Journal of Symbolic Computation 126 (2025) 102345

Contents lists available at ScienceDirect

Journal of Symbolic Computation

journal homepage: www.elsevier.com/locate/jsc

Machine learning parameter systems, Noether normalisations and quasi-stable positions

Amir Hashemi [a,b], Mahshid Mirhashemi [a], Werner M. Seiler [c]

[a] Department of Mathematical Sciences, Isfahan University of Technology, Isfahan 84156-83111, Iran
[b] School of Mathematics, Institute for Research in Fundamental Sciences (IPM), Tehran, 19395-5746, Iran
[c] Institut für Mathematik, Universität Kassel, Heinrich-Plett-Str. 40, 34132 Kassel, Germany

A R T I C L E I N F O

Article history:
Received 9 September 2023
Received in revised form 7 May 2024
Accepted 11 June 2024
Available online 17 June 2024

Dataset link: https://doi.org/10.5281/zenodo.12114165

MSC:
13P10
14Q15
68T05
68W30

Keywords:
Quasi-stable ideals
Noether normalisation
Systems of parameters
Pommaret bases
Machine learning
Multi-class classification

A B S T R A C T

We discuss the use of machine learning models for finding “good coordinates” for polynomial ideals. Our main goal is to put ideals into quasi-stable position, as this generic position shares most properties of the generic initial ideal, but can be deterministically reached and verified. Furthermore, it entails a Noether normalisation and provides us with a system of parameters. Traditional approaches use either random choices which typically destroy all sparsity or rather simple human heuristics which are only moderately successful. Our experiments show that machine learning models provide us here with interesting alternatives that most of the time make nearly optimal choices.

E-mail addresses: Amir.Hashemi@iut.ac.ir (A. Hashemi), mahshidmirhashemi@gmail.com (M. Mirhashemi), seiler@mathematik.uni-kassel.de (W.M. Seiler).

https://doi.org/10.1016/j.jsc.2024.102345

1. Introduction

Many results in algebraic geometry and commutative algebra considerably simplify in generic coordinates. While from a theoretical point of view one may simply exploit that a random transformation (almost) always achieves a generic position, the situation is less simple from a computational point of view. Random transformations are computationally bad, as they destroy all sparsity typically present in generators of polynomial ideals. Furthermore, for many generic positions – like for example the popular GIN position in which one obtains the generic initial ideal (see Sect. 2.1 for a definition) – effective tests are either not known or prohibitively expensive.

Quasi-stable position represents an interesting alternative. It shares most of the properties of the GIN position, but can be effectively verified (Hashemi et al., 2012). It entails a Noether normalisation (Seiler, 2009b) and in a quasi-stable position one easily obtains a system of parameters (Seiler, 2012). Furthermore, in a series of articles, we developed a deterministic approach to obtain a quasi-stable position for arbitrary ideals (Hausdorf and Seiler, 2002; Seiler, 2009b; Hashemi et al., 2018). In this approach, one performs a finite sequence of very sparse transformations until quasi-stability is achieved. The efficiency depends crucially on the number of transformations required.

In each step, one typically has a choice between several possible transformations. The correctness and the termination of the whole procedure is independent of this choice. But the number of transformations required to achieve quasi-stable position can depend strongly on it. Previous computational experiments have indicated that simple human heuristics are not very successful in making consistently good choices here. Therefore we propose in this article to apply methods from *machine learning* for selecting the applied transformations.

Our problem is similar to other proposed applications of machine learning in the context of computational commutative algebra. England and collaborators have studied in a larger number of articles the use of various classification methods for choosing the variable ordering for a cylindrical algebraic decomposition, see e.g. (Huang et al., 2014; England and Florescu, 2019; Florescu and England, 2019; Huang et al., 2019; Florescu and England, 2020; Pickering et al., 2024) and noted that these were better than known human heuristics. The problem of learning a selection strategy in Buchberger's algorithm applied to binomial ideals was studied in (Peifer et al., 2020; Peifer, 2021) using reinforcement learning with a 1D convolutional neural network. In all these works, the authors are concerned with choices within an algorithm which do not affect its correctness or termination, but which possess a significant effect on its efficiency. Somewhat related is also the idea of Simpson et al. (2016) to employ machine learning for choosing the most efficient algorithm for computing resultants. By contrast, Jamshidi and Petrović (2023) presented a machine learning approach for computing Gröbner bases directly.

Our interaction between machine learning and symbolic computing is somewhere in between. On the one hand, we use machine learning to directly solve a geometric problem, namely putting an ideal into a distinguished position. Applying an essentially stochastic tool like machine learning does not compromise mathematical correctness here, as the problem studied by us possesses infinitely many solutions and we are guaranteed to find a correct one. On the other hand, our approach to this geometric problem essentially boils down to making a choice within an algorithm the correctness of which is independent of this choice. As in most of the above cited works, the task of the machine learning is here only to increase the *efficiency* of a symbolic computation. As so far nobody has been able to design good heuristics for choosing the next transformation, we now train different machine learning models and check how well they can solve this problem.

This article is structured into two parts. In the next section, we discuss the mathematical foundations: quasi-stability and Pommaret bases. We recall the necessary notions and their relevant properties. On the algorithmic side, we recall the deterministic approach from (Hashemi et al., 2018) and provide two completion algorithms for monomial Janet and Pommaret bases, respectively. The following section is concerned with applying machine learning in this context. We describe the structure of our feature vectors and how we quantitatively compare different choices. After a discussion of the training process, we present our results. Finally, some conclusions are given.

2. Quasi-stability and pommaret bases

We will work throughout in a polynomial ring $\mathcal{S} = \Bbbk[x_1, \ldots, x_n] = \Bbbk[\mathcal{X}]$ over a field $\Bbbk$ of characteristic zero.[1] We consider exclusively homogeneous polynomials and ideal. A *term* is a power product $x_1^{\mu_1} \cdots x_n^{\mu_n} = x^\mu$ with an exponent vector $\mu = (\mu_1, \ldots, \mu_n) \in \mathbb{N}_0^n$. We write $\max x^\mu = j$, if μ_j is the last non-vanishing entry of μ. We will always take the *degree reverse lexicographic order* with $x_1 \succ \cdots \succ x_n$. The *leading term* of a polynomial $f \in \mathcal{S}$ is written $\operatorname{lt} f$. If $\mathcal{F} \subset \mathcal{S}$ is a finite set of polynomials, we denote by $\operatorname{lt}\mathcal{F} = \{\operatorname{lt} f \mid f \in \mathcal{F}\}$ their leading terms. The set $\mathcal{F}$ is a *Gröbner basis* for an ideal $\mathcal{I}$, if $\mathcal{F} \subset \mathcal{I}$ and $\operatorname{lt}\mathcal{I} = \langle \operatorname{lt} g \mid g \in \mathcal{I}\rangle = \langle \operatorname{lt}\mathcal{F}\rangle$. We refer e.g. to (Cox et al., 2015) for more details on Gröbner bases.

2.1. Quasi-stable ideals and quasi-stable position

Quasi-stable ideals represent a special class of monomial ideals that appear in many different places and which are known under many different names e.g. ideals of nested type (Bermejo and Gimenez, 2006), ideals of Borel type (Herzog et al., 2003) or weakly stable ideals (Caviglia and Sbarra, 2005).

Definition 2.1. A monomial ideal $\mathcal{J} \lhd \mathcal{S}$ is *quasi-stable*, if for any term $t \in \mathcal{J}$ and any index $1 \leq i < j = \max t$ there exists an exponent $s > 0$ such that $x_i^s t / x_j \in \mathcal{J}$. A polynomial ideal $\mathcal{I} \lhd \mathcal{S}$ is in *quasi-stable position*, if its leading ideal $\operatorname{lt}\mathcal{I}$ is quasi-stable.

It suffices to verify the condition in Definition 2.1 for the finitely many minimal generators of $\mathcal{J}$, so that quasi-stability can easily be checked effectively. If t is a minimal generator and for the index $1 \leq i < j = \max t$ no term of the form $x_i^s t/x_j$ lies in $\mathcal{J}$, then we call the pair (t, x_i) an *obstruction to quasi-stability*. There are many equivalent characterisations of quasi-stable ideals (see (Hashemi et al., 2018) for a more detailed discussion, references and some further characterisations).

Proposition 2.2. *Let $\mathcal{J} \lhd \mathcal{S}$ be a D-dimensional monomial ideal. Then the following statements are equivalent:*

(i) *$\mathcal{J}$ is quasi-stable.*
(ii) *Every associated prime ideal of $\mathcal{J}$ is of the form $\langle x_1, \ldots, x_j\rangle$ for some index $1 \leq j \leq n - D$.*
(iii) *The variable x_n is not a zero divisor on $\mathcal{S}/\mathcal{J}^{\mathrm{sat}}$ and the variables x_{n-j} for $1 \leq j < D$ are not zero divisors on $\mathcal{S}/\langle \mathcal{J}, x_n, \ldots, x_{n-j+1}\rangle$.*
(iv) *There is an ascending chain $\mathcal{J} : x_n^\infty \subseteq \mathcal{J} : x_{n-1}^\infty \subseteq \cdots \subseteq \mathcal{J} : x_{n-D+1}^\infty$ and for each index $1 \leq j \leq n - D$ there exists a term $x_j^{\ell_j} \in \mathcal{J}$.*
(v) *We have $\mathcal{J}^{\mathrm{sat}} = \mathcal{J} : x_n^\infty$ and for $1 \leq j < D$*

$$\langle \mathcal{J}, x_n, \ldots, x_{n-j+1}\rangle^{\mathrm{sat}} = \langle \mathcal{J}, x_n, \ldots, x_{n-j+1}\rangle : x_{n-j}^\infty . \tag{2.1}$$

(vi) *For all $1 \leq j < n$ we have*

$$\mathcal{J} : x_{n-j}^\infty = \mathcal{J} : \langle x_1, \ldots, x_{n-j}\rangle^\infty . \tag{2.2}$$

The various characterisations show that quasi-stable ideals possess many special properties which are closely related to the chosen coordinate system, in particular to the ordering of the variables $x_1 \succ \cdots \succ x_n$. Seiler (2009b, 2010) furthermore showed that one can provide for quasi-stable ideals an explicit free resolution.

[1] Some minor adaptions are necessary for fields of positive characteristic. In particular, if the field is too small, a field extension is needed. See (Hashemi et al., 2018) for a more detailed discussion of this situation.

Remark 2.3. For a polynomial ideal $\mathcal{I} \lhd \mathcal{S}$, quasi-stable position is stronger than *Noether position*. By Proposition 2.2(iv), the canonical map $\Bbbk[x_{n-d+1}, \dots, x_n] \to \mathcal{S}/\mathcal{I}$ defines a *Noether normalisation* (see also (Seiler, 2010)). Bermejo and Gimenez (2006) proved that an ideal is in quasi-stable position, if and only if the ideal and all primary components of its leading ideal are simultaneously in Noether position. Hashemi et al. (2018) provided a combinatorial characterisation of Noether position analogous to Definition 2.1 by simply ignoring certain obstructions.

Remark 2.4. For a D-dimensional polynomial ideal $\mathcal{I} \lhd \mathcal{S}$, a *maximal system of parameters* consists of $c = n - D$ ideal members $f_1, \dots, f_c \in \mathcal{I}$ such that the ideal $\tilde{\mathcal{I}} \subseteq \mathcal{I}$ generated by them is also D-dimensional. Such systems of parameters are relevant for many computational tasks in commutative algebra; e.g. when computing primary decompositions, their determination represents a serious bottleneck (Decker et al., 1999). Seiler (2012) showed that in quasi-stable position those elements of a Pommaret basis of $\mathcal{I}$ which have a pure variable power as leading term form a maximal system of parameters making its determination trivial.

