Efficient second-order online kernel learning with adaptive embedding

Daniele Calandriello, Alessandro Lazaric, Michal Valko

Inria

Motivation

Non-parametric models are versatile and accurate.
Computing solution online is still accurate
↳second-order methods' achieve logarithmic regret

Current limitations

- ▶ Curse of kernelization makes them slow down over time:
  ↳ $\mathcal{O}(t)$ space and time *per-step*.
- ▶ Adversary can exploit fixed approximation schemes:
  ↳force linear approximation error

We propose PROS-N-KONS, the first fixed-cost approximate online kernel learning algorithm achieving logarithmic regret

↳ Nyström + leverage score sampling → embed points in $\mathbb{R}^j$
↳ adapts embedding online: cannot be exploited
↳ embedding size j scales only with effective dimension
↳ preserves logarithmic rate

Online kernel learning

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

1. the adversary reveals a new point $\varphi(\mathbf{x}_t) = \phi_t \in \mathcal{H}$
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 $\ell_t(\phi_t^\mathsf{T}\mathbf{w}_t)$ and observes gradient $\mathbf{g}_t$.

Kernel

- $\varphi(\cdot) : \mathcal{X} \to \mathcal{H}$ is the high-dimensional (possibly infinite) map
- $\mathbf{\Phi}_t = [\phi_1, \ldots, \phi_t]$, $\mathbf{\Phi}_t^\mathsf{T}\mathbf{\Phi}_t = \mathbf{K}_t$ (kernel trick)
- $\mathbf{g}_t = \ell_t'(\phi_t^\mathsf{T}\mathbf{w}_t)\phi_t := \dot{g}_t\phi_t$

Minimize **regret**

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

against the best-in-hindsight $\mathbf{w}^* := \arg\min_{\mathbf{w}\in\mathcal{S}} \sum_{t=1}^{T} \ell_t(\phi_t^\mathsf{T}\mathbf{w})$ in feasible space $\mathcal{S} = \cap_t \mathcal{S}_t = \cap_t\{\mathbf{w} : |\phi_t^\mathsf{T}\mathbf{w}| \leq C\}$

