skwdro: a library for Wasserstein distributionally robust machine learning

Florian Vincent, Waïss Azizian, Franck Iutzeler, Jérôme Malick

Journal of Machine Learning Research 27 (2026) 1-7 Submitted 10/24; Revised 12/25; Published 1/26

skwdro: a Library for Wasserstein Distributionally Robust Machine Learning

Florian Vincent, Waïss Azizian, Jérôme Malick
FIRSTNAME.NAME@UNIV-GRENOBLE-ALPES.FR
Univ. Grenoble Alpes, CNRS, Grenoble INP, LJK, F-38000 Grenoble, France

Franck Iutzeler FRANCK.IUTZELER@MATH.UNIV-TOULOUSE.FR
Université de Toulouse, CNRS, UPS, F-31062 Toulouse, France

Editor: Alexandre Gramfort

Abstract

We present `skwdro`, a Python library for training robust machine learning models. The library is based on distributionally robust optimization using Wasserstein distances, popular in optimal transport and machine learnings. The goal of the library is to make the training of robust models easier for a wide audience by proposing a wrapper for PyTorch modules, enabling model loss' robustification with minimal code changes. It comes along with `scikit-learn` compatible estimators for some popular objectives. The core of the implementation relies on an entropic smoothing of the original robust objective, in order to ensure maximal model flexibility. The library is available at `https://github.com/iutzeler/skwdro` and the documentation at `https://skwdro.readthedocs.io`.

Keywords: Distributionally robust optim., distribution shifts, entropic regularization

1. Introduction: ERM, WDRO, and entropic regularization

Training machine learning models typically relies on empirical risk minimization (ERM), which consists in minimizing the expectation of the loss of a model on the empirical distribution of training data. This approach has been questioned for its limited resilience and reliability, as machine learning systems have become widely used and deployed. Recently, distributionally robust optimization (DRO) has emerged as a powerful paradigm towards better generalization and better resilience against data heterogeneity and distribution shifts; see Chen et al. (2020), Blanchet et al. (2021), and Kuhn et al. (2025).

The paradigm consists in minimizing the *average loss* over the *worst distribution* in a neighborhood of the empirical distribution, thus capturing any data uncertainty. A natural way to define this neighborhood is through the Wasserstein distance (see e.g., the textbook Peyré and Cuturi (2019)), as proposed by (Esfahani and Kuhn, 2018). This framework, called Wasserstein distributionally robust optimization (WDRO), is attractive, since it combines powerful modeling capabilities, strong generalization properties, and practical robustness guarantees; see e.g., recent theoretical and practical developments (Kuhn et al., 2019; Azizian et al., 2024; Blanchet et al., 2024). Unfortunately, the resulting optimization problem is tractable only in specific cases (e.g. Esfahani and Kuhn (2018), Belbasi et al. (2023)), which limits the use of WDRO in practice for training robust models.

In this work, we tackle this computational issue and provide a complete and easy-to-use library for numerical WDRO. In Section 2, we formalize the notation and sketch our approach. We present the library `skwdro` in Section 3 and mention the numerical ingredients in Section 4. The online documentation completes this short article by presenting detailed features,tutorials, and numerical illustrations.

2. WDRO models and entropic variants

Formally, we aim at selecting a model among a family parametrized by $\theta \in \Theta$, whose error is measured by some loss function $f_\theta(\xi)$. Given n data points $(\xi_i)_{i=1}^n$ (raw data or input-label pairs), the ERM problem writes

$$\min_{\theta\in\Theta} \quad \frac{1}{n}\sum_{i=1}^n f_\theta(\xi_i) = \mathbb{E}_{\xi\sim \mathrm{P_n}}[f_\theta(\xi)] \quad \text{where the empirical distribution is } \mathrm{P_n} = \frac{1}{n}\sum_{i=1}^n \delta_{\xi_i}.$$

As mentioned in introduction, this approach can fail if the number of data points is too small or the data distribution at deployment differs from the training distribution. WDRO is a way to remedy these issues: it consists in minimizing the *worst expectation of the loss* when the probability distribution Q lives in a *Wasserstein ball* centered at the empirical distribution $\mathrm{P_n}$. Thus, for a robustness radius $\rho \geq 0$, this leads to the WDRO problem

$$\min_{\theta\in\Theta} \quad \sup_{W_c(\mathrm{P_n},\mathrm{Q})\leq\rho} \mathbb{E}_{\xi\sim\mathrm{Q}}[f_\theta(\xi)]\,, \tag{1}$$

