Gröbner basis computation via learning

Hiroshi Kera[1,*], Yuki Ishihara[2], Tristan Vaccon[3] and Kazuhiro Yokoyama[4]

[1]*Chiba University, 1-33 Yayoi-cho, Inage-ku, Chiba-shi, Chiba, 2638522, Japan*
[2]*Nihon University, 1-8-14 Kanda Surugadai, Chiyoda-ku, Tokyo, 1018308, Japan*
[3]*Université de Limoges; CNRS, XLIM, UMR 7252, Limoges, France*
[4]*Rikkyo University, 3-34-1, Nishi-Ikebukuro, Toshima-ku, Tokyo, 1718501, Japan*

Abstract

Solving a polynomial system, or computing an associated Gröbner basis, has been a fundamental task in computational algebra. However, it is also known for its notorious doubly exponential time complexity in the number of variables in the worst case. This paper is the first to address the learning of Gröbner basis computation with Transformers. The training requires many pairs of a polynomial system and the associated Gröbner basis, raising two novel algebraic problems: random generation of Gröbner bases and transforming them into non-Gröbner ones, termed as backward Gröbner problem. We resolve these problems with 0-dimensional radical ideals, the ideals appearing in various applications. The experiments show that our dataset generation method is at least three orders of magnitude faster than a naive approach, overcoming a crucial challenge in learning to compute Gröbner bases, and Gröbner computation is learnable in a particular class.

Keywords

Gröbner Bases, Machine Learning, Transformer

1. Introduction

Understanding the properties of polynomial systems and solving them have been a fundamental problem in computational algebra and algebraic geometry with vast applications in cryptography [1, 2], control theory [3], statistics [4, 5], computer vision [6], systems biology [7], and so forth. Special sets of polynomials called Gröbner bases [8] play a key role to this end. In linear algebra, the Gaussian elimination simplifies or solves a system of linear equations by transforming its coefficient matrix into the reduced row echelon form. Similarly, a Gröbner basis can be regarded as a reduced form of a given polynomial system, and its computation is a generalization of the Gaussian elimination to general polynomial systems. However, computing a Gröbner basis is known for its notoriously bad computational cost in theory and practice. It is an NP-hard problem with the doubly exponential worst-case time complexity in the number of variables [9, 10]. Nevertheless, because of its importance, various algorithms have been proposed in computational algebra to obtain Gröbner bases in better runtime. Examples include Faugère's F4/F5 algorithms [11, 12] and M4GB [13].

In this study, we investigate Gröbner basis computation from a learning perspective, envisioning it as a practical compromise to address large-scale polynomial system solving and understanding, where mathematical algorithms are computationally intractable. The learning approach does not require explicit design of computational procedures, and we only need to train a model using a large amount of (non-Gröbner set, Gröbner basis) pairs. Further, if we restrict ourselves to a particular class of Gröbner bases (or associated *ideals*), the model may internally find some patterns useful for prediction. The success of learning indicates the existence of such patterns, which encourages the improvement of mathematical algorithms and heuristics. Several recent studies have already addressed

SCSS 2024: 10th International Symposium on Symbolic Computation in Software Science, August 28–30, 2024, Tokyo, Japan

[*] Corresponding author.

kera@chiba-u.jp (H. Kera); ishihara.yuki@nihon-u.ac.jp (Y. Ishihara); tristan.vaccon@unilim.fr (T. Vaccon); kazuhiro@rikkyo.ac.jp (K. Yokoyama)
https://hkera.wordpress.com (H. Kera); https://researchmap.jp/yishihara (Y. Ishihara); https://www.unilim.fr/pages_perso/tristan.vaccon/ (T. Vaccon)
0000-0002-9830-0436 (H. Kera); 0000-0003-4057-3703 (Y. Ishihara); 0000-0003-4208-8349 (T. Vaccon); 0000-0002-5072-7799 (K. Yokoyama)

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

mathematical tasks via learning, particularly using Transformers [14, 15, 16]. For example, [14] showed that Transformers can learn symbolic integration simply by observing many $(\mathrm{d}f/\mathrm{d}x\,, f)$ pairs in training. The training samples are generated by first randomly generating f and computing its derivative $\mathrm{d}f/\mathrm{d}x$ and/or by the reverse process.

