Data Augmentation for Mathematical Objects

Tereso del Río[1], Matthew England[1]

[1]Coventry University, UK

Abstract

This paper discusses and evaluates ideas of data balancing and data augmentation in the context of mathematical objects: an important topic for both the symbolic computation and satisfiability checking communities, when they are making use of machine learning techniques to optimise their tools. We consider a dataset of non-linear polynomial problems and the problem of selecting a variable ordering for cylindrical algebraic decomposition to tackle these with. By swapping the variable names in already labelled problems, we generate new problem instances that do not require any further labelling when viewing the selection as a classification problem. We find this augmentation increases the accuracy of ML models by 63% on average. We study what part of this improvement is due to the balancing of the dataset and what is achieved thanks to further increasing the size of the dataset, concluding that both have a very significant effect. We finish the paper by reflecting on how this idea could be applied in other uses of machine learning in mathematics.

Keywords

Machine Learning, Data Balancing, Data Augmentation, Cylindrical Algebraic Decomposition

1. Introduction

1.1. Machine learning and cylindrical algebraic decomposition

Cylindrical Algebraic Decomposition (CAD) is an algorithm which, given a set of polynomials, decomposes the space in which they are defined into regions in which they are sign-invariant [1]. CAD has many potential applications, however, its theoretical and practical complexity is doubly exponential [2], reducing the scope of its use in practice. In recent years, CAD has been a central component of the collaboration between the Symbolic Computation and Satisfiability Checking communities which meet in this SC^2 workshop. For example, there have been adaptions of CAD for use as an SMT theory solver [3], a repackaging of CAD theory into new algorithms better suited for satisfiability (namely cylindrical algebraic coverings [4] and the use of CAD in the model constructing calculus [5]), and the NuCAD algorithm which uses some of these ideas to tackle more general quantifier elimination problems [6].

CAD requires a declared variable ordering. In the satisfiability context, the variable ordering is unspecified (any may be chosen to gain a correct result), and in the quantifier elimination context, there is freedom within quantifier blocks (as swapping the order of quantified variables

8th International Workshop on Satisfiability Checking and Symbolic Computation, July 28, 2023, Tromsø, Norway, Collocated with ISSAC 2023

delriot@coventry.ac.uk (T. d. Río); Matthew.England@coventry.ac.uk (M. England)
https://sites.google.com/view/tereso (T. d. Río); https://matthewengland.coventry.domains (M. England)
0000-0003-3769-5478 (T. d. Río); 0000-0001-5729-3420 (M. England)

changes the meaning only if the quantifiers are different. These choices of variable ordering may not affect the correctness of the end result but they can have a huge impact on the resources required by these algorithms. In fact, Brown and Davenport found in [2] that there are a family of problems for which in the worst ordering the complexity grows doubly exponentially $(2, 4, 16, 256, 4294967296, \dots)$ while another ordering has a constant complexity.

Since the community realised the importance of variable ordering, various human-made heuristics have been developed for the choice, e.g. [7], [8], [9], [10], [11]. There have also been some experiments with dynamical variable orderings in the satisfiability context [12]. None of these heuristics is perfect; all have room for improvement in their choices. This led to a new strain of research which applied Machine Learning (ML) models to make the choice: first in selecting which human-made heuristic to follow [13], and later selecting the ordering directly [14], [15], [16], [17]. While these models have demonstrated good performance, there are barriers to their use such as the lack of meaningful training data, and the unbalanced nature of such data that does exist.

This paper proposes to balance and augment the existing datasets by exploiting the arbitrary nature of the variable representations within (the variable names). We note that this idea has been independently proposed recently in the preprint [18]. The present paper makes similar findings on the benefits of augmentation as [18] and further explores how those benefits split between solving the problem of unbalanced data and increasing the data size.

1.2. Data augmentation

Data augmentation consists of generating new data instances from existing ones. It is a widely-used technique in ML more generally, where the ability to increase the dataset can help tackle over-fitting, and increase the accuracy of the resulting model. Moreover, it can be used to mitigate the biases in the dataset and to reduce the cost of labelling [19].

