Diagnosing Bottlenecks in Deep Q-learning Algorithms

Justin Fu*, Aviral Kumar*, Matthew Soh, Sergey Levine

Berkeley
UNIVERSITY OF CALIFORNIA

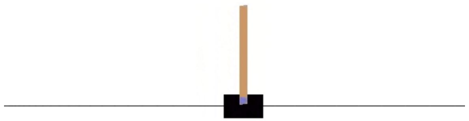
*Equal Contribution

Motivation

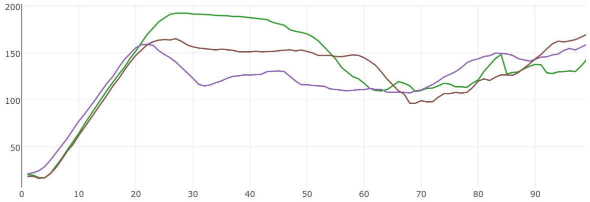
- Deep Q-learning methods are notoriously brittle and hard to tune

Motivation

- Deep Q-learning methods are notoriously brittle and hard to tune



CartPole-v0

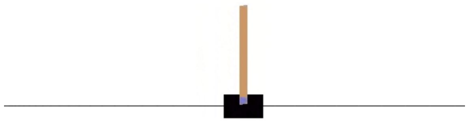


Unstable Training

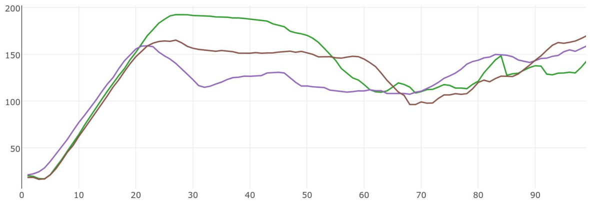
- Compared to supervised learning, Q-learning is poorly understood

Motivation

- Deep Q-learning methods are notoriously brittle and hard to tune



CartPole-v0



Unstable Training

- Compared to supervised learning, Q-learning is poorly understood
- Our goal: **empirically** measure the extent of potential theoretical issues and identify effective research directions.
    - Unit test on tractable domains, verify on standard deep RL tasks

How does function approximation affect convergence?

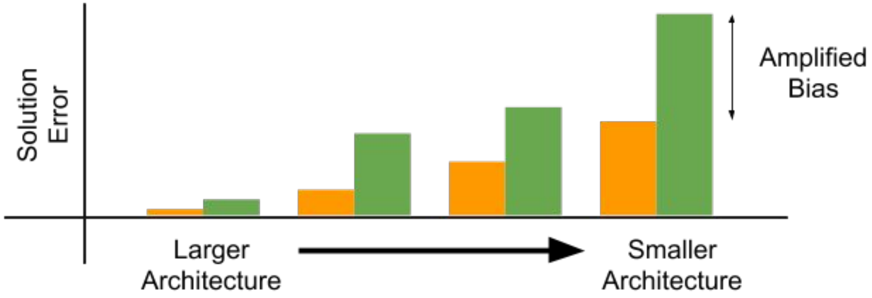
- Divergence is **not** common in practice

How does function approximation affect convergence?

- Divergence is **not** common in practice
- Solution quality deteriorates rapidly with weaker approximators.
  - Bias is **amplified**

How does function approximation affect convergence?

- Divergence is **not** common in practice
- Solution quality deteriorates rapidly with weaker approximators.
  - Bias is **amplified**



(orange) Error of best solution in model class

(green) Error of solution found by approximate Q-learning

How does function approximation affect convergence?

- Divergence is **not** common in practice
- Solution quality deteriorates rapidly with weaker approximators.
    - Bias is **amplified**

(orange) Error of best solution in model class

(green) Error of solution found by approximate Q-learning

Does overfitting occur?

