Escape saddle points by a simple gradient-descent based algorithm

Chenyi Zhang
Tsinghua University

Joint work with Tongyang Li

Nonconvex optimization

Problem: $f: \mathbb{R}^n \to \mathbb{R}, \quad \arg\min_x f(x)$ $f(\cdot)$: non-convex function

Core topic in machine learning and optimization theory

A wide range of applications: matrix & tensor decomposition, neural networks, …

Nonconvex optimization

The most common method for nonconvex optimization: gradient descent (GD)

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

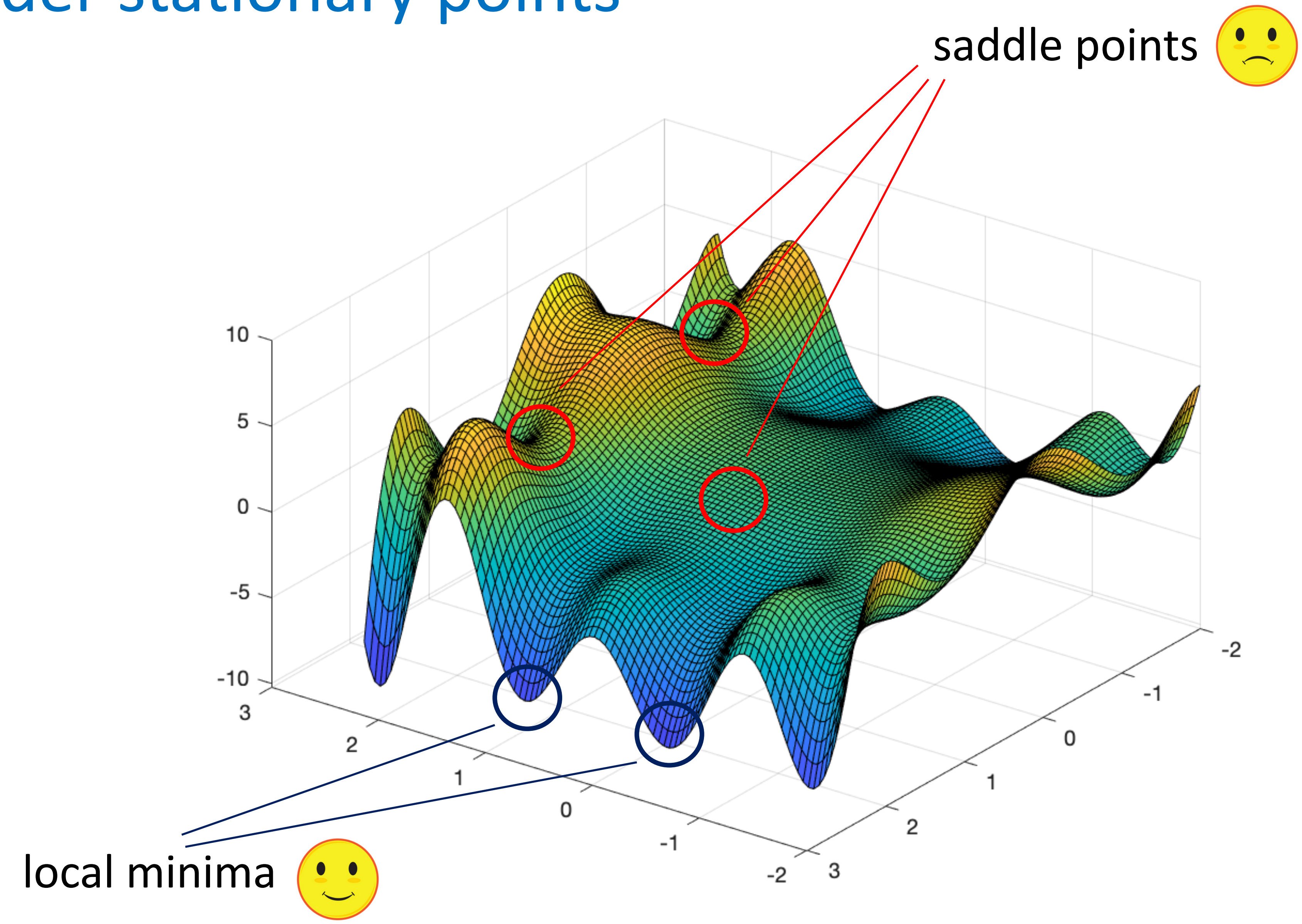
If f is ℓ-smooth: $\|\nabla f(w_1) - \nabla f(w_2)\| \leq \ell \|w_1 - w_2\| \quad \forall\, w_1, w_2 \in \mathbb{R}^n$,

$$t = O(\ell/\epsilon^2) \Rightarrow \|\nabla f(x_t)\| \leq \epsilon.$$

This is an ϵ-approx. first-order stationary point.

