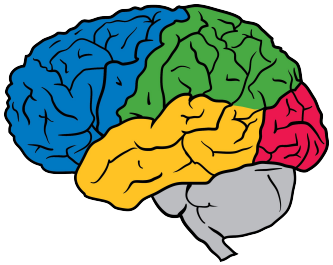
Implicit Under-Parameterization Inhibits Data-Efficient Deep Reinforcement Learning

Aviral Kumar* [2, 3] **Rishabh Agarwal*** [1,3] Dibya Ghosh[2] Sergey Levine[1,2]

1 

2 

3 Mila

*Equal Contribution

Google Research

Reinforcement Learning (RL)

Markov Decision Process (MDP)

S - Set of States

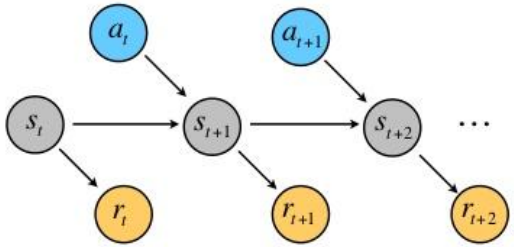
A - Set of Actions

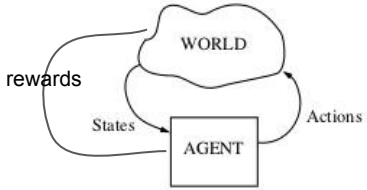
$\Pr(s' \mid a, s)$ - Transitions

α - Starting State Distribution

γ - Discount Factor

r(s) - Reward [or $r(s,a)$]



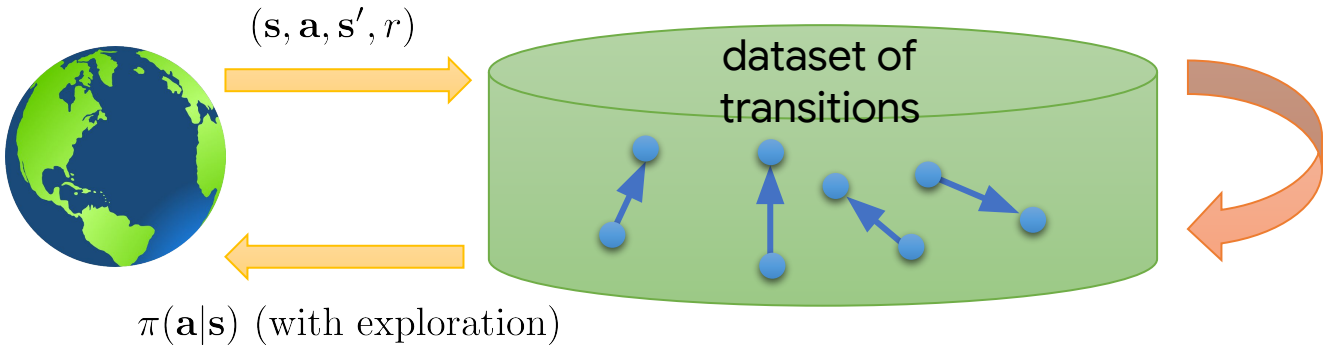


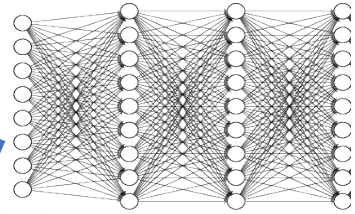
Goal: $\max_{\pi} \mathbb{E}_{\pi} \left[\sum_{t} \gamma^t r(s_t, a_t) \right]$

$s_t \sim P(\cdot \mid s_{t-1}, a_{t-1}), a_t \sim \pi(\cdot|s_t)$

Maximize long-term expected reward.

Modern Deep RL Algorithms





off-policy learning

Training deep neural net policies

Q-Learning

1. Train Q-functions by minimizing **TD Error**:

$$E_{(\mathbf{s},\mathbf{a})\sim\pi_\beta(\mathbf{s},\mathbf{a})}\left[(Q_\phi(\mathbf{s},\mathbf{a}) - (r(\mathbf{s},\mathbf{a}) + \gamma E[Q_\phi(\mathbf{s}',\mathbf{a}')]))^2\right]$$

2. Collect new data in the environment by rolling out the learned policy.

Typically solved "approximately" using gradient descent for a fixed number of steps

How does this approximate optimization procedure affect learning?

Figures adapted from Sergey Levine

Data-Efficient Deep Reinforcement Learning

Data-Efficient Deep RL: Want to learn the most per unit amount of experience/data

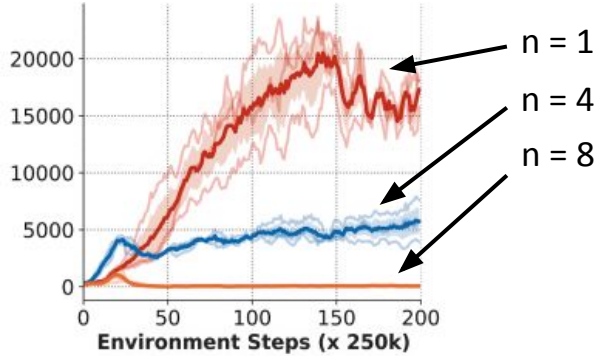
How can we obtain data-efficiency?



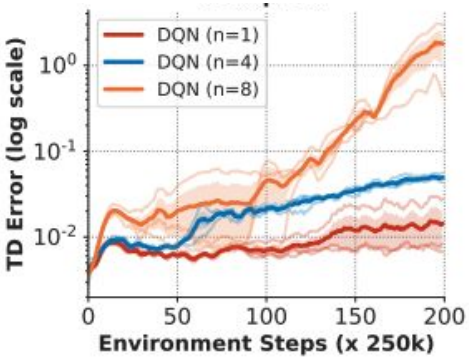
Typically 100-200 updates per datapoint

Supervised Learning:
Train more, control for statistical overfitting
Train error = 0, validation error = high

Reinforcement Learning:
Training >=1 step per datapoint leads to poor



OK, maybe I need to prevent overfitting?



But training error is high with larger n

Why do we see "underfitting" with more training?

Fu*, Kumar* et al. Diagnosing Bottlenecks in Deep Q-Learning Algorithms. ICML 2019.
Fedus*, Ramachandran*, Agarwal et al. Revisiting Fundamentals of Experience Replay. ICML 2020.

