

Interpretable machine learning for Kronecker coefficients

GIORGI BUTBAIA, KYU-HWAN LEE, AND FABIAN RUEHLE

We analyze the saliency of neural networks and employ interpretable machine learning models to predict whether the Kronecker coefficients of the symmetric group are zero or not. Our models use triples of partitions as input features, as well as b -loadings derived from the principal component of an embedding that captures the differences between partitions. Across all approaches, we achieve an accuracy of approximately 83% and derive explicit formulas for a decision function in terms of b -loadings. Additionally, we develop transformer-based models for prediction, achieving the highest reported accuracy of over 99%.

1	Introduction	132
2	Preliminaries	133
3	Saliency analysis of neural networks	139
4	Kolmogorov–Arnold Networks (KAN)	140
5	Small neural networks	143
6	Symbolic regression	144
7	Transformers	145
8	Concluding remarks	146
	References	147

1. Introduction

This article is a sequel to [7, 8], where the second author studied Kronecker coefficients using data-driven approaches, including machine learning (ML) models. In particular, [8] applies convolutional neural networks (CNNs) and LightGBM to datasets of Kronecker coefficients, demonstrating that the trained classifiers achieve high accuracy ($\approx 98\%$) in predicting whether a Kronecker coefficient vanishes. These results suggest that further data-driven analysis may uncover new structural insights into Kronecker coefficients. Indeed, a subsequent paper [7] employs principal component analysis (PCA) and kernel methods to formulate a sufficient condition for the nonvanishing of Kronecker coefficients, providing an effective tool for their study. However, the underlying reasons for the high accuracy of ML models remain largely unexplained, as CNNs and LightGBM are not easily interpretable.

In this article, we employ various interpretable approaches to whitebox the ML results on the binary classification task of predicting which Kronecker coefficients of the symmetric group \mathfrak{S}_n are zero. First, we perform a gradient saliency analysis similar to [3] on a classifier network. Second, we use Kolmogorov–Arnold Networks (KANs), developed by the third author in collaboration [9], both to examine the learned function and as a symbolic regressor. Third, we train a very small neural network and analyze the parameters of the learned function. Finally, we apply PySR [2] for symbolic regression on the datasets.

Our analysis and models use triples of partitions of n as input, as well as the *b-loading* of partitions, introduced in [7] as a result of PCA applied to an embedding that captures differences in partitions. Both the saliency analysis and KANs reveal that when a partition is represented as an n -dimensional vector, the first few and last few entries are more significant than the middle ones. When using *b-loadings*, all three interpretable models yield comparable results, achieving approximately 83% accuracy in each case (the theoretical upper bound on the precision obtainable from *b-loadings* is around 85%). Since these models are interpretable, we can extract explicit formulas for their decision functions.

Additionally, beyond the main theme of this article, we include results from transformer models, which achieve the highest reported accuracy of over 99%. Understanding the interpretability of these transformer models and its implications for the computational complexity or approximation of the underlying decision problem would be an interesting direction for future research.

Acknowledgments. Lee is partially supported by a grant from the Simons Foundation (#712100). The work of Ruehle is supported by the NSF grants PHY-2210333 and PHY-2019786 (The NSF AI Institute for Artificial Intelligence and Fundamental Interactions), as well as startup funding from Northeastern University. We are grateful for support from CMSA for the Mathematics and Machine Learning Program during the Fall 2024 semester, where this project was initiated.

2. Preliminaries

2.1. Kronecker coefficients

Let \mathfrak{S}_n be the symmetric group of degree n . The irreducible representations S_λ of \mathfrak{S}_n over \mathbb{C} are parametrized by partitions λ of n , written as $\lambda \vdash n$. The tensor product of two irreducible representations S_λ and S_μ ($\lambda, \mu \vdash n$) is decomposed into a sum of irreducible representations:

$$S_\lambda \otimes S_\mu = \bigoplus_{\nu \vdash n} g_{\lambda, \mu}^\nu S_\nu \quad (g_{\lambda, \mu}^\nu \in \mathbb{Z}_{\geq 0}).$$

The decomposition multiplicities $g_{\lambda, \mu}^\nu$ are called the *Kronecker coefficients*.

In stark contrast to the analogous Littlewood–Richardson coefficients for $\mathrm{GL}_N(\mathbb{C})$, no combinatorial description of $g_{\lambda, \mu}^\nu$ has been found since Mur-naghan [10] first posed the question in 1938. This remains one of the central open problems in combinatorial representation theory, with only partial results available (e.g., [1]). It has also been shown that determining whether a given Kronecker coefficient is non-zero is NP-hard [6].

For later discussion, we note that there are symmetries among $g_{\lambda, \mu}^\nu$.

Lemma 2.1. [4, p.61] Let $\lambda, \mu, \nu \vdash n$. Then the Kronecker coefficients $g_{\lambda, \mu}^\nu$ are invariant under the permutations of λ, μ, ν . That is, we have

$$g_{\lambda, \mu}^\nu = g_{\mu, \lambda}^\nu = g_{\lambda, \nu}^\mu = g_{\nu, \lambda}^\mu = g_{\mu, \nu}^\lambda = g_{\nu, \mu}^\lambda.$$

2.2. Different matrix and \mathbf{b} -loading

For $n \in \mathbb{Z}_{>0}$, let $\mathcal{P}(n)$ be the set of partitions of n . We identify each element λ of $\mathcal{P}(n)$ with a sequence of length n by appending as many 0-entries as needed. We also consider $\mathcal{P}(n)$ as an ordered set by the lexicographic order.

