Decentralized Stochastic Approximation, Optimization, and Multi-Agent Reinforcement Learning

Justin Romberg, Georgia Tech ECE
CAMDA/TAMIDS Seminar, Texas A&M
College Station, Texas  Streaming live from Atlanta, Georgia
March 16, 2020  October 30, 2020
Collaborators

Thinh Doan
Virginia Tech, ECE

Siva Theja Maguluri
Georgia Tech, ISyE

Sihan Zeng
Georgia Tech, ECE
Reinforcement Learning
Distributed RL is a combination of:

- stochastic approximation
- Markov decision processes
- function representation
- network consensus
Distributed RL is a combination of:

- stochastic approximation
- Markov decision processes
- function representation
- network consensus
- (complicated probabilistic analysis)
Distributed RL is a combination of:

- *stochastic approximation*
- Markov decision processes
- function representation
- network consensus
- (complicated probabilistic analysis)
Classical result (Banach fixed point theorem): when $H(\cdot) : \mathbb{R}^N \to \mathbb{R}^N$ is a contraction

$$\|H(u) - H(v)\| \leq \delta \|u - v\|, \quad \delta < 1,$$

then there is a unique fixed point $x^*$ such that

$$x^* = H(x^*),$$

and the iteration

$$x_{k+1} = H(x_k),$$

finds it

$$\lim_{k \to \infty} x_k = x^*.$$
Choose any point $x_0$, then take

$$x_{k+1} = H(x_k)$$

so

$$x_{k+1} - x^* = H(x_k) - x^* = H(x_k) - H(x^*)$$

and

$$\|x_{k+1} - x^*\| = \|H(x_k) - H(x^*)\| \leq \delta \|x_k - x^*\| \leq \delta^{k+1} \|x_0 - x^*\|,$$

so the convergence is geometric.
Choose any point $x_0$, then take

$$x_{k+1} = H(x_k),$$

then

$$\|x_{k+1} - x^*\| = \|H(x_k) - H(x^*)\| \leq \delta^{k+1}\|x_0 - x^*\|,$$

**Gradient descent** takes

$$H(x) = x - \alpha \nabla f(x)$$

for some differentiable $f$. 
Take
\[ x_{k+1} = x_k + \alpha(H(x_k) - x_k), \quad 0 < \alpha \leq 1. \]
(More conservative, convex combination of new iterate and old.)

Then again
\[ x_{k+1} = (1 - \alpha)x_k + \alpha H(x_k) \]
and
\[
||x_{k+1} - x^*|| \leq (1 - \alpha)||x_k - x^*|| + \alpha||H(x_k) - H(x^*)|| \\
\leq (1 - \alpha - \delta\alpha)||x_k - x^*||.
\]

Still converge, albeit a little more slowly for $\alpha < 1$. 
If our observations of $H(\cdot)$ are *noisy*,

$$x_{k+1} = x_k + \alpha (H(x_k) - x_k + \eta_k), \quad \mathbb{E}[\eta_k] = 0,$$

then we don’t get convergence for fixed $\alpha$,

but we do converge to a “ball” around at a geometric rate
Stochastic approximation

If our observations of $H(\cdot)$ are noisy,

$$x_{k+1} = x_k + \alpha_k (H(x_k) - x_k + \eta_k), \quad \mathbb{E}[\eta_k] = 0,$$

then we need to take $\alpha_k \to 0$ as we approach the solution.

If we take $\{\alpha_k\}$ such that

$$\sum_{k=0}^{\infty} \alpha_k^2 < \infty, \quad \sum_{k=0}^{\infty} \alpha_k = \infty,$$

then we so get (much slower) convergence

Example: $\alpha_k = C/(k + 1)$
Distributed RL is a combination of:

- stochastic approximation
- Markov decision processes
- function representation
- network consensus
- (complicated probabilistic analysis)
At time $t$,

1. An agent finds itself in a **state** $s_t$
2. It takes action $a_t = \mu(s_t)$
3. It moves to state $s_{t+1}$ according to $P(s_{t+1}|s_t, a_t)$...
4. ... and receives reward $R(s_t, a_t, s_{t+1})$. 

