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)**

![Diagram of a Markov decision process](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 $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, \ldots$
- Environment $P(s', r|s, a) = \mathbb{P}[S_{t+1} = s'| S_t = s, R_{t+1} = r| A_t = a]$
Reinforcement learning

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

Source: Wikipedia (modified)
Reinforcement learning

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

- **Goal**: Find a $\theta$ 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 \(S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, \ldots\)
  - Environment \(P(s', r|s, a) = \mathbb{P}\left[ S_{t+1}=s' \mid S_t=s, R_{t+1}=r, A_t=a \right] \)
  - Policy parameterization \(\pi(a|s, \theta) = \mathbb{P}[A_t=a \mid S_t=s] \)

- **Goal:** Find a \(\theta\) 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 $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, \ldots$
  - Environment $P(s', r | s, a) = \mathbb{P}\left[ S_{t+1} = s' \mid S_t = s, A_t = a \right]$
  - Policy parameterization $\pi(a | s, \theta) = \mathbb{P}[A_t = a \mid S_t = s]$

- **Goal:** Find a $\theta$ 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** $(S, A, R) \sim \lim_{t \to +\infty}(S_t, A_t, R_{t+1})$:

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

Stationary distribution of $(S_t, t \geq 0)$ under $\pi(a | s, \theta)$

Source: Wikipedia (modified)
Typical policy-gradient algorithm:

1: Initialize $S_0$ and $\Theta_0$
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
Typical policy-gradient algorithm:

1. Initialize $S_0$ and $\Theta_0$
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 $\left[\nabla J(\Theta_t)\right]$ 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 $\Theta_{t+1} \leftarrow \Theta_t + \alpha \left[\nabla J(\Theta_t)\right]$
3. end for

\[
\left[\cdot\right] = \text{estimate of } \cdot \\
\nabla = \text{gradient with respect to } \theta
\]
Typical **policy-gradient algorithm**:

1. Initialize $S_0$ and $\Theta_0$
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

$\lfloor \cdot \rfloor = \text{estimate of } \cdot$

$\nabla = \text{gradient with respect to } \theta$

How?
Policy-gradient algorithms

- **Typical policy-gradient algorithm:**

  1. Initialize $S_0$ and $\Theta_0$
  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

- **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:**
  1. Initialize $S_0$ and $\Theta_0$
  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 $\left[\nabla J(\Theta_t)\right]$ 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 $\Theta_{t+1} \leftarrow \Theta_t + \alpha \left[\nabla J(\Theta_t)\right]$  
  7. **end for**

- **Actor-critic** applies the policy-gradient theorem (Sutton and Barto, 2018):
  
  $$
  \left[\nabla J(\Theta_t)\right] \leftarrow (R_{t+1} - \mathbb{E}[R]) + \left[\nabla \log \pi(A_t|S_t, \Theta_t)\right]
  $$

- 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 $s \in \{0, 1, 2, \ldots\}$
- Actions: accept or reject
- Admission reward $\alpha$ per job
- Holding cost rate $\eta$ 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 $\alpha$ per job
- Holding cost rate $\eta$ per job per time unit
- Policy $\pi(\text{accept}|s, \theta) = \frac{1}{1 + e^{-\theta s}}$
Example: M/M/1 queue with admission control

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

- Arrival rate $\lambda > 0$, service rate $\mu > 0$
- State: queue length $s \in \{0, 1, 2, \ldots\}$
- Actions: accept or reject
- Admission reward $\alpha$ per job
- Holding cost rate $\eta$ per job per time unit
- Policy $\pi(\text{accept}|s, \theta) = \frac{1}{1 + e^{-\theta_{\min(s,k)}}}$ with parameter vector $\theta = (\theta_0, \theta_1, \ldots, \theta_k)$
- Average reward rate $J(\theta) = \alpha \times \left( \sum_{s=0}^{+\infty} p(s|\theta)\pi(\text{accept}|s, \theta) \right) - \eta \times \left( \sum_{s=0}^{+\infty} p(s|\theta)s \right) \times \frac{1}{\lambda}$
Example: M/M/1 queue with admission control

