Preconditioning Helps: Faster Convergence in Statistical and Reinforcement Learning

Yuejie Chi

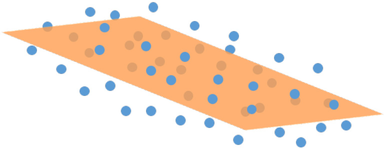
Carnegie Mellon University

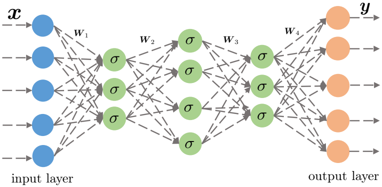
November 2020

Nonconvex problems are ubiquitous

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

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

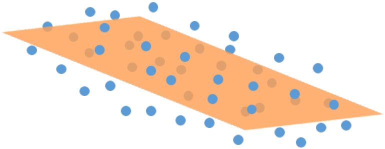


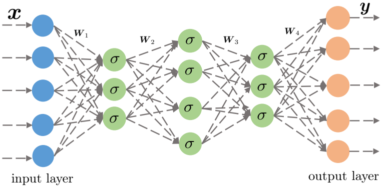


Nonconvex problems are ubiquitous

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

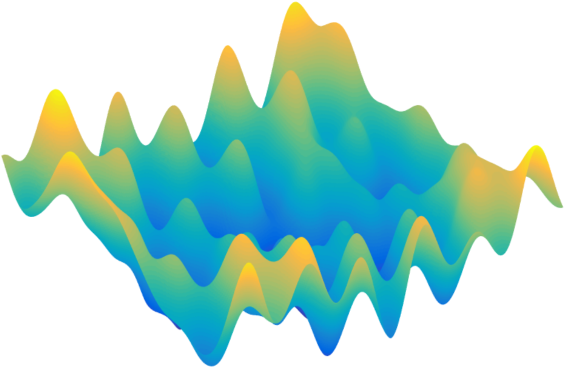
$$\text{optimize}_{\boldsymbol{x}} \quad f(\boldsymbol{x})$$





Often lead to nonconvex problems!

Nonconvex problems are hard!



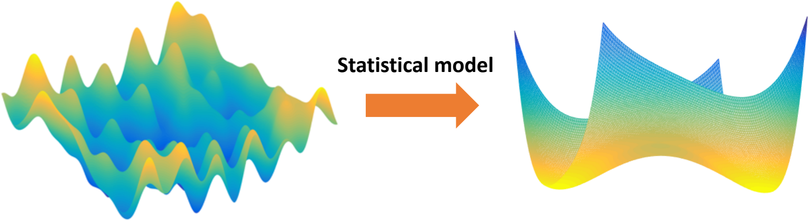
There may be bumps everywhere and exponentially many local optima, e.g. 1-layer neural networks (Auer et.al. '96)

Nonconvex problems are hard!

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

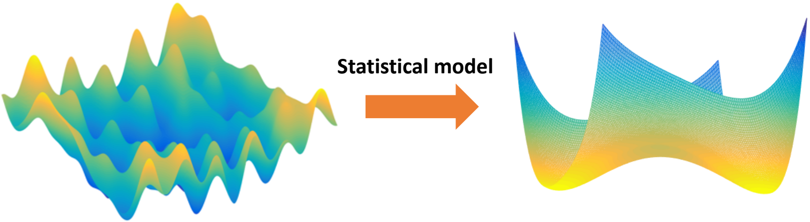
R. T. Rockafellar, in SIAM Review, 1993

Statistics meets optimization



worst case   average case

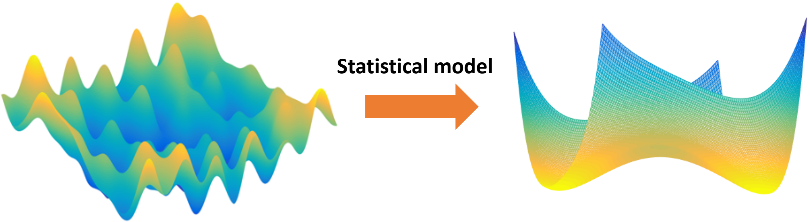
Statistics meets optimization



worst case average case

Simple algorithms can be efficient for nonconvex learning!

Statistics meets optimization



worst case

average case

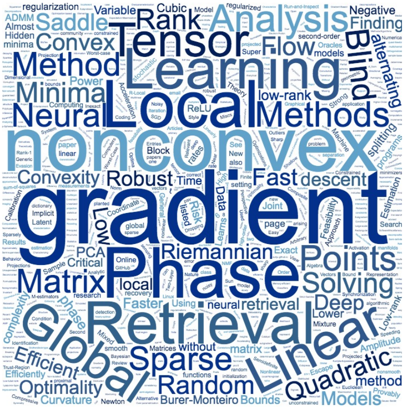
Simple algorithms can be efficient for nonconvex learning!

Vanilla gradient descent (GD):

$$\boldsymbol{x}^{t+1} = \boldsymbol{x}^{t} - \eta \nabla f(\boldsymbol{x}^{t})$$

for $t = 0, 1, \ldots$

Recent testaments: provable nonconvex optimization



"Nonconvex Optimization Meets Low-Rank Matrix Factorization: An Overview," Chi, Lu, Chen TSP 2019

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, ...

Matrix sensing/completion: Keshavan et al. '09, Jain et al. '09, Hardt '13, Jain et al. '13, Sun, Luo '15, Chen, Wainwright '15, Tu et al. '15, Zheng, Lafferty '15, Bhojanapalli et al. 16, Ge, Lee, Ma '16, Jin et al. '16, Ma et al. '17, Chen and Li '17, Cai et al. '18, Li, Zhu, Tang, Wakin '18, Charisopoulos et al. '19, ...

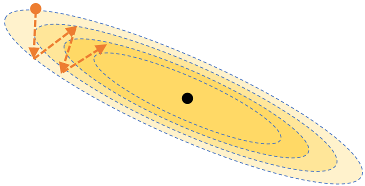
Blind deconvolution/demixing: Li et al. '16, Lee et al. '16, Cambareri, Jacques '16, Ling, Strohmer '16, Huang, Hand '16, Ma et al. '17, Zhang et al. '18, Li, Bresler '18, Dong, Shi '18, Shi, Chi '19, Qu et al. '19...

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...

This talk: acceleration via preconditioning

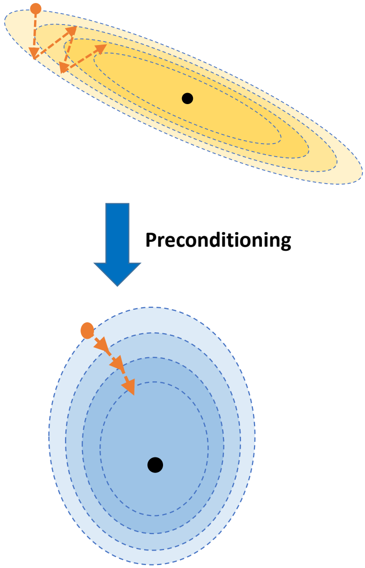


Vanilla GD:

$$\boldsymbol{x}^{t+1} = \boldsymbol{x}^t - \eta \nabla f(\boldsymbol{x}^t)$$

☹ **Slows down with ill-conditioning.**

This talk: acceleration via preconditioning



Vanilla GD:

$$\boldsymbol{x}^{t+1} = \boldsymbol{x}^{t} - \eta \nabla f(\boldsymbol{x}^{t})$$

