One formula to rule all machine learning 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 machine learning problems ...and one algorithm to solve them.

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

- Growing interest in first-order gradient methods\(^1\) due to their scalability and generalization performance
- In the sequel,
  - the set \( \mathcal{X} \) is convex and has a tractable projection operator \( P_{\mathcal{X}} \)
  - all convergence characterizations are with feasible iterates \( x^k \in \mathcal{X} \)
  - gradient mapping means \( G_\eta(x^k) = \frac{1}{\eta} [x^k - P_{\mathcal{X}}(x^k - \eta \nabla f(x^k))], \) where \( \eta \) is the step-size
  - \( L \)-smooth means \( \| \nabla f(x) - \nabla f(y) \| \leq L \| x - y \|, \forall x, y \in \mathcal{X} \)
  - \( \partial \) may refer to the generalized subdifferential, and \( \delta_{\mathcal{X}} \) refers to the indicator function for the set \( \mathcal{X} \)

---

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>$f(x^k) - f^* = O\left(\frac{1}{\sqrt{k}}\right)$</td>
<td>$O\left(\frac{1}{\sqrt{k}}\right)$</td>
<td></td>
</tr>
<tr>
<td>Convex</td>
<td>deterministic</td>
<td>yes</td>
<td>$f(x^k) - f^* = O\left(\frac{1}{k}\right)$</td>
<td>$O\left(\frac{1}{k^2}\right)$</td>
<td></td>
</tr>
<tr>
<td>Convex</td>
<td>stochastic</td>
<td>no</td>
<td>$f(x^k) - f^* = O\left(\frac{1}{\sqrt{k}}\right)$</td>
<td>$O\left(\frac{1}{\sqrt{k}}\right)$</td>
<td></td>
</tr>
<tr>
<td>Nonconvex</td>
<td>stochastic</td>
<td>yes</td>
<td>$|G_\eta(x^k)|^2 = O\left(\frac{1}{\sqrt{k}}\right)^3$</td>
<td>$O\left(\frac{1}{\sqrt{k}}\right)^3$</td>
<td></td>
</tr>
<tr>
<td>Nonconvex</td>
<td>deterministic</td>
<td>yes</td>
<td>$|G_\eta(x^k)|^2 = O\left(\frac{1}{k}\right)^4$</td>
<td>$O\left(\frac{1}{k}\right)^4$</td>
<td></td>
</tr>
<tr>
<td>Nonconvex</td>
<td>stochastic</td>
<td>no</td>
<td>dist$(0, \partial f(x^k) + \delta\chi'(x^k))^2 = \theta^356$</td>
<td>$\theta^356$</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, 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>$f(x^k) - f^* = \mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$</td>
<td>$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$</td>
<td>$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$</td>
</tr>
<tr>
<td>Convex</td>
<td>deterministic</td>
<td>yes</td>
<td>$f(x^k) - f^* = \mathcal{O}\left(\frac{1}{k}\right)$</td>
<td>$\mathcal{O}\left(\frac{1}{k^2}\right)$</td>
<td>$\mathcal{O}\left(\frac{1}{k^2}\right)$</td>
</tr>
<tr>
<td>Convex</td>
<td>stochastic</td>
<td>no</td>
<td>$f(x^k) - f^* = \mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$</td>
<td>$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$</td>
<td>$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$</td>
</tr>
<tr>
<td>Nonconvex</td>
<td>stochastic</td>
<td>yes</td>
<td>$|G_\eta(x^k)|^2 = \mathcal{O}\left(\frac{1}{\sqrt{k}}\right)^3$</td>
<td>$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)^3$</td>
<td>$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)^3$</td>
</tr>
<tr>
<td>Nonconvex</td>
<td>deterministic</td>
<td>yes</td>
<td>$|G_\eta(x^k)|^2 = \mathcal{O}\left(\frac{1}{k}\right)^4$</td>
<td>$\mathcal{O}\left(\frac{1}{k}\right)^4$</td>
<td>$\mathcal{O}\left(\frac{1}{k}\right)^4$</td>
</tr>
<tr>
<td>Nonconvex</td>
<td>stochastic</td>
<td>no</td>
<td>dist$(0, \partial (f(x^k) + \delta \chi(x^k))) = \gamma^{356}$</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 is often too pessimistic

