2020 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, SEPT. 21–24, 2020, ESPOO, FINLAND

# FIRST-ORDER OPTIMIZATION FOR SUPERQUANTILE-BASED SUPERVISED LEARNING

*Yassine Laguel*[⋆] *Jérôme Malick*[⋆] *Zaid Harchaoui*[†]

[⋆]UGA, Lab. J. Kuntzmann, Grenoble, France
[⋆]CNRS, Lab. J. Kunztmann, Grenoble, France
[†] University of Washington, Seattle, USA

## ABSTRACT

Classical supervised learning via empirical risk (or negative log-likelihood) minimization hinges upon the assumption that the testing distribution coincides with the training distribution. This assumption can be challenged in modern applications of machine learning in which learning machines may operate at prediction time with testing data whose distribution departs from the one of the training data. We revisit the superquantile regression method by proposing a first-order optimization algorithm to minimize a superquantile-based learning objective. The proposed algorithm is based on smoothing the superquantile function by infimal convolution. Promising numerical results illustrate the interest of the approach towards safer supervised learning.

***Index Terms*—** supervised learning; risk measure; distributional robustness; nonsmooth optimization

## 1. INTRODUCTION

Classical supervised learning assumes that, at training time, we have access to examples $(x_1, y_1), \ldots, (x_n, y_n)$ drawn i.i.d. from a distribution $\mathbb{P}$, and that at testing time, we may face a new example, also drawn from $\mathbb{P}$. The learned predictor or function can be used by humans or machines to make decisions, or used in as an intermediate component in a greater data processing and computing system.

This common framework is currently challenged by important domain applications [1], in which several of the standard assumptions turn out to be unrealistic or simply incorrect. We may not face the same distribution at test time as we did at training time (train-test distribution shift). Recent failures of learning systems when operating in unknown environments [2, 3] underscore the importance of reconsidering the learning objective used to train learning machines in order to ensure robust behavior in the face of unexpected distributions at prediction time.

The generalized regression framework presented in [4] provides an attractive ground to design learning machines displaying increased robustness in the face of unexpected testing distributions. The framework hinges upon the notion of superquantile, a statistical summary of a distribution tail [5, 6, 7]. This notion of robustness is aligned with the one in distributionally robust optimization [8] and empirical likelihood estimation [9]. It is, however, different, from notions of robustness commonly considered in robust statistics [8, Sec. 12.6].

The superquantile is a risk measure, a family of statistical summaries of distribution tails, well studied in economics and finance [10, 11]. The quantity is, however, a nonsmooth function. We present here a simple approach, based on infimal convolution smoothing, which allows one to easily adapt state-of-the-art gradient-based optimization algorithms for classical supervised learning to the superquantile-based learning framework. Moreover, we provide a companion software package in Python available here `https://github.com/yassine-laguel/spqr`.

### 1.1. Superquantile

Risk measures play a crucial role in optimization under uncertainty, involving problems with an aversion to worst-cases scenarios. Among popular convex risk measures, superquantile (also called Conditional Value at Risk) has received a special attention because of its nice convexity properties; see e.g. the textbook [12, Chap. 6].

We use here the notation and terminology of Rockafellar and Royset [13]. The $p$-quantile $Q_p(U)$ of a random variable $U$ is defined as the general inverse of the cumulative distribution of $U$. More precisely, for a random variable $U$ (admitting a second order moment), the cumulative distribution function $F_U \colon \mathbb{R} \to [0,1]$ is defined as $F_U(x) = \mathbb{P}(U \leq x)$. For any $p \in [0,1]$, the $p$-quantile $Q_p(U)$ and the $p$-superquantile $\bar{Q}_p(U)$, are respectively defined by

$$\begin{aligned} Q_p(U) &= \min\{x \in \mathbb{R}, F_U(x) \geq p\} \\ \bar{Q}_p(U) &= \frac{1}{1-p} \int_{p'=p}^{1} Q_{p'}(U) \mathrm{d}p'. \end{aligned} \tag{1}$$

The superquantile is, therefore, a measure of the upper tail. The parameter $p$ allows one to control the sensitivity to risk.

The authors gratefully acknowledge support from NSF CCF 1740551, DMS 1839371, and faculty research awards.

