Preconditioning Helps: Faster Convergence in Statistical and Reinforcement Learning

Yuejie Chi

Carnegie Mellon University

November 2020
Nonconvex problems are ubiquitous

Given data / model, estimate parameter of interest $x$:

$$\text{optimize}_x \ f(x)$$
Nonconvex problems are ubiquitous

Given data / model, estimate parameter of interest $\mathbf{x}$:

$$\text{optimize}_{\mathbf{x}} \quad f(\mathbf{x})$$

Often lead to nonconvex problems!
Nonconvex problems are hard!

There may be bumps everywhere and exponentially many local optima, e.g. 1-layer neural networks (Auer et.al. '96)
Nonconvex problems are hard!

“...in fact, the great watershed in optimization isn’t between linearity and nonlinearity, but convexity and nonconvexity.

R. T. Rockafellar, in SIAM Review, 1993
Simple algorithms can be efficient for nonconvex learning!

Vanilla gradient descent (GD):

$$x_{t+1} = x_t - \eta \nabla f(x_t)$$

for $t = 0, 1, \ldots$
Simple algorithms can be efficient for nonconvex learning!
Simple algorithms can be efficient for nonconvex learning!

**Vanilla gradient descent (GD):**

\[ \mathbf{x}^{t+1} = \mathbf{x}^t - \eta \nabla f(\mathbf{x}^t) \]

for \( t = 0, 1, \ldots \)
Recent testaments: provable nonconvex optimization

**Phase retrieval:** Netrapalli et al. ’13, Candès, Li, Soltanolkotabi ’14, Chen, Candès ’15, Cai, Li, Ma ’15, Zhang et al. ’16, Wang et al. ’16, Sun, Qu, Wright ’16, Ma et al. ’17, Chen et al. ’18, Soltani, Hegde ’18, Ruan and Duchi, ’18, ...


**Blind deconvolution/demixing:** Li et al. ’16, Lee et al. ’16, Cambareri, Jacques ’16, Ling, Stroudmer ’16, Huang, Hand ’16, Ma et al. ’17, Zhang et al. ’18, Li, Bresler ’18, Dong, Shi ’18, Shi, Chi ’19, Qu et al. ’19, ...

**Dictionary learning:** Arora et al. ’14, Sun et al. ’15, Chatterji, Bartlett ’17, Bai et al. ’18, Gilboa et al. ’18, Rambhatla et al. ’19, Qu et al. ’19, ...

**Robust principal component analysis:** Netrapalli et al. ’14, Yi et al. ’16, Gu et al. ’16, Ge et al. ’17, Cherapanamjeri et al. ’17, Vaswani et al. ’18, Maunu et al. ’19, ...

**Deep learning:** Zhong et al. ’17, Bai, Mei, Montanari ’17, Du et al. ’17, Ge, Lee, Ma ’17, Du et al. ’18, Soltanolkotabi and Oymak, ’18...
Vanilla GD:

\[ \mathbf{x}^{t+1} = \mathbf{x}^t - \eta \nabla f(\mathbf{x}^t) \]

😊 Slows down with ill-conditioning.
This talk: acceleration via preconditioning

Vanilla GD:

\[ \mathbf{x}^{t+1} = \mathbf{x}^t - \eta \nabla f(\mathbf{x}^t) \]

😊 Slows down with ill-conditioning.

Preconditioning accelerates convergence!

Preconditioned GD:

\[ \mathbf{x}^{t+1} = \mathbf{x}^t - \eta \underbrace{\mathbf{H}_t}_{\text{preconditioner}} \nabla f(\mathbf{x}^t) \]
This talk: two recent case studies

Low-rank Matrix Estimation
Statistical Learning

Policy Optimization
Reinforcement Learning
Accelerating ill-conditioned matrix estimation in statistical learning

Tian Tong
CMU

Cong Ma
Berkeley
Low-rank matrices are redundant representations of latent information.
Low-rank matrix sensing

\[ M \in \mathbb{R}^{n_1 \times n_2} \]
\[ \text{rank}(M) = r \]

\[ y = \mathcal{A}(M) + \text{noise} \]

\[ y \in \mathbb{R}^m \]

