Date of acceptance Grade

Instructor

Reinforcement Learning in Games: A Mini Survey

Ege Can Özer

Helsinki April 17, 2017

UNIVERSITY OF HELSINKI
Department of Computer Science

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty Faculty of Science	Laitos — Institution — Department Department of Computer Science

Tekijä — Författare — Author
Ege Can Özer

Työn nimi — Arbetets titel — Title

Reinforcement Learning in Games: A Mini Survey

Oppiaine — Läroämne — Subject
Computer Science

Työn laji — Arbetets art — Level	Aika — Datum — Month and year April 17, 2017	Sivumäärä — Sidoantal — Number of pages 12 pages + 0 appendices

Tiivistelmä — Referat — Abstract

Self-learning programs have been studied and applied in many different fields; but recently, it gains more popularity and familiarity due to its breakthrough success in the game domain. Unlike the examples are taken from the real world, having the predefined set of rules and the less complicated environment in this domain provides greater flexibility to develop reinforcement learning algorithms. However, there are also several difficulties to resolve to be able to study reinforcement learning in games. We will identify these challenges and present an example approach to tackle.

In this paper, we will study the applications of reinforcement learning algorithms in the gaming context by closely focusing on four different articles. Based on the findings, we hope to propose possible improvements for future studies.

ACM Computing Classification System (CCS):
A.1 [Introductory and Survey],
I.7.m [Document and text processing]

Avainsanat — Nyckelord — Keywords
Reinforcement learning, Convergence and learning in games

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

Contents

1 Introduction

During the last decade, reinforcement learning paradigm, has gained a decent amount of popularity. Since middle 80's it has been applied in many fields to address complex problems such as in robotics, control systems, finance, and agent-based systems due to its generic formulation. In its essence, reinforcement learning looks for optimal input actions that maximizes the output states by making use of the feedbacks from the environment [SuB98]. In spite of the fact that reinforcement learning being a mature and widely-used concept, the primary reason to get such attention recently is due to breakthrough success in the gaming context [MKS13, SHM16].

Studying reinforcement learning algorithms in the gaming context contributes several definitive advantages [Tes95]. In general, modeling of the problem, implementation, and analysis of the results in real life examples are harder than the artificial ones such as games. Moreover, having simplified and controlled environment, as well as specifically defined rule sets not only allow an opportunity to explore the new variety of reinforcement learning algorithms, but also enable to have concrete evaluation measurements. Further, research results are reproducible, easy to simulate and provide test-bed for future studies. Therefore, for the given reasons, studying reinforcement learning in games can help the field to progress further and more effectively.

On the other hand, there are many difficulties to resolve to be able to study reinforcement learning algorithms in games. Implementation perspective being one of them that reinforcement learning algorithms require a proper playground for simulations. The performance measure is also another challenge in this domain, mainly because hand-labeled ground truth sets are not relevant in reinforcement learning, unlike in supervised learning. Further, deciding how to handle the complex game environment is a demanding task. Since the agent's performance and features are highly correlated, results may lead to extremely slow convergence or not at all. Given these points, we undertake to inspect the challenges of reinforcement learning algorithms in games.

In this article, we examine four existing reinforcement learning systems in the gaming domain. First, Liao et al. [LYY12] introduce a single learner to play classic platformer game, using *ε-greedy* Q-learning. Second, Amato and Shani [AmS10] demonstrate high-level reinforcement learning approach to improve the built-in AI of a turn-based strategy game. Third, Gerald Tesauro [Tes95] proposes a first successful combination of reinforcement learning and neural network system that achieves human grandmaster level. Finally, Mnih et al. [MKS13] demonstrate a deep reinforcement learning model that learns control policies directly from sensory input and accomplishes remarkable results.

The rest of this paper is organized as follows. In section 2, we discuss common challenges and approaches to the problems in reinforcement learning in games such as dealing with the abstraction of the environment or evaluation of the results. In section 3, we review existing systems in detail each with its unique characteristics and give a comparative analysis. The last section summarizes and concludes the paper.

