Offline Reinforcement Learning

CS 285

Instructor: Sergey Levine
UC Berkeley
The generalization gap

Mnih et al. ‘13
Schulman et al. ‘14 & ‘15
Levine*, Finn*, et al. ‘16

this is done many times

enormous gulf
What makes modern machine learning work?
Can we develop data-driven RL methods?

on-policy RL

- rollout data $\{(s_i, a_i, s'_i, r_i)\}$
- $S, \gamma$
- $\pi_k$
- update $\pi_{k+1}$
- rollout(s)
- $\pi_k + 1$

off-policy RL

- rollout data $\{(s_i, a_i, s'_i, r_i)\}$
- $S, \gamma$
- $\pi_k$
- buffer $\mathcal{D}$
- update $\pi_{k+1}$
- rollout(s)
- $\pi_k + 1$

big datasets from past interaction

occasionally get more data

train for many epochs

What does offline RL mean?

Formally:

\[ D = \{(s_i, a_i, s'_i, r_i)\} \]

\[ s \sim d^{\pi_\beta}(s) \]

\[ a \sim \pi_\beta(a|s) \]

\[ s' \sim p(s'|s, a) \]

\[ r \leftarrow r(s, a) \]

RL objective: \[ \max_{\pi} \sum_{t=0}^{T} E_{s_t \sim d(s), a_t \sim \pi(a|s)} [\gamma^t r(s_t, a_t)] \]
Types of offline RL problems

off-policy evaluation (OPE):

\[ \mathcal{D} = \{(s_i, a_i, s'_i, r_i)\} \]
\[ s \sim d^{\pi_\beta}(s) \]
\[ a \sim \pi_\beta(a|s) \]
\[ s' \sim p(s'|s, a) \]
\[ r \leftarrow r(s, a) \]

given \( \mathcal{D} \), estimate \( J(\pi) = E_\pi \left[ \sum_{t=1}^{T} r(s_t, a_t) \right] \)

offline reinforcement learning: (a.k.a. batch RL, sometimes fully off-policy RL)

given \( \mathcal{D} \), learn the best possible policy \( \pi_\theta \)

not necessarily obvious what this means
How is this even possible?

1. Find the “good stuff” in a dataset full of good and bad behaviors

2. Generalization: good behavior in one place may suggest good behavior in another place

3. “Stitching”: parts of good behaviors can be recombined
What do we expect offline RL methods to do?

**Bad intuition:** It’s like imitation learning

Though it can be shown to be **provably** better than imitation learning even with optimal data, under some structural assumptions!

See: Kumar, Hong, Singh, Levine. Should I Run Offline Reinforcement Learning or Behavioral Cloning?

**Better intuition:** Get order from chaos

“If we have algorithms that properly perform dynamic programming, we can take this idea much further and get near-optimal policies from highly suboptimal data”
A vivid example

RL policies typically don’t generalize to initial conditions that were not seen during training

Can we use previously collected, unlabeled datasets to extend learned skills?

training task  closed drawer  blocked by drawer  blocked by object

Singh, Yu, Yang, Zhang, Kumar, Levine. COG: Connecting New Skills to Past Experience with Offline Reinforcement Learning. ‘20
Why should we care?

this is done many times
Does it work?

stored data from all past experiments
\{(s_i, a_i, s'_i)\}_{i}

live data collection

training buffers
- off-policy \((s, a, s', r)\)
- on-policy \((s, a, s', r)\)
- labeled \((s, a, Q_T(s, a))\)

Bellman updaters
compute \(Q_T(s, a) = r + \max_{a'} Q_\theta(s', a')\)

training threads
\[ \min_\theta ||Q_\theta(s, a) - Q_T(s, a)||^2 \]
Does it work?

<table>
<thead>
<tr>
<th>Method</th>
<th>Dataset</th>
<th>Success</th>
<th>Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offline QT-Opt</td>
<td>580k offline</td>
<td>87%</td>
<td>13%</td>
</tr>
<tr>
<td>Finetuned QT-Opt</td>
<td>580k offline + 28k online</td>
<td>96%</td>
<td>4%</td>
</tr>
</tbody>
</table>
Why is offline RL hard?

amount of data

log scale (massive overestimation)

how well it does

how well it thinks it does (Q-values)

Kumar, Fu, Tucker, Levine. Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction. NeurIPS ‘19
Why is offline RL hard?

**Fundamental problem:** counterfactual queries

**Training data**

---

**What the policy wants to do**

---

Is this good? Bad?

How do we know if we didn’t see it in the data?

