http://www.egejfas.org
Ege Journal of Fisheries and Aquatic Sciences, 42(1), 21-26 (2025)

Su Ürünleri Dergisi (2025)
DOI: 10.12714/egejfas.42.1.03

RESEARCH ARTICLE ARAŞTIRMA MAKALESİ

Detection of moving fish schools using reinforcement learning technique

Takviyeli öğrenme tekniği kullanarak haraketli balık sürülerinin tespiti

Mehmet Yaşar Bayraktar*

Bingol University, Faculty of Engineering, Department of Computer Engineering, 1200, Bingöl, Türkiye

*Corresponding author: mybayraktar@bingol.edu.tr

Received date: 05.09.2024 Accepted date: 15.01.2025

How to cite this paper:
Bayraktar, M.Y. (2025). Detection of moving fish schools using reinforcement learning technique. *Ege Journal of Fisheries and Aquatic Sciences, 42*(1), 21-26. https://doi.org/10.12714/egejfas.42.1.03

Abstract: In this study, it is aimed to contribute to the fishing sector by determining the locations of moving fish schools. With the Q-Learning algorithm, areas where fish schools are frequently seen were marked and autonomous ships were able to reach these areas faster. With the Q-Learning algorithm, one of the machine learning techniques, areas where fish schools are abundant were determined and reward and penalty points were given to each region. In addition, the fish density matrix of the region was extracted thanks to the autonomous systems. Moreover, the algorithm can be automatically updated according to fish species and fishing bans. A different Q-Gain matrix was kept for each fish species to be caught, allowing autonomous ships to move according to the gain matrix. In short, high gains were achieved in terms of time and travel costs in finding or following fish schools by recognizing the region by autonomous ships.

Keywords: Fish school finding, multi-agent systems, reinforcement learning, Q-Learning algorithm

Öz: Bu çalışmada toplu halde hareket eden balık sürülerinin yerlerinin tespit edilerek, balık endüstrisine katkı sağlamaya odaklanılmıştır. Takviyeli öğrenme tekniğini kullanan Q-Learning algoritması ile balıkların sıkça rastlandığı bölgeler işaretlenip, otonom gemilerin bu bölgelere daha hızlı ulaşması sağlanmıştır. Makine öğrenmesi tekniklerinde olan Q-Learning algoritmasıyla, küçük karelere ayrılmış her bir bölgeye verilen ödül ceza puanlarıyla, balık sürülerinin bol olduğu bölgeler tespit edilmiştir. Ayrıca istenen bölgenin balık sürüsü yoğunluk matrisi çıkartılıp, avcı ya da araştırmacılar tarafından daha hızlıca tanınması sağlanmıştır. Sonuç olarak, bölgenin otonom gemiler tarafından tanınmasıyla birlikte, balık sürülerini bulma veya takip etmede zaman ve yol maliyeti açısından yüksek kazançlar elde edilmiştir.

Anahtar kelimeler: Balık yuvası bulma, çoklu etmenler, takviyeli öğrenme, Q-Learning algoritması

INTRODUCTION

Machine learning technique, which is one of the sub-branches of artificial intelligence, is a heuristic algorithm that analyzes the data sources of the machine and produces appropriate solutions to new problems it may encounter(Angiuli et al., 2022; Gümüş, 2016) The network structure of the machine, called artificial intelligence, updates itself according to the expected and obtained results and aims to get better results. Machine learning technique is divided into three types: supervised, unsupervised and reinforcement learning. Reinforcement learning technique works by observing the rewards in the environment. This technique achieves more successful results, especially in cases where the environmental space is very large and cannot be expressed briefly. Q-Learning algorithm is one of the methods that works using the reinforcement learning technique. In the reinforcement learning method, the aim is to trend towards the step with the highest gain by using reward and penalty points. By calculating reward and penalty at every step taken, the profit is brought closer to the best. There is no need for expert or competent instructor knowledge in this type of learning management. Generally, the trained agent (player, robot, etc.) tries to reach its goal by taking random steps. This algorithm keeps the transition matrix between steps and the gain value information of these transitions. The gain matrix values perform the learning process by updating themselves with a certain learning coefficient at each step taken.

