On the Compatibility of Multistep Lookahead and Hessian Approximation for Neural Residual Gradient

Martin Gottwald
Chair for Data Processing,
Technical University of Munich,
Arcisstr. 21, 80333 Munich, Germany
martin.gottwald@tum.de

Hao Shen
fortiss,
Forschungsinstitut des Freistaats Bayern,
Guerickestr. 25, 80805 Munich, Germany
shen@fortiss.org

Abstract

In this work, we investigate, how multistep lookahead affects critical points of Residual Gradient algorithms. We set up a compound Bellman Operator for k consecutive transitions similar to TD(λ) methods and analyse the critical points of the associated Mean Squared Bellman Error (MSBE). By collecting per state multiple successors at once, one can create a more informative objective without increasing the requirements for function approximation architectures. In an empirical analysis, we observe that if one uses Hessian based optimisation to minimise the MSBE, it is not possible to benefit from larger lookahead. Already high convergence speeds and overall lower final error of a Gauss Newton algorithm seem to prevent further improvements by larger lookahead. Only first order gradient descent shows a significant boost in convergence for larger k, emphasizing the importance of multiple steps for existing and successful Deep Reinforcement Learning algorithms. Our results suggest that there are still open questions for Neural Network training in Reinforcement Learning applications.

Keywords: Critical Point Analysis, Gauss Newton Algorithm, Mean Squared Bellman Error, Multistep Lookahead, Residual Gradient

Acknowledgements

Supported by Deutsche Forschungsgemeinschaft (DFG) through TUM International Graduate School of Science and Engineering (IGSSE), GSC 81.

1 Introduction

Reinforcement Learning (RL) is a general approach to solve sequential decision making problems. When facing large or even continuous state spaces, Value Function Approximation (VFA) must be used as inevitable and effective tool [Bertsekas, 2012, Sutton and Barto, 2020]. Recent research efforts have focused more on Non-Linear Value Function Approximation (NL-VFA) methods, which use Neural Networks (NN), e.g. in the form of Multi-Layer Perceptrons (MLP), as approximation architecture. Impressive successes of NNs in solving challenging problems in pattern recognition, computer vision, speech recognition and game playing [LeCun et al., 2015, Yu and Deng, 2015, Mnih et al., 2015, Silver et al., 2017] have further triggered increasing efforts in applying NNs to VFA [van Hasselt et al., 2016].

A key ingredient in well performing RL algorithms, which make use of NL-VFA, are n-step returns [Mnih et al., 2016] or full TD(λ) like multistep lookahead mechanisms [Schulman et al., 2016, 2017]. This means, instead of using (s, a, r, s', a') tuples, one incorporates multiple consecutive transitions at once for learning. Due to the accumulated amount of information in the transition data, algorithms work more reliably with better convergence or higher quality approximations.

In this work, we investigate and analyse the impact of multistep lookahead on Residual Gradient (RG) algorithms [Baird III, 1995], especially when using a Gauss Newton (GN) algorithm for optimisation [Gottwald et al., 2021]. We extend the loss for training with multistep lookahead and derive its differential map. The influence of larger lookahead on critical points is outlined and empirical experiments regarding convergence and generalisation performance are described.

2 Method

We use a Markov Decision Process (MDP) with a continuous state space $\mathcal{S}$, discrete action space $\mathcal{A}$, one step reward r and discount factor γ to model the decision making. In this work, only deterministic transitions are considered to avoid the *Double Sampling* issue when working with RG algorithms [Baird III, 1995]. The goal is to learn an optimal policy π, which maps states to actions in such a way that an accumulated reward signal is maximised. To assess its quality, one employs the value function $V_\pi \colon \mathcal{S} \to \mathbb{R}$, which is defined as

$$V_\pi(s_0) = \sum_{t=0}^{\infty} \gamma^t r(s_t, \pi(s_t), s_{t+1}). \tag{1}$$

Once V_π is known, it can be used to define an improved policy. The value function under policy π is the unique fixed point of the Bellman operator T_π

