Efficient second-order online kernel learning with adaptive embedding

Daniele Calandriello, Alessandro Lazaric, Michal Valko
SequeL, Inria Lille – Nord Europe

COLT 2017

Online Kernel Learning (OKL)

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 a function $f_{\mathbf{w}_t}$ 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 the associated gradient $\mathbf{g}_t$.

Online Kernel Learning (OKL)

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 a function $f_{\mathbf{w}_t}$ 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 the associated 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)

$\mathbf{g}_t = \ell_t'(\phi_t^\mathsf{T}\mathbf{w}_t)\phi_t := \dot{g}_t\phi_t$

Online Kernel Learning (OKL)

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 a function $f_{\mathbf{w}_t}$ 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 the associated 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)

$\mathbf{g}_t = \ell_t'(\phi_t^\mathsf{T}\mathbf{w}_t)\phi_t := \dot{g}_t\phi_t$

Learning to minimize regret $R(\mathbf{w}) = \sum_{t=1}^T \ell_t(\phi_t\mathbf{w}_t) - \ell_t(\phi_t\mathbf{w})$
and compete with best-in-hindsight $\mathbf{w}^* := \arg\min_{\mathbf{w}\in\mathcal{H}} \sum_{t=1}^T \ell_t(\phi_t\mathbf{w})$

Curved losses?

We assume the losses ℓ_t are scalar Lipschitz

$$|\ell_t'(z)| \leq L \text{ whenever } |z| \leq C$$

and curved

$$\ell_t(\phi_t^\mathsf{T}\mathbf{w}) \geq \ell_t(\phi_t^\mathsf{T}\mathbf{u}) + \nabla\ell_t(\phi_t^\mathsf{T}\mathbf{u})^\mathsf{T}(\mathbf{w}-\mathbf{u}) + \sigma\left(\nabla\ell_\mathbf{t}(\phi_\mathbf{t}^\mathsf{T}\mathbf{u})^\mathsf{T}(\mathbf{w}-\mathbf{u})\right)^\mathbf{2}.$$

Curved losses?

We assume the losses ℓ_t are scalar Lipschitz

$$|\ell_t'(z)| \leq L \text{ whenever } |z| \leq C$$

and curved

$$\ell_t(\phi_t^\mathsf{T}\mathbf{w}) \geq \ell_t(\phi_t^\mathsf{T}\mathbf{u}) + \nabla\ell_t(\phi_t^\mathsf{T}\mathbf{u})^\mathsf{T}(\mathbf{w}-\mathbf{u}) + \sigma\left(\nabla\ell_\mathbf{t}(\phi_\mathbf{t}^\mathsf{T}\mathbf{u})^\mathsf{T}(\mathbf{w}-\mathbf{u})\right)^2.$$

You already use curved losses

weaker than strong convexity

↳ strongly convex only along $\nabla\ell_t(\phi_t^\mathsf{T}\mathbf{u})$

satisfied by exp-concave losses

↳ squared loss, squared hinge-loss

Second-Order OKL (Kernel Online Newton Step)

Second-Order Gradient Descent

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \mathbf{A}_t^{-1}\mathbf{g}_t, \qquad \mathbf{A}_t = \sum_{s=1}^{t} \sigma \mathbf{g}_t \mathbf{g}_t^\top + \alpha \mathbf{I}$$

If $\varphi(\mathbf{x}) = \mathbf{x} : \mathbb{R}^d \to \mathbb{R}^d$ is the identity, Online Newton Step Hazan et al. 2006

↳ $\mathcal{O}(\mathbf{d^2})$ time/space per-step (Bottleneck: storing and inverting $\mathbf{X}_t^\top \mathbf{X}_t$)

$R(\mathbf{w}^*) \leq \alpha \|\mathbf{w}^* - \mathbf{w}_0\|_2^2 + \mathbf{d} \log(\mathbf{T})$

Sketched-ONS fast approximation, but only for $\varphi(\mathbf{x})$ identity and low-rank $\mathbf{X}_t^\top \mathbf{X}_t$ Luo et al. 2016

If $\varphi(\mathbf{x}) : \mathbb{R}^d \to \mathcal{H}$ is arbitrary, Kernel-ONS Calandriello et al. 2017

