We thank the useful suggestions from the reviewers. Below please find our responses to the major points raised.

**To Review 1: About the background:** W-DRO problems have received much attention in the machine learning community as they not only provide a probabilistic justification of existing regularization techniques but also offer a powerful alternative approach to tackle ERM problems; see [22]. However, the only known way to solve most existing DRO formulations is to use general-purpose solvers, which limits the scalability of the approach. Our work is motivated by the desire to develop practically efficient methods with provable guarantees for DRO problems, so as to realize the benefits of the approach in large-scale learning settings.

**Q1:** Table 1 looks strange to me. It might better to give more intuitions and explanations.

**R1:** It may be not true that adding a quadratic term always makes the algorithms faster. In a nutshell, the concrete convergence rate depends on the algorithm used and whether it can exploit the regularity condition of the problem at hand. For instance, $f(x) = |x|$ satisfies the sharpness condition, and the subgradient method (SubG) can achieve a linear rate. However, if we add a quadratic term $g(x) = |x| + x^2$, the sharpness condition no longer holds but QG holds. In this case, SubG is only known to converge at the sublinear $\mathcal{O}(\frac{1}{k})$ rate but proximal gradient descent (PGD) is known to converge linearly. However, PGD cannot be applied to our setting mainly due to the non-separability of the non-smooth objective. In particular, the associated proximal mapping cannot be efficiently implemented.

**Q2**: Can the authors prove the convergence beyond the QG condition when $c > 0$, give a possible faster rate?

**R2**: The reviewer raises an interesting question. QG is a rather general regularity condition, which has equivalent relationships with the error bound and PL/KL properties. To the best of our knowledge, almost all problem-specific convergence analyses utilize either a QG- or PL/KL-type regularity condition, including ours (both sharpness and QG can be viewed as a KL-type condition). It is not clear whether our problem possesses some more general regularity conditions, especially for the $\lambda$ variable. Thanks for your question.

**Q3**: From my experience, the proximal point algorithm is always faster than the SubG in general.

**R3**: If the cost of proximal point update is comparable with SubG, PPA is indeed faster than the SubG (i.e., $n = 1$ in our paper). However, this is not the case for incremental methods. As we stated in lines 258-268, the main reason is that IPPA can only update one sample at a time. Thus, if we take batch size = 1 for M-ISG, we can see that IPPA still enjoys substantial advantages over ISG, see Fig.1 (a)-(d). Nevertheless, M-ISG can update the batch data at once and thus less epigraphical projection operations are required for each epoch. Moreover, nested for-loop is not so efficient in MATLAB. Thus, IPPA is slower than M-ISG w.r.t the Wall-clock Time. Thanks for your comments.

**Q4**: The Holderian growth condition is global or local? **R4**: **Global**! Thanks for your kind reminder to clarify this.

**To Review 2:** Thank you for pointing out the interesting research direction. We will explore this in future work.

**To Review 3: Q1**: More details on the BLR condition...? **R1:** BLR is a classic assumption in variational analysis, see section 3.3 in [28]. In your example $\min g(Ax)$, the optimal set can be characterized by a linear system $\{x : Ax = y^*\}$ for some fixed $y^*$, which is polyhedral and satisfies the BLR automatically. More examples can be found in [2,28].

**Q2**: For dataset a3a, it seems that the Hybrid strategy does not run as fast as GS-ADMM and YALMIP.

**R2:** Thanks for your question. The main reason is that the convergence rate of incremental methods depends on the sharpness constant, which in turn is data-dependent (i.e., condition number). You can also observe that a8a is slower that a9a, but a9a has a large problem size. We have to emphasize that both GS-ADMM and YALMIP do not scale well with problem size. Thus, you can observe that the performance gap grows considerably as the problem size increases.

**To Review 4:** Thanks for your comments. We have to emphasize that you can certainly reproduce all experiment results based on the details provided in the Appendix (i.e., Table 5,6; Algorithm 1,2,3). We will release our code later.

**Q1:** ... For example, why GS-ADMM results are not shown for $l_1$ and $l_2$-norm optimization?

**R1:** Thanks for your kind reminder. As we stated in lines 247-249, we just extend GS-ADMM to tackle the $\ell_\infty$ DRSVM problem. Our major concern is that [16] only provides the source code to deal with the $\ell_\infty$ case. For the sake of fairness, we only report this situation. Moreover, the $\ell_\infty$ case is the most efficient (i.e., the faster inner solver - conjugate gradient with an active set method can only tackle the $\ell_\infty$ case in [16]). Thus, Table 4 is enough to verify the efficiency of our proposed method. Based on your advice, we will add more comparison experiments in the revision.

**Q2:** It would be helpful to show the objective function curves of the comparing method in Fig. 1.

**R2:** Thanks for your suggestion. Let us explain a little more to clarify. First, Fig.1 aims to corroborate the theoretical finding in Table 1. Namely, we want to showcase the empirical performance of ISG and IPPA under different regularity conditions and step size strategies. This is why we only report the curve of incremental methods. Second, the YALMIP solver relies on interior-point algorithms, which use second-order (Hessian) information. It is thus not entirely fair to compare the function value curves of our first-order method and the YALMIP solver. For GS-ADMM, it requires an outer loop to search for the $\lambda^*$ (i.e., like a two-stage algorithm). Hence, it is also incomparable to ours in terms of function value. Third, since the problem is convex, all methods will converge to the same objective value.

**Q3:** How the parameters $c$, $\kappa$, and $\epsilon$ determined in the experiments?

**R3:** We mentioned the hyperparameter setting in our original paper (i.e., line 255 and all table titles). $\kappa$ and $\epsilon$ do not affect the convergence rate/computational complexity, see Theorem 4.3 for details. The constant $c$ only influence the regularity condition. Also, some preliminary experiments have been conducted for different hyperparameter sets and all experiment performances have no obvious difference. Thus, we just follow the experimental setup in [16].