Data augmentation is commonly used to generate new images in Computer Vision ML applications. Let us take these ideas used in computer vision as an analogy for generating new mathematical objects. For example, it is clear to any human that a picture of an arrow pointing to the right that is rotated 90 degrees clockwise results gives a picture of an arrow pointing downwards. This can be very useful, imagine that your dataset contains 268 images: 4 images of arrows pointing downwards, 35 pointing left, 56 pointing upwards, and 173 pointing to the right. This dataset is very unbalanced, and any model trained on it would likely have a bias towards predicting that the arrow points to the right and against predicting that the arrow points downwards. However, by simply using image rotations the dataset can be balanced to contain 67 images for each of the classes. Furthermore, since you can obtain three extra images from each of the images in the original dataset, we could actually obtain an augmented dataset of 1072 images with 268 of each class.

Returning to our mathematical context, our objects are sets of polynomials (possibly used to form polynomial constraints). For example, $\{x_1^2 - x_2, x_3^3 - 1\}$. We can determine, by computing and comparing CADs, that the optimal variable ordering to compute a CAD for this set is $x_2 \succ x_1 \succ x_3$. Now observe that simply by swapping the names of the variables x_1 and x_2 we may obtain the new set of polynomials $\{x_2^2 - x_1, x_3^3 - 1\}$, in which we know, without any further CAD computation, that the optimal variable ordering is $x_1 \succ x_2 \succ x_3$.

1.3. Plan of the paper

In this paper, we will use data augmentation to balance our initially unbalanced polynomial dataset, obtaining an improvement in the accuracy of the models. Then, we will see how much more accuracy will be improved by generating the maximum number of instances possible with the data augmentation tools we have. Section 2 outlines our methodology in creating a labelled dataset to use for ML to select a CAD variable ordering and Section 3 how we have balanced and augmented that dataset. Then in Section 4, we compare the performance of ML models trained and tested on these various datasets. We finish in Section 5 with conclusions, a comparison with some similar work in the preprint [18], and ideas for future work.

The dataset and code used to generate the datasets and results described in this paper can be found on GitHub here:

https://github.coventry.ac.uk/delriot/AugmentingMathematicalDataset

2. Creating a Dataset

There are three steps towards creating a dataset suitable for ML in our context: finding a collection of sets of meaningful polynomials, choosing a methodology to represent each of these sets to an ML model, and then a system for labelling them (identifying the best CAD ordering). We describe each of these steps in the following subsections.

2.1. Source of polynomial problems

The collection of sets of polynomials we use will be those problems in the QF_NRA collection of the SMT-LIB library [20] which involve three variables. These examples are all satisfiability problems and thus do not represent the full application range of CAD which can also address quantifier elimination. However, there are no sizeable datasets of QE problems we are aware of. The problems in the SMT-LIB do mostly emit from real applications making performance upon them meaningful. Common sources are problems include the theorem prover MetiTarski [21], attempts to prove termination of term-rewrite systems, verification conditions from Keymaera [22], and curated sets of problems from geometry [23], economics [24] and biology [25].

2.2. Representing sets of polynomials

Representing sets of polynomials for ML is not an easy task. First, their size can vary: we have already chosen to fix the number of variables but there could then still be an arbitrary number of polynomials, and each of these polynomials can have a great many different terms (although in practice each has not very many).

To represent a set of polynomials we will follow the methodology of [14] where polynomial sets are represented by a vector of real (floating point) numbered features, with those features generated algorithmically through simple operations generated in turn for each variable. For example, one feature is the sum across the polynomials of the average of the degree of x_1 across the monomials. For the set of polynomials $\{x_2^2 - x_2x_1, x_3^3x_1 - x_1^2 + 1\}$, this feature is $3/2$, as the average degree of x_1 in the first polynomial is $\frac{1}{2}$ and 1 in the second.

