Chapter 1

Ubiquity of Synonymity

This project is based on the paper "Ubiquity of synonymity: almost all large binary trees are not uniquely identified by their spectra or their immanantal polynomials".

This paper shows that three different matrix representations of a binary tree fail to distinguish different trees based on, as the title would suggest, their spectra or their immanantal polynomials.

1.1 Representations

The 3 matrix representations are the Adjacency matrix, the Laplacian matrix, and the Distance matrix.

The adjacency matrix takes the tree as an undirected graph, represented simply as an adjacency matrix.

The laplacian matrix is the adjacency matrix but with the diagonal changed so that the matrix has zero row and column sum.

The distance matrix is a smaller matrix where D_{ij} is the length of the shortest path from the ith leaf to the jth leaf.

All of these representations are one to one representations of the tree except for the distance matrix.

Relabelling vertices in the tree/graph will simultaneously permute corresponding rows and columns of all of these representations.

This means that the immanant and spectrum have a lot of potential as representations, since they can potentially give the same result under permutation, but different results if two trees really have different shapes.

Of course this does not turn out to be the case, which is the point of the paper.

This is shown by making 3 important arguments, from which the failure of these results can be directly inferred.

1.2 Results

The first result of the paper is that adjacency/laplacian matrices with the same spectrum will have the same immanantal polynomials and vice versa.

The second result of the paper is the definition and existence of an exchange property, wherein some trees not only have the same spectrum, but any tree that contain those subtrees will also have the same spectrum. (this result applied to both distance matrices, and to emphlinear combinations of adjacency/laplacian matrices)

The third result is that the proportion of trees that contains a specified subtree will approach 1 as tree size gets larger.

1.3 Conclusions

Clearly from this the paper has shown that the following 5 functions are what we will later define as "near trivial" representations:

- immanantal polynomial of adjacency matrix
- immanantal polynomial of adjacency matrix
- spectrum of adjacency matrix
- spectrum of laplacian matrix
- spectrum of distance matrix

To say that a representation is "near trivial" only says that for large enough trees the representation will have a lot of distinct trees with the same value.

In particular the paper showed that the rate of approach is fairly slow, so as long as your trees are small enough or your use case is sufficiently permissive of collisions, or some combination of the two, these 5 measurements might still be viable.

Otherwise these representations will not help.

1.4 Usage in This Project

The three main points we use from this paper are as follows:

1. Trees have many convenient representations that are invariant under relabelling and many that are able to distinguish trees that have different shape, but not many with both
2. Matrix spectra are a promising way of removing permutations from a matrix without reducing the matrix to a single number
3. If a tree representation has an exchange property for some pair of trees, then the representation will be near trivial for large trees.

Chapter 2

Tree Construction

Trees will be represented as directed graphs, that is a set of ordered pairs.

Definition 1. *Predecessor/Child*
Given a graph G over verteces V, and a vertex $v \in V$, the predecessors of v are the set $P_v \subset V$ with $u \in P_v \Leftrightarrow (u, v) \in G$
Similarly the children C_v are given by $(v, u) \in G$

The following definitions are not used other than to define a binary tree. [this may change as later reconstruction of trees becomes more rigorous]

Definition 2. *Path*
A path is an ordered list of vertices with the property that adjacent vertices are also adjacent in the tree.

- *$[v]$ is a trivial path in G if $v \in V$*
- *$[u, v, \ldots]$ is a path in G if and only if $(u, v) \in G$ and $[v, \ldots]$ is a path in G*
- *if neither of the above cases are met, the list is not a path*

Definition 3. *Cycle*
A cycle is simply a path with at least 2 elements, and equal endpoints

Definition 4. *Binary Tree with n leaves*
A graph is binary tree with n leaves if and only if

- *there are no cycles*
- *every node has exactly one predecessor, except one node called the root, which has no predecessor*
- *every node has either 0 children (a leaf) or 2 children (a branch)*
- *there are exactly n leaves*

For convenience we will also assume that the tree is labelled with leaves 1 through to $n-1$, hence leaves are labelled n through to $2n-1$.

Hence define B to be the set of branch labels.

$$\{b|b \in \mathbb{Z}, 1 \leq b < n\}$$

Similarly L to be the leaf labels.

$$\{l|l \in \mathbb{Z}, n \leq l < 2n-1\}$$

2.1 Most Recent Common Ancestor

Definition 5. *Ancestry*

If a node r is the root of a tree, then its ancestry is simply the singleton ordered list (r).