Example 2.2. When $n = 6$, we have

$$\begin{aligned} \mathcal{P}(6) = \{ & (6, 0, 0, 0, 0, 0), (5, 1, 0, 0, 0, 0), (4, 2, 0, 0, 0, 0), (4, 1, 1, 0, 0, 0), \\ & (3, 3, 0, 0, 0, 0), (3, 2, 1, 0, 0, 0), (3, 1, 1, 1, 0, 0), (2, 2, 2, 0, 0, 0), \\ & (2, 2, 1, 1, 0, 0), (2, 1, 1, 1, 1, 0), (1, 1, 1, 1, 1, 1)\}. \end{aligned}$$

The size of the set $\mathcal{P}(n)$ will be denoted by $p(n)$. Define a $p(n) \times p(n)$ symmetric matrix $Z_n = [z_{\lambda, \mu}]_{\lambda, \mu \in \mathcal{P}(n)}$ by

$$z_{\lambda, \mu} = \|\lambda - \mu\|_1 := \sum_{i=1}^n |\lambda_i - \mu_i|$$

for $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_n)$ and $\mu = (\mu_1, \mu_2, \dots, \mu_n) \in \mathcal{P}(n)$. The matrix Z_n will be called the *difference matrix* of $\mathcal{P}(n)$.

Example 2.3. When $n = 6$, we obtain

$$Z_6 = \begin{bmatrix} 0 & 2 & 4 & 4 & 6 & 6 & 6 & 8 & 8 & 8 & 10 \\ 2 & 0 & 2 & 2 & 4 & 4 & 4 & 6 & 6 & 6 & 8 \\ 4 & 2 & 0 & 2 & 2 & 2 & 4 & 4 & 4 & 6 & 8 \\ 4 & 2 & 2 & 0 & 4 & 2 & 2 & 4 & 4 & 4 & 6 \\ 6 & 4 & 2 & 4 & 0 & 2 & 4 & 4 & 4 & 6 & 8 \\ 6 & 4 & 2 & 2 & 2 & 0 & 2 & 2 & 2 & 4 & 6 \\ 6 & 4 & 4 & 2 & 4 & 2 & 0 & 4 & 2 & 2 & 4 \\ 8 & 6 & 4 & 4 & 4 & 2 & 4 & 0 & 2 & 4 & 6 \\ 8 & 6 & 4 & 4 & 4 & 2 & 2 & 2 & 0 & 2 & 4 \\ 8 & 6 & 6 & 4 & 6 & 4 & 2 & 4 & 2 & 0 & 2 \\ 10 & 8 & 8 & 6 & 8 & 6 & 4 & 6 & 4 & 2 & 0 \end{bmatrix}.$$

Since Z_n is symmetric, all its eigenvalues are real. Moreover, Z_n is irreducible, and the Perron–Frobenius theorem for matrices with nonnegative entries [5, Section III.2] guarantees that Z_n has a unique eigenvalue of largest magnitude and that the corresponding eigenvector can be chosen to have strictly positive components. Thus the following definition is well-defined.

Definition 2.4. Let $\mathbf{w} = (w_\lambda)_{\lambda \in \mathcal{P}(n)}$ be an eigenvector of the largest eigenvalue of Z_n such that $w_\lambda > 0$ for all $\lambda \in \mathcal{P}(n)$. Denote by w_{\max} (resp. w_{\min}) a maximum (resp. minimum) of $\{w_\lambda\}_{\lambda \in \mathcal{P}(n)}$. Define

$$b_\lambda := 100 \times \frac{w_\lambda - w_{\min}}{w_{\max} - w_{\min}} \quad \text{for } \lambda \in \mathcal{P}(n).$$

The value b_λ is called the *b-loading* of partition $\lambda \in \mathcal{P}(n)$.

Example 2.5. When $n = 6$, we have

$$\mathbf{w} \approx (0.4045, 0.2961, 0.2662, 0.2393, \\ 0.3061, 0.2318, 0.2393, 0.3061, 0.2662, 0.2961, 0.4045),$$

and the b -loadings are given by

$$(b_\lambda)_{\lambda \in \mathcal{P}(n)} = (100.00, 37.25, 19.93, 4.36, \\ 43.01, 0.00, 4.36, 43.01, 19.93, 37.25, 100.00).$$

Remark 2.6. The a -loading is also defined in [7]. However, since it is not used in this article, we do not introduce it here.

For $\mathbf{t} = (\lambda, \mu, \nu) \in \mathcal{P}(n)^3$, we will write

$$g(\mathbf{t}) := g'_{\lambda, \mu}.$$

Definition 2.7. Let $\mathbf{t} = (\lambda, \mu, \nu) \in \mathcal{P}(n)^3$. Define the b -loading of \mathbf{t} , denoted by $b(\mathbf{t})$, to be the sum of the b -loadings of λ, μ and ν , i.e.,

$$b(\mathbf{t}) := b_\lambda + b_\mu + b_\nu.$$

The histograms of b -loadings $b(\mathbf{t})$ clearly indicate that they follow gamma distributions (Figure 1). Moreover, when examining the distributions of loadings based on whether the Kronecker coefficients $g(\mathbf{t})$ are zero or nonzero (Figure 2), it is evident that the minimum values of loadings can be used to define vertical lines that separate the red regions from the blue ones.