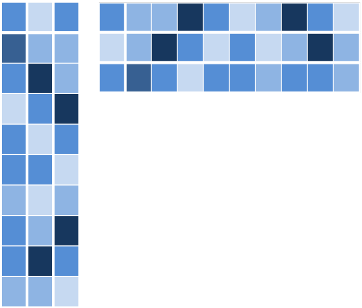
☹ **Slows down with ill-conditioning.**

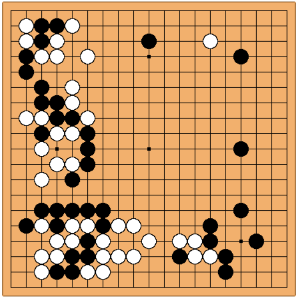
Preconditioned GD:

$$\boldsymbol{x}^{t+1} = \boldsymbol{x}^{t} - \eta \underbrace{\boldsymbol{H}_t}_{\text{preconditioner}} \nabla f(\boldsymbol{x}^{t})$$

☺ **Preconditioning accelerates convergence!**

This talk: two recent case studies





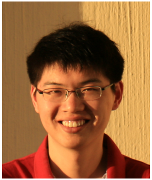
Low-rank Matrix Estimation
Statistical Learning

Policy Optimization
Reinforcement Learning

Accelerating ill-conditioned matrix estimation in statistical learning

Tian Tong

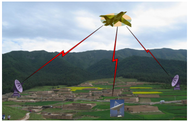
CMU



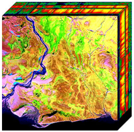
Cong Ma

Berkeley

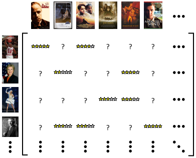
Low-rank matrices in data science



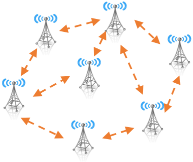
radar imaging



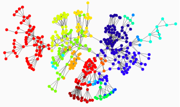
hyperspectral imaging



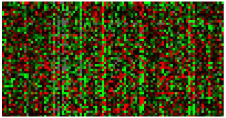
recommendation systems



localization



community detection



bioinformatics

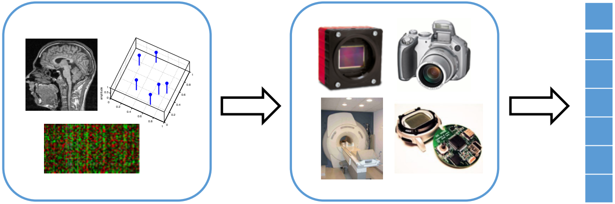
Low-rank matrices are redundant representations of latent information

Low-rank matrix sensing

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

$\mathcal{A}(\cdot)$
linear map

$\boldsymbol{y} \in \mathbb{R}^m$



$$\boldsymbol{y} = \mathcal{A}(\boldsymbol{M}) + \text{noise}$$

Recover M in the sample-starved regime:

$$\underbrace{(n_1 + n_2)r}_{\text{degree of freedom}} \lesssim \underbrace{m}_{\text{sensing budget}} \ll \underbrace{n_1 n_2}_{\text{ambient dimension}}$$

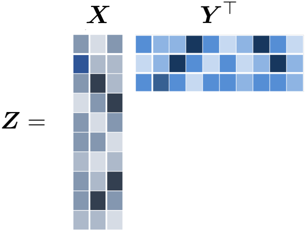
Low-rank matrix factorization

$$\min_{\text{rank}(\boldsymbol{Z})=r} \quad \frac{1}{2} \left\| \boldsymbol{y} - \mathcal{A}(\boldsymbol{Z}) \right\|_2^2$$

Low-rank matrix factorization

$$\min_{\text{rank}(\boldsymbol{Z})=r} \quad \frac{1}{2}\left\|\boldsymbol{y}-\mathcal{A}(\boldsymbol{Z})\right\|_2^2$$

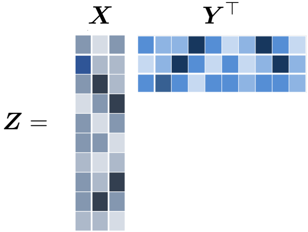
$$\Downarrow$$



Low-rank matrix factorization

$$\min_{\text{rank}(\boldsymbol{Z})=r} \quad \frac{1}{2}\|\boldsymbol{y} - \mathcal{A}(\boldsymbol{Z})\|_2^2$$

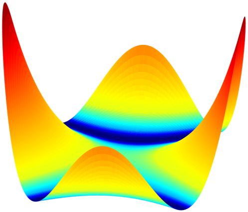
$$\Downarrow$$



$$\min_{\boldsymbol{X},\boldsymbol{Y}} \quad f(\boldsymbol{X},\boldsymbol{Y}) = \frac{1}{2}\left\|\boldsymbol{y} - \mathcal{A}(\boldsymbol{X}\boldsymbol{Y}^\top)\right\|_2^2$$

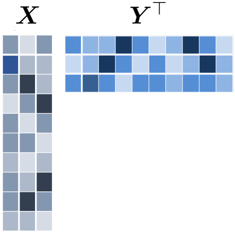
Low-rank matrix factorization

$$\min_{\textcolor{blue}{\operatorname{rank}(\boldsymbol{Z})=r}} \quad \frac{1}{2}\left\|\boldsymbol{y}-\mathcal{A}(\boldsymbol{Z})\right\|_2^2$$



$\Downarrow$

$\boldsymbol{Z} =$

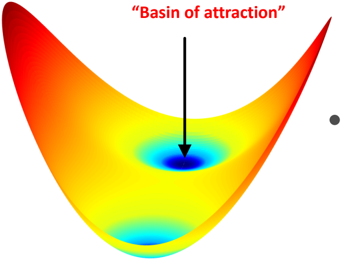


$$\min_{\boldsymbol{X},\boldsymbol{Y}} \quad f(\boldsymbol{X},\boldsymbol{Y}) = \frac{1}{2}\left\|\boldsymbol{y}-\mathcal{A}(\boldsymbol{X}\boldsymbol{Y}^\top)\right\|_2^2$$

Saves memory and computation but introduces nonconvexity!

Prior art: GD with balancing regularization

$$\min_{\boldsymbol{X},\boldsymbol{Y}} \quad f_{\text{reg}}(\boldsymbol{X},\boldsymbol{Y}) = \frac{1}{2}\left\|\boldsymbol{y} - \mathcal{A}(\boldsymbol{X}\boldsymbol{Y}^\top)\right\|_2^2 + \frac{1}{8}\left\|\boldsymbol{X}^\top\boldsymbol{X} - \boldsymbol{Y}^\top\boldsymbol{Y}\right\|_{\text{F}}^2$$



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

$$(\boldsymbol{X}_0, \boldsymbol{Y}_0) \leftarrow \mathsf{SVD}_r(\mathcal{A}^*(\boldsymbol{y}))$$

- **Gradient iterations:**

$$\boldsymbol{X}_{t+1} = \boldsymbol{X}_t - \eta\,\nabla_{\boldsymbol{X}} f_{\text{reg}}(\boldsymbol{X}_t, \boldsymbol{Y}_t)$$
$$\boldsymbol{Y}_{t+1} = \boldsymbol{Y}_t - \eta\,\nabla_{\boldsymbol{Y}} f_{\text{reg}}(\boldsymbol{X}_t, \boldsymbol{Y}_t)$$

for $t = 0, 1, \ldots$

Prior theory for vanilla GD

Theorem (Tu et al., ICML 2016)

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

$$\|\boldsymbol{X}_t \boldsymbol{Y}_t^\top - \boldsymbol{M}\|_{\mathrm{F}} \leq \varepsilon \cdot \sigma_{\min}(\boldsymbol{M})$$