Ordering Name	Ordering
Ordering 0	$x_1 \succ x_2 \succ x_3$
Ordering 1	$x_1 \succ x_3 \succ x_2$
Ordering 2	$x_2 \succ x_1 \succ x_3$
Ordering 3	$x_2 \succ x_3 \succ x_1$
Ordering 4	$x_3 \succ x_1 \succ x_2$
Ordering 5	$x_3 \succ x_2 \succ x_1$

Table 1
The six possible variable orderings

As well as sum and average, the framework we use can apply the operations of maximum, sum, average, and average of non-zero terms. We also have the possibility of taking the sign at any point. Another example feature is the sum across the polynomials of the sign of the sum of the degree of x_2 across the monomials (which simplifies the number of polynomials that contain the variable x_2). For the previous set of polynomials, this feature is 1, because the sum of the degree of x_1 is 3 in the first polynomial and 0 in the second. Moreover, the degree of the variable can be substituted by sv_{x_i}, the total degree of the monomial if the monomial includes such a variable (it is 0 otherwise). E.g. sv_{x_1} is 4 for the monomial $x_1x_2x_3^2$ and 0 for the monomial $x_2^3x_3$ because x_1 does not appear in the latter. See [14] for further details.

Applying this process results in 384 features to describe a set of polynomials in three variables, of which 195 are essentially distinct (not in a linear relationship with any other feature in our dataset). We thus use these 195 features to represent a set of polynomials in 3 variables.

2.3. Labelling the sets of polynomials

In the case of sets of polynomials of three variables, there are six possible variable orderings. A CAD has been computed in Maple [26] for each ordering for every problem in our dataset, and we timed how long this took, discarding any example in which all orderings timed out (took more than 60 seconds). The label of the set of polynomials is the number associated with the ordering, as given in Table 1, whose CAD required the lowest computation time. Thus we form a labelled dataset for an ML classification problem.

3. Modifying the Dataset

The dataset described in the previous section has 1019 instances: 406 labelled 0, 93 labelled 1, 135 labelled 2, 51 labelled 3, 202 labelled 4 and 132 labelled 5. There is hence a clear imbalance in this dataset that will likely result in a bias in models trained upon it.

We split this dataset into an original testing dataset containing 20% of the instances (815) and an original training dataset containing the rest.

3.1. Balancing the dataset

We first randomly changed the label of each instance permuting the variable names in the underlying polynomials. This is done in both of the original datasets (training and testing), obtaining a balanced training dataset and a balanced testing dataset of the same sizes as the original training and testing sets.

3.2. Augmenting the dataset

However, nothing is stopping us from adding all of the six possible re-orderings for each problem to the dataset: each would have a different label which we know without any further labelling. By adding all the possibilities we obtain a perfectly balanced dataset with six times more data than the original one. The sizes of all these datasets are shown in Table 2.

Dataset	0	1	2	3	4	5	Total
`train unbalanced dataset`	326	74	105	41	163	106	815
`train balanced dataset`	126	113	149	138	144	145	815
`train augmented dataset`	815	815	815	815	815	815	4890
`test unbalanced dataset`	80	19	30	10	39	26	204
`test balanced dataset`	31	34	32	38	34	35	204
`test augmented dataset`	204	204	204	204	204	204	1224

Table 2
Number of instances of each class that each dataset has.

4. Performance of Models Trained on Different Datasets

Tables 3, 4 and 5 compare how the ML models trained using the Unbalanced, Balanced and Augmented datasets respectively perform on each of the testing datasets (columns in the tables). The best performance on each dataset (table column) is highlighted in bold. Recall that this problem is making classifications from 6 possible variable orderings. Thus a random classification would on average have an accuracy of 0.17. We see that all models do better than random, but that there are significant differences in performance.

We note that our experiments used models from the Python `sklearn` library [27]: K-Nearest Neighbours (KNN), Decision Tree (DT), Support Vector Classifier (SVC), Random Forest (RF), and Multi-Layer Perceptron (MLP). Each ML model training process followed [14] in first using cross-validation to choose the hyper-parameters of the model.