Deep Q-learning

TD Error

$$L(s, a; \theta) = \left(Q_\theta(s, a) - \left(R(s, a) + \gamma \max_{a'} Q_{\bar{\theta}}(s', a') \right) \right)^2$$

Bootstrapped Target

Google Research

Deep Q-learning

TD Error

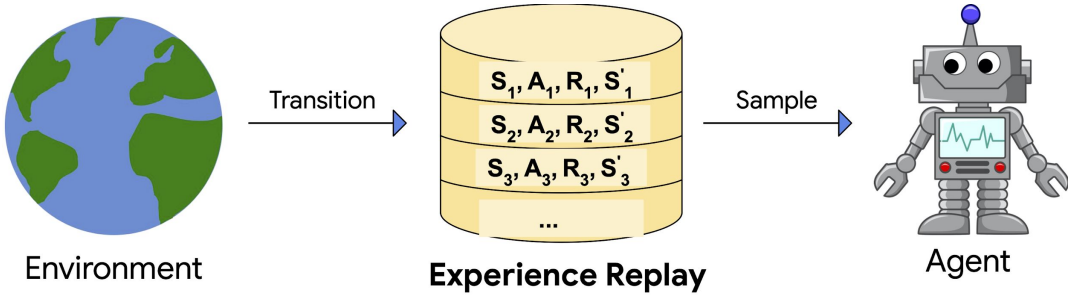
$$L(s, a; \theta) = \left(Q_\theta(s, a) - \left(R(s, a) + \gamma \max_{a'} Q_{\bar{\theta}}(s', a') \right) \right)^2$$

Bootstrapped Target

Regress to targets generated using previous Q-instances.

RL with Experience Replay

- FOR every environment interaction:
  - FOR step= 1, .. , **K**:
    - Sample (s, a) pairs of **size N** from replay
    - Minimize TD-error using above samples



Data-Efficient RL Setting

- FOR every environment interaction:
  - FOR step= 1, .. , **K**:
    - Sample (s, a) pairs of **size N** from replay
    - Minimize TD-error using above samples

More gradient steps (K) per unit data (N)

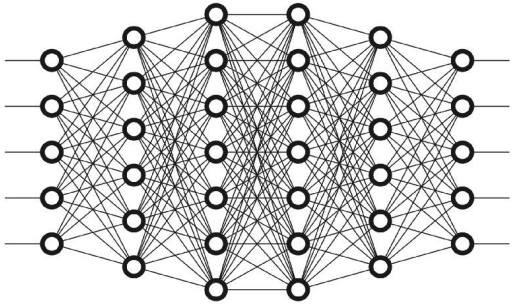


1. Fedus, Ramachandran, Agarwal, Bengio, Larochelle, Rowland, and Dabney. Revisiting fundamentals of experience replay. ICML (2020).
2. Fu*, Kumar*, Soh and Levine. Diagnosing bottlenecks in deep q-learning algorithms. ICML (2019).

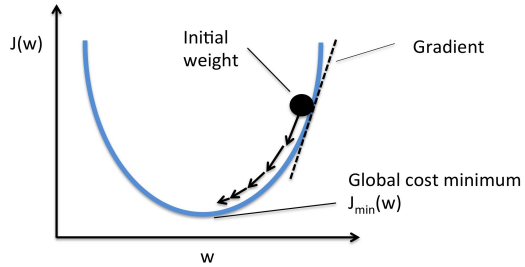
Google Research

Data-Efficient Deep Q-learning

Minimize TD Error



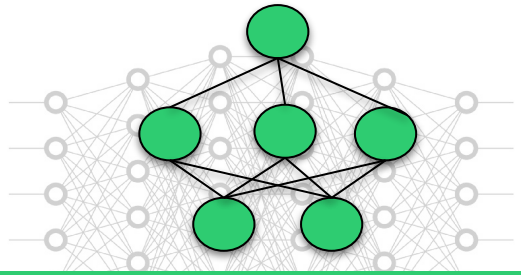
$Q_\theta(s, a)$



Gradient Descent

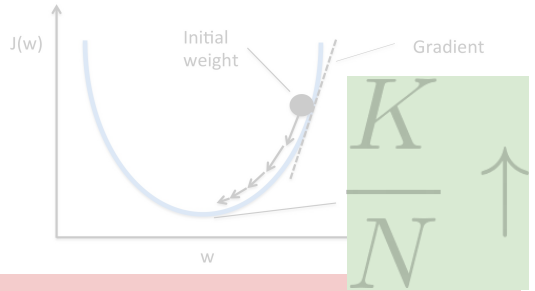
Implicit Under-Parameterization

TD Error



Low Capacity Neural Net

$Q_\theta(s, a)$



Gradient Descent

Q-network implicitly behaves as low capacity network!

Effective Rank of a Matrix

$$Q(s,a) = w^T \Phi(s,a) \qquad \Phi = U \ \mathrm{diag}\{\sigma_i(\Phi)\} \ V$$

$$\Phi \in \mathbb{R}^{|S||A| \times d}$$

Penultimate layer features

Singular Values

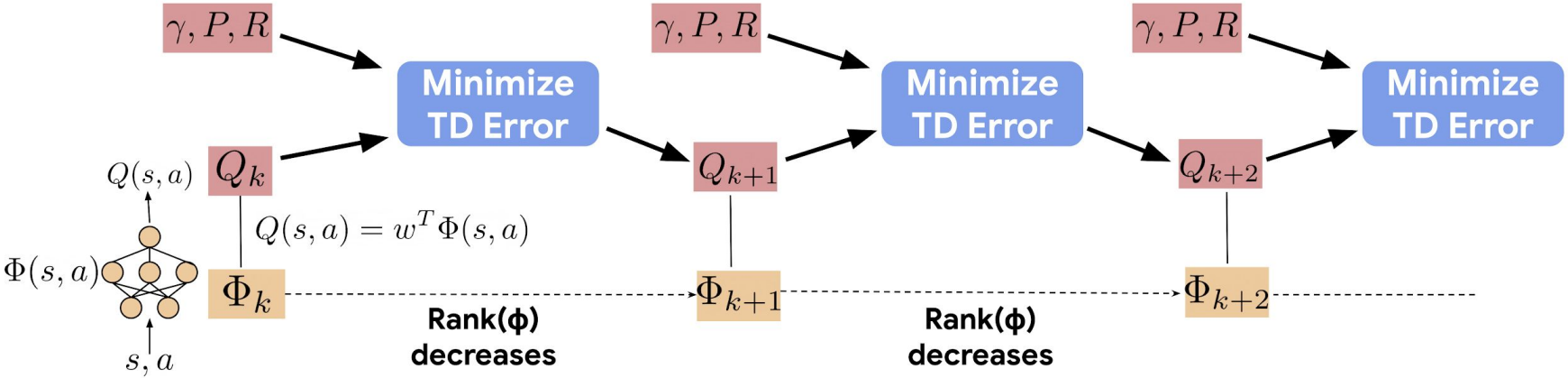
Effective Rank

$$\delta = 0.01$$

$$\mathrm{srank}_\delta(\Phi) = \min \left\{ k : \frac{\sum_{i=1}^{k} \sigma_i(\Phi)}{\sum_{i=1}^{d} \sigma_i(\Phi)} \geq 1 - \delta \right\}$$

1. Yang, Yuzhe, et al. "Harnessing structures for value-based planning and reinforcement learning." *ICLR* (2020).

Implicit Under-Parameterization



Reduction in the effective rank of the features, $\text{srank}_\delta(\Phi)$, that occurs implicitly as a by-product of learning deep neural net Q-functions.