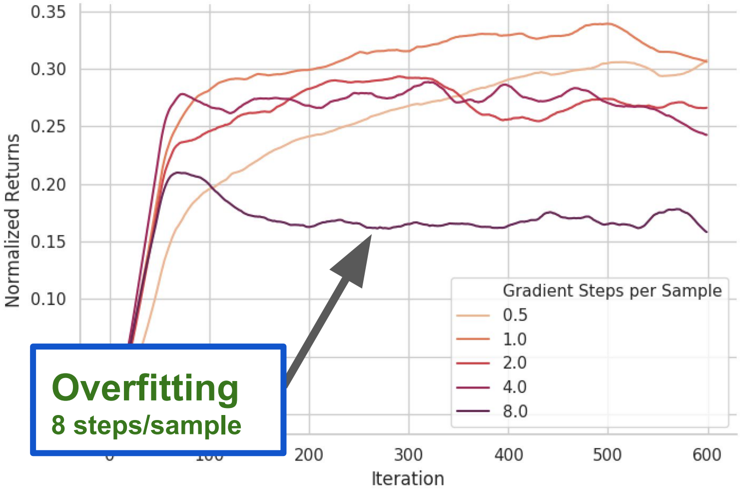
- Large architectures tend to do *better* even in the presence of overfitting.

Does overfitting occur?

- Large architectures tend to do *better* even in the presence of overfitting.
- The **number of gradient steps** per sample is a simple parameter that greatly affects performance.

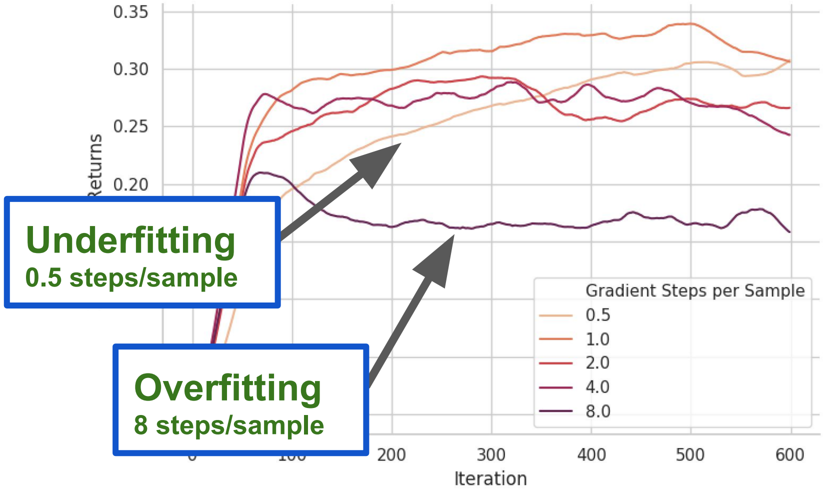
Does overfitting occur?

- Large architectures tend to do *better* even in the presence of overfitting.
- The **number of gradient steps** per sample is a simple parameter that greatly affects performance.



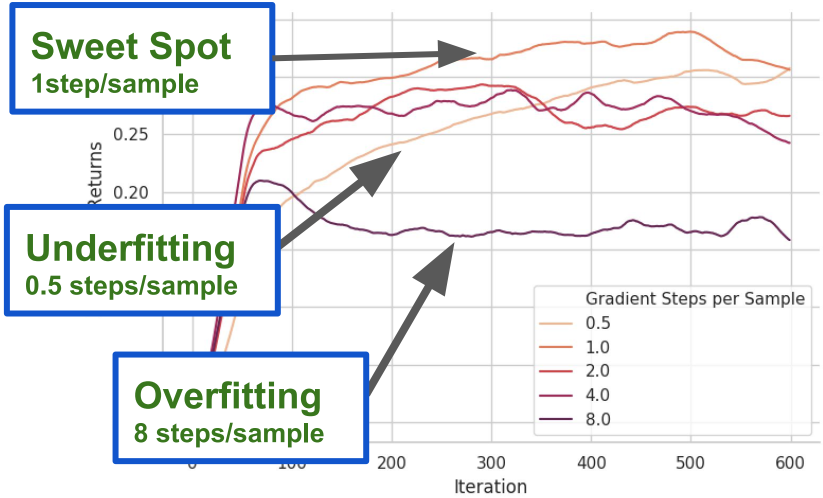
Does overfitting occur?

- Large architectures tend to do *better* even in the presence of overfitting.
- The **number of gradient steps** per sample is a simple parameter that greatly affects performance.



