Hunting the Hessian: Online Methods

Madeleine Udell

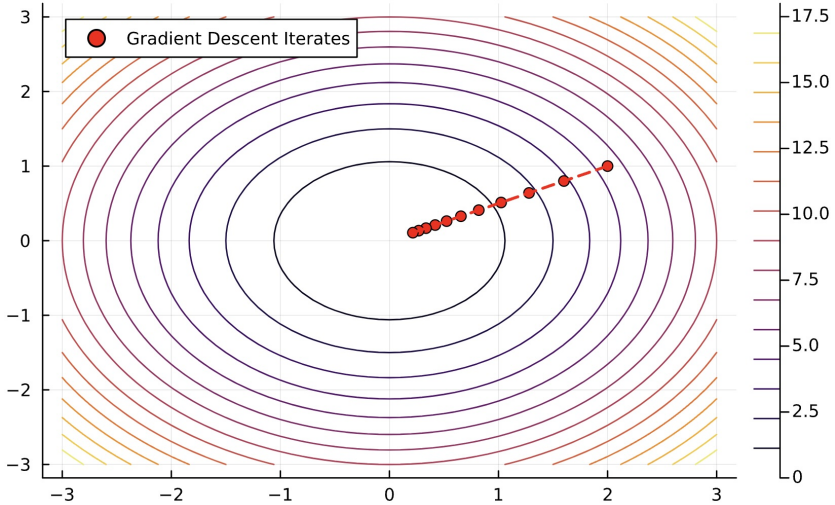
Management Science and Engineering
Stanford University

Joint work with
Wenzhi Gao (Stanford), Ya-Chi Chu (Stanford), Wanyu Zhang (Stanford),
Yinyu Ye (Stanford, Cardinal Optimization)

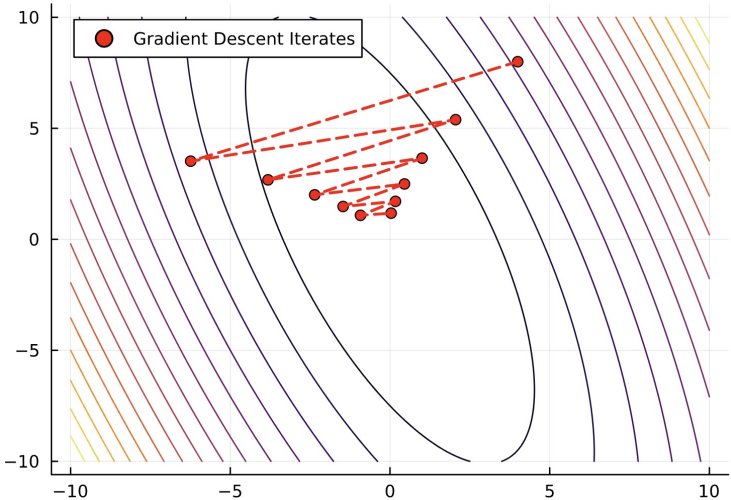
January 13, 2026

Outline

Gradient methods converge quickly on well-conditioned data



Gradient methods converge slowly on ill-conditioned data



Recap: convergence analysis for gradient descent

$$\text{minimize} \quad f(x)$$

recall: we say (twice-differentiable) f is μ-strongly convex and L-smooth if

$$\mu I \preceq \nabla^2 f(x) \preceq LI$$

recall: if f is μ-strongly convex and L-smooth, gradient descent converges linearly

$$f(x^{K+1}) - p^\star \leq \frac{Lc^K}{2}\|x^1 - x^\star\|^2,$$

where $c = (\frac{\kappa-1}{\kappa+1})^2$, $\kappa = \frac{L}{\mu} \geq 1$ is condition number

Recap: convergence analysis for gradient descent

$$\text{minimize} \quad f(x)$$

recall: we say (twice-differentiable) f is μ-strongly convex and L-smooth if

$$\mu I \preceq \nabla^2 f(x) \preceq LI$$

recall: if f is μ-strongly convex and L-smooth, gradient descent converges linearly

$$f(x^{K+1}) - p^\star \leq \tfrac{Lc^K}{2}\|x^1 - x^\star\|^2,$$

where $c = (\frac{\kappa-1}{\kappa+1})^2$, $\kappa = \frac{L}{\mu} \geq 1$ is condition number $\implies$ want $\kappa \approx 1$

Recap: convergence analysis for gradient descent

$$\text{minimize} \quad f(x)$$

recall: we say (twice-differentiable) f is μ-strongly convex and L-smooth if

$$\mu I \preceq \nabla^2 f(x) \preceq LI$$

recall: if f is μ-strongly convex and L-smooth, gradient descent converges linearly

$$f(x^{K+1}) - p^\star \leq \tfrac{Lc^K}{2}\|x^1 - x^\star\|^2,$$

where $c = (\frac{\kappa-1}{\kappa+1})^2$, $\kappa = \frac{L}{\mu} \geq 1$ is condition number $\implies$ want $\kappa \approx 1$

idea: can we minimize another function with $\kappa \approx 1$ whose solution will tell us the minimizer of f?

Preconditioning

for invertible D, the two problems

$$\text{minimize} \quad f(x) \quad \text{and} \quad \text{minimize} \quad f(Dz)$$

have solutions related by $x^\star = Dz^\star$

Preconditioning

for invertible D, the two problems

$$\text{minimize} \quad f(x) \quad \text{and} \quad \text{minimize} \quad f(Dz)$$

have solutions related by $x^\star = Dz^\star$

- gradient of $f(Dz)$ is $D^T \nabla f(Dz)$
- the second derivative (Hessian) of $f(Dz)$ is $D^T \nabla^2 f(Dz) D$

Preconditioning

for invertible D, the two problems

$$\text{minimize} \quad f(x) \quad \text{and} \quad \text{minimize} \quad f(Dz)$$

have solutions related by $x^\star = Dz^\star$

- gradient of $f(Dz)$ is $D^T \nabla f(Dz)$
- the second derivative (Hessian) of $f(Dz)$ is $D^T \nabla^2 f(Dz) D$

a gradient step on $f(Dz)$ with step-size $t > 0$ is

$$\begin{aligned} z^+ &= z - tD^T \nabla f(Dz) \\ Dz^+ &= Dz - tDD^T \nabla f(Dz) \\ x^+ &= x - tDD^T \nabla f(x) \end{aligned}$$

this iteration is *preconditioned gradient descent* (PGD) with preconditioner $P = DD^T$.

Optimal preconditioner

Convergence rate for PGD on function f with preconditioner P can be controlled by the *preconditioned* condition number:

$$\kappa_P(f) := \sup_x \kappa(D^T \nabla^2 f(x) D) = \sup_x \frac{\lambda_{\max}(D^T \nabla^2 f(x) D)}{\lambda_{\min}(D^T \nabla^2 f(x) D)}$$

where $D = P^{1/2}$.

Optimal preconditioner

Convergence rate for PGD on function f with preconditioner P can be controlled by the *preconditioned* condition number:

$$\kappa_P(f) := \sup_x \kappa(D^T \nabla^2 f(x) D) = \sup_x \frac{\lambda_{\max}(D^T \nabla^2 f(x) D)}{\lambda_{\min}(D^T \nabla^2 f(x) D)}$$

where $D = P^{1/2}$.

proof: plug the function $f \circ P$ into the convergence analysis for gradient descent

Optimal preconditioner

Convergence rate for PGD on function f with preconditioner P can be controlled by the *preconditioned* condition number:

$$\kappa_P(f) := \sup_x \kappa(D^T \nabla^2 f(x) D) = \sup_x \frac{\lambda_{\max}(D^T \nabla^2 f(x) D)}{\lambda_{\min}(D^T \nabla^2 f(x) D)}$$

where $D = P^{1/2}$.

proof: plug the function $f \circ P$ into the convergence analysis for gradient descent

The *optimal preconditioner* in set $\mathcal{P}$ minimizes the worst-case condition number,

$$P^\star \quad = \quad \operatorname*{argmin}_{P \in \mathcal{P},\ P = DD^T} \ \sup_x \kappa(D^T \nabla^2 f(x) D),$$

and defines the optimal condition number $\kappa^\star = \kappa((P^\star)^\top \nabla^2 f(x)(P^\star))$.