Recover \( M \) in the sample-starved regime:

\[ (n_1 + n_2)r \lesssim m \ll n_1 n_2 \]

degree of freedom \hspace{1cm} sensing budget \hspace{1cm} ambient dimension
Low-rank matrix factorization

\[
\min_{\text{rank}(Z)=r} \frac{1}{2} \| y - A(Z) \|^2_2
\]
Low-rank matrix factorization

\[
\min_{\text{rank}(Z)=r} \frac{1}{2} \| y - A(Z) \|_2^2
\]

Saves memory and computation but introduces nonconvexity!
Low-rank matrix factorization

\[
\min_{\text{rank}(Z) = r} \frac{1}{2} \| y - A(Z) \|_2^2
\]

A detour: nonconvex optimization
Use low-rank representation
\[ Z = XY \]
\[ \text{with } X, Y \in \mathbb{R}^{n \times r} \]

\[
\min_{X,Y} \quad f(X,Y) = \frac{1}{2} \left\| y - A(XY^T) \right\|_2^2
\]

Saves memory and computation but introduces nonconvexity!
Low-rank matrix factorization

\[
\min_{\text{rank}(Z)=r} \frac{1}{2} \| y - A(Z) \|_2^2
\]

Saves memory and computation but introduces nonconvexity!
Prior art: GD with balancing regularization

\[
\min_{X, Y} f_{\text{reg}}(X, Y) = \frac{1}{2} \left\| y - \mathcal{A}(XY^\top) \right\|^2_2 + \frac{1}{8} \left\| X^\top X - Y^\top Y \right\|_F^2
\]

- **Spectral initialization**: find an initial point in the “basin of attraction”.
  \[
  (X_0, Y_0) \leftarrow \text{SVD}_r(\mathcal{A}^*(y))
  \]

- **Gradient iterations**:
  \[
  X_{t+1} = X_t - \eta \nabla_X f_{\text{reg}}(X_t, Y_t)
  \\
  Y_{t+1} = Y_t - \eta \nabla_Y f_{\text{reg}}(X_t, Y_t)
  \]
  for \( t = 0, 1, \ldots \)
Prior theory for vanilla GD

**Theorem (Tu et al., ICML 2016)**

Suppose $M = X_* Y_*^\top$ is rank-$r$ and has a condition number $\kappa = \sigma_{\max}(M)/\sigma_{\min}(M)$. For low-rank matrix sensing with i.i.d. Gaussian design, vanilla GD (with spectral initialization) achieves

$$\|X_t Y_t^\top - M\|_F \leq \varepsilon \cdot \sigma_{\min}(M)$$

- **Computational**: within $O(\kappa \log \frac{1}{\varepsilon})$ iterations;
- **Statistical**: as long as the sample complexity satisfies

$$m \gtrsim (n_1 + n_2)r^2 \kappa^2.$$
Prior theory for vanilla GD

**Theorem (Tu et al., ICML 2016)**

Suppose $M = X_\ast Y_\ast^\top$ is rank-$r$ and has a condition number $\kappa = \sigma_{\text{max}}(M)/\sigma_{\text{min}}(M)$. For low-rank matrix sensing with i.i.d. Gaussian design, vanilla GD (with spectral initialization) achieves

$$\|X_t Y_t^\top - M\|_F \leq \varepsilon \cdot \sigma_{\text{min}}(M)$$

- **Computational:** within $O(\kappa \log \frac{1}{\varepsilon})$ iterations;
- **Statistical:** as long as the sample complexity satisfies

$$m \gtrsim (n_1 + n_2) r^2 \kappa^2.$$

Similar results hold for many low-rank problems.

(Netrapalli et al. ’13, Candès, Li, Soltanolkotabi ’14, Sun and Luo ’15, Chen and Wainwright ’15, Zheng and Lafferty ’15, Ma et al. ’17, ....)
Convergence slows down for ill-conditioned matrices

\[
\min_{X,Y} \quad f(X, Y) = \frac{1}{2} \left\| \mathcal{P}_\Omega (XY^\top - M) \right\|_F^2
\]

Vanilla GD converges in \( O(\kappa \log \frac{1}{\varepsilon}) \) iterations.
Condition number can be large

chlorine concentration levels
120 junctions, 180 time slots

power-law spectrum

Data source: www.epa.gov/water-research/epanet
Condition number can be large

chlorine concentration levels
120 junctions, 180 time slots

\[ \kappa \approx 20 \]

88%

Data source: www.epa.gov/water-research/epanet
Condition number can be large

chlorine concentration levels
120 junctions, 180 time slots

rank-10 approximation

Data source: www.epa.gov/water-research/epanet
Condition number can be large

chlorine concentration levels
120 junctions, 180 time slots

Can we accelerate the convergence rate of GD to $O(\log \frac{1}{\varepsilon})$?