2 Challenges and approaches

In the last decade, many research has gone into investigating reinforcement learning algorithms in the game context. Concurrently, many challenges and approaches have arisen around this topic repeatedly. In this section, we present some of those common challenges and some approaches that exist mostly in the reviewed systems [Tes95, AmS10, LYY12, MKS13].

Because there exist many different and specific problems as well as methods to tackle them, the challenges and approaches listed in this section are all related but not necessarily belong to every system in the whole research field. Generally, given matters are all common, but still some of them may not appear within another context.

2.1 Implementation perspective

Reinforcement learning researchers require robust playground to train and test the system. Nonetheless, finding the platform, considering the medium, and how to evaluate results has to be chosen in a manner that it does not misdirect the research. Because of the lack of freedom in the software, it can limit the research in a certain way. For example, researchers may end up building the whole playground from scratch. Therefore, finding appropriate software development kits are crucial even before considering the study reinforcement learning algorithms in this domain.

Fortunately, there exists plenty of variety in implementation, ranging from turn-based board games to fast-paced action games. It is not possible to have every specific game mainly due to the trademarking concerns, but some still open source the content or provide a refined framework for everyone. For example, we observe [AmS10, LYY12, MKS13] take advantage of different frameworks to train and as well as to test their system. So that using a common framework provides a reproducible and objective result. In all, having a proper playground to study reinforcement learning in the gaming context is unignorable and certainly an important aspect.

2.2 Abstracting the environment

In order to study the games properly using reinforcement learning, the very first step is abstracting the game environment in terms of Markov decision processes. This is because the modeling choices, as well as the definitions may differ based on the systems (for example, transition probability parameter does not exist in Q-learning). We are not going to give in-depth algorithmic analysis for any of the inspected systems in this paper. Therefore, we will follow one example model by Liao et al. [LYY12] to only illustrate the concept.

$$Q(s_t, a_t) \leftarrow (1 - a_{s,a})Q(s_t, a_t) + \alpha_{s,a}(r + \gamma max(Q(s_{t+1}, a_{t+1}))) \tag{1}$$

We can inspect the required elements for a model more easily directly from the equation 1. In total *five* factors need to be defined in this model. These are s_t, the state space that the agent might be in, a_t, the set of actions that agent capable of taking, r, the reward function that the agent receives after transitioning from s_t to s_{t+1} via a_t. The $\gamma \in [0, 1]$ is the discount factor that serves as a tuning parameter for present and future rewards. Finally $\alpha_{s,a} \in [0, 1]$ is the learning rate that affects agent's convergence rate. Further details are stated in [LYY12].

Liao et al. [LYY12] make use of the Mario AI framework to encode the game environment in the following way. From the figure 1 it can be inferred that nearly any condition of the character and its surroundings can be extracted. According to the article, s_t designed in a way that it requires 39 bits to encode in a state vector, such example attributes are Mario's current mode (small, big, fire), movement states, enemy distance within 3 different ranges and so on. There are 12 different a_t that Mario agent can perform, from the combinations of (LEFT, RIGHT, STAY) x (JUMPY, NOT_JUMP) x (SPEED/FIRE, NO_SPEED). In the article r is defined as combination of weighted state attribute values and the distance that agent performs from the last frame. $a_{s,a}$ and γ parameters chosen based on several try-outs that would lead to non-degenerated performance in the evaluation.

Overall, we revealed an example approach to encode a game environment to be able make use of a reinforcement learning algorithm. Moreover, there are many ways to handle the abstraction of game environment and making use of the reinforcement learning algorithms. However, we briefly categorize them in 3 possible ways, low-level, high-level and context-free abstraction based on the systems we reviewed. We refer the presented approach as low-level abstraction, where the designer can access to nearly all attributes available from the environment and the agent explores all possibilities in the search space. If one can access only the limited resources and be able to utilize inner mechanics of the system, we refer to it as high-level modeling. Compare to previous one, this abstraction approach does not have huge search space. The last one, context-free modeling encodes the environment without any context related knowledge. In the literature we observed that such modeling can achieved by using convolutional neural networks [MKS13, SHM16]. We will go over these abstraction methods as we examine systems in the further sections.

