hhu.

Offline Reinforcement Learning

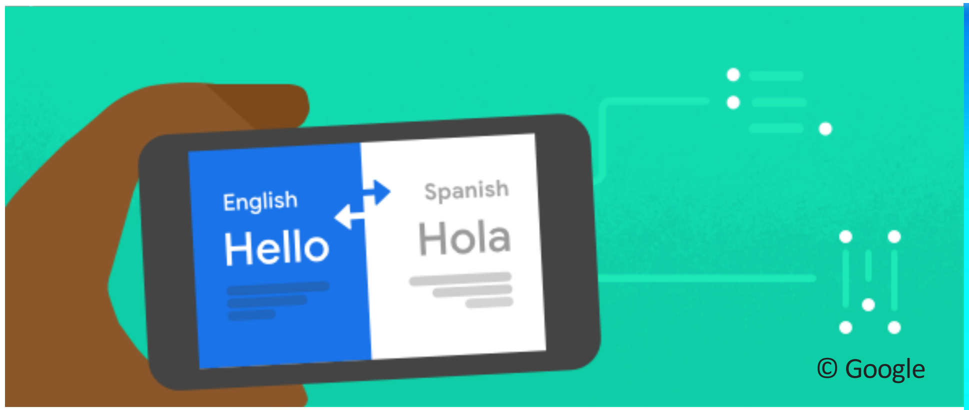
Dr. Nurul Lubis
Dialog Systems and Machine Learning Group
Heinrich-Heine University Düsseldorf

