, and E_{demo} is the expectation over demonstrations. With 0% expectation for demonstrations, R_{T+2} is the rectification of the redistributed reward (Arjona-Medina et al., 2019). $E_{demo}|R_{T+2}| = 0$. Since $\tau_t = e_{0:t}$ and $e_t = f(s_t, a_t)$, we can set $g((s,a)_{o:t}) = S(\tau_t)C$. We ensure strict return equivalence, since $G_O = \sum_{t=0}^{T+1} Rt + 1 = \overline{G_O}$. The redistribution reward depends only on the past, that is, $R_{t+1} = h(s,a)_{o:t})$. The profile alignment of τ_{t-1} can be extended to a profile alignment for τ_t for computational efficiency, just as exact matches are extended to high-scoring sequence pairs with the BLAST algorithm (Altschul et al., 1990; 1997).

Experimental Methods

Align-RUDDER is compared to Behavioral Cloning with Q-learning (BC+Q) and Deep Q-learning from Demonstrations (DQfD) on two fake tasks with sparse and delayed rewards and few demonstrations (Hester et al., 2018). GAIL (Ho and Ermon, 2016), a control system built for continuous observation spaces, failed to solve the two simulated challenges, as it had previously failed to solve similar problems (Reddy et al., 2020). Then we put Align-RUDDER to the test on the difficult MineCraft ObtainDiamond assignment, which has episodic, so long-delayed rewards (Guss et al., 2019b). All of the experiments use MDPs with a finite time horizon of $\gamma = 1$ and episodic rewards.

RESULTS AND DISCUSSION

Artificial tasks I and artificial tasks II. They are variations on the grid world rooms example (Sutton et al., 1999), in which the MDP states are represented by cells (locations). The states do not need to be time-aware in our setting to ensure an MDP, but the unobserved used-up time introduces a random influence. The grid is divided into different rooms, each of which is connected to the next simply by a single cell. The agent's purpose is to attain a target with the fewest steps possible from a starting state. Except for the first chamber, which is only connected to the second room by a portal, it must pass through various rooms that are connected by doors.

The agent is teleported to a fixed portal arrival cell in the second room if it is at the portal entry cell of the first room. The position of the portal entering cell is chosen at random for each episode, although the portal arrival cell remains constant. The position of the portal entry cell is specified in the initial room's state. The portal was created to avoid the task being solved solely via BC startup. It ensures that travelling to the portal entry cells is learned, even if they are not visible in demonstrations. If the agent stays on the grid, it can travel up, down, left, and right at any time. Except for teleportation, all state transitions are random. After T = 200 times, an episode comes to a close steps. If the agent successfully reaches

the intended area, the following stage is for it to enter an absorbing condition. There it will remain until T = 200, with no additional awards.

The five steps of Align-reward RUDDER's redistribution are then described:
(1) Events correspond to clusters of states derived using affinity propagation (Frey and Dueck, 2007), which uses the successor representation of states based on demonstrations as a measure of similarity.
(2) Equation: $\sum_{i=1,j=1}^{n,n} P_i P_j \exp(\lambda s_{i,j})$, is used to get the scoring matrix, with $\in = 0$ and all off-diagonal values of the scoring matrix set to 1.
(3) For the MSA of the demos, ClustalW is used with all gap penalties set to zero and no biological options.
(4) As seen in the MSA provides a profile model and a PSSM.
(5) The agent's generated sequences are mapped to event sequences according to step (1). Reward is reallocated using the PSSM from step 5 and differences in profile alignment scores of consecutive sub-sequences according to Equation:

$$q^{\pi}\left(s_{t,}\, a_t\right) = r\left(s_{t,}\, a_t\right) = \; q^{-\pi}\left(s_{t,}\, a_t\right) - E_{s_{t-1,}\, a_{t-1}}\left|q^{-\pi} s_{t-1,}\, a_{t-1}\right|\, s_{t,}\rfloor = q^{-\pi}\left(s_{t,}\, a_t\right) - \; \psi^{\pi}\left(s_{t,}\right)$$