- **Computational:** *within* $O\big(\kappa \log \frac{1}{\varepsilon}\big)$ *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 $\boldsymbol{M} = \boldsymbol{X}_\star \boldsymbol{Y}_\star^\top$ *is rank-*r *and has a condition number* $\kappa = \sigma_{\max}(\boldsymbol{M})/\sigma_{\min}(\boldsymbol{M})$. *For low-rank matrix sensing with i.i.d. Gaussian design, vanilla GD (with spectral initialization) achieves*

$$\|\boldsymbol{X}_t \boldsymbol{Y}_t^\top - \boldsymbol{M}\|_{\mathrm{F}} \leq \varepsilon \cdot \sigma_{\min}(\boldsymbol{M})$$

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

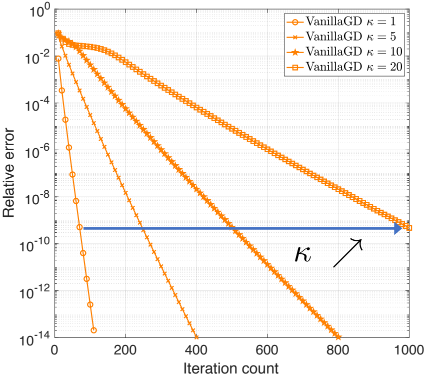
$$m \gtrsim (n_1 + n_2) r^2 \kappa^2.$$

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, ...)

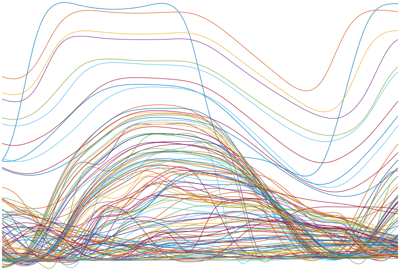
Convergence slows down for ill-conditioned matrices

$$\min_{\boldsymbol{X},\boldsymbol{Y}} \quad f(\boldsymbol{X},\boldsymbol{Y}) = \frac{1}{2}\left\|\mathcal{P}_{\Omega}(\boldsymbol{X}\boldsymbol{Y}^{\top} - \boldsymbol{M})\right\|_{\mathrm{F}}^{2}$$

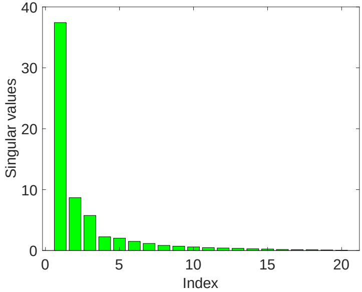


Vanilla GD converges in $O\left(\kappa \log \frac{1}{\varepsilon}\right)$ iterations.

Condition number can be large



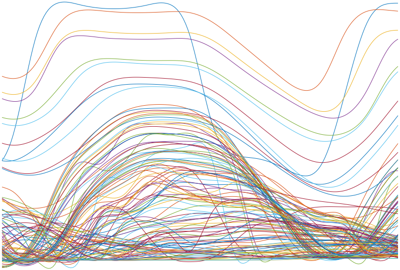
chlorine concentration levels
120 junctions, 180 time slots



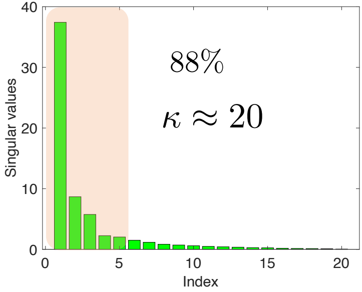
power-law spectrum

Data source: `www.epa.gov/water-research/epanet`

Condition number can be large



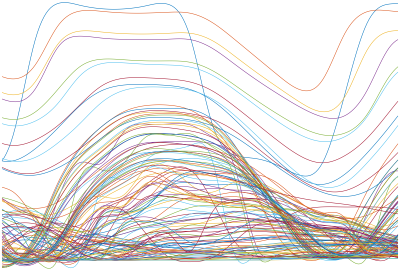
chlorine concentration levels
120 junctions, 180 time slots

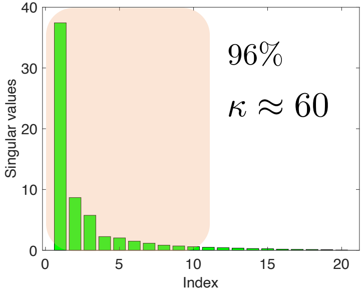


rank-5 approximation

Data source: `www.epa.gov/water-research/epanet`

Condition number can be large



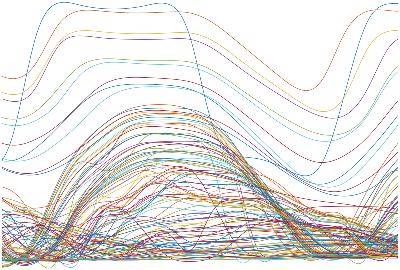


chlorine concentration levels
120 junctions, 180 time slots

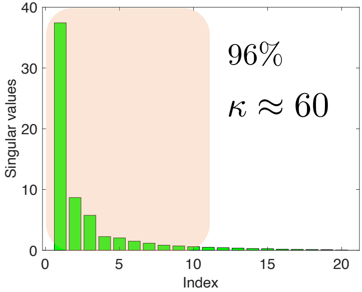
rank-10 approximation

Data source: `www.epa.gov/water-research/epanet`

Condition number can be large



chlorine concentration levels
120 junctions, 180 time slots



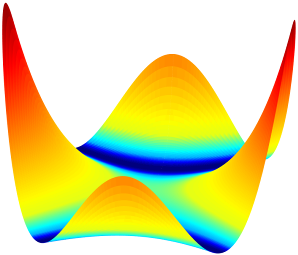
rank-10 approximation

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)

$$f(\boldsymbol{X},\boldsymbol{Y}) = \frac{1}{2}\left\|\boldsymbol{y} - \mathcal{A}(\boldsymbol{X}\boldsymbol{Y}^\top)\right\|_2^2$$



- **Spectral initialization:** find an initial point in the "basin of attraction".
- **Scaled gradient iterations:**

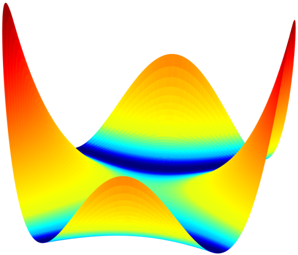
$$\boldsymbol{X}_{t+1} = \boldsymbol{X}_t - \eta\,\nabla_{\boldsymbol{X}} f(\boldsymbol{X}_t,\boldsymbol{Y}_t)\,\underbrace{(\boldsymbol{Y}_t^\top\boldsymbol{Y}_t)^{-1}}_{\texttt{preconditioner}}$$

$$\boldsymbol{Y}_{t+1} = \boldsymbol{Y}_t - \eta\,\nabla_{\boldsymbol{Y}} f(\boldsymbol{X}_t,\boldsymbol{Y}_t)\,\underbrace{(\boldsymbol{X}_t^\top\boldsymbol{X}_t)^{-1}}_{\texttt{preconditioner}}$$

for $t = 0, 1, \ldots$

A new algorithm: scaled gradient descent (ScaledGD)

$$f(\boldsymbol{X},\boldsymbol{Y}) = \frac{1}{2}\left\|\boldsymbol{y} - \mathcal{A}(\boldsymbol{X}\boldsymbol{Y}^\top)\right\|_2^2$$



- **Spectral initialization:** find an initial point in the "basin of attraction".
- **Scaled gradient iterations:**