↳ $\mathcal{O}(\mathbf{t^2})$ time/space per-step (Bottleneck: storing and inverting $\mathbf{K}_t$)

$$R(\mathbf{w}^*) \leq \underbrace{\alpha \|\mathbf{w}^* - \mathbf{w_0}\|_2^2}_{a} + \underbrace{\mathbf{d}_{\text{eff}}{}^{T}(\alpha/(\mathbf{L}\sigma))\log(\mathbf{T})}_{b}$$

(a) start cost, (b) effective dimension (degrees of freedom) of data

Effective Dimension

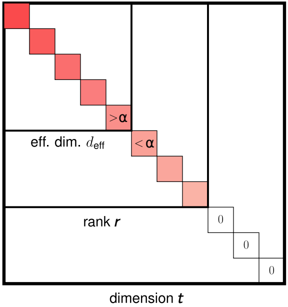
Formally $d_{\text{eff}}^T(\frac{\alpha}{L\sigma})$ is an $\frac{\alpha}{L\sigma}$ soft-thresholded version of the rank defined as

$$d_{\text{eff}}^T\left(\frac{\alpha}{L\sigma}\right) = \operatorname{Tr}\left(\mathbf{K}_T\left(\mathbf{K}_T + \frac{\alpha}{L\sigma}\mathbf{I}\right)^{-1}\right) = \sum_{t=1}^{T} \frac{\lambda_t}{\lambda_t + \frac{\alpha}{L\sigma}} \leq \operatorname{Rank}(\mathbf{K}_T) = r$$

Effective Dimension

Formally $d_{\text{eff}}^T(\frac{\alpha}{L\sigma})$ is an $\frac{\alpha}{L\sigma}$ soft-thresholded version of the rank defined as

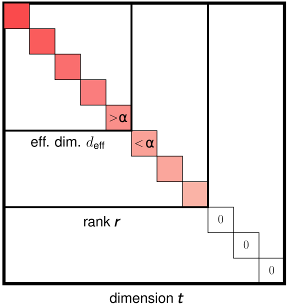
$$d_{\text{eff}}^T\left(\frac{\alpha}{L\sigma}\right) = \operatorname{Tr}\left(\mathbf{K}_T\left(\mathbf{K}_T + \frac{\alpha}{L\sigma}\mathbf{I}\right)^{-1}\right) = \sum_{t=1}^{T}\frac{\lambda_t}{\lambda_t + \frac{\alpha}{L\sigma}} \leq \operatorname{Rank}(\mathbf{K}_T) = r$$



Effective Dimension

Formally $d_{\text{eff}}^T(\frac{\alpha}{L\sigma})$ is an $\frac{\alpha}{L\sigma}$ soft-thresholded version of the rank defined as

$$d_{\text{eff}}^T\left(\frac{\alpha}{L\sigma}\right) = \operatorname{Tr}\left(\mathbf{K}_T\left(\mathbf{K}_T + \frac{\alpha}{L\sigma}\mathbf{I}\right)^{-1}\right) = \sum_{t=1}^{T}\frac{\lambda_t}{\lambda_t + \frac{\alpha}{L\sigma}} \leq \operatorname{Rank}(\mathbf{K}_T) = r$$

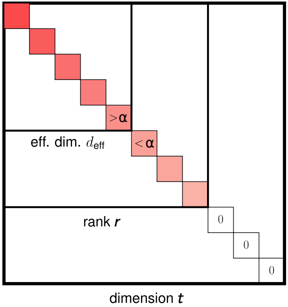


Intuitively, it quantifies the number of relevant orthogonal directions played by the adversary.

Effective Dimension

Formally $d_{\text{eff}}^T(\frac{\alpha}{L\sigma})$ is an $\frac{\alpha}{L\sigma}$ soft-thresholded version of the rank defined as

$$d_{\text{eff}}^T\left(\frac{\alpha}{L\sigma}\right) = \operatorname{Tr}\left(\mathbf{K}_T\left(\mathbf{K}_T + \frac{\alpha}{L\sigma}\mathbf{I}\right)^{-1}\right) = \sum_{t=1}^{T}\frac{\lambda_t}{\lambda_t + \frac{\alpha}{L\sigma}} \leq \operatorname{Rank}(\mathbf{K}_T) = r$$

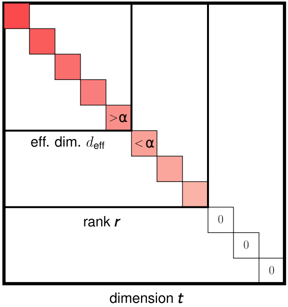


A direction (eigenvector) is relevant if its importance (eigenvalue) is larger than the Lipschitz discounted regularization $\alpha/(L\sigma)$