Q-Learning is a learning method, one of the sub-learning techniques of the machine learning system, was first proposed by Watkins (Watkins,1989). In 1993, it was used by Peter Dayan in calculating return prediction and finding suitability depending on time (Dayan, 1993). At similar times, it was proposed as a method in the field of reinforcement learning in the training and development of agent robots (Lin, 1992). Towards 1995, the algorithm was combined with dynamic programming logic and benefited from real-time application studies (Barto et al., 1995).

This method performs the learning process with a specific function value. By using the Monte Carlo method and the advantages of dynamic programming, it is aimed to predict future steps with current values without creating a model (Jones and Qin, 2022; Liu et al., 2019). The Q-Learning method works by updating the reward value of each step and the step values that can be taken. The aim of this method is to maximize the reward or minimize the punishment in the long term (Watkins and Dayan 1992).

The Q-Learning method, as a type of reinforcement learning, works by updating the reward value of each action taken and the possible future actions based on the current state of the system. This approach helps in making decisions that will maximize rewards or minimize penalties over time, focusing on long-term outcomes rather than immediate feedback. One of the key features of Q-Learning is its ability to perform learning tasks without needing a model of the

environment. Instead, it uses the Monte Carlo method and dynamic programming principles to predict future actions and refine decisions based on the values of the current state (Jones and Qin, 2022; Liu et al., 2019). The learning process relies on a specific function value, where the goal is to optimize the sequence of decisions to achieve the best possible results in the future.

In essence, Q-Learning is designed to operate by updating its understanding of the environment step by step, refining its choices through continuous learning, and aiming to maximize the cumulative reward over time. This makes it a powerful tool in fields where decisions must be made sequentially, and long-term optimization is key (Watkins and Dayan, 1992). As its usage continues to grow, Q-Learning is playing an increasingly important role in both theoretical and applied research, driving innovations in various sectors.

In recent years, the Q–Learning algorithm has contributed to science, especially in areas such as marketing (Jogunola et al. 2021), autonomous robot control systems (Elallid et al. 2022), economy (Meng and Khushi, 2019) and health (Aydındağ Bayrak et al. 2022). Moreover; it has been presented as a suggestion to find solutions to problems in many fields such as information and game theory, operations research, statistics and optimization (Chapman and Kaelbling, 1991). With the reinforcement learning method, it is aimed to predict the next step without labeling the environment or data in the environment, adhering to the reward function (Jordan and Mitchell 2015; Kober et al., 2013), and the use of the algorithm has increased day by day (Parisotto, 2021).

This study aims to enhance fishing operations by combining machine learning techniques with autonomous systems, fostering cost-effective, time-efficient, and sustainable practices. It contributes to the fishing industry by utilizing the Q-Learning algorithm to streamline the identification and tracking of fish schools. Additionally, it determines areas with frequent fish activity and assesses their density. Furthermore, it enables autonomous ships to navigate to high-density fish regions more rapidly and efficiently, minimizing both time and travel expenses.

MATERIALS AND METHODS

'Q-Learning Algorithm', one of the reinforcement learning methods, is a common algorithm that performs the learning process using the reward-punishment system as mentioned above. This algorithm keeps the inter-region transition matrix and the gain value information of the transitions. It is aimed to improve the learning process by updating the gain matrix values with a certain learning coefficient when each step is taken, or a result is reached.

In this model, a fishing ship is considered to start fishing from any point in the sea or lake. It assumes that the ship is casually trying to locate and hunt fish areas. For each position the ship changes, it gives a reward value for transitions between zones. Thus, each other ship that sets out to sea updates the gain values in the location, creating a training network that can optimally find fish areas. In this way, ships use smart systems that detect fish schools autonomously.