Quasi-stable position is a generic notion (see e.g. (Seiler, 2010) for a proof): applying the linear coordinate transformation $\mathbf{x} \mapsto A\mathbf{x}$ with a non-singular random matrix $A \in \Bbbk^{n\times n}$ to an ideal $\mathcal{I}$ yields a transformed ideal $\mathcal{I}_A = A \cdot \mathcal{I}$ which is almost always in quasi-stable position. More precisely, the set of all matrices A such that $\mathcal{I}_A$ is in quasi-stable position contains a Zariski open subset of $\Bbbk^{n\times n}$.

Galligo (1974) in characteristic zero and Bayer and Stillman (1987b) in positive characteristic proved for any ideal $\mathcal{I} \lhd \mathcal{S}$ the existence of a *generic initial ideal* $\operatorname{gin}\mathcal{I}$, i.e. they showed that there exists a Zariski open subset $\mathcal{U} \subseteq \mathrm{GL}(n, \Bbbk)$ such that for all $A, B \in \mathcal{U}$ we have $\operatorname{lt}\mathcal{I}_A = \operatorname{lt}\mathcal{I}_B = \operatorname{gin}\mathcal{I}$. We say that $\mathcal{I}$ is in *GIN position*, if $\operatorname{lt}\mathcal{I} = \operatorname{gin}\mathcal{I}$. This position is very popular among theorists, as in it many invariants of $\mathcal{I}$ can already be read off from $\operatorname{lt}\mathcal{I}$. Computationally, it is very expensive to rigorously verify that an ideal is in GIN position; Hashemi et al. (2018) describe an approach based on Gröbner systems.

GIN position entails quasi-stable position, but the converse is not true. Hashemi et al. (2012) showed that most properties of $\operatorname{gin}\mathcal{I}$ also hold for $\operatorname{lt}\mathcal{I}$ provided $\mathcal{I}$ is in quasi-stable position. We recall here some of most important results.

Theorem 2.5. *Let $\mathcal{I} \lhd \mathcal{S}$ be an ideal in quasi-stable position. Then the ideals $\mathcal{I}$ and $\operatorname{lt}\mathcal{I}$ share the following invariants:*

(i) *satiety* $\operatorname{sat}\mathcal{I} = \operatorname{sat}\operatorname{lt}\mathcal{I}$,
(ii) *projective dimension* $\operatorname{pd}\mathcal{I} = \operatorname{pd}\operatorname{lt}\mathcal{I}$ *(or equivalent* $\operatorname{depth}\mathcal{I} = \operatorname{depth}\operatorname{lt}\mathcal{I}$*),*
(iii) *Castelnuovo–Mumford regularity* $\operatorname{reg}\mathcal{I} = \operatorname{reg}\operatorname{lt}\mathcal{I}$.

Furthermore, $\mathcal{S}/\mathcal{I}$ is Cohen–Macaulay, if and only if the same is true for $\mathcal{S}/\operatorname{lt}\mathcal{I}$.

2.2. Pommaret bases

Involutive bases are a special type of Gröbner bases with additional combinatorial properties and depend not only on a term order but also on an involutive division, a refinement of the divisibility relation of terms. They were introduced by Gerdt and Blinkov (1998a) inspired by the Janet–Riquier theory of partial differential equations. The basic idea of an involutive division L is to associate with any generator h in a finite set $\mathcal{H} \subset \mathcal{S}$ a subset $\mathcal{X}_{L,\mathcal{H}}(h) \subseteq \mathcal{X}$ of *multiplicative variables*. In linear combinations (or normal form computations), the generator h may then only be multiplied with polynomials in the subring $\Bbbk[\mathcal{X}_{L,\mathcal{H}}(h)] \subseteq \mathcal{S}$. Loosely speaking, $\mathcal{H}$ is an involutive basis, if even with this restriction it still generates the whole ideal $\langle\mathcal{H}\rangle$. In contrast to ordinary Gröbner bases, involutive bases are non-trivial even for monomial ideals. For an extensive introduction to the theory, algorithmics and history of involutive bases, we refer to (Seiler, 2009a, 2010). We omit here the rather technical definition of an involutive division L and provide only one for L-involutive bases.

Definition 2.6. Let L be an involutive division. The *L-involutive span* of a finite set $\mathcal{H} \subset \mathcal{S}$ of polynomials is the $\Bbbk$-linear space $\langle \mathcal{H} \rangle_L = \sum_{h\in\mathcal{H}} \Bbbk[\mathcal{X}_{L,\mathcal{H}}(h)] \cdot h \subseteq \langle \mathcal{H} \rangle$. The set $\mathcal{H}$ is an *L-involutive basis* of the ideal $\mathcal{I} = \langle \mathcal{H} \rangle$, if (i) all elements of $\mathcal{H}$ have different leading terms, (ii) $\langle \mathcal{H} \rangle_L = \langle \mathcal{H} \rangle$ and (iii) the above sum is direct.

Obviously, $\mathcal{H}$ is an L-involutive basis of $\mathcal{I}$, if and only if $\operatorname{lt}\mathcal{H}$ is an L-involutive basis of $\operatorname{lt}\mathcal{I}$. Hence any involutive basis is also a – generally non-reduced – Gröbner basis. A key requirement is that any involutive basis induces a direct sum decomposition of the ideal generated by it. It implies for example that involutive standard representations of ideal members are unique. For a more detailed discussion see (Seiler, 2010) and (Hashemi et al., 2022) and references therein.

Example 2.7. Let $\mathcal{H}$ be a set of terms. Then the *Janet division* is defined as follows. Assume $x^\mu \in \mathcal{H}$. We have $x_1 \in \mathcal{X}_{J,\mathcal{H}}(x^\mu)$, if and only if $\mu_1 = \max\left\{\nu_1 \mid x^\nu \in \mathcal{H}\right\}$. For deciding whether any of the other variables are multiplicative, we consider certain subsets of $\mathcal{H}$ defined by initial segments of exponent vectors. Given such a segment $(\mu_1, \dots, \mu_j)$, we write $\mathcal{H}_{(\mu_1,\dots,\mu_j)} = \left\{x^\nu \in \mathcal{H} \mid \nu_1 = \mu_1, \dots, \nu_j = \mu_j\right\}$. Now $x_{j+1} \in \mathcal{X}_{J,\mathcal{H}}(x^\mu)$, if and only if $\mu_{j+1} = \max\left\{\nu_{j+1} \mid x^\nu \in \mathcal{H}_{(\mu_1,\dots,\mu_j)}\right\}$.

Definition 2.8. Let $\mathcal{H} \subset \mathcal{S}$ be a finite set of polynomials and $\mathcal{I} = \langle \mathcal{H} \rangle$. The *volume function* of $\mathcal{I}$ is the numerical function $v_\mathcal{I} \colon \mathbb{N}_0 \to \mathbb{N}_0$ given by $v_\mathcal{I}(q) = \dim_\Bbbk \mathcal{I}_q$ where $\mathcal{I}_q$ denotes the homogeneous component of degree q. Given an involutive division L for which $\mathcal{H}$ is involutively autoreduced, we analogously define a *volume function* of the L-involutive span of $\mathcal{H}$ by setting $v_{\langle \mathcal{H} \rangle_L}(q) = \dim_\Bbbk(\langle \mathcal{H} \rangle_L)_q$.

The induced direct sum decomposition makes it trivial to compute the volume function (and thus also the more commonly used Hilbert function) using an involutive basis.

Lemma 2.9. *Let the finite set $\mathcal{H} \subset \mathcal{S}$ be involutively autoreduced for the involutive division L. If we denote by q_h the degree and by k_h the number of multiplicative variables of a generator $h \in \mathcal{H}$, then we have*[2]

$$v_{\langle \mathcal{H} \rangle_L}(q) = \sum_{h\in\mathcal{H}} \left[q \geq q_h\right] \binom{q - q_h + k_h - 1}{k_h - 1}. \tag{2.3}$$

If $\bar{q} = \max_{h\in\mathcal{H}} q_h$, then $v_{\langle \mathcal{H} \rangle_L}$ is a polynomial for all $q \geq \bar{q}$, the volume polynomial *$V_{\langle \mathcal{H} \rangle_L}$ of the involutive span. An explicit expression for it is obtained by simply dropping the Kronecker–Iversion symbol in (2.3).*

In this article, mainly *Pommaret bases* are relevant. Here, the rule to determine the multiplicative variables is particularly simple, as it depends only on the polynomial h and not on the whole set $\mathcal{H}$: if $\max \operatorname{lt} h = j$, then $\mathcal{X}_P(h) = \{x_j, \dots, x_n\}$. Thus the degree reverse lexicographic order is the optimal choice for obtaining large Pommaret spans.

While for many involutive divisions L (e.g. the Janet division), any ideal $\mathcal{I} \lhd \mathcal{S}$ possesses a finite L-involutive basis, this is not true for the Pommaret division P. The following result follows from (Seiler, 2009b, Prop. 4.4) and relates the existence of finite Pommaret bases to quasi-stability.

Theorem 2.10. *A monomial ideal $\mathcal{J} \lhd \mathcal{S}$ has a finite Pommaret basis, if and only if it is quasi-stable. A polynomial ideal $\mathcal{I} \lhd \mathcal{S}$ has a finite Pommaret basis, if and only if it is in quasi-stable position.*

Thus the possible non-existence of a finite Pommaret basis is only a matter of the coordinate system used: after a generic linear transformation every ideal is in quasi-stable position. Seiler (2009b, Prop. 3.19, Thm. 9.2, Prop. 10.1) showed that Pommaret bases provide us with the following *effective* version of Theorem 2.5.

[2] Here, we use for notational simplicity the Kronecker-Iverson symbol $[C]$ which is 1, if the logical statement C is true, and 0 otherwise.

Theorem 2.11. *Let $\mathcal{H}$ be a finite Pommaret basis of the polynomial ideal $\mathcal{I} \lhd \mathcal{S}$. Let q be the maximal degree* $\deg h$ *and* $n-d+1$ *the maximal value of* $\max h$ *for an element* $h \in \mathcal{H}$. *Then:*

(i) $\mathcal{I}^{\mathrm{sat}} = \mathcal{I} : x_n^{\infty}$ *and* $\operatorname{sat}\mathcal{I} = \max\left\{\deg h \mid h \in \mathcal{H} \wedge \max \operatorname{lt} h = n\right\}$,
(ii) $\operatorname{depth}\mathcal{I} = d$,
(iii) $\operatorname{reg}\mathcal{I} = q$.

With these results, the effective computation of such key invariants like $\operatorname{sat}\mathcal{I}$, $\operatorname{depth}\mathcal{I}$ or $\operatorname{reg}\mathcal{I}$ becomes trivial – provided we can efficiently determine a coordinate transformation to quasi-stable position. Despite the fact that quasi-stability is a generic property, many ideals appearing in applications are not in quasi-stable position. Zero-dimensional ideals are the only exception, as they are always in quasi-stable position.

