

Robust Stochastic Optimization: Robust Gradient Descent vs SEVER

Ian Wang
CPSC 561 - Professor Zampetakis

April 26, 2025

Abstract

Classical statistical estimators rely on strict model assumptions that rarely hold in practice. This limitation has motivated the development of robust statistical methods that provide reliable results, even when underlying data distributions deviate from assumptions. This project explores and compares two different approaches to robust stochastic optimization.

[PSBR18] introduces a robust gradient descent (RGD) approach that hardens each gradient step by replacing the usual sample mean of gradients with a robust gradient estimator. RGD is computationally tractable and robust to both Huber ϵ -contamination and heavy-tailed noise, and is applicable over a broad class of convex optimization problems. In contrast, [DKK⁺19]’s SEVER algorithm only applies robust gradient estimation at critical points in the optimization process, rather than at every gradient step. SEVER first runs a base learner, filters out outlier points based on gradient behavior, and then repeats until convergence.

Although both methods achieve robustness by hardening gradient estimation, the two methods also represent two distinct philosophies of robust learning—RGD hardens the optimizer while SEVER removes corrupted data. This project aims to provide a unified study that compares the algorithmic details, theoretical guarantees, and empirical results for both methods. Special attention will be given to the strengths and weaknesses of each approach and the assumptions that each approach requires. Additionally, because of the orthogonal nature of the two algorithms, I also test a hybrid framework that combines both methodologies on synthetic data.

1 Introduction

Practical applications of machine learning typically involve unclean datasets that may contain corrupted data, but most classical learners are sensitive to even a small fraction of outliers. Consider the traditional empirical risk minimization (ERM) framework. The target parameter θ^* is defined through the population risk expectation:

$$\theta^* = \arg \min_{\theta \in \Theta} R(\theta) = \arg \min_{\theta \in \Theta} \mathbb{E}_{z \sim P^*} [\ell(\theta, z)] \quad (1)$$

But under ERM, $\hat{\theta}$ is estimated by replacing the population risk with the empirical one:

$$\hat{\theta} = \arg \min_{\theta \in \Theta} \mathbb{E}_{z_n} [\hat{\ell}(\theta, z_n)] \quad (2)$$

Importantly, this ERM framework relies on the z_n being iid from distribution P and can fail dramatically under slight model deviations. Further, many of the statistical guarantees from ERM depend on assumptions like sub-Gaussianity or non-heavy tails.

As a result, the problem of developing robust estimators has been thoroughly studied and many methods, such as RANSAC [FB81] or Huberization- [Owe07], have been developed. However, these methods either fail in high dimensions or can only handle obvious outliers that clearly stick out from the rest of the data. Other methods with stronger robustness guarantees are based on computationally intractable approaches like non-convex optimizers [Hub81], tournament algorithms [DG85], or metrics of depth [Miz02].

[PSBR18] and [DKK⁺19] introduce two different learning methods that are both (a) computationally tractable, and (b) robust to outliers in high-dimensional settings. Interestingly, both methods start from the same key insights. While previous robust methods attempt to replace the empirical risk (2) with a robust one (which tends to be computationally intractable), [PSBR18] and [DKK⁺19] find robust algorithms for directly optimizing the population risk. Further, both papers recognize that estimating the gradient of the population risk can be viewed as a mean estimation problem. Then, by applying previous work on robust mean estimation, they can develop robust population risk optimization algorithms.

2 Definitions

Robustness is defined under the ϵ -contamination model and heavy-tail distributions. Informally, a robust estimator is probably almost correct (PAC), even in the presence of model deviations like ϵ -contamination or heavy-tails.

Definition 1. (ϵ -contamination). Observations are sampled from $P_\epsilon = (1 - \epsilon)P + \epsilon Q$ where Q can be any arbitrary distribution.

Definition 2. (Heavy-tailed). A distribution is heavy tailed if it has some infinite moments. Requiring finite n -th moment is a stronger restriction than requiring finite m -th moment for all $n < m$.

[PSBR18]’s algorithm relies on gradient estimators. Informally, these are PAC estimators of the gradient of the population risk.

Definition 3. (gradient estimator). $g(\theta, z, \delta)$ is a gradient estimator if $\|g(\theta, z, \delta) - \nabla \mathcal{R}(\theta)\|_2 \leq \alpha(n, \delta)\|\theta - \theta^*\|_2 + \beta(n, \delta)$ with probability at least $1 - \delta$

[PSBR18] derives constraints on $\alpha(n, \delta)$ and $\beta(n, \delta)$ for different contexts (e.g. linear regression under ϵ -contamination).

[DKK⁺19]’s algorithm relies on γ -approximate critical points, and correspondingly, γ -approximate learners.

Definition 4. (γ -approximate critical point). $w \in \mathcal{H}$ is a γ -approximate critical point of $f : \mathcal{H} \rightarrow \mathbb{R}$ if for all unit vectors v and for all $\delta > 0$ where $w + \delta v \in \mathcal{H}$, we have that $v \cdot \nabla f(w) \geq -\gamma$

Definition 5. (γ -approximate learner). A learning algorithm $\mathcal{L}$ is γ -approximate if for any functions $f_1, \dots, f_n : \mathcal{H} \rightarrow \mathbb{R}$ that are all bounded below, $\mathcal{L}$ returns a γ -approximate critical point of $\bar{f}(x)$.