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

Probability of accepting a job
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 $\alpha$ per job
- Holding cost rate $\eta$ per job per time unit
- Policy $\pi(\text{accept}|s, \theta) = \frac{1}{1 + e^{-\theta_{\min(s,k)}}}$ with parameter vector $\theta = (\theta_0, \theta_1, \ldots, \theta_k)$

Average reward rate $J(\theta) = \alpha \times \left( \sum_{s=0}^{+\infty} p(s|\theta)\pi(\text{accept}|s, \theta) \right) - \eta \times \left( \sum_{s=0}^{+\infty} p(s|\theta)s \right) \times \frac{1}{\lambda}$
Example: M/M/1 queue with admission control

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

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

Average reward rate $J(\theta) = \alpha \times \left( \sum_{s=0}^{\infty} p(s|\theta) \pi(\text{accept}|s, \theta) \right) - \eta \times \left( \sum_{s=0}^{\infty} p(s|\theta)s \right) \times \frac{1}{\lambda}$

Stationary distribution $p(s|\theta) \propto \prod_{i=0}^{k-1} \left( \frac{\lambda}{\mu} \pi(\text{accept}|i, \theta) \right)^{1 \{s \geq i\}} \left( \frac{\lambda}{\mu} \pi(\text{accept}|k, \theta) \right)^{\max(s-k, 0)}$
Example: M/M/1 queue with admission control

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

Average reward rate $J(\theta) = \alpha \times \left( \sum_{s=0}^{+\infty} p(s|\theta) \pi(\text{accept}|s, \theta) \right) - \eta \times \left( \sum_{s=0}^{+\infty} p(s|\theta)s \right) \times \frac{1}{\lambda}$

Stationary distribution $p(s|\theta) \propto \prod_{i=0}^{k-1} \left( \frac{\lambda}{\mu} \pi(\text{accept}|i, \theta) \right) ^ {1 \{s \geq i\}} \left( \frac{\lambda}{\mu} \pi(\text{accept}|k, \theta) \right) ^ {\max(s-k,0)}$

Céline Comte
Reinforcement Learning for Stochastic Networks
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**

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
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
Product-form distribution

\[ p(s|\theta) = \frac{1}{Z(\theta)} \prod_{i=1}^{n} \rho_i(\theta) x_i(s) \]
Product-form distributions as exponential families

**Product-form distribution**

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

Depends on \( \theta \)
Product-form distribution

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

- Depends on \( s \)
- Depends on \( \theta \)

---

Céline Comte
Reinforcement Learning for Stochastic Networks
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** \( x = (x_1, x_2, \ldots, x_n) \)
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** \[ x = (x_1, x_2, \ldots, x_n) \]

- **Load function** \[ \rho = (\rho_1, \rho_2, \ldots, \rho_n) \]
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** \( x = (x_1, x_2, \ldots, x_n) \)
- **Load function** \( \rho = (\rho_1, \rho_2, \ldots, \rho_n) \)
- **Partition function** \( Z \)
  \[
  Z(\theta) = \sum_{s} \prod_{i=1}^{n} \rho_i(\theta) x_i(s)
  \]
Product-form distributions as exponential families

- **Product-form** distribution \( p(s|\theta) = \frac{1}{Z(\theta)} \prod_{i=1}^{n} \rho_i(\theta) x_i(s) \)

\[ \log p(s|\theta) = \log \rho(\theta) \top x(s) - \log Z(\theta) \]

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

- **Load function** \( \rho = (\rho_1, \rho_2, \ldots, \rho_n) \)

- **Partition function** \( Z \)

\[ Z(\theta) = \sum_{s} \prod_{i=1}^{n} \rho_i(\theta) x_i(s) \]
Product-form distributions as exponential families

- **Product-form** distribution \( p(s|\theta) = \frac{1}{Z(\theta)} \prod_{i=1}^{n} \rho_i(\theta) x_i(s) \)  
  \[ \log p(s|\theta) = \log \rho(\theta)^{\top} x(s) - \log Z(\theta) \]

