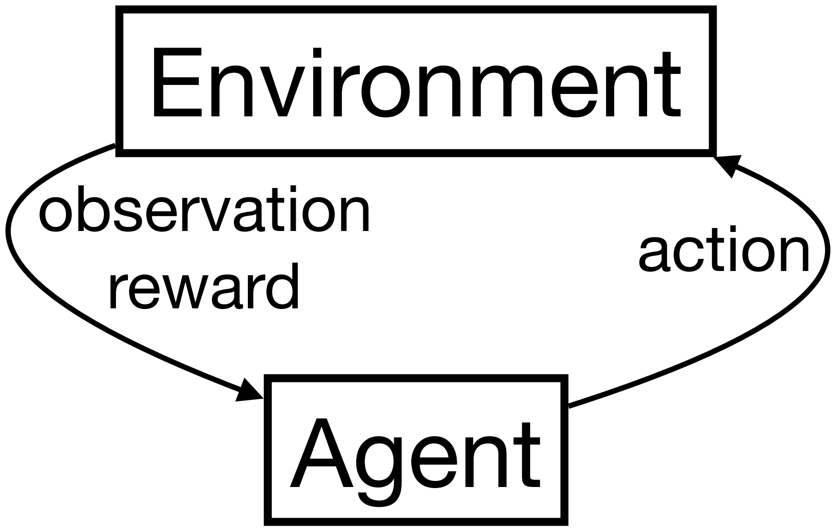
Off-policy Policy Optimization

Dale Schuurmans

Google Brain

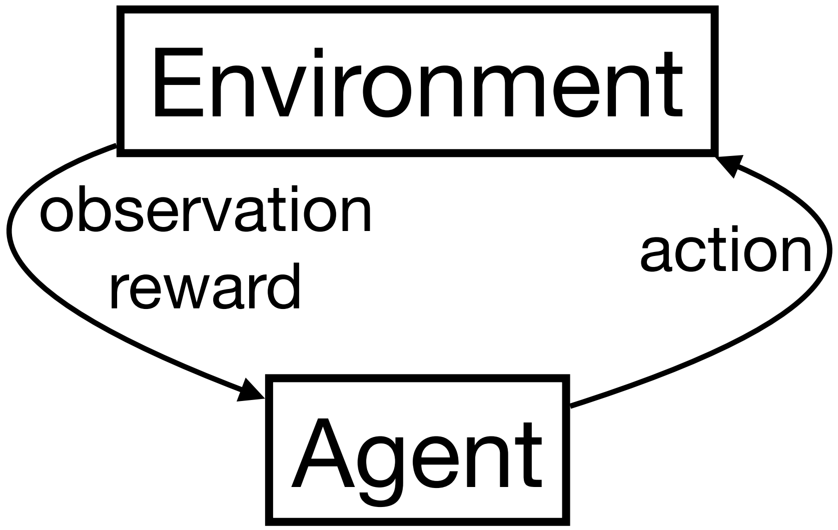
University of Alberta

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

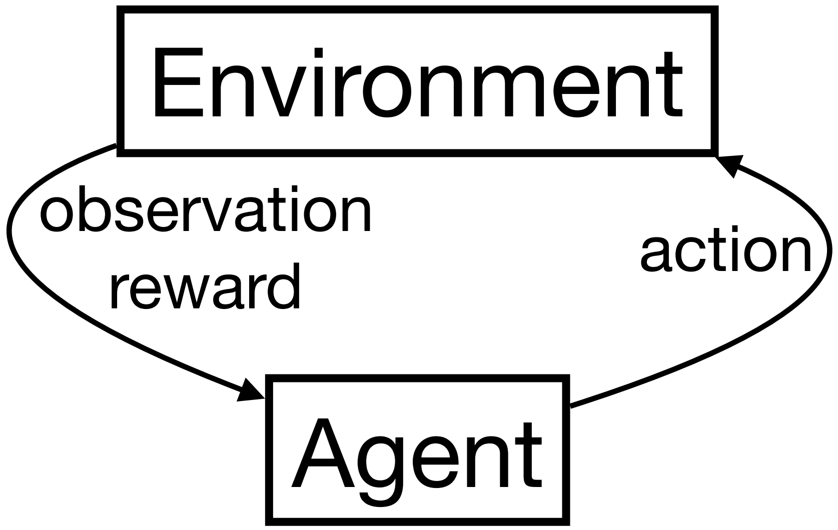
The RL problem



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

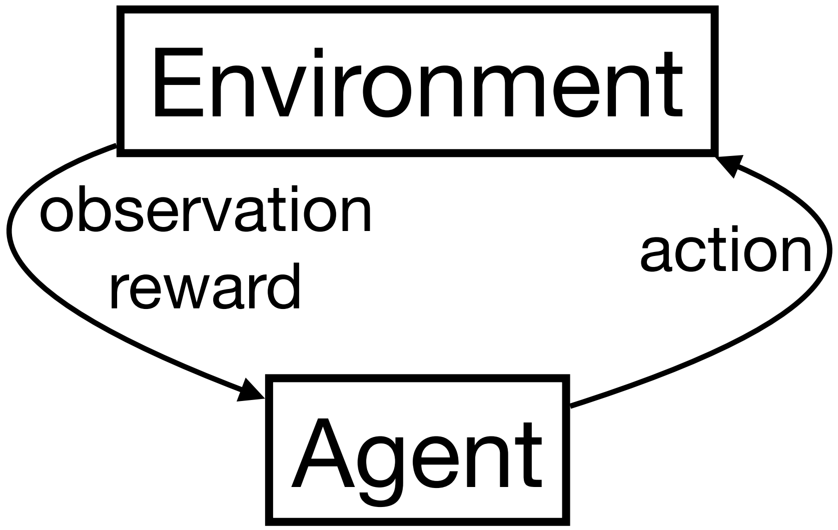
The RL problem



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

	a_1	a_2	$\dots$	a_n
x_1		r_1		
x_2				r_2
x_3	r_3			
x_4			r_4	
x_5	r_5			
x_6		r_6		
$\vdots$			$r_{\vdots}$	
x_m				r_m

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

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

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

$q : X \to \Re^n$ neural network

Batch policy optimization

Given data

	a_1	a_2	$\dots$	a_n
x_1		r_1		
x_2				r_2
x_3	r_3			
x_4			r_4	
x_5	r_5			
x_6		r_6		
$\vdots$			$r_{:}$	
x_m				r_m

Three key issues

1. generalization
2. optimization
3. missing data

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

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

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

$q : X \rightarrow \Re^n$ neural network

Batch policy optimization

Given data

	a_1	a_2	$\dots$	a_n
x_1		r_1, β_1		
x_2				r_2, β_2
x_3	r_3, β_3			
x_4			r_4, β_4	
x_5	r_5, β_5			
x_6		r_6, β_6		
$\vdots$			$r_:, \beta_:$	
x_m				r_m, β_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)

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

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

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

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

$q : X \to \Re^n$ neural network

Batch policy optimization

Given data

	a_1	a_2	$\dots$	a_n
x_1		r_1		
x_2				r_2
x_3	r_3			
x_4			r_4	
x_5	r_5			
x_6		r_6		
$\vdots$			$r_{\vdots}$	
x_m				r_m

Three key issues	Importance corrected expected reward
1. generalization	okay
2. optimization	👎 ✘
3. missing data	👎 ✘

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

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

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

$q : X \to \Re^n$ neural network

Optimization objectives

Given data

	a_1	a_2	$\dots$	a_n
x_1		r_1		
x_2				r_2
x_3	r_3			
x_4			r_4	
x_5	r_5			
x_6		r_6		
$\vdots$			$r_{\vdots}$	
x_m				r_m

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

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

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

$q : X \to \Re^n$ neural network

Optimization objectives

Now assume given **complete** data

	a_1	a_2	$\dots$	a_n
x_1	r_{11}	r_{12}	$r_{1\dots}$	r_{1n}
x_2	r_{21}	r_{22}	$r_{2\dots}$	r_{2n}
x_3	r_{31}	r_{32}	$r_{3\dots}$	r_{3n}
x_4	r_{41}	r_{42}	$r_{4\dots}$	r_{4n}
x_5	r_{51}	r_{52}	$r_{5\dots}$	r_{5n}
x_6	r_{61}	r_{62}	$r_{6\dots}$	r_{6n}
$\vdots$	$r_{:1}$	$r_{:2}$	$r_{:\dots}$	$r_{:n}$
x_m	r_{m1}	r_{m2}	$r_{m\dots}$	r_{mn}

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

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

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

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

Optimization objectives

Now assume given **complete** data

	a_1	a_2	$\dots$	a_n
x_1	r_{11}	r_{12}	$r_{1\dots}$	r_{1n}
x_2	r_{21}	r_{22}	$r_{2\dots}$	r_{2n}
x_3	r_{31}	r_{32}	$r_{3\dots}$	r_{3n}
x_4	r_{41}	r_{42}	$r_{4\dots}$	r_{4n}
x_5	r_{51}	r_{52}	$r_{5\dots}$	r_{5n}
x_6	r_{61}	r_{62}	$r_{6\dots}$	r_{6n}
$\vdots$	$r_{:1}$	$r_{:2}$	$r_{:\dots}$	$r_{:n}$
x_m	r_{m1}	r_{m2}	$r_{m\dots}$	r_{mn}

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

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

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

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

Target objective

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

Done, right?
Not so fast …

This objective has serious problems

- actually trying to solve: $\max \sum_i \mathbf{r}_i \cdot \mathbf{f}(q(x_i))$
- plateaus everywhere

Theorem

can have **exponentially many** local maxima

- nearly impossible to reach a global optima

You already know not to train this way!

