The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-20)

Gradient Boosts the Approximate Vanishing Ideal

Hiroshi Kera, Yoshihiko Hasegawa
Department of Information and Communication Engineering,
Graduate School of Information Science and Technology,
The University of Tokyo, Tokyo, Japan

Abstract

In the last decade, the approximate vanishing ideal and its basis construction algorithms have been extensively studied in computer algebra and machine learning as a general model to reconstruct the algebraic variety on which noisy data approximately lie. In particular, the basis construction algorithms developed in machine learning are widely used in applications across many fields because of their monomial-order-free property; however, they lose many of the theoretical properties of computer-algebraic algorithms. In this paper, we propose general methods that equip monomial-order-free algorithms with several advantageous theoretical properties. Specifically, we exploit the gradient to (i) sidestep the spurious vanishing problem in polynomial time to remove symbolically trivial redundant bases, (ii) achieve consistent output with respect to the translation and scaling of input, and (iii) remove nontrivially redundant bases. The proposed methods work in a fully numerical manner, whereas existing algorithms require the awkward monomial order or exponentially costly (and mostly symbolic) computation to realize properties (i) and (iii). To our knowledge, property (ii) has not been achieved by any existing basis construction algorithm of the approximate vanishing ideal.

Introduction

A set of data points lies in an algebraic variety[1]—this is a common assumption in various methods in machine learning. For example, linear analysis methods such as principal component analysis are designed to work with data lying in linear subspace, which is a class of algebraic varieties. Broader classes of algebraic varieties are considered in subspace clustering (Vidal, Yi Ma, and Sastry 2005), matrix completion (Ongie et al. 2017; Li and Wang 2017), and classification (Livni et al. 2013; Globerson, Livni, and Shalev-Shwartz 2017). In the last decade, the approximate vanishing ideal (Heldt et al. 2009) has been considered for the machine-learning problem in the most general setting, namely, retrieving a polynomial system that describes the algebraic variety where noisy data points approximately lie. An approximate vanishing ideal of a set of points $X \subset \mathbb{R}^n$ is a set of approximate vanishing polynomials, each of which almost takes a zero value for any $\boldsymbol{x} \in X$. Roughly,

$$\mathcal{I}_{\mathrm{app}}(X) = \{g \in \mathcal{P}_n \mid \forall \boldsymbol{x} \in X, g(\boldsymbol{x}) \approx 0\},$$

where $\mathcal{P}_n$ is the set of all n-variate polynomials over $\mathbb{R}$. Various basis construction algorithms for the approximate vanishing ideal have been proposed, first in computer algebra and then in machine learning (Heldt et al. 2009; Fassino 2010; Livni et al. 2013; Limbeck 2013; Király, Kreuzer, and Theran 2014; Kera and Hasegawa 2018). However, existing algorithms suffer from the tradeoff between practicality and theoretical soundness. Basis construction algorithms developed in machine learning are more practically convenient and are used in various fields (Zhao and Song 2014; Hou, Nie, and Tao 2016; Kera and Iba 2016; Iraji and Chitsaz 2017; Wang and Ohtsuki 2018). This is because these algorithms work with numerical computation and without the monomial order, which is a prefixed prioritization of monomials. Different monomial orders can yield different results, but it is unknown how to properly select a monomial order from exponentially many candidates. However, while enjoying the monomial-order-free property, the basis construction algorithms in machine learning lack various theoretical (and advantageous) properties of computer-algebraic algorithms, which use the practically awkward monomial order and symbolic computation.

In this paper, we propose general and efficient methods that enable monomial-order-free basis construction algorithms in machine learning to have various advantageous theoretical properties. In particular, we address the three theoretical issues listed below. To our knowledge, none of the existing basis construction algorithms can resolve the first and third issues in polynomial time without using a monomial order. Furthermore, the second issue has not been addressed by any existing basis construction methods.

The spurious vanishing problem—A polynomial g can approximately vanish for a point $\boldsymbol{x}$, i.e., $g(\boldsymbol{x}) \approx 0$ not because $\boldsymbol{x}$ is close to the roots of g but merely because g is close to the zero polynomial (i.e., the coefficients of the monomi-

[1] An algebraic variety here refers to a set of points that can be described as the solutions of a polynomial system and it is not necessarily irreducible.

als in g are all small)[2]. To sidestep this, g needs to be normalized by some scale. However, intuitive coefficient normalization is exponentially costly for monomial-order-free algorithms (Kera and Hasegawa 2019).