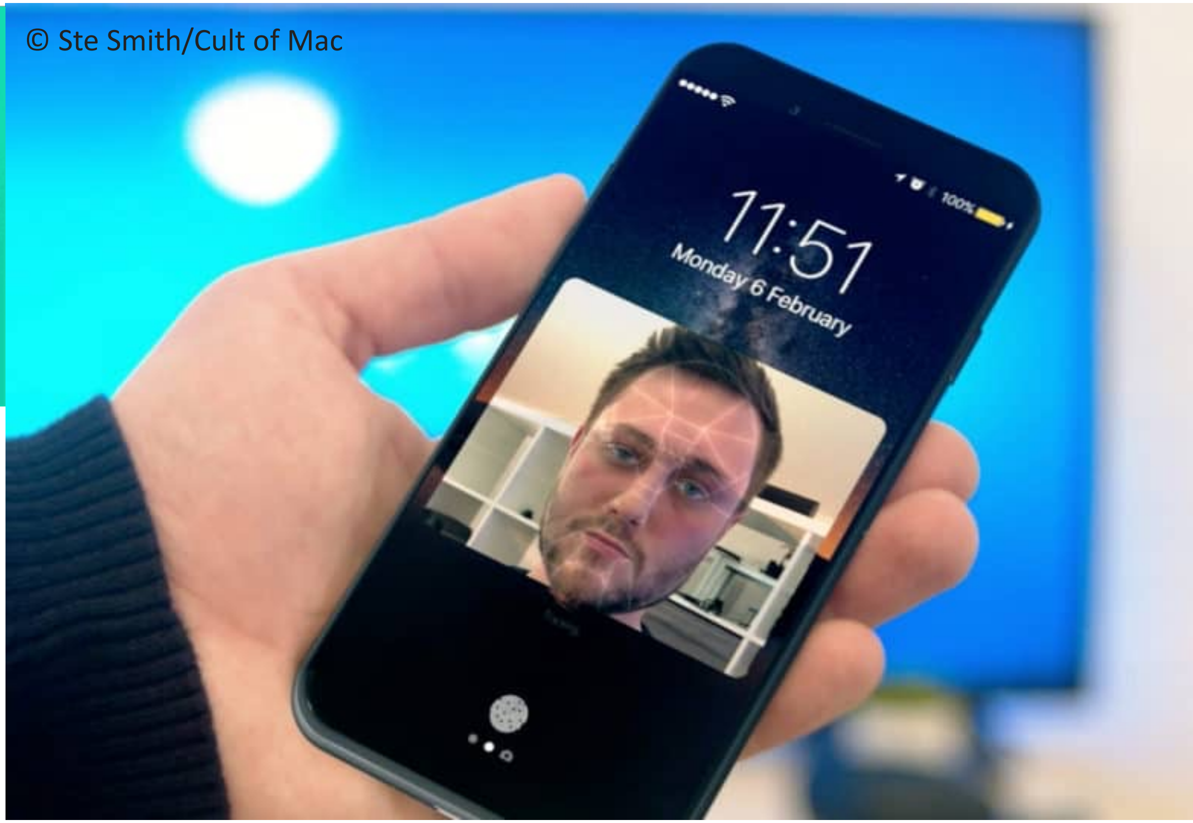
Foreword

- Introduction into the offline RL topic
- Focus on the intuition on the problem and existing solutions
  - Some details on a few approaches
- Many references to 2020 NeurIPS tutorial on Offline RL by Levine and Kumar
  - Highly useful resource for more details on the methods mentioned here

Success in Machine Learning

Heinrich Heine
Universität Düsseldorf





Huge models that generalize well, trained with huge amount of data

Success in Machine Learning

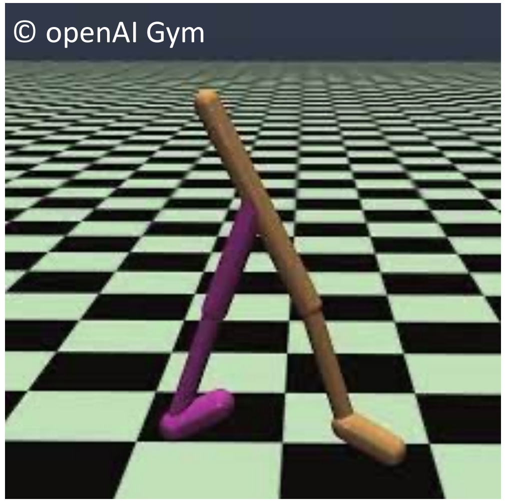
Heinrich Heine
Universität Düsseldorf

Supervised learning...

- Large amounts of data
- Deep NNs
- Generalizes to open world settings

In contrast, reinforcement learning...

- Learning through interactions
- Learn on specific tasks and small domains
- Lacking in terms of generalization



I'm looking for a hotel in the south

What price range do you want?

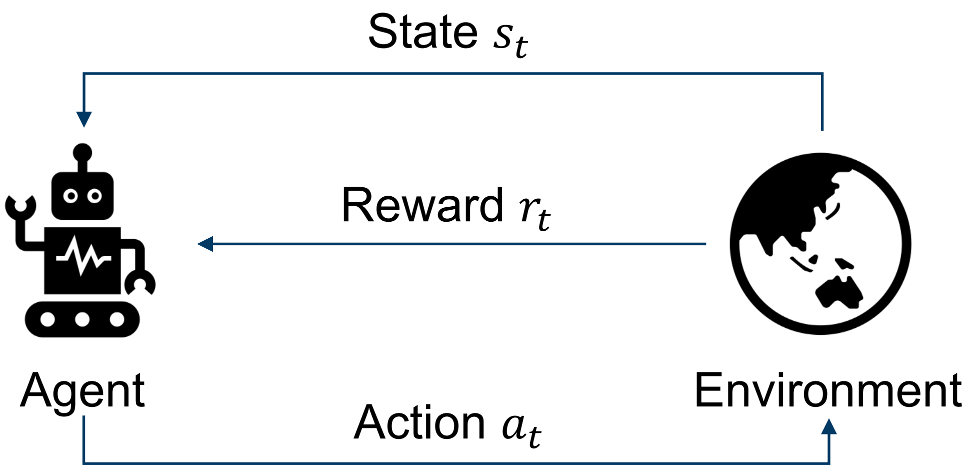
The Need for Data-driven RL

- Some environments are high-risk
  - Factory monitoring, self-driving cars, ..
- Some behavior we want to learn are highly complex
  - Medical field, education, ...
- Learning from online interaction can be expensive and time consuming
  - Dialogue systems
- Simulators?
  - Has its own challenges and limitations
  - Unnecessary in other learning paradigms

Can we leverage offline data to learn a policy?