We can observe in Table 3 that the models trained with unbalanced data perform very well on unbalanced data. However, when tested in data that is been balanced (the Balanced and Augmented datasets), these models perform terribly, showing that the good results on unbalanced data occur only because both datasets are unbalanced in the same way. We note that some models were more affected by this than others (e.g. SVC had the biggest drop in performance and RF the least)

Comparing Tables 3 and 4 it is possible to observe that when testing on data that is balanced, training with a balanced dataset is an asset, in fact, the results improve by 27% on average. Performance on the balanced and augmented datasets in universally better. On the unbalanced dataset, the models trained with balanced data do not perform quite as well as those trained with unbalanced data, but a good deal of the performance is recovered.

Testing dataset	Unbalanced	Balanced	Augmented
`KNN-Unbalanced`	0.51	0.21	0.26
`DT-Unbalanced`	0.53	0.31	0.31
`SVC-Unbalanced`	0.48	0.23	0.2
`RF-Unbalanced`	**0.58**	**0.35**	**0.37**
`MLP-Unbalanced`	0.51	0.32	0.32

Table 3
Accuracy of models trained on the unbalanced dataset, when tested on the different testing datasets.

Testing dataset	Unbalanced	Balanced	Augmented
`KNN-Balanced`	0.41	0.36	0.38
`DT-Balanced`	0.43	0.45	0.45
`SVC-Balanced`	0.25	0.3	0.28
`RF-Balanced`	**0.49**	**0.52**	**0.52**
`MLP-Balanced`	0.45	0.43	0.44

Table 4
Accuracy of models trained on the balanced dataset, when tested on the different testing datasets.

Testing dataset	Unbalanced	Balanced	Augmented
`KNN-Augmented`	0.54	0.55	0.53
`DT-Augmented`	0.54	0.55	0.54
`SVC-Augmented`	0.46	0.48	0.49
`RF-Augmented`	**0.62**	**0.63**	**0.61**
`MLP-Augmented`	0.48	0.5	0.51

Table 5
Accuracy of models trained on the augmented dataset, when tested on the different testing datasets.

Comparing Tables 4 and 5 one observes that fully augmenting the dataset is superior to just balancing it for all models on any of our datasets.

Finally, comparing Tables 3 and 5 it is possible to observe that when testing on unbalanced data the improvement in performance obtained by augmenting the dataset is similar in scale to that gained by training dataset on a dataset that has the same imbalance as the testing data: three of the five models perform better on the unbalanced dataset when trained with augmented data and the other two come close. When comparing performance on a balanced testing dataset is balanced, the improvement from using balanced data or augmented data for training is significant: an increase in 63% of accuracy on average.

5. Final Thoughts

5.1. Conclusions

Our first conclusion is that, for this problem, training on an unbalanced dataset does indeed lead to overfitting and poor performance when the models are utilised on a balanced dataset. The performance in the Unbalanced column in Table 4) is much worse than previous reports on such ML models, e.g. [28], and demonstrates the importance of taking note and care of these balance issues. In general, imbalance is not always inappropriate for ML: some applications will naturally have imbalanced data and the ML models should be aware of this. However, in our case, we seek heuristics for choosing variable orderings for CAD applied in general and there is little rationale to suppose general CAD applications favour one ordering over another[1]. Thus our advice is to ensure ML models for such applications are trained on balanced data.

Our second conclusion is that a good deal of the ML performance can be recovered by simply training on balanced data, re-validating the value of the data-science-led approach to this task that those original papers posited.

Our third conclusion is that it is beneficial to go further and use maximum data augmentation: all models benefitted from this over just balancing the data no matter which dataset they are tested on. Using a balanced dataset instead of an unbalanced one of the same size allowed the accuracy of the models to improve on average by 27%. But using a dataset fully augmented to thus multiply the size by six allowed the accuracy of the models to improve on average by 63%. In fact, the performance lost from the original unbalanced case is basically recovered this way.

Finally, we note that these ideas should generalise easily to variable ordering choice for the other decision procedures of non-linear real arithmetic commonly found in the wider toolchains of the SC^2 community.