---

**Online RL** algorithms don’t have to handle this, because they can simply *try* this action and see what happens

**Offline RL** methods must somehow account for these unseen (“out-of-distribution”) actions, ideally in a safe way

...while still making use of generalization to come up with behaviors that are better than the best thing seen in the data!
Distribution shift in a nutshell

Example empirical risk minimization (ERM) problem:

\[ \theta \leftarrow \arg \min_{\theta} E_{x \sim p(x), y \sim p(y|x)} (f_\theta(x) - y)^2 \]

given some \( x^* \), is \( f_\theta(x^*) \) correct?

\[ E_{x \sim p(x), y \sim p(y|x)} (f_\theta(x) - y)^2 \] is low

\[ E_{x \sim \bar{p}(x), y \sim p(y|x)} (f_\theta(x) - y)^2 \] is not, for general \( \bar{p}(x) \neq p(x) \)

what if \( x^* \sim p(x) \)? not necessarily...

usually we are not worried – neural nets generalize well!

what if we pick \( x^* \leftarrow \arg \max_x f_\theta(x) \)?

Kumar, Fu, Tucker, Levine. Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction. NeurIPS ‘19
Where do we suffer from distribution shift?

\[ Q(s, a) \leftarrow r(s, a) + \max_{a'} Q(s', a') \]

\[ Q(s, a) \leftarrow r(s, a) + E_{a' \sim \pi_{\text{new}}}[Q(s', a')] \]

\[ y(s, a) \]

what is the objective?

\[ \min_{Q} E_{(s,a) \sim \pi_{\beta}(s,a)} [(Q(s, a) - y(s, a))^2] \]

behavior policy

target value

expect good accuracy when \( \pi_{\beta}(a|s) = \pi_{\text{new}}(a|s) \)

even worse: \( \pi_{\text{new}} = \arg \max_{\pi} E_{a \sim \pi(a|s)}[Q(s, a)] \)

(what if we pick \( x^* \leftarrow \arg \max_{x} f_\theta(x) \)?)

how often does that happen?

how well it does

how well it thinks it does (Q-values)

Kumar, Fu, Tucker, Levine. Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction. NeurIPS ’19
Issues with generalization are not corrected

Existing challenges with sampling error and function approximation error in standard RL become much more severe in offline RL.
Batch RL via Importance Sampling
Offline RL with policy gradients

RL objective: $\max_{\pi} \sum_{t=0}^{T} E_{s_t \sim d^\pi(s), a_t \sim \pi(a|s)} [\gamma^t r(s_t, a_t)]$

$\nabla_\theta J(\theta) = E_{\tau \sim \pi_\theta(\tau)} \left[ \sum_{t=0}^{T} \nabla_\theta \gamma^t \log \pi_\theta(a_t|s_t) \hat{Q}(s_t, a_t) \right]$

$\approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T} \nabla_\theta \gamma^t \log \pi_\theta(a_{t,i}|s_{t,i}) \hat{Q}(s_{t,i}, a_{t,i})$

requires sampling from $\pi_\theta$! what if we only have samples from $\pi_\beta$?

importance sampling:

$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \frac{\pi_\theta(\tau_i)}{\pi_\beta(\tau_i)} \sum_{t=0}^{T} \nabla_\theta \gamma^t \log \pi_\theta(a_{t,i}|s_{t,i}) \hat{Q}(s_{t,i}, a_{t,i})$

importance weight
Offline RL with policy gradients

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \frac{\pi_\theta(\tau_i)}{\pi_\beta(\tau_i)} \sum_{t=0}^{T} \nabla_\theta \gamma^t \log \pi_\theta(a_{t,i}|s_{t,i}) \hat{Q}(s_{t,i}, a_{t,i}) \]

\[ \quad \rightarrow E_{\pi_\theta} \left[ \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'} \right] \approx \sum_{t'=t}^{T} \gamma^{t'-t} r_{t',i} \]

\[ \frac{\pi_\theta(\tau)}{\pi_\beta(\tau)} = \frac{p(s_1) \prod_{t=1}^{T} p(s_{t+1}|s_t, a_t) \pi_\theta(a_t|s_t)}{p(s_1) \prod_{t=1}^{T} p(s_{t+1}|s_t, a_t) \pi_\beta(a_t|s_t)} \]

this is exponential in \( T \)
weights likely to be degenerate as \( T \) becomes large

can we fix this?