Effective Dimension

Formally $d_{\text{eff}}^T(\frac{\alpha}{L\sigma})$ is an $\frac{\alpha}{L\sigma}$ soft-thresholded version of the rank defined as

$$d_{\text{eff}}^T\left(\frac{\alpha}{L\sigma}\right) = \mathrm{Tr}\left(\mathbf{K}_T\left(\mathbf{K}_T + \frac{\alpha}{L\sigma}\mathbf{I}\right)^{-1}\right) = \sum_{t=1}^{T}\frac{\lambda_t}{\lambda_t + \frac{\alpha}{L\sigma}} \leq \mathrm{Rank}(\mathbf{K}_T) = r$$



Assume $\|\phi_t\| = 1$, then if all ϕ_t are orthogonal and $\alpha = \sqrt{T}$ then

$$d_{\text{eff}}^T(\sqrt{T}) \sim \sqrt{T}$$

and

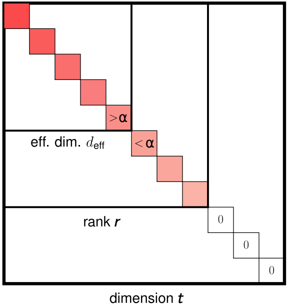
$$R(\mathbf{w}^*) \leq \sqrt{T} + d_{\text{eff}}^T(\sqrt{T})\log(T) \sim \sqrt{T}$$

recover first order bound.

Effective Dimension

Formally $d_{\text{eff}}^T(\frac{\alpha}{L\sigma})$ is an $\frac{\alpha}{L\sigma}$ soft-thresholded version of the rank defined as

$$d_{\text{eff}}^T\left(\frac{\alpha}{L\sigma}\right) = \operatorname{Tr}\left(\mathbf{K}_T\left(\mathbf{K}_T + \frac{\alpha}{L\sigma}\mathbf{I}\right)^{-1}\right) = \sum_{t=1}^{T}\frac{\lambda_t}{\lambda_t + \frac{\alpha}{L\sigma}} \leq \operatorname{Rank}(\mathbf{K}_T) = r$$



If all ϕ_t come from a bounded distribution or a finite set and $\alpha = 1$ then

$$d_{\text{eff}}^T(1) \sim \mathcal{O}(1) \leq r$$

is constant in T and

$$R(\mathbf{w}^*) \leq \mathcal{O}(1) + \mathcal{O}(1)\log(T) \sim \log T$$

logarithmic in T.

Ideal method

How to achieve adaptive $d_{\text{eff}}^T(\alpha)\log(T)$ regret
but without computational complexity depending on t?

Ideal method

How to achieve adaptive $d_{\text{eff}}^T(\alpha)\log(T)$ regret
but without computational complexity depending on t?

Use approximate second order gradient in $\mathcal{H}$ Calandriello et al. 2017

↳ $d_{\text{eff}}^T(\alpha)\log(T)$ regret, but runtime still depends on t

Ideal method

How to achieve adaptive $d_{\text{eff}}^{T}(\alpha)\log(T)$ regret
but without computational complexity depending on t?

Use approximate second order gradient in $\mathcal{H}$ Calandriello et al. 2017

↳ $d_{\text{eff}}^{T}(\alpha)\log(T)$ regret, but runtime still depends on t

Use exact second order gradient in approximate $\widetilde{\mathcal{H}}$

$$\sum_{t=1}^{T} \ell_t(\phi_t \widetilde{\mathbf{w}}_t) - \ell_t(\phi_t \mathbf{w}^*) = \sum_{t=1}^{T} \underbrace{\ell_t(\phi_t \widetilde{\mathbf{w}}_t) - \ell_t(\phi_t \overline{\mathbf{w}})}_{a} + \underbrace{\ell_t(\phi_t \overline{\mathbf{w}}) - \ell_t(\phi_t \mathbf{w}^*)}_{b}$$

Ideal method

How to achieve adaptive $d_{\text{eff}}^T(\alpha)\log(T)$ regret
but without computational complexity depending on t?

