Machine Learning to Steer Symbolic Computation from its Worst Case Complexity

Matthew England
Coventry University

BCTCS and AlgoUK 2020
36th British Colloquium for Theoretical Computer Science
~~Swansea~~ Zoom 6–8 April 2020

Joint work with Dorian Florescu. Supported by EPSRC grant EP/R019622/1.

Outline

Summary: Main Thesis (1/25)

The author works on algorithms for symbolic computation, implemented in Computer Algebra Systems (CASs). CASs prioritise exact mathematical results using algorithms proven to be correct.

However, many algorithms and implementations contain choices and settings which do NOT affect the mathematical correctness of the output, but could greatly affect the time/memory used in reaching that output.

Such choices are currently mostly taken by either the user, by a magic constant (educated choices picked by the developers) or perhaps a human written heuristic.

Our thesis is that could be better taken by a Machine Learning classifier: tools that use statistical techniques to give computer systems the ability to *learn* rules from data.

Summary: Specific Topic (2/25)

Specifically: we 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 as a Zenodo data repository.

Could easily adapted the above to other situations that require the selection of a variable ordering for a set of polynomials.

Publications

M. England and D. Florescu.

Comparing machine learning models to choose the variable ordering for cylindrical algebraic decomposition. Intelligent Computer Mathematics (Proc. CICM '15), 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

Zenodo repository DOI: 10.5281/zenodo.3731703

Outline

Outline

Motivation: Real QE

Real Quantifier Elimination (QE)

Given: Quantified formulae in prenex form with atoms integral polynomial constraints.
Produce: quantifier free formula logically equivalent over $\mathbb{R}$.

Fully quantified

Input: $\forall x, x^2 + 1 > 0$

Output: `True`

Input: $\exists x, x^2 + 1 \leq 0$

Output: `False`

Input: $\exists x, x^2 + 3x + 1 \leq 0$

Output: `True`

Partially quantified

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

Output: $(b \leq -2) \vee (b > 2)$

When partially quantified the equivalent quantifier free formula must depend on the free (unquantified) variables.

Motivation: Real QE

Real Quantifier Elimination (QE)

Given: Quantified formulae in prenex form with atoms integral polynomial constraints.
Produce: quantifier free formula logically equivalent over $\mathbb{R}$.

Fully quantified

Input: $\forall x, x^2 + 1 > 0$
Output: `True`

Input: $\exists x, x^2 + 1 \leq 0$
Output: `False`

Input: $\exists x, x^2 + 3x + 1 \leq 0$
Output: `True`

Partially quantified

Input: $\exists x, x^2 + bx + 1 \leq 0$
Output: $(b \leq -2) \vee (b > 2)$

When partially quantified the equivalent quantifier free formula must depend on the free (unquantified) variables.

Outline

Cylindrical Algebraic Decomposition (5/25)

Cylindrical Algebraic Decomposition (CAD) is the only implemented complete algorithm for Real QE. A CAD is:

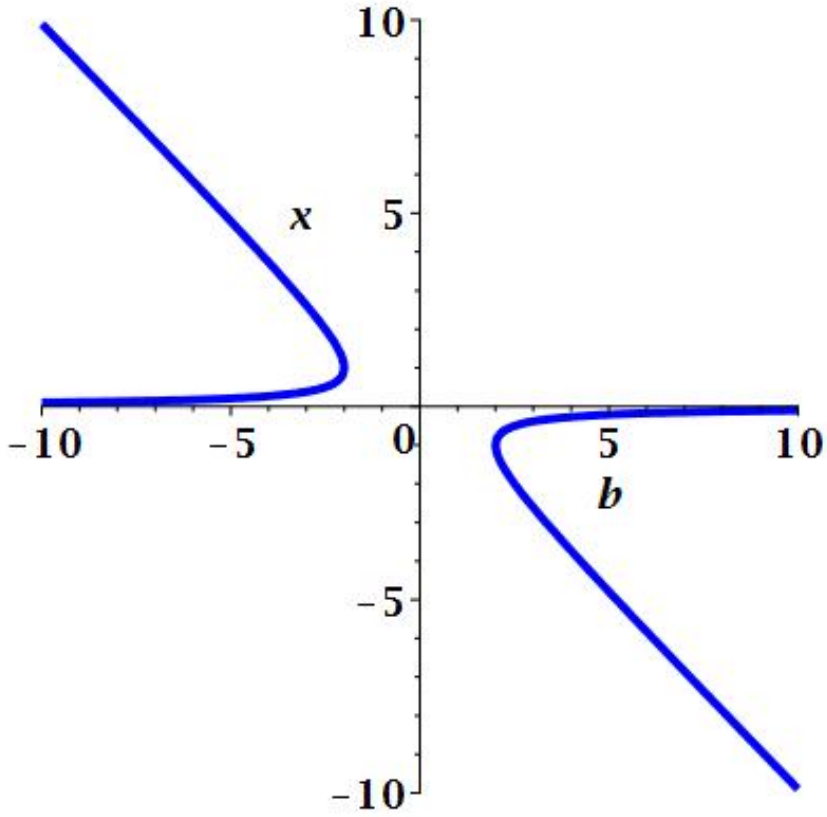
- A **decomposition** of $\mathbb{R}^n$ such that the polynomials involved in the input have constant sign $(+/0/-)$ in each cell. Thus any formulae built with them have constant truth value.
- The cells are **semi-algebraic** meaning they are described by finite sequence of polynomial constraints.
- The cells are **cylindrical** meaning projection (relative to the variable ordering) is trivial from the cell description, and projections of any two cells are identical or disjoint.