Data source: www.epa.gov/water-research/epanet
A new algorithm: scaled gradient descent (ScaledGD)

\[ f(X, Y) = \frac{1}{2} \left\| y - A(XY^\top) \right\|_2^2 \]

- **Spectral initialization:** find an initial point in the “basin of attraction”.

- **Scaled gradient iterations:**

\[
X_{t+1} = X_t - \eta \nabla_X f(X_t, Y_t) \left( Y_t^\top Y_t \right)^{-1} \]

\[
Y_{t+1} = Y_t - \eta \nabla_Y f(X_t, Y_t) \left( X_t^\top X_t \right)^{-1} \]

for \( t = 0, 1, \ldots \)
A new algorithm: scaled gradient descent (ScaledGD)

\[ f(X, Y) = \frac{1}{2} \| y - A(XY^\top) \|_2^2 \]

- **Spectral initialization:** find an initial point in the “basin of attraction”.

- **Scaled gradient iterations:**

  \[
  X_{t+1} = X_t - \eta \nabla_X f(X_t, Y_t) \left( Y_t^\top Y_t \right)^{-1} \]
  
  \[
  Y_{t+1} = Y_t - \eta \nabla_Y f(X_t, Y_t) \left( X_t^\top X_t \right)^{-1} \]

  for \( t = 0, 1, \ldots \)
A new algorithm: scaled gradient descent (ScaledGD)

\[ f(X, Y) = \frac{1}{2} \| y - A(XY^T) \|_2^2 \]

- **Spectral initialization:** find an initial point in the “basin of attraction”.

- **Scaled gradient iterations:**

  \[
  X_{t+1} = \begin{array}{l}
  X_t - \eta \nabla_X f(X_t, Y_t) \\
  \quad \times (Y_t^T Y_t)^{-1}
  \end{array}
  \]

  \[
  Y_{t+1} = \begin{array}{l}
  Y_t - \eta \nabla_Y f(X_t, Y_t) \\
  \quad \times (X_t^T X_t)^{-1}
  \end{array}
  \]

  for \( t = 0, 1, \ldots \)

ScaledGD is a *preconditioned* gradient method *without* balancing regularization!
ScaledGD for low-rank matrix completion

Huge computational saving: ScaledGD converges in an $\kappa$-independent manner with a minimal overhead!
A closer look at ScaledGD

Invariance to invertible transforms: \((\text{Tanner and Wei, '16; Mishra '16})\)

\[
\begin{align*}
(X_t, Y_t) & \quad M_t = X_t Y_t^T \\
(X_{t+1}, Y_{t+1}) & \quad M_{t+1} = X_{t+1} Y_{t+1}^T
\end{align*}
\]

New distance metric as Lyapunov function:

\[
\text{dist}_2\left([X Y], [X \star Y \star]\right) = \inf_{Q \in \text{GL}(r)} \|XQ - X \star\|_F + \|YQ - Y \star\|_F + a \text{ careful trajectory-based analysis}
\]
A closer look at ScaledGD

**Invariance to invertible transforms:** (Tanner and Wei, ’16; Mishra ’16)

\[
M_t = X_t Y_t^T \\
M_{t+1} = X_{t+1} Y_{t+1}^T
\]

**New distance metric as Lyapunov function:**

\[
\text{dist}^2 \left( \begin{bmatrix} X \\ Y \end{bmatrix}, \begin{bmatrix} X^*_r \\ Y^*_r \end{bmatrix} \right) = \inf_{Q \in \text{GL}(r)} \left\| (XQ - X^r) \Sigma_{1/2} \right\|_F^2 + \left\| (YQ^{-T} - Y^r) \Sigma_{r}^{1/2} \right\|_F^2
\]

+ a careful trajectory-based analysis
Theoretical guarantees of ScaledGD

**Theorem (Tong, Ma and Chi, 2020)**

For low-rank matrix sensing with i.i.d. Gaussian design, ScaledGD with spectral initialization achieves

\[ \|X_t Y_t^\top - M\|_F \preceq \varepsilon \cdot \sigma_{\min}(M) \]

