Noname manuscript No.
(will be inserted by the editor)

Superquantile-based learning: a direct approach using gradient-based optimization

Yassine Laguel · Jérôme Malick · Zaid Harchaoui

the date of receipt and acceptance should be inserted later

Abstract We consider a formulation of supervised learning that endows models with robustness to distributional shifts from training to testing. The formulation hinges upon the superquantile risk measure, also known as the conditional value-at-risk, which has shown promise in recent applications of machine learning and signal processing. We show that, thanks to a direct smoothing of the superquantile function, a superquantile-based learning objective is amenable to gradient-based optimization, using batch optimization algorithms such as gradient descent or quasi-Newton algorithms, or using stochastic optimization algorithms such as stochastic gradient algorithms. A companion software `SPQR` implements in Python the algorithms described and allows practitioners to experiment with superquantile-based supervised learning.

Keywords machine learning · risk measure · distributional robustness · nonsmooth optimization

1 Introduction: Superquantile comes into play

Classical supervised learning via empirical risk (or negative log-likelihood) minimization relies on the assumption that the testing distribution coincides

A preliminary version of this work [18] was presented at the IEEE MLSP conference in September 2020. This work is based on Y. Laguel's MSc. thesis defended in Summer 2018.

Yassine Laguel
Univ. Grenoble Alpes, Grenoble INP, LJK, 38000 Grenoble, France
E-mail: yassine.laguel@univ-grenoble-alpes.fr

J. Malick
Univ. Grenoble Alpes, CNRS, Grenoble INP, LJK, 38000 Grenoble, France
E-mail: jerome.malick@univ-grenoble-alpes.fr

Z. Harchaoui
University of Washington, Seattle, USA
E-mail: zaid@uw.edu

with the training distribution. This assumption can be challenged in domain applications of machine learning such as visual systems or dialog systems [30]. Learning machines may then operate at prediction time with testing data whose distribution departs from the one of the training data. Recent failures of learning systems when operating in unknown environments [24,16] underscore the importance of reconsidering the learning objective used to train learning machines in order to ensure robust behavior in the face of prevalence of worst-case scenarios or unexpected distributions at prediction time.

The generalized regression framework presented in [32] provides an attractive ground to design learning machines displaying increased robustness. This framework hinges upon modeling worst-case aversion with superquantile, also known as Conditional Value-at-Risk, a statistical summary of the tail of the distribution considered [21,11,17]. The superquantile stands out as one of prominent examples of risk measures, well-studied in economics and finance [35,3]. The superquantile has recently drawn an increasing attention in machine learning; see e.g. fair learning [40], federated learning [19], adversarial classification [14], submodular optimization [39], and reinforcement learning [7] among others.

The notion of robustness brought by the superquantile is aligned with the one in distributionally robust optimization [2] and empirical likelihood estimation [28]. It is, however, different, from notions of robustness commonly considered in robust statistics [2, Sec. 12.6]. The superquantile provides an efficient and mathematical-grounded adaptive re-weighting scheme of the training data, allowing one to learn predictive models with better worst-case performances that standard models obtained from empirical risk minimization. This has been corroborated empirically by a number of recent papers; see e.g. [40,19, 22,9,38]. Recent work [10] established learning-theoretic generalization bounds for statistical models trained through the minimization of related objectives.

Despite attractive theoretical and practical properties, superquantile-based learning may be less developed than it could have been in machine learning and signal processing. This may be due to the lack of (i) direct scalable algorithms for superquantile-based optimization and (ii) easy-to-use software packages to benchmark superquantile optimization algorithms.

Contributions of this work. In this paper, we present a publicly-available and easy-to-use Python toolbox for superquantile-based learning, building off the popular software library `scikit-learn`. This paper is a follow-up of our IEEE MLSP 2020 conference paper [18], incorporating recent work in an extended literature review, providing additional features to the toolbox, and presenting further empirical illustrations of the robustness brought by superquantile.

More precisely, the contributions of this work are the following:

- We provide a gentle introduction to superquantile-based learning. We present the main notions; we review several choices of optimization algorithms; we also discuss the various numerical components used explicitly or implicitly in recent papers. These components include for instance various strategies to overcome the non-smoothness inherent to the superquantile function.

- We provide elementary analyses as well as template routines within a companion software package. We primarily focus on operational aspects and give pointers to recent theoretical developments.
- We provide numerical experiments illustrating (i) the interest of using batch quasi-Newton optimization algorithms for minimizing superquantile-based objectives and (ii) the robustness of superquantile-based models compared to the standard models obtained from empirical risk minimization.

Outline of the paper. The outline of the paper is as follows. We set the stage by formalizing, in Section 2, the framework of superquantile-based supervised learning, highlighting the three classical formulations of superquantile-based objectives. In Section 3, we study the differentiability of these objective functions, provide practical expressions of their (sub)gradients, together with fast procedures to compute them. In Section 4, we overview batch and mini-batch first-order methods using these fast oracles. In Section 5, we provide a short presentation of the toolbox `SPQR` for superquantile-based learning. Finally, we illustrate in Section 6 the interests of superquantile and `SPQR` for robustness in standard regression/classification tasks.

Most important related work. The introduction has already mentioned a variety of works related to superquantile, robustness, and applications in machine learning and signal processing. Finally, we highlight here the most important articles, in view of the contributions of this work, regarding the algorithms for superquantile optimization and the interest of superquantile in learning.

Classical approaches for superquantile-based optimization consider convex programming techniques, including interior point algorithms; see the review of [31]. The use of first-order algorithms in this context is quite recent and seems to be driven by machine learning considerations. A key reference for our work is [22] which introduces an efficient approximated stochastic gradient algorithm for superquantile-based learning. We have implemented this algorithm within our toolbox and compared it with a simple approach using batch quasi-Newton method (in Section 6.1).

The interest of using superquantile in learning has been shown empirically in several recent papers, including [40,19,22,9,38]. In particular [40], studying fairness issues, empirically demonstrates that superquantile trades predictive accuracy for less fairness violation. In a context of federated learning, [19] compares the performances of models learned by superquantile-based learning to standard models: for heterogeneous data, significant improvements on worst cases are reported for both error testing and accuracy on classification tasks. In our numerical experiments, we use similar representations to visualize the impact of the superquantile. Our experimental results align with those of [9], where the robustness of superquantile models on distributionally shifted datasets is demonstrated.

2 Superquantile-based learning framework

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 an additive 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). Denoting $w \in \mathbb{R}^{\mathrm{d}}$ the parameter ("weights") to be optimized, the classical empirical risk minimization (ERM) problem reads

$$\min_{w \in \mathbb{R}^{\mathrm{d}}} \quad \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \varphi(w, x_i) = \mathbb{E}_{(x,y)\sim\mathcal{D}} \left(\ell(y, \varphi(w, x)) \right), \tag{1}$$

In the above expression as expectation, we identify, by abuse of notation, the training data $\mathcal{D}$ with the empirical measure of the training data. With this ERM problem, we aim at achieving a small loss with an equal weighting across all training data-points. In the event that, at testing time, some probability mass gets shifted from a fraction of them onto another, large losses may then be incurred.

In order to be robust against such uncertainty in the way probability mass will spread at testing time, we can consider, instead, a training objective that involves a minimization problems with respect to a pessimistic re-weighting of the training datapoints. This boils down to replacing the expectation in (1) by a tail-sensitive or *risk-sensitive* quantity. Risk-sensitive measures play a crucial role in optimization under uncertainty. Among popular convex risk measures, the superquantile, also called Conditional Value at Risk, has received particular attention because of its nice convexity properties; we refer to the seminal work [35] and the classical textbook [37, Chap. 6].

We use here the notation and terminology of [34]. Consider a probability space Ω, with probability denoted $\mathbb{P}$. For any $p \in [0, 1]$, the p-quantile of a random variable $U \colon \Omega \to \mathbb{R}$, denoted by $Q_p(U)$, is the inverse of the cumulative distribution function of U: for all $t \in \mathbb{R}$ we have

$$Q_p(U) \leq t \iff \mathbb{P}(U \leq t) \geq p \,. \tag{2}$$

The p-superquantile of U is then defined as the mean of values of quantiles greater than a threshold p

$$\bar{Q}_p(U) = \frac{1}{1-p} \int_{s=p}^{1} Q_s(U) \mathrm{d}s \,. \tag{3}$$

The analogue to (2) for the superquantile is stronger:

$$\bar{Q}_p(U) \leq t \iff U \text{ is lower than } t \text{ on average in its } p\text{-tail.}$$

The superquantile can be therefore interpreted as a measure of the upper tail of the distribution of U with the parameter p controlling the sensitivity to high losses.

