Robust Regression via Model Based Methods[⋆]

Armin Moharrer ✉$^{[0000-0002-8374-7286]}$, Khashayar Kamran$^{[0000-0002-4086-1038]}$, Edmund Yeh$^{[0000-0002-9544-1567]}$, and Stratis Ioannidis$^{[0000-0001-8355-4751]}$

Northeastern University, Boston MA 02115, USA
{amoharrer,kamrank,eyeh,ioannidis}@ece.neu.edu

Abstract. The mean squared error loss is widely used in many applications, including auto-encoders, multi-target regression, and matrix factorization, to name a few. Despite computational advantages due to its differentiability, it is not robust to outliers. In contrast, ℓ_p norms are known to be robust, but cannot be optimized via, e.g., stochastic gradient descent, as they are non-differentiable. We propose an algorithm inspired by so-called model-based optimization (MBO) [35, 36], which replaces a non-convex objective with a convex model function and alternates between optimizing the model function and updating the solution. We apply this to robust regression, proposing SADM, a stochastic variant of the Online Alternating Direction Method of Multipliers (OADM) [48] to solve the inner optimization in MBO. We show that SADM converges with the rate $O(\log T/T)$. Finally, we demonstrate experimentally (a) the robustness of ℓ_p norms to outliers and (b) the efficiency of our proposed model-based algorithms in comparison with gradient methods on autoencoders and multi-target regression.

1 Introduction

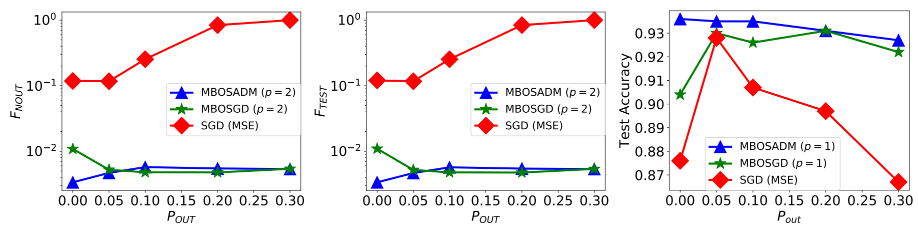
Mean Squared Error (MSE) loss problems are ubiquitous in machine learning and data mining. Such problems have the following form:

$$\min_{\theta} \frac{1}{n}\sum_{i=1}^{n} \|F(\boldsymbol{\theta}; \boldsymbol{x}_i)\|_2^2 + g(\boldsymbol{\theta}), \tag{1}$$

where function $F : \mathbb{R}^d \times \mathbb{R}^m \to \mathbb{R}^N$ captures the contribution of a sample $\boldsymbol{x}_i \in \mathbb{R}^m, i = 1, \dots, n$, to the objective under the parameter $\boldsymbol{\theta} \in \mathbb{R}^d$ and $g : \mathbb{R}^d \to \mathbb{R}$ is a regularizer. Example applications include training auto-encoders [18, 28], matrix factorization [16], and multi-target regression [47].

The MSE loss in (1) is computationally convenient, as the resulting problem is smooth and can thus be optimized efficiently via gradient methods, such as stochastic gradient descent (SGD). However, it is well-known that the MSE loss

⋆ The authors gratefully acknowledge support from the National Science Foundation (Grants CCF-1750539, IIS-1741197, and CNS-1717213), DARPA (Grant HR0011-17-C-0050), and a research grant from American Tower Corp.



(a) Avg. Non-outliers Loss (b) Avg. Test Loss (c) Accuracy

Fig. 1: Robustness of ℓ_p norms vs. MSE to outliers introduced to `MNIST` when training an autoencoder. Figures 1a and 1b show the average loss over the non-outliers and the test set, respectively; values in each figure are normalized w.r.t. the largest value. The test accuracy of a logistic regression on the latent features is shown in Fig. 1c. We see that, under MSE, both for the loss values and classification accuracy are significantly affected by the fraction of outliers $P_{\mathtt{out}}$. Robust embeddings under $p = 1, 2$ norms optimized via our proposed MBO methods exhibit almost constant behavior w.r.t. $P_{\mathtt{out}}$.

is not robust to *outliers* [8, 15, 19, 20, 34], i.e., samples far from the dataset mean. Intuitively, when squaring the error, outliers tend to dominate the objective. To mitigate the effect of outliers, a classic approach is to introduce robustness by replacing the squared error with either the ℓ_2 norm [8, 11, 18, 28, 34, 41] or the ℓ_1 norm [2, 5, 13, 19–21, 24, 40]. This has been applied to several applications, including feature selection [34, 41], PCA [2, 8, 21, 24], K-means clustering [11], training autoencoders [18,28], matrix factorization [5,13,19,20], and regression [40]. Motivated by this approach, we study the following robust variant of Problem (1):

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \|F(\boldsymbol{\theta}; \boldsymbol{x}_i)\|_p + g(\boldsymbol{\theta}), \tag{2}$$

where $\|\cdot\|_p$ denotes an ℓ_p norm ($p \geq 1$). We are particularly interested in cases where F is not affine and, in general, Problem (2) is non-convex. This includes, e.g., feature selection [34], matrix factorization [13, 20], auto-encoders [18], and deep multi-target regression [40, 47].

A significant challenge behind solving Prob. (2) is that its objective is not smooth, precisely because the ℓ_p norm is not differentiable at $\mathbf{0} \in \mathbb{R}^N$. For non-convex and non-smooth problems of the form (2), where the objective contains a composite function, *Model-Based Optimization* (MBO) methods [6, 7, 9, 10, 23, 35] come with good experimental performance as well as theoretical guarantees. In particular, these MBO methods define a convex (but non-smooth) approximation of the main objective, called the *model function*. They then iteratively optimize this model function plus a proximal quadratic term. Under certain conditions, MBO converges to a stationary point of the non-convex problem [23].

In this work, we use MBO to solve Problem (2) for arbitrary ℓ_p norms. In particular, each MBO iteration results in a convex optimization problem. We solve these sub-problems using a novel stochastic variant of the Online Alternating

Direction Method (OADM) [48], which we call *Stochastic Alternating Direction Method* (SADM). Using SADM is appealing, as its resulting steps have efficient gradient-free solutions; in particular, we exploit a bisection method [25,30] for finding the proximal operator of ℓ_p norms. We provide theoretical guarantees for SADM. As an additional benefit, SADM comes with a stopping criterion, which is hard to obtain for gradient methods when the objective is non-smooth [27].

Overall, we make the following contributions:

- We study a general outlier-robust optimization that replaces the MSE with ℓ_p norms. We show that such problems can be solved via Model-Based Optimization (MBO) methods.
- We propose SADM, i.e., a stochastic version of OADM, and show that under strong convexity of the regularizer g, it converges with a $O(\log T/T)$ rate when solving the sub-problems arising at each MBO iteration.
- We conduct extensive experiments on training auto-encoders and multi-target regression. We show (a) the higher robustness of ℓ_p norms in comparison with MSE and (b) the superior performance of MBO, against stochastic gradient methods, both in terms of minimizing the objective and performing down-stream classification tasks. In some cases, we see that the MBO variant using SADM obtains objectives that are $29.6\times$ smaller than the ones achieved by the competitors.

The performance of our MBO approach is illustrated in Fig. 1. An autoencoder trained via SGD over the MSE objective is significantly affected by the presence of outliers; in contrast, our MBO methods applied to ℓ_p objectives are robust to outliers. These relative benefits are also evident in a downstream classification task over the latent embeddings. The remainder of this paper is organized as follows. We review related work in Sec. 2. We introduce our robust formulation along with its applications in Sec. 3. We describe the instance of MBO applied to our problem in Sec. 4. We introduce SADM and its convergence analysis in Sec. 5 and present our experiments in Sec. 6. We finally conclude in Sec. 7.