- GD: $x^{k+1} = x^k - \frac{1}{L} \nabla f(x^k)$

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!
Warmup: $f$ is convex

$$f^* = \min_{x: x \in \mathcal{X}} f(x) \ (\text{argmin} \rightarrow x^*)$$
A classical approach: Line-search

- Long history: Backtracking, Armijo, steepest descent...
- Universal accelerated gradient method\(^1\)

\[
f(x^k) - f^* = O \left( \frac{L\nu \|x^0 - x^*\|^{1+\nu}}{k^{1+3\nu}} \right)
\]

- adapts to Hölder smoothness \((\nu \in [0, 1])\)
- has extensions to primal-dual optimization\(^2\)
- sets accuracy a priori & monotonic step-sizes
- Not as universal as we wish it to be
  - different procedures for stochastic gradients\(^3\)

---

A contemporary approach: Online convex optimization (OCO)

Algorithm: A basic online learning problem

1: for $t = 1, \ldots, k$ do
2: Player chooses some action $x^t \in X \subset \mathbb{R}^p$
3: Environment reveals a convex loss $f_t(\cdot)$
4: Player suffers the loss $f_t(x^t)$
5: end for

- Minimize the total loss vs the best action in hindsight:
  $$R(k) = \sum_{t=1}^{k} f_t(x^t) - \min_{x \in X} \sum_{t=1}^{k} f_t(x).$$

- "somehow" adapts to a "different" function!

- For general convex $f_t$, optimal regret is sublinear:
  $$R(k) = O\left(\sqrt{k}\right).$$

- We can trivially convert regret to rate via $f_t = f$:
  $$f\left(\frac{1}{k} \sum_{t=1}^{k} x^t\right) - f^* \leq \frac{R(k)}{k}.$$
A contemporary approach: Online convex optimization (OCO)

**Algorithm:** A basic online learning problem

1. for \( t = 1, \ldots, k \) do
2. \hspace{1em} Player chooses some action \( x^t \in \mathcal{X} \subset \mathbb{R}^p \)
3. \hspace{1em} Environment reveals a convex loss \( f_t(\cdot) \)
4. \hspace{1em} Player suffers the loss \( f_t(x^t) \)
5. end for

- Minimize the total loss vs the best action in hindsight:

\[
R(k) = \sum_{t=1}^{k} f_t(x^t) - \min_{x \in \mathcal{X}} \sum_{t=1}^{k} f_t(x).
\]

- “somehow” adapts to a “different” function!

- For general convex \( f_t \), optimal regret is sublinear:

\[
R(k) = \mathcal{O} \left( \sqrt{k} \right).
\]

- We can trivially convert regret to rate via \( f_t = f \):

\[
f \left( \frac{1}{k} \sum_{t=1}^{k} x^t \right) - f^* \leq \frac{R(k)}{k}.
\]

- One procedure to rule them all...
  - smooth, non-smooth, stochastic!
- Not as adaptive as we like in optimization
  - The “offline” fast rate \( 1/k^2 \) is not immediate

---

The curious case of AdaGrad

Algorithm: AdaGrad (scalar)

1: Input: Iterations $k$; $x_0 \in \mathcal{X}$
2: for $t = 0, \ldots, k - 1$ do
3: Obtain a gradient estimate $g_t$
4: $\eta_t = D / \left( 2 \sum_{i=1}^{t} \|g_t\|^2 \right)^{1/2}$
5: $x_{t+1} = P_{\mathcal{X}} \left( x_t - \eta_t g_t \right)$
6: end for
7: Output: $\bar{x}_k = \frac{1}{k} \sum_{t=1}^{k} x^t$

