Reinforcement Learning for Stochastic Networks

Exploiting Exponential Families to Tackle Nonconvexity

Céline Comte and Matthieu Jonckheere
CNRS and LAAS

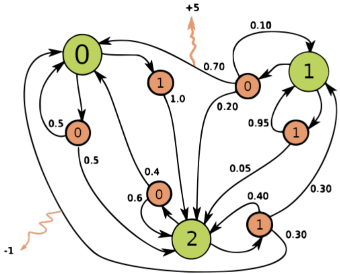
Jaron Sanders and Albert Senen-Cerda
Eindhoven University of Technology

November 20, 2023

Workshop on restless bandits, index policies
and applications in reinforcement learning

Reinforcement learning

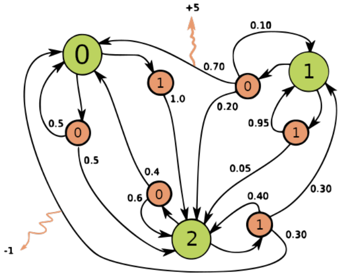
- **Markov decision process** (MDP)



Source: Wikipedia (modified)

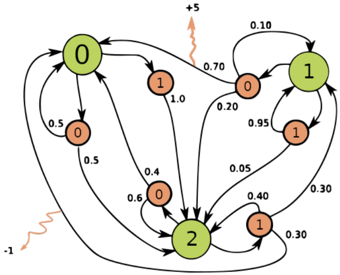
Reinforcement learning

- **Markov decision process** (MDP) with
  - State-action-reward sequence $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, \ldots$



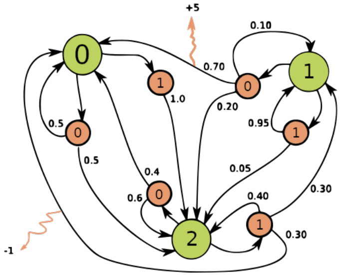
Source: Wikipedia (modified)

- **Markov decision process** (MDP) with
  - State-action-reward sequence $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, \dots$
  - Environment $P(s', r | s, a) = \mathbb{P}\left[\begin{matrix} S_{t+1}{=}s' \\ R_{t+1}{=}r \end{matrix} \middle| \begin{matrix} S_t{=}s \\ A_t{=}a \end{matrix} \right]$



Source: Wikipedia (modified)

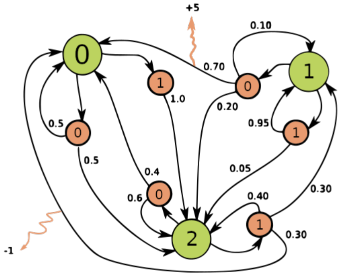
- **Markov decision process** (MDP) with
  - State-action-reward sequence $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, \dots$
  - Environment $P(s', r|s, a) = \mathbb{P}\left[\begin{smallmatrix} S_{t+1}=s' \\ R_{t+1}=r \end{smallmatrix} \middle| \begin{smallmatrix} S_t=s \\ A_t=a \end{smallmatrix}\right]$
  - Policy parameterization $\pi(a|s,\theta) = \mathbb{P}[A_t = a \,|\, S_t = s]$



Source: Wikipedia (modified)

- **Markov decision process** (MDP) with
  - State-action-reward sequence $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, \ldots$
  - Environment $P(s', r|s, a) = \mathbb{P}\left[\begin{smallmatrix} S_{t+1}=s' \\ R_{t+1}=r \end{smallmatrix} \middle| \begin{smallmatrix} S_t=s \\ A_t=a \end{smallmatrix}\right]$
  - Policy parameterization $\pi(a|s,\theta) = \mathbb{P}[A_t = a \,|\, S_t = s]$
- **Goal:** Find a θ that maximizes the **average reward rate**

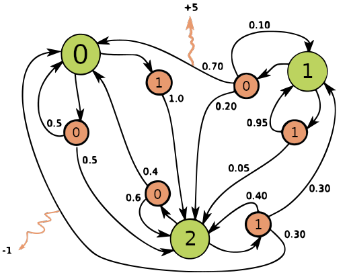
$$J(\theta) = \lim_{T \to +\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[R_t] = \mathbb{E}[R],$$



Source: Wikipedia (modified)

Reinforcement learning

- **Markov decision process** (MDP) with
  - State-action-reward sequence $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, \dots$
  - Environment $P(s', r | s, a) = \mathbb{P}\left[\begin{smallmatrix} S_{t+1}=s' \\ R_{t+1}=r \end{smallmatrix} \middle| \begin{smallmatrix} S_t=s \\ A_t=a \end{smallmatrix}\right]$
  - Policy parameterization $\pi(a|s,\theta) = \mathbb{P}[A_t = a \,|\, S_t = s]$



Source: Wikipedia (modified)

- **Goal:** Find a θ that maximizes the **average reward rate**

$$J(\theta) = \lim_{T \to +\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[R_t] = \mathbb{E}[R],$$

- **Stationary triplet** $(S, A, R) \sim \lim_{t \to +\infty} (S_t, A_t, R_{t+1})$:

$$\mathbb{P}[S = s, A = a, R = r] = p(s|\theta)\pi(a|s,\theta) \sum_{s'} P(s', r|s, a).$$

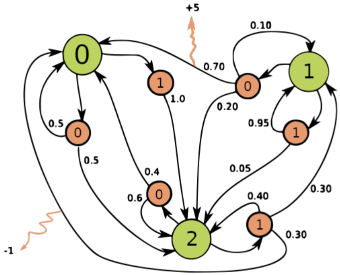
Reinforcement learning

- **Markov decision process** (MDP) with
  - State-action-reward sequence $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, \dots$
  - Environment $P(s', r|s, a) = \mathbb{P}\left[\begin{smallmatrix} S_{t+1}=s' \\ R_{t+1}=r \end{smallmatrix} \middle| \begin{smallmatrix} S_t=s \\ A_t=a \end{smallmatrix}\right]$
  - Policy parameterization $\pi(a|s,\theta) = \mathbb{P}[A_t = a \,|\, S_t = s]$
- **Goal:** Find a θ that maximizes the **average reward rate**

$$J(\theta) = \lim_{T \to +\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[R_t] = \mathbb{E}[R],$$

