International Journal of Electrical and Computer Engineering (IJECE)
Vol. 14, No. 1, February 2024, pp. 904~910
ISSN: 2088-8708, DOI: 10.11591/ijece.v14i1.pp904-910

Reinforcement learning strategies using Monte-Carlo to solve the blackjack problem

Raghavendra Srinivasaiah[1], Vinai George Biju[2], Santosh Kumar Jankatti[3], Ravikumar Hodikehosahally Channegowda[4], Niranjana Shravanabelagola Jinachandra[5]
[1]Department of Computer Science and Engineering, School of Engineering and Technology, CHRIST Deemed to be University, Bengaluru, India
[2]MAI, Faculty of Computer Science and Business Informatics, University of Applied Sciences, Würzburg-Schweinfurt, Germany
[3]Department of Computer Science and Technology, Dayananda Sagar University, Bengaluru, India
[4]Department of Electronics and Communication Engineering, Dayananda Sagar Academy of Technology and Management, Bengaluru, India
[5]Department of Mechanical Engineering, School of Engineering and Technology, CHRIST Deemed to be University, Bengaluru, India

Article Info

Article history:

Received May 18, 2023
Revised Jul 16, 2023
Accepted Jul 17, 2023

Keywords:

Blackjack
Dynamic programming
Monte Carlo
Q-learning
Reinforcement learning
Temporal difference

ABSTRACT

Blackjack is a classic casino game in which the player attempts to outsmart the dealer by drawing a combination of cards with face values that add up to just under or equal to 21 but are more incredible than the hand of the dealer he manages to come up with. This study considers a simplified variation of blackjack, which has a dealer and plays no active role after the first two draws. A different game regime will be modeled for everyone to ten multiples of the conventional 52-card deck. Irrespective of the number of standard decks utilized, the game is played as a randomized discrete-time process. For determining the optimum course of action in terms of policy, we teach an agent-a decision maker-to optimize across the decision space of the game, considering the procedure as a finite Markov decision chain. To choose the most effective course of action, we mainly research Monte Carlo-based reinforcement learning approaches and compare them with q-learning, dynamic programming, and temporal difference. The performance of the distinct model-free policy iteration techniques is presented in this study, framing the game as a reinforcement learning problem.

Corresponding Author:

Raghavendra Srinivasaiah
Department of Computer Science and Engineering, School of Engineering and Technology, CHRIST Deemed to be University
Mysore Road, Bengaluru-560074, Karnataka, India
Email: raghav.trg@gmail.com

1. INTRODUCTION

The experimental version of blackjack is played as follows: at the beginning of each round, the agent and dealer will each get two cards from a deck of size, where size equals 52. The first dealer's card value and the total worth of the dealer's hand both form their features in the construction of state spaces. The agent can then select from the action space, action (state)=hit or stick, with the resulting consequences of drawing a new card for the agent's hand or submitting the existing hand for scoring, respectively. Given the limitations of the game's play, the agent is urged to obtain the maximum squared score possible.

Techniques for reinforcement learning (RL) rely on input from the outside world to help learners progress. The agent is guided in formulating its policy by feedback, which comes as a monetary reward signal. A Markov decision process (MDP) is typically used to represent the environment. An MDP comprises several phases, activities, simulated results, and predicted rewards. Each action has a chance of being chosen

and a value linked to it that reflects the expected benefit of doing the action. The most beneficial activity is one that is motivated by greed. The agent must strike a balance between exploring and making use of the surroundings to learn. The agent tries a greedy approach throughout the exploration to enhance its assessments of their values.

