Learning in Markov Decision Problems

Peter Bartlett

Computer Science and Statistics
University of California at Berkeley

Mathematical Sciences
Queensland University of Technology

UCLA
November 10, 2014
## MDP: Managing Threatened Species

For $t = 1, 2, \ldots$:

<table>
<thead>
<tr>
<th>State $X_t$</th>
<th>Action $A_t$</th>
<th>Loss $\ell(X_t, A_t)$</th>
<th>State Evolution $X_{t+1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Transition matrix:** $P: X \times A \to \Delta(X)$

**Policy:** $\pi: X \to \Delta(A)$

**Stationary distribution:** $\mu$

**Average loss:** $\mu T \ell$.

**Performance Measure:** $\frac{2}{27}$
MDP: Managing Threatened Species

For \( t = 1, 2, \ldots \):

1. See state \( X_t \) of ecosystem
2. Play an action \( A_t \)
3. Incur loss \( \ell(X_t, A_t) \)
4. State evolves to \( X_{t+1} \sim P(X_t, A_t) \)

Transition matrix: \( P : \Delta(X) \times \Delta(A) \rightarrow \Delta(X) \)

Policy: \( \pi : \Delta(X) \rightarrow \Delta(A) \)

Stationary distribution: \( \mu \)

Average loss: \( \mathbb{E}_\mu[T\ell] \)

Performance Measure: \( \mathbb{E}_\mu[T] \)
MDP: Managing Threatened Species

For $t = 1, 2, \ldots$:

1. See state $X_t$ of ecosystem
2. Play an action $A_t$ intervention
MDP: Managing Threatened Species

For $t = 1, 2, \ldots$:

1. See state $X_t$ of ecosystem
2. Play an action $A_t$ intervention
   anti-poaching patrols
Markov Decision Processes

**MDP: Managing Threatened Species**

For $t = 1, 2, \ldots$:

1. See state $X_t$ of ecosystem
2. Play an action $A_t$ intervention anti-poaching patrols
3. Incur loss $\ell(X_t, A_t)$
MDP: Managing Threatened Species

For $t = 1, 2, \ldots$:

1. See state $X_t$ of ecosystem
2. Play an action $A_t$ intervention anti-poaching patrols
3. Incur loss $\ell(X_t, A_t)$, extinction

Performance Measure: $\frac{2}{27}$
MDP: Web Customer Interactions

For $t = 1, 2, \ldots$:

1. See state $X_t$ of customer
2. Play an action $A_t$ interaction offer/advertisement
3. Incur loss $\ell(X_t, A_t)$ missed revenue
### MDP: Managing Threatened Species

For $t = 1, 2, \ldots$:

1. See state $X_t$ of ecosystem
2. Play an action $A_t$ intervention anti-poaching patrols
3. Incur loss $\ell(X_t, A_t)$ $\$, extinction
4. State evolves to $X_{t+1} \sim P_{X_t, A_t}$

**Transition matrix:**

$P : \mathcal{X} \times \mathcal{A} \rightarrow \Delta(\mathcal{X'})$
Markov Decision Processes

MDP: Managing Threatened Species

For \( t = 1, 2, \ldots \):
1. See state \( X_t \) of ecosystem
2. Play an action \( A_t \) intervention anti-poaching patrols
3. Incur loss \( \ell(X_t, A_t) \), extinction
4. State evolves to \( X_{t+1} \sim P_{X_t, A_t} \)

Transition matrix:
\[ P : \mathcal{X} \times \mathcal{A} \rightarrow \Delta(\mathcal{X}) \]
Policy:
\[ \pi : \mathcal{X} \rightarrow \Delta(\mathcal{A}) \]

Performance Measure: Regret
\[
R_T = \mathbb{E} \sum_{t=1}^{T} \ell(X_t, A_t) - \min_{\pi} \mathbb{E} \sum_{t=1}^{T} \ell(X_t^\pi, \pi(X_t^\pi)).
\]
Markov Decision Processes

**MDP: Managing Threatened Species**

For $t = 1, 2, \ldots$:

1. See state $X_t$ of ecosystem
2. Play an action $A_t$ intervention anti-poaching patrols
3. Incur loss $\ell(X_t, A_t)$ $\$, extinction
4. State evolves to $X_{t+1} \sim P_{X_t, A_t}$

**Transition matrix:**

$P : \mathcal{X} \times \mathcal{A} \rightarrow \Delta(\mathcal{X})$

**Policy:**

$\pi : \mathcal{X} \rightarrow \Delta(\mathcal{A})$

**Stationary distribution:**

$\mu$

**Average loss:**

$\mu^T \ell$

**Performance Measure: Regret**

$$R_T = \mathbb{E} \sum_{t=1}^{T} \ell(X_t, A_t) - \min_{\pi} \mathbb{E} \sum_{t=1}^{T} \ell(X_t^\pi, \pi(X_t^\pi)).$$
Markov Decision Processes

MDP: Managing Threatened Species

For $t = 1, 2, \ldots$:

1. See state $X_t$ of ecosystem
2. Play an action $A_t$ intervention anti-poaching patrols
3. Incur loss $\ell(X_t, A_t)$ $\$, extinction
4. State evolves to $X_{t+1} \sim P_{X_t,A_t}$

Transition matrix:

$$P : \mathcal{X} \times \mathcal{A} \rightarrow \Delta(\mathcal{X}')$$

Policy:

$$\pi : \mathcal{X} \rightarrow \Delta(\mathcal{A})$$

Stationary distribution: $\mu$

Average loss:

$$\mu^T \ell.$$

Performance Measure: Excess Average Loss

$$\mu_{\pi}^T \ell - \min_{\pi} \mu_{\pi}^T \ell$$
Large MDP Problems:
When the state space $\mathcal{X}$ is large, we must scale back the ambition of optimal performance.
Large MDP Problems:
When the state space $\mathcal{X}$ is large, we must scale back the ambition of optimal performance:

- In comparison to a restricted family of policies $\Pi$.
  e.g., linear value function approximation.
  Want a strategy that competes with the best policy.
1. Large-Scale Policy Design

- Stochastic gradient convex optimization.
- Competitive with policies near the approximating class.
- Without knowledge of optimal policy.
- Simulation results: queueing, crowdsourcing.

