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

Markov Decision Processes

MDP: Managing Threatened Species

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

Markov Decision Processes

MDP: Managing Threatened Species

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

1. See state X_t of ecosystem

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

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

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 $\ell(X_t, A_t)$

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

Markov Decision Processes

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

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:

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

Markov Decision Processes

MDP: Managing Threatened Species

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

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:

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

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

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:

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

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

Stationary distribution: μ

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:

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

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

Stationary distribution: μ

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-Scale Sequential Decision Problems

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 Π.
  e.g., linear value function approximation.
  Want a strategy that competes with the best policy.

Outline

1. Large-Scale Policy Design

2. Learning Changing Dynamics

Outline

1. Large-Scale Policy Design

- Compete with a restricted family of policies Π:
  Linearly parameterized approximate stationary distributions.

2. Learning Changing Dynamics

Outline

1. Large-Scale Policy Design

- Compete with a restricted family of policies Π:
  Linearly parameterized exponentially transformed value function.

2. Learning Changing Dynamics

Outline

1. Large-Scale Policy Design

- Compete with a restricted family of policies Π:
  Linearly parameterized policies.
- Stochastic gradient convex optimization.

2. Learning Changing Dynamics

Outline

1. Large-Scale Policy Design

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

2. Learning Changing Dynamics

Outline

1. Large-Scale Policy Design

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

2. Learning Changing Dynamics

Outline

1. Large-Scale Policy Design

- Compete with a restricted family of policies Π:
  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 Π.
- Computationally efficient if Π has polynomial size.
- Hard for shortest path problems.

Linear Subspace of Stationary Distributions

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

- Stationary distributions dual to value functions.

Linear Subspace of Stationary Distributions

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** Φ, distribution μ_0, and parameters θ:

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

Linear Subspace of Stationary Distributions

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** Φ, distribution μ_0, and parameters θ:

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

- Let μ_θ denote the stationary distribution of policy π_θ.

Linear Subspace of Stationary Distributions

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** Φ, distribution μ_0, and parameters θ:

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

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

Linear Subspace of Stationary Distributions

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** Φ, distribution μ_0, and parameters θ:

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

- Let μ_θ denote the stationary distribution of policy π_θ.
- Find $\widehat{\theta}$ such that $\mu_{\widehat{\theta}}^\top \ell \le \min_{\theta\in\Theta} \mu_\theta^\top \ell + \epsilon$.
- **Large-scale policy design:** Independent of size of $\mathcal{X}$.

Approach: a Reduction to Convex Optimization

- Define a constraint violation function

$$V(\theta) = \underbrace{\left\| [\mu_0 + \Phi\theta]_- \right\|_1}_{\text{prob. dist.}} + \underbrace{\left\| (P - B)^\top (\mu_0 + \Phi\theta) \right\|_1}_{\text{stationary}}$$

Approach: a Reduction to Convex Optimization

- Define a constraint violation function

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

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

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

  and consider the convex cost function
  $c(\theta) = \ell^\top(\mu_0 + \Phi\theta) + \alpha V(\theta)$.
- Stochastic gradient descent: $\theta_{t+1} = \theta_t - \eta g_t(\theta_t)$, $\widehat{\theta}_T = \sum_{t=1}^T \theta_t / T$,

Approach: a Reduction to Convex Optimization

- Define a constraint violation function

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

  and consider the convex cost function
  $c(\theta) = \ell^\top(\mu_0 + \Phi\theta) + \alpha V(\theta)$.
- Stochastic gradient descent: $\theta_{t+1} = \theta_t - \eta g_t(\theta_t)$, $\widehat{\theta}_T = \sum_{t=1}^T \theta_t / T$,
- ... with cheap, unbiased stochastic subgradient estimates:

$$g_t(\theta) = \ell^\top\Phi - \alpha \frac{\Phi_{(x_t,a_t),:}}{q_1(x_t,a_t)} \mathbb{I}_{\{\mu_0(x_t,a_t) + \Phi_{(x_t,a_t),:}\theta < 0\}} + \alpha \frac{(P-B)^\top_{:,x'_t}\Phi}{q_2(x'_t)} \operatorname{sign}((P-B)^\top_{:,x'_t}\Phi\theta).$$

Performance Bounds

Main Result

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

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

Performance Bounds

Main Result

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

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

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

Performance Bounds

Main Result

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

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

- 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.