Optimization objectives

Special case: **supervised classification**

	a_1	a_2	…	a_n
x_1	0	1	0	0
x_2	0	0	0	1
x_3	1	0	0	0
x_4	0	0	1	0
x_5	1	0	0	0
x_6	0	1	0	0
⋮	0	0	1	0
x_m	0	0	0	1

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

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

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

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

Optimization objectives

Special case: **supervised classification**

	a_1	a_2	$\dots$	a_n
x_1	0	1	0	0
x_2	0	0	0	1
x_3	1	0	0	0
x_4	0	0	1	0
x_5	1	0	0	0
x_6	0	1	0	0
$\vdots$	0	0	1	0
x_m	0	0	0	1

Target objective

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

But you have never trained with this objective

Instead, you used a **surrogate objective**

maximum likelihood

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

What's going on?

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

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

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

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

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

Optimization objectives

Special case: **supervised classification**

	a_1	a_2	…	a_n
x_1	0	1	0	0
x_2	0	0	0	1
x_3	1	0	0	0
x_4	0	0	1	0
x_5	1	0	0	0
x_6	0	1	0	0
⋮	0	0	1	0
x_m	0	0	0	1

Why?

- expected accuracy: $\max \sum_i \mathbf{r}_i \cdot \boldsymbol{\pi}(x_i)$
- maximum likelihood: $\max \sum_i \mathbf{r}_i \cdot \log \boldsymbol{\pi}(x_i)$

Useful properties of maximum likelihood

- $\mathbf{r}_i \cdot \log \boldsymbol{\pi}(x_i)$ is concave in $\mathbf{q}(x_i)$
- it is also calibrated w.r.t. $\mathbf{r}_i \cdot \boldsymbol{\pi}(x_i)$:

$$\forall \epsilon > 0 \exists \delta > 0 \;\; \mathbf{r} \cdot \log \boldsymbol{\pi}^* - \mathbf{r} \cdot \log \boldsymbol{\pi} < \delta \Rightarrow \mathbf{r} \cdot \boldsymbol{\pi}^* - \mathbf{r} \cdot \boldsymbol{\pi} < \epsilon$$

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

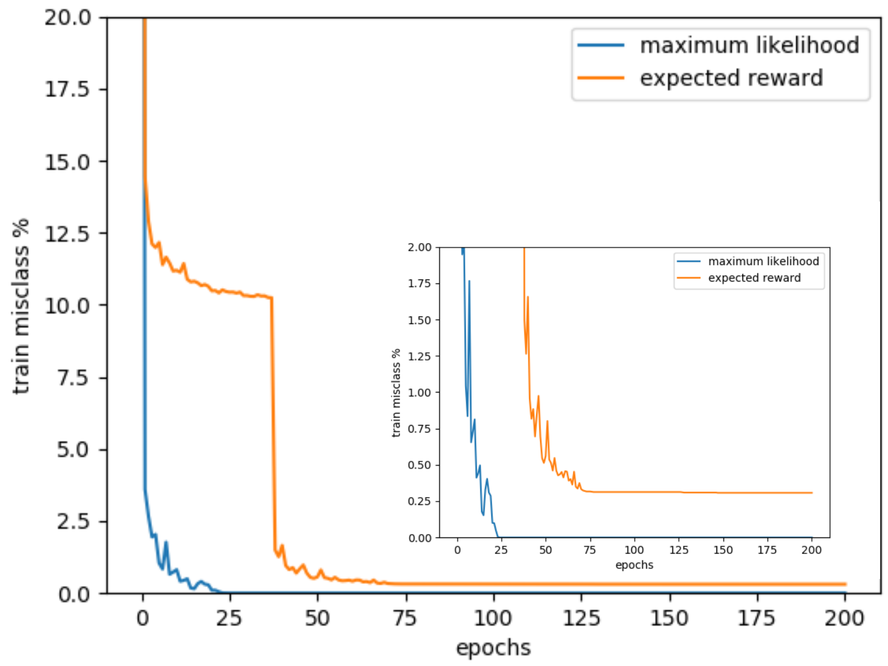
$$\pi(a \mid x) = e^{q(x)_a - F(q(x))}$$

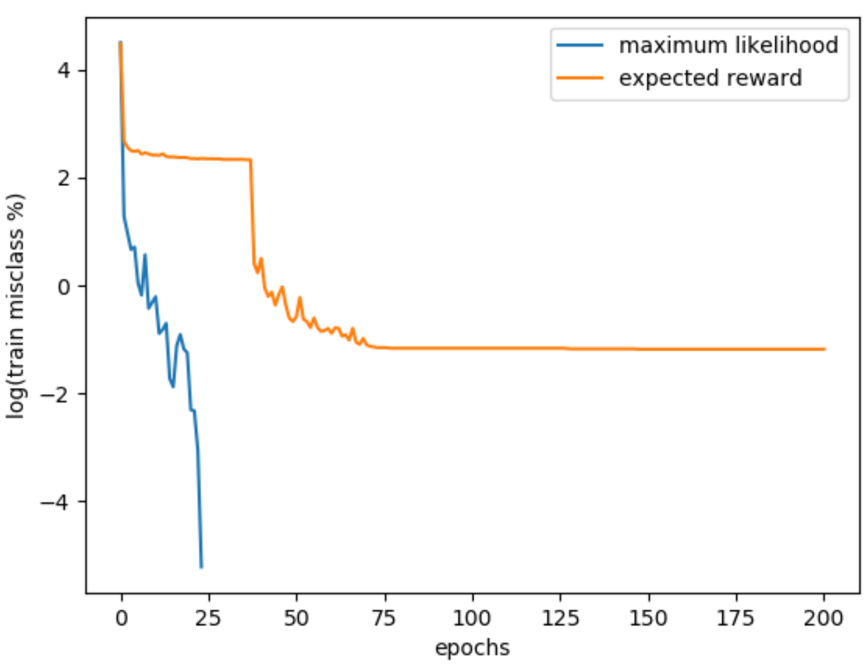
$$F(q(x)) = \log \sum_a e^{q(x)_a}$$

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

Optimization objectives

Misclassification error on MNIST training data





Optimization objectives

Back to **general** rewards

	a_1	a_2	$\dots$	a_n
x_1	r_{11}	r_{12}	$r_{1\dots}$	r_{1n}
x_2	r_{21}	r_{22}	$r_{2\dots}$	r_{2n}
x_3	r_{31}	r_{32}	$r_{3\dots}$	r_{3n}
x_4	r_{41}	r_{42}	$r_{4\dots}$	r_{4n}
x_5	r_{51}	r_{52}	$r_{5\dots}$	r_{5n}
x_6	r_{61}	r_{62}	$r_{6\dots}$	r_{6n}
$\vdots$	$r_{:1}$	$r_{:2}$	$r_{:\dots}$	$r_{:n}$
x_m	r_{m1}	r_{m2}	$r_{m\dots}$	r_{mn}

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

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

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

$q : X \to \Re^n$ neural network

Optimization objectives

Back to **general** rewards

	a_1	a_2	...	a_n
x_1	r_{11}	r_{12}	$r_{1\ldots}$	r_{1n}
x_2	r_{21}	r_{22}	$r_{2\ldots}$	r_{2n}
x_3	r_{31}	r_{32}	$r_{3\ldots}$	r_{3n}
x_4	r_{41}	r_{42}	$r_{4\ldots}$	r_{4n}
x_5	r_{51}	r_{52}	$r_{5\ldots}$	r_{5n}
x_6	r_{61}	r_{62}	$r_{6\ldots}$	r_{6n}
$\vdots$	$r_{:1}$	$r_{:2}$	$r_{:\ldots}$	$r_{:n}$
x_m	r_{m1}	r_{m2}	$r_{m\ldots}$	r_{mn}

Target objective

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

  "cost sensitive classification"

Calibrated surrogates exist for $\mathbf{r}_i \cdot \boldsymbol{\pi}(x_i)$
(Pires et al. ICML-2013)

Interesting alternative

- entropy regularized expected reward

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

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

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

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

$q : X \to \Re^n$ neural network

Optimization objectives

Back to **general** rewards

	a_1	a_2	$\dots$	a_n
x_1	r_{11}	r_{12}	$r_{1\dots}$	r_{1n}
x_2	r_{21}	r_{22}	$r_{2\dots}$	r_{2n}
x_3	r_{31}	r_{32}	$r_{3\dots}$	r_{3n}
x_4	r_{41}	r_{42}	$r_{4\dots}$	r_{4n}
x_5	r_{51}	r_{52}	$r_{5\dots}$	r_{5n}
x_6	r_{61}	r_{62}	$r_{6\dots}$	r_{6n}
$\vdots$	$r_{:1}$	$r_{:2}$	$r_{:\dots}$	$r_{:n}$
x_m	r_{m1}	r_{m2}	$r_{m\dots}$	r_{mn}

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

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

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

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

Entropy regularized expected reward

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

Suggests a natural surrogate

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

- convex in $\mathbf{q}$

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

Optimization objectives

Back to **general** rewards

	a_1	a_2	$\dots$	a_n
x_1	r_{11}	r_{12}	$r_{1\dots}$	r_{1n}
x_2	r_{21}	r_{22}	$r_{2\dots}$	r_{2n}
x_3	r_{31}	r_{32}	$r_{3\dots}$	r_{3n}
x_4	r_{41}	r_{42}	$r_{4\dots}$	r_{4n}
x_5	r_{51}	r_{52}	$r_{5\dots}$	r_{5n}
x_6	r_{61}	r_{62}	$r_{6\dots}$	r_{6n}
$\vdots$	$r_{:1}$	$r_{:2}$	$r_{:\dots}$	$r_{:n}$
x_m	r_{m1}	r_{m2}	$r_{m\dots}$	r_{mn}

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

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

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

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

Comparison to maximum likelihood

before $-\mathbf{r} \cdot \log \boldsymbol{\pi} = F(\mathbf{q}) - \mathbf{q} \cdot \mathbf{r}$

now **KL**$(\mathbf{p} \| \boldsymbol{\pi}) \equiv F(\mathbf{q}) - \mathbf{q} \cdot \mathbf{p}$

If $\mathbf{r} = \mathbf{1}_a$ is an indicator

- become equivalent as $\tau \to 0$
- $\lim_{\tau \to 0} \mathbf{p} = \mathbf{r} = \mathbf{1}_a$
- but $\tau > 0$ gives soft targets for **KL**
  "label smoothing"
  improves generalization in practice