Outline

Gradient methods with online preconditioning

For smooth, strongly convex optimization,

$$x^{k+1} = x^k - P\nabla f(x^k) \implies f(x^{k+1}) - f(x^\star) \leq (1 - \tfrac{1}{\kappa_P})[f(x^k) - f(x^\star)]$$

Gradient methods with online preconditioning

For smooth, strongly convex optimization,

$$x^{k+1} = x^k - P\nabla f(x^k) \implies f(x^{k+1}) - f(x^\star) \leq (1 - \tfrac{1}{\kappa_P})[f(x^k) - f(x^\star)]$$

Can we learn a preconditioner P with $\kappa_P \ll \kappa$ during gradient descent?

$$\text{Gradient descent} \quad x^{k+1} = x^k - P_k \nabla_x f(x^k)$$
$$\text{Preconditioner update} \quad P_{k+1} = \text{Learn}(P_k, x^k)$$

Gradient methods with online preconditioning

For smooth, strongly convex optimization,

$$x^{k+1} = x^k - P\nabla f(x^k) \implies f(x^{k+1}) - f(x^\star) \leq (1 - \tfrac{1}{\kappa_P})[f(x^k) - f(x^\star)]$$

Can we learn a preconditioner P with $\kappa_P \ll \kappa$ during gradient descent?

$$\begin{aligned} \text{Gradient descent} \quad & x^{k+1} = x^k - P_k \nabla_x f(x^k) \\ \text{Preconditioner update} \quad & P_{k+1} = P_k - \eta \nabla_P \ell_{x^k}(P_k) \end{aligned}$$

Yes! By gradient descent on a feedback $\ell_x(P)$

- invented 25 years ago [Almeida, Langlois, Amaral, and Plakhov (1999)] and re-discovered as hypergradient descent [Baydin, Cornish, Rubio, et al. (2018)]
- good performance after tuning, but often unstable and almost no theory

Optimize the convergence rate with online learning

Better $P \Rightarrow$ smaller condition number $\kappa_P \Rightarrow$ better contraction factor

$$f(x - P\nabla f(x)) - f(x^\star) \leq (1 - \tfrac{1}{\kappa_P})[f(x) - f(x^\star)]$$

Optimize the convergence rate with online learning

Better $P \Leftarrow$ smaller condition number $\kappa_P \Leftarrow$ better contraction factor

$$\ell_x(P) = r_x(P) := \frac{f(x - P\nabla f(x)) - f(x^\star)}{f(x) - f(x^\star)}$$

Optimize the convergence rate with online learning

Better $P \Leftarrow$ smaller condition number $\kappa_P \Leftarrow$ better contraction factor

$$\ell_x(P) = r_x(P) := \frac{f(x - P\nabla f(x)) - f(x^\star)}{f(x) - f(x^\star)}$$

$$\frac{f(x^{K+1}) - f(x^\star)}{f(x^1) - f(x^\star)} = \prod_{k=1}^K \frac{f(x^{k+1}) - f(x^\star)}{f(x^k) - f(x^\star)} = \prod_{k=1}^K r_{x^k}(P_k) \leq \left(\frac{1}{K}\sum_{k=1}^K r_{x^k}(P_k)\right)^K$$

Optimize the convergence rate with online learning

Better $P \Leftarrow$ smaller condition number $\kappa_P \Leftarrow$ better contraction factor

$$\ell_x(P) = r_x(P) := \frac{f(x - P\nabla f(x)) - f(x^\star)}{f(x) - f(x^\star)}$$

$$\frac{f(x^{K+1}) - f(x^\star)}{f(x^1) - f(x^\star)} = \prod_{k=1}^K \frac{f(x^{k+1}) - f(x^\star)}{f(x^k) - f(x^\star)} = \prod_{k=1}^K r_{x^k}(P_k) \leq \left(\frac{1}{K}\sum_{k=1}^K r_{x^k}(P_k)\right)^K$$

- $P_{k+1} = P_k - \eta \nabla_P r_{x^k}(P_k)$ is online gradient descent

Optimize the convergence rate with online learning

Better $P \Leftarrow$ smaller condition number $\kappa_P \Leftarrow$ better contraction factor

$$\ell_x(P) = r_x(P) := \frac{f(x - P\nabla f(x)) - f(x^\star)}{f(x) - f(x^\star)}$$

$$\frac{f(x^{K+1}) - f(x^\star)}{f(x^1) - f(x^\star)} = \prod_{k=1}^K \frac{f(x^{k+1}) - f(x^\star)}{f(x^k) - f(x^\star)} = \prod_{k=1}^K r_{x^k}(P_k) \leq \left(\frac{1}{K}\sum_{k=1}^K r_{x^k}(P_k)\right)^K$$

- $P_{k+1} = P_k - \eta \nabla_P r_{x^k}(P_k)$ is online gradient descent
- $\sum_{k=1}^K r_{x^k}(P_k) \leq \sum_{k=1}^K r_{x^k}(P^\star) + O(\sqrt{K})$ guaranteed by online learning

Optimize the convergence rate with online learning

Better $P \Leftarrow$ smaller condition number $\kappa_P \Leftarrow$ better contraction factor

$$\ell_x(P) = r_x(P) := \frac{f(x - P\nabla f(x)) - f(x^\star)}{f(x) - f(x^\star)}$$

$$\frac{f(x^{K+1}) - f(x^\star)}{f(x^1) - f(x^\star)} = \prod_{k=1}^K \frac{f(x^{k+1}) - f(x^\star)}{f(x^k) - f(x^\star)} = \prod_{k=1}^K r_{x^k}(P_k) \leq \left(\frac{1}{K}\sum_{k=1}^K r_{x^k}(P_k)\right)^K$$

- $P_{k+1} = P_k - \eta \nabla_P r_{x^k}(P_k)$ is online gradient descent
- $\sum_{k=1}^K r_{x^k}(P_k) \leq \sum_{k=1}^K r_{x^k}(P^\star) + O(\sqrt{K})$ guaranteed by online learning

Since $r_{x^k}(P^\star) \leq 1 - \frac{1}{\kappa^\star}$, combining these relations gives

$$\frac{f(x^{K+1}) - f(x^\star)}{f(x^1) - f(x^\star)} \leq \left(1 - \frac{1}{\kappa^\star} + O\left(\frac{1}{\sqrt{K}}\right)\right)^K \approx \left(1 - \frac{1}{\kappa^\star}\right)^K,$$

an online acceleration of gradient descent!

More than acceleration

$$\frac{f(x^{K+1}) - f(x^\star)}{f(x^1) - f(x^\star)} \leq \left(1 - \frac{1}{\kappa^\star} + O\left(\frac{1}{\sqrt{K}}\right)\right)^K$$

Important limits:

- Asymptotically, $\frac{1}{\sqrt{K}} \to 0 \implies$ OSGM converges as fast as the optimal $P^\star$.

More than acceleration

$$\frac{f(x^{K+1}) - f(x^\star)}{f(x^1) - f(x^\star)} \leq \left(1 - \frac{1}{\kappa^\star} + O\left(\frac{1}{\sqrt{K}}\right)\right)^K$$

Important limits:

- Asymptotically, $\frac{1}{\sqrt{K}} \to 0 \implies$ OSGM converges as fast as the optimal $P^\star$.
- If f is a quadratic, $\kappa^\star = 1$, so

$$\frac{f(x^{K+1}) - f(x^\star)}{f(x^1) - f(x^\star)} \leq \left(O\left(\frac{1}{\sqrt{K}}\right)\right)^K$$

and we get *superlinear* convergence!