The superquantile enjoys a dual representation [14]

$$\bar{Q}_p(U) = \max_{\substack{0\leq q(\cdot)\leq \frac{1}{1-p} \\ \int_\Omega q\, d\,\mathbb{P}(\nu)=1}} \int_{\nu\in\Omega} U(\nu)q(\nu)\mathrm{d}\mathbb{P}(\nu) \qquad (2)$$

Interestingly, the dual formulation uncovers another interpretation of the superquantile learning objective relating it to the re-weighting of the terms in the empirical risk. In practice, the ambiguity on the data distribution may be formalized before training, for instance by incorporating side information (geographical and/or temporal for instance) that drives the heterogeneity of the data. Superquantile learning is expected to produce models that perform better in case of distributional shifts between the training time and the testing time, compared to models trained using standard empirical risk minimization.

### 1.2. Superquantile-based learning

We are interested in a supervised machine learning setting with training data $\mathcal{D} = (x_i, y_i)_{1\leq i\leq n} \in (\mathbb{R}^{\mathrm{p}} \times \mathbb{R}^{\mathrm{q}})^n$, a prediction function $\varphi : \mathbb{R}^{\mathrm{d}} \times \mathbb{R}^{\mathrm{p}} \to \mathbb{R}^{\mathrm{q}}$ (such as a linear model or a neural network) and a loss function $\ell : \mathbb{R}^{\mathrm{q}} \times \mathbb{R}^{\mathrm{q}} \to \mathbb{R}$ (such as the logistic loss or the least-squares loss). The classical empirical risk minimization writes

$$\min_{w\in\mathbb{R}^{\mathrm{d}}} \mathbb{E}_{(x_i,y_i)\sim\mathcal{D}} \left(\ell(y_i, \varphi(w, x_i)\right). \qquad (3)$$

A natural approach consists then in replacing the expectation in (3) by the superquantile (1) in the case of discrete distributions standing for the training data

$$\min_{w\in\mathbb{R}^{\mathrm{d}}} [\bar{Q}_p]_{(x_i,y_i)\sim\mathcal{D}}\big(\ell(y_i, \varphi(w, x_i)\big) \qquad (4)$$

Introducing $L^i(w) = \ell(y_i, \varphi(w, x_i))$ and $L(w) = (L^i(w))_i$, we simply write the superquantile optimization problem as

$$\min_{w\in\mathbb{R}^d} \quad f(w) = \bar{Q}_p(L(w)). \qquad (5)$$

Note that the objective function can specified by expressing the superquantile in its dual formulation (2) for the discrete distribution

$$f(w) = \sup_{q\in K_p} \sum_{i=1}^n q_i L^i(w) \quad \text{with}$$
$$K_p = \left\{ q \in \mathbb{R}^n, \sum_{i=1}^n q_i = 1, q_i \in \left[0, \frac{1}{n(1-p)}\right] \ \forall i \right\}.$$

This representation is central to the implementation as we shall see in Sec. 3. Existing works on minimizing superquantiles considered linear programming or convex programming including interior point algorithms; see [15]. Our approach considers first-order algorithms instead; although natural, this work seems to be the first one to do so.

## 2. SMOOTHING THE SUPERQUANTILE

In this section, we study the differentiability properties of the superquantile objective (5). We first derive the expression of the subdifferential, when the $L^i$-s are convex[1]. Then, when $L^i$ are smooth (and possibly nonconvex), we show how to smooth the superquantile by infimal convolution in order to apply gradient-based optimization algorithms [16].

### 2.1. Subdifferential expression

The superquantile risk measure (5) is usually nonsmooth and computing its subdifferential (or even a single subgradient) is not straightforward. Using the dual reformulation (2), we get the expression of the entire subdifferential for the convex case. Note that gradients of superquantile-based functions for general distributions are obtained, with advanced tools, in [17]. Interestingly, the nonsmoothness of these functions arises only with discrete distributions.

