Second-order kernel online convex optimization with adaptive sketching

Daniele Calandriello, Alessandro Lazaric, Michal Valko

Inria

Motivation

- Non-parametric models are versatile and accurate
- First-order methods are fast but high regret
- Second-order methods suffer low regret but slow

$\mathcal{O}(t^3)$ *time* $\mathcal{O}(t^2)$ *space* (t steps)

- *Current limitation:* No interpretation for non-parametric regret, no approximate second-order methods

We propose Sketched-KONS, the first approximate algorithm for second-order Kernel Online Convex Optimization

↳ approximation ⇒ $1/\gamma$ times more regret but a γ^2 speedup
↳ using a novel kernel matrix sketching technique
↳ regret scales with the effective dimension of the problem

Kernel Online Convex Optimization

Online game between learner and adversary, at each round $t \in [T]$

1. the adversary reveals a new point $\varphi(\mathbf{x}_t) = \boldsymbol{\phi}_t \in \mathcal{H}$
2. the learner chooses $\mathbf{w}_t$ and predicts $f_{\mathbf{w}_t}(\mathbf{x}_t) = \varphi(\mathbf{x}_t)^\mathsf{T}\mathbf{w}_t$,
3. the adversary reveals the curved loss ℓ_t,
4. the learner suffers $\ell_t(\boldsymbol{\phi}_t^\mathsf{T}\mathbf{w}_t)$ and observes gradient $\mathbf{g}_t$.

Kernel

- $\varphi(\cdot) : \mathcal{X} \to \mathcal{H}$ is the high-dimensional (possibly infinite) map
- $\boldsymbol{\Phi}_t = [\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_t]$, $\boldsymbol{\Phi}_t^\mathsf{T}\boldsymbol{\Phi}_t = \mathbf{K}_t$ (kernel trick)
- $\mathbf{g}_t = \ell_t'(\boldsymbol{\phi}_t^\mathsf{T}\mathbf{w}_t)\boldsymbol{\phi}_t := \dot{g}_t\boldsymbol{\phi}_t$

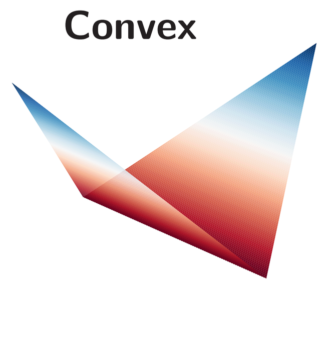
Minimize **regret**

$$R(\mathbf{w}) = \sum_{t=1}^T \ell_t(\boldsymbol{\phi}_t^\mathsf{T}\mathbf{w}_t) - \ell_t(\boldsymbol{\phi}_t^\mathsf{T}\mathbf{w})$$

against the best-in-hindsight $\mathbf{w}^* := \arg\min_{\mathbf{w}\in\mathcal{H}} \sum_{t=1}^T \ell_t(\boldsymbol{\phi}_t^\mathsf{T}\mathbf{w})$

Curvature and first vs second order

Convex



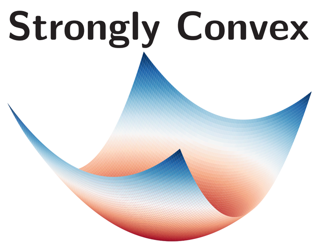
First order (GD)

Zinkevich 2003, Kivinen et al. 2004

- $\mathcal{O}(d)/\mathcal{O}(t)$ time/space per-step
- regret $\sqrt{T}$

Approximation avoids $\mathcal{O}(t)$ runtime
↳ but introduces approximation error (potentially $\mathcal{O}(T)$ regret)

Strongly Convex



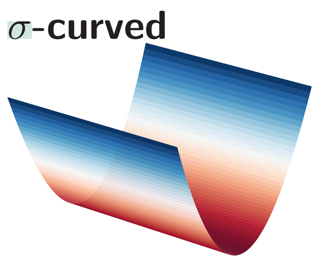
First order (GD)

Hazan, Rakhlin, et al. 2008

- $\mathcal{O}(d)/\mathcal{O}(t)$ time/space per-step
- regret $\log(T)$

but often not satisfied in practice
↳(e.g. $(y_t - \boldsymbol{\phi}_t^\mathsf{T}\mathbf{w}_t)^2$)

σ-curved



First order (GD)

