Incremental eigenvalue computation for graph-structured Sparse PCA search

Daniel Ringwalt

November 29, 2022

1 Background

2 Methods

2.1 Introducing eigenvalue lower bound

The Optimal-SPCA branch-and-bound library is modular, such that one or more upper bound and lower bound implementations should be added to the

2.2 Incrementally computing eigendecomposition

Symmetric eigendecomposition can be performed via a series of rank-one updates to subproblems.

2.2.1 Fortran rank-one update implementation

2.2.2 Serial bidiagonalization implementation

3 Results

3.1 Optimal Sparse PCA tree search algorithm

3.2 Sparse recovery subspace pursuit algorithm

4 Conclusions

We can start considering practicum for exact Sparse PCA adoption, which end-users have not been onboarded onto yet. Our eigenvalue lower bound is practical, because it is simpler than the Optimal SPCA criteria used so far, and is the appropriate fit for this finely tuned LAPACK procedure. If the remainder of Optimal SPCA Julia code had been rewritten at a similarly low level (e.g. Fortran), then the returns from visiting fewer non-leaf nodes (the other, currently Julia code) would have reduced impact. However, micro-optimizing the existing code would significantly limit the practical potential of future high-level restructuring for optimization.

However, other components (especially power iteration at the non-leaf nodes) could be less preferred, because it leverages repeated matrix multiplications when there could be a more sophisticated LAPACK routine meeting its needs.

This also affects the practicum of onboarding end users. Large sparse PCA problems could produce demand to move computation to a cloud service, using large (dedicated CPU) instances or a GPU algorithm. Our users would be particularly likely to offload this task, because latency will grow faster than $O(N)$, and because the inputs/outputs are simple (data matrix input, list of variable names output). This is in contrast to costly normalization/imputation cloud services, where the user must download a dense output, the same size or slightly larger than their input.

5 Future Directions

We can further minimize rank-one update latency on the GPU, retaining a copy of the covariance matrix in VRAM. Within this paper, the secular equations (K equations, each consisting of a sum of K terms plus a constant) has the most potential to utilize a GPU. However, exact sparse PCA was usually considered with K between 5 and 12, and K^2 arithmetic computations are unlikely to make effective use of all GPU cores. Therefore, this is a jumping-off point for new parallel algorithms, which will scale to somewhat larger N.