2 Related Work

Robustness of ℓ_p Norms: To improve the sensitivity of MSE to outliers, Ding et al. [8] first suggested replacing the MSE with the ℓ_2 norm in the context of Principal Component Analysis (PCA). This motivated a line of research for developing robust algorithms using the ℓ_2 norm in different applications, e.g., non-negative matrix factorization [20], feature selection [34,41], training autoencoders [18], and k-means clustering [11]. Attaining robustness via the ℓ_1 norm has also been used in matrix factorization [5,13,19], PCA [2,21,24], and regression [40]. Robustness of the ℓ_1 norm can be linked to robustness of median to outliers in comparison to average value (see, e.g., Friedman et al. [15]). Our problem includes robust variations considered in, e.g., [13,18,20,34,40], as special cases. However, these earlier algorithms are tailored to specific ℓ_p norms and/or do not generalize beyond the studied objective or application (some works, e.g., [34, 40], only consider convex problems). In contrast, we unify these variations for different applications as a non-convex and non-smooth problem, and present a general optimization algorithm for arbitrary ℓ_p norms.

Non-smooth/non-convex Optimization: Non-smooth and non-convex optimization problems arise in many applications, such as non-negative matrix factorization [16], compressed sensing with non-convex norms [1], and ℓ_p norm regularized sparse regression problems [3, 33]. A class of non-smooth non-convex optimization problems, known as *weakly convex problems* [45], i.e., problems in which the objective function is the sum of a convex function and a quadratic function, have attracted a lot of attention [6, 10, 12, 22, 23, 27]. Mai and Johansson [27] provided novel theoretical guarantees on the convergence of stochastic gradient descent with momentum for weakly-convex functions. However, in our experiments in Sec. 6, we show that model-based methods considerably outperform these stochastic gradient methods with momentum.

Our approach falls under the class of *prox-linear* methods [6, 9, 10, 12, 22, 23], that solve problems where the objective is a composition of a non-smooth convex function and a smooth function, exactly as in Prob. (2). Such methods iteratively minimize the composition of the non-smooth function with the first-order approximation of the smooth function [6, 10, 12, 23]. Lewis and Wright [23] prove convergence to a stationary point while Drusvyatskiy et al. prove linear convergence [9] and obtain sample complexity guarantees [10]. Ochs et al. [35, 36] generalize prox-linear methods by proposing *Model-Based Optimization* (MBO) for both smooth and non-smooth non-convex problems. MBO reduces to a prox-linear method when the objective has a composite form, as in our case. Ochs et al. further considered non-quadratic proximal penalties in sub-problems and complemented MBO with an Armijo-like line search. We leverage both their line search and theoretical guarantees (c.f. Prop. 1); our main technical departure is in solving sub-problems per iteration via SADM, which we discuss next.

ADMM. The Alternating Direction Method of Multipliers (ADMM) [4] is a convex optimization algorithm that provides efficient methods for non-smooth problems. Applying ADMM often results in sub-problems that can be solved efficiently via proximal operators [4, 39, 44]. To speed up ADMM, stochastic variants [26, 37, 49] have been proposed for minimizing sum-like objectives. These stochastic variants, similar to SGD, update solutions using the gradients of a small batch of terms in the objective, at each iteration. Another group of works proposed online variants of ADMM [17, 43, 48]. In these variants, the goal is to minimize the summation of loss functions that are revealed by an adversary.

Wang and Banerjee [48] proposed the first online variant of ADMM, termed Online Direction Method of Multipliers (OADM). Here, we propose a stochastic version of OADM, Stochastic Alternating Direction Method (SADM), to solve inner-problems in MBO iterations. SADM is similar to OADM with the difference that functions are sampled uniformly at random and are not given by an adversary. We prove that SADM converges with a $O(\log T/T)$ rate when the regularizer is strongly convex. Other existing stochastic or online ADMM variants either require a smooth objective [26, 49] or bounded sub-gradients [37, 43], neither of which apply for the inner problems we solve. In contrast, we show that applying SADM results in sub-problems that admit gradient-free efficient solutions via a bisection method for finding proximal operators of ℓ_p norms [25, 30].

3 Robust Regression and Applications

Notations. Lowercase boldface letters represent vectors, while capital boldface letters represent matrices. We also use the notation $[n] \triangleq \{1, 2, \ldots, n\}$.
Robust Regression. We first extend Prob. (2) to include constraints via:

$$\min_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i \in [n]} \|F(\boldsymbol{\theta}; \boldsymbol{x}_i)\|_p + g(\boldsymbol{\theta}) + \chi_{\mathcal{C}}(\boldsymbol{\theta}), \tag{3}$$

where, again, $F : \mathbb{R}^d \times \mathbb{R}^m \to \mathbb{R}^N$ is smooth, $|| \cdot ||_p$ is the ℓ_p norm, $g : \mathbb{R}^d \to \mathbb{R}$ is a convex regularizer such that $\inf g > -\infty$, while $\chi_{\mathcal{C}} : \mathbb{R}^d \to \{0, \infty\}$ is the indicator function of the convex set $\mathcal{C} \subseteq \mathbb{R}^d$. In practice, we are often interested in cases where either the regularizer or the constraint is absent.
Applications. For the sake of concreteness, we introduce some applications of Prob. (3). Function g is typically either the lasso (i.e., the ℓ_1 norm $g(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_1$) or ridge regularizer (i.e., the ℓ_2 norm squared $g(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_2^2$). We thus focus on the definition of $F(\cdot; \cdot)$ and constraint set $\mathcal{C}$ in each of these applications.
Auto-encoders [18]. Given n data points $\boldsymbol{x}_i \in \mathbb{R}^m$, $i \in [n]$, auto-encoders embed them in a $m'-$dimensional space, $m' \ll m$, as follows. The mapping to $\mathbb{R}^{m'}$ is done by a possibly non-linear function (e.g., a neural network) with $d_{\mathtt{enc}}$ parameters $F_{\mathtt{enc}} : \mathbb{R}^{d_{\mathtt{enc}}} \times \mathbb{R}^m \to \mathbb{R}^{m'}$, called the *encoder*. An inverse mapping, the *decoder* $F_{\mathtt{dec}} : \mathbb{R}^{d_{\mathtt{dec}}} \times \mathbb{R}^{m'} \to \mathbb{R}^m$ with $d_{\mathtt{dec}}$ parameters re-constructs the original points given latent embeddings. Both the encoder and the decoder are trained jointly over a dataset $\{\boldsymbol{x}_i\}_{i=1}^n$ by minimizing the reconstruction error; cast in our robust setting, this amounts to minimizing (3) with

$$F(\boldsymbol{\theta}; \boldsymbol{x}_i) = \boldsymbol{x}_i - F_{\mathtt{dec}}\left(\boldsymbol{\theta}_{\mathtt{dec}}; F_{\mathtt{enc}}(\boldsymbol{\theta}_{\mathtt{enc}}; \boldsymbol{x}_i)\right), \tag{4}$$

where $\boldsymbol{\theta} = [\boldsymbol{\theta}_{\mathtt{dec}}; \boldsymbol{\theta}_{\mathtt{enc}}] \in \mathbb{R}^{d_{\mathtt{enc}} + d_{\mathtt{dec}}}$ comprises the parameters of the encoder and the decoder. Robustness here aims to ameliorate the effect of outliers in the dataset $\{\boldsymbol{x_i}\}_{i=1}^n$. The constraint set can be $\mathbb{R}^d$ (i.e., the problem is unconstrained) or an ℓ_p-norm ball (i.e., $\{\theta \mid \|\theta\|_p \leq r\}$, for some $r > 0$, $p \geq 1$), when the magnitute of parameters is constrained; this can be used instead of a ℓ_1 or ℓ_2 norm regularizer. In stacked denoising autoencoders [46], the encoder and decoder are shallow and satisfy the additional constraint $\theta_{\mathtt{enc}} = \theta_{\mathtt{dec}}$.
Multi-target Regression [42]. We are given a set of n data points $\boldsymbol{x}_i \in \mathbb{R}^m$, $i \in [n]$ and the corresponding target labels $\boldsymbol{y}_i \in \mathbb{R}^{m'}$. The goal is to train a (again possibly non-linear) function $f : \mathbb{R}^d \times \mathbb{R}^m \to \mathbb{R}^{m'}$, with d parameters, to predict target values for a given vector $\boldsymbol{x} \in \mathbb{R}^m$. This maps to Prob. (3) via:

$$F(\boldsymbol{\theta}; \boldsymbol{x}_i, \boldsymbol{y}_i) = \boldsymbol{y}_i - f(\boldsymbol{\theta}; \boldsymbol{x}_i). \tag{5}$$

Robustness in this setting corresponds to ameliorating the effect of outliers in the *label* space, i.e., among labels $\{\boldsymbol{y}_i\}_{i=1}^n$. The constraint set can again be $\mathbb{R}^d$ or defined through an ℓ_p-norm ball (instead of the corresponding regularizer).

