Off-policy Policy Optimization

Dale Schuurmans
The RL problem

Environment

Agent

1. multi-agent interaction
2. partial observability
3. exploration
4. sequential decisions
5. exploitation

➡ non-stationarity
➡ must construct memory
➡ explore/exploit trade-off
➡ temporal credit assignment
➡ policy optimization
The RL problem

1. multi-agent interaction  ➡  non-stationarity
2. partial observability  ➡  must construct memory
3. exploration  ➡  explore/exploit trade-off
4. sequential decisions  ➡  temporal credit assignment
5. exploitation  ➡  policy optimization

“Textbook” RL
The RL problem

1. multi-agent interaction ➡ non-stationarity
2. partial observability ➡ must construct memory
3. exploration ➡ explore/exploit trade-off
4. sequential decisions ➡ temporal credit assignment
5. exploitation ➡ policy optimization

“Batch” RL
The RL problem

1. multi-agent interaction  ➡  non-stationarity
2. partial observability  ➡  must construct memory
3. exploration  ➡  explore/exploit trade-off
4. sequential decisions  ➡  temporal credit assignment
5. exploitation  ➡  policy optimization

"Batch contextual bandits"
Optimizing *one step* decision making

Batch contextual bandits
Batch policy optimization

Given data

\[
\begin{array}{ccccc}
  & a_1 & a_2 & \ldots & a_n \\
 x_1 & & & & r_1 \\
x_2 & & r_2 & & \\
x_3 & r_3 & & r_4 & \\
x_4 & r_5 & & & \\
x_5 & & & r_6 & \\
x_6 & & & & \vdots \\
\vdots & & & & \\
x_m & & & & r_m \\
\end{array}
\]

Optimize policy \( \pi : X \to \Delta^n \) to maximize expected reward on test contexts

\[
\pi(a \mid x) = e^{q(x)_a - F(q(x))}
\]

\[
F(q(x)) = \log \sum_a e^{q(x)_a}
\]

\[
q : X \to \mathbb{R}^n \quad \text{neural network}
\]
Batch policy optimization

Given data

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$a_1$</th>
<th>$r_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_2$</td>
<td>$a_2$</td>
<td>$r_2$</td>
</tr>
<tr>
<td>$x_3$</td>
<td></td>
<td>$r_3$</td>
</tr>
<tr>
<td>$x_4$</td>
<td></td>
<td>$r_4$</td>
</tr>
<tr>
<td>$x_5$</td>
<td></td>
<td>$r_5$</td>
</tr>
<tr>
<td>$x_6$</td>
<td></td>
<td>$r_6$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td></td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$x_m$</td>
<td></td>
<td>$r_m$</td>
</tr>
</tbody>
</table>

Optimize policy $\pi : X \to \Delta^n$

$\pi(a | x) = e^{q(x)a - F(q(x))}$

$F(q(x)) = \log \sum_a e^{q(x)a}$

$q : X \to \mathbb{R}^n$ neural network

Three key issues

1. generalization
2. optimization
3. missing data

to maximize expected reward on test contexts
## Batch policy optimization

**Given data**

| \( x_1 \) | \( a_1 \) | \( r_1, \beta_1 \) |
| \( x_2 \) | \( a_2 \) | \( r_2, \beta_2 \) |
| \( x_3 \) | \( a_3 \) | \( r_3, \beta_3 \) |
| \( x_4 \) | \( a_4 \) | \( r_4, \beta_4 \) |
| \( x_5 \) | \( a_5 \) | \( r_5, \beta_5 \) |
| \( x_6 \) | \( \vdots \) |
| \( \ldots \) | \( \vdots \) |
| \( x_m \) | \( a_m \) | \( r_m, \beta_m \) |

Isn’t this a solved problem?

We know how to do this, right?

**Default answer**

- maximize importance corrected expected reward
- (assume have proposal probabilities)

\[
\text{max} \sum_i \frac{\pi(a_i | x_i)}{\beta_i} r_i
\]

Optimize policy \( \pi : X \to \Delta^n \)

\[
\pi(a | x) = e^{q(x)a - F(q(x))}
\]

\[
F(q(x)) = \log \sum_a e^{q(x)a}
\]

\( q : X \to \mathbb{R}^n \) neural network
Given data

\[
\begin{array}{cccc}
  x_1 & x_2 & \cdots & x_m \\
  a_1 & a_2 & \cdots & a_n \\
  r_1 & r_2 & \cdots & r_m \\
\end{array}
\]

Optimize policy \( \pi : X \rightarrow \Delta^n \)

\[
\pi(a | x) = e^{q(x)_a - F(q(x))}
\]

\[
F(q(x)) = \log \sum_a e^{q(x)_a}
\]

\[
q : X \rightarrow \mathbb{R}^n \quad \text{neural network}
\]

Three key issues

1. generalization
   - ok
2. optimization
   - bad
3. missing data
   - bad

Importance corrected expected reward

Okay!
Optimization objectives

Given data

\[
\begin{array}{cccc}
  x_1 & a_1 & r_1 \\
  x_2 & a_2 & r_2 \\
  x_3 & & r_3 \\
  x_4 & & r_4 \\
  x_5 & & r_5 \\
  x_6 & & r_6 \\
  \vdots & & \vdots \\
  x_m & & r_m \\
\end{array}
\]