where $W_c(\mathrm{P_n}, \mathrm{Q})$ denotes the optimal transport cost between $\mathrm{P_n}$ and Q with ground cost $c : \Xi \times \Xi \to \mathbb{R}_+$, often referred to as the Wasserstein distance by a slight abuse of terminology[1]. Applying the Lagrangian duality gives the following dual problem

$$\min_{\theta\in\Theta} \quad \min_{\lambda\geq 0} \quad \lambda\rho + \mathbb{E}_{\xi\sim\mathrm{P_n}}\left[\sup_{\zeta} \quad \{f_\theta(\zeta) - \lambda c(\xi,\zeta)\}\right] \tag{2}$$

where $\lambda \geq 0$ is the dual variable associated to the Wasserstein distance constraint. For some specific tasks, this problem can be reformulated as a convex program (Kuhn et al., 2019; Shafieezadeh-Abadeh et al., 2019). However, this convex approach does not allow for complex models, such as neural networks. Thus, though very attractive, this robust approach usually leads to an intractable optimization problem.

To remedy this issue, we consider the following smoothed counterpart of (2)

$$\min_{\theta\in\Theta} \quad \min_{\lambda\geq 0} \quad \lambda\rho + \varepsilon\ \mathbb{E}_{\xi\sim\mathrm{P_n}}\left[\log\left(\mathbb{E}_{\zeta\sim\mathcal{N}(\xi,\sigma^2 I)}\left[\exp\left(\frac{f_\theta(\zeta) - \lambda c(\xi,\zeta)}{\varepsilon}\right)\right]\right)\right] \tag{3}$$

where the regularization strength ε and the sampling variance σ^2 are the two parameters controlling the approximation. This problem interprets as the dual of the entropy regularized

1. The p-Wasserstein distance corresponds to the optimal transport cost when the ground cost is a distance between the two inputs to some power $p \geq 1$ (e.g. $c(\xi,\zeta) = \|\xi - \zeta\|^p$). Notice that our library supports any norm $\|\cdot\|$ and any power p (excepted the special case $p = \infty$), as well any user-provided ground cost.

WDRO problem (Azizian et al., 2023; Wang et al., 2025). Interestingly, such entropic-regularized problems have been proved to retain the generalization guarantees of the original robust problem (2) (see e.g., Azizian et al. (2024); Le and Malick (2025)). They have also opened up interesting convergence analysis (see the recent (Le, 2025)).

In our work, we leverage the practical interest of the approach: (3) can be solved using (stochastic) gradient-based methods, which thus significantly broadens the applicability and numerical tractability of distributionally robust machine learning based on WDRO. This is at the core of `skwdro`.

3. `skwdro`: implementation and features

`skwdro` is an open-source Python library (`https://github.com/iutzeler/skwdro`) for machine learning with WDRO based on the methodology presented in the previous section.

The library provides an easy-to-use `PyTorch` interface for making arbitrary learning models more robust: `skwdro` first formulates the robust training of the model as an optimization problem of the form (3); then the optimization follows the usual pipeline (e.g. using Adam or other stochastic gradient-based optimizers). The key numerical ingredients to handle the robust objective function are mentioned in Section 4 and are further detailed in the online documentation (`https://skwdro.readthedocs.io/`).

Furthermore, for simpler problems (e.g. linear models with $c(\xi, \zeta) = \|\xi - \zeta\|_2^2$) where (1) can be reformulated as a standard convex problem, `skwdro` also proposes a `scikit-learn` interface (Pedregosa et al., 2011), implementing known techniques (e.g. Kuhn et al. (2019)), and relying on `cvxpy` for optimization (Diamond and Boyd, 2016). However this is not the main focus of `skwdro`, and, in this case, we refer users to the library `python-dro` (Liu et al. (2025), `https://python-dro.org`) to experiment with other uncertainty neighborhoods beyond Wasserstein. To summarize, Fig. 1 gives a schematic view of `skwdro`.

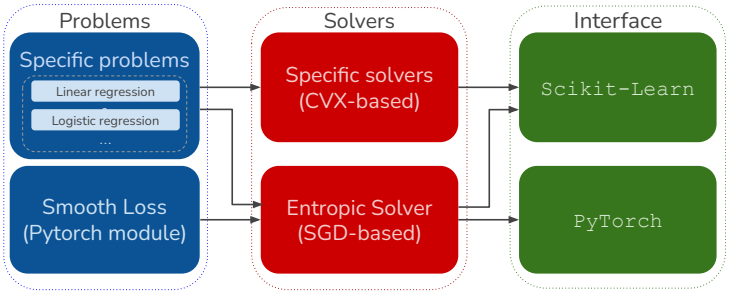

Figure 1: Schematic view of the main blocks of `skwdro`.