**Proposition 2.1.** *Assume the model $\varphi$ and the loss $\ell$ are such that the $L^i$ are convex. For $w \in \mathbb{R}^d$, let $I_p(w)$ be the set of indices $i \in \{1, \dots, n\}$ such that $L^i(w) = Q_p(L(w))$. Then the subdifferential reads as a Minkowski sum*

$$\partial f(w) = \frac{1}{1-p} \sum_{\substack{i\in\{1,\dots n\} \\ L^i(w)>Q_p(L(w))}} \frac{\partial L^i(w)}{n}$$
$$+ \left\{ \frac{1}{1-p} \sum_{i\in I_p(w)} \alpha_i \frac{\partial L^i(w)}{n}, \alpha_i \in [0,1]\, \forall i \in I_p(w) \right.$$
$$\left. \frac{1}{n} \sum_{i\in I_p(w)} \alpha_i = \frac{1}{n} \sum_{i=1}^n \mathbb{1}_{L^i(w)\leq Q_p(L(w))} - p \right\}$$

*In particular, when $L$ is differentiable at $w$, $f$ is differentiable at $w$ if and only if the set $I_p(w)$ is reduced to a singleton.*

*Proof.* The proof consists in applying various convex calculus rules, taken from the textbook [18, Chap D]. First we apply Theorems 4.1.1 and 4.4.2 to $h_i(w, \eta) = \max(L^i(w) - \eta)$

$$\partial h_i(x, \eta) = \{(\partial L^i(w), -1)(\mathbb{1}_{L^i(w)>\eta} + \alpha \mathbb{1}_{L^i(w)=\eta}),\ \alpha \in [0, 1]\}$$

We apply Theorem 4.1.1 with $h(w, \eta) = \eta + \frac{1}{n(1-p)} \sum_{i=1}^n h_i(w, \eta)$

$$\partial h(w, \eta) = \left\{ \left( \frac{1}{1-p} \sum_{i=1}^n \frac{\partial L^i(w)}{n} (\mathbb{1}_{L(w)_i>\eta} + \alpha_i \mathbb{1}_{L^i(w)=\eta}), \right.\right.$$
$$\left. 1 - \frac{1}{1-p} \sum_{i=1}^n \frac{1}{n} (\mathbb{1}_{L^i(w)>\eta} + \alpha_i \mathbb{1}_{L^i(w)=\eta}) \right),$$
$$\left. \alpha_i \in [0, 1], \quad \forall i \in \{1, \dots, n\} \right\}.$$

[1] Convexity of the $L^i$-s is guaranteed when e.g. the model $\varphi$ is linear and the loss $\ell$ is convex with respect to its second variable, as for the $l_2$-squared loss and the the cross-entropy loss.

By [10], $f$ satisfies $f(w) = \min_{\eta \in \mathbb{R}} h(w, \eta)$, with $Q_p(L(w)) = \arg\min_{\eta \in \mathbb{R}} h(w, \eta)$. We can thus apply Corollary 4.5.3 to get

$$\partial f(w) = \left\{ \frac{1}{1-p} \sum_{i=1}^{n} \frac{\partial L^i(w)}{n} \delta^i(w, \alpha) \quad \text{with } \alpha \text{ s.t.} \right.$$
$$\left. 0 = 1 - \frac{1}{1-p} \sum_{i=1}^{n} \frac{\delta^i(w, \alpha)}{n} \text{ and } \alpha_i \in [0,1], \forall i \right\}$$

with $\delta^i(w, \alpha) = (\mathbb{1}_{L^i(w) > Q_p(L(w))} + \alpha_i \mathbb{1}_{L^i(w) = Q_p(L(w))})$. Observe finally that for any sequence $(\alpha_i)_{1 \leq i \leq n}$

$$0 = 1 - \frac{1}{1-p} \sum_{i=1}^{n} \frac{\delta^i(w, \alpha)}{n}$$
$$\Leftrightarrow \frac{1}{n} \sum_{i \in \mathcal{I}_p(z)} \alpha_i = 1 - p - \sum_{i=1}^{n} \frac{1}{n} \mathbb{1}_{L^i(w) > Q_p(L(w))}$$
$$\Leftrightarrow \frac{1}{n} \sum_{i \in \mathcal{I}_p(z)} \alpha_i = 1 - p - (1 - \mathbb{P}[L(w) \leq Q_p(L(w))])$$
$$\Leftrightarrow \frac{1}{n} \sum_{i \in \mathcal{I}_p(z)} \alpha_i = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{L^i(w) \leq Q_p(L(w))} - p$$

which yields the result. □

Thus, the computation of a subgradient can be performed in linear time: the cost essentially stems from the computation of the quantile $Q_p(L(w))$ and the sum of vectors in $\mathbb{R}^d$ (assuming such sums can be computed in constant time).