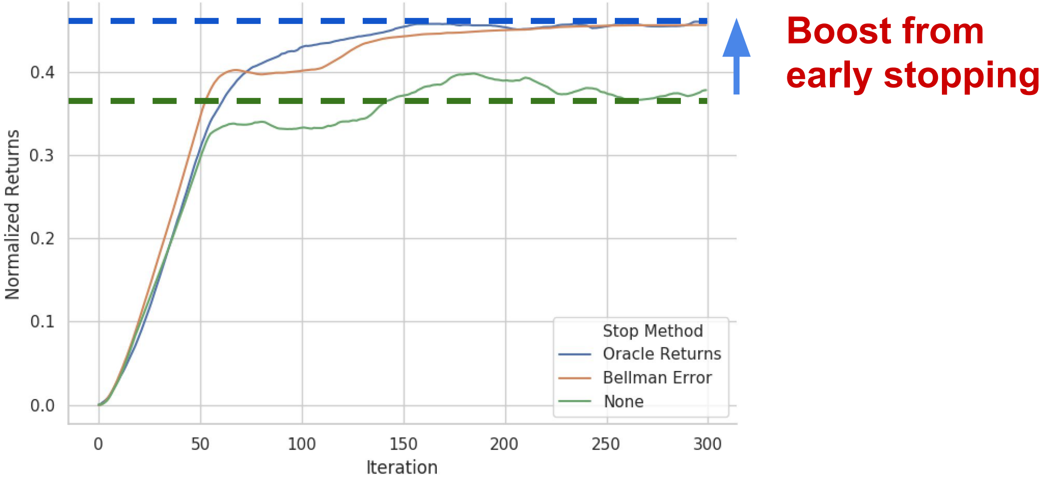
Does overfitting occur?

- Large architectures tend to do *better* even in the presence of overfitting.
- The **number of gradient steps** per sample is a simple parameter that greatly affects performance.



Can early stopping help?

- We can automatically tune the number of steps using some criterion (such as validation error).



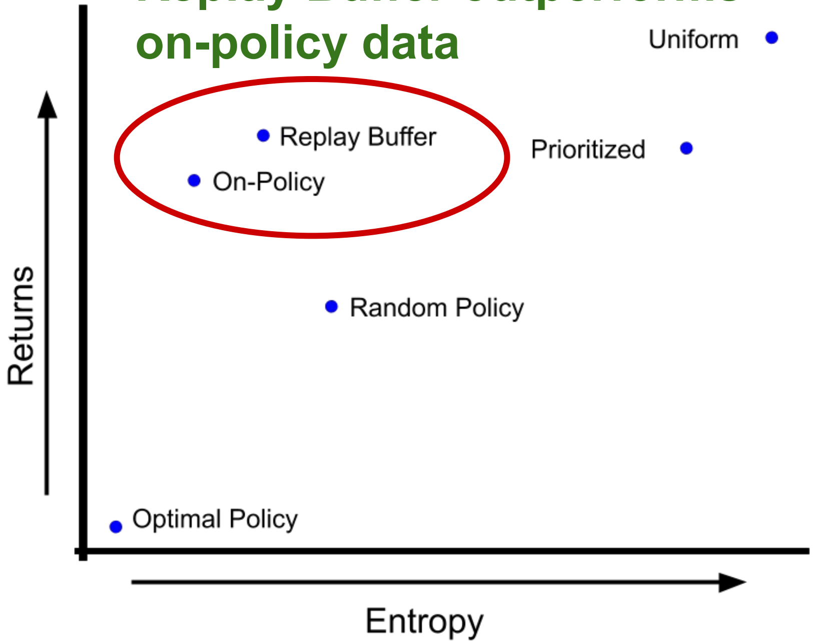
How to choose the sampling distribution?

- On-policy *not* always better.
  - **Intuition:** Narrow distribution; can easily query out-of-distribution values

How to choose the sampling distribution?

- On-policy *not* always better.
  - **Intuition:** Narrow distribution; can easily query out-of-distribution values
- Using data directly from a replay buffer works well, if not better.

