Approximate Matrix Multiplication and Space Partitioning Trees: An Exploration

N.P. Slagle, Lance J. Fortnow

October 23, 2012

Abstract

Herein we explore a dual tree algorithm for matrix multiplication of $A \in \mathbb{R}^{M\times D}$ and $B \in \mathbb{R}^{D\times N}$, very narrowly effective if the normalized rows of A and columns of B, treated as vectors in $\mathbb{R}^D$, fall into clusters of order proportionate to $\Omega(D^\tau)$ with radii less than $\arcsin(\epsilon/\sqrt{2})$ on the surface of the unit D-ball. The algorithm leverages a pruning rule necessary to guarantee ϵ precision proportionate to vector magnitude products in the resultant matrix. *Unfortunately, if the rows and columns are uniformly distributed on the surface of the unit D-ball, then the expected points per required cluster approaches zero exponentially fast in D; thus, the approach requires a great deal of work to pass muster.*

1 Introduction and Related Work

Matrix multiplication, ubiquitous in computing, naively requires $O(MDN)$ floating point operations to multiply together matrices $A \in \mathbb{R}^{M\times D}$ and $B \in \mathbb{R}^{D\times N}$. We present an investigation of our novel approach to matrix multiplication after a brief discussion of related work and an explanation of space-partitioning trees.

1.1 State-of-the-Art for Square Matrices

For $N = D = M$, Strassen [12] gave an $O(N^{\log_2 7})$ algorithm that partitions the matrices into blocks, generalizing the notion that to multiply binary integers a and b, one need only compute $[(a+b)^2 - (a-b)^2]/4$, an operation requiring three additions, two squares, and a left shift. Several improvements appear in the literature [1],[6],[8],[10], the most recent of which give $O(N^{2.3736\ldots})$ [11] and $O(N^{2.3727\ldots})$ [13], both augmentations of the Coppersmith-Winograd algorithm [2]. The latest algorithms feature constants sufficiently large to preclude application on modern hardware [9].

1.2 Motivating the Space

The product of A in $\mathbb{R}^{M\times D}$ and B in $\mathbb{R}^{D\times N}$ features all possible inner products between the row vectors of A and the column vectors of B, each an element of $\mathbb{R}^D$. We investigate whether organizing these two sets of vectors into *space-partitioning trees* can reduce the complexity of the naïve matrix multiplication by exploiting the distribution of the data.

1.2.1 Space-Partitioning Trees

We can organize a finite collection of points $\mathcal{S}$ in Euclidian space $\mathbb{R}^D$ into a space-partitioning tree $\mathcal{T}$ such that the root node $\mathcal{P}_0$ contains all points in $\mathcal{S}$, and for any other node $\mathcal{P}$ in $\mathcal{T}$, all points in $\mathcal{P}$ are in $\pi(\mathcal{P})$, the parent node of $\mathcal{P}$. Figure 1 depicts a space-partitioning tree in $\mathbb{R}^2$.

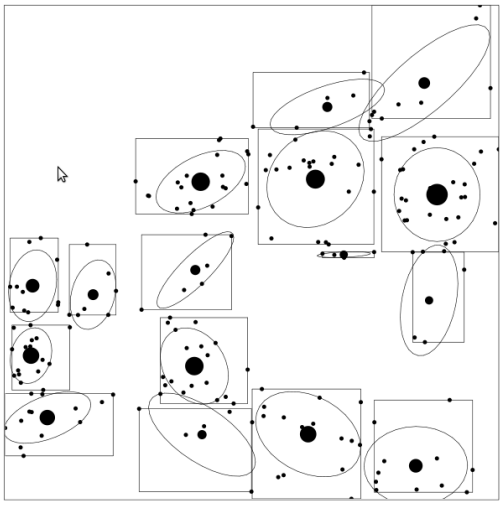

Figure 1: A space-partitioning tree in $\mathbb{R}^2$; the ellipses denote covariances on the node points; the large dots denote node centroids.

A space-partitioning tree definition requires a recursive partitioning rule, such as that appearing in algorithm 1. Organizing $\mathcal{S}$ into such a tree generally requires $O(D|\mathcal{S}|\log(D|\mathcal{S}|))$ time complexity.

