On the Generalization Ability of Online Learning Algorithms for Pairwise Loss Functions

Purushottam Kar*, Bharath Sriperumbudur†, Prateek Jain§ and Harish Karnick*

* Indian Institute of Technology Kanpur
† Center for Mathematical Sciences, University of Cambridge
§ Microsoft Research India

The Goal

Analyze a class of memory-efficient online learning algorithms for pairwise loss functions.

Pairwise Loss Functions ?

Example: *Metric learning for NN classification*
A metric is penalized if it brings oppositely labeled points close or sets points of same label far apart

$$\ell(\mathbf{M},(\mathbf{x},y),(\mathbf{x}',y')) = \phi\left(yy'\left(1-\mathbf{M}^2(\mathbf{x},\mathbf{x}')\right)\right)$$

ϕ is the hinge loss function.

Put simply:

- Loss functions that operate on two input points
- General form $\ell : \mathcal{H}\times\mathcal{Z}\times\mathcal{Z}\to\mathbb{R}^+$

Other examples: Bipartite ranking, AUC maximization, Multiple kernel learning

The Big Question

Can Online Learning techniques be used to obtain scalable learning algorithms for problems involving pairwise loss functions?

Challenges in Higher Order Learning

Algorithmic Challenges

- Training data often not available as i.i.d. pairs
- How to create pairs from online data ?
  - Expensive to process all $\Omega\left(n^2\right)$ data pairs

Learning Theoretic Challenges

- Training data that is used is non-i.i.d
  - Intersection in the pairs causes coupling issues
  - Decoupling method for online setting needed

Existing Work

- [1]: Online AUC maximization
  - **RS** stream sampling algorithm for pair creation
  - Regret bound does not hold
- [2]: Online-to-batch conversion bounds for online algorithms applied to pairwise problems
  - Require online algorithm to store past history
  - Bounds depend linearly on input dimension
  - Don't handle sparsity promoting regularizers

Our Contributions

Algorithmic

- A memory efficient online learning algorithm
  - Memory usage : poly-log in number of samples
  - Sublinear regret w.r.t ∞ memory algorithms
  - Novel stream sampling based pair creation

Online-to-Batch Conversion Bounds

- Applicable to bounded-memory algorithms [1]
- Tight dimension independent bounds
  - Use of Rademacher complexity as capacity
- Handle sparsity-promoting regularizers
- *Fast* rates for strongly convex loss functions

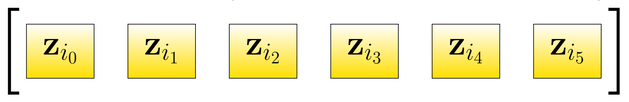
Learning Models

Unbounded Memory Model (UMM)

- At time t, posit a hypothesis $h_{t-1}\in\mathcal{H}$
- Receive a new point $\mathbf{z}_t = (\mathbf{x}_t, y_t)$
- Pair it up with previously seen points
- Incur loss $\hat{\mathcal{L}}_t(h_{t-1}) = \frac{1}{t-1}\sum_{\tau=1}^{t-1}\ell(h_{t-1},\mathbf{z}_t,\mathbf{z}_\tau)$

Bounded Memory Model (BMM)

- Buffer B of size s (updated S.O. at each step)



- At time t, pair new point with points in B
- Incur loss $\hat{\mathcal{L}}_t^{\text{buf}}(h_{t-1}) = \frac{1}{|B_t|}\sum_{\mathbf{z}\in B_t}\ell(h_{t-1},\mathbf{z}_t,\mathbf{z})$

The Desired Bounds

Generalization:

$$\frac{1}{n-1}\sum_{t=1}^{n-1}\mathcal{L}(h_t) \le \inf_{h\in\mathcal{H}}\mathcal{L}(h) + \mathfrak{e}_n$$

All-pairs Regret:

$$\frac{1}{n-1}\sum_{t=1}^{n-1}\hat{\mathcal{L}}_t(h_t) \le \inf_{h\in\mathcal{H}}\frac{1}{n-1}\sum_{t=2}^{n}\hat{\mathcal{L}}_t(h) + \mathfrak{R}_n$$

Finite-buffer Regret:

$$\frac{1}{n-1}\sum_{t=1}^{n-1}\hat{\mathcal{L}}_t^{\text{buf}}(h_t) \le \inf_{h\in\mathcal{H}}\frac{1}{n-1}\sum_{t=2}^{n}\hat{\mathcal{L}}_t^{\text{buf}}(h) + \mathfrak{R}_n^{\text{buf}}$$