If a node r is not the root of a tree, then its ancestry is acquired by taking the ancestry of its predecessor, and prepending r itself.

In other words the ancestry of a node is the list of nodes above that node, starting with itself, and ending with the root of the tree.

Definition 6. *Common Ancestry*

The common ancestry of two nodes is the largest ordered list of nodes that is a suffix of both nodes' ancestries.

Definition 7. *Most Recent Common Ancestor*

The MRCA of two nodes (in particular two leaves) is the first node in the common ancestry of the two nodes.

2.2 Relabelling

Trees can be relabelled by injective functions on their node labels.

Definition 8. *Relabel*

We can use a permutation $\sigma \in S_{2n-1}$ to relabel a tree by taking the group action of S_{2n-1} on a graph.

$$\text{relabel}(\sigma, G) := \{(\sigma(i), \sigma(j))|(i,j) \in G\}$$

Just as group actions give us a concept of relabelling, the orbits of these actions will give a set of trees that are relabellings of eachother.

This can give us an exhaustive set of trees with the same structure, which we can take to represent the structure itself:

Definition 9. *Tree Structure*

The structure of a tree with n leaves, is simply the orbit of that tree under the relabel *action of the group $S_B S_L$.*

Similarly the group S_B could be thought of as that tree without branch labels, and S_L as that tree without leaf labels.

2.3 Representation

Our goal is to find measurements of trees that can identify if trees are relabellings of eachother, and potentially how close they are to being relabellings of eachother.

Such a measurement would, at least, map trees with the same structure to the same values, and map trees with different structures to different values.

Hence we define the following concepts, noting some caveats:

Definition 10. *Invariance Under Relabelling*

A function or operation on binary trees with n leaves, is invariant under relabelling, if and only if trees with the same structure map to the same result.

Similarly a function or operation is invariant under branch/leaf relabellings if trees that are equal with their branches/leaves removed, map to the same result.

These properties could also be invoked "up to isomorphism" or up to some other class of operations.

This simply means that the results of the function don't need to be exactly the same, they just need to be mapped to eachother by some operation in the specified class.

Definition 11. *Distinction of Tree Shapes*

A function or operation on binary trees with n leaves, distinguishes tree shapes, if and only if trees with different structures map to different results.

As before we also define distinguishing trees ignoring leaves/branches, and distinguishing all of these up to isomorphism/permutation/etc.

Note that distinguishing tree shapes up to a class of operations is *stronger* than distinguishing tree shapes in general, whereas invariance under tree shapes up to a class of operations is *weaker* than distinguishing tree shapes.

What we will find in this paper is that our representations either have both properties up to row/column permutation of matrices, or fail to distinguish tree shapes at all. (and become near-trivial as defined below)

In addition to the above properties, if a function doesn't distinguish tree shapes, then there is a stronger failure condition shown in Matsen, which we will call "near triviality":

Definition 12. *Near-Trivial*

A function or operation on binary trees with n leaves, is near trivial, if and only if the proportion of 'failures to distinguish' approaches 1 as n gets larger.

That is, if the proportion of pairs of trees that have different tree shapes but map to the same result, out of all pairs with different shapes, approaches 1.

Definition 13. *Exchange*

Given a function that is invariant under relabelling, regardless of tree size, but does not always distinguish tree shapes, a pair of tree structures A and B 'exchange' if they not only map to the same result under the function, but any tree that contains A as a subtree will map to the same result as that tree with A replaced with B.

[Note that subtrees break our assumption that branch labels are 1..n-1, and leaf labels are n..2n-1. It may be useful to explicitly define subtrees in terms of *tree structure* along with this concept of "replacing" a subtree]

In Matsen it was shown that if a pair of trees exchange then the function for which they exchange will be near-trivial.

2.4 Hierarchy

There is another way of representing a tree without its branch labels, which is by representing it as a hierarchy based on sets of leaves under a node.

[briefly define hierarchies and maybe argue for their equivalence to trees without branch labels]

Chapter 3

Constructing Measurements of the Algebra

3.1 Background

The paper "Ubiquity of synonymity" dealth with 3 different matrix representations of trees: the adjacency matrix, laplacian matrix, and distance matrix. These representations are interesting, since relabellings of the tree result in similar matrices, so their spectra will not change. This means the spectra are themselves a representation of the tree, but the paper shows that these representations are not fair, and in fact 'approach' triviality as the number of leaves increases.

