Approximating Tree Inversion for Matrix Function Estimation

Brian Goldman

May 2015

1 Introduction

In *Methods for Estimating the Diagonal of Matrix Functions*, Jesse Laeuchli seeks to derive unbiased, low-variance estimators for $Tr(A^{-1})$ to be calculated in $O(n)$ time, where A is a symmetric $n \times n$ matrix. He begins by establishing that if $\mathbf{x} \in R^n$ s.t. $x_i = 1$ or $x_i = -1$ with equal probability, then $E[\mathbf{x}'A\mathbf{x}] = E[\sum_{i=1}^{n}\sum_{j=1}^{n} x_i a_{ij} x_j] = tr(A)$, since diagonal entries will always have coefficient 1, but off-diagonal entries will have coefficient 1 or -1 with equal probability. Likewise, $E[\mathbf{x}'A^{-1}\mathbf{x}] = Tr(A^{-1})$. Since we don't have access to A^{-1}, we can compute this quadratic form by first setting $A\mathbf{y} = \mathbf{x}$, solving for $\mathbf{y}$ using an iterative method, and then calculating $\mathbf{x}'\mathbf{y} = \mathbf{x}'A^{-1}\mathbf{x}$.

Laeuchli proceeds by determining other vectors $\mathbf{x}$ such that $\mathbf{x}'A^{-1}\mathbf{x}$ is still an unbiased estimator for $Tr(A^{-1})$, but with lower variance. By appropriately choosing our vectors $\mathbf{x}$, we can eliminate the contribution to our estimator's variance from particular entries of A^{-1}. Since that largest entries in A^{-1} contribute the most to variance, we would like to predict where they are. Laeuchli hypothesizes that if T is a maximum (undirected) spanning tree of the weighted graph underlying A, then, for certain matrices, the largest values of A^{-1} tend to be in the same locations as the largest values of T^{-1}.

Although his initial results are promising, Laeuchli raises a few concerns. In the examples considered so far, T is ill-conditioned. Further, the calculated inverse T^{-1} is sparse– so we could find perhaps only a handful of large values A^{-1} using the calculated inverse.[1] Finally, inversion is slow, but since T is sparse, we may be able to develop a quicker way to invert it.

2 A Paradox

We begin by noting that, for sufficiently large n, an undirected tree T is almost always singular. In particular, suppose T has two leaves i, j which share a single

[1] But we might resolve this problem by considering powers of T^{-1} instead of just T^{-1} itself.

neighbor k. Then $\mathbf{t}_i = t_{ki}\mathbf{e}_k$ and $\mathbf{t}_j = t_{kj}\mathbf{e}_k$, where $\mathbf{t}_i$ is the i_{th} column in T and t_{ki} is the (k, i) entry in T. That is, $\mathbf{t}_i$ and $\mathbf{t}_j$ are scalar multiples of one another. While I don't know the exact probability of an undirected tree being singular, I'd guess it's close to zero for all but the smallest n. Every time I inverted a tree while working on this project, MATLAB output an ill-conditioning warning. And every time I took the determinant of a tree, I got zero.

How, then, could MATLAB's $inv(T)$ predict anything about A^{-1} is T is singular? After reading the documentation and some experimentation, I found that $inv(T)$ should be an approximate right-inverse to T. Recall that if $X = T^{-1}$, $T\mathbf{x_i} = \mathbf{e_i}$. If there does not exist such an X, we may want X such that $T\mathbf{x_i} \approx \mathbf{e_i}$ for all i. More precisely, we want to solve the least squares problems $min||T\mathbf{x_i} - \mathbf{e_i}||$.

Unfortunately, MATLAB doesn't solve these problems very well. For many i, MATLAB finds x_i such that $T\mathbf{x_i} = \mathbf{e_i}$. But for the remaining indices, MATLAB finds $\mathbf{x_i}$ such that $||T\mathbf{x_i} - \mathbf{e_i}|| \approx 1$ or $||T\mathbf{x_i} - \mathbf{e_i}|| > 1$. Of course, we can always achieve $||T\mathbf{x_i} - \mathbf{e_i}|| = 1$ by setting $\mathbf{x_i} = 0$. But even when MATLAB finds a solution such that $\mathbf{x_i}$ such that $||T\mathbf{x_i} - \mathbf{e_i}|| \approx 1$, it usually does so by choosing linear combination of linearly dependent columns in T.

Finally, we observe that MATLAB's calculated inverse $inv(T)$ is sparse with many elements on the main diagonal, even when T is a random tree.

