EFFICIENT GRADIENT-BASED ESTIMATION IN FINITE ECONOMICS' PROBLEMS

Radoslav NEYCHEV
Ph.D. in technical sciences, Moscow institute of physics and technology
Deputy head of department of machine learning
Arman STEPANYAN
Ph.D. student at Solar Blockchain, Moscow institute of physics and technology,
Department of Machine Learning and Digital Humanities, Chief Technical Officer

Key words: Reinforcement learning, finite economics problems, Q-learning, gradient descent, gradient-based value estimation, Bellman equation, loss function, temporal difference, outlier-splitting method, Gauss-Newton method

Introduction In reinforcement learning, very often, we use reward functions that are not differentiable, so optimizing these functions directly is not possible. Because of that, we need to estimate values, which became a core problem in RL [Sutton, et al., 2018, 1-13], [Bhatnagar, et al., 2009, 3-5]. For value estimation, we use mostly modified temporal difference (TD) algorithms such as ESARSA [Van Seijen, et al., 2009, 177-184] and double DQN [Van Hasselt, et al., 2015, 2-5], but they are not gradient-based optimizing methods, so their effectivity from the convergence perspective can't be guaranteed. It is not stable for general approximation [Baird, 1995, 30-37]. This instability was why gradient-based optimizations, such as residual gradient for MSBE and dual gradient-TD algorithms (GTD) [Dai, et al., 2017, 1458-1467], started to evolve. Even though these algorithms have general convergence properties, they are slower than TD-based algorithms in tabular and linear approximation cases [Ghiassian, et al., 2020, 3524-3534].

In this article, we analyze gradient-based algorithms' computational complexity by considering the landscape of MSBE. After we prove theoretically that MSBE is ill-conditioned. This result provides insight into why gradient-based value estimation is slowly converging. On the other hand, Gauss-Newton methods are invariant to the conditioning of loss. We can't use them directly because these methods require inversing matrix (it is computationally hard). We need to provide a linear approximation algorithm called approximate residual Gauss-Newton (RAN). It uses trace to find the proper direction and updates weights along the trace direction. RAN has a problem with double sampling, and because of that, it can be used only in deterministic environments. To properly resolve this problem, we propose RAN extensions which are GTD-based. Our method out-performs residual gradient and GTD having the same computational complexity. Then we analyze the situation of large outliers with essential information in gradients of MSBE. For that, we use one more additional block to RAN of outlier-splitting (RANS). Conducted experiment on a finite economics environment shows that RANS signify-cantly improves RG and competes with TD-based algorithms.

Methodology We define Markov Decision Process (MDP) as a tuple $< S, A, R, p, \gamma >$, where S and A are finite sets for states and actions correspondingly, R is a set of rewards (may be continuous), $p: S\ x\ A\ x\ S\ x\ R \to [0, 1]$ probability distribution function given as probability of next state if we take an action with some immediate reward from previous state, and γ is a discount factor used to calculate the discounted sum of rewards. Then we define Q-function (action-value function) as $q_\pi: S\ x\ A \to R$ is expected discounted sum of rewards from current state s acting a. Value function is defining as $v_\pi : S \to R$ and $v_\pi = E_{a\sim\pi(*|s)}[q_{pi}(s, a)]$. Mostly in production cases, we try to get a parametric estimation on Q-functions through some parameter-dependent (d dimensional vector w) function $q_w : S\ x\ A \to R$. Based on all these definitions we can define Bellman residual $\delta_w : S\ x\ A \to R$ calculated as:

$$\delta_w(s, a) = E_{s',a',r\sim p_\pi(*,*,*|s,a)}[r + \gamma q_w(s', a') - q_w(s, a)]$$

According to Bellman equations $q_w = q_\pi$ <-> $\delta_w(s, a) = 0$ for any $(s, a) \in S\ x\ A$. So, if we have some distribution D over states and actions, MSBE in this case can be defined as proxy to estimate quality of w:

$$MSBE_D(w) = E_{(s,a)\sim D}[\delta_w(s, a)^2]$$

If the distribution is online, then we just write $MSBE(*)$. If instead of paramet-rized Q-function, we use value function $v_w: S \to R$ then:

$$MSBE_D^V(w) = E_{s\sim D}[\delta_w(s)^2]$$

Where D is distribution over the states, and $\delta_w(s) = E_{a\sim\pi(*|s)}[\delta_w(s, a)]$.