Inconsistency of output with respect to the translation and scaling of input—Given translated or scaled data, the output of the basis construction can drastically change in terms of the number of polynomials and their nonlinearity, regardless of how well the parameter is chosen. This contradicts the intuition that the intrinsic structure of an algebraic variety does not change by a translation or scaling on data.

Redundancy in the basis set—The output basis set can contain polynomials that are redundant because they can be generated by other lower-degree polynomials[3]. Determining the redundancy usually needs exponentially costly symbolic procedures and is also unreliable in our approximate setting.

To efficiently address these issues without symbolic computation, we exploit the gradient of the polynomials at the input points, which has been rarely considered in the relevant literature. The advantages of this approach are that (i) gradient can be efficiently and exactly computed in our setting without differentiation and (ii) it provides some information on the symbolic structure of and symbolic relations between polynomials in a numerical manner.

In summary, we propose fully numerical methods for monomial-order-free algorithms to retain two theoretical properties of computer-algebraic algorithms and gain one new advantageous property. Hence, we exploit the gradient to address the aforementioned three fundamental issues as follows.

- We propose gradient normalization to resolve the spurious vanishing problem. A polynomial is normalized by the norm of its gradients at the input points. This approach is based on the intuition that polynomials close to the zero polynomial have a small norm for the gradients at all locations. A rigorous theoretical analysis shows its validity.
- We prove that by introducing gradient normalization, a standard basis construction algorithm can equip a sort of invariance to transformations (scaling and translation) of input data points. The number of basis polynomials at each degree is the same before and after the transformation and the change of each basis polynomial is analytically presented.
- We propose a basis reduction method that considers the linear dependency of gradients between polynomials and removes redundant ones from a basis set without symbolic operations.

Related Work

Based on a classical basis construction algorithm for noise-free points (Möller and Buchberger 1982; Kehrein and Kreuzer 2006), most algorithms of the approximate vanishing ideal in computer algebra efficiently sidestep the issues with the spurious vanishing problem and basis set redundancy using the monomial order and symbolic computation. To our knowledge, there are two algorithms that work without the monomial order in computer algebra (Sauer 2007; Hashemi, Kreuzer, and Pourkhajouei 2019), but both require exponential-time procedures. Although the gradient has been rarely considered in the basis construction of the (approximate) vanishing ideal, Fassino (2010) used the gradient during basis construction to check whether a given polynomial exactly vanishes after slightly perturbing given points. Vidal, Yi Ma, and Sastry et al. (2005) considered a union of subspaces for clustering, where the gradient at some points are used to estimate the dimension of each subspace where a cluster lies. Both of these works use the gradient for purposes that are totally different from ours. The closest work to ours is (Fassino and Torrente 2013), which proposes an algorithm to compute an approximate vanishing polynomial of low degree based on the geometrical distance using the gradient. However, their algorithm does not compute a basis set but only provide a single approximate vanishing polynomial. Furthermore, the computation relies on the monomial order and coefficient normalization.

Preliminaries

Throughout the paper, a polynomial is represented as h without arguments and $h(\boldsymbol{x}) \in \mathbb{R}$ denotes the evaluation of h at a point $\boldsymbol{x}$.

From polynomials to evaluation vectors

Definition 1 (Vanishing Ideal)**.** *Given a set of n-dimensional points X, the vanishing ideal of X is a set of n-variate polynomials that take a zero value, (i.e., vanish) for any point in X. Formally,*

$$\mathcal{I}(X) = \{g \in \mathcal{P}_n \mid \forall \boldsymbol{x} \in X, g(\boldsymbol{x}) = 0\}.$$

Definition 2 (Evaluation vector and evaluation matrix)**.** *Given a set of points $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_{|X|}\}$, the evaluation vector of a polynomial h is defined as follows:*

$$h(X) = \begin{pmatrix} h(\boldsymbol{x}_1) & h(\boldsymbol{x}_2) & \cdots & h(\boldsymbol{x}_{|X|}) \end{pmatrix}^\top \in \mathbb{R}^{|X|},$$

where $|\cdot|$ denotes the cardinality of a set. For a set of polynomials $H = \{h_1, h_2, \ldots, h_{|H|}\}$, its evaluation matrix is $H(X) = (h_1(X)\ h_2(X)\ \cdots\ h_{|H|}(X)) \in \mathbb{R}^{|X| \times |H|}$.