- $\mathcal{O}(d)/\mathcal{O}(t)$ time/space per-step
- regret $\sqrt{T}$

Second order (Newton-like)

Hazan, Kalai, et al. 2006, Zhdanov and Kalnishkan 2010

- regret $\log(T)$
- $\mathcal{O}(d^2)/\mathcal{O}(t^2)$ time/space per-step

Fast approximations for linear case

Luo et al. 2016

↳no approximate methods for kernel case

Assumptions

1: the losses ℓ_t are scalar Lipschitz $|\ell_t'(z)| \le L$

2: $\ell_t(\boldsymbol{\phi}_t^\mathsf{T}\mathbf{w}) \ge \ell_t(\boldsymbol{\phi}_t^\mathsf{T}\mathbf{u}) + \nabla\ell_t(\boldsymbol{\phi}_t^\mathsf{T}\mathbf{u})^\mathsf{T}(\mathbf{w}-\mathbf{u}) + \sigma\left(\nabla\ell_t(\boldsymbol{\phi}_t^\mathsf{T}\mathbf{u})^\mathsf{T}(\mathbf{w}-\mathbf{u})\right)^2$

Challenge

Reduce computational cost without losing logarithmic regret?

References

[1] Elad Hazan, Adam Kalai, et al. "Logarithmic regret algorithms for online convex optimization". In: *COLT*. 2006.

[2] Elad Hazan, Alexander Rakhlin, et al. "Adaptive online gradient descent". In: *NIPS*. 2008.

[3] J. Kivinen et al. "Online Learning with Kernels". In: *IEEE Transactions on Signal Processing* (2004).

[4] Haipeng Luo et al. "Efficient second-order online learning via sketching". In: *Neural Information Processing Systems*. 2016.

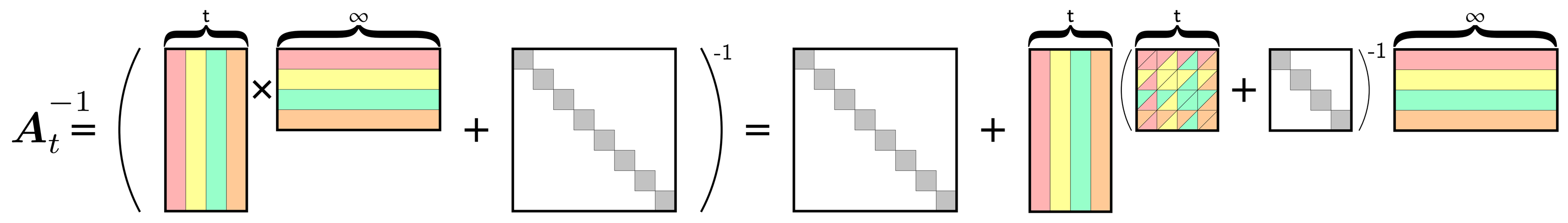
[5] Fedor Zhdanov and Yuri Kalnishkan. "An Identity for Kernel Ridge Regression". In: *Algorithmic Learning Theory*. 2010.

[6] Martin Zinkevich. "Online Convex Programming and Generalized Infinitesimal Gradient Ascent". In: *ICML*. 2003.

Kernel Online Newton Step (KONS)

Second-Order Gradient Descent

1. $\mathbf{A}_0 = \alpha\mathbf{I}$
2. $\mathbf{A}_t = \mathbf{A}_{t-1} + \sigma\mathbf{g}_t\mathbf{g}_t^\mathsf{T}$
3. $\mathbf{w}_{t+1} = \mathbf{w}_t - \mathbf{A}_t^{-1}\mathbf{g}_t$



$$R(\mathbf{w}) \le \mathcal{O}\left(\sum_{t=1}^T \mathbf{g}_t^\mathsf{T}\mathbf{A}_t^{-1}\mathbf{g}_t\right) \le \mathcal{O}\left(\sum_{t=1}^T \mathbf{g}_t^\mathsf{T}\left(\mathbf{G}_t\mathbf{G}_t^\mathsf{T}+\alpha\mathbf{I}\right)^{-1}\mathbf{g}_t\right) \le \mathcal{O}\left(L\sum_{t=1}^T \boldsymbol{\phi}_t^\mathsf{T}\left(\boldsymbol{\Phi}_t\boldsymbol{\Phi}_t^\mathsf{T}+\alpha\mathbf{I}\right)^{-1}\boldsymbol{\phi}_t\right) \le \begin{cases}\text{LOCO: } \mathcal{O}(d\log(T)) \\ \text{KOCO: } \mathcal{O}(\log(\operatorname{Det}(\mathbf{K}_T+\alpha\mathbf{I})))\end{cases}$$

