


Machine learning Kronecker coefficients

Kyu-Hwan Lee 

*Department of Mathematics
University of Connecticut
Storrs, CT 06269, USA*

*Korea Institute for Advanced Study
Seoul 02455, Republic of Korea
khlee@math.uconn.edu*

Received 12 January 2024

Accepted 20 May 2024

Published 26 July 2024

The Kronecker coefficients are the decomposition multiplicities of the tensor product of two irreducible representations of the symmetric group. Unlike the Littlewood–Richardson coefficients, which are the analogues for the general linear group, there is no known combinatorial description of the Kronecker coefficients, and it is an NP-hard problem to decide whether a given Kronecker coefficient is zero or not. In this paper, we show that standard machine learning algorithms such as Nearest Neighbors, Convolutional Neural Networks and Gradient Boosting Decision Trees may be trained to predict whether a given Kronecker coefficient is zero or not. Our results show that a trained machine can efficiently perform this binary classification with high accuracy (≈ 0.98).

Keywords: Kronecker coefficients; symmetric group; machine learning.

1. Introduction

One of the major goals in representation theory is to decompose a representation into irreducible representations, and algebraic combinatorics plays a crucial role as it produces a practical way to describe such a decomposition. A prototype can be found in the decomposition of the tensor product of two irreducible representations of the general linear group $GL_N(\mathbb{C})$, where the Littlewood–Richardson rule completely describes the decomposition using skew semi-standard tableaux. Therefore, it comes with surprise to realize that we do not have a similar success with the symmetric group until now.

To be precise, let \mathfrak{S}_n be the symmetric group of degree n and consider representations of \mathfrak{S}_n over \mathbb{C} . The irreducible representations S_λ of \mathfrak{S}_n are parametrized by partitions $\lambda \vdash n$, and the dimension of S_λ is equal to the number of standard tableaux of shape λ . Consider the tensor product of two irreducible representations

S_λ and S_μ for $\lambda, \mu \vdash n$. Then their tensor product 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 with the Littlewood–Richardson coefficients for $\mathrm{GL}_N(\mathbb{C})$, no combinatorial description has been known for $g_{\lambda, \mu}^\nu$ since Murnaghan [1] initially posed the question in 1938, and it is still considered as one of the main problems in the combinatorial representation theory. For special cases, various combinatorial formulas are known due to Remmel [2], Ballantine–Orellana [3], Remmel–Whitehead [4], Blasiak–Mumuley–Sohoni [5] and Blasiak [6]. Recently, the coefficients $g_{\lambda, \mu}^\nu$ have also been studied from the viewpoint of computational complexity. In particular, it is shown by Ikenmeyer, Mumuley and Walter [7] that deciding whether a given Kronecker coefficient is non-zero is NP-hard, and other interesting results are obtained by Pak and Panova [8, 9].

In this paper, we apply machine learning to the Kronecker coefficients $g_{\lambda, \mu}^\nu$ to show that a machine can be trained to learn these coefficients. Though this seems to be the first time for machine learning to be applied to the coefficients $g_{\lambda, \mu}^\nu$, there has been mounting evidence that a machine can be trained to learn mathematical structures. Indeed, starting with superstring theory [10–14], machine learning has been applied, for example, to group theory [15], graph theory [16, 17], metric geometry [18, 19], knot invariants [20], and to number theory [21–23]. Moreover, a new paradigm of AI-aided research in mathematics is emerging [24].

As the main machine learning methods for this paper, we utilize Nearest Neighbors, Convolutional Neural Networks and Gradient Boosting Decision Trees (specifically, LightGBM [25]) as they perform better than other standard machine learning algorithms. For a broad introduction to machine learning, see [26, 27]. Our datasets of the Kronecker coefficients $g_{\lambda, \mu}^\nu$ are generated by SAGEMATH [28].

The results of our experiments clearly show that a machine can be trained to distinguish between nonzero and zero Kronecker coefficients $g_{\lambda, \mu}^\nu$ with high accuracy ($\approx 98\%$). We focus on \mathfrak{S}_n for $12 \leq n \leq 14$ since general features are clear enough from these cases, and the sizes of datasets are suitable for our purposes. (When $n = 14$, there are 2, 258, 526 data points in our dataset.) We hope that the results in this paper may invite researchers to further study Kronecker coefficients by taking machine learning or AI-aided approaches. Specifically, attribute analysis would be the next step, which may produce some conjectures to describe Kronecker coefficients.