Empirical evidence for Implicit Under-Parameterization

(Data-Efficient) Offline RL

Reinforcement Learning with Online Interactions





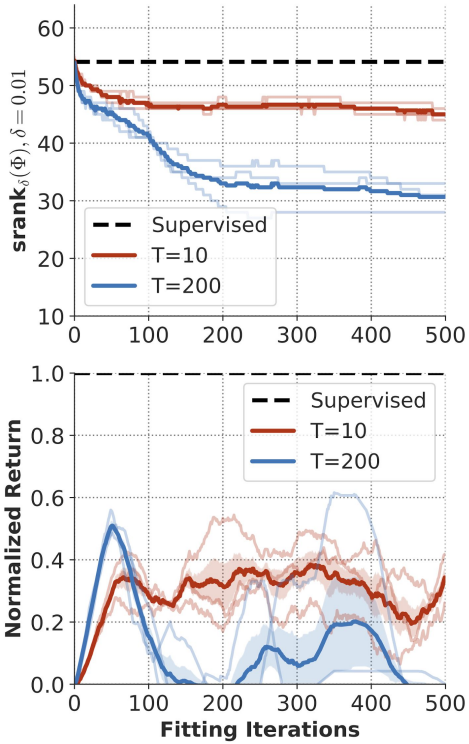
Offline Reinforcement Learning





1. Agarwal, Schuurmans, and Norouzi. "An optimistic perspective on offline reinforcement learning." ICML (2020).

Implicit Under-Parameterization in Offline RL



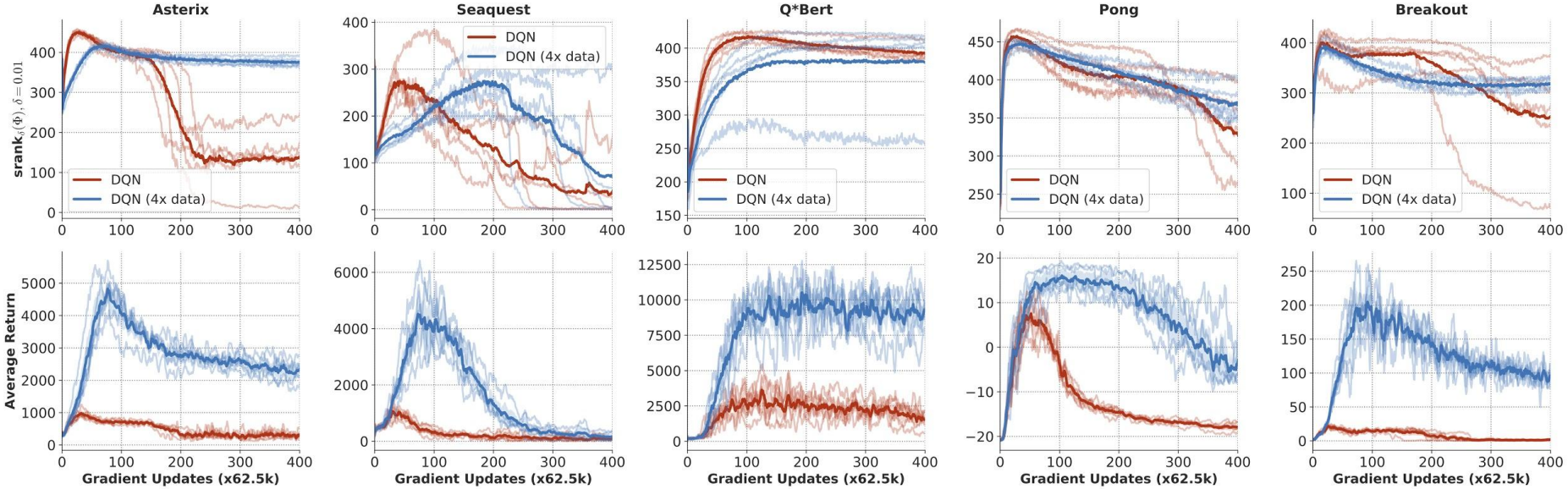
Gridworlds (FQI)

$\mathrm{srank}_\delta(\Phi)$ ⬇ , **Performance** ⬇

T gradient updates per iteration in neural FQI

Google Research

Implicit Under-Parameterization in Offline RL

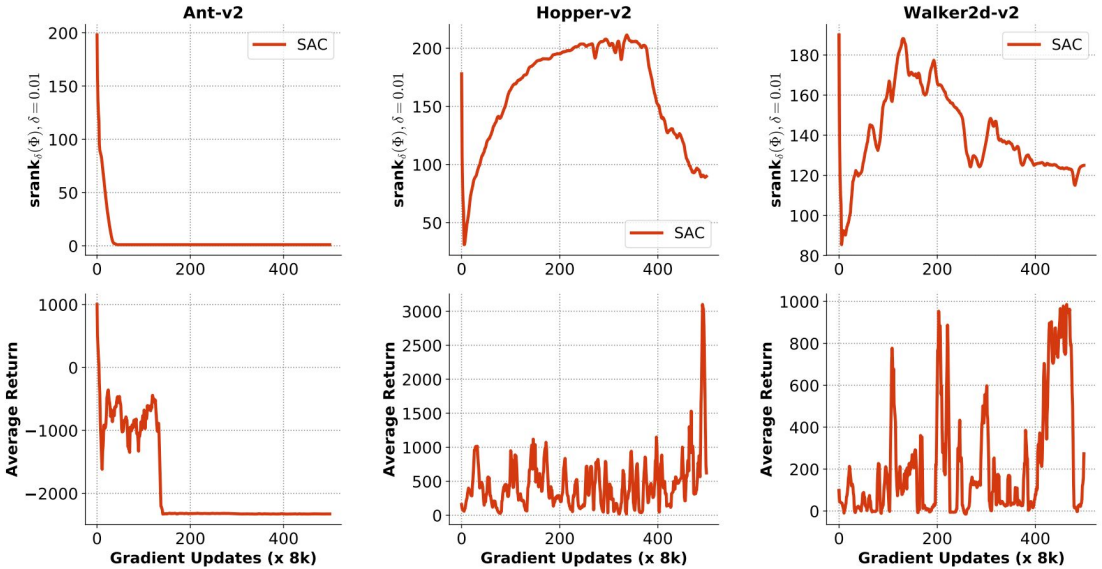


Atari (DQN)