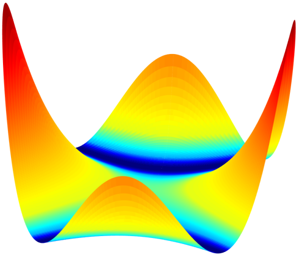
$$\boldsymbol{X}_{t+1} = \boldsymbol{X}_t - \eta\,\nabla_{\boldsymbol{X}} f(\boldsymbol{X}_t,\boldsymbol{Y}_t)\ \underbrace{(\boldsymbol{Y}_t^\top\boldsymbol{Y}_t)^{-1}}_{\texttt{preconditioner}}$$

$$\boldsymbol{Y}_{t+1} = \boldsymbol{Y}_t - \eta\,\nabla_{\boldsymbol{Y}} f(\boldsymbol{X}_t,\boldsymbol{Y}_t)\ \underbrace{(\boldsymbol{X}_t^\top\boldsymbol{X}_t)^{-1}}_{\texttt{preconditioner}}$$

for $t = 0, 1, \ldots$

A new algorithm: scaled gradient descent (ScaledGD)

$$f(\boldsymbol{X}, \boldsymbol{Y}) = \frac{1}{2}\left\| \boldsymbol{y} - \mathcal{A}(\boldsymbol{X}\boldsymbol{Y}^\top)\right\|_2^2$$



- **Spectral initialization:** find an initial point in the "basin of attraction".
- **Scaled gradient iterations:**

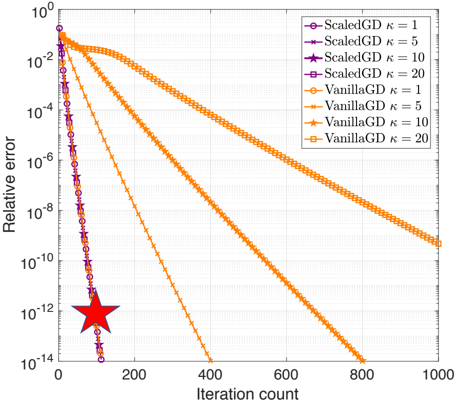
$$\boldsymbol{X}_{t+1} = \boldsymbol{X}_t - \eta \nabla_{\boldsymbol{X}} f(\boldsymbol{X}_t, \boldsymbol{Y}_t) \underbrace{(\boldsymbol{Y}_t^\top \boldsymbol{Y}_t)^{-1}}_{\texttt{preconditioner}}$$

$$\boldsymbol{Y}_{t+1} = \boldsymbol{Y}_t - \eta \nabla_{\boldsymbol{Y}} f(\boldsymbol{X}_t, \boldsymbol{Y}_t) \underbrace{(\boldsymbol{X}_t^\top \boldsymbol{X}_t)^{-1}}_{\texttt{preconditioner}}$$

for $t = 0, 1, \ldots$

ScaledGD is a *preconditioned* gradient method
without balancing regularization!

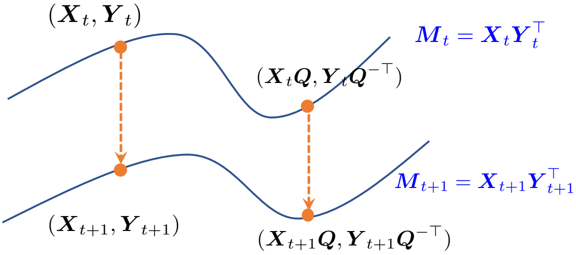
ScaledGD for low-rank matrix completion



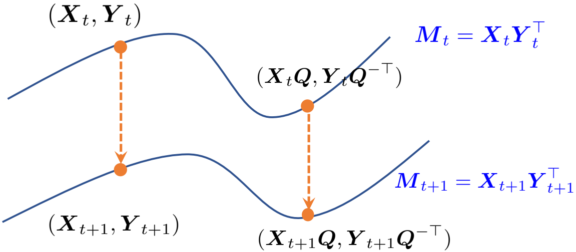
Huge computational saving: ScaledGD converges in an κ-independent manner with a minimal overhead!

A closer look at ScaledGD

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

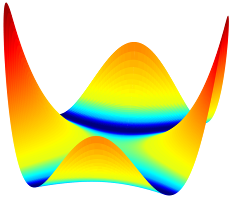


Invariance to invertible transforms: (Tanner and Wei, '16; Mishra '16)



New distance metric as Lyapunov function:

$$\mathrm{dist}^2\left(\begin{bmatrix}\boldsymbol{X}\\ \boldsymbol{Y}\end{bmatrix}, \begin{bmatrix}\boldsymbol{X}_\star\\ \boldsymbol{Y}_\star\end{bmatrix}\right) = \inf_{\boldsymbol{Q}\in\mathrm{GL}(r)} \left\|(\boldsymbol{X}\boldsymbol{Q} - \boldsymbol{X}_\star)\boldsymbol{\Sigma}_\star^{1/2}\right\|_{\mathrm{F}}^2 + \left\|(\boldsymbol{Y}\boldsymbol{Q}^{-\top} - \boldsymbol{Y}_\star)\boldsymbol{\Sigma}_\star^{1/2}\right\|_{\mathrm{F}}^2$$



$+$ 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

$$\|\boldsymbol{X}_t \boldsymbol{Y}_t^\top - \boldsymbol{M}\|_{\mathrm{F}} \lesssim \varepsilon \cdot \sigma_{\min}(\boldsymbol{M})$$

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

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

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

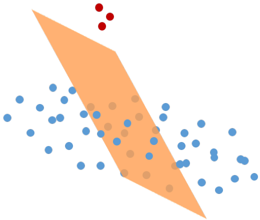
$$\|\boldsymbol{X}_t \boldsymbol{Y}_t^\top - \boldsymbol{M}\|_{\mathrm{F}} \lesssim \varepsilon \cdot \sigma_{\min}(\boldsymbol{M})$$

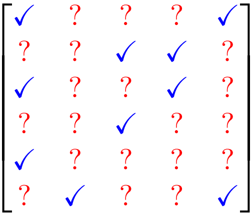
- **Computational:** *within* $O\big(\log \frac{1}{\varepsilon}\big)$ *iterations;*
- **Statistical:** *the sample complexity satisfies*

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

Strict improvement over Tu et al.: ScaledGD provably accelerates vanilla GD at the same sample complexity!

ScaledGD works more broadly





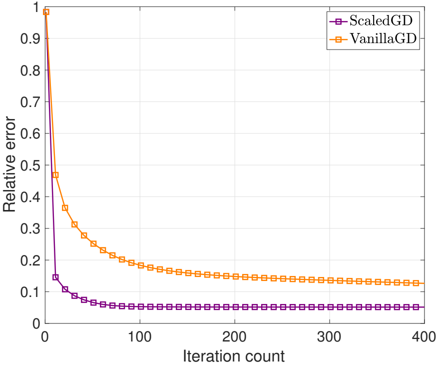
	Robust PCA		Matrix completion	
Algorithms	corruption fraction	iteration complexity	sample complexity	iteration complexity
GD	$\frac{1}{\mu r^{3/2}\kappa^{3/2} \vee \mu r \kappa^2}$	$\kappa \log \frac{1}{\varepsilon}$	$(\mu \vee \log n)\mu n r^2 \kappa^2$	$\kappa \log \frac{1}{\varepsilon}$
ScaledGD	$\frac{1}{\mu r^{3/2}\kappa}$	$\log \frac{1}{\varepsilon}$	$(\mu\kappa^2 \vee \log n)\mu n r^2 \kappa^2$	$\log \frac{1}{\varepsilon}$

Huge computation savings at comparable sample complexities!