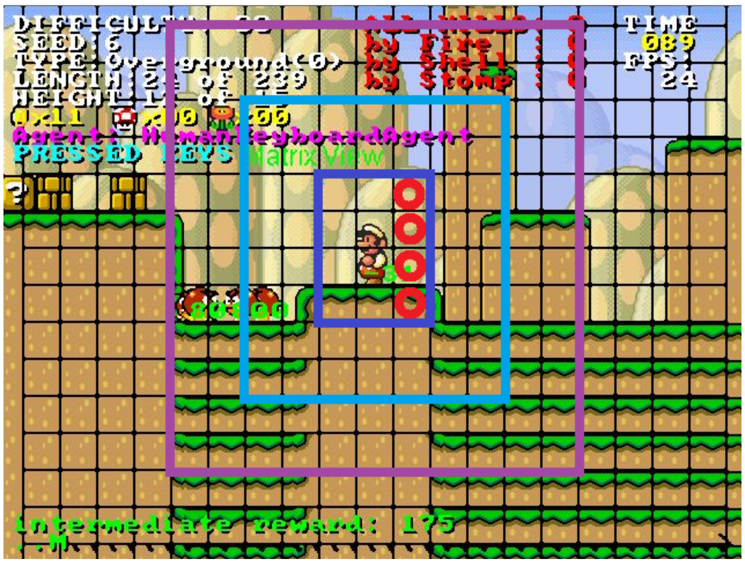

Figure 1: Mario AI framework interface provides control over several variables. In the figure, 3 level distance to nearby enemies can be seen from the square fields around the main character. Liao et al. make use of this AI framework to implement their algorithm [LYY12].

2.3 Performance measures

In reinforcement learning it is crucial to assess the effectiveness of the algorithm, but because of the nature of "learning by experience" evaluation can be challenging. Unlike in supervised learning methods, which there is a ground truth dataset to evaluate the model, reinforcement learning methods would require millions of entries to be effective. Mainly because it takes many iterations to have an intermediate to expert level of an agent, and trying to use the same procedure for reinforcement learning would be trivial and laborious task for researchers. Thus, one needs to consider other means of evaluation methods.

There are several different methods available to assess the quality of the model; however, we consider the groups that Gerald Tesauro [Tes95] states. According to this, we mention there are three different methods to evaluate reinforcement learning model. It is high-level, general enough to cover most of the systems, and actually used without a naming convention by others. Also for this paper, we will refer these groups in our comparative analysis later on.

First method is evaluating the system using automated benchmarking system. This can be seen as a computer opponent, or an objective platform for all other AIs. One obvious advantage of using these kind of automated tool is that the programs can be iterated over many times and accurate statistics can be obtained to support sufficient benchmarks.

Second method is evaluating the system against the master level human players. The major benefit of this method is that it can serve as clue about the reference point of a play performed by the system. However, as the level of complexity increases in the games, the human knowledge may be erroneous and thus unreliable to serve as a ground truth. This seems to be happen frequently in the history especially in the board games such as [Tes95, SHM16]. Also, the number of game plays against humans are more restricted and slower, specifically compared to previous one. But, the evaluation method still can aid for the given reason.

Last evaluation method is done by analyzing individual decisions by system, which is actually closely related to the second method. Moreover, it is valid to say that it also inherits nearly the same problems. Gerald Tesauro [Tes95] states that the method used to analyze the individual decisions via computer roll-outs in a backgammon game. However, one can induce this approach for any systems. The key thing is to set up candidate positions, just as in the supervised learning, and letting the computer to play out the position several times. The results can be deductible from the statistics of good and bad plays. However, it is more suitable for stochastic environments or having large state spaces, generally board games.