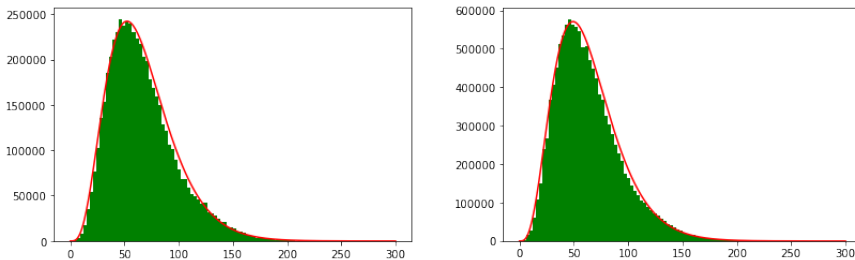


Figure 1: Histograms of b -loadings of $\mathbf{t} \in \mathcal{P}(n)^3$ for $n = 15$ (left) and 16 (right) along with curves (red) of gamma distributions.

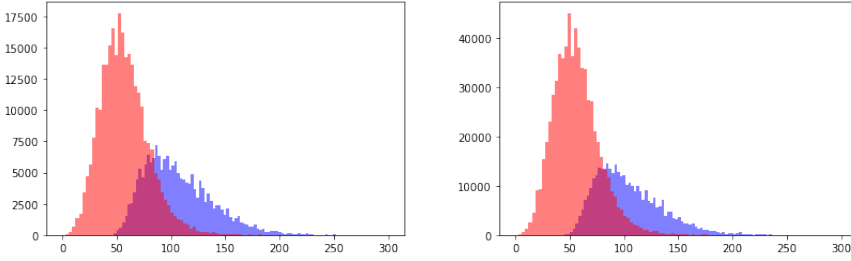


Figure 2: Histograms of b -loadings for $n = 12$ (left) and $n = 13$ (right). The red (resp. blue) region represents the numbers of \mathbf{t} such that $g(\mathbf{t}) \neq 0$ (resp. $g(\mathbf{t}) = 0$).

With this in mind, define

$$b_\star := \min\{b(\mathbf{t}) : g(\mathbf{t}) = 0, \mathbf{t} \in \mathcal{P}(n)^3\}.$$

Once the values of b_\star is determined, this definition provides a sufficient condition for $g(\mathbf{t}) \neq 0$:

- (1) For $\mathbf{t} \in \mathcal{P}(n)^3$, if $b(\mathbf{t}) < b_\star$ then $g(\mathbf{t}) \neq 0$.

Example 2.8.

- 1) When $n = 18$, we obtain $b_\star \approx 44.18$. Now that the b -loading of

$$\mathbf{t} = ((12, 4, 2), (8, 4, 2, 2, 1, 1), (5, 4, 3, 3, 1, 1, 1))$$

is readily computed to be approximately $41.07 < b_\star$, we immediately conclude that $g(\mathbf{t}) \neq 0$ by (1).

- 2) When $n = 20$, there are 246,491,883 triples $\mathbf{t} \in \mathcal{P}(20)$. Among them, 78,382,890 triples satisfy $b(\mathbf{t}) < b_\star \approx 43.74$ so that $g(\mathbf{t}) \neq 0$. The percentage of these triples is about 31.8%.

2.3. The threshold b_\star and additional remarks

The value b_\star determines instances of \mathbf{t} for which $g(\mathbf{t}) \neq 0$ with 100% confidence in a significant number of cases. As mentioned in the introduction, in the following sections, we will derive formulas in terms of b -loadings, using interpretable ML models that predict whether $g(\mathbf{t}) \neq 0$ with approximately 83% accuracy.

The values of b_* are computed for $6 \leq n \leq 20$ in [7] and for $n = 21, 24, 27, 30, 33, 36$ assuming a conjecture. The ratio of \mathbf{t} satisfying $b(\mathbf{t}) < b_*$ exhibits an increasing trend for $n \geq 9$, as shown in Figure 3.

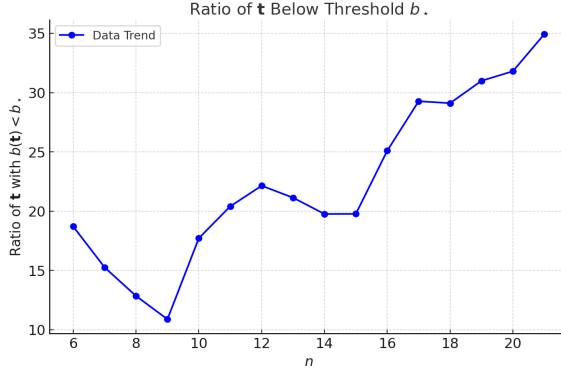


Figure 3: The ratio of \mathbf{t} satisfying $b(\mathbf{t}) < b_*$.

The accuracy of the binary classification networks used in Section 3, where the input features are the triples $\mathbf{t} = (\lambda, \mu, \nu)$ of partitions, is on a par with the accuracy obtained using b -loadings only. For the latter, we find an optimal decision boundary by training a binary decision tree for different values of n . Running the algorithm on a 10% test set with ten-fold cross validation gives a stable performance of around 80% across the range of n studied, see the blue curve in Figure 4. The error bars indicate one-sigma confidence intervals from the cross validation. The maximum classification accuracy that can be achieved based on the b -loadings alone is around 85% (see the orange line).