Matrix Factorization [38]. Given a matrix $\boldsymbol{X} \in \mathbb{R}^{n\times m}$, the goal is to express it a the product of two matrices $\boldsymbol{G}$,$\boldsymbol{H}$. Cast in our setting, each row $\boldsymbol{x}_i \in \mathbb{R}^m, i \in [n]$, of $\boldsymbol{X}$ is mapped to a lower dimensional sub-space as a vector $\boldsymbol{h}_i \in \mathbb{R}^{m'}$, where the sub-space basis is defined by the rows of the matrix $\boldsymbol{G} \in \mathbb{R}^{m\times m'}$. Function F is then given by $F(\boldsymbol{\theta};\boldsymbol{x}_i) = \boldsymbol{x}_i - \boldsymbol{G}\boldsymbol{h}_i$, where $\boldsymbol{\theta} = (\boldsymbol{G},\boldsymbol{H})$ and the rows of the matrix $\boldsymbol{H} \in \mathbb{R}^{n\times m'}$ are the low-dimensional embeddings $\boldsymbol{h}_i$. Robustness here limits sensitivity to outliers in rows; a similar problem can be defined in terms of robustness to outliers in columns. Beyond usual boundedness constraints, additional constraints are introduced in so-called *non-negative matrix factorization* [14,38], where matrices $\boldsymbol{G}$ and $\boldsymbol{H}$ are constrained to be non-negative.

For all three applications, we assume that F is smooth; this requires, e.g., smooth activation functions in deep models. Moreover, in all three examples, Prob. (3) is non-convex and non-smooth, as $\|\cdot\|_p$ is non-differentiable at $\mathbf{0} \in \mathbb{R}^N$.

4 Robust Regression via MBO

In this section, we outline how non-smooth, non-convex Prob. (3) can be solved via *model-based optimization* (MBO) [35]. MBO relies on the use of a model function, which is a convex approximation of the main objective. In short, the algorithm proceeds iteratively, approximating function $F(\cdot;\cdot)$ by it's 1st order Taylor expansion at each iteration. This approximation is affine in $\boldsymbol{\theta}$, and results in a convex optimization problem per iteration.

In more detail, cast into our setting, MBO proceeds as follows. Starting with a feasible solution $\boldsymbol{\theta}^0 \in \mathcal{C}$, it performs the following operations in each step $k \in \mathbb{N}$:

$$\tilde{\boldsymbol{\theta}}^k = \arg\min_{\boldsymbol{\theta}} F_{\boldsymbol{\theta}^k}(\boldsymbol{\theta}) + \frac{h}{2}\|\boldsymbol{\theta} - \boldsymbol{\theta}^k\|_2^2, \tag{6a}$$

$$\boldsymbol{\theta}^{k+1} = (1-\eta^k)\boldsymbol{\theta}^k + \eta^k\tilde{\boldsymbol{\theta}}^k, \tag{6b}$$

where $h > 0$ is a regularization parameter, $\eta^k > 0$ is a step size, and function $F_{\boldsymbol{\theta}^k} : \mathbb{R}^d \to \mathbb{R}$ is the so-called *model function* at $\boldsymbol{\theta}^k$, defined as:

$$F_{\boldsymbol{\theta}^k}(\boldsymbol{\theta}) \triangleq \frac{1}{n}\sum_{i\in[n]} \|F(\boldsymbol{\theta}^k;\boldsymbol{x}_i) + \boldsymbol{D}_{F_i}(\boldsymbol{\theta}^k)(\boldsymbol{\theta} - \boldsymbol{\theta}^k)\|_p + g(\boldsymbol{\theta}) + \chi_{\mathcal{C}}(\boldsymbol{\theta}), \tag{7}$$

where $\boldsymbol{D}_{F_i}(\boldsymbol{\theta}) \in \mathbb{R}^{N\times d}$ is the Jacobian of $F(\boldsymbol{\theta};\boldsymbol{x}_i)$ w.r.t. $\boldsymbol{\theta}$. Thus, in each step, MBO replaces F with its 1st-order Taylor approximation and minimizes the objective plus a proximal penalty; the resulting $\tilde{\boldsymbol{\theta}}^k$ is interpolated with the current solution $\boldsymbol{\theta}^k$.

The above steps are summarized in Alg. 1. The step size η^k is computed via an Armijo-type line search algorithm, which we present in detail in App. A in the extended version [31]. Moreover, the inner-step optimization via (6a) can be

Algorithm 1 Model-based Minimization (MBO)

```
1: Input: Initial solution $\boldsymbol{\theta}^0 \in \mathbf{dom}F$, iteration number $K$ set $\delta, \gamma \in (0,1)$, and $\tilde{\eta} > 0$
2: for $k \in [K]$ do
3:     $\tilde{\boldsymbol{\theta}}^k := \arg\min_{\boldsymbol{\theta}} F_{\boldsymbol{\theta}^k}(\boldsymbol{\theta}) + \frac{h}{2}\|\boldsymbol{\theta} - \boldsymbol{\theta}^k\|_2^2$
4:     Find $\gamma^k$ via Armijo search rule
5:     $\boldsymbol{\theta}^{k+1} := (1-\eta^k)\boldsymbol{\theta}^k + \eta^k\tilde{\boldsymbol{\theta}}^k$
6: end for
```

inexact; the following proposition shows asymptotic convergence of MBO to a stationary point using an inexact solver (see also App. A in [31]):

Proposition 1. *(Theorem 4.1 of [35]) Suppose $\boldsymbol{\theta}^*$ is the limit point of the sequence $\boldsymbol{\theta}^k$ generated by Alg. 1. Assume $F_{\boldsymbol{\theta}^k}(\tilde{\boldsymbol{\theta}}^k) + \frac{h}{2}\|\tilde{\boldsymbol{\theta}}^k - \boldsymbol{\theta}^k\|_2^2 - \inf_{\tilde{\boldsymbol{\theta}}} F_{\boldsymbol{\theta}^k}(\tilde{\boldsymbol{\theta}}) + \frac{h}{2}\|\tilde{\boldsymbol{\theta}} - \boldsymbol{\theta}^k\|_2^2 \leq \epsilon^k$, for all iterations k, and that $\epsilon^k \to 0$. Then $\boldsymbol{\theta}^*$ is a stationary point of Prob.* (3)*.*

For completeness, we prove Proposition 1 in App. B of the extended version [31], by showing that assumptions of Theorem 4.1 of [35] are indeed satisfied. Problem (6a) is convex but still non-smooth; we discuss how it can be solved efficiently via SADM in the next section.

5 Stochastic Alternating Direction Method of Multipliers

After dealing with convexity via MBO, there are still two challenges behind solving the constituent sub-problem (6a). The first is the non-smoothness of $\|\cdot\|_p$; the second is scaling in n, which calls for a the use of a stochastic optimization method, akin to SGD (which, however, is not applicable due to the lack of smoothness). We address both through the a novel approach, namely, SADM, which is a stochastic version of the OADM algorithm by Wang and Banerjee [48]. Most importantly, our approach reduces the solution of Prob. (6a) to several gradient-free optimization sub-steps, which can be computed efficiently. In addition, using an SADM/ADMM variant comes with clear stopping criteria, which is challenging for traditional stochastic subgradient methods [27].