Definition 3 (ϵ-vanishing polynomial)**.** *A polynomial g is an ϵ-vanishing polynomial for a set of points X if $\|g(X)\| \leq \epsilon$, where $\|\cdot\|$ denotes the Euclidean norm; otherwise, g is an ϵ-nonvanishing polynomial.*

As Definition 1 indicates, we are only interested in the evaluation values of polynomials for the given set of points X. Hence, a polynomial h can be represented by its evaluation vector $h(X)$. As a consequence, the product and weighted sum of polynomials become linear algebra operations. Let us consider a set of polynomials $H = \{h_1, h_2, ..., h_{|H|}\}$. A product of $h_1, h_2 \in H$ becomes $h_1(X) \odot h_2(X)$, where $\odot$ denotes the entry-wise

[2]For example, a univariate polynomial $g = x^2 - 1$ approximately vanishes only for points close to its roots $x = \pm 1$. However, once g is scaled to kg by a small nonzero $k \in \mathbb{R}$, then kg can approximately vanish for points far from its roots. A simple remedy is to normalize kg as $kg/\sqrt{2}k$ using the coefficients.

[3]For example, a polynomial gh is unnecessary if g is included in the basis set.

product. A weighted sum $\sum_{i=1}^{|X|} w_i h_i$, where $w_i \in \mathbb{R}$, becomes $\sum_{i=1}^{|H|} w_i h_i(X)$. The weighted sum of polynomials is an important building block in the following discussion. For convenience of notation, we define a special product between a polynomial set and a vector as $H\boldsymbol{w} := \sum_{i=1}^{|H|} w_i h_i$, where w_i is the i-th entry of $\boldsymbol{w} \in \mathbb{R}^{|H|}$. Similarly, we denote the product between a polynomial set H and a matrix $W = (\boldsymbol{w}_1\ \boldsymbol{w}_2\ \cdots \boldsymbol{w}_s) \in \mathbb{R}^{|H|\times s}$ as $HW := \{H\boldsymbol{w}_1, H\boldsymbol{w}_2, ..., H\boldsymbol{w}_s\}$. Note that $(H\boldsymbol{w})(X) = H(X)\boldsymbol{w}$ and $(HW)(X) = H(X)W$. We consider a set of polynomials that is *spanned* by a set F of nonvanishing polynomials or *generated* by a set G of vanishing polynomials. We denote the former as $\mathrm{span}(F) = \{\sum_{f\in F} a_f f \mid a_f \in \mathbb{R}\}$ and the latter as $\langle G\rangle = \{\sum_{g\in G} h_g g \mid h_g \in \mathcal{P}_n\}$.

Simple Basis Construction Algorithm

Our idea of using the gradient is general enough to be integrated with existing monomial-order-free algorithms. However, to avoid a unnecessarily abstract discussion, we focus on the Simple Basis Construction (SBC) algorithm (Kera and Hasegawa 2019), which was proposed by (Kera and Hasegawa 2019) based on Vanishing Component Analysis (VCA; Livni et al. 2013). Most monomial-order-free algorithms can be discussed using SBC; thus, the following discussion is sufficiently general.

The input to SBC is a set of points $X \subset \mathbb{R}^n$ and error tolerance $\epsilon \geq 0$. SBC outputs a basis set G of ϵ-vanishing polynomials and a basis set of ϵ-nonvanishing polynomials F. We later discuss the conditions that G and F are required to satisfy (cf., Theorem 1). SBC proceeds from degree-0 polynomials to those of higher degree. At each degree t, a set of degree-t ϵ-vanishing polynomials G_t and a set of degree-t ϵ-nonvanishing polynomials F_t are generated. We use notations $F^t = \bigcup_{\tau=0}^{t} F_\tau$ and $G^t = \bigcup_{\tau=0}^{t} G_\tau$. For $t = 0$, $F_0 = \{m\}$ and $G_0 = \emptyset$, where $m \neq 0$ is a constant polynomial. At each degree $t \geq 1$, the following procedures (`Step 1`, `Step 2`, and `Step 3`) are conducted[4].