Question: Is this good enough?

First order stationary points



Nonconvex optimization

Common fact about many learning problems:

- Ubiquitous saddle points (including local maxima) can give highly suboptimal solutions

- We would want to escape from saddle points, but finding an ϵ-approx. local minimum x_ϵ suffices:

$$\|\nabla f(x_\epsilon)\| \leq \epsilon, \quad \lambda_{\min}(\nabla^2 f(x_\epsilon)) \geq -\sqrt{\rho\epsilon}.$$

Here f is ρ-Hessian Lipschitz: $\|\nabla^2 f(w_1) - \nabla^2 f(w_2)\| \leq \rho\|w_1 - w_2\| \quad \forall w_1, w_2 \in \mathbb{R}^n$.

Escaping from saddle points

Oracle	Reference	Iterations	Simplicity
Hessian	Nesterov and Polyak 2006	$O(1/\epsilon^{1.5})$	Single-loop
Hessian-vector product	Agarwal et al.2017; Carmon et al. 2018	$\tilde{O}(\log n/\epsilon^{1.76})$	Nested-loop
Gradient	Xu et al. 2017; Allen-Zhu et al. 2017	$\tilde{O}(\log n/\epsilon^{1.75})$	Nested-loop
Gradient	Jin et al. 2017, 2019	$\tilde{O}(\log^4 n/\epsilon^2)$	Single-loop
Gradient	Jin et al. 2018	$\tilde{O}(\log^6 n/\epsilon^{1.75})$	Single-loop

Our result:

Gradient	**this work**	$\tilde{O}(\log n/\epsilon^{1.75})$	Single-loop

Two main considerations:

Complexity:

- Reduce the dependence on both accuracy ϵ and dimension n

Simplicity:

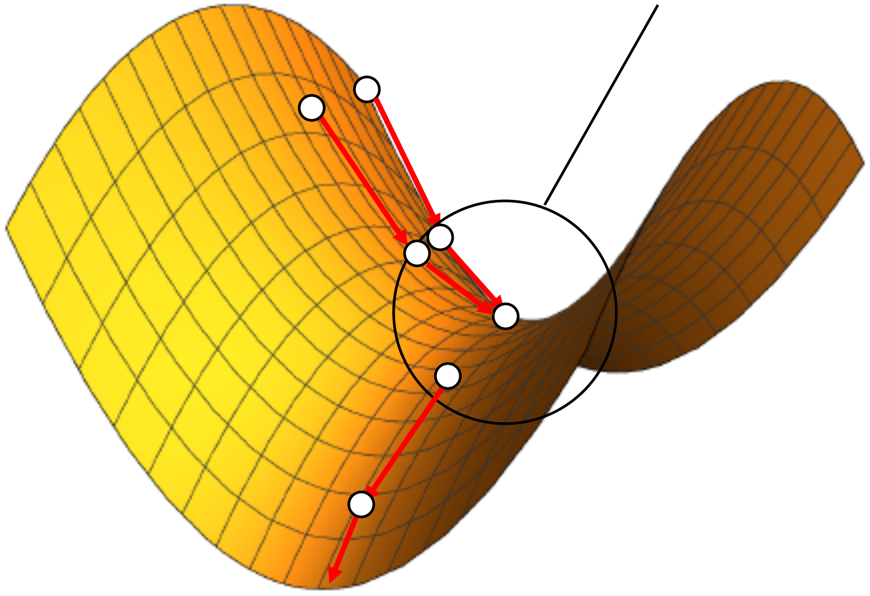
- Simpler oracle
- Simpler structure (single-loop, less hyperparameters)

Escaping from saddle points

The main idea: perturbed gradient descent

Main thoughts:

- **Radius of perturbation**: If it is too large, then we may backtrack too much. If it is too small, we may need many iterations to leave the saddle.
- **Way of perturbation**: What's the most efficient approach?
- **Gradient descent**: Faster versions?



Perturbed accelerated gradient descent (simplified)

Jin et al. 2017

- Throughout the algorithm, use Nesterov's **accelerated gradient descent** (AGD):

$$y_t \leftarrow x_t + (1-\theta)v_t, \; x_{t+1} \leftarrow y_t - \eta \nabla f(y_t), \; v_{t+1} \leftarrow x_{t+1} - x_t.$$

- If $\|\nabla f(x_t)\| \leq \epsilon$ and no perturbation happened in $O(\log n)$ steps:
  Perturb by the **uniform distribution** in the ball of radius $r = \Theta(\epsilon / \log^5 n)$.

  Bottleneck of the algorithm

  Fact: Perturbed AGD takes $O(\log n)$ steps to decrease the the Hamiltonian

$$f(x_t) + \|v_t\|^2 / 2\eta$$

  by $\Omega(1/\log^5 n)$, convergence rate $O(1/\epsilon^{1.75})$. Total cost: $\tilde{\Theta}(\log^6 n / \epsilon^{1.75})$.

- **Question**: can we do better than uniform perturbation and improve dependence on $\log n$?

Better than uniform perturbation

Intuition: add perturbation in the negative curvature direction

Observation 1: Consider the Hessian matrix at the saddle point, its eigenvectors with negative eigenvalue indicate negative curvature direction

- Agarwal et al. 2017; Carmon et al. 2018: it takes $O(\log n)$ Hessian-vector products to find negative curvature by Hessian power method.

Observation 2: For Hessian-Lipschitz functions, Hessian-vector product can be approximated via two gradient queries of two near enough points:

$$\mathcal{H}(\mathbf{x}) \cdot \Delta\mathbf{x} = \nabla f(\mathbf{x} + \Delta\mathbf{x}) - \nabla f(\mathbf{x}) + O(\|\Delta\mathbf{x}\|^2)$$

- Xu et al. 2017; Allen-Zhu et al. 2017: it takes $O(\log n)$ gradient calls to find negative curvature and then escape from saddle points.

End of the story?

Simplicity

Simplicity is of great importance in the design of optimization algorithms

- Empirical observation: simple algorithms often have good performance in practice
- It is hard to train machine learning models and adjust parameters using a complicated optimizer

Xu et al. 2017

- *Complicated for practical use*
- *Numerically instable*

Accelerated Gradient methods for Extracting NC from Noise: $\text{NEON}^+(f, \mathbf{x}, t, \mathcal{F}, U, \zeta, r)$

Input: $f, \mathbf{x}, t, \mathcal{F}, U, \zeta, r$
Generate $\mathbf{y}_0 = \mathbf{u}_0$ randomly from the sphere of an Euclidean ball of radius r
for $\tau = 0, \ldots, t$ **do**
  if $\Delta_{\mathbf{x}}(\mathbf{y}_\tau, \mathbf{u}_\tau) < -\frac{\gamma}{2}\|\mathbf{y}_\tau - \mathbf{u}_\tau\|^2$ **then**
    return $\mathbf{v} = \text{NCFind}(\mathbf{y}_{0:\tau}, \mathbf{u}_{0:\tau})$
  end if
  compute $(\mathbf{y}_{\tau+1}, \mathbf{u}_{\tau+1})$ by (14)
end for
if $\min_{\|\mathbf{y}_\tau\| \le U} \hat{f}_{\mathbf{x}}(\mathbf{y}_\tau) \le -2\mathcal{F}$ **then**
  let $\tau' = \arg\min_{\tau, \|\mathbf{y}_\tau\| \le U} \hat{f}_{\mathbf{x}}(\mathbf{y}_\tau)$
  return $\mathbf{y}_{\tau'}$
else
  return 0
end if

Simplicity

Simplicity is of great importance in the design of optimization algorithms

- Empirical observation: simple algorithms often have good performance in practice
- It is hard to train machine learning models and adjust parameters using a complex optimizer

Xu et al. 2017

- *Complicated for practical use*
- *Numerically instable*

- Can we have gradient-descent based, more numerically stable algorithms with much simpler structure which enable possible practical application, while preserving the dependence on $\log n$?
- Our work answers this question in the affirmative.

Simpler algorithm

- **Basic idea**: adopt the structure of PAGD (Jin et al. 2017)

Algorithm 2 Perturbed Accelerated Gradient Descent $(\mathbf{x}_0, \eta, \theta, \gamma, s, r, \mathscr{T})$

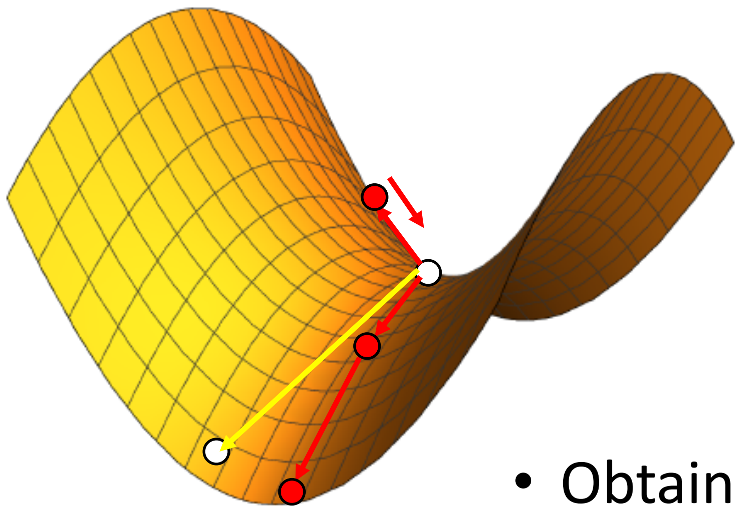
```
v_0 ← 0
for t = 0, 1, ..., do
    if ‖∇f(x_t)‖ ≤ ϵ and no perturbation in last 𝒯 steps then
        x_t ← x_t + ξ_t        ξ_t ~ Unif(𝔹_0(r))
    y_t ← x_t + (1 − θ)v_t
    x_{t+1} ← y_t − η∇f(y_t)
    v_{t+1} ← x_{t+1} − x_t
    if f(x_t) ≤ f(y_t) + ⟨∇f(y_t), x_t − y_t⟩ − (γ/2)‖x_t − y_t‖^2 then
        (x_{t+1}, v_{t+1}) ← Negative-Curvature-Exploitation(x_t, v_t, s)
```

Replace it by a simple, gradient-based subroutine that can find negative curvature near saddle points

Simpler algorithm

- **Basic idea**: adopt the structure of PAGD (Jin et al. 2017), while use a simple, gradient-based subroutine to find negative curvature near saddle points

Near a saddle point, the function is well-approximated by a quadratic function



- Add a perturbation and run AGD for some time

The total motion of AGD can be decomposed into several independent one-dimensional motions

- Obtain a vector which has a large overlap with the negative curvature direction

Simpler algorithm

- **Basic idea**: adopt the structure of PAGD (Jin et al. 2017), while use a simple, gradient-based subroutine to find negative curvature near saddle points

Accelerated Negative Curvature Finding

$\mathbf{y}_0 \leftarrow \mathrm{Uniform}(\mathbb{B}_{\tilde{\mathbf{x}}}(r))$ where $\mathbb{B}_{\tilde{\mathbf{x}}}(r)$ is the ℓ_2-norm ball centered at $\tilde{\mathbf{x}}$ with radius r;

$\mathbf{v}_0 \leftarrow \mathbf{0}$;

for $t = 1, ..., \mathscr{T}'$ **do**

  $\mathbf{z}_t \leftarrow \mathbf{y}_t + (1-\theta)\mathbf{v}_t$;

  $\mathbf{y}_{t+1} \leftarrow \mathbf{z}_t - \eta \nabla f(\mathbf{z}_t)$;

  $\mathbf{v}_{t+1} \leftarrow \mathbf{y}_{t+1} - \mathbf{y}_t$;

  $\mathbf{v}_t \leftarrow \mathbf{v}_t \cdot \frac{r}{\|\mathbf{y}_t\|}$, $\mathbf{y}_t \leftarrow \mathbf{y}_t \cdot \frac{r}{\|\mathbf{y}_t\|}$;

Output $\mathbf{y}_{\mathscr{T}}/r$.

Simplicity preserving

- No additional hyperparameters compared to original PAGD
- Approximately the same structure as PAGD

Numerically stable

- An additional renormalization step

Quantitative result

- **Basic idea**: adopt the structure of PAGD (Jin et al. 2017), while use a simple, gradient-based subroutine to find negative curvature near saddle points

Proposition (informal). *For any* $0 < \delta \leq 1$, *we specify our choice of parameters:*

$$\mathscr{T} = \tilde{O}(\log n/\epsilon^{1/4}), \quad r = \tilde{O}\Big(\frac{\delta\epsilon^{1/4}}{\mathscr{T}\sqrt{n}}\Big).$$

Then for any $\tilde{\mathbf{x}}$ *satisfying* $\lambda_{\min}(\nabla^2 f(\tilde{\mathbf{x}})) \leq -\sqrt{\rho\epsilon}$, *with probability at least* $1-\delta$, *the subroutine* ***Accelerated Negative Curvature Finding*** *outputs a unit vector* $\hat{\mathbf{e}}$ *satisfying*

$$\hat{\mathbf{e}}^T\mathcal{H}(\tilde{\mathbf{x}})\hat{\mathbf{e}} \leq -\sqrt{\rho\epsilon}/4,$$

where $\mathcal{H}$ *stands for the Hessian matrix of function* f, *using* $\tilde{O}(\log n/\epsilon^{1/4})$ *iterations.*

Putting everything together

Algorithm 2: Perturbed Accelerated Gradient Descent with Accelerated Negative Curvature Finding($\mathbf{x}_0, \eta, \theta, \gamma, s, \mathscr{T}, r$)

$\mathbf{v}_0 = \mathbf{0}$, $t_{\text{perturb}} = 0$, $\tilde{\mathbf{x}} = \mathbf{x}_0$;
for $t = 0, 1, \ldots, T$ **do**
  if $\|\nabla f(\mathbf{x}_t)\| \leq \epsilon$ *and* $t - t_{perturb} > \mathscr{T}$ **then**
    $\tilde{\mathbf{x}} = \mathbf{x}_t$;
    $\mathbf{x}_t \leftarrow \text{Uniform}(\mathbb{B}_{\tilde{\mathbf{x}}}(r))$ where $\text{Uniform}(\mathbb{B}_{\tilde{\mathbf{x}}}(r))$ is the ℓ_2-norm ball centered at $\tilde{\mathbf{x}}$ with radius r; $\mathbf{v}_t \leftarrow \mathbf{0}$, $t_{\text{perturb}} \leftarrow t$;
  if $t - t_{perturb} = \mathscr{T}$ **then**
    $\hat{\mathbf{e}} := \frac{\mathbf{x}_t - \tilde{\mathbf{x}}}{\|\mathbf{x}_t - \tilde{\mathbf{x}}\|}$; $\mathbf{x}_t \leftarrow \tilde{\mathbf{x}} - \frac{f'_{\hat{\mathbf{e}}}(\tilde{\mathbf{x}})}{4|f'_{\hat{\mathbf{e}}}(\tilde{\mathbf{x}})|}\sqrt{\frac{\epsilon}{\rho}} \cdot \hat{\mathbf{e}}$, $\mathbf{v}_t \leftarrow \mathbf{0}$;
  $\mathbf{z}_t \leftarrow \mathbf{x}_t + (1-\theta)\mathbf{v}_t$;
  $\mathbf{x}_{t+1} \leftarrow \mathbf{z}_t - \eta \nabla f(\mathbf{z}_t)$;
  $\mathbf{v}_{t+1} \leftarrow \mathbf{x}_{t+1} - \mathbf{x}_t$;
  if $t_{perturb} \neq 0$ *and* $t - t_{perturb} < \mathscr{T}$ **then**
    $\mathbf{x}_{t+1} = \mathbf{x}_{t+1} + \eta \nabla f(\tilde{\mathbf{x}})$, $\mathbf{v}_{t+1} = \mathbf{v}_{t+1} + \eta \nabla f(\tilde{\mathbf{x}})$;
    $\mathbf{v}_{t+1} \leftarrow r \cdot \frac{\mathbf{v}_{t+1}}{\|\mathbf{x}_{t+1} - \tilde{\mathbf{x}}\|}$, $\mathbf{x}_{t+1} \leftarrow \tilde{\mathbf{x}} + r \cdot \frac{\mathbf{x}_{t+1} - \tilde{\mathbf{x}}}{\|\mathbf{x}_{t+1} - \tilde{\mathbf{x}}\|}$;
  else
    if $f(\mathbf{x}_t) \leq f(\mathbf{z}_t) + \langle \nabla f(\mathbf{z}_t), \mathbf{x}_t - \mathbf{z}_t \rangle - \frac{\gamma}{2}\|\mathbf{x}_t - \mathbf{z}_t\|^2$ **then**
      $(\mathbf{x}_{t+1}, \mathbf{v}_{t+1}) \leftarrow$ NegativeCurvatureExploitation($\mathbf{x}_t, \mathbf{v}_t, s$);

Single-looped

Simplicity and numerical stability are preserved

Final result

Theorem 7 (informal). *For any $\epsilon > 0$ and any constant $0 < \delta \leq 1$, Algorithm 2 satisfies that at least one of the iterations $\mathbf{x}_t$ will be an ϵ-approximate second-order stationary point in*

$$\tilde{O}\Big(\frac{(f(\mathbf{x}_0) - f^*)}{\epsilon^{1.75}} \cdot \log n\Big)$$

iterations, with probability at least $1 - \delta$, where f^* is the global minimum of f.

- Matches the iteration number of Allen-Zhu et al. 2017 using a simpler, single-looped algorithm with numerical stability.

- In addition, we essentially show the robustness of this algorithm, which may be of independent interest.

Extension to stochastic settings

- A stochastic version of the our simple, numerically-stable negative curvature finding subroutine:

Algorithm 4: Stochastic Negative Curvature Finding$(\mathbf{x}_0, r_s, \mathscr{T}_s, m)$.

$\mathbf{y}_0 \leftarrow 0,\ L_0 \leftarrow r_s$;
for $t = 1, ..., \mathscr{T}_s$ **do**
  Sample $\left\{\theta^{(1)}, \theta^{(2)}, \cdots, \theta^{(m)}\right\} \sim \mathcal{D}$;
  $\mathbf{g}(\mathbf{y}_{t-1}) \leftarrow \frac{1}{m}\sum_{j=1}^{m}\left(\mathbf{g}(\mathbf{x}_0 + \mathbf{y}_{t-1}; \theta^{(j)}) - \mathbf{g}(\mathbf{x}_0; \theta^{(j)})\right)$;
  $\mathbf{y}_t \leftarrow \mathbf{y}_{t-1} - \frac{1}{\ell}(\mathbf{g}(\mathbf{y}_{t-1}) + \xi_t / L_{t-1}), \qquad \xi_t \sim \mathcal{N}\left(0, \frac{r_s^2}{d} I\right)$;
  $L_t \leftarrow \frac{\|\mathbf{y}_t\|}{r_s} L_{t-1}$ and $\mathbf{y}_t \leftarrow \mathbf{y}_t \cdot \frac{r_s}{\|\mathbf{y}_t\|}$;
Output $\mathbf{y}_{\mathscr{T}} / r_s$.

Extension to stochastic settings

- Quantitative result:

Theorem 9 (informal). *For any* $\epsilon > 0$ *and any constant* $0 < \delta \leq 1$, *our algorithm using only stochastic gradient descent satisfies that at least one of the iterations* $\mathbf{x}_t$ *will be an* ϵ*-approximate second-order stationary point in*

$$\tilde{O}\Big(\frac{(f(\mathbf{x}_0) - f^*)}{\epsilon^4} \cdot \log^2 n\Big)$$

iterations, with probability at least $1 - \delta$, *where* f^* *is the global minimum of* f.

Numerical experiments

Comparison between our algorithm (ANCGD) and Jin et al. (PAGD)

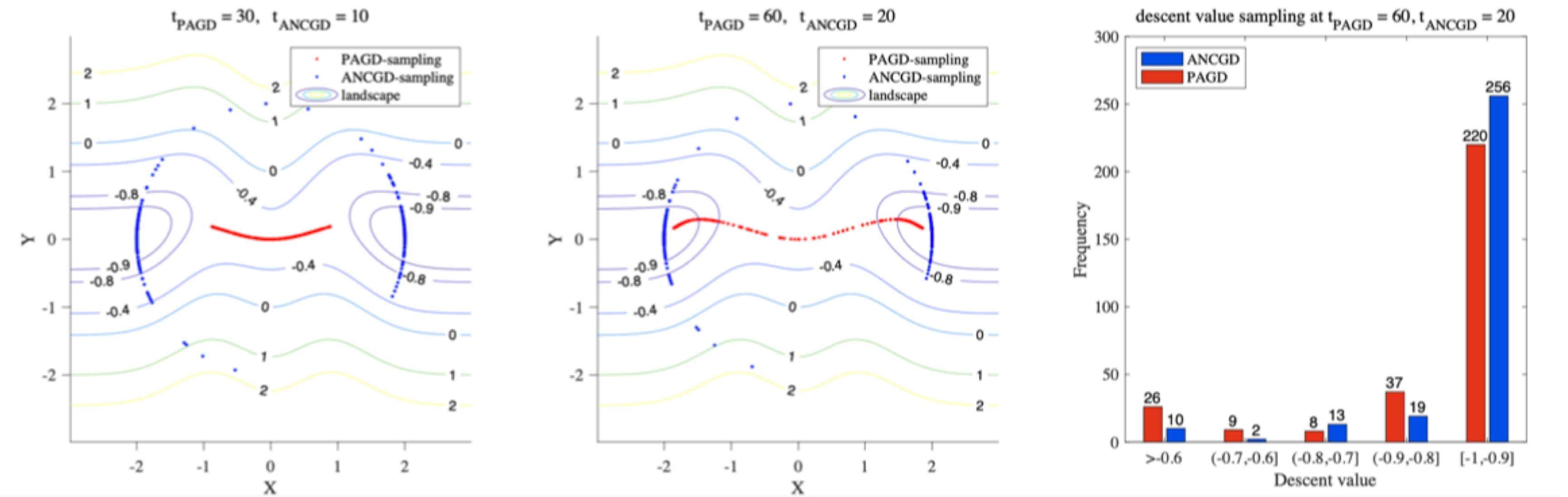


Figure 6: Run ANCGD and PAGD on landscape $f(x_1, x_2) = f(x_1, x_2) = \frac{1}{1+e^{x_1^2}} + \frac{1}{2}\left(x_2 - x_1^2 e^{-x_1^2}\right)^2 - 1$. Parameters: $\eta = 0.03$ (step length), $r = 0.1$ (ball radius in PAGD and parameter r in ANCGD), $M = 300$ (number of samplings).

Left: The contour of the landscape is placed on the background with labels being function values. Blue points represent samplings of ANCGD at time step $t_{\text{ANCGD}} = 10$ and $t_{\text{ANCGD}} = 20$, and red points represent samplings of PAGD at time step $t_{\text{PAGD}} = 30$ and $t_{\text{PAGD}} = 60$. ANCGD converges faster than PAGD even when $t_{\text{ANCGD}} \ll t_{\text{PAGD}}$.

Right: A histogram of descent values obtained by ANCGD and PAGD, respectively. Set $t_{\text{ANCGD}} = 20$ and $t_{\text{PAGD}} = 60$. Although we run three times of iterations in PAGD, its performance is still dominated by our ANCGD.

Conclusions

Main result: A single-looped, simple algorithm for an ϵ-approx. local minimum x_ϵ using $\tilde{O}(\log n/\epsilon^{1.75})$ iterations.

Open questions:

- Can we achieve the polynomial speedup in $\log n$ for more advanced stochastic optimization algorithms with complexity $\tilde{O}(\text{poly}(\log n)/\epsilon^{3.5})$ (Allen-Zhu et al. 2018) or $\tilde{O}(\text{poly}(\log n)/\epsilon^{3})$ (Fang et al. 2018)?
- How is the performance of our algorithms for escaping saddle points in real-world applications, such as tensor decomposition, matrix completion, etc.?