RL Primer

RL Primer: Notations



- Through interactions with the environment, the agent try to find the best policy based on some measure of reward.
- Huge amount of interactions are needed

Trajectory τ
Sequence of state, action, reward tuples from a sequence of time steps (we assume a finite horizon case)

Return R_t
Discounted cumulative reward

$$R_t = \sum_{n \geq t} \gamma^{n-t} r_n$$

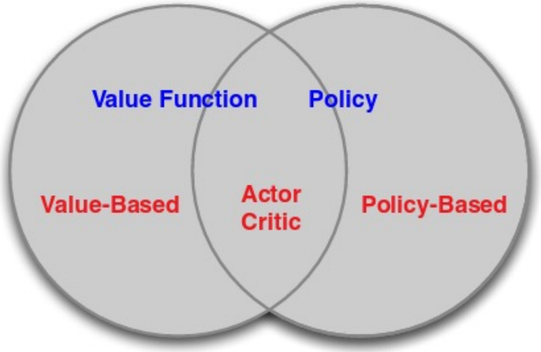
Policy $\pi(a|s)$
Probability distribution over actions in a given state

Value functions

- $V^{\pi}(s)$ expected return of being in state s and following policy π afterwards
- $Q^{\pi}(s, a)$ expected return of being in state s, taking action a, and following policy π afterwards

RL Primer: Learning Methods

- **Value-based**: Learn the optimal Q-function Q^* and act greedily
  - Starting with an arbitrary value function $Q(s,a)$, update at each time step to enforce the Bellman equation
    - $Q(s,a) = r(s,a) + \gamma \mathbb{E}_{a' \sim \pi(a'|s')} Q(s',a')$
  - For example with temporal difference (TD) target
  - Policy is defined implicitly
    - $\pi(s) = argmax_{a'}\, Q^*(s,a')$



- **Policy-based**: Directly parametrise the policy $\pi_\theta(a|s)$ using θ
  - For instance using a neural that outputs a softmax over possible actions given a state
  - REINFORCE (Williams, 1992): Update paramater to encourage actions that maximize return
    - $\nabla_\theta J(\theta) = \mathbb{E}_\theta[\sum_{t=0}^{T} R_t \nabla_\theta \log p_\theta(a_t|s_t)]$
- **Actor-Critic**: Learn both an actor $\pi_\theta(a|s)$ and a critic $\mathrm{Q}_\psi(s,a)$
  - Critic tries to approximate $Q^\pi(s,a)$
  - Improves on policy-based methods by trying to reduce variance

Slide adapted from Chris' talk on RL in 2020

RL Primer: Set Ups

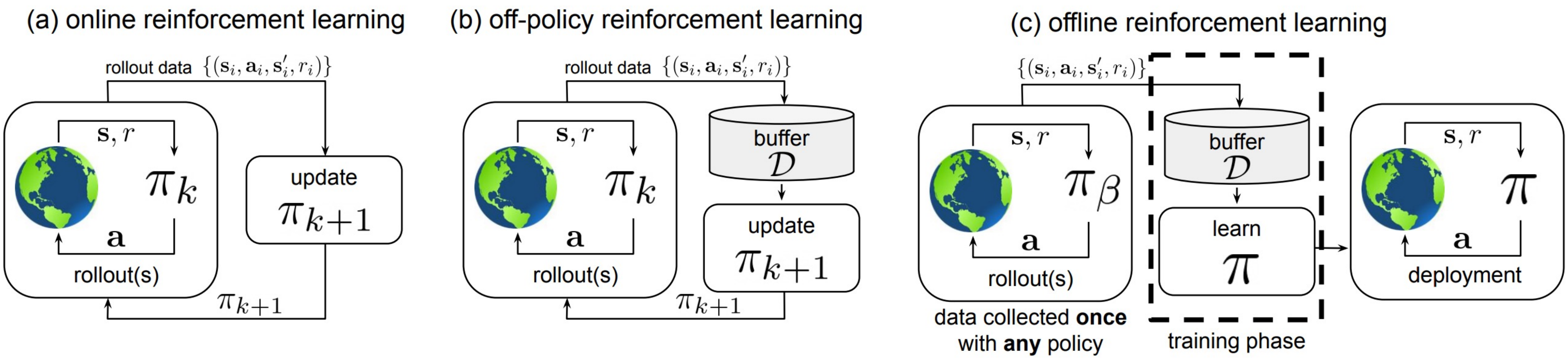


Figure from (Levine et al., 2020)

What makes offline RL challenging?