More intuitively, for our purposes, each f is the loss function for a data point. A γ -approximate critical point is one where f decreases at a rate of at most γ for small deviations. Most common learning algorithms like stochastic gradient descent are γ -approximate learners.

Now, we have the tools we need to explain each algorithm.

3 Robust Gradient Descent

[PSBR18] introduces Robust Gradient Descent (RGD), a variant of gradient descent (GD) that is robust under ϵ -contamination and heavy-tailed settings. Like GD, RGD works by incrementally updating the parameter estimates. Notably, RGD doesn't seek to optimize a new, hardened loss function. Instead, RGD uses robust estimates of the population gradient by treating it as a mean estimation problem. Informally, the algorithm is as follows:

1. Split samples into T subsets
2. For $t = 0, \dots, T - 1$: calculate θ_{t+1} as $\theta_t - \eta g(\theta_t, z, \delta)$ where g is a robust gradient estimator

The specific g that is used can vary. For ϵ -contamination settings, [PSBR18] uses the robust estimator proposed in [LRV16] as it makes weak moment assumptions and works well in high dimensions. The key insight behind [LRV16]'s robust mean estimator is that (a) robust estimation in one-dimension is easy, and (b) if the direction in which corrupted data shifts the mean can be estimated, then mean estimation in high dimensions can be reduced to one-dimensional mean estimation. More specifically, once the gradient shift direction is identified, we can apply robust mean estimation along that direction and a non-robust sample mean estimation in the orthogonal directions. Inspired by this, [PSBR18] develop the Huber Gradient Estimator. Informally, the algorithm is as follows:

1. Find the smallest ball that contains $\sim 1 - \epsilon$ of the data points S and only keep the data points that are inside
2. If the current dimension p is 1, return the mean of the current data points
3. Else, let V be the subspace spanned by the top $p/2$ principal components of the covariance matrix and W be its complement
4. Let $\hat{\mu}_V$ be the result of recursing the Huber Gradient Estimator on the data points projected onto V

5. Let $\mu_{\hat{W}}$ be the mean of the current data points projected onto W
6. Return the sum of $\mu_{\hat{V}} + \mu_{\hat{W}}$

For heavy-tailed settings, [PSBR18] uses the robust estimator proposed by [Min15]. This estimator extends the classical median-of-means estimator to multivariate distributions. Informally, this estimator works as follows:

1. Split the data points and estimate the sample mean for each of the splits
2. Return the geometric median of the sample means

To limit the scope of this project, I will focus on the ϵ -contamination setting for my experiments.

4 SEVER

[DKK⁺19] introduces SEVER, a meta-algorithm that wraps around a base γ -approximate learner and filters out outliers over multiple rounds. Informally, each round of the algorithm proceeds as following:

1. The base learner finds a γ -approximate critical point w from the current data points
2. Calculate matrix G by stacking the centered gradients for the current data points:
 $G = [\nabla f_i(w) - \text{avg}(\nabla f_i(w))]$
3. Calculate v , the top right singular vector of G
4. Compute the outlier score τ_i for each data point by the square of the projection of the centered gradient on v : $\left((\nabla f_i(w) - \text{avg}(\nabla f_i(w))) \cdot v \right)^2$
5. Filter out the data points with the largest outlier scores
6. If no data points were filtered, return w . Else, repeat.

For statistical guarantees, [DKK⁺19] uses the following filtering algorithm.

1. If $\sum_i \tau_i \leq C$ for some parameter C , don't filter out any data points
2. Else, sample T from the uniform distribution $[0, \max(\text{outlierscore})]$ and return the data points with $\tau_i < T$

The intuition for SEVER's filtering algorithm is that for outliers to have a significant effect on $\hat{\theta}$, the gradients of the outliers should be large and aligned in the same direction. This is very similar to the insight in [LRV16]. Such an effect should result in a large singular value in the singular value decomposition of the gradients matrix. Hence, SEVER filters out the points whose gradient aligns most strongly in this direction.

[DKK⁺19] also note that empirically, good results can be obtained if SEVER is run for a fixed number of rounds where the top p fraction of outliers are removed with each filter. This is the version of SEVER that they—and I—choose to implement in our experiments.

5 RGD vs SEVER

At a high level, both RGD and SEVER utilize the same building blocks: robust mean estimation and a base optimizer. However, they differ in how they combine the two. Whereas RGD applies robust mean estimation to estimate the population gradient at every step of the base optimizer, SEVER only applies robust mean estimation after the base optimizer has converged to a γ -approximate critical point. Intuitively, SEVER can be thought of a version of RGD where many steps of gradient descent are applied between rounds of robust mean estimation. However, this framework also comes with the additional benefit that SEVER doesn't only work with step-based gradient descent optimizers, but also any learner that finds γ -approximate critical point. [PSBR18] discusses the use of plugin estimators, but notes they have weaker statistical guarantees than with gradient descent methods.

Another important difference is that SEVER applies a filter-based robust estimator, by which outliers are explicitly identified and filtered out. On the other hand, RGD's uses various robust estimators, but they all directly estimate the population gradient without removing outliers. This difference also reflects the two different philosophies. Because SEVER only applies robust mean estimation between rounds, it must introduce robustness by improving the data quality that is given to the base learner. However, RGD repeatedly computes robust gradients at every step, so it is better to keep all the data.