- **Stationary triplet** $(S, A, R) \sim \lim_{t \to +\infty} (S_t, A_t, R_{t+1})$:

$$\mathbb{P}[S = s, A = a, R = r] = \underbrace{\boxed{p(s|\theta)}}_{\substack{\text{Stationary distribution of} \\ (S_t,\, t \ge 0) \text{ under } \pi(a|s,\theta)}} \pi(a|s,\theta) \sum_{s'} P(s', r|s, a).$$



Source: Wikipedia (modified)

Policy-gradient algorithms

- Typical **policy-gradient algorithm**:

```
Initialize $S_0$ and $\Theta_0$
for $t = 0, 1, 2, \ldots$ do
    Sample $A_t \sim \pi(\cdot|S_t, \Theta_t)$
    Take action $A_t$ and observe $S_{t+1}, R_{t+1}$
    Estimate $[\![\nabla J(\Theta_t)]\!]$ using the history $S_0, \Theta_0, A_0, R_1, \ldots, S_t, \Theta_t, A_t, R_{t+1}, S_{t+1}$
    Update $\Theta_{t+1} \leftarrow \Theta_t + \alpha[\![\nabla J(\Theta_t)]\!]$
end for
```

Policy-gradient algorithms

- Typical **policy-gradient algorithm**:

  1: Initialize S_0 and Θ_0
  2: **for** $t = 0, 1, 2, \ldots$ **do**
  3: Sample $A_t \sim \pi(\cdot | S_t, \Theta_t)$
  4: Take action A_t and observe S_{t+1}, R_{t+1}
  5: Estimate $[\![\nabla J(\Theta_t)]\!]$ using the history $S_0, \Theta_0, A_0, R_1, \ldots, S_t, \Theta_t, A_t, R_{t+1}, S_{t+1}$
  6: Update $\Theta_{t+1} \leftarrow \Theta_t + \alpha [\![\nabla J(\Theta_t)]\!]$
  7: **end for**

$[\![\cdot]\!]$ = estimate of $\cdot$
∇ = gradient with respect to θ

Policy-gradient algorithms

- Typical **policy-gradient algorithm**:

$[\![\cdot]\!]$ = estimate of $\cdot$
∇ = gradient with respect to θ

1: Initialize S_0 and Θ_0
2: **for** $t = 0, 1, 2, \dots$ **do**
3: Sample $A_t \sim \pi(\cdot | S_t, \Theta_t)$
4: Take action A_t and observe S_{t+1}, R_{t+1}
5: Estimate $[\![\nabla J(\Theta_t)]\!]$ using the history $S_0, \Theta_0, A_0, R_1, \dots, S_t, \Theta_t, A_t, R_{t+1}, S_{t+1}$ How?
6: Update $\Theta_{t+1} \leftarrow \Theta_t + \alpha [\![\nabla J(\Theta_t)]\!]$
7: **end for**

Policy-gradient algorithms

$[\![\cdot]\!]$ = estimate of $\cdot$
∇ = gradient with respect to θ

- Typical **policy-gradient algorithm**:

```
Initialize S_0 and Θ_0
for t = 0, 1, 2, ... do
    Sample A_t ~ π(·|S_t, Θ_t)
    Take action A_t and observe S_{t+1}, R_{t+1}
    Estimate [[∇J(Θ_t)]] using the history S_0, Θ_0, A_0, R_1, ..., S_t, Θ_t, A_t, R_{t+1}, S_{t+1}
    Update Θ_{t+1} ← Θ_t + α[[∇J(Θ_t)]]
end for
```

How?

- **Actor-critic** applies the policy-gradient theorem (Sutton and Barto, 2018):

$$[\![\nabla J(\Theta_t)]\!] \leftarrow (R_{t+1} - [\![\mathbb{E}[R]]\!] + [\![v(S_{t+1})]\!] - [\![v(S_t)]\!])\nabla \log \pi(A_t|S_t, \Theta_t).$$

Policy-gradient algorithms

- Typical **policy-gradient algorithm**:

  Initialize S_0 and Θ_0
  for $t = 0, 1, 2, \dots$ **do**
      Sample $A_t \sim \pi(\cdot|S_t, \Theta_t)$
      Take action A_t and observe S_{t+1}, R_{t+1}
      $\boxed{\text{Estimate } [\![\nabla J(\Theta_t)]\!] \text{ using the history } S_0, \Theta_0, A_0, R_1, \dots, S_t, \Theta_t, A_t, R_{t+1}, S_{t+1}}$ How?
      Update $\Theta_{t+1} \leftarrow \Theta_t + \alpha [\![\nabla J(\Theta_t)]\!]$
  end for

$[\![\cdot]\!]$ = estimate of $\cdot$
∇ = gradient with respect to θ

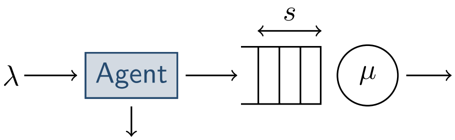
- **Actor-critic** applies the policy-gradient theorem (Sutton and Barto, 2018):

$$[\![\nabla J(\Theta_t)]\!] \leftarrow (R_{t+1} - [\![\mathbb{E}[R]]\!] + [\![v(S_{t+1})]\!] - [\![v(S_t)]\!]) \nabla \log \pi(A_t|S_t, \Theta_t).$$

- Can we do better by exploiting the **system structure**?

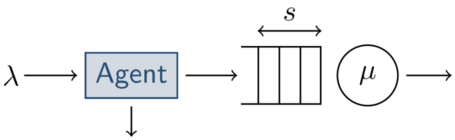
Example: M/M/1 queue with admission control

- Arrival rate $\lambda > 0$, service rate $\mu > 0$
- State: queue length $s \in \{0, 1, 2, \dots\}$
- Actions: accept or reject
- Admission reward α per job
- Holding cost rate η per job per time unit



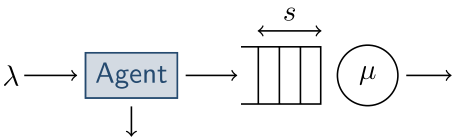
Example: M/M/1 queue with admission control

- Arrival rate $\lambda > 0$, service rate $\mu > 0$
- State: queue length $s \in \{0, 1, 2, \ldots\}$
- Actions: accept or reject
- Admission reward α per job
- Holding cost rate η per job per time unit
- Policy $\pi(\text{accept}|s, \theta) = \dfrac{1}{1 + e^{-\theta_s}}$