More than acceleration

$$\frac{f(x^{K+1}) - f(x^\star)}{f(x^1) - f(x^\star)} \leq \left(1 - \frac{1}{\kappa^\star} + O\left(\frac{1}{\sqrt{K}}\right)\right)^K$$

Important limits:

- Asymptotically, $\frac{1}{\sqrt{K}} \to 0 \implies$ OSGM converges as fast as the optimal $P^\star$.
- If f is a quadratic, $\kappa^\star = 1$, so

$$\frac{f(x^{K+1}) - f(x^\star)}{f(x^1) - f(x^\star)} \leq \left(O\left(\frac{1}{\sqrt{K}}\right)\right)^K$$

  and we get *superlinear* convergence!

Online Scaled Gradient Method: a new acceleration scheme for first-order methods

Outline

OSGM framework

Two agents: stepsize scheduler, landscape

Procedure: at each iteration k, two agents interact as follows:

1. Scheduler makes decision P_k and suggests $x^{k+1/2} = x^k - P_k \nabla f(x^k)$.
2. Landscape steps to $x^{k+1} = \mathcal{M}(x^k, x^{k+1/2})$ and provides feedback $\ell_{x^k}(P_k)$ to $\mathcal{S}$.
3. Scheduler updates the stepsize using online learning $P_{k+1} = \mathcal{A}(P_k, \{\ell_j\}_{j \le k})$.

Algorithm:

$$\begin{aligned} \text{(PGD)} \qquad & x^{k+1/2} = x^k - P_k \nabla f(x^k) \\ \text{(Landscape action)} \qquad & x^{k+1} = \mathcal{M}(x^k, x^{k+1/2}) \\ \text{(Stepsize learning)} \qquad & P_{k+1} = \mathcal{A}(P_k, \{\ell_j\}_{j \le k}) \end{aligned}$$

More generally

OSGM converts a *measure of convergence* into an *improved algorithm* by adaptively learning hyperparameters of the algorithm, such as a stepsize or gradient scaling.

Choices:

- algorithm to update iterates x ▷ e.g., PGD
- algorithm to update stepsize P ▷ e.g., online GD
- domain for stepsize ▷ e.g., scalar, diagonal, positive definite, general
- convergence measure $\ell_{x^k}(P^\star)$ ▷ e.g., ratio feedback $r_x(P)$
- checks to ensure stability ▷ e.g., monotonicity, linesearch

Preview of results

With appropriate choices, OSGM guarantees

- **Competitive convergence:** OSGM converges asymptotically at least as fast as the best fixed stepsize $P^\star$, and always at least as fast as the initial stepsize P_1.
- **Local convergence:** OSGM converges superlinearly.
- **Trajectory-dependent convergence:** OSGM converges nearly as fast as any *sequence* of stepsizes, even one adapted to the iterate sequence $x^1, \ldots, x^K$, up to a term depending on the distance between subsequent stepsizes.
- **Good practical performance:** OSGM beats other adaptive first order methods, and even quasi-Newton methods like (L-)BFGS.

Feedback

Ratio feedback. Contraction factor of suboptimality.

$$r_x(P) := \frac{f(x - P\nabla f(x)) - f(x^\star)}{f(x) - f(x^\star)}, \quad \nabla r_x(P) := \frac{\nabla f(x - P\nabla f(x))\nabla f(x)^T}{f(x) - f(x^\star)}.$$

Feedback

Ratio feedback. Contraction factor of suboptimality.

$$r_x(P) := \frac{f(x - P\nabla f(x)) - f(x^\star)}{f(x) - f(x^\star)}, \quad \nabla r_x(P) := \frac{\nabla f(x - P\nabla f(x))\nabla f(x)^T}{f(x) - f(x^\star)}.$$

Hypergradient feedback. Function value progress relative to size of gradient.

$$h_x(P) := \frac{f(x - P\nabla f(x)) - f(x)}{\|\nabla f(x)\|^2}, \quad \nabla h_x(P) := \frac{\nabla f(x - P\nabla f(x))\nabla f(x)^T}{\|\nabla f(x)\|^2}.$$

Feedback

Ratio feedback. Contraction factor of suboptimality.

$$r_x(P) := \frac{f(x - P\nabla f(x)) - f(x^\star)}{f(x) - f(x^\star)}, \quad \nabla r_x(P) := \frac{\nabla f(x - P\nabla f(x))\nabla f(x)^T}{f(x) - f(x^\star)}.$$

Hypergradient feedback. Function value progress relative to size of gradient.

$$h_x(P) := \frac{f(x - P\nabla f(x)) - f(x)}{\|\nabla f(x)\|^2}, \quad \nabla h_x(P) := \frac{\nabla f(x - P\nabla f(x))\nabla f(x)^T}{\|\nabla f(x)\|^2}.$$

- Motivated by the descent lemma: $f\left(x - \frac{1}{L}\nabla f(x)\right) - f(x) \leq -\frac{1}{2L}\|\nabla f(x)\|^2$.

Feedback

Ratio feedback. Contraction factor of suboptimality.

$$r_x(P) := \frac{f(x - P\nabla f(x)) - f(x^\star)}{f(x) - f(x^\star)}, \quad \nabla r_x(P) := \frac{\nabla f(x - P\nabla f(x))\nabla f(x)^T}{f(x) - f(x^\star)}.$$

Hypergradient feedback. Function value progress relative to size of gradient.

$$h_x(P) := \frac{f(x - P\nabla f(x)) - f(x)}{\|\nabla f(x)\|^2}, \quad \nabla h_x(P) := \frac{\nabla f(x - P\nabla f(x))\nabla f(x)^T}{\|\nabla f(x)\|^2}.$$

- Motivated by the descent lemma: $f\left(x - \frac{1}{L}\nabla f(x)\right) - f(x) \leq -\frac{1}{2L}\|\nabla f(x)\|^2$.
- No need to know optimal value $f(x^\star)$

gradient scaling P in OSGM need not be positive definite or even symmetric!

Hypergradient Reduction

Hypergradient feedback controls convergence if the iterations are monotone.

Theorem Suppose $f(x)$ is convex and $h_k := \frac{f(x^{k+1})-f(x^k)}{\|\nabla f(x^k)\|^2}$ and the iterates $\{x^k\}$ are **non-increasing** in f: $f(x^{k+1}) \leq f(x^k) \leq \cdots$. Then, for any $K \geq 1$, the iterates $\{x^k\}$ and the stepsizes $\{P_k\}$ generated by `OSGM-H` satisfy

$$f(x^{K+1}) - f(x^\star) \leq \min\left\{\frac{\Delta^2}{\sum_{k=1}^K -h_k}, f(x^1) - f(x^\star)\right\},$$

where diameter $\Delta := \max_{x:\ f(x)\leq f(x^1)} \min_{x^\star \in \mathcal{X}^\star} \|x - x^\star\|$.

Hypergradient Reduction

Hypergradient feedback controls convergence if the iterations are monotone.

Theorem Suppose $f(x)$ is convex and $h_k := \frac{f(x^{k+1})-f(x^k)}{\|\nabla f(x^k)\|^2}$ and the iterates $\{x^k\}$ are **non-increasing** in f: $f(x^{k+1}) \leq f(x^k) \leq \cdots$. Then, for any $K \geq 1$, the iterates $\{x^k\}$ and the stepsizes $\{P_k\}$ generated by `OSGM-H` satisfy

$$f(x^{K+1}) - f(x^\star) \leq \min\left\{\frac{\Delta^2}{\sum_{k=1}^K -h_k}, f(x^1) - f(x^\star)\right\},$$

where diameter $\Delta := \max_{x:\ f(x)\leq f(x^1)} \min_{x^\star \in \mathcal{X}^\star} \|x - x^\star\|$.

More negative $\sum_{k=1}^K h_k \implies$ faster convergence!

Hypergradient Reduction

Hypergradient feedback controls convergence if the iterations are monotone.

Theorem Suppose $f(x)$ is convex and $h_k := \frac{f(x^{k+1})-f(x^k)}{\|\nabla f(x^k)\|^2}$ and the iterates $\{x^k\}$ are **non-increasing** in f: $f(x^{k+1}) \leq f(x^k) \leq \cdots$. Then, for any $K \geq 1$, the iterates $\{x^k\}$ and the stepsizes $\{P_k\}$ generated by `OSGM-H` satisfy

$$f(x^{K+1}) - f(x^\star) \leq \min\left\{\frac{\Delta^2}{\sum_{k=1}^K -h_k}, f(x^1) - f(x^\star)\right\},$$

where diameter $\Delta := \max_{x:\ f(x)\leq f(x^1)} \min_{x^\star \in \mathcal{X}^\star} \|x - x^\star\|$.