However, a crucial challenge in the learning of Gröbner basis computation is that it is mathematically unknown how to efficiently generate many (non-Gröbner set, Gröbner basis) pairs. We need an efficient backward approach (i.e., *solution-to-problem* computation) because, as discussed above, the forward approach (i.e., *problem-to-solution* computation) is prohibitively expensive. To this end, we frame two problems: (i) a random generation of Gröbner bases and (ii) a backward transformation from a Gröbner basis to an associated non-Gröbner set. To our knowledge, neither of them has been addressed in the study of Gröbner bases because of the lack of motivations; all the efforts have been dedicated to the forward computation from a non-Gröbner set to Gröbner basis.

Tackling aforementioned two unexplored algebraic problems, we investigates the first learning approach to the Gröbner computation using Transformers and experimentally show its learnability uncovered two unexplored algebraic problems in the 0-dimensional case. Our experiments show that the proposed dataset generation is highly efficient and faster than a baseline method by three or four orders of magnitude. Further, we observe a learnability gap between polynomials on finite fields and infinite fields while predicting polynomial supports are more tractable. Full version of this paper can be found in [17].

2. New algebraic problems for dataset generation

Our notations and definitions follow [18] except that we call power products of indeterminate terms instead of monomials. By Gröbner basis computation, we mean computation of reduced Gröbner bases. Our goal is to realize Gröbner basis computation through learning. To this end, we need a large training set $\{(F_i, G_i)\}_{i=1}^m$ with finite polynomial set $F_i \subset k[x_1, \dots, x_n]$ and Gröbner basis G_i of ideal $\langle F_i \rangle$. As the computation from F_i to G_i is computationally expensive in general, we instead resort to *backward generation* (i.e., solution-to-problem process); that is, we generate a Gröbner basis G_i randomly and transform it to non-Gröbner set F_i.

Problem 2.1 (Random generation of Gröbner bases)**.** *Find a collection $\mathscr{G} = \{G_i\}_{i=1}^m$ with the reduced Gröbner basis $G_i \subset k[x_1, \dots, x_n]$ of $\langle G_i \rangle$, $i = 1, \dots, m$. The collection should contain diverse bases, and we need an efficient algorithm for constructing them.*

Problem 2.2 (Backward Gröbner problem)**.** *Given a Gröbner basis $G \subset k[x_1, \dots, x_n]$, find a collection $\mathscr{F} = \{F_i\}_{i=1}^\mu$ of polynomial sets that are not Gröbner bases but $\langle F_i \rangle = \langle G \rangle$ for $i = 1, \dots, \mu$. The collection should contain diverse sets, and we need an efficient algorithm for constructing them.*

Problems. 2.1 and 2.2 require the collections $\mathscr{G}, \mathscr{F}$ to contain diverse polynomial sets. Thus, the algorithms for these problems should not be deterministic but should have some controllable randomness.

What makes the learning of Gröbner basis computation hard is that, to our knowledge, neither (i) a random generation of Gröbner basis nor (ii) the backward transform from Gröbner basis to non-Gröbner set has been considered in computational algebra. Its primary interest has been instead posed on Gröbner basis computation (i.e., forward generation), and nothing motivates the random generation of Gröbner basis nor the backward transform. Interestingly, machine learning now sheds light on them. Formally, we address the following problems for dataset generation.