2. Learning Changing Dynamics

- Changing MDP; complete information.
- Exponential weights strategy.
- Competitive with small comparison class $\Pi$.
- Computationally efficient if $\Pi$ has polynomial size.
- Hard for shortest path problems.
1. Large-Scale Policy Design

- Compete with a restricted family of policies $\Pi$:
  Linearly parameterized approximate stationary distributions.

2. Learning Changing Dynamics
1. Large-Scale Policy Design
   - Compete with a restricted family of policies $\Pi$:
     - Linearly parameterized exponentially transformed value function.

2. Learning Changing Dynamics
Outline

1. Large-Scale Policy Design
   - Compete with a restricted family of policies $\Pi$:
     - Linearly parameterized policies.
   - Stochastic gradient convex optimization.

2. Learning Changing Dynamics
1. Large-Scale Policy Design

- Compete with a restricted family of policies $\Pi$:
  - Linearly parameterized policies.
  - Stochastic gradient convex optimization.
- Competitive with policies near the approximating class.

2. Learning Changing Dynamics
1. Large-Scale Policy Design

- Compete with a restricted family of policies \( \Pi \):
  - Linearly parameterized policies.
  - Stochastic gradient convex optimization.
  - Competitive with policies near the approximating class.
  - Without knowledge of optimal policy.

2. Learning Changing Dynamics
1. Large-Scale Policy Design

- Compete with a restricted family of policies $\Pi$:
  - Linearly parameterized policies.
- Stochastic gradient convex optimization.
- Competitive with policies near the approximating class.
- Without knowledge of optimal policy.
- Simulation results: queueing, crowdsourcing.

2. Learning Changing Dynamics

- Changing MDP; complete information.
- Exponential weights strategy.
- Competitive with small comparison class $\Pi$.
- Computationally efficient if $\Pi$ has polynomial size.
- Hard for shortest path problems.
Large-scale policy design

Stationary distributions dual to value functions.

Let $\mu_\theta$ denote the stationary distribution of policy $\pi_\theta$. Find $\hat{\theta}$ such that $\mu_\theta^\top \ell \leq \min_{\theta \in \Theta} \mu_\theta^\top \ell + \epsilon$. Large-scale policy design: Independent of size of $X$. (with Yasin Abbasi-Yadkori and Alan Malek. ICML2014)
Large-scale policy design

Stationary distributions dual to value functions.

Consider a class of policies defined by feature matrix $\Phi$, distribution $\mu_0$, and parameters $\theta$:

$$
\pi_\theta(a|x) = \frac{[\mu_0(x, a) + \Phi(x,a) \cdot \theta]^+}{\sum_{a'}[\mu_0(x, a') + \Phi(x,a') \cdot \theta]^+}.
$$
Large-scale policy design

- Stationary distributions dual to value functions.
- Consider a class of policies defined by **feature matrix** $\Phi$, distribution $\mu_0$, and parameters $\theta$:

$$
\pi_\theta(a|x) = \frac{[\mu_0(x,a) + \Phi(x,a)\cdot \theta]^+}{\sum_{a'}[\mu_0(x,a') + \Phi(x,a')\cdot \theta]^+}.
$$

- Let $\mu_\theta$ denote the stationary distribution of policy $\pi_\theta$. 

(with Yasin Abbasi-Yadkori and Alan Malek. ICML2014)
Linear Subspace of Stationary Distributions

Large-scale policy design

- Stationary distributions dual to value functions.
- Consider a class of policies defined by feature matrix $\Phi$, distribution $\mu_0$, and parameters $\theta$:

  $$\pi_{\theta}(a|x) = \frac{[\mu_0(x, a) + \Phi(x,a),:\theta]^+}{\sum_{a'}[\mu_0(x, a') + \Phi(x,a'),:\theta]^+}.$$

- Let $\mu_\theta$ denote the stationary distribution of policy $\pi_{\theta}$.
- Find $\hat{\theta}$ such that $\mu_{\hat{\theta}}^\top \ell \leq \min_{\theta \in \Theta} \mu_\theta^\top \ell + \epsilon$. 

(with Yasin Abbasi-Yadkori and Alan Malek. ICML2014)
Large-scale policy design (with Yasin Abbasi-Yadkori and Alan Malek. ICML2014)

- Stationary distributions dual to value functions.
- Consider a class of policies defined by feature matrix $\Phi$, distribution $\mu_0$, and parameters $\theta$:

$$
\pi_\theta(a|x) = \frac{[\mu_0(x, a) + \Phi(x,a)_\cdot \theta]^+}{\sum_{a'}[\mu_0(x, a') + \Phi(x,a')_\cdot \theta]^+}.
$$

- Let $\mu_\theta$ denote the stationary distribution of policy $\pi_\theta$.
- Find $\hat{\theta}$ such that $\mu_{\hat{\theta}}^\top \ell \leq \min_{\theta \in \Theta} \mu_\theta^\top \ell + \epsilon$.
- Large-scale policy design: Independent of size of $X$. 