In light of the NP-hardness of the decision problem, this suggests that the average-case complexity is not NP-hard, since computing the b -values is polynomial, and the data science methods we employ run in polynomial time. Finally, we want to point out that the accuracy of predicting zero vs. non-zero coefficients remains almost constant, albeit the percentage of non-zero coefficients varies (see the right plot of Figure 4). At $n = 9$, the majority class shifts from zero to non-zero coefficients. However, the classifier accuracy is unaffected by this.

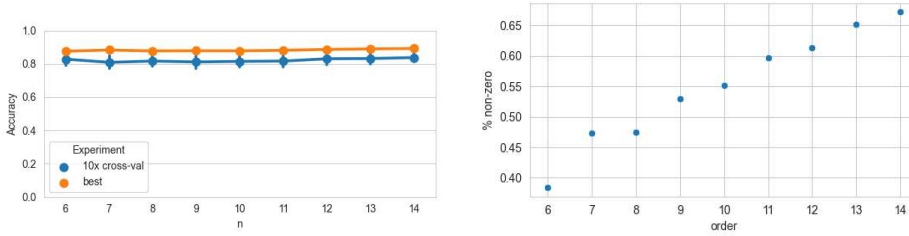


Figure 4: Left: Accuracies for classification based on b -loadings for different decision boundaries. Right: Percentage of non-zero Kronecker coefficients.

2.4. Data structures

2.4.1. $3n$ -dimensional input. The decision problem is uniquely determined by the triple of integer partitions $\mathbf{t} = (\lambda, \mu, \nu)$, which we represent as a $3n$ -dimensional vector. PCA reveals a three-fold degeneracy in the principal components (PCs), arising from the symmetric role of (λ, μ, ν) in the problem (recall Lemma 2.1), and a rapid drop in the explained variance ratio. Examining the Spearman correlation matrix also shows a negative correlation among the first few features and the rest, and a positive correlation among the last features. This stems from the lexicographic ordering and the constraint that the total sum of parts equals n for each partition. We plot the explained variance of the PCs, their loadings, and the correlation matrix in Figure 5.

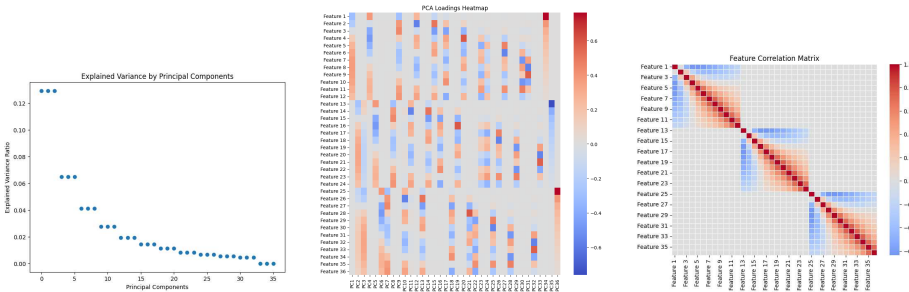


Figure 5: Explained variance ratio of Principal Components, a heatmap of their loadings, and the Spearman correlation matrix of the inputs for $n = 12$.

2.4.2. 18-dimensional input. PCA, saliency analysis (see Section 3), and the Kolmogorov–Arnold Network (see Section 4) all indicate that the

first and last three entries of each partition vector are the most significant. Based on this observation, we conduct some of the experiments using only these 18 components instead of the full $3n$ -dimensional vector.

2.4.3. 1-dimensional input. Finally, we conduct experiments using only the b -loading $b(\mathbf{t})$ for $\mathbf{t} = (\lambda, \mu, \nu)$. We also tested using the triple $(b_\lambda, b_\mu, b_\nu)$ instead of their sum $b(\mathbf{t})$. However, the accuracy remained unchanged, suggesting that the individual values $(b_\lambda, b_\mu, b_\nu)$ do not carry more information than the sum.

3. Saliency analysis of neural networks

Even very simple neural networks (NNs) achieve relatively high accuracy on the binary classification task. We found that the results are not highly sensitive to architectural choices, such as the number of hidden layers, the number of nodes per layer, or the activation function. The plots below correspond to a NN with two hidden layers, each containing 16 nodes with ReLU activation, trained for 60 epochs with early stopping if the test set accuracy begins to diverge. The $3n$ -dimensional dataset from §2.4.1 is used in this experiment. Training is performed using the Adam optimizer with cross-entropy loss. While the embedding dimension seems very small for this NN, we performed a PCA of the embedded features and found that the network effectively utilizes only a few dimensions. Depending on n , 80% of the explained variance is captured within the first 3–7 PCs of the embedding vectors.

In order to understand what the algorithm bases its decision on, we perform a gradient saliency analysis, which means we compute the gradient of the NN output with respect to its input. Larger absolute values of the gradient indicate the final result depends more strongly on these particular input parameters. While we present the input as a $3n$ -dimensional vector to the NN, we display the saliency map as a $3 \times n$ matrix, where each row corresponds to one of the three integer partitions (λ, μ, ν) . We also scale our input to zero mean and unit variance and use a train:test split of 90:10.