5.1 Theoretical Guarantees

To reduce the scope of this project, I will limit my comparison of theoretical guarantees to the linear regression setting. Note, these theoretical guarantees are based on the underlying robust mean estimators each paper uses, and as better robust mean estimators are developed, both SEVER and RGD will yield better results

In the ϵ -contamination setting, RGD's Huber Gradient Estimator requires that the covariates have bounded 8-th moments and the noise to have bounded 4-th moments. In the heavy tailed setting, with the geometric median of means estimator, this is relaxed so that the covariates just need bounded 4-th moments and the noise just needs bounded 2nd moments. SEVER also requires the uncorrupted covariates to be bounded in the 4-moment and the noise to be bounded in the 2nd moment, but operates in both the ϵ -contamination and heavy tailed settings. Thus, SEVER is slightly better in terms of moment restrictions.

In the ϵ -contamination setting, RGD requires $n > \frac{\tau_u^2}{\tau_l^2} T d \log(T/\delta)$ samples to compute $\hat{\theta}$ such that asymptotically, $\|\hat{\theta} - \theta^*\|_2 = O(\sqrt{\epsilon} \log d)$ with probability at least $1 - \delta$. Additionally, it requires $\epsilon < C_1 \frac{\tau_l^2}{\tau_u^2 \log p}$ for some universal constant C_1 . τ_l and τ_u are the strong-convexity and smoothness parameters, respectively. T is the number of training steps, and d is the dimension. In the heavy tailed setting, RGD requires the same samples to compute $\hat{\theta}$ such that $\|\hat{\theta} - \theta^*\|_2 \leq \|\theta^0 - \theta^*\|_2 + C\sigma \sqrt{\|\Sigma\|_2 \left(\sqrt{\frac{dT \log(T/\delta)}{n}} \right)}$ with probability at least $1 - \delta$. C_2 is another universal constant. Σ is the covariance of the covariates. SEVER requires $O(d^5/\epsilon^2)$ ϵ -contaminated samples to compute $\hat{\theta}$ such that $\|\hat{\theta} - \theta^*\|_2 = O(\sqrt{\epsilon})$ with probability at least $9/10$. Note, because of differences in the formulations of the two papers, [DKK⁺19] does not

provide arbitrary probabilistic guarantees. The tradeoffs between sample size, dimensions, error bounds, and ϵ further complicate direct comparisons between the two methods. The choice of one method over another depends on sample availability and empirical success.

5.2 Other Strengths/Weaknesses

RGD requires the population risk to be strongly convex, but SEVER makes no such assumptions. SEVER can easily be integrated in non-convex settings, given a non-convex base learner, although the resulting statistical guarantees are weaker. Without convexity, SEVER only guarantees approximation of a γ -approximate critical points rather than the global optimum.

SEVER is also much easier to implement and can easily wrap around any γ -approximate learner with minimal effort and requires very few hyper-parameters (in the simplified implementation, just an estimate of ϵ and a number of rounds). On the other hand, RGD requires selecting an appropriate robust gradient estimator and integrating it into a gradient descent framework. Additionally, it requires many hyper-parameters such as step size and number of iterations, on top of what is needed for the chosen gradient estimator.

Robust mean estimation methods tend to be computationally expensive. Because SEVER uses them sparsely, it tends to be much faster than RGD (see Figure 5 in the Appendix).

SEVER’s outlier detection works by picking out the single direction of largest variation. On the other hand, RGD recursively picks subspaces with the largest variation. Consequently, we might expect that SEVER works best when the corrupted data points are coherent, while RGD handles dispersed outliers better. I found this to be true in my experiments.

6 Simulation Results

Because RGD works by hardening the optimizer and SEVER works by hardening the data, it is possible to combine both in a hybrid framework. I re-create the results of [PSBR18] and [DKK⁺19] for linear regression on simulated ϵ -contaminated data ¹.

Both papers generate the covariates X from a normal distribution and add Gaussian noise to the clean y . However, [PSBR18] uses fixed $\theta^* = \mathbf{1}$ and chooses outliers from $N(0, d^2)$ where d is the dimension (set to 500 in my experiments). This essentially picks outlier points with extreme leverage. [DKK⁺19] picks random $\theta^* \sim N(0, I_d/d)$ and chooses outliers $(X_{bad}, y_{bad}) \ni (X_{bad} = \frac{1}{\alpha \cdot \epsilon * n_{total}} y^T X + noise \text{ and } y_{bad} = -\beta)$. These outliers are aligned with the direction of the regression fit and designed to attack a least squares fit. Specifically, by choosing $\alpha = \beta$, we can systematically cause a ridge regression fit to estimate $\theta^* = \mathbf{0}$.

For my baselines, I used:

- **uncorrupted:** a standard ridge regression (λ tuned to $d * \sigma^2$)
- **l2:** SEVER algorithm but filtering points based on ℓ_2 distance of covariates x from mean

¹<https://github.com/Ian2x/Robust-Stochastic-Optimization>

- **loss**: SEVER algorithm but filtering points based on loss
- **gradientCentered**: SEVER algorithm but filtering points based on ℓ_2 -norm of centered gradient
- **RANSAC**: repeatedly fit subsamples and choose the best fit among them

Finally, I implemented the RGD algorithm with the Huber Gradient Estimator, SEVER, and a hybrid SEVER-RGD (using RGD as the base learner for SEVER).