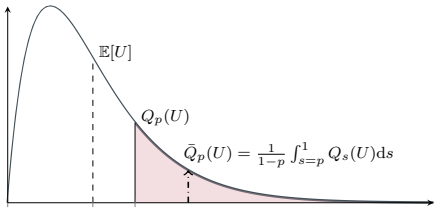

Fig. 1: Illustration of the expectation $\mathbb{E}(U)$, the p-quantile $Q_p(U)$, and the $(1-p)$-superquantile $\bar{Q}_p(U)$ of a random variable U.

In the case where the random variable U takes equi-probable realizations $u_1, \ldots, u_n$, the integral (3) reduces to an average of the u_i that are greater or equal than the quantile. This sum can be further split in two parts with the u_i that are equal to the quantile and those are that strictly larger (indexed by $I_>$). Mathematically, this writes

$$\bar{Q}_p(U) = \frac{1}{n(1-p)} \sum_{i \in I_>} u_i + \frac{\delta}{1-p} Q_p(U) \qquad \text{with } I_> = \{i : u_i > Q_p(U)\}. \tag{4}$$

where $\delta = F_U(Q_p(U)) - p = \frac{1}{n}(n - |I_>|) - p$. This expression involves the distance from p to the next discontinuity point of the quantile function. Thus, (4) provides a direct way to compute the superquantile from the computation of the quantile.

Going back to the context of learning described at the beginning of this section, we consider the superquantile of discrete distributions standing for the training data, that we denote by $[\bar{Q}_p]_{(x,y)\sim\mathcal{D}}$. A risk-sensitive statistical learning framework using the superquantile of losses rather than the expected loss thus formally replaces in (1) the expectation by the superquantile

$$\min_{w \in \mathbb{R}^{\mathrm{d}}} \quad f(w) = [\bar{Q}_p]_{(x,y)\sim\mathcal{D}}\big(\ell(y, \varphi(w, x))\big). \tag{5}$$

This superquantile-based objective function has some special properties. First is has a nice variational formulation [35]:

$$f(w) = \min_{\eta \in \mathbb{R}} \left\{ \eta + \frac{1}{n(1-p)} \sum_{i=1}^{n} \max\{\ell(y_i, \varphi(w, x_i)) - \eta, 0\} \right\}. \tag{6}$$

This formulation opens the way to treating (5) as a joint minimization over (w, η); this is discussed in Section 4. Note here that the minimization with respect to η in (6) exactly gives the p-quantile of the losses and can be done efficiently in linear time.

Using standard duality, we can also write the min problem (6) as a max, which takes the form

$$f(w) = \max_{q \in \Delta_n} \left\{ \sum_{i=1}^{n} q_i \, \ell(y_i, \varphi(w, x_i)) : 0 \le q_i \le \frac{1}{n(1-p)} \right\} \tag{7}$$

where Δ_n denotes the probability simplex $\Delta_n = \{q \in (\mathbb{R}_+)^n, \sum_{i=1}^n q_i = 1\}$ Interestingly, this third formulation uncovers another interpretation of the superquantile objective. The set of admissible probability q_i in (7) acts as a so-called ambiguity set around the uniform probability distribution $(\frac{1}{n}, \dots, \frac{1}{n})$, relating (5) to an instance of distributionally robust optimization: (7) considers the worst possible combination among possible re-weightings of the individual losses, with the probability distributions in this ambiguity set. Superquantile-based learning is then expected to produce models that perform better in case of small distributional re-weighting between the training time and the testing time, compared to models trained using standard empirical risk minimization.

The three above formulations (5) (6) and (7) of the superquantile-based objective reveal an inherent non-smoothness. We discuss in the next section how to obtain first-order information from a superquantile-based criterion. Note, though, that training with such loss is not straightforward: replacing the expectation by the superquantile in (5) completely changes the situation, making stochastic gradient algorithms, popular methods for solving (1), which are somewhat flexible to the smoothness properties of the objective, not directly applicable; we will come back to this in Section 4.

Let us finally mention that the probability threshold p should be considered as an hyperparameter of the superquantile-based learning problem (5). The standard way to set p is then to perform a cross validation over a grid of values and chose the best one with respect to a risk sensitive metric, such as e.g. the 90^{th} percentile of the validation loss.

We finish this section by illustrating on a toy problem that superquantile-based learning allows one, as expected, to learn models with better worst-case performance.

Example 1 (Superquantile-based learning gives better worst-case performance) We consider a linear regression task on a synthetic training dataset[1] to provide a striking illustration of the benefit of superquantile-based learning in terms of worst-case performance. For a given model parameter $\bar{w}$, we generate the data according to

$$y_i = x_i^\top \bar{w} + \varepsilon_i \qquad \text{with } \varepsilon_i = \beta\varepsilon_{\mathcal{N}} + (1-\beta)\varepsilon_{\mathcal{L}}.$$

The noise ε_i is generated from a mixture of two distributions: $\varepsilon_{\mathcal{N}}$ follows a standard normal distribution, $\varepsilon_{\mathcal{L}}$ follows a Laplace distribution with location $\mu = 10$ and scale $s = 1$, and β follows a Bernoulli distribution with parameter 0.8. We solve the ordinary ℓ_2^2-regularized least squares problem and its superquantile counterpart:

$$\min_{w\in\mathbb{R}^d} \mathbb{E}_{(x,y)\sim\mathcal{D}}\big((y - w^\top x)^2\big) \quad \text{vs.} \quad \min_{w\in\mathbb{R}^d} [\bar{Q}_p]_{(x,y)\sim\mathcal{D}}\big((y - w^\top x)^2\big).$$

Figure 2 reports the distribution of losses obtained on the training dataset and on a test dataset of 2000 data points independently generated with the same

[1] We take $n = 10^4$ points in $\mathbb{R}^{40} \times \mathbb{R}$. The design matrix $X = (x_i)_{1\le i\le n}$ is generated with the `make_low_rank_matrix` procedure of *scikit_learn* [29] with a rank 30.

procedure. Thanks to the superquantile-based learning, the upper tail of the error is shift to the left of the plot, which in other words means an improved performance in extreme cases. □

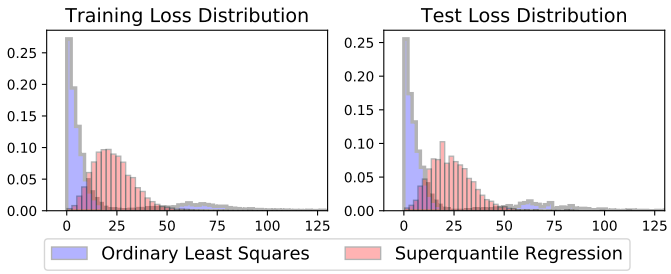

Fig. 2: Illustration of the reshaping of the distribution of errors resulting from superquantile-based learning (model trained with $p = 0.9$).

3 First-order oracles for superquantile function

The expression (4) gives an efficient way to compute superquantiles. We have indeed a three step procedure: (i) compute the p-quantile with the specialized algorithm (called `quickfind`) of complexity $O(n)$ (with n the number of data points); (ii) select all values greater or equal than the quantile; (iii) average values along (4). To minimize the superquantile-based objective (5), we would also need, in addition to an objective evaluation oracle, an oracle to obtain first-order information.

In this section, we study the differentiability properties of the superquantile objective and we describe how to obtain subgradient or gradient information with the same complexity $O(n)$ as for computing a standard quantile. We denote by

$$L^i(w) = \ell(y_i, \varphi(w, x_i)) \tag{8}$$

the underlying data-dependent functions in (1) and (5). We will distinguish two cases: (a) L^i convex in Section 3.1 and (b) L^i smooth in Section 3.2.

3.1 Subgradient oracle

We assume here that the functions L^i defined in (8) are convex. This is the case when e.g. the model φ is linear and the loss ℓ is convex with respect to its second variable, as for the l_2-squared or the cross-entropy loss. This encompasses several situations including p-least-squares regression with $p \geq 1$

$$L^i(w) = |y_i - w^\top x_i|^p$$

or the logistic regression which can be written with $\hat{y}_i = 1/(1+e^{-w^\top x_i})$ as

$$L^i(w) = -y_i \log(\hat{y}_i) - (1-y_i)\log(1-\hat{y}_i).$$

In this case, the superquantile-based function f of (5) is convex as well: we can see it on (7) which expresses f is a max, over q, of convex functions in w. We give here the expression of the entire subdifferential for the convex case. This result is not new: it is mentioned in several recent papers including [22, 9]; it is part of the thorough study of [36] where gradients[2] for general distributions are obtained from advanced tools. We give here a short proof using elementary convex analysis [13].