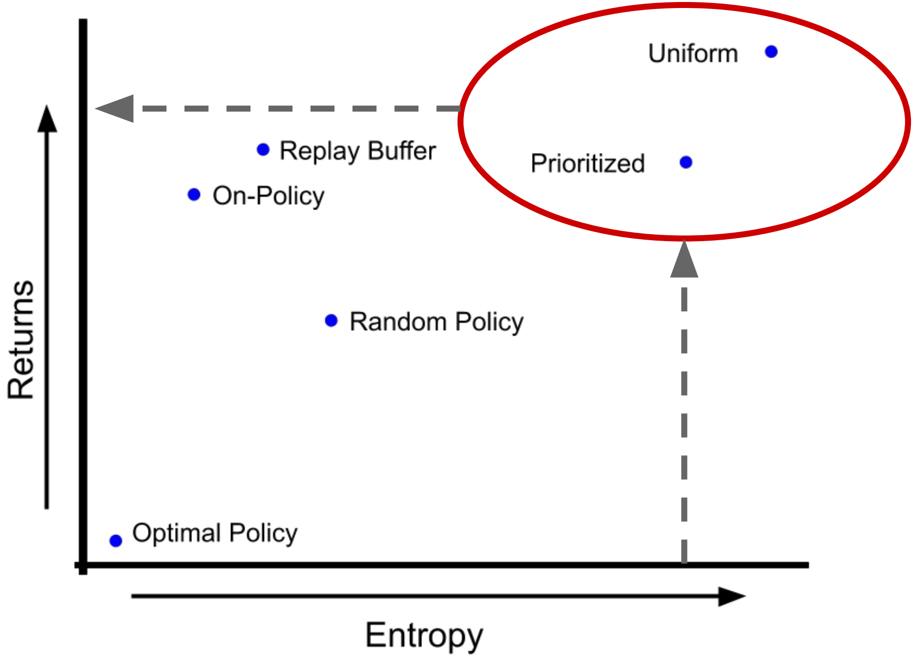
Replay Buffer outperforms on-policy data



How to choose the sampling distribution?

- On-policy *not* always better.
  - **Intuition:** Narrow distribution; can easily query out-of-distribution values
- Using data directly from a replay buffer works well, if not better.
- **High-entropy** distributions over the state space are generally effective

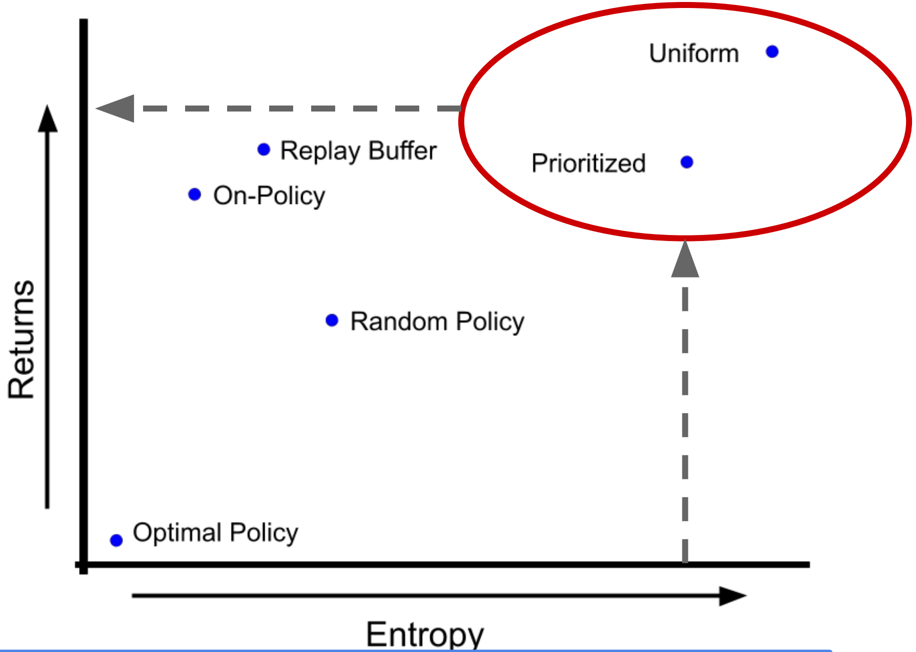
High-entropy generally performs better



How to choose the sampling distribution?

High-entropy generally performs better

- On-policy *not* always better.
  - **Intuition:** Narrow distribution; can easily query out-of-distribution values
- Using data directly from a replay buffer works well, if not better.
- **High-entropy** distributions over the state space are generally effective



Our new work on being robust to static datasets: [arxiv/1906.00949]

Adversarial Feature Matching (AFM)

- How can we create a sampling distribution that incorporates all major insights found so far?

Adversarial Feature Matching (AFM)

- How can we create a sampling distribution that incorporates all major insights found so far?