6.1 Simulated Isotropic Corruption

The following results are on simulated data generated according to [PSBR18].

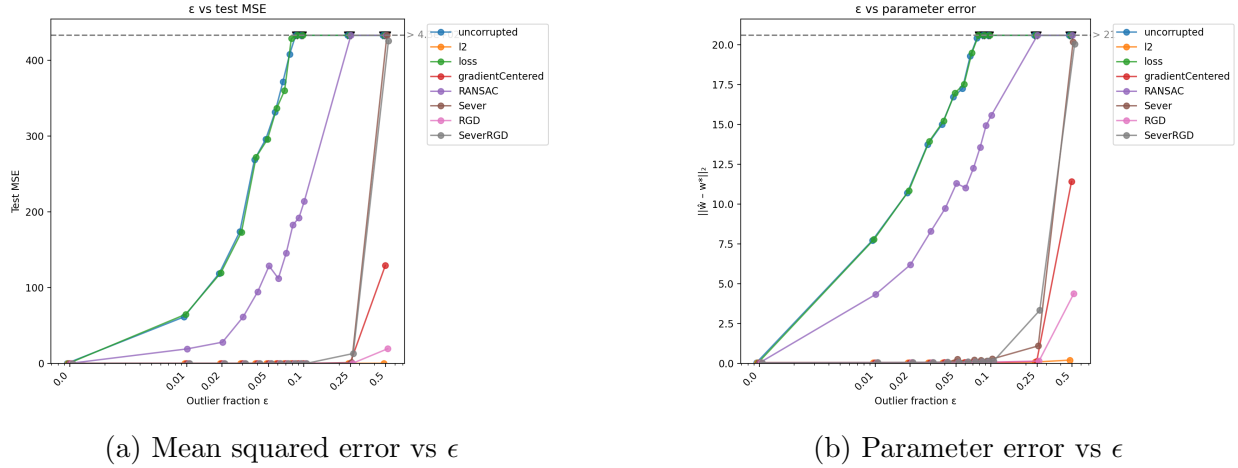


Figure 1: High leverage, isotropic corruption. MSE should approach 500 as the outliers dominate. Parameter error should approach $\sqrt{500}$ as the outliers dominate.

We can see that the **l2** algorithm works very well. This makes sense in this setting we are generating outliers with vary large covariates. This means that the **l2** algorithm has an easy time identifying such outliers. However, as we will see in 6.2, it is also possible for outliers to have a powerful effect on the overall fit, while maintain small covariates.

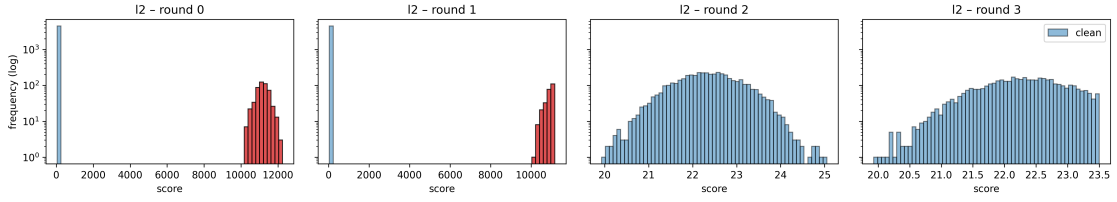
RGD also does very well, which is unsurprising given that this is the type of dataset used in [PSBR18]. **gradientCentered** also does fairly well, likely for a similar reason as **l2**; the larger the covariates, the more likely that the overall gradient will be large.

SEVER and SEVER-RGD do similarly, indicating that the addition of SEVER’s filtering methodology to RGD makes it worse. As SEVER removes outliers, it also may remove some clean datapoints. Given that RGD is designed to handle outliers, the impact of removing those clean datapoints may have a greater effect than the outliers that were removed.

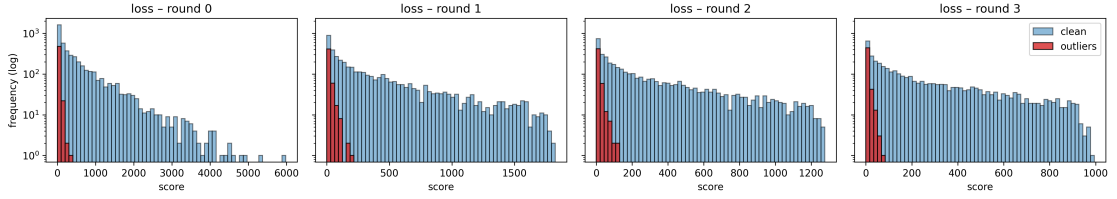
We can also see that the more traditional algorithms like RANSAC and ridge regression do relatively poorly, as they are not equipped to handle such extreme outliers.