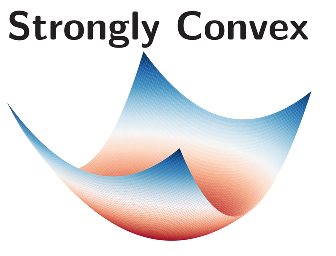
Curvature and first vs second order

Convex

First order (GD) Zinkevich 2003, Kivinen et al. 2004

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

Strongly Convex

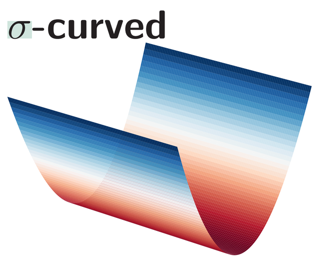


First order (GD) Hazan, Rakhlin, et al. 2008

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

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

σ-curved

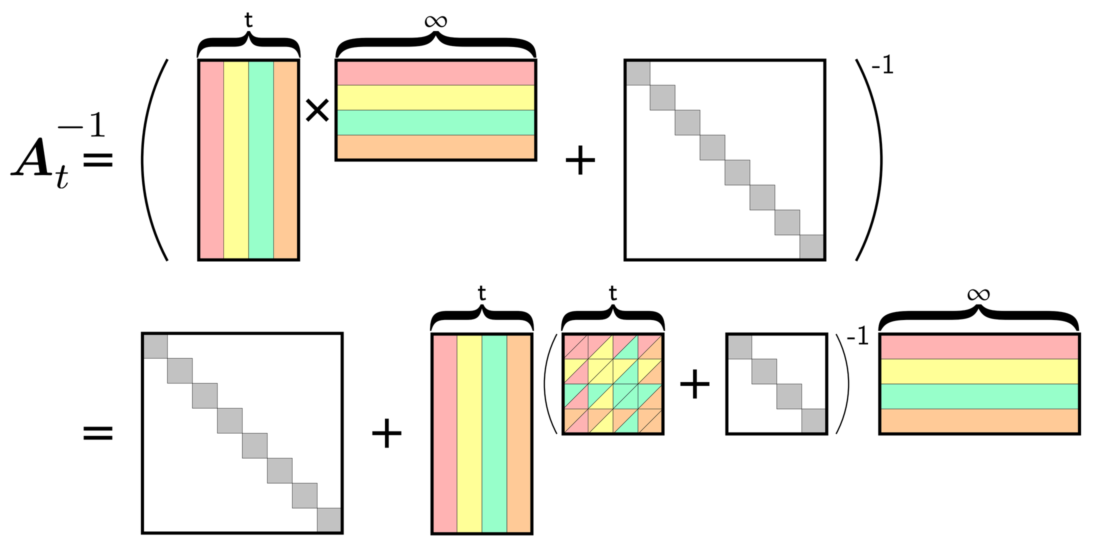


Second order (Newton-like)
Hazan, Kalai, et al. 2006, Zhdanov and Kalnishkan 2010

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

Kernelized Online Newton Step (KONS)

$$\mathbf{A}_0 = \alpha\mathbf{I}, \quad \mathbf{A}_t = \mathbf{A}_{t-1} + \sigma\mathbf{g}_t\mathbf{g}_t^\mathsf{T}, \quad \mathbf{w}_{t+1} = \Pi_{\mathcal{S}_{t+1}}^{\mathbf{A}_t}(\mathbf{w}_t - \mathbf{A}_t^{-1}\mathbf{g}_t).$$



Assumptions

1: the losses ℓ_t are scalar Lipschitz $|\ell_t'(z)| \leq L$
2: $\ell_t(\phi_t^\mathsf{T}\mathbf{w}) \geq \ell_t(\phi_t^\mathsf{T}\mathbf{u}) + \nabla\ell_t(\phi_t^\mathsf{T}\mathbf{u})^\mathsf{T}(\mathbf{w}-\mathbf{u}) + \sigma\left(\nabla\ell_t(\phi_t^\mathsf{T}\mathbf{u})^\mathsf{T}(\mathbf{w}-\mathbf{u})\right)^2$

Challenge

Reduce computational cost without losing logarithmic regret?

References

[1] Daniele Calandriello et al. "Second-Order Kernel Online Convex Optimization with Adaptive Sketching". In: *ICML*. 2017.

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

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

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

[5] Haipeng Luo et al. "Efficient second-order online learning via sketching". In: *NIPS*. 2016.

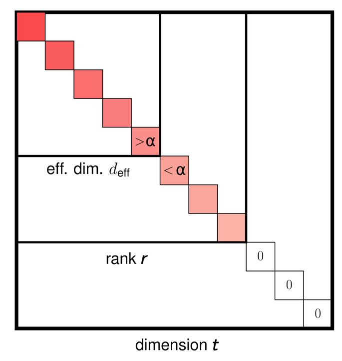
[6] Fedor Zhdanov and Yuri Kalnishkan. "An Identity for Kernel Ridge Regression". In: *Algorithmic Learning Theory*. 2010.

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

Fast rates in online kernel learning

Proposition 1: $R(\mathbf{w}) \leq \mathcal{O}\left(\sum_{t=1}^T \mathbf{g}_t^\mathsf{T}\mathbf{A}_t^{-1}\mathbf{g}_t\right) \leq \mathcal{O}\left(\sum_{t=1}^T \mathbf{g}_t^\mathsf{T}(\mathbf{G}_t\mathbf{G}_t^\mathsf{T} + \alpha\mathbf{I})^{-1}\mathbf{g}_t\right) \leq \mathcal{O}\left(L\sum_{t=1}^T \phi_t^\mathsf{T}(\mathbf{\Phi}_t\mathbf{\Phi}_t^\mathsf{T} + \alpha\mathbf{I})^{-1}\phi_t\right)$.