Markov decision process

At time $t$,

1. An agent finds itself in a **state** $s_t$
2. It takes action $a_t = \mu(s_t)$
3. It moves to state $s_{t+1}$ according to $P(s_{t+1} \mid s_t, a_t)$...
4. ... and receives reward $R(s_t, a_t, s_{t+1})$.

Long-term reward of **policy** $\mu$:

$$V_\mu(s) = E \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t, \mu(s_t), s_{t+1}) \mid s_0 = s \right]$$
At time $t$,

1. An agent finds itself in a state $s_t$
2. It takes action $a_t = \mu(s_t)$
3. It moves to state $s_{t+1}$ according to $P(s_{t+1}|s_t, a_t)$...
4. ... and receives reward $R(s_t, a_t, s_{t+1})$.

Bellman equation: $V_\mu$ obeys

$$V_\mu(s) = \sum_{z \in S} P(z|s, \mu(s)) \left[ R(s, \mu(s), z) + \gamma V_\mu(z) \right]$$

This is a fixed point equation for $V_\mu$. 
At time $t$, 

1. An agent finds itself in a state $s_t$
2. It takes action $a_t = \mu(s_t)$
3. It moves to state $s_{t+1}$ according to $P(s_{t+1}|s_t, a_t)$...
4. ... and receives reward $R(s_t, a_t, s_{t+1})$.

State-action value function ($Q$ function):

$$Q_\mu(s, a) = E \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t, \mu(s_t)s_{t+1}) \mid s_0 = s, a_0 = a \right]$$
Markov decision process

At time $t$,

1. An agent finds itself in a state $s_t$
2. It takes action $a_t = \mu(s_t)$
3. It moves to state $s_{t+1}$ according to $P(s_{t+1}|s_t, a_t)$...
4. ... and receives reward $R(s_t, a_t, s_{t+1})$.

State-action value for the optimal policy obeys

$$Q^*(s, a) = \mathbb{E} \left[ R(s, a, s') + \gamma \max_{a'} Q^*(s', a') \mid s_0 = s, a_0 = a \right]$$

and we take $\mu^*(s) = \arg \max_a Q^*(s, a)$ ...

... this is another fixed point equation
Fixed point iteration for finding $V_\mu(s)$:

$$V_{t+1}(s) = V_t(s) + \alpha \left( \sum_z P(z|s) \left[ R(s, z) + \gamma V_t(z) \right] - V_t(s) \right)$$

$$H(V_t) - V_t$$
Stochastic approximation for policy evaluation

Fixed point iteration for finding $V_\mu(s)$:

$$V_{t+1}(s) = V_t(s) + \alpha \left( \sum_z P(z|s) [R(s,z) + \gamma V_t(z)] - V_t(s) \right)$$

In practice, we don’t have the model $P(z|s)$, only observed data $\{(s_t, s_{t+1})\}$
Stochastic approximation for policy evaluation

Fixed point iteration for finding $V_\mu(s)$:

$$V_{t+1}(s) = V_t(s) + \alpha \left( \sum_z P(z|s) [R(s, z) + \gamma V_t(z)] - V_t(s) \right)$$

$$H(V_t) - V_t$$

Stochastic approximation iteration

$$V_{t+1}(s_t) = V_t(s_t) + \alpha_t (R(s_t, s_{t+1}) + \gamma V_t(s_{t+1}) - V_t(s_t))$$

The “noise” is that $s_{t+1}$ is sampled, rather than averaged over
Fixed point iteration for finding $V_\mu(s)$:

$$V_{t+1}(s) = V_t(s) + \alpha \left( \sum_z P(z|s) [R(s, z) + \gamma V_t(z)] - V_t(s) \right)$$

Stochastic approximation iteration

$$V_{t+1}(s_t) = V_t(s_t) + \alpha_t \left( R(s_t, s_{t+1}) + \gamma V_t(s_{t+1}) - V_t(s_t) \right)$$