Q-Learning algorithm, the training process is based on rewarding or punishing the values in the Q-Gain matrix. This process will be achieved by determining the region where the fish flock is located and increasing the profit value of the region that allows reaching it (Nykjaer, 2022).

$$Q_G[start, end] = (R_N[start, end] + (LCV * max))$$

Q_G: Q_{Gain}; R_N: $R_{Neighborhood}$; LCV: Learning coefficient value

As seen in the equation; “QGain[start, end]” represents the "gain" or reward for taking a certain path (from start to end) in the Q-learning algorithm. It tracks how valuable that path is based on prior experience and learning. “RNeighborhood[start, end]” matrix is the reward for being in the current "neighborhood" or region. It could represent how close the path is to the fish school. “LearningCoefficientValue” is a factor that influences how much the algorithm values past experiences versus the current observation. It helps control the learning rate. “Max” value refers to the maximum value of the Q-Gain for all possible actions from the end state. Essentially, this term captures the best possible future reward from that point (after taking action). By increasing the gain value of the roads leading to the location of the fish schools, faster discovery of the schools is ensured. This process can be compared to the process of increasing the efficiency of that path by ants in the ant swarm algorithm by constantly secreting phenomena. Thus, as the amount of gain/phenomenon increases, many of the ants will use this path, and many of the ships will move using the profitable path.

Matrix model representation of an area

The area where fishing ships hunt can be represented with squares of 50 or 100 meters in length. In this way, the autonomous ship can determine the fish density in the region every time it changes location. Thus, the gain value in that location is updated for each step taken. In fact, this is done in a similar way by ship captains. With his previous experience, the captain moves towards areas where fish are concentrated and determines the fish value in that location. However, while the captain uses his own or his immediate surroundings' experiences when making this evaluation, autonomously moving ships will have the opportunity to progress with more experience by updating the gain/learning data they receive from the common learning matrix.

In Figure 1, the sea is shown as a matrix space and the regions where fish are found are expressed representatively. Using this matrix, autonomous ships start hunting from any point and try to find the location of fish schools. Ships randomly sail around the sea while the learning process is taking place. When they reach the position where the fish schools are, they increase the gain values of that position and the positions that help them reach that position, thus creating a Q-Gain matrix.

Establishing neighborhoods between determined regions

In order to represent the search space, transitions and gain information between each region must be determined. Thus, information on inter-regional neighborhood and its earnings value can be obtained.

First of all, with the expression 'RNeighor[i,j] = 0', the permissions for passage from all regions to other regions are disabled. Then, by using certain mathematical conditions, the upper neighbor, lower neighbor and diagonal neighbor numbers are determined and the adjacency matrix value is set to 1, and you are informed that the transition can be made. As seen in Figure 2, in order to define the transition to the right (right neighborhood), it is sufficient to meet the "i==j-1" condition to define the neighborhood from the active area (i) to the area one number higher (j). However, this code makes all consecutive fields neighbors. However, as can be seen from Figure 1, although square 15 and square 16 have consecutive values, they are not neighbors. Therefore, it is necessary to remove the neighborhood for all elements to the right of the sea space, even if they are consecutive. In this case, it will be sufficient to meet the condition " (j % numberofelements) != 0) " (Figure 2).