Performance Bounds

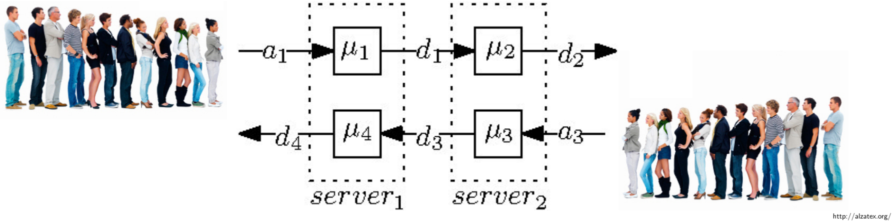
Main Result

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

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

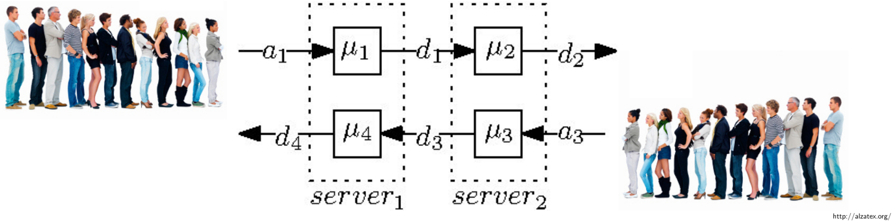
- Competitive with all policies (stationary distributions) in the linear subspace (i.e., $V(\theta) = 0$).
- Competitive with other policies; comparison more favorable near some stationary distribution in the subspace.
- Previous results of this kind:
  - require knowledge about optimal policy, or
 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

Simulation Results: Queueing

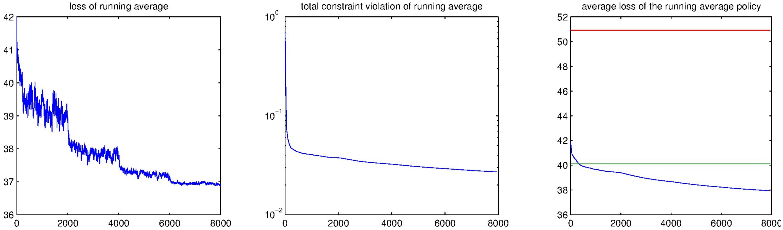


(Rybko and Stolyar, 1992; de Farias and Van Roy, 2003a)

Simulation Results: Queueing



(Rybko and Stolyar, 1992; de Farias and Van Roy, 2003a)



Outline

1. Large-Scale Policy Design

- Compete with a restricted family of policies Π:
  Linearly parameterized approximate stationary distributions.

Outline

1. Large-Scale Policy Design

- Compete with a restricted family of policies Π:
  Linearly parameterized approximate stationary distributions.
  Linearly parameterized exponentially transformed value function.

Outline

1. Large-Scale Policy Design

- Compete with a restricted family of policies Π:
  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.

Total Cost, Kullback-Leibler Penalty

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

Total Cost, Kullback-Leibler Penalty

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

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

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

Total Cost, Kullback-Leibler Penalty

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

- 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*

$$P_0 : \mathcal{X} \to \Delta(\mathcal{X}).$$

Total Cost, Kullback-Leibler Penalty

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

- 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*

$$P_0 : \mathcal{X} \to \Delta(\mathcal{X}).$$

- 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 (with Yasin Abbasi-Yadkori, Xi Chen and Alan Malek)

- Consider a class of policies defined by **feature matrix** Φ, and parameters θ:

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

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

$$\widehat{J}_\theta = -\log(\Phi\theta). \qquad \text{log linear}$$

Total Cost, Kullback-Leibler Penalty

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

- Consider a class of policies defined by **feature matrix** Φ, and parameters θ:

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

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

$$\widehat{J}_\theta = -\log(\Phi\theta). \qquad \text{log linear}$$

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

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

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

Total Cost, Kullback-Leibler Penalty

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

- Consider a class of policies defined by **feature matrix** Φ, and parameters θ:

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

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

$$\widehat{J}_\theta = -\log(\Phi\theta). \qquad \text{log linear}$$

- Find parameters $\widehat{\theta}$ (hence policy $\hat{\pi} = G\widehat{J}_{\hat{\theta}}$) 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 $\mathcal{X}$.

