Machine learning classification for variable ordering choice in Maple

Matthew England and Dorian Florescu
Coventry University, UK

2020 Maple Conference
2–6 November 2020
Online

Supported by EPSRC grant EP/R019622/1.

Outline

ML and Maple (Slide 1)

Machine Learning (ML) tools use statistics and large quantities of data to learn how to perform tasks not explicitly programmed. There are a wide variety of ML algorithms in existence.

Deep Learning refers to the use artificial neural networks for ML – responsible for many of the recent headline advances in AI.

Since 2018 Maple has a `DeepLearning` Package: a front end to the Google TensorFlow Python API. This lets you use and explore the results of deep learning from within Maple.

So ML is now available within Maple, but to date there has been very little use of ML to improve the code and symbolic algorithms of Maple, or any other computer algebra system (CAS).

ML tools can only offer probabilistic guidance, but computer algebra prizes exact results. So is there really scope to use ML to improve Maple's symbolic algorithms?

ML and Maple (Slide 1)

Machine Learning (ML) tools use statistics and large quantities of data to learn how to perform tasks not explicitly programmed. There are a wide variety of ML algorithms in existence.

Deep Learning refers to the use artificial neural networks for ML – responsible for many of the recent headline advances in AI.

Since 2018 Maple has a `DeepLearning` Package: a front end to the Google TensorFlow Python API. This lets you use and explore the results of deep learning from within Maple.

So ML is now available within Maple, but to date there has been very little use of ML to improve the code and symbolic algorithms of Maple, or any other computer algebra system (CAS).

ML tools can only offer probabilistic guidance, but computer algebra prizes exact results. So is there really scope to use ML to improve Maple's symbolic algorithms?

ML to safely improve Computer Algebra (Slide 2)

We propose to use ML techniques to make non-critical choices or guide searches in a CAS.

- Thus the performance of the ML has no affect on the mathematical correctness of the final result.
- But such choices can make a substantial contribution to both computational efficiency and presentation of the end result.

Our thesis is that ML can make such choices better than the user, developer, or currently employed human-made heuristics.

This approach differs from say the Facebook Research work of Lample and Charton (2020) which used ML to predict the end result of a mathematical process directly.

ML to safely improve Computer Algebra (Slide 2)

We propose to use ML techniques to make non-critical choices or guide searches in a CAS.

- Thus the performance of the ML has no affect on the mathematical correctness of the final result.
- But such choices can make a substantial contribution to both computational efficiency and presentation of the end result.

Our thesis is that ML can make such choices better than the user, developer, or currently employed human-made heuristics.

This approach differs from say the Facebook Research work of Lample and Charton (2020) which used ML to predict the end result of a mathematical process directly.

Inspiration from the literature (Slide 3)

- Liang, Ganesh, Poupart, Czarnecki (2016) used a machine learnt branching heuristic in their MapleSAT solver for formulae in Boolean logic.
- Alemi, Chollet, Een, Irving, Szegedy and Urban (2016) use deep neural networks for premise selection in a theorem prover.
- Gryak, Haralick and Kahrobaei (2020) used pattern matching technology to determine whether a pair of elements from a specified group are conjugate.
- Huang, England, Wilson, Davenport, Paulson, Bridge (2014) used a support vector machine to pick CAD variable ordering.
- Kuipers, Ueda, Vermaseren (2015) used a Monte-Carlo tree search to find the representation of polynomials that are most efficient to evaluate in their CAS.

See also the papers by Brown, and Chen & Zhu in ICMS 2020 AIMS Session.

Inspiration from the literature (Slide 3)

- Liang, Ganesh, Poupart, Czarnecki (2016) used a machine learnt branching heuristic in their MapleSAT solver for formulae in Boolean logic.
- Alemi, Chollet, Een, Irving, Szegedy and Urban (2016) use deep neural networks for premise selection in a theorem prover.
- Gryak, Haralick and Kahrobaei (2020) used pattern matching technology to determine whether a pair of elements from a specified group are conjugate.
- Huang, England, Wilson, Davenport, Paulson, Bridge (2014) used a support vector machine to pick CAD variable ordering.
- Kuipers, Ueda, Vermaseren (2015) used a Monte-Carlo tree search to find the representation of polynomials that are most efficient to evaluate in their CAS.

See also the papers by Brown, and Chen & Zhu in ICMS 2020 AIMS Session.