More negative $\sum_{k=1}^K h_k \implies$ faster convergence!

how to guarantee monotonicity?

Monotone landscape

Monotone landscape accepts proposal from scheduler if objective value decreases:

$$x^{k+1} = \arg\min\{f(x) \mid x \in \{x^{k+1/2}, x^k\}\}$$

Hypergradient OSGM (`OSGM-H`). Monotone landscape justifies

- Hypergradient reduction: $f(x^{K+1}) - f(x^\star) \leq \min\left\{\frac{\Delta^2}{\sum_{k=1}^K -h_k}, f(x^1) - f(x^\star)\right\}$
- $\sum_{k=1}^K h_k = \sum_{k=1}^K \frac{f(x^{k+1}) - f(x^k)}{\|\nabla f(x^k)\|^2} \leq \sum_{k=1}^K \frac{f(x^k - P_k \nabla f(x^k)) - f(x^k)}{\|\nabla f(x^k)\|^2} = \sum_{k=1}^K h_{x^k}(P_k)$

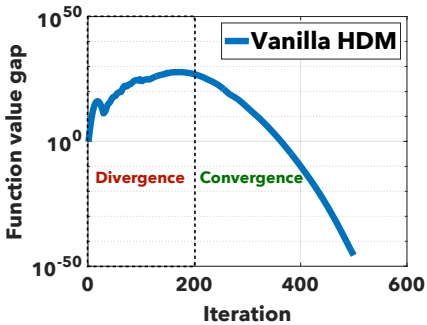
So, smaller feedback $h_{x^k}(P_k) \Rightarrow$ better progress $h_k \Rightarrow$ faster convergence!

Convergence of OSGM-H

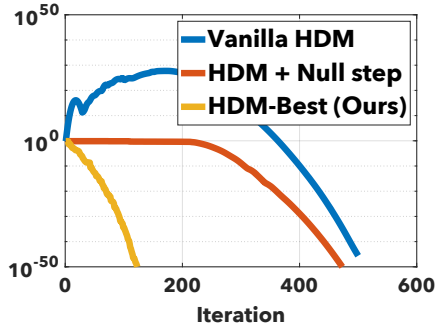
$$
\begin{aligned}
f(x^{K+1}) - f(x^\star) &\le \min\left\{ \frac{\Delta^2}{\sum_{k=1}^K -h_k}, f(x^1) - f(x^\star) \right\} && \text{(reduction)} \\
&\le \min\left\{ \frac{\Delta^2}{\sum_{k=1}^K -h_{x^k}(P_k)}, f(x^1) - f(x^\star) \right\} && \text{(monotone)} \\
&\le \min\left\{ \frac{\Delta^2}{K\left(\frac{1}{K}\sum_{k=1}^K -h_{x^k}(\hat{P}) - \mathcal{O}(\frac{1}{\sqrt{K}})\right)}, f(x^1) - f(x^\star) \right\} && \text{(regret guarantee)} \\
&\le \min\left\{ \frac{\Delta^2}{K\left(\frac{1}{2L} - \mathcal{O}(\frac{1}{\sqrt{K}})\right)}, f(x^1) - f(x^\star) \right\} && \left(\text{descent lemma } h_x\left(\tfrac{1}{L} I\right) \le -\tfrac{1}{2L}\right)
\end{aligned}
$$

- The hypergradient descent method (HDM) is OSGM-H (without monotonicity).
- Our work provides first convergence guarantee for (a simple variant of) HDM.

The importance of monotonicity



(a) Two-phase behavior



(b) Addressing instability

Figure: The behavior of different HDM variants on a toy quadratic optimization problem. Figure 1a: two-phase convergence behavior of vanilla HDM. Figure 1b: effect of monotone landscape and our best variant.

Outline

Regret guarantees vs any fixed stepsize

OSGM satisfies

$$\frac{1}{K}\sum_{k=1}^{K}\ell_{x^k}(P_k) \leq \frac{1}{K}\sum_{k=1}^{K}\ell_{x^k}(\hat{P}) + \mathcal{O}\Big(\frac{1}{\sqrt{K}}\Big) \quad \text{for any } \hat{P} \in \mathcal{P}.$$

Regret guarantees vs any fixed stepsize

OSGM satisfies

$$\frac{1}{K}\sum_{k=1}^{K} \ell_{x^k}(P_k) \leq \frac{1}{K}\sum_{k=1}^{K} \ell_{x^k}(\hat{P}) + \mathcal{O}\Big(\frac{1}{\sqrt{K}}\Big) \quad \text{for any } \hat{P} \in \mathcal{P}.$$

$\Longrightarrow$ convergence rate of OSGM is (asymptotically) no worse than that of *any fixed stepsize* $\hat{P} \in \mathcal{P}$,

Regret guarantees vs any fixed stepsize

OSGM satisfies

$$\frac{1}{K}\sum_{k=1}^{K}\ell_{x^k}(P_k) \leq \frac{1}{K}\sum_{k=1}^{K}\ell_{x^k}(\hat{P}) + \mathcal{O}\Big(\frac{1}{\sqrt{K}}\Big) \quad \text{for any } \hat{P} \in \mathcal{P}.$$

$\implies$ convergence rate of OSGM is (asymptotically) no worse than that of *any fixed stepsize* $\hat{P} \in \mathcal{P}$, including

- global optimal stepsize $P^\star$
- classical constant stepsize $\frac{1}{L}I$
- stepsize that minimizes the average feedback over trajectory

$$\operatorname*{argmin}_{P\in\mathcal{P}} \frac{1}{K}\sum_{k=1}^{K}\ell_{x^k}(P)$$

Regret guarantee $\Rightarrow$ Superlinear convergence

$$\frac{1}{K}\sum_{k=1}^{K}\ell_{x^k}(P_k) \le \frac{1}{K}\sum_{k=1}^{K}\ell_{x^k}(\hat{P}) + \mathcal{O}\Big(\frac{1}{\sqrt{K}}\Big) \quad \text{for any } \hat{P}\in\mathcal{P}.$$

The fixed stepsize $\hat{P} = [\nabla^2 f(x^\star)]^{-1}$ achieves local quadratic convergence, hence as a consequence of the regret guarantee, OSGM achieves

Superlinear convergence. For example, for `OSGM-R` on a quadratic,

$$f(x^{K+1}) - f(x^\star) \le [f(x^1) - f(x^\star)](\tfrac{1}{K}\underbrace{\textstyle\sum_{k=1}^{K} r_{x^k}([\nabla^2 f(x^\star)]^{-1})}_{=0} + \mathcal{O}(\tfrac{1}{\sqrt{K}}))^K$$

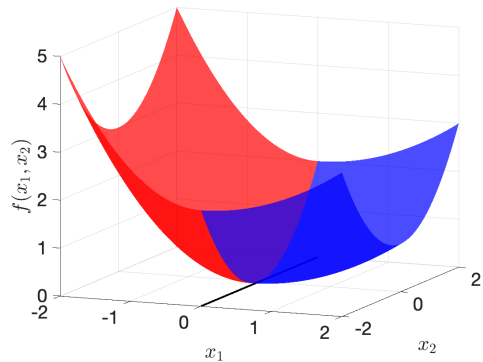
$$\le [f(x^1) - f(x^\star)](\tfrac{C'}{\sqrt{K}})^K.$$

Is a good fixed stepsize enough?