The agent will update the state values and, in addition to that, state-action values on these methods independently. We then present the results of applying each technique in terms of the win, draw, and loss percentage per game regime and each process for each game regime, respectively. The framework to mathematically formalize this optimization problem is modelled as RL. It was proposed that, given a discrete step, t, the values of states are linked recursively by the Bellman equation. Value functions allow the agent to determine the immediate value of being a resident of a state s or acting in a way that starts with s while adhering to specified norms. These value functions might technically be defined as in (1) and (2):

$$V^{\pi}(S) = E_{\pi}\{ \sum_{k=0}^{\infty} \gamma^{k} r_{t+k+1} \mid S_t = S \} \quad (1)$$

$$Q^{\pi}(S,a) = E_{\pi}\{ \sum_{k=0}^{\infty} \gamma^{k} r_{t+k+1} \mid S_t = S, a_t = a \} \quad (2)$$

The agent can calculate the predicted benefit using $V^{\pi}(S)$, the state-value function, of being in state s and following policy. The agent can determine the immediate value of being in state S, acting in method a, and subsequently carrying out policy based on the action-value function Q*(S, a). The activities that produce the highest long-term reward make up an optimum policy. The ideal state-value function is denoted by V*(s), while the perfect action-value part is characterized by Q*(s, a). The properties of different learning algorithms and their behavior are listed in the following section, namely Monte Carlo (MC) algorithm, q-learning (QL) algorithm, dynamic programming (DP) algorithm, and temporal difference (TD) learning algorithm.

The MC technique is used to solve the blackjack problem. It is simple to set up exploration states that consider all potential outcomes because the episodes are based on simulations of games. In this instance, the player's total, the dealer's cards, and the player possessing a usable ace are all chosen randomly with the same chance. The estimated policy that stays on only 20 or 21 from the last blackjack instance is used as the initial strategy. For all state-action couples, the initial action-value function can be zero.

Consider the situation where we need to approximate v(s), the assessment of state' s' using policy, having a collection of episodes we acquired by subsequent and transitory through s. A visit to s refers to each instance of state' s' in an episode. Although s could appear more than once in a single episode, let's guide to the initial appearance as the first visit to s. Each visit using the MC technique is considered as the average of the yields after all trips to s, whereas the initial visit MC technique approximates v(s) as the mean of the returns after initial visits to s. Although the theoretical characteristics of these two MC approaches are similar, they differ somewhat. The independent estimations for each state are a crucial aspect of MC techniques. Unlike DP, the estimate for the state does not add to the approximation of other states [1]–[7].

The heuristic moves update the means of all actions performed first on their intersections with the same color as the first move with the score after a random game with a score rather than updating the mean of the actions of the random game. Symmetrically, the steps as-first heuristic modifies with the opposite score the means of all movements performed first on their intersections with various colors from the first move. Overall, this heuristic updates nearly every move's mean as though it were the initial move in the random game. This heuristic could be more accurate since it may update two movements with the same score even if they have distinct impacts based on the timing of their play. Since TD errors in RL are dependent on estimations of the value function, which are dynamically changing, it is evident that they are pretty noisy.

Furthermore, in this issue, the policy is also altering. Hence one would anticipate that the TD errors would not be stable. Estimating q(s, a), the predictable return while being in initial state s, using action a, and after policy, is the goal of the policy analysis problem for action values. The MC techniques are substantially similar to those previously discussed for matters for states, except we now speak about trips to state-action duo rather than a state. When a state is reached, and action is reserved in a given episode, the pair of action-state s is said to have been visited. The value of the action-state duo is estimated by each visit using the MC technique as the returns average that trailed all the visits to it. The returns after the state were visited and the action was chosen for the first time in each episode are averaged using the first-visit MC approach. As more visits are made to each state-action pair, the techniques, as previously, converge quadratically to the valid anticipated values. The agent is restricted to searching for optimal actions via updates. In other words, the agent may find an optimal action from trajectories identical to the current state, and this is contrary to QL, where the agent may search for any state that matches the current rather than being restricted to those states that arose from identical trajectories.