This paper is organized as follows. In Sec. 2, the generation of training data and the experimental set-up is explained. In Sec. 3, we document the experimental

outcomes and make some observations. In Sec. 4, we offer some concluding remarks and directions for further research.

2. Methodology

In this section, we explain our experimental set-up. In particular, we present details about how to construct our datasets and overview the machine learning strategies adopted.

2.1. Generating datasets

We keep the notations in the introduction. Additionally, for a partition $\lambda = (\lambda_1 \geq \lambda_2 \geq \dots)$ of n , define $d_\lambda := n - \lambda_1$, called the *depth* of λ . The following theorem suggests a mild restriction for (λ, μ, ν) to be considered.

Theorem 2.1 ([29, Theorem 2.9.22]). *If $g_{\lambda,\mu}^\nu \neq 0$ then*

$$|d_\lambda - d_\mu| \leq d_\nu \leq d_\lambda + d_\mu. \quad (2.1)$$

The inequalities in (2.1) provide necessary conditions for $g_{\lambda,\mu}^\nu \neq 0$ and the converse is not true. Actually, when $n = 12$, there are 456,533 triples (λ, μ, ν) in total and 406,919 of them satisfy (2.1). Among them, 280,009 triples give $g_{\lambda,\mu}^\nu \neq 0$ and the remaining 126,910 triples $g_{\lambda,\mu}^\nu = 0$. We will consider classification problems for datasets satisfying (2.1).

There are symmetries among $g_{\lambda,\mu}^\nu$ as stated in the lemma below, which will be explicitly taken into account when we make some datasets.

Lemma 2.2 ([30, 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*

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

Now, 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 adding 0-entries as many as needed. For example, when $n = 5$, we have

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

Let $\mathcal{Q}(n)$ be the set of triples $(\lambda, \mu, \nu) \in \mathcal{P}(n)^3$ that satisfies (2.1). For each triple $(\lambda, \mu, \nu) \in \mathcal{Q}(n)$, we form the following data points:

(1) a 1d-array of size $1 \times 3n$ given by concatenation:

$$v_1(\lambda, \mu, \nu) = [\lambda, \mu, \nu].$$

(2) a 2d-array of size $n \times 3$ given by

$$v_2(\lambda, \mu, \nu) = [[\lambda], [\mu], [\nu]],$$

where $[\lambda], [\mu], [\nu]$ are column vectors.

(3) a 3d-array of size $6 \times n \times 3$ consisting of the permutations of the triple:

$$v_3(\lambda, \mu, \nu) = [[[\lambda], [\mu], [\nu]], [[\lambda], [\nu], [\mu]], [[\mu], [\lambda], [\nu]], \\ [[\mu], [\nu], [\lambda]], [[\nu], [\lambda], [\mu]], [[\nu], [\mu], [\lambda]]].$$

The array $v_3(\lambda, \mu, \nu)$ is defined to reflect Lemma 2.2 into the dataset.

Define a class-valued function $t(\lambda, \mu, \nu)$ on $\mathcal{Q}(n)$ by

$$t(\lambda, \mu, \nu) = \begin{cases} 0 & \text{if } g_{\lambda, \mu}^{\nu} = 0, \\ 1 & \text{if } g_{\lambda, \mu}^{\nu} \neq 0. \end{cases}$$

Finally, we have the following labeled datasets:

$$\mathcal{D}_n^{(a)} = \{v_a(\lambda, \mu, \nu) \rightarrow t(\lambda, \mu, \nu) : (\lambda, \mu, \nu) \in \mathcal{Q}(n)\} \quad \text{for } a = 1, 2, 3. \quad (2.2)$$

2.2. *Experimental strategy*

We perform our experiments following the steps explained below.