Example 2.12. A celebrated result by Bayer and Stillman (1987a) asserts that generically the maximal degree of a Gröbner basis with respect to the degree reverse lexicographic order is the Castelnuovo–Mumford regularity of the ideal. However, there is no way to effectively verify whether a given ideal is in the required generic position whereas this is trivial for quasi-stability. The difference is nicely demonstrated by the following ideal from (Seiler, 2009b, Ex. 9.9):

$$\mathcal{I} = \langle x_1^8 - x_2^6 x_3 x_4,\ x_2^7 - x_1 x_3^6,\ x_1^7 x_2 - x_3^7 x_4 \rangle \lhd \Bbbk[x_1, x_2, x_3, x_4]\,. \tag{2.4}$$

In the given coordinates, the three generators define already the reduced Gröbner basis for the degree reverse lexicographic order. Thus one might expect that $\operatorname{reg}\mathcal{I} = 8$. However, if we swap two of the coordinates and consider $\mathcal{I}$ as an ideal in $\Bbbk[x_1, x_3, x_2, x_4]$, we obtain a completely different Gröbner basis for the corresponding degree reverse lexicographic order:

$$\Big\{ x_2^7 - x_1 x_3^6,\ x_1^7 x_2 - x_3^7 x_4,\ x_1^8 - x_2^6 x_3 x_4,\ x_1^6 x_2^8 - x_3^{13} x_4,\ x_1^5 x_2^{15} - x_3^{19} x_4,\ x_1^4 x_2^{22} - x_3^{25} x_4,\ x_1^3 x_2^{29} - x_3^{31} x_4,\ x_1^2 x_2^{36} - x_3^{37} x_4,\ x_1 x_2^{43} - x_3^{43} x_4,\ x_2^{50} - x_3^{49} x_4 \Big\}. \tag{2.5}$$

It seems to indicate that $\operatorname{reg}\mathcal{I} = 50$. However, none of the used coordinates are generic in the sense of Bayer and Stillman (1987a). In the original coordinates, the ideal $\mathcal{I}$ is in quasi-stable position. A Pommaret basis is obtained by adding to the Gröbner basis the six polynomials $x_1^k(x_2^7 - x_1 x_3^6)$ for $1 \leq k \leq 6$. According to Theorem 2.11, we thus have $\operatorname{reg}\mathcal{I} = 13$ and we can conclude that the result by Bayer and Stillman (1987a) does not even provide either a lower or an upper bound. Without a method to verify the genericity of the used coordinates, it is rather useless for concrete computations.

Although the definitions of the Janet and the Pommaret division, respectively, look very different, the two divisions are actually closely related (see (Gerdt, 2000) for a detailed discussion). One consequence of this is the following observation that is an immediate corollary to statements in (Seiler, 2009b, Sect. 2).

Proposition 2.13. *Assume that the polynomial ideal $\mathcal{I} \lhd \mathcal{S}$ is in quasi-stable position. Then any minimal Janet basis of $\mathcal{I}$ is also a Pommaret basis.*

2.3. Related algorithms

As extensively discussed in (Seiler, 2010) and references therein, there exists a direct algorithm – developed by Gerdt and Blinkov (1998a,b) – for computing involutive bases and many optimisations have been proposed for it (see e.g. (Gerdt, 2005) and references therein). Several implementations of various variants of the basic algorithm exist, but they are not as mature as current implementations of Gröbner bases. In our experiments, we therefore computed first reduced Gröbner bases and then completed only the leading terms to an involutive basis of the leading ideal. As one can see from results like Theorem 2.5, this is sufficient for most purposes.

As a consequence of this strategy, we discuss here only monomial completion algorithms. The basic idea of any involutive completion algorithm is to consider what happens if a generator is multiplied with one of its non-multiplicative variables. If the obtained term does not lie in the involutive span, it is added to the basis. Under modest assumptions, one can show correctness and termination of such algorithms; in fact, one can even show that if the input is a minimal generating set, then the output will be a minimal involutive basis (Seiler, 2010).

Algorithm 1: JanetCompletion.

Data: Minimal generating set $\mathcal{F}$ for the monomial ideal $\mathcal{J} \trianglelefteq \mathcal{S}$
Result: Minimal Janet basis of $\mathcal{J}$
1 **begin**
2 $\quad \mathcal{H} \longleftarrow \left\{ \left[f, \emptyset, \mathcal{X}_{J,\mathcal{F}}(f) \right] \mid f \in \mathcal{F} \right\}$ *// the second component contains the already treated non-multiplicative variables*
3 $\quad$ Sort $\mathcal{H}$ w.r.t. degree from the smallest to the largest one
4 $\quad$ **repeat**
5 $\qquad$ $flag \longleftarrow$ `true`
6 $\qquad$ **if** $\exists h \in \mathcal{H} \colon \{x_1, \dots, x_n\} \setminus \big(h[2] \cup h[3]\big) \neq \emptyset$ **then**
7 $\qquad\quad$ $x_i \longleftarrow$ first element of $\{x_1, \dots, x_n\} \setminus (h[2] \cup h[3])$
8 $\qquad\quad$ $h[2] \longleftarrow h[2] \cup \{x_i\}$; $\quad flag \longleftarrow$ `false`
9 $\qquad\quad$ **if** $x_i \cdot h[1]$ has no Janet divisor in $\left\{ h[1] \mid h \in \mathcal{H} \right\}$ **then**
10 $\qquad\qquad$ $\mathcal{H} \longleftarrow \mathcal{H} \cup \left\{ \left[x_i \cdot h[1], \emptyset, h[3] \cap \{x_1, \dots, x_{i-1}\} \right] \right\}$
11 $\qquad\qquad$ For the elements $g \in \mathcal{H}$ such that $g[1]$ and $x_i \cdot h[1]$ have the same exponents in the variables $x_1, \dots, x_i$, update the multiplicative variables among the variables $x_i, \dots, x_n$ (the variable x_i must be checked only for $x_i \cdot h[1]$).
12 $\qquad\qquad$ Sort $\mathcal{H}$ w.r.t. degree from the smallest to largest one
13 $\quad$ **until** $flag$
14 $\quad$ **return** $\left\{ h[1] \mid h \in \mathcal{H} \right\}$

Algorithm 1 allows us to compute efficiently (minimal) Janet bases, although it does not explicitly use more sophisticated data structures like Janet trees (Gerdt et al., 2001). The key optimisations in this algorithm are that we keep track of the already considered non-multiplicative variables and that we do not compute the multiplicative variables from scratch after each change in the basis, but only perform some necessary adaptions. Experiments with 2000 random ideals show that such simple measures suffice to provide a rather efficient algorithm: compared with a naive completion algorithm, the runtimes were on average 10 times faster.

We do not provide a detailed proof of the correctness of Algorithm 1. It follows rather immediately from the definition of the Janet division that adding in Line 10 the new generator $x_i \cdot h[1]$ affects the assignment of multiplicative variables only for the generators considered in Line 11. The properties of an involutive division, in particular the so-called filter axiom, ensure that the addition of a further generator can only lead to smaller sets of multiplicative variables. One might worry that some earlier considered non-multiplicative prolongation could then lose its Janet divisor. But it is shown in (Seiler, 2010, Sect. 4.4) that this does not affect the correctness of the algorithm.

If the given monomial ideal is quasi-stable, then we can resort to a simpler algorithm. As the Pommaret division is global, i.e. the multiplicative variables associated to a term are independent of the remaining terms in the considered set, it is not necessary to manage multiplicative variables or to keep track of already considered non-multiplicative variables and we arrive at Algorithm 2 (in Line 4, any term order $\prec$ may be used). Correctness and termination is extensively discussed in (Seiler, 2010).

Hashemi et al. (2018) developed a deterministic approach to achieve quasi-stable (and related) position for arbitrary ideals based on *elementary moves*. These very sparse transformations generate the Borel group of lower triangular, non-singular matrices. Each elementary move is characterised by

Algorithm 2: PommaretCompletion.

Data: Finite generating set $\mathcal{F}$ of quasi-stable ideal $\mathcal{J} \trianglelefteq \mathcal{S}$
Result: Pommaret basis $\mathcal{H}$ of $\mathcal{J}$

begin
  $\mathcal{H} \longleftarrow \mathcal{F}$; $\mathcal{S} \longleftarrow \{x_i h \mid h \in \mathcal{H},\ i < \max h\}$
  while $\mathcal{S} \neq \emptyset$ **do**
    $s \longleftarrow \min_{\prec} \mathcal{S}$; $\mathcal{S} \longleftarrow \mathcal{S} \setminus \{s\}$
    if s has no Pommaret divisor in $\mathcal{H}$ **then**
      $\mathcal{H} \longleftarrow \mathcal{H} \cup \{s\}$; $\mathcal{S} \longleftarrow \mathcal{S} \cup \{x_i s \mid i < \max s\}$
  return $\mathcal{H}$

a pair of indices (i, j) with $1 \le i < j \le n$ and the move $\mu_{(i,j)}$ maps $x_j \mapsto x_i + x_j$ and leaves all other variables unchanged. Thus in $\mathcal{S}$ we have a total of $\frac{1}{2}n(n-1)$ different elementary moves.

Assume that the ideal $\mathcal{I}$ is not in quasi-stable position. Hence its leading ideal $\operatorname{lt}\mathcal{I}$ has at least one obstruction (x^μ, i) to quasi-stability. Hashemi et al. (2018) showed that if $\max x^\mu = j$, then the transformation $x_j \mapsto x_i + a x_j$ will remove this obstruction for almost any choice of $a \in \Bbbk$. We will always choose $a = 1$, i.e. apply the elementary move $\mu_{(i,j)}$. In "unlucky" situations, cancellations may allow the obstruction to persist. However, after finitely many iterations of $\mu_{(i,j)}$, it will always disappear.

These considerations lead to Algorithm 3 (in (Hashemi et al., 2018), it was given for the case of strongly stable position, but the adaption is trivial). For proving its termination and for its formulation, one needs the following ordering. Let $\mathcal{F}$ be an autoreduced finite set of polynomials and write $\mathcal{L}(\mathcal{F}) = (t_1, \dots, t_\ell)$ for the tuple of leading terms of $\mathcal{F}$ sorted such that $t_1 \prec_{\text{revlex}} \cdots \prec_{\text{revlex}} t_\ell$ for the purely reverse lexicographic order (not the *degree* reverse lexicographic order!). Given two such sets $\mathcal{F}$ and $\tilde{\mathcal{F}}$ with $\mathcal{L}(\mathcal{F}) = (t_1, \dots, t_\ell)$ and $\mathcal{L}(\tilde{\mathcal{F}}) = (\tilde{t}_1, \dots, \tilde{t}_{\tilde{\ell}})$, we define

$$\mathcal{F} \prec_{\mathcal{L}} \tilde{\mathcal{F}} \quad \Longleftrightarrow \quad \begin{cases} \exists j \le \min(\ell, \tilde{\ell}) \colon \big(\forall i < j \colon t_i = \tilde{t}_i\big) \wedge t_j \prec_{\text{revlex}} \tilde{t}_j \text{ or} \\ \big(\forall j \le \min(\ell, \tilde{\ell}) \colon t_j = \tilde{t}_j\big) \wedge \ell < \tilde{\ell} \,. \end{cases} \tag{2.6}$$

Algorithm 3: QSPos – Quasi-Stable Position.

Data: Reduced Gröbner basis $\mathcal{G}$ of homogeneous ideal $\mathcal{I} \lhd \mathcal{S}$
Result: Linear change of coordinates Ψ such that $\operatorname{lt}\Psi(\mathcal{I})$ is quasi-stable

1 **begin**
2   $\Psi \longleftarrow \mathrm{id}$; $\mathcal{F} \longleftarrow \mathcal{G}$
3   **while** *obstruction to quasi-stability of* $\langle \operatorname{lt}\mathcal{F}\rangle$ *exists* **do**
4   choose elementary move ψ related to obstruction; $\Psi \longleftarrow \psi \circ \Psi$
5     $\tilde{\mathcal{F}} \longleftarrow$ `ReducedGröbnerBasis`$\big(\psi(\mathcal{F})\big)$
6   **while** $\mathcal{F} \succeq_{\mathcal{L}} \tilde{\mathcal{F}}$ **do**
7       $\Psi \longleftarrow \psi \circ \Psi$
8       $\tilde{\mathcal{F}} \longleftarrow$ `ReducedGröbnerBasis`$\big(\psi(\tilde{\mathcal{F}})\big)$
9     $\mathcal{F} \longleftarrow \tilde{\mathcal{F}}$
10   **return** Ψ

The strategy behind Algorithm 3 is quite simple. As long as obstructions exist, we apply a related elementary move. Generically, this move will eliminate at least one obstruction; in rare cases we may have to iterate the move. In (Hashemi et al., 2018), it is shown that a given ideal $\mathcal{I}$ possesses modulo linear coordinate transformations only finitely many leading ideals and that after an elementary move related to an obstruction a polynomial set can never descend with respect to the ordering $\prec_{\mathcal{L}}$. This ensures the correctness and termination of the algorithm.