Sub-tasks

Sub-tasks, which are alignment places that earn a high redistributed reward, are implicitly defined by reward redistribution (doors and portal arrival). The sub-tasks divide the Q-table into sub-tables, each of which corresponds to a sub-agent. When opposed to a single Q-table, defining sub-tasks has no effect on learning in the tabular scenario.
All of the approaches that were compared learned a Q-table and used a -greedy policy with a ratio of $\in = 0.2$. Behavioral cloning is used to set up the Q-table (BC). The state-action pairings that are not initialized because they are not visited in the demonstrations are given an optimistic start by selecting a sample from a normal distribution inferred from the demonstration returns (avoiding equal Q-values). RUDDER's Q-value estimation with correction is used by Align-RUDDER to learn the Q-table (Type A variant ii from above). Q-learning is used to learn a Q-table for BC+Q and DQfD. Grid search was used to select hyperparameters, and each approach took the same amount of time. Performance is determined by the number of episodes required to produce 80% of the average return on the demos for various numbers of demonstrations. The significance of performance differences between Align-RUDDER and the other approaches is determined using the Wilcoxon rank-sum test.

The environment for Task (I) is a 12 x 12 gridworld with four rooms. The goal is in room #4, while the starting point is in room #1, which includes 20 portal access points. The gateway entry is noted in the state for each episode. Figure 4 depicts the number of episodes required to get 80% of the average demonstration reward for various numbers of demos. Over 100 trials, the results are averaged. Align-RUDDER surpasses all other approaches, especially when there are few demonstrations (p-values of $<10^{-10}$ for up to ten demos).

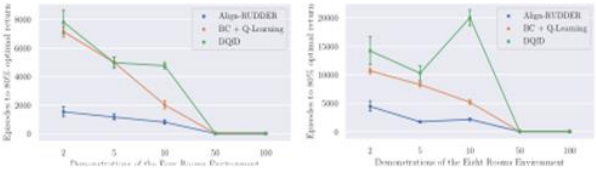

Figure 4: Comparison of Align-RUDDER and other methods on Task (I) (left) and Task (II) (right)

The environment for Task (II) is a 12 x 24 gridworld with eight rooms. The goal is in room #8, while the starting point is in room #1, which includes 20 portal access points. The findings are shown in Figure 4, with the same parameters and evaluation criteria as Task 1. (I). Align-RUDDER surpasses all other approaches, especially when there are few demonstrations (p-values of $<10^{-26}$ for up to 10 demos). No pure learning method (sub-goals are also learned) has yet to mine a diamond, to the best of our knowledge (Scheller et al., 2020). Demonstrations from human players are included in the dataset. The amount of demonstrations, however, is insufficient to immediately learn a policy that can mine a diamond from them (out of 117 demonstrations, 67 mined a diamond).

Align-five RUDDER's phases are implemented as follows:
(1) There are two parts to a state: a visual input and an inventory (including the equip state). Both sections are scaled to have the same amount of data, such as the same number of components and variance. According to Arjona-Medina et al (2019) explaining Away Problem," we cluster the differences of consecutive states. We combined minor clusters and deleted very big clusters, leaving roughly 20 clusters corresponding to events characterized by inventory changes. Finally, demonstrations are assigned to event sequences.
(2) The equation below is used to calculate the score matrix.

$$q^{\pi}\left(s_{t,}\, a_t\right) = r\left(s_{t,}\, a_t\right) = \; q^{-\pi}\left(s_{t,}\, a_t\right) - E_{s_{t-1,}\, a_{t-1}}\left|q^{-\pi} s_{t-1,}\, a_{t-1}\right|\, s_{t,}\rfloor = q^{-\pi}\left(s_{t,}\, a_t\right) - \; \psi^{\pi}\left(s_{t,}\right)$$

(3) ClustalW aligns the ten shortest demos that obtained a diamond, with gap penalties set to zero and no biological alternatives.

(4) From the multiple alignment, a profile model and a PSSM are extracted.
(5) According to Equation:

$$c = (S(\tau_t) - S(\tau_{t-1}) C = g((s; a)0:t) - g((s; a)0:t-1);$$

redistributed reward is based on the variations in profile alignment scores of consecutive sub-sequences. Using the PSSM from the previous step (4).