Approach: a Reduction to Convex Optimization

- Define a constraint violation function

\[
V(\theta) = \left\| \left[ \mu_0 + \Phi \theta \right]_+ \right\|_1 + \left\| (P - B)^\top (\mu_0 + \Phi \theta) \right\|_1
\]

prob. dist. stationary
Define a constraint violation function

\[ V(\theta) = ||[\mu_0 + \Phi \theta]_-||_1 + \left\|(P - B)^\top(\mu_0 + \Phi \theta)\right\|_1 \]

and consider the convex cost function

\[ c(\theta) = \ell^\top(\mu_0 + \Phi \theta) + \alpha V(\theta). \]
Approach: a Reduction to Convex Optimization

- Define a constraint violation function

\[ V(\theta) = \| [\mu_0 + \Phi \theta]_- \|_1 + \| (P - B)^\top (\mu_0 + \Phi \theta) \|_1 \]

and consider the convex cost function

\[ c(\theta) = \ell^\top (\mu_0 + \Phi \theta) + \alpha V(\theta). \]

- Stochastic gradient descent: \( \theta_{t+1} = \theta_t - \eta g_t(\theta_t), \hat{\theta}_T = \sum_{t=1}^T \theta_t / T, \)
Approach: a Reduction to Convex Optimization

- Define a constraint violation function

$$V(\theta) = \|[\mu_0 + \Phi \theta]_-\|_1 + \|(P - B)^\top (\mu_0 + \Phi \theta)\|_1$$

and consider the convex cost function

$$c(\theta) = \ell^\top (\mu_0 + \Phi \theta) + \alpha V(\theta).$$

- Stochastic gradient descent: $\theta_{t+1} = \theta_t - \eta g_t(\theta_t)$, $\hat{\theta}_T = \sum_{t=1}^T \theta_t / T$,

- ... with cheap, unbiased stochastic subgradient estimates:

$$g_t(\theta) = \ell^\top \Phi - \alpha \frac{\Phi(x_t, a_t)}{q_1(x_t, a_t)} I\{\mu_0(x_t, a_t) + \Phi(x_t, a_t), \theta < 0\}$$

$$+ \alpha \frac{(P - B)^\top x_t' \Phi}{q_2(x_t')} \text{sign}((P - B)^\top x_t' \Phi \theta).$$
Main Result

For $T = 1/\epsilon^4$ gradient estimates, with high probability (under a mixing assumption),

$$\mu_{\hat{\theta}_T}^T \ell \leq \min_{\theta \in \Theta} \left( \mu_\theta^T \ell + \frac{V(\theta)}{\epsilon} + O(\epsilon) \right).$$
For $T = 1/\epsilon^4$ gradient estimates, with high probability (under a mixing assumption),

$$\mu_{\hat{\theta}_T}^T \ell \leq \min_{\theta \in \Theta} \left( \mu_\theta^T \ell + \frac{V(\theta)}{\epsilon} + O(\epsilon) \right).$$

- Competitive with all policies (stationary distributions) in the linear subspace (i.e., $V(\theta) = 0$).
Main Result

For $T = 1/\epsilon^4$ gradient estimates, with high probability (under a mixing assumption),

$$\mu_{\hat{\theta}_T}^T \ell \leq \min_{\theta \in \Theta} \left( \mu_{\theta}^T \ell + \frac{V(\theta)}{\epsilon} + O(\epsilon) \right).$$

- Competitive with all policies (stationary distributions) in the linear subspace (i.e., $V(\theta) = 0$).
- Competitive with other policies; comparison more favorable near some stationary distribution in the subspace.
Performance Bounds

Main Result

For $T = 1/\epsilon^4$ gradient estimates, with high probability (under a mixing assumption),

$$\mu_{\hat{\theta}_T}^\top \ell \leq \min_{\theta \in \Theta} \left( \mu_\theta^\top \ell + \frac{V(\theta)}{\epsilon} + O(\epsilon) \right).$$

- Competitive with all policies (stationary distributions) in the linear subspace (i.e., $V(\theta) = 0$).
- Competitive with other policies; comparison more favorable near some stationary distribution in the subspace.
- Previous results of this kind:
  - require knowledge about optimal policy, or
  - require that the comparison class $\Pi$ contains a near-optimal policy.
Simulation Results: Queueing

(Rybko and Stolyar, 1992; de Farias and Van Roy, 2003a)
Simulation Results: Queueing

(Rybko and Stolyar, 1992; de Farias and Van Roy, 2003a)
1. Large-Scale Policy Design

- Compete with a restricted family of policies $\Pi$:
  - Linearly parameterized approximate stationary distributions.
1. Large-Scale Policy Design

- Compete with a restricted family of policies $\Pi$:
  - Linearly parameterized approximate stationary distributions.
  - Linearly parameterized exponentially transformed value function.
1. Large-Scale Policy Design

- Compete with a restricted family of policies $\Pi$:
  - Linearly parameterized approximate stationary distributions.
  - Linearly parameterized exponentially transformed value function.
- Stochastic gradient convex optimization.
- Competitive with policies in the approximating class.
- Simulation results: crowdsourcing.
Consider total cost:

\[ E_{\infty} \sum_{t=1}^{\infty} \ell(X_t) \].

Parameterized value functions, close to a reference policy \( P_0 \):

\[ X \rightarrow \Delta(X) \].

Regularize with KL-divergence to \( P_0 \):

\[ \ell(x, P) = \ell(x) + d_{KL}(P(\cdot|x), P_0(\cdot|x)) \].