- AdaGrad does not need to know smoothness
  1. $g_t \in \partial f(x^t)$
  2. $g_t = \nabla f(x^t)$
  3. $\mathbb{E}g_t = \nabla f(x^t)$ & $\mathbb{E}[\|g - \nabla f(x)\|^2 | x] \leq \sigma^2$
- AdaGrad adapts and achieves optimal regret
  \[ R(k) \leq \sqrt{2D^2 \sum_{t=1}^{k} \|g_t\|^2}, \]
  where $D = \sup_{x,y \in \mathcal{X}} \|x - y\|_2$.
- When $f$ is $L$-smooth, AdaGrad output satisfies
  \[ \mathbb{E} \left[ f(\bar{x}_k) \right] - f^* = \mathcal{O} \left( \frac{LD^2}{k} + \frac{\sigma D}{\sqrt{k}} \right). \]

---

The curious case of AdaGrad

Algorithm: AdaGrad (scalar)

1: **Input:** Iterations $k$; $x_0 \in \mathcal{X}$
2: **for** $t = 0, ..., k - 1$ **do**
3: Obtain a gradient estimate $g_t$
4: $\eta_t = D/(2 \sum_{i=1}^{t} \|g_t\|^2)^{1/2}$
5: $x^{t+1} = P_X(x^t - \eta_t g_t)$
6: **end for**
7: **Output:** $\bar{x}_k = \frac{1}{k} \sum_{t=1}^{k} x^t$

**Is it an adaptive optimization method?**

**Is it a universal optimization method?**

- AdaGrad does not need to know smoothness
  1. $g_t \in \partial f(x^t)$
  2. $g_t = \nabla f(x^t)$
  3. $E[g_t = \nabla f(x^t) & E[\|g - \nabla f(x)\|^2|x] \leq \sigma^2$

- AdaGrad adapts and achieves optimal regret

$$R(k) \leq \sqrt{2D^2 \sum_{t=1}^{k} \|g_t\|_2^2},$$

where $D = \sup_{x, y \in \mathcal{X}} \|x - y\|_2$.

- When $f$ is $L$-smooth, AdaGrad output satisfies

$$E[f(\bar{x}_k)] - f^* = O\left(\frac{LD^2}{k} + \frac{\sigma D}{\sqrt{k}}\right).$$

---

Enter AcceleGrad:¹ Exploiting the linear coupling idea²

Algorithm: AcceleGrad for unconstrained optimization

1: **Input:** Iterations \( k \); \( y_0, z_0 \in \mathbb{R}^p \)
2: **for** \( t = 0, \ldots, k - 1 \) **do**
3: Obtain a gradient estimate \( g_t \)
4: \( \alpha_t = \max \left( 1, \frac{t+1}{4} \right) \)
5: \( \eta_t = \frac{\sqrt{G^2 + \sum_{i=0}^{t} \alpha_i^2 \|g_i\|^2}}{2D} \)
6: \( x_{t+1} = \frac{1}{\alpha_t} y_t + \left( 1 - \frac{1}{\alpha_t} \right) z_t \)
7: \( z_{t+1} = P_X (z_t - \alpha_t \eta_t g_t) \)
8: \( y_{t+1} = x_{t+1} - \eta_t g_t \)
9: **end for**
10: **Output:** \( \bar{y}_k \propto \alpha \sum_{t=1}^{k} \alpha_t - 1 y_t \)

- AcceleGrad does not need to know smoothness
  1. \( g_t \in \partial f(x^t) \)
  2. \( g_t = \nabla f(x^t) \)
  3. \( \mathbb{E} g_t = \nabla f(x^t) \) & \( \|g\| \leq G \)

- AcceleGrad output satisfies:¹ \( \mathbb{E} f(\bar{y}_k) - f^* = \)
  1. \( \mathcal{O} \left( \frac{GD \sqrt{\log(k)}}{\sqrt{k}} \right) \)
  2. \( \mathcal{O} \left( \frac{DG + LD^2 \log(LD/G)}{k^2} \right) \)
  3. \( \mathcal{O} \left( \frac{GD \sqrt{\log(k)}}{\sqrt{k}} \right) \)

- Caveats:
  - needs a bound \( G \) on the subgradient norms
  - needs a bound \( D \) on \( X \) where the solution lives
  - cannot handle constraints!