Step 1: Generate a set of candidate polynomials Pre-candidate polynomials of degree t for $t > 1$ are generated by multiplying nonvanishing polynomials across F_1 and F_{t-1}.

$$C_t^{\mathrm{pre}} = \{pq \mid p \in F_1, q \in F_{t-1}\}.$$

At $t = 1$, $C_1^{\mathrm{pre}} = \{x_1, x_2, ..., x_n\}$, where x_k are variables. The candidate basis is then generated through the orthogonalization.

$$C_t = C_t^{\mathrm{pre}} - F^{t-1}F^{t-1}(X)^{\dagger}C_t^{\mathrm{pre}}(X), \tag{1}$$

where $\cdot^{\dagger}$ is the pseudo-inverse of a matrix.

Step 2: Solve a generalized eigenvalue problem We solve the following generalized eigenvalue problem:

$$C_t(X)^{\top}C_t(X)V = \mathfrak{N}(C_t)V\Lambda, \tag{2}$$

where a matrix V that has generalized eigenvectors $\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_{|C_t|}$ for its columns, Λ is a diagonal matrix with generalized eigenvalues $\lambda_1, \lambda_2, ..., \lambda_{|C_t|}$ along its diagonal, and $\mathfrak{N}(C_t) \in \mathbb{R}^{|C_t|\times|C_t|}$ is the normalization matrix, which will soon be introduced.

Step 3: Construct sets of basis polynomials Basis polynomials are generated by linearly combining polynomials in C_t with $\{\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_{|C_t|}\}$.

$$G_t = \{C_t\boldsymbol{v}_i \mid \sqrt{\lambda_i} \leq \epsilon\},$$
$$F_t = \{C_t\boldsymbol{v}_i \mid \sqrt{\lambda_i} > \epsilon\}.$$

If $|F_t| = 0$, the algorithm terminates with output $G = G^t$ and $F = F^t$.

Remark 1. *At `Step 1`, Eq. (1) makes the column space of $C_t(X)$ orthogonal to that of $F^{t-1}(X)$. The aim is to focus on the subspace of $\mathbb{R}^{|X|}$ that cannot be spanned by the evaluation vectors of polynomials of degree less than t.*

Remark 2. *At `Step 3`, a polynomial $C_t\boldsymbol{v}_i$ is classified as an ϵ-vanishing polynomial if $\sqrt{\lambda_i} \leq \epsilon$ because $\sqrt{\lambda_i}$ equals the extent of vanishing of $C_t\boldsymbol{v}_i$. Actually,*

$$\|(C_t\boldsymbol{v}_i)(X)\| = \sqrt{\boldsymbol{v}_i^{\top}C_t(X)^{\top}C_t(X)\boldsymbol{v}_i} = \sqrt{\lambda_i}.$$

At `Step 2`, we have a normalization matrix $\mathfrak{N}(C_t) \in \mathbb{R}^{|C_t|\times|C_t|}$ to resolve the spurious vanishing problem (Kera and Hasegawa 2019). For the coefficient normalization, the coefficient vector[5] of c_i is denoted by $\mathfrak{n}_{\mathrm{c}}(c_i)$, and the (i,j)-th entry of $\mathfrak{N}(C_t)$ is $\mathfrak{n}_{\mathrm{c}}(c_i)^{\top}\mathfrak{n}_{\mathrm{c}}(c_j)$. Solving the generalized eigenvalue problem Eq. (2) with this normalization matrix leads to polynomials $C_t\boldsymbol{v}_1, ..., C_t\boldsymbol{v}_{|C_t|}$ at `Step 3`, which are normalized with respect to their coefficient vectors, i.e., $\forall i, \|\mathfrak{n}_{\mathrm{c}}(C_t\boldsymbol{v}_i)\| = 1$. Instead of $\mathfrak{n}_{\mathrm{c}}$, we can also define the normalization matrix from another mapping as long as it satisfies some requirements (Kera and Hasegawa 2019). Later, we propose a novel mapping $\mathfrak{n}_{\mathrm{g}}$, which is based on the gradient of a given polynomial. Although this mapping only satisfy the relaxed version of the requirements, we show that the same guarantee for the SBC output (Theorem 2 in Kera and Hasegawa 2019) can still be stated with these relaxed requirements (see the supplementary material for proof).