A motivation indicated by the paper is to derive a distance function between the structure of two trees. Such a function would still have potential use if it was derived from an unfair representation, but the result that representations become near trivial means most trees would also have zero distance As such while it would be nice to have a fair representation, it is crucial that it is at least substantially non-trivial.

The Tas Phylo group has shown in another paper that trees can be represented as an algebra of matrices, and that this algebra is commutative.

The algebra is as follows:

Definition 14. *Algebra of a Tree*

Given a binary tree with n leaves, that tree's algebra is an $n-1$-dimensional commutative algebra spanned by $n-1$ basis matrices corresponding to the $n-1$ branch nodes of the tree.

The basis matrix r in a tree with n leaves has the following definition:

$$L_{ij} = \begin{cases} 1 & i \neq j \text{ and } \mathrm{MRCA}(i,j) = r \\ 0 & i \neq j \text{ and } \mathrm{MRCA}(i,j) \neq r \\ -\sum_{k \neq j} L_{ik} & \text{otherwise} \end{cases}$$

Similar to the previous paper, these matrices relate rows/columns to leaf labels, and hence are similar to matrices associated with relabellings of the tree.

That is, this algebra representation is invariant under relabelling up to simultaneous row/column permutation.

It also distinguishes tree shapes up to row/column permutation.

As a set of matrices, the algebra no longer has branch information, however the basis as an ordered set will still have branch information, so we can see that the basis is not invariant under tree shapes, but will distinguish tree shapes, and that the algebra will in fact be invariant under branch relabelling.

[also the basis is invariant under branch relabelling up to basis permutations, invariant under leaf relabelling up to row/column permutation, these properties seem too expressive to thoroughly explore]

Further, since the algebra is commutative, we have the property that if it diagonalizable at all, then it is simultaneously diagonalizable. Together these properties suggest that the spectra of matrices in this algebra might be useful.

3.2 Canonical Forms of Spectra

If we assume that the matrices in the algebra are diagonalizable for now, (we can show this later) then we can start to reason about the spectra of these matrices. With this assumption, it will also follow that all of the matrices in the algebra are simultaneously diagonalizable, and hence that a set of eigenvectors can be found common to the whole algebra. We shall therefore refer to these as the Algebra's eigenvectors.

With this in place, we consider the spectra of the matrices in the algebra. When representing each tree as a single matrix, comparing the spectra of these matrices was a simple matter of comparing the multiplicity of each eigenvalue. This could be thought of as either comparing the spectrum as a multiset, or as a sorted sequence of values. Then since the order of the eigenvectors doesn't matter, we can freely relabel trees without changing their representation.

We, on the other hand, are considering a multidimensional space of matrices, and it might be nice to have this same property of invariance under relabelling. As such we might consider the set of sorted spectra of the whole algebra.

This representation seems to be useful, but once we consider the simultaneous diagonalization of the algebra, we see a missed opportunity to generate a homomorphism from the algebra to the vector space $\mathbb{C}^n$. That is, by taking the algebra's eigenvectors under some ordering, and generating the spectra of matrices corresponding to these eigenvectors, we will get the property that the spectrum of a linear combination of matrices is the same linear combination of the individual spectra. This means the "spectrum" map is a homomorphism, and so its image will be a vector space.

Definition 15. *Spectral Space*

The spectral space of a binary tree will here mean the set of spectra of matrices from the tree's algebra.

This set is a vector space since the algebra is commutative.

This will not apply to the sorted image, e.g. $< 1, 1 > - < 0, 1 > = < 1, 0 >$

This spectral space will be invariant under relabelling, up to permutation of the axes of the space.

Additionally, since it was constructed from the algebra and not from an ordered basis, it will be invariant under branch relabelling.

This seems like a promising representation; if we could find a convenient measurement of the space to get rid of the axis labels, that would be very useful.

One such measurement would be the dimension, but when we consider the dimension we find something else about these spaces...

3.3 Subspace Triviality

It will turn out that its dimension is always the same as the algebra, which seems odd when the algebra seems to contain matrices that are just relabellings of eachother. As an example every cherry in a tree will correspond to a matrix equivalent to the following:

$$\begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}$$

The exact matrices are simply the above with added rows and columns of zero. The spectrum of any matrix in this form should be the same as any other, and in the sorted-list representation they will be, but our vector space will distinguish these, in the same way that $< 0, 1 >$ and $< 1, 0 >$ are linearly independent.