- **Computational:** within \( O(\log \frac{1}{\varepsilon}) \) iterations;
- **Statistical:** the sample complexity satisfies

\[ m \gtrsim (n_1 + n_2)r^2 \kappa^2. \]
Theoretical guarantees of ScaledGD

**Theorem (Tong, Ma and Chi, 2020)**

For low-rank matrix sensing with i.i.d. Gaussian design, ScaledGD with spectral initialization achieves

\[
\|X_t Y_t^\top - M\|_F \lesssim \varepsilon \cdot \sigma_{\min}(M)
\]

- **Computational:** within \(O\left(\log \frac{1}{\varepsilon}\right)\) iterations;
- **Statistical:** the sample complexity satisfies

\[
m \gtrsim (n_1 + n_2) r^2 \kappa^2.
\]

**Strict improvement over Tu et al.:** ScaledGD provably accelerates vanilla GD at the same sample complexity!
## ScaledGD works more broadly

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Robust PCA</th>
<th>Matrix completion</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>corruption fraction</td>
<td>iteration complexity</td>
</tr>
<tr>
<td>GD</td>
<td>$\frac{1}{\mu r^{3/2} \kappa^{3/2} \vee \mu r \kappa^2}$</td>
<td>$\kappa \log \frac{1}{\varepsilon}$</td>
</tr>
<tr>
<td>ScaledGD</td>
<td>$\frac{1}{\mu r^{3/2} \kappa}$</td>
<td>$\log \frac{1}{\varepsilon}$</td>
</tr>
</tbody>
</table>

Huge computation savings at comparable sample complexities!

Numerical stability

ScaledGD converges faster than vanilla GD in a small number of iterations (they eventually reach the same accuracy).
Outlier-corrupted low-rank matrix sensing

\[ M \in \mathbb{R}^{n_1 \times n_2} \]
\[ \text{rank}(M) = r \]

linear map

\[ \mathbf{y} \in \mathbb{R}^{m} \]

Sensor failures
Malicious attacks

\[ \mathbf{y} = A(M) + \text{sparse outliers} \]

a small fraction (e.g. \( p_s \approx 5\% \))
Outlier-corrupted low-rank matrix sensing

\[ M \in \mathbb{R}^{n_1 \times n_2} \]
\[ \text{rank}(M) = r \]

\( A(\cdot) \)
linear map

\[ y \in \mathbb{R}^m \]

Sensor failures
Malicious attacks

\[ y = A(M) + \text{sparse outliers} \]
\[ \text{a small fraction (e.g. } p_s \approx 5\%) \]

Least absolute deviation (LAD)

\[
\min_{X, Y} \quad f(X, Y) = \frac{1}{2} \left\| y - A(XY^\top) \right\|_1
\]
Scaled subgradient iterations:

\[ X_{t+1} = X_t - \eta_t \partial_X f(X_t, Y_t) (Y_t^\top Y_t)^{-1} \]

\[ Y_{t+1} = Y_t - \eta_t \partial_Y f(X_t, Y_t) (X_t^\top X_t)^{-1} \]

where \( \eta_t \) is set as Polyak’s or geometric decaying stepsize.
Scaled subgradient methods

Scaled subgradient iterations:

\[ X_{t+1} = X_t - \eta_t \partial_X f(X_t, Y_t) \left( Y_t^\top Y_t \right)^{-1} \]

\[ Y_{t+1} = Y_t - \eta_t \partial_Y f(X_t, Y_t) \left( X_t^\top X_t \right)^{-1} \]

where \( \eta_t \) is set as Polyak’s or geometric decaying stepsize.

<table>
<thead>
<tr>
<th></th>
<th>matrix sensing</th>
<th>quadratic sensing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subgradient Method</td>
<td>( \frac{\kappa}{(1-2p_s)^2} \log \frac{1}{\epsilon} )</td>
<td>( \frac{r\kappa}{(1-2p_s)^2} \log \frac{1}{\epsilon} )</td>
</tr>
<tr>
<td>(Charisopoulos et al, '19)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ScaledSM</td>
<td>( \frac{1}{(1-2p_s)^2} \log \frac{1}{\epsilon} )</td>
<td>( \frac{r}{(1-2p_s)^2} \log \frac{1}{\epsilon} )</td>
</tr>
</tbody>
</table>
Scaled subgradient methods

**Scaled subgradient iterations:**

\[
X_{t+1} = X_t - \eta_t \partial_X f (X_t, Y_t) \left( Y_t^\top Y_t \right)^{-1}
\]

\[
Y_{t+1} = Y_t - \eta_t \partial_Y f (X_t, Y_t) \left( X_t^\top X_t \right)^{-1}
\]

where \( \eta_t \) is set as Polyak’s or geometric decaying stepsize.