5.1 SADM

We first describe how our SADM can be applied to solve Prob. (6a). We introduce the following notation to make our exposition more concise:

$$F^{(k)}(\boldsymbol{\theta}; \boldsymbol{x}_i) \triangleq \|F(\boldsymbol{\theta}^k; \boldsymbol{x}_i) + \boldsymbol{D}_{F_i}(\boldsymbol{\theta}^k)(\boldsymbol{\theta} - \boldsymbol{\theta}^k)\|_p + \frac{h}{2}\|\boldsymbol{\theta} - \boldsymbol{\theta}^k\|_2^2, \tag{8a}$$

$$F^{(k)}(\boldsymbol{\theta}) \triangleq \frac{1}{n}\sum_{i\in[n]} F^{(k)}(\boldsymbol{\theta}, \boldsymbol{x}_i), \tag{8b}$$

$$G(\boldsymbol{\theta}) \triangleq g(\boldsymbol{\theta}) + \chi_{\mathcal{C}}(\boldsymbol{\theta}). \tag{8c}$$

We can then rewrite Prob. (6a) as the following equivalent problem:

$$\text{Minimize} \quad F^{(k)}(\boldsymbol{\theta}_1) + G(\boldsymbol{\theta}_2) \tag{9a}$$

$$\text{subject to:} \quad \boldsymbol{\theta}_1 = \boldsymbol{\theta}_2, \tag{9b}$$

where $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \mathbb{R}^d$ are auxiliary variables.

Note that the objective in (9a) is equivalent to $F^{(k)}(\boldsymbol{\theta}_1) + G(\boldsymbol{\theta}_2)$. SADM starts with initial solutions, i.e., $\boldsymbol{\theta}_1^0 = \boldsymbol{\theta}_2^0 = \boldsymbol{u}^0 = 0$. At the t-th iteration, the algorithm performs the following steps:

$$\boldsymbol{\theta}_1^{t+1} := \arg\min_{\boldsymbol{\theta}_1} F^{(k)}(\boldsymbol{\theta}_1; \boldsymbol{x}_t) + \frac{\rho_t}{2}\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2^t + \boldsymbol{u}^t\|_2^2 + \frac{\gamma_t}{2}\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_1^t\|_2^2, \tag{10a}$$

$$\boldsymbol{\theta}_2^{t+1} := \arg\min_{\boldsymbol{\theta}_2} G(\boldsymbol{\theta}_2) + \frac{\rho_t}{2}\|\boldsymbol{\theta}_1^{t+1} - \boldsymbol{\theta}_2 + \boldsymbol{u}^t\|_2^2, \tag{10b}$$

$$\boldsymbol{u}^{t+1} := \boldsymbol{u}^t + \boldsymbol{\theta}_1^{t+1} - \boldsymbol{\theta}_2^{t+1}, \tag{10c}$$

where variables $\boldsymbol{x}_t$ are sampled uniformly at random from $\{\boldsymbol{x}_i\}_{i=1}^n$, $\boldsymbol{u}^t \in \mathbb{R}^d$ is the dual variable, the $\rho_t, \gamma_t > 0$ are scaling coefficients at the t-th iteration. We explain how to set ρ_t, γ_t in Thm. 1.

The solution to Problem (10b) amounts to finding the proximal operator of function G. In general, given that g is smooth and convex, this is a strongly convex optimization problem and can be solved via standard techniques. Nevertheless, for several of the practical cases we described in Sec. 3 this optimization can be done efficiently with gradient-free methods. For example, in the case where the regularizer g is the either a ridge or lasso penalty, and $\mathcal{C} = \mathbb{R}^d$, it is well-known that proximal operators for ℓ_1 and ℓ_2 norms have closed-form solutions [4]. For general ℓ_p norms, an efficient (gradient-free) bi-section method due to Liu and Ye [25] (see App. I in [31]) can be used to compute the proximal operator. Moreover, in the absence of the regularizer, the proximal operator for the indicator function $\chi_{\mathcal{C}}$ is equivalent to projection on the convex set $\mathcal{C}$. This again has closed-form solution, e.g., when $\mathcal{C}$ is the simplex [29] or an ℓ_p-norm ball [25,32]. Problem (10a) is harder to solve; we show however that it can also reduced to the (gradient-free) bisection method due to Liu and Ye [25] in the next section.

5.2 Inner ADMM

We solve Problem (10a) using another application of ADMM. In particular, note that (10a) assumes the following general form:

$$\min_{\boldsymbol{x}} \quad \|\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}\|_p + \lambda\|\boldsymbol{x} - \boldsymbol{c}\|_2^2, \tag{11}$$

where $\boldsymbol{A} = \boldsymbol{D}_{F_t}(\boldsymbol{\theta}^{(k)})$, the constituent parameter vectors are $\boldsymbol{c} = \frac{\rho_t}{\rho_t+\gamma_t+h}(\boldsymbol{\theta}_2^t - \boldsymbol{u}^t) + \frac{\gamma_t}{\rho_t+\gamma_t+h}\boldsymbol{\theta}_1^t + \frac{h}{\rho_t+\gamma_t+h}\boldsymbol{\theta}^{(k)}$, $\boldsymbol{b} = F(\boldsymbol{\theta}^{(k)}; \boldsymbol{x}_t) - \boldsymbol{D}_{F_t}(\boldsymbol{\theta}^{(k)})\boldsymbol{\theta}^{(k)}$, and $\lambda = \frac{\rho_t+\gamma_t+h}{2}$.

We solve (11) via ADMM by reformulating it as the following problem:

$$\min \quad \|\boldsymbol{y}\|_p + \lambda\|\boldsymbol{x} - \boldsymbol{c}\|_2^2 \tag{12a}$$

$$\text{s.t} \quad \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b} - \boldsymbol{y} = 0. \tag{12b}$$

The ADMM steps at the k-th iteration for (12) are the following:

$$\boldsymbol{y}^{k+1} := \arg\min_{\boldsymbol{y}} \|\boldsymbol{y}\|_p + \rho'/2\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}^k - \boldsymbol{b} + \boldsymbol{z}^k\|_2^2, \tag{13a}$$

$$\boldsymbol{x}^{k+1} := \arg\min_{\boldsymbol{x}} \lambda\|\boldsymbol{x} - \boldsymbol{c}\|_2^2 + \rho'/2\|\boldsymbol{y}^{k+1} - \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} + \boldsymbol{z}^k\|_2^2, \tag{13b}$$

$$\boldsymbol{z}^{k+1} := \boldsymbol{z}^k + \boldsymbol{y}^{k+1} - \boldsymbol{A}\boldsymbol{x}^{k+1} - \boldsymbol{b}, \tag{13c}$$

where $\boldsymbol{z}^k \in \mathbb{R}^N$ denotes the dual variable at the k-th iteration and $\rho' > 0$ is a hyper-parameter of ADMM.

Problem (13a) is again equivalent to computing the proximal operator of the ℓ_p-norm, which, as mentioned earlier, has closed-form solution for $p = 1, 2$. Moreover, for general ℓ_p-norms the proximal operator can be computed via the bisection algorithm by Liu and Ye [25]. This bisection method yields a solution with an ϵ accuracy in $O(\log_2(1/\epsilon))$ rounds [25,30] (see App. I in [31]).

5.3 Convergence

To attain the convergence guarantee of MBO given by Proposition 1, we need to solve the inner problem within accuracy ϵ^k at iteration k, where $\epsilon^k \to 0$. As our major technical contribution, we ensure this by proving the convergence of SADM when solving Prob. (6a).