Online-to-Batch Conversion Bounds

$$\sum_{t=1}^{n-1}\left(\mathcal{L}(h_t) - \hat{\mathcal{L}}_t(h_t)\right)$$

Martingale term [Azuma/Bernstein] — Residual term [Uniform Convergence]

- Problem: Classical symmetrization fails

$$\ell_h(\mathbf{z}_t,\mathbf{z}_\tau) - \ell_h(\tilde{\mathbf{z}}_t,\tilde{\mathbf{z}}_\tau) \Leftrightarrow \ell_h(\mathbf{z}_t,\tilde{\mathbf{z}}_\tau) - \ell_h(\tilde{\mathbf{z}}_t,\mathbf{z}_\tau)$$

- Solution: Symmetrization of expectations

$$\mathop{\mathbb{E}}_{\mathbf{z}}[\![\ell_h(\mathbf{z},\mathbf{z}_\tau)]\!] - \mathop{\mathbb{E}}_{\mathbf{z}}[\![\ell_h(\mathbf{z},\tilde{\mathbf{z}}_\tau)]\!]$$

- Rademacher avg. for bivariate function classes

$$\mathcal{R}_n(\mathcal{H}) \triangleq \mathop{\mathbb{E}}_{\mathbf{z},\mathbf{z}_\tau,\epsilon_\tau}\left[\!\!\left[\sup_{h\in\mathcal{H}}\frac{1}{n}\sum_{\tau=1}^{n}\epsilon_\tau h(\mathbf{z},\mathbf{z}_\tau)\right]\!\!\right] \sim \frac{C_d}{\sqrt{n}}$$

	Convex loss	Strongly Convex loss
UMM	$\mathfrak{e}_n \le \mathfrak{R}_n + \frac{C_d+\sqrt{\log n}}{\sqrt{n}}$	$\mathfrak{e}_n \le \mathfrak{R}_n + \frac{C_d^2\log^2 n}{n}$
BMM	$\mathfrak{e}_n \le \mathfrak{R}_n^{\text{buf}} + \frac{C_d+\sqrt{\log n}}{\sqrt{s}}$	$\mathfrak{e}_n \le \mathfrak{R}_n^{\text{buf}} + \frac{C_d^2\log n}{s}$

Applications

AUC Max.: maximize area under ROC curve

- $\ell(\mathbf{w},\mathbf{z},\mathbf{z}') = \phi\left((y-y')\mathbf{w}^\top(\mathbf{x}-\mathbf{x}')\right)$
- L_p regularized $\mathbf{w}$, $p>1$: $C_d = \mathcal{O}(1)$
- L_1 regularized *sparse* $\mathbf{w}$: $C_d = \mathcal{O}\left(\sqrt{\log d}\right)$

Mahalanobis metric learning

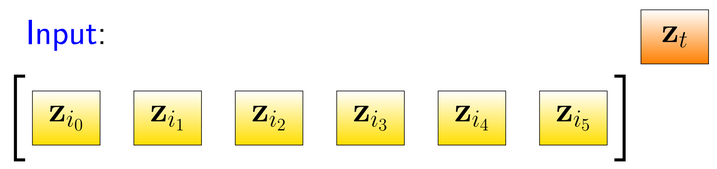
- $\ell(\mathbf{M},\mathbf{z},\mathbf{z}') = \phi\left(yy'\left(1-\mathbf{M}^2(\mathbf{x},\mathbf{x}')\right)\right)$
- Frobenius norm regularized $\mathbf{M}$: $C_d = \mathcal{O}(1)$
- Trace norm regularized $\mathbf{M}$: $C_d = \mathcal{O}\left(\sqrt{\log d}\right)$

MKL: $K_{\boldsymbol{\mu}}(\mathbf{x},\mathbf{x}') = \sum_{i=1}^{p}\boldsymbol{\mu}_i K_i(\mathbf{x},\mathbf{x}')$

- $\ell(\boldsymbol{\mu},\mathbf{z},\mathbf{z}') = \phi\left(yy'K_{\boldsymbol{\mu}}(\mathbf{x},\mathbf{x}')\right)$
- L_2 norm regularized $\boldsymbol{\mu}$: $C_d = \mathcal{O}\left(\sqrt{p}\right)$
- L_1 norm regularized $\boldsymbol{\mu}$: $C_d = \mathcal{O}\left(\sqrt{\log p}\right)$