- Counterfactual decision making
  - Taking a different action than shown in data
  - A necessity in offline RL
- Distributional shift
  - State distribution
    - π has different state distribution than π_β
    - Even though we get high value on data, policy still could be bad during deployment
  - Sampling and estimation error
    - Erroneous estimates for unseen state and action pairs are not corrected
    - Exacerbated by choosing action to maximize the value function

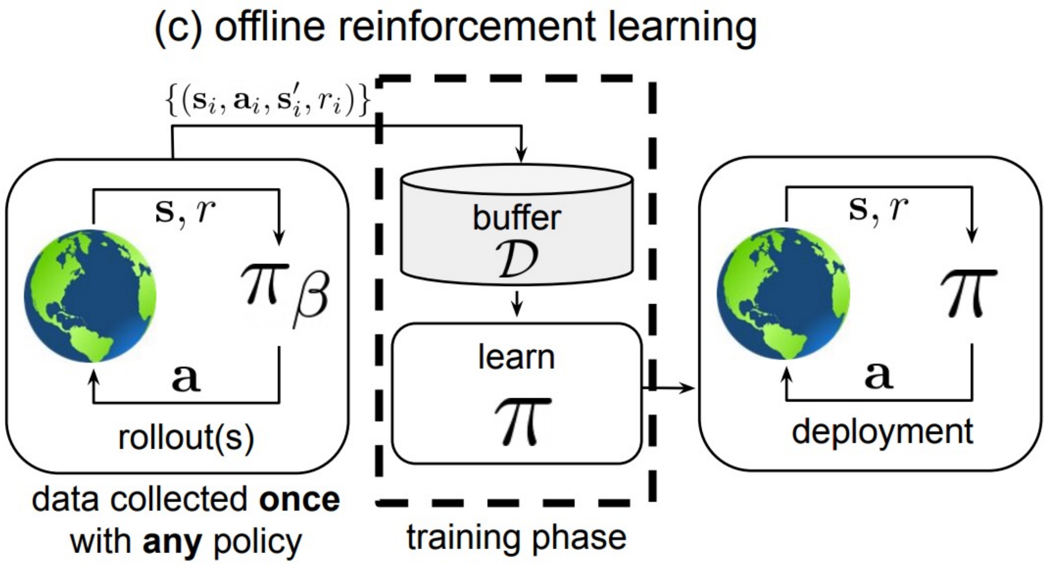


Figure from (Levine et al., 2020)

Offline RL methods

- Importance sampling
- Policy constraints
- Value regularization
- Model-based methods
- Uncertainty based methods
- ...

Importance sampling

Off-policy evaluation for offline RL

How can we estimate the return of our current policy, given trajectories from another policy?

- **Importance sampling** (Rubinstein, 1981) can be used to derive and unbiased estimator of $J(\pi)$ based on trajectories sampled from the behavior policy (Precup, 2000)
  - $J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\beta(\tau)}\left[\frac{\pi_\theta(\tau)}{\pi_\beta(\tau)} \sum_{t=0}^{T} \gamma^t r(s_t, a_t)\right]$
  - $\frac{\pi_\theta(\tau)}{\pi_\beta(\tau)} = \prod_{t=0}^{T} \frac{\pi_\theta(a_t|s_t)}{\pi_\beta(a_t|s_t)}$
- Drawbacks: very high variance and potentially unbounded

Off-policy evaluation for offline RL

How can we estimate the return of our current policy, given trajectories from another policy?