$$V_\pi(s) = (\mathrm{T}_\pi V_\pi)(s) = r(s, \pi(s), s') + \gamma V_\pi(s') \quad \forall s \in \mathcal{S}, \tag{2}$$

where s' is the successor of s when executing the action $a = \pi(s)$. For an arbitrary $V \colon \mathcal{S} \to \mathbb{R}$, the squared difference of the left and right hand side in Eq. (2) can be used as an objective function to convert the fixed point problem into an optimisation task. Multistep lookahead is now obtained by k repeated applications of the operator. This can either be done by summing all powers of T_π with exponential weighting to obtain the TD(λ) method, or by using only finite many powers and combining them as simple average. In the latter case, one defines T_π for several steps k by introducing the compound operator

$$\mathrm{T}_\pi^{(k)} = \frac{1}{k} \sum_{i=1}^{k} \mathrm{T}_\pi^i \,. \tag{3}$$

As an example, three step lookahead results in $\mathrm{T}_\pi^{(3)} = 1/3 \left(\mathrm{T}_\pi^1 + \mathrm{T}_\pi^2 + \mathrm{T}_\pi^3\right)$ with $(\mathrm{T}_\pi^2 V)(s) = r(s, \pi(s), s') + \gamma r(s', \pi(s'), s'') + \gamma^2 V(s'')$ and $(\mathrm{T}_\pi^3 V)(s) = r(s, \pi(s), s') + \gamma r(s', \pi(s'), s'') + \gamma^2 r(s'', \pi(s''), s''') + \gamma^3 V(s''')$. Higher powers of T_π are defined similarly. The fixed point of $\mathrm{T}_\pi^{(k)}$ is still V_π as defined in Eq. (1) and yields the condition $V_\pi(s) = (\mathrm{T}_\pi^{(k)} V_\pi)(s)$. Hence, $\mathrm{T}_\pi^{(k)}$ also allows for a conversion of the fixed point iteration into a root finding problem by defining

$$\delta(s, s', s'', \ldots, s^{(k)}) \coloneqq V(s) - (\mathrm{T}_\pi^{(k)} V)(s). \tag{4}$$

The current state s and its k successors s', s'' until $s^{(k)}$ are collected from the dynamical system under control according to the current policy. After collecting a batch of N start states s_i, which are uniformly distributed in the whole state space, and their successors $s_i^{(j)}$, one can approximate a solution to the root finding problem with the MLP $f \colon \boldsymbol{\mathcal{W}} \times \mathcal{S} \to \mathbb{R}$ by minimising the multistep Neural Mean Squared Bellman Error (NMSBE)

$$\mathcal{J}(\mathbf{W}) \coloneqq \frac{1}{2N} \sum_{i=1}^{N} \Big(f(\mathbf{W}, s_i) - \sum_{j=0}^{k-1} (k-j) \frac{\gamma^j}{k} r_{ij} - \sum_{j=1}^{k} \frac{\gamma^j}{k} f(\mathbf{W}, s_i^{(j)}) \Big)^2 = \frac{1}{2N} \Delta_\pi^{(k)}(\mathbf{W})^\intercal \Delta_\pi^{(k)}(\mathbf{W}), \tag{5}$$

where the expression $\Delta_\pi^{(k)}(\mathbf{W}) \in \mathbb{R}^N$ takes the form

$$\Delta_\pi^{(k)}(\mathbf{W}) \coloneqq F(\mathbf{W}) - \frac{1}{k} R_\pi^{(k)} - \frac{1}{k} \sum_{j=1}^{k} \gamma^j F^{(j)}(\mathbf{W}) \tag{6}$$

by collecting for all i the evaluation of f for the states as vector $F(\mathbf{W}) \coloneqq [f(\mathbf{W}, s_1) \dots f(\mathbf{W}, s_N)]^\mathsf{T} \in \mathbb{R}^N$. Further, we use the abbreviation $r_{ij} = r(s_i^{(j)}, \pi(s_i^{(j)}), s_i^{(j+1)})$ and accumulate all reward terms in $R_\pi^{(k)}$. Critical points of $\mathcal{J}(\mathbf{W})$ are now characterised by the equation