Consider the sequence $\{\boldsymbol{\theta}_1^t, \boldsymbol{\theta}_2^t, \boldsymbol{u}^t\}_{t=1}^T$ generated by our SADM algorithm (10), where $\boldsymbol{x}_t$, $t \in [T]$, are sampled u.a.r. from $\{\boldsymbol{x}_i\}_{i=1}^n$. Let also

$$\bar{\boldsymbol{\theta}}_1^T \triangleq \frac{1}{T}\sum_{t=1}^T \boldsymbol{\theta}_1^t,\ \bar{\boldsymbol{\theta}}_2^T \triangleq \frac{1}{T}\sum_{t=1}^T \boldsymbol{\theta}_2^{t+1}, \tag{14}$$

denote the time averages of the two solutions. Let also $\boldsymbol{\theta}^* = \boldsymbol{\theta}_1^* = \boldsymbol{\theta}_2^*$ be the optimal solution of Prob. (9). Finally, denote by

$$R^T \triangleq F^{(k)}(\bar{\boldsymbol{\theta}}_1^T) + G(\bar{\boldsymbol{\theta}}_2^T) - F^{(k)}(\boldsymbol{\theta}^*) - G(\boldsymbol{\theta}^*) \tag{15}$$

the residual error of the objective from the optimal. Then, the following holds:

Theorem 1. *Assume that $\mathcal{C}$ is convex, closed, and bounded, while $g(0) = 0$, $g(\boldsymbol{\theta}) \geq 0$, and $g(\cdot)$ is both Lipschitz continuous and β-strongly convex over $\mathcal{C}$. Moreover, assume that both the function $F(\boldsymbol{\theta}; \boldsymbol{x}_i)$ and its Jacobian $\boldsymbol{D}_{F_i}(\boldsymbol{\theta})$ are bounded on the set $\mathcal{C}$, for all $i \in [n]$. We set $\gamma_t = ht$ and $\rho_t = \beta t$. Then,*

$$\|\bar{\boldsymbol{\theta}}_1^T - \bar{\boldsymbol{\theta}}_2^T\|_2^2 = O\left(\frac{\log T}{T}\right) \tag{16a}$$

$$\mathbb{E}[R^T] = O\left(\frac{\log T}{T}\right) \tag{16b}$$

$$\mathbb{P}\left(R^T \geq k_1\frac{\log T}{T} + k_2\frac{M}{\sqrt{T}}\right) \leq e^{-\frac{M^2}{16}} \quad \textit{for all } M > 0, T \geq 3, \tag{16c}$$

where $k_1, k_2 > 0$ are constants (see (32) in App. C in [31] for exact definitions).

We prove Theorem 1 in App. C in [31]. The theorem has the following important consequences. First, (16a) implies that the infeasibility gap between θ_1 and θ_2 decreases as $O(\frac{\log T}{T})$ *deterministically.* Second, by (16b) the residual error R^T decreases as $O(\frac{\log T}{T})$ in expectation. Finally, (16c) shows that the tail of the residual error as iterations increase is exponentially bounded. In particular, given a desirable accuracy ϵ_k, (16c) gives the number of iterations necessary be within ϵ_k of the optimal with any probability $1-\delta$. Therefore, according to Proposition 1, using SADM will result in convergence of Algorithm 1 with high probability. Finally, we note that, although we write Theorem 1 for updates using only one random sample per iteration, the analysis and guarantees readily extend to the case where a batch selected u.a.r. is used instead. A formal statement and proof can be found in App. E in the extended version [31].

6 Experiments

Algorithms. We run two variants of MBO; the first one, which we call `MBOSADM`, uses SADM (see Sec. 5) for solving the inner problems (6a). The second one, which we call `MBOSGD`, solves inner problems via a sub-gradient method. We also apply stochastic gradient descent with momentum directly to Prob. (3); we refer to this algorithm as `SGD`. This corresponds to the algorithm by [27], applied to our setting. We also solve the problem instances with an MSE objective using `SGD`, as the MSE is smooth and `SGD` is efficient in this case. Hyperparameters and implementation details are in App. F in [31]. Our code is publicly available.[1]

Applications and Datasets. We focus on two applications: training autoencoders and multi-target regression, with a ridge regularizer and $\mathcal{C} = \mathbb{R}^d$. The architectures we use are described in App. F in [31]. For autoencoders, We use `MNIST` and `Fashion-MNIST` to train autoencoders and `SCM1d` [42] for multi target regression. All three datasets, including training and test splits, are also described in App. F in [31].

Outliers. We denote the outliers ratio with $P_{\texttt{out}}$; each datapoint $\boldsymbol{x}_i$, $i \in [n]$, is independently corrupted with outliers with probability $P_{\texttt{out}}$. The probability $P_{\texttt{out}}$ ranges from 0.0 to 0.3 in our experiments. In particular, we corrupt training samples by replacing them with samples randomly drawn from a Gaussian distribution whose mean is α away from the original data and its standard deviation equals that of the original dataset. For `MNIST` and `FashionMNIST`, we set α to 1.5 times the original standard deviation, while for `SCM1d`, we set α to 2.5 times the standard deviation.

Metrics. We evaluate the solution obtained by different algorithms by using the following three metrics. The first is $F_{\texttt{OBJ}}$, the regularized objective of Prob. (3) evaluated over the training set. The other two are: $F_{\texttt{NOUTL}} \triangleq \frac{\sum_{i \notin \mathcal{S}_{\texttt{OUTL}}} \|F(\boldsymbol{\theta};\boldsymbol{x}_i)\|_p}{n-|\mathcal{S}_{\texttt{OUTL}}|}$, and $F_{\texttt{TEST}} \triangleq \frac{\sum_{i \in \mathcal{S}_{\texttt{TEST}}} \|F(\boldsymbol{\theta};\boldsymbol{x}_i)\|_p}{|\mathcal{S}_{\texttt{TEST}}|}$, where $\mathcal{S}_{\texttt{OUTL}}$, $\mathcal{S}_{\texttt{TEST}}$ are the outlier and test sets, respectively. Metric $F_{\texttt{NOUTL}}$ measures the robustness of algorithms w.r.t. outliers;

[1] https://github.com/neu-spiral/ModelBasedOptimization

Table 1: Time and Objective Performance. We report objective and time metrics for under different outlier ratios and different p-norms. We observe from the table that `MBOSADM` significantly outperforms other competitors in terms of objective metrics. In terms of running time, `SGD` is generally fastest, due to fast gradient updates. However, we see that the time MBO variants take to get to the same or better objective value (i.e., T^*), ware comparable to running time of `SGD`.