Algorithm 1 $[\mathcal{L}, \mathcal{R}]$ =partition$(\mathcal{P}, m)$

1: If $|\mathcal{P}| \leq m$, then **RETURN** $[NULL, NULL]$.
2: Pick the dimension k that maximizes the range of x_k for $x \in \mathcal{P}$.
3: Sort the points in $\mathcal{P}$ according to dimension k.
4: Split $\mathcal{P}$ into $\mathcal{L}$ and $\mathcal{R}$ using the median (or mean) of x_k.
5. **RETURN** $[\mathcal{L}, \mathcal{R}]$.

1.2.2 Dual Tree Algorithm

Given a reference tree $\mathcal{R}$ and a query tree $\mathcal{Q}$ of data points, we can perform pairwise operations such as kernel summations and inner products across across nodes rather than points, performing a depth-first search on both trees. The algorithm leverages a pruning criterion to guarantee ϵ level approximation in the outputs. Algorithm 2 exhibits this approach.

Algorithm 2 dualTreeCompareNodes($\mathcal{R}, \mathcal{Q}$, operation *op*, pruning rule $R, \epsilon, \widehat{C}$)

1: If $\mathcal{R}$ and $\mathcal{Q}$ are leaf nodes, then perform the point-wise operation, filling in appropriate entries of $\widehat{C}$. **RETURN**
2: If rule $R(\mathcal{R}, \mathcal{Q})$ is true, approximate *op* between points in the nodes using their centroids, filling in appropriate entries of $\widehat{C}$; then **RETURN**.
3: Call

- dualTreeCompareNodes($\mathcal{R}.left, \mathcal{Q}.left$)
- dualTreeCompareNodes($\mathcal{R}.left, \mathcal{Q}.right$)
- dualTreeCompareNodes($\mathcal{R}.right, \mathcal{Q}.left$)
- dualTreeCompareNodes($\mathcal{R}.right, \mathcal{Q}.right$)

1.2.3 Space-Partitioning Trees in the Literature

Applied statistical methods such as dual tree approximate kernel summations [3], [4], [5] and other pairwise statistical problems [7] partition the query and test samples into respective space-partitioning trees for efficient look-ups. Using cover trees, Ram [7] demonstrates linear time complexity for naïve $O(N^2)$ pairwise algorithms.

2 Dual Tree Investigation

2.1 Product Matrix Entries

Given the two matrices $A \in \mathbb{R}^{M \times D}$ and $B \in \mathbb{R}^{D \times N}$, we can think of the entries of $C = AB$ as $c_{ij} = |a_i||b_j|\cos\theta_{ij}$, where a_i is the ith row of A, b_j is the jth column of B, and θ_{ij} is the angle between a_i and b_j. We can compute the magnitudes of these vectors in time $O(D(M+N))$ and all products of the magnitudes in time $O(MN)$, for a total time complexity of $O(MN + D(M+N))$. Thus, computing the cosines of the angles for $M, N \in O(D)$ is the $O(MDN)$ bottleneck. We give narrow conditions under which we can reduce this complexity.

2.2 Algorithm

In our investigation, we normalize the row vectors of A and the column vectors of B, then organize each set into a ball tree, a space-partitioning tree such that each node is a D-ball. To compute the cosines of the angles between all pairs, we apply the dual tree algorithm. The pruning rule must guarantee that the relative error of our estimate $\widehat{c}_{ij}$ with respect to the full magnitude $|a_i||b_j|$ be no more than ϵ, or, more formally,

$$|c_{ij} - \widehat{c}_{ij}| \leq \epsilon |a_i||b_j|. \tag{1}$$

Thus, we require

$$|\cos\theta_{ij} - \cos\widehat{\theta}_{ij}| \leq \epsilon. \tag{2}$$

The pruning rule guaranteeing the above error bound appears in algorithm 3.