5.2. Comparison with the work of Hester et al. (2023)

Let us now compare the results on this paper with the ones obtained in the recent preprint [18]. Table 2 in [18] presents the accuracies of trained models on different datasets: note that both their 'Training Set 2' and 'Dataset 1' contain instances in which the models have been trained, meaning that 'Testing Set 2' is the most appropriate column for evaluation in that table. That column shows similar results to the ones shown in this paper.

We note that the original dataset in [18] contained 6895 instances while in our paper the initial dataset only contained 1019 instances. This is because, even though both datasets have the same ultimate source (the SMT-LIB), our dataset had been stripped of duplicate instances (those problem instances whose CAD tree structure is identical for every variable ordering), as described in detail in Section 4.1 of [11]. We view this as a necessary step to meaningful use of the QF_NRA section of the SMT-LIB where there are many *very* similar problems.

This comparison with [18] shows that the size alone of the dataset is not what matters (since [18] has similar accuracy to the models presented here despite training with much more data). Rather it is the number of qualitatively different problems within the dataset. I.e. there is little benefit to including multiple very similar problems. It may seem that data augmentation adds no new information, but since the ML models are not aware of these symmetries by exposing them with augmentation we actually give them access to this information.

5.3. Future work

Given the success of this data augmentation, an obvious area for future work is to look for additional augmentation techniques. Returning to the computer vision analogy: rotations

[1]except perhaps the existence of the SMT-LIB data!

are not the only augmentation tool, there are others also e.g. mirror reflections. Regarding mathematical objects, a corresponding augmentation technique may be substituting a variable with its negative, which would create a new instance without the need for any further labelling. We could also consider more involved variable transformations, however, these would most likely require additional CAD computations for data labelling, which is the most expensive part of this whole process.

We note that these ideas of data augmentation could be generalised to other mathematical object datasets. One should reflect on which parts of the representation of a mathematical object are arbitrary to the problem at hand. For example, in [29] the authors consider symbolic integration by ML, with mathematical expressions represented as natural text. The order of the operands in commutative operations is arbitrary (e.g. $x \wedge 2 + y * z$ is the same expression as $z * y + x \wedge 2$). This could be exploited to generate an exorbitant amount of new instances that do not require any relabelling!

Acknowledgments

TdR is supported by Coventry University and a travel grant from the London Mathematical Society (LMS). ME is supported by UKRI EPSRC Grant EP/T015748/1, *Pushing Back the Doubly-Exponential Wall of Cylindrical Algebraic Decomposition* (the DEWCAD Project).

References

[1] G. E. Collins, Quantifier elimination for real closed fields by cylindrical algebraic decomposition, in: Proceedings of the 2nd GI Conference on Automata Theory and Formal Languages, Springer-Verlag (reprinted in the collection [30]), 1975, pp. 134–183. doi:10.1007/3-540-07407-4_17.

[2] C. W. Brown, J. H. Davenport, The complexity of quantifier elimination and cylindrical algebraic decomposition, in: Proceedings of the International Symposium on Symbolic and Algebraic Computation, ISSAC, ACM, 2007, pp. 54–60. doi:10.1145/1277548.1277557.

[3] G. Kremer, E. Ábrahám, Fully incremental cylindrical algebraic decomposition, Journal of Symbolic Computation 100 (2020) 11–37. doi:10.1016/j.jsc.2019.07.018.

[4] E. Ábrahám, J. H. Davenport, M. England, G. Kremer, Deciding the consistency of non-linear real arithmetic constraints with a conflict driven search using cylindrical algebraic coverings, Journal of Logical and Algebraic Methods in Programming 119 (2021) 100633. doi:10.1016/j.jlamp.2020.100633.

[5] D. Jovanović, L. De Moura, Solving non-linear arithmetic, in: Lecture Notes in Computer Science, volume 7364, Springer, Berlin, Heidelberg, 2012, pp. 339–354. doi:10.1007/978-3-642-31365-3_27.