Use approximate second order gradient in $\mathcal{H}$ Calandriello et al. 2017

↳ $d_{\text{eff}}^T(\alpha)\log(T)$ regret, but runtime still depends on t

Use exact second order gradient in approximate $\widetilde{\mathcal{H}}$

$$\sum_{t=1}^{T} \ell_t(\phi_t\widetilde{\mathbf{w}}_t) - \ell_t(\phi_t\mathbf{w}^*) = \sum_{t=1}^{T} \underbrace{\ell_t(\phi_t\widetilde{\mathbf{w}}_t) - \ell_t(\phi_t\overline{\mathbf{w}})}_{a} + \underbrace{\ell_t(\phi_t\overline{\mathbf{w}}) - \ell_t(\phi_t\mathbf{w}^*)}_{b}$$

(a) error between online and batch in $\widetilde{\mathcal{H}}$:

↳ $d_{\text{eff}}^T(\alpha)\log(T)$ bound using KONS analysis

Ideal method

How to achieve adaptive $d_{\text{eff}}^{T}(\alpha)\log(T)$ regret
but without computational complexity depending on t?

Use approximate second order gradient in $\mathcal{H}$ Calandriello et al. 2017

↳ $d_{\text{eff}}^{T}(\alpha)\log(T)$ regret, but runtime still depends on t

Use exact second order gradient in approximate $\widetilde{\mathcal{H}}$

$$\sum_{t=1}^{T} \ell_t(\phi_t \widetilde{\mathbf{w}}_t) - \ell_t(\phi_t \mathbf{w}^*) = \sum_{t=1}^{T} \underbrace{\ell_t(\phi_t \widetilde{\mathbf{w}}_t) - \ell_t(\phi_t \overline{\mathbf{w}})}_{a} + \underbrace{\ell_t(\phi_t \overline{\mathbf{w}}) - \ell_t(\phi_t \mathbf{w}^*)}_{b}$$

(a) error between online and batch in $\widetilde{\mathcal{H}}$:

↳ $d_{\text{eff}}^{T}(\alpha)\log(T)$ bound using KONS analysis

(b) error between $\overline{\mathbf{w}}$ best in $\widetilde{\mathcal{H}}$ and $\mathbf{w}^*$ best in $\mathcal{H}$: <u>bound how</u>?

Subspace approximation error

$\widetilde{\mathcal{H}}$ cannot be fixed

$\hookrightarrow$ the adversary will find orthogonal points and exploit this

Subspace approximation error

$\widetilde{\mathcal{H}}$ cannot be fixed

↳ the adversary will find orthogonal points and exploit this

Use Nyström approximation instead and adapt it online

↳ $\widetilde{\mathcal{H}}_t = \mathrm{Span}(\mathcal{I}_t)$ defined using m_t inducing points $\mathcal{I}_t = \{\phi_s\}_{s=1}^{m_t}$

If the adversary plays a "sufficiently orthogonal" ϕ_t, add it to $\mathcal{I}_{t+1}$

Subspace approximation error

$\widetilde{\mathcal{H}}$ cannot be fixed

↳ the adversary will find orthogonal points and exploit this

Use Nyström approximation instead and adapt it online

↳ $\widetilde{\mathcal{H}}_t = \mathrm{Span}(\mathcal{I}_t)$ defined using m_t inducing points $\mathcal{I}_t = \{\phi_s\}_{s=1}^{m_t}$

If the adversary plays a "sufficiently orthogonal" ϕ_t, add it to $\mathcal{I}_{t+1}$

$\widetilde{\mathcal{H}}_t$ is finite dimensional: runtime independent of t

↳ Easy to embed (project) points as

$$\widetilde{\varphi}(\cdot) = \mathbf{\Sigma}^{-1}\mathbf{U}^\mathsf{T}\mathbf{\Phi}_\mathcal{I}^\mathsf{T}\varphi(\cdot) : \mathbb{R}^d \to \mathbb{R}^{m_t}$$

with $\mathbf{K}_\mathcal{I} = \mathbf{\Phi}_\mathcal{I}^\mathsf{T}\mathbf{\Phi}_\mathcal{I} = \mathbf{U}\mathbf{\Sigma}\mathbf{\Sigma}\mathbf{U}^\mathsf{T}$

$\mathcal{O}(m_t^2)$ time/space cost to run exact KONS in $\widetilde{\mathcal{H}}_t$

Subspace approximation error

"sufficiently orthogonal" is measured using γ-**ridge leverage scores**

$$\mathbb{P}(\text{include } \phi_t \text{ in } \mathcal{I}_t) \sim \phi_t^\mathsf{T}\phi_t - \phi_t^\mathsf{T}(\mathbf{\Phi}_{\mathcal{I}_{t-1}}\mathbf{\Phi}_{\mathcal{I}_{t-1}}^\mathsf{T} + \gamma\mathbf{I})^{-1}\phi_t$$

Also computable in m_t^2 time.