$$\nabla_\mathbf{W} \mathcal{J}(\mathbf{W}) = \frac{1}{N} \underbrace{\Big(G(\mathbf{W}) - \frac{1}{k} \sum_{j=1}^{k} \gamma^j G^{(j)}(\mathbf{W}) \Big)^\mathsf{T}}_{=:\widetilde{G}(\mathbf{W}) \in \mathbb{R}^{N \times N_{net}}} \Delta_\pi^{(k)}(\mathbf{W}) = 0, \tag{7}$$

where $G(\mathbf{W})$ is the differential map of F with respect to the parameters $\mathbf{W}$ evaluated at all start states. With $G^{(j)}(\mathbf{W})$ we denote the same item but use the j-th successor states as input. We see that $\mathrm{T}_\pi^{(k)}$ provides a richer objective for the optimisation problem than $\mathrm{T}_\pi^{(1)}$ without increasing the required number of network parameters to allow for a full rank of $\widetilde{G}(\mathbf{W})$. A full rank of $\widetilde{G}(\mathbf{W})$ is important to ensure that any critical point achieves zero error, since in this case $\Delta_\pi^{(k)}(\mathbf{W}) = 0$ is the only way to satisfy the critical point condition of Eq. (7). This property is important, as it removes local minima and saddle points. In practical applications, the rank of $\widetilde{G}(\mathbf{W})$ is full, since sampling and numerical inaccuracies are present. If exact learning is out of reach, i.e., $\Delta_\pi^{(k)}(\mathbf{W}) \neq 0 \; \forall \; \mathbf{W} \in \mathcal{W}$, the objective is free of critical points. One can run a descent algorithm as long as possible.

There remains an open problem in the definition of $\Delta_\pi^{(k)}(\mathbf{W})$, which exists due to the sampling of states and their successors. We hypothesize that with only one step transitions, it is relatively easy to sample states with large pairwise distances. Especially in high dimensional spaces, this could be rather common. In such a case, an MLP could approximate a function, where $\Delta_\pi^{(1)}(\mathbf{W})$ becomes zero without being everywhere close to V_π, because for some states the transition information to remaining parts of the state space might not be present in the sampled data. Thus, the sampling based loss of Eq. (5) has uncontrollable degrees of freedom and for $k = 1$, one can even construct examples to demonstrate this issue. We expect that a multistep lookahead algorithm helps here. Due to the use of trajectories, one ensures that there are proper transitions between states and their various successors available. The chance that parts of the state space are not sufficiently connected is reduced. Hence, an MLP has less possibilities to approximate functions, which would shrink the length of $\Delta_\pi^{(k)}(\mathbf{W})$ to zero without becoming close to V_π.

From a theoretical perspective, one would expect a better performance when switching to multistep methods. This expectation is supported by the empirical behaviour of existing algorithms. When using k-step returns T_π^k for training, e.g. as done in [Mnih et al., 2016], or when employing full TD(λ)-like methods as in [Schulman et al., 2016, 2017], one obtains well performing algorithms. Multiple steps can compensate the convergence issues of Semi-Gradient algorithms, which exist due to the missing derivates of the MLP for successor states with respect to the parameters. Because expressions receive higher powers of the discount factor, they have less impact. If the lookahead becomes large, the terms vanish naturally such that omitted dependencies no longer cause any harm. Opposed to that, we observe that a Gauss Newton Residual Gradient algorithm converges already without problems with only single-step lookahead. Hence, we argue that a beneficial impact of multistep-lookahead depends directly on the algorithm and problem at hand. To clarify our considerations, we compare first order and Hessian based descent algorithms and test, whether different values for k create a meaningful difference.