Optimization objectives

Back to **general** rewards

	a_1	a_2	$\ldots$	a_n
x_1	r_{11}	r_{12}	$r_{1\ldots}$	r_{1n}
x_2	r_{21}	r_{22}	$r_{2\ldots}$	r_{2n}
x_3	r_{31}	r_{32}	$r_{3\ldots}$	r_{3n}
x_4	r_{41}	r_{42}	$r_{4\ldots}$	r_{4n}
x_5	r_{51}	r_{52}	$r_{5\ldots}$	r_{5n}
x_6	r_{61}	r_{62}	$r_{6\ldots}$	r_{6n}
$\vdots$	$r_{:1}$	$r_{:2}$	$r_{:\ldots}$	$r_{:n}$
x_m	r_{m1}	r_{m2}	$r_{m\ldots}$	r_{mn}

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

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

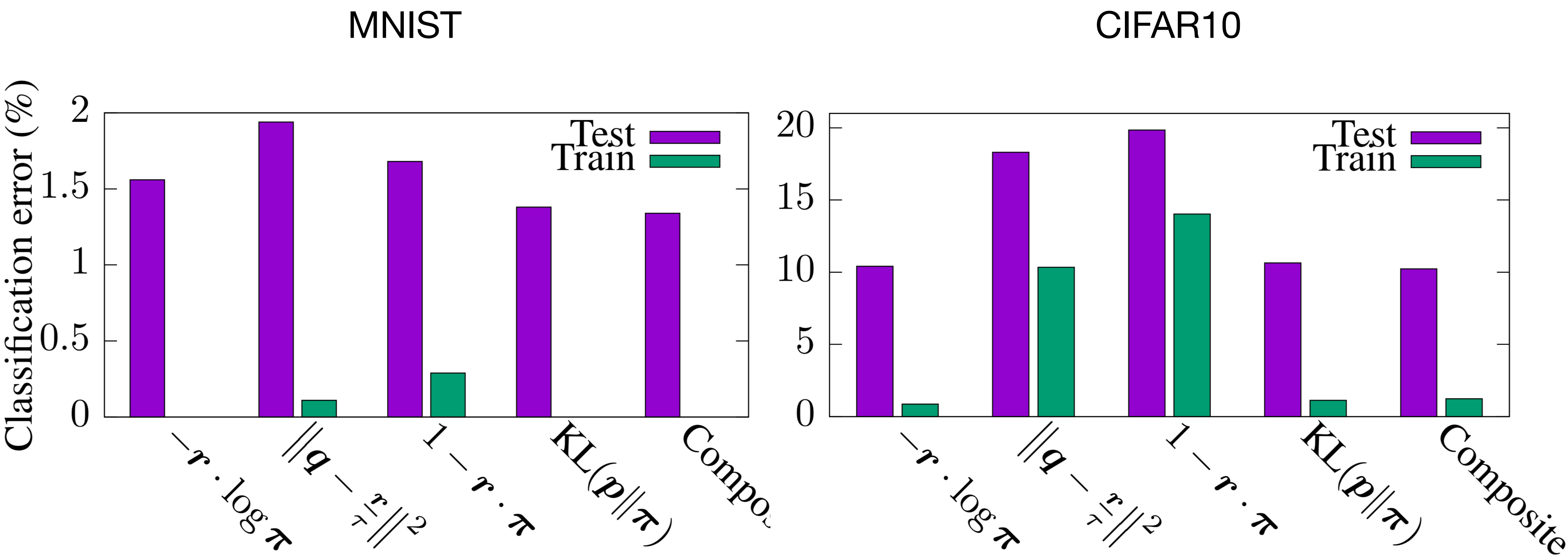
$$F(q(x)) = \log \sum_a e^{q(x)_a}$$

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

A convex, calibrated upper bound

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

Optimization objectives



Batch policy optimization

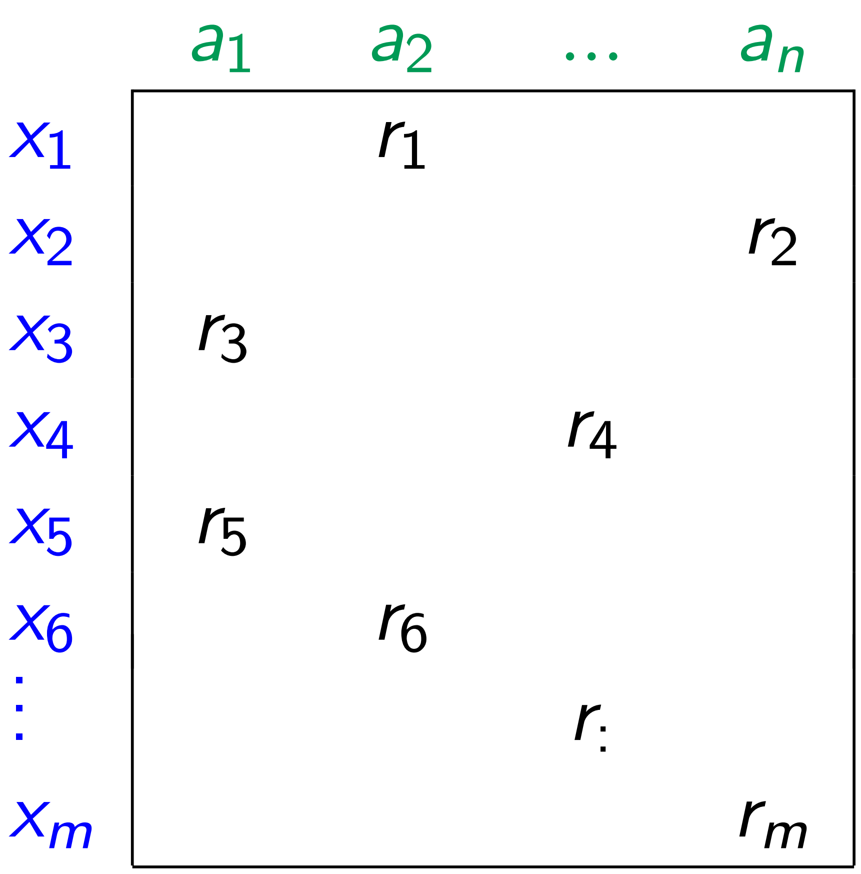
	a_1	a_2	$\dots$	a_n
x_1	r_{11}	r_{12}	$r_{1\dots}$	r_{1n}
x_2	r_{21}	r_{22}	$r_{2\dots}$	r_{2n}
x_3	r_{31}	r_{32}	$r_{3\dots}$	r_{3n}
x_4	r_{41}	r_{42}	$r_{4\dots}$	r_{4n}
x_5	r_{51}	r_{52}	$r_{5\dots}$	r_{5n}
x_6	r_{61}	r_{62}	$r_{6\dots}$	r_{6n}
$\vdots$	$r_{:1}$	$r_{:2}$	$r_{:\dots}$	$r_{:n}$
x_m	r_{m1}	r_{m2}	$r_{m\dots}$	r_{mn}

Three key issues

1. generalization
2. optimization
3. missing data

training objective
≠
target objective

Batch policy optimization



Three key issues

1. generalization
2. optimization
3. missing data

Supervised vs reinforcement learning

supervised classification

	a_1	a_2	$\dots$	a_n
x_1	r_{11}	r_{12}	$r_{1\dots}$	r_{1n}
x_2	r_{21}	r_{22}	$r_{2\dots}$	r_{2n}
x_3	r_{31}	r_{32}	$r_{3\dots}$	r_{3n}
x_4	r_{41}	r_{42}	$r_{4\dots}$	r_{4n}
x_5	r_{51}	r_{52}	$r_{5\dots}$	r_{5n}
x_6	r_{61}	r_{62}	$r_{6\dots}$	r_{6n}
$\vdots$	$r_{:1}$	$r_{:2}$	$r_{:\dots}$	$r_{:n}$
x_m	r_{m1}	r_{m2}	$r_{m\dots}$	r_{mn}

batch policy optimization

	a_1	a_2	$\dots$	a_n
x_1		r_1		
x_2				r_2
x_3	r_3			
x_4			r_4	
x_5	r_5			
x_6		r_6		
$\vdots$			$r_:$	
x_m				r_m

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?

	a_1	a_2	$\dots$	a_n
x_1		r_1		
x_2				r_2
x_3	r_3			
x_4			r_4	
x_5	r_5			
x_6		r_6		
$\vdots$			$r_{\vdots}$	
x_m				r_m

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

Missing data inference

	a_1	a_2	$\dots$	a_n
x_1	q_{11}	r_1	$q_{1\dots}$	q_{1n}
x_2	q_{21}	q_{22}	$q_{2\dots}$	r_2
x_3	r_3	q_{32}	$q_{3\dots}$	q_{3n}
x_4	q_{41}	q_{42}	r_4	q_{4n}
x_5	r_5	q_{52}	$q_{5\dots}$	q_{5n}
x_6	q_{61}	r_6	$q_{6\dots}$	q_{6n}
$\vdots$	$q_{:1}$	$q_{:2}$	$r_:$	$q_{:n}$
x_m	q_{m1}	q_{m2}	$q_{m\dots}$	r_m

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

Missing data inference

	a_1	a_2	…	a_n
x_1	0	$\frac{r_1}{\beta_1}$	0	0
x_2	0	0	0	$\frac{r_2}{\beta_2}$
x_3	$\frac{r_3}{\beta_3}$	0	0	0
x_4	0	0	$\frac{r_4}{\beta_4}$	0
x_5	$\frac{r_5}{\beta_5}$	0	0	0
x_6	0	$\frac{r_6}{\beta_6}$	0	0
⋮	0	0	$\frac{r_:}{\beta_:}$	0
x_m	0	0	0	$\frac{r_m}{\beta_m}$