<table>
<thead>
<tr>
<th></th>
<th><strong>matrix sensing</strong></th>
<th><strong>quadratic sensing</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Subgradient Method</strong> (Charisopoulos et al, ’19)</td>
<td>( \frac{\kappa}{(1-2p_s)^2} \log \frac{1}{\varepsilon} )</td>
<td>( \frac{r\kappa}{(1-2p_s)^2} \log \frac{1}{\varepsilon} )</td>
</tr>
<tr>
<td>ScaledSM</td>
<td>( \frac{1}{(1-2p_s)^2} \log \frac{1}{\varepsilon} )</td>
<td>( \frac{r}{(1-2p_s)^2} \log \frac{1}{\varepsilon} )</td>
</tr>
</tbody>
</table>

Robustness to both ill-conditioning and adversarial corruptions!
Accelerating convergence of policy optimization in reinforcement learning
Reinforcement learning (RL)

In RL, an agent learns by interacting with an environment.

Policy optimization is a major driver to these successes.
Markov decision process (MDP)

- $S$: state space
- $A$: action space
- $r(s,a) \in [0,1]$: immediate reward
- $\pi(\cdot|s)$: policy (or action selection rule)
- $P(\cdot|s,a)$: transition probabilities

![Diagram of an agent interacting with an environment](image)
Markov decision process (MDP)

- $S$: state space
- $A$: action space
- $r(s, a) \in [0, 1]$: immediate reward

$\pi(s | s_t)$: policy (or action selection rule)

$P(\cdot | s, a)$: transition probabilities
Markov decision process (MDP)

- $S$: state space
- $A$: action space
- $r(s, a) \in [0, 1]$: immediate reward
- $\pi(\cdot|s)$: policy (or action selection rule)
Markov decision process (MDP)

- $S$: state space
- $A$: action space
- $r(s, a) \in [0, 1]$: immediate reward
- $\pi(\cdot|s)$: policy (or action selection rule)
- $P(\cdot|s, a)$: transition probabilities
Value function and Q-function of policy $\pi$:

\[ \forall s \in S : \quad V^\pi(s) := \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s \right] \]

\[ \forall (s, a) \in S \times A : \quad Q^\pi(s, a) := \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, a_0 = a \right] \]
Value function and Q-function

Value function and Q-function of policy $\pi$:

$\forall s \in S : \quad V^\pi(s) := \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s \right]$ 

$\forall (s, a) \in S \times A : \quad Q^\pi(s, a) := \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, a_0 = a \right]$ 

- $\gamma \in [0, 1)$ is the discount factor; $\frac{1}{1-\gamma}$ is effective horizon
- Expectation is w.r.t. the sampled trajectory under $\pi$
To encourage exploration, promote the stochasticity of the policy using the “soft” value function:

\[
\forall s \in \mathcal{S} : \quad V^\pi_\tau (s) := \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t (r_t - \tau \log \pi(a_t | s_t)) \mid s_0 = s \right]
\]

where \( \tau \) is the entropy regularization parameter.
Entropy-regularized RL

To encourage exploration, promote the stochasticity of the policy using the "soft" value function:

$$\forall s \in S : \quad V^\pi_\tau(s) := \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t (r_t - \tau \log \pi(a_t|s_t)) \mid s_0 = s \right]$$

where $\tau$ is the entropy regularization parameter.