Approach: a Reduction to Convex Optimization

- Define a transformed Bellman error function

$$V(\theta) = \underbrace{\|\Phi\theta - \exp(-\ell(x))P_0\Phi\theta\|}_{\text{convex in } \theta}$$

Approach: a Reduction to Convex Optimization

- Define a transformed Bellman error function (‖ · ‖ is a 1-norm over trajectories)

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

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

Approach: a Reduction to Convex Optimization

- Define a transformed Bellman error function ($\|\cdot\|$ is a 1-norm over trajectories)

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

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

and consider the convex cost function
$c(\theta) = \widehat{J}_\theta + \alpha V(\theta)$.

Approach: a Reduction to Convex Optimization

- Define a transformed Bellman error function (‖ · ‖ is a 1-norm over trajectories)

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

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

  and consider the convex cost function
  $c(\theta) = \widehat{J}_\theta + \alpha V(\theta)$.
- Stochastic gradient descent

Approach: a Reduction to Convex Optimization

- Define a transformed Bellman error function (‖ · ‖ is a 1-norm over trajectories)

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

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

  and consider the convex cost function
  $c(\theta) = \widehat{J}_\theta + \alpha V(\theta)$.
- Stochastic gradient descent
- ... with cheap, unbiased stochastic subgradient estimates.

Performance Bounds

Main Result

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

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

Performance Bounds

Main Result

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

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

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

Performance Bounds

Main Result

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

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

- Competitive with all policies in the parameterized class, up to penalties involving the Bellman errors.
- Unfortunately:
  - require that the comparison class Π contains a near-optimal policy.

Simulation Results: Crowdsourcing

- Classification task.

http://www.technicalinfo.net/

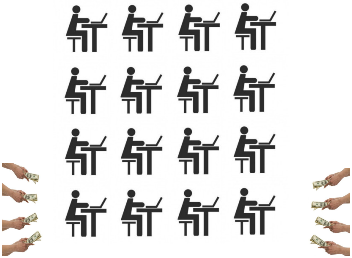
Simulation Results: Crowdsourcing

- Classification task.
- Crowdsource labels.

http://www.technicalinfo.net/
http://cdns2.freepik.com/

Simulation Results: Crowdsourcing

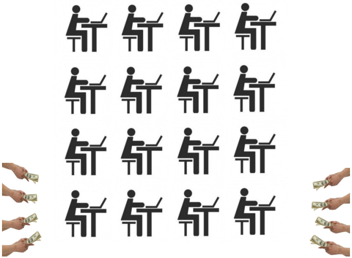
- Classification task.
- Crowdsource: $ for labels.
- Fixed budget;
  minimize errors.



http://www.technicalinfo.net/
http://cdns2.freepik.com/

Simulation Results: Crowdsourcing

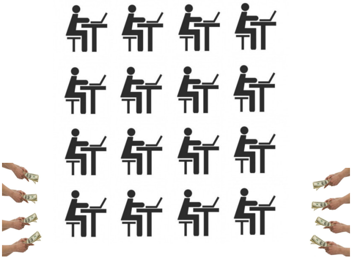
- Classification task.
- Crowdsource: \$ for labels.
- Fixed budget;
  minimize errors.
- Bayesian model:
  binary labels, i.i.d. crowd;
  $Y_i \sim \text{Bernoulli}(p_i)$

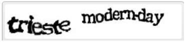


http://www.technicalinfo.net/
http://cdns2.freepik.com/

Simulation Results: Crowdsourcing

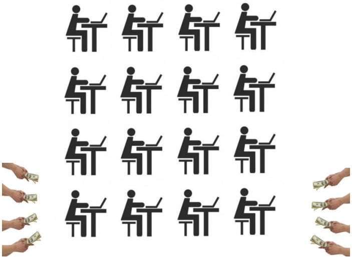
- Classification task.
- Crowdsource: $ for labels.
- Fixed budget;
  minimize errors.
- Bayesian model:
  binary labels, i.i.d. crowd;
  $Y_i \sim \text{Bernoulli}(p_i)$