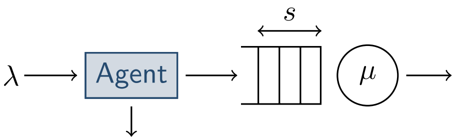
Example: M/M/1 queue with admission control

- Arrival rate $\lambda > 0$, service rate $\mu > 0$
- State: queue length $s \in \{0, 1, 2, \dots\}$
- Actions: accept or reject
- Admission reward α per job
- Holding cost rate η per job per time unit
- Policy $\pi(\text{accept}|s, \theta) = \dfrac{1}{1 + e^{-\theta_{\min(s,k)}}}$ with parameter vector $\theta = (\theta_0, \theta_1, \dots, \theta_k)$



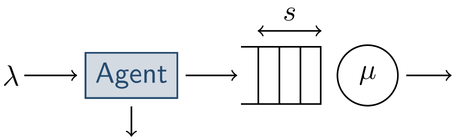
Example: M/M/1 queue with admission control

- Arrival rate $\lambda > 0$, service rate $\mu > 0$
- State: queue length $s \in \{0, 1, 2, \dots\}$
- Actions: accept or reject
- Admission reward α per job
- Holding cost rate η per job per time unit



- Policy $\pi(\text{accept}|s,\theta) = \dfrac{1}{1 + e^{-\theta_{\min(s,k)}}}$ with parameter vector $\theta = (\theta_0, \theta_1, \dots, \theta_k)$
- Average reward rate $J(\theta) = \alpha \times \left(\sum_{s=0}^{+\infty} p(s|\theta)\pi(\text{accept}|s,\theta) \right) - \eta \times \left(\sum_{s=0}^{+\infty} p(s|\theta) s \right) \times \dfrac{1}{\lambda}$

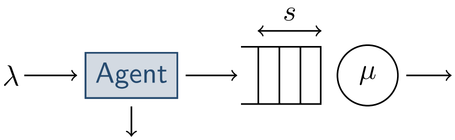
Example: M/M/1 queue with admission control



- Arrival rate $\lambda > 0$, service rate $\mu > 0$
- State: queue length $s \in \{0, 1, 2, \dots\}$
- Actions: accept or reject
- Admission reward α per job
- Holding cost rate η per job per time unit
- Policy $\pi(\text{accept}|s,\theta) = \dfrac{1}{1 + e^{-\theta_{\min(s,k)}}}$ with parameter vector $\theta = (\theta_0, \theta_1, \dots, \theta_k)$
- Average reward rate $J(\theta) = \alpha \times \underbrace{\left(\sum_{s=0}^{+\infty} p(s|\theta)\pi(\text{accept}|s,\theta) \right)}_{\text{Probability of accepting a job}} - \eta \times \left(\sum_{s=0}^{+\infty} p(s|\theta) s \right) \times \dfrac{1}{\lambda}$

Example: M/M/1 queue with admission control

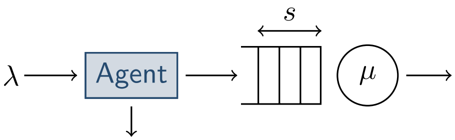
- Arrival rate $\lambda > 0$, service rate $\mu > 0$
- State: queue length $s \in \{0, 1, 2, \dots\}$
- Actions: accept or reject
- Admission reward α per job
- Holding cost rate η per job per time unit



- Policy $\pi(\text{accept}|s,\theta) = \dfrac{1}{1 + e^{-\theta_{\min(s,k)}}}$ with parameter vector $\theta = (\theta_0, \theta_1, \dots, \theta_k)$
- Average reward rate $J(\theta) = \alpha \times \underbrace{\left(\sum_{s=0}^{+\infty} p(s|\theta)\pi(\text{accept}|s,\theta) \right)}_{\text{Probability of accepting a job}} - \eta \times \underbrace{\left(\sum_{s=0}^{+\infty} p(s|\theta) s \right)}_{\text{Mean queue size}} \times \dfrac{1}{\lambda}$

Example: M/M/1 queue with admission control

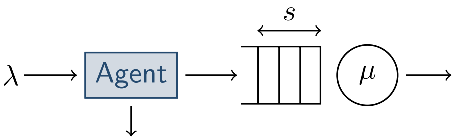
- Arrival rate $\lambda > 0$, service rate $\mu > 0$
- State: queue length $s \in \{0, 1, 2, \dots\}$
- Actions: accept or reject
- Admission reward α per job
- Holding cost rate η per job per time unit



- Policy $\pi(\text{accept}|s, \theta) = \dfrac{1}{1 + e^{-\theta_{\min(s,k)}}}$ with parameter vector $\theta = (\theta_0, \theta_1, \dots, \theta_k)$
- Average reward rate $J(\theta) = \alpha \times \left(\sum_{s=0}^{+\infty} p(s|\theta)\pi(\text{accept}|s, \theta) \right) - \eta \times \left(\sum_{s=0}^{+\infty} p(s|\theta) s \right) \times \dfrac{1}{\lambda}$
- Stationary distribution $p(s|\theta) \propto \prod_{i=0}^{k-1} \left(\dfrac{\lambda}{\mu} \pi(\text{accept}|i, \theta) \right)^{1_{\{s \geq i\}}} \left(\dfrac{\lambda}{\mu} \pi(\text{accept}|k, \theta) \right)^{\max(s-k,0)}$

Example: M/M/1 queue with admission control

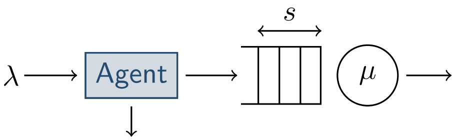
- Arrival rate $\lambda > 0$, service rate $\mu > 0$
- State: queue length $s \in \{0, 1, 2, \dots\}$
- Actions: accept or reject
- Admission reward α per job
- Holding cost rate η per job per time unit



- Policy $\pi(\text{accept}|s,\theta) = \dfrac{1}{1 + e^{-\theta_{\min(s,k)}}}$ with parameter vector $\theta = (\theta_0, \theta_1, \dots, \theta_k)$
- Average reward rate $J(\theta) = \alpha \times \left(\sum_{s=0}^{+\infty} p(s|\theta)\pi(\text{accept}|s,\theta) \right) - \eta \times \left(\sum_{s=0}^{+\infty} p(s|\theta) s \right) \times \dfrac{1}{\lambda}$
- Stationary distribution $p(s|\theta) \propto \prod_{i=0}^{k-1} \left(\boxed{\dfrac{\lambda}{\mu}\pi(\text{accept}|i,\theta)} \right)^{1_{\{s \geq i\}}} \left(\boxed{\dfrac{\lambda}{\mu}\pi(\text{accept}|k,\theta)} \right)^{\max(s-k,0)}$