### 2.2. Gradient of smoothed approximation

As shown in Proposition 2.1, the objective function is not differentiable in general (even when $L$ is differentiable), and we propose to smooth it using infimal convolution as in [16]. More precisely, we follow the methodology of [19] and we propose to smooth only the superquantile $\bar{Q}_p$ rather than the whole function $f$. Given formulation (2), we introduce

$$f_\mu(w) = \max_{q \in K_p} \sum_{i=1}^{n} q_i \, L^i(w) - \mu \, d(q) \qquad \text{for } \mu > 0 \quad (6)$$

where $d : \mathbb{R}^n \to \mathbb{R}$ is a fixed non-negative strongly convex function that satisfies $\min_{q \in K} d(q) = 0$. In this paper, we consider the euclidean distance to the uniform probability measure and the entropic penalty function

$$d(q) = \frac{1}{2} \left\| q - \frac{1}{n} \mathrm{e} \right\|^2 \quad \text{and} \quad d(q) = \log(n) + \sum_{i=1}^{n} q_i \log(q_i)$$

where $e = (1, \ldots, 1)^\top$ is the usual vectors of all ones. As a direct application of [16, Th. 1], we have the following proposition establishing that $f_\mu$ is a smooth approximation of $f$.

**Algorithm 1:** Fast subroutine for smoothed oracle

**Initialization:** $u = L(w) + \frac{\mu}{n} \mathrm{e}$,
$\ell = \frac{1}{n(1-p)}$, $q_\mu = 0 \in \mathbb{R}^n$
$\mathcal{P} = \{u_i, i \in \{1, \ldots, n\}\} \cup \{u_i - \mu\ell, i \in \{1, \ldots, n\}\}$
Find $a := \max\{s \in \mathcal{P}, \theta'(s) \leq 0\}$
  $b := \min\{s \in \mathcal{P}, \theta'(s) > 0\}$;
**if** $\theta'(a) = 0$ **then**
  $\lambda := a$;
**else**
  $\lambda := a - \frac{\theta'(a)(b-a)}{\theta'(b) - \theta'(a)}$
**for** $1 \leq k \leq n$ **do**
  **if** $\lambda < u_k - \mu\ell$ **then**
    $[q_\mu]_k = \ell$;
  **else if** $u_k - \mu\ell \leq \lambda < u_k$ **then**
    $[q_\mu]_k = \frac{u_k - \lambda}{\mu}$;
  **else**
    $[q_\mu]_k = 0$
  **end**
**end**
**Output:** $q_\mu \in \mathbb{R}^n$ : solution of (6)

**Proposition 2.2** (Gradient of smoothed approximation)**.** *Assume the model $\varphi$ and the loss $\ell$ are such that the $L^i$ are smooth for any $i$. In the above setting, the convex function $f_\mu$ provides a global approximation of $f$, i.e. $f_\mu(w) \leq f(w) \leq f_\mu(w) + \frac{\mu}{2}$ for any $w \in \mathbb{R}^d$. If $L$ is differentiable, then $f_\mu$ is differentiable as well, with*

$$\nabla f_\mu(w) = \mathrm{J}L(w)^T q_\mu(w), \quad (7)$$

*where $\mathrm{J}L(w)$ is the Jacobian of $L$ at $w$ and $q_\mu(w)$ is the optimal solution of* (6)*, unique by strong convexity of $d$.*

To be made practical, the previous result needs to be equipped with a fast and efficient procedure to solve (6). As stated in the next proposition, Algorithm 1 addresses this issue. The procedures follows closely the ones in [20], where convex duality and one-dimensional search ideas are fruitfully combined.