		`MBOSADM`					`MBOSGD`					`SGD`			
P_{out}	p	$F_{\texttt{NOUTL}}$	$F_{\texttt{OBJ}}$	$F_{\texttt{TEST}}$	T(h)	T^*(h)	$F_{\texttt{NOUTL}}$	$F_{\texttt{OBJ}}$	$F_{\texttt{TEST}}$	T(h)	T^*(h)	$F_{\texttt{NOUTL}}$	$F_{\texttt{OBJ}}$	$F_{\texttt{TEST}}$	T(h)
`MNIST`															
0.0	2.0	**2.50**	**2.51**	**2.50**	5.69	0.14	8.08	8.08	8.12	64.17	6.47	9.21	9.22	9.30	9.83
0.0	1.5	**2.63**	**2.63**	**2.63**	11.67	0.79	20.19	20.20	20.39	65.67	59.98	20.35	20.36	20.57	14.71
0.0	1.0	**3.46**	**3.47**	**3.44**	17.82	3.09	102.79	102.80	104.24	81.53	NA	102.44	102.46	103.89	11.50
0.05	2.0	**3.48**	**5.35**	**3.46**	6.33	0.31	3.89	6.36	3.86	54.92	38.31	8.03	12.52	8.09	13.96
0.05	1.5	**4.10**	**9.74**	**4.08**	45.32	2.08	5.86	11.70	5.82	57.03	25.12	20.34	34.69	20.57	14.60
0.05	1.0	**5.23**	**20.20**	**5.24**	44.03	5.61	27.68	73.53	27.56	32.76	9.67	102.40	236.43	103.90	11.70
0.1	2.0	4.27	**7.77**	4.23	11.67	1.34	**3.56**	7.83	**3.54**	64.20	33.70	7.02	11.64	7.04	13.97
0.1	1.5	**4.18**	**11.84**	**4.17**	68.74	0.29	5.50	13.77	5.45	67.04	9.88	20.34	48.79	20.57	14.04
0.1	1.0	**5.90**	**36.02**	**5.92**	37.73	8.20	30.08	109.79	30.16	39.77	6.72	102.36	368.22	103.90	11.81
0.2	2.0	4.07	8.97	4.04	51.69	4.39	**3.54**	**8.23**	**3.52**	57.08	19.19	7.48	16.44	7.51	14.25
0.2	1.5	**3.90**	**11.58**	**3.89**	195.69	1.56	7.00	20.63	6.95	45.46	6.44	20.36	77.78	20.59	15.06
0.2	1.0	**3.85**	**28.25**	**3.83**	36.98	5.15	40.12	224.47	40.11	19.71	2.13	102.37	639.32	103.90	8.25
0.3	2.0	**3.99**	14.21	**3.98**	9.83	4.93	4.02	**10.55**	3.99	55.60	24.14	7.46	20.63	7.48	13.53
0.3	1.5	20.55	**22.56**	20.78	159.92	39.24	**7.22**	24.30	**7.16**	42.65	15.09	23.90	58.52	23.89	16.25
0.3	1.0	102.70	**99.36**	104.27	51.48	0.87	**56.60**	438.89	**56.17**	20.48	3.32	102.34	910.68	103.90	8.52
`Fashion-MNIST`															
0.0	2.0	**3.51**	**3.51**	**3.51**	4.33	0.31	5.01	5.01	5.01	42.13	14.80	8.72	8.73	8.70	9.78
0.0	1.5	**6.13**	**6.14**	**6.14**	14.35	2.18	8.87	8.88	8.89	63.39	12.80	22.62	22.63	22.56	14.70
0.0	1.0	**10.59**	**10.61**	**10.56**	29.69	2.63	41.24	41.26	41.35	50.41	3.32	224.26	224.28	224.89	9.72
0.05	2.0	**3.80**	**5.82**	**3.80**	14.70	1.40	4.53	6.75	4.54	67.29	19.15	8.30	11.98	8.27	9.71
0.05	1.5	**7.38**	14.57	**7.40**	96.73	2.56	7.91	**11.97**	7.93	64.25	16.16	20.88	27.22	20.83	10.94
0.05	1.0	**16.64**	**30.51**	**16.68**	43.55	16.04	65.01	109.94	65.31	29.60	9.70	158.65	227.48	158.27	9.97
0.1	2.0	**4.05**	**6.73**	**4.06**	14.66	2.35	4.28	7.90	4.29	65.06	16.46	8.96	14.35	8.94	13.67
0.1	1.5	11.08	32.69	11.10	20.07	NA	**8.46**	**15.23**	**8.49**	65.78	9.92	17.98	31.41	17.95	10.63
0.1	1.0	**9.79**	**27.96**	**9.81**	35.50	2.43	58.70	126.18	58.90	45.06	2.08	235.02	452.45	234.49	13.32
0.2	2.0	6.07	10.19	6.08	14.25	3.67	**4.77**	**9.28**	**4.77**	69.84	30.16	5.71	14.34	5.71	9.31
0.2	1.5	28.51	53.69	28.49	39.42	NA	**10.94**	**27.12**	**10.97**	39.11	16.09	19.36	42.97	19.36	10.87
0.2	1.0	**10.50**	**27.95**	**10.50**	94.72	6.57	140.00	390.02	140.08	17.03	3.33	204.88	644.99	205.13	14.72
0.3	2.0	6.63	23.18	6.63	32.87	NA	**5.84**	**13.04**	**5.85**	50.52	29.95	7.45	20.12	7.46	13.65
0.3	1.5	**7.08**	**22.51**	**7.10**	86.27	30.02	11.09	24.73	11.12	52.41	12.08	19.52	58.26	19.56	11.05
0.3	1.0	**14.43**	**50.91**	**14.46**	95.08	19.48	404.77	893.56	404.52	9.51	NA	410.82	522.50	411.84	10.74
`SCMD1d`															
0.0	2.0	2.88	2.88	3.02	1.82	0.12	**2.85**	**2.85**	**2.99**	0.36	0.04	3.62	3.63	3.72	1.37
0.0	1.5	4.23	4.24	**4.39**	7.22	0.43	**4.22**	**4.23**	4.44	0.36	0.04	5.47	5.47	5.60	1.58
0.0	1.0	**9.78**	**9.79**	**10.18**	7.13	0.47	9.86	9.86	10.32	0.37	0.04	12.95	12.95	13.25	1.22
0.05	2.0	2.88	3.13	2.99	2.52	0.13	**2.86**	**3.11**	3.00	0.54	0.05	3.64	3.89	3.71	1.31
0.05	1.5	4.23	4.61	**4.37**	10.23	4.61	**4.22**	**4.59**	4.46	0.50	0.05	5.50	5.87	5.61	1.23
0.05	1.0	**9.69**	**10.52**	**10.09**	0.59	0.18	9.86	10.66	10.35	0.51	0.05	13.03	13.87	13.29	1.17
0.1	2.0	2.90	3.41	3.01	2.22	0.13	**2.84**	**3.34**	**3.00**	0.46	0.05	3.63	4.12	3.69	1.30
0.1	1.5	4.23	4.99	4.42	9.42	0.68	**4.18**	**4.90**	**4.40**	0.50	0.05	5.52	6.11	5.62	1.15
0.1	1.0	**9.56**	**10.99**	**10.18**	9.92	0.78	9.77	11.11	10.54	0.54	0.07	13.09	13.72	13.32	1.11
0.2	2.0	2.93	3.90	3.03	1.83	0.95	**2.86**	**3.79**	**3.02**	0.5	0.3	3.63	3.97	3.66	1.15
0.2	1.5	4.23	5.60	**4.37**	8.17	3.60	**4.21**	**5.48**	4.47	0.36	0.2	5.50	5.83	5.56	1.17
0.2	1.0	**9.46**	**11.00**	**10.04**	6.55	1.50	9.82	11.30	10.60	0.45	0.20	13.09	13.38	13.28	1.11
0.3	2.0	2.93	4.32	3.03	1.80	NA	**2.85**	4.10	**3.03**	0.46	NA	3.61	**3.90**	3.64	1.18
0.3	1.5	4.25	6.05	**4.44**	8.19	NA	**4.21**	5.73	4.49	0.50	NA	5.43	**5.69**	5.43	1.18
0.3	1.0	**9.53**	11.07	**10.00**	6.44	2.72	9.68	**11.05**	10.32	0.51	0.28	12.95	13.13	12.96	1.11

ideally, $F_{\texttt{NOUTL}}$ should remain unchanged as the fraction of outliers increases. Metric $F_{\texttt{TEST}}$ evaluates the generalization ability of algorithms on unseen (test) data, which also does not contain outliers; ideally, $F_{\texttt{TEST}}$ be similar $F_{\texttt{NOUTL}}$. Moreover, we report total running time (T) of all algorithms. For the two variants of MBO, we additionally report the time (T^*) until the they reach the optimal value attained by `SGD` (N/A if never reached). Finally, for autoencoders, we also use dataset labels to train a logistic regression classifier over latent embeddings, and also report the prediction accuracy on the test set. Classifier hyperparameters are described in App. G in [31].

6.1 Time and Objective Performance Comparison

We evaluate our algorithms w.r.t. both objective and time metrics, which we report for different outlier ratios $P_{\texttt{out}}$ and p-norms in Table 1. By comparing objective metrics, we see that MBOSADM and MBOSGD significantly outperform SGD. SGD achieves a better $F_{\texttt{OBJ}}$ in only 2 out of 45 cases, i.e., SCM1d dataset for $p = 1.5, 2$ and $P_{\texttt{out}} = 0.3$; however, even for these two cases, MBOSADM and MBOSGD obtain better $F_{\texttt{NOUTL}}$ and $F_{\texttt{TEST}}$ values. In terms of overall running time T, SGD is generally faster than MBOSADM and MBOSGD; this is expected, as each iteration of SGD only computes the gradient of a mini-batch of terms in the objective, while the other methods need to solve an inner-problem. Nonetheless, by comparing T^*, we see that the MBO variants obtain the same or better objective as SGD in a comparable time. In particular, T^* is less than T for SGD in 33 and 15 cases (out of 45) for MBOSADM and MBOSGD, respectively.