Thus existential QE via projection of true cells onto free variables. Universal QE via $\forall x F(x) = \neg\exists x \neg F(x)$.

QE via CAD Example

Recall from earlier the problem:

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



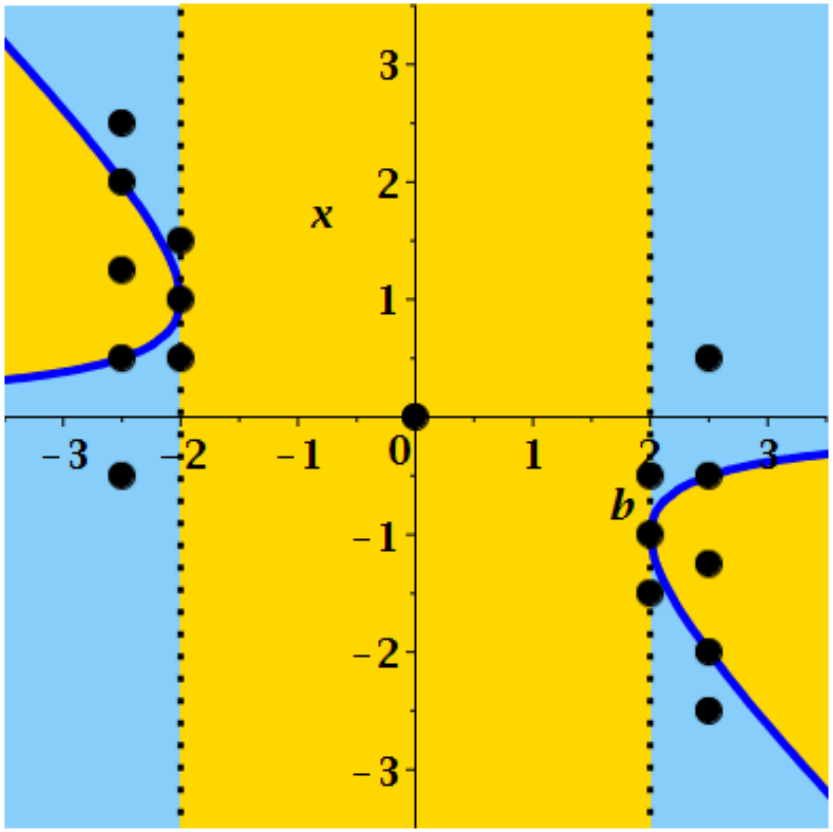
QE via CAD Example

Recall from earlier the problem:

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

To solve we:

Build a sign-invariant CAD for $f = x^2 + bx + 1$.