Depends on θ

Example: M/M/1 queue with admission control



- Arrival rate $\lambda > 0$, service rate $\mu > 0$
- State: queue length $s \in \{0, 1, 2, \dots\}$
- Actions: accept or reject
- Admission reward α per job
- Holding cost rate η per job per time unit
- Policy $\pi(\text{accept}|s,\theta) = \dfrac{1}{1 + e^{-\theta_{\min(s,k)}}}$ with parameter vector $\theta = (\theta_0, \theta_1, \dots, \theta_k)$
- Average reward rate $J(\theta) = \alpha \times \left(\sum_{s=0}^{+\infty} p(s|\theta)\pi(\text{accept}|s,\theta) \right) - \eta \times \left(\sum_{s=0}^{+\infty} p(s|\theta) s \right) \times \dfrac{1}{\lambda}$
- Stationary distribution $p(s|\theta) \propto \prod_{i=0}^{k-1} \left(\boxed{\frac{\lambda}{\mu}\pi(\text{accept}|i,\theta)} \right)^{\boxed{1_{\{s \geq i\}}}} \left(\boxed{\frac{\lambda}{\mu}\pi(\text{accept}|k,\theta)} \right)^{\boxed{\max(s-k,0)}}$

Depends on s

Depends on θ

Our approach

- We consider MDPs and policy parameterizations $\pi(a|s,\theta)$ such that the Markov chain $(S_t, t \geq 0)$ has a **product-form stationary distribution** $p(s|\theta)$

Our approach

- We consider MDPs and policy parameterizations $\pi(a|s,\theta)$ such that the Markov chain $(S_t, t \geq 0)$ has a **product-form stationary distribution** $p(s|\theta)$
- We exploit the product form to introduce a new **policy-gradient algorithm**

Our approach

- We consider MDPs and policy parameterizations $\pi(a|s,\theta)$ such that the Markov chain $(S_t, t \geq 0)$ has a **product-form stationary distribution** $p(s|\theta)$
- We exploit the product form to introduce a new **policy-gradient algorithm**
- We show that this algorithm has nice **convergence properties**

Our approach

- We consider MDPs and policy parameterizations $\pi(a|s,\theta)$ such that the Markov chain $(S_t, t \geq 0)$ has a **product-form stationary distribution** $p(s|\theta)$
- We exploit the product form to introduce a new **policy-gradient algorithm**
- We show that this algorithm has nice **convergence properties**

- Main contributions:
  1. Product-form distributions as exponential families
  2. Score-aware gradient estimator (SAGE)
  3. SAGE-based policy-gradient algorithm
  4. Nonconvex convergence result

① Product-form distributions as exponential families

- **Product-form** distribution

$$p(s|\theta) = \frac{1}{Z(\theta)} \prod_{i=1}^{n} \rho_i(\theta)^{x_i(s)}$$

① Product-form distributions as exponential families

- **Product-form** distribution

$$p(s|\theta) = \frac{1}{Z(\theta)} \prod_{i=1}^{n} \boxed{\rho_i(\theta)}^{x_i(s)}$$

Depends on θ

① Product-form distributions as exponential families

- **Product-form** distribution

Depends on s

$$p(s|\theta) = \frac{1}{Z(\theta)} \prod_{i=1}^{n} \boxed{\rho_i(\theta)}^{\boxed{x_i(s)}}$$

Depends on θ

① Product-form distributions as exponential families

- **Product-form** distribution

$$p(s|\theta) = \frac{1}{Z(\theta)} \prod_{i=1}^{n} \rho_i(\theta)^{x_i(s)}$$

- **Feature function** $x = (x_1, x_2, \dots, x_n)$

① Product-form distributions as exponential families

- **Product-form** distribution

$$p(s|\theta) = \frac{1}{Z(\theta)} \prod_{i=1}^{n} \rho_i(\theta)^{x_i(s)}$$

- **Feature function** $x = (x_1, x_2, \dots, x_n)$
- **Load function** $\rho = (\rho_1, \rho_2, \dots, \rho_n)$

① Product-form distributions as exponential families

- **Product-form** distribution

$$p(s|\theta) = \frac{1}{Z(\theta)} \prod_{i=1}^{n} \rho_i(\theta)^{x_i(s)}$$

- **Feature function** $x = (x_1, x_2, \dots, x_n)$
- **Load function** $\rho = (\rho_1, \rho_2, \dots, \rho_n)$
- **Partition function** Z

$$Z(\theta) = \sum_{s} \prod_{i=1}^{n} \rho_i(\theta)^{x_i(s)}$$

① Product-form distributions as exponential families

- **Product-form** distribution ⟶ **Exponential family** of distributions

$$p(s|\theta) = \frac{1}{Z(\theta)} \prod_{i=1}^{n} \rho_i(\theta)^{x_i(s)}$$

$$\log p(s|\theta) = \log \rho(\theta)^\intercal x(s) - \log Z(\theta)$$

- **Feature function** $x = (x_1, x_2, \ldots, x_n)$
- **Load function** $\rho = (\rho_1, \rho_2, \ldots, \rho_n)$
- **Partition function** Z

$$Z(\theta) = \sum_s \prod_{i=1}^{n} \rho_i(\theta)^{x_i(s)}$$

① Product-form distributions as exponential families

- **Product-form** distribution ⟶ **Exponential family** of distributions

$$p(s|\theta) = \frac{1}{Z(\theta)} \prod_{i=1}^{n} \rho_i(\theta)^{x_i(s)}$$

$$\log p(s|\theta) = \log \rho(\theta)^\intercal x(s) - \log Z(\theta)$$

- **Feature function** $x = (x_1, x_2, \dots, x_n)$ ⟶ **Feature function** $x = (x_1, x_2, \dots, x_n)$
- **Load function** $\rho = (\rho_1, \rho_2, \dots, \rho_n)$
- **Partition function** Z

$$Z(\theta) = \sum_{s} \prod_{i=1}^{n} \rho_i(\theta)^{x_i(s)}$$

① Product-form distributions as exponential families

- **Product-form** distribution ⟶ **Exponential family** of distributions

$$p(s|\theta) = \frac{1}{Z(\theta)} \prod_{i=1}^{n} \rho_i(\theta)^{x_i(s)} \qquad \log p(s|\theta) = \log \rho(\theta)^\intercal x(s) - \log Z(\theta)$$