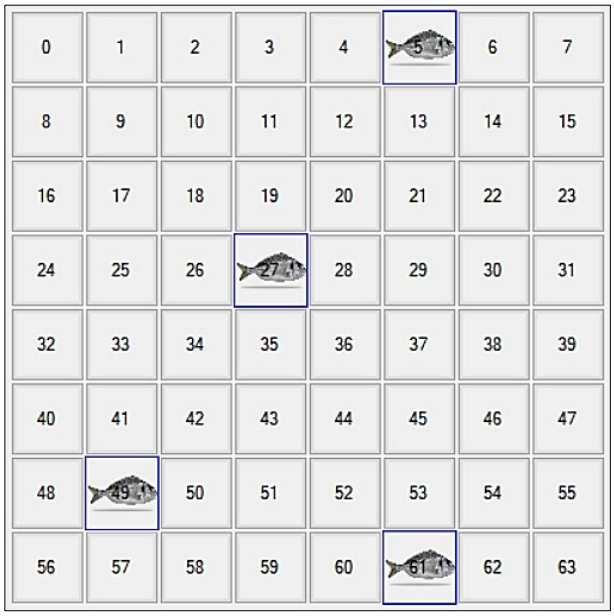

Figure 1. Representative illustration of sea space and schools of fish

```
private void NeighborhoodTransitions(int matrixSize, int i, int j)
{
    if (i == j)//no transition
    {
        RNeighbor[i, j] = 0;
    }
    if (i == j - 1 && ((j % matrixSize) != 0))// transition right,
    {
        RNeighbor[i, j] = 1;
    }
    if (i == j - matrixSize)//transition down
    {
        RNeighbor[i, j] = 1;
    }

    if ((i == j + 1 && (i % matrixSize) != 0))//transition left,
    {
        RNeighbor[i, j] = 1;
    }
```

Figure 2. Defining interregional neighborhood

Development of the interface and application

The program was developed as a desktop application in C Sharp programming language, using the Visual Studio IDE (integrated development environment). The aim is that the ship left in a random location will find schools of fish using the optimal path, looking at the earnings updated by the learning method. As seen in Figure 3, the sea the ships navigated was represented as a 7*7 array.

The location of the fish schools was determined dynamically by the person using the application. Thus, a ship departing from any point will try to find schools of fish by moving randomly (Figure 3).

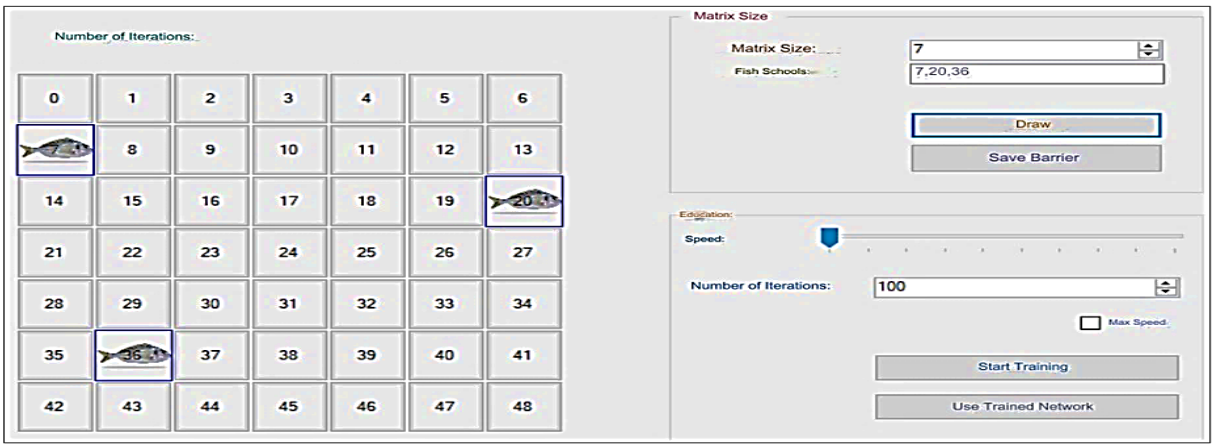

Figure 3. Application interface and training phase

Application speed

In order to make the visual of the application understandable, it would be meaningful to show the steps taken with a painting tool. Thus, people running the application will have more information about how the application works by following the steps taken. However, since the learning process will be long and laborious, you will need to accelerate after seeing a few steps in slow motion. In this way, results can be achieved quickly, without waiting too long. At the same time, since it takes a long time to update the drawing and matrix visual, the processing time will be much reduced by using the max speed control.

Uses of the trained network

Once the training process is repeated a certain/sufficient number of times, the training process is completed and ready to be used. In this case, the earnings values have reached a certain coefficient and the process of finding the school of fish is easily completed by sailing a new ship. Figure 4 shows how the ship placed in area 18 after the training found the school of fish (Figure 5). Since the herd in area 46 was closer, an orientation towards that area was quickly achieved.

