We thank the reviewers for their constructive comments and suggestions. We respond to each point individually.

R1: *More description to highlight the unique contribution.*
Compared with previous metric learning methods by using low-rank/online/stochastic strategies, that still encounter the scalability problem when handling large data, our paper has two unique contributions as follows. (1) Our method embeds the triplet constraints into a matrix, and further reduces the size of involved matrices by replacing $\mathbf{Y}$ with $\mathbf{BV^T}$, which ensures the existence of the optimal solution in the reduced matrices. (2) By substituting the closed-form optimal solution of $\mathbf{s}$, the optimization of positive semidefinite matrices is converted into the optimization on the Stiefel manifold, which can be optimized more efficiently. These two contributions significantly reduce the complexity and the size of involved matrices, which makes our method scalable to both high dimensions and large numbers of samples.

R1: *Explain the difference with some recent methods.*
We will add the missing references in "Related Work" and explain the difference with them in the revision. *Mini-SGD* [Qian et al. ML 2015] uses the mini-batch strategy to optimize the metric matrix in the positive semidefinite cone. Using the online metric learning strategy, *OPML* [Li et al. PR 2018] introduces a closed-form formula for updating the metric matrix. However, the above two methods all require updating a large $D \times D$ matrix (D is the dimensionality of original data) at each iteration, and they can only learn from samples with hundreds of dimensions in their experiments. In contrast, our method models the metric learning problem on the Stiefel manifold with much smaller size $r \times d$, which significantly reduces the complexity and memory usage in optimization. In our experiments, the datasets can be with up to one million dimensions (see Table 1 in our paper).

R1: *Theoretical results using $\mathbf{BV^T}$ to replace $\mathbf{Y}$ to ensure performance, and differences with anchor-based strategy.*
(1) If the linear equation system $\mathbf{Y} = \mathbf{LX}$ has a feasible solution of $\mathbf{L}$, it can always be transformed into a full-rank linear equation system $\mathbf{YV} = \mathbf{LU\Sigma}$. According to Theorem 1 in our paper, since $\mathbf{YV} = \mathbf{B}$, all the possible solutions can be covered by $\mathbf{B} \in \mathbb{R}^{d\times r}$, which ensures the performance of accelerated low-rank metric learning.
(2) For the anchor-based strategy in *AnchorGraph* [Liu et al. ICML 2010], the anchors are "local samples" with high dimension D, and the algorithm represents each data point as a convex combination of its closest anchors. In contrast, our method optimizes a smaller matrix $\mathbf{B} \in \mathbb{R}^{d\times r}$ to obtain the global optimal solution of a larger matrix $\mathbf{L}$, which is different with the "local sampling" strategy used in *AnchorGraph*, as will be added in "Related Work" in the revision.

R2: *The upper bound of r.*
As a preprocessing parameter, the upper bound of r is an empirical value. In general, a larger r will increase memory and computational costs, while a smaller r may lose some intrinsic values. In our preliminary experiments under dataset "TDT2", when r varies in 500, 1000, 2000, 3000 and 4000, the accuracy just slightly changes to 0.948, 0.964, 0.965, 0.963 and 0.948, respectively. This shows that the variation of r within a reasonable range merely affects computational time and memory, but has little effect on accuracy. For fair comparison, we use the same r for all compared methods.

R2: *What about the approximation loss of $\mu(x)$, and how to optimize the problem without using $\mu(x)$?*
The authors guess the approximation loss concerned by the reviewer is the difference between $\mu(x) = -\log(\sigma(-x))$ and $\max(0, x)$, where its maximum value is $-\log\frac{1}{2} \approx 0.69$ at the point $x = 0$. If not using $\mu(x)$, our model can still be solved. However, due to the discontinuous gradient, the convergence requires more iterations, and the results are less stable. For example, under the dataset "TDT2" in five repeated runs, when using $\mu(x)$, the average number of iterations, average accuracy, and standard deviation are 6.6, 0.962, and 0.0008, respectively. In contrast, when only using $\max(0, x)$, these values become 12.6, 0.955, and 0.012, respectively.

R2: *Theoretical analysis of stochastic strategy.*
The theoretical analysis of the stochastic strategy which updates $\mathbf{L}$ in step sizes by $1/\sqrt{I}$ decay can refer to the reference [14] in our paper. We will add more description for [14] about the stochastic strategy in the revision.

R2: *Parameter sensitivity examinations.*
- The examination of different mini-batch sizes has been provided in Fig. 4, where N_t is an indicator of mini-batch size.
- The examination of different m has been provided in Fig. 3. We will explicitly add the definition of m in the revision.
- d is the target rank value in low-rank metric learning tasks, which is usually set according to user preference. We use $d = 100$ in all experiments for a fair comparison, which is a moderate value so that most tested methods can achieve good performance on most datasets, as mentioned in Section 4.1.
- Triplet sizes: 5 triplets are randomly generated for each sample (which is a moderate value for most datasets), and the same triplet sets are used for all tested methods for a fair comparison, as mentioned in Section 4.1.

R3: *Potential applications and more explicit gaps on the previous literature.*
A more detailed highlight of contributions and differences with recent methods can refer to 1st Answer for Reviewer 1. Metric learning has been widely used in various areas, such as dimensionality reduction, feature extraction, and information retrieval. Our method can be applied to the scenario of learning metrics quickly on large numbers of high-dimensional data with limited computing resources. We will introduce some application scenarios in the revision.