We can also gather more insight into the performance of **l2** and SEVER from the histogram of scores over the filtering rounds. As a reminder, for each filtering round of the SEVER algorithm (and the baselines like **l2**), scores are given to each datapoint, and the

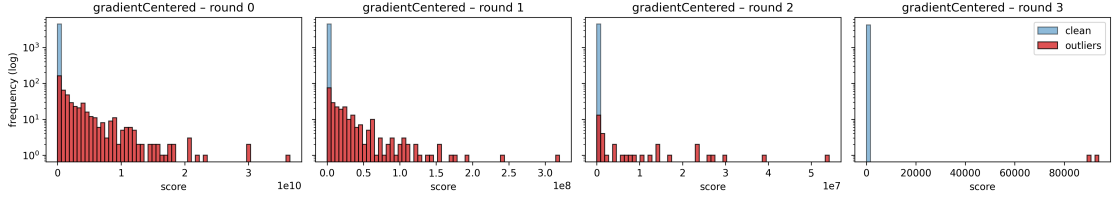
datapoints with the largest scores are filtered out. In the following histogram, the outlier data scores are shown in red, and the clean data scores are shown in blue.



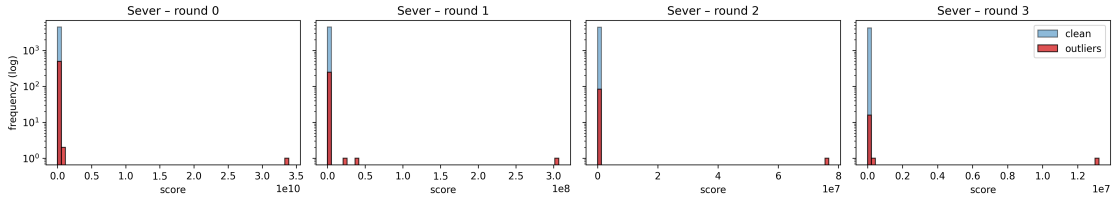
(a) **I2** filter



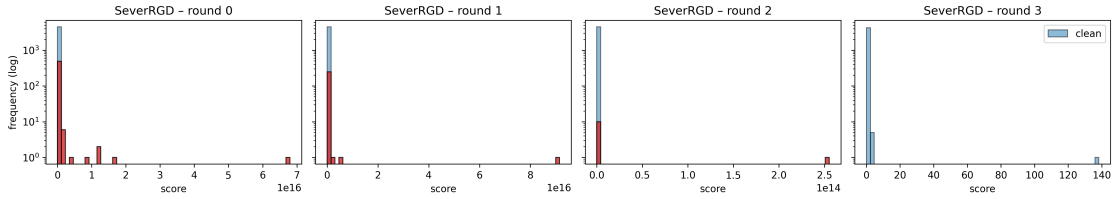
(b) **loss** filter



(c) **gradientCentered** filter



(d) **SEVER** filter



(e) **SEVER-RGD** filter

Figure 2: Filter score histograms for high leverage, isotropic synthetic corrupted data ($\epsilon = 0.1$)

From these histograms, it becomes clear why **I2** does so well. The outlier scores are glaringly obvious under this metric. **SEVER-RGD** also manages to remove all the outliers at $\epsilon = 0.1$. For **gradientCentered** and **SEVER**, the outlier scores tend to be larger, but the 4 filtering rounds are unable to completely filter out the outliers. Nonetheless, enough

are removed that a good fit can still be obtained for $\epsilon = 0.1$. On the other hand, for a metric like **loss**, the outlier scores are low and hidden within the clean scores.

It is interesting that SEVER can work as well as it does, even when the outliers are isotropic. Each round of SEVER only focuses on removing outliers in one direction at a time. However, even though the outliers are not coherent, their cumulative effective can be described by a singular vector shift.

6.2 Simulated Adversarial Corruption

The following results are on simulated data generated according to [DKK⁺19].

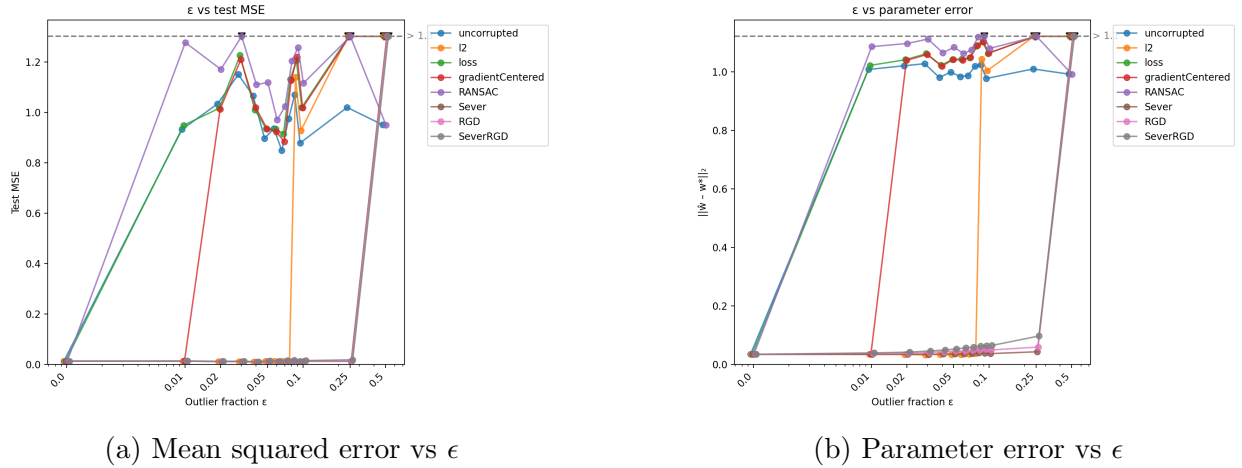


Figure 3: Adversarial corruption. MSE and parameter error should approach 1 as the outliers shift $\hat{\theta}$ to $\mathbf{0}$.