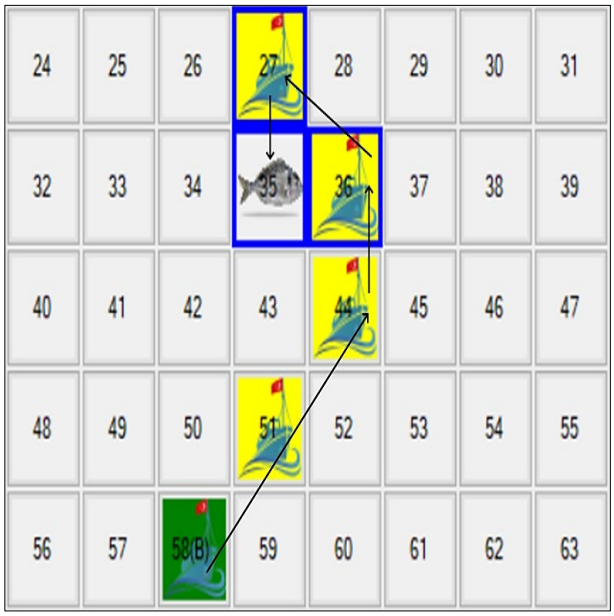

Figure 4. Fish shoal movement situation

Movement of the fish school

Generally, fish do not stay in a fixed place but move around a certain nest or area. In this sense, updating the determined fish points in certain steps will allow us to characterize the real fish schools a little better. The school of fish seen at number 35 has been allowed to pass to areas number 36 and 27 (Figure 4). Although the fish school changed location during the training process, the density of the gain matrix towards that area helped control and step in other areas where the fish roamed.

However, if the schools of fish move quickly and do not focus on a certain point, it will become more difficult for the schools to be caught by autonomous ships.

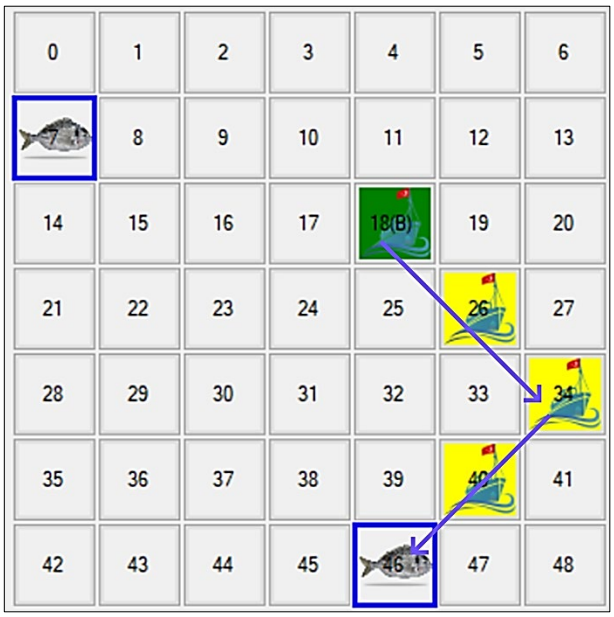

Figure 5. Finding schools of fish using trained network

Fish school density matrix

The matrix-represented region will be recognized by autonomous ships and the creation of a density matrix showing areas where fish density may be high will be beneficial for hunters. In this sense, a representative matrix showing fish density was created in Figure 6. In this matrix, the red color represents the region where fish are abundant, while the areas close to black represent the regions where fish are rare. Thanks to this density matrix, fishermen or researchers who know little or nothing about the region will have information about the region.