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

- **Load function** \( \rho = (\rho_1, \rho_2, \ldots, \rho_n) \)

- **Partition function** \( Z \)
  \[ Z(\theta) = \sum_s \prod_{i=1}^{n} \rho_i(\theta) x_i(s) \]
Product-form distributions as exponential families

- **Product-form distribution** \( p(s|\theta) = \frac{1}{Z(\theta)} \prod_{i=1}^{n} \rho_i(\theta)^{x_i(s)} \) \rightarrow **Exponential family** of distributions
  \[ \log p(s|\theta) = \log \rho(\theta)^\top x(s) - \log Z(\theta) \]

- **Feature function** \( x = (x_1, x_2, \ldots, x_n) \) \rightarrow **Feature function** \( x = (x_1, x_2, \ldots, x_n) \)

- **Load function** \( \rho = (\rho_1, \rho_2, \ldots, \rho_n) \) \rightarrow **Log-load function** \( \log \rho = (\log \rho_1, \ldots, \log \rho_n) \)

- **Partition function** \( Z \)
  \[ Z(\theta) = \sum_s \prod_{i=1}^{n} \rho_i(\theta)^{x_i(s)} \]
Product-form distributions as exponential families

- **Product-form distribution** $p(s|\theta) = \frac{1}{Z(\theta)} \prod_{i=1}^{n} \rho_i(\theta) x_i(s)$
- **Exponential family** of distributions

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

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

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

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

$$\log Z(\theta) = \log \left( \sum_{s} e^{\log \rho(\theta)^\top x(s)} \right)$$
The **score** is the gradient of the log-likelihood with respect to the parameter vector:

\[
\text{“Likelihood”} = p(s|\theta) \quad \rightarrow \quad \text{“Score”} = \nabla \log p(s|\theta).
\]
The **score** is the gradient of the log-likelihood with respect to the parameter vector:

\[
"\text{Likelihood" } = p(s|\theta) \rightarrow "\text{Score" } = \nabla \log p(s|\theta).
\]

**Theorem**

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

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

\[
\nabla J(\theta) = D \log \rho(\theta)^\top \text{Cov}[R, x(S)] + \mathbb{E}[R \nabla \log \pi(A|S, \theta)].
\]
Score-aware gradient estimator (SAGE)

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

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

\[
\nabla J(\theta) = D \log \rho(\theta)^\top \operatorname{Cov}[R, x(S)] + \mathbb{E}[R \nabla \log \pi(A|S, \theta)].
\]

- **Main take-away:** If we can evaluate \(D \log \rho(\theta)\), this gives us an estimator for \(\nabla J(\theta)\).
Typical **policy-gradient algorithm**:  

1. Initialize $S_0$ and $\Theta_0$  
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**

How?

Céline Comte  
Reinforcement Learning for Stochastic Networks
Typical **policy-gradient algorithm:**

1. Initialize $S_0$ and $\Theta_0$
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**

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

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

1: Initialize $S_0$ and $\Theta_0$
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

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

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

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

$$\nabla J(\Theta_t) \leftarrow \mathbf{D} \log \rho(\Theta_t)^\top [\text{Cov}[R, x(S)]] + \mathbb{E}[R \nabla \log \pi(A|S, \Theta_t)].$$
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 $\pi_0(\theta^*) = \pi_1(\theta^*) = \pi_2(\theta^*) = 1$ and $\pi_3(\theta^*) = 0$. 

**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 $\pi_0(\theta^*) = \pi_1(\theta^*) = 1$ and $\pi_2(\theta^*) = \pi_3(\theta^*) = 0$. 

**Simulation setup**
- 10^6 steps
- Convergence time $T$: $J(\Theta_t) > J(\theta^*) - \epsilon$ for each $t \in \{T, T+1, \ldots, 10^6\}$
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 $\pi(\Theta_0) = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$
- Optimal policy $\pi_0(\theta^*) = \pi_1(\theta^*) = \pi_2(\theta^*) = 1$ and $\pi_3(\theta^*) = 0$.