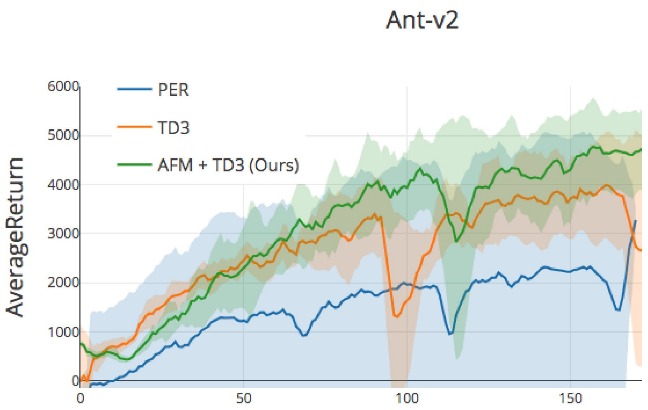
Key Idea: Learn distribution as a minimax game, with a feature matching constraint

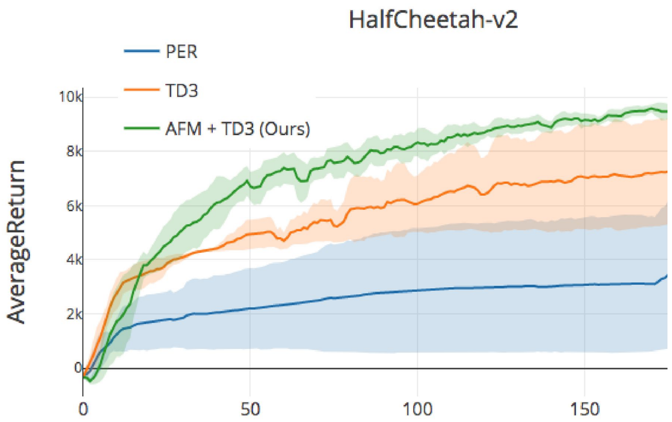
Minimax Objective

- **Prioritize** on states with high Bellman error *(Function Approx)*
- Enforce independence of features for different states *(Overfitting + Function Approx)*

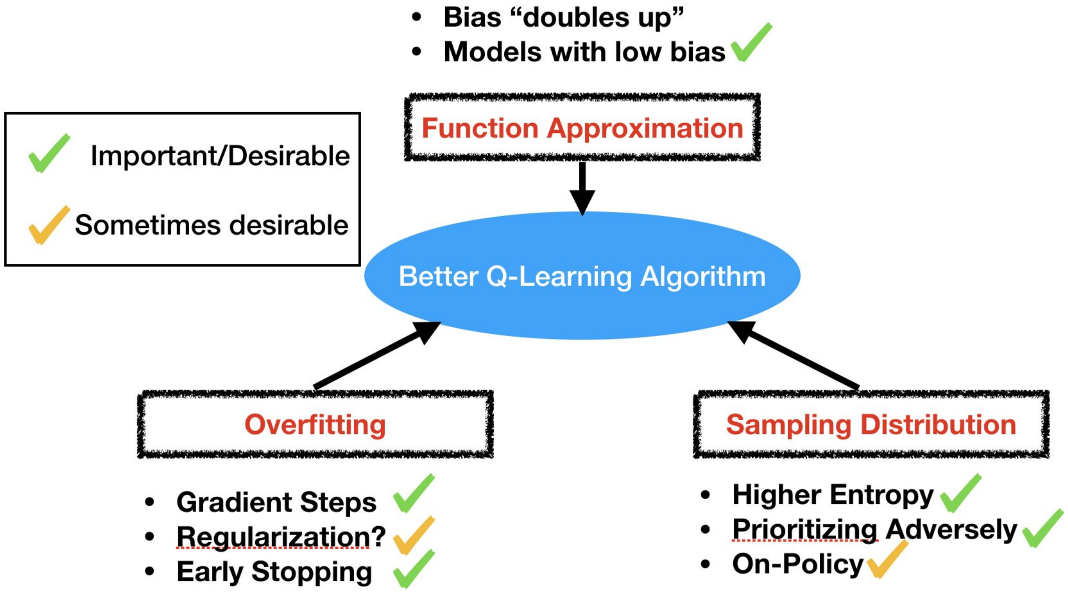
Feature Matching

Adversarial Feature Matching (AFM)





Generous improvement on MuJoCo tasks



Check out Poster #44

Code, Colab Notebooks available online!