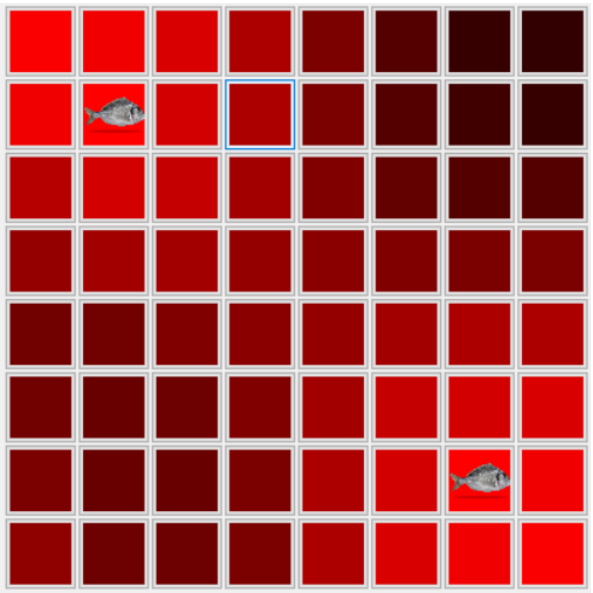

Figure 6. Fish school density matrix

RESULTS

Since fishing vessels do not know where the fish are, it will be time-consuming for them to locate the school. The vessels will have to visit almost every area one by one and detect the schools of fish. Moreover, if the school has moved a little when the fish nests are reached, it will cause the school of fish not to be found. In this sense, autonomous vessels- which are used Q-learning algorithm- will helped to detect both the area where the school is located and its movement area.

In Figure 7, the average number of steps to be taken for a region according to the matrix size is given statistically. A ship newly included in the system can reach the fish shoal in approximately 75 steps by taking random steps in an 8*8 matrix array, while autonomous ships moving with a trained network data can find the fish area in approximately 4.6 steps. These results show that defining the region with a trained network is approximately 20 times more advantageous. In addition, as the matrix size increases, the number of steps taken by ships moving with an untrained network increases exponentially, while a linear increase is observed when the trained network is used (Figure 7).

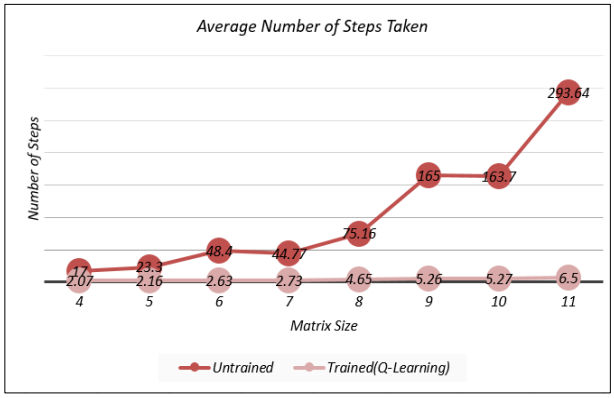

Figure 7. Average number of steps to find schools of fish

DISCUSSION

In the Q-Learning algorithm, the decision-making process is evaluated based on the reward function. The step to be taken is preferred according to the size of the profit in the reward function. In the relative Q-Learning algorithm, the step with the highest instant reward is selected by taking the previous step into account (Pandey et al., 2010).In some cases, problems caused by over-learning can occur in the Reinforcement Learning algorithm (D'Eramo et al. 2017; 2021). In addition, as stated in the article, due to events such as the movement of fish schools, the reward function may need to be corrupted or updated (Everitt et al., 2017). A solution to these problems was proposed by expanding the reward and penalty functions (Wang et al., 2020). Moreover, studies were conducted to identify potential new rewards and focus on different rewards, and the flexibility of the Q-Learning algorithm was improved (Devlin et al., 2014).In this article, by using the Q-Learning algorithm, in addition to detecting moving fish schools, it has been ensured that obstacles and areas forbidden to fish are avoided. In addition, it has been used in ATARI games, and the steps taken by the player have been optimized (Christiano et al., 2017; Van Seijen et al., 2017).

