Fast SVD for Large-Scale Matrices

Approximate PCA, eigenvector methods and more via stratified Monte Carlo

Michael P. Holmes
Alexander G. Gray
Charles Lee Isbell, Jr.

College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0760
{mph,agray,isbell}@cc.gatech.edu

SVD. The singular value decomposition (including eigendecomposition) is a key linear algebraic operation at the heart of many machine learning methods. Principal components analysis is perhaps the most well known, including variants such as kernel PCA and local embedding techniques, and there are countless other eigenvector/SVD-based methods in dimension reduction, manifold learning, metric learning, collaborative filtering, etc.

Exact SVD of a $m \times n$ matrix has time complexity $O(\min\{mn^2, m^2n\})$. Though feasible for small datasets or offline processing, many modern applications involve real-time learning and/or massive dataset dimensionality and size. For instance, deployed adaptive systems and exploratory data anlysis require near-instant reactivity, while large-scale simulations can produce unbounded quantities of complex data. Further, parallel architectures are generally not available to deployed systems or to many practicioners who would like to run experiments in seconds rather than days. We therefore have a need for *algorithmic* acceleration of key computational bottlenecks such as the SVD.

Monte Carlo and the SVD. An $m \times n$ matrix A can be written in the form $A = U\Sigma V^T$, where the columns u_i and v_i of U and V form a orthonormal basis sets, referred to as the left and right singular vectors, and Σ contains values σ_i known as the singular values in descending order along its diagonal, with zeroes everywhere else.

We commonly wish to find the best rank-k approximation to the original matrix A. It can be shown that the best approximation in the sense of minimizing $||A - A_k||_F^2$ is $A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T = U_k \Sigma_k V_k^T$. However, for k large enough to capture most of the variance in A, computing A_k requires essentially the same amount of time as the full SVD. We would like a fast approximation to A_k with rigorous error control.

The seminal work in [1] proposed the following idea: assume the data points are in the columns of A. Then AA^T is the dataset covariance matrix, and it's SVD can be written in terms of A's SVD as $AA^T = U\Sigma^2 U^T$, meaning that the SVD of A can be recovered from that of AA^T and vice versa. Now, instead of computing the SVD on A, let us instead choose a subset of its columns to form a subsampled matrix S. Statistically, the subsampled covariance SS^T should approach AA^T, and therefore the SVD of SS^T will be close to that of AA^T. The SVD of SS^T can therefore be used to recover an approximation to the SVD of A, while being computed from the SVD of S. SVD on S is only $O(s^2 m)$ (assuming $s < m$), thus giving a fast approximation.

Several other papers in the theory and algorithms community have followed in this vein, typically giving algorithms with error bounds such as

$$||A - \tilde{A}_k||_F^2 \leq ||A - A_k||_F^2 + \epsilon ||A||_F^2 \ , \tag{1}$$

where $\tilde{A_k}$ is the sample-based approximation to the optimal A_k [2]. Samples are usually taken from a distribution where the probability of choosing a column is proportional to its length squared, and the sample size is required to be set to some complex combination of constants in order to guarantee the error bound.

Theoretical and algorithmic contributions. One problem with these error bounds is the they tend, empirically, to be quite loose [3], forcing many more samples to be taken than necessary for a targeted error level ϵ. The length-squared distribution is problematic when some data points have much larger values than others, resulting in loss of fine structure that exact SVD readily picks out. While the length-squared distribution minimizes variance for a limited class of sampling schemes, it does not incorporate any of the approaches to variance reduction found in classical Monte Carlo approximation. Finally, the additive error $\epsilon||A||_F^2$ above the optimal $||A - A_k||_F^2$ is not relative to the optimal error but to the unbounded $||A||_F^2$.

We propose to move from the simple randomization sense of Monte Carlo to the classical empirical sense of sampling, estimating variance, relating variance to error, then continuing to sample until target error levels are achieved. By moving to this empirical style of Monte Carlo, we can automatically determine the minimum number of samples needed based on variance while still maintaining the same error bounds as above. This cuts out theoretical slack and sample inefficiency and therefore increases speedup.

Additionally, we propose the use of stratified sampling for variance reduction and even more significant increase in sample efficiency. The idea of stratified sampling is to divide the dataset into disjoint regions and focus sampling in the regions of higher variance, thus achieving an overall large variance reduction. Specifically, we use spatial partitioning structures such as *kd*-trees for stratifying the columns of A. In addition to improving sample efficiency, stratification has the added advantage over length-squared sampling of being able to separate out columns with dominant values from those with finer features so that all are properly represented in the subsampled matrix.

Stratified empirical Monte Carlo yields an algorithm with error bounds analagous to those of [2], but with sample efficiency that is both theoretically and empirically much better.

Results. We present results showing that 1) our method empirically meets its theoretical error guarantees, 2) error vs. sample size is much better using stratified empirical Monte Carlo than using flat sampling schemes without empirical error control, and 3) our approximation demonstrates orders-of-magnitude speedup over exact SVD.

References

[1] A. Frieze, R. Kannan, and S. Vempala. Fast monte-carlo algorithms for finding low-rank approximations. *Journal of the ACM (JACM)*, 51(6):1025–1041, 1998.

[2] P. Drineas, R. Kannan, and M. W. Mahoney. Fast monte carlo algorithms for matrices ii: Computing a low-rank approximation to a matrix. *SIAM Journal on Computing*, 36:158–183, 2006.

[3] P. Drineas, E. Drinea, and P. S. Huggins. An experimental evaluation of a monte-carlo algorithm for singular value decomposition. *Lectures Notes in Computer Science*, 2563:279–296, 2003.