The QL algorithm is the TD method that gets the closest to Q*(s, a). Q*(s, a) estimations are modified at every step using incremental algorithms in TD approaches like QL. Below is a description of the QL in (3). Given that learning may occur while playing, the QL algorithm is a great way to approximate the best blackjack strategy. As a result, it is an excellent option for the blackjack problem domain. Blackjack is phrased as a serialized activity, with each hand's finish signifying the end of a single episode. The agent's current point total, the dealer's face-up card value, the hand's softness, and the possibility of splitting are all included in the state representation [8]–[16].

$$Q(s,a) <= Q(s,a) + \propto [r + \gamma \, max_{a'} \, Q(s',a) - Q(s,a)] \quad (3)$$

A class of techniques known as DP combines solutions to sub-problems to solve more significant, complicated issues. A known MDP may be solved using DP approaches in planning by identifying the optimal value function and its accompanying optimal policy. A fundamental tenet of optimal control is Bellman's principle, which argues that if an optimum policy has already been chosen, the subsequent choices must also be optimal in light of the state created by the prior decisions, and this is often referred to as the discrete-time Hamilton Jacobi Bellman equation or the Bellman optimality equation. The best course of action would then be as in (4).

$$h^*(x_k) = arg \; min_{h(.)} \left(r\left(x_k, h(x_k)\right) \gamma^{V^*}(x_{k+1})\right. \quad (4)$$

Bellman's principle in (5) results in a time approach calculated backward for resolving the optimum control issue since one has to know the ideal policy at time usage to derive the optimal policy at time k. It is the foundation for the DP techniques widely used in operations research, control system theory, and other fields. These are offline planning techniques.

$$V^{\pi}(S) = E_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t \, r_t \mid S_0 = S\right] = \sum_{s' \epsilon \, S} P_{ss'}^{\pi(s)} \left[R_{ss'}^{\pi(s)} + \gamma \, V^{\pi}(s')\right] \quad (5)$$

Action-value functions are used to store the policy implicitly. The policy is greedy concerning the value-action-function Q is known if, for any state' s', a value-action Q (s, a) is accessible for all actions. With this modification, policies won't need to be stored explicitly. To keep the value function, it should be noted that moving from V to Q increases the memory need by a factor of |A| [17]–[26].

One approach to solving RL issues is using TD. The predicted long-term payoff for performing a particular action in a state is estimated by TD techniques using a value function. The TD method is an online learning approach. The agent updates the state value during a trajectory or episode rather than waiting until the game is finished and then updating over the explored trajectory sequentially using (6).

$$\delta_t = r_{t+1} + \gamma Q(S_{t+1}, a_{t+1}) - Q(S_t, a_t) \quad (6)$$

The value function is trained in TD models by calculating a value-prediction error signal every instant the agent switches the states. The transformation between the estimated value and the actual value, which includes the instant reward, as seen during the switchover, is represented by the letter “d”. The previous state's value estimate can be reorganized based on d to be closer to the observed value. This “d” signal manifests in response to unexpected rewards, propagates with learning from rewards to pre-emptive incentives, and modifies in response to variations in predicted reward. The interest of RL is to create the best possible policy that maximizes value across all possible states. The methods employed could iteratively build such a policy from data. For the policy to be evaluated by on-policy algorithms, it must primarily be greedy regarding value function estimations [27]–[35].

2. EXPERIMENTS AND EVALUATION

Hit-and-stick actions are included, and an effort is made to maintain the integrity of the game using all feasible measures. The following summarizes the reward system: For every action that does not change to a terminal state, a zero reward is awarded. The agent is rewarded according to the game's outcome when a terminal condition is attained. For instance, if the agent gains $1 and wins the hand, they are rewarded with a +1. If the hand is lost, the agent is rewarded -1. Due to the state representation's inability to foresee which cards may come into subsequent hands, a static betting approach is employed.

The performance of the learning agent was evaluated using two hardcoded players (dealer and player). The acts of the first player are wholly arbitrary, whereas the second player employs a basic plan.

Using a basic strategy, the best course of action for the state representation selected reduces the casino edge to less than one percent. We utilize several runs in the experiment, each consisting of a loop of test and training hands. During training, the agent interacts with the environment states and uses the same lookup tables. The agent competes against the player, and the player uses basic strategy during the test hands. After performing a series of first random trials, every trial places the opening card. While the number of test hands stays fixed, numerous bets are eventually reached by gradually increasing the amount of training hands towards each run.