**Goal:** find the optimal policy $\pi^*_\tau$ that maximize $V^\pi_\tau(s)$
Policy gradient methods

Given an initial state distribution \( s \sim \rho \), find policy \( \pi \) such that

\[
\text{maximize}_{\pi} \quad V_{\pi}^{\pi}(\rho) := \mathbb{E}_{s \sim \rho} [V_{\pi}^{\pi}(s)]
\]

Policy gradient methods (Sutton et al., 2000)

For \( t = 0, 1, \cdots \),

\[
\theta(t+1) = \theta(t) + \eta \nabla_{\theta} V_{\pi}^{\pi}(\rho)
\]

where \( \eta \) is the learning rate.
Policy gradient methods

Given an initial state distribution \( s \sim \rho \), find policy \( \pi \) such that

\[
\text{maximize}_{\pi} \quad V^\pi_\tau(\rho) := \mathbb{E}_{s \sim \rho} [V^\pi_\tau(s)]
\]

softmax parameterization:

\[
\pi_\theta(a|s) = \frac{\exp(\theta(s, a))}{\sum_a \exp(\theta(s, a))}
\]
Policy gradient methods

Given an initial state distribution \( s \sim \rho \), find policy \( \pi \) such that

\[
\text{maximize}_\pi \quad V^\pi_\tau (\rho) := \mathbb{E}_{s \sim \rho} [V^\pi_\tau (s)]
\]

softmax parameterization:

\[
\pi_\theta (a|s) = \frac{\exp(\theta (s, a))}{\sum_a \exp(\theta (s, a))}
\]

\[
\text{maximize}_\theta \quad V^{\pi_\theta}_\tau (\rho) := \mathbb{E}_{s \sim \rho} [V^{\pi_\theta}_\tau (s)]
\]
Policy gradient methods

Given an initial state distribution \( s \sim \rho \), find policy \( \pi \) such that

\[
\max_{\pi} \quad V_\pi (\rho) := \mathbb{E}_{s \sim \rho} [V_\pi (s)]
\]

softmax parameterization:

\[
\pi_\theta (a|s) = \frac{\exp(\theta (s, a))}{\sum_a \exp(\theta (s, a))}
\]

\[
\max_{\theta} \quad V_{\pi_\theta} (\rho) := \mathbb{E}_{s \sim \rho} [V_{\pi_\theta} (s)]
\]

Policy gradient methods (Sutton et al., 2000)

For \( t = 0, 1, \cdots \)

\[
\theta^{(t+1)} = \theta^{(t)} + \eta \nabla_\theta V_{\pi_\theta}^{(t)} (\rho)
\]

where \( \eta \) is the learning rate.
Natural policy gradient (Kakade, 2002)

For $t = 0, 1, \ldots$

$$\theta^{(t+1)} = \theta^{(t)} + \eta (\mathcal{F}_\rho^\theta)^\dagger \nabla_\theta V_{\tau\theta}^{\pi(\theta)} (\rho)$$

where $\eta$ is the learning rate and $\mathcal{F}_\rho^\theta$ is the Fisher information matrix:

$$\mathcal{F}_\rho^\theta := \mathbb{E} \left[ (\nabla_\theta \log \pi_\theta(a|s)) (\nabla_\theta \log \pi_\theta(a|s))^\top \right].$$
Natural gradient helps!

**Toy example:** a bandit with 3 arms of rewards 1, 0.9 and 0.1.
**Toy example:** a bandit with 3 arms of rewards 1, 0.9 and 0.1.

NPG follows a more direct path to find the optimal policy.
Advantages of policy gradient methods:

- directly optimize the policy, which is the quantity of interest;
- allow flexible differentiable parameterizations of the policy;
- work with both continuous and discrete problems.

TRPO = NPG + line search
(Schulman et al., 2015)

A3C (Mnih et al., 2016)
SAC (Haarnoja et al., 2018)
Advantages of policy gradient methods:

- directly optimize the policy, which is the quantity of interest;
- allow flexible differentiable parameterizations of the policy;
- work with both continuous and discrete problems.

TRPO = NPG + line search
(Schulman et al., 2015)

A3C (Mnih et al., 2016)
SAC (Haarnoja et al., 2018)

Can we justify the efficacy of NPG in entropy-regularized RL?
Recent breakthroughs on understanding global convergence of

- policy gradient methods for control (Fazel et al., 2018; Bhandari and Russo, 2019);
- (un)regularized policy gradients for tabular MDPs (Agarwal et al., 2019, Bhandari and Russo, 2019; Mei et al. 2020);
- unregularized NPG for tabular MDPs (Agarwal et al., 2019);

and many others.
Entropy-regularized NPG in the tabular setting

For $t = 0, 1, \cdots$, the policy is updated via

$$
\pi^{(t+1)}(a|s) \propto \pi^{(t)}(a|s)^{1-\frac{n\tau}{1-\gamma}} \exp(Q^{\pi(t)}_\tau(s, a)/\tau)^{\frac{n\tau}{1-\gamma}}
$$

where $Q^{\pi(t)}_\tau$ is the soft $Q$-function of $\pi^{(t)}$, and $0 < \eta \leq \frac{1-\gamma}{\tau}$.