Gradient-based value estimation is done through minimizing MSBE by gradient optimization. If during timestamp t two independent transitions are required, we call it double sampling. RAN algorithm iteratively updates 3 values:

$$\delta_t = R_t + \gamma q_w(S_{t+1}, A_{t+1}) - q_w(S_t, A_t)$$
$$\delta_t' = R_t' + \gamma q_w(S_{t+1}', A_{t+1}') - q_w(S_t, A_t)$$
$$w = w - \alpha\delta_t'\nabla_w\delta_t$$

Where $w_t = w$. Apart from that, we also update Gauss-Newton directions.

MBSE Loss Illness: Condition-number of symmetric square matrix H is defined as:

$$C(H) = \frac{\max\limits_{x:||x||=1} |x^T H x|}{\min\limits_{y:||y||=1} |y^T H y|}$$

In other words, it is the ratio of largest and lowest singular values. After that we define condition number for quadratic function $f(x) = x^T H x$.

$$C(f) = C(H)$$

Where H is hessian matrix of function f. We consider linear approximation case, and because MSBE is a quadratic function, then it has a condition number as well denoted C.

Theorem: *a) For any Markov decision process and with any policy, the following takes place:*

$$C \geq \frac{(1-\gamma h)^2}{4} min(\frac{l^2}{\gamma^2}, \frac{1}{(1-\gamma)^2})$$

b) For any ≥ 1 *, there exists a policy and* n *state Markov decision process such as:*

$$C \geq \frac{\gamma^4 n^2}{(1-\gamma)^2}$$

Proof: Part b is proved in [Zhang, et al., 2020, P. 1611-1619], so we will prove only part a. Let's note that fixed policy gets to Markov process with termination point. We consider having a Markov chain with n non-terminal states, let P be associated transition matrix. Let

$$A = (I-\gamma P)^T(I-\gamma P)$$

If we take tabular approximation and uniform state distribution D, we have:

$$MSBE^V(w) = \frac{w^T A w}{n}$$

We denote largest and smallest eigenvalues of A as λ_{max} and λ_{min}. It follows that $C = \frac{\lambda}{\lambda_{min}}$

We estimate from above λ_{max} and from below λ_{min}. Let's denote l_i the expected number of steps until termination for state i. In that case it is obvious that

$$l_i = 1 + \sum_{j=1}^{n} P_{ij} l_j$$

Let's make a vector from these expected numbers and denote as $l = [l_1, \dots, l_n]^T$, then we can rewrite our expectation formula:

$$l = 1 + Pl$$

Where $1 = [1, \dots, 1]^T$. From above, we get:

$$(1-\gamma P)l = l - \gamma Pl = l - \gamma(l-1) = (1-\gamma)l + \gamma * 1$$

Let $l = \frac{l_1 + \dots + l_n}{n}$ as mean of $\{l_i\}_{i=1}^n$. From Cauchy-Schwarz inequality we get:

$$\frac{||l^2||}{n} = \frac{1}{n}\sum_{i=1}^{n} l_i^2 \geq \left(\frac{1}{n}\sum_{i=1}^{n} l_i\right)^2 = l^2$$

Let's estimate largest eigenvalue firstly:

$$\lambda_{max} \geq \frac{1}{n} trace(A) = \frac{1}{n}\sum_{i}^{n} A_{ii}$$
$$= \frac{1}{n}\sum_{i=1,j=1}^{i=n,j=n} \left(I_{ji} - \gamma P_{ji}\right)^2 = \frac{1}{n}\sum_{i=1}^{n}\left((1-\gamma P_{ii})^2 + \sum_{j\neq i} \gamma^2 P_{ji}^2\right)$$
$$\geq \frac{1}{n}\sum_{i=1}^{n}(1-\gamma P_{ii})^2 \geq \left(\frac{1}{n}\sum_{i=1}^{n} 1 = \gamma P_{ii}\right)^2 = \left(1 - \frac{\gamma}{n}\sum_{i=q}^{n} P_{ii}\right)^2$$
$$= (1-\gamma h)^2$$