The “noise” is that $s_{t+1}$ is sampled, rather than averaged over
Fixed point iteration for finding $V_\mu(s)$:

$$V_{t+1}(s) = V_t(s) + \alpha \left( \sum_z P(z|s) [R(s, z) + \gamma V_t(z)] - V_t(s) \right)$$

Stochastic approximation iteration

$$V_{t+1}(s_t) = V_t(s_t) + \alpha_t \left( R(s_t, s_{t+1}) + \gamma V_t(s_{t+1}) - V_t(s_t) \right)$$

The “noise” is that $s_{t+1}$ is sampled, rather than averaged over

This is different from stochastic gradient descent, since $H(\cdot)$ is in general not a gradient map
Distributed RL is a combination of:

- stochastic approximation
- Markov decision processes
- function representation
- network consensus
- (complicated probabilistic analysis)
State space can be large (or even infinite) ... 

... we need a natural way to parameterize/simplify
Simple (but powerful) model: linear representation

\[
V(s; \theta) = \sum_{k=1}^{K} \theta_k \phi_k(s) = \phi(s)^T \theta, \quad \phi(s) = \begin{bmatrix} \phi_1(s) \\ \vdots \\ \phi_K(s) \end{bmatrix}
\]
Linear function approximation

Simple (but powerful) model: linear representation

\[ V(s; \theta) = \sum_{k=1}^{K} \theta_k \phi_k(s) = \phi(s)^T \theta, \quad \phi(s) = \begin{bmatrix} \phi_1(s) \\ \vdots \\ \phi_K(s) \end{bmatrix} \]
Bellman equation:

\[ V(s) = \sum_{z \in S} P(z|s) [R(s, \mu(s), z) + \gamma V(z)] \]

Linear approximation:

\[ V(s; \theta) = \sum_{k=1}^{K} \theta_k \phi_k(s) = \phi(s)^T \theta \]

These can conflict ....
Bellman equation:

$$V(s) = \sum_{z \in S} P(z|s) [R(s, \mu(s), z) + \gamma V(z)]$$

Linear approximation:

$$V(s; \theta) = \sum_{k=1}^{K} \theta_k \phi_k(s) = \phi(s)^T \theta$$

These can conflict ...

... but the following iterations

$$\theta_{t+1} = \theta_t + \alpha_t (R(s_t, s_{t+1}) + \gamma V(s_{t+1}; \theta_t) - V(s_t; \theta_t)) \nabla \theta V(s_t, \theta_t)$$

$$= \theta_t + \alpha_t \left( R(s_t, s_{t+1}) + \gamma \phi(s_{t+1})^T \theta_t - \phi(s_t)^T \theta_t \right) \phi(s_t)$$

converge to a “near optimal” $\theta^*$

Tsitsiklis and Roy, '97
Distributed RL is a combination of:

- stochastic approximation
- Markov decision processes
- function representation
- network consensus
- (complicated probabilistic analysis)
Network consensus

- Each node in a network has a number $x(i)$
- We want each node to agree on the average

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x(i) = 1^T x$$

- Node $i$ communicates with its neighbors $\mathcal{N}_i$
- Iterate, take $v_0 = x$, then

$$v_{k+1}(i) = \sum_{j \in \mathcal{N}_i} W_{ij} v_k(i)$$

$$v_{k+1} = Wv_k, \quad W \text{ doubly stochastic}$$
Nodes reach “consensus” quickly:

\[ \mathbf{v}_{k+1} = W \mathbf{v}_k \]
\[ \mathbf{v}_{k+1} - \bar{x} \mathbf{1} = W \mathbf{v}_k - \bar{x} \mathbf{1} \]
\[ = W (\mathbf{v}_k - \bar{x} \mathbf{1}) \]
\[ \| \mathbf{v}_{k+1} - \bar{x} \mathbf{1} \| = \| W (\mathbf{v}_k - \bar{x} \mathbf{1}) \| \]
Nodes reach “consensus” quickly:

\[ \mathbf{v}_{k+1} = W \mathbf{v}_k \]
\[ \mathbf{v}_{k+1} - \bar{x} \mathbf{1} = W \mathbf{v}_k - \bar{x} \mathbf{1} \]
\[ = W (\mathbf{v}_k - \bar{x} \mathbf{1}) \]
\[ \| \mathbf{v}_{k+1} - \bar{x} \mathbf{1} \| = \| W (\mathbf{v}_k - \bar{x} \mathbf{1}) \| \]
\[ \leq \sigma_2 \| \mathbf{v}_k - \bar{x} \mathbf{1} \| \]
\[ \leq \sigma_2^{k+1} \| \mathbf{v}_0 - \bar{x} \mathbf{1} \| \]
Network consensus convergence