$\text{srank}_\delta(\Phi)$ ⬇ , **Performance** ⬇

Google Research

Implicit Under-Parameterization in Offline RL

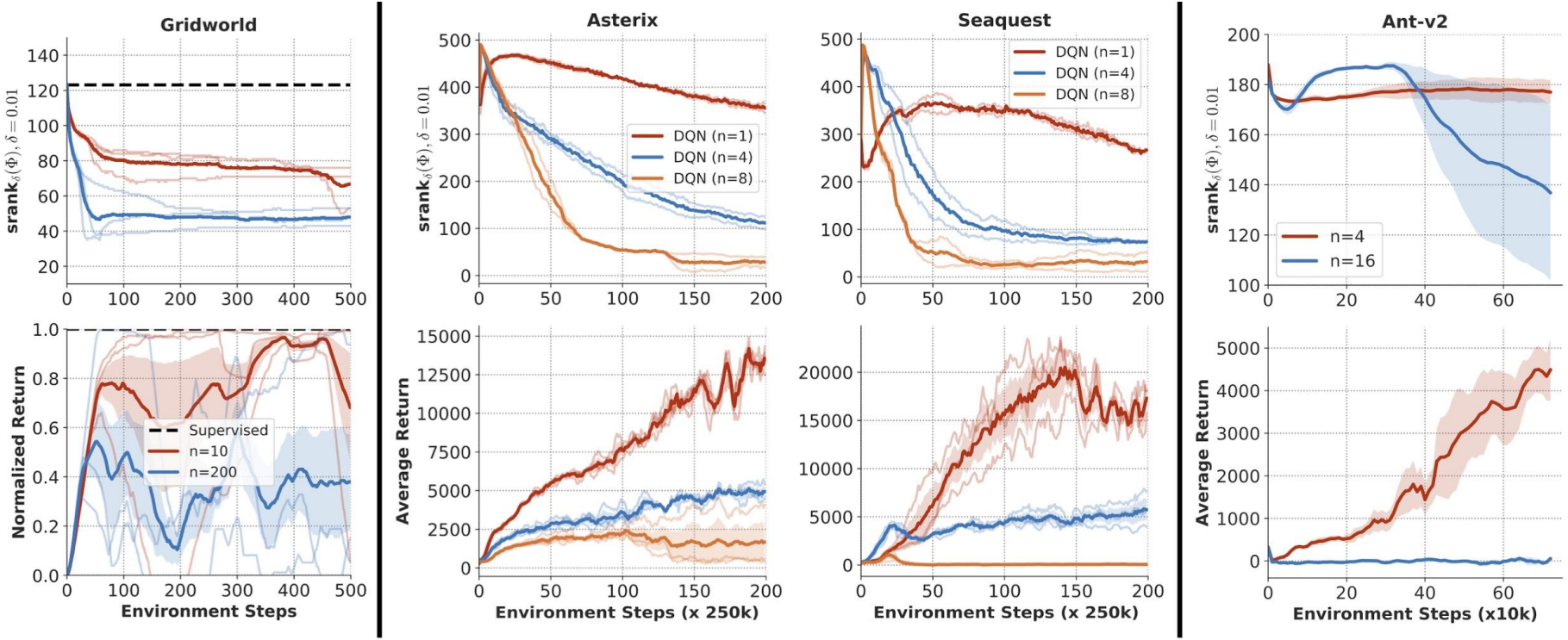


MuJoCo (SAC)

$\text{srank}_\delta(\Phi)$ ⬇

Google Research

IUP in Data-Efficient Online RL



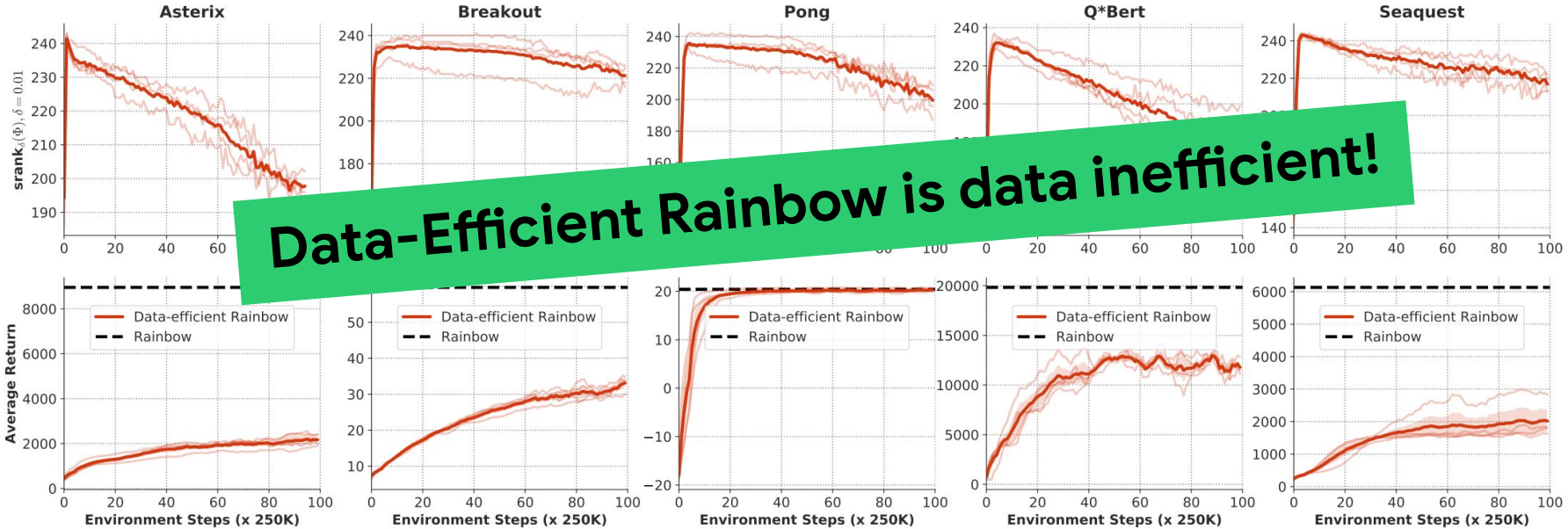
Gridworlds (FQI)

Atari (DQN)

MuJoCo (SAC)

Google Research

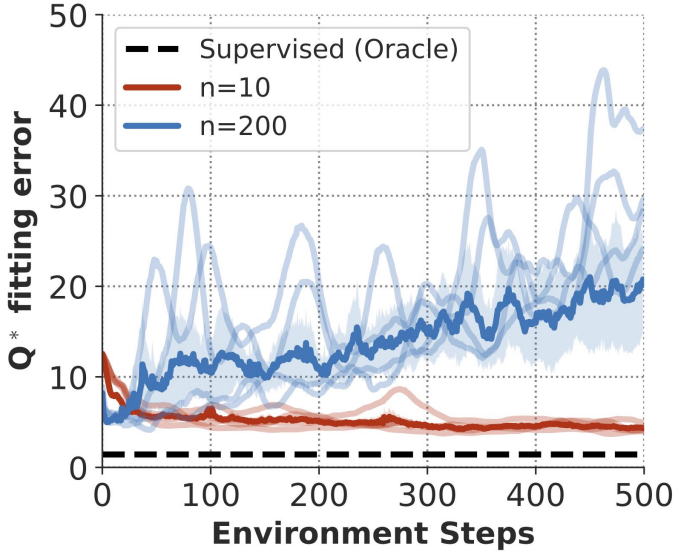
Data-Efficient Online Rainbow



1. van Hasselt, Hessel, and Aslanides. When to use parametric models in reinforcement learning? NeurIPS (2019)