Definition 1. *Given a kernel matrix* $\mathbf{K}_T \in \mathbb{R}^{T\times T}$, *define*
α*-ridge leverage score:* $\tau_{T,i}(\alpha) = \mathbf{e}_{T,i}\mathbf{K}_T^\mathsf{T}(\mathbf{K}_T + \alpha\mathbf{I}_T)^{-1}\mathbf{e}_{T,i} = \phi_i^\mathsf{T}(\mathbf{\Phi}_T\mathbf{\Phi}_T^\mathsf{T} + \alpha\mathbf{I})^{-1}\phi_i$
Effective dimension: $d_{\text{eff}}(\alpha)_T = \sum_{i=1}^T \tau_{T,i}(\alpha) = \sum_{i=1}^{T} \frac{\lambda_i(\mathbf{K}_T)}{\lambda_i(\mathbf{K}_T) + \alpha} \leq \text{Rank}(\mathbf{K}_T) = r$



Proposition 2: $d_{\text{onl}}^T(\alpha) := \sum_t \phi_t^\mathsf{T}(\mathbf{\Phi}_t\mathbf{\Phi}_t^\mathsf{T} + \alpha\mathbf{I})^{-1}\phi_t \leq \log\text{Det}(\mathbf{K}_T/\alpha + \mathbf{I}) \leq 2d_{\text{eff}}^T(\alpha)\log(T/\alpha)$.

Kernel online row sampling (KORS)

A dictionary $\mathcal{I} = \{(s_i, \phi_i)\}$ is a (weighted) collection of samples.
$\mathbf{P}_\mathcal{I} = \mathbf{\Phi}_\mathcal{I}({\mathbf{\Phi}_\mathcal{I}}^\mathsf{T}\mathbf{\Phi}_\mathcal{I})^+\mathbf{\Phi}_\mathcal{I})$ is the projection on the dictionary.

$$\widetilde{\tau}_{t,i} = \frac{1+\varepsilon}{\rho\gamma}\left(k_{i,i} - \mathbf{k}_{t,i}\overline{\mathbf{S}}(\overline{\mathbf{S}}^\mathsf{T}\mathbf{K}_t\overline{\mathbf{S}} + \gamma\mathbf{I})^{-1}\overline{\mathbf{S}}^\mathsf{T}\mathbf{k}_{t,i}\right)$$

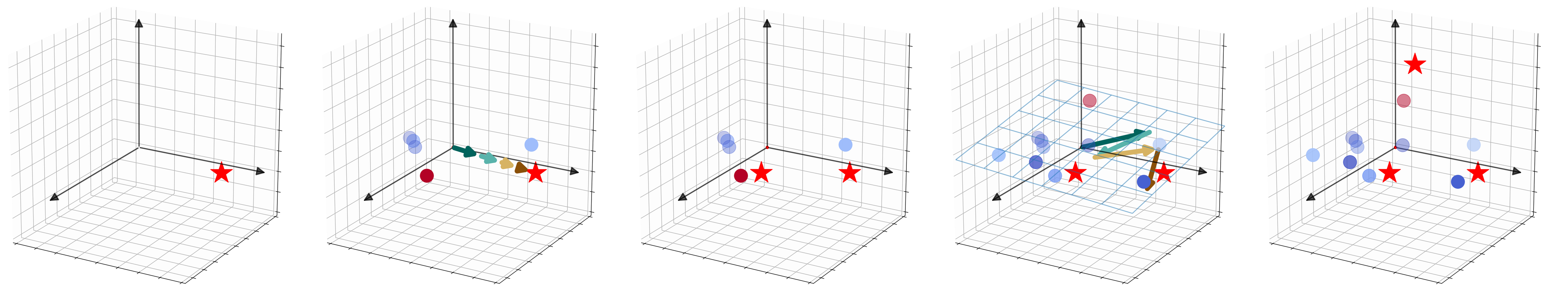
Proposition 3. *Given parameters* $0 < \varepsilon \leq 1$, $0 < \gamma$, $0 < \delta < 1$, $\rho = \frac{1+\varepsilon}{1-\varepsilon}$, *if* $\beta \geq 3\log(T/\delta)/\varepsilon^2$ *then the dictionary learned by* KORS *is such that w.p.* $1-\delta$ *and for all* $t \in [T]$, *we have*