For the smallest one we get:

$$\lambda_{min} \leq \frac{l^T A l}{\left||l|\right|^2} = \frac{\left||(I-\gamma P)l|\right|^2}{\left||l|\right|^2} = \frac{\left||(1 - gamma\,)l + \gamma * 1|\right|^2}{||l||^2}$$
$$= \frac{(I-\gamma)^2||l||^2 + 2\gamma(1-\gamma)1^T l + \gamma^2 n}{\left||l|\right|^2}$$
$$= (I-\gamma)^2 + \frac{2\gamma(1-\gamma)nl + \gamma^2 n}{\left||l|\right|^2} \leq (I-\gamma)^2 + \frac{2\gamma(1-\gamma)l + \gamma^2}{l^2}$$
$$= \left(I - \gamma + \frac{\gamma}{l}\right)^2$$

So, connecting these 2 estimations together we can write down:

$$C = \frac{\lambda_{max}}{\lambda_{min}} \geq \frac{(1-\gamma h)^2}{(1-\gamma+\gamma/l)^2} \geq \frac{(1-\gamma h)^2}{2(1-\gamma)^2 + 2\gamma^2/l^2} \geq \frac{(1-\gamma h)^2}{4}\min(\frac{l^2}{\gamma^2}, \frac{1}{(1-\gamma)^2})$$

Which completes the proof of point a.

This theorem implies slow convergence of MSE gradient optimization methods. For example, if we set the following parameters for our MDP $\gamma = 0.99, l_{min} = 100, p = 0.1$, then from point a, we get that $C > 2000$.

Double Sampling Freedom: In RAN algorithm, we need double sampling, but such a requirement is possible only in theoretically created environments, because 2 sequential states are very likely dependent [Dabney, et al., 2014, 3-6]. To solve this issue, we use parametrization for 2 sequential samples, so the only difference compared RAN is one more additional parameter added and a separate update of that parameter.

Outlier Sampling: RAN also has a problem with rare samples with big gradients and it can't be clipped [Zhang, et al., 2019, 3-9] because of the information contained in them. Let's consider the idea of how we can do that. Let's note a task of minimization of $f_1 + \cdots + f_n$ for smooth functions $\{f_i\}_i^n$ and f_j -s gradient is k times larger than the gradient of norms of other functions for some $k > 1$. Instead of optimizing sum of functions we optimize $\frac{f_1}{k} + \cdots + \frac{f_1}{k} + f_2 + \cdots + f_n$ random based. The first updates are getting rid of outliers, while latter ones are equivalent to updating initial sum. Let's set outlier sampling probability as σ (will be used later as a hyperparameter).

RANS: Final algorithm is a combination of RAN, double sampling freedom, and outlier splitting. To update Gauss-Newton directions more effectively, we insert adaptive step-size β using the algorithm from [Kochenderfer, et al., 2019, 95-99]. Let's denote a trace vector v_t of $(\nabla\delta_t)^2$. Here is the update rule:

$$v_t = \lambda' v_{t-1} + (1-\lambda')(\nabla\delta_t)^2$$

We set $\eta \in (0,1)$ and

$$\epsilon_t = < \frac{1}{\sqrt{v_t}} * \nabla\delta_t, \nabla\delta_t >$$

Where $*$ is entrywise product, $<>$ - inner product. After we compute trace ϕ and k the following way: $\phi_{t+1} = \lambda'\phi_t + (1-\lambda')\epsilon_t$ and $k = \lfloor \epsilon_t / \rho\phi_t \rfloor + 1$. Step size is set:

$$\beta_t = \frac{\eta}{\rho\phi_t}\frac{1}{\sqrt{v_t}}$$

Let's show that RANS prevents problems that RAN has. Let actions space consist of one element $A = \{a\}$. When we approximate functions, 2 successive states ($\{S_t, S_{t+1}\}$) often doesn't differ a lot from the representation standpoint, consequently their gradients of Q function are close to each other, which implies

$$\Delta\delta_t = \gamma\Delta q_w(S_{t+1}, a) - \Delta q_w(S_t, a) \sim 0$$

The problem here is not in the fact that $\Delta\delta_t$ is small. If it was small for all the consecutive states, then we could easily solve the problem by using a step-size scheduler and constantly increasing it (that action should happen slowly as a smooth function). The problem appears when the next state is terminal and in that case that state will be far from its previous state. Despite the fact that mostly such scenarios happen with a low probability, we risk to lose too much information. The updating rule in RAN has a momentum and correction terms (corrects direction to Gauss-Newton). In an outlier case, correction term gets too large, which means that direction change to Gauss-Newton happens much faster than it should have been. As a result, tracking that direction becomes challenging. As long as, we have defined updating β_t, we have