Google Research

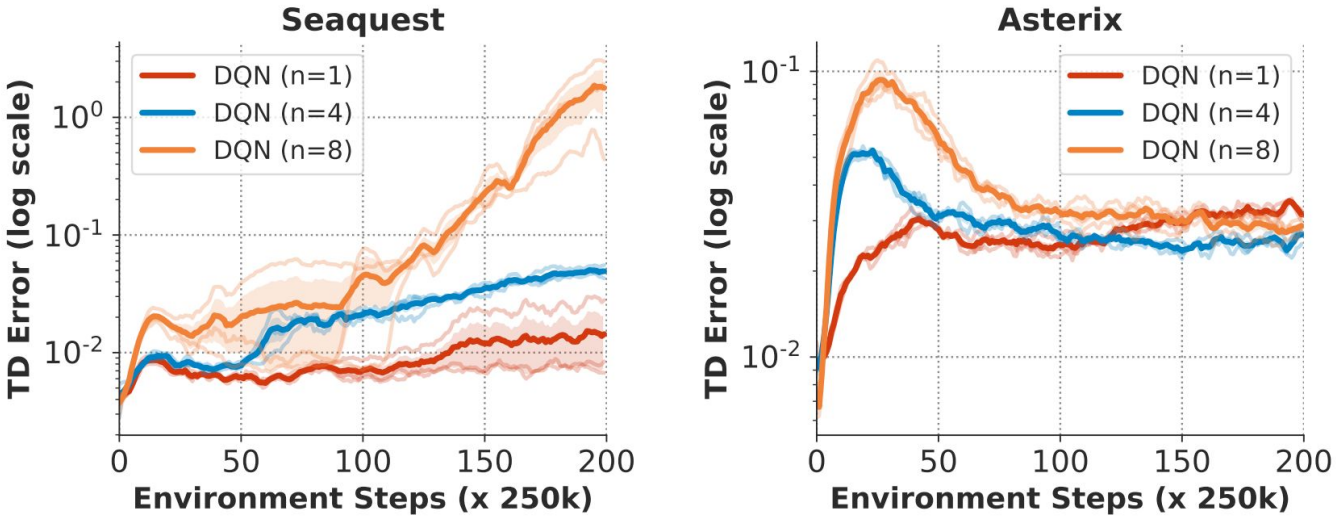
How does implicit under-parameterization degrade performance?



Supervised Regression to Q* with learned features

Google Research

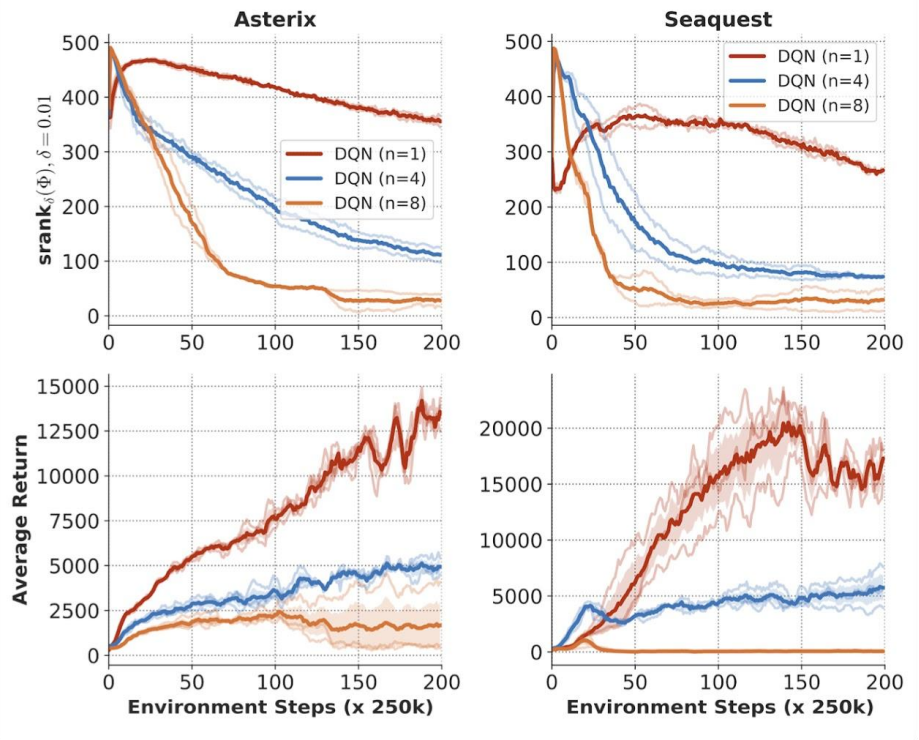
How does implicit under-parameterization degrade performance?



TD error typically increases for features which show higher rank collapse.

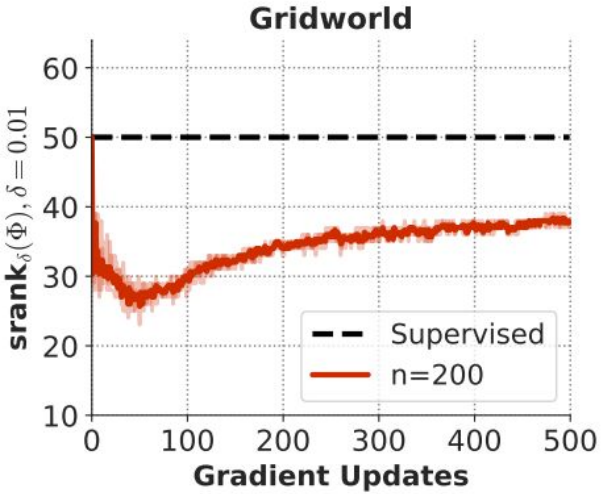
Google Research

How does implicit under-parameterization degrade performance?



Google Research

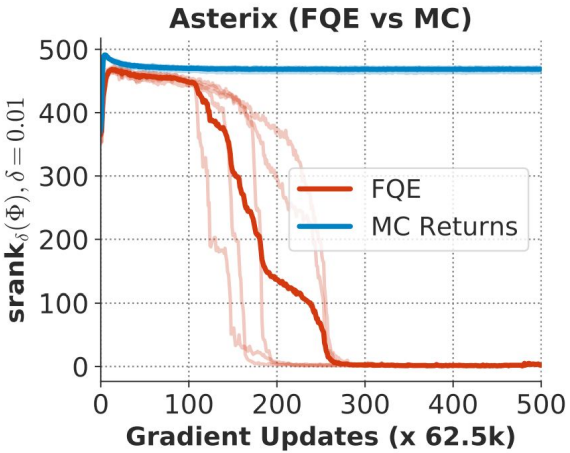
Implicit under-parameterization Is NOT caused by non-stationarity in RL

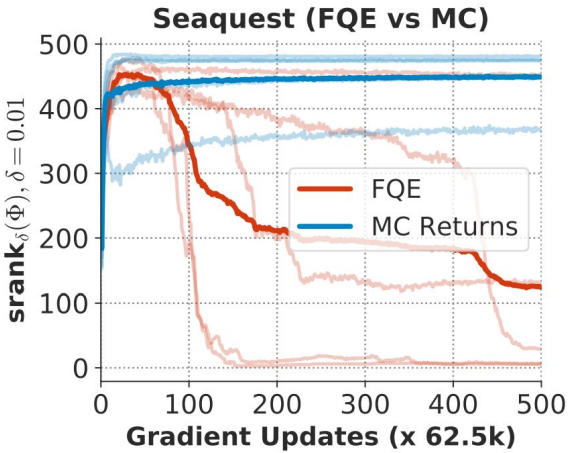


Using oracle access to exact Q^π for computing the target values does not significantly decrease in rank, and training for longer can increase feature rank.

