GPU Rank-one Update Eigenvalue Initialization

Daniel Ringwalt

November 16, 2022

1 Background

Positive definite matrices of different sizes are useful in a model where variables can be added or removed. The positive definite matrix is the covariance matrix of some multivariate normal distribution. With data available offline (full data matrix, or precomputed Gram matrix), efficient rank-one updates (adding a data column) have been neglected for sparse PCA models (which use a small subset of variables). From the small data matrix

Given squared singular values (eigenvalues) accurate to machine precision, the augmented eigenvectors (rotated by the original data's singular vectors) were already computed efficiently from a matrix multiplication identity (Bunch 1978).

We have Σ, which is a diagonal singular values matrix. Our original, full-rank data matrix is augmented with a column of zeros, so Σ has rank $n - 1$ because the last row and column are zero. For a new column of data, there is a $\mathbb{R}^n$ vector update v to the last column of Σ, and new additional left and right singular vector, to reproduce the data. Bunch analyzed the eigenvalues and eigenvectors of $\Sigma'\Sigma'^T = \Sigma^2 + vv^T$, which is a dense matrix.

Note that the matrix $\Sigma'^T\Sigma' = D'$ is sparse and positive definite, with the same eigenvalues as Bunch's matrix. It has the following nonzero entries:

$$D' = \begin{pmatrix} d_1 & 0 & 0 & w_1 \\ 0 & d_2 & 0 & w_2 \\ 0 & 0 & d_3 & w_3 \\ w_1 & w_2 & w_3 & w_4 \end{pmatrix}$$

SVD on sparse Σ' was performed by Ross 2008, but assumed a highly rectangular data matrix, so a naive platform-provided SVD on square Σ' sufficed. The singular vectors of Σ' are dense. We could try to decompose Σ' into n rank-one matrices which have orthogonal ranges (vv^T is not orthogonal to the standard basis of Σ), but we would immediately be dealing with n dense left and right singular vectors. Most useful similarity transformations (e.g. Householder) would also densify the Σ' or D' matrix after one or two steps. However, we can multiply D' by $Q_1 D' Q_2$, without explicitly solving for the similarity transformation

matrix, preserving the eigenvalues but not the eigenvectors. One Givens rotation from the left side can eliminate w_1 while introducing an upper off-diagonal entry, while a Givens rotation from the right side can eliminate the sole upper off-diagonal entry without affecting the w column.

Eigenvalue perturbation

We have an analytic formula (Bunch 1978) for the eigenvalues of D', $n-1$ of which lie between distinct open intervals found between $0, d_1, d_2, d_3$. We can pigeonhole the largest eigenvalue into the interval $(d_3, \text{Tr } D')$. Bunch's formula is a rational function with n zeros (where the updated eigenvalues are found) and n poles. Bunch's zero-finding iterative algorithm should be used when there is an initialization for the updated eigenvalues. The initialization should not be separated from the actual updated eigenvalue by a pole. However, various Newton's methods (including Padé approximation) can skip over the pole and find a different eigenvalue than intended, even if the initialization was closer to the intended eigenvalue than it is to any other pole or zero. Therefore, the initialization algorithm must be good-quality.

2 Eigenvalue initialization algorithm

We can use Givens rotations on both sides to efficiently rotate the entries of D' to upper triangular. We want $n-1$ matrices on the LHS which progressively push down the w entries in the last column. The last column of D' may be isolated, and the effect of the series of Givens rotations will be computed in parallel.

Now, suppose that we have applied a Givens rotation to zero out D'_{in} (the top right entry). We need to consider two progressive 2x2 rotations affecting certain diagonal entries: $d_i \to d'_i \to d''_i$

Algorithm for last column Givens rotations

First, d_1 will only have one 2x2 rotation touching it, so initialize $d'_1 = d_1$.

Next, consider $d_i, i > 1$. We have an angle: $\sin\theta_i = \frac{w'_{i-1}}{w'_i}$. For w'_2, we formed the hypotenuse of the two entries, and by reversing the angle (which is in the first quadrant; both positive), we push $\sin\frac{w_1}{\sqrt{w_1^2+w_2^2}}$ to zero. After one Givens rotation on the LHS of the matrix, we have $w'_2 = \sqrt{w_1^2+w_2^2}$. On the GPU, we calculate the first non-zero entry of the w column after N Givens rotations, by taking the cumsum of w_i^2. Only one value in this array may be used at any given time. We materialize the results, but at any given point in time in our algorithm,

To simplify this stage, we can apply a diagonal matrix with 1 or -1 entries to the LHS of the matrix first. The w_i entries will be nonnegative, so we have a $0 \le \theta_i \le \frac{\pi}{2}, \cos\theta_i \ge 0, \sin\theta_i \ge 0$.

Intermediate result for diagonal entries

Apply the 2x2 rotation to the 2x2 block:

$$\begin{pmatrix}\cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i\end{pmatrix}\begin{pmatrix}d_{i-1} & 0 \\ 0 & d_i\end{pmatrix} = \begin{pmatrix}\ell_1 & r_1 \\ \ell_2 & r_2\end{pmatrix} = \begin{pmatrix}d'_{i-1}\cos\theta_i & -d_i\sin\theta_i \\ d'_{i-1}\sin\theta_i & d_i\cos\theta_i\end{pmatrix}$$

Before writing our updates (d''_{i-1} and d'_i), we need r_1 to go to zero. Apply a 2x2 rotation matrix to the adjoint:

$$\begin{pmatrix}\cos\phi_i & -\sin\phi_i \\ \sin\phi_i & \cos\phi_i\end{pmatrix}\begin{pmatrix}\ell_1 & r_1 \\ \ell_2 & r_2\end{pmatrix}^T = \begin{pmatrix}d''_{i-1} & 0 \\ * & d'_i\end{pmatrix}^T$$

