Irlbpy – A fast partial SVD for Python

J. Baglama [1], M. Kane [2], and B. Lewis [3]

[1]Department of Mathematics, The University of Rhode Island
[2]Department of Biostatistics, Yale University
[3]Paradigm4

Overview

The singular value decomposition (SVD) is central to many important data-analytic methods and applications including principal component analysis, canonical correlation analysis, correspondence analysis, latent semantic indexing and non-linear iterative partial least squares to name a few. However, numerical implementations of the SVD are intensive, generally incurring a computational complexity of $O(m^2n + n^3)$ for an $m \times n$ real matrix (when $m \geq n$). As a result, data scientist's have fewer analytical tools to understand the structure of data as those data become large and the resulting computational cost becomes too expensive to carry out.

Many important analysis methods and applications only require a subset of the largest singular values and corresponding singular vectors. With this in mind, some numerical analysts have focused on the developement of *truncated* SVD algorithms that calculate the largest or smallest singular values and vectors for a matrix. The *augmented implicitly restarted Lanczos bidiagonalization* (IRLB) algorithm [**?**] is a fast and efficient approach for calculating truncated singular value decompositions, generally scaling linearly in the size of the matrix. This innovative algorithm calculates a key numerical decomposition for statistical and machine learning procedures and it effectively scales to massive data. Standard analyses can now be applied to much larger data sets than previously possible. Furthermore, the approach used by the IRLB algorithm suggests a general class of new approximation algorithms that can be used for big-data challenges where current approaches fall short.

While IRLB implementations have existed for some time now in both Matlab [**?**] and R [**?**] the scientific Python community has not enjoyed the computational savings offered by the algorithm until now. This poster introduces the irlbpy package for Python, a pip-installable open-source implementation of the IRLB algorithm that is available from github at https://github.com/bwlewis/irlbpy. The package is compatible with dense and sparse data, accepting either numpy 2D arrays or matrices, or scipy sparse matrices as input. The rest of this poster gives an overview of the algorithm and benchmarks performance. The benchmarks were performed on a Mac Book Pro with a quad-core 2.7 GHz Intel Core i7 with 16 GB of 1600 MHz DDR3 RAM running Python version 2.7.3, Numpy version 1.7.0, and SciPy version 0.12.0.

Partial SVD Definition

A truncated SVD of a matrix $X \in \mathcal{R}^{m \times n}$ computes the decomposition

$$XV = U\Sigma,$$

where V is an $n \times p$ matrix with orthonormal columns, U is an $m \times p$ matrix with orthonormal columns, Σ is an $p \times p$ diagonal matrix, and $1 \leq p \leq \min(m, n)$. The diagonal values of Σ are called singular values of X and are nonnegative and ordered, $\sigma_1 \geq \sigma_2 \geq \cdots \sigma_p \geq 0$. The columns of U and V are called the left and right singular vectors, repsectively.

When $p = \min(m, n)$, the truncated singular value decomposition coincides with the usual singular value decomposition and $A = U\Sigma V^T$. When $p < \min(m, n)$, then $U\Sigma V^T$ only approximates the matrix A.

The irlb algorithm estimates a truncated SVD of X, computing

$$XV \approx U\Sigma$$

by projecting the matrix into an augmented sequence of Krylov subspaces. The quality of the approximation is user-controlled by a subspace iteration tolerance parameter. The method often converges to a solution quickly even when the tolerance is set to a small value. The availability of the parameter allows users to make very quick estimates of a few singular values and their corresponding vectors, even for large problems.

Discussion

In practice the IRLB algorithm scales linearly in the size of the data. This gives it a tremendous advantage over the traditional SVD methods when only the largest singular value information is needed.

The IRLB algorithm and irlbpy implementation are compatible with both dense numpy matrices and sparse scipy matrices.

A distributed version of the IRLB algorithm is availablle for SciDB [**?**] (and soon indirectly for Python through the SciDBPy package under development). This version has computed a truncated SVD on sparse matrices with 50 million rows, 40 million columns and 4 billion non-zero elements on a cluster of 4 dual-Xeon machines in approximately five minutes per singular value/vector.

The irlbpy package can compute a few singular vectors and corresponding singular values of problems similar to the Netflix prize problem–a sparse 500000 × 18000 matrix with about 100 million nonzero entries–in a few minutes on a laptop.