Algorithm 3 dualTreeMatrixMultiplication(A, B, ϵ)

1: Allocate $M \times N$ matrix $\widehat{C}$.
2: Compute the magnitudes of a_i and b_j for $i = 1, \ldots, M$, $j = 1, \ldots, N$.
3: Fill in $\widehat{C}$ so that $\widehat{c}_{ij} = |a_i||b_j|$.
4: Compute $u_i = a_i/|a_i|$, $v_j = b_j/|b_j|$.
5: Allocate trees $\mathcal{U}$ and $\mathcal{V}$ with root($\mathcal{U}$) = $\{u_i\}$ and root($\mathcal{V}$) = $\{v_j\}$.
6: Call partition(root($\mathcal{U}$), $size$), partition(root($\mathcal{V}$), $size$), with $size$ the minimum number of points (defaulted to one) per tree node.
7: Let $op(s, t) =< s, t >$.
8: For node balls $\mathcal{R} \in \mathcal{U}$, $\mathcal{Q} \in \mathcal{V}$, define

- α:=angle between the centers of $\mathcal{R}$,$\mathcal{Q}$,
- β:=angle subtending half of the node ball $\mathcal{R}$, and
- γ:=angle subtending half of the node ball $\mathcal{Q}$,

all angles in $[0, \pi]$.
9: Define the pruning rule R as an evaluation of $|\beta + \gamma| \leq \frac{\epsilon}{|\sin\alpha|+|\cos\alpha|}$.
10: Call dualTreeCompareNodes(root($\mathcal{U}$),root($\mathcal{V}$), $op, R, \epsilon, \widehat{C}$).
11: **RETURN** $\widehat{C}$.

We could define a more conservative pruning rule of $|\beta + \gamma| \leq \epsilon/\sqrt{2} \leq \frac{\epsilon}{|\sin\alpha|+|\cos\alpha|}$. For future analyses, we apply the more conservative bound.

Figure 2 exhibits the relationships between angles α, β, and γ.

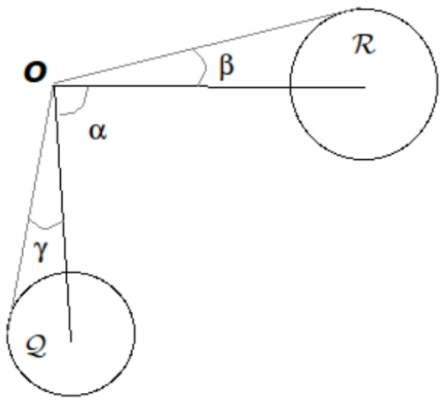

Figure 2: Angles in algorithm 3.

2.2.1 Proof of the Pruning Rule

Simply put, the pruning rule in algorithm 3 bounds the largest possible error on the cosine function in terms of the center-to-center angle (our approximation) and the angles subtending the balls $\mathcal{R}$ and $\mathcal{Q}$, formally stated in theorem 1.

Theorem 1. *Given ball nodes $\mathcal{R}$ and $\mathcal{Q}$ and angles as defined in algorithm 3, if $\beta+\gamma \leq \frac{\epsilon}{|\sin\alpha|+|\cos\alpha|}$, then $|r \cdot q - \cos\alpha| \leq \epsilon$ for all $r \in \mathcal{R}$, $q \in \mathcal{Q}$.*

To prove theorem 1, we need the following lemma.

Lemma 2. *Given both the ball nodes $\mathcal{R}$ and $\mathcal{Q}$ and angles listed in theorem 1, let error$(r, q) = |r \cdot q - \cos\alpha|$. The maximum of error occurs when r and q are in the span of the two centers of $\mathcal{R}$ and $\mathcal{Q}$. Furthermore, the maxima of error are $|\cos(\alpha \widehat{\mp} \beta \mp \gamma) - \cos\alpha|$.*

Proof. Let $\bar{r}$ and $\bar{q}$ be the centers of $\mathcal{R}$ and $\mathcal{Q}$, respectively. Since $\cos\theta$ is monotone for $\theta \in [0, \pi]$, the extrema of the error function occur when r and q fall on the surface of $\mathcal{R}$ and $\mathcal{Q}$, respectively. Furthermore, we only care about the extrema of $r \cdot q$ since the maxima and minima of this function bound the error about $\cos\alpha$. Thus, we optimize $r \cdot q$ subject to $\bar{r} \cdot \bar{q} = \cos\alpha$, $\bar{r} \cdot r = \cos\beta$, $\bar{q} \cdot q = \cos\gamma$, and $r \cdot r = q \cdot q = \bar{r} \cdot \bar{r} = \bar{q} \cdot \bar{q} = 1$.