\( P \) is transition matrix under policy.
Large-scale policy design

Consider total cost:

\[ \mathbb{E} \sum_{t=1}^{\infty} \ell(X_t). \]

(assume a.s. hit absorbing state with zero loss)
Large-scale policy design

Consider total cost:

\[ \mathbb{E} \sum_{t=1}^{\infty} \ell(X_t). \]

Parameterized value functions, close to a reference policy

\( P_0 : \mathcal{X} \rightarrow \Delta(\mathcal{X}) \).
Total Cost, Kullback-Leibler Penalty

Large-scale policy design

- Consider total cost: (assume a.s. hit absorbing state with zero loss)

\[
\mathbb{E} \sum_{t=1}^{\infty} \ell(X_t).
\]

- Parameterized value functions, close to a reference policy

\[
P_0 : \mathcal{X} \rightarrow \Delta(\mathcal{X}).
\]

- Regularize with KL-divergence to \( P_0 \): (so optimization is linear; Todorov/Kappen/Fleming)

\[
\ell(x, P) = \ell(x) + d_{KL}(P(\cdot|x), P_0(\cdot|x)).
\]

\( P \) is transition matrix under policy.
Total Cost, Kullback-Leibler Penalty

Large-scale policy design

Consider a class of policies defined by feature matrix $\Phi$, and parameters $\theta$:

$$\Pi = \left\{ G\hat{J}_\theta : \theta \in \Theta \right\}$$

$$G\hat{J}(x) := \arg\min_{\pi} \left( \ell(x, P^\pi) + \mathbb{E}^\pi \left[ \hat{J}(x') | x \right] \right)$$

Greedy policies

$$\hat{J}_\theta = -\log(\Phi\theta).$$

Log linear

Large-scale policy design:
Independent of size of $X$. 

(with Yasin Abbasi-Yadkori, Xi Chen and Alan Malek)
**Large-scale policy design**

Consider a class of policies defined by **feature matrix** \( \Phi \), and parameters \( \theta \):

\[
\Pi = \{ G\hat{J}_\theta : \theta \in \Theta \}
\]

\[
G\hat{J}(x) := \arg \min_\pi \left( \ell(x, P^\pi) + \mathbb{E}^\pi \left[ \hat{J}(x') | x \right] \right)
\]

\[
\hat{J}_\theta = -\log(\Phi \theta).
\]

Greedy policies

Log linear

Find parameters \( \hat{\theta} \) (hence policy \( \hat{\pi} = G\hat{J}_{\hat{\theta}} \)) such that

\[
J_{\hat{\pi}}(x_1) \leq \min_{\pi \in \Pi} J_\pi(x_1) + \epsilon
\]

\[
J_\pi(x) := \mathbb{E}^\pi \left[ \sum_{t=1}^{\infty} \ell(X_t) \middle| X_1 = x \right]
\]
Large-scale policy design

Consider a class of policies defined by feature matrix $\Phi$, and parameters $\theta$:

$$\Pi = \left\{ G\hat{J}_\theta : \theta \in \Theta \right\}$$

$G\hat{J}(x) := \arg\min_\pi \left( \ell(x, P^\pi) + \mathbb{E}^\pi \left[ \hat{J}(x') | x \right] \right)$  

$\hat{J}_\theta = -\log(\Phi\theta)$.  

Find parameters $\hat{\theta}$ (hence policy $\hat{\pi} = G\hat{J}_{\hat{\theta}}$) such that

$$J_{\hat{\pi}}(x_1) \leq \min_{\pi \in \Pi} J_{\pi}(x_1) + \epsilon$$

$$J_{\pi}(x) := \mathbb{E}^\pi \left[ \sum_{t=1}^{\infty} \ell(X_t) \middle| X_1 = x \right]$$

Large-scale policy design: Independent of size of $\mathcal{X}$. 
Define a transformed Bellman error function

\[ V(\theta) = \| \Phi \theta - \exp(-\ell(x))P_0 \Phi \theta \| \]

convex in \( \theta \)
Define a transformed Bellman error function

\[ V(\theta) = \| \Phi \theta - \exp(-\ell(x)) P_0 \Phi \theta \| = \| \exp(-\hat{J}_\theta) - \exp(-T \hat{J}_\theta) \| \]

\[ T \hat{J}(x) := \min_\pi \left( \ell(x, P^\pi) + \mathbb{E}^{\pi} \left[ \hat{J}(x') | x \right] \right) \quad \text{(dynamic prog operator)} \]
Define a transformed Bellman error function

\[ V(\theta) = \| \Phi \theta - \exp(-\ell(x)) P_0 \Phi \| = \| \exp(-\hat{J}_\theta) - \exp(-T \hat{J}_\theta) \| \]

\[ T \hat{J}(x) := \min_{\pi} \left( \ell(x, P^\pi) + \mathbb{E}^{\pi} \left[ \hat{J}(x') \right| x \right) \] (dynamic prog operator)

and consider the convex cost function

\[ c(\theta) = \hat{J}_\theta + \alpha V(\theta). \]
Define a transformed Bellman error function

\[ V(\theta) = \| \Phi \theta - \exp(-\ell(x)) P_0 \Phi \| = \left\| \exp(-\hat{J}_\theta) - \exp\left(-T \hat{J}_\theta\right) \right\| \]

\[ T \hat{J}(x) := \min_{\pi} \left( \ell(x, P^\pi) + \mathbb{E}^\pi \left[ \hat{J}(x') | x \right] \right) \quad \text{(dynamic prog operator)} \]

and consider the convex cost function

\[ c(\theta) = \hat{J}_\theta + \alpha V(\theta). \]