3.1. Results

The saliency analysis for $6 \leq n \leq 14$ is given in Figure 6. Red indicates high saliency while blue indicates low saliency. We see that the middle entries of each partition are the least relevant for the decision. The asymmetry in the

saliencies of the three partition vectors comes from the fact that the train and test set are selected randomly, which breaks their symmetric appearance.

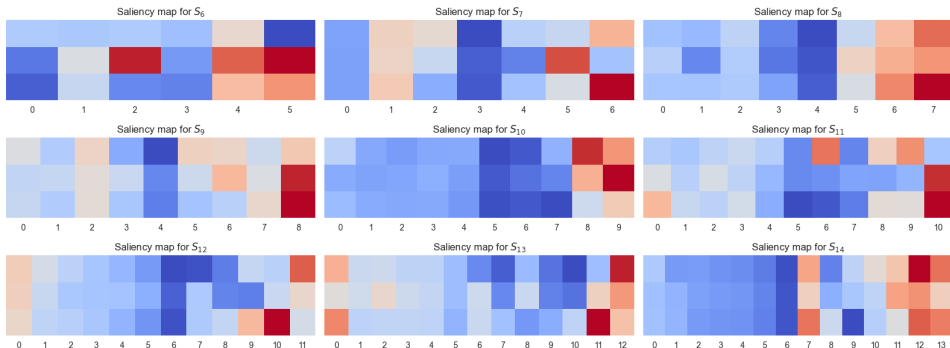


Figure 6: Saliency heatmap for the partition vectors from least relevant (cool blue) to most relevant (dark red).

Based on our analysis, we trained a classifier using only the first three and last three entries of each partition—corresponding to the dataset described in §2.4.2. This reduction had little to no impact on prediction accuracy, confirming the saliency analysis result that these entries are the most important.

Given that the dataset is somewhat unbalanced—the majority class comprises around 60% of the dataset for larger n -values in the range we studied, though the majority class shifts between small and large n (see the right plot of Figure 4)—we also examined the confusion matrix. While the confusion matrix is skewed, it is not always towards the majority class. For example, for $n = 8$, where the dataset is relatively balanced, 57% of the misclassified Kronecker coefficients were misclassified as non-vanishing (but 47.5% of all coefficients are non-vanishing), while for $n = 14$, 53% of the misclassified coefficients were misclassified as non-vanishing (and 67% of all Kronecker coefficients are non-vanishing in this case).

4. Kolmogorov–Arnold Networks (KAN)

KANs are by design more interpretable than neural networks and are well-suited for applications in scientific discovery [9]. Overall, they are smaller models, and the non-linear functions on the edges of the computational graph, which we fit with cubic splines, allow for easy visualization.

4.1. Model architecture

We find that the classification performance shows architecture dependence. We investigate KAN architectures of shapes $[3n, \ell, 1]$ and $[3n, \ell, \ell, 1]$. We follow the notation of [9], which indicates that the KAN has a $3n$ -dimensional input, one (respectively two) “hidden layers” of width ℓ , and one scalar output. Throughout our experiments, we use cubic splines, a grid size of 5, and train for 200 steps with a batch size of 1024, binary cross entropy loss with logits (meaning a sigmoid activation function is applied to the output of the KAN), and standard parameters otherwise.

We also trained a $[1, 1]$ -KAN (essentially a cubic spline) to classify based on the dataset of b -loadings in §2.4.3 and use KAN’s `auto_symbolic` function for symbolic regression.

4.2. Results

For the $[3n, \ell, 1]$ KANs, we find that, after pruning, only a single node in the hidden layer is utilized. This aligns with our observation that the feed-forward NN also primarily utilizes a low-dimensional embedding manifold. Effectively, the KAN is learning a non-linear transformation of each entry in the partition vectors, which it then sums up and applies another non-linearity to obtain a prediction for the output. Accuracy percents are in the low 80’s (for example, 83% for $n = 12$).

For the $[3n, \ell, \ell, 1]$ KANs, accuracy is higher (for example, 89% for $n = 12$). The KAN is utilizing more than one node in the first hidden layer, but still typically not all ℓ at its disposal. In the second hidden layer, it is utilizing even fewer.

For both architectures, we see that the KAN is mostly paying attention to roughly the first half of the entries (in contrast to the NNs, where a saliency analysis showed that the NN is mostly paying attention to the first few and last few entries). The information contained in these entries is comparable, as knowing either the first half or the first and last few entries of a partition vector significantly constrains the remaining entries. Indeed, we observe that providing either set to the classifier results in accuracies in the mid 80% range.