Sub-goals are defined based on the dispersion of rewards. Profile model positions that achieve an average redistributed reward beyond a threshold for demonstrations are recognized as sub-goals. Sub-sequences of demonstration between sub-goals are regarded demonstrations for the sub-tasks. To decide whether a sub-goal is met, the agent generates new sub-sequences that are aligned to the profile model. Because the redistributed reward across sub-goals is granted at the end of the sub-sequence, the sub-tasks receive episodic reward as well. Figure 5 shows how reward redistribution is used to identify sub-goals. Behavioral Cloning is used to pre-train sub-agents on the sub-task demos, and then Proximal Policy Optimization (PPO) is used to train them in the environment (Schulman et al., 2018).

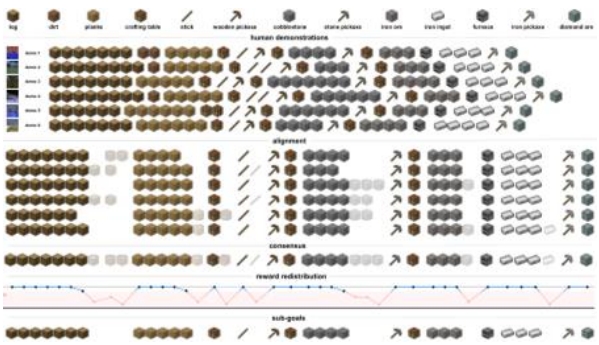

Figure 5: Example of alignment and reward redistribution for demonstrations of ObtainDiamond. Thresholding the redistributed reward identifies sub-goals

Our primary agent is capable of performing all tasks, as well as executing sub-agents and learning from the redistributed reward (return-equivalent MDP). The primary agent is started by executing sub-agents in accordance with the alignment, but it is free to diverge from this technique. The Appendix contains more information on architectures, hyperparamters, and other technical specifics. With just ten demonstrations, Align-RUDDER can learn how to mine diamonds. When the 31 extracted sub-tasks from the ObtainDiamond environment are considered, a diamond is obtained in 0.1 percent of the cases. To put this proportion into perspective, consider a 0.5 chance of success for each extracted sub-task, which already necessitates a highly trained agent. The success rate for mining the diamond as a result would be around 4:66 1010.

CONCLUSION

From a few examples, Align-RUDDER can learn to accomplish exceedingly complicated problems with delayed and sparse rewards. Align-RUDDER is based on the reward redistribution theory, which ensures that optimal policies are maintained while the reward delay is significantly decreased. Alignment techniques from bioinformatics are used in reward redistribution. With few demos, Align- RUDDER surpasses competitors on artificial tasks. Align-RUDDER was able to mine a diamond in 0.1 percent of the MineCraft ObtainDiamond tasks.

REFERENCES

Ahmed, A.A.A. (2021). Event Ticketing Accounting Information System using RFID within the COVID-19 Fitness Etiquettes. Academia Letters, Article 1379. https://doi.org/10.20935/AL1379

Altschul, S. F., Gish, W., Miller, W., Myers, E. W. and Lipman, D. J. 1990. Basic local alignment search tool. J. Molec. Biol., 214:403–410, 1990.

Altschul, S. F., Madden, T. L., Schäffer, A. A., Zhang, J., Zhang, Z., Miller, W. and Lipman D. J. 1997. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Research, 25(17):3389–3402, 1997. doi: 10.1093/nar/25.17.3389.

Antonoglou, I., V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. P. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and Hassabis. Mastering the game of Go with deep neural networks and tree search. Nature, 529(7587):484–489. doi:10.1038/nature16961.

Arjona-Medina, J. A., Gillhofer, M., Widrich, M., Unterthiner, T., Brandstetter, J. and Hochreiter, S. 2019. RUDDER: return decomposition for delayed rewards. In Advances in Neural Information Processing Systems 32, pp. 13566–13577.