Stochastic gradient descent
Approach: a Reduction to Convex Optimization

- Define a transformed Bellman error function

\[ V(\theta) = \| \Phi \theta - \exp(-\ell(x)) P_0 \Phi \theta \| = \| \exp(-\hat{J}_\theta) - \exp(-T\hat{J}_\theta) \| \]

\[ T\hat{J}(x) := \min_{\pi} \left( \ell(x, P^\pi) + \mathbb{E}^\pi \left[ \hat{J}(x')|x \right] \right) \]

(dynamic prog operator)

and consider the convex cost function

\[ c(\theta) = \hat{J}_\theta + \alpha V(\theta). \]

- Stochastic gradient descent

- ... with cheap, unbiased stochastic subgradient estimates.
Performance Bounds

Main Result

For \( T = 1/\epsilon^4 \) gradient estimates, with high probability,

\[
J_{\hat{\pi}}(x_1) \leq \min_{\pi \in \Pi} \left( J_\pi(x_1) + \frac{1}{\epsilon} \| J_\theta - T \hat{J}_\theta \| + \| \hat{J}_\theta - T \hat{J}_\hat{\theta} \|' + O(\epsilon) \right).
\]
Main Result

For $T = 1/\epsilon^4$ gradient estimates, with high probability,

$$J_{\hat{\pi}}(x_1) \leq \min_{\pi \in \Pi} \left( J_\pi(x_1) + \frac{1}{\epsilon} \| \hat{J}_\theta - T \hat{J}_\theta \| + \| \hat{J}_\hat{\theta} - T \hat{J}_{\hat{\theta}} \|' + O(\epsilon) \right).$$

- Competitive with all policies in the parameterized class, up to penalties involving the Bellman errors.
Main Result

For $T = 1/\epsilon^4$ gradient estimates, with high probability,

$$J_{\hat{\pi}}(x_1) \leq \min_{\pi \in \Pi} \left( J_\pi(x_1) + \frac{1}{\epsilon} \| \hat{J}_\theta - T \hat{J}_\theta \| \right) + \| \hat{J}_\theta - T \hat{J}_\theta \|' + O(\epsilon).$$

- Competitive with all policies in the parameterized class, up to penalties involving the Bellman errors.
- Unfortunately:
  - require that the comparison class $\Pi$ contains a near-optimal policy.
Simulation Results: Crowdsourcing

- Classification task.
Simulation Results: Crowdsourcing

- Classification task.
- Crowdsore labels.
Simulation Results: Crowdsourcing

- Classification task.
- Crowdsource: $ for labels.
- Fixed budget; minimize errors.
Classification task.
Crowdsourcing: $ for labels.
Fixed budget; minimize errors.
Bayesian model: binary labels, i.i.d. crowd;
\( Y_i \sim \text{Bernoulli}(p_i) \)
Simulation Results: Crowdsourcing

- Classification task.
- Crowdsource: $ for labels.
- Fixed budget; minimize errors.
- Bayesian model: binary labels, i.i.d. crowd; $Y_i \sim \text{Bernoulli}(p_i)$
Simulation Results: Crowdsourcing

- Classification task.
- Crowdsource: $ for labels.
- Fixed budget; minimize errors.
- Bayesian model: binary labels, i.i.d. crowd; \( Y_i \sim \text{Bernoulli}(p_i) \)
Simulation Results: Crowdsourcing

- Classification task.
- Crowdsource: $ for labels.
- Fixed budget; minimize errors.
- Bayesian model:
  - binary labels, i.i.d. crowd;
  - $Y_i \sim \text{Bernoulli}(p_i)$;
  - $p_i \sim \text{Beta}$. 
Simulation Results: Crowdsourcing

- Classification task.
- Crowdsource: $ for labels.
- Fixed budget; minimize errors.
- Bayesian model: binary labels, i.i.d. crowd; $Y_i \sim \text{Bernoulli}(p_i);$ $p_i \sim \text{Beta}.$
- State = posterior.
Simulation Results: Crowdsourcing

- Classification task.
- Crowdsource: $ for labels.
- Fixed budget; minimize errors.
- Bayesian model:
  - binary labels, i.i.d. crowd; $Y_i \sim \text{Bernoulli}(p_i)$;
  - $p_i \sim \text{Beta}$.
- State = posterior.

![Graph showing posterior classification error vs. budget (B)]
1. Large-Scale Policy Design
   - Compete with a restricted family of policies $\Pi$:
     - Linearly parameterized policies.
     - Stochastic gradient convex optimization.
     - Competitive with policies near the approximating class.
     - Without knowledge of optimal policy.
   - Simulation results: queueing, crowdsourcing.

2. Learning Changing Dynamics
1. Large-Scale Policy Design

- Compete with a restricted family of policies $\Pi$:
  - Linearly parameterized policies.
  - Stochastic gradient convex optimization.
  - Competitive with policies near the approximating class.
  - Without knowledge of optimal policy.
  - Simulation results: queueing, crowdsourcing.

2. Learning Changing Dynamics

- Changing MDP; complete information.
1. Large-Scale Policy Design

- Compete with a restricted family of policies $\Pi$:
  - Linearly parameterized policies.
  - Stochastic gradient convex optimization.
  - Competitive with policies near the approximating class.
  - Without knowledge of optimal policy.
  - Simulation results: queueing, crowdsourcing.

2. Learning Changing Dynamics

- Changing MDP; complete information.
  - Exponential weights strategy.
Outline

1. Large-Scale Policy Design
   - Compete with a restricted family of policies $\Pi$:
     - Linearly parameterized policies.
   - Stochastic gradient convex optimization.
   - Competitive with policies near the approximating class.
   - Without knowledge of optimal policy.
   - Simulation results: queueing, crowdsourcing.