QE via CAD Example

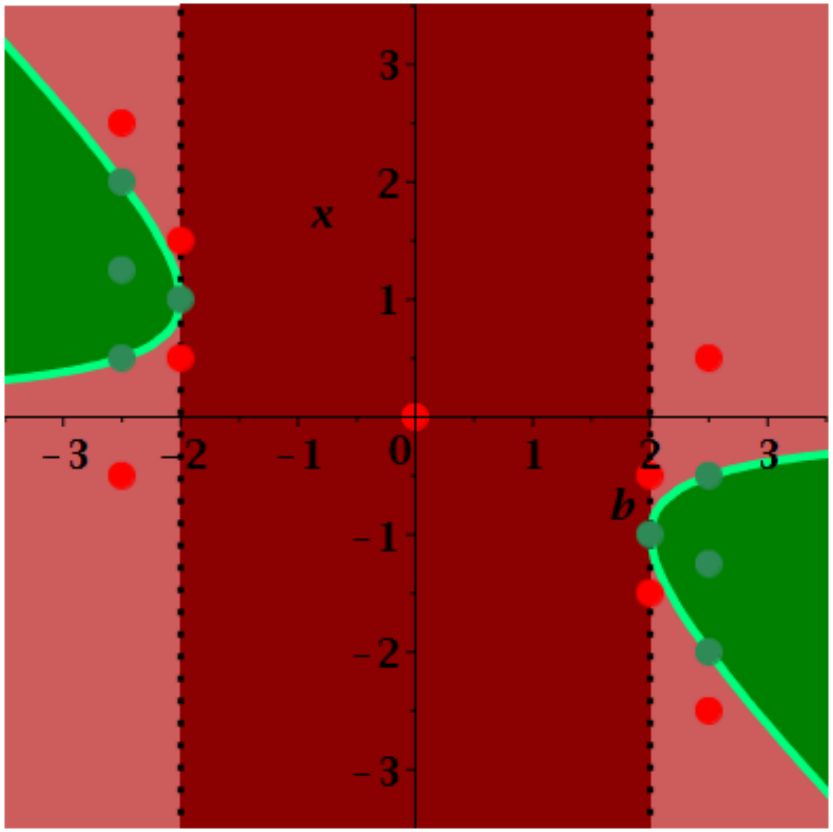
Recall from earlier the problem:

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

To solve we:

Build a sign-invariant CAD for $f = x^2 + bx + 1$.

Tag each cell true or false according to $f \leq 0$.



QE via CAD Example

Recall from earlier the problem:

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

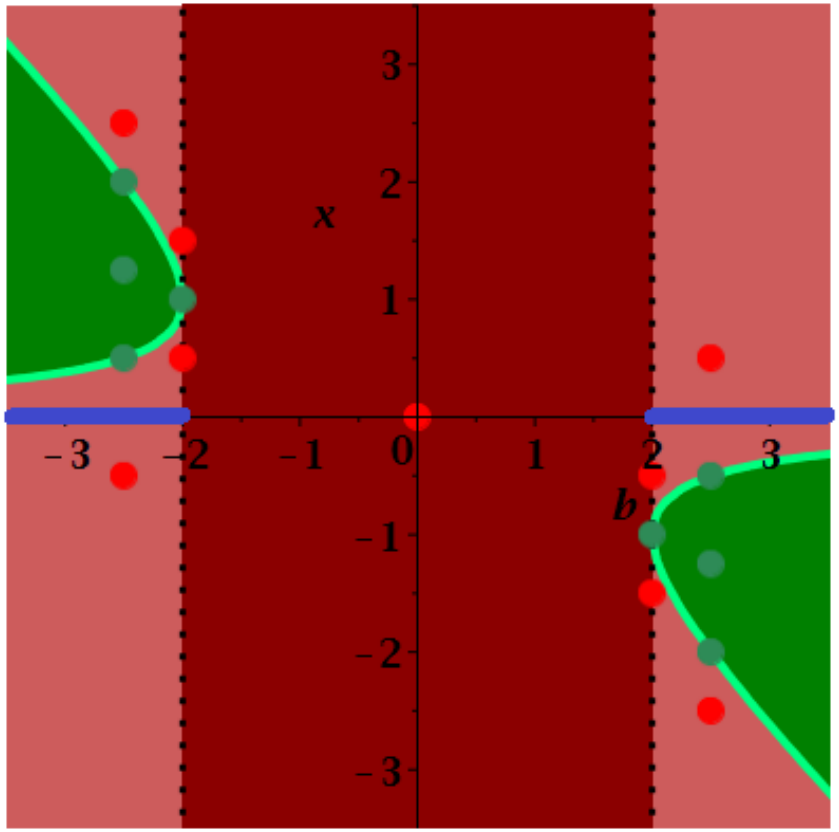
To solve we:

Build a sign-invariant CAD for $f = x^2 + bx + 1$.

Tag each cell true or false according to $f \leq 0$.