(1) $\mathbf{0} \preceq \mathbf{\Phi}_t^\mathsf{T}(\mathbf{P}_t - \mathbf{P}_{\mathcal{I}_t})\mathbf{\Phi}_t \preceq +\frac{\varepsilon}{1-\varepsilon}\gamma\mathbf{I}_t$
(2) $J = \max_t |\mathcal{I}_t|$ *is bounded by* $\mathcal{O}(d_{\text{eff}}^T(\gamma)\log^2(T/\delta))$.

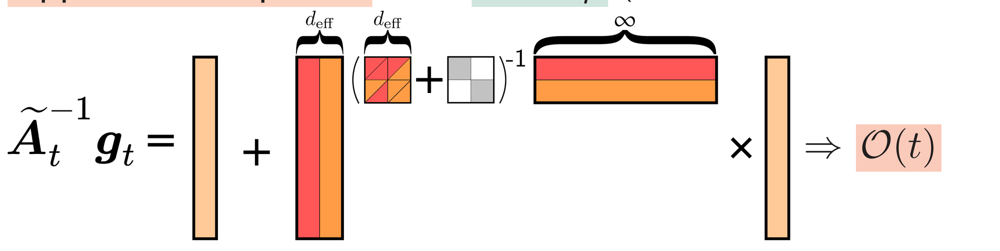
KORS *runs in* $\mathcal{O}(d_{\text{eff}}^T(\gamma)^2\log^4(T))$ *space and* $\widetilde{\mathcal{O}}(d_{\text{eff}}^T(\gamma)^3)$ *per-step time.*

Input: Regularization γ, accuracy ε, budget β
1: Initialize $\mathcal{I}_0 = \emptyset$
2: **for** $t = \{0, \ldots, T-1\}$ **do**
3: receive ϕ_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 $\overline{\mathcal{I}}_t$.
6: draw $z_t \sim \mathcal{B}(\widetilde{p}_t)$ and if $z_t = 1$, add $(1/\widetilde{p}_t, \phi_t)$ to $\mathcal{I}_t$
7: **end for**

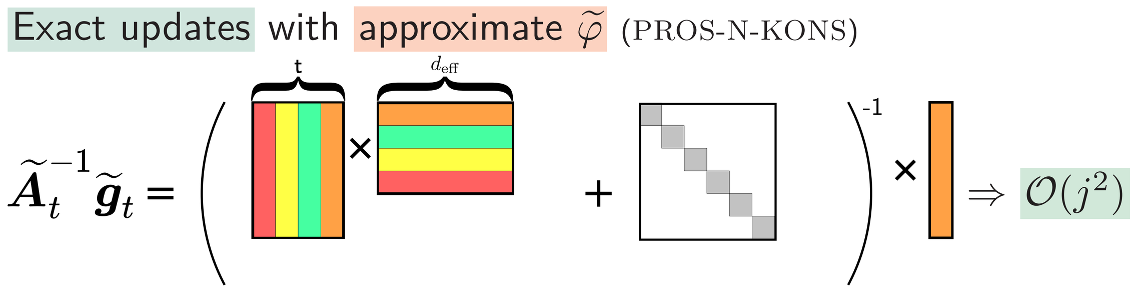
PROS-N-KONS



Approximate updates with exact φ (Luo et al. 2016; Calandriello et al. 2017)



Exact updates with approximate $\widetilde{\varphi}$ (PROS-N-KONS)



- ▶ near-linear time $\widetilde{\mathcal{O}}(Td_{\text{eff}}^T(\gamma)^2)$, near-constant space $\widetilde{\mathcal{O}}(d_{\text{eff}}^T(\gamma)^2)$
- ▶ adapt embedding using online RLS sampling
  ↳finite time guarantees, unlike approximate linear dependency
- ▶ Adversary influences steps and starting point
  ↳adaptively reset solution, keep dictionary, not too often!