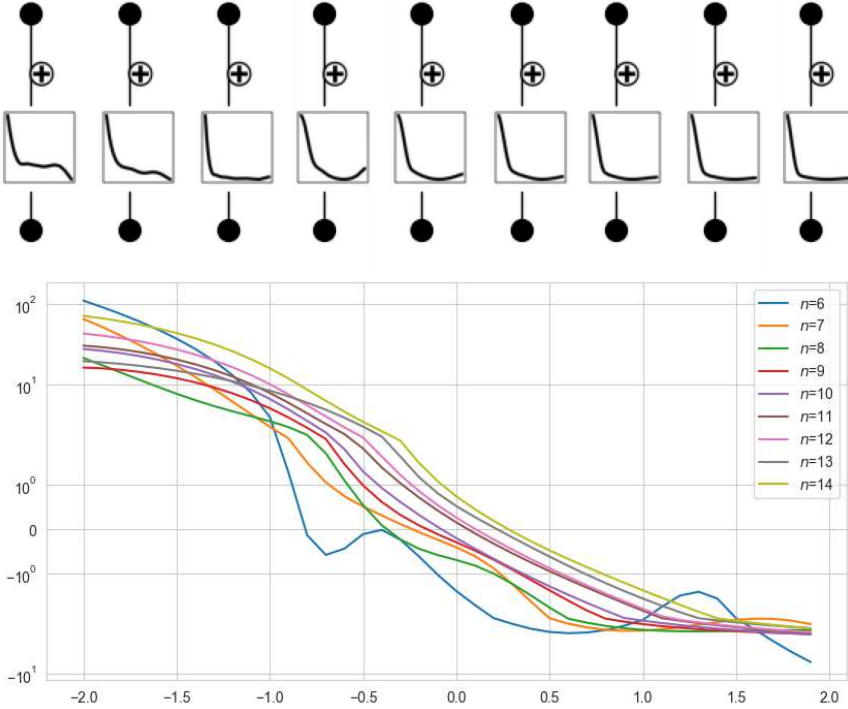


Figure 7: Top: [1,1] KANs for binary classification based on b -loadings for $6 \leq n \leq 14$. Bottom: Plot of the spline for $-2 \leq x \leq 2$.

For the [1, 1]-KAN, we find accuracies in the low to mid-80s, comparable to those in Figure 4. The function learned by the KAN to classify based on the b -loadings looks similar for all n ; see Figure 7. For binary classification with the sigmoid function

$$(2) \quad \sigma(x) := \frac{1}{1 + \exp(-x)}$$

the decision boundary is at $x = 0$ where $\sigma(0) = 0.5$. In our [1, 1]-KAN, we use $\sigma(f(x))$. While the optimizer will find an $f(x)$ that makes the transition between the classes sharper, the relevant property of f for the final class assignment is where it changes sign.

We train the KAN with normalized inputs $\tilde{b} := \frac{b - \mathbf{m}}{\mathbf{s}}$, where \mathbf{m} is the mean of the distribution of the b -loadings and \mathbf{s} is the standard deviation. We find that all splines are monotonic for a wide range of \tilde{b} -values, and they

have a unique zero close to $\tilde{b} = 0$, i.e., close to $b = \mathbf{m}$. Hence, when using \tilde{b} -values, we can approximate the spline by $\tilde{f}(x) \approx -x + c$ with a constant c . We observe from the spline that c lies between -0.5 and 0.5 and is increasing as n increases. If we set $c = 0$, this function predicts the zero/non-zero classes with accuracies in the low 80's range, which is comparable to those in Figure 4. Consequently,

$$(3) \quad \tilde{F}_1(\tilde{b}) = \sigma(-\tilde{b}) \quad \text{or} \quad F_1(b) = \sigma(-b + \mathbf{m})$$

serves as an effective decision function for $6 \leq n \leq 14$. Understanding why the mean \mathbf{m} of the b -loadings can act as a decision boundary would be an interesting direction for further investigation.

5. Small neural networks

5.1. Model architecture

The next interpretable ML model is a small neural networks (NNs) with one hidden layer containing k nodes, using ReLU activation for the hidden layer and a sigmoid activation for the output. Since there is no improvement in accuracy when using the triple $(b_\lambda, b_\mu, b_\nu)$ as input instead of the sum $b(\mathbf{t}) = b_\lambda + b_\mu + b_\nu$, $\mathbf{t} = (\lambda, \mu, \nu)$, we take $b(\mathbf{t})$ as the input, hence using the dataset in §2.4.3. Then the layers in the model are described by

$$\mathbb{R} \rightarrow \mathbb{R}^k \rightarrow \mathbb{R},$$

with a total of $3k + 1$ parameters.

To be more precise, let $\sigma(x)$ be the sigmoid function as before and $r(x) := \max(0, x)$ the ReLU function. Then, the NN model is given by the function:

$$(4) \quad F(x) = \sigma \left(\gamma_0 + \sum_{i=1}^k \gamma_i r(\alpha_i + \beta_i x) \right),$$

where $\alpha_i, \beta_i, \gamma_i$ ($i = 1, 2, \dots, k$) and γ_0 are the parameters (or weights).

5.2. ML results

We run the model in (4) for $9 \leq n \leq 14$ with a 66:33 train-test split. Across all n and various values of k , we obtain a test accuracy of approximately 83%. For example, when $n = 14$ and $k = 7$, the model has 22 parameters

and we can easily look at them; the matrix $\begin{bmatrix} \alpha_i \\ \beta_i \\ \gamma_i \end{bmatrix}_{1 \leq i \leq 7}$ along with γ_0 are given by

$$\begin{bmatrix} 0 & 0 & 0 & -2.4159462 & -2.8786569 & 2.3823507 & -3.1325226 \\ -1.1845468 & -2.146678 & -0.7272987 & 0.26937068 & 0.32904944 & 0.8263043 & 0.12642458 \\ -0.591498 & -0.35002795 & -0.27688783 & -0.52213764 & -0.24775109 & 0.27229485 & -0.6243779 \end{bmatrix},$$

$$\gamma_0 = 2.1719017.$$

Substituting these values into (4) yields an accuracy of approximately 83%. However, we observe that $r(\alpha_i + \beta_i x) \equiv 0$ with $\alpha_i = 0$ and $\beta_i < 0$ for $i = 1, 2, 3$, indicating that $k = 4$ would have been sufficient.

