Using spectra of MRCA algebras to distinguish binary tree shapes.

Jarvis Carroll
Supervised by Jeremy Sumner

2019
February

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"[1].

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. (i.e. the sum of each row is subtracted from the diagonal)

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 with labels, except for the distance matrix, which corresponds to trees without internal labels.

Relabelling vertices in the tree/graph will simultaneously permute corresponding rows and columns of all of these representations, a concept we will come to define as covariance.

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'. (i.e. they aren't relabellings of eachother)

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: some trees not only have the same spectrum, but substituting those subtrees for eachother in larger trees will not change their spectrum either. (this result applied to both distance matrices, and to *linear 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 operations are what we will later define as “near trivial”:

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

They did not prove or disprove triviality for the immanantal polynomial of the distance matrix; it is still an open question.

To say that an operation is “near trivial” says that for large enough trees the operation will map most pairs of distinct trees to the same value despite not being relabellings of eachother. (The proportion of pairs with this property approaches 1)

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

Otherwise these operations will not be useful representations of the tree.

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 an operation applied to a tree has an exchange property for some pair of trees, then the operation will be near trivial for large trees.

It should also be noted that a motivation indicated by the paper, and shared by this project, is to derive a distance function between the structure of two trees.

Such a function would have potential use if it was derived from an operation that couldn’t perfectly distinguish tree shapes, but should it have a proportion of failures that approaches 1, then of course most trees will have zero distance.

As such while it would be nice to have an operation which perfectly distinguishes tree shapes, it is crucial that it isn’t near-trivial, the way that the previous 5 operations are.

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 $(v, u) \in G$

A child or parent is of course an element of the set of children or parents.

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 a path with at least 2 elements, and equal endpoints

Definition 4. *Binary Tree*

A graph is a binary tree, or a tree with labels, if and only if

- *there are no cycles*
- *every vertex has exactly one predecessor, except one vertex called the root, which has no predecessor*
- *every vertex has either 0 children (a leaf) or 2 children (an interior vertex)*

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

Hence define B to be the set of interior vertex ('branch') labels.

$$B := [n-1] := \{b | b \in \mathbb{Z}, 1 \leq b \leq n-1\}$$

Similarly L to be the leaf labels.

$$L := [2n-1] \setminus [n-1]$$

This notation:

$$[n] = \{r | r \in \mathbb{Z}, 1 \leq r \leq n\}$$

will be used throughout this report for dealing with sets of vertex labels and matrix indeces.

2.1 Most Recent Common Ancestor

Definition 5. *Ancestor/Descendant*
A vertex a is an ancestor of a vertex b, and similarly b is a descendant of a if there is a path from a to b.

Definition 6. *Ancestry*
The ancestry of a vertex is the unique path from the root to that vertex.
The ancestry is unique in binary trees, since they are defined to have only one predecessor on each vertex.

The ancestry gets its name from the fact that it is an ordered list of ancestors for a given vertex.

Definition 7. *Common Ancestry*
The common ancestry of two verteces is the largest ordered list of verteces that is a prefix of both verteces' ancestries.

Definition 8. *Most Recent Common Ancestor*
The MRCA of two verteces (in particular two leaves) is the last element of the common ancestry of the two verteces.

2.2 Relabelling

Trees can be relabelled by injective operations on their vertex labels.

Definition 9. *Relabel*
We can use a permutation $\sigma \in \mathcal{S}_[2n-1]$ to relabel a tree by taking the group action of $\mathcal{S}_[2n-1]$ 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 labelled 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 10. *Trees Without Labels*
A tree without labels is an orbit under the relabel *action of the group $\mathcal{S}_B\mathcal{S}_L$.*
Similarly a tree without leaf labels or interior labels is that of the groups $\mathcal{S}_L$ and $\mathcal{S}_B$ respectively.
In addition to this, given a tree with labels, that *tree without labels is the orbit of that tree. (Without labels, without leaf labels, without interior labels)*

2.3 Representation

Our goal is to find measurements of (operations on) trees with labels, 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 which are the same without labels to the same values, and map trees which aren't, to different values.

Hence we define the following concepts:

Definition 11. *Covariance*
An operation on binary trees is covariant, if it commutes with some group actions.
I.e. if we map a set of trees T to some set of objects X via the mapping f, then for any relabelling $\sigma \in \mathcal{S}_L\mathcal{S}_B$, it is true that $f \circ \sigma = \sigma \circ f$. (where in the LHS σ is the relabel action of the permutation on T, and in the RHS is some other action on X)

If an operation is covariant then it not only forms a well defined mapping from trees to objects in the codomain, but also a well defined mapping from orbits of trees to orbits of objects in the codomain.