$$\frac{1}{k} < \beta_t * \Delta\delta_t, \Delta\delta_t > = \frac{1}{k}\frac{\eta}{\rho\phi_t} < \frac{1}{\sqrt{v_t}} * \Delta\delta_t, \Delta\delta_t > \le \frac{\rho\phi_t}{\epsilon_t}\frac{\eta}{\rho\phi_t} < \frac{1}{\sqrt{v_t}} * \Delta\delta_t, \Delta\delta_t > = \eta$$

From there we get that

$$\left|\frac{1}{k} < \beta_t * (\Delta\delta_t^T m)\Delta\delta_t, \Delta\delta_t > \right| \le \eta |\Delta\delta_t^T m|$$

Consequently, outlier sampling problem doesn't occur in RANS.

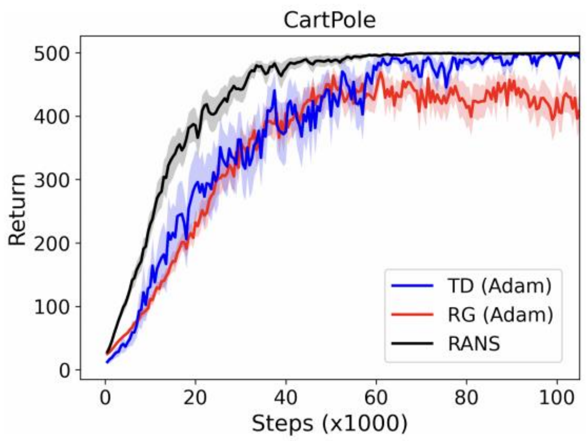

Figure 1. Performance of models on economical decision-making task CartPole, where Q-values are learnt

Literature review MSBE illness and the challenges that appeared with it were a complex challenge to solve. In the work [Wang, et al., 2021, 3-6] poor conditions related to MSBE were examined in detail. Their research was concentrated on Markov chains with a fixed length. In their work, it was found out that condition-number of chains increases quadratically to the length of that chain. Second important finding was that there is one more dependence and it is reverse quadratic related to γ. These findings provide a clear understanding of current problems with the corner cases.

Gradient-based value estimation methods are powerful, but they lack computationnal efficiency. For details, one can refer to Baird's article, where it is revealed that each update of RG method is an update of 2 components. First one is called TD component and is responsible for keeping a right direction during the optimization and wrong direction component. Here the main idea was to reduce second components' influence. The experiments have shown that the strategy clearly works on early epochs of training but after gaining a certain momentum, it is not effective at all.

The open question on how to get an adequate gradient value estimation had other alternatives proposed. For example, Gauss-Newton method was in [Gottwald et al., 2022, 1-5]. This development wouldn't have been done if Newton's method wasn't studied in [Sun, et al., 2015, 3-7], where authors discussed minimization problem of MSBE and solving it through Newton method.

Natural gradient-based methods for value estimations were presented in [Kakade, 2001, 1-3], where the author proposed a method to solve basic problems without complex data. Along with them a similar architecture to RAN was proposed. The model was similar from the algorithmic perspective to RAN but was 2 times larger than the original

RAN. In both articles the outlier sampling problem wasn't considered as a serious one and as an implication – wasn't solved.

To solve the problem with outliers, there were numerous tries. The most popular of them are [Karampatziakis, et al., 2010, 3-6] and [Tian, et al., 2019, 64-76], where step scheduling was proposed to decrease the effect of outliers. With outliers, there is a higher chance that we got a bigger direction change, so fixing of it will take longer rather than without outliers (which means that next state is not terminal and the gradient values' difference is not too big).

