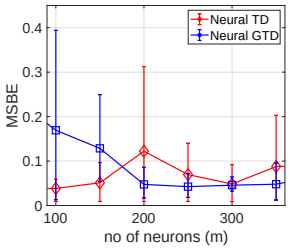
We would like to thank the three reviewers for their feedback. Upon acceptance, we will include (a) *a preliminary experiment results on Neural GTD*, (b) *an expanded discussion on the theoretical results and relation to prior works*. We first discuss the concern about experiments shared by **reviewer 1**, **reviewer 2**, **reviewer 3**.

●●● **Experiments**: The main motivation and contribution of this work have been theoretical – proving that off-policy learning using neural network (NN) functions approximation can learn the value function with (almost) zero MSBE together with a finite time convergence bound. However, we fully agree with the reviewers that a numerical experiment would strengthen this claim. As a preliminary study, we consider an MDP taken from the `Garnet` class with $|\mathsf{S}| = 500$ states, $|\mathsf{A}| = 5$ possible actions per state with uniformly distributed rewards, and the discount factor is $\gamma = 0.9$. We generate two random policies with the same support as the behavior/target policies. We compare the average MSBE against the number of neurons m (for 2-layer, ReLU NN with random init.) after $T = 2 \times 10^5$ iterations of neural GTD and neural TD [Cai et al., 2019] from 3 independent run of state/action. From the figure, the average MSBE decreases with m stably for neural GTD, while it fluctuates with m with neural TD, indicating that the latter can be unstable in the off-policy setting ([Cai et al., 2019] analyzed the neural TD for on-policy). We will include simulations that averages with more trajectories to obtain the expected performance.



Reviewer 1: We thank you for the review and constructive comments. See the point-to-point response below.

NN Architecture: As discussed in (4) and the abstract, our analysis are based on a **2-layer, fully connected, ReLU** NN with d-dimensional input, and m hidden neurons. Having said that, it is an interesting future direction to analyze neural GTD with other types of NN architecture. Lastly, our analysis is based on the surrogate (linearized) NN function (14)-(16), which is akin to a kernel approximation (called the Neural Tangent Kernel, see [Jacot et al., 2018]). In the final version, we will discuss about these connections in detail.

Saddle-point Reformulation: Just as the reviewer said, the reformulation of the min MSBE problem as saddle point optimization follows from prior work such as [Dai et al., 2017,2018], [Shapiro 2011]. We do not claim this as our main contribution either. Instead, the introduction of the **dual NN** in (9) is new. Particularly, a simple application of saddle point reformulation to min MSBE results in (8), which involves an $|\mathsf{S}|$-dim. sub-problem. As $|\mathsf{S}|$ is large (can be infinite), the dual NN is used to circumvent this intractability. We will include your references in the final version.

Dai et al. [2018] derived a similar reformulation to our paper. Besides only guaranteeing convergence to a stationary point (whilst we showed convergence to a global MSBE minimizer for neural GTD), their approximation error is characterized by the ℓ_∞-norm (Theorem 7). The ℓ_∞-norm requirement is restrictive as it requires the function approximation to be **uniformly accurate**, i.e., $\min_\theta |V^\pi(s) - \hat{V}(s;\theta)|$ is small for **every** $s \in \mathsf{S}$. On the other hand, we only require a small L^2 variation (H4), i.e., $\min_\theta \mathbb{E}_s[|V^\pi(s) - \hat{V}(s;\theta)|^2]$. Lastly, their algorithm requires a computation oracle which finds an **exact** solution to the inner optimization (see line 7 of their algorithm 1), which can be intractable. E.g., if an NN approximation is used, this oracle will need to solve an NN regression problem.

Technical Contributions: On top of providing an explicit theoretical analysis with convergence to a *global minimum* MSBE, we emphasize on the technical novelty developed in our paper – We show the convergence rates of the L^2 variation w.r.t. the optimal NN over the **function space** (see (18)) that is *independent of the no. of neurons* m. This is different from existing analysis on GTD learning with linear function approximation (LFA), which show the mean square error of parameter in Euclidean norm. Moreover, the latter analysis may not work if the LFA used involve $p \to \infty$ parameters, since in this case the expected GTD update matrix may cease to be Hurwitz. On the other hand, the convergence rate measured in this L^2 variation (as well as MSBE) is unaffected by the no. of parameters; as seen in Theorem 3.1-3.4 and the numerical experiments above.

For a comparison, as analyzed in [Dalal et al. 2019], GTD with LFA finds an optimal parameter at $\mathcal{O}(1/k)$ (w.h.p.), while neural GTD's rate is $\mathcal{O}(\log k/\sqrt{k})$. However, as mentioned the rate of GTD with LFA is valid only for finite p. Besides, the reference [Kumar et al. 2019] is on actor-critic for policy improvement. Though the subroutine of GTD with LFA is used as the critic, the analysis therein is not directly comparable to this work.

Off-policy Learning: This is when the states/actions received during policy evaluation follow a *behavior policy* which is different from the *target policy* that we want to evaluate. It is a common setting, e.g., when only one set of data is available and our goal is to evaluate different policies without gathering more data. Our study is important as it is known that classical TD learning with LFA can diverge in off-policy learning [Sutton et al. 2009b], and by extension the neural TD may diverge as well. Particularly we show neural GTD is still efficient for off-policy.

Reviewer 2: We thank you for the positive comments. As mentioned, we will now provide a small numerical experiment to strengthen our claims in the final version.

Reviewer 3: We thank you for the positive comments. In the final version, we will extend the discussions of the theoretical results and relation to prior works. We also provide a small numerical experiment to verify our claims (see above). In addition, we are considering to extend the analysis for more general NN architectures.