- **Feature function** $x = (x_1, x_2, \dots, x_n)$ ⟶ **Feature function** $x = (x_1, x_2, \dots, x_n)$
- **Load function** $\rho = (\rho_1, \rho_2, \dots, \rho_n)$ ⟶ **Log-load function** $\log \rho = (\log \rho_1, \dots, \log \rho_n)$
- **Partition function** Z

$$Z(\theta) = \sum_s \prod_{i=1}^{n} \rho_i(\theta)^{x_i(s)}$$

① Product-form distributions as exponential families

- **Product-form** distribution ⟶ **Exponential family** of distributions

$$p(s|\theta) = \frac{1}{Z(\theta)} \prod_{i=1}^{n} \rho_i(\theta)^{x_i(s)} \qquad \log p(s|\theta) = \log \rho(\theta)^\intercal x(s) - \log Z(\theta)$$

- **Feature function** $x = (x_1, x_2, \ldots, x_n)$ ⟶ **Feature function** $x = (x_1, x_2, \ldots, x_n)$
- **Load function** $\rho = (\rho_1, \rho_2, \ldots, \rho_n)$ ⟶ **Log-load function** $\log \rho = (\log \rho_1, \ldots, \log \rho_n)$
- **Partition function** Z ⟶ **Log-partition function** $\log Z$

$$Z(\theta) = \sum_s \prod_{i=1}^{n} \rho_i(\theta)^{x_i(s)} \qquad \log Z(\theta) = \log\left(\sum_s \mathrm{e}^{\log \rho(\theta)^\intercal x(s)}\right)$$

② Score-aware gradient estimator (SAGE)

- The **score** is the gradient of the log-likelihood with respect to the parameter vector:

$$\text{“Likelihood”} = p(s|\theta) \quad \to \quad \text{“Score”} = \nabla \log p(s|\theta).$$

② Score-aware gradient estimator (SAGE)

- The **score** is the gradient of the log-likelihood with respect to the parameter vector:

$$\text{“Likelihood”} = p(s|\theta) \quad \to \quad \text{“Score”} = \nabla \log p(s|\theta).$$

Theorem

Recalling that $(S, A, R) \sim$ *stationary distribution of* $((S_t, A_t, R_{t+1}), t \geq 0)$*, we have*

$$\begin{aligned} \nabla \log p(s|\theta) &= \mathrm{D} \log \rho(\theta)^\intercal (x(s) - \mathbb{E}[x(S)]), \\ \nabla J(\theta) &= \mathrm{D} \log \rho(\theta)^\intercal \mathrm{Cov}[R, x(S)] + \mathbb{E}[R \, \nabla \log \pi(A|S, \theta)]. \end{aligned}$$

② Score-aware gradient estimator (SAGE)

- The **score** is the gradient of the log-likelihood with respect to the parameter vector:

$$\text{“Likelihood”} = p(s|\theta) \quad \rightarrow \quad \text{“Score”} = \nabla \log p(s|\theta).$$

Theorem

Recalling that $(S, A, R) \sim$ *stationary distribution of* $((S_t, A_t, R_{t+1}), t \geq 0)$, *we have*

$$\begin{aligned} \nabla \log p(s|\theta) &= \mathrm{D} \log \rho(\theta)^\intercal (x(s) - \mathbb{E}[x(S)]), \\ \nabla J(\theta) &= \mathrm{D} \log \rho(\theta)^\intercal \mathrm{Cov}[R, x(S)] + \mathbb{E}[R \nabla \log \pi(A|S, \theta)]. \end{aligned}$$

- **Main take-away:** If we can evaluate $\mathrm{D} \log \rho(\theta)$, this gives us an estimator for $\nabla J(\theta)$.

③ SAGE-based policy-gradient algorithm

- Typical **policy-gradient algorithm**:

Initialize S_0 and Θ_0
for $t = 0, 1, 2, \dots$ **do**
    Sample $A_t \sim \pi(\cdot | S_t, \Theta_t)$
    Take action A_t and observe S_{t+1}, R_{t+1}
    Estimate $[\![\nabla J(\Theta_t)]\!]$ using the history $S_0, \Theta_0, A_0, R_1, \dots, S_t, \Theta_t, A_t, R_{t+1}, S_{t+1}$ How?
    Update $\Theta_{t+1} \leftarrow \Theta_t + \alpha [\![\nabla J(\Theta_t)]\!]$
end for

③ SAGE-based policy-gradient algorithm

- Typical **policy-gradient algorithm**:

```
Initialize S_0 and Θ_0
for t = 0, 1, 2, ... do
    Sample A_t ~ π(·|S_t, Θ_t)
    Take action A_t and observe S_{t+1}, R_{t+1}
    Estimate ⟦∇J(Θ_t)⟧ using the history S_0, Θ_0, A_0, R_1, ..., S_t, Θ_t, A_t, R_{t+1}, S_{t+1}    How?
    Update Θ_{t+1} ← Θ_t + α⟦∇J(Θ_t)⟧
end for
```

- **Actor-critic** applies the policy-gradient theorem (Sutton and Barto, 2018):

$$[\![\nabla J(\Theta_t)]\!] \leftarrow (R_{t+1} - [\![\mathbb{E}[R]]\!] + [\![v(S_{t+1})]\!] - [\![v(S_t)]\!])\nabla \log \pi(A_t|S_t, \Theta_t).$$

③ SAGE-based policy-gradient algorithm

- Typical **policy-gradient algorithm**:

  1: Initialize S_0 and Θ_0
  2: **for** $t = 0, 1, 2, \ldots$ **do**
  3: Sample $A_t \sim \pi(\cdot|S_t, \Theta_t)$
  4: Take action A_t and observe S_{t+1}, R_{t+1}
  5: Estimate $[\![\nabla J(\Theta_t)]\!]$ using the history $S_0, \Theta_0, A_0, R_1, \ldots, S_t, \Theta_t, A_t, R_{t+1}, S_{t+1}$ How?
  6: Update $\Theta_{t+1} \leftarrow \Theta_t + \alpha [\![\nabla J(\Theta_t)]\!]$
  7: **end for**