- invariant with the choice of $\rho$
- Reduces to soft policy iteration when $\eta = \frac{1-\gamma}{\tau}$. 

Entropy-regularized NPG (Tabular setting)
Linear convergence with exact gradient

**Exact oracle:** perfect evaluation of $Q_{\pi}^{\pi(t)}$ given $\pi^{(t)}$;
— Read our paper for the inexact case!

**Theorem (Cen, Cheng, Chen, Wei, Chi ’20)**

For any learning rate $0 < \eta \leq (1 - \gamma)/\tau$, the entropy-regularized NPG updates satisfy

- **Linear convergence of soft Q-functions:**

$$\|Q^{\star}_\tau - Q^{(t+1)}_\tau\|_\infty \leq C_1 \gamma (1 - \eta T)^t$$

for all $t \geq 0$, where $Q^{\star}_\tau$ is the optimal soft Q-function, and

$$C_1 = \|Q^{\star}_\tau - Q^{(0)}_\tau\|_\infty + 2\tau \left(1 - \frac{\eta T}{1 - \gamma}\right) \|\log \pi^{\star}_\tau - \log \pi^{(0)}\|_\infty.$$
Linear convergence with exact gradient

**Exact oracle:** perfect evaluation of $Q^{\pi(t)}_\tau$ given $\pi(t)$;

— *Read our paper for the inexact case!*

**Theorem (Cen, Cheng, Chen, Wei, Chi ’20)**

For any learning rate $0 < \eta \leq (1 - \gamma)/\tau$, the entropy-regularized NPG updates satisfy

- **Linear convergence of log policies:**

\[
\| \log \pi^*_\tau - \log \pi^{(t+1)} \|_\infty \leq 2C_1 \tau^{-1} (1 - \eta \tau)^t
\]

for all $t \geq 0$, where $\pi^*_\tau$ is the optimal policy, and

\[
C_1 = \| Q^*_\tau - Q^{(0)}_\tau \|_\infty + 2\tau \left(1 - \frac{\eta \tau}{1 - \gamma}\right) \| \log \pi^*_\tau - \log \pi^{(0)} \|_\infty.
\]
Implications

To reach $\|Q^*_\tau - Q^{(t+1)}_\tau\|_\infty \leq \epsilon$, the iteration complexity is at most

- **General learning rates** ($0 < \eta < \frac{1-\gamma}{\tau}$):
  \[
  \frac{1}{\eta \tau} \log \left( \frac{C_1 \gamma}{\epsilon} \right)
  \]

- **Soft policy iteration** ($\eta = \frac{1-\gamma}{\tau}$):
  \[
  \frac{1}{1-\gamma} \log \left( \frac{\|Q^*_\tau - Q^{(0)}_\tau\|_\infty \gamma}{\epsilon} \right)
  \]
Implications

To reach $\|Q^*_\tau - Q^{(t+1)}_\tau\|_\infty \leq \epsilon$, the iteration complexity is at most

- **General learning rates** $(0 < \eta < \frac{1-\gamma}{\tau})$:
  \[
  \frac{1}{\eta \tau} \log \left( \frac{C_1 \gamma}{\epsilon} \right)
  \]

- **Soft policy iteration** $(\eta = \frac{1-\gamma}{\tau})$:
  \[
  \frac{1}{1 - \gamma} \log \left( \frac{\|Q^*_\tau - Q^{(0)}_\tau\|_\infty \gamma}{\epsilon} \right)
  \]

Global linear convergence of entropy-regularized NPG at a rate independent of $|S|, |A|$!
Comparisons with entropy-regularized PG

(Mei et. al. ’20) showed entropy-regularized PG achieves

\[ V^*_\tau(\rho) - V^{(t)}_\tau(\rho) \leq \left( V^*_\tau(\rho) - V^{(0)}_\tau(\rho) \right) \]

\[
\cdot \exp \left( - \frac{(1 - \gamma)^4 t}{(8/\tau + 4 + 8 \log |A|)|S|} \right) \left\| \frac{d\pi^*_\tau}{\rho} \right\|^{-1} \min_{s} \rho(s) \left( \inf_{0 \leq k \leq t - 1} \min_{s,a} \pi^{(k)}(a|s) \right)^2 \]

unclear dependence with \(|S|, |A|, \gamma\)

Much faster convergence of entropy-regularized NPG at a dimension-free rate!
Aside: entropy helps!