Motivated by this observation, we examine the networks for different values of k and find that logistic regression is sufficient. Indeed, a logistic regression model achieves an accuracy of approximately 83% across all n . For instance, when $n = 14$, the decision function for a threshold of 0.5 is given by

$$(5) \quad F_2(b) = \sigma(-0.08550718 \times b + 7.27970193),$$

with a decision boundary at approximately $b = 85.14$ and an accuracy of about 84%. In comparison, the decision boundary from (3) is $b = \mathbf{m} \approx 72.07$ with an accuracy 83%. Examining all coefficients, we find that the best separation occurs near $b = 80$, achieving an accuracy of about 85%.

6. Symbolic regression

As another approach to interpretable ML for predicting whether $g(\mathbf{t}) \neq 0$ from the b -loading of the partition triple $\mathbf{t} = (\mu, \nu, \lambda)$, we perform symbolic regression on the neural networks (NNs). Specifically, we first train a fully-connected NN with 4 hidden layers, each containing 128 nodes. We alternate activation functions between `tanh` and `Gelu` across the hidden layers and apply the σ activation function to the final 1-dimensional output layer. The NN is trained over 40,000 epochs using an Adam optimizer with learning rate 10^{-3} . Finally, we use the trained NN to generate data for symbolic regression, which we perform using PySR [2].

As a result, we identify the following candidate function, which approximates the NN model reasonably well:

$$(6) \quad F_3(b) = \left[\cos \left(\sqrt{\cos(\sin(\log(b^2)))^2} + \cos(\log(b)) \right) \right]^3.$$

The predictor based on the condition $F_3(b) \geq 1/2$ achieves an accuracy of approximately 83%. The performance is summarized in Figure 8. In particular, the decision boundary is again around $b = 80$.

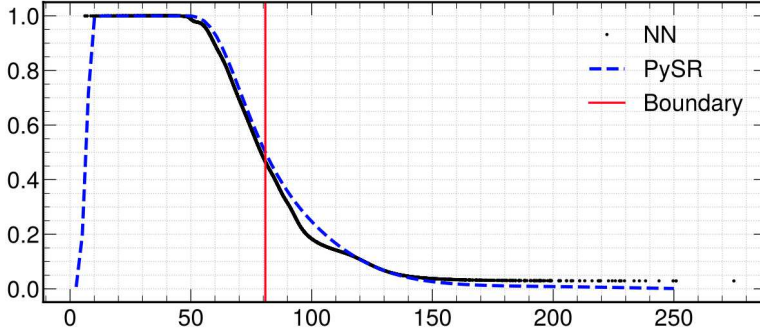


Figure 8: Comparison of the expression (6), obtained through symbolic regression, with the output of a trained neural network.

7. Transformers

In this section, aside from the main theme of this article, we present ML results from transformer models, which achieve the highest reported accuracy of over 99%. We switch back to the notation $g_{\mu,\nu}^\lambda = g(\mathbf{t})$ for $\mathbf{t} = (\mu, \nu, \lambda)$.

Evidently highly non-trivial combinations of the components of the partitions μ , ν and λ influence $g_{\mu,\nu}^\lambda$. Motivated with the success of the attention mechanism [11], we use transformer-like architecture to build our model. In particular, we use 12 multi-head attention blocks, each with 8 heads. For a given n , we use $\{0, 1, \dots, n\}$ as our token table, thus, each entry in a given partition μ is a token. Our context window is equal to $3n$. For the positional embedding, we use encoding that does not distinguish between individual parts of a given partition μ . Due to this, we do not apply causal mask to the attention matrices. We do, however, distinguish different arguments μ , ν and λ of $g_{\mu,\nu}^\lambda$ by assigning them positions 0, 1 or 2, respectively. The input to the transformer is then produced by interleaving position 0, 1 and 2 of the stacked array $[\mu, \nu, \lambda]$, as shown on Figure 9.

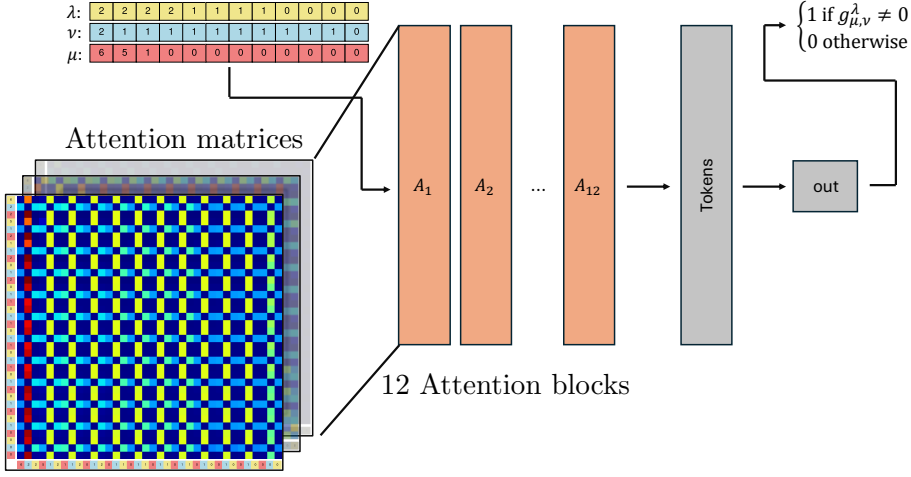


Figure 9: Transformer architecture for deciding whether $g_{\mu,\nu}^\lambda \neq 0$ based on the partitions (μ, ν, λ) . The coloring of the input blocks μ, ν and λ corresponds to the positional encoding used in the transformer.