Nodes reach “consensus” quickly:

\[ v_{k+1} = W v_k \]
\[ v_{k+1} - \bar{x}1 = W v_k - \bar{x}1 \]
\[ = W(v_k - \bar{x}1) \]
\[ \|v_{k+1} - \bar{x}1\| = \|W(v_k - \bar{x}1)\| \]
\[ \leq \sigma_2 \|v_k - \bar{x}1\| \]
\[ \leq \sigma_2^{k+1} \|v_0 - \bar{x}1\| \]
Multi-agent Reinforcement Learning, Scenario 1:
Multiple agents in a single environment, common state, different rewards

What is the value of a particular policy?
Multi-agent reinforcement learning

- $N$ agents, communicating on a network
- One environment, common state $s_t \in S$
  transition probabilities $P(s_{t+1}|s_t)$
- Individual actions $a_t^i \in A^i$
- Individual rewards $R^i(s_t, s_{t+1})$
- evaluate policies $\mu^i : S \rightarrow A^i$
Multi-agent reinforcement learning

- $N$ agents, communicating on a network
- One environment, common state $s_t \in S$
  transition probabilities $P(s_{t+1}|s_t)$
- Individual actions $a_t^i \in A^i$
- Individual rewards $R^i(s_t, s_{t+1})$
- compute average cumulative reward

$$V(s) = \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t \frac{1}{N} \sum_{i=1}^{N} R^i(s_t, s_{t+1}) \mid s_0 = s \right]$$
Multi-agent reinforcement learning

- \( N \) agents, communicating on a network
- One environment, common state \( s_t \in S \)
  - transition probabilities \( P(s_{t+1} | s_t) \)
- Individual actions \( a_t^i \in A^i \)
- Individual rewards \( R^i(s_t, s_{t+1}) \)
- find \( V \) that satisfies

\[
V(s) = \sum_{z \in S} P(z | s) \left[ \frac{1}{N} \sum_{n=1}^{N} R^i(s, z) + \gamma V(z) \right]
\]
Distributed temporal difference learning

**Initialize:** Each agent starts at $\theta^i_0$

**Iterations:** Observe: $s_t$, take action to go to $s_{t+1}$, get reward $R(s_t, s_{t+1})$

Communicate: average estimates from neighbors

$$y^i_t = \sum_{j \in N_i} W_{ij} \theta^j_t$$

Local updates:

$$\theta^i_{t+1} = y^i_t + \alpha_t d^i_t \phi(s_t),$$

where

$$d^i_t = R^i(s_t, s_{t+1}) + \gamma \phi(s_{t+1})^T \theta^i_t - \phi(s_t)^T \theta^i_t$$
Distributed RL is a combination of:

- stochastic approximation
- Markov decision processes
- function representation
- network consensus
- (complicated probabilistic analysis)
Previous work

Subset of existing results:

- Unified convergence theory: Borkar and Meyn '00
- Convergence rates with “independent noise” (centralized):
  Thoppe and Borkar '19, Dalal et al '18, Lakshminarayanan and Szepesvari '18
- Convergence rates under Markovian noise (centralized):
  Bhandari et al COLT '18. Srikant and Ying COLT '19
- Multi-agent RL: Mathkar and Bokar '17, Zhang et al '18, Kar et al '13, Stankovic and Stankovic '16, Macua et al '15
Fixed step size $\alpha_t = \alpha$, for small enough $\alpha$

$$E[\|\theta^i_t - \theta^*\|] \leq O(\sigma^{t-\tau}) + O(\eta^{t-\tau}) + O(\alpha)$$

where $\sigma < 1$ is network connectivity, $\eta < 1$ are problem parameters, and $\tau$ is the mixing time for the underlying Markov chain.
Rate of convergence for distributed TD