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T} \left( \prod_{t'=0}^{t-1} \frac{\pi_\theta(a_{t',i}|s_{t',i})}{\pi_\beta(a_{t',i}|s_{t',i})} \right) \nabla_\theta \gamma^t \log \pi_\theta(a_{t,i}|s_{t,i}) \left( \prod_{t'=t}^{T} \frac{\pi_\theta(a_{t',i}|s_{t',i})}{\pi_\beta(a_{t',i}|s_{t',i})} \right) \hat{Q}(s_{t,i}, a_{t,i}) \]

accounts for difference in probability of landing in \( s_{t,i} \)
we have \( s_t \sim d^\pi(s_t) \), but want \( s_t \sim d^\pi_\alpha(s_t) \)

why is this a reasonable approximation?
Estimating the returns

\[ \nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T} \nabla_{\theta} \gamma^t \log \pi_\theta(a_{t,i} \mid s_{t,i}) \left( \prod_{t' = t}^{T} \frac{\pi_\theta(a_{t',i} \mid s_{t',i})}{\pi_\beta(a_{t',i} \mid s_{t',i})} \right) \hat{Q}(s_{t,i}, a_{t,i}) \]

\[ \approx E_{\pi_\theta} \left[ \sum_{t' = t}^{T} \gamma^{t' - t} r_{t'} \right] \approx \sum_{t' = t}^{T} \gamma^{t' - t} r_{t',i} \]

\[ \sum_{t' = t}^{T} \left( \prod_{t'' = t}^{t'} \frac{\pi_\theta(a_{t'',i} \mid s_{t'',i})}{\pi_\beta(a_{t'',i} \mid s_{t'',i})} \right) \gamma^{t' - t} r_{t',i} \]

but this is still exponential!

To avoid exponentially exploding importance weights, we must use value function estimation!

Imagine we knew \( Q^{\pi_\theta}(s, a) \)

We’ll see how to do this shortly, but first let’s conclude our discussion of importance sampling.
The doubly robust estimator

\[ V^{\pi_{\theta}}(s_0) \approx \sum_{t' = 0}^{T} \left( \prod_{t' = t}^{t'} \frac{\pi_{\theta}(a_{t'}|s_{t'}, s_{t'})}{\pi_{\beta}(a_{t'}|s_{t'}, s_{t'})} \right) r_{t'} - r_{t', i} \]

\[ = \sum_{t=0}^{T} \left( \prod_{t' = 0}^{t} \rho_{t'} \right) \gamma^{t} r_{t} \]

\[ = \rho_{0} r_{0} + \rho_{0} \gamma \rho_{1} r_{1} + \rho_{0} \gamma \rho_{1} \gamma \rho_{2} r_{2} + \ldots \]

\[ = \rho_{0}(r_{0} + \gamma(\rho_{1}(r_{1} + \gamma(\rho_{2}(r_{2} + \gamma(\ldots)))))) \]

\[ = \bar{V}^{T} \quad \text{where} \quad \bar{V}^{T+1-t} = \rho_{t}(r_{t} + \gamma \bar{V}^{T-t}) \]

\[ \bar{V}_{\text{DR}}^{T+1-t} = \hat{V}(s_{t}) + \rho_{t}(r_{t} + \gamma \bar{V}_{\text{DR}}^{T-t} - \hat{Q}(s_{t}, a_{t})) \]

Marginalized importance sampling

**Main idea:** instead of using $\prod_t \frac{\pi_\theta(a_t|s_t)}{\pi_\beta(a_t|s_t)}$, estimate $w(s, a) = \frac{d^{\pi_\theta}(s,a)}{d^{\pi_\beta}(s,a)}$

if we can do this, we can estimate $J(\theta) \approx \frac{1}{N} \sum_i w(s_i, a_i)r_i$

typically this is done for off-policy evaluation, rather than policy learning

how to determine $w(s, a)$? typically solve some kind of consistency condition

example (Zhang et al., GenDICE):

$$d^{\pi_\beta}(s', a')w(s', a') = (1-\gamma)p_0(s')\pi_\theta(a'|s') + \gamma \sum_{s,a} \pi_\theta(a'|s')p(s'|s,a)d^{\pi_\beta}(s,a)w(s,a)$$

probability of starting in $(s', a')$  

probability of transitioning into $(s', a')$

solving for $w(s, a)$ typically involves some fixed point problem
Additional readings: importance sampling

Classic work on importance sampled policy gradients and return estimation:

Doubly robust estimators and other improved importance-sampling estimators:

Analysis and theory:

Marginalized importance sampling:

Batch RL via Linear Fitted Value Functions
Offline value function estimation

How have people thought about it before?

Extend existing ideas for approximate dynamic programming and Q-learning to offline setting

Derive tractable solutions with simple (e.g., linear) function approximators

How are people thinking about it now?

Derive approximate solutions with highly expressive function approximators (e.g., deep nets)

The primary challenge turns out to be **distributional shift**

generally not concerned with distributional shift before

(maybe it was not such a big problem with linear models)

We’ll discuss some older offline/batch RL methods next for completeness
Warmup: linear models

\( \Phi \) – feature matrix, \(|S| \times K \)

could also think of as a vector-valued function \( \Phi(s) \)

Can we do (offline) model-based RL in feature space?