Given that the predicted object is $g_{\mu,\nu}^\lambda$, rather than a sequence, we modify the head of the transformer to apply over the full context window, thereby reducing the dimensionality of the output to 1. The model is then trained as a binary classifier, with the classes being $g_{\mu,\nu}^\lambda = 0$ or $g_{\mu,\nu}^\lambda \neq 0$. We use binary cross-entropy as a loss function and train for 1.25×10^5 epochs using an AdamW optimizer with no scheduler. We find that a learning rate of $\alpha = 3 \times 10^{-4}$ with $\beta = (0.9, 0.95)$ and $\epsilon = 10^{-8}$ performs reasonably well for the classification task. To ensure stability during training, we clip gradient norms to 1.0. The training was performed on a single Nvidia P100 GPU, on which we can use a batch size of 128.

The accuracies on a test set (with a 33% split) significantly outperform previous models [8], as shown in Table 1.

8. Concluding remarks

We applied interpretable ML models to the decision problem of which Kronecker coefficients are vanishing. Using a $3n$ -dimensional input, our saliency analysis and KANs indicate knowing either the first half or the first and last few entries of a partition vector is sufficient for classification. Indeed,

n	Accuracy %		
	Transformer	CNN	LGBM
12	99.42	96.97	97.14
13	99.65	97.73	98.37
14	99.06	97.72	98.45

Table 1: Performance of the transformer-like model for different values of n .

an 18-dimensional input that omits the middle part performed equally well. When using a one-dimensional input of b -loadings, our interpretable ML models revealed simple formulas, as shown in (3), (5) and (6), all achieving approximately 83% accuracy.

These results also show the limitations of using a one-dimensional input such as the b -loadings. It would be ideal to find a simple decision function on two- or three-dimensional vectors with significantly higher accuracy. This suggests that developing additional informative features beyond the b -loading may be necessary.

Despite the decision problem being NP-hard, much better classifiers exist. We constructed transformer models and achieved the highest reported accuracy of over 99%. Understanding the decision process of the transformer models—for example, through mechanistic interpretability—would be an interesting direction for future research.

Overall, the ability to analyze dataset saliency and derive simple decision functions highlights the importance and potential of interpretable ML in tackling research problems in mathematics.

References

- [1] J. Blasiak. *An insertion algorithm for catabolizability*. European J. Combin. **33** (2012), 267–276.
- [2] M. Cranmer. *Interpretable machine learning for science with PySR and SymbolicRegression.jl*. (2023). [arXiv:2305.01582](https://arxiv.org/abs/2305.01582).
- [3] A. Davies, P. Veličković, L. Buesing, et al. *Advancing mathematics by guiding human intuition with AI*. Nature **600** (2021), 70–74.
- [4] W. Fulton and J. Harris. *Representation Theory*. Graduate Texts in Mathematics **129** (1991).

- [5] F. Gantmacher. Applications of The Theory of Matrices. Interscience Publishers (1959).
- [6] C. Ikenmeyer, K. D. Mulmuley, and M. Walter. *On vanishing of Kronecker coefficients*. Comput. Complex. **26** (2017), 949–992.
- [7] K.-H. Lee. *Data-scientific study of Kronecker coefficients*. (2023). [arXiv:2310.17906](https://arxiv.org/abs/2310.17906).
- [8] K.-H. Lee. *Machine-learning Kronecker coefficients*. (2023). [arXiv:2306.04734](https://arxiv.org/abs/2306.04734).
- [9] Z. Liu, Y. Wang, S. Vaidya, et al. *KAN: Kolmogorov–Arnold Networks*. (2024). [arXiv:2404.19756](https://arxiv.org/abs/2404.19756).
- [10] D. Murnaghan. *The analysis of the direct product of irreducible representations of the symmetric groups*. Amer. J. Math. **60** (1938), 44–65.
- [11] A. Vaswani, N. Shazeer, N. Parmar, et al. *Attention is all you need*. Advances in Neural Information Processing Systems **30** (2017), 5998–6008.

DEPARTMENT OF PHYSICS AND ASTRONOMY, UNIVERSITY OF NEW HAMPSHIRE
DURHAM, NH 03824, USA
E-mail address: giorgi.butbaia@unh.edu

DEPARTMENT OF MATHEMATICS, UNIVERSITY OF CONNECTICUT
STORRS, CT 06269, USA
KOREA INSTITUTE FOR ADVANCED STUDY
SEOUL, 02455, REPUBLIC OF KOREA
E-mail address: khlee@math.uconn.edu

DEPARTMENT OF PHYSICS AND DEPARTMENT OF MATHEMATICS
NORTHEASTERN UNIVERSITY, BOSTON, MA 02115, USA
NSF INSTITUTE FOR ARTIFICIAL INTELLIGENCE AND FUNDAMENTAL INTERAC-
TIONS CAMBRIDGE, MA 02139, USA
E-mail address: f.ruehle@northeastern.edu

RECEIVED FEBRUARY 17, 2025

ACCEPTED JUNE 03, 2025