Effective dimension

Lemma 1

$$d_{\text{onl}}^T(\alpha) := \sum_{t=1}^T \boldsymbol{\phi}_t^\mathsf{T}\left(\boldsymbol{\Phi}_t\boldsymbol{\Phi}_t^\mathsf{T}+\alpha\mathbf{I}\right)^{-1}\boldsymbol{\phi}_t$$
$$\le \log(\operatorname{Det}(\mathbf{K}_T/\alpha+\mathbf{I})) \le 2\mathbf{d}_{\text{eff}}^T(\alpha)\log(T/\alpha).$$

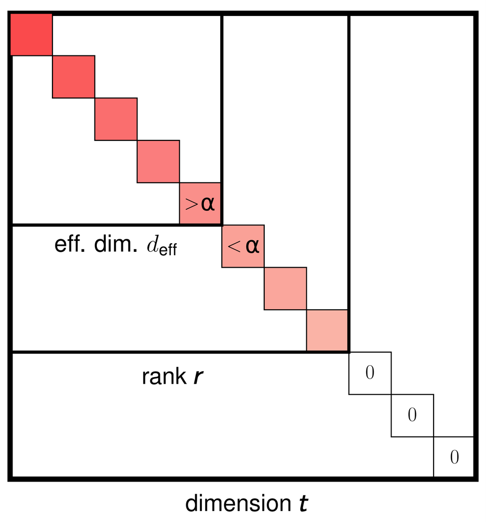
Given a kernel matrix $\mathbf{K}_T \in \mathbb{R}^{t\times t}$

⇒ α-ridge leverage score

$$\tau_{T,i}(\alpha) = \mathbf{e}_{T,i}\mathbf{K}_T^\mathsf{T}(\mathbf{K}_T+\alpha\mathbf{I}_T)^{-1}\mathbf{e}_{T,i} = \boldsymbol{\phi}_i^\mathsf{T}(\boldsymbol{\Phi}_T\boldsymbol{\Phi}_T^\mathsf{T}+\alpha\mathbf{I})^{-1}\boldsymbol{\phi}_i$$

⇒ Effective dimension

$$\mathbf{d}_{\text{eff}}(\alpha)_T = \sum_{i=1}^T \tau_{T,i}(\alpha) = \operatorname{Tr}\left(\mathbf{K}_T(\mathbf{K}_T+\alpha\mathbf{I}_T)^{-1}\right) = \sum_{i=1}^T \frac{\lambda_i(\mathbf{K}_T)}{\lambda_i(\mathbf{K}_T)+\alpha} \le \operatorname{Rank}(\mathbf{K}_T) = r$$



Kernel Online Row Sampling (KORS)

Input: Regularization α, accuracy ε, budget β

1: Initialize $\mathcal{I}_0 = \emptyset$
2: **for** $t = \{0, \ldots, T-1\}$ **do**
3: receive $\boldsymbol{\phi}_t$
4: construct temporary dictionary $\overline{\mathcal{I}}_t := \mathcal{I}_{t-1} \cup (t,1)$
5: compute $\widetilde{p}_t = \min\{\beta\widetilde{\tau}_{t,t}, 1\}$ using $\overline{\mathcal{I}}_t$ and Eq. 4 in the paper.
6: draw $z_t \sim \mathcal{B}(\widetilde{p}_t)$ and if $z_t = 1$, add $(t, 1/\widetilde{p}_t)$ to $\mathcal{I}_t$
7: **end for**