Take disjunction of projections of true cells:

$$b < -2 \vee b = -2 \vee b = 2 \vee b > -2$$



QE via CAD Example

Recall from earlier the problem:

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

To solve we:

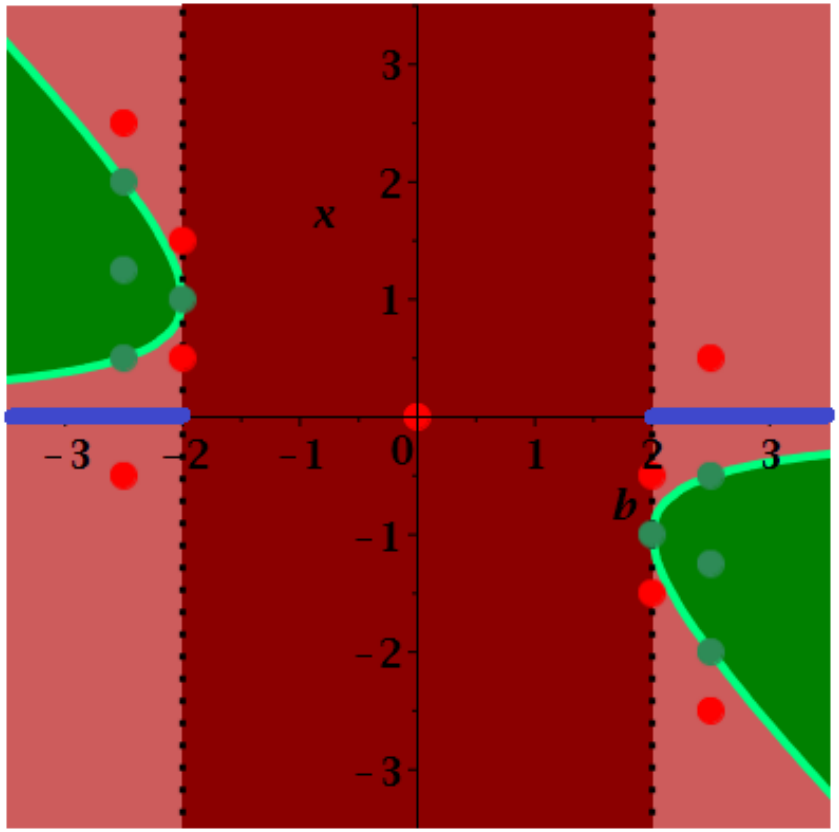
Build a sign-invariant CAD for $f = x^2 + bx + 1$.

Tag each cell true or false according to $f \leq 0$.

Take disjunction of projections of true cells:

$$\Longrightarrow$$

$$b \leq -2 \vee b \geq 2$$



Implementations and Applications (7/25)

QE and CAD implementations are traditionally found in Computer Algebra Systems, e.g. MAPLE (`RegularChains`, `SyNRAC`), MATHEMATICA, REDUCE (`Redlog`), QEPCAD-B.
SMT for non-linear real arithmetic is a special case of Real QE.
Thus CAD now in SMT-solvers, e.g. SMT-RAT, YICES, Z3.

QE can solve problems throughout engineering & science. E.g.

- derivation of optimal numerical schemes (Erascu-Hong, 2014)
- artificial intelligence (Todai Robot Project)
- automated theorem proving (Paulson, 2012)
- bio-chemical network analysis (Bradford et al., 2017)
- automated loop parellisation (Grösslinger et al. 2006)
- analysis of economic hypotheses (Mulligan et al., 2018)

CAD Complexity

Building the CAD relies on a repeated projection of polynomials to track key geometric information through the use of e.g. resultants. Then repeated use of real root isolation to track where the root structure of these polynomials changes. This process has doubly exponential complexity in the number of variables!

C. Brown and J.H. Davenport.
The complexity of quantifier elimination and cylindrical algebraic decomposition.
In Proc. ISSAC '07, pages 54–60. ACM, 2007.

By the end of projection you have doubly exponentially many polynomials of doubly exponential degree.

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.

Outline

CAD Variable Ordering

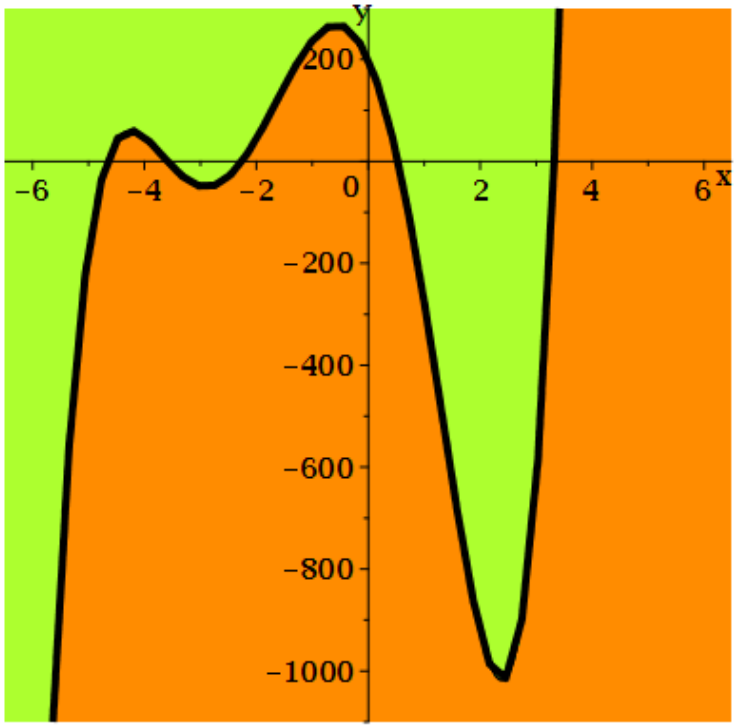
CADs are defined with respect to an ordering on variables (for cylindricity, projection etc.) For QE one must order variables as they are quantified; but there is no restriction on free variables and adjacent quantifiers of the same type may be swapped. Thus for SMT on NRA with n real variables we have $n!$ choices.