Optimize policy \( \pi : X \to \Delta^n \) to maximize expected reward on test contexts

\[
\pi(a | x) = e^{q(x)_a - F(q(x))}
\]

\[
F(q(x)) = \log \sum_a e^{q(x)_a}
\]

\( q : X \to \mathbb{R}^n \) neural network
Optimization objectives

Now assume given **complete** data

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$r_{11}$</th>
<th>$r_{12}$</th>
<th>$r_{1...}$</th>
<th>$r_{1n}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_2$</td>
<td>$r_{21}$</td>
<td>$r_{22}$</td>
<td>$r_{2...}$</td>
<td>$r_{2n}$</td>
</tr>
<tr>
<td>$x_3$</td>
<td>$r_{31}$</td>
<td>$r_{32}$</td>
<td>$r_{3...}$</td>
<td>$r_{3n}$</td>
</tr>
<tr>
<td>$x_4$</td>
<td>$r_{41}$</td>
<td>$r_{42}$</td>
<td>$r_{4...}$</td>
<td>$r_{4n}$</td>
</tr>
<tr>
<td>$x_5$</td>
<td>$r_{51}$</td>
<td>$r_{52}$</td>
<td>$r_{5...}$</td>
<td>$r_{5n}$</td>
</tr>
<tr>
<td>$x_6$</td>
<td>$r_{61}$</td>
<td>$r_{62}$</td>
<td>$r_{6...}$</td>
<td>$r_{6n}$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$x_m$</td>
<td>$r_{m1}$</td>
<td>$r_{m2}$</td>
<td>$r_{m...}$</td>
<td>$r_{mn}$</td>
</tr>
</tbody>
</table>

Optimize policy $\pi : X \rightarrow \Delta^n$ to maximize expected reward on test contexts

$$\pi(a | x) = e^{q(x)a - F(q(x))}$$

$$F(q(x)) = \log \sum_a e^{q(x)a}$$

$q : X \rightarrow \mathbb{R}^n$ neural network
Optimization objectives

Now assume given **complete** data

<table>
<thead>
<tr>
<th>$a_1$</th>
<th>$a_2$</th>
<th>...</th>
<th>$a_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>$r_{11}$</td>
<td>$r_{12}$</td>
<td>$r_{1...}$</td>
</tr>
<tr>
<td>$x_2$</td>
<td>$r_{21}$</td>
<td>$r_{22}$</td>
<td>$r_{2...}$</td>
</tr>
<tr>
<td>$x_3$</td>
<td>$r_{31}$</td>
<td>$r_{32}$</td>
<td>$r_{3...}$</td>
</tr>
<tr>
<td>$x_4$</td>
<td>$r_{41}$</td>
<td>$r_{42}$</td>
<td>$r_{4...}$</td>
</tr>
<tr>
<td>$x_5$</td>
<td>$r_{51}$</td>
<td>$r_{52}$</td>
<td>$r_{5...}$</td>
</tr>
<tr>
<td>$x_6$</td>
<td>$r_{61}$</td>
<td>$r_{62}$</td>
<td>$r_{6...}$</td>
</tr>
<tr>
<td>...</td>
<td>$r_{.1}$</td>
<td>$r_{.2}$</td>
<td>$r_{...}$</td>
</tr>
<tr>
<td>$x_m$</td>
<td>$r_{m1}$</td>
<td>$r_{m2}$</td>
<td>$r_{m...}$</td>
</tr>
</tbody>
</table>

Target objective
- expected reward: $\max \sum_i r_i \cdot \pi(x_i)$

Done, right?
Not so fast …

**This objective has serious problems**
- actually trying to solve: $\max \sum_i r_i \cdot f(q(x_i))$
- plateaus everywhere

**Theorem**
- can have **exponentially many** local maxima
- nearly impossible to reach a global optima

You already know not to train this way!

to maximize expected reward on **test** contexts

Optimize policy $\pi : X \to \Delta^n$

$\pi(a \mid x) = e^{q(x)_a - F(q(x))}$

$F(q(x)) = \log \sum_a e^{q(x)_a}$

$q : X \to \mathbb{R}^n$ neural network
Optimization objectives

Special case: supervised classification

Optimize policy \( \pi : X \rightarrow \Delta^n \) to maximize expected accuracy on test contexts.

\[
\begin{align*}
\pi(a \mid x) &= e^{q(x) a - F(q(x))} \\
F(q(x)) &= \log \sum_a e^{q(x) a}
\end{align*}
\]

\( q : X \rightarrow \mathbb{R}^n \) neural network
Optimization objectives

Special case: **supervised classification**

$$\pi(a | x) = e^{q(x)a - F(q(x))}$$

$$F(q(x)) = \log \sum_{a} e^{q(x)a}$$

$$q : X \rightarrow \mathcal{R}^n$$  neural network

**Target objective**

- expected **accuracy**: $\max \sum_i \mathbf{r}_i \cdot \pi(x_i)$

But you have never trained with this objective instead, you used a **surrogate objective**