Even the optimal stepsize might not work well:

$$f(x_1, x_2) = \begin{cases} \frac{1}{2}(0.5x_1^2 + x_2^2), & x_1 \geq 0 \\ \frac{1}{2}(1.5x_1^2 + x_2^2), & x_1 < 0 \end{cases}$$

$$[\nabla^2 f(x)]^{-1} = \begin{cases} \text{diag}(0.5, 1), & x_1 \geq 0 \\ \text{diag}(1.5, 1), & x_1 < 0 \end{cases}$$

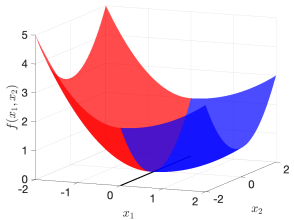


Trajectory-optimal stepsize

Hessian inverse preconditioner

$$P(x) = \begin{cases} \mathrm{diag}(0.5, 1)^{-1} = 0 & x_1 \geq 0 \\ \mathrm{diag}(1.5, 1)^{-1} = 0 & x_1 < 0 \end{cases}$$

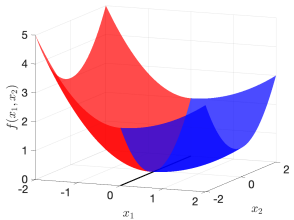
achieves zero ratio feedback $r_x(P(x)) = 0$ for all x.



Trajectory-optimal stepsize

Hessian inverse preconditioner

$$P(x) = \begin{cases} \mathrm{diag}(0.5, 1)^{-1} = 0 & x_1 \geq 0 \\ \mathrm{diag}(1.5, 1)^{-1} = 0 & x_1 < 0 \end{cases}$$



achieves zero ratio feedback $r_x(P(x)) = 0$ for all x.

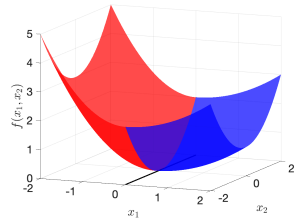
If trajectory $\{x^k\}$ of PGD stays in the same half-plane as the initial point x^1, best preconditioner *for the trajectory* achieves ratio feedback 0:

$$\min_P \sum_{k=1}^{K} r_{x^k}(P) = 0.$$

Trajectory-optimal stepsize

Hessian inverse preconditioner

$$P(x) = \begin{cases} \mathrm{diag}(0.5, 1)^{-1} = 0 & x_1 \geq 0 \\ \mathrm{diag}(1.5, 1)^{-1} = 0 & x_1 < 0 \end{cases}$$



achieves zero ratio feedback $r_x(P(x)) = 0$ for all x.

If trajectory $\{x^k\}$ of PGD stays in the same half-plane as the initial point x^1, best preconditioner *for the trajectory* achieves ratio feedback 0:

$$\min_P \sum_{k=1}^K r_{x^k}(P) = 0.$$

OSGM adapts stepsize to region traversed by the algorithm!

$$\tfrac{1}{K}\textstyle\sum_{k=1}^K r_{x^k}(P_k) \leq \tfrac{1}{K}\sum_{k=1}^K r_{x^k}(\hat{P}) + \mathcal{O}\big(\tfrac{1}{\sqrt{K}}\big) = 0 + \mathcal{O}\big(\tfrac{1}{\sqrt{K}}\big)$$

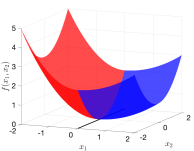
Dynamic regret

What if the trajectory $\{x^k\}$ traverses different regions?
OSGM satisfies a dynamic regret bound wrt. a sequence of competitors $\{\hat{P}_k\}$:

$$\frac{1}{K}\sum_{k=1}^{K} \ell_{x^k}(P_k) \leq \frac{1}{K}\sum_{k=1}^{K} \ell_{x^k}(\hat{P}_k) + \mathcal{O}\left(\frac{1+\mathrm{PL}(\{\hat{P}_k\})}{\sqrt{K}}\right),$$

where $\mathrm{PL}(\{\hat{P}_k\}) := \sum_{k=1}^{K} \|\hat{P}_k - \hat{P}_{k+1}\|_F^2$ is the path-length.

Example.



Take $\hat{P}_k = [\nabla^2 f(x^k)]^{-1}$.

- $\frac{1}{K}\sum_{k=1}^{K} r_{x^k}(\hat{P}_k) = 0.$
- $\mathrm{PL}(\{\hat{P}_k\}) \propto$ number of times $\{x^k\}$ switches from one region to another.

- `OSGM` chooses $\hat{P}_k$ that adapts to each local region at the price of extra regret.

Outline

Practical OSGM

Ingredients for the best practical version of OSGM:

1. **Iterate update: heavy-ball GD.** Parameters include stepsize P_k and momentum parameter β_k:
$$x^{k+1/2} = x^k - P_k \nabla f(x^k) + \beta_k (x^k - x^{k-1})$$
2. **Stepsize update: AdaGrad.** Learn (P_k, β_k) jointly by `AdaGrad`:
$$(P_{k+1}, \beta_{k+1}) = \texttt{AdaGrad}((P_k, \beta_k), \{h_{x^j, x^{j-1}}\}_{j \le k})$$
3. **Feedback: hypergradient.** Define hypergradient feedback wrt potential function $\varphi(x, x^-) = f(x) + \frac{\omega}{2}\|x - x^-\|^2$ to measure the quality of parameters (P_k, β_k) at point x^k.
4. **Landscape: monotone.** Use monotone landscape to ensure convergence.
5. **Domain: diagonal.** Learn diagonal preconditioner for scalability.

Source: Still works provably, see [Chu, Gao, Ye, and Udell (2025)].

Projected hypergradient can be easy to compute

$$\nabla h_x(P) = \frac{\nabla f\big(x - P\,\nabla f(x)\big)\ \nabla f(x)^\mathsf{T}}{\|\nabla f(x)\|^2}$$

$$\mathcal{P} = \left\{ \begin{bmatrix} \alpha & & \\ & \ddots & \\ & & \alpha \end{bmatrix} \right\} \qquad \mathcal{P} = \left\{ \begin{bmatrix} d_1 & & \\ & \ddots & \\ & & d_n \end{bmatrix} \right\} \qquad \mathcal{P} = \{P \succeq 0\}$$

Projected hypergradient:
$$\frac{\nabla f\big(x - P\nabla f(x)\big) \cdot \nabla f(x)}{\|\nabla f(x)\|^2}$$

Projected hypergradient:
$$\frac{\nabla f\big(x - P\nabla f(x)\big) \odot \nabla f(x)}{\|\nabla f(x)\|^2}$$

Projected hypergradient:
$$\mathrm{Proj}_{\succeq 0}\big(P - \eta \nabla h_x(P)\big)$$

Projected hypergradient can be easy to compute

$$\nabla h_x(P) = \frac{\nabla f\big(x - P\,\nabla f(x)\big)\ \nabla f(x)^\mathsf{T}}{\|\nabla f(x)\|^2}$$

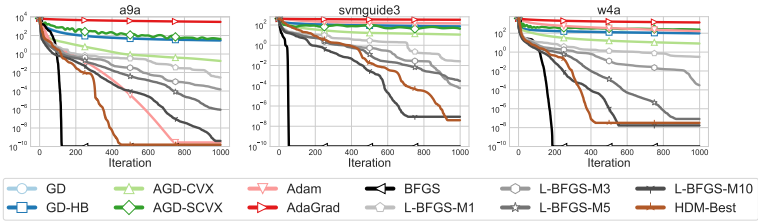
$$\mathcal{P} = \left\{ \begin{bmatrix} \alpha & & \\ & \ddots & \\ & & \alpha \end{bmatrix} \right\} \qquad \mathcal{P} = \left\{ \begin{bmatrix} d_1 & & \\ & \ddots & \\ & & d_n \end{bmatrix} \right\} \qquad \mathcal{P} = \{P \succeq 0\}$$

Projected hypergradient:
$$\frac{\nabla f\big(x - P\nabla f(x)\big) \cdot \nabla f(x)}{\|\nabla f(x)\|^2}$$

Projected hypergradient:
$$\frac{\nabla f\big(x - P\nabla f(x)\big) \odot \nabla f(x)}{\|\nabla f(x)\|^2}$$

Projected hypergradient:
$\mathrm{Proj}_{\succeq 0}\big(P - \eta \nabla h_x(P)\big)$
(don't do this!)