Code available at `https://github.com/Titan-Tong/ScaledGD`

Numerical stability

ScaledGD converges faster than vanilla GD in a small number of iterations (they eventually reach the same accuracy).

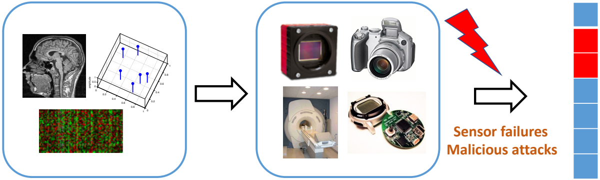


Outlier-corrupted low-rank matrix sensing

$\boldsymbol{M} \in \mathbb{R}^{n_1 \times n_2}$
$\mathrm{rank}(\boldsymbol{M}) = r$

$\mathcal{A}(\cdot)$
linear map

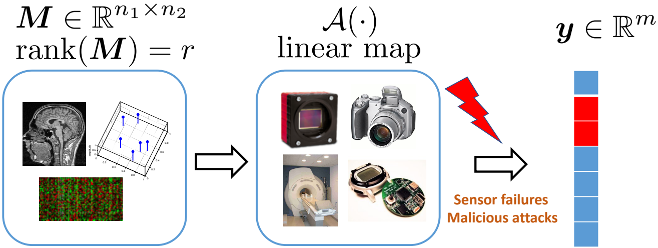
$\boldsymbol{y} \in \mathbb{R}^m$



$$\boldsymbol{y} = \mathcal{A}(\boldsymbol{M}) + \underbrace{\text{sparse outliers}}$$

a small fraction (e.g. $p_s \approx 5\%$)

Outlier-corrupted low-rank matrix sensing

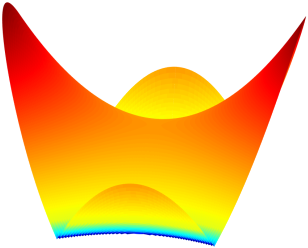


$$y \quad = \quad \mathcal{A}(M) + \underbrace{\text{sparse outliers}}_{\text{a small fraction (e.g. } p_s \approx 5\%)}$$

Least absolute deviation (LAD)

$$\min_{X,Y} \quad f(X,Y) = \frac{1}{2} \left\| y - \mathcal{A}(XY^\top) \right\|_1$$

Scaled subgradient methods



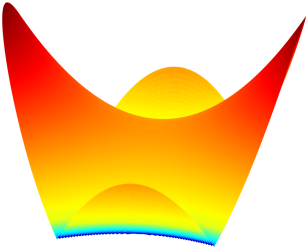
Scaled subgradient iterations:

$$\boldsymbol{X}_{t+1} = \boldsymbol{X}_t - \eta_t \, \partial_{\boldsymbol{X}} f(\boldsymbol{X}_t, \boldsymbol{Y}_t) \underbrace{(\boldsymbol{Y}_t^\top \boldsymbol{Y}_t)^{-1}}_{\texttt{preconditioner}}$$

$$\boldsymbol{Y}_{t+1} = \boldsymbol{Y}_t - \eta_t \, \partial_{\boldsymbol{Y}} f(\boldsymbol{X}_t, \boldsymbol{Y}_t) \underbrace{(\boldsymbol{X}_t^\top \boldsymbol{X}_t)^{-1}}_{\texttt{preconditioner}}$$

where η_t is set as Polyak's or geometric decaying stepsize.

Scaled subgradient methods



Scaled subgradient iterations:

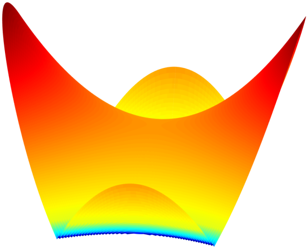
$$X_{t+1} = X_t - \eta_t \, \partial_X f(X_t, Y_t) \underbrace{(Y_t^\top Y_t)^{-1}}_{\texttt{preconditioner}}$$

$$Y_{t+1} = Y_t - \eta_t \, \partial_Y f(X_t, Y_t) \underbrace{(X_t^\top X_t)^{-1}}_{\texttt{preconditioner}}$$

where η_t is set as Polyak's or geometric decaying stepsize.

	matrix sensing	**quadratic sensing**
Subgradient Method (Charisopoulos et al, '19)	$\frac{\kappa}{(1-2p_s)^2} \log \frac{1}{\varepsilon}$	$\frac{r\kappa}{(1-2p_s)^2} \log \frac{1}{\varepsilon}$
ScaledSM	$\frac{1}{(1-2p_s)^2} \log \frac{1}{\varepsilon}$	$\frac{r}{(1-2p_s)^2} \log \frac{1}{\varepsilon}$

Scaled subgradient methods



Scaled subgradient iterations:

$$\boldsymbol{X}_{t+1} = \boldsymbol{X}_t - \eta_t \, \partial_{\boldsymbol{X}} f(\boldsymbol{X}_t, \boldsymbol{Y}_t) \underbrace{(\boldsymbol{Y}_t^\top \boldsymbol{Y}_t)^{-1}}_{\texttt{preconditioner}}$$

$$\boldsymbol{Y}_{t+1} = \boldsymbol{Y}_t - \eta_t \, \partial_{\boldsymbol{Y}} f(\boldsymbol{X}_t, \boldsymbol{Y}_t) \underbrace{(\boldsymbol{X}_t^\top \boldsymbol{X}_t)^{-1}}_{\texttt{preconditioner}}$$

where η_t is set as Polyak's or geometric decaying stepsize.

	matrix sensing	**quadratic sensing**
Subgradient Method (Charisopoulos et al, '19)	$\frac{\kappa}{(1-2p_s)^2} \log \frac{1}{\varepsilon}$	$\frac{r\kappa}{(1-2p_s)^2} \log \frac{1}{\varepsilon}$
ScaledSM	$\frac{1}{(1-2p_s)^2} \log \frac{1}{\varepsilon}$	$\frac{r}{(1-2p_s)^2} \log \frac{1}{\varepsilon}$

Robustness to both ill-conditioning and adversarial corruptions!

Accelerating convergence of policy optimization in reinforcement learning

Shicong Cen
CMU

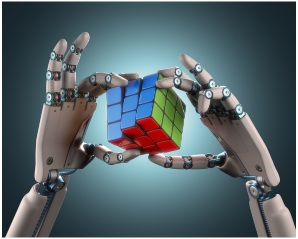
Chen Cheng
Stanford

Yuxin Chen
Princeton

Yuting Wei
CMU

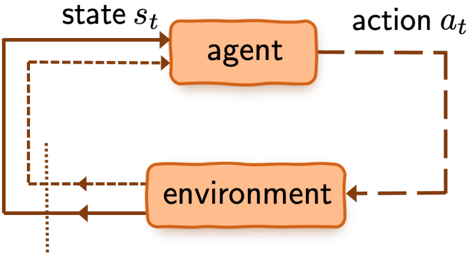
Reinforcement learning (RL)

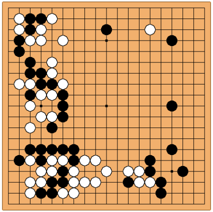
In RL, an agent learns by interacting with an environment.



Policy optimization is a major driver to these successes.