Leveraging Lagrange multipliers, we obtain the solutions

$$r = \bar{r}[\cos\beta \widehat{\mp} \cot\alpha \sin\beta] + \bar{q}\left[\widehat{\pm}\frac{\sin\beta}{\sin\alpha}\right]$$

and

$$q = \bar{r}\left[\pm\frac{\sin\gamma}{\sin\alpha}\right] + \bar{q}[\cos\gamma \mp \cot\alpha \sin\gamma],$$

with

$$r \cdot q = \widehat{\mp} \pm \cos\alpha \sin\beta \sin\gamma + \cos\alpha \cos\beta \cos\gamma \pm \sin\alpha \cos\beta \sin\gamma \widehat{\pm} \sin\alpha \sin\beta \cos\gamma = \cos(\alpha \widehat{\mp} \beta \mp \gamma).$$

□

Notice, the possible values of r and q maximizing the error are simply the edges of the cones subtending balls $\mathcal{R}$ and $\mathcal{Q}$ in the hyperplane spanned by $\bar{r}$ and $\bar{q}$. Now, we prove theorem 1.

Proof. By hypothesis, $|\beta + \gamma|\,[|\sin\alpha| + |\cos\alpha|] \leq \epsilon$. Since $|\beta + \gamma| \geq |\widehat{\mp}\beta \mp \gamma|$, $|\sin h| \leq |h|$, $|1 - \cos h| \leq |h|$ for $\beta, \gamma \in [0, \pi]$ and $h \in [-\pi, \pi]$, we have

$$\epsilon \geq |\widehat{\mp}\beta \mp \gamma|\left[|\sin\alpha|\left|\frac{\sin(\widehat{\mp}\beta \mp \gamma)}{\widehat{\mp}\beta \mp \gamma}\right| + |\cos\alpha|\left|\frac{1 - \cos(\widehat{\mp}\beta \mp \gamma)}{\widehat{\mp}\beta \mp \gamma}\right|\right] \geq |\cos\alpha\cos(\widehat{\mp}\beta \mp \gamma) - \sin\alpha\sin(\widehat{\mp}\beta \mp \gamma) - \cos\alpha|,$$

and so

$$\epsilon \geq |\cos\alpha\cos(\widehat{\mp}\beta \mp \gamma) - \sin\alpha\sin(\widehat{\mp}\beta \mp \gamma) - \cos\alpha| = |\cos(\alpha\widehat{\mp}\beta \mp \gamma) - \cos\alpha|.$$

□

2.2.2 Analysis of Algorithm 3

Given matrices A in $\mathbb{R}^{M\times D}$ and B in $\mathbb{R}^{D\times N}$, computing the magnitudes, normalizing the rows of A and columns of B, and computing magnitude products for $\widehat{C}$ requires $O(D(M+N)+MN)$. Organizing the normalized points into space-partitioning trees requires $O(MD\log MD + ND\log ND)$. Finally, an analysis of the dual tree algorithm requires conditions on the data points. We suppose that given the approximation constant ϵ, the number of points falling in node balls of appropriate size, say radius roughly $\arcsin(\epsilon/\sqrt{2})$, is bounded below by $f_D(\epsilon)$. If the points are clustered into such balls, each prune saves the computation of at least $D[f_D(\epsilon)]^2$. So we can fill into $\widehat{C}$ $[f_D(\epsilon)]^2$ entries with a constant number of inner products at cost $O(D)$, for a total complexity of $O(MDN/[f_D(\epsilon)]^2)$. Thus, we have the following theorem.

Theorem 3. *The total time complexity of algorithm 3 is $O(MD\log MD + ND\log ND + MN(1 + D/[f_D(\epsilon)]^2))$.*

2.3 Gaping Caveat

An obvious caveat in the analysis is the behavior of $f_D(\epsilon)$ as D increases without bound. For a rough sketch of the expected behavior of f_D, recall that the volume and surface area of a D-ball of radius r are

$$V_D(r) = \frac{2^D \pi^{\frac{D-1}{2}} \Gamma\left(\frac{D+1}{2}\right)}{D\Gamma(D)} r^D \tag{3}$$

and

$$SA_D(r) = V_D\prime(r) = \frac{2^D \pi^{\frac{D-1}{2}} \Gamma\left(\frac{D+1}{2}\right)}{\Gamma(D)} r^{D-1}. \tag{4}$$

Since uniformly distributed data represents something of a worst-case scenario with respect to clustering algorithms, we explore the expected cluster sizes by dividing the surface of the unit D-ball by the node balls of appropriate size.