Input: Feasible parameter C, step-sizes η_t, regularizer α
1: Initialize $j = 0$, $\widetilde{\mathbf{w}}_0 = \mathbf{0}$, $\widetilde{\mathbf{g}}_0 = \mathbf{0}$, $\widetilde{\mathbf{P}}_0 = \mathbf{0}$, $\widetilde{\mathbf{A}}_0 = \alpha\mathbf{I}$,
2: Start a KORS instance with an empty dictionary $\mathcal{I}_0$ and parameter γ
3: **for** $t = \{1, \ldots, T\}$ **do**
4: Receive $\mathbf{x}_t$, feed it to KORS.
5: Receive z_t (point added to dictionary or not)
6: **if** $z_{t-1} = 1$ **then** { Dictionary changed, reset.}
7: $j = j + 1$
8: Build $\mathbf{K}_j$ from $\mathcal{I}_j$ and decompose it in $\mathbf{U}_j\mathbf{\Sigma}_j\mathbf{\Sigma}_j^\mathsf{T}\mathbf{U}_j^\mathsf{T}$
9: Set $\widetilde{\mathbf{A}}_{t-1} = \alpha\mathbf{I} \in \mathbb{R}^{j\times j}$, $\widetilde{\boldsymbol{\omega}}_t = \mathbf{0} \in \mathbb{R}^j$
10: **else** {Execute a gradient-descent step.}
11: Compute map ϕ_t and approximate map $\widetilde{\phi}_t = \mathbf{\Sigma}_j^{-1}\mathbf{U}_j^\mathsf{T}\mathbf{\Phi}_j^\mathsf{T}\phi_t \in \mathbb{R}^j$.
12: Compute $\widetilde{\boldsymbol{\upsilon}}_t = \widetilde{\boldsymbol{\omega}}_{t-1} - \widetilde{\mathbf{A}}_{t-1}^{-1}\widetilde{\mathbf{g}}_{t-1}$.
13: Compute $\widetilde{\boldsymbol{\omega}}_t = \widetilde{\boldsymbol{\upsilon}}_t - \frac{\text{sign}(\widetilde{\phi}_t^\mathsf{T}\widetilde{\boldsymbol{\upsilon}}_t)\max\{|\widetilde{\phi}_t^\mathsf{T}\widetilde{\boldsymbol{\upsilon}}_t| - C, 0\}}{\widetilde{\phi}_t^\mathsf{T}\widetilde{\mathbf{A}}_{t-1}^{-1}\widetilde{\phi}_t}\widetilde{\mathbf{A}}_{t-1}^{-1}\widetilde{\phi}_t$
14: **end if**
15: Predict $\widetilde{y}_t = \widetilde{\phi}_t^\mathsf{T}\widetilde{\boldsymbol{\omega}}_t$.
16: Observe $\widetilde{\mathbf{g}}_t = \nabla_{\widetilde{\boldsymbol{\omega}}_t}\ell_t(\widetilde{\phi}_t^\mathsf{T}\widetilde{\boldsymbol{\omega}}_t) = \ell_t'(\widetilde{y}_t)\widetilde{\phi}_t$.
17: Update $\widetilde{\mathbf{A}}_t = \widetilde{\mathbf{A}}_{t-1} + \frac{\sigma_t}{2}\widetilde{\mathbf{g}}_t\widetilde{\mathbf{g}}_t^\mathsf{T}$.
18: **end for**

