

EVALUATION  
ONLINE LEARNING  
LINKS WITH OPTIMIZATION AND GAMES  
UNIVERSITÉ PARIS–SACLAY



VARIANTS OF ADAGRAD-NORM AND APPLICATION TO GAMES

Let  $d \geq 1$ ,  $\mathcal{X} \subset \mathbb{R}^d$  a nonempty closed convex set,  $\gamma, \eta, L > 0$ ,  $x_1 \in \mathcal{X}$ , and  $(u_t)_{t \geq 1}$  a sequence in  $\mathbb{R}^d$ . We consider

- *AdaGrad-Norm*, defined as

$$x_{t+1} = \Pi_{\mathcal{X}} \left( x_t + \frac{\gamma}{\sqrt{\sum_{s=1}^t \|u_s\|_2^2}} u_t \right), \quad t \geq 1,$$

with convention  $0/0 = 0$ ;

- *AdaGrad-DA-Norm*, defined as

$$x_{t+1}^{(\text{DA})} = \Pi_{\mathcal{X}} \left( \frac{\eta}{\sqrt{L^2 + \sum_{s=1}^t \|u_s\|_2^2}} \sum_{s=1}^t u_s \right), \quad t \geq 1;$$

- *AdaGrad-Hybrid-Norm*, defined as

$$x_{t+1}^{(\text{hyb})} = \Pi_{\mathcal{X}} \left( \eta_{t+1} \left( \frac{x_t^{(\text{hyb})}}{\eta_t} + u_t \right) \right), \quad t \geq 1,$$

where  $\eta_t = \eta / \sqrt{L^2 + \sum_{s=1}^{t-1} \|u_s\|_2^2}$ .

- 1) Assume that for all  $t \geq 1$ ,  $\|u_t\|_2 \leq L$ . For  $T \geq 1$  and  $x \in \mathcal{X}$ , establish an upper bound on the regret

$$\sum_{t=1}^T \langle u_t, x - x_t^{(\text{DA})} \rangle \quad \left( \text{resp.} \quad \sum_{t=1}^T \langle u_t, x - x_t^{(\text{hyb})} \rangle \right).$$

*Hint.* — For  $t \geq 1$ , consider mirror map

$$H_t(x) = \frac{\sqrt{L^2 + \sum_{s=1}^{t-1} \|u_s\|_2^2}}{2\eta} \|x\|_2^2, \quad x \in \mathbb{R}^d,$$

and associated regularizer  $h_t = H_t + I_{\mathcal{X}}$ .

- 2) Let  $m, n \geq 1$  be integers, and  $A \in \mathbb{R}^{m \times n}$ . Apply each of the above algorithms for solving the two-player zero-sum game associated with  $A$  and derive corresponding guarantees.
- 3) Perform numerical experiments in the context of solving two-player zero-sum games and compare the performance of the three above algorithms with RM, RM+, the exponential weights algorithm, as well as *optimistic* counterparts of all previous algorithms. Use the following function to compute the Euclidean projection onto the simplex.

```
def projection_simplex(y):
    n_features = y.shape[0]
    z = np.sort(y)[:, -1]
    cssv = np.cumsum(z) - 1
    ind = np.arange(n_features) + 1
    cond = z - cssv / ind > 0
    rho = ind[cond] [-1]
    theta = cssv[cond] [-1] / float(rho)
    w = np.maximum(y - theta, 0)
    return w
```

- 4) BONUS. — Also study the diagonal variants of each above algorithm and include them in the numerical experiments.

- 5) BONUS. — Also perform numerical experiments for extensive-form games using the counterfactual regret minimization approach.