Unsurprisingly, SEVER does very well on this simulated dataset (Figure 3) as it was designed with this sort of corruption in mind.

Again, we can turn to the histogram of scores for further insight into what is happening. Figure 4 shows the score distribution for each of the SEVER-based filters at each round of filtering for one trial with $\epsilon = 0.1$.

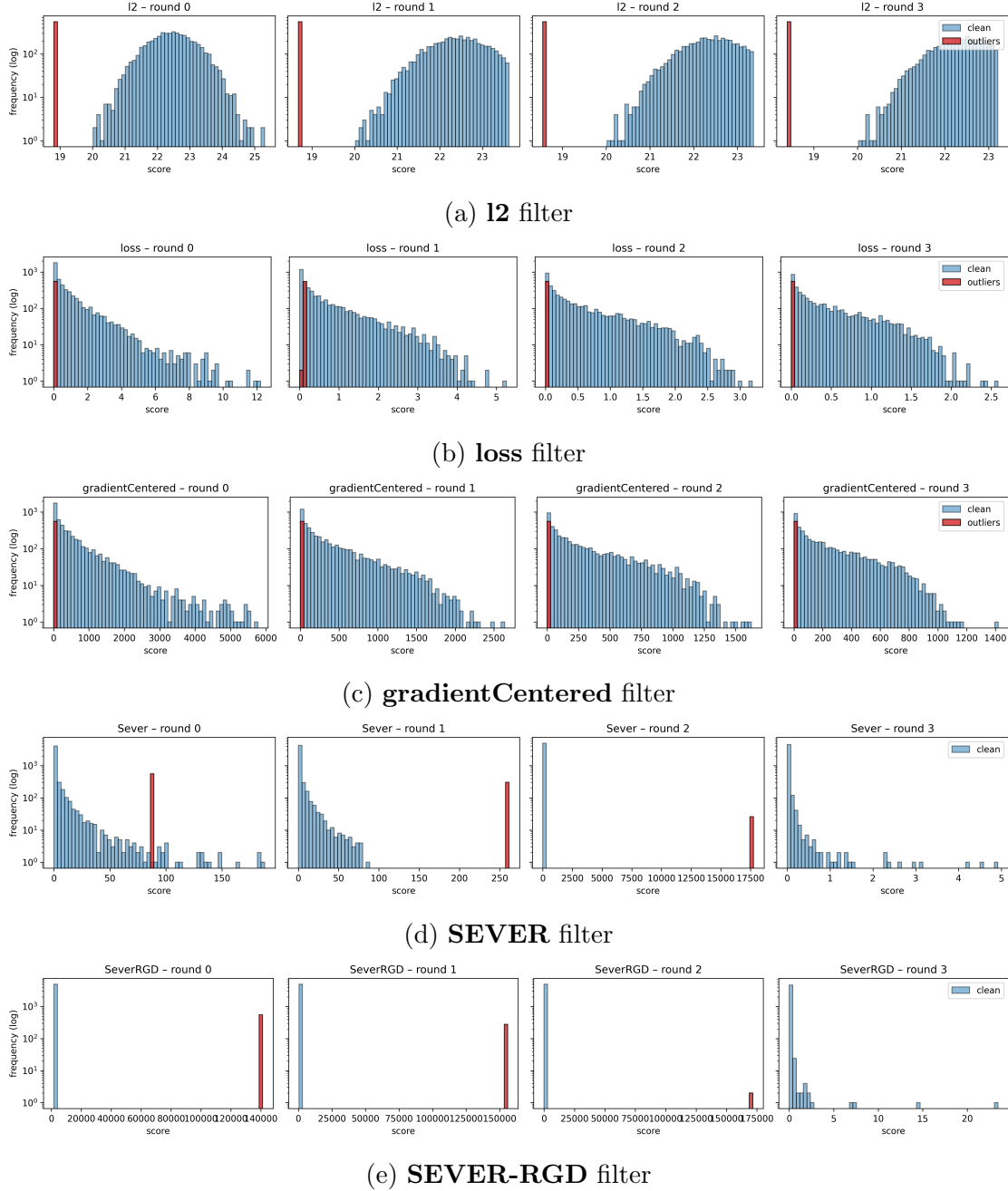


Figure 4: Filter score histograms for adversarial synthetic corrupted data ($\epsilon = 0.1$)

The **l2** score of the corrupted data is actually very low in this case—lower than the clean data. Nonetheless, the corrupted data still maintains a powerful effect on the estimated parameters because of its targeted placement.

For both the **loss** and **gradientCentered** filters, the scores of the corrupted data are hidden within the scores of the clean data, making them hard to identify.

Both **SEVER** and **SEVER-RGD** assign scores to the corrupted data that make them stand out and they are effectively removed.