3. RESULTS AND DISCUSSION

Mean-variance optimization in MDPs and work on resilient MDPs both focus on increasing the safety of policies. However, it should be noted that the method suggests a combined criterion that considers both the mean and variance of the value function, which is much more expensive to optimize. Furthermore, even if needed later on, optimal values and policies cannot be retrieved in that work since the value function is not learned individually. The controllability only influences the action options; the optimal values are unaffected by controllability values and are done to retain the value function's correctness.

The initial policy of the RL agent is entirely arbitrary. Therefore, it will likely produce similar results to the random player. Because the QL method closely approximates Q*, the learning agent's strategy should reach a fundamental approach as the quantity of training hands rises (s, a) and is accurate even for the best practice, highlighting the difficulty of gaining money when playing blackjack. Based on a comparison between the two, the learning agent outperforms the random player significantly. It is clear that perhaps the agent is picking up relevant knowledge during the iteration. The efficiency of the learning agent improves as it asymptotically approaches that of a player adopting a primary strategy and is the predicted Q* since the QL process directly approximates the ideal action-value function Q* (s, a).

Figure 1 illustrates the strategy to hit to stick for each card the learner adopts during the game. The learning is improved over a higher number of iterations. The value function differs when using the ace card, as shown in Figure 2. The dealer and player values are plotted against the state value. This investigation involved an agent learning to model a finite MC decision chain. A natural extension is to consider an infinite-sized deck such that now the agent would attempt to model a Markov chain-MC-based RL; For this; we expect the agent to find the stationary per-game score value, which appears to be converging. Including additional blackjack game actions like split and double would make the application more attractive and, in turn, cause the action space to become A = hit; stick and edge the agent's policy closer to real-life application, but in addition to that, exponentially increasing the size of the state space. The snapshot of the state-action space of the implementation is indicated in Figure 1.

	2	3	4	5	6	7	8	9	10	11
3	H	H	H	H	H	H	H	H	H	H
4	S	H	S	H	H	H	H	H	H	S
5	S	S	H	H	H	H	S	H	S	H
6	H	H	H	H	H	H	H	H	H	S
7	S	H	H	H	H	H	H	H	H	H
8	H	H	H	H	H	H	S	H	S	S
9	H	H	H	H	H	H	H	H	H	H
10	H	H	H	H	H	H	H	H	H	S
11	H	H	H	H	H	H	H	H	H	H
12	H	S	S	S	H	H	S	H	H	H
13	S	S	S	S	H	S	H	H	H	S
14	S	S	H	S	S	H	S	S	H	H
15	H	S	H	S	H	H	S	H	H	S
16	H	H	S	H	S	H	S	H	S	S
17	S	S	S	S	S	S	S	S	S	S
18	S	S	S	S	S	S	S	S	S	H
19	S	S	S	S	S	S	S	S	S	S
20	S	S	S	S	S	S	S	S	S	S

Figure 1. Strategy outcome and state space snapshot

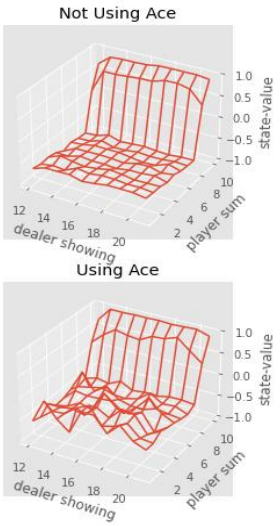

Figure 2. Value function with/without ace

4. CONCLUSION

The blackjack game is studied and explored for optimal strategies using the MC-based RL technique. The foundation study of different RL techniques like QL, DP, and TD learning is also being

carried out in the work. This investigation found that training an agent to play a simplified version of blackjack produces desirable results; the agent can see a policy p that makes an average win, draw and loss rate of 38:26 across different policy iteration methods. The best blackjack strategy employing the MC method was investigated in this study using RL. It has been shown that the learning agent outperformed random and converged to a nearly ideal policy. Although the results are positive, there is still potential for development. The outcomes could be improved with a better exploration technique like the Bayesian MC Learning Algorithm. Additionally, a policy outperforming basic strategy may result from a more robust state representation combined with a dynamic betting approach. The effect on value function based on the rule change related to including or not including the ace is also studied.