In this paper, we tackle these problems in the case of radical 0-dimensional ideals. We first address Prob. 2.1 using the fact that 0-dimensional radical ideals are generally *in shape position.*

Definition 2.3 (Shape position)**.** Ideal $I \subset k[x_1, \dots, x_n]$ is called in *shape position* if some univariate polynomials $h, g_1, \dots, g_{n-1} \in k[x_n]$ form the reduced $\prec_{\mathrm{lex}}$-Gröbner basis of I as follows.

$$G = \{h, x_1 - g_1, \dots, x_{n-1} - g_{n-1}\}. \tag{2.1}$$

Particularly, 0-dimensional radical ideals are almost always in shape position if k is an infinite field or finite field with large field order [19, 20]. With this fact, an efficient sampling of Gröbner bases of 0-dimensional radical ideals can be realized by sampling n polynomials in $k[x_n]$, i.e., $h, g_1, \dots, g_{n-1}$ with $h \neq 0$. We have to make sure that the degree of h is always greater than that of $g_1, \dots, g_{n-1}$, which is necessary and sufficient for G to be a reduced Gröbner basis. This approach involves efficiency and randomness, and thus resolving Prob. 2.1. To address Prob. 2.2, we consider the following problem.

Problem 2.4. *Let $I \subset k[x_1, \dots, x_n]$ be a 0-dimensional ideal, and let $G = (g_1, \dots, g_t)^\top \in k[x_1, \dots, x_n]^t$ be its $\prec$-Gröbner basis with respect to term order $\prec$.[1] Find a polynomial matrix $A \in k[x_1, \dots, x_n]^{s\times t}$ giving a non-Gröbner set $F = (f_1, \dots, f_s)^\top = AG$ such that $\langle F\rangle = \langle G\rangle$.*

Namely, we generate a set of polynomials $F = (f_1, \dots, f_s)^\top$ from $G = (g_1, \dots, g_t)^\top$ by $f_i = \sum_{j=1}^{t} a_{ij} g_j$ for $i = 1, \dots, s$, where $a_{ij} \in k[x_1, \dots, x_n]$ denotes the (i, j)-th entry of A. Note that $\langle F\rangle$ and $\langle G\rangle$ are generally not identical, and the design of A such that $\langle F\rangle = \langle G\rangle$ is of our question.

A similar question was studied without the Gröbner condition in [21, 22]. They provided an algebraic necessary and sufficient condition for the polynomial system of F to have a solution outside the variety defined by G. This condition is expressed explicitly by multivariate resultants. However, strong additional assumptions are required: A, F, G are homogeneous, G is a regular sequence, and in the end, $\langle F\rangle = \langle G\rangle$ is only satisfied up to saturation. Thus, they are not compatible with our setting and method for Prob. 2.1. Our analysis gives the following results for the design A to achieve $\langle F\rangle = \langle G\rangle$ for the 0-dimensional case.

Theorem 2.5. *Let $G = (g_1, \dots, g_t)^\top$ be a Gröbner basis of a 0-dimensional ideal in $k[x_1, \dots, x_n]$. Let $F = (f_1, \dots, f_s)^\top = AG$ with $A \in k[x_1, \dots, x_n]^{s\times t}$.*

1. *If $\langle F\rangle = \langle G\rangle$, it implies $s \geq n$.*
2. *If A has a left-inverse in $k[x_1, \dots, x_n]^{t\times s}$, $\langle F\rangle = \langle G\rangle$ holds.*
3. *The equality $\langle F\rangle = \langle G\rangle$ holds if and only if there exists a matrix $B \in k[x_1, \dots, x_n]^{t\times s}$ such that each row of $BA - E_t$ is a syzygy of G, where E_t is the identity matrix of size t.*

We now assume $\prec = \prec_{\text{lex}}$ and 0-dimensional ideals in shape position. Then, G has exactly n generators. When $s = n$, we have the following.

Proposition 2.6. *For any $A \in k[x_1, \dots, x_n]^{n\times n}$ with $\det(A) \in k \setminus \{0\}$, we have $\langle F\rangle = \langle G\rangle$.*