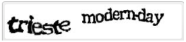




Simulation Results: Crowdsourcing

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



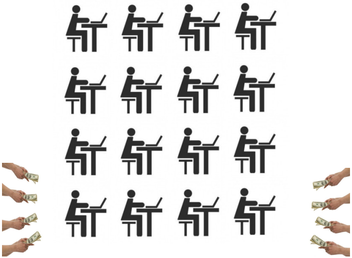


http://www.technicalinfo.net/

http://cdns2.freepik.com/

Simulation Results: Crowdsourcing

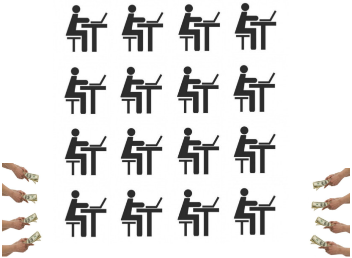
- Classification task.
- Crowdsource: \$ for labels.
- Fixed budget;
  minimize errors.
- Bayesian model:
  binary labels, i.i.d. crowd;
  $Y_i \sim \text{Bernoulli}(p_i)$;
  $p_i \sim \text{Beta}$.

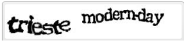


http://www.technicalinfo.net/
http://cdns2.freepik.com/

Simulation Results: Crowdsourcing

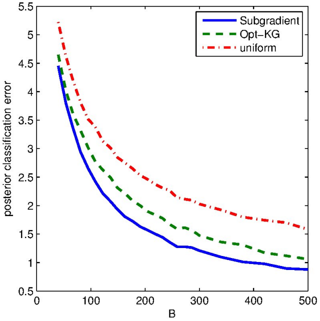
- Classification task.
- Crowdsource: $ for labels.
- Fixed budget;
  minimize errors.
- Bayesian model:
  binary labels, i.i.d. crowd;
  $Y_i \sim \text{Bernoulli}(p_i)$;
  $p_i \sim \text{Beta}$.
- State $=$ posterior.





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.



Outline

1. Large-Scale Policy Design

- Compete with a restricted family of policies Π:
  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

Outline

1. Large-Scale Policy Design

- Compete with a restricted family of policies Π:
  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.

Outline

1. Large-Scale Policy Design

- Compete with a restricted family of policies Π:
  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 Π:
  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 Π.

Outline

1. Large-Scale Policy Design

- Compete with a restricted family of policies Π:
  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 Π.
- Computationally efficient if Π has polynomial size.

Outline

1. Large-Scale Policy Design

- Compete with a restricted family of policies Π:
  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 Π.
- Computationally efficient if Π has polynomial size.
- Hard for shortest path problems.

Changing Dynamics

(with Yasin Abbasi-Yadkori, Varun Kanade, Yevgeny Seldin, Csaba Szepesvari, NIPS2013)

- Observe P_t, ℓ_t after round t.

Changing Dynamics

(with Yasin Abbasi-Yadkori, Varun Kanade, Yevgeny Seldin, Csaba Szepesvari, NIPS2013)

- Observe P_t, ℓ_t after round t.
- Consider a comparison class: $\Pi \subset \{\pi \mid \pi : \mathcal{X} \to \mathcal{A}\}$

Changing Dynamics

(with Yasin Abbasi-Yadkori, Varun Kanade, Yevgeny Seldin, Csaba Szepesvari, NIPS2013)

- Observe P_t, ℓ_t after round t.
- Consider a comparison class: $\Pi \subset \{\pi \mid \pi : \mathcal{X} \to \mathcal{A}\}$
- $\pi^* = \operatorname{argmin}_{\pi \in \Pi} \sum_{t=1}^{T} \ell_t(x_t^\pi, \pi(x_t^\pi))$

Changing Dynamics

(with Yasin Abbasi-Yadkori, Varun Kanade, Yevgeny Seldin, Csaba Szepesvari, NIPS2013)

- Observe P_t, ℓ_t after round t.
- Consider a comparison class: $\Pi \subset \{\pi \mid \pi : \mathcal{X} \to \mathcal{A}\}$
- $\pi^* = \operatorname{argmin}_{\pi \in \Pi} \sum_{t=1}^{T} \ell_t(x_t^\pi, \pi(x_t^\pi))$
- $R_T = \sum_{t=1}^{T} \ell_t(x_t, a_t) - \sum_{t=1}^{T} \ell_t(x_t^{\pi^*}, \pi^*(x_t^{\pi^*}))$

Changing Dynamics

(with Yasin Abbasi-Yadkori, Varun Kanade, Yevgeny Seldin, Csaba Szepesvari, NIPS2013)