- Fixed step size $\alpha_t = \alpha$, for small enough $\alpha$
  \[
  \mathbb{E}[\|\theta^i_t - \theta^*\|] \leq O(\sigma^{t-\tau}) + O(\eta^{t-\tau}) + O(\alpha)
  \]

  where $\sigma < 1$ is network connectivity, $\eta < 1$ are problem parameters, and $\tau$ is the mixing time for the underlying Markov chain.

- Time-varying step size $\alpha_t \sim 1/(t+1)$
  \[
  \mathbb{E}[\|\theta^i_t - \theta^*\|] \leq O(\sigma^{t-\tau}) + O(T(1-\sigma^2)^2 \log(t+1)/t+1)
  \]
Fixed **step size** $\alpha_t = \alpha$, for small enough $\alpha$

$$\mathbb{E} \left[ \| \theta^i_t - \theta^* \| \right] \leq O(\sigma^{t-\tau}) + O(\eta^{t-\tau}) + O(\alpha)$$

where $\sigma < 1$ is network connectivity, $\eta < 1$ are problem parameters, and $\tau$ is the mixing time for the underlying Markov chain.
Rate of convergence for distributed TD

- **Fixed step size** $\alpha_t = \alpha$, for small enough $\alpha$

  $$
  \mathbb{E} \left[ \| \theta^i_t - \theta^* \| \right] \leq O(\sigma^{t-\tau}) + O(\eta^{t-\tau}) + O(\alpha)
  $$

  where $\sigma < 1$ is network connectivity, $\eta < 1$ are problem parameters, and $\tau$ is the mixing time for the underlying Markov chain.

- **Time-varying step size** $\alpha_t \sim 1/(t + 1)$

  $$
  \mathbb{E} \left[ \| \theta^i_t - \theta^* \| \right] \leq O(\sigma^{t-\tau}) + O\left( \frac{T}{(1 - \sigma_2)^2} \frac{\log(t + 1)}{t + 1} \right)
  $$
Rate of convergence for distributed TD

- **Fixed step size** $\alpha_t = \alpha$, for small enough $\alpha$
  \[
  \mathbb{E} \left[ \| \theta^i_t - \theta^* \| \right] \leq O(\sigma^{t-\tau}) + O(\eta^{t-\tau}) + O(\alpha)
  \]
  where $\sigma < 1$ is network connectivity, $\eta < 1$ are problem parameters, and $\tau$ is the mixing time for the underlying Markov chain

- **Time-varying step size** $\alpha_t \sim 1/(t + 1)$
  \[
  \mathbb{E} \left[ \| \theta^i_t - \theta^* \| \right] \leq O(\sigma^{t-\tau}) + O\left( \frac{T}{(1 - \sigma_s)^2} \frac{\log(t + 1)}{t + 1} \right)
  \]
Goal: Find $\theta^*$ such that $\bar{F}(\theta^*) = 0$, where

$$\bar{F}(\theta) = \sum_{i=1}^{N} \mathbb{E}[F_i(X_i; \theta)],$$

using decentralized communications between agents with access to $F_i(X_i; \theta)$.

Using the iteration

$$\theta_{i+1}^{k} = \sum_{i \in \mathcal{N}(i)} W_{i,j} \theta^k_j + \epsilon F_i(X^k_i, \theta^k_i)$$

gives us

$$\max_j \mathbb{E}\left[\|\theta^k_i - \theta^*\|_2^2\right] \to O\left(\frac{\epsilon \log(1/\epsilon)}{1 - \sigma^2_2}\right)$$

at a linear rate when the $F_i$ are Lipschitz, $\bar{F}_i$ are strongly monotone, and the $\{X^k_i\}$ are Markov.
Multi-agent Reinforcement Learning, Scenario 2:
Multiple agents in different environments (dynamics, rewards)