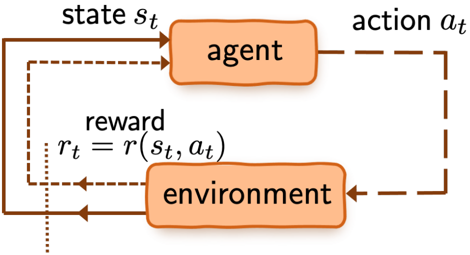
Markov decision process (MDP)

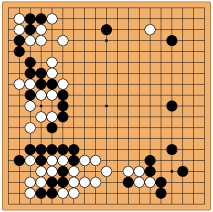




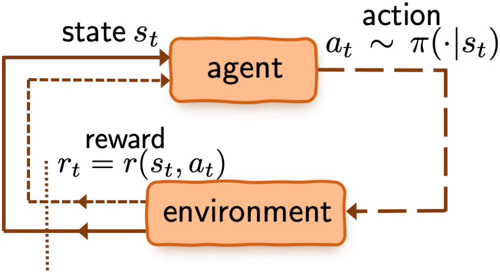
- $\mathcal{S}$: state space
- $\mathcal{A}$: action space

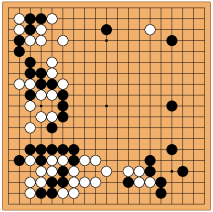
Markov decision process (MDP)





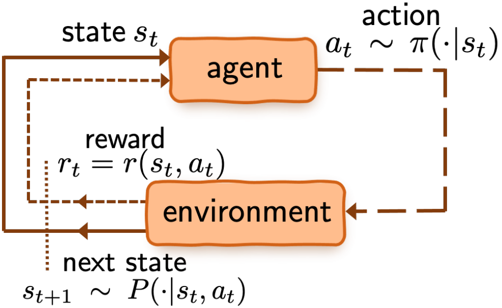
- $\mathcal{S}$: state space
- $\mathcal{A}$: action space
- $r(s, a) \in [0, 1]$: immediate reward

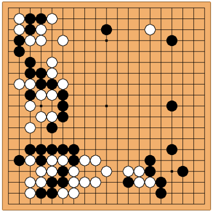




- $\mathcal{S}$: state space
- $\mathcal{A}$: action space
- $r(s, a) \in [0, 1]$: immediate reward
- $\pi(\cdot|s)$: policy (or action selection rule)

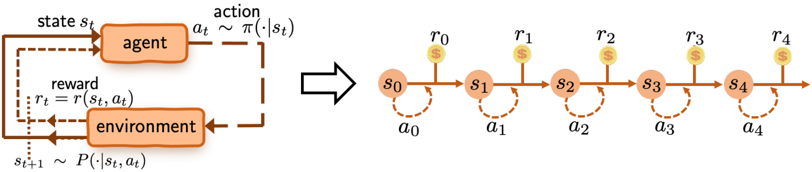
Markov decision process (MDP)





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

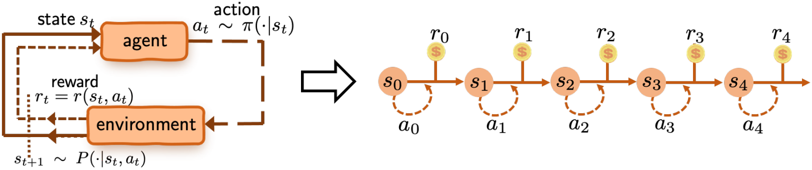
Value function and Q-function



Value function and **Q function** of policy π:

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

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



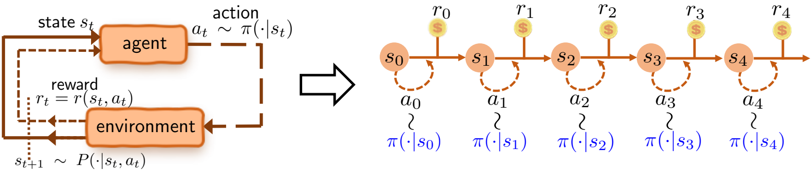
Value function and **Q function** of policy π:

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

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

- $\gamma \in [0,1)$ is the discount factor; $\frac{1}{1-\gamma}$ is effective horizon
- Expectation is w.r.t. the sampled trajectory under π

Entropy-regularized RL

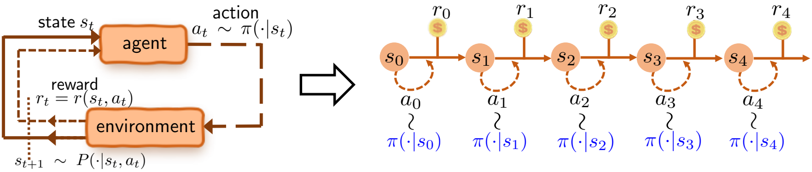


To encourage exploration, promote the stochasticity of the policy using the **"soft"** value function:

$$\forall s \in \mathcal{S}: \qquad V_\tau^\pi(s) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \big(r_t - \tau \log \pi(a_t|s_t)\big) \,\Big|\, s_0 = s\right]$$

where τ is the entropy regularization parameter.

Entropy-regularized RL



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

$$\forall s \in \mathcal{S}: \qquad V_\tau^\pi(s) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \big(r_t - \tau \log \pi(a_t|s_t)\big) \,\Big|\, s_0 = s\right]$$

where τ is the entropy regularization parameter.

Goal: find the optimal policy $\pi_\tau^\star$ that maximize $V_\tau^\pi(s)$

Policy gradient methods

Given an initial state distribution $s \sim \rho$, find policy π such that

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

Policy gradient methods

Given an initial state distribution $s \sim \rho$, find policy π such that

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

$$\Downarrow$$

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 π such that

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

$\Downarrow$

softmax parameterization:

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

$$\text{maximize}_\theta \quad V_\tau^{\pi_\theta}(\rho) := \mathbb{E}_{s\sim\rho}\left[V_\tau^{\pi_\theta}(s)\right]$$

Policy gradient methods

Given an initial state distribution $s \sim \rho$, find policy π such that

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

$$\Downarrow$$

softmax parameterization:

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

$$\text{maximize}_\theta \quad V_\tau^{\pi_\theta}(\rho) := \mathbb{E}_{s\sim\rho}\left[V_\tau^{\pi_\theta}(s)\right]$$

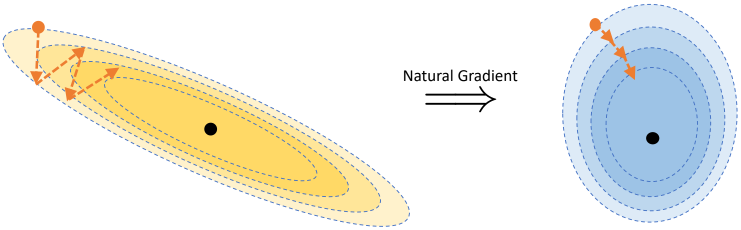
Policy gradient methods (Sutton et al., 2000)

For $t = 0, 1, \cdots$

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

where η *is the learning rate.*

Natural policy gradient



Natural policy gradient (Kakade, 2002)

For $t = 0, 1, \cdots$

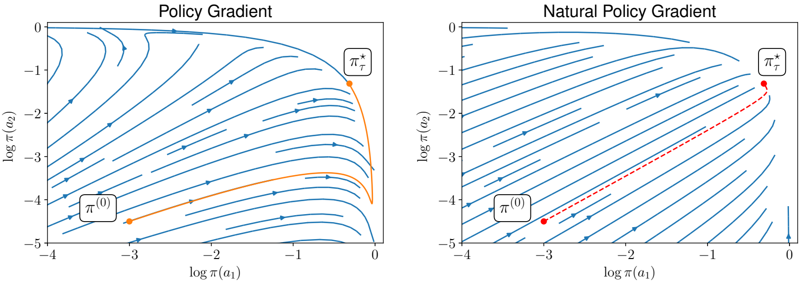
$$\theta^{(t+1)} = \theta^{(t)} + \eta (\mathcal{F}_\rho^\theta)^\dagger \nabla_\theta V_\tau^{\pi_\theta^{(t)}}(\rho)$$

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

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

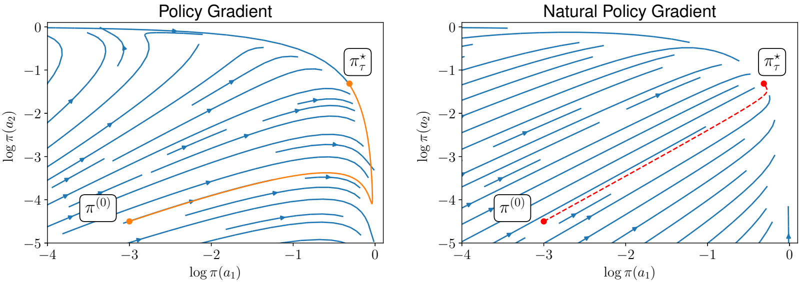
Natural gradient helps!