I.e. we get operations that are well defined on trees without labels.

Definition 12. *Invariance*

An operation on binary trees is invariant, if it is covariant, but with the additional property that orbits of its image are all singleton sets.

Invariance of course shows that the maps from orbit to orbit, are in a sense maps from orbit to individual objects.

Definition 13. *Orbit Map*

Given a covariant operation, and a subgroup of permutations inside $\mathcal{S}_B\mathcal{S}_L$, the orbit map of that operation is the map from the set of orbits in the domain under the subgroup, to the set of orbits in the codomain.

It is defined to simply apply the covariant operation to each element of the orbit to which it is applied.

Since the operation is covariant, it can be shown that this new set will itself be an orbit in the codomain.

These orbit maps are sometimes injective, though we find in this paper that none of the operations we investigate are both invariant and give injective orbit-maps.

Not only that, the invariant maps we investigate happen to have a stronger failure condition as described in Matsen, which we will call "near triviality":

It is worth noting that by taking the orbit of a tree, then applying an orbit map, immediately gives an invariant operation which will have the same injectivity properties that the original operation had.

This means any injective covariant operation can be used to construct an operation with the properties we want, invariant with injective orbit maps, but we do not consider them, as the sets of objects are not capable of being used to derive a distance formula, and are not computationally scalable due to the combinatorics of large binary trees.

As an extreme example we could simply take the orbit of a tree to be an invariant, injective operation on the tree, but each orbit would be a set with $n!(n-1)!$ binary trees in it.

Definition 14. *Near-Trivial*

An operation on binary trees is near trivial, if and only if the proportion of pairs of trees that are different even without labels, but that give the same result, out of all pairs that are different without labels, approaches 1, as the tree size n gets larger.

Definition 15. *Exchange*

Given an invariant, non-injective operation on binary trees, a pair of trees without labels A and B exchange if they not only map to the same result under the operation, but any tree that contains A as a subtree will map to the same result as that tree with A replaced with B.

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

2.4 Additional Concepts

We use permutation groups to understand the operations on binary trees which we will derive.

We also use other graphs/relations to reconstruct trees based on our knowledge of which verteces are descendants of which.

Definition 16. *Symmetric Group*

The symmetric group $\mathcal{S}_A$ is the set of maps which are bijections on the finite set A.

There are a few group-theoretic concepts such as characters of a permutation, group actions, orbits, and row/column permutations of a matrix, which are not defined here but are crucial to many constructs described in this paper.

Additionally some concepts are important for explaining some of the background research of this project as well as for finishing the final theorems sketched in this report:

- Immanantal polynomials of a matrix
- Subtrees, subtree substitution
- Trees as a hierarchy on their leaves
- Trees as a partial order on their verteces
- Acquiring the directed graph of a tree from the above via transitive reduction
- Adding leaves to a tree which isn't binary, to form a tree which is binary.

Chapter 3

Tree Algebras and Consequent Operations

3.1 Tree Algebras

The Tasmania Phylo group has shown in current work that trees can be represented as an algebra of matrices, and that this algebra is commutative.[2]

The algebra is as follows:

Definition 17. *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 verteces of the tree.*

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

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

Note that we don't use $\mathrm{MRCA}(i,j)$ but $\mathrm{MRCA}(n-1+i, n-1+j)$ since i and j are matrix indeces not leaf labels.

We have the result that this algebra is commutative, which means that once we know its elements can be diagonalized, it will be simultaneously diagonalizable.

This suggests that similar to Matsen, we may be able to use the spectrum of elements of the algebra in order to distinguish binary trees.

But first, in order for such spectra to have distinguishing power, it would need to be true that the algebra still has distinguishing power.

3.2 Algebra Covariance

Theorem 1. *Basis Covariance*

The construction of the basis as an ordered set of matrices, is a covariant operation

Proof.

The basis matrices are defined in terms of the $MRCA$ of two leaves, i and j, against the index of one interior vertex r.

The group action on the trees permutes their labels, which gives matrices based on $\sigma(i)$ and $\sigma(j)$, against the index of one interior vertex $\sigma(r)$.