Bynagari, N. B. & Ahmed, A. A. A. (2021). Anti-Money Laundering Recognition through the Gradient Boosting Classifier. *Academy of Accounting and Financial Studies Journal*, *25*(5), 1–11. https://doi.org/10.5281/zenodo.5523918

Bynagari, N. B. (2017). Prediction of Human Population Responses to Toxic Compounds by a Collaborative Competition. *Asian Journal of Humanity, Art and Literature*, *4*(2), 147-156. https://doi.org/10.18034/ajhal.v4i2.577

Bynagari, N. B. (2018). On the ChEMBL Platform, a Large-scale Evaluation of Machine Learning Algorithms for Drug Target Prediction. *Asian Journal of Applied*

Science and Engineering, *7*, 53–64. Retrieved from https://upright.pub/index.php/ajase/article/view/31

Bynagari, N. B. (2019). GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. *Asian Journal of Applied Science and Engineering*, *8*, 25–34. Retrieved from https://upright.pub/index.php/ajase/article/view/32

Bynagari, N. B., & Amin, R. (2019). Information Acquisition Driven by Reinforcement in Non-Deterministic Environments. *American Journal of Trade and Policy*, *6*(3), 107-112. https://doi.org/10.18034/ajtp.v6i3.569

Bynagari, N. B., & Fadziso, T. (2018). Theoretical Approaches of Machine Learning to Schizophrenia. *Engineering International*, *6*(2), 155-168. https://doi.org/10.18034/ei.v6i2.568

Ganapathy, A., Vadlamudi, S., Ahmed, A. A. A., Hossain, M. S., Islam, M. A. (2021). HTML Content and Cascading Tree Sheets: Overview of Improving Web Content Visualization. Turkish Online Journal of Qualitative Inquiry, 12(3), 2428-2438. https://doi.org/10.5281/zenodo.5522159

Hester, T., M. Vecerík, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, G. Dulac-Arnold, J. Agapiou, J. Z. Leibo, and A. Gruslys. 2018. Deep q-learning from demonstrations. In The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18). Association for the Advancement of Artificial Intelligence, 2018.

Ho J. and Ermon S. 2016. Generative adversarial imitation learning. In Advances in Neural Information Processing Systems 29, pp. 4565–4573, 2016.

Hochreiter S. and Schmidhuber J. 1995. Long short-term memory. Technical Report FKI-207-95, Fakultätfür Informatik, Technische Universität München, 1995.

Hochreiter S. and Schmidhuber J. 1997a. Long short-term memory. Neural Comput., 9(8):1735–1780.

Hochreiter S. and Schmidhuber J. 1997b. LSTM can solve hard long time lag problems. In M. C. Mozer,

Hussain, S., Ahmed, A. A. A., Kurniullah, A. Z., Ramirez-Asis, E., Al-Awawdeh, N., Al-Shamayleh, N. J. M., Julca-Guerrero, F. (2021). Protection against Letters of Credit Fraud. Journal of Legal, Ethical and Regulatory Issues, 24(Special Issue 1), 1-11. https://doi.org/10.5281/zenodo.5507840

Luoma, J., Ruutu, S., King, A. W. and Tikkanen H. 2017. Time delays, competitive interdependence, and firm performance. Strategic Management Journal, 38(3):506–525. doi: 10.1002/smj.2512.

Manavalan, M. (2016). Biclustering of Omics Data using Rectified Factor Networks. *International Journal of Reciprocal Symmetry and Physical Sciences*, *3*, 1–10. Retrieved from https://upright.pub/index.php/ijrsps/article/view/40

Manavalan, M. (2018). Do Internals of Neural Networks Make Sense in the Context of Hydrology?. *Asian Journal of Applied Science and Engineering*, *7*, 75–84. Retrieved from https://upright.pub/index.php/ajase/article/view/41

Manavalan, M. (2019a). P-SVM Gene Selection for Automated Microarray Categorization. *International Journal of Reciprocal Symmetry and Physical Sciences*, *6*, 1–7. Retrieved from https://upright.pub/index.php/ijrsps/article/view/43

Manavalan, M. (2019b). Using Fuzzy Equivalence Relations to Model Position Specificity in Sequence Kernels. *Asian Journal of Applied Science and Engineering*, *8*, 51–64. Retrieved from https://upright.pub/index.php/ajase/article/view/42