- Observe P_t, ℓ_t after round t.
- Consider a comparison class: $\Pi \subset \{\pi \mid \pi : \mathcal{X} \to \mathcal{A}\}$
- $\pi^* = \mathrm{argmin}_{\pi \in \Pi} \sum_{t=1}^{T} \ell_t(x_t^{\pi}, \pi(x_t^{\pi}))$
- $R_T = \sum_{t=1}^{T} \ell_t(x_t, a_t) - \sum_{t=1}^{T} \ell_t(x_t^{\pi^*}, \pi^*(x_t^{\pi^*}))$

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

Changing Dynamics

(with Yasin Abbasi-Yadkori, Varun Kanade, Yevgeny Seldin, Csaba Szepesvari, NIPS2013)

- Observe P_t, ℓ_t after round t.
- Consider a comparison class: $\Pi \subset \{\pi \mid \pi : \mathcal{X} \to \mathcal{A}\}$
- $\pi^* = \mathrm{argmin}_{\pi\in\Pi} \sum_{t=1}^{T} \ell_t(x_t^{\pi}, \pi(x_t^{\pi}))$
- $R_T = \sum_{t=1}^{T} \ell_t(x_t, a_t) - \sum_{t=1}^{T} \ell_t(x_t^{\pi^*}, \pi^*(x_t^{\pi^*}))$

- Aim for low regret: $R_T / T \to 0$
- Computationally efficient low regret strategies?

Regret Bound

Main Result

There is a strategy that (under a τ-mixing assumption) achieves

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

Exponential weights:

Strategy for a repeated game:

Choose action $a \in \mathcal{A}$ with probability proportional to

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

Exponential weights:

Strategy for a repeated game:

Choose action $a \in \mathcal{A}$ with probability proportional to

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

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

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

Exponential weights:

Strategy for a repeated game:

Choose action $a \in \mathcal{A}$ with probability proportional to

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

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

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

- Long history.

Exponential weights:

Strategy for a repeated game:

Choose action $a \in \mathcal{A}$ with probability proportional to

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

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

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

- 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. $\beta_t = \dfrac{w_{\pi_{t-1},t-1}}{w_{\pi_{t-1},t-2}}$, $\pi_t = \pi_{t-1}$. Otherwise $\pi_t \sim p_{.,t}$.
  Choose action $a_t \sim \pi_t(.|x_t)$.
  Observe dynamics P_t and loss ℓ_t.
  Suffer $\ell_t(x_t, a_t)$.
  For all policies π, $w_{\pi,t} = w_{\pi,t-1} \exp\left(-\eta \mathbb{E}\left[\ell_t(x_t^\pi, \pi)\right]\right)$.
end for

- Exponential weights on Π.

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 ℓ_t.
  Suffer $\ell_t(x_t, a_t)$.
  For all policies π, $w_{\pi,t} = w_{\pi,t-1} \exp\left(-\eta \mathbb{E}\left[\ell_t(x_t^\pi, \pi)\right]\right)$.
end for

- Exponential weights on Π.
- Rare, random changes to π_t.

Regret Bound

Main Result

There is a strategy that (under a τ-mixing assumption) achieves

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

- Adversarial dynamics and loss functions.

Regret Bound

Main Result

There is a strategy that (under a τ-mixing assumption) achieves

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

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

Regret Bound

Main Result

There is a strategy that (under a τ-mixing assumption) achieves

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

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

Regret Bound

Main Result

There is a strategy that (under a τ-mixing assumption) achieves

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

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

Regret Bound

Main Result

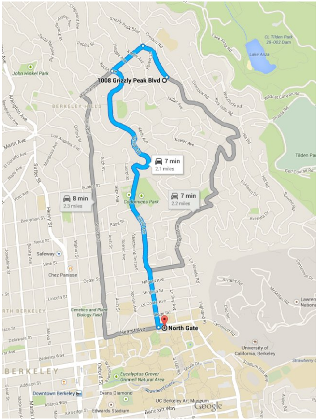
There is a strategy that (under a τ-mixing assumption) achieves

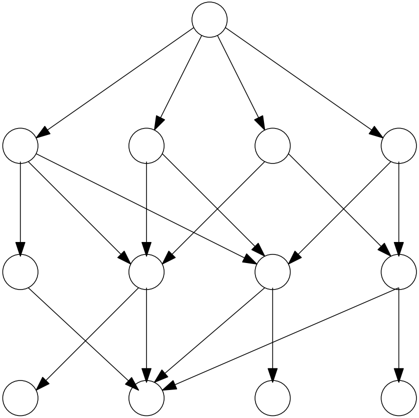
$$\mathbb{E}\left[R_T\right] \leq (4 + 2\tau^2)\sqrt{T \log |\Pi|} + \log |\Pi| .$$

- Adversarial dynamics and loss functions.
- Large state and action spaces.
- $\mathbb{E}\left[R_T\right] / T \to 0$ 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.