References


J.H. Liang and V. Ganesh and P. Poupart and K. Czarnecki.
Learning Rate Based Branching Heuristic for SAT Solvers. Theory and Applications of Satisfiability Testing (Proc. SAT 2016), LNCS 9710, pp. 123–140. Springer, 2016.


A.A. Alemi and F. Chollet and N. Een and G. Irving and C. Szegedy and J. Urban.
DeepMath – Deep Sequence Models for Premise Selection Proc. NIPS 2016, 2243–2251, Curran Associates Inc., 2016.


J. Gryak and R.M. Haralick and D. Kahrobaei.
Solving the Conjugacy Decision Problem via Machine Learning. Experimental Mathematics, 29(1), pp. 66–78. Taylor & Francis, 2020.


Z. Huang and M. England and D. Wilson and J.H. Davenport and L. Paulson and J. Bridge.
Applying machine learning to the problem of choosing a heuristic to select the variable ordering for cylindrical algebraic decomposition. Intelligent Computer Mathematics (Proc. CICM 2014), LNCS 8543, pp. 92–107. Springer, 2014.


J. Kuipers and T. Ueda and J.A.M. Vermaseren.
Code optimization in FORM. Computer Physics Communications, 189, pp.1–19, 2015.

Summary: Authors' Work (Slide 5)

Specifically: the authors have been experimenting on using Machine Learning (ML) to choose the variable ordering for a Maple implementation of the Cylindrical Algebraic Decomposition (CAD) algorithm acting on a set of polynomials. We have:

- Experimented with several ML classifiers in `sklearn`: they all do better than the existing human-made heuristics.
- New approach to generate features of the input polynomials.
- Proposed a more suitable measure of ML classifier accuracy.
- Used this to write an improved method for cross-validation hyper-parameter selection in `sklearn`.
- Released our software pipeline for free as a Zenodo repository: `https://doi.org/10.5281/zenodo.3731703`

This talk is a survey: details in papers at CICM 2019, SC^2 2019, MACIS 2019, ICMS 2020.

Surveyed Publications (Slide 6)

M. England and D. Florescu.
Comparing machine learning models to choose the variable ordering for cylindrical algebraic decomposition. Intelligent Computer Mathematics (Proc. CICM '19), LNCS 11617, pp. 93–1082. Springer, 2019. DOI: 10.1007/978-3-030-23250-4_7

D. Florescu and M. England.
Algorithmically generating new algebraic features of polynomial systems for machine learning. Proc. SC^2 '19, CEUR Workshop Proceedings 2460, 2019. `http://ceur-ws.org/Vol-2460/`

D. Florescu and M. England.
Improved cross-validation for classifiers that make algorithmic choices to minimise runtime without compromising output correctness. Mathematical Aspects of Computer and Information Sciences (Proc. MACIS '19), LNCS 11989, pp. 341–356. Springer, 2020. DOI: 10.1007/978-3-030-43120-4_27

D. Florescu and M. England.
A Machine Learning Based Software Pipeline to Pick the Variable Ordering for Algorithms with Polynomial Inputs. Mathematical Software (Proc. ICMS '20), LNCS 12097, pp. 302–322. Springer, 2020. DOI: 10.1007/978-3-030-52200-1_30

Should I care if I don't work on CAD? (Slide 7)

Our results could generalise beyond the context of choosing a variable ordering for CAD.

- Our feature identification process is useful whenever one needs to derive features for ML from a set of polynomials.
- Our hyper-parameter selection process is useful whenever we are seeking to take a choice that minimises computation.

Further, the broader topic of ML to safely improve the performance of symbolic algorithms has huge potential:

- Most CAS developers will likely be able to identify similar heuristic choices to be made in their algorithms.
- Many CAS users will be familiar with the changes of performance that can arise through changing algorithm settings not critical to their application.

In both cases ML offers a structured route forward.

Should I care if I don't work on CAD? (Slide 7)

Our results could generalise beyond the context of choosing a variable ordering for CAD.

- Our feature identification process is useful whenever one needs to derive features for ML from a set of polynomials.
- Our hyper-parameter selection process is useful whenever we are seeking to take a choice that minimises computation.

Further, the broader topic of ML to safely improve the performance of symbolic algorithms has huge potential:

- Most CAS developers will likely be able to identify similar heuristic choices to be made in their algorithms.
- Many CAS users will be familiar with the changes of performance that can arise through changing algorithm settings not critical to their application.