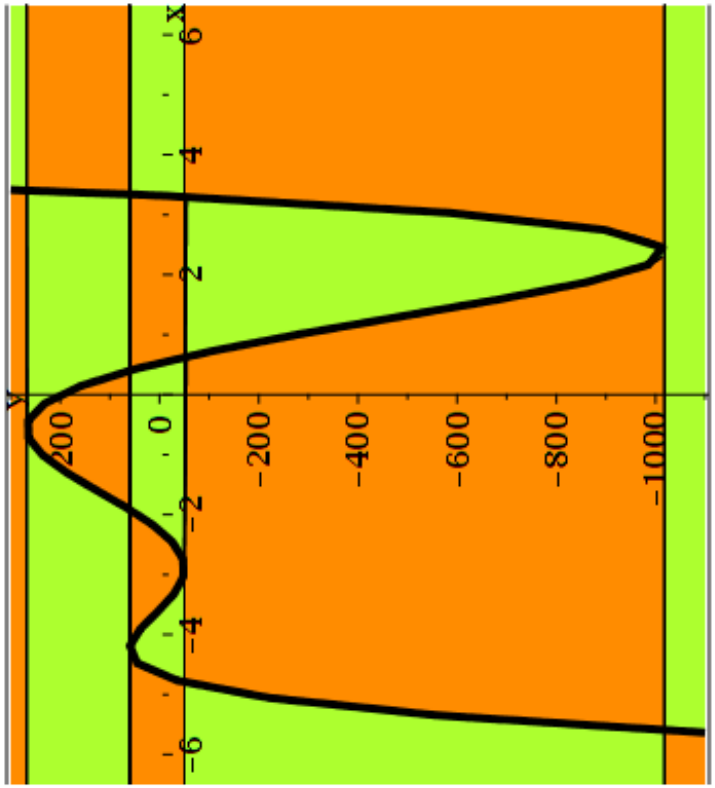
There is a class of problems in which one variable ordering gives output of double exponential complexity in the number of variables and another output of a constant size!

More generally, it is well observed that choice of variable ordering can dramatically affect both the number of cells in the decomposition and the time required to compute them, often to the point of feasibility.

CAD Variable Ordering Example

CAD for polynomial $y - 3x^5 + 20x^4 - 10x^3 - 240x^2 - 250x + 200$.
With $y \succ x$ a sign-invariant CAD has 3 cells, with $y \prec x$ it is 59.





Outline

ISSAC 2004 Heuristics

C. Brown.
Tutorial Notes: Cylindrical algebraic decomposition.
Presented at ISSAC 2004.

It uses three simple criteria on degrees and frequency of occurrence of variables in the input, applied in turn with ties broken by the next. Used in `Qepcad` and `Maple`.

A. Dolzmann, A. Seidl and T. Sturm . In: Proc.
Efficient projection orders for CAD.
Proc. ISSAC 2004, pp.111-118, ACM (2004).

Identified heuristic based on `sotd`: **s**um **o**f **t**otal **d**egrees of all monomials of all polynomials in the projection set. Used in `Redlog`. More expensive as it requires projection operations, but there is an obvious greedy variant where we choose one variable at a time based on a single projection.

More recent human-made heuristics (12/25)

R. Bradford, M. England, J.H. Davenport and D. Wilson.
Optimising problem formulations for cylindrical algebraic decomposition.
Proc. CICM 2013, LNCS 7961, pp. 19-34. Springer 2013.

Found examples where `sotd` misled (differences between $\mathbb{R}$ and $\mathbb{C}$).
Hence `ndrr`: number of distinct real roots in decomposition of $\mathbb{R}$.