Google Research

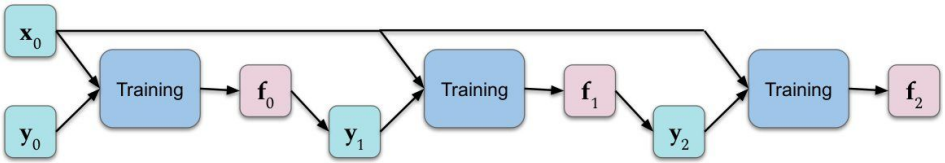
Is implicit under-parameterization caused by bootstrapping?





Theoretical Analysis of Implicit Under-Parameterization

Self-distillation induces regularization



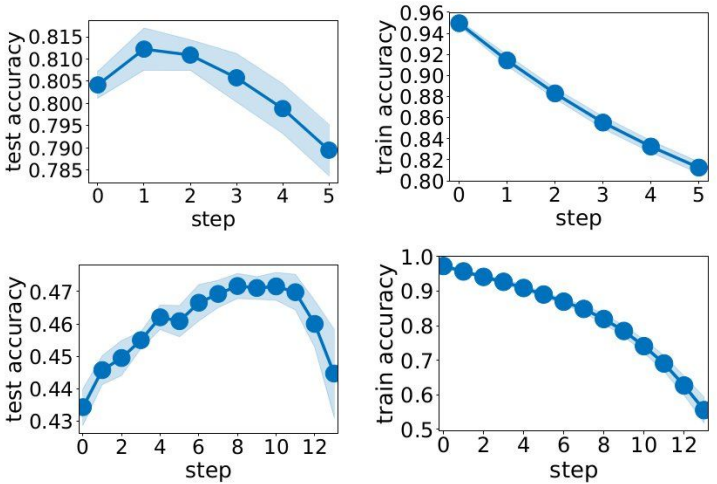


Figure 5: Self-distillation with cross entropy loss on predictions for Resnet50 (top) on CIFAR-10, (bottom) on CIFAR-100.

Mobahi et al. Self-distillation amplifies regularization in hilbert space. NeurIPS (2020).

Google Research

Analysis via Kernel Regression (NTK)

$$\mathbf{Q}_{k+1} \leftarrow \arg\min_{Q \in \mathcal{Q}} \sum_{\mathbf{s}_i, \mathbf{a}_i \in \mathcal{D}} \left(Q(\mathbf{s}_i, \mathbf{a}_i) - y_k(\mathbf{s}_i, \mathbf{a}_i)\right)^2 + c \sum_{(\mathbf{s},\mathbf{a})} \sum_{(\mathbf{s}',\mathbf{a}')} u((\mathbf{s},\mathbf{a}),(\mathbf{s}',\mathbf{a}')) Q(\mathbf{s},\mathbf{a}) Q(\mathbf{s}',\mathbf{a}').$$

Self-distillation:

$$\mathbf{y}_k = \mathbf{y}_{k-1}$$

RL with bootstrapping:

$$\mathbf{y}_k = \mathbf{R} + \gamma P^\pi \mathbf{Q}_{k-1}$$

Kernel Regularizer capturing implicit bias of gradient descent

1. Mobahi et al. Self-distillation amplifies regularization in hilbert space. NeurIPS (2020).
2. Jacot, A. et al. Neural tangent kernel: Convergence and generalization in neural networks. NeurIPS (2018).

Google Research

Analysis via Kernel Regression (NTK)

$$\mathbf{Q}_{k+1} \leftarrow \arg\min_{\mathbf{Q}\in\mathcal{Q}} \sum_{\mathbf{s}_i,\mathbf{a}_i\in\mathcal{D}} \left(Q(\mathbf{s}_i,\mathbf{a}_i) - y_k(\mathbf{s}_i,\mathbf{a}_i)\right)^2 + c \sum_{(\mathbf{s},\mathbf{a})} \sum_{(\mathbf{s}',\mathbf{a}')} u((\mathbf{s},\mathbf{a}),(\mathbf{s}',\mathbf{a}'))Q(\mathbf{s},\mathbf{a})Q(\mathbf{s}',\mathbf{a}').$$

$$Q_{k+1}(\mathbf{s},\mathbf{a}) = \mathbf{g}_{(\mathbf{s},\mathbf{a})}^T (c\mathbf{I} + \mathbf{G})^{-1} \mathbf{y}_k$$

Green's function

1. Mobahi et al. Self-distillation amplifies regularization in hilbert space. NeurIPS (2020).
2. Jacot, A. et al. Neural tangent kernel: Convergence and generalization in neural networks. NeurIPS (2018).

Google Research

Green's function

Let $g(\boldsymbol{x}, \boldsymbol{t})$ be a function such that,

$$\int_{\mathcal{X}} u(\boldsymbol{x}, \boldsymbol{x}^\dagger)\, g(\boldsymbol{x}^\dagger, \boldsymbol{t})\, d\boldsymbol{x}^\dagger = \delta(\boldsymbol{x} - \boldsymbol{t})\,. \tag{2}$$

Such g is called the ***Green's function*** of the linear operator L satisfying $[Lf](\boldsymbol{x}) = \int_{\mathcal{X}} u(\boldsymbol{x}, \boldsymbol{x}^\dagger)\, f^*(\boldsymbol{x}^\dagger)\, d\boldsymbol{x}^\dagger$. Let $\boldsymbol{y}_{K\times 1} \triangleq (y_1, y_2, \ldots, y_K)$.

1. Dean G Duffy. Green's functions with applications. CRC Press, 2015
2. Mobahi et al. Self-distillation amplifies regularization in hilbert space. NeurIPS (2020).

Google Research

Green's function

$$x_i = (s_i, a_i), K = |\mathcal{D}|$$

Let the matrix $G_{K\times K}$ and the vector $g_{x\,K\times 1}$ be defined as,

$$\boldsymbol{G}[j,k] \triangleq \frac{1}{K} g(\boldsymbol{x}_j, \boldsymbol{x}_k)$$

$$\boldsymbol{g}_{\boldsymbol{x}}[k] \triangleq \frac{1}{K} g(\boldsymbol{x}, \boldsymbol{x}_k)\,.$$

Notice similarity with NTK!