- **Actor-critic** applies the policy-gradient theorem (Sutton and Barto, 2018):

$$[\![\nabla J(\Theta_t)]\!] \leftarrow (R_{t+1} - [\![\mathbb{E}[R]]\!] + [\![v(S_{t+1})]\!] - [\![v(S_t)]\!]) \nabla \log \pi(A_t|S_t, \Theta_t).$$

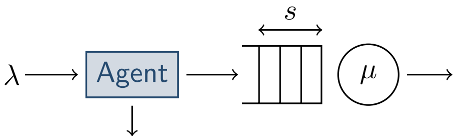
- We instead estimate $[\![\nabla J(\Theta_t)]\!]$ with a **score-aware gradient estimator (SAGE)**:

$$[\![\nabla J(\Theta_t)]\!] \leftarrow \mathrm{D} \log \rho(\Theta_t)^\intercal [\![\mathrm{Cov}[R, x(S)]]\!] + [\![\mathbb{E}[R \nabla \log \pi(A|S, \Theta_t)]]\!].$$

Example: M/M/1 queue with admission control

Stable case

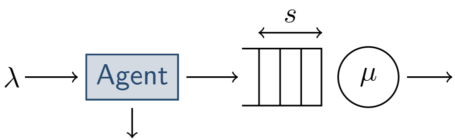
- Arrival rate $\lambda = 0.7$, service rate $\mu = 1$
- Admission reward $\alpha = 5$
- Holding cost rate $\eta = 1$
- Initial policy $\pi(\Theta_0) = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2})$
- Optimal policy $\pi_0(\theta^\star) = \pi_1(\theta^\star) = \pi_2(\theta^\star) = 1$ and $\pi_3(\theta^\star) = 0$.



Example: M/M/1 queue with admission control

Stable case

- Arrival rate $\lambda = 0.7$, service rate $\mu = 1$
- Admission reward $\alpha = 5$
- Holding cost rate $\eta = 1$
- Initial policy $\pi(\Theta_0) = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2})$
- Optimal policy $\pi_0(\theta^\star) = \pi_1(\theta^\star) = \pi_2(\theta^\star) = 1$ and $\pi_3(\theta^\star) = 0$.



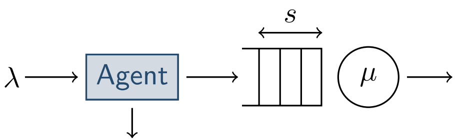
Possibly-unstable case

- Arrival rate $\lambda = 1.4$, service rate $\mu = 1$
- Admission reward $\alpha = 5$
- Holding cost rate $\eta = 1$
- Initial policy $\pi(\Theta_0) = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2})$
- Optimal policy $\pi_0(\theta^\star) = \pi_1(\theta^\star) = 1$ and $\pi_2(\theta^\star) = \pi_3(\theta^\star) = 0$.

Example: M/M/1 queue with admission control

Stable case

- Arrival rate $\lambda = 0.7$, service rate $\mu = 1$
- Admission reward $\alpha = 5$
- Holding cost rate $\eta = 1$
- Initial policy $\pi(\Theta_0) = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2})$
- Optimal policy $\pi_0(\theta^\star) = \pi_1(\theta^\star) = \pi_2(\theta^\star) = 1$ and $\pi_3(\theta^\star) = 0$.



Possibly-unstable case

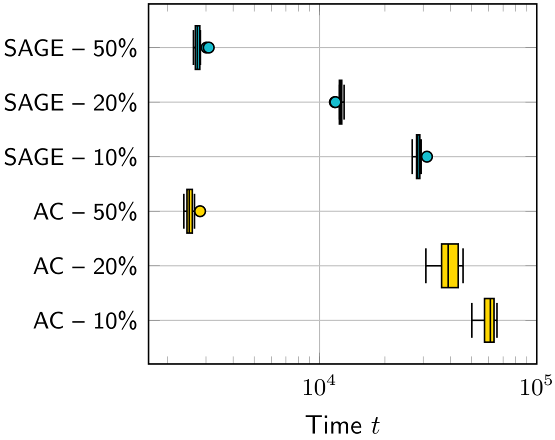
- Arrival rate $\lambda = 1.4$, service rate $\mu = 1$
- Admission reward $\alpha = 5$
- Holding cost rate $\eta = 1$
- Initial policy $\pi(\Theta_0) = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2})$
- Optimal policy $\pi_0(\theta^\star) = \pi_1(\theta^\star) = 1$ and $\pi_2(\theta^\star) = \pi_3(\theta^\star) = 0$.

Simulation setup

- 10^6 steps
- Convergence time T: $J(\Theta_t) > J(\theta^\star) - \epsilon$ for each $t \in \{T, T+1, \dots, 10^6\}$

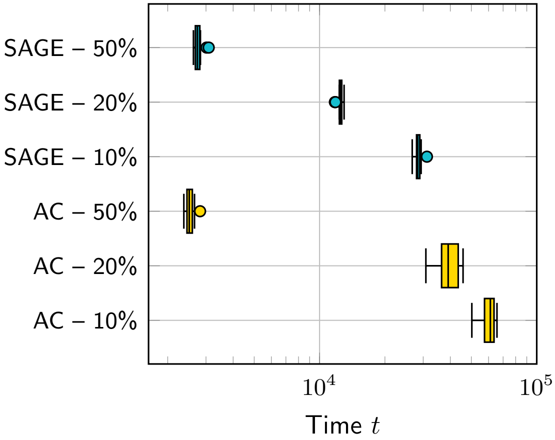
Example: M/M/1 queue with admission control