Like many algorithms in commutative algebra, Algorithm 3 is not completely specified: in Line 4, it is not said how the next elementary move should be chosen. In general, there are several possibilities here. Although the choice does not affect the correctness and the termination of the algorithm, it has

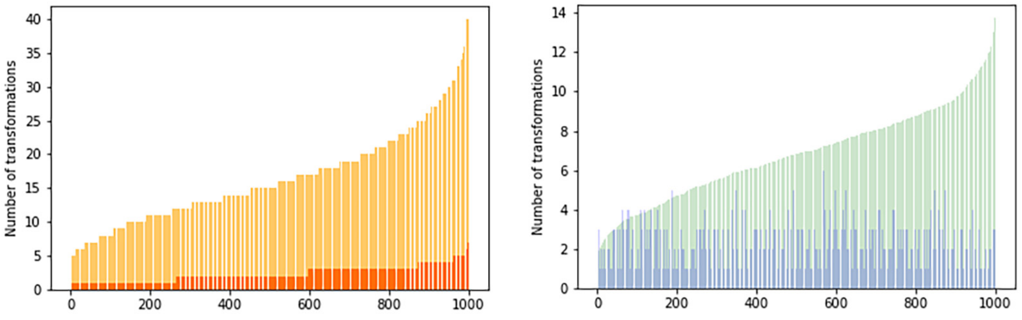


Fig. 1. Performance of random choices. Left: comparison of minimal and maximal number of transformations needed to reach quasi-stable position. Right: Comparison of average number of transformations required by random strategy (green) with democratic strategy (blue). (For interpretation of the colours in the figure(s), the reader is referred to the web version of this article.)

an effect on the efficiency. A coarse measure for the efficiency is the number of elementary moves required until quasi-stable position is reached.

So far, only very elementary strategies were used for choosing a move based on the obstructions to quasi-stability. A Maple implementation of the algorithm from (Hashemi et al., 2012) just looks for the first obstruction it detects and then uses the move suggested by it (thus the choice depends on the way how the search for obstructions is implemented). In other (unpublished) works, we experimented with strategies like taking an obstruction of minimal degree or an obstruction related to a generator with a minimal number of multiplicative variables. However, for any of these strategies, one can easily design examples where they work badly.

We now propose a slightly more sophisticated strategy, the *"democratic" strategy*, which considers all obstructions instead of just one. We note for each existing obstruction of quasi-stability for which elementary move it "votes", i.e. to which move it corresponds; the move which gets the most votes is taken. For estimating the influence of good or bad choices, we compared it in a preparatory experiment with a *random strategy*: in each iteration of the outer `while` loop of Algorithm 3, we randomly pick an elementary move.

We applied both strategies to 1,000 ideals (these were a sample of 10% of a large test set of random ideals described in Section 3.3 below). The random strategy was applied to each ideal 10 times. The outcome of the experiment is shown in the two plots in Fig. 1. The plot on the left shows for each ideal the minimal and the maximal number of transformations needed to reach a quasi-stable position using random choices. Obviously, the numbers differ significantly: the maximum is almost 10 times as large as the minimum in the worst cases and even in the best cases the numbers differ by a factor of five. The plot on the right compares the average number of transformations in a random strategy with the number required by the democratic strategy. The latter one needed typically between one and three elementary moves and thus was much more efficient. These observations clearly indicate that it is worthwhile thinking about good ways to perform the choice in Algorithm 3.

Remark 2.14. It seems natural to consider only elementary moves which are related to obstructions, i.e. which obtain in the democratic strategy at least one vote. All human strategies we know work this way. However, in the systematic analysis of a large set of random ideals (see Section 3.3 below), we found cases where the optimal choice is not related to any obstruction. However, such cases are fairly rare: we observed this phenomenon only for about 6% of the considered ideals.

It follows from Remarks 2.3 and 2.4 that all problems mentioned in the title can be simultaneously solved by achieving quasi-stable position (for a system of parameters one needs in addition the Pommaret basis). Therefore, we will consider in the sequel only this problem. If the goal is a Noether position, then this may lead to unnecessary transformations, as it is weaker than quasi-stable position. Hashemi et al. (2019) constructed a special involutive division, called D-Noether division, such that a finite Noether basis exists, if and only if the ideal is in Noether position. In principle, one could adapt the approach presented here to this division and thus obtain slightly more efficient computations. As bringing an ideal into a quasi-stable position has so many further benefits, we refrain from detailing such an adaption.

3. Machine learning quasi-stable position

We will now show how methods from supervised machine learning can successfully be applied to selecting the "right" elementary move in Line 4 of Algorithm 3. We will apply a greedy approach: the models are trained to estimate which elementary move will lead to the largest Pommaret span. Following ideas used by England and collaborators for machine learning good variable orders for cylindrical algebraic decompositions (see the references in the Introduction), we will consider the choice of moves as a *multi class classification problem*: each class corresponds to one possible elementary move so that we have $\frac{1}{2}n(n-1)$ different classes.

In our study, we will compare the following five well established and much used classification models:

- k Nearest Neighbours (kNN)
- Support Vector Machine (SVM)
- Decision Tree (DT)
- Multilayer Perceptron (MLP)
- Logistic Regression (LR)

Details about them can be found in most textbooks on machine learning; see e.g. (Aggarwal, 2015) or (Bishop, 2006). We emphasise that with the exception of the multilayer perceptron, none of them is based on a neural network. This means that training is comparatively cheap and less data are needed. In particular the latter point is of some relevance, as we will discuss below.

All our computations were done in Python using the Scikit-learn library (Pedregosa et al., 2011). For polynomial computations, the Python based computer algebra system SageMath[3] was used. This concerns in particular the determination of Gröbner bases. For the extension to Janet or Pommaret bases, we implemented the monomial algorithms presented in Section 2.3 in SageMath.

3.1. Feature selection

For most machine learning models, the data must be mapped into a *feature space* of fixed dimension. This fact excludes the typical computer algebra approach of using a generating set as input: as the number of generators and the number of terms in each generator vary widely from ideal to ideal, generating sets cannot be interpreted as elements of a feature space. We could also observe in experiments that typical algebraic invariants like dimension, depth or regularity are not relevant for deciding which elementary move to choose. Therefore, we omit them in our feature vectors.

Our feature vectors contain mainly what we call *statistical data* about the given generating set. This includes e.g. information about degrees (total or in individual variables) or the distribution of the variables over all terms or over the leading terms. In addition, we incorporate a *transformation part* which encodes for each elementary moves how many obstructions to quasi-stability vote for it. This makes the length of the feature vector independent of the size of the generating set, but it still depends on the number n of variables in the underlying polynomial ring $\mathcal{S}$ so that we must fix n. All experiments reported in this work have been done for $n=4$. For smaller values of n, it is not so hard to construct by hand good coordinates. For larger values of n, the computational costs rapidly increase (both for preparing the training data, but also for the learning, as the number of our features grows exponentially with n). Hence $n=4$ is a good choice for first experiments with different machine learning models, although $n=5$ or $n=6$ definitely appear manageable.

Table 1 lists $3\cdot 2^n+6n+8$ statistical features for a polynomial ring with n variables. The second column describes the number of features each row defines. For instance, in the fourth row we have three values for each variable which gives a total of $3n$ features. For explaining the third column, we

[3] https://www.sagemath.org/.

Table 1
Used statistical features.

Description	#	Formula
Number of generators	1	k
total number of terms	1	$\sum_i T_i$
Min/max/average total degree of generators	3	$\min_i \sum_{j=1}^n d_{i,t}^j$, $\max_i \sum_{j=1}^n d_{i,t}^j$ and $\mathrm{av}_i \sum_{j=1}^n d_{i,t}^j$
Min/max/average degree in single variable of generators	$3n$	$\min_i d_{i,t}^j$, $\max_i d_{i,t}^j$ and $\mathrm{av}_i\, d_{i,t}^j$ for $j \in \{1,\dots,n\}$
Min/max/average degree in single variable of leading terms	$3n$	$\min_i D_i^j$, $\max_i D_i^j$ and $\mathrm{av}_i\, D_i^j$ for $j \in \{1,\dots,n\}$
Min/max/average number of terms in generators	3	$\min_i T_i$, $\max_i T_i$ and $\mathrm{av}_i\, T_i$
Number of generators containing certain variables	$2^n - 1$	$\sum_i \mathrm{sgn}\left(\sum_t \prod_{x_j \in \bar{\mathcal{X}}} \mathrm{sgn}\,(d_{i,t}^j)\right)$ for $\emptyset \neq \bar{\mathcal{X}} \subseteq \mathcal{X}$
Number of terms containing certain variables	$2^n - 1$	$\sum_{i,t} \prod_{x_j \in \bar{\mathcal{X}}} \mathrm{sgn}\,(d_{i,t}^j)$ for $\emptyset \neq \bar{\mathcal{X}} \subseteq \mathcal{X}$
Number of leading terms containing certain variables	$2^n - 1$	$\sum_i \prod_{x_j \in \bar{\mathcal{X}}} \mathrm{sgn}\,(D_i^j)$ for $\emptyset \neq \bar{\mathcal{X}} \subseteq \mathcal{X}$
Sum of total degrees of generators	1	$\sum_i \deg(f_i)$
Number of pure variable powers among terms	1	$\sum_i T_i - \sum_{i,t,j} \mathrm{sgn}\left(\lvert d_{i,t}^j - \deg(f_i^t)\rvert\right)$
Number of pure variable powers among leading terms	1	$k - \sum_{i,t,j} \mathrm{sgn}\left(\lvert D_{i,t}^j - \deg f_i\rvert\right)$

need the following notations. Let $\mathcal{F} = \{f_1, \dots, f_k\} \subset \mathcal{S}$ be the considered finite generating set of an ideal $\mathcal{I}$. We can write each polynomial as

$$f_i = \sum_{t=1}^{T_i} c_{i,t} x_1^{d_{i,t}^1} \cdots x_n^{d_{i,t}^n}, \quad i \in \{1, \dots, k\}. \tag{3.1}$$

Here T_i is the number of terms of f_i. We also write f_i^t for the term t in f_i. For the leading term, we use capital letters:

$$\mathrm{lt}\, f_i = C_{i,t} x_1^{D_{i,t}^1} \cdots x_n^{D_{i,t}^n}. \tag{3.2}$$

In some rows, we used the sign function to count the number of nonzero exponents and denoted by av the arithmetic mean of some values.

As one can see, this statistical part of the feature vector provides information about how the variables are distributed over all the terms, over the different generators and over the leading terms, as such information are crucial for estimating how different possible transformations may affect the size of the Pommaret span (and the sparsity) of the transformed set.

In addition, we have a smaller *transformation part* in the feature vector consisting of $\frac{1}{2}n(n-1)$ entries: we store for each elementary move how many obstructions vote for it (the democratic strategy is based exclusively on these values). The total number of features is thus $3 \cdot 2^n + \frac{1}{2}(n^2 + 11n) + 8$ which leads for $n = 4$ to 86 features.