**maximum likelihood**

$$\max \sum_i \mathbf{r}_i \cdot \log \pi(x_i)$$

**What's going on?**

- $\mathbf{r}_i \cdot \pi(x_i)$ is differentiable, that's not the issue
- training with $\mathbf{r}_i \cdot \log \pi(x_i)$ actually achieves better values of $\mathbf{r}_i \cdot \pi(x_i)$ on the training data

Optimize policy $\pi : X \rightarrow \Delta^n$

$$\pi(a | x) = e^{q(x)a - F(q(x))}$$

$$F(q(x)) = \log \sum_{a} e^{q(x)a}$$

$$q : X \rightarrow \mathcal{R}^n$$  neural network
Optimization objectives

Special case: **supervised classification**

<table>
<thead>
<tr>
<th></th>
<th>a1</th>
<th>a2</th>
<th>...</th>
<th>an</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>x3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>x5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x6</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xm</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Optimize policy \( \pi : X \to \Delta^n \)

\[ \pi(a | x) = e^{q(x)_a - F(q(x))} \]

\[ F(q(x)) = \log \sum_a e^{q(x)_a} \]

\( q : X \to \mathcal{R}^n \) neural network

Why?

- expected accuracy: \( \max \sum_i r_i \cdot \pi(x_i) \)
- maximum likelihood: \( \max \sum_i r_i \cdot \log \pi(x_i) \)

Useful properties of maximum likelihood

- \( r_i \cdot \log \pi(x_i) \) is concave in \( q(x_i) \)
- it is also calibrated w.r.t. \( r_i \cdot \pi(x_i) \):

\[ \forall \epsilon > 0 \exists \delta > 0 \ \ r \cdot \log \pi^* - r \cdot \log \pi < \delta \Rightarrow r \cdot \pi^* - r \cdot \pi < \epsilon \]
Optimization objectives

Misclassification error on MNIST training data
Optimization objectives

Back to general rewards

<table>
<thead>
<tr>
<th></th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>...</th>
<th>$a_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>$r_{11}$</td>
<td>$r_{12}$</td>
<td>$r_{1...}$</td>
<td>$r_{1n}$</td>
</tr>
<tr>
<td>$x_2$</td>
<td>$r_{21}$</td>
<td>$r_{22}$</td>
<td>$r_{2...}$</td>
<td>$r_{2n}$</td>
</tr>
<tr>
<td>$x_3$</td>
<td>$r_{31}$</td>
<td>$r_{32}$</td>
<td>$r_{3...}$</td>
<td>$r_{3n}$</td>
</tr>
<tr>
<td>$x_4$</td>
<td>$r_{41}$</td>
<td>$r_{42}$</td>
<td>$r_{4...}$</td>
<td>$r_{4n}$</td>
</tr>
<tr>
<td>$x_5$</td>
<td>$r_{51}$</td>
<td>$r_{52}$</td>
<td>$r_{5...}$</td>
<td>$r_{5n}$</td>
</tr>
<tr>
<td>$x_6$</td>
<td>$r_{61}$</td>
<td>$r_{62}$</td>
<td>$r_{6...}$</td>
<td>$r_{6n}$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$x_m$</td>
<td>$r_{m1}$</td>
<td>$r_{m2}$</td>
<td>$r_{m...}$</td>
<td>$r_{mn}$</td>
</tr>
</tbody>
</table>

Optimize policy $\pi : X \rightarrow \Delta^n$

$$\pi(a|x) = e^{q(x)_a - F(q(x))}$$

$$F(q(x)) = \log \sum_a e^{q(x)_a}$$

$q : X \rightarrow \mathbb{R}^n$ neural network
to maximize expected reward on test contexts
Optimization objectives

Back to general rewards

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>...</th>
<th>$a_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$r_{11}$</td>
<td>$r_{12}$</td>
<td>...</td>
<td>$r_{1n}$</td>
</tr>
<tr>
<td>$x_2$</td>
<td>$r_{21}$</td>
<td>$r_{22}$</td>
<td>...</td>
<td>$r_{2n}$</td>
</tr>
<tr>
<td>$x_3$</td>
<td>$r_{31}$</td>
<td>$r_{32}$</td>
<td>...</td>
<td>$r_{3n}$</td>
</tr>
<tr>
<td>$x_4$</td>
<td>$r_{41}$</td>
<td>$r_{42}$</td>
<td>...</td>
<td>$r_{4n}$</td>
</tr>
<tr>
<td>$x_5$</td>
<td>$r_{51}$</td>
<td>$r_{52}$</td>
<td>...</td>
<td>$r_{5n}$</td>
</tr>
<tr>
<td>$x_6$</td>
<td>$r_{61}$</td>
<td>$r_{62}$</td>
<td>...</td>
<td>$r_{6n}$</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>$x_m$</td>
<td>$r_{m1}$</td>
<td>$r_{m2}$</td>
<td>...</td>
<td>$r_{mn}$</td>
</tr>
</tbody>
</table>

Optimize policy $\pi : X \rightarrow \Delta^n$

$\pi(a | x) = e^{q(x)_a - F(q(x))}$

$F(q(x)) = \log \sum_a e^{q(x)_a}$

$q : X \rightarrow \mathbb{R}^n$ neural network

Target objective

- expected reward: $\max \sum_i r_i \cdot \pi(x_i)$

"cost sensitive classification"