Can we find a jointly optimal policy?
Policy Optimization, Framework

We will set this up as a distributed optimization program with decentralized communications

- One agent explores each environment
- Agent collaborate by sharing their models
- Performance guarantees:
  - number of gradient iterations
  - sample complexity (future)
Policy Optimization, Framework

Environments $i = 1, \ldots, N$, each with similar state/action spaces

Key quantities:

- $\pi(\cdot|s)$: policy that maps states into actions
- $r_i(s, a)$: reward function in environment $i$
- $\rho_i(s)$: initial state distribute in environment $i$
- $L_i(\pi)$: long-term reward of $\pi$ in environment $i$

\[
L_i(\pi) = E \left[ \sum_{k=0}^{\infty} \gamma^k r_i(s^k_i, a^k_i) \right], \quad a^k_i \sim \pi(\cdot|s^{k-1}_i), \quad s^0_i \sim \rho_i
\]

We want to solve

\[
\max_{\pi} \sum_{i=1}^{N} L_i(\pi)
\]
Decentralized Policy Optimization, Challenges

\[
\text{maximize} \sum_{i=1}^{N} L_i(\pi) \rightarrow \text{maximize} \sum_{i=1}^{N} L_i(\theta), \quad \pi_{\theta}(a|s) = \frac{e^{\theta_{s,a}}}{\sum_{a'} e^{\theta_{s,a'}}}
\]

- Natural parameterization (softmax) is ill-conditioned at solution
Decentralized Policy Optimization, Challenges

\[
\max_{\pi} \sum_{i=1}^{N} L_i(\pi) \quad \rightarrow \quad \max_{\theta} \sum_{i=1}^{N} L_i(\theta) - \lambda \text{RE}(\theta), \quad \pi_{\theta}(a|s) = \frac{e^{\theta_{s,a}}}{\sum_{a'} e^{\theta_{s,a'}}}
\]

- Natural parameterization (softmax) is **ill-conditioned at solution**
Decentralized Policy Optimization, Challenges

\[
\text{maximize } \prod_{i=1}^{N} L_i(\pi) \rightarrow \text{maximize } \prod_{i=1}^{N} L_i(\theta) - \lambda \text{RE}(\theta), \quad \pi_{\theta}(a|s) = \frac{e^{\theta_{s,a}}}{\sum_{a'} e^{\theta_{s,a'}}}
\]

- Natural parameterization (softmax) is ill-conditioned at solution

- Even for a single agent, this problem is nonconvex ...
  ... ability to find global optimum tied to “exploration conditions” (Agarwal et al ’19)
Decentralized Policy Optimization, Challenges

\[
\max_{\pi} \sum_{i=1}^{N} L_i(\pi) \quad \Rightarrow \quad \max_{\theta} \sum_{i=1}^{N} L_i(\theta) - \lambda \text{RE}(\theta), \quad \pi_{\theta}(a|s) = \frac{e^{\theta_{s,a}}}{\sum_{a'} e^{\theta_{s,a'}}}
\]

- Natural parameterization (softmax) is ill-conditioned at solution

- Even for a single agent, this problem is nonconvex ...
  ... ability to find global optimum tied to “exploration conditions” (Agarwal et al ’19)

- Agents have competing interests (global solution suboptimal for every agent)
Decentralized Policy Optimization, Challenges

\[
\text{maximize } \prod_{\pi} \sum_{i=1}^{N} L_i(\pi) \rightarrow \text{maximize } \prod_{\theta} \sum_{i=1}^{N} L_i(\theta) - \lambda \text{RE}(\theta), \quad \pi_\theta(a|s) = \frac{e^{\theta s,a}}{\sum_{a'} e^{\theta s,a'}}
\]

- Natural parameterization (softmax) is ill-conditioned at solution.

- Even for a single agent, this problem in nonconvex ...
  