Comparing the performance between MBOSADM and MBOSGD, we first note that MBOSADM has a superior performance w.r.t. all three objective metrics for 25 out of 45 cases. In some cases, MBOSADM obtains considerably smaller objective values; for example, for MNIST and $P_{\texttt{out}} = 0.0, p = 1$, $F_{\texttt{NOUT}}$ is 0.03 of the value obtained by MBOSGD (also see Figures 2c and 2f). However, it seems that in the high-outlier setting $P_{\texttt{out}} = 0.3$ the performance of MBOSADM deteriorates; this is mostly due to the fact that the high number of outliers adversely affects the convergence of SADM and it takes more iterations to satisfy the desired accuracy.

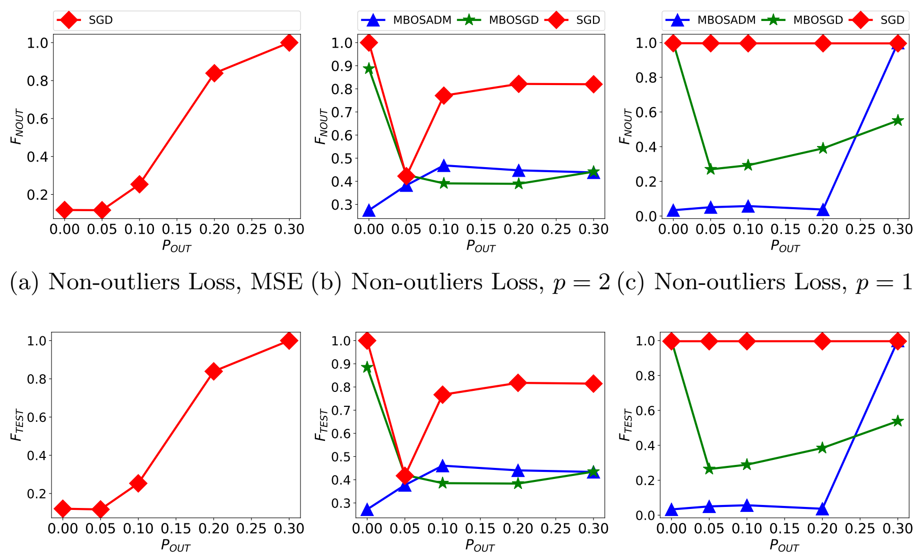
6.2 Robustness Analysis

We further study the robustness of different p-norms and MSE to the presence of outliers. For brevity, we only report results for MNIST and for $p = 1, 2$, and MSE. For more results refer to Fig. 4a in App. H in [31]. We show the scaling of $F_{\texttt{NOUT}}$ and $F_{\texttt{TEST}}$ w.r.t. the fraction $P_{\texttt{out}}$ in Fig. 4b, for different norms. To make comparisons between different objectives interpretable, we normalize all values in each figure by the largest value in that figure.

By comparing Figures 2a and 2d, corresponding to MSE, with other plots in Fig. 2, we see that the loss values considerably increase by adding outliers. For other p-norms, we see that SGD generally stays unchanged, w.r.t. outliers. However, the loss for SGD is higher than MBO variants. Loss values for MBOSGD also do not increase significantly by adding outliers. Moreover, we see that, when no outliers are present $P_{\texttt{out}} = 0.0$, MBOSGD obtains higher loss values. MBOSADM generally achieves the lowest loss values and these values again do not increase with increasing $P_{\texttt{out}}$; however, for the highest outliers ($P_{\texttt{out}} = 0.3$), the performance of MBOSADM is considerably worse for $p = 1$. As we emphasize in Sec. 6.1, high number of outliers adversely affects the convergence of SADM, and hence the poor performance of MBOSADM for $P_{\texttt{out}} = 0.3$.

6.3 Classification Performance

Fig. 3 shows the quality of the latent embeddings obtained by different trained autoencoders on the downstream classification over MNIST and FashionMNIST.



(a) Non-outliers Loss, MSE (b) Non-outliers Loss, $p = 2$ (c) Non-outliers Loss, $p = 1$

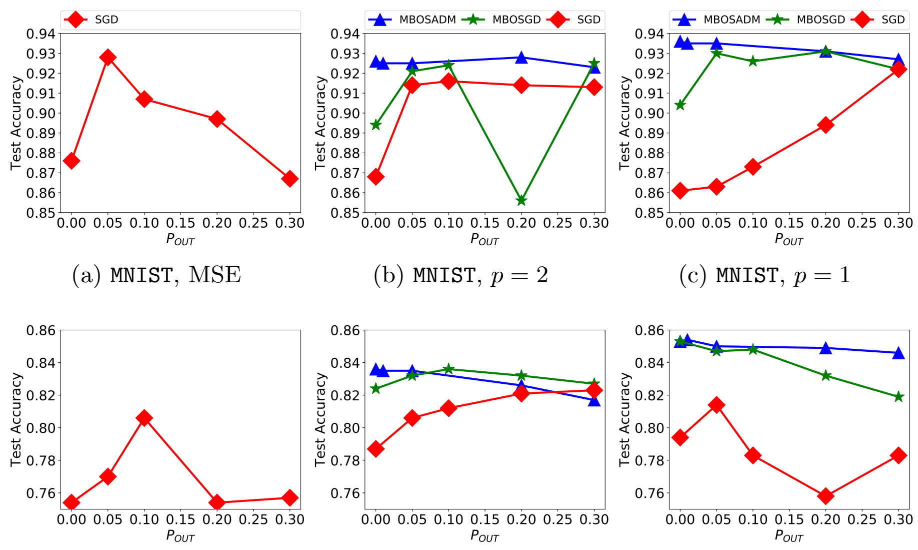
(d) Test Loss, MSE (e) Test Loss, $p = 2$ (f) Test Loss, $p = 1$

Fig. 2: A comparison of scalability of the non-outlioers loss $F_{\texttt{NOUT}}$ and the test loss $F_{\texttt{TEST}}$ for different p-norms, w.r.t., outliers fraction $P_{\texttt{out}}$. We normalize values in each figure by the largest observed value, to make comparisons between different objectives possible. We see that MSE in Figures 2a and 2 are drastically affected by outliers and scale with outliers fraction $P_{\texttt{out}}$. Other ℓ_p norms for different methods in Figures 2b, 2c, 2e, and 2f generally stay unchanged w.r.t. $P_{\texttt{out}}$. However, `MBOSADM` in the high outlier regime and $p = 1$ performs poorly.

Additional results are shown in Tables 2 and 3 in App. G. We see that MBO variants again outperform `SGD`. For `MNIST`, (reported in Figures 3a to 3c), we see that `MBOSADM` for $p = 1$ obtains the highest accuracy. Moreover, for `Fashion-MNIST` (reported in Fig. 3d to 3f), we observe that again `MBOSADM` for $p = 1$ outperforms other methods. We also observe that MSE (reported in Figures 3a and 3d) is sensitive to outliers; the corresponding accuracy drastically drops for $P_{\texttt{out}} \geq 0.1$. An interesting observation is that adding outliers improves the performance of `SGD`; however, we see that `SGD` always results in lower accuracy, except in two cases ($P_{\texttt{out}} = 0.2$ in Fig. 3b and $P_{\texttt{out}} = 0.3$ in Fig. 3e).

7 Conclusion

We present a generic class of robust formulations that includes many applications, i.e., auto-encoders, multi-target regression, and matrix factorization. We show that SADM, in combination with MBO, provides efficient solutions for our class of robust problems. Studying other proximal measures described by Ochs et al. [35] is an open area. Moreover, characterizing the sample complexity of our



(a) MNIST, MSE (b) MNIST, $p = 2$ (c) MNIST, $p = 1$

(d) Fashion-MNIST, MSE (e) Fashion-MNIST, $p = 2$ (f) Fashion-MNIST, $p = 1$

Fig. 3: Classification performance for different methods and datasets. We use the embeddings obtained by auto-encoders trained via different algorithms to train a logistic regression model for classification. We generally observe that MBOSADM results in higher accuracy on the test sets. Moreover, we see that MSE is evidently sensitive to outliers, see Figures 3a and 3d for $P_{\text{out}} \geq 0.2$.

proposed method for obtaining a stationary point, as in MBO variants that use gradient methods [6, 10], is an interesting future direction.

