Robustify your data-driven models with `SkWDRO`

Florian Vincent[1]

Inria Grenoble & Laboratoire Jean Kuntzmann

`florian.vincent(at)inria.fr`

Keywords : *robust optimization, Scikit-Learn, PyTorch, deep learning, optimal transport*

This talk will present `SkWDRO`, a user-friendly `Python` library for distributionally robust optimization. It will focus on modeling aspects, the algorithmic approach, and some application to classical learning/optimization problems with an emphasis on the numerical difficulties that are handled by the library.

1 Distributionally Robust Optimization

In operations research and machine learning, it is fundamental to take into account data uncertainty in decision-making or prediction systems. One way to capture (random or adversarial) data perturbations is to consider *uncertainty sets* around the empirical distribution $\mathbb{P}^N$ of the dataset; this is the so-called *distributionally robust* approach. Popular uncertainty sets [3] are balls for the optimal-transport-based distance (also called *Wasserstein distance*).

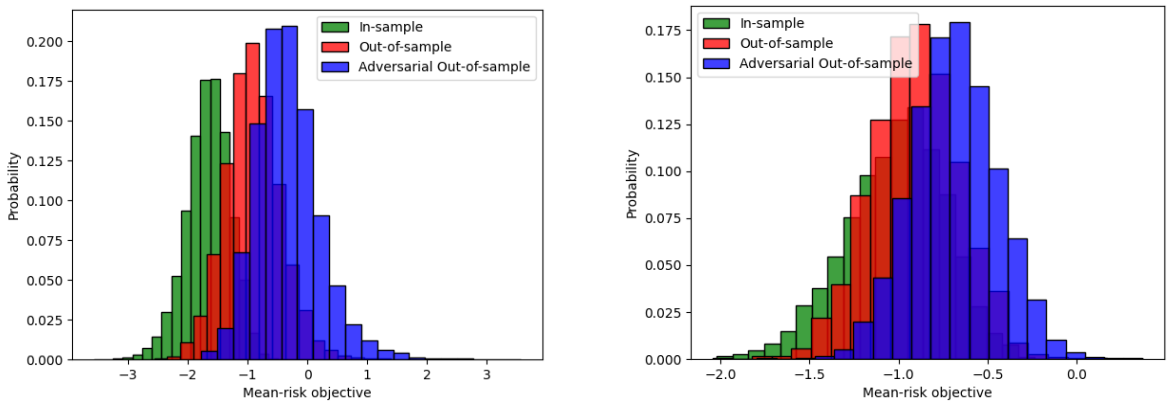

FIG. 1: Illustration on a problem of portfolio management where we plot the distributions of negative returns, with and without robustness. In green, the loss observed in the samples used to fit the decision variable; in red the loss for new samples; and in blue an adversarial perturbation of the data. On the left, the standard portfolio promises higher returns than what it provides during testing. In contrast, the robust portfolio on the right better fits its promises by displaying a closer training performance to its testing returns. Moreover the robust portfolio is able to keep most of its adversarial returns positive unlike the standard portfolio. (See the difference in the x-axis.)

Thus for a given loss function $L(\xi;\theta)$ of two parameters, ξ being the data and θ the decision variable, this approach consists in replacing the standard decision problem (called ERM for empirical risk minimization or SAA for sample average approximation) by its robust counterpart:

$$\min_\theta \frac{1}{N}\sum_{i=1}^N L(\xi_i;\theta) \qquad \text{v.s.} \qquad \min_\theta \sup_{W(\mathbb{Q},\mathbb{P}^N)\leq\rho} \mathbb{E}_{\zeta\sim\mathbb{Q}}\left[L(\zeta;\theta)\right]. \tag{1}$$

During the presentation, we shall study the applications of this framework for various problems such as those presented in [4], and further investigate deep learning models.

2 `SkWDRO`: an easy-to-use software for regularized WDRO

Formulation: In specific cases [4], one can solve (1) by reformulating it as tractable optimization problems. In general though, such a reformulation is not available, e.g. for neural networks. Our approach, following [2], consists in regularizing the objective function which leads to the smooth dual formulation:

$$\min_{\theta,\lambda\geq 0} \quad \rho\lambda + \frac{1}{N}\sum_{i=1}^{N} \varepsilon \underbrace{\ln \mathbb{E}_{\zeta\sim\mathcal{N}(\xi_i,\sigma^2 I_d)}\left[e^{\frac{1}{\varepsilon}[L(\zeta;\theta)-\lambda c(\xi_i,\zeta)]}\right]}_{=:J_{\lambda,\theta}(\xi_i)}. \tag{2}$$

The toolbox: In this talk I will present `SkWDRO`, which is a `Python` toolbox for approximating (1) by efficiently solving (2). We have taken a particular care to handle numerical issues related to the efficient computation of $J_{\theta,\lambda}$, in particular by an importance sampling technique to reduce the variance of this estimation. This enables the use of stochastic gradient techniques to minimize (2). `SkWDRO` proposes easy-to-use interfaces with popular machine learning libraries. First, an interface with `Scikit-Learn` allows to obtain robust counterparts of standard models with only one additional line of code. Second, an interface with `PyTorch` allows to wrap deep learning models inside of the dual loss of (2) so that only a couple of additional lines of code are needed to robustify a given model. Finally, `SkWDRO` proposes an automatic calibration of the hyperparameters σ and ε with heuristics from [1].

Range of applications: `SkWDRO` is general enough to allow the use of various loss functions L:

- L can be an error between a prediction and a target for supervised learning;
- L can be the negative log-likelihood/posterior of a model in an unsupervised setting;
- L can be an usual cost in operations research problems, such as the opposite of gains in the portfolio management example of Figure 1.

References

[1] Waïss Azizian, Franck Iutzeler, and Jérôme Malick. Exact Generalization Guarantees for (Regularized) Wasserstein Distributionally Robust Models. In *NeurIPS 2023*.

[2] Waïss Azizian, Franck Iutzeler, and Jérôme Malick. Regularization for wasserstein distributionally robust optimization. *ESAIM: COCV*, 2023.

[3] Daniel Kuhn, Peyman Mohajerin Esfahani, Viet Anh Nguyen, and Soroosh Shafieezadeh-Abadeh. Wasserstein distributionally robust optimization: Theory and applications in machine learning. In *Operations research & management science in the age of analytics*. Informs, 2019.

[4] Peyman Mohajerin Esfahani and Daniel Kuhn. Data-driven distributionally robust optimization using the wasserstein metric: performance guarantees and tractable reformulations. *Mathematical Programming*, 171, 2018.