Enter AcceleGrad:¹ Exploiting the linear coupling idea²

**Algorithm:** AcceleGrad for unconstrained optimization

1. **Input:** Iterations $k$; $y_0, z_0 \in \mathbb{R}^p$
2. for $t = 0, \ldots, k - 1$ do
3. Obtain a gradient estimate $g_t$
4. $\alpha_t = \max \left( 1, \frac{t+1}{4} \right)$
5. $\eta_t = \frac{1}{\sqrt{\alpha_t} + \sum_{i=0}^{t} \alpha_i^2 \|g_i\|^2}$
6. $x_{t+1} = \frac{1}{\alpha_t} y_t + (1 - \frac{1}{\alpha_t}) z_t$
7. $z_{t+1} = P_{\mathcal{X}}(z_t - \alpha_t \eta_t g_t)$
8. $y_{t+1} = x_{t+1} - \eta_t g_t$
9. end for
10. **Output:** $\bar{y}_k \propto \alpha \sum_{t=1}^{k} \alpha_t - 1 y_t$

- AcceleGrad does not need to know smoothness
  1. $g_t \in \partial f(x^t)$
  2. $g_t = \nabla f(x^t)$
  3. $\mathbb{E}g_t = \nabla f(x^t) \& \|g\| \leq G$

- AcceleGrad output satisfies:¹ $\mathbb{E}f(\bar{y}_k) - f^* =$
  1. $\mathcal{O} \left( \frac{GD \sqrt{\log(k)}}{\sqrt{k}} \right)$
  2. $\mathcal{O} \left( \frac{D G^2 \log(LD / \|g_0\|)}{k^2} \right)$
  3. $\mathcal{O} \left( \frac{GD \sqrt{\log k}}{\sqrt{k}} \right)$

- Caveats:
  - needs a bound $G$ on the subgradient norms
  - needs a bound $D$ on $\mathcal{X}$ where the solution lives
  - cannot handle constraints!

---

Enter AcceleGrad:¹ Exploiting the linear coupling idea²

Algorithm: AcceleGrad for unconstrained optimization

1: **Input:** Iterations \( k \); \( y_0, z_0 \in \mathbb{R}^p \)
2: **for** \( t = 0, \ldots, k - 1 \) **do**
3: Obtain a gradient estimate \( g_t \)
4: \( \alpha_t = \max \left( 1, \frac{t+1}{4} \right) \)
5: \( \eta_t = \frac{\sqrt{D + \sum_{i=0}^{t} \alpha_i^2 \|g_i\|^2}}{2D} \)
6: \( x_{t+1} = \frac{1}{\alpha_t} y_t + (1 - \frac{1}{\alpha_t}) z_t \)
7: \( z_{t+1} = P_{\mathcal{X}}(z_t - \alpha_t \eta_t g_t) \)
8: \( y_{t+1} = x_{t+1} - \eta_t g_t \)
9: **end for**
10: **Output:** \( \tilde{y}_k \propto_{\alpha} \sum_{t=1}^{k} \alpha t - 1 y_t \)

- Is it an adaptive optimization method?
- Is it a universal optimization method?

- AcceleGrad does not need to know smoothness
  1. \( g_t \in \partial f(x^t) \)
  2. \( g_t = \nabla f(x^t) \)
  3. \( \mathbb{E} g_t = \nabla f(x^t) \& \|g\| \leq G \)

- AcceleGrad output satisfies:¹ \( \mathbb{E} f(\tilde{y}_k) - f^* = \)
  1. \( \mathcal{O} \left( \frac{GD \sqrt{\log(k)}}{\sqrt{k}} \right) \)
  2. \( \mathcal{O} \left( \frac{DG^2 + LD^2 \log(LD/\|g_0\|)}{k^2} \right) \)
  3. \( \mathcal{O} \left( \frac{GD \sqrt{\log(k)}}{\sqrt{k}} \right) \)

- Caveats:
  ▶ needs a bound \( G \) on the subgradient norms
  ▶ needs a bound \( D \) on \( \mathcal{X} \) where the solution lives
  ▶ cannot handle constraints!