D. Wilson, M. England, R. Bradford and J.H. Davenport.
Using the distribution of cells by dimension in a cylindrical algebraic decomposition.
Proc. SYNASC 2014, pp. 53-60. IEEE 2014.

Went even further and counted full dimensional cells in a CAD (so no algebraic number computations).

Heuristics getting too expensive. Also – still not fully accurate and no obvious greedy variants.

Outline

Huang et al. (CICM 2014)

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

Used a Support Vector Machine (SVM) to choose which of three human made heuristics to follow when picking an ordering.

- Experiments on 7000 problems identified substantial subclasses on which each made a better decision.
- Trained three SVMs and used relative magnitude of their margin values to pick which heuristic to follow.
- ML choice did significantly better than any one heuristic.

Outline

Our CICM 2019 Paper

Repeated Huang *et al.*'s experiments but this time:

- Choose variable ordering directly.

Many examples where no human-made heuristic had a good choice: greater savings but harder to scale for larger n.

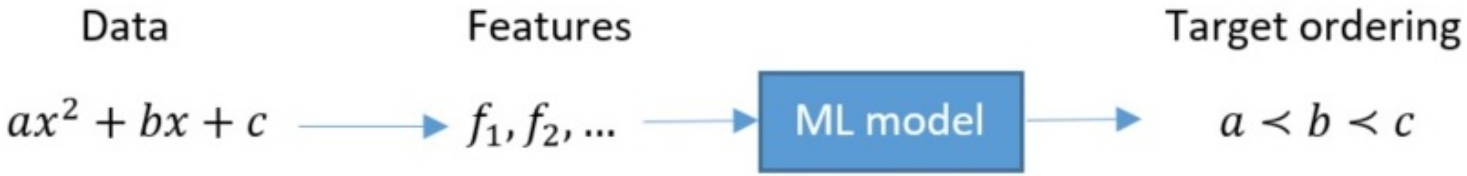
- Experimented with different ML classifiers
  - Support Vector Machine (SVM) classifier with RBF kernel.
  - K−Nearest Neighbours (KNN) classifier.
  - Multi-Layer Perceptron (MLP) classifier.
  - Decision Tree (DT) classifier.

All did better than human-made heuristics but SVM actually beaten significantly by the other three.

Outline

ML Features

The ML classifiers are trained not on the input polynomials but vectors of real numbers derived from them. Each of these numbers corresponds to a feature of the input.



Work above used 11 features inspired by Brown's heuristic, e.g.:

- Degree of a variable in the input.
- Proportion of input polynomials which contact a variable.
- Proportion of input monomials which contact a variable.

(Q) Are there more / better features we can extract from the input polynomials without resorting to expensive projection operations?

Our SC^2 2019 Paper

Presented a framework to enumerate (all appropriate) combinations of some (basic) functions on a set of polynomials. This encompasses all previously used features but also many more.

- *Basic functions*: `sign`, `max`, `sum`, `average`. All cheap!
- *All appropriate combinations*: i.e. taking care of the different dimensions being applied to.

All possibilities in 3 variables gave 1728 features. But:

- Many easily seen as mathematically identical.
- Some others are certainly identical for the dataset in question.
- A handful were constant (evaluate to the same number) for the whole dataset (making them useless for ML).

After this: 78 features for 3-variable problems, compared to 11 previously: seven times more! 105 features for 4-variable problems.

Our SC^2 2019 Paper Results

		DT	KNN	MLP	SVM
11 features	Accuracy (%)	62.6	63.3	61.6	58.8
	Time (s)	9,994	10,105	9,822	10,725
78 features	Accuracy (%)	65.2	66.3	**67**	65
	Time (s)	9,603	**9,178**	9,399	9,487

	Virtual Best	Virtual Worst	random	sotd	Brown
Accuracy (%)	*100*	*0*	22.7	49.5	51
Time (s)	*8,623*	*64,534*	30,235	11,938	10,951

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

Time: the computation time if all the testing set had CADs computed with that heuristic's suggested orderings.

Our SC^2 2019 Paper Results

		DT	KNN	MLP	SVM
11 features	Accuracy (%)	62.6	63.3	61.6	58.8
	Time (s)	9,994	10,105	9,822	10,725
78 features	Accuracy (%)	65.2	66.3	**67**	65
	Time (s)	9,603	**9,178**	9,399	9,487

	Virtual Best	Virtual Worst	random	sotd	Brown
Accuracy (%)	*100*	*0*	22.7	49.5	51
Time (s)	*8,623*	*64,534*	30,235	11,938	10,951

The human made heuristics achieved times that are 27% above the minimum possible. ML with similar features reduced that to 14% above. The additional features reduced it to only 6% above. All ML classifiers improved performance with extra features.