This means that symmetries in the tree might have created interesting redundancies in the set of sorted spectra, but our vector space will not reflect these symmetries in the same way.

In particular if we say that the number of leaves on the original tree is n, then our algebra will be generated from $n - 1$ internal nodes, and will be a set of $n \times n$ matrices.

This means that the vector space will be an $n - 1$ dimensional subspace of $\mathbb{C}^n$.

Unfortunately since the matrices all have zero row-sum, that corresponds to an eigenvector of all 1s, and an eigenvalue of zero, which means the subspace will simply be $\mathbb{C}^{n-1}$ extended by a zero, regardless of the tree in question.

Not only is the vector space not a fair representation of the binary tree, it is a trivial representation.

Theorem 1. *The spectral space is trivial.*

Given a binary tree with n leaves, its spectral space will always be $\mathbb{C}^{n-1} \prod 0$.

The proof of this will come later once we have the eigenvalues in closed form.

3.4 Parameter-Free Representations

While it is not useful to map the whole algebra to $\mathbb{C}^n$, if we just look at the basis it turns out to be more useful. As such we shall consider the basis that was used to define the algebra.

If the original tree has n leaves, then we have

- $n-1$ internal nodes
- $n-1$ basis matrices
- n common eigenvectors
- $(n-1) \times n$ eigenvalues

This could be summarized in a pair of matrices:

- An $n \times n$ matrix of eigenvectors, and
- an $(n-1) \times n$ matrix of eigenvalues.

This second matrix will always have a column of zeroes in it, corresponding to the eigenvector of all 1s, since the algebra has zero row-sum. As such we do not lose any information by removing this column, and getting a square $(n-1) \times (n-1)$ matrix. We could do the same for the eigenvector and get an $n \times (n-1)$ matrix, but obviously this has the opposite effect.

These matrices are not invariant under relabelling:

- relabelling leaves permutes the rows of the eigenvector matrix
- relabelling internal nodes permutes the rows of the eigenvalue matrix
- relabelling eigenvectors permutes the columns of both matrices.

It seems like either of these matrices on their own would be enough to reconstruct the algebra, and hence the tree, so on their own they are interesting representations of a tree. Further still, measurements that are invariant under the row/column permutations above have potential for distinguishing tree-shapes as was originally desired.

One clear possibility is to take the determinant of either of these matrices, and remove the effect of row/column permutations with an absolute value. As such we shall first derive the exact value of both of the matrices.

Chapter 4

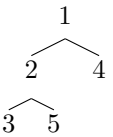
Simultaneous Diagonalization

When working towards a simultaneous diagonalization of the algebra, the first thing we need is to show that the matrices can be diagonalized at all. In showing this we expect to get the eigenvalues as well.

4.1 Simple Matrix Format

For simplicity we will assume that the tree leaves have been labelled "from left to right".

As an example of a tree that isn't labelled left to right, take the following tree:



The issue we have is that 2 has a descendant that belongs to the left of 4 and another to the right of 4, but 4 is not itself a descendant.

Definition 16. *Left to Right*

A binary tree with n leaves is labelled Left to Right if and only if the descendents of any vertex are a set of consecutive integers.

Other labellings will simply permute the coordinates of the eigenvectors we derive here.

Then if we consider a vertex in such a tree, and suppose that:

- its descendents are in the range $(l, l + x + y]$,
- its left child's descendents are in the range $(l, l + x]$,
- its right child's are in the range $(l + x, l + x + y]$

where the above ranges are taken to be $subset\mathbb{Z}$, then we can infer the following basis matrix corresponding to this vertex:

Definition 17. *The M form of a basis matrix*

$$M_{ij} = \begin{cases} -y & l < i = j \leq l + x \\ -x & l + x < i = j \leq l + x + y \\ 1 & l < i \leq l + x < j \leq l + x + y \\ 1 & l < j \leq l + x < i \leq l + x + y \\ 0 & otherwise \end{cases}$$

Lemma 1. *The* MRCA *of two leaves i and j is r if and only if they are descendants of different children of r.*

Proof:

If i and j descend from the same child of r, then that child will also be in the common ancestry of i and j, and hence r will not be the most recent one.