3 Perfect Column Solutions

Given these problems, I wondered if I could write a more accurate, faster inverse approximation algorithm with which we could better test Laeuchli's initial hypothesis. To begin, note that if a vertex i in our tree is connected to a leaf j, we can solve $T\mathbf{x_i} = \mathbf{e_i}$ by setting $\mathbf{x_i} = \mathbf{e_j}/t_{ij}$ since $\mathbf{t_j} = t_{ij}\mathbf{e_i}$. We'll mark vertex i to remember that we've already solved $T\mathbf{x_i} = \mathbf{e_i}$.

After we have done this for all vertices adjacent to leaves, suppose a vertex i is adjacent to another vertex, j, such that all of j's neighbors have already been marked except for i. If we set $\mathbf{x_i} = \mathbf{e_j}/t_{ij}$, then the i_{th} entry in $T\mathbf{x_i}$ will be 1, as desired, but we will have non-zero entries t_{kj}/t_{ij} for each of j's neighbors $k \neq i$. Fortunately, we have already solved $T\mathbf{x_k} = \mathbf{e_k}$, so we can cancel those non-zero entries by setting $\mathbf{x_i} = \mathbf{e_j}/t_{ij} - \sum \frac{t_{kj}}{t_{ij}}\mathbf{x_k}$ so that $T\mathbf{x_i} = \mathbf{e_i}$.

Suppose we track how many unmarked neighbors each vertex still has. Whenever a vertex is marked, we subtract 1 from the remaining degree of each of its neighbors. When a vertex j has only one remaining unmarked neighbor i, we know that we can solve for $\mathbf{x}_i$ so that $T\mathbf{x_i} = \mathbf{e_i}$. Since T has $2n - 2$ directed edges, tracking degrees will use $O(n)$ time. And, this procedure guarantees that we will consider each equation $T\mathbf{x_i} = \mathbf{e_i}$ at most once. But if we want a numeric

solution for each i, we do not guarantee that our algorithm will terminate in $O(n)$ time since solving these equations may take more time for a vertex considered at the end of the algorithm than for one considered earlier (note that our solutions $\mathbf{x_i}$ build recursively from one another). We implemented two versions of this algorithm: one which tracks degrees, and another that may consider solving the equation $T\mathbf{x_i} = \mathbf{e_i}$ several times for each vertex. Surprisingly, the second implementation runs faster. But both are much faster than MATLAB's inversion algorithm.

Our marking algorithm will not necessarily mark every vertex in T. For example, if T is a root and two leaves, the algorithm will only mark the root. Nor is the matrix X that it outputs necessarily unique; rather, it may be sensitive to the order in which the indices are considered.[2] But I suspect that it would find T^{-1} in the rare case that T is invertible. Each time I compared the set of indices i such that $T\mathbf{x_i} = \mathbf{e_i}$ was solved by our marking algorithm with the set of indices i such that $T\mathbf{x_i} = \mathbf{e_i}$ was solved by MATLAB's inversion algorithm, I found they were equal to each other.

4 Approximate Column Solutions

By performing trials over various tree sizes, I found that our marking algorithm finds a solution $T\mathbf{x_i}$ for about 60% of indices i. We'll find an approximate solution to $min||T\mathbf{x_i} - \mathbf{e_i}||$ for the remaining indices, so that the objective value is always less than 1.

For a given node i, if l is one of i's neighbors, then $t_{il} \neq 0$. Otherwise, $t_{il} \neq 0$. Then, let i have neighbors j. We will project $\mathbf{e}_i$ into the space spanned by the vectors $\mathbf{t}_j$. But rather than find an orthogonal basis for this space, we will find the projection using calculus. For each vector $\mathbf{t}_j$, let $\mathbf{s}_j = \mathbf{t}_j - t_{ij}\mathbf{e}_i$. Then, the set of vectors $\mathbf{s}_j$ are mutually orthogonal: Suppose $\mathbf{s}_{j_1}$ and $\mathbf{s}_{j_2}$ each have a non-zero k_{th} entry for some k. Then, both i and k are adjacent to j_1 and j_2 in T, forming a cycle. But T is a tree– contradiction.