UniXGrad: Universal eXtra Gradient method

**Algorithm: UniXGrad**

**Input:** Iterations $k$; $y_0 \in \mathcal{X}$; $\alpha_t = t$

1. for $t = 0, \ldots, k - 1$ do
2. \[ \tilde{y}_t \propto \alpha_t y_{t-1} + \sum_{i=1}^{t-1} \alpha_i x_i \]
3. Obtain a gradient estimate $g_t^{(1)} = g_t(\tilde{y}_t)$
4. \[ \eta_t = 2D \sqrt{1 + \sum_{i=1}^{t-1} \alpha_i^2 \left\| g_i^{(1)} - g_i^{(2)} \right\|_*^2} \]
5. \[ x^t = P_X \left( y_{t-1} - \alpha_t \eta_t g_t^{(1)} \right) \]
6. \[ \bar{x}_t \propto \alpha_t x^t + \sum_{i=1}^{t-1} \alpha_i x_i \rightarrow \text{output} \]
7. Obtain a gradient estimate $g_t^{(2)} = g_t(\bar{x}_t)$
8. \[ y_t = P_X \left( y_{t-1} - \alpha_t \eta_t g_t^{(2)} \right) \]
9. end for

---

2. A. Nemirovski, “Prox-method with rate of convergence ... smooth convex-concave saddle point problems,” SIOPT, 2005.
UniXGrad: Universal eXtra Gradient method

**Algorithm:** UniXGrad

**Input:** Iterations $k$; $y_0 \in \mathcal{X}$; $\alpha_t = t$

1. for $t = 0, \ldots, k - 1$ do
   2. $\tilde{y}_t \propto \alpha_t y_{t-1} + \sum_{i=1}^{t-1} \alpha_i x_i$
   3. Obtain a gradient estimate $g^{(1)}_t = g_t(\tilde{y}_t)$
   4. $\eta_t = 2D \sqrt{\frac{1}{t} + \sum_{i=1}^{t-1} \alpha_i^2 \left\| g^{(1)}_i - g^{(2)}_i \right\|_2^2}$
   5. $x^t = P_{\mathcal{X}} \left( y_{t-1} - \alpha_t \eta_t g^{(1)}_t \right)$
   6. $\bar{x}_t \propto \alpha_t x^t + \sum_{i=1}^{t-1} \alpha_i x_i$ → output
   7. Obtain a gradient estimate $g^{(2)}_t = g_t(\bar{x}_t)$
   8. $y_t = P_{\mathcal{X}} \left( y_{t-1} - \alpha_t \eta_t g^{(2)}_t \right)$
2. end for

- UniXGrad does not need to know smoothness
  1. $g_t(\cdot) \in \partial f(\cdot)$
  2. $g_t(\cdot) = \nabla f(\cdot)$
  3. $\mathbb{E}g_t(\cdot) = \nabla f(\cdot) \& \mathbb{E}[\|g_t(x) - \nabla f(x)\|^2|x] \leq \sigma^2$

- UniXGrad output satisfies:
  1. $\mathbb{E}f(\bar{x}_k) - f^* =$
     1. $\frac{6D}{k^2} + \frac{14GD}{\sqrt{k}}$
     2. $\frac{20\sqrt{7}D^2L}{k^2}$
     3. $\frac{224\sqrt{14}D^2L}{k^2} + \frac{14\sqrt{2}\sigma D}{\sqrt{k}}$

- First universal and adaptive algorithm
  - optimal rates in the “offline” setting
  - builds on mirror-prox$^2$ & optimistic MD$^3$
  - new online-to-offline conversion lemma$^4$
$f$ is nonconvex

\[ f^* = \min_{x : x \in \mathcal{X}} f(x) \text{ (argmin } \to x^*) \]
Detour: Weak convexity (WeCo) & approximate stationarity

- **Smooth:** Gradient mapping norm
  \[ \|G_{\eta}(x^k)\|^2 = \frac{1}{\eta^2} \|x^k - P_X(x^k - \eta \nabla f(x^k))\|^2 \]
  - possible to compute

- **Non-smooth:** Generalized subdifferential distance
  \[ \text{dist}(0, \partial(f(x^k) + \delta_X(x^k)))^2 \]
  - hard in general (even approximately)

- \( f \) is \( \rho \)-weakly convex if \( f(x) + \frac{\rho}{2} \|x\|^2 \) is convex.