England and Florescu (2019) used 11 similar statistical features in their work on learning good variables orderings for cylindrical algebraic decomposition. Later, they proposed in (Florescu and England, 2019) a brute force approach to automatically generate new features which provided them with almost 2,000 features (for $n = 3$ variables) out of which they extracted with a statistical variance analysis 78 relevant and independent ones. We believe that we understand our problem sufficiently well to select the relevant features by hand and refrained from such an automatised approach. The results presented below seem to indicate that this belief is justified.

3.2. Scoring

Given a finite polynomial set $\mathcal{F} \subset \mathcal{S}$ generating an ideal $\mathcal{I} \lhd \mathcal{S}$, our final goal is to put $\mathcal{I}$ into quasi-stable position and to obtain a Pommaret basis $\mathcal{H}$ of $\mathcal{I}$. We first compute a reduced Gröbner basis $\mathcal{G}$ out of $\mathcal{F}$ so that $\mathrm{lt}\,\mathcal{G}$ is the minimal generating set of $\mathrm{lt}\,\mathcal{I}$. For determining the transformation part of the feature vector, we need the obstructions to quasi-stability of $\operatorname{lt}\mathcal{G}$. If no obstructions exist, $\mathcal{I}$ is already in quasi-stable position and we can determine a Pommaret basis (of its leading ideal) with Algorithm 2.

If obstructions exist, then we need a more quantitative measure for assessing how far away from quasi-stable position we are. Taking $\operatorname{lt} G$ as input for Algorithm 1, we determine a minimal Janet basis $\mathcal{H}$ of $\operatorname{lt}\mathcal{I}$ and read off from it the volume polynomial $V_{\mathcal{I}}$ via Lemma 2.9. With the same lemma, we can also compute the volume polynomial $V_{\langle\mathcal{H}\rangle_P}$ of the Pommaret span of $\mathcal{H}$. In quasi-stable position, these two polynomials are identical, as then the Janet basis is simultaneously a Pommaret basis by Proposition 2.13 and thus $\langle\mathcal{H}\rangle_P = \operatorname{lt}\mathcal{I}$. Otherwise, we have for all sufficiently large degrees q that $V_{\langle\mathcal{H}\rangle_P}(q) < V_{\mathcal{I}}(q)$.

The formula in Lemma 2.9 expresses the volume polynomials as a sum of binomial coefficients. But it is of course trivial to expand these explicitly into the standard form of a univariate polynomial

$$V(q) = v_{n-1}q^{n-1} + v_{n-2}q^{n-2} + \cdots + v_1 q + v_0 \tag{3.3}$$

and it also follows immediately from Lemma 2.9 that the maximal degree of any volume polynomial is $n-1$. We associate with each volume polynomial written in the form (3.3), its coefficient vector $\mathfrak{v} = (v_{n-1}, \ldots, v_1, v_0) \in \mathbb{Q}^n$. Given two volume polynomials, we can compare their coefficient vectors using a lexicographic ordering. The volume polynomial with the larger vector will have the stronger asymptotic growth, i.e. from some degree on it will always produce larger values. For example, if we are not in quasi-stable position, then $\mathfrak{v}_{\langle\mathcal{H}\rangle_P} \prec_{\text{lex}} \mathfrak{v}_{\mathcal{I}}$.

Based on this observation, we can compare the effect of different elementary moves. Let $\mu_{(i,j)}$ and $\mu_{(k,\ell)}$ be two moves. Applying each of them to the generating set $\mathcal{F}$, we obtain two new polynomial sets $\mathcal{F}_{(i,j)}$ and $\mathcal{F}_{(k,\ell)}$, respectively. Out of them, we compute first reduced Gröbner bases $\mathcal{G}_{(i,j)}$ and $\mathcal{G}_{(k,\ell)}$ and then complete their leading terms to Janet bases $\mathcal{H}_{(i,j)}$ and $\mathcal{H}_{(k,\ell)}$. If now $\mathfrak{v}_{\langle\mathcal{H}_{(i,j)}\rangle_P} \prec_{\text{lex}} \mathfrak{v}_{\langle\mathcal{H}_{(k,\ell)}\rangle_P}$, then we consider the elementary move $\mu_{(k,\ell)}$ as the better choice in Algorithm 3.

Obviously, it is very costly to analyse all possible elementary moves in this way, in particular for a larger number n of variables. One may say that we try to bypass this expensive computation by using machine learning. The task of the different models is to predict the best move purely on the basis of the above described features – without actually applying any transformation and without computing any Gröbner basis.

3.3. Producing training data

A fundamental problem in the application of tools from machine learning to commutative algebra is the lack of a sufficiently large repository of polynomial ideals. We are aware of only two publicly available collections: the home page of Jan Verschelde[4] with well over 1oo ideals and the old POSSO test suite[5] with a bit more than 30 ideals. The former one is geared towards the numerical solution of polynomial systems; the latter one comprises examples which are considered as hard for Gröbner bases computations. Thus none of them corresponds exactly to the application we have in mind: computations in algebraic geometry. If one eliminates all ideals in these collections which are not in four variables, have floating point coefficients or are already in quasi-stable position, then there remain only five ideals: *cohn2, cohn3* (Verschelde), *bronstein2, gerdt3, gerdt6* (POSSO). The training of a typical machine learning model will require at least a few thousand examples. Hence we must resort to *random ideals*. It is well known that the properties of polynomial ideals appearing in applications often differ from those of random ideals, but we are not aware of any alternative.

Instead of relying on some built-in functions of SageMath, we wrote our own random generator for homogeneous ideals to have full control over all relevant parameters like degrees, number of generators or number of terms. It uses a *Poisson distribution* for choosing these parameters. Hence like in many applications, most generators have rather small degrees and a low number of terms, but some generators may have fairly high degrees or a larger number of terms.

4 https://homepages.math.uic.edu/~jan/.
5 https://www-sop.inria.fr/saga/POL/BASE/3.posso/.

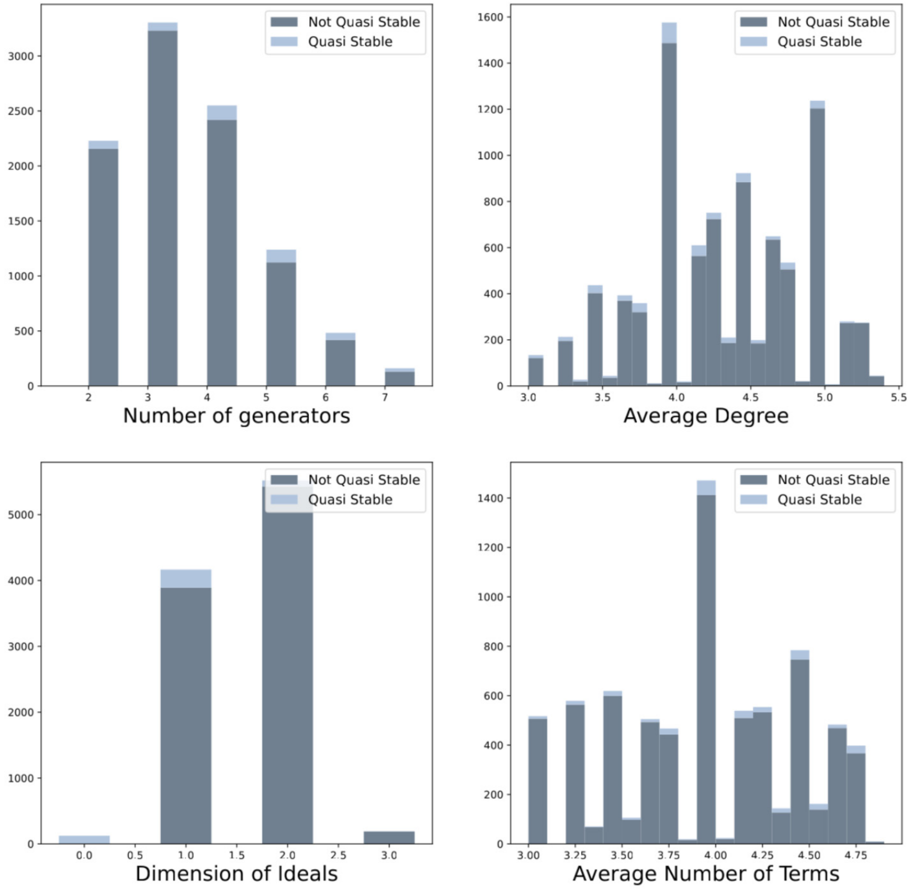


Fig. 2. Some statistics of the 10,000 generated random ideals and their generating sets.

An important goal was that most generated ideals are not in quasi-stable position, as only these are useful for our purposes. Furthermore, the dimension of the ideals is important. Any zero-dimensional ideal is automatically in quasi-stable position and for ideals of dimension $n-1$, i.e. hypersurfaces, it is usually easy to find by hand good coordinates. Thus we chose the parameters of the Poisson distributions in such a way that many ideals of intermediate dimensions are produced.

With this random generator, we produced $10,000$ ideals in a polynomial ring with 4 variables over the rational numbers. About 95% of them, $9,509$ to be precise, were not in quasi-stable position and we only used these ideals for our experiments. The histograms in Fig. 2 show some key properties of the generated polynomial sets and their ideals. The first histogram of the number of generators shows the typical shape of a Poisson distribution with most of the sets containing two to four generators. The second and the fourth histogram depicts the average degree and the average number of terms in a generator. The third histogram plots the dimensions of the generated ideals. It shows that most ideals have their dimension in the desired intermediate range.

Remark 3.1. We used the occasion of having so many ideals available to do some statistics on their properties and on their bases. More precisely, we were interested in comparing the number of generators and the maximal degree of a generator in the original generating set, in the reduced Gröbner basis and in the Janet basis as an empirical way to assess the practical complexity of the latter two types of bases.

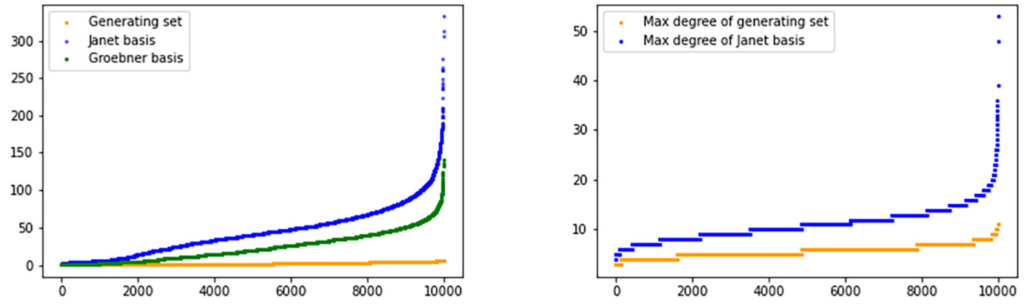


Fig. 3. Sizes (left) and maximal degrees (right) of the original generating sets, the reduced Gröbner bases and the Janet bases of the random ideals.

The left diagram in Fig. 3 concerns the sizes of the different types of generating sets. Each curve contains one point for each of the 10,000 random ideals and each curve is separately sorted by size. It thus makes no sense to compare individual points, as they will typically belong to different ideals. One clearly sees that for the vast majority of ideals the size of the Gröbner and Janet bases, respectively, grow only moderately; only for a very small fraction of the ideals large bases with more than one hundred elements occur. As Janet bases are non-reduced Gröbner bases, they contain of course more generators, but on average they seem to be larger by a relatively modest factor.