Practical OSGM on LIBSVM



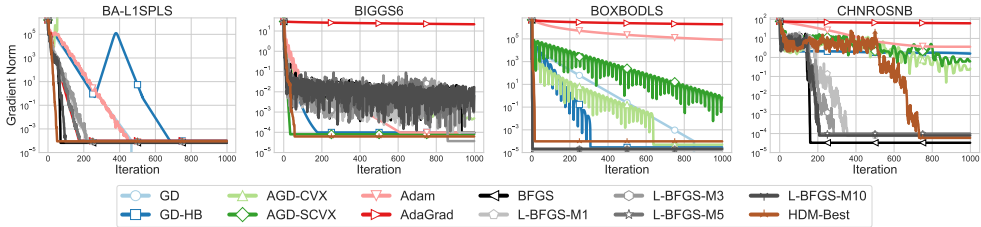
On deterministic convex problems

- comparable performance to L-BFGS-M5/M10
- same memory as L-BFGS-1 and cheaper iterations

Algorithm	SVM	Log. Reg
GD	5	2
GD-HB	9	7
AGD-CVX	8	3
AGD-SCVX	7	6
Adam	26	11
AdaGrad	9	8
L-BFGS-M1	13	11
L-BFGS-M3	20	14
L-BFGS-M5	26	16
L-BFGS-M10	31	18
BFGS	32	26
OSGM	32	21

solved instances in LIBSVM

Practical OSGM on nonconvex CUTEst problems



On deterministic nonconvex problems

- Performance compares well with other first-order adaptive methods
- Sometimes outperforms BFGS!

Which is better? Spectral or diagonal preconditioning?

Both! OSGM + sketch-based preconditioners improve stochastic optimization:

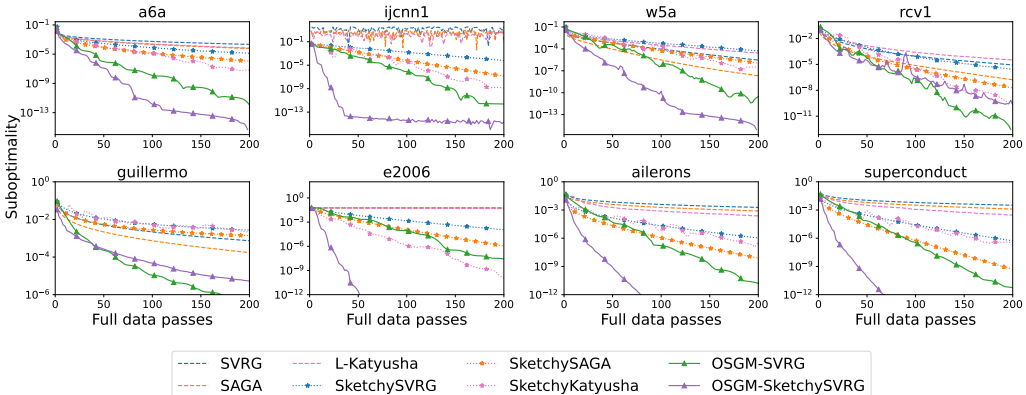
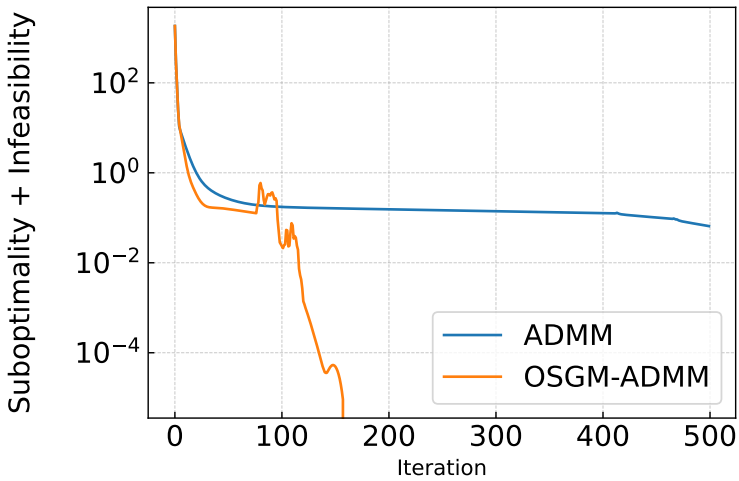


Figure: First row: logistic regression. Second row: ridge regression.

OSGM for ADMM

Use OSGM to learn a diagonal preconditioner for the dual problem in ADMM:



Outline

Where are we now?

OSGM represents

- a new framework for stepsize adaptation of first-order methods
- a new framework to learn a good stepsize (locally, $[\nabla f(x^\star)]^{-1}$)
- a new way to achieve non-asymptotic superlinear convergence

Where are we now?

OSGM represents

- a new framework for stepsize adaptation of first-order methods
- a new framework to learn a good stepsize (locally, $[\nabla f(x^\star)]^{-1}$)
- a new way to achieve non-asymptotic superlinear convergence

There is an existing framework that achieves similar goals:

Where are we now?

OSGM represents

- a new framework for stepsize adaptation of first-order methods
- a new framework to learn a good stepsize (locally, $[\nabla f(x^\star)]^{-1}$)
- a new way to achieve non-asymptotic superlinear convergence

There is an existing framework that achieves similar goals:

Quasi-Newton methods (1959 $\sim$): DFP, BFGS, Broyden ...

OSGM and QN achieve similar guarantees. Why?

Online Hessian learning

Our goal: find a good stepsize $P^\star$. For a (locally) quadratic function,

$$f(x) \sim f(x^\star) + \tfrac{1}{2}\langle x - x^\star, \nabla^2 f(x^\star)(x - x^\star)\rangle + \mathcal{O}(\|x - x^\star\|^3).$$

Optimal stepsize: $P^\star = [\nabla^2 f(x^\star)]^{-1}$.

Online Hessian learning

Our goal: find a good stepsize $P^\star$. For a (locally) quadratic function,

$$f(x) \sim f(x^\star) + \tfrac{1}{2}\langle x - x^\star, \nabla^2 f(x^\star)(x - x^\star)\rangle + \mathcal{O}(\|x - x^\star\|^3).$$

Optimal stepsize: $P^\star = [\nabla^2 f(x^\star)]^{-1}$.

Can we learn $P^\star$ without access to second-order information?

Online Hessian learning

Our goal: find a good stepsize $P^\star$. For a (locally) quadratic function,

$$f(x) \sim f(x^\star) + \tfrac{1}{2}\langle x - x^\star, \nabla^2 f(x^\star)(x - x^\star)\rangle + \mathcal{O}(\|x - x^\star\|^3).$$

Optimal stepsize: $P^\star = [\nabla^2 f(x^\star)]^{-1}$.

Can we learn $P^\star$ without access to second-order information?

Let's take an ML approach.

Online Hessian learning

Our goal: find a good stepsize $P^\star$. For a (locally) quadratic function,

$$f(x) \sim f(x^\star) + \tfrac{1}{2}\langle x - x^\star, \nabla^2 f(x^\star)(x - x^\star)\rangle + \mathcal{O}(\|x - x^\star\|^3).$$

Optimal stepsize: $P^\star = [\nabla^2 f(x^\star)]^{-1}$.

Can we learn $P^\star$ without access to second-order information?

Let's take an ML approach. Stepsize $P^\star$ locally brings any point x to $x^\star$ in one step:

$$x - P^\star \nabla f(x) = x^\star, \text{ for all } x$$

Online Hessian learning

Our goal: find a good stepsize $P^\star$. For a (locally) quadratic function,

$$f(x) \sim f(x^\star) + \tfrac{1}{2}\langle x - x^\star, \nabla^2 f(x^\star)(x - x^\star)\rangle + \mathcal{O}(\|x - x^\star\|^3).$$

Optimal stepsize: $P^\star = [\nabla^2 f(x^\star)]^{-1}$.

Can we learn $P^\star$ without access to second-order information?