The `PyTorch` interface of `skwdro` allows to robustify machine learning models, specified as `PyTorch` modules, with minimal code changes. We display below a simple color-coded example: in blue, the code for learning a linear classifier with MSE loss; in green (resp. red), the lines to add (resp. remove) in order to robustify the model.

```
import torch as pt
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, TensorDataset
```

```python
from skwdro.torch import robustify
# Toy data
n_features = 3
X = pt.randn(32, n_features)
y = X @ pt.rand(n_features, 1) + 1.
train_loader = DataLoader(TensorDataset(X, y), batch_size=4)
# Define the model
model = nn.Linear(n_features, 1)
# Define the loss function
loss_fn = nn.MSELoss(reduction='none')
# Robust loss
rho = pt.tensor(.1)
robust_loss = robustify(loss_fn, model, rho, X, y)
# Define the optimizer
optimizer = optim.AdamW(model.parameters(), lr=.1)
optimizer = robust_loss.optimizer  # or use your favorite optimizer
# Training loop
for epoch in range(100):
    avg_loss = 0.
    robust_loss.get_initial_guess_at_dual(X, y)
    for batch_x, batch_y in train_loader:
        optimizer.zero_grad()
        loss = loss_fn(model(batch_x), batch_y))
        loss = robust_loss(batch_x, batch_y)
        loss.backward()
        optimizer.step()
        avg_loss += loss.detach().item()
```

4. Numerical aspects

The generic robustification of `PyTorch` models through the smoothed objective (3) presents some numerical challenges, before even optimizing the model, indeed:

- Given an implementation of the loss function $f_\theta(\xi)$, the objective (3) is not yet amenable to stochastic first-order optimization, due to the presence of a (continuous) expectation inside the logarithm. We approximate this integral by Monte-Carlo sampling, with a bound on the induced bias.
- The presence of the exponential in (3) makes the objective sharply peaked, which results in a high variance in the gradient estimate. We mitigate this with importance sampling shifting the samples following the gradient $\nabla_\xi f_\theta(\xi_i)$.
- The smoothing depends on some parameters σ, ε that have to be tuned. We provide an automatic calibration for them, based on the problem parameters and the following theoretical guidelines of Azizian et al. (2024).

Once the model loss is robustified, the user has two options: either use the new loss in their own training/optimization loop, or use the optimization procedure provided by the library (relying on adaptive optimizers (Cutkosky et al., 2023)). Numerical illustrations as well as features and implementation details are available in the online documentation at `https://skwdro.readthedocs.io`.

Acknowledgments

This work has been partially supported by MIAI@Grenoble Alpes (ANR-19-P3IA-0003)

References

Waïss Azizian, Franck Iutzeler, and Jérôme Malick. Regularization for Wasserstein distributionally robust optimization. *ESAIM: COCV*, 2023.

Waïss Azizian, Franck Iutzeler, and Jérôme Malick. Exact generalization guarantees for (regularized) wasserstein distributionally robust models. *NeurIPS*, 2024.

Reza Belbasi, Aras Selvi, and Wolfram Wiesemann. It's all in the mix: Wasserstein machine learning with mixed features. *arXiv preprint arXiv:2312.12230*, 2023.

Jose Blanchet, Karthyek Murthy, and Viet Anh Nguyen. Statistical analysis of Wasserstein distributionally robust estimators. In *Tutorials in Operations Research: Emerging Optimization Methods and Modeling Techniques with Applications*. INFORMS, 2021.

Jose Blanchet, Jiajin Li, Sirui Lin, and Xuhui Zhang. Distributionally robust optimization and robust statistics. *arXiv preprint arXiv:2401.14655*, 2024.

Ruidi Chen, Ioannis Ch Paschalidis, et al. Distributionally robust learning. *Foundations and Trends® in Optimization*, 2020.

Ashok Cutkosky, Aaron Defazio, and Harsh Mehta. Mechanic: A learning rate tuner. *NeurIPS*, 2023.

Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 2016.

Peyman Mohajerin Esfahani and Daniel Kuhn. Data-driven distributionally robust optimization using the Wasserstein metric: Performance guarantees and tractable reformulations. *Mathematical Programming*, 2018.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning*, pages 5637–5664. PMLR, 2021.

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.

Daniel Kuhn, Soroosh Shafiee, and Wolfram Wiesemann. Distributionally robust optimization. *Acta Numerica*, 34:579–804, 2025.