2.4 Further challenges

Until now we bring up common problems that every reinforcement learning systems strives. Still, underlying all possible challenges, especially after the combinations of other fields, not only out of scope of this paper but also is intractable and demanding task. For this reason, we will additionally cover only a few prominent ones from what we have studied. Briefly these are: exploration versus exploitation trade-off and various reinforcement learning challenges from deep learning point of view.

To begin with, exploitation versus exploration dilemma is truly one of the classic challenges in the reinforcement learning. Problem comes along with the agent's main learning objective. That is, the agent must be able to obtain the maximum possible reward from the actions that it has performed before. Still, it must discover such new actions that would also lead to maximize the reward in future [SuB98]. The dilemma is basically deciding on what is the perfect moment to explore and exploit, yet it has no single solution. Sutton and Barto [SuB98] demonstrate ways to balance between exploration and exploitation, including *ε-greedy* and *softmax* which all are perform near-optimally reward. Even though there exists many possible techniques in the literature, upon closer examination *ε-greedy* appears to be common approach, also in [AmS10, LYY12].

On the other hand, reinforcement learning suffers many problems from deep learning point of view which some are identified by Mnih et al. [MKS13]. First, data needs to be independent and identically distributed based on the assumptions of deep learning algorithms. In contrast to this, reinforcement learning usually consists of sequences of highly correlated actions. Moreover, vast majority of deep learning algorithms requires extensive amount of labeled data to achieve "automatic" feature extraction, whereas reinforcement learning algorithms' experience-based learning demands reward that is usually noisy and delayed. Also, agents in reinforcement learning algorithms likely to change their behaviors as they explore, this indefinite data distribution violates the fixed distribution assumption of the deep learning algorithms. Fortunately, there exists deep learning approach to overcome these issues [MKS13].

3 Review of existing systems

Until now we have seen that there are several challenges and approaches for the reinforcement learning algorithms. In order to get more intuition behind the matter, we have examined four unique systems. We demonstrate their adaptation to the challenges, and how they handle and make use of reinforcement learning in games. At the end of the section, we present a comparative analysis of reviewed articles.

Reinforcement Learning to Play Mario [LYY12]. First article authored by Liao et al. [LYY12] focuses on the design of a single agent, which plays a classic platformer game called Super Mario Bros using, *ε-greedy* Q-learning. The aim of the agent is to get the highest score. To acquire that it has to overcome various obstacles and gain rewards, such as beating enemies, avoiding gaps, or collecting items, until it reaches the finish line.

The authors used a framework called, Mario AI which helps to abstract the game environment at low-level, meaning many attributes tailored to encode into one state. The main motivation to use the *ε-greedy* Q-learning algorithm is because the state transition probabilities do not need to be known by the algorithm beforehand. Moreover, the method ensures larger state space and faster computation time. *ε-greedy* action selection extension provides balanced way to explore more states, as well as decaying learning rate ensures faster converge policy rate than fixed learning rate.

Using the same framework, training and evaluation of the agent are done under different learning parameters. The optimum one is selected based on the four performance metrics. After 5000 training iterations the agent is able to reach approximately 90% of winning probability ($\approx$ 9000 average score) of the game. On the other, evaluation of the system is only compared against baseline. Even though the tool by Karakovskiy and Togelius [KaT12] provides several benchmarking tracks, none of the performance metrics present an objective comparison to existing systems. To be able to present definitive results, a possible benchmark or a human-level comparison could be applied.