ACKNOWLEDGEMENTS

I would like to thank CHRIST Deemed to be University for providing me an opportunity and facility in completing this work.

REFERENCES

[1] V. S. Borkar, "Reinforcement learning - a bridge between numerical methods and Monte Carlo," in *Perspectives In Mathematical Science I: Probability And Statistics*, WORLD SCIENTIFIC, 2009, pp. 71–91.
[2] J. Asmuth and M. Littman, "Learning is planning: near Bayes-optimal reinforcement learning via Monte-Carlo tree search," *arXiv preprint arXiv:1202.3699*, Feb. 2012.
[3] F. Bai, X. Ju, S. Wang, W. Zhou, and F. Liu, "Wind farm layout optimization using adaptive evolutionary algorithm with Monte Carlo tree search reinforcement learning," *Energy Conversion and Management*, vol. 252, Jan. 2022, doi: 10.1016/j.enconman.2021.115047.
[4] N. A. Vien, W. Ertel, V. H. Dang, and T. Chung, "Monte-Carlo tree search for Bayesian reinforcement learning," *Applied Intelligence*, vol. 39, no. 2, pp. 345–353, Feb. 2013, doi: 10.1007/s10489-012-0416-2.
[5] J. Veness, K. S. Ng, M. Hutter, W. Uther, and D. Silver, "A Monte-Carlo AIXI approximation," *Journal of Artificial Intelligence Research*, vol. 40, pp. 95–142, Sep. 2011, doi: 10.1613/jair.3125.
[6] I. P. Pinto and L. R. Coutinho, "Hierarchical reinforcement learning with Monte Carlo tree search in computer fighting game," *IEEE Transactions on Games*, vol. 11, no. 3, pp. 290–295, Sep. 2019, doi: 10.1109/TG.2018.2846028.
[7] D. Silver and G. Tesauro, "Monte-carlo simulation balancing," in *ACM International Conference Proceeding Series*, Jun. 2009, vol. 382, doi: 10.1145/1553374.1553495.
[8] D. Zha *et al.*, "RLCard: a toolkit for reinforcement learning in card games," *arXiv preprint arXiv:1910.04376*, Oct. 2019.
[9] S. A. Kakvi, "Reinforcement learning for Blackjack," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5709 LNCS, Springer Berlin Heidelberg, 2009, pp. 300–301.
[10] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-Learning," *30th AAAI Conference on Artificial Intelligence, AAAI 2016*, vol. 30, no. 1, pp. 2094–2100, Mar. 2016, doi: 10.1609/aaai.v30i1.10295.
[11] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 2020-Decem, Jun. 2020.
[12] S. Gu, T. Lillicrap, U. Sutskever, and S. Levine, "Continuous deep q-learning with model-based acceleration," *33rd International Conference on Machine Learning, ICML 2016*, vol. 6, pp. 4135–4148, Mar. 2016.
[13] H. Xu, X. Zhan, and X. Zhu, "Constraints penalized q-learning for safe offline reinforcement learning," *Proceedings of the 36th AAAI Conference on Artificial Intelligence, AAAI 2022*, vol. 36, no. 8, pp. 8753–8760, Jun. 2022, doi: 10.1609/aaai.v36i8.20855.
[14] C. E. Mariano and E. F. Morales, "DQL: a new updating strategy for reinforcement learning based on q-learning," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 2167, Springer Berlin Heidelberg, 2001, pp. 324–335.
[15] B. O'Donoghue, R. Munos, K. Kavukcuoglu, and V. Mnih, "Combining policy gradient and q-learning," *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2017, doi: 10.1007/978-1-4842-6809-4_8.
[16] R. Carmona, M. Laurière, and Z. Tan, "Model-free mean-field reinforcement learning: mean-field MDP and mean-field q-learning," *arXiv preprint arXiv:1910.12802*, Oct. 2019.
[17] L. Buşoniu, R. Babuška, B. De Schutter, and D. Ernst, *Reinforcement learning and dynamic programming using function approximators*. CRC Press, 2010.
[18] F. L. Lewis and D. Vrabie, "Reinforcement learning and adaptive dynamic programming for feedback control," *IEEE Circuits and Systems Magazine*, vol. 9, no. 3, pp. 32–50, 2009, doi: 10.1109/MCAS.2009.933854.
[19] A. Geramifard, T. J. Walsh, S. Tellex, G. Chowdhary, N. Roy, and J. P. How, "A tutorial on linear function approximators for dynamic programming and reinforcement learning," *Foundations and Trends in Machine Learning*, vol. 6, no. 4, pp. 375–454, 2013, doi: 10.1561/2200000042.
[20] H. Lee, C. Song, N. Kim, and S. W. Cha, "Comparative analysis of energy management strategies for HEV: dynamic programming and reinforcement learning," *IEEE Access*, vol. 8, pp. 67112–67123, 2020, doi: 10.1109/ACCESS.2020.2986373.
[21] D. Zhao, D. Liu, F. L. Lewis, J. C. Principe, and S. Squartini, "Special issue on deep reinforcement learning and adaptive dynamic programming," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 6, pp. 2038–2041, Jun. 2018, doi: 10.1109/TNNLS.2018.2818878.
[22] A. Yamaguchi and C. G. Atkeson, "Neural networks and differential dynamic programming for reinforcement learning problems," in *Proceedings - IEEE International Conference on Robotics and Automation*, May 2016, vol. 2016-June, pp. 5434–5441, doi: 10.1109/ICRA.2016.7487755.
[23] M. Zolfpour-Arokhlo, A. Selamat, S. Z. Mohd Hashim, and H. Afkhami, "Modeling of route planning system based on q value-based dynamic programming with multi-agent reinforcement learning algorithms," *Engineering Applications of Artificial Intelligence*, vol. 29, pp. 163–177, Mar. 2014, doi: 10.1016/j.engappai.2014.01.001.
[24] D. K. Sharma, J. J. P. C. Rodrigues, V. Vashishth, A. Khanna, and A. Chhabra, "RLProph: a dynamic programming based