Subspace approximation error

"sufficiently orthogonal" is measured using γ-**ridge leverage scores**

$$\mathbb{P}(\text{include } \phi_t \text{ in } \mathcal{I}_t) \sim \phi_t^\mathsf{T}\phi_t - \phi_t^\mathsf{T}(\boldsymbol{\Phi}_{\mathcal{I}_{t-1}}\boldsymbol{\Phi}_{\mathcal{I}_{t-1}}^\mathsf{T} + \gamma\mathbf{I})^{-1}\phi_t$$

Also computable in m_t^2 time.

Guarantees that Calandriello et al. 2017

$$m_t \leq d_{\text{eff}}^t(\gamma)\log^2(T) \quad \text{(space/time)}, \quad \mathbf{K}_t - \widetilde{\mathbf{K}}_t \preceq \gamma\mathbf{I} \quad \text{(accuracy)}$$

Online/batch error

Use KONS guarantees to bound $\sum_{t=1}^{T} \ell_t(\phi_t \widetilde{\mathbf{w}}_t) - \ell_t(\phi_t \overline{\mathbf{w}})$ separately for each $\widetilde{\mathcal{H}}_t$ and associated $\overline{\mathbf{w}}_j$

$$\sum_{t=1}^{T} \ell_t(\phi_t \widetilde{\mathbf{w}}_t) - \ell_t(\phi_t \overline{\mathbf{w}}) \leq J d_{\text{eff}}^{T}(\alpha/(L\sigma)) \log(T) + \sum_{j=1}^{J} \underbrace{\alpha \|\overline{\mathbf{w}}_j - \mathbf{w}_{t_j}\|_2^2}_{\text{start costs}}$$

Online/batch error

Use KONS guarantees to bound $\sum_{t=1}^{T} \ell_t(\phi_t \widetilde{\mathbf{w}}_t) - \ell_t(\phi_t \overline{\mathbf{w}})$ separately for each $\widetilde{\mathcal{H}}_t$ and associated $\overline{\mathbf{w}}_j$

$$\sum_{t=1}^{T} \ell_t(\phi_t \widetilde{\mathbf{w}}_t) - \ell_t(\phi_t \overline{\mathbf{w}}) \leq J d_{\text{eff}}^{T}(\alpha/(L\sigma)) \log(T) + \sum_{j=1}^{J} \underbrace{\alpha \|\overline{\mathbf{w}}_j - \mathbf{w}_{t_j}\|_2^2}_{\text{start costs}}$$

Every time we change $\widetilde{\mathcal{H}}$ we pay $\alpha \|\overline{\mathbf{w}}_j - \mathbf{w}_{t_j}\|_2^2$

↳ the adversary can influence $\mathbf{w}_{t_j}$ and make it large

Online/batch error

Use KONS guarantees to bound $\sum_{t=1}^{T} \ell_t(\phi_t \widetilde{\mathbf{w}}_t) - \ell_t(\phi_t \overline{\mathbf{w}})$ separately for each $\widetilde{\mathcal{H}}_t$ and associated $\overline{\mathbf{w}}_j$

$$\sum_{t=1}^{T} \ell_t(\phi_t \widetilde{\mathbf{w}}_t) - \ell_t(\phi_t \overline{\mathbf{w}}) \leq J d_{\text{eff}}^{T}(\alpha/(L\sigma)) \log(T) + \sum_{j=1}^{J} \underbrace{\alpha \|\overline{\mathbf{w}}_j - \mathbf{w}_{t_j}\|_2^2}_{\text{start costs}}$$

Every time we change $\widetilde{\mathcal{H}}$ we pay $\alpha \|\overline{\mathbf{w}}_j - \mathbf{w}_{t_j}\|_2^2$

↳ the adversary can influence $\mathbf{w}_{t_j}$ and make it large

Reset $\widetilde{\mathbf{w}}_t$ and $\widetilde{\mathbf{A}}_t$ when $\widetilde{\mathcal{H}}_t$ changes

↳ Wasteful, but not too often. At most $J \leq d_{\text{eff}}^{T}(\gamma)$ times.