Scientific novelty Compared to previous works related to gradient-based value estimations, mostly all the challenges have unpleasant corner cases, which can't be handled through provided methods in real cases. Benchmark solutions like RAN effectively give value estimation in theoretical cases, when the data is not noisy and is consistent, which in production tasks never happens. That is the main reason why TD is still commonly used as an easy alternative to complex gradient-estimation approaches, because it behaves well on noisy data. This research shows that not all the complex methods have problems with corner cases, it theoretically proves how double and outlier sampling can be included in the common value-estimation solution, without making the method more complex. The newly provided gradient-based value estimation method – RANS gives an alternative to TD, which was one and only baseline for noisy and inconsistent data. It works faster than TD and convergence happens smoother.

Not only as a new method but also as a new way of considering both gradient-value and value estimations, RANS provides a way to improve already existing algorithms that work on unbiased, less noisy data. Overall, RANS is not only a new gradient-value estimation algorithm, but a new way of generalization of solutions with outlier sampling cases.

Analysis After analyzing previously developed methods and showing how RANS is better compared to its competitors, we will experimentally show the results that we have proven above. As baseline comparison methods we are going to take TD and RG. RG is not able to handle outlier sampling problem but provides good results otherwise. TD is faster, with worse quality but handles outlier sampling.

RANS has the following tuple of hyperparameters $< \alpha, \eta, \rho, \lambda, \lambda', \sigma >$. We set for experiment the following values as default without effecting performance: $\eta = 0.2$, $\rho = 1.2$, $\lambda = 0.999$,

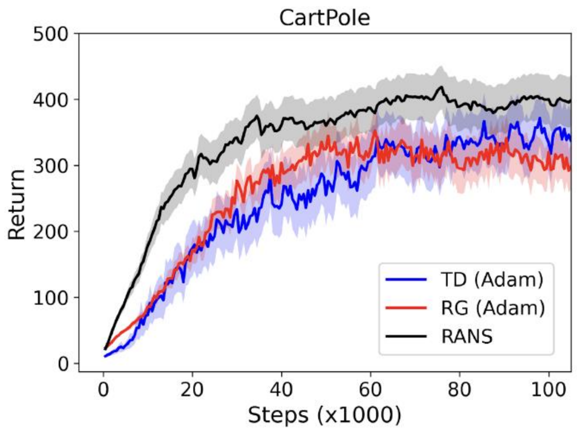

Figure 2. Performance of models on economical decision-making task CartPole, where average of expected Q-values are learnt

$\lambda' = 0.9999$, $\sigma = 0.02$. So, the only hyperparameter to optimize is α like TD and RG algorithms. Computational complexity in one iteration in the worst case is 2 times more than TD and RG.

Next step is formally defining the environment. Here, we are going to use a finite economic game "CartPole", which has the same naming as the balance game. The game itself can be considered as a benchmark to test the quality of RL-based models, especially in cases where the algorithm is not complex and doesn't need a lot of data to feed. In finite economy case, we have 2 types of states – stable and unstable. There are 2 types of instabilities, called negative and positive. The distinction between these two from task's goal perspective doesn't change, but for the understanding, we need to mention that there is a big difference on how economics can become unstable and based on type scores are different.

The player has a finite set of actions. There are three choices that in each state can be taken. First action pushes economy to the negative side of instability second action pushes the economy to positive side of it. Third action is more passive than previous 2 and it is designed to be more inactive and mostly causes not a dramatic move of economy stability score to neither one nor other direction. Fortunately, if the next state is terminal, then dramatic move may happen even in case of third type of action. It is logical because sometimes doing nothing to your economy can make situations worse or better.

The objective of this environment is not just about reacting to the current policy. It is created as a new RL task, where the model should find an optimal policy – sequence of actions, which gives the economy stability for some amount of time. If there are state changes with a low probability that may cause outlier sampling, this environment fully matches the case to use RANS and make sure that theoretical results have experimental confirmation too.

First experiment is taken on a basic economics task with discrete environment [Charpentier, et al., 2020, 29-34]. We remind that the goal is to keep the economy stable between negative and positive instability during k consecutive steps. The final model is a neural network with 128 hidden units and Leaky-ReLU for actions distribution and Sigmoid for Q-function distribution. Results can be viewed in figure 1. where we compare our algorithm with TD and RG using Adam optimizer.

For the second experiment we used a multi-layer network with 256 hidden Leaky-ReLU activations, so we can learn action-values (AV). Actions are being chosen using sigmoid distribution on AV. The network for Q function is being trained by three algorithms, as the first one: TD, RG and RANS. Basic RANS has an advantage over its basic competitors because of adaptive step size updating, and because of that we are using TD and RG with Adam optimizer.