  ... ability to find global optimum tied to “exploration conditions” (Agarwal et al '19)

- Agents have competing interests (global solution suboptimal for every agent)

- Gradients can only be computed imperfectly for large or partially specified problems
Algorithm: Decentralized Policy Optimization

maximize \( \sum_{i=1}^{N} L_i(\theta_i) \), subject to \( \theta_i = \theta_j, (i, j) \in \mathcal{E} \)

- Each agent stores a local version of policy \( \theta_i \), initialized to \( \theta_i^0 \)
- At each node, iterate from policy \( \pi_{\theta_i^k} \)
  - Compute “advantage function” \( A(s, a) = Q(s, a) - V(s) \)
  - Compute gradient
    \[
    \nabla L_i(\theta_i^k) = (\text{complicated function of } \pi_{\theta_i^k} \text{ and } A(s, a))
    \]
  - Meanwhile, exchange \( \theta_i^k \) with neighbors
  - Update policy
    \[
    \theta_i^{k+1} = \sum_{j \in \mathcal{N}(i)} W_{i,j} \theta_j^k + \alpha_k \nabla L_i(\theta_i^k)
    \]
\[ \theta^{k+1}_i = \sum_{j \in \mathcal{N}(i)} W_{i,j} \theta^k_j + \alpha_k \nabla L_i(\theta^k_i) \]

For small enough step sizes \( \alpha_k \), after \( k \) iterations we have

\[ \left\| \frac{1}{N} \sum_{i=1}^{N} \nabla L_i(\theta^k_i) \right\|^2 \leq O \left( \frac{1}{\sqrt{k}} + \frac{C_g}{k} \right) \]

- Convergence to stationary point (not global max)
- Graph properties expressed in \( C_g \)
- Other constants come from \( \lambda, N \), and MDP properties
Algorithm: Mathematical Guarantees

$$\theta^{k+1}_i = \sum_{j \in \mathcal{N}(i)} W_{i,j} \theta^k_j + \alpha_k \nabla L_i(\theta^k_i)$$

If common states are “equally explored” across environments, then after $k$ iterations

$$\max_j \left\{ \sum_{i=1}^{N} L_i(\theta^*) - L_i(\theta^k_j) \right\} \leq \epsilon \quad \text{when} \quad k \geq \frac{C}{\epsilon^2}$$

- Convergence to **global optimum**
- Requires careful choice of regularization parameter $\lambda$
- “Equal exploration” hard to verify
- Can make this stochastic, but not with finite-sample guarantee
MultiTask RL4 – Unconflicted Goals

Simulation: GridWorld
MultiTask RL4 – Resolvable Conflicted Goals 1
MultiTask RL4 – Un-resolvable Conflicted Goals 1

Simulation: GridWorld
Simulation: Drones in D-PEDRA
Simulation: Drones in D-PEDRA

![Graphs showing Mean Safe Flight in different environments](image)

### Table 1: MSF of the learned policy

<table>
<thead>
<tr>
<th>Policy</th>
<th>Env0</th>
<th>Env1</th>
<th>Env2</th>
<th>Env3</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA-0</td>
<td>15.9</td>
<td>4.5</td>
<td>4.1</td>
<td>3.6</td>
<td>28.1</td>
</tr>
<tr>
<td>SA-1</td>
<td>3.0</td>
<td>55.4</td>
<td>9.7</td>
<td>8.1</td>
<td>76.2</td>
</tr>
<tr>
<td>SA-2</td>
<td>1.5</td>
<td>0.8</td>
<td>21.1</td>
<td>2.0</td>
<td>25.4</td>
</tr>
<tr>
<td>SA-3</td>
<td>2.3</td>
<td>0.8</td>
<td>8.6</td>
<td>40.1</td>
<td>51.8</td>
</tr>
<tr>
<td>DCPG</td>
<td>25.2</td>
<td>67.9</td>
<td>40.5</td>
<td>61.8</td>
<td>195.4</td>
</tr>
<tr>
<td>Random</td>
<td>2.5</td>
<td>3.9</td>
<td>4.7</td>
<td>3.7</td>
<td>14.8</td>
</tr>
</tbody>
</table>

Interesting, unexplained result:

*Learning a joint policy is easier than learning individual policies*
Thank you!

References:


