Adaptive Optimization Methods for Machine Learning and Signal Processing

Volkan Cevher
volkan.cevher@epfl.ch

Ali Kavis
ali.kavis@epfl.ch

Kfir Y. Levy
kfirylevy@technion.ac.il

Ahmet Alacaoglu
ahmet.alacaoglu@epfl.ch

Part I/IV: An introduction

Laboratory for Information and Inference Systems (LIONS)
École Polytechnique Fédérale de Lausanne (EPFL)
Further acknowledgements

- LIONS group members (current & alumni): [https://lions.epfl.ch](https://lions.epfl.ch)
  - EE-556 (Mathematics of Data): Course material
- Optimization for Machine Learning (OPT-ML) 2020 Workshop at NeurIPS: [https://opt-ml.org/](https://opt-ml.org/)
One formula to rule all ML & SP problems

\[ f^* = \min_{x:x \in \mathcal{X}} f(x) \ (\text{argmin} \to x^*) \]

- Growing interest in first-order gradient methods\(^1\) due to their scalability and generalization performance

---

One formula to rule all ML & SP problems ... and one algorithm to solve them.

\[ f^* = \min_{x: x \in \mathcal{X}} f(x) \quad \text{(argmin} \rightarrow x^*) \]

- Growing interest in first-order gradient methods\(^1\) due to their scalability and generalization performance
- In the sequel, the set \( \mathcal{X} \) is convex:
  \[- \forall x, y \in \mathcal{X}, \forall \alpha \in [0, 1], \quad \alpha x + (1 - \alpha)y \in \mathcal{X}. \]
- In the sequel, the function \( f \) may be convex:
  \[- f(\alpha x + (1 - \alpha)y) \leq \alpha f(x) + (1 - \alpha)f(y), \quad \forall x, y \in \mathcal{X}, \quad \forall \alpha \in [0, 1]. \]

One formula to rule all ML & SP problems ...and one algorithm to solve them.

\[ f^* = \min_{x: x \in \mathcal{X}} f(x) \quad (\text{argmin} \rightarrow x^*) \]

- Growing interest in first-order gradient methods\(^1\) due to their scalability and generalization performance
- In the sequel, the set \( \mathcal{X} \) is convex:
  - \( \forall x, y \in \mathcal{X}, \forall \alpha \in [0, 1], \alpha x + (1 - \alpha)y \in \mathcal{X}. \)
- In the sequel, the function \( f \) may not be convex:
  - \( f(\alpha x + (1 - \alpha)y) \leq \alpha f(x) + (1 - \alpha)f(y), \quad \forall x, y \in \mathcal{X}, \forall \alpha \in [0, 1]. \)

---
Application: Deep learning via empirical risk minimization

Definition (Optimization formulation)

The deep-learning training problem is given by

\[ x^*_{DL} \in \arg \min_{x \in \mathcal{X}} \left\{ f(x) := \frac{1}{n} \sum_{i=1}^{n} L(h_x(a_i), b_i) \right\}, \]

where \( \mathcal{X} \) denotes the constraints on the parameters.

\( \circ \) A single hidden layer neural network with params \( x := [X_1, X_2, \mu_1, \mu_2] \)

\[ h_x(a) := \left[ \begin{array}{c} X_2 \\ X_1 \end{array} \right] \sigma \left( \left[ \begin{array}{c} a \\ \mu_1 \end{array} \right] + \left[ \begin{array}{c} \mu_2 \end{array} \right] \right) \]

activation \( \downarrow \)

weight \( \downarrow \)

input \( \downarrow \)

bias \( \downarrow \)

bias \( \downarrow \)

hidden layer = learned features
**Loss function examples**

**Definition (Hinge loss)**

For a binary classification problem, the hinge loss for a score value $b_1 \in \mathbb{R}$ and class label $b_2 \in \pm 1$ is given by $L(b_1, b_2) = \max(0, 1 - b_1 \times b_2)$.

**Definition ($\ell_q$-losses)**

For all $b_1, b_2 \in \mathbb{R}^n \times \mathbb{R}^n$, we can use $L_q(b_1, b_2) = \|b_1 - b_2\|_q^q$, where

$$\ell_q\text{-norm: } \|b\|_q^q := \sum_{i=1}^n |b_i|^q \text{ for } b \in \mathbb{R}^n \text{ and } q \in [1, \infty)$$

**Definition (Wasserstein distance)**

Let $\mu$ and $\nu$ be two probability measures on $\mathbb{R}^d$ and define their couplings as $\Gamma(\mu, \nu) := \{\pi \text{ probability measure on } \mathbb{R}^d \times \mathbb{R}^d \text{ with marginals } \mu, \nu\}$.

$$W(\mu, \nu) := \left(\inf_{\pi \in \Gamma(\mu, \nu)} \mathbb{E}_{(x,y) \sim \pi} \|x - y\|^2\right)^{1/2}$$
A basic *iterative* strategy

\[ f^* = \min_{x: x \in \mathcal{X}} f(x) \ (\text{argmin} \rightarrow x^*) \]

**General idea of an optimization algorithm**

*Guess* a solution, and then *refine* it based on *oracle information*.

*Repeat* the procedure until the result is *good enough*. 
Basic principles of descent methods

Template for iterative descent methods

1. Let \( x_0 \in \mathcal{X} \) be a starting point.
2. Generate a sequence of vectors \( x_1, x_2, \cdots \in \mathcal{X} \) so that we have descent:

\[
f(x_{t+1}) < f(x_t), \quad \text{for all } t = 0, 1, \ldots
\]

until \( x_t \) satisfies \( f(x_t) - f^* \leq \epsilon \).

Such a sequence \( \{x_t\}_{t \geq 0} \) can be generated as:

\[
x_{t+1} = x_t + \alpha_t p_t
\]

where \( p_t \) is a descent direction and \( \alpha_t > 0 \) a step-size.

Remarks:
- Iterative algorithms can use various oracle information in the optimization problem
- The type of oracle information used becomes a defining characteristic of the algorithm
- Example oracles: Objective value, gradient, and Hessian result in 0-th, 1-st, 2-nd order methods
- The oracle choices determine \( \alpha_k \) and \( p_t \) as well as the overall convergence rate and complexity

First-order methods use subdifferentials & gradients

The subdifferential of $f$ at $x$, denoted $\partial f(x)$, is the set of all vectors $v$ satisfying

$$f(y) \geq f(x) + \langle v, y - x \rangle + o(\|y - x\|) \quad \text{as} \quad y \to x.$$ 

If the function $f$ is differentiable, then its subdifferential contains only the gradient.
Basic principles of descent methods ($\mathcal{X} = \mathbb{R}^p$)

○ Recall the representation of the algorithmic iterates:

$$x_{t+1} = x_t + \alpha_t p_t.$$  

○ For a differentiable $f$, apply Taylor’s theorem with $\alpha_t = o(1)$

$$f(x_{t+1}) = f(x_t) + \alpha_t \langle \nabla f(x_t), p_t \rangle + \mathcal{O}(\alpha_t^2 \|p_t\|_2^2).$$

○ To obtain $f(x_{t+1}) < f(x_t)$, we need $\langle \nabla f(x_t), p_t \rangle < 0!$

Observations:

○ The local *steepest descent* direction is the negative gradient $p_t := -\nabla f(x_t)$

  ▶ $\langle \nabla f(x_t), p_t \rangle = \|\nabla f(x_t)\| \|p_t\| \cos \theta$

  ▶ $\theta$ is the angle between $\nabla f(x_t)$ and $p_t$

○ We can use a subgradient $p_t \in -\partial f(x_t)$ as a descent direction

![Figure: Descent directions in 2D should be an element of the cone of descent directions $\mathcal{D}(f, \cdot)$.](image-url)
Brief detour: Gradients of vector valued functions

### Jacobian

When \( f : \mathbb{R}^n \to \mathbb{R}^d \) is a vector valued function, the following \( d \times n \) matrix \( J \) of partial derivatives

\[
\begin{bmatrix}
  \frac{\partial f_1}{\partial x_1}(x) \\
  \frac{\partial f_2}{\partial x_1}(x) \\
  \vdots \\
  \frac{\partial f_n}{\partial x_1}(x)
\end{bmatrix}
\]

is called the Jacobian of \( f \) at \( x \).

**Observations:**
- The Jacobian is the transpose of the gradient, when \( f \) is real valued.
- Thinking in terms of Jacobians is really helpful when we need to use the chain rule.

### Chain Rule via Jacobians

Let \( \circ \) denote the functional composition: \( g \circ f := g(f(x)) \). If \( g \circ f \) is differentiable at \( x \), then the following holds

\[
J_{g \circ f}(x) = J_g(f(x))J_f(x).
\]

Hence, the chain rule, which is helpful in differentiating function compositions, can be related to a simple product of Jacobian matrices.

---

1 We overload the notation \( x_i \) to denote \( i^{th} \) coordinate when it is clear from the context. When we have \( x_t \), we use \( x_{t,i} \).
An example

Example

The gradient of \( f : x \mapsto w_2^\top \sigma(W_1x + \mu) \) is given by the following expression:

\[
\nabla f(x) = J_f(x)^\top = W_1^\top (\sigma'(W_1x + \mu) \odot w_2),
\]

where \( \sigma \) is a non-linear function that applies to each coordinate, and \( \odot \) denotes the component wise product.

Proof: \( f \) is a composition of the functions \( k \circ g \circ h \)

- \( h(x) = W_1x + \mu \), whose Jacobian is \( J_h(x) = W_1 \).
- \( g(x) = \begin{bmatrix} \sigma(x_1) \\ \vdots \\ \sigma(x_n) \end{bmatrix} \), whose Jacobian is \( J_g(x) = \text{diag}(\sigma'(x_1), \ldots, \sigma'(x_n)) \).
- \( k(x) = w_2^\top x \) whose Jacobian is \( J_k(x) = w_2^\top \).
- By the chain rule, we have that

\[
J_f(x) = J_k(g(h(x))) \cdot J_g(h(x)) \cdot J_h(x)
\]

\[
= w_2^\top \cdot \text{diag}(\sigma'([W_1x + \mu]_1), \ldots, \sigma'([W_1x + \mu]_n)) \cdot W_1
\]

Simply transpose the Jacobian to get the gradient and use \( \odot \) to replace the diagonal matrix.
A simple iterative algorithm: Gradient descent

Choose initial point: $x_0$. 

$x^* \leftarrow $ best direction
$x_{t+1} = x_t - \alpha \nabla f(x_t)$

Repeat until $x_t$ is accurate enough.
A simple iterative algorithm: Gradient descent

- Choose initial point: \( x_0 \).
- Take a step in the negative gradient direction with a step size \( \alpha > 0 \): 
  \[
  x_{t+1} = x_t - \alpha \nabla f(x_t).
  \]
A simple iterative algorithm: Gradient descent

- Choose initial point: $x_0$.
- Take a step in the negative gradient direction with a step size $\alpha > 0$: $x_{t+1} = x_t - \alpha \nabla f(x_t)$.
- Repeat this procedure until $x_t$ is accurate enough.
Challenges for an iterative optimization algorithm

**Problem**

*Find the minimum $x^*$ of $f(x)$, given starting point $x_0$ based on only local information.*

- *Fog of war*
Challenges for an iterative optimization algorithm

Problem

*Find the minimum $x^*$ of $f(x)$, given starting point $x_0$ based on only local information.*

- Fog of war, non-differentiability, discontinuities, local minima, stationary points...
A notion of convergence: Stationarity

- Let $f : \mathbb{R}^p \rightarrow \mathbb{R}$ be twice-differentiable and $x^* = \min_{x \in \mathbb{R}^p} f(x)$

**Gradient method**

Choose a starting point $x_0$ and iterate

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

where $\alpha > 0$ is a step-size to be chosen so that $x_t$ converges to $x^*$.

**Definition (First order stationary point (FOSP))**

A point $\bar{x}$ is a first order stationary point of a twice differentiable function $f$ if

$$\nabla f(\bar{x}) = 0.$$ 

**Fixed-point characterization**

Multiply by -1 and add $\bar{x}$ to both sides to obtain the fixed point condition:

$$\bar{x} = \bar{x} - \alpha \nabla f(\bar{x}) \quad \text{for all } \alpha \in \mathbb{R}.$$
Geometric interpretation of stationarity

Observation: Neither \( \bar{x} \), nor \( \tilde{x} \) is necessarily equal to \( x^* \) !!

Proposition (*Local minima, maxima, and saddle points)

Let \( \bar{x} \) be a stationary point of a twice differentiable function \( f \).

- If \( \nabla^2 f(\bar{x}) \succ 0 \), then the point \( \bar{x} \) is called a local minimum or a second order stationary point (SOSP).
- If \( \nabla^2 f(\bar{x}) \prec 0 \), then the point \( \bar{x} \) is called a local maximum.
- If \( \nabla^2 f(\bar{x}) = 0 \), then the point \( \bar{x} \) can be a saddle point, a local minimum, or a local maximum.
Local minima

\[
\min_{x \in \mathbb{R}} \{x^4 - 3x^3 + x^2 + \frac{3}{2}x\}
\]
\[
\frac{df}{dx} = 4x^3 - 9x^2 + 2x + \frac{3}{2}
\]

Choose \(x_0 = 0\) and \(\alpha = \frac{1}{6}\)
\[
x_1 = x_0 - \alpha \frac{df}{dx} \bigg|_{x=x_0} = 0 - \frac{1}{6} \cdot \frac{3}{2} = -\frac{1}{4}
\]
\[
x_2 = -\frac{5}{16}
\]

\(x_t\) converges to a local minimum!
From local to global optimality

Definition (Local minimum)
Given $f : \mathbb{R}^p \to \mathbb{R} \cup \{+\infty\}$, a vector $x^* \in \mathbb{R}^p$ is called a local minimum of $f$ if there exists $\epsilon > 0$ s.t.

$$f(x^*) \leq f(x) \quad \forall x \in \mathbb{R}^p \quad \text{with} \quad \|x - x^*\| \leq \epsilon.$$  

Theorem
If $Q \subset \mathbb{R}^p$ is a convex set and $f : \mathbb{R}^p \to (-\infty, +\infty]$ is a proper convex function, then a local minimum of $f$ over $Q$ is also a global minimum of $f$ over $Q$.

Proof.
Suppose $x^*$ is a local minimum but not global, i.e. there exist $x \in \mathbb{R}^p$ s.t. $f(x) < f(x^*)$. By convexity,

$$f(\alpha x^* + (1 - \alpha)x) \leq \alpha f(x^*) + (1 - \alpha)f(x) < f(x^*), \forall \alpha \in [0, 1]$$

which contradicts the local minimality of $x^*$.

Theorem
Let $f : \mathbb{R}^p \to \mathbb{R}$ be a convex differentiable function. Then any stationary point of $f$ is a global minimum.
Effect of very small step-size $\alpha$...

$$\min_{x \in \mathbb{R}} \frac{1}{2} (x - 3)^2$$

$$\frac{df}{dx} = x - 3$$

Choose $x_0 = 5$ and $\alpha = \frac{1}{10}$

$$x_1 = x_0 - \alpha \frac{df}{dx} \bigg|_{x=x_0} = 5 - \frac{1}{10} \cdot 2 = 4.8$$

$$x_2 = x_1 - \alpha \frac{df}{dx} \bigg|_{x=x_1} = 4.8 - \frac{1}{10} \cdot 1.8 = 4.62$$

$x_t$ converges very slowly.
Effect of very large step-size $\alpha$...

\[
\min_{x \in \mathbb{R}} \frac{1}{2} (x - 3)^2
\]

\[
\frac{df}{dx} = x - 3
\]

Choose $x_0 = 5$ and $\alpha = \frac{5}{2}$

\[
x_1 = x_0 - \alpha \frac{df}{dx} \bigg|_{x=x_0} = 5 - \frac{5}{2} \cdot 2 = 0
\]

\[
x_2 = x_1 - \alpha \frac{df}{dx} \bigg|_{x=x_1} = 0 - \frac{5}{2} (-3) = \frac{15}{2}
\]

$x_t$ diverges.
A geometrical intuition: How does smoothness help?
A geometrical intuition: How does smoothness help?

Structure in optimization:

\[(1) \quad f(x) \geq f(x^k) + \langle \nabla f(x^k), x - x^k \rangle \]
A geometrical intuition: How does smoothness help?

Majorize:

\[ f(x) \leq f(x^k) + \langle \nabla f(x^k), x - x^k \rangle + \frac{L}{2} \| x - x^k \|_2^2 := Q_L(x, x^k) \]

Minimize:

\[ x^{k+1} = \arg \min_x Q_L(x, x^k) \]
\[ = \arg \min_x \| x - \left( x^k - \frac{1}{L} \nabla f(x^k) \right) \|_2^2 \]
\[ = x^k - \frac{1}{L} \nabla f(x^k) \]

Structure in optimization:

(1) \[ f(x) \geq f(x^k) + \langle \nabla f(x^k), x - x^k \rangle \]
(2) \[ f(x) \leq f(x^k) + \langle \nabla f(x^k), x - x^k \rangle + \frac{L}{2} \| x - x^k \|_2^2 \]
A geometrical intuition: How does smoothness help?

Majorize:

\[ f(x) \leq f(x^k) + \langle \nabla f(x^k), x - x^k \rangle + \frac{L'}{2} \| x - x^k \|_2^2 := Q_{L'}(x, x^k) \]

Minimize:

\[ x^{k+1} = \arg\min_x Q_{L'}(x, x^k) \]

\[ = \arg\min_x \left\| x - \left( x^k - \frac{1}{L'} \nabla f(x^k) \right) \right\|^2 \]

\[ = x^k - \frac{1}{L'} \nabla f(x^k) \]

Structure in optimization:

(1) \[ f(x) \geq f(x^k) + \langle \nabla f(x^k), x - x^k \rangle \]
(2) \[ f(x) \leq f(x^k) + \langle \nabla f(x^k), x - x^k \rangle + \frac{L}{2} \| x - x^k \|_2^2 \]
Stationarity measures with constraints & non-smoothness

- **Smooth**: Gradient mapping norm
  - $\|G_{\alpha}(x_t)\|^2 = \frac{1}{\alpha^2} \|x_t - P_{\mathcal{X}}(x_t - \alpha \nabla f(x_t))\|^2$
  - $P_{\mathcal{X}}$ denotes the projection operator to $\mathcal{X}$
  - possible to compute

- **Non-smooth**: Generalized subdifferential distance
  - $\text{dist}(0, \partial(f(x_t) + \delta_{\mathcal{X}}(x_t)))^2$
  - $\delta_{\mathcal{X}}$ refers to the indicator function for the set $\mathcal{X}$
  - hard in general (even approximately)
The one formula is very flexible

\[ \Phi^* = \min_{x \in \mathcal{X}} \max_{y \in \mathcal{Y}} \Phi(x, y) \ (\text{argmin argmax} \rightarrow x^*, y^*) \]
The one formula is very flexible

\[ \Phi^* = \min_{x: x \in \mathcal{X}} \max_{y: y \in \mathcal{Y}} \Phi(x, y) \quad (\text{argmin } \text{argmax } \rightarrow x^*, y^*) \]

\[ f^* = \min_{x: x \in \mathcal{X}} f(x) \quad (\text{argmin } \rightarrow x^*) \]
Application: Adversarial training

Figure: (Left) An $\ell_\infty$-attack: The alteration is hard to perceive. (Right) An $\ell_1$-attack: The alteration in this case is obvious.

Adversarial Training

Let $h_x : \mathbb{R}^n \to \mathbb{R}$ be a model with parameters $x$ and let $\{(a_i, b_i)\}_{i=1}^n$, with the data $a_i \in \mathbb{R}^p$ and the labels $b_i$. The problem of adversarial training is the following adversarial optimization problem

\[
\min_x \frac{1}{n} \sum_{i=1}^n \max_{\eta : \|\eta\| \leq \epsilon} L(h_x(a_i + \eta), b_i) \approx \min_x \mathbb{E}_{(a, b) \sim P} \left[ \max_{\eta : \|\eta\| \leq \epsilon} L(h_x(a_i + \eta), b_i) \right].
\]

Note the similarity with the template $\min_{x \in \mathcal{X}} \max_{y \in \mathcal{Y}} \Phi(x, y)$. 

Danskin’s theorem

Danskin’s theorem (Bertsekas variant)

Let \( \Phi(x, y) : \mathbb{R}^p \times Y \to \mathbb{R} \), where \( Y \subset \mathbb{R}^m \) is a compact set and define \( f(x) := \max_{y \in Y} \Phi(x, y) \). Let \( \Phi(x, y) \) is an extended real-valued closed proper convex function for each \( y \) in the compact set \( Y \); the interior of the domain of \( f \) is nonempty; \( \Phi(x, y) \) is jointly continuous on the relative interior of the domain of \( f \) and \( Y \).

Define \( Y^* := \arg \max_{y \in Y} \Phi(x, y) \) as the set of maximizers and \( y^* \in Y^* \) as an element of this set. We have

1. \( f(x) \) is a convex function.
2. If \( y^* = \arg \max_{y \in Y} \Phi(x, y) \) is unique, then the function \( f(x) = \max_{y \in Y} \Phi(x, y) \) is differentiable at \( x \):

\[
\nabla_x f(x) = \nabla_x \left( \max_{y \in Y} \phi(x, y) \right) = \nabla_x \Phi(x, y^*).
\]

3. If \( y^* = \arg \max_{y \in Y} \Phi(x, y) \) is not unique, then the subdifferential \( \partial_x f(x) \) of \( f \) is given by

\[
\partial_x f(x) = \text{conv} \{ \partial_x \Phi(x, y^*) : y^* \in Y^* \}.
\]

Remarks:

- The adversarial problem is not convex in \( x \) in general.
- With proper initialization, overparameterization works argue that it is effectively convex.
- (Sub)Gradients of \( f_i \) are calculated as \( \partial f_i(x) = \nabla_x L(h_x(a_i + \eta^*(x)), b_i) \).
A corollary to Danskin’s theorem

**Adversarial Training**

Let \( h_x : \mathbb{R}^n \to \mathbb{R} \) be a model with parameters \( x \) and let \( \{(a_i, b_i)\}_{i=1}^n \), with \( a_i \in \mathbb{R}^p \) and \( b_i \) be the corresponding labels. The adversarial training optimization problem is given by

\[
\min_x \left\{ \frac{1}{n} \sum_{i=1}^n f_i(x) := \frac{1}{n} \sum_{i=1}^n \max_{\eta: \|\eta\|_\infty \leq \epsilon} L(h_x(a_i + \eta), b_i) \right\}.
\]

\( L \) is not continuously differentiable due to ReLU, max-pooling, etc.

**Descent directions [4]**

Define \( \mathcal{Y}^* := \arg \max_{y \in \mathcal{Y}} \Phi(x, y) \) as the set of maximizers, \( y^* \in \mathcal{Y}^* \), and \( f(x) := \max_{y \in \mathcal{Y}} \Phi(x, y) \). As long as \( \nabla_x \Phi(x, y^*) \) is non-zero, it is a descent direction (and not a subgradient!) for \( f(x) \).

**Remarks:**
- \( \nabla_x L(h_x(a_i + \eta^*(x)), b_i) \) is a descent direction for \( f_i(x) \).
- We cannot find global maximizers \( \mathcal{Y}^* \).
- Only when \( y^* \) is a singleton, \( \nabla_x L(h_x(a_i + \eta^*(x)), b_i) \) is a (sub)gradient [1].
A more general minimax problem: Generative adversarial networks

**Vanilla GAN [2]**

\[
\min_{x \in \mathcal{X}} \max_{y \in \mathcal{Y}} \mathbb{E}_{a \sim \hat{\mu}_n} [\log d_y(a)] + \mathbb{E}_{\omega \sim p_{\Omega}} [\log (1 - d_y(h_x(\omega)))]
\]  

- Binary cross-entropy modeling.
- \(d_y(a) : \mathcal{Y} \rightarrow [0, 1]\) represents the probability that \(a\) came from the real data distribution \(\mu^\natural\).

**Figure:** Schematic of a generative model, \(h_x(\omega) [2, 3]\).
Worst-case iteration complexities of classical projected first-order methods

<table>
<thead>
<tr>
<th>$f(x)$</th>
<th>gradient oracle</th>
<th>$L$-smooth</th>
<th>Stationarity measure</th>
<th>GD/SGD</th>
<th>Accelerated GD/SGD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convex</td>
<td>stochastic</td>
<td>yes</td>
<td>$f(x_t) - f^* = \mathcal{O}\left(\frac{1}{\sqrt{t}}\right)$</td>
<td>(\mathcal{O}\left(\frac{1}{\sqrt{t}}\right))</td>
<td>(\mathcal{O}\left(\frac{1}{\sqrt{t}}\right))</td>
</tr>
<tr>
<td>Convex</td>
<td>deterministic</td>
<td>yes</td>
<td>$f(x_t) - f^* = \mathcal{O}\left(\frac{1}{t}\right)$</td>
<td>(\mathcal{O}\left(\frac{1}{t^2}\right))</td>
<td>(\mathcal{O}\left(\frac{1}{t^2}\right))</td>
</tr>
<tr>
<td>Convex</td>
<td>stochastic</td>
<td>no</td>
<td>$f(x_t) - f^* = \mathcal{O}\left(\frac{1}{\sqrt{t}}\right)$</td>
<td>(\mathcal{O}\left(\frac{1}{\sqrt{t}}\right))</td>
<td>(\mathcal{O}\left(\frac{1}{\sqrt{t}}\right))</td>
</tr>
<tr>
<td>Nonconvex</td>
<td>stochastic</td>
<td>yes</td>
<td>$|G_\eta(x_t)|^2 = \mathcal{O}\left(\frac{1}{\sqrt{t}}\right)^3$</td>
<td>(\mathcal{O}\left(\frac{1}{\sqrt{t}}\right)^3)</td>
<td>(\mathcal{O}\left(\frac{1}{\sqrt{t}}\right)^3)</td>
</tr>
<tr>
<td>Nonconvex</td>
<td>deterministic</td>
<td>yes</td>
<td>$|G_\eta(x_t)|^2 = \mathcal{O}\left(\frac{1}{t}\right)^4$</td>
<td>(\mathcal{O}\left(\frac{1}{t}\right)^4)</td>
<td>(\mathcal{O}\left(\frac{1}{t}\right)^4)</td>
</tr>
<tr>
<td>Nonconvex</td>
<td>stochastic</td>
<td>no</td>
<td>$\text{dist}(0, \partial(f(x_t) + \delta\chi(x_t)))^2 = \gamma^{356}$</td>
<td>(\gamma^{356})</td>
<td>(\gamma^{356})</td>
</tr>
</tbody>
</table>

- Basic structures, such as smoothness or strong convexity, help, but there are more structures that can be used:
  - max-form, metric subregularity, Polyak-Lojasiewicz, Kurdyka-Lojasiewicz, weak convexity, growth cond...

---

### Worst-case iteration complexities of classical projected first-order methods

<table>
<thead>
<tr>
<th>$f(x)$</th>
<th>gradient oracle</th>
<th>$L$-smooth</th>
<th>Stationarity measure</th>
<th>GD/SGD</th>
<th>Accelerated GD/SGD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convex</td>
<td>stochastic</td>
<td>yes</td>
<td>$f(x_t) - f^* = \mathcal{O}\left(\frac{1}{\sqrt{t}}\right)$</td>
<td>$\mathcal{O}\left(\frac{1}{\sqrt{t}}\right)$</td>
<td></td>
</tr>
<tr>
<td>Convex</td>
<td>deterministic</td>
<td>yes</td>
<td>$f(x_t) - f^* = \mathcal{O}\left(\frac{1}{t}\right)$</td>
<td>$\mathcal{O}\left(\frac{1}{t^2}\right)$</td>
<td></td>
</tr>
<tr>
<td>Convex</td>
<td>stochastic</td>
<td>no</td>
<td>$f(x_t) - f^* = \mathcal{O}\left(\frac{1}{\sqrt{t}}\right)$</td>
<td>$\mathcal{O}\left(\frac{1}{\sqrt{t}}\right)$</td>
<td></td>
</tr>
<tr>
<td>Nonconvex</td>
<td>stochastic</td>
<td>yes</td>
<td>$|G_\eta(x_t)|^2 = \mathcal{O}\left(\frac{1}{\sqrt{t}}\right)^3$</td>
<td>$\mathcal{O}\left(\frac{1}{\sqrt{t}}\right)^3$</td>
<td></td>
</tr>
<tr>
<td>Nonconvex</td>
<td>deterministic</td>
<td>yes</td>
<td>$|G_\eta(x_t)|^2 = \mathcal{O}\left(\frac{1}{t}\right)^4$</td>
<td>$\mathcal{O}\left(\frac{1}{t}\right)^4$</td>
<td></td>
</tr>
<tr>
<td>Nonconvex</td>
<td>stochastic</td>
<td>no</td>
<td>$\text{dist}(0, \partial(f(x_t) + \delta \mathcal{X}(x_t))^2 = \gamma^{356}$</td>
<td>$\gamma^{356}$</td>
<td></td>
</tr>
</tbody>
</table>

- Basic structures, such as smoothness or strong convexity, help, but there are more structures that can be used:
  - max-form, metric subregularity, Polyak-Lojasiewicz, Kurdyka-Lojasiewicz, weak convexity,\(^3\) growth cond...

---

Worst-case is often too pessimistic

- GD: \( x_{t+1} = x_t - \frac{1}{L} \nabla f(x_t) \)

- Rates are not everything!
  - overall computational effort is what matters
  - constants & implementations are key

- Knowledge of smoothness, the value of \( L \),...
  - challenging

- Must “somehow” adapt to a “different” function
  - online and without knowing \( L \)
  - can reduce overall computational effort!
References