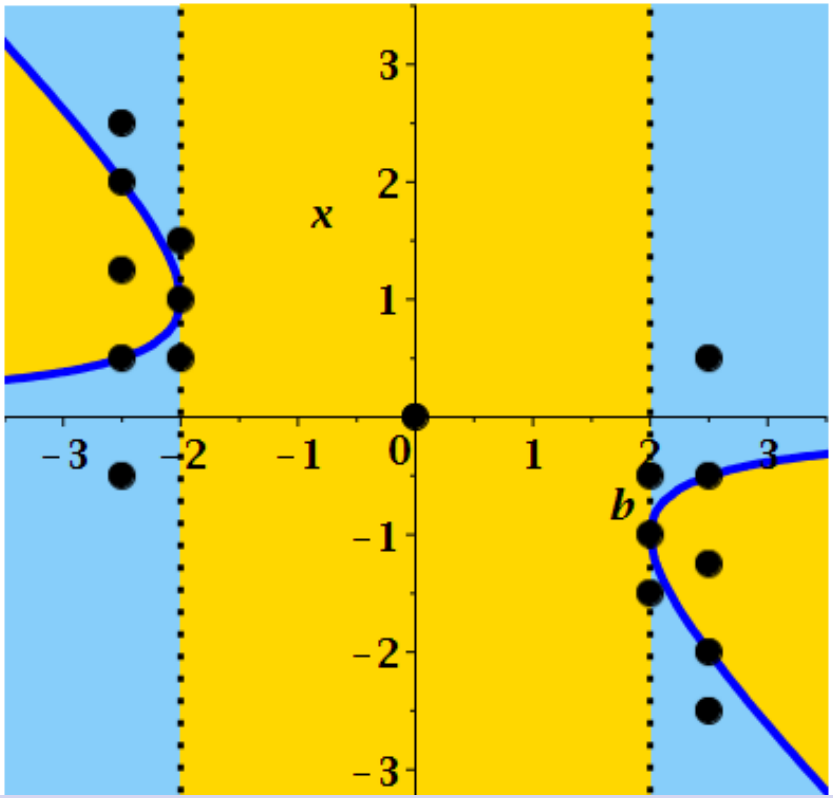
In both cases ML offers a structured route forward.

Outline

CAD and QE

(Slide 8)

A CAD is a decomposition of $\mathbb{R}^n$ that may be algorithmically produced from a set of polynomials such that each polynomial has invariant sign on each cell. E.g. CAD below for $\{x^2 + bx + 1\}$. The invariance means we can answer questions on polynomials by testing a finite number of sample points.



CAD and QE

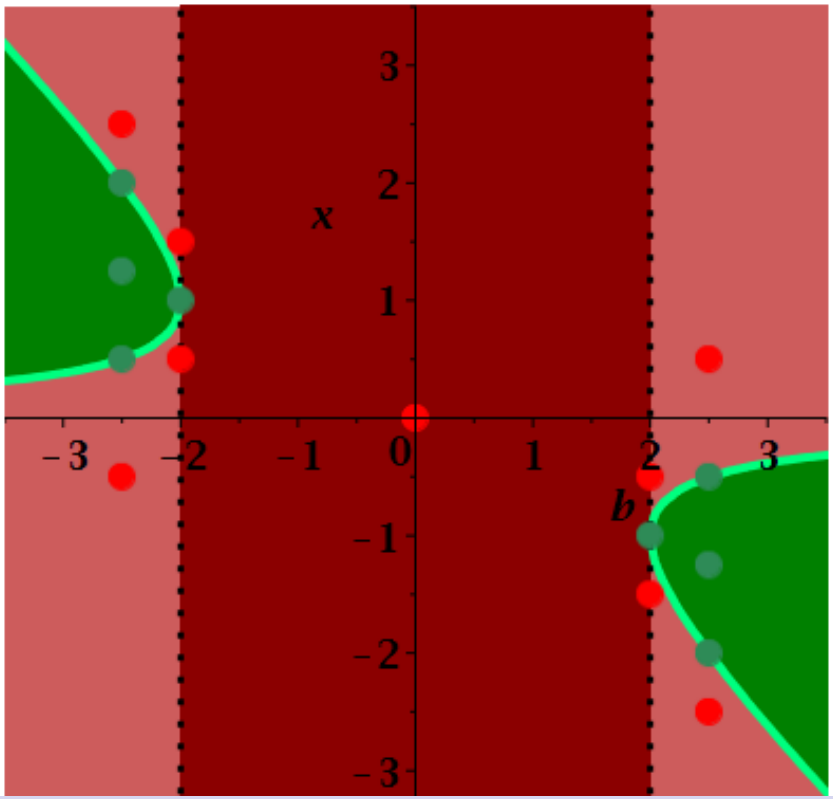
(Slide 8)

A CAD is a decomposition of $\mathbb{R}^n$ that may be algorithmically produced from a set of polynomials such that each polynomial has invariant sign on each cell. E.g. CAD below for $\{x^2 + bx + 1\}$. The invariance means we can answer questions on polynomials by testing a finite number of sample points.