- (1) Choose n . In this paper, we mainly consider $12 \leq n \leq 14$. When $n = 14$, we have $\#(\mathcal{Q}(14)) = 2, 258, 526$, which makes the size of dataset reasonably big.
- (2) Determine an ML classifier. We mostly use Nearest Neighbors (NearN), Convolutional Neural Networks (CNNs) and LightGBM (LGBM).
- (3) Generate datasets of the form $\mathcal{D}_n^{(a)}$ as in (2.2) using SAGEMATH. According to the choice of an ML classifier, we use $a = 1$ for NearN and LGBM and $a = 2, 3$ for CNNs.
- (4) Form a balanced dataset \mathcal{D} out of $\mathcal{D}_n^{(a)}$ by sampling the same number of data points from each class $t(\lambda, \mu, \nu) = i$ for $i = 0, 1$. Choose a subset $\mathcal{T} \subset \mathcal{D}$ and denote its complement by $\mathcal{V} = \mathcal{D} - \mathcal{T}$. We will refer to \mathcal{T} as the training dataset, and \mathcal{V} as the validation dataset. We will typically choose 70% of data points for \mathcal{T} and 30% for \mathcal{V} .
- (5) Train the classifier in step (2) on the set \mathcal{T} . We implement the algorithms using MATHEMATICA [31], TENSORFLOW [32] and LIGHTGBM [25].
- (6) For all triples (λ, μ, ν) in \mathcal{V} , ask the classifier to determine $t(\lambda, \mu, \nu)$, that is, whether $g_{\lambda, \mu}^{\nu} = 0$ or not. We record the precision and confusion matrix, which together constitute a good measure of performance of the classifier.
- (7) Repeat steps (4)–(6) for different choices of \mathcal{T} . The precision values and confusion matrices recorded below are representative of several repetitions.
- (8) Repeat steps (1)–(7) for different choices of n, a and an ML classifier.

3. Results

In this section, we describe our experimental results for the binary classification: $g_{\lambda, \mu}^{\nu} = 0$ or $\neq 0$.

Table 1. The above table shows the precisions of NearN, CNN₂, CNN₃ and LGBM classifiers when asked to distinguish (λ, μ, ν) with $g_{\lambda, \mu}^{\nu} = 0$ from those with $g_{\lambda, \mu}^{\nu} \neq 0$.

n	# \mathcal{D}	Precision			
		NearN	CNN ₂	CNN ₃	LGBM
12	$126,900 \times 2$	0.9155	0.9529	0.9697	0.9714
13	$260,000 \times 2$	0.9318	0.9618	0.9773	0.9837
14	$600,000 \times 2$	0.9364	0.9635	0.9772	0.9845

Table 2. The above table shows the confusion matrices of NearN, CNN₂, CNN₃ and LGBM classifiers in the experiments recorded in Table 1. Here class 0 represents the case $g_{\lambda, \mu}^{\nu} = 0$ and class 1 the case $g_{\lambda, \mu}^{\nu} \neq 0$.

n	NearN	CNN ₂	CNN ₃	LGBM
12				
13				
14				

For $n = 12, 13, 14$, we use datasets $\mathcal{D}_n^{(1)}$ to train NearN and LGBM classifiers. On the other hand, $\mathcal{D}_n^{(2)}$ and $\mathcal{D}_n^{(3)}$ are used for CNN classifiers to utilize the benefit of convolution layers. The results of CNN using $\mathcal{D}_n^{(2)}$ will be denoted by CNN₂ and those using $\mathcal{D}_n^{(3)}$ by CNN₃. The results are recorded in Tables 1 and 2.

3.1. Specifics

In the architecture of CNNs, we put one $2d$ -convolution layer with 32 filters and one dense layer after flattening the outcome of the convolution layer. Though there are no significant differences in precisions for different kernel sizes of the convolution layers, the following kernel sizes perform slightly better than others: for CNN₂, kernel size $(n - 4) \times 3$ with input size $n \times 3$, and for CNN₃, kernel size $2 \times (n - 4)$ with input size $6 \times n \times 3$, where $n = 12, 13, 14$. The outcomes for CNNs in Table 1

are from these kernel sizes. The total numbers of parameters are shown below:

$$\text{CNN}_2: (96 \times n - 30), \quad \text{CNN}_3: (192 \times n + 866).$$

That is, 1122, 1218, 1314 for CNN_2 and 3170, 3362, 3554 for CNN_3 when $n = 12, 13, 14$, respectively.