Proposition 1 *Assume that the L^i are convex. Fix $w \in \mathbb{R}^d$, compute $L(w) \in \mathbb{R}^n$ and $Q_p(L(w)) \in \mathbb{R}$. Consider $I_>$ the set of indices such that $L^i(w) > Q_p(L(w))$ and $I_=$ the set of indices such that $L^i(w) = Q_p(L(w))$. Then the subdifferential at w of the convex function f reads as the Minkowski sum*

$$\partial f(w) \;=\; \frac{1}{n(1-p)} \sum_{i \in I_>} \partial L^i(w) \;+\; \frac{\delta}{1-p}\, \mathrm{conv}\left\{ \partial L^i(w) : i \in I_= \right\}, \quad (9)$$

with $\delta = \frac{1}{n}(n - |I_>|) - p$. In particular, when the L^i is differentiable at w, f is differentiable at w if and only if the set $I_=$ is reduced to a singleton.

Proof The proof simply consists in applying convex calculus rules; the reader may find them in [13, Chap D]. First we apply Theorems 4.1.1 and 4.4.2 to $h_i(w,\eta) = \max\{L^i(w) - \eta, 0\})$ to get

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

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

$$\partial h(w,\eta) = \left\{ \left(\frac{1}{(1-p)n} \sum_{i=1}^n \partial L^i(w) \delta^i(w,\alpha), \right.\right.$$
$$\left.\left. 1 - \frac{1}{(1-p)n} \sum_{i=1}^n \delta^i(w,\alpha) \right), \alpha_i \in [0,1], \forall i \right\}.$$

with $\delta^i(w,\alpha) = (\mathbb{1}_{L^i(w)>Q_p(L(w))} + \alpha_i \mathbb{1}_{L^i(w)=Q_p(L(w))})$. We finish with writing $f(w) = \min_{\eta\in\mathbb{R}} h(w,\eta)$ from (6). We can thus apply Corollary 4.5.3 to get (9) after simplification. □

This proposition thus tells us that the computation of a subgradient can be performed in linear time from the subgradients $g_i \in \partial L^i(w)$ for $i \in I_> \cup I_=$: the computing cost essentially stems from the computation of the p-quantile of the losses $L^i(w)$ and the sum of vectors in $\mathbb{R}^d$.

[2] Interestingly, the nonsmoothness of superquantile-based functions arises only with discrete distributions, as we consider here.

3.2 Gradient oracle (for smoothed approximation)

We assume in this section that the functions L^i defined by (8) are smooth, which holds locally when both the model φ and the loss ℓ are smooth. Unfortunately, the superquantile breaks the smoothness (see e.g. Proposition 1 with smooth convex functions L^i), so that superquantile-based function f is usually nonsmooth.

We propose here to smooth f using infimal convolution as in [25]. More precisely, we follow the methodology of [1] and we propose to smooth only the superquantile $\bar{Q}_p$ rather than the whole function f. Given the formulation (7), we consider the function f_μ for $\mu > 0$, as the composition of the L^i by the infimal convolution smoothing of $\bar{Q}_p$

$$f_\mu(w) = \max_{q \in \Delta_n, q_i \leq \frac{1}{n(1-p)}} \sum_{i=1}^n q_i \, L^i(w) - \mu \, d(q) \tag{10}$$

where $d : \mathbb{R}^n \to \mathbb{R}$ is a fixed non-negative strongly convex function. As a direct application of [25, Th. 1], we have the following proposition establishing that f_μ is a smooth approximation of f.

Proposition 2 (Gradient of smoothed approximation) *Assume that the L^i are smooth for any i. Then, the function f_μ of (10) 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_μ is differentiable as well, with*

$$\nabla f_\mu(w) = \mathrm{J}L(w)^T q_\mu(w), \tag{11}$$

where $\mathrm{J}L(w)$ is the Jacobian of L at w and $q_\mu(w)$ is the optimal solution of (10), unique by strong convexity of d.

In practice, the previous result requires an efficient subroutine solving (10). Here, we consider the euclidean distance to the uniform probability measure

$$d(q) = \sum_{i=1}^n \left(q_i - \tfrac{1}{n}\right)^2. \tag{12}$$

For this distance, Algorithm 1 provides an efficient procedure for solving (10). The procedure follows the one in [8], where convex duality and one-dimensional search ideas are fruitfully combined. Thanks to the particular smoothing distance d, computing (10) by duality is equivalent to finding the zero of a non-decreasing, piecewise affine and continuous function (the derivative of the dual function), which has an explicit expression after sorting the kinky points. We formalize this in Algorithm 1 and the proposition below.

Proposition 3 *Algorithm 1 computes the optimal solution of the problem (10) with d as (12) at a cost of $O(n)$ operations.*

Algorithm 1: Fast subroutine for smoothed oracle

Initialize: $e = (1, \dots, 1)^\top, \quad u = L(w) + \frac{\mu}{n}\,\mathrm{e}, \quad \ell = \frac{1}{n(1-p)}, \;\; q_\mu = 0 \in \mathbb{R}^n$

Find in the points of non-differentiability $\mathcal{P}$, a and b such that,
$\mathcal{P} := \{u_i, u_i - \mu\ell, i \in \{1, \dots, n\}\}$
$a := \max\{s \in \mathcal{P}, \theta'(s) \leq 0\}$
$b := \min\{s \in \mathcal{P}, \theta'(s) > 0\}$;

Set the dual optimal solution $\lambda := a - \frac{\theta'(a)(b-a)}{\theta'(b)-\theta'(a)}$;

Construct the primal solution component-wise:

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 (10)

Proof 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 ℓ 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

$$\arg\max_{0 \leq q_i \leq l} h_i(q_i) = \begin{cases} 0 & \text{if } \lambda \geq u_i \\ \frac{u_i - \lambda}{\mu} & \text{if } u_i \geq \lambda \geq u_i - \mu\ell \\ \ell & \text{if } \lambda \leq u_i - \mu\ell \end{cases} \tag{13}$$

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

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

Observe 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, θ' 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 (13) for $\lambda = \lambda^\star$) is the solution of the

primal problem (10). Finally, we compute $\lambda^\star$ zeroing θ'. Since θ' is piecewise affine, we just need to evaluate θ' at points belonging to the set $\mathcal{P}$ and at a and b as defined in Algorithm 1. Thus we have $\lambda^* = a - \frac{\theta'(a)(b-a)}{\theta'(b)-\theta'(a)}$. Regarding computational costs, this algorithm boils down to the search of a and b, and the assignment of the coordinates of q_μ. This also sums up to a $\mathcal{O}(n)$ cost. □

Combining Propositions 2 and 3 provides a gradient oracle for the smoothed approximation f_μ. For a given $w \in \mathbb{R}^d$, we run Algorithm 1 to get $q_\mu(w)$; we select the indexes i of non-zeros entries of $q_\mu(w)$; and from the oracles of L^i we get

$$f_\mu(w) = \sum_{i:q_\mu(w)_i \neq 0} \big(q_\mu(w)\big)_i \, L^i(w) \quad \text{and} \quad \nabla f_\mu(w) = \sum_{i:q_\mu(w)_i \neq 0} \big(q_\mu(w)\big)_i \nabla L^i(w).$$

We finish this section by a short discussion on the two extreme cases for the smoothing parameters : μ close to 0 and μ very large. Small $\mu \sim 0$ imply exploding entries of $q_\mu(w)$ (see line 8 in Algorithm 1) and then instability of $\nabla f_\mu(w)$. Large $\mu \sim +\infty$ imply $q_\mu(w) = (1/n, \ldots, 1/n)$ constant (see line 6 in Algorithm 1) and therefore the smoothed function $f_\mu(w)$ and its gradient $\nabla f_\mu(w)$ coincide with the function and gradient of the corresponding ERM objective. We illustrate these two extreme cases in Section 6.

4 First-order optimization for superquantile-based learning

Minimization of superquantile-based objectives comes with a number of technical challenges on the structure of the problem tackled, the size of the dataset or the non-smoothness of the objective. Standard works on minimizing superquantiles considered linear programming or convex programming techniques, including interior point algorithms; see the review of [31]. Perhaps surprisingly, the use of first-order algorithms for superquantile-based optimization is quite recent and seems to have been driven by domain applications of machine learning.

In this section, we provide an overview of the range of first-order methods to minimize superquantile-based objective functions expressed as (5), (6), or (7). Our discussion focuses on practical considerations; we give pointers to references presenting more details and theoretical analysis (in particular, convergence results and convergence rates if any).