reinforcement learning approach for optimal routing in opportunistic IoT networks," *Wireless Networks*, vol. 26, no. 6, pp. 4319–4338, Apr. 2020, doi: 10.1007/s11276-020-02331-1.
[25] A. Gonzalez-Garcia, D. Barragan-Alcantar, I. Collado-Gonzalez, and L. Garrido, "Adaptive dynamic programming and deep reinforcement learning for the control of an unmanned surface vehicle: experimental results," *Control Engineering Practice*, vol. 111, p. 104807, Jun. 2021, doi: 10.1016/j.conengprac.2021.104807.
[26] J. Wu, Y. Zou, X. Zhang, T. Liu, Z. Kong, and D. He, "An online correction predictive EMS for a hybrid electric tracked vehicle based on dynamic programming and reinforcement learning," *IEEE Access*, vol. 7, pp. 98252–98266, 2019, doi: 10.1109/ACCESS.2019.2926203.
[27] K. Krawiec and M. Szubert, "Coevolutionary temporal difference learning for small-board Go," *2010 IEEE World Congress on Computational Intelligence, WCCI 2010 - 2010 IEEE Congress on Evolutionary Computation, CEC 2010*, Jul. 2010, doi: 10.1109/CEC.2010.5586054.
[28] I. Menache, S. Mannor, and N. Shimkin, "Basis function adaptation in temporal difference reinforcement learning," *Annals of Operations Research*, vol. 134, no. 1, pp. 215–238, Feb. 2005, doi: 10.1007/s10479-005-5732-z.
[29] M. E. Taylor, S. Whiteson, and P. Stone, "Comparing evolutionary and temporal difference methods in a reinforcement learning domain," in *GECCO 2006 - Genetic and Evolutionary Computation Conference*, Jul. 2006, vol. 2, pp. 1321–1328, doi: 10.1145/1143997.1144202.
[30] Z. Kurth-Nelson and A. D. Redish, "Temporal-difference reinforcement learning with distributed representations," *PLoS ONE*, vol. 4, no. 10, p. e7362, Oct. 2009, doi: 10.1371/journal.pone.0007362.
[31] A. H. Tan, N. Lu, and D. Xiao, "Integrating temporal difference methods and self-organizing neural networks for reinforcement learning with delayed evaluative feedback," *IEEE Transactions on Neural Networks*, vol. 19, no. 2, pp. 230–244, Feb. 2008, doi: 10.1109/TNN.2007.905839.
[32] P. Tano, P. Dayan, and A. Pouget, "A local temporal difference code for distributional reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 2020-Decem, 2020, doi: 10.5555/3495724.
[33] A. Filos, C. Lyle, Y. Gal, S. Levine, N. Jaques, and G. Farquhar, "PsiPhi-learning: reinforcement learning with demonstrations using successor features and inverse temporal difference learning," *Proceedings of Machine Learning Research*, vol. 139, pp. 3305–3317, Feb. 2021.
[34] K. De Asis, A. Chan, S. Pitis, R. S. Sutton, and D. Graves, "Fixed-horizon temporal difference methods for stable reinforcement learning," *AAAI 2020 - 34th AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, pp. 3741–3748, Sep. 2019, doi: 10.1609/aaai.v34i04.5784.
[35] M. Ruiz-Montiel, L. Mandow, and J. L. Pérez-de-la-Cruz, "A temporal difference method for multi-objective reinforcement learning," *Neurocomputing*, vol. 263, pp. 15–25, Nov. 2017, doi: 10.1016/j.neucom.2016.10.100.