6.3 Optimal MSEs

In the above analysis (Figure 1, Figure 3), we looked at the thresholds for which each algorithm failed. However, another important aspect of robust algorithms is how they perform when "successful". The data for the numerical results depicted in Figure 1 and Figure 3 is shown in the Appendix (Figure 6, Figure 7). In all simulations, we used a noise standard deviation of 0.1. Thus, the maximum achievable MSE is 0.01. We see that in the [DKK⁺17] simulation data, SEVER, RGD, and SEVER-RGD achieve nearly optimal MSEs for ϵ up to 0.25. In the [PSBR18] simulation data, "successful" MSEs are generally higher because the outliers are more spread out, and hence successful removal of the outliers is less of a binary outcome. We can see that SEVER and SEVER-RGD work well through $\epsilon = 0.1$, but only RGD works well at $\epsilon = 0.25$.

7 Discussion

In this paper I provided a comparison between two different approaches to robust stochastic optimization. [PSBR18] introduces Robust Gradient Descent, an algorithm that uses robust gradient estimators to harden each step in gradient descent. [DKK⁺19] introduces SEVER, a meta-algorithm that can harden a base learner by filtering out outliers over multiple rounds. I compared both algorithms in their overall structure, strengths/weaknesses, and theoretical guarantees in the context of linear regression. Finally, I tested both algorithms—and a hybrid framework combining both—on simulated data.

Overall, I found that each algorithm performed best under the specific outlier conditions they were designed for. Robust Gradient Descent works well under isotropic, high leverage outliers. SEVER works well under adversarial, directionally-coherent outliers. I also found that the hybrid framework generally performed worse than either of the individual algorithms. I theorize that this is because the two approaches do not work synergistically, but rather detract from one another. RGD is capable of working with corrupted data, but SEVER tries to provide its base learner with clean data. In the hybrid approach, the SEVER portion is removing data that RGD could have handled and used.

Because the core of both algorithms is estimating the population gradient, future improvements will likely come from harnessing better robust mean estimators. Additionally, steps could be taken to try to improve the range of conditions under which each algorithm performs well. For instance, SEVER is designed to filter out outliers based on the top singular value of the gradients, but could potentially be improved to filter out outliers based on vulnerable subspaces (like RGD's Huber Gradient Estimator).

References

- [DG85] Luc Devroye and László Györfi. *Nonparametric Density Estimation: The L_1 View*. Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons, 1985.
- [DKK⁺17] Ilias Diakonikolas, Gautam Kamath, Daniel M. Kane, Jerry Li, Ankur Moitra, and Alistair Stewart. Being robust (in high dimensions) can be practical. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pages 999–1008. PMLR, 2017.
- [DKK⁺19] Ilias Diakonikolas, Gautam Kamath, Daniel Kane, Jerry Li, Jacob Steinhardt, and Alistair Stewart. SEVER: A robust meta-algorithm for stochastic optimization. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pages 1596–1606. PMLR, 2019.
- [FB81] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.
- [Hub81] Peter J. Huber. *Robust Statistics*. John Wiley & Sons, 1981.
- [LRV16] Kevin A. Lai, Anup B. Rao, and Santosh Vempala. Agnostic estimation of mean and covariance. In *Proceedings of the 57th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 665–674. IEEE, 2016.
- [Min15] Stanislav Minsker. Geometric Median and Robust Estimation in Banach Spaces. *Bernoulli*, 21(4):2308–2335, 2015.
- [Miz02] Ivan Mizera. On depth and deep points: A calculus. *Annals of Statistics*, 30(6):1681–1736, 2002.
- [Owe07] Art B. Owen. A robust hybrid of lasso and ridge regression. In *Contemporary Mathematics*, volume 443, pages 59–72. American Mathematical Society, 2007.
- [PSBR18] Aaditya Prasad, Akshay S. Suggala, Sivaraman Balakrishnan, and Pradeep K. Ravikumar. Robust estimation via robust gradient estimation. *CoRR*, abs/1802.06485, 2018.

Appendix

Runtimes at $\epsilon=0.1$	
	median runtime (s)
RANSAC	2.3211
RGD	46.2412
Sever	1.3056
SeverRGD	212.5080
gradientCentered	0.2116
l2	0.1966
loss	0.1670
uncorrupted	0.0316

Figure 5: Median runtime for $\epsilon = 0.1$ for [PSBR18] generated data

(a) Median MSE								
ϵ	uncorr.	ℓ_2	loss	grad. C.	RANSAC	Sever	RGD	S-RGD
0.000	0.0127	0.0127	0.0127	0.0127	0.0127	0.0127	0.0127	0.0127
0.010	0.9327	0.0129	0.9490	0.0130	1.2776	0.0129	0.0131	0.0131
0.020	1.0341	0.0112	1.0167	1.0121	1.1711	0.0111	0.0119	0.0120
0.030	1.1508	0.0107	1.2262	1.2097	1.4273	0.0113	0.0116	0.0127
0.040	1.0660	0.0096	1.0104	1.0201	1.1107	0.0095	0.0096	0.0100
0.050	0.8968	0.0121	0.9359	0.9351	1.1189	0.0117	0.0121	0.0140
0.060	0.9358	0.0117	0.9345	0.9236	0.9703	0.0115	0.0116	0.0124
0.070	0.8492	0.0119	0.9147	0.8836	1.0246	0.0116	0.0136	0.0145
0.080	0.9750	0.0121	1.1319	1.1277	1.2052	0.0127	0.0135	0.0159
0.090	1.0704	1.1397	1.2065	1.2199	1.2576	0.0093	0.0113	0.0127
0.100	0.8786	0.9280	1.0203	1.0185	1.1163	0.0120	0.0137	0.0150
0.250	1.0199	1.5181	1.3704	1.3621	1.4567	0.0119	0.0132	0.0182
0.500	0.9508	1.3128	1.3291	1.3138	0.9500	1.3138	1.4748e+37	1.4092e+37