- Some solutions to reduce variance (in tradeoff with some bias) (Precup, 2000)
  - Self-normalizing: divide with the sum of weights
  - Per-decision importance sampling estimator: drop the weights from future time steps
- We can also use an estimate of the value $\hat{Q}(s_t, a_t)$ in place of the reward (Jiang and Li, 2015; Thomas and Brunskill 2016)
  - Reduces variance while keeping the estimate unbiased if π_β is known or $\hat{Q}(s_t, a_t)$ is correct

Off-policy evaluation for offline RL

Can we use this to estimate a policy gradient, and use that for policy update?

- Importance sampling can also be used to directly estimate policy *gradients* using trajectories from π_β
  - REINFORCE:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\beta(\tau)}\left[\frac{\pi_\theta(\tau)}{\pi_\beta(\tau)}\sum_{t=0}^{T}\gamma^t\nabla_\theta\log\pi_\theta(a_t|s_t)\,r(s_t,a_t)\right]$$

- The objective can also be derived for per-decision importance weight (Precup, 2000), or with value estimate instead of the return (Gu et al., 2017; Cheng et al., 2019; ...)

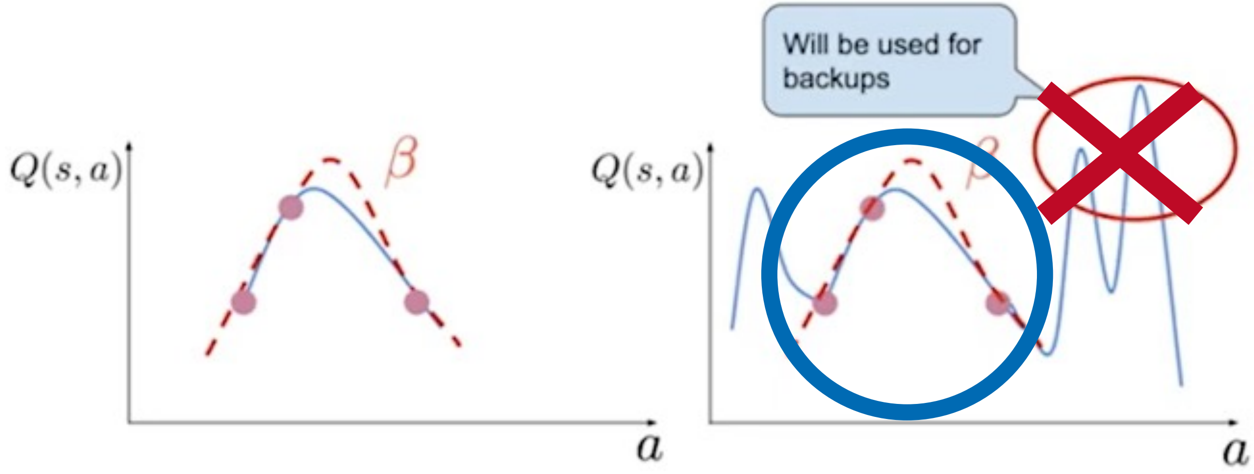
Challenges and open problems

- Typically used in off-policy setting, where we assume that we can collect additional data from interaction
- Application in offline RL has been limited
  - In practice, the variance is too high to work well in problems of interest
  - In sequential problems (with long horizon), exponential blowup could happen
- If π_β is too far from π_θ, the weights quickly become degenerate

Constraint methods

Policy constraints

- Constraining the action distribution of $\pi(a|s)$ to match the density of $\pi_\beta(a|s)$
  - Make sure that the actions taken by the learned policy is close enough to the action density of the behavior policy
  - $D\left(\pi(a|s), \pi_\beta(a|s)\right) \leq \epsilon$