Theorem 1. *Let $\mathfrak{n}$ be a valid normalization mapping for SBC (cf., Definition 4). When SBC with $\mathfrak{n}$ runs with $\epsilon = 0$ for a set of points X, the output basis sets G and F satisfy the following.*

- *Any vanishing polynomial $g \in \mathcal{I}(X)$ can be generated by G, i.e., $g \in \langle G\rangle$.*
- *Any polynomial h can be represented by $h = f' + g'$, where $f' \in \mathrm{span}(F)$ and $g' \in \langle G\rangle$.*
- *For any t, any degree-t vanishing polynomial $g \in \mathcal{I}(X)$ can be generated by G^t, i.e., $g \in \langle G^t\rangle$.*
- *For any t, any degree-t polynomial h can be represented by $h = f' + g'$, where $f' \in \mathrm{span}(F^t)$ and $g' \in \langle G^t\rangle$.*

[4]For ease of understanding, we describe the procedures in the form of symbolic computation, but these can be numerically implemented (i.e., by matrix-vector calculations)

[5]The coefficient vector of a polynomial is defined as a vector that lists the coefficients of the monomials of the polynomial. For instance, a degree-3 univariate polynomial $g = 1 - x + 2x^3$ has the coefficient vector $(1, -1, 0, 2)^{\top}$.

Proposed Method

In the literature on the vanishing ideal, polynomials are represented by their evaluation vectors at an input set of points X. However, two vanishing polynomials, say g_1 and g_2, share identical evaluation vectors $g_1(X) = g_2(X) = \mathbf{0}$, and thus any information about their symbolic forms cannot be inferred from these vectors. In this paper, we propose to use the gradient as a key tool to deal with polynomials in a fully numerical way. Specifically, given a polynomial h, we consider the evaluation of its partial derivatives $\nabla h := \{\partial h/\partial x_1, \partial h/\partial x_2, \cdots, \partial h/\partial x_n\}$ at the given set of data points $X \subset \mathbb{R}^n$; that is, from the definition of the evaluation matrix, we consider

$$\nabla h(X) = \begin{pmatrix} \frac{\partial h}{\partial x_1}(X) & \frac{\partial h}{\partial x_2}(X) & \cdots & \frac{\partial h}{\partial x_n}(X) \end{pmatrix},$$

which can be efficiently and exactly calculated without differentiation by taking advantage of the iterative framework of the basis construction. Interestingly, one can infer the symbolic structure of a vanishing polynomial g from $\nabla g(X)$. For example, if $(\partial g/\partial x_k)(X) \approx \mathbf{0}$, then the variable x_k is unlikely to be dominant in g; if $(\partial g/\partial x_k)(X) \approx \mathbf{0}$ for all k, then g can be close to the zero polynomial. One may argue that a nonzero vanishing polynomial g can take $\nabla g(X) = O$. However, such g is revealed to be *redundant* in the basis set, and thus it can be excluded from our consideration (cf., Lemmas 1 and 2). Next, we ask whether any symbolic relation between vanishing polynomials g_1 and g_2 is reflected in the relation between $\nabla g_1(X)$ and $\nabla g_2(X)$. The answer is yes; if g_2 is a polynomial multiple of g_1, i.e., $g_2 = g_1 h$ for some $h \in \mathcal{P}_n$, then for any $\boldsymbol{x} \in X$, $\nabla g_1(\boldsymbol{x})$ and $\nabla g_2(\boldsymbol{x})$ are identical up to a constant scale. A more general symbolic relation between polynomials is discussed in Conjecture 1. The proofs of our claims are provided in the supplementary material for reasons of space.