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

Example

importance corrected expected reward

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

where β are proposal probabilities from behavior strategy

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

Equivalent to $\max \hat{\mathbf{r}} \cdot \boldsymbol{\pi}$ using

$$\hat{\mathbf{r}}_i = \mathbf{1}_{a_i} \frac{r_i}{\beta_i}$$

That is

- exaggerate observed values by $1/\beta_i$
- fill in all unobserved values with 0

Missing data inference

	a_1	a_2	…	a_n
x_1	0	$\frac{r_1}{\beta_1}$	0	0
x_2	0	0	0	$\frac{r_2}{\beta_2}$
x_3	$\frac{r_3}{\beta_3}$	0	0	0
x_4	0	0	$\frac{r_4}{\beta_4}$	0
x_5	$\frac{r_5}{\beta_5}$	0	0	0
x_6	0	$\frac{r_6}{\beta_6}$	0	0
⋮	0	0	$\frac{r_:}{\beta_:}$	0
x_m	0	0	0	$\frac{r_m}{\beta_m}$

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!

$$\mathbb{E}[\hat{\mathbf{r}} \,|\, x] = \sum_a \beta_a \mathbf{1}_a \frac{r_a}{\beta_a} = \sum_a \mathbf{1}_a r_a = \mathbf{r}$$

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

Missing data inference

	a_1	a_2	$\dots$	a_n
x_1	τq_{11}	$\lambda(r_1 - \tau q_{12})$	$\tau q_{1\dots}$	τq_{1n}
x_2	τq_{21}	τq_{22}	$\tau q_{2\dots}$	$\lambda(r_2 - \tau q_{2n})$
x_3	$\lambda(r_3 - \tau q_{31})$	τq_{32}	$\tau q_{3\dots}$	τq_{3n}
x_4	τq_{41}	τq_{42}	$\lambda(r_4 - \tau q_{4\dots})$	τq_{4n}
x_5	$\lambda(r_5 - \tau q_{51})$	τq_{52}	$\tau q_{5\dots}$	τq_{5n}
x_6	τq_{61}	$\lambda(r_6 - \tau q_{62})$	$\tau q_{6\dots}$	τq_{6n}
$\vdots$	$\tau q_{:1}$	$\tau q_{:2}$	$\lambda(r_: - \tau q_{:\dots})$	$\tau q_{:n}$
x_m	τq_{m1}	τq_{m2}	$\tau q_{m\dots}$	$\lambda(r_m - \tau q_{mn})$

Improvement

"douby robust estimation"

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

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

Also unbiased

- as long as $\lambda = 1/\beta_i$

but still alters observed data

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

Missing data inference

	a_1	a_2	$\dots$	a_n
x_1	τq_{11}	$\lambda(r_1 - \tau q_{12})$	$\tau q_{1\dots}$	τq_{1n}
x_2	τq_{21}	τq_{22}	$\tau q_{2\dots}$	$\lambda(r_2 - \tau q_{2n})$
x_3	$\lambda(r_3 - \tau q_{31})$	τq_{32}	$\tau q_{3\dots}$	τq_{3n}
x_4	τq_{41}	τq_{42}	$\lambda(r_4 - \tau q_{4\dots})$	τq_{4n}
x_5	$\lambda(r_5 - \tau q_{51})$	τq_{52}	$\tau q_{5\dots}$	τq_{5n}
x_6	τq_{61}	$\lambda(r_6 - \tau q_{62})$	$\tau q_{6\dots}$	τq_{6n}
$\vdots$	$\tau q_{:1}$	$\tau q_{:2}$	$\lambda(r_: - \tau q_{:\dots})$	$\tau q_{:n}$
x_m	τq_{m1}	τq_{m2}	$\tau q_{m\dots}$	$\lambda(r_m - \tau q_{mn})$

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

Improvement

"douby robust estimation"

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

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

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 $\boldsymbol{\pi}$
- 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

	a_1	a_2	$\dots$	a_n
x_1	τq_{11}	$\lambda(r_1 - \tau q_{12})$	$\tau q_{1\dots}$	τq_{1n}
x_2	τq_{21}	τq_{22}	$\tau q_{2\dots}$	$\lambda(r_2 - \tau q_{2n})$
x_3	$\lambda(r_3 - \tau q_{31})$	τq_{32}	$\tau q_{3\dots}$	τq_{3n}
x_4	τq_{41}	τq_{42}	$\lambda(r_4 - \tau q_{4\dots})$	τq_{4n}
x_5	$\lambda(r_5 - \tau q_{51})$	τq_{52}	$\tau q_{5\dots}$	τq_{5n}
x_6	τq_{61}	$\lambda(r_6 - \tau q_{62})$	$\tau q_{6\dots}$	τq_{6n}
$\vdots$	$\tau q_{:1}$	$\tau q_{:2}$	$\lambda(r_: - \tau q_{:\dots})$	$\tau q_{:n}$
x_m	τq_{m1}	τq_{m2}	$\tau q_{m\dots}$	$\lambda(r_m - \tau q_{mn})$

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

Unified approach

- actor and critic are same model
- $\boldsymbol{\pi} = e^{\mathbf{q} - F(\mathbf{q})}$ where $F(\mathbf{q}) = \log \mathbf{1} \cdot e^{\mathbf{q}}$
- use logits $\tau \mathbf{q}(x)$ to predict rewards

$$q(x, a) \approx \frac{r(x, a)}{\tau}$$

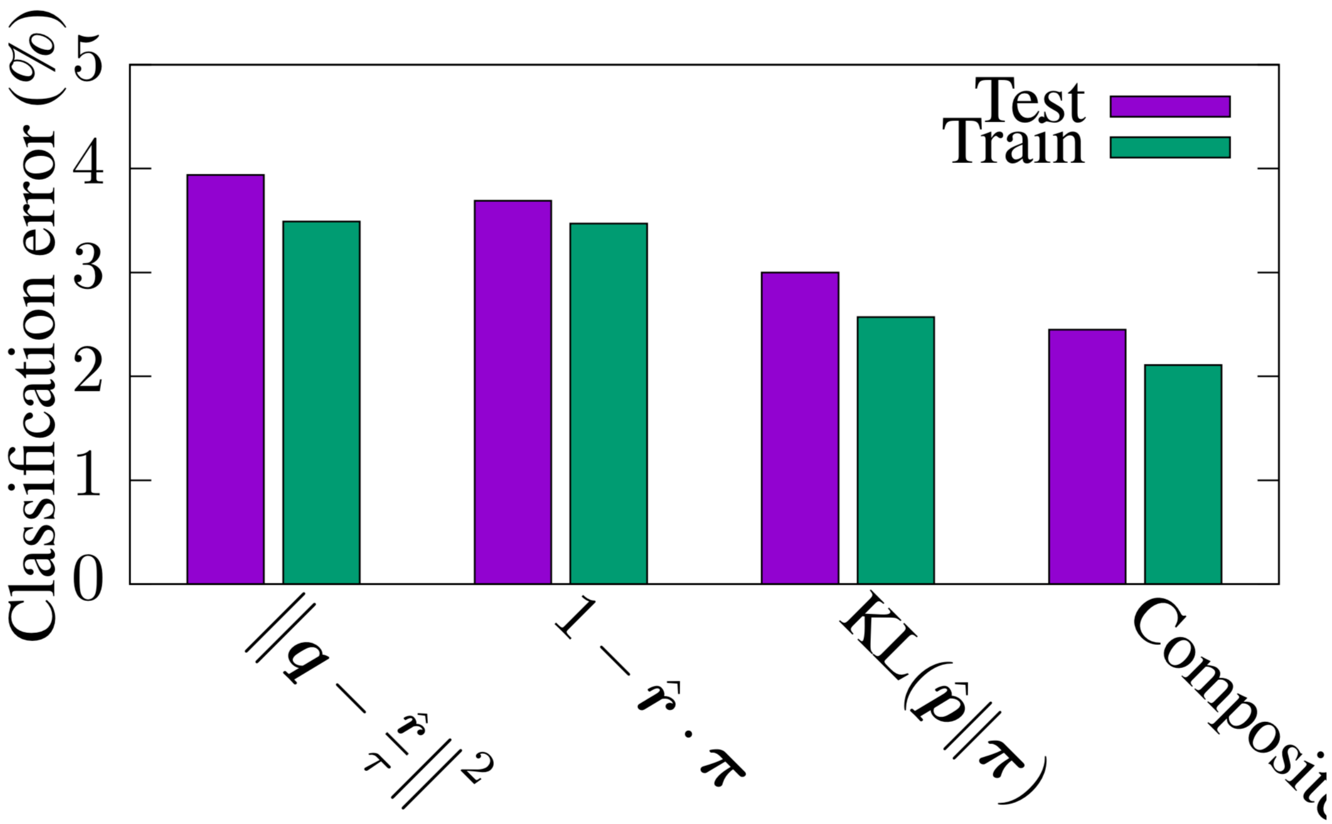
Can combine with previous objectives

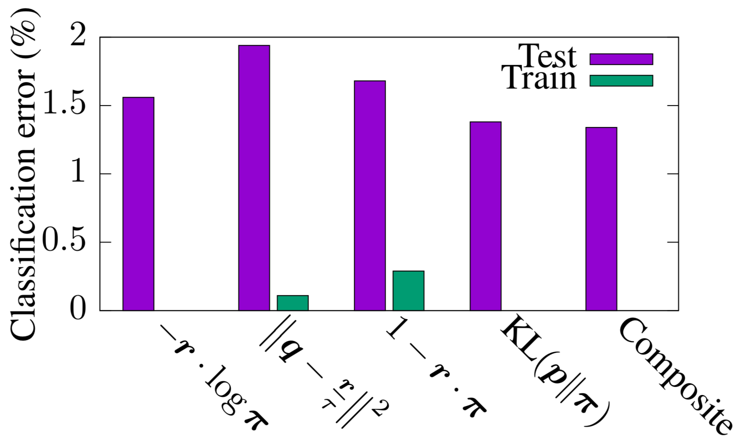
- **KL**$(\boldsymbol{\pi} \| \hat{\mathbf{p}})$ where $\hat{\mathbf{p}} = e^{\hat{\mathbf{r}}/\tau - F(\hat{\mathbf{r}}/\tau)}$
- **KL**$(\hat{\mathbf{p}} \| \boldsymbol{\pi})$
- **KL**$(\boldsymbol{\pi} \| \hat{\mathbf{p}}) \leq$ **KL**$(\hat{\mathbf{p}} \| \boldsymbol{\pi}) + \frac{\tau}{4} \|\hat{\mathbf{r}}/\tau - \mathbf{q}\|^2$