3 Experiments

We use the Mountain Car environment from [Brockman et al., 2016] with additional transitions from the goal region to the valley to obtain an infinite horizon MDP. Training is performed in a batch setting with both a GN and first order only RG algorithm. For the first, we use a constant learning rate $\alpha = 0.1$, for the latter $\alpha = 0.01$. Start states are sampled uniformly from the whole space. We use $N = 300$ samples for training. Additional states, which are arranged on a grid with a high resolution (500×500), serve as test dataset. For representing value functions, we take an MLP consisting of two hidden layers with ten units each and employ Bent-Id activation functions. The input layer accepts two dimensional state vectors. The output is scalar with linear activation. Initial parameters are drawn elementwise uniformly from the interval $[-1, 1]$. We set the discount factor to $\gamma = 0.99$ and the policy for evaluation to accelerating in the direction of movement. For the lookahead we consider $k \in \{1, 2, 3, 4, 5\}$. We show results for 25 repetitions, where we randomise the training data and initial network parameters in each run.

When visualising the descent behaviour in Fig. 1, we observe a missing impact of $\mathrm{T}_\pi^{(k)}$ with increasing k for GN based optimisation. Using multiple transitions during training does not help with convergence. Opposed to that, when training with a first order only RG algorithm, the convergence speed can be enhanced significantly by varying k. The well-known slow convergence of RG algorithms for $k = 1$ vanishes with larger values. For both first order and Hessian based optimisation, a stepwise larger k results in worse final training errors. This could be explained by the fact that an MLP with identical capacity must fit a more complex objective.