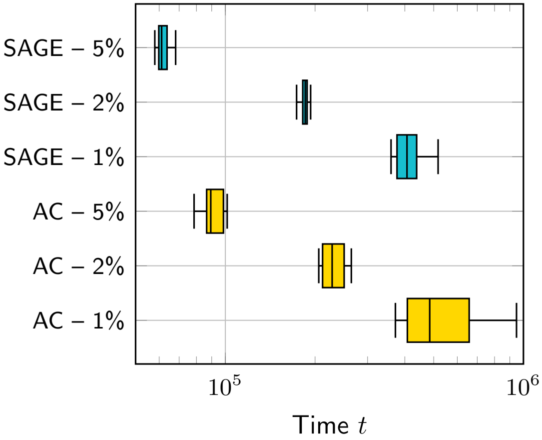
Stable case – Convergence times



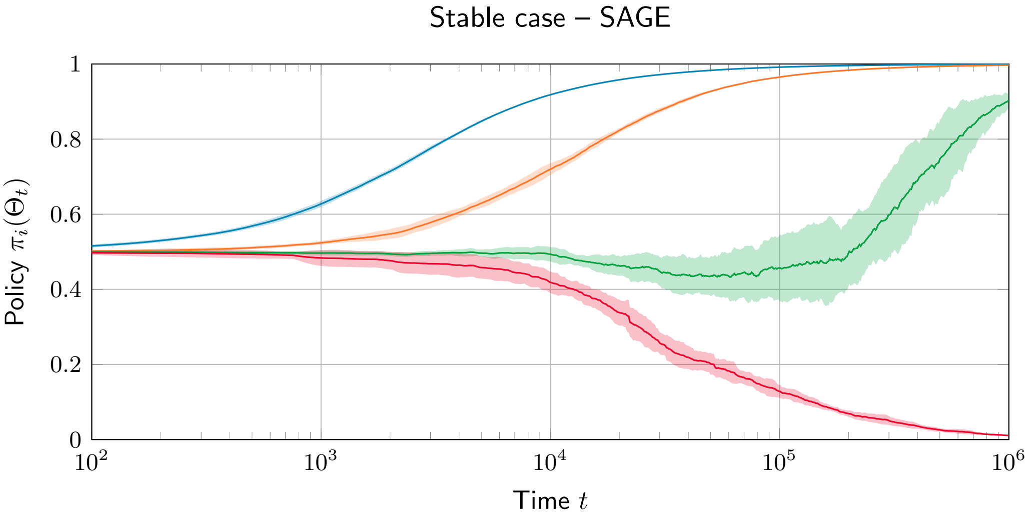
Example: M/M/1 queue with admission control

Stable case – Convergence times

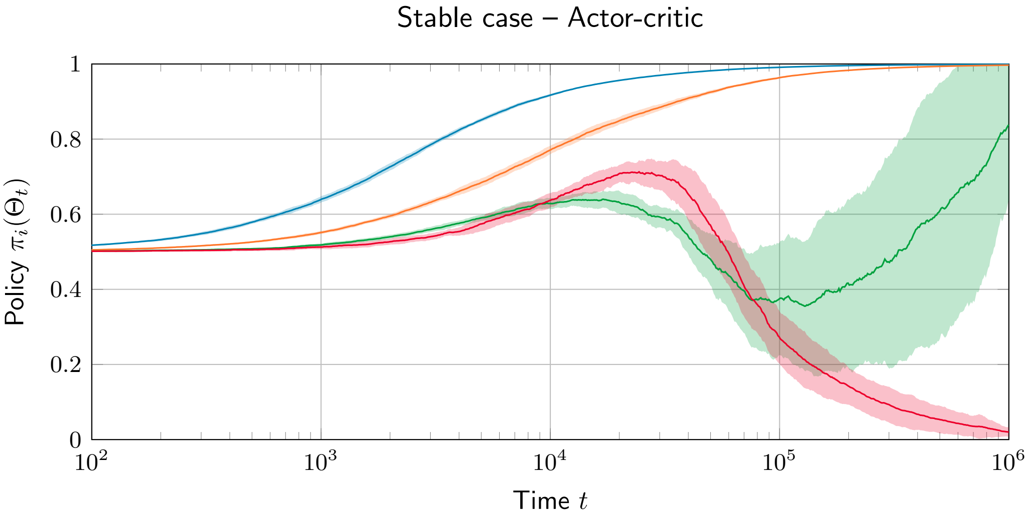




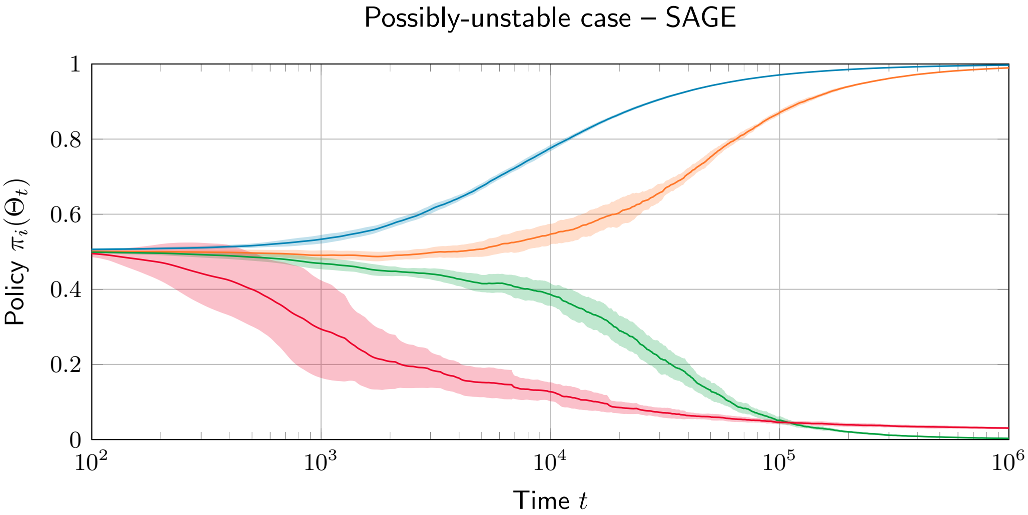
Example: M/M/1 queue with admission control



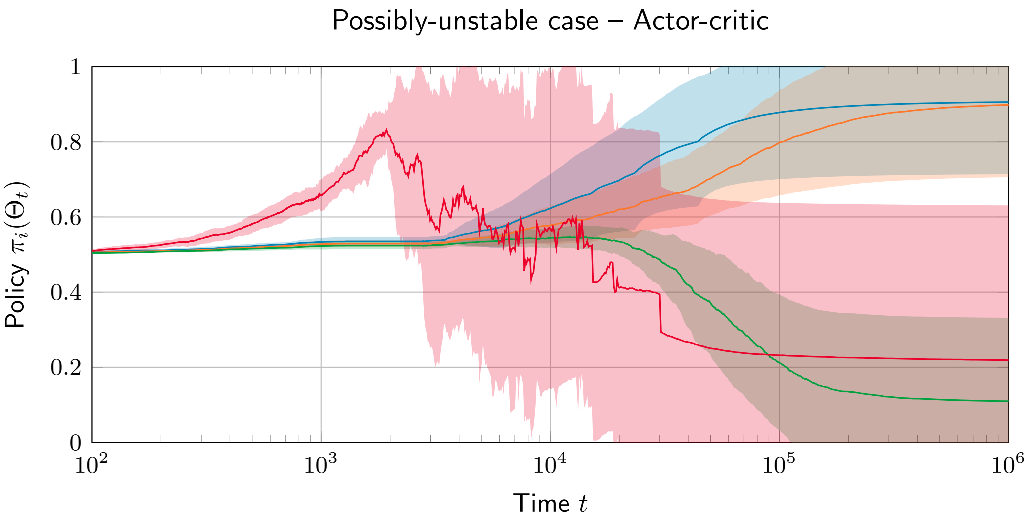
Example: M/M/1 queue with admission control