Tam Le. Unregularized limit of stochastic gradient method for wasserstein distributionally robust optimization. *arXiv preprint arXiv:2506.04948*, 2025.

Tam Le and Jerome Malick. Universal generalization guarantees for wasserstein distributionally robust models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=0h6v4SpLCY`.

Jiashuo Liu, Tianyu Wang, Henry Lam, Hongseok Namkoong, and Jose Blanchet. Dro: A python library for distributionally robust optimization in machine learning. *arXiv preprint arXiv:2505.23565*, 2025.

Ronak Mehta, Vincent Roulet, Krishna Pillutla, and Zaid Harchaoui. Distributionally robust optimization with bias and variance reduction. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=TTrzgEZt9s`.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 2011.

Gabriel Peyré and Marco Cuturi. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 2019.

Soroosh Shafieezadeh-Abadeh, Daniel Kuhn, and Peyman Mohajerin Esfahani. Regularization via mass transportation. *Journal of Machine Learning Research*, 2019.

Jie Wang, Rui Gao, and Yao Xie. Sinkhorn distributionally robust optimization. *Operations Research*, pages 1–23, 2025. doi: 10.1287/opre.2023.0294.

Appendix A. Illustration

In this appendix, we showcase the ability of `SkWDRO` to handle non-convex loss function f_θ, like neural networks. This is a significative feature of our library compared to existing software. We refer to the online documentation (`https://skwdro.readthedocs.io/`) for further material (discussion, comparisons, and illustrations).

We consider an image classification problem with the `iWildsCam` data set (Koh et al. (2021)), pretreated to extract a set of frozen features as per Mehta et al. (2024). This data noticeably suffers from a distribution shift. For the objective function (3), we consider a neural network with one hidden layer of 64 neurons, equipped with a `Leaky-ReLU` activation function and cross-entropy loss. The optimal transport ground cost is the squared euclidean norm on the input features, specified as `"t-NC-2-2"`. The regularization strength ε is set to 10^{-3}, and the noise level σ on the reference gaussian distribution is set to 10^{-4}.

Figure 2 reports the testing accuracy for a range of robustness radii $\rho \in \{10^{-6}, \dots, 10^{-1}\}$. We observe that, for small values of ρ, the accuracy raises at the beginning, and drops as the training procedure overfits the training set. In contrast, for higher values of ρ, the test performances are superior and do not degrade along training, better defending against the distribution shift.

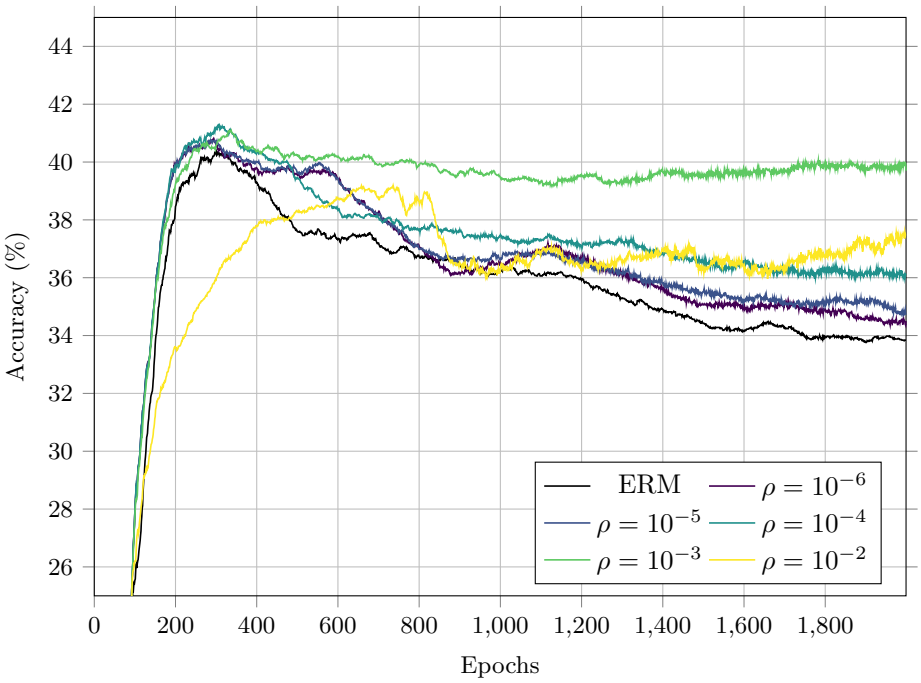

Figure 2: Evolution, over the training epochs, of the test accuracy the neural network on the data set `iWildsCam`. Colors represent different robustness radii ρ.