For each of the experiments we performed 200 randomly generated data samples with random seeds. We are estimating expected values of Q function, so we took an average of 200 environment simulations each 400 steps. The results can be viewed in Figure 2, where an average of expected returns on random samples is plotted.

There is a need to mention separately that once the algorithm achieves a score of 400 and stabilizes, the following happens. Once the algorithm reaches a certain equilibrium it starts to forget actions that were making summary reward higher. The absence of failures during some period causes the model to have catastrophic forgetting problem.

To solve of this problem, during the experiment the following strategy was used. Recognizing the fact that with specific scenarios that causes random samples, the environment won’t be able to pass enough information to the model, which will cause it to have abovementioned problem. For that reason, 60 worst average return scenarios are dropped off. This helps the algorithm to concentrate on a certain set of challenges, preventing the rest from affecting the performance of the model.

To eliminate catastrophic forgetting at least partly, we used a replay buffer. It serves as a memory reserve, where we store previous replay experiences which allows us to revisit them later. It is a way to force the model to learn from previous experience which allows to overcome catastrophic forgetting. Anyway, to overcome this problem fully, we need to use updates through batches, which allows us to replay the algorithm with multiple sessions at the same time.

Ideal way of excluding catastrophic forgetting in case of big size of configuration space we need to increase replay buffer size significantly, because it will ensure more extensive range of experiences stored.

For both experiments there is a specific list of parameters that were used to ensure that the model is consistent and generalizable for any randomly generated finite environments. The choice of parameters is done through Grid Search on a set of parameters with size 2000, running them for small number of randomly generated environments, and getting the best results from that set. For each of below 3 algorithms the process is being repeated and as a result here are the parameters with their values that each algorithm uses as an optimal:

- For TD, sigmoid coefficient 0.01 with Adam optimizer with step-size 0.3 was used.
- For RG, sigmoid coefficient 0.005 with Adam optimizer with step-size 0.3 was used.
- For RANS, sigmoid coefficient 2, $\alpha = 0.001$, and the rest of the parameters were set to default.

Above experiments with set parameters show that RANS overperforms from the quality perspective both TD and RG algorithms, and from the speed perspective it is better than RG. On the other hand, it is slower than TD because each iteration RANS has adaptive step-size updating rule, which isn't included in TD but on the other hand, RANS handles outlier cases with higher accuracy, and the model doesn't lose its consistence if probability of outlier samplings increases.

As a further analysis for future works, we haven't covered off-policy cases, showing how RANS work experimentally only on-policy way. Of course, it can be easily transferred from on-policy to off-policy applying any importance sampling method. Also, there is a direction to test this method on an environment, which configuretion set's power is continuous and compare RANS to other baseline algorithms as well as applying unbiased gradient estimate instead of biased one. On continuous environments it would be also interesting to prove alike theoretical results as we did here to apply the same methods here and exclude corner cases and make the models for any types of environments (consequently, continuous economics problems) stable.

Conclusion In this article we explained the most common challenges that MSBE's gradient-based value estimations have. The main issue that was identified was the slow convergence. We theoretically explained why gradient-based value estimations for MSBE currently are slowly converging: double sampling and outlier sampling. Both problems cause inconsistency during the estimation process and add noise to it. Because of that overall efficiency of method becomes not competitive.

To fix that, we considered RAN algorithm, which provides robustness in many common cases. After that, we considered solutions for both double and outlier samplings and combined them with RAN getting more advanced algorithm called RANS. It includes RAN and corner case solutions lowering the noise and inconsistence for both problematic samplings. Theoretically we proved that RANS is improving overall convergence and solved problems of double sampling and outlier sampling.

Further taken experiments with RANS, particularly on an economical problem with a finite number of states and 3 decisions' environment has shown that it improved previous baseline algorithms like RG and TD. Compared to TD, RANS has shown competitive results, showing a potential to be used in certain scenarios, where the generated samples include a lot of noise.

This research provides a new way of looking at challenges in gradient-based estimations, where a smart combination of corner cases and ordinary solutions can give a significant enhancements and RANS as a perfect example, provides promising results for the future research and applications in finite decision-making environments.