High-level Reinforcement Learning in Strategy Games [AmS10]. Second article, Amato and Shani [AmS10] study to improve the built-in AI of a turn based strategy game called Civilization IV using high-level reinforcement learning learners. The single/multiple can accomplish this by switching between high-level strategies of "leader traits", and leaving the low-level actions (technology investment, unit creation etc.) to the built-in AI. To begin with the abstraction of the game environment, states are defined in terms of the resource values, namely population difference, military power, land difference, and remaining land. Then, action space is based on the four leaders that have all possible variety of leader traits, such as financial, aggressive and so on. Lastly, reward model is based on the built-in scoring heuristic of the game, by which score difference of players used as the reward.

Article compares three reinforcement learning algorithms, particularly Q-learning, Dyna-Q, and Dyna-Q with conditionally independence assumptions between states. Evaluation of the algorithms done by comparing with the build-in AI under distinct leader match-ups. After different training episodes, overall Q-learning performs worst, the former Dyna-Q best, the latter one second due to obvious state dependency between transitions, but still performs better than Q-learning and faster than the usual Dyna-Q. Nevertheless, they all perform better than the built-in AI, which means reinforcement learning even can contribute to environments such that it is highly complex and the low-level behavior cannot be adjusted.

One side note, considering each leader has different initial advantages, match-ups could also be organized in a way that the designed agent has the less favorable one, a mirror match-up with increasing difficulties of built-in AI (Prince to Immortal level), or to test the multi-agent capability a team match-up could be planned. With the current evaluation scheme (limited leaders and match-ups) only finite number of approaches to win a game can be observed.

Temporal Difference Learning: TD-Gammon [Tes95]. So far we discussed two systems that have converged into decent results either by exploiting the heuristics and/or with the aid of built-in AI. Gerald Tesauro [Tes95] presents a temporal difference (TD) neural network learner that trains itself and automatically discovers the features where agent able to reach world-class human grandmaster level in the backgammon game.

Initially, the total number of checkers, from only one side, are vectorized according to each position on the board. This representation is then given as input pattern to neural network, which also has output patterns consisting of 4 possible outcomes for example a white win by gammon. At each time step, edge weights of neural network is calculated by using TD(λ) algorithm. By the end of game, again the "outcomes" given as the final reward for weight calculation.

In the training phase, agent makes the move for both sides and scores possible legal moves, without any heuristics, at each step in neural network. Then from neural network maximum likely outcome is selected to make the next move. This learning methodology with no heuristics managed to reach at intermediate-level of play. Yet, after adding hand-crafted features and training for about 1.5 millions of self-played game, TD-Gammon achieved master level of play [Tes95].

But surprisingly, upon further analysis by Pollack and Blair [PoB97] it is concluded that the main reason to achieve this level of play is not related with TD(λ) and neural network approach, but the self-playing scheme and the stochastic environment of the backgammon game.

Playing Atari with Deep Reinforcement Learning [MKS13]. Heretofore, game specific information would needed to be encoded such that diverse reinforcement learning algorithms could make progress towards learning. Mnih et al. [MKS13] introduce a context-free method called *deep Q-network (DQN)*, that is a combination of convolutional neural network and reinforcement learning by which the model takes information nothing but from the video input. Their approach overcome several challenges between reinforcement learning and deep learning, as we discussed in the previous section.

First of all sensory inputs taken from Atari games consist of high dimensional frames, therefore, preprocessing is crucial to reduce observation dimensionality. Then, a state is described as the sequence of observation and action pairs, because it is impossible to completely relate the situation from only one observation. However, this scheme can cause the training sample to dominate others, especially if the maximizing action is only particular one. For example, if the maximizing action that moves to right, then it dominates the other actions. For that reason, experience replay mechanism is used to smooth out the sample distributions. Moreover, Q-learning is used for the same rationale, meaning unlike the on-policy learning like SARSA, off-policy updates Q-values using the greedy action not current policy's action. Actions are executed based on ε-greedy policy, after that the reward and next frame is observed then stored in the replay memory. To alleviate highly correlated consecutive samples, each replay memory is randomly sampled and optimal action-value pair is set based on Q-learning algorithm. Finally, gradient descent is performed to calculate the local minimum of the network.