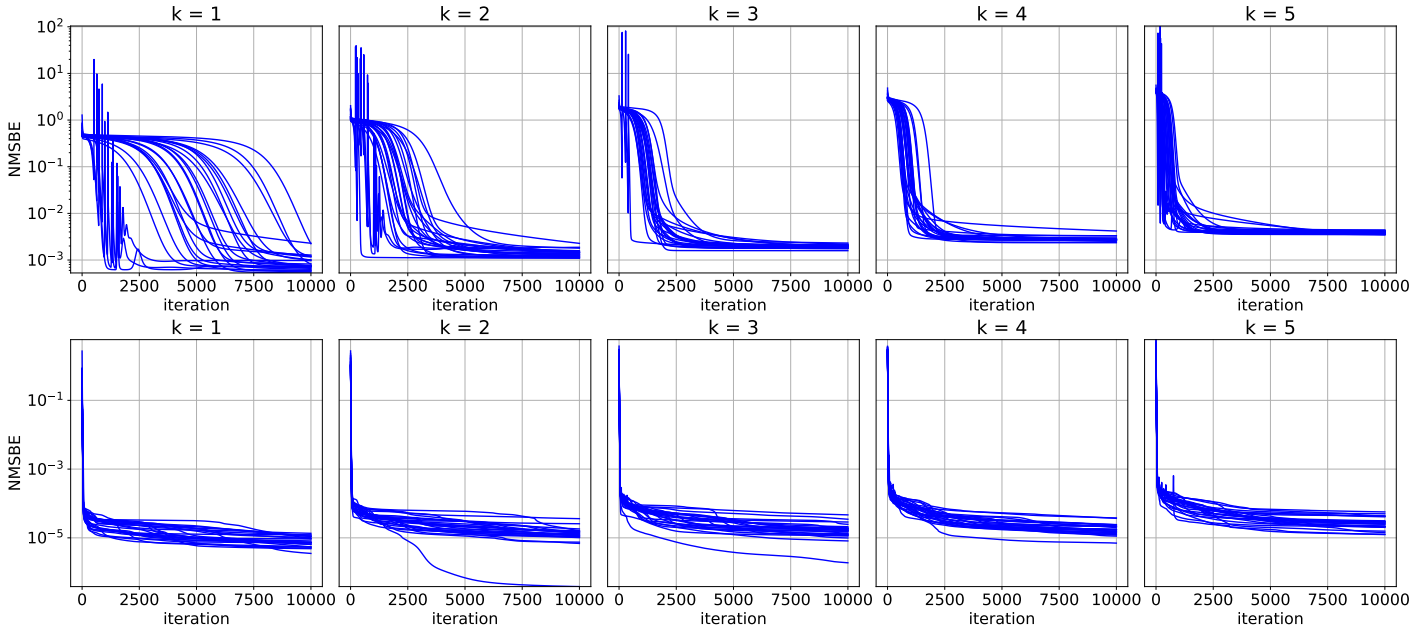


Figure 1: The impact of multistep lookahead onto first order and Hessian based RG algorithms. Only first order optimisation profits from larger k. For both methods, the final achieved training error increases slightly with bigger k. **Top:** First order optimisation. **Bottom:** Gauss Newton algorithm.

To compare training outcomes fairly, we evaluate the MLP at the end of training with the held out test set. Figs. 2a and 2b show the final test errors for all k for both optimisation methods. We see, that the average test error increases with k for first order gradient descent, implying that the solutions become worse. This matches the increase of training errors. For Hessian based optimisation, we observe a similar rise, but not as pronounced. Interestingly, the best approximated value function for $k = 5$, which is shown in Fig. 2f, possesses a qualitatively better shape despite a higher test error than the best value function for $k = 1$ in Fig. 2e. The value functions for first order optimisation are for all k almost identical to that of Fig. 2d and do not reflect the details of the actual value function V_π, which is obtained from Monte Carlo methods and shown in Fig. 2c. We conclude that using a test dataset is not a perfect reliable assessment. This could be due to the aforementioned open problems regarding the objective of Eq. (7) and its critical point condition.

4 Conclusion

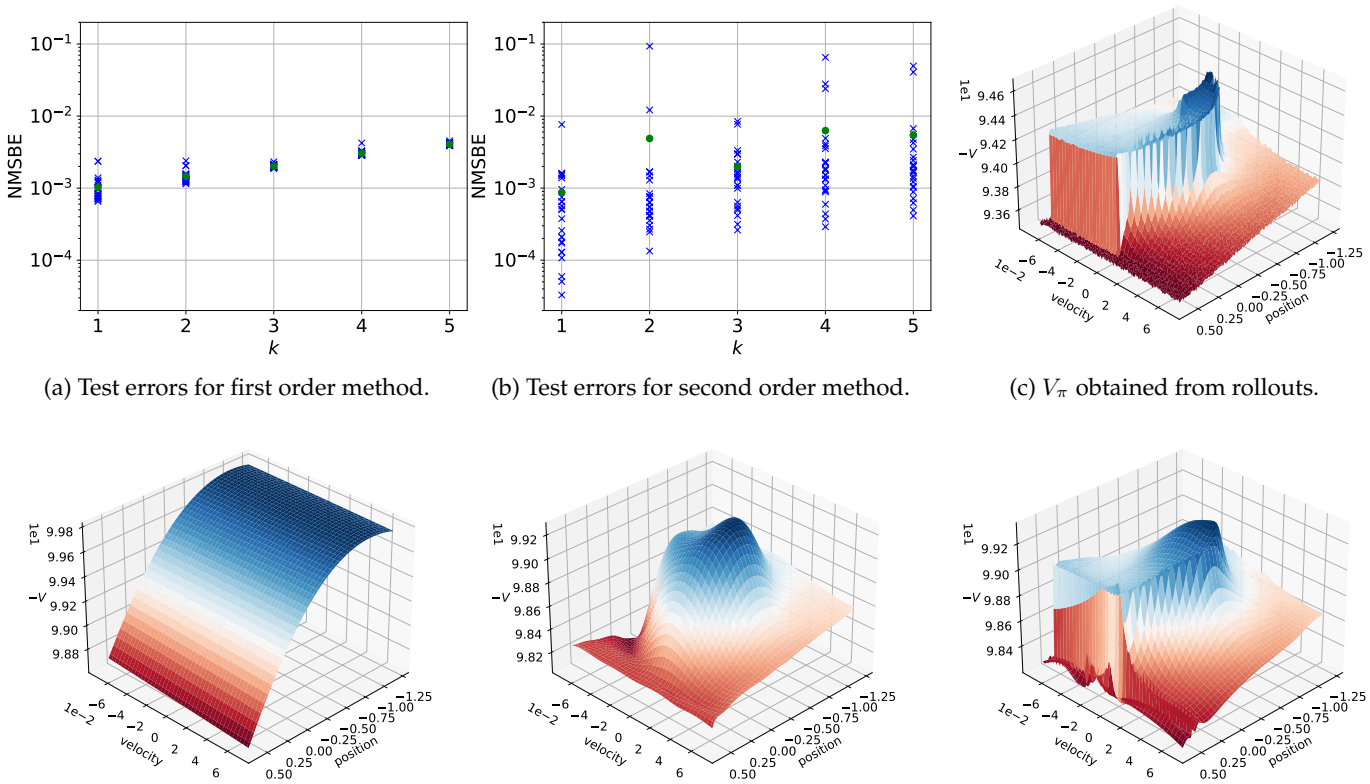
We have seen that the compatibility of multistep lookahead formulations is not given when employing Gauss Newton Residual Gradient algorithms. Although we can demonstrate the beneficial impact of multiple steps on a first order only Residual Gradient algorithm and we also see that a Hessian based optimisation performs well when using single transitions, their combination does not lead to an even more impressive algorithm. Since this happens in relatively easy control problems, our experiments suggest that a proper formulation of the optimisation problem is mandatory and, unfortunately, to some extent still an unsolved problem. There are open questions about the critical points of the objective, in particular when using sampling based approximations, and it is not known, whether this loss is suited for Deep Reinforcement Learning with Residual Gradient algorithms. Furthermore, a sound performance evaluation is also a challenge on its own.