References

1. Attouch, H., Bolte, J., Redont, P., Soubeyran, A.: Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the Kurdyka-Lojasiewicz inequality. Mathematics of operations research **35**(2), 438–457 (2010)
2. Baccini, A., Besse, P., Falguerolles, A.: A l1-norm PCA and a heuristic approach. Ordinal and symbolic data analysis **1**(1), 359–368 (1996)
3. Blumensath, T., Davies, M.E.: Iterative hard thresholding for compressed sensing. Applied and computational harmonic analysis **27**(3), 265–274 (2009)
4. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends® in Machine learning **3**(1), 1–122 (2011)
5. Croux, C., Filzmoser, P.: Robust factorization of a data matrix. In: COMPSTAT. pp. 245–250. Springer (1998)
6. Davis, D., Drusvyatskiy, D.: Stochastic model-based minimization of weakly convex functions. SIAM Journal on Optimization **29**(1), 207–239 (2019)

7. Davis, D., Grimmer, B.: Proximally guided stochastic subgradient method for nonsmooth, nonconvex problems. SIAM Journal on Optimization **29**(3), 1908–1930 (2019)
8. Ding, C., Zhou, D., He, X., Zha, H.: R1-PCA: rotational invariant l 1-norm principal component analysis for robust subspace factorization. In: ICML (2006)
9. Drusvyatskiy, D., Lewis, A.S.: Error bounds, quadratic growth, and linear convergence of proximal methods. Mathematics of Operations Research **43**(3), 919–948 (2018)
10. Drusvyatskiy, D., Paquette, C.: Efficiency of minimizing compositions of convex functions and smooth maps. Mathematical Programming **178**(1), 503–558 (2019)
11. Du, L., Zhou, P., Shi, L., Wang, H., Fan, M., Wang, W., Shen, Y.D.: Robust multiple kernel k-means using l21-norm. In: IJCAI (2015)
12. Duchi, J.C., Ruan, F.: Stochastic methods for composite and weakly convex optimization problems. SIAM Journal on Optimization **28**(4), 3229–3259 (2018)
13. Eriksson, A., Van Den Hengel, A.: Efficient computation of robust low-rank matrix approximations in the presence of missing data using the l1 norm. In: CVPR (2010)
14. Févotte, C., Idier, J.: Algorithms for nonnegative matrix factorization with the β-divergence. Neural computation **23**(9), 2421–2456 (2011)
15. Friedman, J., Hastie, T., Tibshirani, R.: The Elements of Statistical Learning, vol. 1. Springer Series in Statistics New York (2001)
16. Gillis, N.: Nonnegative Matrix Factorization. SIAM - Society for Industrial and Applied Mathematics, Philadelphia, PA (2020)
17. Hosseini, S., Chapman, A., Mesbahi, M.: Online distributed ADMM via dual averaging. In: CDC (2014)
18. Jiang, W., Gao, H., Chung, F.l., Huang, H.: The l2,1-norm stacked robust autoencoders for domain adaptation. In: AAAI (2016)
19. Ke, Q., Kanade, T.: Robust l1 factorization in the presence of outliers and missing data by alternative convex programming. In: CVPR (2005)
20. Kong, D., Ding, C., Huang, H.: Robust nonnegative matrix factorization using l21-norm. In: CIKM (2011)
21. Kwak, N.: Principal component analysis based on l1-norm maximization. IEEE transactions on pattern analysis and machine intelligence **30**(9), 1672–1680 (2008)
22. Le, H., Gillis, N., Patrinos, P.: Inertial block proximal methods for non-convex non-smooth optimization. In: ICML (2020)
23. Lewis, A.S., Wright, S.J.: A proximal method for composite minimization. Mathematical Programming **158**(1), 501–546 (2016)
24. Li, X., Pang, Y., Yuan, Y.: l1-norm-based 2DPCA. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **40**(4), 1170–1175 (2010)
25. Liu, J., Ye, J.: Efficient l1/lq NormRregularization. arXiv preprint arXiv:1009.4766 (2010)
26. Liu, Y., Shang, F., Cheng, J.: Accelerated variance reduced stochastic admm. In: AAAI (2017)
27. Mai, V., Johansson, M.: Convergence of a stochastic gradient method with momentum for non-smooth non-convex optimization. In: ICML (2020)
28. Mehta, J., Gupta, K., Gogna, A., Majumdar, A., Anand, S.: Stacked robust autoencoder for classification. In: NeurIPS (2016)
29. Michelot, C.: A finite algorithm for finding the projection of a point onto the canonical simplex of n. Journal of Optimization Theory and Applications (1986)
30. Moharrer, A., Gao, J., Wang, S., Bento, J., Ioannidis, S.: Massively distributed graph distances. IEEE Transactions on Signal and Information Processing over Networks **6**, 667–683 (2020)

31. Moharrer, A., Kamran, K., Yeh, E., Ioannidis, S.: Robust regression via model based methods. arXiv preprint arXiv:2106.10759 (2021)
32. Moreau, J.J.: Décomposition orthogonale d'un espace hilbertien selon deux cônes mutuellement polaires. Comptes rendus hebdomadaires des séances de l'Académie des sciences **255**, 238–240 (1962)
33. Natarajan, B.K.: Sparse approximate solutions to linear systems. SIAM journal on computing **24**(2), 227–234 (1995)
34. Nie, F., Huang, H., Cai, X., Ding, C.H.: Efficient and robust feature selection via joint l2,1-norms minimization. In: NIPS (2010)
35. Ochs, P., Fadili, J., Brox, T.: Non-smooth non-convex bregman minimization: Unification and new algorithms. Journal of Optimization Theory and Applications **181**(1), 244–278 (2019)
36. Ochs, P., Malitsky, Y.: Model function based conditional gradient method with Armijo-like line search. In: Proceedings of the 36th International Conference on Machine Learning (2019)
37. Ouyang, H., He, N., Tran, L., Gray, A.: Stochastic alternating direction method of multipliers. In: International Conference on Machine Learning. pp. 80–88. PMLR (2013)
38. Paatero, P., Tapper, U.: Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. Environmetrics **5**(2), 111–126 (1994)
39. Peng, Y., Ganesh, A., Wright, J., Xu, W., Ma, Y.: Rasl: Robust alignment by sparse and low-rank decomposition for linearly correlated images. IEEE transactions on pattern analysis and machine intelligence **34**(11), 2233–2246 (2012)
40. Pesme, S., Flammarion, N.: Online robust regression via sgd on the l1 loss. In: NeurIPS (2020)
41. Qian, M., Zhai, C.: Robust unsupervised feature selection. In: IJCAI (2013)
42. Spyromitros-Xioufis, E., Tsoumakas, G., Groves, W., Vlahavas, I.: Multi-target regression via input space expansion: treating targets as inputs. Machine Learning **104**(1), 55–98 (2016)
43. Suzuki, T.: Dual averaging and proximal gradient descent for online alternating direction multiplier method. In: ICML (2013)
44. Tao, M., Yuan, X.: Recovering low-rank and sparse components of matrices from incomplete and noisy observations. SIAM Journal on Optimization **21**(1), 57–81 (2011)
45. Vial, J.P.: Strong and weak convexity of sets and functions. Mathematics of Operations Research **8**(2), 231–259 (1983)
46. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.A.: Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. Journal of machine learning research **11**(Dec), 3371–3408 (2010)
47. Waegeman, W., Dembczyński, K., Hüllermeier, E.: Multi-target prediction: a unifying view on problems and methods. Data Mining and Knowledge Discovery **33**(2), 293–324 (2019)
48. Wang, H., Banerjee, A.: Online alternating direction method. In: ICML (2012)
49. Zheng, S., Kwok, J.T.: Fast-and-light stochastic ADMM. In: IJCAI. pp. 2407–2613 (2016)