2. Learning Changing Dynamics
   - Changing MDP; complete information.
   - Exponential weights strategy.
   - Competitive with small comparison class $\Pi$. 
1. Large-Scale Policy Design
- Compete with a restricted family of policies $\Pi$:
  - Linearly parameterized policies.
  - Stochastic gradient convex optimization.
  - Competitive with policies near the approximating class.
  - Without knowledge of optimal policy.
- Simulation results: queueing, crowdsourcing.

2. Learning Changing Dynamics
- Changing MDP; complete information.
- Exponential weights strategy.
- Competitive with small comparison class $\Pi$.
- Computationally efficient if $\Pi$ has polynomial size.
Outline

1. Large-Scale Policy Design
   - Compete with a restricted family of policies $\Pi$:
     - Linearly parameterized policies.
   - Stochastic gradient convex optimization.
   - Competitive with policies near the approximating class.
   - Without knowledge of optimal policy.
   - Simulation results: queueing, crowdsourcing.

2. Learning Changing Dynamics
   - Changing MDP; complete information.
   - Exponential weights strategy.
   - Competitive with small comparison class $\Pi$.
   - Computationally efficient if $\Pi$ has polynomial size.
   - Hard for shortest path problems.
Observe $P_t$, $\ell_t$ after round $t$. 

(With Yasin Abbasi-Yadkori, Varun Kanade, Yevgeny Seldin, Csaba Szepesvari, NIPS2013)
Observe $P_t, \ell_t$ after round $t$.

Consider a comparison class: $\Pi \subset \{\pi | \pi : \mathcal{X} \rightarrow \mathcal{A}\}$
Observe $P_t, \ell_t$ after round $t$.

Consider a comparison class: $\Pi \subset \{\pi | \pi : \mathcal{X} \rightarrow \mathcal{A}\}$

$\pi^* = \arg\min_{\pi \in \Pi} \sum_{t=1}^{T} \ell_t(x_t^\pi, \pi(x_t^\pi))$
Observe $P_t, \ell_t$ after round $t$.

Consider a comparison class: $\Pi \subset \{\pi \mid \pi : \mathcal{X} \to \mathcal{A}\}$

$\pi^* = \argmin_{\pi \in \Pi} \sum_{t=1}^{T} \ell_t(x_t^\pi, \pi(x_t^\pi))$

$R_T = \sum_{t=1}^{T} \ell_t(x_t, a_t) - \sum_{t=1}^{T} \ell_t(x_t^{\pi^*}, \pi^*(x_t^{\pi^*}))$
Observe $P_t, \ell_t$ after round $t$.

Consider a comparison class: $\Pi \subset \{\pi | \pi : X \rightarrow A\}$

$\pi^* = \text{argmin}_{\pi \in \Pi} \sum_{t=1}^{T} \ell_t(x_t^\pi, \pi(x_t^\pi))$

$R_T = \sum_{t=1}^{T} \ell_t(x_t, a_t) - \sum_{t=1}^{T} \ell_t(x_t^{\pi^*}, \pi^*(x_t^{\pi^*}))$

Aim for low regret: $R_T / T \rightarrow 0$
Observe $P_t, \ell_t$ after round $t$.

Consider a comparison class: $\Pi \subset \{\pi \mid \pi : X \to A\}$

$$\pi^* = \arg\min_{\pi \in \Pi} \sum_{t=1}^{T} \ell_t(x^\pi_t, \pi(x^\pi_t))$$

$$R_T = \sum_{t=1}^{T} \ell_t(x_t, a_t) - \sum_{t=1}^{T} \ell_t(x^\pi^*_t, \pi^*(x^\pi^*_t))$$

Aim for low regret: $R_T/T \to 0$

Computationally efficient low regret strategies?
Main Result

There is a strategy that (under a $\tau$-mixing assumption) achieves

$$\mathbb{E} [R_T] \leq (4 + 2\tau^2) \sqrt{T \log |\Pi|} + \log |\Pi|.$$
### Exponential weights:

#### Strategy for a repeated game:

Choose action $a \in A$ with probability proportional to

$$\exp(\text{total loss } a \text{ has incurred so far}).$$

Regret (total loss versus best in hindsight) for $T$ rounds:

$$O(\sqrt{T \log |A|}).$$

Long history.

Unreasonably broadly applicable:
- Zero-sum games.
- AdaBoost.
- Bandit problems.
- Linear programming.
- Shortest path problems.
- Fast max-flow.
- Fast graph sparsification.
- Model of evolution.
Exponential weights:

Strategy for a repeated game:
Choose action $a \in A$ with probability proportional to

$$\exp(\text{total loss } a \text{ has incurred so far}).$$

- Regret (total loss versus best in hindsight) for $T$ rounds:

$$O\left(\sqrt{T \log |A|}\right).$$
Strategy for a repeated game:
Choose action $a \in A$ with probability proportional to

$$\exp(\text{total loss } a \text{ has incurred so far}).$$

- Regret (total loss versus best in hindsight) for $T$ rounds:
  $$O\left(\sqrt{T \log |A|}\right).$$

- Long history.
Exponential weights:

Strategy for a repeated game:
Choose action \( a \in A \) with probability proportional to
\[
\exp(\text{total loss } a \text{ has incurred so far}).
\]

Regret (total loss versus best in hindsight) for \( T \) rounds:
\[
O\left(\sqrt{T \log |A|}\right).
\]

Long history.