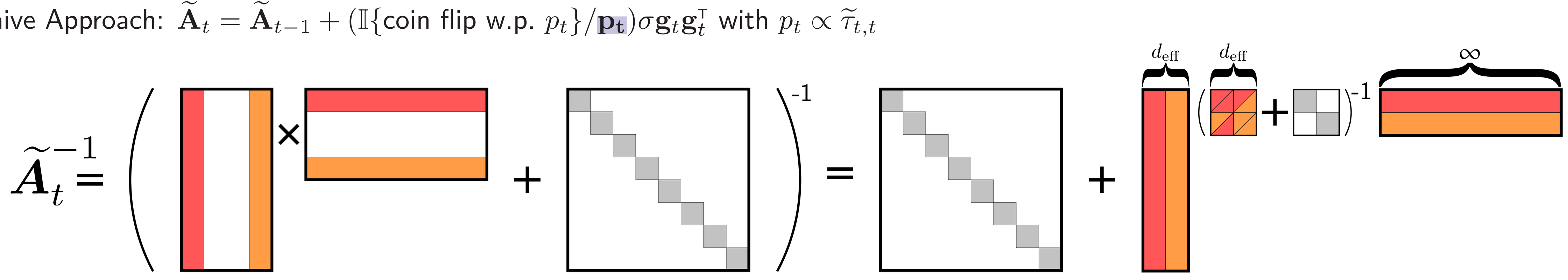
Theorem 1. *Given parameters $0 < \varepsilon \le 1$, $0 < \alpha$, $0 < \delta < 1$, let $\rho = \frac{1+\varepsilon}{1-\varepsilon}$ and run KORS with $\beta \ge 3\log(T/\delta)/\varepsilon^2$. Then w.p. $1-\delta$, for all steps $t \in [T]$,*

(1) $(1-\varepsilon)\mathbf{A}_t \preceq \mathbf{A}_t^{\mathcal{I}_t} \preceq (1+\varepsilon)\mathbf{A}_t$.
(2) $|\mathcal{I}_t| \le d_{\text{eff}}^t(\alpha)\frac{6\rho\log^2\left(\frac{2T}{\delta}\right)}{\varepsilon^2}$.
(3) Satisfies $\tau_{t,t} \le \widetilde{\tau}_{t,t} \le \rho\tau_{t,t}$.

Moreover, the algorithm runs in $\mathcal{O}(d_{\text{eff}}^t(\alpha)^2\log^4(T))$ space, and $\mathcal{O}(d_{\text{eff}}^t(\alpha)^2\log^4(T))$ time per iteration.

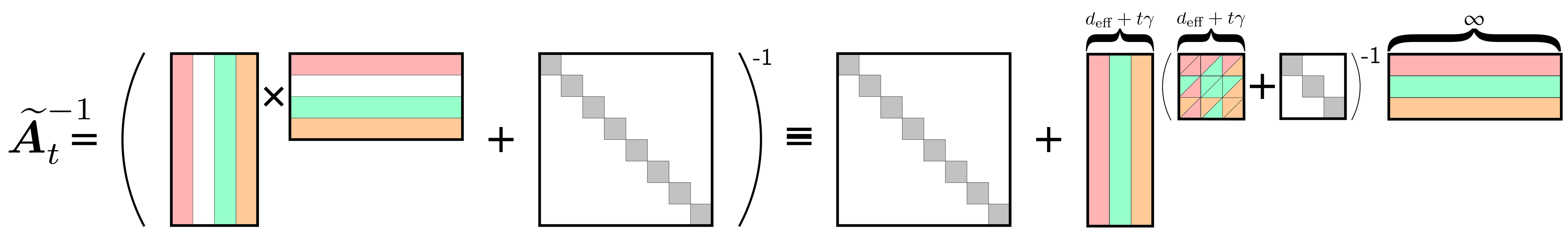
Sketched-KONS

Naive Approach: $\widetilde{\mathbf{A}}_t = \widetilde{\mathbf{A}}_{t-1} + (\mathbb{I}\{\text{coin flip w.p. } p_t\}/\mathbf{p_t})\sigma\mathbf{g}_t\mathbf{g}_t^\mathsf{T}$ with $p_t \propto \widetilde{\tau}_{t,t}$



- w.h.p. $\widetilde{A}_t$ updated only $d_{\text{eff}}^T(\alpha)\log^2(T)$ times
- $\widetilde{\mathcal{O}}(d_{\text{eff}}^T(\alpha)^2+t)$ per-step space/time complexity
- Expected regret $d_{\text{eff}}^T(\alpha)\log(T)$
- The weights $1/p_t \sim 1/\widetilde{\tau}_{t,t}$ can be large
  ↳ large variance

Sketched-KONS $\widetilde{\mathbf{A}}_t = \widetilde{\mathbf{A}}_{t-1} + (\mathbb{I}\{\text{coin flip w.p. } p_t\} \quad)\sigma\mathbf{g}_t\mathbf{g}_t^\mathsf{T}$ with $p_t \propto \max\{\gamma, \widetilde{\tau}_{t,t}\}$