The hyper-parameters of LightGBM used in our experiments are below:

```
'metric': 'auc',           'boosting': 'gbdt',       'num_leaves': 63,
'feature_fraction': 0.5,   'bagging_fraction': 0.5,
'bagging_freq': 20,       'learning_rate': 0.01
```

3.2. Observations

Let us make some observations from the results recorded in Tables 1 and 2.

- (1) The precisions of different ML classifiers can be compared as

$$\text{NearN} < \text{CNN}_2 < \text{CNN}_3 < \text{LGBM}.$$

When $n = 14$, LGBM achieved precision 0.9845.

- (2) The classifiers generally perform better as n increases. However, the difference between $n = 13$ and $n = 14$ is much less, compared to that between $n = 12$ and $n = 13$. It seems that the precision is being stabilized as n grows.
- (3) In the confusion matrix one can see that class 1 ($g'_{\lambda,\mu} \neq 0$) is more often misclassified than class 0 ($g'_{\lambda,\mu} = 0$). However, though rarely, the opposite may occur depending on samples. For example, see $n = 13$ of CNN_3 in Table 2.

4. Conclusions and Outlook

The experimental results in this paper show that an ML classifier can be trained to distinguish nonzero Kronecker coefficients from zero ones with high accuracy. This provides somewhat convincing evidence that ML can be used to recognize structures in Kronecker coefficients, adding another example to new opportunities of developing ML techniques for studying mathematical structures.

This paper focuses on the binary classification of $g'_{\lambda,\mu} = 0$ or $g'_{\lambda,\mu} > 0$. One can perform several variations of this experiment. For example, multi-class classification of $g'_{\lambda,\mu} = 0, 1, \dots, K$ or $g'_{\lambda,\mu} > K$ can be considered for some $K \geq 1$. A different direction is to consider *induction* and *reduction* with respect to the inclusions $\mathfrak{S}_n \subset \mathfrak{S}_N$ for $n < N$. That is, we can train a machine with \mathfrak{S}_N and apply it to \mathfrak{S}_n , and vice versa. Yet another direction is to investigate the datasets using dimensionality reduction techniques such as principal component analysis (PCA) or t-distributed stochastic neighborhood embedding (t-SNE) [33].

With all these experimental results at hand, the next step would be to perform attribute analysis to understand what ML classifiers actually recognize in the datasets. This may lead to some conjectures on how to combinatorially determine

whether a Kronecker coefficient is zero or not, or how to combinatorially describe Kronecker coefficients. Hopefully, it may be that ML can even provide some hints about how to prove the conjectures.

Acknowledgments

We are grateful to Greta Panova for helpful comments. This work was partially supported by a grant from the Simons Foundation (#712100).