If i and j descend from different children of r, say v_L and v_R respectively, and we suppose that j is *also* a descendant of v_L, then the ancestry of j would show either a path from v_L to v_R or the other way around. A path ending in either v_L or v_R must contain r penultimately, since vertices in a tree only have one predecessor, which means we can construct a non-trivial path from r to itself. Since such a cycle is also impossible, it must *not* be the case that j descends from v_L, and similarly we would see that i does not descend from v_R. In other words, neither one is a common ancestor of i and j, and yet r *is* a common ancestor, so r is the most recent common ancestor.

Lemma 2. *The M form matrix on a node r is a basis matrix*

Proof:

Our basis matrix was defined by 3 cases, the first two of which:

$$L_{ij} = 1 \text{ if } i \neq j \text{ and } \text{MRCA}(i, j) = r \tag{4.1}$$

$$L_{ij} = 0 \text{ if } i \neq j \text{ and } \text{MRCA}(i, j) \neq r \tag{4.2}$$

correspond to 3 of the cases in our M form matrix:

$$M_{ij} = 1 \text{ if } l < i \leq l + x < j \leq l + x + y \tag{4.3}$$

$$M_{ij} = 1 \text{ if } l < j \leq l + x < i \leq l + x + y \tag{4.4}$$

$$M_{ij} = 0 \text{ otherwise, as long as } i \neq j \tag{4.5}$$

We see that the conditions for (4.3) in a left to right labelled tree directly correspond to i descending from the left child of r and j from the right child of r.

(4.4) instead shows these for j and i respectively.

(4.5) takes up the remaining cases, so we conclude that in M if $i \neq j$, then $M_{ij} = 1$ if and only if i and j descend from different children of r, and $M_{ij} = 0$ otherwise.

By the lemma above this means that MRCA$(i, j) = r$ when $M_{ij} = 1$, and $MRCA(i, j) \neq r$ when $M_{ij} = 0$, so in these cases $M_{ij} = L_{ij}$

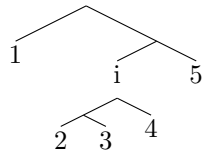
The remaining case is $i = j$, which in our basis matrix is:

$$L_{ii} = -\sum_{k \neq i} L_{ik} \tag{4.6}$$

By inspecting the cases of M_{ik} one can see that if $i \leq l$ or $l + x + y < i$ then $L_{ii} = 0 = M_{ii}$ similarly if $l < i \leq l + x$ then $L_{ii} = -sum_{l+x<k\leq l+x+y} 1 = -y = M_{ii}$ and so on.

So M is in fact the basis matrix corresponding to r.

So for example if we take the tree:



Then the node marked i will have $l = 1$, $x = 2$, $y = 1$ which gives

$$L_i = M = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 1 & 1 & -2 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

From this it becomes clear that all but $x + y$ of the eigenvalues will be zero, and zero-rowsum brings this down to $x + y - 1$ non-zero eigenvalues.

As a side note this means that the matrix of all of the eigenvalues will be at least half zeroes, which simplifies a lot of calculations on this matrix, especially once we find a way of permuting the eigenvalue matrix to be upper triangular.

4.2 Diagonalization

Once we are working with the M form, the eigenvalues become straight-forward to enumerate.

Theorem 2. *Eigenvalues of a basis matrix.*

In a binary tree with n leaves, the basis matrix L_r has 0, $-x$, $-y$, $-(x + y)$ as eigenvalues, with multiplicity $n - (x + y) + 1$, $y - 1$, $x - 1$, and 1 respectively, where x is the number of leaves descending from the left child of r, and y is the number of leaves descending from the right child.

In addition to this, the n corresponding eigenvectors are linearly independent.

If we take the root of the 3-tree for example, we get the following eigenvectors:

$$\begin{bmatrix} -1 & 0 & 1 \\ 0 & -1 & 1 \\ 1 & 1 & -2 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 0 & -2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 0 & -2 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -3 \end{bmatrix}$$

the first eigenvector is trivial to understand, having an eigenvalue of 0 it simply says that the row-sum of our matrix is 0, an intended feature of its construction.

The second and third are more interesting, and can be understood by how they act in the first two rows, (where i ¡= x) vs the last row (where x ¡ i ¡= x+y)

$$\begin{bmatrix} -1 & 0 & 1 \\ 0 & -1 & 1 \\ 1 & 1 & -2 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix} = -1 \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}$$

In the first two rows the eigenvalue directly appears in the matrix, and is the only term that doesn't become a zero in the series. In the last row the repeated 1s sum the coordinates of the matrix, which give zero.