Theorem 1. *For any sequence of losses* ℓ_t *satisfying Asm.1-2, let* $\alpha \leq \sqrt{T}$, $\beta \geq 3\log(T/\delta)/\varepsilon^2$, *then the regret of* PROS-N-KONS *over* T *steps is bounded w.p.* $1-\delta$ *as*

$$R_T(\mathbf{w}) \leq \mathcal{O}\Big(\underbrace{J}_{\textit{restarts}}\big(\alpha\|\mathbf{w}\|^2 + \underbrace{d_{\text{eff}}^T(\alpha)\log(T/\alpha)}_{\textit{online-offline gap}}\big) + \underbrace{\gamma T}_{\mathcal{H}\text{-}\widetilde{\mathcal{H}}\textit{ gap}}/\alpha\Big),$$

where $J \leq 3\beta d_{\text{eff}}^T(\gamma)\log(2T)$ *is the number of epochs. If* $\gamma = \alpha/T$ *the previous bound reduces to*

$$R_T(\mathbf{w}) \leq \mathcal{O}(d_{\text{eff}}^T(\alpha/T)(\alpha\|\mathbf{w}\|^2\log(T) + d_{\text{eff}}^T(\alpha)\log^2(T))).$$

- ▶ If eigenvalues decay as $\lambda_t = t^{-q}$, regret is $o(d_{\text{eff}}(1/T)) \leq o(T^{1/q})$
- ▶ If eigenvalues decay as $\lambda_t = e^-t$ (Gaussian $\mathcal{H}$), regret is $o(\log(T))$
- ▶ If $\mathcal{H} = \mathbb{R}^d$ regret is $\mathcal{O}(r\log(T))$, improve over Luo et al. 2016

Theorem 2. *For any sequence* $\ell_t = (y_t - \widehat{y}_t)^2$ *of squared losses, let* $\alpha \leq \sqrt{T}$, $\gamma \leq \alpha$, $\beta \geq 3\log(T/\delta)/\varepsilon^2$, *then the regret of* PROS-N-KONS *over* T *steps is bounded w.p.* $1-\delta$ *as*

$$R_T(\mathbf{w}) \leq \mathcal{O}\left(J\left(d_{\text{eff}}^T(\alpha)\log(T) + \alpha\max_j \mathcal{L}_j^* + \alpha\|\mathbf{w}\|_2^2\right)\right)$$

where $\mathcal{L}_j^* = \min_{\mathbf{w}\in\mathcal{S}}\left(\sum_{t=t_j}^{t_{j+1}-1}(\phi_t^\mathsf{T}\mathbf{w} - y_t)^2 + \alpha\|\mathbf{w}\|_2^2\right)$ *is the best regularized cumulative loss in* $\mathcal{H}$ *within epoch* j.

- ▶ First-order regret bound, $\mathcal{L}^*$ constant if model is correct
  ↳constant $\mathcal{H}$-$\widetilde{\mathcal{H}}$ gap is enough if instantaneous loss goes to 0.
- ▶ near-linear time online Gaussian process optimization
  ↳adaptive choice of inducing points.
- ▶ Analysis can be applied to first-order methods too.

Experiments

Algorithm	parkinson $n = 5,875$, $d = 20$			cpusmall $n = 8,192$, $d = 12$		
	avg. squared loss	#SV	time	avg. squared loss	#SV	time
FOGD	**0.04909** ± 0.00020	30	—	0.02577 ± 0.00050	30	—
NOGD	**0.04896** ± 0.00068	30	—	0.02559 ± 0.00024	30	—
PROS-N-KONS	0.05798 ± 0.00136	18	5.16	0.02494 ± 0.00141	20	7.28
Con-KONS	0.05696 ± 0.00129	18	5.21	0.02269 ± 0.00164	20	7.40
B-KONS	0.05795 ± 0.00172	18	5.35	0.02496 ± 0.00177	20	7.37
BATCH	0.04535 ± 0.00002	—	—	0.01090 ± 0.00082	—	