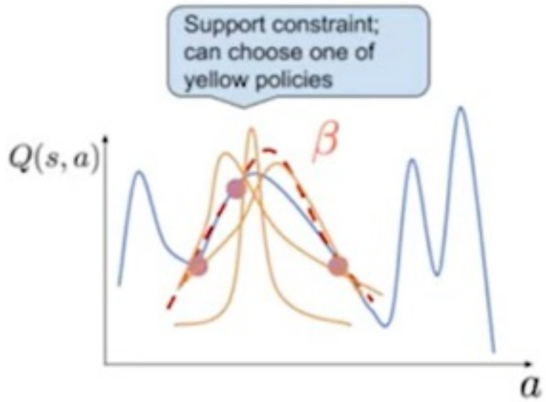


- Constraints can be solved explicitly
  - KL-Divergence (Jacques et al., 2019; Wu et al., 2019a)
  - F-Divergence (Wu et al., 2019b)
- Or implicitly
  - Add distance minimization into the objective, and express in closed form (Pang et al., 2019; Seigel et al., 2019; Wang et al., 2020; Nair et al. 2020)

Figures from NeurIPS 2020 tutorial on Offline RL

Support constraints

- Consider only actions that are within the support of the behavior policy π_β
  - Support: a set of action that are likely under the behavior policy
- Instead of matching the density of $\pi_\beta(a|s)$ as in policy constraints, here we compare the samples
  - Results in a more spiked density

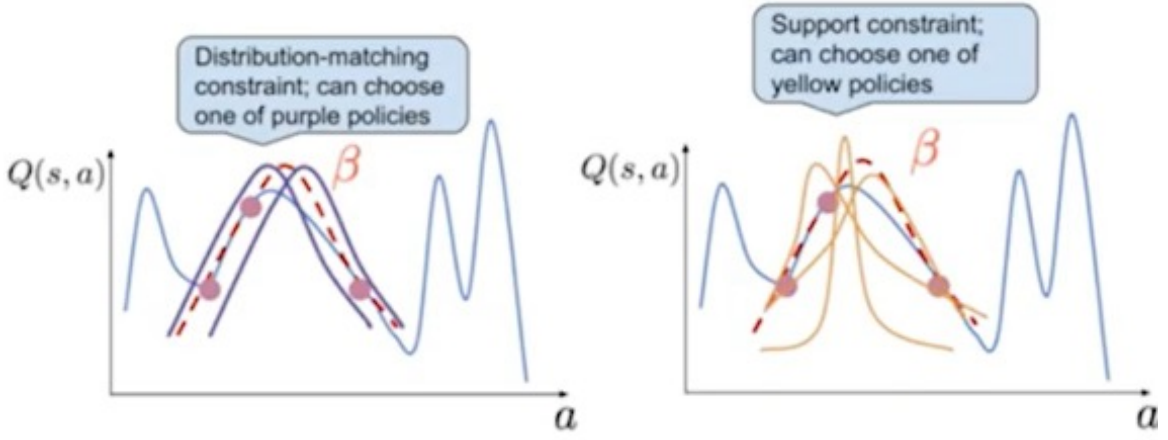


  - $argmax_{a \in \mathcal{D}[s]} Q(s,a)$ (Fujimoto et al., 2019; Ghasemipour et al., 2020; ...)

Figures from NeurIPS 2020 tutorial on Offline RL

Which one works better?

- *In theory*, support constraints
  - Can choose actions deterministically
  - More flexibility in choosing a policy
- With distribution matching, we always match the distribution even in suboptimal cases
  - May be too conservative



- Support constraints can outperform behavior cloning, but do not work well yet in more complex environments (Fu et al., 2020; Wu et al., 2020)
  - One major shortcoming is the need to estimate behavior policy π_β
  - If π_β is wrongly estimated, the learned policy will fail
    - as is the case in more complex environments

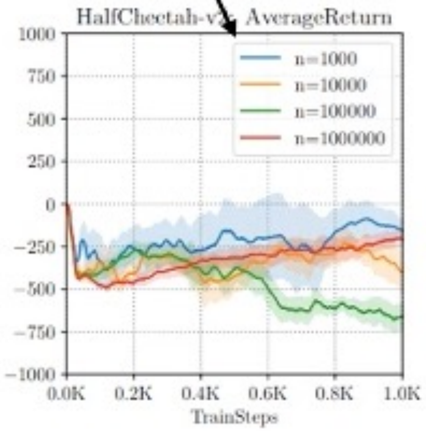
Figures from NeurIPS 2020 tutorial on Offline RL