Then, by the Pythagorean theorem,

$$||T\mathbf{x_i} - \mathbf{e_i}||^2 \quad = \quad (1 - \sum t_{ij}x_{ij})^2 + \sum ||\mathbf{s}_j||^2 x_{ij}^2$$

where x_{ij} are our decision variables. Setting

$$\frac{\partial}{\partial x_{il}}\left[(1 - \sum t_{ij}x_{ij})^2 + \sum ||\mathbf{s}_j||^2 x_{ij}^2\right] \quad = \quad -2t_{il}(1 - \sum t_{ij}x_{ij}) + 2||\mathbf{s}_l||^2 x_{il} \quad = \quad 0$$

[2] Although I believe the set of marked vertices will always be the same.

we find

$$x_{il} \quad = \quad \frac{t_{il}}{||\mathbf{s}_l||^2}(1 - \sum t_{ij}x_{ij}).$$

If we set $C = 1 - \sum t_{ij}x_{ij}$, then $x_{il} = \frac{t_{il}}{||\mathbf{s}_l||^2}C$. So, $C = 1 - \sum \frac{t_{il}^2}{||\mathbf{s}_l||^2}C$, and $C = 1/(1 + \sum \frac{t_{il}^2}{||\mathbf{s}_l||^2})$.

So, we will begin by computing C according to this last formula, and use C to immediately calculate x_{il} for each neighbor l. We can perform these calculations for all indices i in $O(n)$ time if we store the norms $||\mathbf{s}_l||^2$. Finally, note that we have indeed found a minimum since

$$\frac{\partial^2}{\partial x_{il}^2}\left[(1 - \sum t_{ij}x_{ij})^2 + \sum ||\mathbf{s}_j||^2 x_{ij}^2\right] \quad = \quad 2t_{il}^2 + 2||\mathbf{s}_l||^2 \quad > \quad 0 \text{ for all } l.$$

5 Results

Let X be the approximate right inverse computed by first applying the algorithm found in section 3 and then computing approximate solutions for unmarked indices as described in section 4. We expect $TX \approx I$. But unless T is invertible, we don't make any claim about whether $XT \approx I$. Note, though, that $X'T = X'T' = TX \approx I$, so X' is a corresponding approximate left inverse. We suggest that $(X + X')/2$ may be an appropriate inverse if we don't prefer either direction. Note that $(X+X')/2$ is symmetric (in general, X is not).

I generated 25 random trees with 1000 vertices, each. For each tree T, I computed X, $(X + X')/2$, and MATLAB's $inv(T)$. I pre- and post-multiplied each of these approximate inverses by T, subtracted the identity matrix, and found the Frobenius norm. Although $inv(T)$ is not symmetric, pre- and post-multiplication yielded the same mean error norm: 35.05. Multiplication by $(X + X')/2$ resulted in a mean error norm of 29.02. The mean of $||TX - I||$ was 25.15; the mean of $||XT - I||$ was 39.72.

On average, it took 1.55 seconds to find $inv(T)$ but only .08 seconds to find X.

Finally, we found no evidence that large entries in A^{-1} tended lie in the same places as large entries in X when we chose T to be the maximum spanning tree. In fact, when we generated random $n \times n$ matrices B with no relation to A, probing A with X_A was no more effective than probing A with X_B. The same held for probing with $(X_A + X_A')/2$ and $(X_B + X_B')/2$

6 Discussion

We are left to ask why probing with MATLAB's $inv(T)$ is effective. Perhaps the function's idiosyncratic behavior on singular matrices somehow aligns with a property of the Laplacian matrices Laeuchli was considering. Or perhaps the Laplacians were sufficiently structured that probing with a broad class of matrices would give good results.

Barik et al and Bapat et al discuss conditions for tree invertibility. In particular, they observe that if T is a symmetric unweighted tree, and there exists a unique perfect matching on T, then T^{-1} exists and can be calculated using the matching. A maximum matching on a bipartite graph (such as T) can be calculated in $O(\sqrt{(}n)m) = O(n^{1.5})$ time, where $m = n - 1$ is the number of undirected edges in the graph.

7 Acknowledgements

Thanks to Professor Stathopoulos and Jesse Laeuchli; this project leans heavily on their help and prior work. Thanks also to Professor van Zuylen in the Mathematics department, who suggested tracking how many unmarked neighbors each vertex has in our marking algorithm in order to reduce the run time.

References

Bapat, R.b., and E. Ghorbani. "Inverses of Triangular Matrices and Bipartite Graphs." Linear Algebra and Its Applications 447 (2014): 68-73.

Barik, S., M. Neumann, and S. Pati. "On Nonsingular Trees and a Reciprocal Eigenvalue Property." Linear and Multilinear Algebra 54.6 (2006): 453-65.

Toledo, S. "Vaidya's preconditioners: Implementation and experimental study." Electronic Transactions on Numerical Analysis 16 (2003): 30-49.

Laeuchli, J. "Methods for Estimating the Diagonal of Matrix Functions."