Martinsson, Rokhlin, and others have developed fast truncated SVD and related decomposition methods using randomized methods, see for example [**?**] These methods are indeed competetive with the IRLB approach, and problems exist where each outperforms the other.

Dense Matrix Comparison

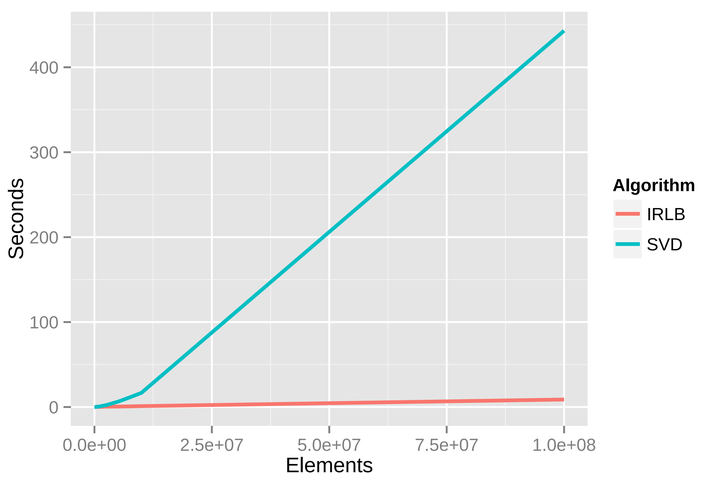

Figure 1: Performance comparison of the IRLB and the numpy implementation of the SVD calculating the 10 larges singular values and vectors.

Dense Matrix Scaling

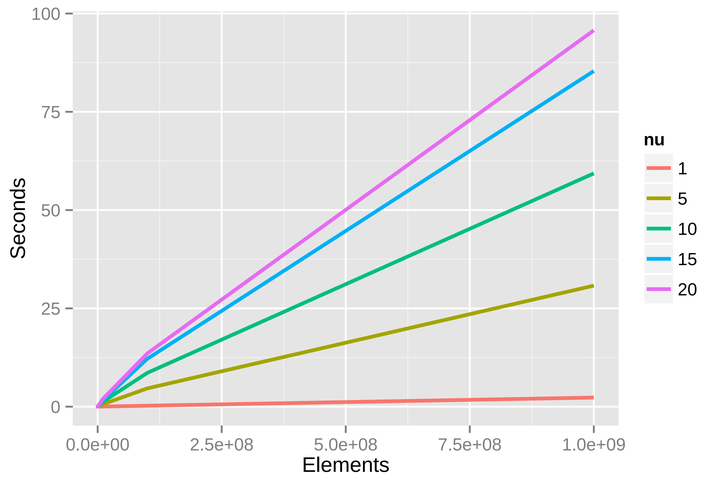

Figure 2: The time required to calculate the IRLB on dense matrices for specified values of nu (the number of singular vectors).

Sparse Matrix Scaling

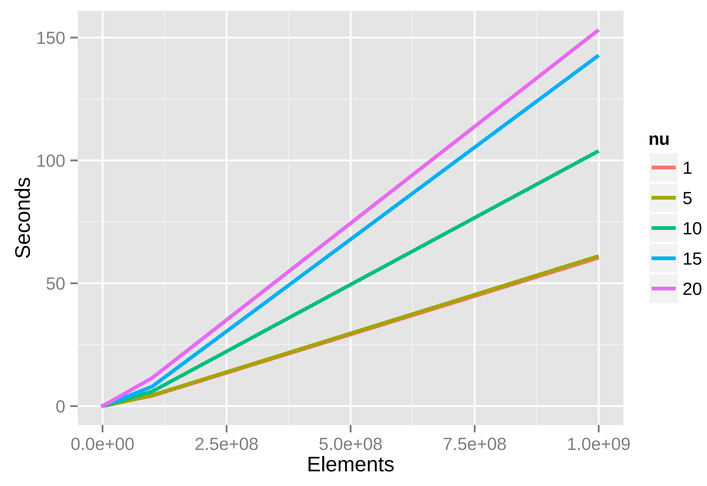

Figure 3: The time required to calculate the IRLB on sparse matrices for specified values of nu (the number of singular vectors).