If we simply define the group action on the sequence of basis matrices to permute the matrices themselves, as well as their rows, and their columns, in the same way, then they will have the exact same definition:

$$L\prime^{(}_{ij}r) = \begin{cases} 1 & i \neq j \text{ and } \mathrm{MRCA}(\sigma(n-1+i), \sigma(n-1+j)) = \sigma(r) \\ 0 & i \neq j \text{ and } \mathrm{MRCA}(\sigma(n-1+i), \sigma(n-1+j)) \neq \sigma(r) \\ -\sum_{k \neq i} L\prime_{ik} & i = j \end{cases}$$

This gives us covariance. □

The construction of the algebra itself, as the span of these matrices, will also be covariant, which can be shown with the same reasoning.

Theorem 2. *Basis Injectivity*

If a binary tree with labels is used to generate its sequence of basis matrices, then the binary tree can be recovered exactly from the sequence.

Proof.

Given the basis matrices, we immediately know that a leaf i is a descendant of an interior vertex r, if the ith row/column of the rth matrix is not empty.

This is because the tree is specifically a binary tree:

If i is a descendant of r, then whichever child i descends from, there must be another child and hence another leaf descending from that child

This means there is a pair of leaves whose MRCA is r (This kind of reasoning will be made explicit in a later lemma)

This in turn means the ith row/column of the rth matrix is not empty.

Once we know whether any leaf i descends from any interior vertex r, the rest of the tree is determined.

The reason for this is that we can construct a hierarchy which is equivalent to the tree, and then apply transitive reduction to recover the tree itself.

□

This means that if two sequences of basis matrices are different, then the trees they were constructed from must also be different, though they may simply be relabellings.

It can be shown that any orbit maps that come from the action of subgroups of $\mathcal{S}_L\mathcal{S}_B$ will also be injective.

The algebra itself will not have this full injectivity, but its orbit maps under specifically $\mathcal{S}_B$ will be injective.

The proof will not be written here but the approach would be to find the unique basis of the algebra whose non-diagonal entries are each either 0 or 1, then pick an arbitrary order for this basis and repeat the above process.

The reason that this injectivity only applies to $\mathcal{S}_B$'s orbit map is that our choice of basis labels is arbitrary, and so we can only reconstruct the corresponding tree without interior labels. (Trees without interior labels of course being orbits of trees with labels, under the group action of $\mathcal{S}_B$)

3.3 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 also follows 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.

Note that we will call these unordered representations of the spectrum, even though the sorted list is ordered in a sense. Additionally, spectra are ordered if not specified otherwise.

Then since the order of the eigenvectors doesn't matter for an unordered spectrum, we can freely relabel trees without changing their corresponding spectrum. (invariance)

We, on the other hand, are considering a multidimensional space of matrices, and it would be nice to attempt to construct an invariant operation with it. 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 18. *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 covariant, with a group action that permutes the axes of the space based on the permutation's effect on leaf labels, and ignores its effect on interior labels.

Note that the ordering of the eigenvectors is free anyway, so covariance doesn't really apply since you can already permute the axes without relabelling the tree. In that way our spectral space isn't well defined.

This seems like a promising representation; if we could find a convenient measurement of the space to get rid of the label information implicit in the ordering of the axes, 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...

3.4 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 unordered representations they will be, but our vector space will distinguish these, in the same way that $(0,1)$ and $(1,0)$ are linearly independent, but the same when sorted.

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 vertices, 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 (operation with injective orbit maps) of the binary tree, it is a trivial representation.

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

3.5 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 verteces
- $n-1$ 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 of creating an oblong matrix.

These matrices are not invariant under relabelling:

- relabelling leaves permutes the rows of the eigenvector matrix
- relabelling internal verteces 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 tree, so on their own they are interesting representations of a tree.

Further still, by continuing the process of applying covariant operations, we could find a method of distinguishing tree-shapes as originally desired, i.e. an invariant operation with injective orbit maps.

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

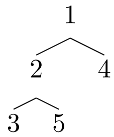
Simultaneous Diagonalization

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 19. *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 left child’s descendents are in the range $[l+x] \setminus [l]$,
- its right child’s are in the range $[l+x+y] \setminus [l+x]$

then we can infer the following basis matrix corresponding to this vertex:

Definition 20. *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 vertex r is a basis matrix*

Proof:

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

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

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

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 $\mathrm{MRCA}(i,j) = r$ when $M_{ij} = 1$, and $MRCA(i,j) \neq r$ when $M_{ij} = 0$, so in these cases $M_{ij} = L_{ij}$

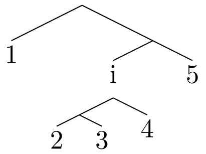
The remaining case is $i = j$, which in our basis matrix is:

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

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 vertex 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.