Gradient normalization for the spurious vanishing problem

The spurious vanishing problem is resolved by normalizing polynomials for some scale. Here, we propose gradient normalization, which normalizes polynomials using the norm of their gradient. Specifically, a polynomial h is normalized with the norm of the vector

$$\mathfrak{n}_{\mathrm{g}}(h; X) = \mathrm{vec}(\nabla h(X)) \in \mathbb{R}^{|X|n}, \tag{3}$$

where $\mathrm{vec}(\cdot)$ denotes the vectorization of a given matrix. We refer to the norm $\|\mathfrak{n}_{\mathrm{g}}(h; X)\|$ as the gradient norm of h. By solving Eq. (2) in `Step 2`, the basis polynomials of vanishing polynomials and nonvanishing polynomials (say, h) are normalized such that $\|\mathfrak{n}_{\mathrm{g}}(h; X)\| = 1$. Conceptually, this rescales h with respect to the gradient norm as $h/\|\mathfrak{n}_{\mathrm{g}}(h; X)\|$, but in an optimal way (Kera and Hasegawa 2019). The gradient normalization is superior to the coefficient normalization in terms of computational cost; the former works in polynomial time complexity (cf., Proposition 2) and the latter requires exponential time complexity. SBC using $\mathfrak{n}_{\mathrm{g}}$ (SBC-$\mathfrak{n}_{\mathrm{g}}$) set m of $F_0 = \{m\}$ to the mean absolute value of X for consistency.

The gradient normalization is based on a shift in thinking on "being close to the zero polynomial". Traditionally, the closeness was measured based on the coefficients—polynomials with small coefficients are considered close to the zero polynomial. On the other hand, the gradient normalization is based on the gradient norm; that is, if the gradient of a polynomial has a small norm at all the given points, then the polynomial is considered close to the zero polynomial.

A natural concern about the gradient normalization is that the gradient norm $\|\mathfrak{n}_{\mathrm{g}}(h; X)\|$ can be equal to zero even for a nonzero polynomial h. In other words, what if all partial derivatives $\partial h/\partial x_k$ are vanishing for X, i.e., $(\partial h/\partial x_k)(X) = \mathbf{0}$? Solving the generalized eigenvalue problem Eq. (2) only provides polynomials with the nonzero gradient norm. Is it sufficient for basis construction to only collect such polynomials? The following two lemmas answer this question affirmatively.

Lemma 1. *Suppose that $G^t \subset \mathcal{P}_n$ is a basis set of vanishing polynomials of degree at most t for a set of points X such that for any $\widetilde{g} \in \mathcal{I}(X)$ of degree at most t, $\widetilde{g} \in \langle G^t \rangle$. Then, for any $g \in \mathcal{I}(X)$ of degree $t+1$, if $(\partial g/\partial x_k)(X) = \mathbf{0}$ for all $k = 1, 2, ..., n$, then $g \in \langle G^t \rangle$.*

Lemma 2. *Suppose that $F^t \subset \mathcal{P}_n$ is a basis set of nonvanishing polynomials of degree at most t for a set of points X such that for the evaluation vector $\widetilde{f}(X)$ of any nonvanishing polynomial $\widetilde{f}$ of degree at most t, $\widetilde{f}(X) \in \mathrm{span}(F^t(X))$. Then, for any nonvanishing polynomial $f \in \mathcal{P}_n$ of degree $t+1$, if $(\partial f/\partial x_k)(X) = 0$ for all $k = 1, 2, ..., n$, then $f(X) \in \mathrm{span}(F^t(X))$.*

These two lemmas imply that we do not need polynomials with zero gradient norms for constructing basis sets because these polynomials can be described by basis polynomials of lower degrees. Therefore, it is valid to use $\mathfrak{n}_{\mathrm{g}}$ for the normalization in SBC. Formally, we define the validity of the normalization mapping for a basis construction as follows.

Definition 4 (Valid normalization mapping for $\mathcal{A}$)**.** *Let $\mathfrak{n} : \mathcal{P}_n \to \mathbb{R}^\ell$ be a mapping that satisfies the following.*

- *$\mathfrak{n}$ is a linear mapping, i.e., $\mathfrak{n}(ah_1 + bh_2) = a\mathfrak{n}(h_1) + b\mathfrak{n}(h_2)$, for any $a, b \in \mathbb{R}$ and any $h_1, h_2 \in \mathcal{P}_n$.*
- *The dot product is defined between normalization components; that is, $\langle \mathfrak{n}(h_1), \mathfrak{n}(h_2) \rangle$ is defined for any $h_1, h_2 \in \mathcal{P}_n$.*
- *In a basis construction algorithm $\mathcal{A}$, $\mathfrak{n}(h)$ takes the zero value only for polynomials that can be generated by basis polynomials of lower degrees.*

Then, $\mathfrak{n}$ is a valid normalization mapping for $\mathcal{A}$, and $\mathfrak{n}(h)$ is called the normalization component of h.