Value regularization methods

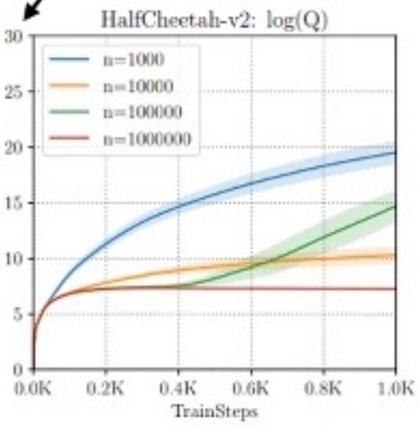
Overconfidence in Q-value estimation

amount of data

log scale (massive overestimation)



how well it does

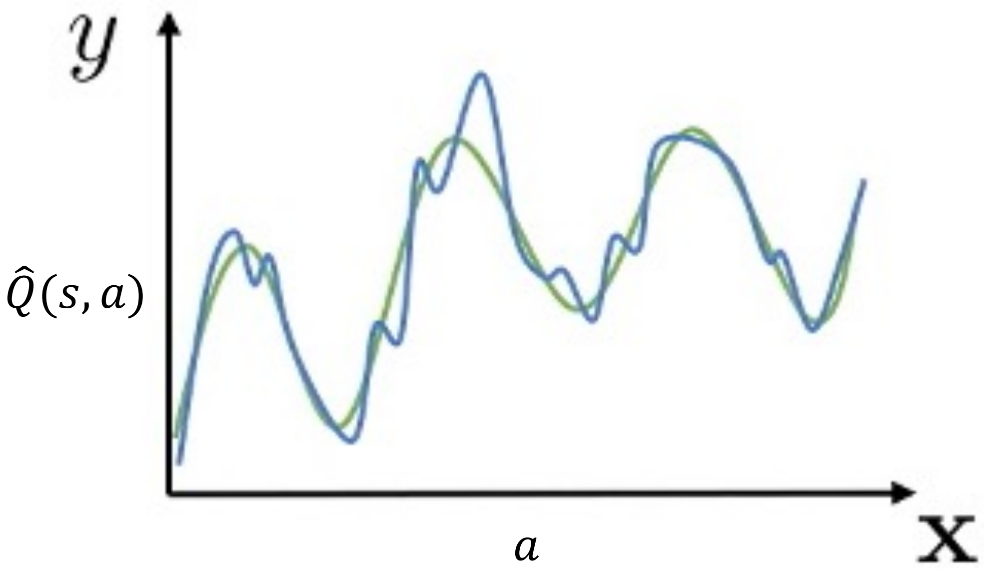


how well it *thinks* it does (Q-values)

- Huge discrepancy between its estimation and real return
  - Not something that larger amount of data can fix
- Value estimation on OOD actions can be unpredictable
- We expect good estimation when $\pi_\beta(a|s) = \pi(a|s)$
  - However, that is rarely the case, and may even be an *undesirable* case.

Figures from Kumar et al. (2019)

Overconfidence in Q-value estimation



Figures from NeurIPS 2020 tutorial on Offline RL

- Huge discrepancy between its estimation and real return
  - Not something that larger amount of data can fix
- Value estimation on OOD actions can be unpredictable
- We expect good estimation when $\pi_\beta(a|s) = \pi(a|s)$
  - However, that is rarely the case, and may even be an *undesirable* case.
  - Even worse, by choosing an action that maximizes the Q-value, we essentially choose actions where the estimate is most overconfident

Pessimistic value functions

Learning a conservative Q-function to act as a lowerbound of the true value

- Avoid overestimation, especially on OOD state-action pairs, where the estimation could be erroneously high

- We can add penalty to the value function based on the distance between the policies
  - $Q(s,a) \leftarrow r(s,a) + \gamma \mathbb{E}_{a' \sim \pi(a'|s')} Q(s',a') - \alpha D(\pi_\theta, \pi_\beta)$
  - For example, with KL-control (Jacques et al., 2018) or BRAC-v (Wu et al., 2019)
  - Drawback: still needs to estimate π_β