As non-zero constant scaling does not change the ideal, we focus on A with $\det(A) = \pm 1$ without loss of generality. Such A can be constructed using the Bruhat decomposition $A = U_1 P U_2$, where $U_1, U_2 \in \mathrm{ST}(n, k[x_1, \dots, x_n])$ are upper-triangular matrices with all-one diagonal entries (i.e., unimodular upper-triangular matrices) and $P \in \{0, 1\}^{n\times n}$ denotes a permutation matrix. Noting that A^{-1} satisfies $A^{-1}A = E_n$, we have $\langle AG\rangle = \langle G\rangle$ from Thm. 2.5. Therefore, random sampling (U_1, U_2, P) of unimodular upper-triangular matrices U_1, U_2 and a permutation matrix P resolves the backward Gröbner problem for $s = n$. We extend this idea to the case of $s > n$ using a rectangular unimodular upper-triangular matrix $U_2 = \begin{pmatrix} U_2' \\ O_{s-n,n} \end{pmatrix} \in k[x_1, \dots, x_n]^{s\times n}$, where $U_2' \in k[x_1, \dots, x_n]^{n\times n}$ is a unimodular upper-triangular matrix and $O_{s-n,n} \in k[x_1, \dots, x_n]^{(s-n)\times n}$ is the zero matrix. The permutation matrix is now $P \in \{0, 1\}^{s\times s}$. Our strategy is to compute $F = U_1 P U_2 G$, which only requires a sampling of $\mathcal{O}(s^2)$ polynomials in $k[x_1, \dots, x_n]$, and $\mathcal{O}(n^2 + s^2)$-times multiplications of polynomials.

3. Experiments

We present the efficiency of our dataset generation method and the learnability of Gröbner basis computation. The experiments were conducted with 48-core CPUs, 768GB RAM, and NVIDIA RTX A6000ada GPUs. Due to the space limitation, we cannot present full experimental setup. See the full version in [17].

[1] We surcharge notations to mean that the set $\{g_1, \dots, g_t\}$ defined by the vector G is a $\prec$-Gröbner basis.

Table 1
Runtime comparison (in seconds) of forward generation (F.) and backward generation (B.) of dataset $\mathscr{D}_n(\mathbb{F}_7)$ of size 1,000. The forward generation used either of the three algorithms provided in SageMath with the libSingular backend. We set a timeout limit to five seconds (added to the total runtime at every occurrence) for each Gröbner basis computation. The numbers with † and ‡ include the timeout for more than 13 % and 24 % of the runs, respectively.

Method	$n = 2$	$n = 3$	$n = 4$	$n = 5$
F. (STD)	4.65	129	873†	1354‡
F. (SLIMGB)	4.67	149	712†	1259‡
F. (STDFGLM)	5.78	12.6	44.2	360
B. (ours)	**.003**	**.005**	**.009**	**.014**

Dataset generation. We constructed 12 datasets $\mathscr{D}_n(k)$ for $n \in \{2, 3, 4, 5\}$ and $k \in \{\mathbb{F}_7, \mathbb{F}_{31}, \mathbb{Q}\}$ and measured the runtime of our backward generation and naive forward generation (i.e., Gröbner basis computation). In the backward generation, we sampled Gröbner bases of ideals in shape position. In this step, univariable polynomials were generically sampled in $k[x_1, \ldots, x_n]_{\leq 5}$. Next, Gröbner bases were transformed to non-Gröbner sets based on Thm. 2.5. Random polynomials in Bruhat decomposition (i.e., U_1 and U_2') were sampled from $k[x_1, \ldots, x_n]_{\leq 3}$ and restricted to monomials and binomials. For $\mathbb{Q}$, coefficients of all sampled polynomials were bounded as a/b with $a, b \in \{-5, \ldots, 5\}$ and we only accepted F with coefficients such as $a, b \in \{-100, \ldots, 100\}$. This restriction is required from our machine learning model and learning framework. For forward generation, we adopted three algorithms given by SageMath [23] with the libSingular backend. For a fair comparison, forward generation computed Gröbner bases of the non-Gröbner sets given by the backward generation, leading to the identical dataset. As Tab. 1 shows, our backward generation is significant orders of magnitude faster than the forward generation. A sharp runtime growth is observed in the forward generation as the number of variables increases. Note that these numbers only show the runtime on 1,000 samples, while training typically requires millions of samples. Therefore, the forward generation is almost infeasible, and the proposed method resolves a bottleneck in the learning of Gröbner basis computation.