[6] C. W. Brown, Open non-uniform cylindrical algebraic decompositions, in: Proceedings of the International Symposium on Symbolic and Algebraic Computation, ISSAC, ACM, 2015, pp. 85–92. doi:10.1145/2755996.2756654.

[7] C. W. Brown, Companion to the ISSAC 2004 tutorial: Cylindrical algebraic decomposition, URL http://www.usna.edu/Users/cs/wcbrown/research/ISSAC04/handout.pdf, 2004.

[8] A. Dolzmann, A. Seidl, T. Sturm, Efficient projection orders for CAD, in: Proceedings of the 2004 International Symposium on Symbolic and Algebraic Computation, ISSAC, ACM, 2004, pp. 111–118. doi:10.1145/1005285.1005303.

[9] R. Bradford, J. H. Davenport, M. England, D. Wilson, Optimising problem formulation for cylindrical algebraic decomposition, in: Intelligent Computer Mathematics (CICM 2013), volume 7961 of *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, 2013, pp. 19–34. doi:10.1007/978-3-642-39320-4_2.

[10] D. Wilson, M. England, R. Bradford, J. H. Davenport, Using the distribution of cells by dimension in a cylindrical algebraic decomposition, in: 16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2014), IEEE, 2015, pp. 53–60. doi:10.1109/SYNASC.2014.15.

[11] T. del Río, M. England, New heuristic to choose a cylindrical algebraic decomposition variable ordering motivated by complexity analysis, in: F. Boulier, M. England, T. M. Sadykov, E. V. Vorozhtsov (Eds.), Computer Algebra in Scientific Computing, volume 13366 of *Lecture Notes in Computer Science*, Springer International Publishing, 2022, pp. 300–317. doi:10.1007/978-3-031-14788-3_17.

[12] J. Nalbach, G. Kremer, E. Ábrahám, On variable orderings in MCSAT for non-linear real arithmetic, in: J. Abbott, A. Griggio (Eds.), Proceedings of the 4th Workshop on Satisfiability Checking and Symbolic Computation (SC^2 2019), number 2460 in CEUR Workshop Proceedings, 2019. URL: http://ceur-ws.org/Vol-2460/.

[13] Z. Huang, M. England, D. Wilson, J. H. Davenport, L. C. Paulson, J. Bridge, Applying machine learning to the problem of choosing a heuristic to select the variable ordering for cylindrical algebraic decomposition, in: S. M. Watt, J. H. Davenport, A. P. Sexton, P. Sojka, J. Urban (Eds.), Lecture Notes in Computer Science, volume 8543, Springer Verlag, 2014, pp. 92–107. doi:10.1007/978-3-319-08434-3_8.

[14] D. Florescu, M. England, Algorithmically generating new algebraic features of polynomial systems for machine learning, in: J. Abbott, A. Griggio (Eds.), Proceedings of the 4th Workshop on Satisfiability Checking and Symbolic Computation (SC^2 2019), number 2460 in CEUR Workshop Proceedings, 2019. URL: https://ceur-ws.org/Vol-2460/.

[15] D. Florescu, M. England, Improved cross-validation for classifiers that make algorithmic choices to minimise runtime without compromising output correctness, in: D. Slamanig, E. Tsigaridas, Z. Zafeirakopoulos (Eds.), Mathematical Aspects of Computer and Information Sciences (Proc. MACIS '19), volume 11989 of *Lecture Notes in Computer Science*, Springer, 2020, pp. 341–356. doi:10.1007/978-3-030-43120-4_27.

[16] D. Florescu, M. England, A machine learning based software pipeline to pick the variable ordering for algorithms with polynomial inputs, in: A. Bigatti, J. Carette, J. H. Davenport, M. Joswig, T. de Wolff (Eds.), Mathematical Software – ICMS 2020, volume 12097 of *Lecture Notes in Computer Science*, Springer International Publishing, 2020, pp. 302–322. doi:10.1007/978-3-030-52200-1_30.