- Moreau envelope (ME):
  \[ \varphi_{1/\rho}(x) = \min_{y \in X} \left\{ f(y) + \frac{\rho}{2} \|y - x\|^2 \right\} \]
  \[ \hat{x} \leftarrow \arg \min \nabla \varphi_{1/\rho}(x) = \rho(x - \hat{x}) \]

  - Small \( \|\nabla \varphi_{1/\rho}(x)\| \) implies near-stationarity:
  \[ \text{dist}(0, \partial(f(x^k) + \delta_X(x^k)))^2 \leq \|\nabla \varphi_{1/\rho}(x^k)\|^2 \]
  - also implies small \( \|G_{\eta}(x^k)\|^2 \) if \( f \) is smooth

---

The King of all optimization algorithms: Adam\(^1\) (60K+ citations)

**Algorithm:** (variable metric) Adam

1. **Input:** Iterations \(k\); \(x_0 \in \mathcal{X}, \beta_{1,2} \in [0, 1]\)
2. for \(t = 0, \ldots, k - 1\) do
3. Obtain a gradient estimate \(g_t\)
4. \(m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t\)
5. \(\hat{v}_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2\)
6. \(x^{t+1} = P^{{\hat{v}_t}^{1/2}}_{\mathcal{X}} \left( x^t - \alpha_t \hat{v}_t^{-1/2} m_t \right)\)
7. end for
8. **Output:** \(x^*_{t}(k); t^*(k)\) is randomly chosen in \(\{1, \ldots, k\}\).

- The King does not need to know smoothness
  1. \(g_t \in \partial f(x^t)\)
  2. \(g_t = \nabla f(x^t)\)
  3. \(\mathbb{E}g_t = \nabla f(x^t) \& \mathbb{E}[\|g - \nabla f(x^t)\|^2] \leq \sigma^2\)

- The King adapts and achieves optimal regret\(^3\)
  \[R(k) = \mathcal{O}\left(\sqrt{k}\right),\]
  with constant \(\beta_1\) in OCO.

- The King’s output satisfies for WeCo\(^4\)
  \[\mathbb{E}\|\nabla \phi^t_{1/\rho}(x^*_{t}(k))\|^2 = \mathcal{O}\left(\frac{1}{\sqrt{k}}\right).\]

---


\(^3\)A. Alacaoglu, Y. Malitsky, P. Mertikopoulos, and V. Cevher, "A new regret analysis for adam-type algorithms," ICML 2020

The King of all optimization algorithms: Adam\(^1\) (60K+ citations)

Algorithm: (variable metric) Adam-type

1. **Input:** Iterations \(k\); \(x_0 \in \mathcal{X}, \beta_{1,2} \in [0,1]\)

2. **for** \(t = 0, \ldots, k - 1\) **do**

3. Obtain a gradient estimate \(g_t\)

4. \(m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t\)

5. \(\hat{v}_t = \phi(g_t)\)

6. \(x_{t+1} = P_{\mathcal{X}} \left( x_t - \alpha_t \hat{v}_t^{-1/2} m_t \right)\)

7. **end for**

8. **Output:** \(x_{t^*(k)}\): \(t^*(k)\) is randomly chosen in \(\{1, \ldots, k\}\).

- The King is naked:\(^2\) AMSGrad
  - \(\phi(g_t) = \max(\hat{v}_{t-1}, v_t)\), and \(v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2\)
  - F. Orabona: parameterfree.com (Dec 6)

- The King does not need to know smoothness
  1. \(g_t \in \partial f(x^t)\)
  2. \(g_t = \nabla f(x^t)\)
  3. \(\mathbb{E} g_t = \nabla f(x^t) \& \mathbb{E}[\|g - \nabla f(x)\|^2 | x] \leq \sigma^2\)

- The King adapts and achieves optimal regret:\(^3\)
  \[ R(k) = \mathcal{O} \left( \sqrt{k} \right), \]
  with constant \(\beta_1\) in OCO.