**Proposition 2.3.** *Algorithm 1 computes the optimal solution of the problem* (6) *(with the euclidean or the entropic penalty) at a cost of $\mathcal{O}(n)$ operations.*

*Proof.* We detail the proof for $d(q) = \frac{1}{2}\|q - 1/n \, \mathrm{e}\|^2$; the second case of the entropy follows the same lines. We dualize the constraint $\sum_{i=1}^n q_i - 1 = 0$ to get the Lagrangian:

$$\mathcal{L}(q, \lambda) = \sum_{i=1}^{n} q_i L^i(w) - \frac{\mu}{2} \sum_{i=1}^{n} \left( q_i - \frac{1}{n} \right)^2 + \lambda \left( 1 - \sum_{i=1}^{n} q_i \right).$$

With the notation $\ell$ and $u$ introduced in the algorithm, the dual function writes:

$$\theta(\lambda) = \max_{\substack{q \in \mathbb{R}^n \\ 0 \leq q_i \leq l}} \mathcal{L}(q, \lambda) = \lambda - \frac{\mu}{2n} + \sum_{i=1}^{n} \max_{0 \leq q_i \leq l} (u_i - \lambda) q_i - \frac{\mu}{2} q_i^2$$

For $\lambda \in \mathbb{R}$ and $i \in \{1, \dots, n\}$ fixed, let us introduce the function $h_i(q_i) = (u_i - \lambda)q_i - \frac{\mu}{2}q_i^2$. Then, we get

$$\arg\max_{0 \le q_i \le l} h_i(q_i) = \begin{cases} 0 & \text{if } \lambda \ge u_i \\ \frac{u_i - \lambda}{\mu} & \text{if } u_i \ge \lambda \ge u_i - \mu\ell \\ \ell & \text{if } \lambda \le u_i - \mu\ell \end{cases} \quad (8)$$

As a result, we get the explicit expression of $\theta(\lambda)$. Observing that it is differentiable, we get

$$\theta'(\lambda) = 1 - \sum_{i=1}^{n} \left( \frac{u_i - \lambda}{\mu} \mathbb{1}_{u_i \ge \lambda \ge u_i - \mu\ell} + \ell \mathbb{1}_{u_i - \mu\ell > \lambda} \right).$$

Observe now that $\lim_{\lambda \to +\infty} \theta'(\lambda) = 1$ and since $n\ell = \frac{1}{1-p} > 1$, $\lim_{\lambda \to -\infty} \theta'(\lambda) < 0$. Therefore, $\theta'$ is a non-decreasing and continuous (piecewise affine) function that takes negative and positive values: by the intermediate value theorem, there exists a solution $\lambda^\star \in \mathbb{R}$ such that $\theta'(\lambda^\star) = 0$. By duality theory, the associated $q^\star$ (the optimal solution of (8) for $\lambda = \lambda^\star$) is the solution of the primal problem (6). Finally, we compute $\lambda^\star$ zeroing $\theta'$. Since $\theta'$ is piecewise affine, we just need to evaluate $\theta'$ at points belonging to the set $\mathcal{P}$ and at $a$ and $b$ as defined in Algorithm 1. One can then find $\lambda^\star$ by testing three simple cases (i) if $\theta'(a) = 0$, take $\lambda^* = a$, if $\theta'(b) = 0$, take $\lambda^* = b$, else, take $\lambda^* = a - \frac{\theta'(a)(b-a)}{\theta'(b) - \theta'(a)}$.

Regarding computational costs, this algorithm boils down to the search of $a$ and $b$, and the assignment of the coordinates of $q_\mu$. This also sums up to a $\mathcal{O}(n)$ cost. □

Thus Algorithm 1 provides an efficient oracle for minimizing of $f$ with first-order algorithms.

## 3. A PYTHON TOOLBOX FOR SUPERQUANTILE OPTIMIZATION

We provide a Python software package called `SPQR` to the =community for research in superquantile-based optimization and learning. The software package includes optimization and modeling tools to solve problems of the form (5) with just a few lines of code. The implementation builds off basic structures of `scikit-learn` [21].The code is publicly available at `https://github.com/yassine-laguel/spqr`.

We describe here the optimization methods used in in the toolbox and how to call the basic functions. We refer to the online documentation for more details, custom options, and parameter settings.