The demonstrated method by authors manages to play 7 Atari games using Arcade

Learning Environment with no game specific information other than raw images. After 50 hours of training (one training epoch takes 30 minutes of training time), their approach (DQN) outperforms all best performing methods from the literature including a neuro-evolution policy search approach. Whereas, DQN manages to surpass human performance only in 3 games. Remaining game results are far away from being close to expert human player. Even though these are only preliminary results, it is an important step towards general artificial intelligence.

Comparative Analysis. In the literature, there is no clear-cut definition to categorize reinforcement learning applications in the gaming domain under any framework. So, we compare the systems based on the abstraction approaches, as we have already defined in the earlier sections (Section 2.2). Table 1 demonstrates a comparison matrix among four reviewed reinforcement learning articles based on the criteria.

First, previous articles [Tes95, LYY12] utilize low-level abstraction in two different game genres board game and platformer game. Usually, board games implemented in this research field requires low-level abstraction, such as position of the

Criteria		Gerald Tesauro [Tes95]	Amato and Shani [AmS10]	Liao et al. [LYY12]	Mnih et al. [MKS13]
Environment	Name	Temporal Difference Learning and TD-Gammon	High-evel Reinforcement Learning in Strategy Games	Reinforcement Learning to Play Mario	Playing Atari with Reinforcement Learning
	Game	Backgammon	Civilization IV	Mario	7 Atari games
	Genre	Board game	Turn-based strategy	Platformer	Arcade games
	Single vs Multiple Agent	Single	Single&Multiple	Single	Single
	Playground	Own implementation	Civilization IV SDK	Mario AI Framework	Arcade Learning Environment
Methodology	Algorithms	Neural network + TD(λ)	Q-Learning, Dyna-Q	ε-greedy Q-Learning	DQN
	Abstraction approach	Low-level	High-level	Low-level	Context-free
Evaluation	Automated evaluation	Yes	Yes	Yes	Yes
	Human comparison	Yes	No	No	Yes
	Individual decision eval.	Yes	No	No	No
	Benchmark comparison	Yes	No	No	Yes
	Comparison against baseline	Yes	Yes	Yes	Yes
	Outcome	Grandmaster level	Improved the built-in AI	Fast convergence and 90% win prob.	Beat all existing algorithms

Table 1: The comparison matrix presents the differences among four reviewed reinforcement learning articles in the gaming context based on the evaluation criteria.

4 Conclusion

Recap everything, summary + potential future work at the end.

Sandbox learning. Context-free, model-free, application-free training scheme.

References

AmS10 Amato, C. and Shani, G., High-level reinforcement learning in strategy games. *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, 2010, pages 75–82.

KaT12 Karakovskiy, S. and Togelius, J., The mario ai benchmark and competitions. *IEEE Transactions on Computational Intelligence and AI in Games*, 4,1(2012), pages 55–67.

LYY12 Liao, Y., Yi, K. and Yang, Z., Cs229 final report reinforcement learning to play mario. Technical Report, Technical report, Stanford University, 2012.

MKS13 Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. and Riedmiller, M., Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

PoB97 Pollack, J. B. and Blair, A. D., Why did td-gammon work? *Advances in Neural Information Processing Systems*. MORGAN KAUFMANN PUBLISHERS, 1997, pages 10–16.

SuB98 Sutton, R. S. and Barto, A. G., *Introduction to reinforcement learning*, volume 135. MIT Press Cambridge, 1998.

SHM16 Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M. et al., Mastering the game of go with deep neural networks and tree search. *Nature*, 529,7587(2016), pages 484–489.

SSS15 Sukhbaatar, S., Szlam, A., Synnaeve, G., Chintala, S. and Fergus, R., Mazebase: A sandbox for learning from games. *arXiv preprint arXiv:1511.07401*.

Tes95 Tesauro, G., Temporal difference learning and td-gammon. *Communications of the ACM*, 38,3(1995), pages 58–68.