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 rows corresponding to leaves in the left subtree, vs those of the right subtree.

The second eigenvector can be generalized to multiple vectors when looking at vertices with more descendants:

Lemma 3. *Child Eigenvalues*

The basis matrix corresponding to each vertex, has an eigenvalue equal to the negative of the order of one child subtree, with an eigenspace whose dimension is at least one less than the order of the other child subtree. (And vice versa)

Proof.

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

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 $< 1, -1, 0 \ldots 0 >$ 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 $< 0 \ldots 0, -1, 1 >$ 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 ≤ x, and the above argument would still apply.

Then the spans of these vectors gives eigenspaces of the size we require. □

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:

Lemma 4. *Final Eigenvalue*

The basis matrix of each internal vertex has an eigenvalue equal to the negative of the number of leaves descendant from that vertex.

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

If we guess that the general form is $< a, a \ldots a, b, b \ldots b >$ 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

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

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

$b = a$ 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):

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

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. □

Theorem 3. *The Basis Matrices are Diagonalizable*

The previous lemmas mean 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

Definition 21. *Suo Eigenvector*

The suo vector of an internal vertex is the eigenvector corresponding to the $-(x + y)$ eigenvalue written above.

That is it is the eigenvector associated with the simple eigenvalue that every basis matrix has, equal to the negative of the number of leaves descending from that vertex.

Lemma 5. *Simultaneous Suo Vectors*

The suo vectors are eigenvectors of the whole algebra.

Proof:

Since the algebra is commutative, and any matrix in the algebra is diagonalizable, we know that the algebra is simultaneously diagonalizable.

This simultaneously diagonalizing set of eigenvectors needs to include eigenvectors corresponding to the simple eigenvalues of the suo vectors.

This implies that the suo vectors are all proportional to one of these eigenvectors, which in turn implies that the suo vectors are themselves eigenvectors of the whole algebra. □

These vectors have a few interesting properties, such as coordinates adding up to zero, and subtrees' leaves having equal coordinates.

As such we define another kind of vector, related to suo vectors:

Definition 22. *Leaf Vector*

The leaf vector of a vertex is the vector whose ith coordinate is 1 if the ith leaf is a descendant of the vertex, and 0 otherwise.

Lemma 6. *Eigenvector Orthogonality*

The suo vectors and the zero-rowsum vector are all orthogonal.

Proof:

Since suo vectors are a linear combination of two leaf vectors, and the zero-rowsum vector is the leaf vector of the root of the tree, we can consider a few cases:

First, if one vertex i is a descendant of another j, then it will also either equal or descend from one of the other vertex's children, c, say. In this case the dot product of the ith and jth suo vector will be the dot product of the cth leaf vector and the jth suo vector.

This will be proportional to the sum of the jth vector's coordinates, which is zero, so they are orthogonal.

If the two verteces are not descendants of eachother, then they won't share any non-zero coordinates, so their product will be zero.

Finally if one vector is the zero-rowsum vector, then again the product will simply be the sum of the other vector's coordinates, which is zero.

Therefore all the eigenvectors we have constructed are orthogonal. □

Theorem 4. *Simultaneous Diagonalization*

The eigenvectors of the algebra are exactly the $n-1$ suo vectors, and the zero-rowsum vector.

By the lemmas above, the suo vectors are eigenvectors of the whole algebra, and the zero-rowsum vector is also common to the whole algebra.

Additionally they are orthogonal to eachother, and hence clearly not linearly dependent.

As such we have n linearly independent eigenvectors and can simultaneously diagonalize the algebra. □

Lemma 7. *Structural Eigenvalues*

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 vertex*
- $L_i v_j = -x_i v_j$ *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 5. *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 verteces so that their children always come after them, then the matrix of eigenvalues will be upper triangular. (Once the zero-rowsum column is removed)

This is simply the pre-order traversal of the vertices, a traversal which by definition satisfies this requirement.

So if the interior vertices are labelled 1 through to n by one of the preorder traversals of the tree, and the eigenvectors are given the same labels as the vertex they correspond to, then the matrix of eigenvalues we construct will be upper triangular. □

One consequence of this is that the determinant will simply be

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

where d_i refers to the ith diagonal entry of the matrix of eigenvalues.