4.1 Batch algorithms

As explained in Section 3, computing the function values and (sub)gradients of the superquantile-based function f in (5) (or its smoothed counterpart f_μ) requires sorting loss values on the whole data set, which is not directly amenable to classical stochastic gradient algorithms. This rehabilitates batch optimization algorithms, at least for small to medium datasets. Thus the

first approach for minimizing the superquantile-based objective functions is to use standard subgradient-based methods (subgradient and dual averaging) or gradient-based methods (gradient, accelerated gradient, Quasi-Newton). This is essentially what we described in [18], and it is the first set of methods available in our toolbox. More precisely, we have two cases:

- *Convex case.* If $w \mapsto \ell(y_i, \varphi(w, x_i))$ are convex, then f is convex and we have a subgradient oracle (from Proposition 1) enjoying the same complexity as the one for computing a quantile. We can use standard convex nonsmooth optimization methods, such as subgradient methods and dual averaging. We implement in particular the "weighted" version of the latter with a Euclidean prox-function [26, Eq. 2.22]. These algorithms satisfy ergodic convergence guarantees in objective values [4].
- *Smooth case.* If $w \mapsto \ell(y_i, \varphi(w, x_i))$ are differentiable, then we have a gradient oracle of the smooth approximation f_μ (from Proposition 3), again with a $\mathcal{O}(n)$ complexity. We can use standard methods for smooth optimization: gradient method, accelerated gradient method, and quasi-Newton (L-BFGS). If furthermore we have convexity, these algorithms satisfy convergence guarantees in objective values [4, 5].

For small to medium-size dataset, such batch methods are shown to be simple and efficient; see forthcoming Section 6.1. For large-scale problems though, the oracles become too costly as they require sorting loss values on the whole data set. We turn to the other formulations to introduce stochastic and mini-batch algorithms, that usually are the methods of choice for the case of standard learning using empirical risk minimization.

4.2 Mini-batch algorithms

From the perspective of the formulation (6) of the objective, the superquantile-based learning problem writes

$$\min_{w\in\mathbb{R}^{\mathrm{d}}} \ \min_{\eta\in\mathbb{R}} \ \left\{ \frac{1}{n(1-p)} \sum_{i=1}^{n} \max\{\ell(y_i, \varphi(w, x_i)) - \eta, 0\} + \eta \right\}. \tag{14}$$

When the loss is assumed to be smooth, one may again smooth the inner $\max\{\cdot, 0\}$ term to get a smooth approximation of this joint objective. One can then perform a joint minimization[3] with respect to the model w and the dual variable η. In other words, superquantile learning reduces to a standard empirical risk minimization with a modified loss function truncated by the max-term. In practice, batch methods may not be interesting here, since they would not leverage the fact that the minimization over η can be performed explicitly. Thus [19] proposes, in a context of federated learning, to rather

[3] Such approach is well-suited to problems with a particular decomposable structure such as non-anticipativity constraints in multi-stage programming problems; see [33].

perform independent minimization over w and η alternatively. In general, this min-min approach (14) paves the way to stochastic and mini-batch algorithms.

Several works, including [38] and [40] (as well as [12] without mentioning superquantile), use successfully standard stochastic optimization algorithms on this modified objective. Observe though that, if a mini-batch of data is sampled uniformly at random from the data, only a fraction $(1-p)$ carry (sub)gradient information. Furthermore, the (sub)gradients of these examples are scaled by $\frac{1}{1-p}$, leading to exploding directions. Thus mini-batch estimates of (sub)gradients of superquantile-based objectives may suffer from high variance. A solution proposed by [9] is to perform an adaptive sampling rather than a uniform one. This algorithm gradually adjusts its sampling distribution to increasingly sample tail events, until it eventually minimizes the superquantile. This approach has a nice two-player interpretation related to the third formulation, recalled below.

The third expression (7) of f leads to the following formulation (or, as previously, its smoothed counterpart with a quadratic term on q as in (10))

$$\min_{w\in\mathbb{R}^{\mathrm{d}}} \max_{q\in\Delta_n} \left\{ \sum_{i=1}^{n} q_i\, \ell(y_i, \varphi(w, x_i)) : 0 \le q_i \le \frac{1}{n(1-p)} \right\}. \tag{15}$$

This min-max formulation offers several ways to solve the superquantile-based learning. A first approach would consist in considering it as a generic saddle point problem and using standard (extra-)gradient algorithms or recent extensions exploiting some aspects of the problem (see e.g. [23] for a variance-reduced min-max with strongly concave max). In our specific case, computing the max can be done systematically by a greedy algorithm with linear time complexity (see Section 3). This key feature is exploited by the stochastic algorithm of [11], and also by the one of [15] without relating it to superquantile. This algorithm uses a biased sampling approximation to f or f_μ which has nice guarantees. We briefly describe below this approach.

We sample a mini-batch of $\mathcal{S}$ uniformly in $\mathcal{D}$ and we consider the restriction

$$\begin{aligned} \widetilde{f}(w) &= [\bar{Q}_p]_{(x,y)\sim\mathcal{S}}\big(\ell(y, \varphi(w, x)\big) \\ &= \max_{q\in\Delta_n} \left\{ \sum_{i\in\mathcal{S}} q_i\, \ell(y_i, \varphi(w, x_i)) : 0 \le q_i \le \frac{1}{n(1-p)} \right\}. \end{aligned}$$

We can now use the (sub)gradient oracles of Section 3 on $\widetilde{f}$ and apply gradient-based algorithms with biased mini-batch estimator. Indeed, even if $\mathbb{E}[\widetilde{f}(w)] \neq f(w)$ and $\mathbb{E}[\nabla\widetilde{f}(w)] \neq \nabla f(w)$, under standard assumptions, the bias is controlled by uniform bounds and variance bounds, which gives in turn complexity guarantees when using gradient-based algorithms; see [22, Sec. 3]. The algorithm requires a number of gradient evaluations independent of training set size and number of parameters, making it suitable for large-scale applications. This algorithm is implemented in our toolbox and tested in Section 6.

5 SPQR: Python Toolbox for Superquantile-based learning

We provide a Python software package for superquantile-based learning; it is named `SPQR` for SuPer Quantile Risk optimization. The software package includes modeling tools and optimization algorithms to solve problems of the form (5) with just a few lines of code. The implementation builds off basic structures of `scikit-learn` [29] the popular python machine learning library. `SPQR` routines rely on just-in-time compilation [20] to ensure efficient running times. The software package is publicly available at `https://github.com/yassine-laguel/spqr`. We now walk the reader through the toolbox `SPQR`.

5.1 Basic usage: input format and execution

The user provides a dataset modeled as a couple $(X, Y) \in \mathbb{R}^{n\times p} \times \mathbb{R}^p$ and a first-order oracle for the function L^i. The dataset is stored into two 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 variable and `(x,y)` a datapoint. For instance, the oracle for superquantile linear regression are the following one.

```
# 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 (5), we instantiate the `RiskOptimizer` object with the oracles, following the standard usage of `scikit-learn`. The basic instantiation is:

```
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 provided data, to get a solution of (5).

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

5.2 Advanced use: parameters and SPQR objects

Options and parameters. The customizable parameters are stored in a python dictionary `params` which is designed as an attribute of the `RiskOptimizer` class. The main parameters to tune are: the choice of the oracle, the choice of the algorithm, the safety probability level `p`, the starting point of the algorithm `w_start`, the maximum number of iterations `max_iter`. The user can specify some of these parameters as an input and the others will be filled with defaults values when instantiating a `RiskOptimizer`. For example:

```
custom_params = {'algorithm': 'dualaveraging', # selected algorithm
    'p': 0.2 } # safety probability level
custom_optimizer = RiskOptimizer(loss, loss_prime, params=custom_params)
```

Some important parameters (such as the safety probability level, the algorithm chosen, or the smoothing parameter μ) can be given directly to the constructor of the class `RiskOptimizer` when instantiating the object. For example:

```
other_custom_optimizer = RiskOptimizer(loss, loss_prime, p=0.95,
     algorithm='bfgs', mu=0.1)
```

Oracle classes. The selection of the oracle is automatically done when the user instantiate the `RiskOptimizer` object. Four different oracles are implemented as python objects: two oracles for batch methods (`OracleSubgradient` to be used when the chosen algorithm is `'subgradient'` or `'dual_averaging` and `OracleSmoothGradient` when the chosen algorithm is `'gradient'`, `'nesterov'` or `'bfgs'`) and two mini-batch oracles (`OracleStochasticSubgradient` and `OracleStochasticGradient`).

To avoid the treatment of optional parameters when instantiating an oracle, we advise to go through the instantiation of a `RiskOptimizer` first.

```
custom_params = {'algorithm': 'nesterov', # selected algorithm
    'p': 0.5 # safety probability level
}
#  Instantiation of the Risk Optimizer
custom_optimizer = RiskOptimizer(loss, loss_prime, params=custom_params)
#  Recovery of the oracle
smooth_oracle = custom_optimizer.oracle
```

Algorithms class The algorithm chosen is a parameter for the instantiation of the `RiskOptimizer` class. This parameter can either be given in the input dictionary `params` or directly to the constructor of `RiskOptimizer`. The user has the choice among `'subgradient'`, `'dual_averaging'`, `'gradient'`, `'nesterov'`, `'bfgs'` and `'sgd'`.

```
#  Risk Optimizer class with nesterov accelerated gradient algorithm
custom_optimizer = RiskOptimizer(loss, loss_prime, algorithm='nesterov')
```

Each algorithm is implemented as a python class that stores the oracle, together with relevant parameters for the optimization process. The main method of this class is `run`, which is run when `RiskOptimizer.fit` is called. The parameters of the algorithm selected are stored in the dictionary `params` that is an input of the class `RiskOptimizer`. Hence, in a standard usage, there is no need to interact with the algorithm python object.