BIOGRAPHIES OF AUTHORS

Raghavendra Srinivasaiah is currently working as associate professor in the Department of Computer Science and Engineering at CHRIST Deemed to be University, Bangalore. He completed his Ph.D. degree in Computer Science and Engineering from VTU, Belgaum, India in 2017 and has more than 18+ years of teaching experience. His interests include data mining, artificial intelligence and big data. He can be contacted through e-mail: raghav.trg@gmail.com.

Vinai George Biju is currently pursuing M.S. at University of Applied Sciences, Faculty of Computer Science and Business Informatics, Germany. He completed his Ph.D. degree in computer science and engineering from VTU, Belgaum, India in 2021 and has more than 10 years of teaching experience. His interests include data mining, artificial intelligence and big data. He can be contacted through e-mail: vinaigb@gmail.com.

Santosh Kumar Jankatti is currently working as associate professor in the Department of Computer Science and Technology at Dayananda Sagar University, Bangalore. He completed his Ph.D. degree in computer science and engineering from VTU, Belgaum, India in 2022 and has more than 12 years of teaching experience and 3 years of IT industry experience. His interests include data mining, artificial intelligence and big data. He can be contacted through e-mail: sjankatti@gmail.com.

Ravikumar Hodikehosahally Channegowda completed his Ph.D. from VTU, Belagavi in 2021. He has done his masters in VLSI design and embedded systems from VTU Extension Centre, PESCE, Mandya. His areas of interest are image processing, machine learning, pattern recognition and multimedia concepts. He is currently working as an assistant professor at Dayananda Sagar Academy of Technology and Management, Bengaluru. He can be contacted through e-mail: raviec40@gmail.com.

Niranjana Shravanabelagola Jinachandra completed his Ph.D. from VTU, Belagavi in 2022. He has done his masters in machine design from VTU, Belagavi. His areas of interest are image processing, machine learning, and fluid dynamics. He is currently working as assistant professor in the Department of Mechanical Engineering at CHRIST Deemed to be University. He can be contacted through e-mail: sjniranjan86@gmail.com.