We expect: $d''_{i-1} = \sqrt{\ell_1^2 + r_1^2}$

Apply the reverse rotation matrix to the transposed entries:

$$\begin{pmatrix}\cos\phi_i & -\sin(-\phi_i) \\ \sin(-\phi_i) & \cos\phi_i\end{pmatrix}\begin{pmatrix}d''_{i-1} \\ 0\end{pmatrix} = \begin{pmatrix}\ell_1 \\ r_1\end{pmatrix}$$

Therefore: $\cos\phi_i = \frac{\ell_1}{d''_{i-1}}$

Our last step is to solve for d'_i:

$$\begin{pmatrix}\cos\phi_i & -\sin\phi_i \\ \sin\phi_i & \cos\phi_i\end{pmatrix}\begin{pmatrix}\ell_2 \\ r_2\end{pmatrix} = \begin{pmatrix}* \\ d'_i\end{pmatrix}$$

Using the results of the two rotations:

$$d'_i = \ell_2\sin\phi_i + r_2\cos\phi_i = d'_{i-1}\sin\theta_i\sin\phi_i + d_i\cos\theta_i\cos\phi_i$$

Multiply through by the hypotenuse of the ϕ_i trigonometric functions (d''_{i-1}).

$$d'_i\sqrt{d'^2_{i-1}\cos^2\theta_i + d_i^2\sin^2\theta_i} = d'_{i-1}r_1\sin\theta_i + d_i\ell_1\cos\theta_i$$

$$d'_i\sqrt{d'^2_{i-1}\cos^2\theta_i + d_i^2\sin^2\theta_i} = -\frac{d'_{i-1}d_iw_{i-1}}{w'_i} + \frac{d_id'_{i-1}w_i}{w'_i} = \frac{d_id'_{i-1}}{w'_i}(w_i - w_{i-1})$$

Square both sides.

$$d'^2_i = \frac{d_i^2d'^2_{i-1}}{w'^2_i(d'^2_{i-1}\cos^2\theta_i + d_i^2\sin^2\theta_i)}(w_i - w_{i-1})^2$$

Evaluate:

$$d'^2_{i-1}\cos^2\theta_i + d_i^2\sin^2\theta_i = \frac{d'^2_{i-1}w_i^2 + d_i^2w'^2_{i-1}}{w'^2_i}$$

Therefore, the explicit use of the trigonometric function will be avoided:

$$d'^2_i = \frac{d_i^2d'^2_{i-1}}{d'^2_{i-1}w_i^2 + d_i^2w'^2_{i-1}}(w_i - w_{i-1})^2$$

Harmonic mean

The harmonic mean of two numbers a, b is: $\frac{2ab}{a+b}$. The formula of squared diagonal entries cannot be approximated well by successive harmonic means. However, additional steps are needed - we cannot just take cumulative harmonic means using a parallel algorithm, without weighting/scaling.

Use weighted $\omega_a d'^2_{i-1}$ and $\omega_b d_i^2$, and find the harmonic mean of this expression.

$$\frac{\omega_a + \omega_b}{\frac{\omega_a}{d'^2_{i-1}} + \frac{\omega_b}{d_i^2}} = \frac{(\omega_a + \omega_b) d'^2_{i-1} d_i^2}{\omega_a d_i^2 + \omega_b d'^2_{i-1}}$$

We produce a formula for $1/d'^2_i$ based on harmonic mean. Take:

$$\frac{w'^2_{i-1}}{d'^2_{i-1}}, \frac{w_i^2}{d_i^2}$$

Then, take the arithmetic sum of these values. For d'^2_i, we have an additional scale factor after taking the harmonic mean:

$$d'^2_i = \left(\frac{d_i^2 d'^2_{i-1} (w_i^2 + w'^2_{i-1})}{d'^2_{i-1} w_i^2 + d_i^2 w'^2_{i-1}} \right) \left(\frac{(w_i - w_{i-1})^2}{w_i^2 + w'^2_{i-1}} \right)$$

Using a relationship between sum of reciprocals and the weighted harmonic mean:

$$\frac{1}{d'^2_i} = \left(\frac{w'^2_{i-1}}{d'^2_{i-1}} + \frac{w_i^2}{d_i^2} \right) \left(\frac{(w_i - w_{i-1})^2}{w_i^2 + w'^2_{i-1}} \right)$$

On the reciprocals array, there is a weight multiplier which will increase progressive, so that we can take the cumsum of $\frac{1}{d'^2_i} \Pi_i \text{scale}_i$ (and then divide each result by a normalizing value). The successive weights should use a cumsum in log-space.

Eigenvalue formula

We found:

$$\lambda_{i-1} = d''_{i-1} = \frac{\sqrt{d'^2_{i-1} w_i + d_i^2 w'_{i-1}}}{w'_i}$$

The final eigenvalue will be calculated from the trace and $n-1$ eigenvalues.

Sign of eigenvalue

During this algorithm, we operated on nonnegative (squared) values. Our system will assert a positive definite matrix (the matrix before rank-one update may be positive semidefinite, but its leading $n-1$ x $n-1$ block will be positive

definite). We can add additional assertions, analytically finding the phase of ϕ element-wise using the arrays created in parallel so far. Therefore, we could find fatal errors such as an eigenvalue which analytically should take the negative square root.

3 Analysis

The log, reciprocal, and weighting steps (where the exponent of the weight may grow large) will reduce the significant figures of the true diagonal entries (eigenvalues). We will benchmark our method over covariance matrices taken on random data. Our method's accuracy should have significant figures at least one greater than the relative separation between eigenvalues in the data. This would make it a worthwhile initialization step so that we can assume that Bunch's zero-finding will find the true eigenvalues, to machine precision.