(b) Median Parameter Error $\ \hat{w} - w^*\ _2$								
ϵ	uncorr.	ℓ_2	loss	grad. C.	RANSAC	Sever	RGD	S-RGD
0.000	0.0344	0.0344	0.0344	0.0344	0.0344	0.0344	0.0344	0.0344
0.010	1.0085	0.0341	1.0225	0.0348	1.0867	0.0343	0.0375	0.0392
0.020	1.0213	0.0336	1.0421	1.0397	1.0975	0.0345	0.0379	0.0418
0.030	1.0283	0.0326	1.0626	1.0598	1.1127	0.0341	0.0404	0.0459
0.040	0.9812	0.0341	1.0229	1.0199	1.0659	0.0354	0.0433	0.0491
0.050	0.9997	0.0331	1.0429	1.0431	1.0845	0.0354	0.0451	0.0529
0.060	0.9842	0.0335	1.0443	1.0407	1.0637	0.0358	0.0458	0.0565
0.070	0.9878	0.0352	1.0491	1.0504	1.0748	0.0376	0.0479	0.0589
0.080	1.0196	0.0369	1.0886	1.0906	1.1204	0.0396	0.0497	0.0630
0.090	1.0245	1.0440	1.1035	1.1035	1.1227	0.0383	0.0513	0.0641
0.100	0.9778	1.0041	1.0636	1.0649	1.0805	0.0368	0.0491	0.0646
0.250	1.0104	1.2439	1.1615	1.1625	1.1674	0.0431	0.0593	0.0969
0.500	0.9928	1.1800	1.1807	1.1801	0.9921	1.1802	4.0142e+18	3.9167e+18

Figure 6: Median errors over 3 simulations of [DKK⁺17] outlier data generation (c) MSE, (d) Parameter Error

(a) Median MSE

ϵ	uncorr.	ℓ_2	loss	grad. C.	RANSAC	Sever	RGD	S-RGD
0.000	0.0138	0.0138	0.0138	0.0138	0.0138	0.0138	0.0138	0.0138
0.010	61.3077	0.0117	64.7275	0.0119	19.1137	0.0116	0.0122	0.0131
0.020	118.7966	0.0109	119.6077	0.0114	27.9716	0.0110	0.0116	0.0124
0.030	174.0216	0.0099	173.2523	0.0097	61.2574	0.0118	0.0103	0.0121
0.040	268.7753	0.0141	272.3642	0.0149	94.5670	0.0141	0.0153	0.0179
0.050	296.1034	0.0141	295.7747	0.0139	128.6085	0.1057	0.0171	0.0189
0.060	331.6001	0.0121	336.9072	0.0127	112.1585	0.0130	0.0163	0.0213
0.070	371.6665	0.0120	360.1518	0.0128	145.3692	0.0538	0.0151	0.0226
0.080	408.3129	0.0127	428.8432	0.0128	182.9381	0.0393	0.0152	0.0231
0.090	440.4467	0.0156	444.0772	0.0163	192.1148	0.0353	0.0171	0.0277
0.100	484.1813	0.0114	489.3572	0.0127	214.0527	0.0748	0.0160	0.0310
0.250	499.7160	0.0150	499.7370	0.0225	499.7370	1.0899	0.0288	12.9807
0.500	523.7999	0.0671	523.8046	129.2801	523.8046	434.9654	19.3144	425.9304

(b) Median Parameter Error $\|\hat{w} - w^*\|_2$

ϵ	uncorr.	ℓ_2	loss	grad. C.	RANSAC	Sever	RGD	S-RGD
0.000	0.0420	0.0420	0.0420	0.0420	0.0420	0.0420	0.0420	0.0420
0.010	7.7188	0.0421	7.7792	0.0447	4.3267	0.0423	0.0483	0.0526
0.020	10.7001	0.0446	10.8364	0.0465	6.1947	0.0448	0.0543	0.0623
0.030	13.7435	0.0455	13.9428	0.0482	8.2905	0.0485	0.0576	0.0705
0.040	15.0107	0.0459	15.2336	0.0515	9.7415	0.0466	0.0630	0.0838
0.050	16.7288	0.0462	16.9804	0.0508	11.2989	0.2594	0.0667	0.0908
0.060	17.2560	0.0491	17.5362	0.0559	11.0199	0.0510	0.0730	0.1044
0.070	19.2959	0.0471	19.4965	0.0567	12.2516	0.2171	0.0732	0.1169
0.080	20.4208	0.0500	20.6789	0.0588	13.5691	0.1949	0.0764	0.1222
0.090	21.4543	0.0508	21.6189	0.0608	14.9328	0.1537	0.0794	0.1400
0.100	22.2249	0.0497	22.2898	0.0596	15.5788	0.2750	0.0824	0.1482
0.250	22.3602	0.0830	22.3607	0.1183	22.3607	1.0974	0.1459	3.3305
0.500	22.3606	0.2040	22.3607	11.4216	22.3607	20.1955	4.3884	20.0598

Figure 7: Median errors over 3 simulations of [PSBR18] outlier data generation (c) MSE, (d) Parameter Error