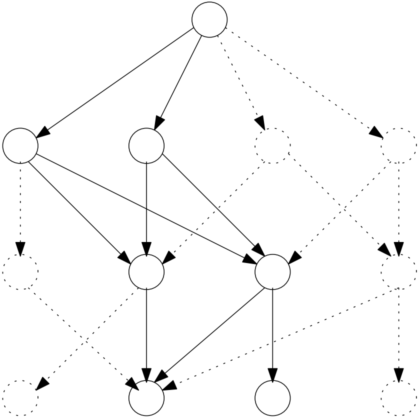




http://www.google.com/

Shortest Path Problem

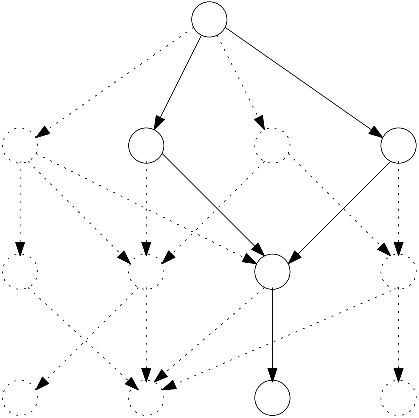
Special case of MDP: node=state; action=edge; loss=weight.



http://www.google.com/
http://www.meondirect.com/

Shortest Path Problem

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



http://www.google.com/
http://www.meondirect.com/

Computational Efficiency

Hardness Result

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

1. runs in time $\mathrm{poly}(n, T)$, and
2. has regret $R_T = O(\mathrm{poly}(n) T^{1-\delta})$ 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 $\{0,1\}^n$:
  $\mathrm{PARITIES} = \{\mathrm{PAR}_S \mid S \subset [n], \mathrm{PAR}_S(x) = \oplus_{i \in S} x_i\}$

Online Agnostic Parity Learning

- Class of parity functions on $\{0,1\}^n$:
  $\mathrm{PARITIES} = \{\mathrm{PAR}_S \mid S \subset [n], \mathrm{PAR}_S(x) = \oplus_{i \in S} x_i\}$
- Learning problem: given $x_t \in \{0,1\}^n$, learner predicts $\widehat{y}_t \in \{0,1\}$, observes the true label y_t and suffers loss $\mathbb{I}_{\{\widehat{y}_t \neq y_t\}}$

Online Agnostic Parity Learning

- Class of parity functions on $\{0,1\}^n$:
  $\mathrm{PARITIES} = \{\mathrm{PAR}_S \mid S \subset [n], \mathrm{PAR}_S(x) = \oplus_{i \in S} x_i\}$
- Learning problem: given $x_t \in \{0,1\}^n$, learner predicts $\widehat{y}_t \in \{0,1\}$, observes the true label y_t and suffers loss $\mathbb{I}_{\{\widehat{y}_t \neq y_t\}}$
- $R_T = \sum_{t=1}^{T} \mathbb{I}_{\{\widehat{y}_t \neq y_t\}} - \min_{\mathrm{PAR}_S \in \mathrm{PARITIES}} \sum_{t=1}^{T} \mathbb{I}_{\{\mathrm{PAR}_S(x_t) \neq y_t\}}$

Online Agnostic Parity Learning

- Class of parity functions on $\{0,1\}^n$:
  $\mathrm{PARITIES} = \{\mathrm{PAR}_S \mid S \subset [n], \mathrm{PAR}_S(x) = \oplus_{i \in S} x_i\}$
- Learning problem: given $x_t \in \{0,1\}^n$, learner predicts $\widehat{y}_t \in \{0,1\}$, observes the true label y_t and suffers loss $\mathbb{I}_{\{\widehat{y}_t \neq y_t\}}$
- $R_T = \sum_{t=1}^T \mathbb{I}_{\{\widehat{y}_t \neq y_t\}} - \min_{\mathrm{PAR}_S \in \mathrm{PARITIES}} \sum_{t=1}^T \mathbb{I}_{\{\mathrm{PAR}_S(x_t) \neq y_t\}}$