these are somewhat sensitive to ranking, unlike least squares

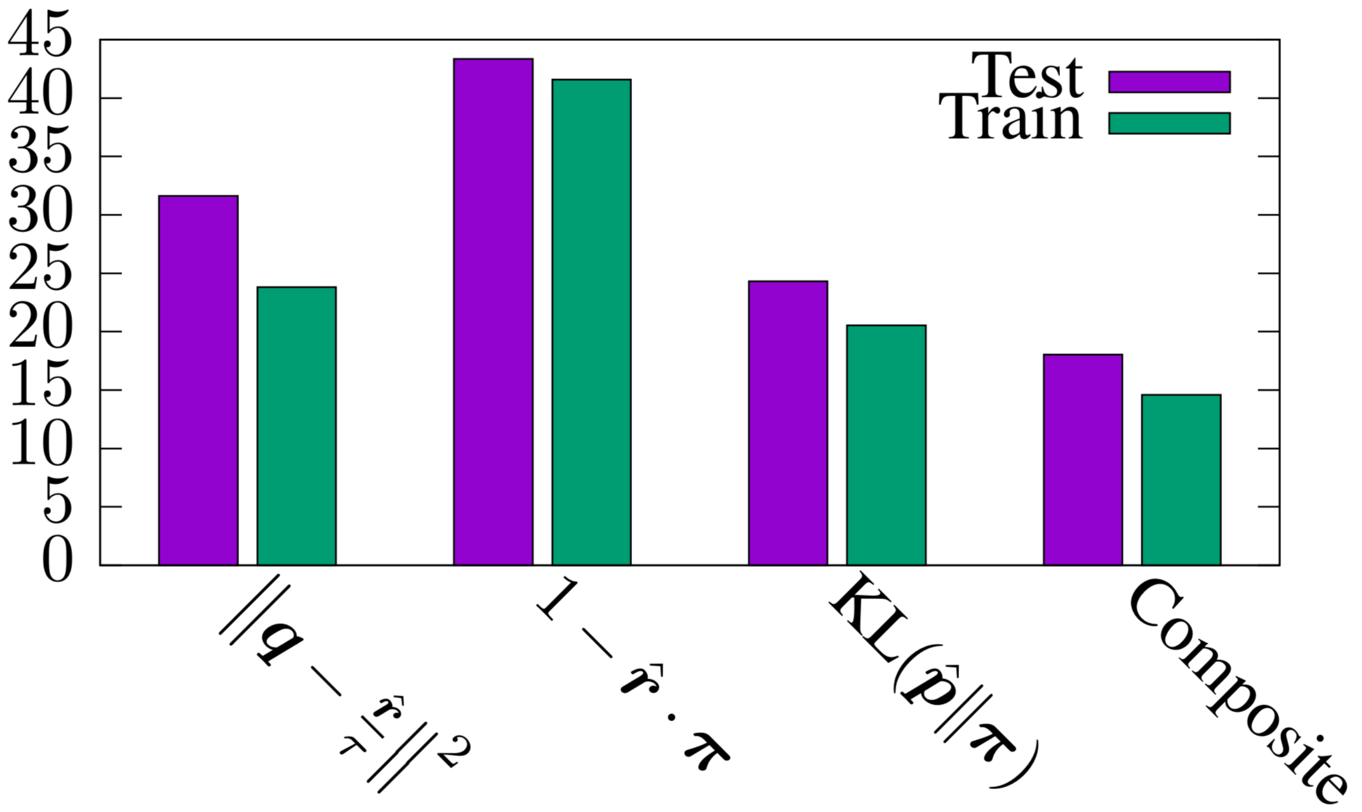
Missing data inference

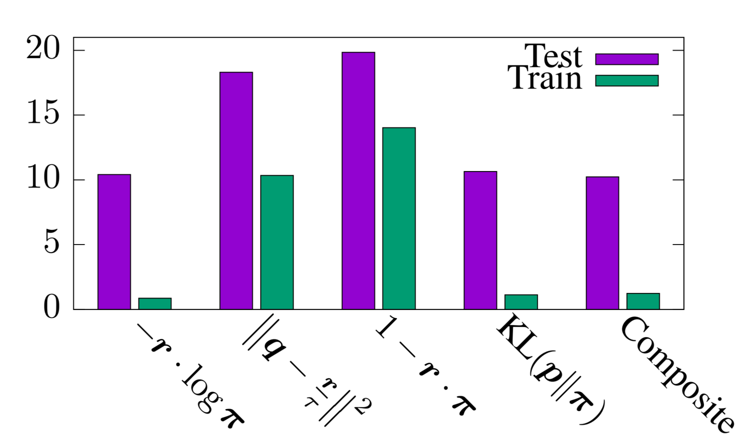
MNIST





CIFAR10





Missing data inference

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

surrogate loss $L(\mathbf{q}, \mathbf{r}, x) = \tau D_F(\mathbf{q}(x) \| \frac{\mathbf{r}}{\tau}) + \frac{\tau}{4} \|\mathbf{q}(x) - \frac{\mathbf{r}}{\tau}\|^2$ $\hat{L}(\mathbf{q}, \mathscr{D}) = \frac{1}{T} \sum_{i \in \mathscr{D}} L(\mathbf{q}, \hat{\mathbf{r}}_i, x_i)$

smoothed risk $\mathscr{S}_\tau(\boldsymbol{\pi}, \mathbf{r}, x) = -\mathbf{r} \cdot \boldsymbol{\pi}(x) + \tau \boldsymbol{\pi}(x) \cdot \log \boldsymbol{\pi}(x)$ $\mathscr{S}_\tau(\boldsymbol{\pi}) = \mathbb{E}[\mathscr{S}_\tau(\boldsymbol{\pi}, \mathbf{r}, x)]$

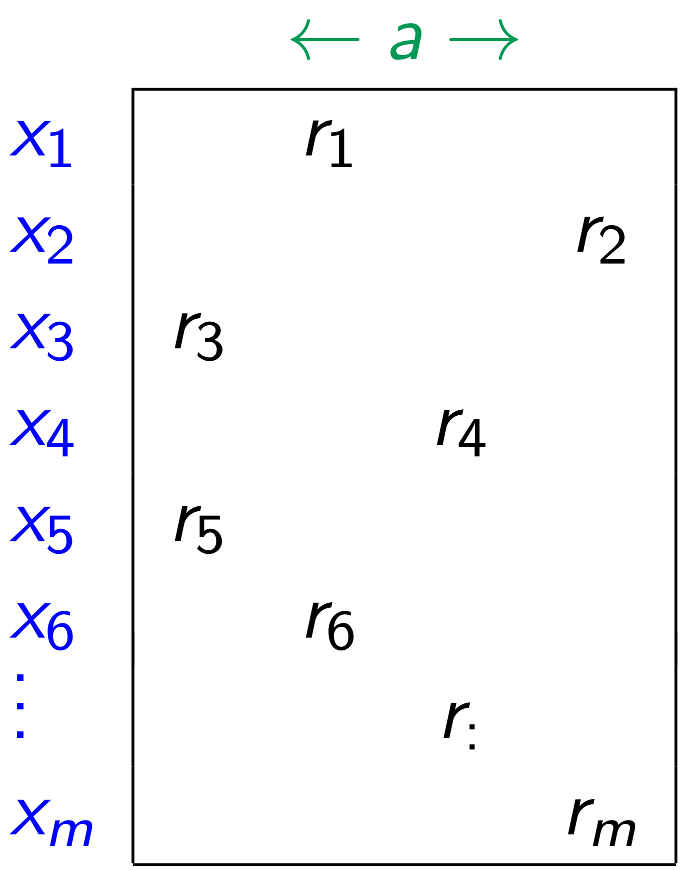
suboptimality gap $\mathscr{G}_\tau(\boldsymbol{\pi}) = \mathscr{S}_\tau(\boldsymbol{\pi}) - \mathscr{S}_\tau^*$ $\mathscr{S}_\tau^* = \inf_{\mathbf{q} \in \mathscr{Q}} \mathscr{S}_\tau(\mathbf{f} \circ \mathbf{q})$

Theorem (informally): If $\mathscr{H}, \beta, p(x, \mathbf{r}), \hat{\mathbf{r}}$ are "well behaved", then:

$\forall \tau, \delta > 0 \, \exists C$ s.t. w.p. $\geq 1 - \delta$: if $\hat{L}(\mathbf{q}, \mathscr{D}) < \frac{\tau C}{\sqrt{T}}$ for $\mathbf{q} \in \mathscr{H}$ then $\mathscr{G}_\tau(\mathbf{f} \circ \mathbf{q}) \leq \frac{2\tau C}{\sqrt{T}}$

small empirical surrogate implies small true suboptimality gap

Missing data inference



Optimize policy $\pi : X \to$ exp-family$(\mathfrak{R})$

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

$$F(q(x)) = \log \int e^{q(x)_a} \mu(da)$$

$q : X \to \mathfrak{R}^k$ neural network

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

$$q \to \xi \to r$$

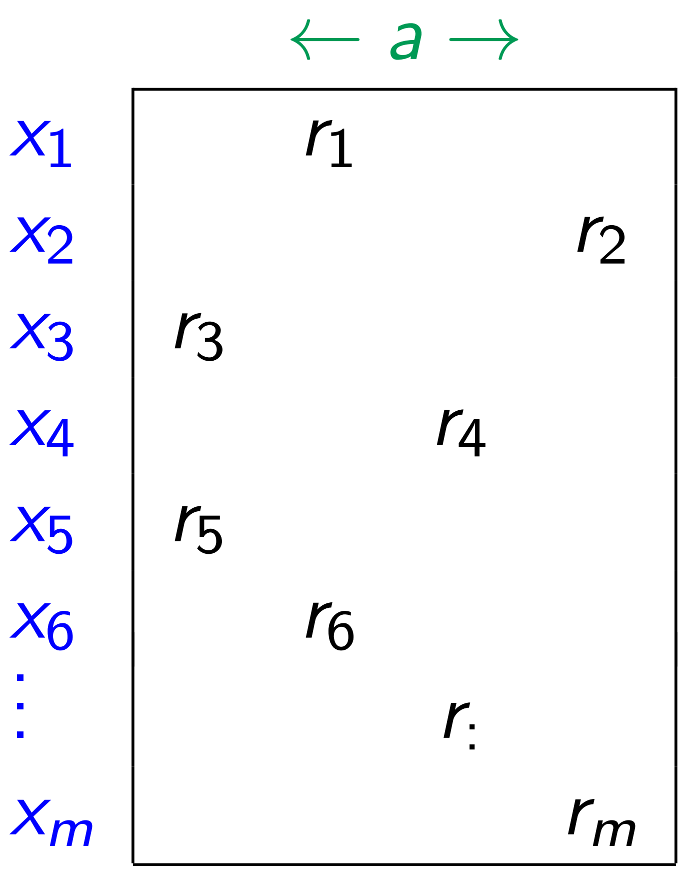
- **e.g. Gaussian**
  - prior $\xi \sim \mathcal{N}(q, Q)$
  - likelihood $r\,|\,a, \xi \sim \mathcal{N}(\phi(a) \cdot \xi, \sigma^2)$
  - posterior $\xi\,|\,r_0, a_0 \sim \mathcal{N}(\mu, C)$

$$\mu = C(\phi(a_0) r_0 \sigma^{-2} + Q^{-1} q)$$
$$C = (Q^{-1} + \sigma^{-2} \phi(a_0) \phi(a_0)^\top)^{-1}$$

  - predictive $r\,|\,a, r_0, a_0 \sim \mathcal{N}(\hat{\mu}, \hat{\sigma}^2)$

$$\hat{\mu} = \phi(a) \cdot \mu$$
$$\hat{\sigma}^2 = \sigma^2 + \phi(a_0)^\top C \phi(a_0)$$

Missing data inference



Optimize policy $\pi : X \to$ exp-family$(\mathfrak{R})$

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

$$F(q(x)) = \log \int e^{q(x)_a} \mu(da)$$

$q : X \to \mathfrak{R}^k$ neural network

Empirical Bayes estimation

- optimize hyperparameters $\mathbf{q}$ (neural network)
- integrate out parameters ξ

Example

marginal likelihood

$$-\log p(r_0 \,|\, a_0, \mathbf{q})$$

$$= -\log \int p(r_0 \,|\, a_0, \xi) p(\xi \,|\, \mathbf{q}) \, d\xi$$

$$= \frac{1}{2\sigma^2}(\phi(a_0) \cdot q - r_0)^2 + \frac{1}{2} \log \sigma^2 + c$$

- essentially least squares regression

Can alternatively use surrogates

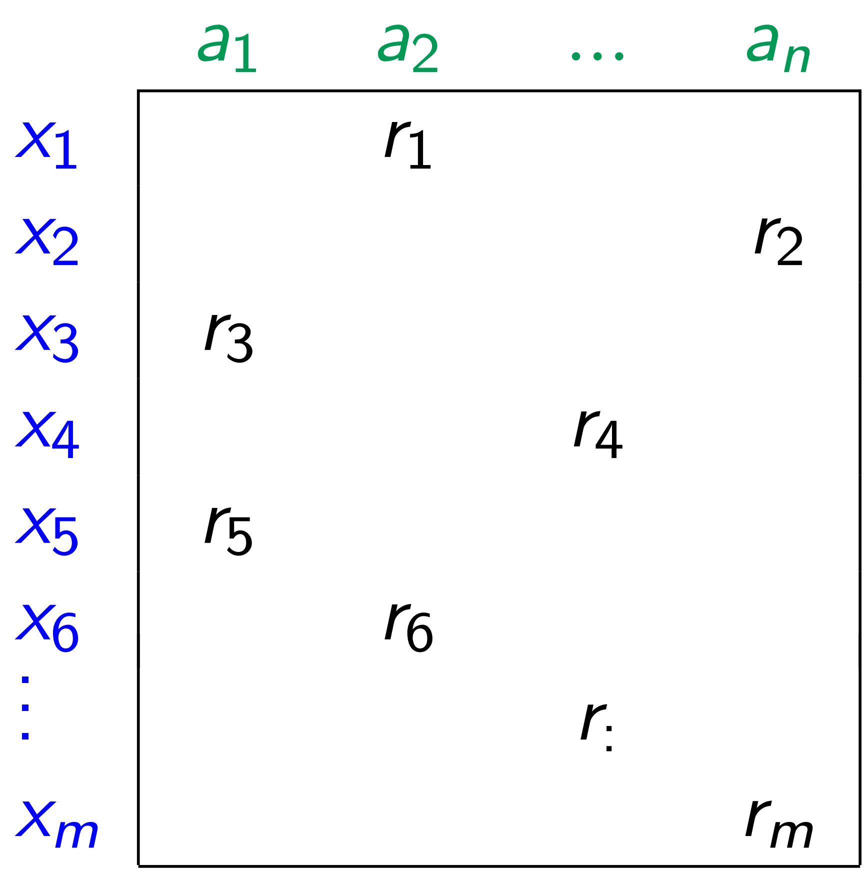
$\min$ **KL**(prior||posterior)

$\min$ **KL**(posterior||prior) $\approx \min I(\xi; r_0)$

Missing data inference

Sum of squared test error on continuous action MNIST ($a \in \Re^{10}$)

Batch policy optimization



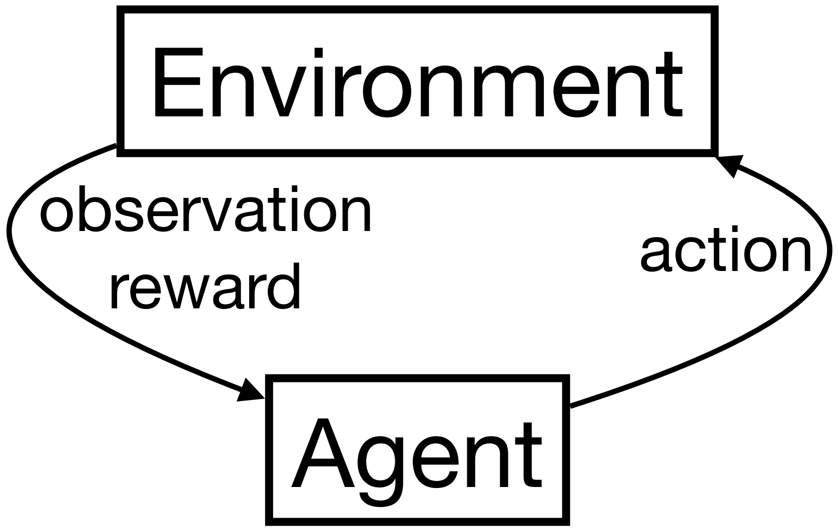
Three key issues

1. generalization
2. optimization
3. missing data

training objective $\neq$ target objective

classical methods still help

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 $\quad \forall s, a \quad q_{sa} = r_{sa} + \gamma \sum_{s'} p_{sas'} \max_{a'} q_{s'a'}$

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

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

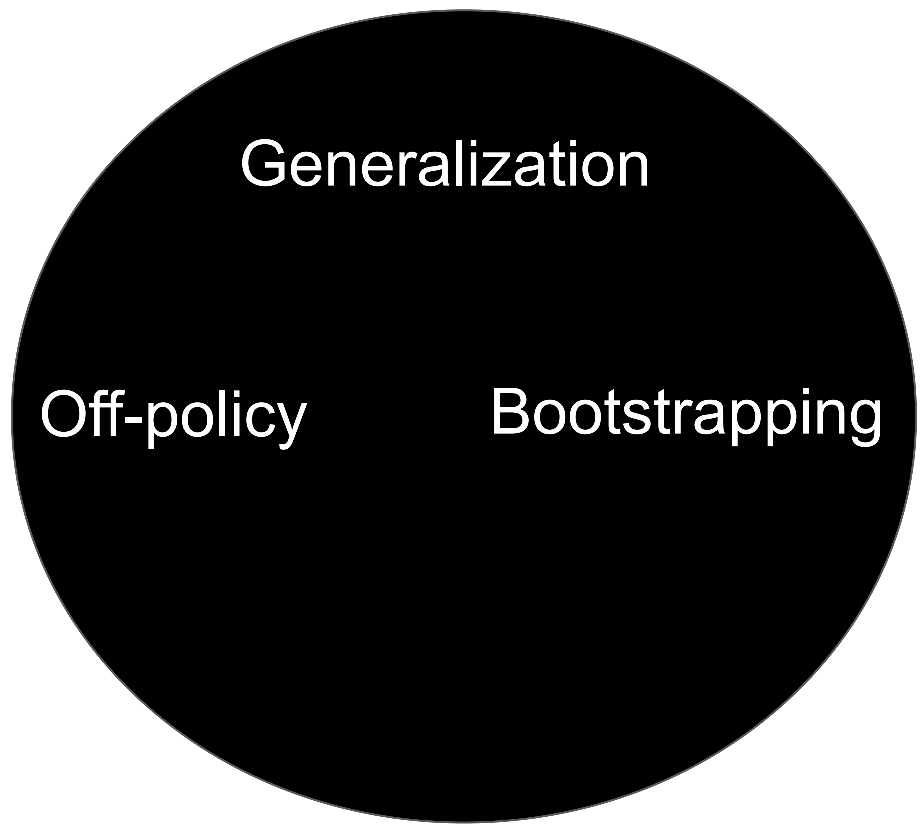
$$\sum_{s,a} d_{sa} \left(\hat{q}_{sa} - r_{sa} - \gamma \sum_{s'} p_{sas'} \max_{a'} \hat{q}_{s'a'} \right) \left(\frac{d\hat{q}_{sa}}{d\theta} - \gamma \sum_{s'} p_{sas'} \frac{d\hat{q}_{sa^*(s)}}{d\theta} \right)$$