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



Natural gradient helps!

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

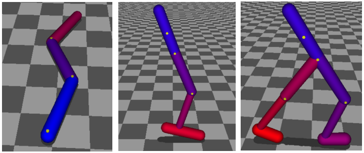


NPG follows a more direct path to find the optimal policy.

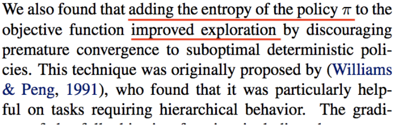
Unreasonable effectiveness in practice

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)



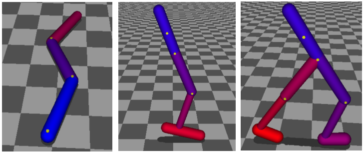
We also found that adding the entropy of the policy π to the objective function improved exploration by discouraging premature convergence to suboptimal deterministic policies. This technique was originally proposed by (Williams & Peng, 1991), who found that it was particularly helpful on tasks requiring hierarchical behavior. The gradi-

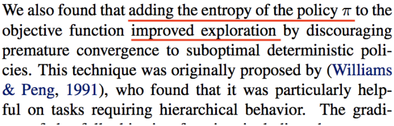
A3C (Mnih et al., 2016)
SAC (Haarnoja et al., 2018)

Unreasonable effectiveness in practice

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.





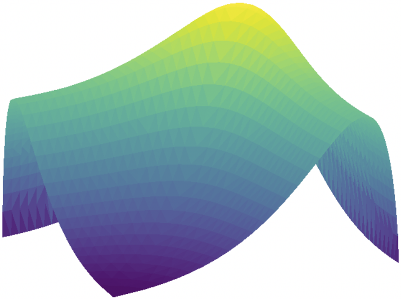
We also found that adding the entropy of the policy π to the objective function improved exploration by discouraging premature convergence to suboptimal deterministic policies. This technique was originally proposed by (Williams & Peng, 1991), who found that it was particularly helpful on tasks requiring hierarchical behavior. The gradi-

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?

Theoretical challenges: non-concavity

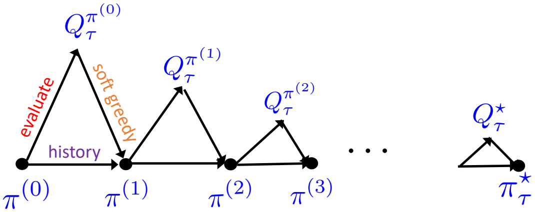


Recent breakthroughs on understanding global convergence of

- 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



Entropy-regularized NPG (Tabular setting)

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

$$\pi^{(t+1)}(a|s) \propto \underbrace{\pi^{(t)}(a|s)}_{\textit{current policy}}{}^{1-\frac{\eta\tau}{1-\gamma}} \underbrace{\exp(Q_\tau^{\pi^{(t)}}(s,a)/\tau)}_{\textit{soft greedy}}{}^{\frac{\eta\tau}{1-\gamma}}$$

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

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

Linear convergence with exact gradient

Exact oracle: perfect evaluation of $Q_\tau^{\pi^{(t)}}$ 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:**

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

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

$$C_1 = \|Q_\tau^\star - Q_\tau^{(0)}\|_\infty + 2\tau \left(1 - \frac{\eta\tau}{1-\gamma}\right) \|\log \pi_\tau^\star - \log \pi^{(0)}\|_\infty.$$

Linear convergence with exact gradient

Exact oracle: perfect evaluation of $Q_\tau^{\pi^{(t)}}$ 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 log policies:**

$$\|\log \pi_\tau^\star - \log \pi^{(t+1)}\|_\infty \leq 2C_1 \tau^{-1} (1-\eta\tau)^t$$

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

$$C_1 = \|Q_\tau^\star - Q_\tau^{(0)}\|_\infty + 2\tau \left(1 - \frac{\eta\tau}{1-\gamma}\right) \|\log \pi_\tau^\star - \log \pi^{(0)}\|_\infty.$$

Implications

To reach $\|Q_\tau^\star - Q_\tau^{(t+1)}\|_\infty \leq \epsilon$, 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 ($\eta = \frac{1-\gamma}{\tau}$):**

$$\frac{1}{1-\gamma} \log\left(\frac{\|Q_\tau^\star - Q_\tau^{(0)}\|_\infty \gamma}{\epsilon}\right)$$

Implications

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

- **General learning rates ($0 < \eta < \frac{1-\gamma}{\tau}$):**

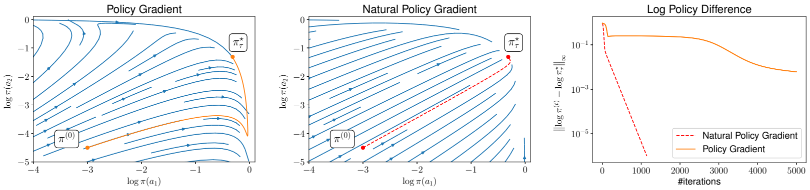
$$\frac{1}{\eta\tau} \log\left(\frac{C_1\gamma}{\epsilon}\right)$$

- **Soft policy iteration ($\eta = \frac{1-\gamma}{\tau}$):**

$$\frac{1}{1-\gamma} \log\left(\frac{\|Q_\tau^\star - Q_\tau^{(0)}\|_\infty \gamma}{\epsilon}\right)$$

Global linear convergence of entropy-regularized NPG
at a rate independent of $|\mathcal{S}|, |\mathcal{A}|$!

Comparisons with entropy-regularized PG



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

$$V_\tau^\star(\rho) - V_\tau^{(t)}(\rho) \le \Big(V_\tau^\star(\rho) - V_\tau^{(0)}(\rho)\Big)$$

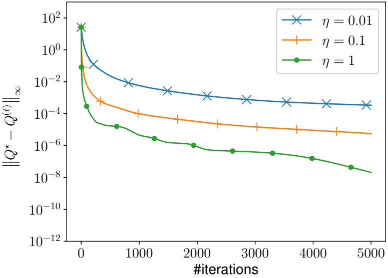
$$\cdot \exp\left(- \frac{(1-\gamma)^4 t}{(8/\tau + 4 + 8\log|\mathcal{A}|)|\mathcal{S}|} \left\| \frac{d_\rho^{\pi_\tau^\star}}{\rho} \right\|_\infty^{-1} \min_s \rho(s) \underbrace{\Big(\inf_{0\le k\le t-1} \min_{s,a} \pi^{(k)}(a|s) \Big)^2}_{\text{unclear dependence with } |\mathcal{S}|,\, |\mathcal{A}|,\, \gamma} \right)$$

Much faster convergence of entropy-regularized NPG
at a **dimension-free** rate!