Our SC^2 2019 Paper Results

		DT	KNN	MLP	SVM
11 features	Accuracy (%)	62.6	63.3	61.6	58.8
	Time (s)	9,994	10,105	9,822	10,725
78 features	Accuracy (%)	65.2	66.3	**67**	65
	Time (s)	9,603	**9,178**	9,399	9,487

	Virtual Best	Virtual Worst	random	sotd	Brown
Accuracy (%)	*100*	*0*	22.7	49.5	51
Time (s)	*8,623*	*64,534*	30,235	11,938	10,951

Quickest computation times achieved by KNN, although MLP had slightly higher accuracy. I.e. MLP makes best choice more often but the occasions it makes a poor choice drag its times down considerably.

Outline

Feedback we received

We received sensible feedback about our definition of accuracy.
Accuracy: *the percentage of problems in the testing set for which a heuristic picked the optimal ordering.*
Standard meaning of accuracy in ML, but does not distinguish between an "almost optimal" ordering and a "very bad" ordering.
A classifier that always picks second best is probably preferred than one that picks best half the time and worst the other half!

We agreed that a better evaluation metric for accuracy would be
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 use $x = 20$ later.

But how to use something like this in training?

Parameters vs Hyperparameters

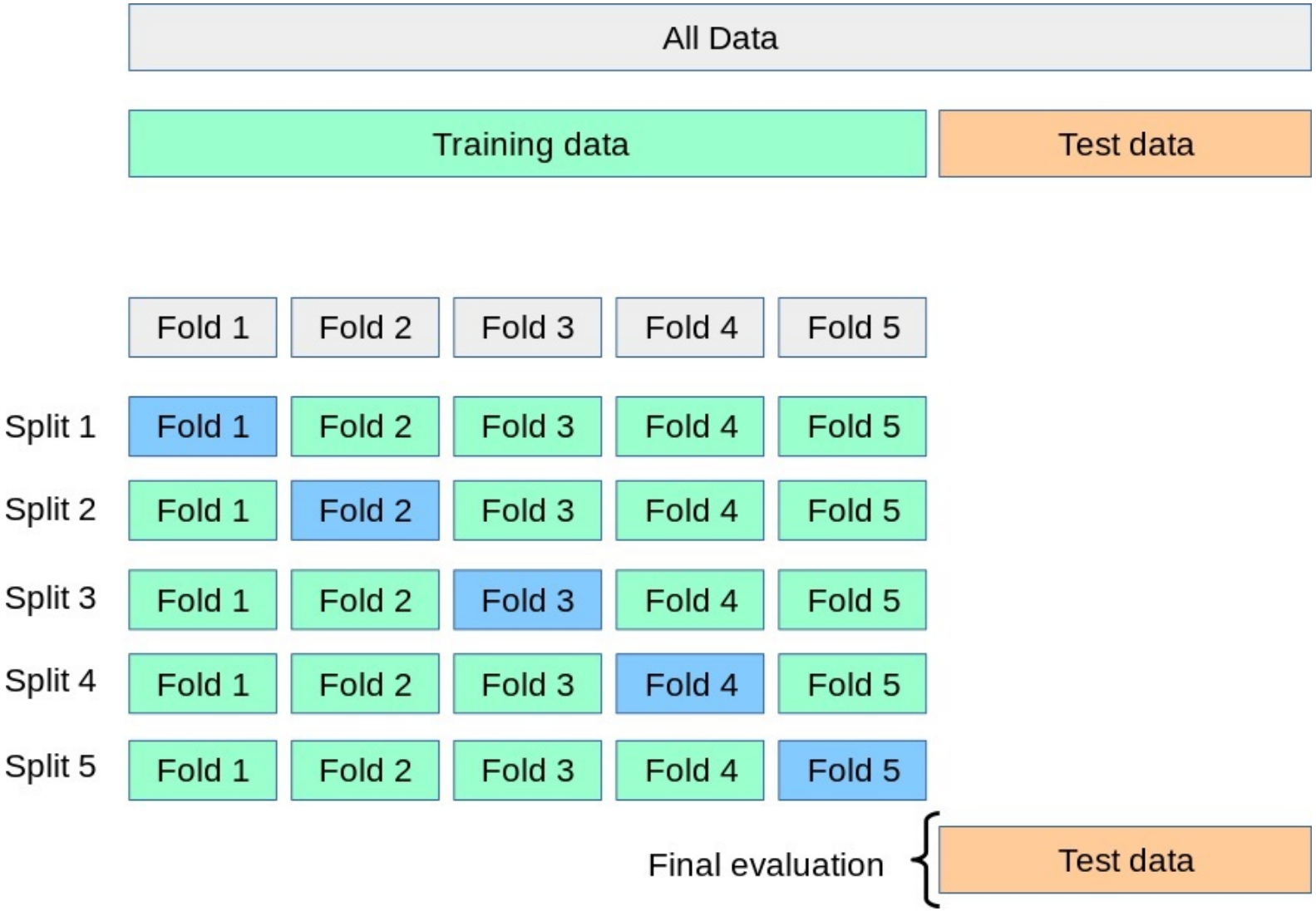
ML models requires the fixing of

Parameters: variables that can be fine-tuned during training so that the prediction error reaches a minimum, e.g. weights in an artificial neural network.

hyperparameters: model configurations selected before training e.g. the number of layers in a neural network. Often specified by the practitioner based on experience or a simple grid search.

To prevent over-fitting the hyperparameters and parameters are recommended to be tuned on different datasets. One approach is to do this as part of k-fold cross-validation. Here the training data is split into k groups: each combination of $k-1$ groups is used to train the parameters with the final group used to select the hyper-parameters for that model.

E.g. 5−fold Cross Validation



New approach to choosing hyperparameters (20/25)

Typically in ML the hyperparameters are selected to maximise

$$h_{opt} = \operatorname*{argmax}_h \left(\frac{1}{k} \sum_{g=1}^{k} \text{score}_h^g \right),$$

where h are the different hyperparameter options, and score_h^g denotes the F1-score of that group with that hyperparameter choice for the model prediction.

Our new idea is to instead maximise

$$h_{opt} = \operatorname*{argmax}_h \left(\frac{1}{k} \sum_{g=1}^{k} -\text{ctime}_h^g \right),$$

where ctime_h^g denotes the time for computing CADs on this group using the variable ordering predicted by the model with those hyperparameters.