Algorithm	cadata $n = 20,640$, $d = 8$			casp $n = 45,730$, $d = 9$		
	avg. squared loss	#SV	time	avg. squared loss	#SV	time
FOGD	0.04097 ± 0.00015	30	—	0.08021 ± 0.00031	30	—
NOGD	0.03983 ± 0.00018	30	—	0.07844 ± 0.00008	30	—
PROS-N-KONS	0.03095 ± 0.00110	20	18.59	**0.06773** ± 0.00105	21	40.73
Con-KONS	**0.02850** ± 0.00174	19	18.45	**0.06832** ± 0.00315	20	40.91
B-KONS	0.03095 ± 0.00118	19	18.65	**0.06775** ± 0.00067	21	41.13
BATCH	0.02202 ± 0.00002	—	—	0.06100 ± 0.00003	—	—

Algorithm	slice $n = 53,500$, $d = 385$			year $n = 463,715$, $d = 90$		
	avg. squared loss	#SV	time	avg. squared loss	#SV	time
FOGD	**0.00726** ± 0.00019	30	—	0.01427 ± 0.00004	30	—
NOGD	0.02636 ± 0.00460	30	—	0.01427 ± 0.00004	30	—
Dual-SGD	—	—	—	0.01440 ± 0.00000	100	—
PROS-N-KONS	did not complete	—	—	0.01450 ± 0.00014	149	884.82
Con-KONS	did not complete	—	—	0.01444 ± 0.00017	147	889.42
B-KONS	0.00913 ± 0.00045	100	60	**0.01302** ± 0.00006	100	505.36
BATCH	0.00212 ± 0.00001	—	—	0.01147 ± 0.00001	—	—

← Regression datasets | Binary classification datasets ↴

$\alpha = 1$, $\gamma = 1$

Algorithm	ijcnn1 $n = 141,691$, $d = 22$			cod-rna $n = 271,617$, $d = 8$		
	accuracy	#SV	time	accuracy	#SV	time
FOGD	9.06 ± 0.05	400	—	10.30 ± 0.10	400	—
NOGD	9.55 ± 0.01	100	—	13.80 ± 2.10	100	—
Dual-SGD	**8.35** ± 0.20	100	—	**4.83** ± 0.21	100	—
PROS-N-KONS	9.70 ± 0.01	100	211.91	13.95 ± 1.19	38	270.81
Con-KONS	9.64 ± 0.01	101	215.71	18.99 ± 9.47	38	271.85
B-KONS	9.70 ± 0.01	98	206.53	13.99 ± 1.16	38	274.94
BATCH	8.33 ± 0.03	—	—	3.781 ± 0.01	—	—

$\alpha = 0.01$, $\gamma = 0.01$

Algorithm	ijcnn1 $n = 141,691$, $d = 22$			cod-rna $n = 271,617$, $d = 8$		
	accuracy	#SV	time	accuracy	#SV	time
FOGD	9.06 ± 0.05	400	—	10.30 ± 0.10	400	—
NOGD	9.55 ± 0.01	100	—	13.80 ± 2.10	100	—
Dual-SGD	8.35 ± 0.20	100	—	4.83 ± 0.21	100	—
PROS-N-KONS	10.73 ± 0.12	436	1003.82	4.91 ± 0.04	111	459.28
Con-KONS	6.23 ± 0.18	432	987.33	5.81 ± 1.96	111	458.90
B-KONS	**4.85** ± 0.08	100	147.22	**4.57** ± 0.05	100	333.57
BATCH	5.61 ± 0.01	—	—	3.61 ± 0.01	—	—

- ▶ effective dimension empirically small $d_{\text{eff}}(1) \lesssim 4d$
- ▶ Restarts sometimes disrupt learning. Are they necessary?