Calibrated surrogates exist for $r_i \cdot \pi(x_i)$ (Pires et al. ICML-2013)

Interesting alternative

- entropy regularized expected reward

$\max \sum_i r_i \cdot \pi(x_i) - \tau \pi(x_i) \cdot \log \pi(x_i)$

to maximize expected reward on test contexts
Optimization objectives

Back to general rewards

<table>
<thead>
<tr>
<th>X1</th>
<th>a1</th>
<th>a2</th>
<th>...</th>
<th>an</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>r11</td>
<td>r12</td>
<td>r1...</td>
<td>r1n</td>
</tr>
<tr>
<td>X2</td>
<td>r21</td>
<td>r22</td>
<td>r2...</td>
<td>r2n</td>
</tr>
<tr>
<td>X3</td>
<td>r31</td>
<td>r32</td>
<td>r3...</td>
<td>r3n</td>
</tr>
<tr>
<td>X4</td>
<td>r41</td>
<td>r42</td>
<td>r4...</td>
<td>r4n</td>
</tr>
<tr>
<td>X5</td>
<td>r51</td>
<td>r52</td>
<td>r5...</td>
<td>r5n</td>
</tr>
<tr>
<td>X6</td>
<td>r61</td>
<td>r62</td>
<td>r6...</td>
<td>r6n</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Xm</td>
<td>rm1</td>
<td>rm2</td>
<td>rm...</td>
<td>rmn</td>
</tr>
</tbody>
</table>

Entropy regularized expected reward

\[
\text{arg max } r \cdot \pi - \tau \pi \cdot \log \pi \\
= \text{arg min } \tau F(r/\tau) - r \cdot \pi + \tau F^*(\pi) \\
= \text{arg min } F(r/\tau) - r \cdot \pi/\tau + F^*(\pi) \\
= \text{arg min } \text{KL}(\pi \| p) \text{ where } p = e^{r/\tau - F(r/\tau)}
\]

Suggests a natural surrogate

arg min \text{KL}(p \| \pi) = \text{arg min } F(q) - q \cdot p

• convex in q

Optimize policy \( \pi : X \rightarrow \Delta^n \)

\[
\pi(a | x) = e^{q(x)_a - F(q(x))}
\]

\[
F(q(x)) = \log \sum_a e^{q(x)_a}
\]

Let \( F^*(\pi) = \pi \cdot \log \pi \)

q : X → \( \mathcal{R}^n \) neural network
Optimization objectives

Back to general rewards

<table>
<thead>
<tr>
<th></th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>\ldots</th>
<th>$a_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>$r_{11}$</td>
<td>$r_{12}$</td>
<td>$r_1\ldots$</td>
<td>$r_{1n}$</td>
</tr>
<tr>
<td>$x_2$</td>
<td>$r_{21}$</td>
<td>$r_{22}$</td>
<td>$r_2\ldots$</td>
<td>$r_{2n}$</td>
</tr>
<tr>
<td>$x_3$</td>
<td>$r_{31}$</td>
<td>$r_{32}$</td>
<td>$r_3\ldots$</td>
<td>$r_{3n}$</td>
</tr>
<tr>
<td>$x_4$</td>
<td>$r_{41}$</td>
<td>$r_{42}$</td>
<td>$r_4\ldots$</td>
<td>$r_{4n}$</td>
</tr>
<tr>
<td>$x_5$</td>
<td>$r_{51}$</td>
<td>$r_{52}$</td>
<td>$r_5\ldots$</td>
<td>$r_{5n}$</td>
</tr>
<tr>
<td>$x_6$</td>
<td>$r_{61}$</td>
<td>$r_{62}$</td>
<td>$r_6\ldots$</td>
<td>$r_{6n}$</td>
</tr>
<tr>
<td>\vdots</td>
<td>$r_{1\cdot}$</td>
<td>$r_{2\cdot}$</td>
<td>$r_{\cdot\ldots}$</td>
<td>$r_{n\cdot}$</td>
</tr>
<tr>
<td>$x_m$</td>
<td>$r_{m1}$</td>
<td>$r_{m2}$</td>
<td>$r_{m\cdot}$</td>
<td>$r_{mn}$</td>
</tr>
</tbody>
</table>

Optimize policy  

$\pi : X \to \Delta^n$

$\pi(a \mid x) = e^{q(x)a - F(q(x))}$

$F(q(x)) = \log \sum a e^{q(x)a}$

$q : X \to \mathbb{R}^n$ neural network

Comparison to maximum likelihood

before  

$-r \cdot \log \pi = F(q) - q \cdot r$

now  

$\text{KL}(p \mid \pi) \equiv F(q) - q \cdot p$

If $r = 1_a$ is an indicator

- become equivalent as $\tau \to 0$

  \[ \lim_{\tau \to 0} p = r = 1_a \]

- but $\tau > 0$ gives soft targets for KL

  "label smoothing"

  improves generalization in practice
Optimization objectives

Back to **general** rewards

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>...</th>
<th>$a_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$r_{11}$</td>
<td>$r_{12}$</td>
<td>$r_1$...</td>
<td>$r_{1n}$</td>
</tr>
<tr>
<td>$x_2$</td>
<td>$r_{21}$</td>
<td>$r_{22}$</td>
<td>$r_2$...</td>
<td>$r_{2n}$</td>
</tr>
<tr>
<td>$x_3$</td>
<td>$r_{31}$</td>
<td>$r_{32}$</td>
<td>$r_3$...</td>
<td>$r_{3n}$</td>
</tr>
<tr>
<td>$x_4$</td>
<td>$r_{41}$</td>
<td>$r_{42}$</td>
<td>$r_4$...</td>
<td>$r_{4n}$</td>
</tr>
<tr>
<td>$x_5$</td>
<td>$r_{51}$</td>
<td>$r_{52}$</td>
<td>$r_5$...</td>
<td>$r_{5n}$</td>
</tr>
<tr>
<td>$x_6$</td>
<td>$r_{61}$</td>
<td>$r_{62}$</td>
<td>$r_6$...</td>
<td>$r_{6n}$</td>
</tr>
<tr>
<td>...</td>
<td>$r_{\cdot 1}$</td>
<td>$r_{\cdot 2}$</td>
<td>$r_{\cdot \text{...}}$</td>
<td>$r_{\cdot n}$</td>
</tr>
<tr>
<td>$x_m$</td>
<td>$r_{m1}$</td>
<td>$r_{m2}$</td>
<td>$r_m$...</td>
<td>$r_{mn}$</td>
</tr>
</tbody>
</table>

**A convex, calibrated upper bound**

$$\text{KL}(\pi \| p) \leq \text{KL}(p \| \pi) + \frac{\tau}{4} \| r/\tau - q \|^2$$

Optimize policy

$$\pi : X \to \Delta^n$$

$$\pi(a \mid x) = e^{q(x)_a - F(q(x))}$$

$$F(q(x)) = \log \sum_a e^{q(x)_a}$$

$$q : X \to \mathbb{R}^n \quad \text{neural network}$$
Optimization objectives

MNIST

CIFAR10
Batch policy optimization

Three key issues

1. generalization
2. optimization
3. missing data

training objective ≠ target objective
Batch policy optimization

<table>
<thead>
<tr>
<th></th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$\ldots$</th>
<th>$a_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_5$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_6$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\vdots$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_m$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$r_1$</td>
<td></td>
<td></td>
<td>$r_2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$r_3$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$r_4$</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$r_5$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Three key issues:
1. generalization
2. optimization
3. missing data
Supervised vs reinforcement learning

supervised classification

\[
\begin{array}{cccc}
  x_1 & a_1 & r_{11} & r_{12} & r_{1\ldots} & r_{1n} \\
  x_2 & a_2 & r_{21} & r_{22} & r_{2\ldots} & r_{2n} \\
  x_3 & a_3 & r_{31} & r_{32} & r_{3\ldots} & r_{3n} \\
  x_4 & a_4 & r_{41} & r_{42} & r_{4\ldots} & r_{4n} \\
  x_5 & a_5 & r_{51} & r_{52} & r_{5\ldots} & r_{5n} \\
  x_6 & a_6 & r_{61} & r_{62} & r_{6\ldots} & r_{6n} \\
  \vdots & \vdots & r_{1:1} & r_{1:2} & r_{1:\ldots} & r_{1:n} \\
  x_m & a_m & r_{m1} & r_{m2} & r_{m\ldots} & r_{mn}
\end{array}
\]

batch policy optimization

\[
\begin{array}{cccc}
  x_1 & a_1 & r_1 \\
  x_2 & a_2 & r_2 \\
  x_3 & a_3 & r_3 \\
  x_4 & a_4 & r_4 \\
  x_5 & a_5 & r_5 \\
  x_6 & a_6 & r_6 \\
  \vdots & \vdots & r_1:1 \\
  x_m & a_m & r_{m1} \\
\end{array}
\]

Optimize policy \( \pi: X \to \Delta^n \) to maximize expected reward on test contexts

key difference is missing data
Missing data inference

How to handle missing data?

Optimize policy $\pi : X \rightarrow \Delta^n$
### Missing data inference

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$X_4$</th>
<th>$X_5$</th>
<th>$X_6$</th>
<th>...</th>
<th>$X_m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>$q_{11}$</td>
<td>$r_1$</td>
<td>$q_{12}$</td>
<td>$q_{13}$</td>
<td>$q_{14}$</td>
<td>...</td>
<td>$q_{1n}$</td>
</tr>
<tr>
<td>$a_2$</td>
<td>$q_{21}$</td>
<td>$q_{22}$</td>
<td>$q_{23}$</td>
<td>$r_2$</td>
<td>$q_{24}$</td>
<td>...</td>
<td>$q_{2n}$</td>
</tr>
<tr>
<td></td>
<td>$r_3$</td>
<td>$q_{32}$</td>
<td>$q_{33}$</td>
<td>$q_{34}$</td>
<td>$q_{35}$</td>
<td>...</td>
<td>$q_{3n}$</td>
</tr>
<tr>
<td></td>
<td>$q_{41}$</td>
<td>$q_{42}$</td>
<td>$r_4$</td>
<td>$q_{44}$</td>
<td>$q_{45}$</td>
<td>...</td>
<td>$q_{4n}$</td>
</tr>
<tr>
<td></td>
<td>$r_5$</td>
<td>$q_{52}$</td>
<td>$q_{53}$</td>
<td>$q_{54}$</td>
<td>$q_{55}$</td>
<td>...</td>
<td>$q_{5n}$</td>
</tr>
<tr>
<td></td>
<td>$q_{61}$</td>
<td>$r_6$</td>
<td>$q_{63}$</td>
<td>$q_{64}$</td>
<td>$q_{65}$</td>
<td>...</td>
<td>$q_{6n}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$q_{:1}$</td>
<td>$q_{:2}$</td>
<td>$r_:$</td>
<td>$q_{:n}$</td>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$q_{m1}$</td>
<td>$q_{m2}$</td>
<td>$q_{m3}$</td>
<td>$r_m$</td>
<td></td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

**Simple idea**

**imputation**
- fill in guesses for missing values
- reduce to fully observed case

**Might sound naive**
- but this is actually a dominant approach

Optimize policy $\pi : X \rightarrow \Delta^n$
Missing data inference

Optimize policy \( \pi : X \to \Delta^n \)

Example

Importance corrected expected reward

\[
\max \sum_i \pi(a_i \mid x_i) \frac{r_i}{\beta_i}
\]

where \( \beta \) are proposal probabilities from behavior strategy.

We already know this is a poor objective but what about missing data inference?

Equivalent to \( \max \hat{r} \cdot \pi \) using

\[
\hat{r}_i = 1 \frac{r_i}{a_i \beta_i}
\]

That is

- exaggerate observed values by \( \frac{1}{\beta_i} \)
- fill in all unobserved values with 0
Missing data inference

This is a pretty lame inference principle

- altering the data we do see
- to compensate for a bad guess about the data we don't see

But … its unbiased!

\[ E[\hat{r} | x] = \sum_a \beta_a 1_a r_a = \sum_a 1_a r_a = r \]

Optimize policy \( \pi : X \to \Delta^n \)
Improvement

• instead of filling in with 0s
• fill in with guesses from a model \( q(x) \)

\[
\hat{r} = \tau q + \lambda \mathbf{1}_a(r - \tau q_a)
\]

Also unbiased

• as long as \( \lambda = 1/\beta_i \)
but still alters observed data

Missing data inference

Optimize policy \( \pi : X \rightarrow \Delta^n \)
### Missing data inference

Optimize policy $\pi : X \rightarrow \Delta^n$

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$a_1$</th>
<th>$\tau q_{11}$</th>
<th>$\lambda(r_{12} q_{12})$</th>
<th>$\tau q_{11}$</th>
<th>$\tau q_{1n}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_2$</td>
<td>$a_2$</td>
<td>$\tau q_{21}$</td>
<td>$\tau q_{22}$</td>
<td>$\tau q_{22}$</td>
<td>$\lambda(r_{23} q_{2n})$</td>
</tr>
<tr>
<td>$x_3$</td>
<td>$\lambda(r_{31} q_{31})$</td>
<td>$\tau q_{32}$</td>
<td>$\tau q_{32}$</td>
<td>$\tau q_{32}$</td>
<td>$\tau q_{3n}$</td>
</tr>
<tr>
<td>$x_4$</td>
<td>$\tau q_{41}$</td>
<td>$\tau q_{42}$</td>
<td>$\lambda(r_{44} q_{44})$</td>
<td>$\tau q_{44}$</td>
<td></td>
</tr>
<tr>
<td>$x_5$</td>
<td>$\lambda(r_{51} q_{51})$</td>
<td>$\tau q_{52}$</td>
<td>$\tau q_{52}$</td>
<td>$\tau q_{52}$</td>
<td>$\tau q_{5n}$</td>
</tr>
<tr>
<td>$x_6$</td>
<td>$\tau q_{61}$</td>
<td>$\lambda(r_{62} q_{62})$</td>
<td>$\tau q_{62}$</td>
<td>$\tau q_{62}$</td>
<td>$\tau q_{6n}$</td>
</tr>
<tr>
<td>...</td>
<td>$\tau q_{i1}$</td>
<td>$\tau q_{i2}$</td>
<td>$\lambda(r_{in} q_{in})$</td>
<td>$\tau q_{in}$</td>
<td></td>
</tr>
<tr>
<td>$x_m$</td>
<td>$\tau q_{m1}$</td>
<td>$\tau q_{m2}$</td>
<td>$\tau q_{m2}$</td>
<td>$\tau q_{m2}$</td>
<td>$\lambda(r_{mn} q_{mn})$</td>
</tr>
</tbody>
</table>

### Improvement

“doubly robust estimation”

- instead of filling in with 0s
- fill in with guesses from a model $q(x)$

$$\hat{r} = \tau q + \lambda 1_a (r - \tau q_a)$$

Also unbiased

- as long as $\lambda = 1/\beta_i$
- but still alters observed data

### Where should the model come from?

- could use a separate critic
- train via least squares, then optimize $\pi$
- works okay, but not great

### Note

- there is only one action value function for single-step decision making, $r(x, a)$
- actor-critic approaches trivialized
Missing data inference

Dr. X

Unified approach
- actor and critic are same model
  \[ \pi = e^{q - F(q)} \] where \( F(q) = \log 1 \cdot e^{q} \)
- use logits \( \tau q(x) \) to predict rewards
  \[ q(x, a) \approx \frac{r(x, a)}{\tau} \]

Can combine with previous objectives
- \( KL(\pi || \hat{p}) \) where \( \hat{p} = e^{\hat{r}/\tau - F(\hat{r}/\tau)} \)
- \( KL(\hat{p} || \pi) \)
- \( KL(\pi || \hat{p}) \leq KL(\hat{p} || \pi) + \frac{\tau}{4} \| \hat{r}/\tau - q \|^2 \)

these are somewhat sensitive to ranking, unlike least squares

Optimize policy \( \pi : X \rightarrow \Delta^n \)
Missing data inference

MNIST

CIFAR10
The unified combination is **sound**
(i.e. single model, doubly robust est., calibrated surrogate)

**surrogate loss**
\[ L(q, r, x) = \tau D_F(q(x)\| \frac{r}{\tau}) + \frac{\tau}{4} \| q(x) - \frac{r}{\tau} \|^2 \]

**smoothed risk**
\[ \mathcal{S}_\tau(\pi, r, x) = -r \cdot \pi(x) + \tau \pi(x) \cdot \log \pi(x) \]

**suboptimality gap**
\[ \mathcal{G}_\tau(\pi) = \mathcal{S}_\tau(\pi) - \mathcal{S}_\tau^* \]

**Theorem** (informally): If \( \mathcal{H}, \beta, p(x, r), \hat{r} \) are “well behaved”, then:
\[ \forall \tau, \delta > 0 \exists C \text{ s.t. w.p. } \geq 1 - \delta: \text{ if } \hat{L}(q, \mathcal{D}) < \frac{\tau C}{\sqrt{T}} \text{ for } q \in \mathcal{H} \text{ then } \mathcal{G}_\tau(f \circ q) \leq \frac{2\tau C}{\sqrt{T}} \]

small empirical surrogate implies small true suboptimality gap
Missing data inference

Even more principled approach
• back to first principles
• how do we reason about missing data in the rest of ML and statistics?

Bayesian inference
• postulate a generative model of reward $q \rightarrow \xi \rightarrow r$
• e.g. Gaussian
  • prior $\xi \sim \mathcal{N}(q, Q)$
  • likelihood $r|a, \xi \sim \mathcal{N}(\phi(a) \cdot \xi, \sigma^2)$
  • posterior $\xi | r_0, a_0 \sim \mathcal{N}(\mu, C)$
    $$\mu = C(\phi(a_0)r_0\sigma^{-2} + Q^{-1}q)$$
    $$C = (Q^{-1} + \sigma^{-2}\phi(a_0)\phi(a_0)^\top)^{-1}$$
• predictive $r | a, r_0, a_0 \sim \mathcal{N}(\hat{\mu}, \hat{\sigma}^2)$
    $$\hat{\mu} = \phi(a) \cdot \mu$$
    $$\hat{\sigma}^2 = \sigma^2 + \phi(a_0)^\top C\phi(a_0)$$

Optimize policy $\pi : X \rightarrow \text{exp-family}(\mathcal{R})$
$$\pi(a | x) = e^{q(x)a - F(q(x))}$$
$$F(q(x)) = \log \int e^{q(x)a} \mu(da)$$
$q : X \rightarrow \mathcal{R}^k$ neural network
Missing data inference

Empirical Bayes estimation
- optimize hyperparameters $\mathbf{q}$ (neural network)
- integrate out parameters $\xi$

Example
marginal likelihood
$$- \log p(r_0 \mid a_0, \mathbf{q}) = - \log \int p(r_0 \mid a_0, \xi)p(\xi \mid \mathbf{q}) \, d\xi$$
$$= \frac{1}{2\sigma^2}(\phi(a_0) \cdot q - r_0)^2 + \frac{1}{2} \log \sigma^2 + c$$
- essentially least squares regression

Can alternatively use surrogates
$$\min_{\mathbf{KL}}(\text{prior} \| \text{posterior})$$
$$\min_{\mathbf{KL}}(\text{posterior} \| \text{prior}) \approx \min I(\xi; r_0)$$

Optimize policy
$$\pi : X \rightarrow \exp\text{-family}(\mathcal{R})$$
$$\pi(a \mid x) = e^{q(x)a - F(q(x))}$$
$$F(q(x)) = \log \int e^{q(x)a} \mu(da)$$
$$q : X \rightarrow \mathcal{R}^k$$ neural network
Missing data inference

Sum of squared test error on continuous action MNIST ($a \in \mathbb{R}^{10}$)
Batch policy optimization

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( \ldots )</th>
<th>( x_m )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r_1 )</td>
<td>( r_2 )</td>
<td>( r_3 )</td>
<td>( r_4 )</td>
</tr>
<tr>
<td>( r_5 )</td>
<td>( r_6 )</td>
<td>( r_7 )</td>
<td>( \ldots )</td>
</tr>
<tr>
<td>( r_m )</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Three key issues**