The right diagram in Fig. 3 shows the maximal degree of a generator; we omitted here a curve for the reduced Gröbner bases, as it is indistinguishable from the one for the Janet bases. As already mentioned, Bayer and Stillman (1987a) proved that generically the maximal degree of a Gröbner basis with respect to the degree reverse lexicographic order is the Castelnuovo–Mumford regularity of the ideal. But as we demonstrated in Example 2.12, this degree is neither a lower nor an upper bound of the regularity. In (Albert et al., 2015), it is shown that the degree of a Janet basis is always at least the Castelnuovo–Mumford regularity and thus represents an upper bound. The diagram confirms a well-known empirical fact: although the Castelnuovo–Mumford regularity may grow double exponentially with the number n of variables and the degree d of the generating set, this rarely occurs in practise. In other words, our random ideals exhibit here again the same behaviour as ideals appearing in applications.

Hashemi et al. (2021, Ex. 7.7) exhibited a family of ideals in an arbitrary number of variables such that the difference between the degree of the Janet basis and the Castelnuovo–Mumford regularity can become arbitrarily large (based on an example in three variables from (Albert et al., 2015)). The benchmarks presented in (Albert et al., 2015) indicate, however, that in practise large differences are rare. Our 10,000 random ideals confirm this observation: 5,863 have a Janet basis with exactly the same degree as the Gröbner basis, 4,081 Janet bases have a degree which is one higher and 56 a degree which is two higher; larger differences do not occur. Thus, although the Janet bases are typically considerable larger, the additional generators do not increase noticably the maximal degree.

To each of the not quasi-stable ideals, we applied each of the six possible elementary moves and determined which move yields the largest Pommaret span in the sense that the coefficient vector of the volume polynomial is maximal for the lexicographic ordering. We sorted the elementary moves as $(2,1)$, $(3,1)$, $(3,2)$, $(4,1)$, $(4,2)$ and $(4,3)$ and labelled the corresponding classes as $0,1,\dots,5$. Fig. 4 shows how the ideals distribute over the six different classes. Obviously, the distribution is very unbalanced: almost half of the ideals belong to class 3, whereas the classes 0 and 2 are very rare. As a consequence, we used for the determination of the hyperparameters of the different models a *stratified* cross-validation. It is an interesting question whether this unevenness is an artefact of working with random ideals or whether it can also be observed in ideals from applications.

The prevalence of class 3 is not surprising, but a simple combinatorial fact. The total number of terms of degree q in n variables is given by $\binom{n+q-1}{q}$; the number of terms x^μ with $\max x^\mu = k \le n$ among these is $\binom{k+q-2}{q-1}$. Thus terms with $\max x^\mu = 4$ (and only one multiplicative variable) are the

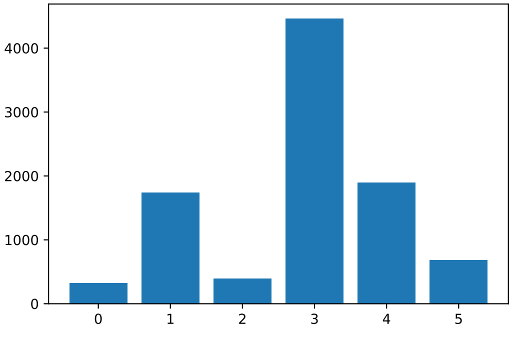


Fig. 4. Distribution of classes in data set.

Table 2
Hyperparameters determined by a grid search with 5-fold stratified cross-validation.

Model	Hyperparameter	Value
k-NN	k	24
	Weights	Uniform
	Algorithm	KDtree
SVM	Regularisation param. C	0.1
	Kernel	Linear
DT	Criterion	Gini impurity
	Maximum depth	5
MLP	Hidden layer size	100
	Activation function	Logistic sigmoid
	Optimizer	Stochastic gradient descent
	Regularisation param. α	1.0
LR	Class	Multinomial
	Max. iterations	1.000
	Penalty	L2

most frequent ones. For $n = 4$ half of the terms of degree 3 have only one multiplicative variable; at degree 6 this is the case for two thirds of the terms and at degree 9 even three quarters. The move $(4, 1)$ replaces every term $x_1^{\mu_1} x_2^{\mu_2} x_3^{\mu_3} x_4^{\mu_4}$ with $\mu_4 > 0$ by the term $x_1^{\mu_1+\mu_4} x_2^{\mu_2} x_3^{\mu_3}$ having at least two multiplicative variables (plus some other terms). Thus it possesses a very high chance of having a significant impact on the volume function of the Pommaret span.

3.4. Experimental results

We tuned the hyperparameters of the different machine learning models by a grid search with a 5-fold stratified cross-validation. We first put 1,902 ideals aside as holdout set. Then we partitioned the remaining ideals into five sets of equal size taking care that the distribution of the ideals over the different classes remains the same in all sets. Then always four sets were used for training and one set for validating. The best values obtained for the various hyperparameters of the different models in a grid search are shown in Table 2.

As the different models depend on different numbers of hyperparameters and also the possible ranges of values for the various hyperparameters can be very different (sometimes a finite number of discrete values, sometimes a real interval), it makes no sense to directly compare the computation times for the determination of the hyperparameters. We only mention that the multilayer perceptron required by far the longest time and that the decision trees were the fastest model in this respect.

Table 3
Comparison of the results for the different models and strategies.

Model	k-NN	SVM	DT	MLP	LR	Demo.	Rand.
Test accuracy	0.75	0.89	0.78	0.81	0.85	0.42	0.17
M. av. precision	0.74	0.86	0.68	0.63	0.88	0.42	0.17
S. av. precision	0.74	0.89	0.75	0.77	0.88	0.60	0.17
M. av. recall	0.55	0.84	0.55	0.58	0.85	0.37	0.16
S. av. recall	0.75	0.89	0.78	0.81	0.88	0.42	0.17
Train. time (sec.)	7.87	11.19	0.37	31.92	11.18	517.12	0.24

For assessing the quality of the models defined by the obtained hyperparameters, we computed for each model a confusion matrix based on the holdout set. All confusion matrices can be found in Appendix A. Table 3 reports for each model the accuracy, the precision and the recall. The classwise precision and recall can be found in the borders of our confusion matrices; the numbers in Table 3 are the macro and the sample weighted averages of the classwise ones.

The accuracy is a standard tool for assessing the quality of classification methods, but it can be misleading in unbalanced data sets as ours. In our experiments, however, it turned out that other tools like precision and recall lead essentially to the same results. But as they are computed classwise, they provide deeper insight into the performance of the different models.

We observed accuracies between 75% (k-NN) and 89% (SVM). For comparison: England and Florescu (2019) reported for their use of machine learning in the context of cylindrical algebraic decomposition accuracies around 60% for all tested models. This seems to indicate that our problem is very well suited for the use of machine learning. However, one should take into account that because of the imbalance within our data set, always answering class 3 yields already an accuracy of almost 50%.

It is interesting to note that for the recall, one can observe for some models considerable differences between the macro average and the weighted average. In the former one, all classes are considered as equally important and the multilayer perceptron and decision trees had considerable problems in detecting the rare classes 0 and 2, whereas the other models showed at much more homogeneous behaviour (which in the case of k-nearest neighbours means homogeneously not so great results). Thus the “winners” are the support vector machine and logistic regression with a very similar performance.

In the last two columns of Table 3, we contrast the observed values with a human strategy – the democratic strategy mentioned above – and a simple random approach. For the random approach, we used a uniform random generator, as in real application situations the unbalanced distribution of the classes shown for our data in Fig. 4 is not known. Somewhat surprisingly, we observed for random choices nevertheless an accuracy of 17% which is exactly the value one would expect for a uniform distribution. Not surprising is the fact that this is by far the lowest value in the table. The democratic strategy as one of the most natural human strategies achieved only an accuracy of 42% which is even worse than always saying class 3 and far inferior to any used machine learning model. Thus this human strategy cannot be considered as very successful.

The last row in Table 3 provides for each model and the two other strategies the training time. For the machine learning models, this means the time needed to train the model after the hyperparameters have been tuned. For the two other strategies, we give the time to handle all ideals in the our data set. Trivially, the random strategy is the fastest. As the democratic strategy requires the determination of all obstructions, it is by far the slowest. Among the machine learning models, the decision tree is the fastest model to train and the multilayer perceptron the slowest model; the other three models are at roughly the same level.

A comparison based on measures like accuracy, precision, recall or specificity is “binary”: if the model does not predict the *optimal* move, its answer is considered as false. However, it might be that the move selected by the model has almost the same effect on the Pommaret span as the optimal one. We therefore computed for each model for each ideal where it did not predict the optimal move a “volume ratio”, i.e. the ratio $V_{\mathrm{pred}}(\bar{q})/V_{\mathrm{opt}}(\bar{q})$ where V_{pred} denotes the volume polynomial of the Pommaret span after the predicted move, V_{opt} the volume polynomial of the Pommaret span after

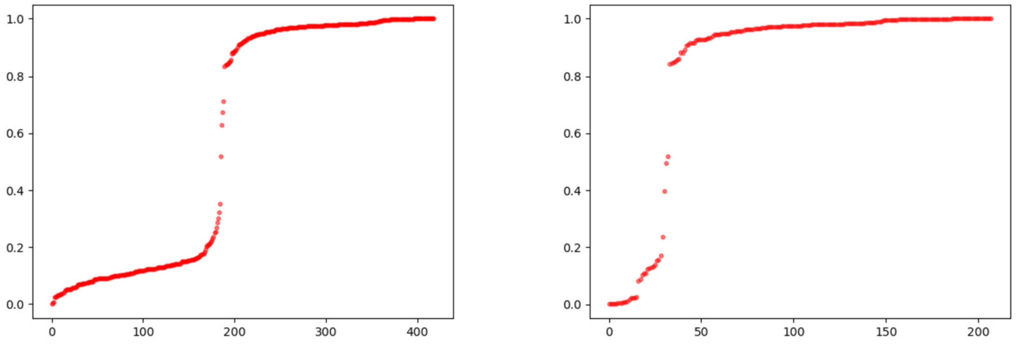


Fig. 5. Volume ratios for two models: decision tree (left) and support vector machine (right).

the optimal move and the degree $\bar{q}$ was chosen as ten times the degree of the Janet basis of the original ideal, i.e. so high that it approximates well the asymptotic behaviour. As one can clearly see in Fig. 5, in many cases the predicted move was actually not much worse than the optimal move with a volume ratio above 90%. For the support vector machine the volume ratio was really bad for less than 50 ideals out of almost 2,000. By comparison, the decision tree classified about 200 ideals really badly, i.e. more than four times as many. Without defining a precise criterion, we consider a classification as "really bad", if the volume ratio is below 80%, as in both diagrams one observes at roughly this value a steep drop.

3.5. Complete determination of quasi-stable position

So far, we have only been concerned with one intermediate step of Algorithm 3, the choice of the next elementary move in Line 4. In principle, it is straightforward to embed machine learning into the algorithm: in Line 4, the choice of the next move is done by a trained model. Note, however, the following difference: in Algorithm 3, the next move is chosen among all moves related to an obstruction, whereas our machine learning models always consider *all* possible elementary moves. As discussed in Remark 2.14, in a small number of cases the optimal move is not related to an obstruction and hence it appears useful to allow also such moves. On the other hand, the termination proof in (Hashemi et al., 2018) does not take such moves into account.