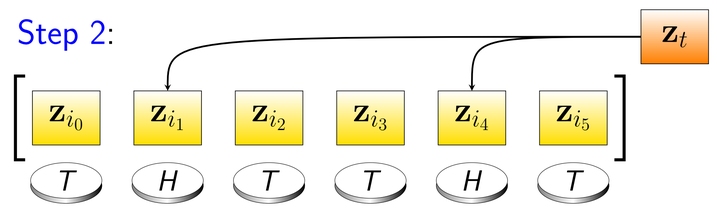
The **RS-x** Algorithm

Input:

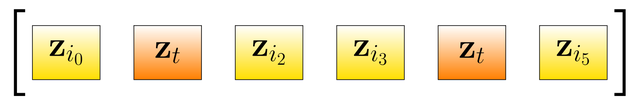


Step 1: $\sim B(1/t)$

Step 2:



Output:



Theorem: At any time step $t \ge s$, each buffer item is an i.i.d sample from the set $\{\mathbf{z}_1,\ldots,\mathbf{z}_{t-1}\}$.

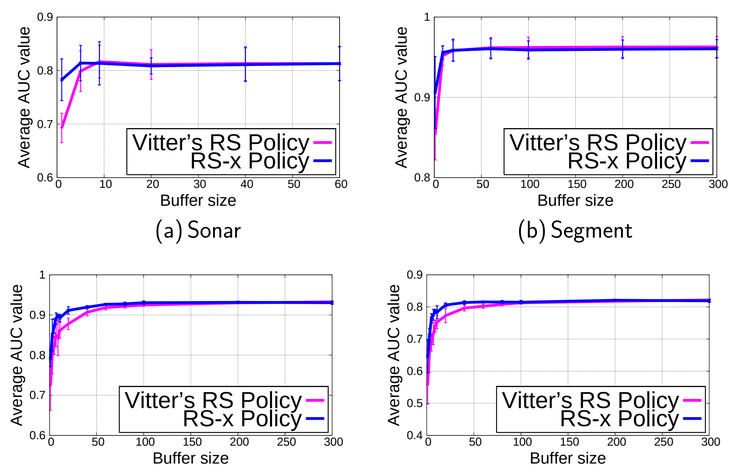
The **OLP** Algorithm

1. Start with $h_0 \leftarrow \mathbf{0}, B \leftarrow \phi$
   At each time step t
2. Obtain new point $\mathbf{z}_t$
3. $h_t \leftarrow \Pi_\Omega\left[h_{t-1} - \frac{\eta}{\sqrt{t}}\nabla_h\hat{\mathcal{L}}_t^{\text{buf}}(h_{t-1})\right]$
4. Update buffer B with $\mathbf{z}_t$ using **RS-x**
5. Return $\bar{h} = \sum_{t=1}^{n}h_{t-1}$

Theorem: **OLP** with an s-sized buffer guarantees $\mathfrak{R}_n^{\text{buf}} \le \sqrt{\frac{1}{n}}$ and w.h.p., $\mathfrak{R}_n \le C_d\sqrt{\frac{\log n}{s}}$.

Proof idea: Prove $\mathfrak{R}_n^{\text{buf}}$ bound using GIGA analysis [3]. Then prove w.h.p. $\hat{\mathcal{L}}_t(h_{t-1}) \le \hat{\mathcal{L}}_t^{\text{buf}}(h_{t-1}) + \epsilon$.

Experiments on AUC Maximization



(a) Sonar (b) Segment (c) IJCNN (d) Covertype

Fig. : Performance of **OLP** + **RS-x** and OAM_{gra}+ **RS** [1] with varying buffer sizes

References

[1] P. Zhao, S. C. H. Hoi, R. Jin, and T. Yang. Online AUC Maximization. In *ICML*, 2011.

[2] Y. Wang, R. Khardon, D. Pechyony, and R. Jones. Generalization Bounds for Online Learning Algorithms with Pairwise Loss Functions. In *COLT*, 2012.

[3] M. Zinkevich. Online Convex Programming and Generalized Infinitesimal Gradient Ascent. In *ICML*, 2003.

Full Paper : http://tinyurl.com/peozstm