Additionally in this article, a density matrix has been obtained that will allow them to go to the nearest school in a short time. Moreover, by updating the matrix according to the school and fish type, suitable schools will be reached during suitable fishing periods. In particular, the areas prohibited from fishing will be removed from the matrix, and the protection of endangered schools will be ensured.

CONCLUSION

So, the process of discovering fish shoals, which is a difficult and time-consuming process for fishing vessels, has been made fast and effective thanks to autonomous ships. In addition, the detection of herds that may not be detected due to movement was ensured. With this method, it has become possible to quickly identify areas that are not known to fishermen or captains or that have not yet been discovered.

With the Q-Learning algorithm, areas such as seas or lakes in a region are represented in a matrix and the areas where fish schools and varieties are located are marked. In addition, the algorithm can update itself according to fish types and hunting prohibitions. By keeping a different Q-Earning matrix for each type of fish to be caught, the Q-Total-Earning matrix was obtained according to the types of fish desired to be caught, and autonomous ships were enabled to act according to this matrix. In addition, the region was recognized by autonomous ships and the fish density matrix representing that region was created.

For more complex and high-dimensional state spaces, such as large areas with fluctuating conditions, Deep Q-Learning (DQN) can be utilized. This method leverages deep neural networks to approximate the Q-value function, allowing autonomous ships to process greater volumes of data and perform more effectively in diverse settings.

Furthermore, autonomous ships can collaborate by sharing information about fish shoals and successful fishing regions. This cooperative strategy would enable a more comprehensive and dynamic understanding of fish movements, helping the ships to operate more efficiently and work together.

The Q-Learning algorithm can also be improved by incorporating adaptability, adjusting strategies based on the real-time behavior of various fish species, which may exhibit different movement patterns influenced by their environment, season, and other factors.

ACKNOWLEDGEMENTS AND FUNDING

This study did not receive any financial support, grant, or assistance from any public, commercial, or nonprofit funding organization.

AUTHOR CONTRIBUTIONS

The data collected and findings obtained for this article were provided by Mehmet Yaşar Bayraktar.

CONFLICT OF INTEREST

The authors declare that there is no conflict of interest.

ETHICAL APPROVAL

For this type of study, formal consent is not required.

DATA AVAILABILITY

The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

REFERENCES

Angiuli, A., Fouque, J.P., & Laurière, M. (2022). Unified reinforcement Q-learning for mean field game and control problems. *Mathematics of Control, Signals, and Systems*, *34*(2), 217-271. https://doi.org/10.1007/s00498-021-00310-1

Aydındağ Bayrak, E., Kırcı, P., Ensari, T., Seven, E., & Dağtekin, M. (2022). Diagnosing breast cancer using machine learning methods. (in Turkish with English abstract) *Journal of Intelligent Systems: Theory and Applications*, *5*(1), 35-41. https://doi.org/10.38016/jista.966517

Barto, A.G., Bradtke, S.J., & Singh, S.P. (1995). Learning to act using real-time dynamic programming. *Artificial Intelligence*, *72*(1-2), 81-138. https://doi.org/10.1016/0004-3702(94)00011-O

Chapman, D., & Kaelbling, L.P. (1991). Input generalization in delayed reinforcement learning: An algorithm and performance comparisons. Proceedings of the 1991. International Joint Conference on Artificial Intelligence, 726–731 pp., Sydney, Australia.

Christiano, P.F., Leike, J., Brown, T., Martic, M., Legg, S., & Amodei, D. (2017). Deep reinforcement learning from human preferences. *Advances in Neural Information Processing Systems*, *30*. http://dx.doi.org/10.48550/arXiv.1706.03741

D'Eramo, C., Cini, A., Nuara, A., Pirotta, M., Alippi, C., Peters, J., & Restelli, M. (2021). Gaussian approximation for bias reduction in Q-learning. *Journal of Machine Learning Research*, *22*(277), 1-51.

D'Eramo, C., Nuara, A., Pirotta, M., & Restelli, M. (2017). Estimating the maximum expected value in continuous reinforcement learning problems. In Proceedings of the AAAI Conference on Artificial Intelligence, *31*(1), 1846-1846.