**Vanilla NPG**
\[ \tau = 0 \]

**Regularized NPG**
\[ \tau = 0.001 \]

**Sublinear rate:**
\[
\frac{1}{\min\{\eta,(1-\gamma)^2\} \epsilon}
\]
(Agarwal et.al. 2019)

**Linear rate:**
\[
\frac{1}{\eta \tau} \log \left( \frac{1}{\epsilon} \right)
\]
Ours
Aside: entropy helps!

**Sublinear rate:** \( \frac{1}{\min\{\eta, (1-\gamma)^2\}} \epsilon \)

(Agarwal et al. 2019)

**Linear rate:** \( \frac{1}{\eta \tau} \log \left( \frac{1}{\epsilon} \right) \)

Ours

Entropy regularization enables fast convergence!
Recall: Bellman’s optimality principle

Bellman operator

\[ T(Q)(s, a) := r(s, a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s, a)} \left[ \max_{a' \in A} Q(s', a') \right] \]

- one-step look-ahead
Recall: Bellman’s optimality principle

Bellman operator

\[ T(Q)(s, a) := r(s, a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s, a)} \left[ \max_{a' \in A} Q(s', a') \right] \]

- immediate reward
- next state's value

• one-step look-ahead

Bellman equation: \( Q^* \) is unique solution to

\[ T(Q^*) = Q^* \]

\( \gamma \)-contraction of Bellman operator:

\[ \| T(Q_1) - T(Q_2) \|_{\infty} \leq \gamma \| Q_1 - Q_2 \|_{\infty} \]
Soft Bellman operator

\[ T_\tau(Q)(s, a) := r(s, a) \]

\[ + \gamma \mathbb{E}_{s' \sim P(\cdot|s, a)} \left[ \max_{a' \sim \pi(\cdot|s')} \mathbb{E}_{\pi(\cdot|s')} \left[ Q(s', a') - \tau \log \pi(a'|s') \right] \right] \]

immediate reward

next state's value

entropy
Soft Bellman operator

\[ T_\tau(Q)(s, a) := r(s, a) \]

immediate reward

\[ + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[ \max_{a' \sim \pi(\cdot|s')} \mathbb{E}_{a' \sim \pi(\cdot|s')} \left[ Q(s', a') - \tau \log \pi(a' | s') \right] \right], \]

next state's value entropy

Soft Bellman equation: \( Q^*_\tau \) is unique solution to

\[ T_\tau(Q^*_\tau) = Q^*_\tau \]

\( \gamma \)-contraction of soft Bellman operator:

\[ \| T_\tau(Q_1) - T_\tau(Q_2) \|_\infty \leq \gamma \| Q_1 - Q_2 \|_\infty \]
Analysis of soft policy iteration ($\eta = \frac{1-\gamma}{\tau}$)

Policy iteration

Bellman operator
Analysis of soft policy iteration \( (\eta = \frac{1-\gamma}{\tau}) \)

Policy iteration

\[ \pi(0) \xrightarrow{\text{evaluate}} Q\pi(0) \xrightarrow{\text{greedy}} \pi(1) \xrightarrow{\text{evaluate}} Q\pi(1) \xrightarrow{\text{greedy}} \pi(2) \xrightarrow{\text{...}} Q^* \xrightarrow{\pi^*} \]

Soft policy iteration

\[ \pi(0) \xrightarrow{\text{evaluate}} Q^\pi(0) \xrightarrow{\text{soft greedy}} \pi(1) \xrightarrow{\text{evaluate}} Q^\pi(1) \xrightarrow{\text{soft greedy}} \pi(2) \xrightarrow{\text{...}} Q^*_\tau \xrightarrow{\pi^*_\tau} \]
Preconditioning dramatically increases the efficiency of vanilla gradient methods even for challenging nonconvex problems!
Preconditioning dramatically increases the efficiency of vanilla gradient methods even for challenging nonconvex problems!

**Promising directions:** unveiling the power of preconditioning in

- Statistical learning
- Reinforcement learning
- Many more ...
Thanks!


https://users.ece.cmu.edu/~yuejiec/