In our experiments, it turned out that these moves may indeed lead to termination problems. We encountered surprisingly often situations where only a very small number of elementary moves was related to an obstruction, but the selected machine learning model proposed another move. Unfortunately, this move did not change the leading ideal and thus the algorithm ran into an infinite loop. We therefore designed Algorithm 4 which is modified in two respects. (i) If only one elementary move is related to an obstruction, it always applies this move without asking the machine learning model. This reflects that in such a situation one can expect that after this move a quasi-stable position will be reached and thus the question of choosing a move does not arise. (ii) If the machine learning model proposes a move which is *not* related to an obstruction, then the algorithm uses this move only tentatively. This means that it checks afterwards whether the move has led to a new set of leading terms which is larger than the old one with respect to the ordering $\prec_{\mathcal{L}}$ introduced in (2.6). Only if this is the case, it continues with the transformed set. Otherwise, it rejects the move and instead uses the democratic strategy to choose the next move. This modification makes the algorithm consistent with the termination proof in (Hashemi et al., 2018) – which is based on producing a sequence of leading ideals which is strictly increasing with respect to the ordering $\prec_{\mathcal{L}}$ – and thus ensures termination of the adapted algorithm.

We applied Algorithm 4 to the 9,509 test ideals not in quasi-stable position and counted for each how many elementary moves were necessary to reach a quasi-stable position. In this experiment, we used the support vector machine as a machine learning model, as it has almost the same accuracy as the multilayer perceptron, but requires less computation time. Fig. 6 shows a histogram depicting

Algorithm 4: QSPosML – Quasi-Stable Position with ML.

Data: Reduced Gröbner basis $\mathcal{G}$ of homogeneous ideal $\mathcal{I} \lhd \mathcal{S}$
Result: Linear change of coordinates Ψ such that $\operatorname{lt}\Psi(\mathcal{I})$ is quasi-stable

begin
  $\Psi \longleftarrow \mathrm{id}$; $\mathcal{F} \longleftarrow \mathcal{G}$
  $\mathcal{M} \longleftarrow \big\{$elem. moves related to obstr. to quasi-stability of $\langle \operatorname{lt} \mathcal{F} \rangle\big\}$
  while $\mathcal{M} \neq \emptyset$ **do**
    if $|\mathcal{M}| = 1$ **then**
      $\psi \longleftarrow \mathcal{M}[1]$;
    else
      let ML model choose elementary move ψ
    $\tilde{\mathcal{F}} \longleftarrow$ `ReducedGröbnerBasis`$\big(\psi(\mathcal{F})\big)$
    if $(\psi \notin \mathcal{M}) \wedge (\mathcal{F} \succeq_{\mathcal{L}} \tilde{\mathcal{F}})$ **then**
      $\psi \longleftarrow$ move in $\mathcal{M}$ with most votes
      $\tilde{\mathcal{F}} \longleftarrow$ `ReducedGröbnerBasis`$\big(\psi(\mathcal{F})\big)$
    $\Psi \longleftarrow \psi \circ \Psi$
    while $\mathcal{F} \succeq_{\mathcal{L}} \tilde{\mathcal{F}}$ **do**
      $\Psi \longleftarrow \psi \circ \Psi$
      $\tilde{\mathcal{F}} \longleftarrow$ `ReducedGröbnerBasis`$\big(\psi(\tilde{\mathcal{F}})\big)$
    $\mathcal{F} \longleftarrow \tilde{\mathcal{F}}$
    $\mathcal{M} \longleftarrow \big\{$elem. moves related to obstr. to quasi-stability of $\langle \operatorname{lt} \mathcal{F} \rangle\big\}$
  return Ψ

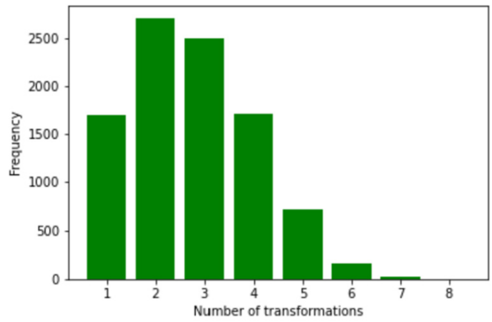


Fig. 6. Number of transformations required to achieve a quasi-stable position.

how many ideals needed how many transformations to reach a quasi-stable position. 90% of the ideals require at most four transformations with 55% needing two or three; more than eight transformations were never necessary. This indicates that most of our test ideals are not far away from a quasi-stable position. We also monitored the effect of the modifications introduced in the design of Algorithm 4. About two thirds of the ideals reached a situation where only one elementary move was related to an obstruction; one may conjecture that this typically happens for the last transformation. Given the observation from Remark 2.14 that moves not related to obstructions are rarely optimal, it is surprising that it happened also for about two thirds of the ideals that the support vector machine proposed in at least one iteration such a move. But on average only about every fourth such move could be accepted.

4. Conclusions

Our preliminary experiments already indicate that the problem of obtaining a quasi-stable position is well suited for the use of machine learning models and definitely better than the coarse human heuristics used so far. This is not very surprising, as the latter ones are based only on a rather superficial analysis of the leading terms, whereas our feature vector takes the complete support of all generators into account.

Of course, the results presented here still have to be confirmed by further experiments with polynomials in a larger number of variables, say $n = 5$ and $n = 6$. Such experiments will also provide some information on how our approach scales with n. It is clear that both the preparation of the training data (more precisely the scoring of the data which involves several Gröbner bases) and the determination of the hyperparameters will become significantly more expensive with an increasing number of variables and features, but we believe that at least $n = 5$ and $n = 6$ should still be managable and we are working on it. Once the models are trained, the costs of applying them should be neglectable compared to the costs of computing a Gröbner basis.

In this context, it is also of interest to study how much training data is really necessary. The number of 10,000 random ideals was basically chosen ad hoc. It could well be that 1,000 or 2,000 ideals might have also produced satisfactory models with considerable less computational costs. One should also expect that different algorithms require different amounts of training data, an important aspect (in particular for higher values of n) which can serve as a further criterion for selecting a specific approach.

In fact, we believe that in particular the multilayer perceptron might profit from a larger data set. In a preliminary experiment, we used all almost 10,000 ideals for the training and assessed the accuracy via the results on the test sets in the cross-validation. With this larger set, we observed for it a considerably higher accuracy, whereas for the other models the accuracies were more or less the same. Also the problems with recognising the rare classes disappeared. This seems to imply that the multilayer perceptron is not only the most expensive model concerning the tuning of the hyperparameters, but also needs the largest training data set. If one is willing to spend all the necessary time for the training, one obtains then, however, a very good prediction tool.

The performance of both Algorithm 4 and the machine learning models is not yet completely satisfactory. In particular, the large number of instances where the machine learning model proposes a move which is not related to an obstruction to quasi-stability is surprising and requires further study. As currently only about one fourth of these moves can be accepted, they lead to a considerable waste of computation time.

We have not yet analysed how good our choice of the used features is. Only support vector machines offer here a simple possibility by looking at the support vectors defining the hyperplanes separating the different classes. While one can observe in this model differences in the relative importance of the various features, it turned out that at least for this model all features are relevant and used for the classification. Thus, it seems that our choice of features was not so bad, but we plan to confirm this in the future with a more rigorous statistical analysis of all used models similar to what was done by Pickering et al. (2024). Hopefully, such an analysis will also provide some insight into the inner working of the models, i.e. offer some explanations how they reach their decisions. This in turn may allow us to come up with better human heuristics applicable for an arbitrary number of variables.

Recall from the discussion in the Introduction that a key aspect in determining a quasi-stable position is to preserve as much sparsity as possible during the transformations, as otherwise all subsequent computations are getting rather expensive. Currently, this additional goal is neglected in our scoring, which is solely based on the size of the Pommaret span. As we can see in Fig. 5, sometimes two different moves provide Pommaret spans of almost the same size. In such a situation, the move producing the slightly smaller span might preserve more sparsity and thus might be preferable from the point of view of the full process of determining a quasi-stable position. We plan to include sparsity considerations into the scoring and to perform a multi-objective optimisation in the training phase. In fact, our features have already been selected in such a way that they should provide the necessary information for estimating also the sparsity of the transformed ideal.

In this context, it is also of interest to extend the set of allowed transformations by considering besides elementary moves also permutations. Our assignment of multiplicative variables depends on the numbering of the variables and hence can be influenced by simply permuting the variables. Obviously, permutations are optimal in preserving sparsity. However, in other respects they are more difficult to handle. An elementary move can never decrease the Pommaret span and by simply applying sufficiently many elementary moves one is guaranteed to reach a quasi-stable position. By contrast, a badly chosen permutation can considerably reduce the size of the Pommaret span and with permutations even cycles are possible where after several transformations one has exactly the same leading terms as before.

CRediT authorship contribution statement

Amir Hashemi: Writing – review & editing, Writing – original draft, Supervision, Investigation, Conceptualization. **Mahshid Mirhashemi:** Writing – review & editing, Writing – original draft, Software, Investigation, Data curation, Conceptualization. **Werner M. Seiler:** Writing – review & editing, Writing – original draft, Supervision, Investigation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The computations reported in this article were performed with Python and SageMath. Corresponding Jupiter notebooks and our data set of 10,000 polynomial ideals are freely available at Zenodo using the DOI https://doi.org/10.5281/zenodo.12114165.

Acknowledgements

The authors thank Markus Lange-Hegermann (Lemgo) and Rüdiger Nather (Kassel) for enlightening discussions on machine learning. The research of the first author was in part supported by a grant from IPM (No. 1400130214).

Appendix A. Confusion matrices for the different models and strategies

Fig. A.7 displays bordered confusion matrices for all used machine learning models and Fig. A.8 for the two discussed human strategies (democratic and random). The right border of the matrices contains the precision for each class and the bottom row the recall for each class; in the lower right corner the accuracy is given.

In a comparison of the different machine learning models, it is in particular striking that the multilayer perceptron and the decision tree never correctly identified class 0 and only very rarely class 2. As these classes occur with the lowest frequency in our data set, this is not so visible in the accuracy or in sample weighted averages of precision and recall. Over all models one can observe that class 1 seems to be hard to detect and that it is often confused with class 3.

The confusion matrices for the democratic and the random strategy are reported only for completeness. The random strategy shows exactly the expected behaviour. The democratic strategy profits for the recall for the prevalence of class 3, but generally both precision and recall are rather modest over all classes.