Example: M/M/1 queue with admission control



Example: M/M/1 queue with admission control



④ Local convergence result

Sketch of Theorem

Under additional assumptions, a batch variant of the algorithm that starts in a basin of attraction of a global maximizer will converge to a global maximizer with large probability.

Sketch of Theorem

Under additional assumptions, a batch variant of the algorithm that starts in a basin of attraction of a global maximizer will converge to a global maximizer with large probability.

④ Local convergence result

Sketch of Theorem

Under additional assumptions, a <u>*batch variant*</u> *of the algorithm that starts in a basin of attraction of a global maximizer will converge to a global maximizer with large probability.*

Sketch of Theorem

Under additional assumptions, a batch variant of the algorithm that starts in a <u>basin of attraction of a global maximizer</u> will converge to a global maximizer with large probability.

Sketch of Theorem

Under additional assumptions, a batch variant of the algorithm that starts in a basin of attraction of a global maximizer will <u>converge to a global maximizer</u> with large probability.

Sketch of Theorem

Under additional assumptions, a batch variant of the algorithm that starts in a basin of attraction of a global maximizer will converge to a global maximizer with large probability.

Sketch of Theorem

Under additional assumptions, a batch variant of the algorithm that starts in a basin of attraction of a global maximizer will converge to a global maximizer with large probability.

Proof: See preprint when available.

Sketch of Theorem

Under additional assumptions, a batch variant of the algorithm that starts in a basin of attraction of a global maximizer will converge to a global maximizer with large probability.

Proof: See preprint when available.

What are these "additional assumptions"?

④ Local convergence result

Sketch of Theorem

Under additional assumptions, a batch variant of the algorithm that starts in a basin of attraction of a global maximizer will converge to a global maximizer with large probability.

Proof: See preprint when available.

What are these "additional assumptions"?

- There exists a neighborhood of the global maximizer where:
  - The Markov chain of state-action pairs is geometrically ergodic.
  - The objective function behaves approximately in a convex manner in directions that are perpendicular to the set of global maximizers.
  - The function $\mathrm{D}\log\rho$ is bounded and the functions x, r, and $r\,\nabla\log\pi$ grow slowly enough.

④ Local convergence result

Sketch of Theorem

Under additional assumptions, a batch variant of the algorithm that starts in a basin of attraction of a global maximizer will converge to a global maximizer with large probability.

Proof: See preprint when available.

What are these "additional assumptions"?

- There exists a neighborhood of the global maximizer where:
  - The Markov chain of state-action pairs is geometrically ergodic.
  - The objective function behaves approximately in a convex manner in directions that are perpendicular to the set of global maximizers.
  - The function $\mathrm{D} \log \rho$ is bounded and the functions x, r, and $r \nabla \log \pi$ grow slowly enough.
- The step sizes are decreasing and the batch sizes are increasing.

Conclusion

- **Main contributions**
  1. Product-form distributions as exponential families
  2. Score-aware gradient estimator (SAGE)
  3. SAGE-based policy-gradient algorithm
  4. Nonconvex convergence result

Product-form stationary distribution

$$\log p(s|\theta) = \log \rho(\theta)^\intercal x(s) - \log Z(\theta)$$

$$\downarrow$$

$$\nabla \log p(s|\theta) = \mathrm{D} \log \rho(\theta)^\intercal (x(s) - \mathbb{E}[x(S)])$$

Score-aware gradient estimator (SAGE)

Conclusion

- **Main contributions**
  1. Product-form distributions as exponential families
  2. Score-aware gradient estimator (SAGE)
  3. SAGE-based policy-gradient algorithm
  4. Nonconvex convergence result

Product-form stationary distribution

$$\log p(s|\theta) = \log \rho(\theta)^\intercal x(s) - \log Z(\theta)$$

$$\downarrow$$

$$\nabla \log p(s|\theta) = \mathrm{D} \log \rho(\theta)^\intercal (x(s) - \mathbb{E}[x(S)])$$

Score-aware gradient estimator (SAGE)

- **Future research directions**
  - Run extensive numerical results on more challenging examples.

Conclusion

- **Main contributions**
  1. Product-form distributions as exponential families
  2. Score-aware gradient estimator (SAGE)
  3. SAGE-based policy-gradient algorithm
  4. Nonconvex convergence result

Product-form stationary distribution

$$\log p(s|\theta) = \log \rho(\theta)^\intercal x(s) - \log Z(\theta)$$

$$\downarrow$$

$$\nabla \log p(s|\theta) = \mathrm{D} \log \rho(\theta)^\intercal (x(s) - \mathbb{E}[x(S)])$$

Score-aware gradient estimator (SAGE)

- **Future research directions**
  - Run extensive numerical results on more challenging examples.
  - Find better estimators for covariance and expectation, such as robust estimators.

Conclusion

- **Main contributions**
  1. Product-form distributions as exponential families
  2. Score-aware gradient estimator (SAGE)
  3. SAGE-based policy-gradient algorithm
  4. Nonconvex convergence result

Product-form stationary distribution

$$\log p(s|\theta) = \log \rho(\theta)^\intercal x(s) - \log Z(\theta)$$

$$\downarrow$$

$$\nabla \log p(s|\theta) = \mathrm{D} \log \rho(\theta)^\intercal (x(s) - \mathbb{E}[x(S)])$$

Score-aware gradient estimator (SAGE)

- **Future research directions**
  - Run extensive numerical results on more challenging examples.
  - Find better estimators for covariance and expectation, such as robust estimators.
  - Apply to (queueing) systems where the stationary distribution is known only *up to a multiplicative constant*.