Another important consequence is related to our previously defined spectral space:

Theorem 6. *The spectral space is trivial.*

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

Proof.

The spectral space is simply the span of the rows of the eigenvalue matrix, which can always be permuted to be a column of zeros followed by an upper triangular matrix.

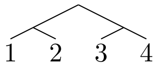
Further the diagonal of this triangular matrix has no zeros, so there is no linear dependance between these rows.

Clearly the span of these rows will be the set of vectors with zero in their first coordinate, and all other $n-1$ coordinates free.

As a result the spectral space of any tree will be some axis relabelling of such a vector space, which means an axis relabelling of $\mathbb{C}^{n-1} \times \{0\}$

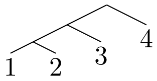
4.5 Eigenvalue Examples

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

Injectivity of Orbit Maps

5.1 Ubiquity in Eigenvalue Matrix

As we will see, this eigenvalue matrix can be used to reconstruct the tree without interior labels, but *with leaf labels*, and so we would like to remove some more information to reach our goal of having a convenient representation of trees without labels.

The first guess would be the determinant, but now that we know the eigenvalue matrix can be made upper triangular, this means we are really considering an operation on the unordered spectrum of the eigenvalue matrix.

Lemma 8. *Exchange Property of Spectrum*

If two binary trees have eigenvalue matrices with the same unordered spectrum, then they also have the exchange property.

Proof.

Suppose the two trees are called T_1 and T_2 respectively, and we have a larger tree that contains T_1 as a subtree, called T_1', along with this larger tree after substitution of T_2 into T_1, T_2'.

This larger tree must either be T_1 itself, or a root vertex with two child subtrees, one of which is *smaller, and itself contains T_1 as a subtree.*

In the former case, the lemma is trivial, as $T_1' = T_1$, and clearly $T_2' = T_2$, and these already have the same spectrum by definition.

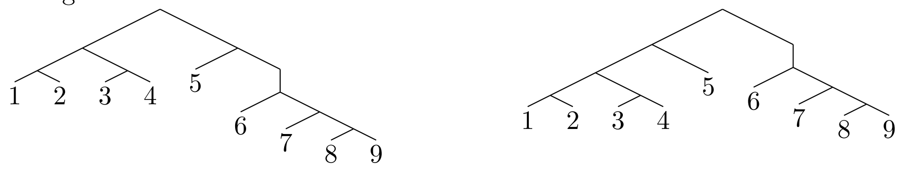
In the latter case, we can see that the result also follows, as the spectrum is just the multiset containing the order of each subtree of the tree, which will be the spectrum of each subtree combined with the order of the whole tree. One of the subtrees is already the same in both T_1' and T_2', and the order of the whole tree is the same, so by assuming that the other subtree has the same spectrum in each of T_1' and T_2' as an inductive hypothesis, then clearly the spectrum of the whole trees will be the same as well. □

Theorem 7. *Near Triviality of Determinant*

The spectrum of the eigenvalue matrix of a tree algebra is near trivial.

Since trees with the same spectrum immediately have the exchange property, and existance of trees with the exchange property immediately shows near-triviality, all we need is to find a pair of trees with the same spectrum.

A simple search gives these trees:



So we have near-triviality. □

The above search was done in haskell with code that can be found at the following locations:
`https://github.com/spiveeworks/treeshapes/blob/master/det_of_values.hs/`
`https://github.com/spiveeworks/treeshapes/blob/master/Tree.hs/`

One consequence of this result is that any operation which is a symmetric function of the diagonal, such as the determinant or the trace, will also be near-trivial, since the spectra will already be the same most of the time.

Two notable cases outside of this are, the trace of the eigenvalue matrix when not triangular, and the *ordered* spectrum of the eigenvalue matrix.

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.[3]

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 verteces in unbalanced trees can be expected to make both the trace and the determinant larger.

5.3 Fairness of Matrices

Theorem [Sketch] 1. *Fairness of eigenvalue matrix.*

The group $\mathcal{S}_L$ and the operation of constructing eigenvalue matrices, give injective orbit maps.

Proof.

We will show this injectivity by reconstructing a partial order which will give a tree once transitively reduced.

This tree will have arbitrary leaf labels which is why we can only show the orbit maps are injective.

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

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

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

$$Size(i) := \max\{W_{ij} | j \in [n-1]\}$$

$$Col : [n-1] \to [n-1]$$

$$\forall i \in [n-1], W_{iCol(i)} = Size(i)$$

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

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 verteces with leaves until it is a binary tree, should give the original tree.