### 3.1. First-order optimization algorithms

Although stochastic gradient algorithms are popular methods to solve empirical risk minimization problems at scale (3), replacing the expectation by the superquantile in (4) completely changes the situation making these algorithms not directly applicable. Indeed computing the function values and gradients requires sorting loss values on the whole data set, which is not directly amenable to classical stochastic gradient algorithms. This rehabilitates batch optimization algorithms in our context. We cover a variety of methods

- when $L$ is convex: subgradient method and dual averaging. We implement in particular the "weighted" version of dual averaging with a Euclidean prox-function [22]. For an iterate $x_k$ and a gradient $g_k$ of $f$ at $x_k$, the update writes:

$$x_{k+1} = \frac{-s_{k+1}}{\alpha_k} \text{ with } s_{k+1} = \sum_{i=0}^{k} \frac{g_k}{\|g_k\|}$$

  where $(\alpha_k)_{k \ge 0}$ denotes the tuned step-size of the method. The tuning is carried through a line-search strategy performed at the first iteration. To use these algorithms, we provide a subgradient oracle (from Proposition 2.1) with the same complexity as computing a quantile (ie. $\mathcal{O}(n)$ with $n$ the number of data points).

- when $L$ is smooth, we can use the smoothed objective: gradient method, accelerated gradient method and quasi-Newton (BFGS). In particular the accelerated gradient method relies on the following scheme [23]:

$$\alpha_0 = 0,\ \alpha_s = \frac{1 + \sqrt{1 + 4\alpha_{s-1})}}{2} \text{ and } \gamma_s = \frac{1 - \alpha_s}{\alpha_{s+1}}$$

$$x_{s+1} = y_s - \frac{1}{\beta}\nabla f(y_s),\ y_{s+1} = (1 - \gamma_s)x_{s+1} + \gamma_s x_s$$

  with $x_0 = y_0 = 0$. To use these algorithms, we provide a gradient oracle using Algorithm 1, again with a $\mathcal{O}(n)$ complexity (Proposition 2.3).

### 3.2. Basic usage: input format and execution

The user provides a dataset $(X, Y) \in \mathbb{R}^p \times \mathbb{R}^m$ and an oracle for the function $L$ and its gradient. The dataset is stored into two python lists (or numpy arrays) `X` and `Y`; for instance, for realizations of random variables:

```
import numpy as np
X = np.random.rand(100, 2)
alpha = np.array([1., 2.])
Y = np.dot(X, alpha) + np.random.rand(100)
```

The two python functions `L` and `L_prime` are assumed to be functions of the triplet `(w,x,y)` where `w` is the optimization variable and `(x,y)` a data point. For instance, one can perform risk-sensitive linear regression with:

```
# Define the loss and its derivative
def L(w,x,y):
    return 0.5 * np.linalg.norm(y -
        np.dot(x,w))**2
```

```
def L_prime(w,x,y):
    return -1.0 * (y - np.dot(x,w)) * x
```

Before solving the problem (5), we have to instantiate the `RiskOptimizer` object of `SPQR` with the two oracles, following standard usage of `scikit-learn`. The basic instantiation is as follows.

```
from SPQR import RiskOptimizer
# Instantiate a risk optimizer object
optimizer = RiskOptimizer(L, L_prime)
```

`RiskOptimizer` inherits from `scikit-learn`'s estimators: we use the `fit` method to run the optimization algorithm on the data, providing a solution of (5).

```
# Running the algorithm
optimizer.fit(X,Y)
lst_iterates = optimizer.list_iterates
sol = optimizer.solution
```

## 4. NUMERICAL ILLUSTRATIONS

We compare the proposed approach (4) with the common approach using empirical risk minimization on synthetic and real data. We solve the ordinary least squares problem

$$\min_{w\in\mathbb{R}^d} \mathbb{E}_{(x_i,y_i)\sim\mathcal{D}}\big((y_i - w^\top x_i)^2\big)$$

using the corresponding function of `scikit-learn` (by calling `LinearRegression.fit(X,Y)` method). We solve its risk-sensitive counterpart

$$\min_{w\in\mathbb{R}^d} [\bar{Q}_p]_{(x_i,y_i)\sim\mathcal{D}}\big((y_i - w^\top x_i)^2\big)$$

using our toolbox with risk-sensitive linear regression, Euclidean smoothing (with $\mu = 1000$), and L-BGFS as optimizer (see Sec. 3).