This form of $< 1, -1, 0 \dots 0 >$ will generalize to any M_{ij} as above, as long as $x \geq 2$, with an eigenvalue of -y. Similarly if y ¿= 2 we could reverse the labels temporarily and use the same argument, we get an eigenvector of $< 0 \dots 0, -1, 1 >$ with an eigenvalue of -x.

Further still, we can get $x-1$ of the former and $y-1$ of the latter by moving the -1 coordinate to any other row $\leq$ x, and the above argument would still apply.

This gives $x-1$ repeated eigenvalues of $-y$, $y-1$ repeated eigenvalues of $-x$, which when combined with the 0 eigenvalue shown, and the $n-(x+y)$ trivial 0 eigenvalues coming from rows that are all zero, we get $n-1$ total eigenvalues, meaning there is 1 more before we have a general solution.

This must correspond to the third eigenvector we get in the simple 3-tree case:

$$\begin{bmatrix} -1 & 0 & 1 \\ 0 & -1 & 1 \\ 1 & 1 & -2 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ -2 \end{bmatrix} = -3 \begin{bmatrix} 1 \\ 1 \\ -2 \end{bmatrix}$$

If we guess that the general form is $< a, a \dots a, b, b \dots b >$ so that the a repeats y times, and the b repeats x times, then upon application of M as above, we get

$$[Mv]_i = \begin{cases} -y * a + y * b & \text{if } i <= x \\ x * a - x * b & \text{if } i > x \end{cases}$$

Then if we suppose this vector Mv equals λv then that gives two equations:

$$\lambda a = (yb - ay) \tag{4.7}$$

$$\lambda b = (ax - xb)$$

Next we eliminate λ and start to solve for b

$$(yb - ay)/a = (ax - xb)/b$$
$$b^2y - aby = a^2x - abx$$
$$b^2y + ab(x - y) - a^2x = 0$$

$$\begin{aligned} r &= \frac{-a(x-y) \pm \sqrt{a(x-y)^2 + 4a^2xy}}{2y} \\ &= a\frac{y - x \pm (x+y)}{2y} \\ &= a or a\frac{-x}{y} \end{aligned}$$

$b = a$ corresponds to the zero-rowsum eigenvector from above, so it is not new. $b = a\frac{-x}{y}$ is new however, so set $a = y$, $b = -x$.

then our eigenvalue λ can be derived from (1):

$$\lambda = \frac{y(b-a)}{a} \qquad\qquad = -(x+y)$$

This is our last eigenvalue, corresponding to the following eigenvector:
$\langle y, y \ldots y, -x, -x \ldots - x\rangle$
where y is repeated x times, and $-x$ is repeated y times.

This means that all of our basis vectors are fully diagonalizable, as we have $x + y$ nontrivial eigenvectors, plus $n - x + y$ empty rows, giving a total of n eigenvectors.

4.3 Simultaneous Diagonalization

Theorem 3. *Simultaneous Diagonalization*

The eigenvectors of the basis are exactly the eigenvectors for the $-(x + y)$ values above.

[could we use symmetry concepts to create a simple description of the eigenvectors?]

The matrices that we are looking at form a commutative algebra, which means there is some set of n linearly independent vectors that are eigenvectors for all n-1 of our basis matrices at once.

If we look at each matrix L_i, we can observe that the last eigenvalue in our list above, $\lambda = -(x + y)$ is unique, and hence the eigenvector that corresponds to it is uniquely determined by it. (modulo scaling)

Further in the matrix corresponding to the root of the tree, the 0 eigenvalue is also unique, so this node will actually give 2 uniquely determined eigenvectors.

So in total we have determined n eigenvectors!

Inspection of these shows that none of them are proportional to eachother, since they all contain zeroes or negatives in different places to eachother.

Then we know that:

- a set of linearly independent eigenvectors must exist,
- each of our eigenvalues is proportional to one of these vectors
- none of our eigenvalues are proportional to eachother

From this we can conclude that this must be a rescaling of some linearly independent set of simultaneous eigenvectors, and hence is also a linearly independent set of simultaneous eigenvectors.

If we label each eigenvector with the basis matrix that determined it, plus v_0 as the zero-rowsum eigenvector, then the the eigenvalues of these eigenvectors has a novel relationship with the structure of the binary tree:

- $L_i v_i = -(x_i + y_i)v_i$ noting v_i and L_i come from the same node
- $L_i v_j = -x_i v_j$ if j sits on the left subtree under i
- $L_i v_j = -y_i v_j$ if j sits on the right subtree under i
- $L_i v_j = 0$ if j sits outside of the subtree under i, or $j = 0$

These results follow from constructing v_j as a linear combination of the eigenvectors described previously for L_i

So we have the exact value of our n eigenvectors and eigenvalues.