Theorem 2. *For any sequence of losses ℓ_t satisfying Asm.1-2, let $\widetilde{\tau}_{\min} = \min_{t=1}^T \widetilde{\tau}_{t,t}$. For all t, $\alpha \le \sqrt{T}$, $\beta \ge 3\log(T/\delta)/\varepsilon^2$, then w.p. $1-\delta$ the regret of* Sketched-KONS *satisfies*

$$\widetilde{R}_T \le \alpha\|\mathbf{w}^*\|^2 + 2\frac{\boldsymbol{d}_{\text{eff}}^T(\boldsymbol{\alpha}/(\boldsymbol{\sigma L}^2))\log(2\boldsymbol{\sigma L}^2\boldsymbol{T})}{\boldsymbol{\sigma}\max\{\boldsymbol{\gamma}, \boldsymbol{\beta}\widetilde{\boldsymbol{\tau}}_{\min}\}}, \qquad (1)$$

and the algorithm runs in $\mathcal{O}(d_{\text{eff}}^t(\alpha)^2+t^2\gamma^2)$ time and $\mathcal{O}(d_{\text{eff}}^t(\alpha)^2+t^2\gamma^2)$ space complexity for each iteration t.

- Trade-off computation and regret
  ↳ $1/\gamma$ increase in regret for γ^2 space/time improvement
- Neither uniform nor RLS
  ↳ keep updates with high $\tau_{t,t}$ for accuracy
  uniformly update for stability
- Can we get rid of dependency on t?
  ↳ not when $\mathbf{A}_t - \mathbf{A}_{t-1} = w_t\mathbf{g}_t\mathbf{g}_t^\mathsf{T}$

Counterexample

Adversary always plays same sample ϕ_{exp}, but alternates label $\{+1, -1\}$
Class of updates: $\mathbf{A}_t - \mathbf{A}_{t-1} = w_t\mathbf{g}_t$

#SV budget $B = \#\mathbb{I}\{w_t \neq 0\}$ drives complexity
cumulative weight $W_t = \sum_{s=1}^t w_s$ drives regret

$$\begin{aligned}R(\mathbf{w}^*) &\le \sum_{t=1}^T \mathbf{g}_t^\mathsf{T}\mathbf{A}_t^{-1}\mathbf{g}_t + \sum_{t=1}^T (\mathbf{w}_t-\mathbf{w}^*)^\mathsf{T}(\mathbf{A}_t - \mathbf{A}_{t-1} - \sigma_t\mathbf{g}_t\mathbf{g}_t^\mathsf{T})(\mathbf{w}_t-\mathbf{w}^*) \\ &\le \sum_{t=1}^T \mathbf{g}_t^\mathsf{T}\mathbf{A}_t^{-1}\mathbf{g}_t + \sum_{t=1}^T (w_t-\sigma_t)^2(\mathbf{g}_t^\mathsf{T}(\mathbf{w}_t-\mathbf{w}^*))^2 \\ &\le \underbrace{\sum_{s=1}^t \frac{1}{W_s+\alpha}}_{R_G} + \underbrace{\sum_{s=1}^t \max\{0, w_t - \sigma\}}_{R_D}\end{aligned}$$

(1) Increase W_t quickly ↳reduce R_G (2) Increase W_t slowly ↳reduce R_D (3) Increase W_t sparsely ↳reduce B

Contrasting goals cannot be satisfied at the same time.

Only constant speedup over exact

How can we avoid this?

Support Removal
Learn how to remove old $\mathbf{g}_{t-1}$ from $\mathbf{A}_t$?
↳ $(\mathbf{w}_t - \mathbf{w}^*)^\mathsf{T}(\mathbf{g}_t\mathbf{g}_t^\mathsf{T} - \mathbf{g}_{t-1}\mathbf{g}_{t-1}^\mathsf{T})(\mathbf{w}_t - \mathbf{w}^*)$ could be large

Functional embedding
Instead of approximating $\mathbf{A}_t$, approximate ϕ_t
↳ Random features not strong enough (yet)

Avron et al. ICML'17 satisfy guarantee (1) of Thm. 1
↳ only in batch setting

Nyström-based embeddings?
↳ ongoing work