6 Numerical experiments

In this section, we report two types of numerical experiments:

- "Optimization" experiments in Section 6.1. There are many algorithmic options within the toolbox `SPQR`; we provide a comparison of batch vs. mini-batch algorithms and a discussion on tuning the smoothness parameter.
- "Learning" experiments in Section 6.2. The interest of using superquantile in learning has been shown empirically in several recent papers, including [40, 19, 22, 9, 38]. We provide here complementary experiments highlighting the robustness of superquantile-learnt models.

All experiments are run using `SPQR`. The optimization algorithms are initialized at $w = 0 \in \mathbb{R}^{\mathrm{d}}$. For these experiments, we use a bunch of standard datasets from the UCI repository, which scale from 352 to 94644 datapoints. For each dataset, categorical features were one-hot encoded so that the total number of features ranges from 3 to 287.

For the experiment of Table 1, we report the agreggated results for all the datasets. For the other experiments, we report, in the main text, the detailed results obtained with one representative dataset, and we provide, in appendix, complementary results for others datasets.

6.1 Solving superquantile-based learning.

In this section, we illustrate two different aspects of the optimization methods available in `SPQR`. First, we compare the two families of algorithms available (batch vs. mini-batch) showing the interest of using batch algorithms for superquantile-based learning within `scikitlearn/SPQR`. Second, we experiment with all the range of the smoothing parameter, advocating to avoid extreme values.

Batch vs. mini-batch. We compare, on a standard problem, a stochastic gradient algorithm (more precisely, SGD with momentum, denoted SGD) and a batch quasi-Newton algorithm (more precisely, low-memory BFGS [27], denoted BFGS).

For this experiment, the set-up is similar to the one of [22]. We consider a supervised multi-class classification task with the superquantile multinomial logistic loss on the `MNIST` dataset. We perform feature extraction from the

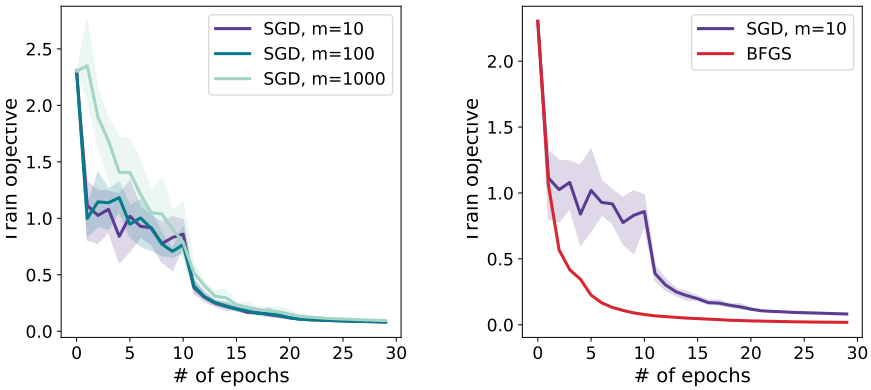

Fig. 3: A comparison between batch/mini-batch algorithms in `SPQR` on a superquantile logistic regression problem with `MNIST`. Left: comparison of the runs of SGD with different batch sizes. Right: best SGD vs. batch quasi-Newton.

images using a pre-trained convolutional network similarly to [22]. For a fixed probability threshold set to $p = 0.8$, we then train a linear multi-class classifier on top of the transformed data. For SGD, we use a momentum term of 0.9 and we use a step decay scheme $\eta_t = \eta_0 d^{-\lfloor t/t_0 \rfloor}$, where η_0 is tuned with respect to the size of the mini-batch m, and where $d = 0.5$ and $t_0 = 10$ epochs are fixed throughout all the experiments. For each mini-batch size $m \in \{10, 100, 1000\}$, we tune η_0 via a grid-search and take the highest initial value yielding a non-diverging sequence of iterates. In constrast with SGD, the quasi-Newton algorithm does not require specific tuning as it automatically calibrates stepsizes by line-searches at each iteration.

On the left part of Figure 3, we compare the performance of SGD for the different mini-batch sizes. Each color corresponds to a mini batch size $m \in \{10, 100, 1000\}$. Along iterates, the bold line represents the mean value over the five seeds of the functions and the shaded region represent the difference between the min and max values across the seeds. We observe that there is no substantial difference among the sizes of the mini-batches: all curves show a noisy behaviour (caused by the stochastic approximation of the gradient at each step) and eventually converge to a suboptimal value. Unlike SGD, L-BFGS (right part of Figure 3) presents a stable convergence. We observe also that a large number of epochs is necessary for SGD to catch up with BFGS for superquantile-based training. This is to be contrasted with the usually small number of epochs necessary for SGD to catch with BFGS for expectation-based training or ERM. Note that a final bias remains visible between the stochastic methods and the deterministic BFGS, as expected by the theory laid down in [22].

Impact of the smoothing parameter. We consider a logistic regression on the `Australian Credit` dataset. For a sequence of smoothing parameters ν evenly spread on a log scale, we train $w_\nu^\star$ by solving the superquantile learning objective with L-BFGS and $p = .99$.

On Figure 4, we report both the value of the smoothed .99-superquantile (purple) and the non-smoothed .99-superquantile (dashed green) at the $w_\nu^\star$. We also train the standard empirical risk minimizer $w^\star$ and we report both the average loss (solid black line) and the non-smoothed .99-superquantile loss (dashed black line) at $w^\star$.

For very small values of ν ($< 10^{-3}$), we observe unsuccessful termination of the L-BFGS algorithm, due to the failure of the line-search. For medium values of ν (< 1), the value of smooth superquantile-based function at $w_\nu^\star$ roughly coincides the non-smooth one. Finally for high values of ν ($> 10^3$), we observe that the smooth superquantile tends to the same optimal function value of the empirical risk minimizer $w^\star$, as expected from Section 3.2.

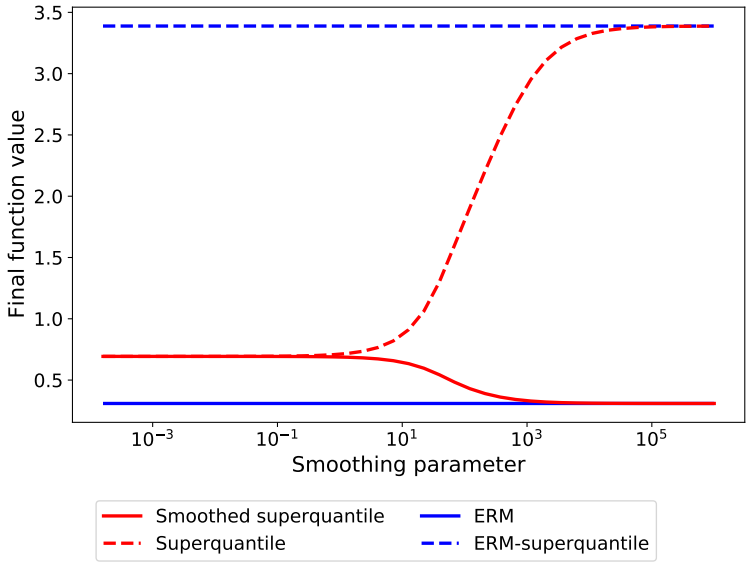

Fig. 4: Impact of the smoothing parameter ν on the results obtained by the quasi-Newton algorithm solving a superquantile logistic regression on the `Australian Credit` dataset. Medium values are preferable: small values compromise convergence and large values give solutions close to the standard ERM.

6.2 Superquantile brings robustness against distributional shift

In the second part of the numerical experiments, we show the benefits of the superquantile by comparing superquantile-based minimization vs. empirical risk minimization, when a distributional shift occurs, similarly to [9]. For the three next standard regression or classification tasks, we proceed as follows. For each dataset, we first perform a 80%-20% train-test split. Second, we minimize with respect to the train set a regularized objective, both in expectation and with respect to the superquantile:

$$\begin{aligned} &\min_{w\in\mathbb{R}^{\mathrm{d}}} \ \mathbb{E}_{(x,y)\sim D_{\mathrm{train}}}\ell(y,w^\top x)+\frac{\lambda}{2}\|w\|_2^2 \\ &\min_{w\in\mathbb{R}^{\mathrm{d}}} \ [\bar{Q}_p]_{(x,y)\sim D_{\mathrm{train}}}\ell(y,w^\top x)+\frac{\lambda}{2}\|w\|_2^2 \end{aligned} \tag{16}$$

We set the regularization parameter λ to be the inverse of the number of training data-points: $\lambda = 1/n_{\mathrm{train}}$. The above problems are solved with `SPQR` using L-BGFS. Then we perform three different types of distributional shifts on the testing set and we compare the behaviour of the superquantile-based models and the ERM models. We develop this approach in the next three settings.