4.4 Ordering the Nodes

Theorem 4. *Spectrum Matrix is Upper Triangular*

The spectrum matrix is similar to an upper triangular matrix by simultaneous row + column permutation.

In other words there is a relabelling of any tree that gives it an upper triangular spectrum.

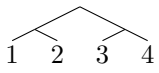
If we can order these nodes so that their children always come after them, then the matrix of eigenvalues will be upper triangular. (Once the zero-rowsum column is removed)

One consequence of this is that the determinant will simply be

$$\prod_{i=1}^{n} d_i$$

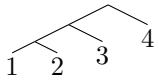
This is simply the pre-order traversal of the vertices, and so we have a closed form for the determinant.

As an example of what all of the eigenvectors and eigenvalues look like in matrix form, take two trees with 4 leaves:



$$\text{eigenvectors} = \begin{bmatrix} 1 & 2 & 1 & 0 \\ 1 & 2 & -1 & 0 \\ 1 & -2 & 0 & 1 \\ 1 & -2 & 0 & -1 \end{bmatrix}$$

$$\text{eigenvalues} = \begin{bmatrix} 0 & -4 & -2 & -2 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & -2 \end{bmatrix}$$



$$\text{eigenvectors} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & -1 \\ 1 & 1 & -2 & 0 \\ 1 & -3 & 0 & 0 \end{bmatrix}$$

$$\text{eigenvalues} = \begin{bmatrix} 0 & -4 & -1 & -1 \\ 0 & 0 & -3 & -1 \\ 0 & 0 & 0 & -2 \end{bmatrix}$$

Chapter 5

Deciding Fairness

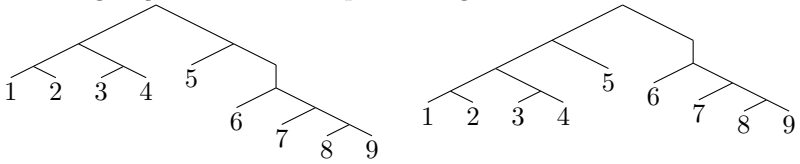
5.1 Ubiquity in Eigenvalue Matrix

As we will see, this eigenvalue matrix can be used to reconstruct the tree *with labels*, and so we would like to remove some more information to reach our goal of having a finite representation of tree shapes.

Theorem 5. *Near Triviality of Determinant*
The spectrum of the eigenvalue matrix of a tree algebra is near trivial under relabelling, up to reordering.

The spectrum of the eigenvalue immediately has the exchange property, as the spectrum of our upper triangular matrices is simply the diagonal, and this diagonal is simply the number of leaves in each subtree.

So then as long as there is a pair of trees with the same spectrum, we'll get the same strong negative result for spectra in general.



As a result any measurement which is a symmetric function of the diagonal, such as the determinant or the trace, will be at least as likely to coincide as the spectrum itself, and so these functions will also be near-trivial.

The two exceptions to this are, the trace of the eigenvalue matrix when not triangular, and the spectrum of the eigenvalue matrix with ordering information.

5.2 Spectrum as Measure of Balance

Interestingly, the trace of these upper triangular matrices will equal to the Sackin index of the tree, which is the sum of the path lengths from the root to each leaf of the tree.

The determinant seems quite unrelated to path lengths, but will still measure tree balance for similar reasons to the trace.

As an example the two trees of order four have a spectrum of $(4, 2, 2)$ and $(4, 3, 2)$ respectively. The larger nodes in unbalanced trees can be expected to make both the trace and the determinant larger.

5.3 Fairness of Matrices

Theorem 6. *Fairness of eigenvalue matrix.*

Construction of the eigenvalue matrix distinguishes and is invariant under tree shapes without leaf labels.

The eigenvalue matrix has 3 non-redundant sets of data in it:

- The most extreme number in a row indicates the number of leaves under a node
- The column of that most extreme number indicates the eigenvector labelling
- Nonzero cells indicate that that column's node exists in the subtree of that row's node

So the first thing one could do is decompose the matrix into those 3 functions.

Define W to be the matrix of eigenvalues, *without the zero-rowsum column*, and with negated (and hence positive) entries.