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

Online Agnostic Parity Learning

- Class of parity functions on $\{0,1\}^n$:
  $\mathsf{PARITIES} = \{\mathsf{PAR}_S \mid S \subset [n], \mathsf{PAR}_S(x) = \oplus_{i \in S} x_i\}$
- Learning problem: given $x_t \in \{0,1\}^n$, learner predicts $\widehat{y}_t \in \{0,1\}$, observes the true label y_t and suffers loss $\mathbb{I}_{\{\widehat{y}_t \neq y_t\}}$
- $R_T = \sum_{t=1}^{T} \mathbb{I}_{\{\widehat{y}_t \neq y_t\}} - \min_{\mathsf{PAR}_S \in \mathsf{PARITIES}} \sum_{t=1}^{T} \mathbb{I}_{\{\mathsf{PAR}_S(x_t) \neq y_t\}}$
- Is there an efficient (time polynomial in n, T) learning algorithm with sublinear regret ($R_T = O(\mathrm{poly}(n) T^{1-\delta})$ for some $\delta > 0$)?
- Very well-studied.

Online Agnostic Parity Learning

- Class of parity functions on $\{0,1\}^n$:
  $\mathsf{PARITIES} = \{\mathsf{PAR}_S \mid S \subset [n], \mathsf{PAR}_S(x) = \oplus_{i \in S} x_i\}$
- Learning problem: given $x_t \in \{0,1\}^n$, learner predicts $\widehat{y}_t \in \{0,1\}$, observes the true label y_t and suffers loss $\mathbb{I}_{\{\widehat{y}_t \neq y_t\}}$
- $R_T = \sum_{t=1}^{T} \mathbb{I}_{\{\widehat{y}_t \neq y_t\}} - \min_{\mathsf{PAR}_S \in \mathsf{PARITIES}} \sum_{t=1}^{T} \mathbb{I}_{\{\mathsf{PAR}_S(x_t) \neq y_t\}}$

- Is there an efficient (time polynomial in n, T) learning algorithm with sublinear regret ($R_T = O(\mathrm{poly}(n) T^{1-\delta})$ for some $\delta > 0$)?
- Very well-studied.
- Widely believed to be hard: used for cryptographic schemes.

Computational Efficiency

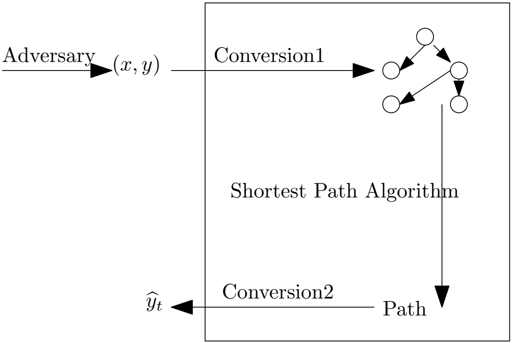
Hardness Result

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

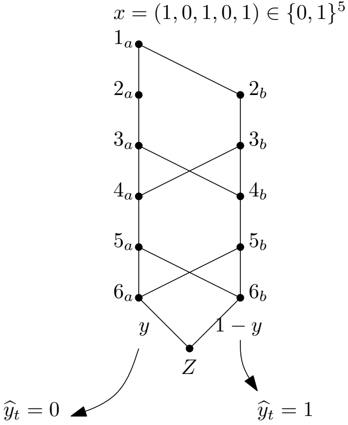
1. runs in time $\mathrm{poly}(n, T)$, and
2. has regret $R_T = O(\mathrm{poly}(n) T^{1-\delta})$ for some constant $\delta > 0$.

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

Reduction



Adversary
(x, y)
Conversion1
Shortest Path Algorithm
Conversion2
Path
$\widehat{y}_t$



$x = (1, 0, 1, 0, 1) \in \{0, 1\}^5$
1_a
2_a
2_b
3_a
3_b
4_a
4_b
5_a
5_b
6_a
6_b
y
$1 - y$
Z
$\widehat{y}_t = 0$
$\widehat{y}_t = 1$

Online shortest path: Hard versus easy

Edges (dynamics)	Weights (costs)	
Adversarial	Adversarial	As hard as noisy parity.
Stochastic	Adversarial	Efficient algorithm.
Adversarial	Stochastic	Efficient algorithm.

Outline

1. Large-Scale Policy Design

- Compete with a restricted family of policies Π:
  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 Π.
- Computationally efficient if Π has polynomial size.
- Hard for shortest path problems.