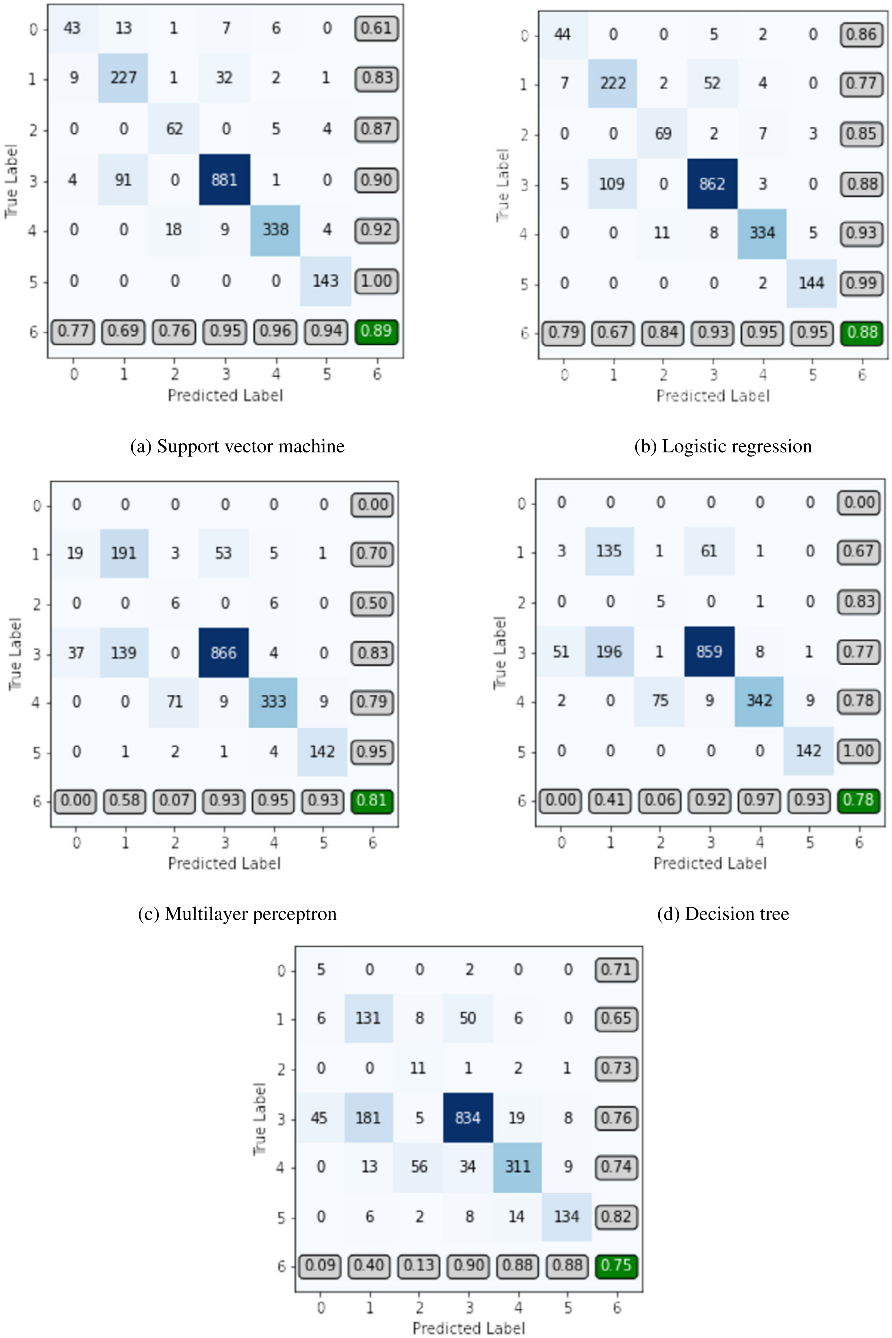
References

Aggarwal, C. (Ed.), 2015. Data Classification: Algorithms and Applications. Data Mining and Knowledge Discovery Series. CRC Press, Boca Raton.

Albert, M., Fetzer, M., Seiler, W., 2015. Janet bases and resolutions in CoCoALib. In: Gerdt, V., Koepf, W., Seiler, W., Vorozhtsov, E. (Eds.), Computer Algebra in Scientific Computing — CASC 2015. In: Lecture Notes in Computer Science, vol. 9301. Springer-Verlag, Cham, pp. 15–29.

Bayer, D., Stillman, M., 1987a. A criterion for detecting m-regularity. Invent. Math. 87, 1–11.

Bayer, D., Stillman, M., 1987b. A theorem on refining division orders by the reverse lexicographic orders. Duke J. Math. 55, 321–328.



(a) Support vector machine

(b) Logistic regression

(c) Multilayer perceptron

(d) Decision tree

(e) k-nearest neighbours

Fig. A.7. Bordered confusion matrices for the different machine learning models.

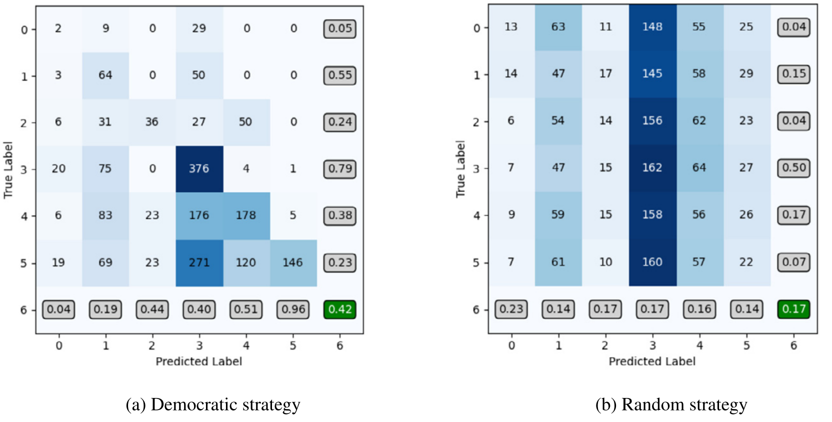


Fig. A.8. Bordered confusion matrices for the discussed human strategies.

Bermejo, I., Gimenez, P., 2006. Saturation and Castelnuovo-Mumford regularity. J. Algebra 303, 592–617.

Bishop, C., 2006. Pattern Recognition and Machine Learning. Information Science and Statistics. Springer-Verlag, New York.

Caviglia, G., Sbarra, E., 2005. Characteristic-free bounds for the Castelnuovo-Mumford regularity. Compos. Math. 141, 1365–1373.

Cox, D.A., Little, J., O'Shea, D., 2015. Ideals, Varieties, and Algorithms, 4th revised ed. Undergraduate Texts in Mathematics. Springer-Verlag, Cham.

Decker, W., Greuel, G., Pfister, G., 1999. Primary decomposition: algorithms and comparisons. In: Matzat, B., Greuel, G., Hiss, G. (Eds.), Algorithmic Algebra and Number Theory. Springer-Verlag, Heidelberg, pp. 187–220.

England, M., Florescu, D., 2019. Comparing machine learning models to choose the variable ordering for cylindrical algebraic decomposition. In: Kaliszyk, C., Brady, E., Kohlhase, A., Coen, C. (Eds.), Intelligent Computer Mathematics – CICM 2019. In: Lecture Notes in Computer Science, vol. 11617. Springer-Verlag, Cham, pp. 93–108.

Florescu, D., England, M., 2019. Algorithmically generating new algebraic features of polynomial systems for machine learning. In: Abbott, J., Griggio, A. (Eds.), Proc. 4th Int. Workshop Satisfiability Checking and Symbolic Computation. In: CEUR Workshop Proceedings, vol. 2460.

Florescu, D., England, M., 2020. A machine learning based software pipeline to pick the variable ordering for algorithms with polynomial inputs. In: Bigatti, A., Carette, J., Davenport, J., Joswig, M., de Wolff, T. (Eds.), Mathematical Software – ICMS 2020. In: Lecture Notes in Computer Science, vol. 12097. Springer-Verlag, Cham, pp. 302–311.

Galligo, A., 1974. A propos du théorème de préparation de Weierstrass. In: Norguet, F. (Ed.), Fonctions de Plusieurs Variables Complexes. In: Lecture Notes in Mathematics, vol. 409. Springer-Verlag, Berlin, pp. 543–579.

Gerdt, V., 2000. On the relation between Pommaret and Janet bases. In: Ghanza, V., Mayr, E., Vorozhtsov, E. (Eds.), Computer Algebra in Scientific Computing — CASC 2000. Springer-Verlag, Berlin, pp. 167–182.

Gerdt, V., 2005. Involutive algorithms for computing Gröbner bases. In: Cojocaru, S., Pfister, G., Ufnarovski, V. (Eds.), Computational Commutative and Non-Commutative Algebraic Geometry. In: NATO Science Series III: Computer & Systems Sciences, vol. 196. IOS Press, Amsterdam, pp. 199–225.

Gerdt, V., Blinkov, Y., 1998a. Involutive bases of polynomial ideals. Math. Comput. Simul. 45, 519–542.

Gerdt, V., Blinkov, Y., 1998b. Minimal involutive bases. Math. Comput. Simul. 45, 543–560.

Gerdt, V., Blinkov, Y., Yanovich, D., 2001. Construction of Janet bases I: monomial bases. In: Ghanza, V., Mayr, E., Vorozhtsov, E. (Eds.), Computer Algebra in Scientific Computing — CASC 2001. Springer-Verlag, Berlin, pp. 233–247.

Hashemi, A., Orth, M., Seiler, W., 2022. Complementary decompositions of monomial ideals and involutive bases. Appl. Algebra Eng. Commun. Comput. 33, 791–821.

Hashemi, A., Parnian, H., Seiler, W., 2019. Noether bases and their applications. Bull. Iran. Math. Soc. 45, 1283–1301.

Hashemi, A., Parnian, H., Seiler, W., 2021. Degree upper bounds for involutive bases. Math. Comput. Sci. 15, 233–254.

Hashemi, A., Schweinfurter, M., Seiler, W., 2012. Quasi-stability versus genericity. In: Gerdt, V., Koepf, W., Mayr, E., Vorozhtsov, E. (Eds.), Computer Algebra in Scientific Computing — CASC 2012. In: Lecture Notes in Computer Science, vol. 7442. Springer-Verlag, Berlin, pp. 172–184.

Hashemi, A., Schweinfurter, M., Seiler, W., 2018. Deterministic genericity for polynomial ideals. J. Symb. Comput. 86, 20–50.

Hausdorf, M., Seiler, W., 2002. An efficient algebraic algorithm for the geometric completion to involution. Appl. Algebra Eng. Commun. Comput. 13, 163–207.

Herzog, J., Popescu, D., Vladoiu, M., 2003. On the ext-modules of ideals of Borel type. In: Avramov, L., Chardin, M., Morales, M., Polini, C. (Eds.), Commutative Algebra: Interactions with Algebraic Geometry. In: Contemp. Math., vol. 331. Amer. Math. Soc., Providence, pp. 171–186.

Huang, Z., England, M., Wilson, D., Bridge, J., Davenport, J., Paulson, L., 2019. Using machine learning to improve cylindrical algebraic decomposition. Math. Comput. Sci. 13, 461–488.

Huang, Z., England, M., Wilson, D., Davenport, J., Paulson, L., Bridge, J., 2014. Applying machine learning to the problem of choosing a heuristic to select the variable ordering for cylindrical algebraic decomposition. In: Watt, S., Davenport, J., Sexton, A.,

Sojka, P., Urban, J. (Eds.), Intelligent Computer Mathematics – CICM 2014. In: Lecture Notes in Computer Science, vol. 8543. Springer-Verlag, Cham, pp. 92–107.
Jamshidi, S., Petrović, S., 2023. The Spark Randomizer: a learned randomized framework for computing Gröbner bases. Preprint. arXiv:2306.08279.
Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, É., 2011. Scikit-learn: machine learning in Python. J. Mach. Learn. Res. 12, 2825–2830.
Peifer, D., 2021. Reinforcement Learning in Buchberger's Algorithm. Ph.D. thesis. Graduate School. Cornell University.
Peifer, D., Stillman, M., Halpern-Leistner, D., 2020. Learning selection strategies in Buchberger's algorithm. Proc. Mach. Learn. Res. 119, 7575–7585.
Pickering, L., Del Rio Almanjano, T., England, M., Cohen, K., 2024. Explainable AI insights for symbolic computation: a case study on selecting the variable ordering for cylindrical algebraic decomposition. J. Symb. Comput. 123, 102276.
Seiler, W., 2009a. A combinatorial approach to involution and δ-regularity I: involutive bases in polynomial algebras of solvable type. Appl. Algebra Eng. Commun. Comput. 20, 207–259.
Seiler, W., 2009b. A combinatorial approach to involution and δ-regularity II: structure analysis of polynomial modules with Pommaret bases. Appl. Algebra Eng. Commun. Comput. 20, 261–338.
Seiler, W., 2010. Involution — The Formal Theory of Differential Equations and Its Applications in Computer Algebra. Algorithms and Computation in Mathematics, vol. 24. Springer-Verlag, Berlin.
Seiler, W., 2012. Effective genericity, δ-regularity and strong Noether position. Commun. Algebra 40, 3933–3949.
Simpson, M., Yi, Q., Kalita, J., 2016. Automatic algorithm selection in computational software using machine learning. In: 15th IEEE Int. Conf. Machine Learning and Applications – ICMLA 2016. IEEE, pp. 355–360.