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

We know that this would in fact be the original tree, as we've shown that the eigenvalues of each basis matrix correspond to the descendance relation on the tree exactly as we have reconstructed.

This means that the orbit map must be injective. □

Theorem [Sketch] 2. *Fairness of eigenvector matrix.*

The group $\mathcal{S}_B$ and the operation of constructing eigenvector matrices, give injective orbit maps.

The construction here is simple, if we use the eigenvector column indeces as labels for the internal verteces, 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 vertex needs to be supplemented with any leaves, taking their labels from the row indeces of V that each new leaf must correspond to.

This should also give us the tree we want, showing that the orbit maps must be injective. □

Without explicit definitions of transitive reduction or leaf supplementation, and without explicit proof this tree sits inside the correct orbit, or is even a tree at all, this proof is somewhat incomplete.

This injectivity result has taken all of the previous work to *discover*, but should this result be published it would require this completely separate set of constructions to prove, which is beyond the scope of what this project has achieved.

Chapter 6

Further Notes

6.1 Sets of Eig Matrices

Our eigenvalue and eigenvector matrices are both constructed with an undefined ordering on the columns, which means there is still a set of $n!$ equivalent representations, so our original goal of removing complexity from the orbit representation of a tree shape to get something that might be suitable for a distance function, is in a sense no closer to being attained.

That said one of them may still achieve this result in the future, or have another application.

It seems like the sorted diagonal of an upper triangular eigenvalue matrix might still be able to recover the tree without leaf labels, which could in fact reduce the size of the sets. Surely this is a known representation? Subtree sizes sorted by preorder traversal

6.2 Invariance

Removing the permutation information from either of our eig matrices without creating a trivial representation is still an interesting question.

Perhaps some kind of matrix norm won't have the exchange property?

What could be done with the eigenvector matrix? Investigating its spectrum found that the eigenvectors are orthogonal, so the matrix is orthogonal once its columns are normalized.

What would happen if you split the eigenvector matrix into another orthogonal matrix transforming the basis of a sequence of 2d rotations?

matrix differences might be useful once the matrices have been sorted somehow? but then a tree will not have 0 distance from its own relabellings.

6.3 Redundancy of Root

Information at the root of a binary tree is redundant, since it is always just the parent of the two trees described in the rest of your tree representation. This applies to every representation we have discovered here:

- a second column can be removed from the eigenvector matrix
- a second column and a row can be removed from the eigenvalue matrix, all without removing any information.
- the basis we construct of a tree's algebra always sums to a matrix of all 1s/$-(n-1)$s, so one basis matrix is redundant

If one column of the eigenvectors can be removed because the original matrices always coordinate-sum to zero, then it seems like these further columns can be removed for a reason somehow related to the eigenvectors always coordinate-summing to zero.

(A neat parallel to the V_+, V_- distinction which the later section "Symmetries under relabelling" explores.)

It is not clear however that this would help our original goal. If anything it is the redundancies of our representation that seem to open up possibilities for matrix theoretic measurements of a tree.

6.4 Eigenvector 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, this technique does not save any time in this case but is certainly more general to some extent.

6.5 Geometry of Eigenvectors and Eigenvalues

We have constructed a lot of vectors in this process, do they have a geometric interpretation?

Relabelling trees may transform volumes in some way.

6.6 General bases of the algebra

Our constructs use a specific basis of the algebra, the unique basis with 0 or 1 non-diagonal entries.

Is it possible to create constructs with more general choice of basis?

The determinant of the eigenvalue matrix is preserved by the special linear group acting on the original basis, for example.

6.7 Symmetries under relabelling

The eigenspaces of the basis matrices all have a coordinate sum of zero, except the all-1 eigenvector.

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

In particular each vertex is agnostic to relabellings of its left subtree, and of its right subtree, so each vertex 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" for example.

Under what would be perhaps the wrong restrictions, we should get imanant polynomials out, but steering clear of that might give something useful.

6.9 References

1. Matsen FA and Evans SN: Ubiquity of synonymity: almost all large binary trees are not uniquely identified by their spectra or their immanantal polynomials. *Algorithms for Molecular Biology* 2012, 7:14

2. Work not yet published, by Tasmania Phylogeny group

3. Steel M: Phylogeny, Discrete and Random Processes in Evolution. p53. *University of Canterbury Christchurch, New Zealand* 2016