Learning is preserved through $\widetilde{\mathcal{H}}_t$ that always improves

Adaptive doubling trick

Final regret guarantees

For any curved loss

$$R(\mathbf{w}^*) \leq J\left(\alpha\|\mathbf{w}^*\|_2^2 + d_{\text{eff}}^T \log(\alpha/(L\sigma))\right) + \frac{\gamma}{\alpha}T$$

Final regret guarantees

For any curved loss

$$R(\mathbf{w}^*) \leq J\left(\alpha\|\mathbf{w}^*\|_2^2 + d_{\text{eff}}^T \log(\alpha/(L\sigma))\right) + \frac{\gamma}{\alpha}T$$

Setting $\gamma = \alpha/T$ removes second term

↳ computational cost is $\mathcal{O}(d_{\text{eff}}^T(1/T)^2)$, still small in many cases

Final regret guarantees

For any curved loss

$$R(\mathbf{w}^*) \leq J\left(\alpha\|\mathbf{w}^*\|_2^2 + d_{\text{eff}}^T \log(\alpha/(L\sigma))\right) + \frac{\gamma}{\alpha}T$$

Setting $\gamma = \alpha/T$ removes second term

↳ computational cost is $\mathcal{O}(d_{\text{eff}}^T(1/T)^2)$, still small in many cases

For squared loss only and $\gamma = \alpha$

$$R(\mathbf{w}^*) \leq J\left(\alpha\|\mathbf{w}^*\|_2^2 + d_{\text{eff}}^T \log(\alpha/(L\sigma))\right) + J\left(\sum_{t=1}^{T} \ell_t(\phi_t \mathbf{w}^*) + \alpha\|\mathbf{w}^*\|_2^2\right)$$

Last term $J(\sum_{t=1}^{T} \ell_t(\phi_t \mathbf{w}^*) + \alpha\|\mathbf{w}^*\|_2^2)$ replaces $\frac{\gamma}{\alpha}T$

↳ regularized cumulative loss of $\mathbf{w}^*$

if $\mathcal{H}$ is good, very small

Conclusions

Algorithm	cadata $n = 20k$, $d = 8$			casp $n = 45k$, $d = 9$		
	Avg. Squared Loss	#SV	Time	Avg. Squared Loss	#SV	Time
FOGD	0.04097 $\pm$ 0.00015	30	—	0.08021 $\pm$ 0.00031	30	—
NOGD	0.03983 $\pm$ 0.00018	30	—	0.07844 $\pm$ 0.00008	30	—
PROS-N-KONS	0.03095 $\pm$ 0.00110	20	18.59	**0.06773** $\pm$ 0.00105	21	40.73
CON-KONS	**0.02850** $\pm$ 0.00174	19	18.45	**0.06832** $\pm$ 0.00315	20	40.91
B-KONS	0.03095 $\pm$ 0.00118	19	18.65	**0.06775** $\pm$ 0.00067	21	41.13
BATCH	0.02202 $\pm$ 0.00002	—	—	0.06100 $\pm$ 0.00003	—	—

Algorithm	slice $n = 53k$, $d = 385$			year $n = 463k$, $d = 90$		
	Avg. Squared Loss	#SV	Time	Avg. Squared Loss	#SV	Time
FOGD	0.00726 $\pm$ 0.00019	30	—	0.01427 $\pm$ 0.00004	30	—
NOGD	0.02636 $\pm$ 0.00460	30	—	0.01427 $\pm$ 0.00004	30	—
DUAL-SGD	—	—	—	0.01440 $\pm$ 0.00000	100	—
PROS-N-KONS	did not complete	—	—	0.01450 $\pm$ 0.00014	149	884.82
CON-KONS	did not complete	—	—	0.01444 $\pm$ 0.00017	147	889.42
B-KONS	**0.00913** $\pm$ 0.00045	100	60	**0.01302** $\pm$ 0.00006	100	505.36
BATCH	0.00212 $\pm$ 0.00001	—	—	0.01147 $\pm$ 0.00001	—	—

Bibliography

Daniele Calandriello, Alessandro Lazaric, and Michal Valko. "Second-Order Kernel Online Convex Optimization with Adaptive Sketching". In: International Conference on Machine Learning. 2017.

Elad Hazan, Adam Kalai, Satyen Kale, and Amit Agarwal. "Logarithmic regret algorithms for online convex optimization". In: Conference on Learning Theory. Springer, 2006, pp. 499–513.

Haipeng Luo, Alekh Agarwal, Nicolo Cesa-Bianchi, and John Langford. "Efficient second-order online learning via sketching". In: Neural Information Processing Systems. 2016.