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

Simulation setup
- 10^6 steps
- Convergence time $T$: $J(\Theta_t) > J(\theta^*) - \epsilon$ for each $t \in \{T, T + 1, \ldots, 10^6\}$
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 $\pi_0(\theta^*) = \pi_1(\theta^*) = \pi_2(\theta^*) = 1$ and $\pi_3(\theta^*) = 0$.

**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 $\pi_0(\theta^*) = \pi_1(\theta^*) = 1$ and $\pi_2(\theta^*) = \pi_3(\theta^*) = 0$.

**Simulation setup**
- $10^6$ steps
- Convergence time $T$: $J(\Theta_t) > J(\theta^*) - \epsilon$
  for each $t \in \{T, T + 1, \ldots, 10^6\}$
Example: M/M/1 queue with admission control

Stable case – Convergence times

Time $t$

Céline Comte
Reinforcement Learning for Stochastic Networks
Example: M/M/1 queue with admission control

Stable case – Convergence times

<table>
<thead>
<tr>
<th></th>
<th>Time $t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAGE – 50%</td>
<td>$10^4$</td>
</tr>
<tr>
<td>SAGE – 20%</td>
<td>$10^5$</td>
</tr>
<tr>
<td>SAGE – 10%</td>
<td>$10^5$</td>
</tr>
<tr>
<td>AC – 50%</td>
<td>$10^6$</td>
</tr>
<tr>
<td>AC – 20%</td>
<td>$10^6$</td>
</tr>
<tr>
<td>AC – 10%</td>
<td>$10^6$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Time $t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAGE – 5%</td>
<td>$10^5$</td>
</tr>
<tr>
<td>SAGE – 2%</td>
<td>$10^6$</td>
</tr>
<tr>
<td>SAGE – 1%</td>
<td>$10^6$</td>
</tr>
<tr>
<td>AC – 5%</td>
<td>$10^6$</td>
</tr>
<tr>
<td>AC – 2%</td>
<td>$10^6$</td>
</tr>
<tr>
<td>AC – 1%</td>
<td>$10^6$</td>
</tr>
</tbody>
</table>

Céline Comte
Reinforcement Learning for Stochastic Networks
Example: M/M/1 queue with admission control

Stable case – SAGE

Policy $\pi_i(\Theta_t)$ vs Time $t$
Example: M/M/1 queue with admission control

Stable case – Actor-critic

Policy $\pi_i(\Theta_t)$

Time $t$

Céline Comte
Reinforcement Learning for Stochastic Networks
Example: M/M/1 queue with admission control

Possibly-unstable case – SAGE

Policy \( \pi_i(\Theta_t) \)
Example: M/M/1 queue with admission control
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:

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

1. The Markov chain of state-action pairs is geometrically ergodic.
2. The objective function behaves approximately in a convex manner in directions that are perpendicular to the set of global maximizers.
3. The function $D \log \rho$ is bounded and the functions $x$, $r$, and $r \nabla \log \pi$ grow slowly enough.
4. The step sizes are decreasing and the batch sizes are increasing.
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 $D \log \rho$ is bounded and the functions $x, r, r \nabla \log \pi$ grow slowly enough.
- The step sizes are decreasing and the batch sizes are increasing.
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.
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.
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 $D \log \rho$ 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.
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 $D \log \rho$ is bounded and the functions $x$, $r$, and $r \nabla \log \pi$ grow slowly enough.
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 $D \log \rho$ 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

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

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

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

**Future research directions**

- Run extensive numerical results on more challenging examples.

Product-form stationary distribution

\[
\log p(s|\theta) = \log \rho(\theta)^T x(s) - \log Z(\theta)
\]

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

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

Future research directions

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

Product-form stationary distribution

\[
\log p(s|\theta) = \log \rho(\theta)^\top x(s) - \log Z(\theta)
\]

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

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

- **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.

---

Product-form stationary distribution

\[
\log p(s|\theta) = \log \rho(\theta)^\top x(s) - \log Z(\theta)
\]

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

Score-aware gradient estimator (SAGE)