1. Mobahi et al. Self-distillation amplifies regularization in hilbert space. NeurIPS (2020).

Google Research

Solution using Green's Function

$$\mathbf{Q}_{k+1} \leftarrow \arg\min_{\mathbf{Q}\in\mathcal{Q}} \sum_{\mathbf{s}_i,\mathbf{a}_i\in\mathcal{D}} \left(Q(\mathbf{s}_i,\mathbf{a}_i) - y_k(\mathbf{s}_i,\mathbf{a}_i)\right)^2 + c \sum_{(\mathbf{s},\mathbf{a})} \sum_{(\mathbf{s}',\mathbf{a}')} u((\mathbf{s},\mathbf{a}),(\mathbf{s}',\mathbf{a}'))Q(\mathbf{s},\mathbf{a})Q(\mathbf{s}',\mathbf{a}').$$

$$Q_{k+1}(\mathbf{s},\mathbf{a}) = \mathbf{g}_{(\mathbf{s},\mathbf{a})}^T (c\mathbf{I} + \mathbf{G})^{-1} \mathbf{y}_k$$

$$\mathbf{Q}_{k+1} = \mathbf{G}(c\mathbf{I}+\mathbf{G})^{-1}\mathbf{y}_k = \underbrace{\mathbf{G}(c\mathbf{I}+\mathbf{G})^{-1}}_{\mathbf{A}} [\mathbf{R} + \gamma P^\pi \mathbf{Q}_k] = \mathbf{A}\left(\sum_{i=1}^k \gamma^{k-i} \left(P^\pi \mathbf{A}\right)^{k-i}\right)\mathbf{R} := \mathbf{A}\mathbf{M}_k\mathbf{R}.$$

Google Research

Analysis via Kernel Regression (NTK)

$$\mathbf{Q}_{k+1} = \mathbf{A}\mathbf{M}_k\mathbf{R}$$

Solution

$$\mathbf{M}_k = \sum_{i=1}^{k} \gamma^{k-i} \left(P^{\pi}\mathbf{A}\right)^{k-i-1}$$

There exists an increasing sequence $\{k_1, k_2, .. k_l, .., k_\infty\}$ of iterations such that effective rank of M_k decreases.

$$\forall\, l \in \mathbb{N} \;\; and \;\; l' \geq l, \;\; \frac{\sigma_i(\mathbf{M}_{k_{l'}})}{\sigma_j(\mathbf{M}_{k_{l'}})} < \frac{\sigma_i(\mathbf{M}_{k_l})}{\sigma_j(\mathbf{M}_{k_l})} \leq \frac{\sigma_i(\mathbf{S})}{\sigma_j(\mathbf{S})}.$$

Google Research

Analysis via Deep Linear Networks

Gradient descent with deep networks has an implicit regularization effect in supervised learning.

$$\min_X \quad ||AX - y||_2^2$$

$$\min_X ||X||_F^2 \text{ s.t. } AX = y$$

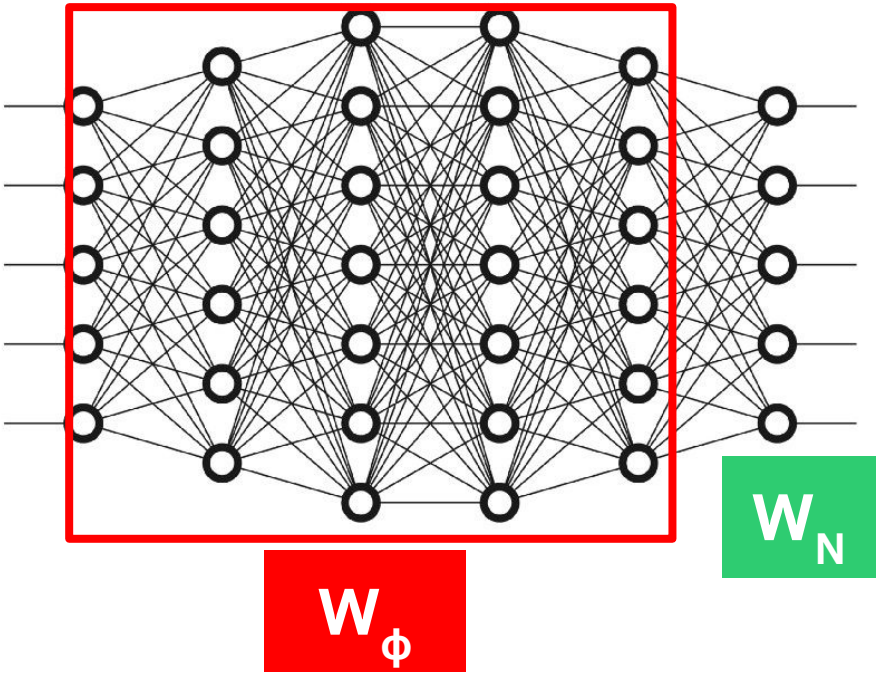
If gradient descent converges to a good solution, it converges to a minimum norm solution

Check Arora et al. (2019) for how this regularization is more complex ..

Gunasekar et al. Implicit Regularization in Matrix Factorization. NeurIPS 2017.
Arora et al. Implicit Regularization in Deep Matrix Factorization. NeurIPS 2019

Google Research

Analysis via Deep Linear Networks



3 or more linear layers

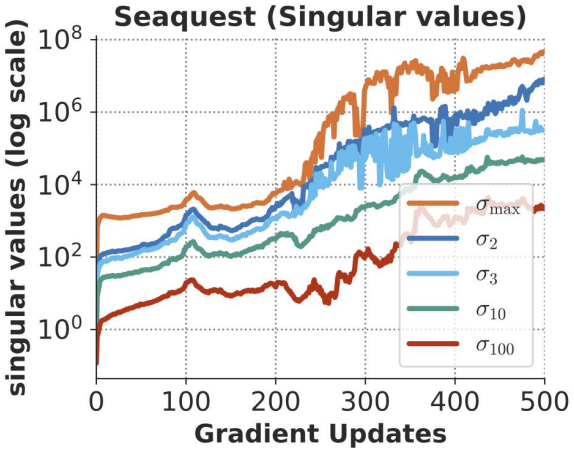
$$Q(\mathbf{s}, \mathbf{a}) = \mathbf{W}_N \mathbf{W}_\phi [\mathbf{s}; \mathbf{a}]$$

Analysis via Deep Linear Networks

$$\dot{\sigma}_r(k,t) \propto \left(\sigma_r^2(k,t)\right)^{1-\frac{1}{N-1}}$$

"Rich gets richer" effect!

Relaxed assumptions .. Arora et al. trivially leads to rank 1 features



Arora et al. Implicit Regularization in Deep Matrix Factorization. NeurIPS 2019

Google Research

Analysis via Deep Linear Networks

$$Q(\mathbf{s}, \mathbf{a}) = \mathbf{W}_N \mathbf{W}_\phi [\mathbf{s}; \mathbf{a}]$$

$$h_{k,t}(i) = \sum_{j=1}^{i} \frac{\sigma_j(\mathbf{W}_\phi(k,t))}{\sum_l \sigma_l(\mathbf{W}_\phi(k,t))} = \frac{1}{1 + \frac{\sum_{j=i+1}^{n} \sigma_j(\mathbf{W}_\phi(k,t))}{\sum_{j=1}^{i} \sigma_j(\mathbf{W}_\phi(k,t))}} \geq \frac{1}{1 + \sum_{j=i+1}^{n} \frac{\sigma_j(\mathbf{W}_\phi(k,t))}{\sigma_1(\mathbf{W}_\phi(k,t))}}$$

$$\mathrm{srank}_\delta(\mathbf{W}_\phi(k,t)) = \min_j \{\, h_k(j) \geq 1 - \delta \,\}.$$

$$\min_{\mathbf{W}_\phi, \mathbf{W}_N} ||\mathbf{W}_N \mathbf{W}_\phi [\mathbf{s}; \mathbf{a}] - \mathbf{y}_k(\mathbf{s}, \mathbf{a})||^2 + \lambda_k \mathrm{srank}_\delta(\mathbf{W}_\phi).$$