Our results show that whenever one is restricted to first order only optimisation, it is important to use multistep methods to overcome the slow convergence of Residual Gradient algorithms. If the quality of the solution is more important, one should use Hessian information, at least in an approximated form. The final NMSBE is significantly smaller, convergence is fast and one only requires transition data consisting of a single step.

In this work, we have analysed a compound Bellman operator with uniform averaging. Hence, as a next step, it is important to know, whether also TD(λ) methods show this incompatibility.

References

L. C. Baird III. Residual algorithms: Reinforcement learning with function approximation. In *Proceeding of the* 12th *International Conference on Machine Learning*, pages 30–37, 1995.



(a) Test errors for first order method.

(b) Test errors for second order method.

(c) V_π obtained from rollouts.

(d) Gradient only with $k = 5$

(e) Hessian based with $k = 1$

(f) Hessian based with $k = 5$

Figure 2: **a)** and **b)**: Final test errors for first and second order optimisation. **c)** to **f)**: Ground truth value function and the best approximations according to the smallest test errors.

D. P. Bertsekas. *Dynamic Programming and Optimal Control: Approximate Dynamic Programming*, volume 2. Athena Scientific, 4th edition, 2012.

G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *Computing Research Repository*, arXiv:1606.01540, 2016.

M. Gottwald, S. Gronauer, H. Shen, and K. Diepold. Analysis and optimisation of bellman residual errors with neural function approximation. *Computing Research Repository*, arXiv:2106.08774, 2021.

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521:436–444, 2015.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Peterson, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of The* 33rd *International Conference on Machine Learning*, pages 1928–1937, 2016.

J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. In *Proceedings of the* 4th *International Conference on Learning Representations*, 2016.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *Computing Research Repository*, arXiv:1707.06347, 2017.

D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 550:354—359, 2017.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. The MIT Press, 2nd edition, 2020.

H. van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double Q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, pages 2094–2100, 2016.

D. Yu and L. Deng. *Automatic Speech Recognition: A Deep Learning Approach*. Springer-Verlag, London, 2015.