[17] C. Chen, Z. Zhu, H. Chi, Variable Ordering Selection for Cylindrical Algebraic Decomposition with Artificial Neural Networks, in: A. Bigatti, J. Carette, J. H. Davenport, M. Joswig, T. de Wolff (Eds.), Mathematical Software – ICMS 2020, volume 12097 of *Lecture Notes in Computer Science*, Springer, 2020, pp. 281–291. doi:10.1007/978-3-030-52200-1_28.

[18] J. Hester, B. Hitaj, G. Passmore, S. Owre, N. Shankar, E. Yeh, Revisiting variable ordering

for real quantifier elimination using machine learning, ArXiv Preprint (2023). doi:10.48550/ARXIV.2302.14038.

[19] C. Shorten, T. M. Khoshgoftaar, A survey on image data augmentation for deep learning, Journal of Big Data 6 (2019) 60. doi:10.1186/s40537-019-0197-0.

[20] C. Barrett, P. Fontaine, C. Tinelli, The Satisfiability Modulo Theories Library (SMT-LIB), 2016. URL: http://smtlib.cs.uiowa.edu/.

[21] L. C. Paulson, Metitarski: Past and future, in: L. Beringer, A. Felty (Eds.), Interactive Theorem Proving, volume 7406 of *Lecture Notes in Computer Science*, Springer, 2012, pp. 1–10. doi:10.1007/978-3-642-32347-8_1.

[22] A. Platzer, J. Quesel, P. Rümmer, Real world verification, in: R. A. Schmidt (Ed.), Automated Deduction (CADE-22), volume 5663 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2009, pp. 485–501. doi:10.1007/978-3-642-02959-2_35.

[23] C. Brown, Z. K., T. Reico, , R. Vajda, M. Pilar Vélez, Is computer algebra ready for conjecturing and proving geometric inequalities in the classroom?, Mathematics in Computer Science 16 (2021) 31. doi:10.1007/s11786-022-00532-9.

[24] C. B. Mulligan, J. H. Davenport, M. England, TheoryGuru: A Mathematica package to apply quantifier elimination technology to economics, in: J. H. Davenport, M. Kauers, G. Labahn, J. Urban (Eds.), Mathematical Software – Proc. ICMS 2018, volume 10931 of *Lecture Notes in Computer Science*, Springer International Publishing, 2018, pp. 369–378. doi:10.1007/978-3-319-96418-8_44.

[25] R. Bradford, J. H. Davenport, M. England, H. Errami, V. Gerdt, D. Grigoriev, C. Hoyt, M. Košta, O. Radulescu, T. Sturm, A. Weber, Identifying the parametric occurrence of multiple steady states for some biological networks, Journal of Symbolic Computation 98 (2020) 84–119. doi:10.1016/j.jsc.2019.07.008.

[26] C. Chen, M. Moreno Maza, Cylindrical algebraic decomposition in the RegularChains library, in: H. Hong, C. Yap (Eds.), Mathematical Software – ICMS 2014, volume 8592 of *Lecture Notes in Computer Science*, Springer Heidelberg, 2014, pp. 425–433. doi:10.1007/978-3-662-44199-2_65.

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830. URL: http://www.jmlr.org/papers/v12/pedregosa11a.html.

[28] M. England, D. Florescu, Comparing machine learning models to choose the variable ordering for cylindrical algebraic decomposition, in: C. Kaliszyk, E. Brady, A. Kohlhase, C. Sacerdoti Coen (Eds.), Intelligent Computer Mathematics, volume 11617 of *Lecture Notes in Computer Science*, Springer International Publishing, 2019, pp. 93–108. doi:10.1007/978-3-030-23250-4_7.

[29] G. Lample, F. Charton, Deep Learning for Symbolic Mathematics, in: S. Mohamed, M. White, K. Cho, D. Song (Eds.), Eighth International Conference on Learning Representations (ICLR 2020), 2020. URL: https://iclr.cc/virtual_2020/poster_S1eZYeHFDS.html.

[30] B. Caviness, J. Johnson, Quantifier Elimination and Cylindrical Algebraic Decomposition, Texts & Monographs in Symbolic Computation, Springer-Verlag, 1998. doi:10.1007/978-3-7091-9459-1.