Learning results. We used a standard Transformer (e.g., 6 encoder/decoder layers and 8 attention heads) and a standard training setup. The batch size was set to 16, and models were trained for 8 epochs. Each polynomial set in the datasets is converted into a sequence using the prefix representation and the separator tokens. To make the input sequence length manageable for vanilla Transformers, we used simpler datasets $\mathscr{D}_n^-(k)$ using U_1, U_2' of a moderate density $\sigma \in (0, 1]$. This makes the maximum sequence length less than 5,000. Specifically, we used $\sigma = 1.0, 0.6, 0.3, 0.2$ for $n = 2, 3, 4, 5$, respectively. The training set has one million samples, and the test set has one thousand samples. Table 2 shows that trained Transformers successfully compute Gröbner bases with moderate/high accuracy. Not shown here, but we found several examples in the datasets for which Transformer successfully compute Gröbner bases significantly faster than math algorithms. The accuracy shows that the learning is more successful on infinite field coefficients $k \in \{\mathbb{Q}, \mathbb{R}\}$ than finite field ones $k = \mathbb{F}_p$. This may be a counter-intuitive observation because there are more possible coefficients in G and F for $\mathbb{Q}$ than $\mathbb{F}_p$. Specifically, for G, the coefficient $a/b \in \mathbb{Q}$ is restricted to those with $a, b \in \{-5, \ldots, 5\}$ (i.e., roughly 50 choices), and $a, b \in \{-100, \ldots, 100\}$ (i.e., roughly 20,000 choices) for F. In contrast, there are only p choices for $\mathbb{F}_p$. The performance even degrades for the larger order $p = 31$. Interestingly, the support accuracy shows that the terms forming the polynomial (i.e., the *support* of polynomial) are correctly identified well. Thus, Transformers have difficulty determining the coefficients in finite fields. Several studies have also reported that learning to solve a problem involving modular arithmetic may encounter some difficulties [24, 25, 26].

Table 2

Accuracy [%] / support accuracy [%] of Gröbner basis computation by Transformer on $\mathscr{D}_n^-(k)$. In the support accuracy, two polynomials are considered identical if they consist of an identical set of terms (i.e., identical *support*), Note that the datasets for $n = 3, 4, 5$ are here constructed using U_1, U_2' with density $\sigma = 0.6, 0.3, 0.2$, respectively.

Ring	$n = 2, \sigma = 1$	$n = 3, \sigma = 0.6$	$n = 4, \sigma = 0.3$	$n = 5, \sigma = 0.2$
$\mathbb{Q}[x_1, \dots, x_n]$	94.6 / 97.9	96.1 / 98.6	96.2 / 98.6	91.8 / 97.9
$\mathbb{F}_7[x_1, \dots, x_n]$	66.6 / 76.6	78.8 / 87.6	80.9 / 91.1	83.2 / 91.4
$\mathbb{F}_{31}[x_1, \dots, x_n]$	44.7 / 82.7	58.5 / 89.3	73.9 / 93.9	80.0 / 93.4

4. Conclusion

This study proposed the first learning approach to a fundamental algebraic task, the Gröbner basis computation. While various recent studies have reported the learnability of mathematical problems by Transformers, we addressed the first problem with nontriviality in the dataset generation. Ultimately, the learning approach may be useful to address large-scale problems that cannot be approached by Gröbner basis computation algorithms because of their computational complexity. Transformers can output predictions in moderate runtime. The outputs may be incorrect, but there is a chance of obtaining a hint of a solution, as shown in our experiments. We believe that our study reveals many interesting open questions to achieve Gröbner basis computation learning.

Acknowledgments

This research was supported by JST ACT-X Grant Number JPMJAX23C8 and JSPS KAKENHI Grant Number JP22K13901. Yuta Kambe (Mitsubishi Electric Information Technology R&D Center) is not included in the authors due to a technical reason at submission.

References

[1] G. V. Bard, Algorithms for Solving Polynomial Systems, Springer US, 2009.

[2] T. Yasuda, X. Dahan, Y.-J. Huang, T. Takagi, K. Sakurai, MQ challenge: hardness evaluation of solving multivariate quadratic problems, Cryptology ePrint Archive (2015).

[3] H. Park, G. Regensburger (Eds.), Gröbner Bases in Control Theory and Signal Processing, De Gruyter, 2007.