Superquantile ridge regression. We consider a ridge regression problem, that is (16) with $\ell(y,w^\top x)=(y-w^\top x)^2$, on the dataset `Cpu-small`. We minimize the two problems, first, in expectation and, second, with respect to the superquantile with several safety thresholds $p\in\{0.3,0.5,0.7,0.8,0.9,0.95,0.99\}$.

We report in Figure 5 the histogram of losses on the test set and compare each trained superquantile model (in red) with the ERM model (in blue). We observe that as the probability threshold p grows, the right tail distribution of losses on the test set gets shifted to the left. In particular, a dramatic decrease of the 90^{th} quantile of the losses can be observed. Thus superquantile learning allows us to reduce worst-case losses. This comes with the price of lower performances on the left tail distribution. This trade-off between gain on extreme cases and loss on average is typical of the impact of superquantiles. We observe a similar trade-off for other datasets; see Figures 7 in appendix.

Superquantile logistic regression. We consider a regularized logistic regression problem, that is (16) with $\ell(y,w^\top x)=-y\sigma(w^\top x)-(1-y)\sigma(w^\top x)$ (where $\sigma(z):=\frac{1}{1+e^{-z}}$ denotes the sigmoid function). We use 10 classification datasets from the UCI repository library and we perform a distributional shift on the train sets: we subsample the majority class so that it accounts for only 10% of the minority class. Then we train a ERM and superquantile models. The safety parameter p is tuned via a cross validation procedure on the shifted train set. We finally compute, for the best parameter obtained, the test accuracy and the test loss.

We report our results in Table 1. For most datasets, we note a significant decrease of the test loss with the superquantile model, when compared to ERM

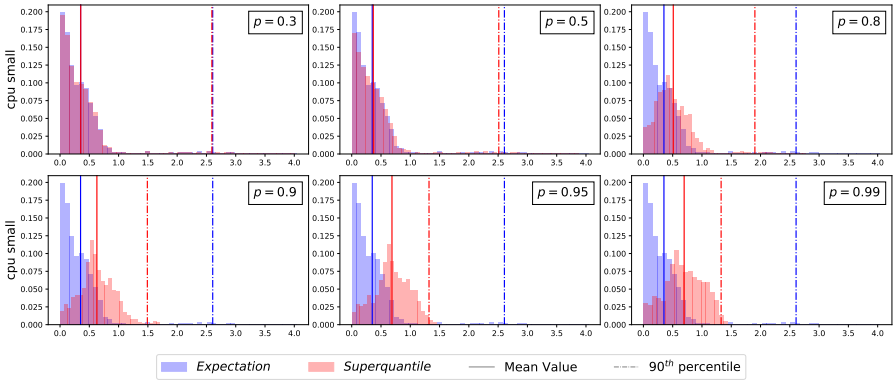

Fig. 5: Reshaping of the histogram of testing losses for superquantile regression models (in red) as p grows. We observe a shift to the left of the 90^{th} quantile of losses, at the price of degrading the average value.

Dataset	Superquantile Accuracy	Superquantile Loss	Expectation Accuracy	Expectation Loss
Adult	53.2 ± 0.67	0.693 ± 0.00	$\mathbf{55.4 \pm 0.48}$	1.072 ± 0.01
Monks	$\mathbf{64.4 \pm 2.65}$	0.714 ± 0.05	54.0 ± 1.57	1.207 ± 0.08
Splice	$\mathbf{82.7 \pm 0.62}$	0.681 ± 0.05	81.7 ± 0.78	0.557 ± 0.04
Diabetes	42.5 ± 4.72	0.694 ± 0.00	$\mathbf{45.1 \pm 4.51}$	1.325 ± 0.12
Spambase	$\mathbf{78.4 \pm 1.23}$	0.761 ± 0.15	77.1 ± 0.87	0.635 ± 0.07
Mammography	$\mathbf{39.1 \pm 7.59}$	0.730 ± 0.01	$\mathbf{39.1 \pm 6.90}$	1.293 ± 0.09
Electricity	42.8 ± 0.40	0.693 ± 0.00	$\mathbf{47.5 \pm 0.63}$	1.060 ± 0.01
Phoneme	37.3 ± 5.38	0.737 ± 0.01	$\mathbf{50.5 \pm 3.10}$	1.292 ± 0.04
Nomao	$\mathbf{87.5 \pm 0.22}$	0.413 ± 0.03	$\mathbf{87.4 \pm 0.23}$	0.394 ± 0.02
Skin-segmentation	$\mathbf{92.1 \pm 0.11}$	0.420 ± 0.00	$\mathbf{91.9 \pm 0.05}$	0.537 ± 0.01

Table 1: Comparison of performances between a superquantile model and a risk-neutral model for a logistic regression on a distributionally shifted dataset.

model. In terms of accuracy, the superquantile model offers better performance for this particular distributional shift.

Robustness to all possible distributional shifts. We take the same setting as before, focusing on the `splice` dataset, and now we perform a sequence of distributional shifts on the training set by rebalancing all the proportions of the two classes. More precisely, for a fixed $\alpha \in (0, 1)$, we compute the number $n_{\min}$ of samples from the minority class; we randomly select $\lceil \alpha n_{\min} \rceil$ points from the majority class and $\lceil (1-\alpha) n_{\min} \rceil$ from the minority class. We train on the shifted train set the two logistic regression models of (16). We repeat this experiment for 5 different seeds and we compute the average test losses and test accuracies of both models. The experiment is conducted for 100 values of α evenly spread on $(0, 1)$.

The histograms of Figure 6 depicts the performances, as α varies, of ERM against the superquantile (for a fixed probability threshold p). In terms of losses, the superquantile model brings better performances for almost all values of p. In particular, the 90^{th} quantile of the losses over all considered shifts gets notably decreased for p. In terms of accuracy, the superquantile models brings better performance with respect to distributional shifts for all values of p. Such behaviours are also observed with other datasets, as those depicted in Figures 8 and 9 in Appendix.

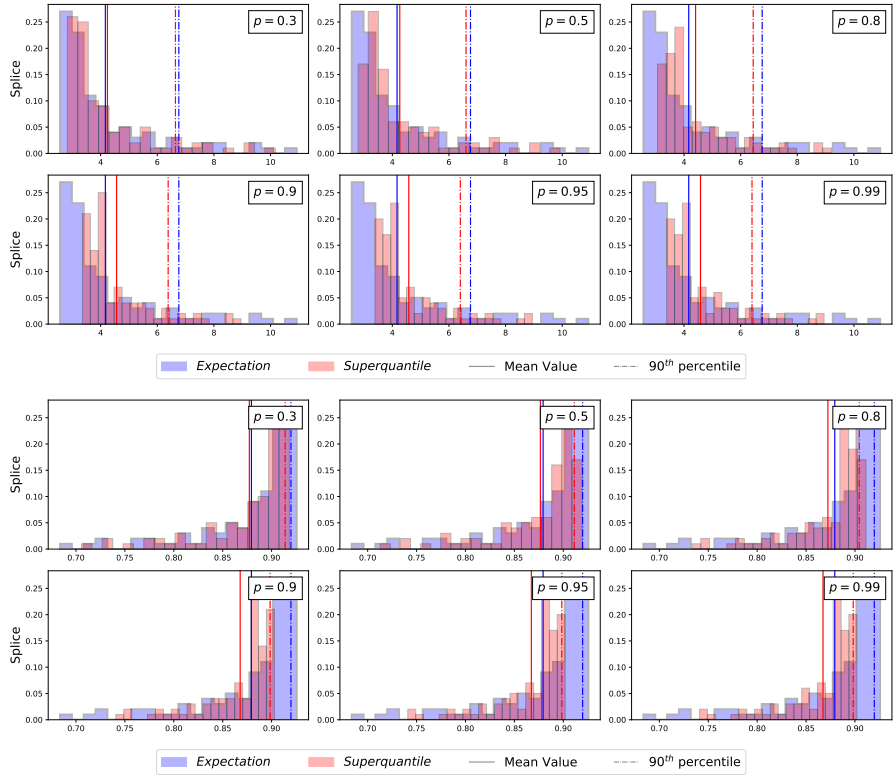

Fig. 6: Reshaping of histograms of test losses (top) and test accuracies (bottom) over all class imbalances (for a classification task with logistic regression and the `splice` dataset).

7 Conclusion, perspectives

Risk-sensitive optimization can play an important role in the design of safer machine learning models involved in automated decision making. We provide

here a software package to tackle superquantile-based learning problems using standard first-order optimization algorithms. The software package is publicly available on the authors' websites. We have described the main components of the optimization algorithms and how they can be made to tackle superquantile-based learning problems using smoothing techniques in particular. We would tend to recommend the use of a combination of smoothed oracles and batch gradient algorithms to experiment with superquantile-based objectives.

This work can be included in the more general research stream on developing operational tools for distributionally robust learning, which has recently gained interest and focus in the machine learning community; see e.g. the recent textbook [6]. Recent work on related topics developing optimization algorithms with improved complexity bounds [9, 22], exploring fairness challenges [40], tackling data heterogeneity problems [19], shows the burst of activity in this general area and suggests a number of venues for future investigation.

Acknowledgements We acknowledge support from ANR-19-P3IA-0003 (MIAI – Grenoble Alpes), as well as NSF DMS 2023166, DMS 1839371, CCF 2019844, CIFAR LMB, and faculty research awards.

References

1. Beck, A., Teboulle, M.: Smoothing and first order methods: A unified framework. SIAM Journal on Optimization (2012). URL https://doi.org/10.1137/100818327
2. Ben-Tal, A., El Ghaoui, L., Nemirovski, A.: Robust optimization. Princeton University Press (2009)
3. Ben-Tal, A., Teboulle, M.: An old-new concept of convex risk measures: The optimized certainty equivalent. Mathematical Finance (2007)
4. Bertsekas, D.: Convex Optimization Algorithms. Athena Scientific (2015)
5. Bertsekas, D.: Nonlinear Programming. Athena Scientific (2016)
6. Chen, R., Paschalidis, I.C., et al.: Distributionally robust learning. Foundations and Trends® in Optimization **4**(1-2), 1–243 (2020)
7. Chow, Y., Tamar, A., Mannor, S., Pavone, M.: Risk-sensitive and robust decision-making: a cvar optimization approach. In: Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 1, pp. 1522–1530 (2015)
8. Condat, L.: Fast projection onto the simplex and the l1 ball. Mathematical Programming (2016)
9. Curi, S., Levy, K.Y., Jegelka, S., Krause, A.: Adaptive sampling for stochastic risk-averse learning. Advances in Neural Information Processing Systems **33** (2020)
10. Duchi, J., Namkoong, H.: Learning models with uniform performance via distributionally robust optimization. arXiv preprint arXiv:1810.08750 (2018)
11. Duchi, J.C., Namkoong, H.: Variance-based Regularization with Convex Objectives. Journal of Machine Learning Research (2019)
12. Fan, Y., Lyu, S., Ying, Y., Hu, B.G.: Learning with average top-k loss. In: NIPS (2017)
13. Hiriart-Urruty, J.B., Lemaréchal, C.: Convex analysis and minimization algorithms I: Fundamentals. Springer science & business media (2013)
14. Ho-Nguyen, N., Wright, S.J.: Adversarial classification via distributional robustness with wasserstein ambiguity. preprint arXiv:2005.13815 (2020)
15. Kawaguchi, K., Lu, H.: Ordered sgd: A new stochastic optimization framework for empirical risk minimization. In: International Conference on Artificial Intelligence and Statistics, pp. 669–679. PMLR (2020)
16. Knight, W.: A self-driving Uber has killed a pedestrian in Arizona. Ethical Tech (2018)
17. Kuhn, D., Esfahani, P., Nguyen, V.A., Shafieezadeh-Abadeh, S.: Wasserstein distributionally robust optimization: Theory and applications in machine learning. In: Operations Research & Management Science in the Age of Analytics. INFORMS (2019)
18. Laguel, Y., Malick, J., Harchaoui, Z.: First-order optimization for superquantile-based supervised learning. In: 2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP), pp. 1–6. IEEE (2020)
19. Laguel, Y., Pillutla, K., Malick, J., Harchaoui, Z.: A superquantile approach to federated learningwith heterogeneous devices. In: 55th Annual Conference on Information Sciences and Systems, CISS. IEEE (2021)
20. Lam, S.K., Pitrou, A., Seibert, S.: Numba: A llvm-based python jit compiler. In: Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC, LLVM '15. Association for Computing Machinery, New York, NY, USA (2015)
21. Lee, J., Raginsky, M.: Minimax statistical learning with Wasserstein distances. In: Advances in Neural Information Processing Systems (2018)
22. Levy, D., Carmon, Y., Duchi, J.C., Sidford, A.: Large-scale methods for distributionally robust optimization. Advances in Neural Information Processing Systems **33** (2020)
23. Luo, L., Ye, H., Huang, Z., Zhang, T.: Stochastic recursive gradient descent ascent for stochastic nonconvex-strongly-concave minimax problems. Advances in Neural Information Processing Systems **33** (2020)
24. Metz, R.: Microsoft's neo-Nazi sexbot was a great lesson for makers of AI assistants. Artificial Intelligence (2018)
25. Nesterov, Y.: Smooth minimization of non-smooth functions. Mathematical programming (2005)
26. Nesterov, Y.: Primal-dual subgradient methods for convex problems. Mathematical programming (2009)

27. Nocedal, J., Wright, S.: Numerical optimization. Springer Science & Business Media (2006)
28. Owen, A.: Empirical Likelihood. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. CRC Press (2001)
29. Pedregosa, F., et al.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research (2011)
30. Recht, B., Roelofs, R., Schmidt, L., Shankar, V.: Do imagenet classifiers generalize to imagenet? arXiv:1902.10811 (2019)
31. Rockafellar, R., Royset, J., Miranda, S.: Superquantile regression with applications to buffered reliability, uncertainty quantification, and conditional value-at-risk. European Journal of Operational Research (2014)
32. Rockafellar, R., Uryasev, S., Zabarankin, M.: Risk tuning with generalized linear regression. Mathematics of Operations Research (2008)
33. Rockafellar, R.T.: Solving stochastic programming problems with risk measures by progressive hedging. Set-Valued and Variational Analysis **26**(4), 759–768 (2018)
34. Rockafellar, R.T., Royset, J.O.: Superquantiles and their applications to risk, random variables, and regression. In: Theory Driven by Influential Applications. INFORMS (2013)
35. Rockafellar, T., Uryasev, S.: Optimization of Conditional Value-at-Risk. Journal of Risk (2000)
36. Ruszczyński, A., Shapiro, A.: Optimization of convex risk functions. Mathematics of operations research (2006)
37. Shapiro, A., Dentcheva, D., Ruszczyński, A.: Lectures on stochastic programming: modeling and theory. SIAM (2014)
38. Soma, T., Yoshida, Y.: Statistical learning with conditional value at risk. arXiv preprint arXiv:2002.05826 (2020)
39. Wilder, B.: Risk-sensitive submodular optimization. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32 (2018)
40. Williamson, R.C., Menon, A.K.: Fairness Risk Measures. In: International Conference on Machine Learning (2019)

A Additional numerical results

In this Appendix section, we collect additional results comparing classical supervised learning and superquantile-based supervised learning using the optimization algorithms described in the main text. The experimental setting is exactly the one of Section 6.2, yet we consider here other datasets from UCI repository. The results obtained are essentially the same as the ones presented in Section 6.2, suggesting a greater control of extreme losses and a greater robustness to distributional shift of superquantile-based supervised learning. We refer to the main text for the discussions on these main observations, and we give here additional observations.

Figure 7 shows the same behaviour as in Figure 5: as the probability threshold p grows, the right tail distribution of losses on the test set gets shifted to the left. We can indeed see, on the subfigures, the reshaping of the histogram and the translation of the 90^{th} quantile to the left. Two exceptions should be noticed though: for the dataset `boston housing` with $p = 0.95$ and 0.99, the superquantile approach was not able decrease the 90^{th} quantile (see the last two subplots at the bottom). This would suggests to avoid in general using too large values of p that would restrict the computational effort to a too small fraction of extreme scenarios only.

Figures 8 and 9 show results similar to the ones presented in Figure 6 about the resistance to distributional shifts. The three datasets considered here provide a variety of histograms shapes. We see on Figure 8 that the superquantile brings better performances on the worst-case test losses for all values of p (except for the `skin-segmentation` with $p \geq 0.9$). Similarly, on Figure 9, we see, in most cases, improvements of the worst-case test accuracy: sometimes the improvement is important (e.g. `Australian` with $p = 0.9$), sometimes it is more marginal or even negative (e.g. `monks-problem1` with $p = 0.9$).