Theorem 4. *Assuming that the normalized rows of A and columns of B are uniformly distributed about the unit D-ball, let W be the number of points in each ball of radius* $\arcsin(\epsilon/\sqrt{2})$. *Then*

$$\mathbb{E}[W] \approx MN \frac{V_{D-1}(\arcsin(\epsilon/\sqrt{2}))}{SA_D(1)} = \frac{MN\Gamma\left(\frac{D}{2}\right)}{2\sqrt{\pi}\Gamma\left(\frac{D+1}{2}\right)} \arcsin^{D-1}(\epsilon/\sqrt{2})$$

Thus, since $\Gamma(x+1/2)/\Gamma(x) \in O(x)$, exponentially few points fall into each ball of radius $\arcsin(\epsilon/\sqrt{2})$. Thus, we require strong clustering conditions, stated formally below, if the dual tree approach described in algorithm 3 is to defeat naïve matrix multiplication.

Theorem 5. *If $M = D = N$ and the normalized rows of A and columns of B form clusters of size $\Omega(D^\tau)$ for $\tau > 0$ where cluster radii are approximately* $\arcsin(\epsilon/\sqrt{2})$ *for $\sqrt{2} > \epsilon > 0$, then algorithm 3 runs in time $O(D^2 \log D + D^{3-2\tau})$.*

3 Concluding Remarks and Future Work

Given the problem of multiplying together matrices $A \in \mathbb{R}^{M\times D}$ and $B \in \mathbb{R}^{D\times N}$, we present a dual tree algorithm effective if row vectors of the left matrix and column vectors of the right matrix fall into clusters of size proportionate to some positive power τ of the dimension D of said vectors. Unfortunately, worst-case uniformly distributed vectors give exponentially small cluster sizes. Possible improvements include partitioning columns of A and rows of B so that the size of clusters increases slightly while incurring a greater cost in tree construction and the number of magnitudes to calculate, or appealing to the asymptotic orthogonality of vectors as D becomes arbitrarily large. Clearly, the approach needs a great deal of work to be of practical interest.

4 References

1. D. Bini, M. Capovani, F. Romani, and G. Lotti. $O(n^{2.7799})$ complexity for $n \times n$ approximate matrix multiplication. *Inf. Process. Lett.*, 8(5):234235, 1979.
2. D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *J. Symbolic Computation*, 9(3):251280, 1990.
3. A.G. Gray and A.W. Moore. N-Body Problems in Statistical Learning. T.K. Leen, T.G. Dietterich, and V. Tresp, editors, *Advances in Information Processing Systems 13 (December 2000)*. MIT Press, 2001.

4. A.G. Gray and A. W. Moore. Rapid Evaluation of Multiple Density Models. In *Artificial Intelligence and Statistics 2003*, 2003.
5. M.P. Holmes, A.G. Gray, and C.L. Isbell Jr. Fast Kernel Conditional Density Estimation: A Dual Tree Monte Carlo Approach. *Computational Statistics and Data Analysis*, 1707-1718, 2010.
6. V. Y. Pan. Strassens algorithm is not optimal. In Proc. FOCS, volume 19, pages 166176, 1978.
7. P. Ram, D. Lee, W. March, A.G. Gray. Linear-time Algorithms for Pairwise Statistical Problems, *NIPS*, 2010.
8. F. Romani. Some properties of disjoint sums of tensors related to matrix multiplication, *SIAM J. Comput.*, pages 263267, 1982.
9. S. Robinson. Toward an Optimal Algorithm for Matrix Multiplication, *SIAM News 38 (9)*, 2005.
10. A. Schönhage. Partial and total matrix multiplication. SIAM J. Comput., 10(3):434455, 1981.
11. A. Stothers. Ph.D. Thesis, U. Edinburgh, 2010.
12. V. Strassen. Gaussian Elimination is not Optimal, *Numer. Math. 13*, p. 354-356, 1969.
13. V.V. Williams. Multiplying matrices faster than Coppersmith-Winograd, *STOC*, 2012.