Define I to be the set of integers 1 thru $n-1$, i.e. $\mathbb{Z} \cap [1, n-1]$

$$Size(i) := \max\{W_{ij} | j \in I\}$$

$$Col : I \to I$$

$$\forall i \in I, W_{iCol(i)} = Size(i)$$

$$j \preccurlyeq i \Leftrightarrow W_{iCol(j)} > 0$$

Col is well defined since we know $Size(i)$ is the unique eigenvalue of L_i.

$\preccurlyeq$ is the original tree understood as a partial order, but with leaves removed.

Simply reconstructing the tree from this partial order, and supplementing nodes with leaves until it is a binary tree, should give the original tree.

On the other hand, one can use the $Size$ function, by taking the two largest descendants as the children of each node, again supplemented with leaves until each node has 2 children.

We know that this is in fact the original tree, as we've shown that the eigenvalues of each matrix correspond to the structure of the tree.

Theorem 7. *Fairness of eigenvector matrix.*

Construction of the eigenvector matrix distinguishes and is invariant under tree shapes without branch *labels.*

The construction here is simple, if we use the eigenvector column indeces as labels for the internal nodes, then the partial order is defined as:

$$j_1 \preccurlyeq j_2 \Leftrightarrow \forall i \in J, V_{ij_1} > 0 \Rightarrow V_{ij_2} > 0$$

Then as before each node neads to be supplemented with any leaves, taking their labels from the row indeces of V.

Chapter 6

loose notes

6.1 proof notes

theres a bunch of things that could help some of the proofs above become more clear.

First eigenvectors with non-zero eigenvalues have $v_j \neq 0 \Rightarrow$ leaf j is a descendant of node i

In particular for the most extreme/unique eigenvalue the converse is also true.

Recovering tree structure from the vectors is then simple. Recover the set of leaf descendants from above, then all if the descendants of one node are also descendants of another node, the former is a descendant of the latter.

Further we know that none of these can be proportional to eachother, or they would belong to the same node, and hence be the same anyway. The exception here is the zero-rowsum eigenvector, but we know none of the constructed eigenvectors are proportional to that because its eigenvalue is always 0.

Second the non-unique eigenspaces of the matrices correspond to tree structure as well. Leaves that dont descend from the node form an eigenspace on 0. leaves that descend from the left child, restricted to have a sum of zero, form An eigenspace on the size of the left subtree. Likewise for the right child.

Then each of the eigenvectors in our set will sit in one of these three spaces for all other nodes in the tree, which is very convenient.

This gives us the structural information we need to show that the eigenvalue matrix is a fair representation of the original tree.

The fact that a binary tree can be reconstructed from the tree of its internal nodes is probably a reference?

6.2 minimum determinant

each row of a balanced tree with 2^n leaves, is 2^i lots of 2^{n-i}... right?

6.3 Meeting 2

the determinants seem to measure how balanced the trees are, from maximally balanced: (expr?), to maximally unbalanced: n!

The original question still remains, how much could be removed from the matrix without breaking injectivity. Could you reconstruct the tree from an ordered spectrum?

6.4 eig matrices

why do the two matrices need leaf/branch info to recover the original? could you get rid of it? if you cant is it impossible to use these matrices to distinguish tree shapes?

well chosen matrix norms of either matrix might stifle the exchange property matrix differences might be useful once the matrices have been sorted? but then a tree will not have 0 distance from its own relabellings.

6.5 eigenvalue polynomials

the eigenvectors can be discovered through a polynomial process, by looking at the product of each pair of basis matrices, as a linear combination of other basis matrices

6.6 hierarchies

the eigenvector subset argument could be quite simple if described in terms of hierarchies?

6.7 Symmetries under relabelling

The eig spaces of the basis matrices all have a sum of zero, except the all-1 eigenvector

corresponds exactly to the symmetries of the tree under the action of leaf relabellings.

In particular each node is agnostic to relabellings of its left subtree, and of its right subtree, so each node restricts the eigenspaces only with local knowledge, until eventually the full structure of the tree has been represented.

6.8 Different kind of symmetric function

When trying to summarize the eigenvalue matrix, we specifically want an operation that is symmetric under row permutations, and under column permutations, but not on arbitrary value permutations.

$f(X) = f(K_1 X K_2)$

Then we can take further restrictions like "must be homogeneous" or whatever.

Under the wrong restrictions, we should get imanant polynomials out, but steering clear of that might give something useful?