Let's take an ML approach. Stepsize $P^\star$ locally brings any point x to $x^\star$ in one step:

$$P^\star \nabla f(x) = x - x^\star, \text{ for all } x$$

For each x near $x^\star$, $(x - x^\star, \nabla f(x))$ provides information about $P^\star$ in a 1D subspace.

Online Hessian learning

$$P^\star \nabla f(x) = x - x^\star$$

With a dataset $\{(x^k - x^\star, \nabla f(x^k))\}$, we can estimate $P^\star$:

Online Hessian learning

$$P^\star \nabla f(x) = x - x^\star$$

With a dataset $\{(x^k - x^\star, \nabla f(x^k))\}$, we can estimate $P^\star$:

- linear regression problem ✓

Online Hessian learning

$$P^\star \nabla f(x) = x - x^\star$$

With a dataset $\{(x^k - x^\star, \nabla f(x^k))\}$, we can estimate $P^\star$:

- linear regression problem ✓
- well-defined loss function ✓

$$\ell(P) := \frac{1}{2}\|P\nabla f(x) - (x - x^\star)\|^2 \qquad \text{or} \qquad \ell(P) := \mathbb{I}\{P\nabla f(x) = x - x^\star\}$$

Online Hessian learning

$$P^\star \nabla f(x) = x - x^\star$$

With a dataset $\{(x^k - x^\star, \nabla f(x^k))\}$, we can estimate $P^\star$:

- linear regression problem ✓
- well-defined loss function ✓

$$\ell(P) := \frac{1}{2}\|P\nabla f(x) - (x - x^\star)\|^2 \qquad \text{or} \qquad \ell(P) := \mathbb{I}\{P\nabla f(x) = x - x^\star\}$$

- scalable online update ✓ (e.g. recursive least squares)

$$P_{k+1} = \arg\min_{P \in \mathcal{P}} \{\ell_k(P) + \text{dist}(P, P_k)\}$$

Online Hessian learning

$$P^\star \nabla f(x) = x - x^\star$$

With a dataset $\{(x^k - x^\star, \nabla f(x^k))\}$, we can estimate $P^\star$:

- linear regression problem ✓
- well-defined loss function ✓

$$\ell(P) := \frac{1}{2}\|P\nabla f(x) - (x - x^\star)\|^2 \qquad \text{or} \qquad \ell(P) := \mathbb{I}\{P\nabla f(x) = x - x^\star\}$$

- scalable online update ✓ (e.g. recursive least squares)

$$P_{k+1} = \arg\min_{P \in \mathcal{P}} \{\ell_k(P) + \text{dist}(P, P_k)\}$$

- Uhoh: part of the data, $x^\star$, is unknown!

$$\implies \text{eliminate } x^\star \text{ to learn } P^\star$$

Quasi-Newton methods

This relation is useless unless we know $x^\star$:

$$P\nabla f(x) = x - x^\star.$$

Quasi-Newton methods

This relation is useless unless we know $x^\star$:

$$P\nabla f(x) = x - x^\star.$$

This relation is useless unless we know $x^\star$:

$$P\nabla f(y) = y - x^\star.$$

Quasi-Newton methods

This relation is useless unless we know $x^\star$:

$$P\nabla f(x) = x - x^\star.$$

This relation is useless unless we know $x^\star$:

$$P\nabla f(y) = y - x^\star.$$

Subtract them to eliminate $x^\star$:

$$P[\nabla f(x) - \nabla f(y)] = (x - x^\star) - (y - x^\star) = x - y$$

- We've derived the secant equation, the workhorse of quasi-Newton methods.

Quasi-Newton methods

- A computable relation ✓

$$P[\nabla f(x) - \nabla f(y)] = x - y$$

- linear regression + well-defined loss function ✓
- scalable online update ✓

$$P_{k+1} = \arg\min_{P \in \mathcal{P}} \{\ell_k(P) + \text{dist}(P, P_k)\}$$

Variants of Quasi-Newton methods

$$P_{k+1} = \arg\min_{P \in \mathcal{P}} \{\ell_k(P) + \text{dist}(P, P_k)\}$$

Broyden's method

$$\mathcal{P} = \mathbb{R}^{n \times n},\ \text{dist}(P, Q) = \tfrac{1}{2}\|P - Q\|_F^2, \quad \ell_k(P) = \mathbb{I}\{P[\nabla f(x^{k+1}) - \nabla f(x^k)] = x^{k+1} - x^k\}.$$

Powell-symmetric-Broyden

$$\mathcal{P} = \mathbb{S}^n,\ \text{dist}(P, Q) = \tfrac{1}{2}\|P - Q\|_F^2, \quad \ell_k(P) = \mathbb{I}\{P[\nabla f(x^{k+1}) - \nabla f(x^k)] = x^{k+1} - x^k\}.$$

DFP & BFGS

$$\mathcal{P} = \mathbb{S}^n_+,\ \text{dist}(P, Q) = \log\det \text{ div.}, \quad \ell_k(P) = \mathbb{I}\{P[\nabla f(x^{k+1}) - \nabla f(x^k)] = x^{k+1} - x^k\}.$$

Penalized DFP & BFGS

$$\mathcal{P} = \mathbb{S}^n_+,\ \text{dist}(P, Q) = \log\det \text{ div.}, \quad \ell_k(P) = \tfrac{1}{2}\|P[\nabla f(x^{k+1}) - \nabla f(x^k)] - (x^{k+1} - x^k)\|^2.$$

OSGM

Let's take a step back.

OSGM

Let's take a step back. Again, this relation is useless unless we know $x^\star$:

$$P\nabla f(x) = x - x^\star$$

OSGM

Let's take a step back. Again, this relation is useless unless we know $x^\star$:

$$\ell(P) = \tfrac{1}{2}\|P\nabla f(x) - (x - x^\star)\|^2$$

- The least-square loss cannot be computed in the Euclidean norm due to $x^\star$.

OSGM

Let's take a step back. Again, this relation is useless unless we know $x^\star$:

$$\ell(P) = \tfrac{1}{2}\|P\nabla f(x) - (x - x^\star)\|^2$$

- The least-square loss cannot be computed in the Euclidean norm due to $x^\star$.
- What if we use another norm? In the Hessian norm, locally

$$\ell(P) = \tfrac{1}{2}\|P\nabla f(x) - (x - x^\star)\|^2_{\nabla^2 f(x^\star)} = f(x - P\nabla f(x)) - f(x^\star)$$

OSGM

Let's take a step back. Again, this relation is useless unless we know $x^\star$:

$$\ell(P) = \tfrac{1}{2}\|P\nabla f(x) - (x - x^\star)\|^2$$

- The least-square loss cannot be computed in the Euclidean norm due to $x^\star$.
- What if we use another norm? In the Hessian norm, locally

$$\ell(P) = \tfrac{1}{2}\|P\nabla f(x) - (x - x^\star)\|^2_{\nabla^2 f(x^\star)} = f(x - P\nabla f(x)) - f(x^\star)$$

- Compare to OSGM feedback functions, which scale and translate this loss:

$$r_x(P) = \frac{\ell(P)}{f(x) - f(x^\star)}, \qquad h_x(P) = \frac{\ell(P) + f(x^\star) - f(x)}{\|\nabla f(x)\|^2}$$

Variants of OSGM

$$P_{k+1} = \arg\min_{P \in \mathcal{P}} \{\ell_k(P) + \text{dist}(P, P_k)\}$$

`OSGM-R`

$$\mathcal{P} = \mathbb{R}^{n \times n}, \quad \text{dist}(P, Q) = \tfrac{1}{2}\|P - Q\|_F^2, \quad \ell_k(P) = \tfrac{f(x^k - P\nabla f(x^k)) - f^\star}{f(x^k) - f^\star}.$$

`OSGM-H`

$$\mathcal{P} = \mathbb{R}^{n \times n}, \quad \text{dist}(P, Q) = \tfrac{1}{2}\|P - Q\|_F^2, \quad \ell_k(P) = \tfrac{f(x^k - P\nabla f(x^k)) - f^\star}{\|\nabla f(x^k)\|^2} + \tfrac{f^\star - f(x^k)}{\|\nabla f(x^k)\|^2}.$$

OSGM vs. Quasi-Newton

- Same idea: learn the Hessian from $P\nabla f(x) - (x - x^\star)$
- Different ways of removing $x^\star$
- Similar convergence guarantees

OSGM vs. Quasi-Newton

- Same idea: learn the Hessian from $P\nabla f(x) - (x - x^\star)$
- Different ways of removing $x^\star$
- Similar convergence guarantees

Why not both?