Unreasonably broadly applicable:
- Zero-sum games.
- AdaBoost.
- Bandit problems.
- Linear programming.
- Shortest path problems.
- Fast max-flow.
- Fast graph sparsification.
- Model of evolution.
Strategy:

For all policies $\pi \in \Pi$, $w_{\pi,0} = 1$.

$W_t = \sum_{\pi \in \Pi} w_{\pi,t}$, $p_{\pi,t} = w_{\pi,t-1}/W_{t-1}$.

for $t := 1, 2, \ldots$ do

w.p. $\beta_t = \frac{w_{\pi_{t-1},t-1}}{w_{\pi_{t-1},t-2}}$, $\pi_t = \pi_{t-1}$. Otherwise $\pi_t \sim p_{.,t}$.

Choose action $a_t \sim \pi_t(.|x_t)$.

Observe dynamics $P_t$ and loss $\ell_t$.

Suffer $\ell_t(x_t, a_t)$.

For all policies $\pi$, $w_{\pi,t} = w_{\pi,t-1} \exp (-\eta \mathbb{E} [\ell_t(x_t^\pi, \pi)])$.

end for

- Exponential weights on $\Pi$. 
Strategy:

For all policies $\pi \in \Pi$, $w_{\pi, 0} = 1$.

$$W_t = \sum_{\pi \in \Pi} w_{\pi, t}, \quad p_{\pi, t} = w_{\pi, t-1} / W_{t-1}.$$  

for $t := 1, 2, \ldots$ do  

w.p. $\beta_t = \frac{w_{\pi_{t-1}, t-1}}{w_{\pi_{t-1}, t-2}}$, $\pi_t = \pi_{t-1}$. Otherwise $\pi_t \sim p_{., t}$.  

Choose action $a_t \sim \pi_t(. | x_t)$.  

Observe dynamics $P_t$ and loss $\ell_t$.  

Suffer $\ell_t(x_t, a_t)$.  

For all policies $\pi$, $w_{\pi, t} = w_{\pi, t-1} \exp (-\eta \mathbb{E} [\ell_t(x_t^\pi, \pi)])$.  

end for

- Exponential weights on $\Pi$.

- Rare, random changes to $\pi_t$.  

Main Result

There is a strategy that (under a $\tau$-mixing assumption) achieves

$$\mathbb{E} [R_T] \leq (4 + 2\tau^2) \sqrt{T \log |\Pi|} + \log |\Pi|.$$

- Adversarial dynamics and loss functions.
Main Result

There is a strategy that (under a $\tau$-mixing assumption) achieves

$$\mathbb{E}[R_T] \leq (4 + 2\tau^2) \sqrt{T \log |\Pi|} + \log |\Pi|.$$  

- Adversarial dynamics and loss functions.
- Large state and action spaces.
Regret Bound

Main Result

There is a strategy that (under a $\tau$-mixing assumption) achieves

$$\mathbb{E} [R_T] \leq (4 + 2\tau^2) \sqrt{T \log |\Pi|} + \log |\Pi|.$$ 

- Adversarial dynamics and loss functions.
- Large state and action spaces.
- $\mathbb{E} [R_T] / T \rightarrow 0$ for $T = \omega(\log |\Pi|)$. 
Regret Bound

Main Result

There is a strategy that (under a $\tau$-mixing assumption) achieves

$$\mathbb{E} [R_T] \leq (4 + 2\tau^2) \sqrt{T \log |\Pi|} + \log |\Pi|.$$ 

- Adversarial dynamics and loss functions.
- Large state and action spaces.
- $\mathbb{E} [R_T] / T \to 0$ for $T = \omega(\log |\Pi|)$.
- Computationally efficient as long as $|\Pi|$ is polynomial.
Main Result

There is a strategy that (under a $\tau$-mixing assumption) achieves

$$E[R_T] \leq (4 + 2\tau^2)\sqrt{T \log |\Pi|} + \log |\Pi|.$$ 

- Adversarial dynamics and loss functions.
- Large state and action spaces.
- $E[R_T]/T \to 0$ for $T = \omega(\log |\Pi|)$.
- Computationally efficient as long as $|\Pi|$ is polynomial.
- No computationally efficient algorithm in general
Shortest Path Problem

Special case of MDP: node=state; action=edge; loss=weight.
Shortest Path Problem

Special case of MDP: node=state; action=edge; loss=weight.
Shortest Path Problem

Special case of MDP: node=state; action=edge; loss=weight.
Computational Efficiency

Hardness Result

Suppose there is a strategy for the online adversarial shortest path problem that:

1. runs in time $\text{poly}(n, T)$, and
2. has regret $R_T = O(\text{poly}(n) T^{1-\delta})$ for some constant $\delta > 0$.

Then there is an efficient algorithm for online agnostic parity learning with sublinear regret.
Online Agnostic Parity Learning

- Class of parity functions on \( \{0, 1\}^n \):
  \[
  \text{PARITIES} = \{ \text{PAR}_S \mid S \subset [n], \text{PAR}_S(x) = \bigoplus_{i \in S} x_i \}
  \]
Online Agnostic Parity Learning

- Class of parity functions on \(\{0, 1\}^n\):
  \[
  \text{PARITIES} = \{\text{PAR}_S \mid S \subseteq [n], \text{PAR}_S(x) = \bigoplus_{i \in S} x_i\}
  \]

- Learning problem: given \(x_t \in \{0, 1\}^n\), learner predicts \(\hat{y}_t \in \{0, 1\}\), observes the true label \(y_t\) and suffers loss \(\mathbb{II}_{\hat{y}_t \neq y_t}\).
Online Agnostic Parity Learning