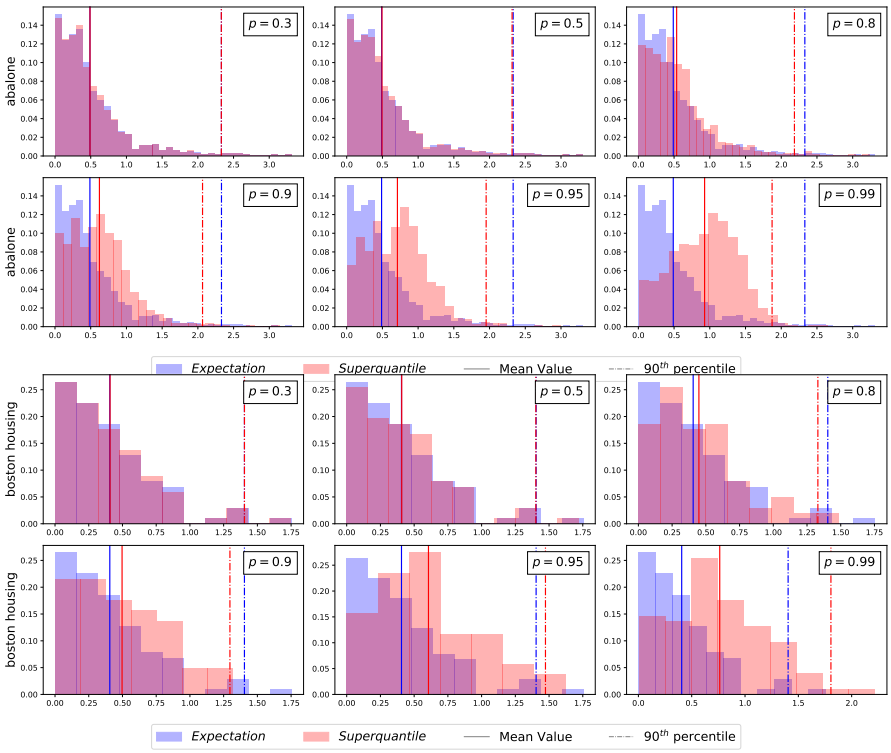

Fig. 7: Ridge regression: comparison of performances between a superquantile model and a ERM model for `Abalone` and `Boston Housing`.

Interestingly, one observes that, for each dataset, there is a particular value of p (depending on the dataset) for which the histogram of losses gets shifted to the left uniformly ($p = 0.8$ for `monks-problem-1` and $p = 0.3$ for the `skin-segmentation` dataset). This highlights the importance of a careful tuning of p to address the worst-case outcomes.

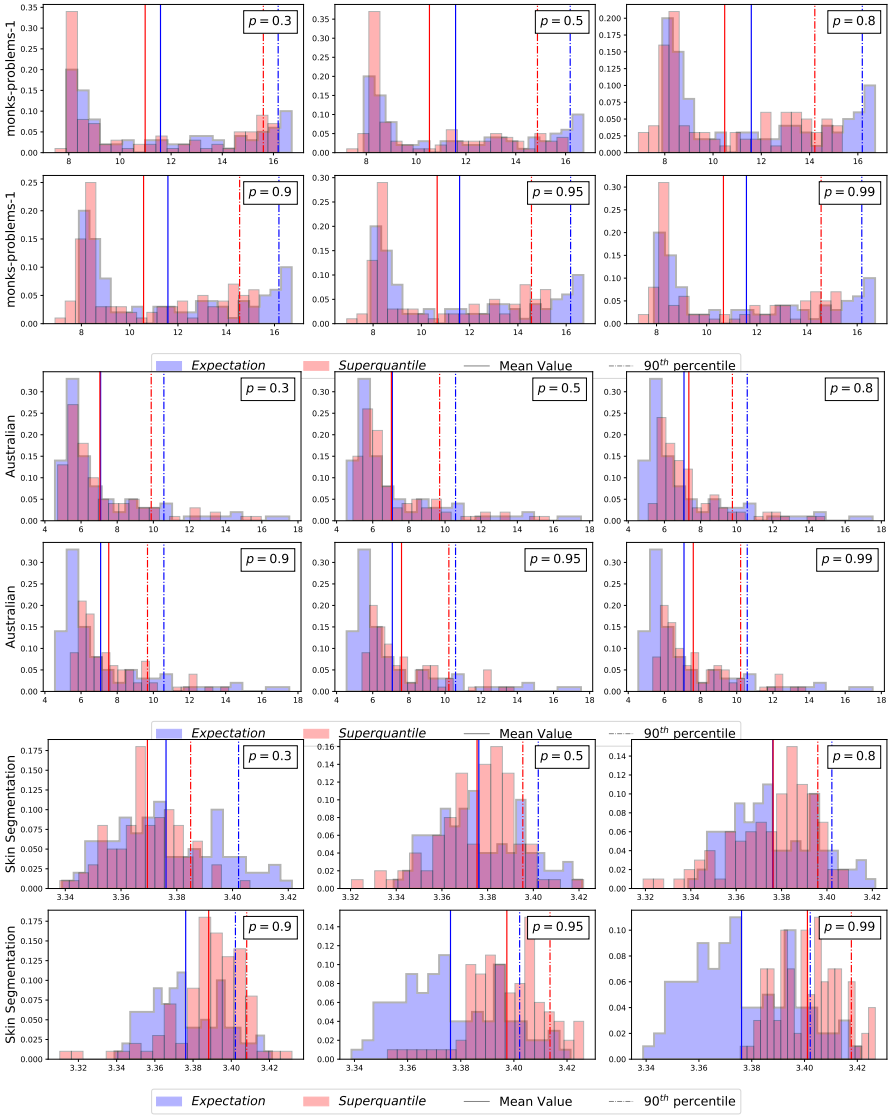

Fig. 8: Histogram of test losses over all distributional shifts for the datasets `monks-problem-1`, `australian-credit`, and `skin-segmentation`.

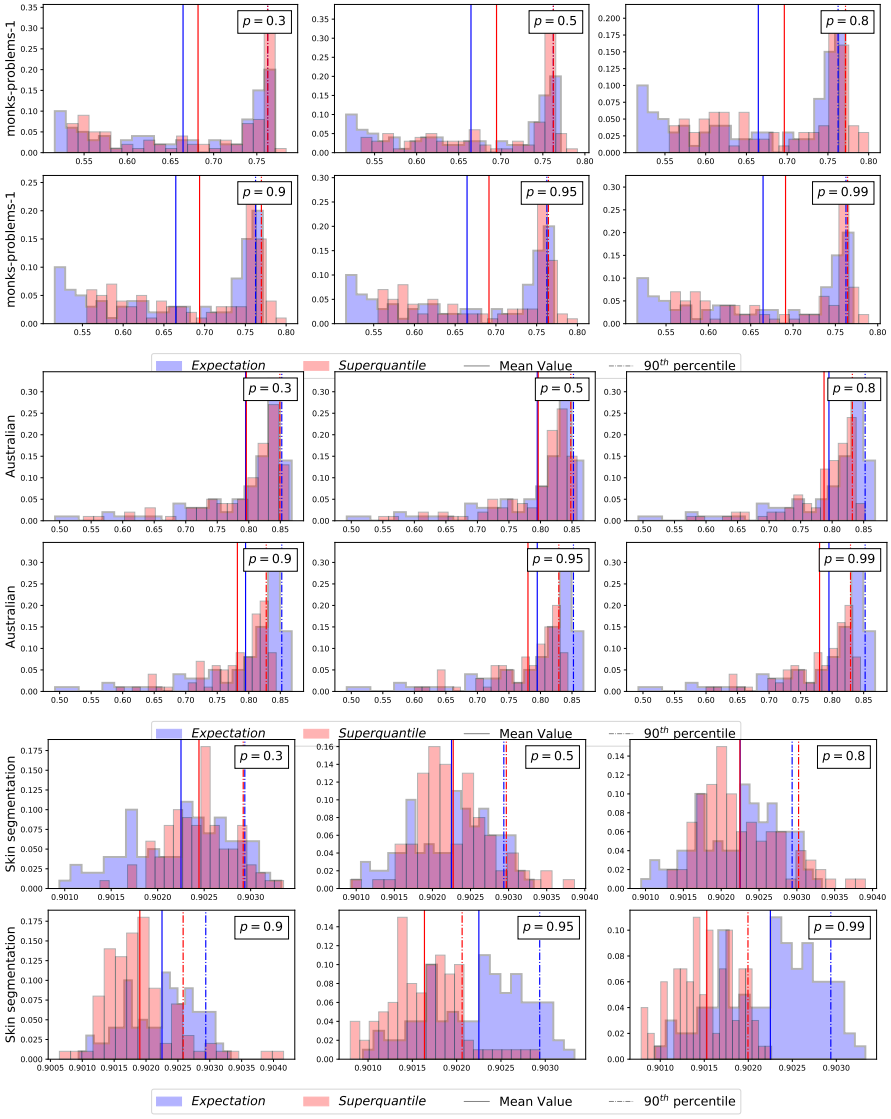

Fig. 9: Histogram of test accuracy over all distributional shifts for the datasets `monks-problem-1`, `australian-credit`, and `skin-segmentation`.