1. Estimate the reward
2. Estimate the transitions
3. Recover the value function
4. Improve the policy

1. Reward model: \( \Phi w_r \approx r \)  
   least squares: \( w_r = (\Phi^T \Phi)^{-1} \Phi^T r \)

2. Transition model: \( \Phi P_\Phi \approx P_\pi \Phi \)  
   least squares: \( P_\Phi = (\Phi^T \Phi)^{-1} \Phi^T P_\pi \Phi \)

estimated feature-space transition matrix \( K \times K \)  
real transition matrix (on states) \( |S| \times |S| \)

all of this is for a fixed policy \( \pi \)

data adapted from Ron Parr
Recovering the value function

1. Reward model: $\Phi w_r \approx r$
   least squares: $w_r = (\Phi^T \Phi)^{-1} \Phi r$

2. Transition model: $\Phi P_\Phi \approx P_\pi \Phi$
   least squares: $P_\Phi = (\Phi^T \Phi)^{-1} \Phi P_\pi \Phi$

3. Estimate $V_\pi \approx V_\Phi = \Phi w_V$
   can apply the same equation in feature space:
   $w_V = (I - \gamma P_\Phi)^{-1} w_r$

   but wait – do we even need the model?

   $w_V = (I - \gamma (\Phi^T \Phi)^{-1} \Phi^T P_\pi \Phi)^{-1} (\Phi^T \Phi)^{-1} \Phi^T r$

   after a bit of algebra...

   $w_V = (\Phi^T \Phi - \gamma \Phi^T P_\pi \Phi)^{-1} \Phi^T r$

   this is called least-squares temporal difference (LSTD)

Aside: solving for $V_\pi$ in terms of $P_\pi$ and $r$:

$V_\pi = r + \gamma P_\pi V_\pi$

$(I - \gamma P_\pi)V_\pi = r$

$V_\pi = (I - \gamma P_\pi)^{-1} r$

---

material adapted from Ron Parr
Doing it all with samples

$$w_V = (\Phi^T \Phi - \gamma \Phi^T P^\pi \Phi)^{-1} \Phi^T \vec{r}$$

$$D = \{(s_i, a_i, r_i, s'_i)\}$$

replace with $$\Phi'$$

$$\Phi'_i = \phi(s'_i)$$

$$\vec{r}_i = r(s_i, a_i)$$

Everything else works **exactly** the same way, only now we have some sampling error

material adapted from Ron Parr
Improving the policy

1. Estimate the reward
2. Estimate the transitions
3. Recover the value function
4. Improve the policy

or just do these together with LSTD!

That’s not going to work for offline RL!

\[ \mathbf{w}_V = (\Phi^T \Phi - \gamma \Phi^T \mathbf{P}^\pi \Phi)^{-1} \Phi^T \tilde{r} \]

\[ \mathcal{D} = \{(s_i, a_i, r_i, s'_i)\} \]

\[ \tilde{r}_i = r(s_i, a_i) \]

replace with \( \Phi' \)

\[ \Phi'_i = \phi(s'_i) \]

this requires samples from \( \pi \)!

material adapted from Ron Parr
Least-squares policy iteration (LSPI)

Main idea: replace LSTD with LSTDQ – LSTD but for Q-functions

\[ w_Q = (\Phi^T \Phi - \gamma \Phi^T \Phi')^{-1} \Phi^T \bar{r} \]

\[ \mathcal{D} = \{(s_i, a_i, r_i, s'_i)\} \]

\[ \Phi'_i = \phi(s'_i, \pi(s'_i)) \]

LSPI:
1. compute \( w_Q \) for \( \pi_k \)
2. \( \pi_{k+1}(s) = \arg \max_a \phi(s, a)w_Q \)
3. Set \( \Phi'_i = \phi(s'_i, \pi_{k+1}(s'_i)) \)

Material adapted from Ron Parr
What’s the issue?

In general, all approximate dynamic programming (e.g., fitted value/Q iteration) methods will suffer from action distributional shift, and we must fix it!

\[ Q(s, a) \leftarrow r(s, a) + E_{a' \sim \pi_{\text{new}}}[Q(s', a')] \]

\[ y(s, a) \]

\[ \min_{Q} E_{(s,a) \sim \pi_{\beta}(s,a)} [(Q(s, a) - y(s, a))^2] \]

behavior policy
target value

expect good accuracy when \( \pi_{\beta}(a|s) = \pi_{\text{new}}(a|s) \)

how often does \textit{that} happen?

even worse: \( \pi_{\text{new}} = \arg \max_{\pi} E_{a \sim \pi(a|s)}[Q(s, a)] \)