- The King’s output satisfies for WeCo:\(^4\)
  \[ \mathbb{E}\|\nabla \phi_1^t (x_{t^*(k)})\|^2 = \mathcal{O} \left( \frac{1}{\sqrt{k}} \right). \]

---


\(^3\) A. Alacaoglu, Y. Malitsky, P. Mertikopoulos, and V. Cevher, “A new regret analysis for adam-type algorithms,” ICML 2020

## A comparison of algorithms

<table>
<thead>
<tr>
<th></th>
<th>GD/SGD</th>
<th>Accelerated GD/SGD</th>
<th>AdaGrad</th>
<th>AcceleGrad/UniXgrad</th>
<th>Adam/AMSGrad</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Convex, stochastic</strong></td>
<td>$O\left(\frac{1}{\sqrt{k}}\right)$ $^1$</td>
<td>$O\left(\frac{1}{\sqrt{k}}\right)$ $^1$</td>
<td>$O\left(\frac{1}{k}\right)$ $^2$</td>
<td>$O\left(\frac{1}{\sqrt{k}}\right)$ $^{3,4}$</td>
<td>$O\left(\frac{1}{k}\right)$ $^5$</td>
</tr>
<tr>
<td><strong>Convex, deterministic, $L$-smooth</strong></td>
<td>$O\left(\frac{1}{k}\right)$ $^1$</td>
<td>$O\left(\frac{1}{k^2}\right)$ $^1$</td>
<td>$O\left(\frac{1}{k}\right)$ $^3$</td>
<td>$O\left(\frac{1}{k^2}\right)$ $^{3,4}$</td>
<td>$O\left(\frac{1}{k}\right)$ $^6$</td>
</tr>
<tr>
<td><strong>Nonconvex, stochastic, $L$-smooth</strong></td>
<td>$O\left(\frac{1}{\sqrt{k}}\right)$ $^1$</td>
<td>$O\left(\frac{1}{\sqrt{k}}\right)$ $^1$</td>
<td>$O\left(\frac{1}{k}\right)$ $^7$</td>
<td>?</td>
<td>$O\left(\frac{1}{\sqrt{k}}\right)$ $^8$</td>
</tr>
<tr>
<td><strong>Nonconvex, deterministic, $L$-smooth</strong></td>
<td>$O\left(\frac{1}{k}\right)$ $^1$</td>
<td>$O\left(\frac{1}{k}\right)$ $^1$</td>
<td>$O\left(\frac{1}{k}\right)$ $^7$</td>
<td>?</td>
<td>$O\left(\frac{1}{k}\right)$ $^6$</td>
</tr>
</tbody>
</table>

---

Conclusions

◦ Simple algorithms automatically adapt to strong convexity under broad assumptions
  ▶ GD achieves linear rate with $\eta = 1/L^1$
  ▶ SGD achieves $O(1/k)$-rate with $\eta_k = O(1/k)^2$
  ▶ PDHG achieves linear rate under metric subregularity

◦ Adaptive methods are promising but are not yet truly universal...
  ▶ Accelegrad/UniXgrad does not adapt to strong convexity
  ▶ AdaGrad needs a different step-size policy
  ▶ Adam-type does not adapt to strong convexity
  ▶ MetaGrad comes close but is not universal yet

◦ Still seeking one algorithm to rule them all!

---

Acknowledgements

- Faculty: Kfir Levy, Francis Bach, Yura Malitsky, Panayotis Mertikopoulos.

- PhD:
  - Ahmet Alacaoglu
    ahmet.alacaoglu@epfl.ch
  - Ali Kavis
    ali.kavis@epfl.ch
  - Alp Yurtsever
    alpy@mit.edu

- Postdoc positions available at LIONS. Email: volkan.cevher@epfl.ch
Logistic regression

- Data: a4a
- Oracle: Deterministic

Figure: Logistic regression on a4a
Neural network training: ADAM vs. AcceleGrad

Figure: Resnet classifier optimization (train loss)

Figure: Resnet classifier optimization (test loss)