### 4.1. Synthetic Dataset

We consider a regression task on a synthetic training dataset of $n = 10^4$ points in $\mathbb{R}^{40} \times \mathbb{R}$. The design matrix $X = (x_i)_{1\leq i\leq n}$ is generated with the `make_low_rank_matrix` procedure of `scikit_learn` [21] with a rank 30. For a given model parameter $\bar{w} \in \mathbb{R}$, we generate the data according to

$$y_i = x_i^\top \bar{w} + \varepsilon_i.$$

The noise $\varepsilon_i$ is defined here as a mixture

$$\varepsilon_i = \beta\varepsilon_{\mathcal{N}} + (1-\beta)\varepsilon_{\mathcal{L}}$$

where all random variables are independent, $\varepsilon_{\mathcal{N}}$ follows a standard normal distribution, $\varepsilon_{\mathcal{L}}$ follows a Laplace distribution with location $\mu = 10$ and scale $s = 1$, and $\beta$ follows a Bernoulli distribution with parameter $p = 0.8$. Define the *squared residuals* (or losses)

$$r_i^2 = (y_i - w^\top x_i)^2 \quad \text{for } i = 1,\dots,n$$

and the $p$-quantiles of the empirical distribution of $(r_i^2)_{i=1,\dots,n}$ for $p = 0.5$ and $p = 0.9$.

| Model | Mean | $p$-quantile of the loss | |
|---|---|---|---|
| | | $p = 0.5$ | $p = 0.9$ |
| $\mathbb{E}$ | 16.45 | 5.55 | 60.2 |
| $\bar{Q}_p \ - \ p = 0.5$ | 18.75 | 13.9 | 41.2 |
| $\bar{Q}_p \ - \ p = 0.7$ | 22.3 | 20.7 | 36.6 |
| $\bar{Q}_p \ - \ p = 0.9$ | 23.7 | 22.5 | 37.7 |

**Table 1**. Quantiles of the empirical distribution of residuals on the test.

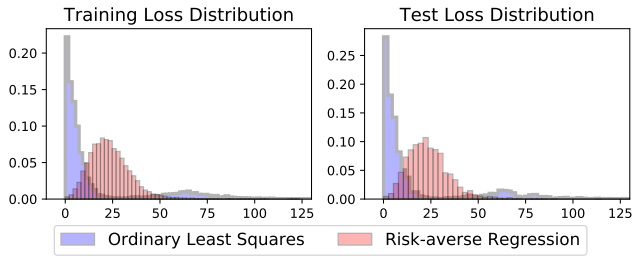


**Fig. 1**. Quantiles of the empirical distribution of residuals on the test. The risk-sensitive model was trained with $p = 0.9$.

We report the $p$-quantiles and the distribution of losses obtained on the training dataset and on a test dataset of 2000 data points independently generated with the same procedure; see Table 1 and Figure 1. As $p$ grows, the superquantile-based or risk-sensitive model shifts the upper tail on errors to the left, which shows an improved performance on extreme cases. This comes with the price of lower performances on inputs well managed by the standard approach (see metrics for $p = 0.5$ in Table 1).

### 4.2. Real Dataset

We consider the superconductivity dataset [24] which contains the information of $21,263$ superconductors. The learning task is to predict the critical temperature of a superconductor from the $10$ most important features as selected by [24]. We split the dataset into a training set and a testing set with a ratio $80\%/20\%$.

We report in Figure 2 the comparison between the quantiles of the testing and training loss distribution respectively. In terms of the quantile at $90\%$, the proposed approach display better statistical behavior on the testing loss than the common approach based on empirical risk minimization. This is in line with the aim of the formulation considered, which seeks to gain a better control on the tails of the loss distribution.

| Model | Mean | $p$-quantile of the loss | | |
|---|---|---|---|---|
| | | $p = 0.9$ | $p = 0.95$ | $p = 0.99$ |
| $\mathbb{E}$ | 16.5 | 35.8 | 42.7 | 55.7 |
| $\bar{Q}_p \ - \ p = 0.8$ | 17.4 | 34.7 | 41.0 | 53.8 |
| $\bar{Q}_p \ - \ p = 0.9$ | 18.1 | 35.6 | 41.0 | 53.6 |
| $\bar{Q}_p \ - \ p = 0.95$ | 18.9 | 36.5 | 41.4 | 53.6 |

**Table 2**. Metrics of the distribution of the loss values $r_i$ on the test superconductivity dataset

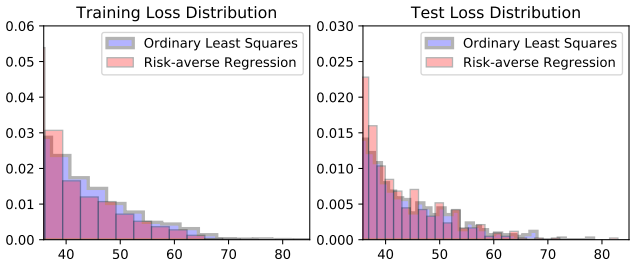


**Fig. 2**. Distribution of the loss values $r_i$ on the train and test superconductivity dataset. The risk-sensitive model is trained with $p = 0.9$.