1. **generalization**

2. **optimization**

3. **missing data**

<table>
<thead>
<tr>
<th>Training objective</th>
<th>Target objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \neq )</td>
<td>classical methods still help</td>
</tr>
</tbody>
</table>
The RL problem

1. multi-agent interaction ➡ non-stationarity
2. partial observability ➡ must construct memory
3. exploration ➡ explore/exploit trade-off
4. sequential decisions ➡ temporal credit assignment
5. exploitation ➡ policy optimization

“Batch” RL
Optimizing sequential decision making

Batch RL
Sequential decision making

Major differences from one step decision making

- Target values are not immediate rewards
  - Temporal credit assignment problem
- Target values must be inferred
Sequential decision making

Target value inference: Bellman optimality principle

Bellman optimality
\[ \forall s, a \quad q_{sa} = r_{sa} + \gamma \sum_{s'} p_{sas'} \max_{a'} q_{s'a'} \]

Consider approximation
\[ \forall s, a \quad \hat{q}_{sa} \approx r_{sa} + \gamma \sum_{s'} p_{sas'} \max_{a'} \hat{q}_{s'a'} \]

Violation penalty
\[ \sum_{s,a} \frac{d_{sa}}{2} \left( \hat{q}_{sa} - r_{sa} - \gamma \sum_{s'} p_{sas'} \max_{a'} \hat{q}_{s'a'} \right)^2 \]

Gradient
\[ \sum_{s,a} d_{sa} \left( \hat{q}_{sa} - r_{sa} - \gamma \sum_{s'} p_{sas'} \max_{a'} \hat{q}_{s'a'} \right) \left( \frac{d\hat{q}_{sa}}{d\theta} - \gamma \sum_{s'} p_{sas'} \frac{d\hat{q}_{sa}(s)}{d\theta} \right) \]

Textbook “Update”
\[ \sum_{s,a} d_{sa} \left( \hat{q}_{sa} - r_{sa} - \gamma \sum_{s'} p_{sas'} \max_{a'} \hat{q}_{s'a'} \right) \frac{d\hat{q}_{sa}}{d\theta} \]
Sequential RL

The Deadly Triad

Generalization

Off-policy

Bootstrapping

$q_{sa} = r_{sa} + \gamma \sum_{s'} p(s' | s, a) \max_{a'} q_{s' a'}$
Sequential RL

The Deadly Triad

DQN

Generalization

Off-policy

Bootstrapping

$q_{sa} = r_{sa} + \gamma \sum_{s'} p(s' | s, a) \max_{a'} q_{s'a'}$
Sequential RL

Back to Basics

Generalization

Off-policy  Bootstrapping

\[ q_{sa} = r_{sa} + \gamma \sum_{s'} p(s' | s, a) \max_{a'} q_{s'a'} \]
Sequential RL

Back to Basics

Off-policy

Bootstrapping

Don’t be crazy

\[ q_{sa} = r_{sa} + \gamma \sum_{s'} p(s' \mid s, a) \max_{a'} q_{s'a'} \]
Sequential RL

Back to Basics

Generalization

On policy methods
- Policy gradient
- Actor-critic

Off-policy

Bootstrapping

Data inefficient

$q_{sa} = r_{sa} + \gamma \sum_{s'} p(s' | s, a) \sum_{a'} \pi_{s' a'} q_{s'a'}$

$q_{sa} = r_{sa} + \gamma \sum_{s'} p(s' | s, a) \sum_{a'} \pi_{s' a'} q_{s'a'}$
Sequential RL

Back to Basics

Generalization

Off-policy

Bootstrapping

$q_{sa} = r_{sa} + \gamma \sum_{s'} p(s' | s, a) \max_{a'} q_{s'a'}$
Sequential RL

**Avoiding the bootstrap**

1. Multiple hypothesis tracking *(NeurIPS-2018)*

2. Monte Carlo policy iteration

3. Lagrange dual: joint state-action distributions
Sequential RL

Avoiding the bootstrap

1. Multiple hypothesis tracking (NeurIPS-2018)
2. Monte Carlo policy iteration
3. Lagrange dual: joint state-action distributions
Sequential RL

Cart-Pole

using coordinate features

given random walk data
Sequential RL

Avoiding the bootstrap

1. Multiple hypothesis tracking (NeurIPS-2018)

2. Monte Carlo policy iteration

3. Lagrange dual: joint state-action distributions

$$\max_{\mathbf{d}} \mathbf{d}^T \mathbf{r} \text{ subject to } \mathbf{d} \geq 0, (I \otimes \mathbf{1}^T)\mathbf{d} = (1 - \gamma)\mu + \gamma \mathbf{P}^T \mathbf{d}$$

$$\forall s' \sum_{a'} d_{s'a'} = \sum_{s,a} \tilde{P}(s'|s,a) d_{sa}$$
Conclusion

• Classical (within domain) generalization might not have been fully exploited in RL
  • generalization destabilizes bootstrapping
  • but should prioritize generalization over bootstrap

• It is possible to infer improved policies from log data, without policy-directed exploration

• Surrogate training objectives and missing data inference improve solution quality

• Batch RL amenable to classical generalization theory