Value-Based Deep RL Scales Predictably

Oleh Rybkin, Michal Nauman, Preston Fu, Charlie Snell, Pieter Abbeel, Sergey Levine, Aviral Kumar

Berkeley UNIVERSITY OF CALIFORNIA · UNIVERSITY OF WARSAW · Carnegie Mellon University

Motivation and Problem Statement

Q1: RL is sensitive to hyperparameters. How to set batch size and learning rate for large-scale runs?

Q2: To achieve performance level J, RL requires a combination of *data* $\mathcal{D}$ and *compute* $\mathcal{C}$. What is their tradeoff?

Q3: I have a requirement on the total *budget*, a combination $\mathcal{F} = \mathcal{C} + \delta \cdot \mathcal{D}$ of data and compute. How should I configure the algorithm to maximize performance given this budget?

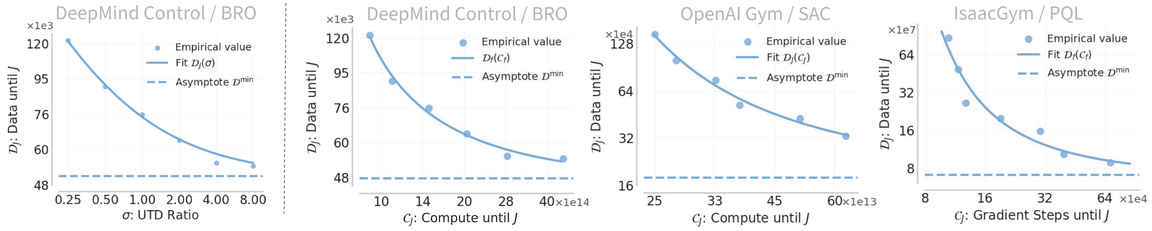
Q2, Q3: Configuring Algorithm

Q2 With the right hyperparameters, data and compute are predictable functions of UTD! Data follows a power law across environments/algorithms.

Tradeoff along the data-compute Pareto frontier parameterized by UTD.

$$\mathcal{D}_J(\text{UTD}) \approx \mathcal{D}_J^{\min} + \left(\frac{\beta_J}{\text{UTD}}\right)^{\alpha_J}$$

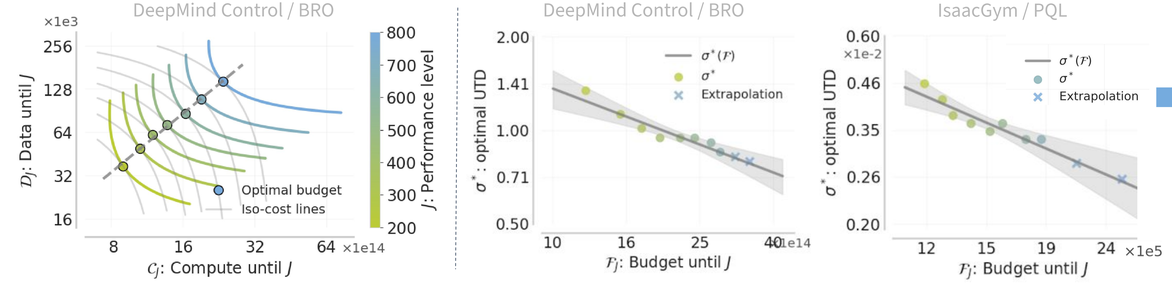
$$\mathcal{C}_J(\text{UTD}) \approx 10 \cdot N \cdot B^*(\text{UTD}) \cdot \text{UTD} \cdot \mathcal{D}_J(\text{UTD})$$



Q3 Each J corresponds to a $\mathcal{D}$–$\mathcal{C}$ Pareto frontier. Each frontier has a unique budget-optimal point, corresponding to an optimal UTD* and budget $\mathcal{F}$.

Optimal UTD follows a power law wrt budget!

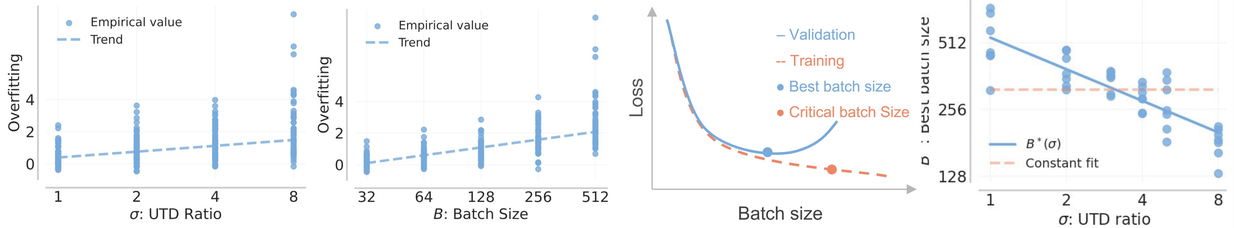
$$\text{UTD}^*(\mathcal{F}) \approx \left(\frac{\beta_{\text{UTD}}}{\mathcal{F}}\right)^{\alpha_{\text{UTD}}}$$



Q1: Setting Hyperparameters

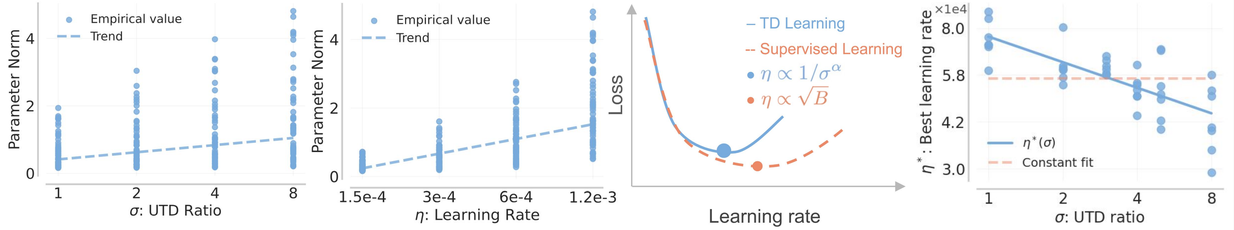
Batch size *Overfitting* = (TD error on newest data) – (TD error on uniformly sampled data). High *updates-to-data ratio* (UTD) and batch size both increase overfitting, since they train on the average data point more. **Decrease batch size with higher UTD to counteract overfitting.**

$$B^*(\text{UTD}) = \left(\frac{\beta_B}{\text{UTD}}\right)^{\alpha_B}$$



Learning rate High UTD ratios and high learning rates both lead to *plasticity loss*. **Decrease learning rate to counteract high parameter norm.**

$$\text{lr}^*(\text{UTD}) \approx \left(\frac{\beta_{\text{lr}}}{\text{UTD}}\right)^{\alpha_{\text{lr}}}$$



Summary and Takeaways

- Batch size, learning rate predictable from UTD; outperforms constant.
- UTD controls the tradeoff between data, compute.
- Optimal UTD follows a power law wrt budget.