## 5. CONCLUSION

Risk-sensitive optimization plays a major role in the design of safer models for decision-making and has recently gained interest in machine learning. We provide a toolbox to tackle superquantile-based learning problems using first-order optimization algorithms. Numerical illustrations on regression tasks show an improved statistical behavior in terms of higher quantiles of the testing loss.

## 6. REFERENCES


[1] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar, "Do imagenet classifiers generalize to imagenet?," *arXiv:1902.10811*, 2019.

[2] Rachel Metz, "Microsoft's neo-Nazi sexbot was a great lesson for makers of AI assistants," *Artificial Intelligence*, March 2018.

[3] Will Knight, "A self-driving Uber has killed a pedestrian in Arizona," *Ethical Tech*, March 2018.

[4] R.T. Rockafellar, S. Uryasev, and M. Zabarankin, "Risk tuning with generalized linear regression," *Mathematics of Operations Research*, 2008.

[5] J. Lee and M. Raginsky, "Minimax statistical learning with Wasserstein distances," in *Advances in Neural Information Processing Systems*, 2018.

[6] J. C. Duchi and H. Namkoong, "Variance-based Regularization with Convex Objectives.," *Journal of Machine Learning Research*, 2019.

[7] D. Kuhn, P.M. Esfahani, V. Anh Nguyen, and S. Shafieezadeh-Abadeh, "Wasserstein distributionally robust optimization: Theory and applications in machine learning," in *Operations Research & Management Science in the Age of Analytics*. INFORMS, 2019.

[8] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski, *Robust optimization*, Princeton University Press, 2009.

[9] A.B. Owen, *Empirical Likelihood*, Chapman & Hall/CRC Monographs on Statistics & Applied Probability. CRC Press, 2001.

[10] T. Rockafellar and S. Uryasev, "Optimization of Conditional Value-at-Risk," *Journal of Risk*, 2000.

[11] A. Ben-Tal and M. Teboulle, "An old-new concept of convex risk measures: The optimized certainty equivalent," *Mathematical Finance*, 2007.

[12] A. Shapiro, D. Dentcheva, and A. Ruszczyński, *Lectures on stochastic programming: modeling and theory*, SIAM, 2014.

[13] R. T. Rockafellar and J. O Royset, "Superquantiles and their applications to risk, random variables, and regression," in *Theory Driven by Influential Applications*. INFORMS, 2013.

[14] H. Föllmer and A. Schied, "Convex measures of risk and trading constraints," *Finance and stochastics*, 2002.

[15] R.T. Rockafellar, J.O. Royset, and S.I. Miranda, "Superquantile regression with applications to buffered reliability, uncertainty quantification, and conditional value-at-risk," *European Journal of Operational Research*, 2014.

[16] Y. Nesterov, "Smooth minimization of non-smooth functions," *Mathematical programming*, 2005.

[17] A. Ruszczyński and A. Shapiro, "Optimization of convex risk functions," *Mathematics of operations research*, 2006.

[18] J.-B. Hiriart-Urruty and C. Lemaréchal, *Convex analysis and minimization algorithms I: Fundamentals*, Springer science & business media, 2013.

[19] A. Beck and M. Teboulle, "Smoothing and first order methods: A unified framework," *SIAM Journal on Optimization*, 2012.

[20] L. Condat, "Fast projection onto the simplex and the l1 ball," *Mathematical Programming*, 2016.

[21] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, 2011.

[22] Y. Nesterov, "Primal-dual subgradient methods for convex problems," *Mathematical programming*, 2009.

[23] Y. Nesterov, "A method for solving the convex programming problem with convergence rate $\mathcal{O}(1/k^2)$," *Dokl. Akad. Nauk SSSR*, 1983.

[24] K. Hamidieh, "A data-driven statistical model for predicting the critical temperature of a superconductor," *Computational Materials Science*, 2018.