References

1. Sutton R.S., Barto A.G., Reinforcement learning: An introduction, MIT Press, P. 1-13, 2018.
2. Bhatnagar S., Sutton R.S., Ghavamzadeh M., Lee M., Natural actor-critic algorithms, Automatica, P. 3-5, 2009.
3. Van Seijen H., Van Hasselt H., Whiteson S., Wiering M., A theoretical and empirical analysis of expected Sarsa, IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning, P. 177-184, 2009.
4. Van Hasselt H., Guez A., Silver D., Deep Reinforcement Learning with Double Q-learning, Association for the Advancement of Artificial Intelligence, P. 2-5, 2015.
5. Baird L., Residual algorithms: Reinforcement learning with function approximation., Proceedings of the International Conference on Machine Learning, P. 30-37, 1995.
6. Dai B., He N., Pan Y., Boots B., Song L., Learning from conditional distributions via dual embeddings, In Proceedings of the International Conference on Artificial Intelligence and Statistics, P. 1458-1467, 2017.
7. Ghiassian S., Patterson A., Garg S., Gupta D., White A., White M., Gradient temporal-difference learning with regularized corrections, In Proceedings of the International Conference on Machine Learning, P. 3524-3534, 2020.
8. Zhang S., Boehmer W., Whiteson S., Deep residual reinforcement learning, Proceedings of the 19th International Conference on Autonomous Agents and Multiagent systems, P. 1611-1619, 2020.
9. Dabney W., Thomas P., Natural temporal difference learning, Proceedings of the AAAI Conference on Artificial Intelligence, V. 28, P. 3-6, 2014.

10. Zhang J, He T., Sra S., Jadbabaie A., why gradient clipping accelerates training: A theoretical justification for adaptivity, arXiv preprint, P. 3-9, 2019.
11. Kochenderfer M. J., Wheeler T. A., Algorithms for optimization, MIT Press, P. 95-99, 2019.
12. Wang. Z.T., Ueda M., A convergent and efficient deep Q network algorithm, arXiv preprint, P. 3-6, 2021.
13. Gottwald M., Shen H., On the compatibility of multistep lookahead and hessian approximation for neural residual gradient. In Proceedings of the Multi-disciplinary Conference on Reinforcement Learning and Decision Making, P. 1-5, 2022.
14. Sun W., Bagnell J. A., Online bellman residual algorithms with predictive error guarantees, P. 3-7, 2015.
15. Kakade S. M., A natural policy gradient, In Advances in Neural Information Processing Systems, volume 14, P. 1-3, 2001.
16. Karampatziakis N., Langford J., Online importance weight aware updates, arXiv preprint, P. 3-6, 2010.
17. Tian T., Sutton R., Extending sliding-step importance weighting from supervised learning to reinforcement learning, In Proceedings of the International Joint Conference on Artificial Intelligence, P. 64-76, Springer, 2019.
18. Charpentier A., Romuald E., Remlinger C., Reinforcement Learning in Economics and Finance, arXiv preprint, P. 29-34, 2020.

Radoslav NEYCHEV, Arman STEPANYAN

Efficient gradient-based estimation in finite economics' problems

Key words: reinforcement learning, finite economics problems, Q-learning, gradient descent, gradient-based value estimation, Bellman equation, loss function, temporal difference, outlier-splitting method, Gauss-Newton method

Gradient-based methods (GBMs) for estimating values have stability properties, but the temporal difference (TD) and its modifications' learning methods are much faster in reinforcement learning (RL). We prove a theorem stating the cause of GBMs being slow and show that the mean square of Bellman error (MSBE) is a not appropriate loss function if its second derivative matrix has a significant determinant. To resolve the problem with MSBE on GBMs we propose residual approximate Gauss-Newton with an outlier-splitting method (RANS). This method adds outlier-spitting on gradient methods and learning adapter ideas to residual gradient methods making them more stable from the estimation perspective. We show that it is faster than its residual competitors having the same computation time and competing with TD on the baseline problem of economics (CartPole) in RL that we tested. Further analysis and future contributions are considered to make the result of these methods better on any types of economics, which can be built as a finite set of state-action pairs. After proving that claim, GBMs can be used as a baseline in any types of RL-based problems including finite economics problems.