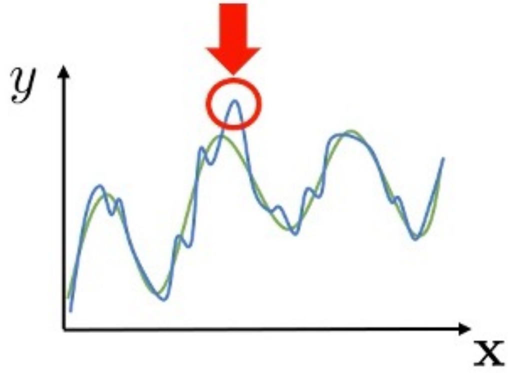
Conservative Q-learning

- We can regularize the objective directly to make this behavior inherent (Kumar et al., 2020)
  - Big Q-values for actions that are likely under the current policy is minimized
  - Q-values for state action pairs in the data will still be pushed up by the TD error
  - Shown to learn a lowerbound of Q-values on state action pairs contained in the data

$$\min_{Q}\max_{\mu} \overbrace{\mathbb{E}_{\mathbf{s}\sim\mathcal{D}}\mathbb{E}_{\mathbf{a}\sim\mu(\mathbf{a}|\mathbf{s})}\left[Q(\mathbf{s},\mathbf{a})\right]}^{\text{Minimize the “big” Q-values}} + \frac{1}{2\alpha}\overbrace{\mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'\sim\mathcal{D}}\left[\left(Q(\mathbf{s},\mathbf{a}) - y(\mathbf{s},\mathbf{a})\right)^2\right]}^{\text{Standard TD error objective}}$$

$$y(\mathbf{s},\mathbf{a}) = r(\mathbf{s},\mathbf{a}) + \gamma\mathbb{E}_{\mathbf{a}'\sim\pi_\theta(\mathbf{a}'|\mathbf{s}')}\left[Q(\mathbf{s}',\mathbf{a}')\right]$$

Target values



  - The learned Q-function can then be used in a standard Q-learning or actor-critic algorithms
  - Shown to work on more complex simulation environments
- Drawback: the policy and the value function works in an adversarial manner, so training can be unstable

Figures from NeurIPS 2020 tutorial on Offline RL

Open challenges

Heinrich Heine Universität Düsseldorf

- Conservative value estimation may underfit on small data, leading to excessive pessimism
  - Value for undersampled actions may be estimated too low
  - How to balance the risk of overestimation while still exploring OOD actions?

Other approaches

Heinrich Heine Universität Düsseldorf

- Model-based
  - Train an ensemble of dynamics model
  - Use their agreement as a measure of uncertainty to penalize the reward
- Other uncertainty-based methods
  - Train multiple Q-functions and use multiple predictions to estimate uncertainty
- ...

Applications

- Robotics
  - Object grasping is particularly interesting as it requires generalization.
  - Navigating a room using human demonstration
- Healthcare
  - Exclusively offline, due to high risk of online exploration
  - Works towards treatments for epilepsy, schizophrenia, and more
- Autonomous driving
  - More datasets containing human driving activity are being released
  - Offline RL hasn't been successfully applied yet
- Advertising and recommender systems
  - Off policy evaluation is commonly used to perform A/B testing
  - Optimize visit and clicks based on user activity logs
- Language and Dialogue
  - Learning from readily available human dialogue, e.g. dialogue data from customer service

Conclusion

- Offline RL aims to learn a policy using previously collected data, without further interaction with the deployment environment
  - Towards scalable RL towards solving more complex real-world problems
- Major challenges:
  - Counterfactual decision making
  - Distributional shift
- Some solutions:
  - Importance sampling to address the distribution mismatch
  - Constrained policy update, making sure the policy stays close to the behavior policy
  - Conservative value estimation, underestimate the value on OOD state-action pairs
  - ...
- Have been applied in various fields, from robotics to dialogue
- Actively developing area of research

References

Levine, Sergey, et al. "Offline reinforcement learning: Tutorial, review, and perspectives on open problems." *arXiv preprint arXiv:2005.01643* (2020).

Papers cited in this talk can be found on the bibliography