OSGM vs. Quasi-Newton

- Same idea: learn the Hessian from $P\nabla f(x) - (x - x^\star)$
- Different ways of removing $x^\star$
- Similar convergence guarantees

Why not both?

OSGM vs. Quasi-Newton

- Same idea: learn the Hessian from $P\nabla f(x) - (x - x^\star)$
- Different ways of removing $x^\star$
- Similar convergence guarantees

Why not both?

- OSGM + QN can be combined to get the best of both worlds.
  e.g., designing $\ell_k(P)$ to combine both online feedback and secant equation

OSGM + Quasi-Newton

We can learn via both the secant equation and hypergradient feedback:

$$\ell_k(P) = h_x(P) + \tfrac{\omega}{2}\|x - y - P(\nabla f(x) - \nabla f(y))\|^2$$

OSGM + Quasi-Newton

We can learn via both the secant equation and hypergradient feedback:

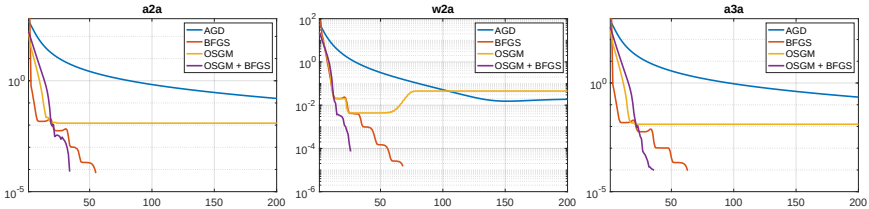
$$\ell_k(P) = h_x(P) + \tfrac{\omega}{2}\|x - y - P(\nabla f(x) - \nabla f(y))\|^2$$

This instantiation of the algorithm updates P by a rank-4 matrix every iteration,

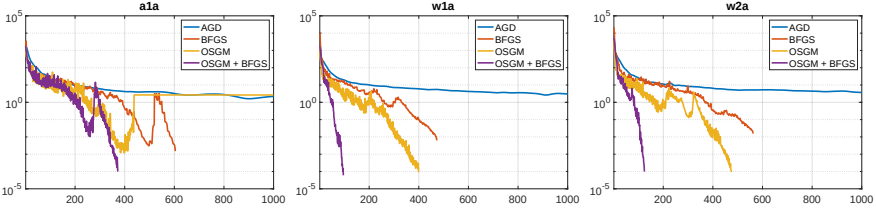
$$P_{k+1} = \underbrace{(I - \tfrac{s^k(y^k)^\top}{\langle y^k, s^k\rangle})P_k(I - \tfrac{y^k(s^k)^\top}{\langle s^k, y^k\rangle}) + \tfrac{s^k(s^k)^\top}{\langle y^k, s^k\rangle}}_{\text{BFGS}} + \underbrace{\alpha_k(P_k u^k - v^k)(P_k u^k - v^k)^\top + \beta_k v^k (v^k)^\top}_{\text{OSGM}},$$

with $s^k = x^{k+1} - x^k$, $y^k = \nabla f(x^{k+1}) - \nabla f(x^k)$, $u^k = \frac{\nabla f(x^{k+1})}{\|\nabla f(x^k)\|}$, $v^k = \frac{\nabla f(x^k)}{\|\nabla f(x^k)\|}$

OSGM + BFGS



Regularized logistic regression (AGD: Accelerated Gradient Descent). Y-axis: $\|\nabla f(x^k)\|$



Regularized SVM problems. Y-axis: $\|\nabla f(x^k)\|$

Conclusion

- OSGM is a new framework for adapting hyperparameters in first-order methods.
- OSGM achieves competitive convergence rates, superlinear convergence, and trajectory-dependent convergence.
- OSGM is practical and outperforms existing adaptive first-order methods on real-world problems.

Conclusion

- OSGM is a new framework for adapting hyperparameters in first-order methods.
- OSGM achieves competitive convergence rates, superlinear convergence, and trajectory-dependent convergence.
- OSGM is practical and outperforms existing adaptive first-order methods on real-world problems.

Where can I learn more?

- Wenzhi Gao, Ya-Chi Chu, Yinyu Ye, and Madeleine Udell. "Gradient Methods with Online Scaling." Conference on Learning Theory (COLT) 2025.
- Ya-Chi Chu, Wenzhi Gao, Yinyu Ye, and Madeleine Udell. "Provable and Practical Online Learning Rate Adaptation with Hypergradient Descent." International Conference on Machine Learning (ICML) 2025. arXiv:2502.11229 (2025).
- Wenzhi Gao, Ya-Chi Chu, Yinyu Ye, and Madeleine Udell. "Gradient Methods with Online Scaling Part I. Theoretical Foundations." arXiv: 2025.
- Ya-Chi Chu, Wenzhi Gao, Yinyu Ye, and Madeleine Udell. "Gradient Methods with Online Scaling Part II. Practical Aspects Foundations." arXiv: 2025.

Online gradient descent

Analysis of online gradient descent: $P_{k+1} = P_k - \eta \nabla r_{x^k}(P_k)$: for any $\hat{P} \in \mathcal{P}$,

$$\begin{aligned}
\|P_{k+1} - \hat{P}\|_F^2 &= \|P_k - \eta \nabla r_{x^k}(P_k) - \hat{P}\|_F^2 \\
&= \|P_k - \hat{P}\|_F^2 - 2\eta \langle \nabla r_{x^k}(P_k), P_k - \hat{P} \rangle + \eta^2 \|\nabla r_{x^k}(P_k)\|_F^2 \\
&\qquad \text{(Expand)} \\
&\leq \|P_k - \hat{P}\|_F^2 - 2\eta [r_{x^k}(P_k) - r_{x^k}(\hat{P})] + \eta^2 \|\nabla r_{x^k}(P_k)\|_F^2 \\
&\qquad \text{(Convexity)} \\
&\leq \|P_k - \hat{P}\|_F^2 - 2\eta [r_{x^k}(P_k) - r_{x^k}(\hat{P})] + \eta^2 G^2 \qquad \text{(Bounded gradient)}
\end{aligned}$$

Re-arranging:

$$\frac{1}{2\eta} [\|P_{k+1} - \hat{P}\|_F^2 - \|P_k - \hat{P}\|_F^2] \leq -[r_{x^k}(P_k) - r_{x^k}(\hat{P})] + \frac{\eta}{2} G^2$$

Online gradient descent

$$\underbrace{\frac{1}{2\eta}[\|P_{k+1} - \hat{P}\|_F^2 - \|P_k - \hat{P}\|_F^2]}_{\text{What we learn}} \leq -\underbrace{[r_{x^k}(P_k) - r_{x^k}(\hat{P})]}_{\text{What we regret}} + \frac{\eta}{2} G^2$$

- If P_k underperforms $\hat{P}$, $r_{x^k}(P_k) > r_{x^k}(\hat{P}) \Rightarrow P_{k+1}$ gets closer to $\hat{P}$ up to error $\frac{\eta}{2} G^2$
  We learn something whenever we regret a lot
- If there is not too much to learn ($\|P_1 - \hat{P}\|_F^2 < \infty$), then there's not too much regret

$$\underbrace{\sum_{k=1}^K r_{x^k}(P_k) - r_{x^k}(\hat{P})}_{\text{Regret}} \leq \underbrace{\frac{1}{2\eta}\|P_1 - \hat{P}\|_F^2}_{\text{What we can learn}} + \underbrace{\frac{\eta}{2} K G^2}_{\text{Error}} = \mathcal{O}(\tfrac{1}{\eta} + \eta K) \overset{\eta = \frac{1}{\sqrt{K}}}{=} \mathcal{O}(\sqrt{K})$$