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

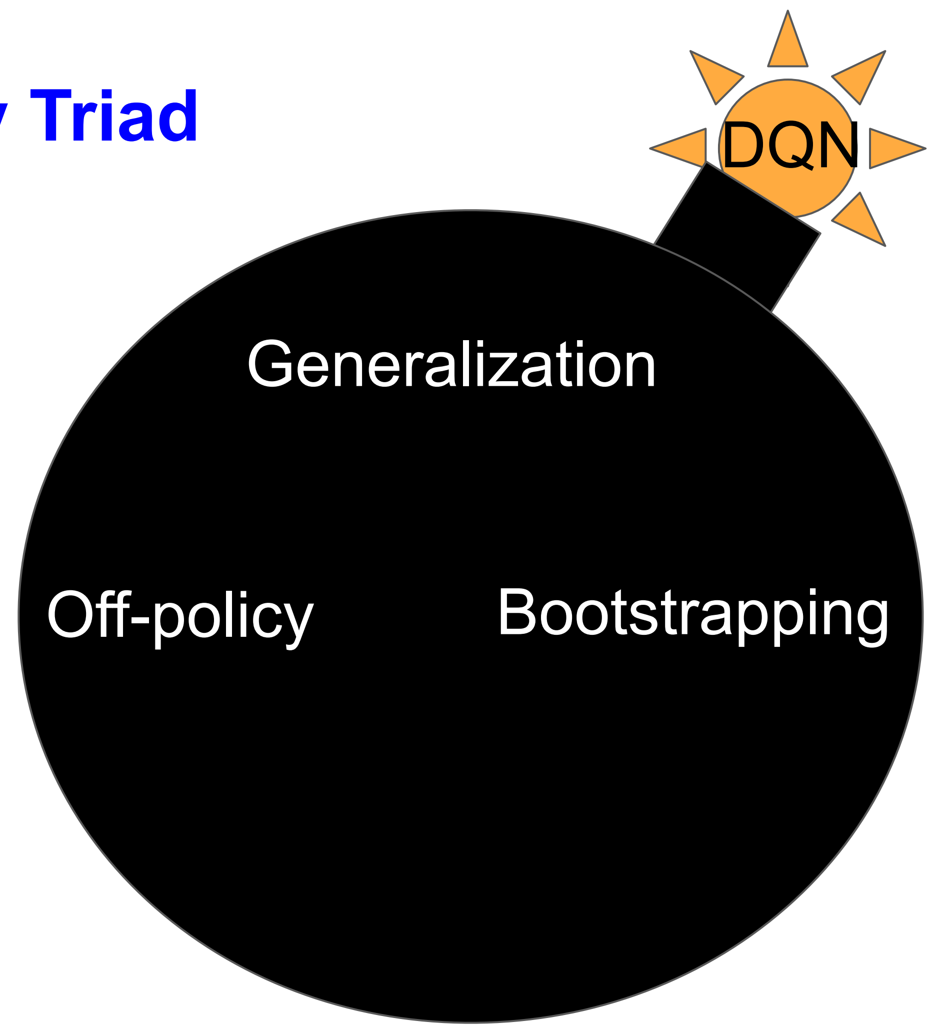
The Deadly Triad



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

Sequential RL

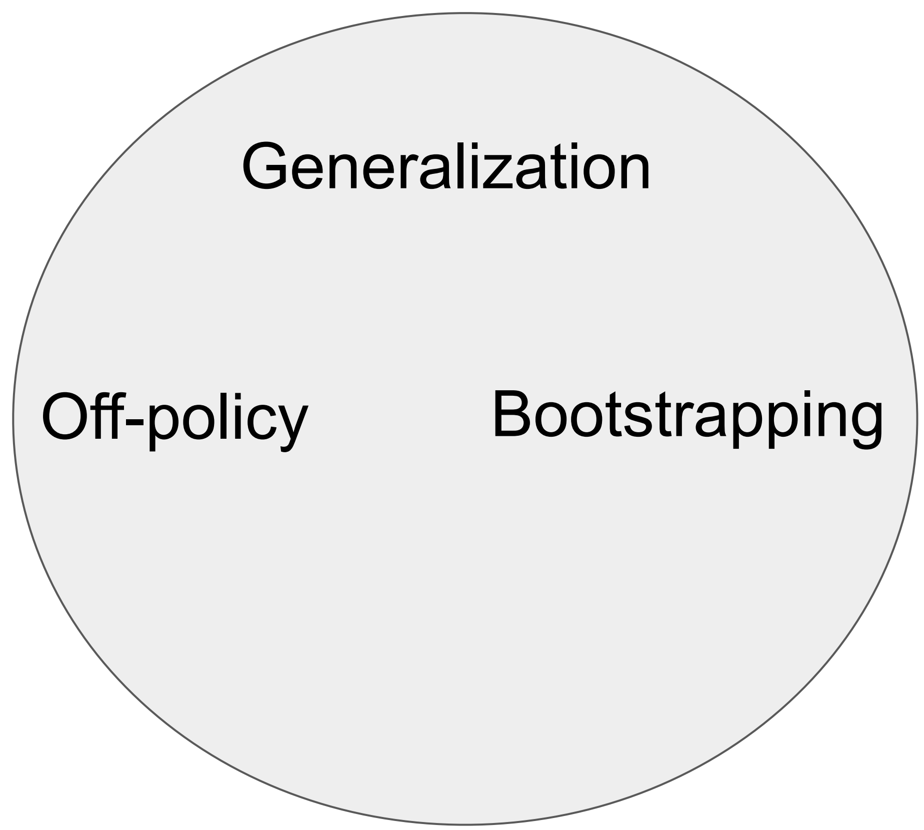
The Deadly Triad



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

Sequential RL

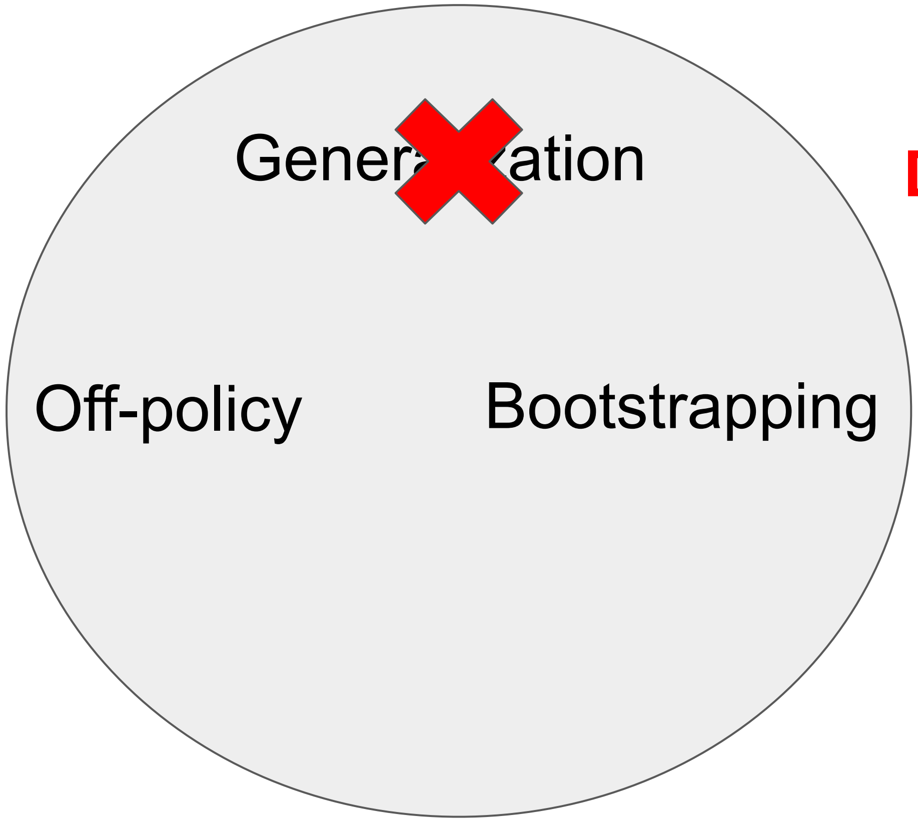
Back to Basics



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

Sequential RL

Back to Basics

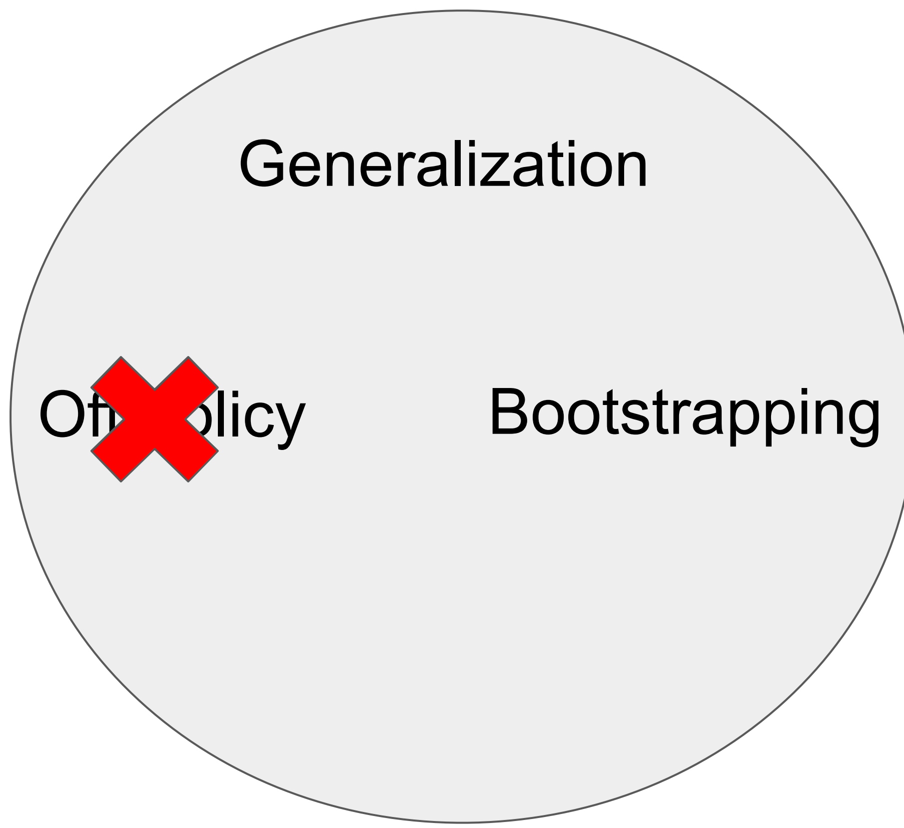


Don't be crazy

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

Sequential RL

Back to Basics



On policy methods

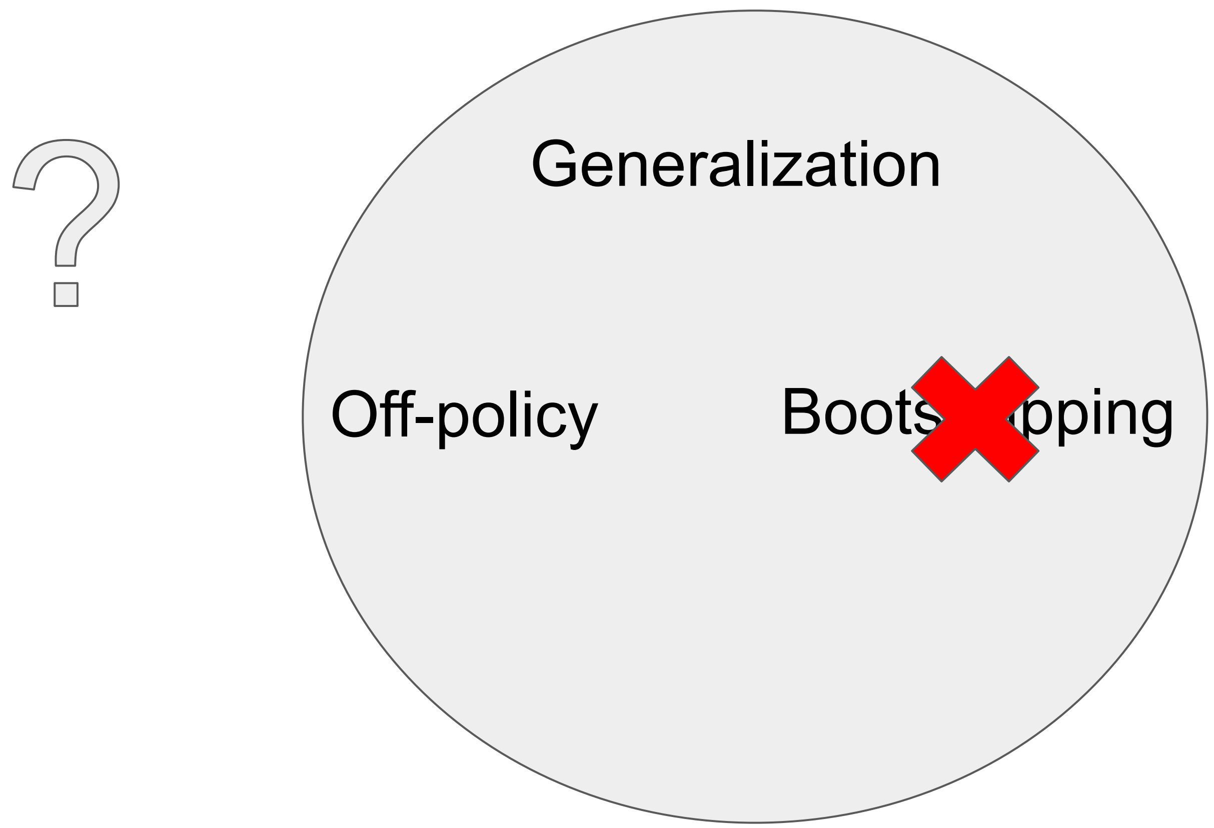
- Policy gradient
- Actor-critic

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