[4] P. Diaconis, B. Sturmfels, Algebraic algorithms for sampling from conditional distributions, The Annals of Statistics 26 (1998) 363 – 397.

[5] T. Hibi, Gröbner bases. Statistics and software systems., Springer Tokyo, 2014.

[6] H. Stewenius, Gröbner Basis Methods for Minimal Problems in Computer Vision, Ph.D. thesis, Mathematics (Faculty of Engineering), 2005.

[7] R. Laubenbacher, B. Sturmfels, Computer algebra in systems biology, American Mathematical Monthly 116 (2009) 882–891.

[8] B. Buchberger, Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal (An Algorithm for Finding the Basis Elements in the Residue Class Ring Modulo a Zero Dimensional Polynomial Ideal), Ph.D. thesis, Mathematical Institute, University of Innsbruck, Austria, 1965. English translation in J. of Symbolic Computation, Special Issue on Logic, Mathematics, and Computer Science: Interactions. Vol. 41, Number 3-4, Pages 475–511, 2006.

[9] E. W. Mayr, A. R. Meyer, The complexity of the word problems for commutative semigroups and polynomial ideals, Advances in Mathematics 46 (1982) 305–329.

[10] T. W. Dubé, The structure of polynomial ideals and Gröbner bases, SIAM Journal on Computing 19 (1990) 750–773.

[11] J.-C. Faugère, A new efficient algorithm for computing Gröbner bases (F4), Journal of Pure and Applied Algebra 139 (1999) 61–88.

[12] J.-C. Faugère, A new efficient algorithm for computing Gröbner bases without reduction to zero (F5), in: Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation, ISSAC '02, Association for Computing Machinery, New York, NY, USA, 2002, p. 75–83.

[13] R. H. Makarim, M. Stevens, M4GB: An efficient Gröbner-basis algorithm, in: Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation, ISSAC'17, Association for Computing Machinery, New York, NY, USA, 2017, pp. 293–300.

[14] G. Lample, F. Charton, Deep learning for symbolic mathematics, in: International Conference on Learning Representations, 2020. URL: https://openreview.net/forum?id=S1eZYeHFDS.

[15] L. Biggio, T. Bendinelli, A. Neitz, A. Lucchi, G. Parascandolo, Neural symbolic regression that scales, in: Proceedings of the 38th International Conference on Machine Learning, volume 139, 2021, pp. 936–945.

[16] F. Charton, Linear algebra with transformers, Transactions on Machine Learning Research (2022).

[17] H. Kera, Y. Ishihara, Y. Kambe, T. Vaccon, K. Yokoyama, Learning to compute gröbner bases, 2024. arXiv:2311.12904.

[18] D. A. Cox, J. Little, D. O'Shea, Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra, Undergraduate Texts in Mathematics, Springer International Publishing, 2015.

[19] P. Gianni, T. Mora, Algebraic solution of systems of polynomial equations using Groebner bases, in: Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, Springer Berlin Heidelberg, Berlin, Heidelberg, 1989, pp. 247–257.

[20] M. Noro, K. Yokoyama, A modular method to compute the rational univariate representation of zero-dimensional ideals, Journal of Symbolic Computation 28 (1999) 243–263.

[21] L. Busé, M. Elkadi, B. Mourrain, Resultant over the residual of a complete intersection, Journal of Pure and Applied Algebra 164 (2001) 35–57.

[22] L. Busé, Étude du résultant sur une variété algébrique, Theses, Université Nice Sophia Antipolis, 2001.

[23] The Sage Developers, SageMath, the Sage Mathematics Software System (Version 10.0), 2023. https://www.sagemath.org.

[24] A. Power, Y. Burda, H. Edwards, I. Babuschkin, V. Misra, Grokking: Generalization beyond overfitting on small algorithmic datasets, arXiv abs/2201.02177 (2022).

[25] F. Charton, Can transformers learn the greatest common divisor?, arXiv abs/2308.15594 (2023).

[26] A. Gromov, Grokking modular arithmetic, arXiv abs/2301.02679 (2023).