A key application is quantifier elimination. E.g.

$$\exists x, x^2 + bx + 1 \leq 0$$

Tag cells in the CAD true or false according to polynomial sign.



CAD and QE

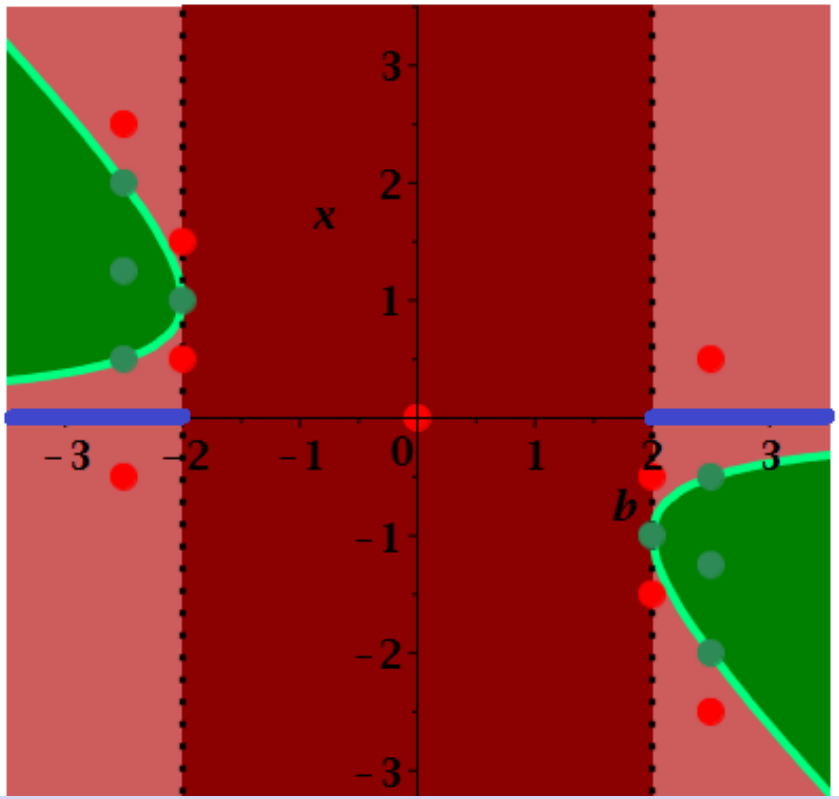
(Slide 8)

A CAD is a decomposition of $\mathbb{R}^n$ that may be algorithmically produced from a set of polynomials such that each polynomial has invariant sign on each cell. E.g. CAD below for $\{x^2 + bx + 1\}$. The invariance means we can answer questions on polynomials by testing a finite number of sample points.

A key application is quantifier elimination. E.g.

$$\exists x, x^2 + bx + 1 \leq 0$$

Tag cells in the CAD true or false according to polynomial sign. Then project true cells and take disjunction. $\implies b \leq -2 \vee b \geq 2$.



CAD Variable Ordering Choice (Slide 9)

CAD has doubly exponential complexity in the number of variables! The complexity is felt in practice. But careful optimisations and pre-processing can *push back the doubly exponential wall* and bring new applications in scope.

A particularly important optimisation is the variable ordering. For QE one must order variables as they are quantified; but there is no restriction on free variables and adjacent quantifiers of the same type may be swapped.

The ordering is well known to dramatically affect both the number of cells and the time to compute them.

Usually systems use human made heuristics which are far from perfect. In 2014 we used a support vector machine to pick from the 6 possible orderings for 3 variable problems.

CAD Variable Ordering Choice (Slide 9)

CAD has doubly exponential complexity in the number of variables! The complexity is felt in practice. But careful optimisations and pre-processing can *push back the doubly exponential wall* and bring new applications in scope.

A particularly important optimisation is the variable ordering. For QE one must order variables as they are quantified; but there is no restriction on free variables and adjacent quantifiers of the same type may be swapped.

The ordering is well known to dramatically affect both the number of cells and the time to compute them.