Data inefficient

Sequential RL

Back to Basics



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

Sequential RL

Avoiding the bootstrap

1. Multiple hypothesis tracking (NeurIPS-2018)

2. Monte Carlo policy iteration

3. Lagrange dual: joint state-action distributions

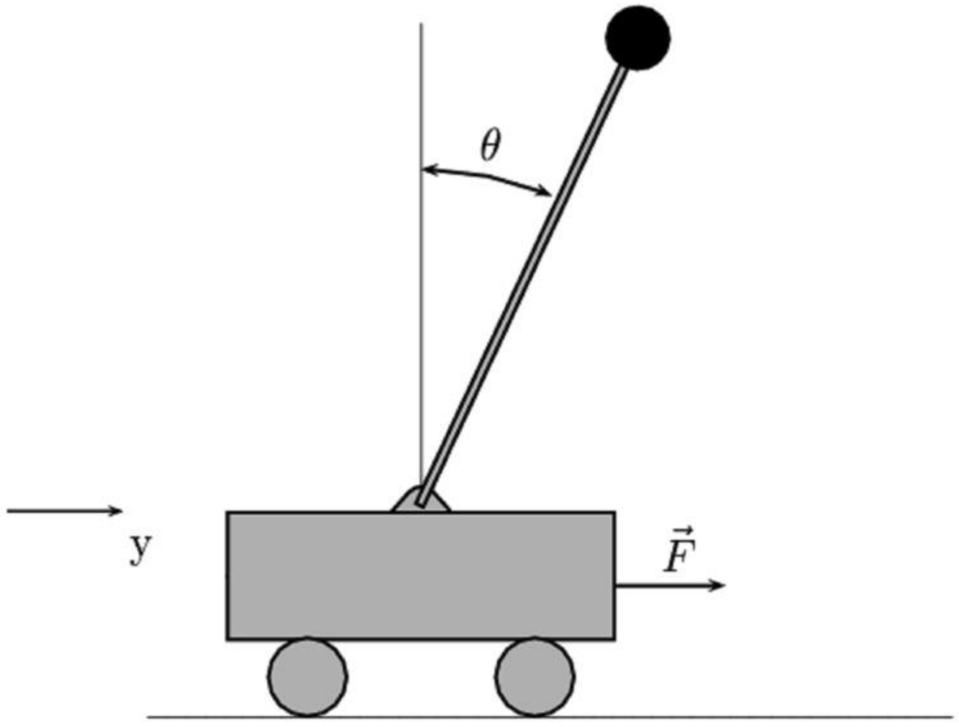
Sequential RL

Avoiding the bootstrap

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

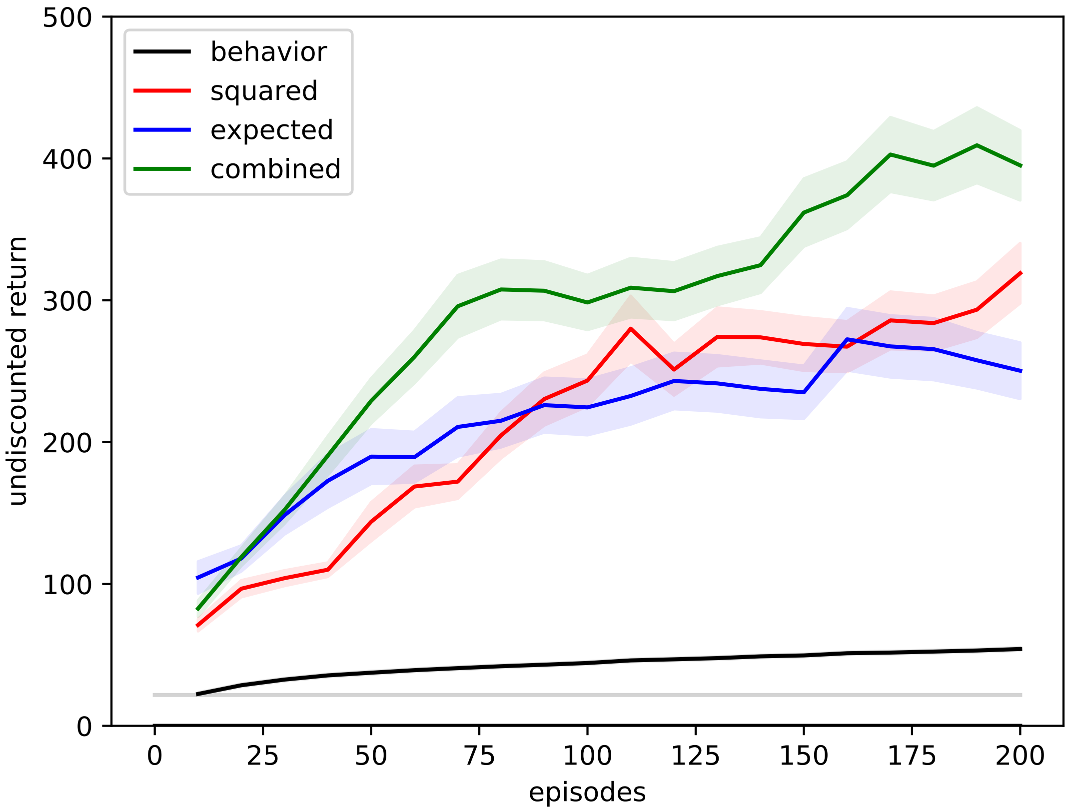
Sequential RL

Cart-Pole



using coordinate features

Policy improvement



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

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

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

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