Dayan, P. (1993). Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, *5*(4), 613-624. https://doi.org/10.1162/neco.1993.5.4.613

Devlin, S., Yliniemi, L., Kudenko, D., & Tumer, K. (2014). Potential-based difference rewards for multiagent reinforcement learning. In Proceedings of the 2014 International Conference on Autonomous Agents and Multi-Agent Systems, 165-172 pp.

Elallid, B.B., Benamar, N., Hafid, A.S., Rachidi, T., & Mrani, N. (2022). A comprehensive survey on the application of deep and reinforcement learning approaches in autonomous driving. *Journal of King Saud University-Computer and Information Sciences*, *34*(9), 7366-7390. https://doi.org/10.1016/j.jksuci.2022.03.013

Everitt, T., Krakovna, V., Orseau, L., Hutter, M., & Legg, S. (2017). Reinforcement learning with a corrupted reward channel. Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17), 4705-4713.

Gümüş, E. (2016). Q-Learning Algoritması ile Labirentte Yol Bulmak. *7*(2), 1–23. https://github.com/emrahgumus/java-q-learning-labirent.git(Erişim Tarihi: 10.09.2024)

Jogunola, O., Adebisi, B., Ikpehai, A., Popoola, S.I., Gui, G., Gačanin, H., & Ci, S. (2021). Consensus algorithms and deep reinforcement learning in energy market: A review. *IEEE Internet of Things Journal*, *8*(6), 4211-4227. https://doi.org/10.1109/JIOT.2020.3032162

Jones, G.L., & Qin, Q. (2022). Markov chain Monte Carlo in practice. *Annual Review of Statistics and Its Application*, *9*(1), 557-578. https://doi.org/10.1146/annurev-statistics-040220-090158

Jordan, M.I., & Mitchell, T.M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, *349*(6245), 255-260. https://doi.org/10.1126/science.aaa8415

Kober, J., Bagnell, J.A., & Peters, J. (2013). Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, *32*(11), 1238-1274. https://doi.org/10.1177/0278364913495721

Nykjaer, K. (2022). Q-Learning Library. https://kunuk.wordpress.com/2012/01/14/q-learning-library-example-with-csharp/ (Erişim Tarihi: 11.09.2024)

Lin, L.J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, *8*, 293-321. https://doi.org/10.1007/BF00992699

Liu, H., Bi, W., Teo, K.L., & Liu, N. (2019). Dynamic optimal decision making for manufacturers with limited attention based on sparse dynamic programming. *Journal of Industrial & Management Optimization*, *15*(2). https://doi.org/10.3934/jimo.2018050

Meng, T.L., & Khushi, M. (2019). Reinforcement learning in financial markets. *Data*, *4*(3), 110. https://doi.org/10.3390/data4030110

Pandey, P., Pandey, D., & Kumar, S. (2010). Reinforcement learning by comparing immediate reward. *International Journal of Computer Science and Information Security*, *8*(5), 1009.2566. https://doi.org/10.48550/arXiv.1009.2566

Parisotto, E. (2021). Meta reinforcement learning through memory. Doctoral dissertation, Pittsburgh, Carnegie Mellon University.

Van Seijen, H., Fatemi, M., Romoff, J., Laroche, R., Barnes, T., & Tsang, J. (2017). Hybrid reward architecture for reinforcement learning. Advances in Neural Information Processing Systems, 30. ISBN: 9781510860964.

Wang, J., Liu, Y., & Li, B. (2020, April). Reinforcement learning with perturbed rewards. In Proceedings of the AAAI conference on artificial intelligence, *34*(04), 6202-6209. https://doi.org/10.1609/aaai.v34i04.6086

Watkins, C.J.C.H. (1989). Learning from delayed rewards. Doctoral dissertation, King's College, London, UK.

Watkins, C.J.C.H., & Dayan, P. (1992). Q-learning. *Machine Learning*, *8*, 279-292. https://doi.org/10.1007/BF00992698