Google Research

Compounding rank drop

Closure. If function class can represent Q, then it can also represent the bootstrapped target, that is, $W_N^P W_\phi^P[s,a] = R + \gamma P^\pi Q(s,a)$

Change in srank. $srank_\delta(\mathbf{W}_\phi^P(k)) \leq srank_\delta(\mathbf{W}_\phi(k-1)) + c_k.$

$$\text{srank}_\delta(\mathbf{W}_\phi(k)) \leq \boxed{\text{srank}_\delta(\mathbf{W}_\phi(0)) + \sum_{j=1}^{k} c_j - \sum_{j=1}^{k} \frac{||\mathbf{Q}_j - \mathbf{y}_j||}{\lambda_j}}$$

Google Research

Proof outline: Compounding drop

$$\min_{\mathbf{W}_\phi, \mathbf{W}_N} ||\mathbf{W}_N \mathbf{W}_\phi [\mathbf{s}; \mathbf{a}] - \mathbf{y}_k(\mathbf{s}, \mathbf{a})||^2 + \lambda_k \text{srank}_\delta(\mathbf{W}_\phi).$$

$$\text{srank}_\delta(\mathbf{W}_\phi(k)) \leq \text{srank}_\delta(\mathbf{W}_\phi^P(k)) - \frac{||\mathbf{Q}_k - \mathbf{y}_k||}{\lambda_k} \quad \text{(since we can copy over the weights)}$$

$$\text{srank}_\delta(\mathbf{W}_\phi(k)) \leq \boxed{\text{srank}_\delta(\mathbf{W}_\phi(0)) + \sum_{j=1}^{k} c_j - \sum_{j=1}^{k} \frac{||\mathbf{Q}_j - \mathbf{y}_j||}{\lambda_j}}$$

Google Research

Analysis via Deep Linear Networks

$$\min_{\mathbf{W}_\phi, \mathbf{W}_N} ||\mathbf{W}_N \mathbf{W}_\phi [\mathbf{s}; \mathbf{a}] - \mathbf{y}_k(\mathbf{s}, \mathbf{a})||^2 + \lambda_k \mathrm{srank}_\delta(\mathbf{W}_\phi).$$

"Self-training" regime gets activated near convergence!

Near convergence

$$\mathbf{y}_k = \mathbf{R} + \gamma P^\pi \mathbf{Q}_{k-1} \approx \mathbf{Q}_{k-1}$$

Consequence of rank drop near convergence

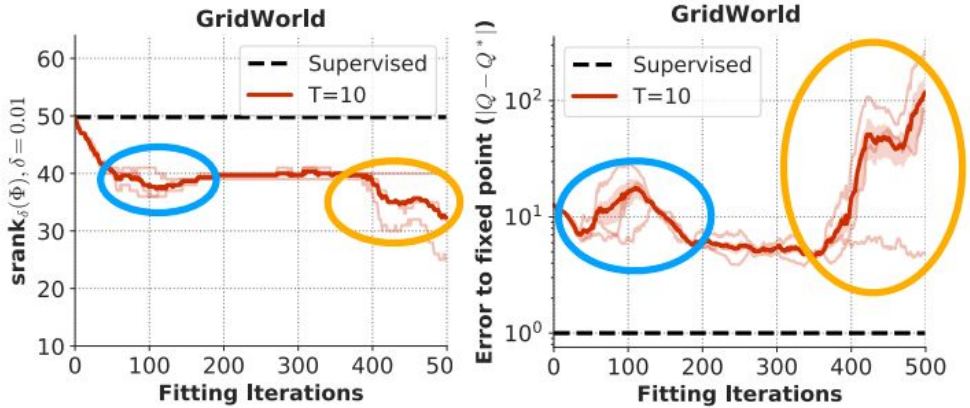
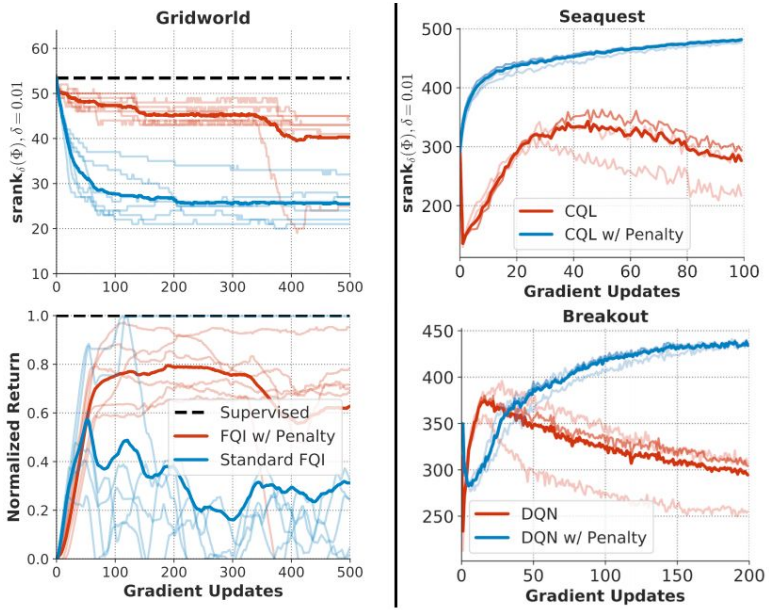


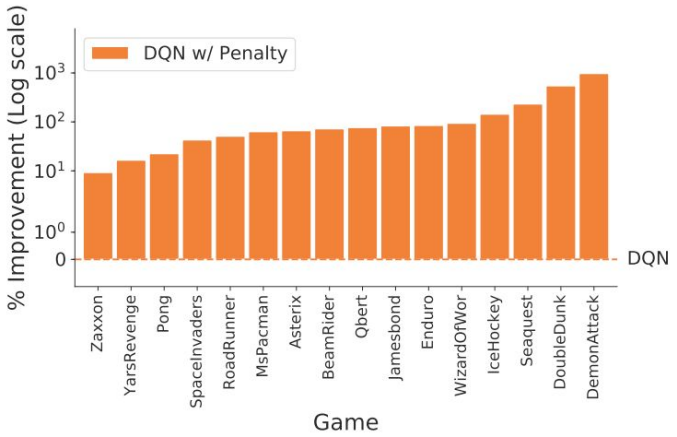
Figure 4: Trend of $\text{srank}_\delta(\Phi)$ *v*.s. error on log scale to the projected TD fixed point. A drop in $\text{srank}_\delta(\Phi)$ (shown as blue and yellow circles) corresponds to a corresponding increase in distance to the fixed point.

Mitigating Implicit Under-Parameterization Improves Deep Q-learning

Google Research

$$\mathcal{L}_p(\Phi) = \sigma^2_{\max}(\Phi) - \sigma^2_{\min}(\Phi).$$





Thank you!