Experimental methodology (21/25)

In our MACIS 2019 paper we present an experiment to see if this new cross-validation approach is effective. We also experiment with 4-variable problems for the first time (24 possible orderings).

- Use the `nlsat` dataset of D. Jovanović. Extracted 2080 problems (polynomial sets).
- Split randomly into sets for training (1546) and testing (534).
- CADs built for all orderings in Maple with time limit of 64s.
- Classifiers trained to minimise computation time using 105 features and 3-fold cross-validation.
- Compared classifiers to each other and human-made heuristics: the latter do not always distinguish and in such cases the average time of all their selections is computed.

MACIS 2019 Paper Results (4 variables)

$-O$ uses original cross validation and $-N$ the new one.

	DT-O	DT-N	KNN-O	KNN-N
Acc.	51.7%	54.3%	53.9%	54.5%
Time	4,022	**3,627**	3,808	3,748

	MLP-O	MLP-N	SVM-O	SVM-N
Acc.	53.6%	**56.9%**	53.9%	54.9%
Time	3,972	3,784	3,795	3,672

	Virtual Best	Virtual Worst	random	`Brown`	`sotd`
Acc.	100%	0%	17.0%	20.1%	47.8%
Time	2,177	22,735	8,291	8,292	4,348

Results Analysis

- For each ML model the performance when trained with the new cross-validation was better (measured using either accuracy or total computation time).
- The scale of the improvement varied: Decision Tree 9.8% quicker but KNN only 1.6% quicker.
- All ML classifiers outperform human-made heuristics (with or without the new cross validation). Thus our ML methodology can be extended to 4-variable problems.
- All heuristics (ML and human-made) are further away from the optimum on this 4-variable dataset than with 3-variables: to be expected as choosing from 24 rather than 6 orderings.
- Best performing ML model achieves timings 67% greater than the minimum; best human-made heuristic is 98% greater.

What Next?

We have demonstrated a methodology that can do well on the standard community dataset (at least up to 4 variables).
But classifiers trained on this dataset do not do so well when applied on examples from outside the dataset (such as those from biology or economics).

We have developed a software pipeline which can retrain the classifiers to a new dataset which a single click (available open source on Zenodo).

But this is reliant on there being a dataset of sufficient size: our collection of examples in economics number less than 100.
Now experimenting with training classifiers on random polynomial data in which certain features are selected according to a probability distribution based on a real dataset.

The End

Contact Details

Matthew.England@coventry.ac.uk

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

Thanks for listening!