- Class of parity functions on \( \{0, 1\}^n \):
  \[
  \text{PARITIES} = \{ \text{PAR}_S \mid S \subset [n], \text{PAR}_S(x) = \bigoplus_{i \in S} x_i \}
  \]

- Learning problem: given \( x_t \in \{0, 1\}^n \), learner predicts \( \hat{y}_t \in \{0, 1\} \), observes the true label \( y_t \) and suffers loss \( \mathbb{I}_{\{\hat{y}_t \neq y_t\}} \)

- \( R_T = \sum_{t=1}^{T} \mathbb{I}_{\{\hat{y}_t \neq y_t\}} - \min_{\text{PAR}_S \in \text{PARITIES}} \sum_{t=1}^{T} \mathbb{I}_{\{\text{PAR}_S(x_t) \neq y_t\}} \)
Class of parity functions on $\{0, 1\}^n$:

$$\text{PARITIES} = \{\text{PAR}_S \mid S \subset [n], \text{PAR}_S(x) = \bigoplus_{i \in S} x_i\}$$

Learning problem: given $x_t \in \{0, 1\}^n$, learner predicts $\hat{y}_t \in \{0, 1\}$, observes the true label $y_t$ and suffers loss $I\{\hat{y}_t \neq y_t\}$

$$R_T = \sum_{t=1}^{T} I\{\hat{y}_t \neq y_t\} - \min_{\text{PAR}_S \in \text{PARITIES}} \sum_{t=1}^{T} I\{\text{PAR}_S(x_t) \neq y_t\}$$

Is there an efficient (time polynomial in $n$, $T$) learning algorithm with sublinear regret ($R_T = O(\text{poly}(n)T^{1-\delta})$ for some $\delta > 0$)?
Online Agnostic Parity Learning

- Class of parity functions on \( \{0, 1\}^n \):
  \[
  \text{PARITIES} = \{ \text{PAR}_S \mid S \subset [n], \text{PAR}_S(x) = \bigoplus_{i \in S} x_i \} 
  \]

- Learning problem: given \( x_t \in \{0, 1\}^n \), learner predicts \( \hat{y}_t \in \{0, 1\} \), observes the true label \( y_t \) and suffers loss \( \mathbb{I}_{\{\hat{y}_t \neq y_t\}} \)

- \( R_T = \sum_{t=1}^{T} \mathbb{I}_{\{\hat{y}_t \neq y_t\}} - \min_{\text{PAR}_S \in \text{PARITIES}} \sum_{t=1}^{T} \mathbb{I}_{\{\text{PAR}_S(x_t) \neq y_t\}} \)

- Is there an efficient (time polynomial in \( n, T \)) learning algorithm with sublinear regret (\( R_T = O(\text{poly}(n) T^{1-\delta}) \) for some \( \delta > 0 \))?

- Very well-studied.
Online Agnostic Parity Learning

- Class of parity functions on $\{0, 1\}^n$:
  \[ \text{PARITIES} = \{ \text{PAR}_S \mid S \subset [n], \text{PAR}_S(x) = \bigoplus_{i \in S} x_i \} \]

- Learning problem: given $x_t \in \{0, 1\}^n$, learner predicts $\hat{y}_t \in \{0, 1\}$, observes the true label $y_t$ and suffers loss $\mathbb{I}_{\{\hat{y}_t \neq y_t\}}$

- $R_T = \sum_{t=1}^T \mathbb{I}_{\{\hat{y}_t \neq y_t\}} - \min_{\text{PAR}_S \in \text{PARITIES}} \sum_{t=1}^T \mathbb{I}_{\{\text{PAR}_S(x_t) \neq y_t\}}$

- Is there an efficient (time polynomial in $n, T$) learning algorithm with sublinear regret ($R_T = O(\text{poly}(n) T^{1-\delta})$ for some $\delta > 0$)?

- Very well-studied.

- Widely believed to be hard: used for cryptographic schemes.
Computational Efficiency

Hardness Result

Suppose there is a strategy for the online adversarial shortest path problem that:

1. runs in time $\text{poly}(n, T)$, and
2. has regret $R_T = O(\text{poly}(n) T^{1-\delta})$ for some constant $\delta > 0$.

Then there is an efficient algorithm for online agnostic parity learning with sublinear regret.
Reduction

Adversary \((x, y)\) → Conversion1 → Shortest Path Algorithm → Conversion2 → Path

\[ \hat{y}_t = 0 \] \[ \hat{y}_t = 1 \]

\[ x = (1, 0, 1, 0, 1) \in \{0, 1\}^5 \]
## Online shortest path: Hard versus easy

<table>
<thead>
<tr>
<th>Edges (dynamics)</th>
<th>Weights (costs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adversarial</td>
<td>Adversarial</td>
</tr>
<tr>
<td>Stochastic</td>
<td>Adversarial</td>
</tr>
<tr>
<td>Adversarial</td>
<td>Stochastic</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>As hard as noisy parity.</td>
</tr>
<tr>
<td></td>
<td>Efficient algorithm.</td>
</tr>
<tr>
<td></td>
<td>Efficient algorithm.</td>
</tr>
</tbody>
</table>
Outline

1. Large-Scale Policy Design
   - Compete with a restricted family of policies $\Pi$:
     - Linearly parameterized policies.
   - Stochastic gradient convex optimization.
   - Competitive with policies near the approximating class.
   - Without knowledge of optimal policy.
   - Simulation results: queueing, crowdsourcing.

2. Learning changing dynamics
   - Changing MDP; complete information.
   - Exponential weights strategy.
   - Competitive with small comparison class $\Pi$.
   - Computationally efficient if $\Pi$ has polynomial size.
   - Hard for shortest path problems.