ORCID

Kyu-Hwan Lee  <https://orcid.org/0009-0001-2387-7089>

References

- [1] D. Murnaghan, The analysis of the Direct product of irreducible representations of the symmetric groups, *Am. J. Math.* **60** (1938) 44–65.
- [2] J. B. Remmel, A formula for the Kronecker products of Schur functions of hook shapes, *J. Algebra* **120** (1989) 100–118.
- [3] C. M. Ballantine and R. C. Orellana, A combinatorial interpretation for the coefficients in the Kronecker product $s_{(n-p,p)} * s_\lambda$, *Sém. Lothar. Combin.* **54A** (2006) Art. B54Af.
- [4] J. B. Remmel and T. Whitehead, On the Kronecker product of Schur functions of two row shapes, *Bull. Belg. Math. Soc. Simon Stevin* **1** (1994) 649–683.
- [5] J. Blasiak, K. D. Mulmuley and M. Sohoni, Geometric complexity theory IV: Non-standard quantum group for the Kronecker problem, *Mem. Amer. Math. Soc.* **235** (2015) 1109.
- [6] J. Blasiak, An insertion algorithm for catabolizability, *Eur. J. Combin.* **33** (2012) 267–276.
- [7] C. Ikenmeyer, K. D. Mulmuley and M. Walter, On vanishing of Kronecker coefficients, *Comput. Complex.* **26** (2017) 949–992.
- [8] I. Pak and G. Panova, Bounds on certain classes of Kronecker and q-binomial coefficients, *J. Combin. Theory Ser. A* **147** (2017) 1–17.
- [9] I. Pak and G. Panova, On the complexity of computing Kronecker coefficients, *Comput. Complexity* **26** (2017) 1–36.
- [10] Y.-H. He, Deep-learning the landscape, arXiv:1706.02714 [hep-th].
- [11] Y.-H. He, Machine-learning the string landscape, *Phys. Lett. B* **774** (2017) 564–568.
- [12] D. Krefl and R. K. Seong, Machine learning of Calabi–Yau volumes, *Phys. Rev. D* **96** (2017) 066014.
- [13] F. Ruehle, Evolving neural networks with genetic algorithms to study the string landscape, *J. High Energy Phys.* **038** (2017).
- [14] J. Carifio, J. Halverson, D. Krioukov and B. D. Nelson, Machine learning in the string landscape, *J. High Energy Phys.* **157** (2017).
- [15] Y.-H. He and M. Kim, Learning algebraic structures: Preliminary investigations, arXiv:1905.02263 [cs.LG].
- [16] Y.-H. He and S.-T. Yau, Graph Laplacians, Riemannian manifolds and their machine-learning, arXiv:2006.16619 [math.CO].
- [17] A. Z. Wagner, Constructions in combinatorics via neural networks, preprint, arXiv:2104.14516.

- [18] A. Ashmore, Y.-H. He and B. A. Ovrut, Machine learning Calabi–Yau metrics, arXiv:1910.08605 [hep-th].
- [19] M. Douglas, S. Lakshminarasimhan and Y. Qi, Numerical Calabi–Yau metrics from holomorphic networks, *Proc. Mach. Learn. Res.* **145** (2022) 223–252.
- [20] V. Jejjala, A. Kar and O. Parrikar, Deep learning the hyperbolic volume of a knot, *Phys. Lett. B* **799** (2019) 135033.
- [21] Y.-H. He, K.-H. Lee and T. Oliver, Machine-learning the Sato–Tate conjecture, *J. Symb. Comput.* **111** (2022) 61–72.
- [22] Y.-H. He, K.-H. Lee and T. Oliver, Machine-learning number fields, *Math. Comput. Geometr. Data* **2** (2022) 49–66.
- [23] Y.-H. He, K.-H. Lee and T. Oliver, Machine learning invariants of arithmetic curves, *J. Symb. Comput.* **115** (2023) 478–491.
- [24] A. Davies, P. Veličković, L. Buesing, S. Blackwell, D. Zheng, N. Tomašev, R. Tanburn, P. Battaglia, C. Blundell, A. Juhász, M. Lackenby, G. Williamson, D. Hassabis and P. Kohli, Advancing mathematics by guiding human intuition with AI, *Nature* **600** (2021) 70–74.
- [25] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye and T.-Y. Liu, Lightgbm: A highly efficient gradient boosting decision tree, *Adv. Neural Inf. Process. Syst.* **30** (2017) 3146–3154.
- [26] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning — Adaptive Computation and Machine Learning* (MIT Press, 2016).
- [27] T. Hastie, R. Tibshirani and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (Springer, NY, 2001).
- [28] The Sage Development Team, *SageMath, the Sage Mathematics Software System (Version 9.1.0)* (2020), <http://www.sagemath.org>.
- [29] G. James and A. Kerber, *The Representation Theory of the Symmetric Group*, Encyclopedia of Mathematics and Its Applications, Vol. 16 (Addison-Wesley Publishing Co., Reading, Mass., 1981).
- [30] W. Fulton and J. Harris, *Representation theory*, Graduate Texts in Mathematics, Vol. 129 (Springer-Verlag, New York, 1991).
- [31] Wolfram Research, Inc., *Mathematica 12.1*, Champaign, Illinois (2020), <https://www.wolfram.com/mathematica>.
- [32] M. Abadi *et al.*, TensorFlow: Large-scale machine learning on heterogeneous systems (2015), <https://www.tensorflow.org>.
- [33] L. van der Maaten and G. Hinton, Visualizing data using t-sne, *J. Mach. Learn. Res.* **9** (2008) 2579–2605.