Usually systems use human made heuristics which are far from perfect. In 2014 we used a support vector machine to pick from the 6 possible orderings for 3 variable problems.

Our Recent Work I

We experimented with different ML classifiers in `sklearn`

- Support Vector Machine (SVM) classifier with RBF kernel.
- K−Nearest Neighbours (KNN) classifier.
- Multi-Layer Perceptron (MLP) classifier.
- Decision Tree (DT) classifier.

The classifiers are trained not on the input polynomials but vectors of real numbers derived from them. These are called *features*.

We started with a handful of features from the human made heuristics. Then developed a framework to enumerate (all appropriate) combinations of some (basic) functions on a set of polynomials. Led to 78 unique features for 3-variable problems, and 105 features for 4-variable problems.

Our Recent Work II

We then considered the definition of accuracy (used to train and evaluate classifiers). We started with the standard definition of predicting the *right* answer.

Accuracy: the percentage of problems in the testing set for which a heuristic picked the optimal ordering.

But this is not a meaningful evaluation as it ranks the second best ordering the same as the worst. We now evaluate using:

Accuracy: *the percentage of problems where a classifier's predicted variable ordering led to a computing time closer than x% of the time of the optimal ordering*.

We then rewrote the `sklearn` function for selecting classifier hyperparameters via cross validation to use the corresponding CAD computation times instead of the optimal ordering choice.

Successes (Slide 12)

Experiments showed that:

- All machine learning classifiers made better choices on average over the data set than pre-existing human-made heuristics.
- All the new classifiers tried did better than SVM (the one used in 2014).
- All ML classifiers benefited from using the additional features.
- All classifiers benefited from the new cross validation procedure (both with and without the additional features).

On our 3-variable dataset: human made heuristics achieved times 27% above the minimum; ML achieved times 6% above.

On our 4-variable dataset: human made heuristics achieved times 98% above the minimum; ML achieved times 67% above.

Successes (Slide 12)

Experiments showed that:

- All machine learning classifiers made better choices on average over the data set than pre-existing human-made heuristics.
- All the new classifiers tried did better than SVM (the one used in 2014).
- All ML classifiers benefited from using the additional features.
- All classifiers benefited from the new cross validation procedure (both with and without the additional features).

On our 3-variable dataset: human made heuristics achieved times 27% above the minimum; ML achieved times 6% above.

On our 4-variable dataset: human made heuristics achieved times 98% above the minimum; ML achieved times 67% above.

Methodological Choices (Slide 13)

- We adhered to standard good practice for ML experimentation (e.g. separate data sets for training and testing, cross validation procedure for selecting hyper-parameters separately to training).
- We choose to avoid training on polynomials as textual data to reduce the risk of overfitting to the training data.
- We used data derived from real world applications of CAD, from the SMT-LIB (non-linear real arithmetic section).
- We choose to use the classical supervised learning ML algorithms in sklearn rather than a deep learning framework like TensorFlow as deep learning required quantities of data that were orders of magnitude greater than our dataset.

Challenges (Slide 14)

- Dealing with exponential growth in number of choices.
- Adapting the problem to the unsupervised learning ML paradigm. Sample choices?
- Finding datasets that are sufficiently varied, and representative of all use-cases. Pre-trained adaptable classifiers?
- Finding datasets large enough for deep learning. Random generation to fit statistical properties? Perturbation of real world examples? Random generation of the real world scenario?

A new PhD student has just started at Coventry University to grapple with these challenges!

Challenges

- Dealing with exponential growth in number of choices.
- Adapting the problem to the unsupervised learning ML paradigm. Sample choices?
- Finding datasets that are sufficiently varied, and representative of all use-cases. Pre-trained adaptable classifiers?
- Finding datasets large enough for deep learning. Random generation to fit statistical properties? Perturbation of real world examples? Random generation of the real world scenario?

A new PhD student has just started at Coventry University to grapple with these challenges!

Challenges

- Dealing with exponential growth in number of choices.
- Adapting the problem to the unsupervised learning ML paradigm. Sample choices?
- Finding datasets that are sufficiently varied, and representative of all use-cases. Pre-trained adaptable classifiers?
- Finding datasets large enough for deep learning. Random generation to fit statistical properties? Perturbation of real world examples? Random generation of the real world scenario?

A new PhD student has just started at Coventry University to grapple with these challenges!

The End

Contact Details

Matthew.England@coventry.ac.uk

http://computing.coventry.ac.uk/~mengland/

Thanks for listening!