Aside: entropy helps!

Vanilla NPG

$\tau = 0$

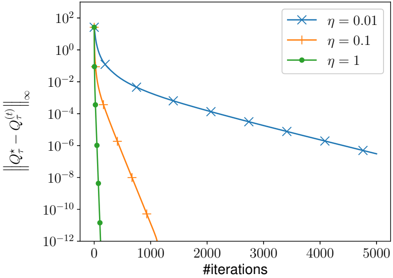


Sublinear rate: $\frac{1}{\min\{\eta,(1-\gamma)^2\}\epsilon}$

(Agarwal et.al. 2019)

Regularized NPG

$\tau = 0.001$



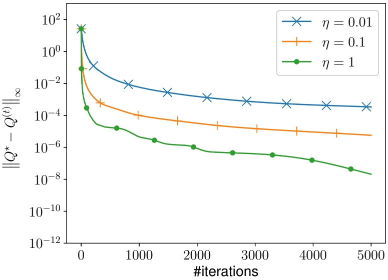
Linear rate: $\frac{1}{\eta\tau}\log\left(\frac{1}{\epsilon}\right)$

Ours

Aside: entropy helps!

Vanilla NPG

$\tau = 0$

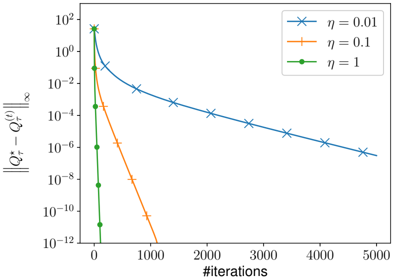


Sublinear rate: $\frac{1}{\min\{\eta,(1-\gamma)^2\}\epsilon}$

(Agarwal et.al. 2019)

Regularized NPG

$\tau = 0.001$



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

$$\mathcal{T}(Q)(s,a) := \underbrace{r(s,a)}_{\text{immediate reward}} + \gamma \underset{s' \sim P(\cdot|s,a)}{\mathbb{E}} \underbrace{\left[\max_{a' \in \mathcal{A}} Q(s',a')\right]}_{\text{next state's value}}$$

- one-step look-ahead

Recall: Bellman's optimality principle

Bellman operator

$$\mathcal{T}(Q)(s,a) := \underbrace{r(s,a)}_{\text{immediate reward}} + \gamma \mathop{\mathbb{E}}_{s' \sim P(\cdot|s,a)} \underbrace{\left[\max_{a' \in \mathcal{A}} Q(s',a')\right]}_{\text{next state's value}}$$

- one-step look-ahead

Bellman equation: $Q^\star$ is *unique* solution to

$$\mathcal{T}(Q^\star) = Q^\star$$

γ-contraction of Bellman operator:

$$\|\mathcal{T}(Q_1) - \mathcal{T}(Q_2)\|_\infty \le \gamma \|Q_1 - Q_2\|_\infty$$

Richard Bellman

Soft Bellman operator

$$\mathcal{T}_\tau(Q)(s,a) := \underbrace{r(s,a)}_{\text{immediate reward}} + \gamma \mathop{\mathbb{E}}_{s' \sim P(\cdot|s,a)} \left[\max_{\pi(\cdot|s')} \mathop{\mathbb{E}}_{a' \sim \pi(\cdot|s')} \Big[\underbrace{Q(s',a')}_{\text{next state's value}} - \underbrace{\tau \log \pi(a'|s')}_{\text{entropy}} \Big] \right],$$

Soft Bellman operator

Soft Bellman operator

$$\mathcal{T}_\tau(Q)(s,a) := \underbrace{r(s,a)}_{\text{immediate reward}} + \gamma \underset{s' \sim P(\cdot|s,a)}{\mathbb{E}} \left[\max_{\pi(\cdot|s')} \underset{a' \sim \pi(\cdot|s')}{\mathbb{E}} \left[\underbrace{Q(s',a')}_{\text{next state's value}} - \underbrace{\tau \log \pi(a'|s')}_{\text{entropy}} \right] \right],$$

Soft Bellman equation: $Q_\tau^\star$ is *unique* solution to

$$\mathcal{T}_\tau(Q_\tau^\star) = Q_\tau^\star$$

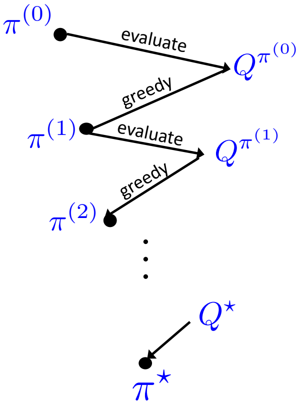
γ**-contraction of soft Bellman operator:**

$$\|\mathcal{T}_\tau(Q_1) - \mathcal{T}_\tau(Q_2)\|_\infty \le \gamma \|Q_1 - Q_2\|_\infty$$

Richard Bellman

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

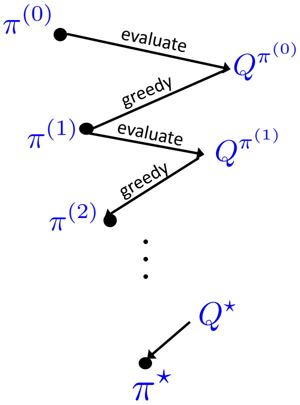
Policy iteration



Bellman operator

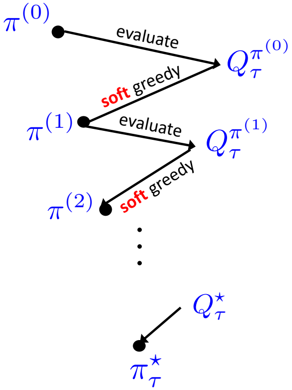
Analysis of soft policy iteration ($\eta = \frac{1-\gamma}{\tau}$)

Policy iteration



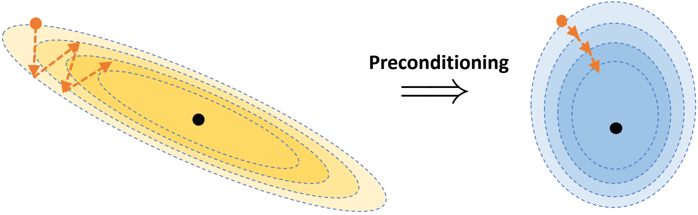
Bellman operator

Soft policy iteration



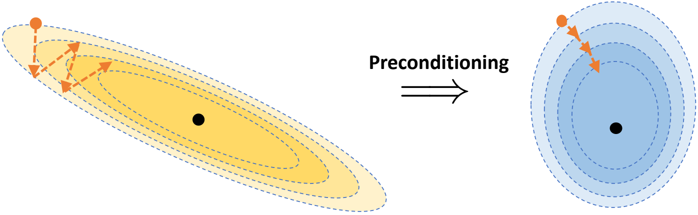
Soft Bellman operator

Concluding remarks



Preconditioning dramatically increases the efficiency of vanilla gradient methods even for challenging nonconvex problems!

Concluding remarks



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!

- Accelerating Ill-Conditioned Low-Rank Matrix Estimation via **Scaled Gradient Descent**, `arXiv 2005.08898`.
- Low-Rank Matrix Recovery with **Scaled Subgradient Methods**: Fast and Robust Convergence Without the Condition Number, `arXiv 2010.13364`.
- Fast Global Convergence of **Natural Policy Gradient** Methods with Entropy Regularization, `arXiv 2007.06558`.

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