Manavalan, M., & Bynagari, N. B. (2015). A Single Long Short-Term Memory Network can Predict Rainfall-Runoff at Multiple Timescales. *International Journal of Reciprocal Symmetry and Physical Sciences*, *2*, 1–7. Retrieved from https://upright.pub/index.php/ijrsps/article/view/39

Manavalan, M., & Chisty, N. M. A. (2019). Visualizing the Impact of Cyberattacks on Web-Based Transactions on Large-Scale Data and Knowledge-Based Systems. *Engineering International*, *7*(2), 95-104. https://doi.org/10.18034/ei.v7i2.578

Manavalan, M., & Donepudi, P. K. (2016). A Sample-based Criterion for Unsupervised Learning of Complex Models beyond Maximum Likelihood and Density Estimation. *ABC Journal of Advanced Research*, *5*(2), 123-130. https://doi.org/10.18034/abcjar.v5i2.581

Manojkumar, P., Suresh, M., Ahmed, A. A. A., Panchal, H., Rajan, C. C. A., Dheepanchakkravarthy, A., Geetha, A., Gunapriya, B., Mann, S., & Sadasivuni, K. K. (2021). A novel home automation distributed server management system using Internet of Things. International Journal of Ambient Energy, https://doi.org/10.1080/01430750.2021.1953590

Needleman S. B. and Wunsch C. D. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of Molecular Biology, 48(3):443–453, 1970.

Panchal, H., Sadasivuni, K. K., Ahmed, A. A. A., Hishan, S. S., Doranehgard, M. H., Essa, F. A., Shanmugan, S., & Khalid, M. (2021). Graphite powder mixed with black paint on the absorber plate of the solar still to enhance yield: An experimental investigation. Desalination, Volume 520. https://doi.org/10.1016/j.desal.2021.115349

Rahmandad, H., Repenning, N. and Sterman J. 2009. Effects of feedback delay on learning. System Dynamics Review, 25(4):309–338. doi: 10.1002/sdr.427.

Raya, I., Kzar, H. H., Mahmoud, Z. H., Ahmed, A. A. A., Ibatova, A. Z., & Kianfar, E. (2021). A review of gas sensors based on carbon nanomaterial. Carbon Letters. Article No: 276. https://doi.org/10.1007/s42823-021-00276-9

Reddy, S., Dragan, A. D. and. Levine S. 2020. SQIL: imitation learning via regularized behavioral cloning. ArXiv, 2020. Eighth International Conference on Learning Representations (ICLR).

Scheller, C., Y. Schraner, and M. Vogel. 2020. Sample efficient reinforcement learning through learning from demonstrations in Minecraft. arXiv, abs/2003.06066, 2020.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A. and Klimov O. 2018. Proximal policy optimization algorithms. ArXiv, 2018.

Sharma, D. K., Chakravarthi, D. S., Shaikh, A. A., Ahmed, A. A. A., Jaiswal, S., Naved, M. (2021). The aspect

of vast data management problem in healthcare sector and implementation of cloud computing technique. Materials Today: Proceedings. https://doi.org/10.1016/j.matpr.2021.07.388

Silver, D., A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, Frey B. J. and Dueck D. 2007. Clustering by passing messages between data points. Science, 315(5814): 972–976, 2007. doi: 10.1126/science.1136800.

Smith T. F. and Waterman M. S. 1981. Identification of common molecular subsequences. Journal of Molecular Biology, 147(1):195–197, 1981

Stormo, G. D., Schneider, T. D., Gold, L. and Ehrenfeucht A. 1982. Use of the 'Perceptron' algorithm to distinguish translational initiation sites in E. coli. Nucleic Acids Research, 10(9):2997–3011, 1982.

Sutton R. S. and Barto A. G. 2018. Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA, 2 edition.

Sutton, R. S., Precup, D. and Singh S. P. 1999. Between MDPs and Semi-MDPs: A framework for temporal abstraction in reinforcement learning. Artificial Intelligence, 112(1-2):181–211, 1999.

--0--