Supplementary Materials for Efficient Optimization Methods for Extreme Similarity Learning with Nonlinear Embeddings

Bowen Yuan
National Taiwan University
f03944049@csie.ntu.edu.tw

Pengrui Quan
University of California, Los Angeles
prquan@g.ucla.edu

Yu-Sheng Li
National Taiwan University
r07922087@csie.ntu.edu.tw

Chih-Jen Lin
National Taiwan University
cjlin@csie.ntu.edu.tw

1 MORE IMPLEMENTATION DETAILS

In evaluating $L(\theta)$, the required $P, Q, \hat{P}_c, \hat{Q}_c, P_c, Q_c$, and Frobenius inner products can be directly computed by the forward and other built-in functions. However, when we use GPUs for large data sets, we face two difficulties. First, as the memory consumption of a forward process is proportional to the number of data, calculating the whole $P$ and $Q$ at once may be infeasible for GPUs. Second, for later computations, we need to cache $P$ and $Q$ in $O((m+n)k)$ space, which may also be infeasible.

For the first difficulty, fortunately, as the memory consumption of a forward process is proportional to the number of data, the computation of each subset is independent of others, we follow [2] to have a mini-batch setting. Specifically, by splitting $U$ and $V$ into multiple subsets, we sequentially calculate rows of $P$ or $Q$ corresponding to indices in each subset. For $\hat{P}_c, \hat{Q}_c, \hat{Q}_c, P_c,$ and $Q_c$, each of which is the sum of $m$ or $n$ matrices of size $k \times k$, we can calculate the partial results on each subset and accumulate them for the final output.

To overcome the second difficulty, we apply a heterogeneous computing scheme to store $P$ and $Q$ in CPU, which has a larger memory capacity than GPU. After finishing the above computation in GPU on each subset, we feed the required partial matrices to the CPU memory. Then with the cached $W, H, P,$ and $Q$, we compute $ZQ$ and $Z^\top P$ on CPUs. Finally, we call the automatic differentiation function to finish the remaining transposed Jacobian-vector products.

REFERENCES