As the third condition implies, this definition is dependent on the algorithm $\mathcal{A}$. The third condition is the relaxed condition of that in (Kera and Hasegawa 2019), where $\mathfrak{n}(h)$ is required to take a zero value if and only if h is the zero polynomial. Now, we can readily show that $\mathfrak{n}_{\mathrm{g}}$ is a valid normalization mapping for SBC.

Theorem 2. *The mapping $\mathfrak{n}_{\mathrm{g}}$ of Eq. (3) is a valid normalization mapping for SBC.*

We emphasize that gradient normalization is essentially different from coefficient normalization because it is a data-dependent normalization. The following proposition holds thanks to this data-dependent nature, which argues for consistency in the output of SBC-$\mathfrak{n}_{\mathrm{g}}$ with respect to a translation or scaling of the input data points.

Proposition 1. *Suppose SBC-$\mathfrak{n}_{\mathrm{g}}$ outputs (G, F) for input (X, ϵ), $(\widetilde{G}, \widetilde{F})$ for input $(X - \boldsymbol{b}, \epsilon)$, and $(\widehat{G}, \widehat{F})$ for input $(\alpha X, |\alpha|\epsilon)$, where $X - \boldsymbol{b}$ denotes the translation of each point in X by $\boldsymbol{b}$ and αX denotes the scaling by $\alpha \neq 0$.*

- *$G, \widetilde{G}$, and $\widehat{G}$ have exactly the same number of basis polynomials at each degree.*
- *$F, \widetilde{F}$, and $\widehat{F}$ have exactly the same number of basis polynomials at each degree.*
- *Any pair of the corresponding polynomials $\widetilde{h} \in \widetilde{G} \cup \widetilde{F}$ and $h \in G \cup F$ satisfies $h(x_1, x_2, ..., x_n) = \widetilde{h}(x_1 + b_1, x_1 + b_2, ..., x_n + b_n)$, where $h(x_1, x_2, ..., x_n)$ here denotes a polynomial in n variables $x_1, x_2, ..., x_n$ and $\boldsymbol{b} = (b_1, b_2, ..., b_n)^\top$.*
- *For any pair of the corresponding polynomials $\widehat{h} \in \widehat{G} \cup \widehat{F}$ and $h \in G \cup F$, $\widehat{h}$ is the $(1, \alpha)$-degree-wise identical[6] to h.*

The first two statements of Proposition 1 argue that translation and scaling on the input points do not affect the inferred dimensionality of the algebraic set where the noisy data approximately lie; an algebraic variety should be described by the same number of polynomials of the same nonlinearity before and after these data transformations. Although this intuition seems natural, to our knowledge, no existing basis construction algorithms have this property. The third statement of Proposition 1 argues that basis polynomials for translated data are polynomials with a variable translation from those of the untranslated data. Note that it is not trivial that the *algorithm* outputs these translated polynomials. VCA has this translation-invariance property, whereas most other basis construction algorithms, including SBC with the coefficient normalization, do not. The last statement of Proposition 1 is the most interesting property and is not held by any other basis construction algorithms, to our knowledge. The $(1, \alpha)$-degree-wise identicality between the corresponding $\widehat{h} \in \widehat{G} \cup \widehat{F}$ and $h \in G \cup F$ implies the following relation:

$$\widehat{h}(\alpha X) = \alpha h(X). \tag{4}$$

In words, scaling by α on input X of SBC-$\mathfrak{n}_{\mathrm{g}}$ only affects *linearly* the evaluation vectors of the *nonlinear* output polynomials. Thus, we only need linearly scaled threshold $\alpha\epsilon$ for αX. Without this property, linear scaling on the input leads to nonlinear scaling on the evaluation of the output polynomials; thus, a consistent result cannot be obtained regardless of how well ϵ is chosen. Symbolically, $(1, \alpha)$-degree-wise identicality implies that h and $\widehat{h}$ consist of the same terms up to a scale, and the corresponding terms m of h and $\widehat{m}$ of $\widehat{h}$ relate as $\widehat{m} = \alpha^{1-\tau} m$. This implies that larger α decreases the coefficients of higher-degree terms more sharply. This is quite natural because highly nonlinear terms grow sharply as the input value increases. One may argue that any basis construction algorithm could obtain translation- and scale-invariance by introducing a preprocessing stage for input X, such as mean-centralization and normalization. Although preprocessing can be helpful in some practical scenarios, it discards the mean and scale information, and thus the output basis sets do not reflect this information. In contrast, the output polynomials of SBC-$\mathfrak{n}_{\mathrm{g}}$ reflect the mean and scale, but in a convenient form.

[6]The gist of this property will be explained soon. The definition can be found in the supplementary material.

Removal of redundant basis polynomials

The monomial-order-free algorithms tend to output a large basis set of vanishing polynomials that contains redundant basis polynomials. Specifically, let G be an output basis set of vanishing polynomials ($\epsilon = 0$). Then, G can contain redundant polynomials (say, $g \in G$) that can be generated from polynomials of lower degrees in G; that is, with some polynomials $\{h_{g'}\} \subset \mathcal{P}_n$,

$$g = \sum_{g' \in G^{\deg(g)-1}} h_{g'} g', \tag{5}$$

which is equivalent to $g \in \langle G^{\deg(g)-1} \rangle$. To determine whether $g \in \langle G^{\deg(g)-1} \rangle$ or not for a given g, a standard approach in computer algebra is to divide g by the Gröbner basis of $G^{\deg(g)-1}$. However, the complexity of computing a Gröbner basis is known to be doubly exponential (Cox, Little, and O'shea 1992). Polynomial division also needs an expanded form of g, which is also computationally costly to obtain. Moreover, this polynomial division-based approach is not suitable for the approximate setting, where g may be approximately generated by polynomials in $G^{\deg(g)-1}$. Thus, we would like to handle the redundancy in a numerical way using the evaluation values at points. However, (exact) vanishing polynomials have the same evaluation vectors $\mathbf{0}$.

Here again, we can resort to the gradient of the polynomials, whose evaluation values are proven to be nonvanishing at input points (Lemma 1). In short, we consider g as redundant if for any point $\boldsymbol{x} \in X$, the gradient $\nabla g(\boldsymbol{x})$ is linearly dependent on that of the polynomials in $G^{\deg(g)-1}$.

Conjecture 1. *Let G be a basis set of a vanishing ideal $\mathcal{I}(X)$, which is output by SBC with $\epsilon = 0$. Then, $g \in G$ is $g \in \langle G^{\deg(g)-1} \rangle$ if and only if for any $\boldsymbol{x} \in X$,*

$$\nabla g(\boldsymbol{x}) = \sum_{g' \in G^{\deg(g)-1}} \alpha_{g', \boldsymbol{x}} \nabla g'(\boldsymbol{x}), \tag{6}$$

for some $\alpha_{g', \boldsymbol{x}} \in \mathbb{R}$.

The sufficient condition ("if" statement) can be readily proven by differentiating $g = \sum_{g' \in G^{\deg(g)-1}} g' h_{g'}$ and using $g'(\boldsymbol{x}) = 0$ (see the supplementary material). Using the sufficiency, we can remove all the redundant polynomials in the form of Eq. (5) from the basis set by checking whether or not Eq. (6) holds. Note that we may accidentally remove some basis polynomials that are not redundant because the

necessity ("only if" statement) remains to be proven. Conceptually, the necessity implies that one can know the global (symbolic) relation $g \in \langle G^{\deg(g)-1} \rangle$ from the local relation Eq. (6) at finitely many points X. This may not be true for general g and $G^{\deg(g)-1}$. However, g and $G^{\deg(g)-1}$ are both generated in a very restrictive way, and this is why we suspect that this conjecture can be true.

We can support the validity of using Conjecture 1 from another perspective. When Eq. (6) holds, this implies the following: using the basis polynomials of lower degrees, one can generate a polynomial $\widehat{g}$ that takes the same value and gradient as g at all the given points; in short, $\widehat{g}$ behaves identically to g up to the first order for all the points. According to the spirit of the vanishing ideal—identifying a polynomial only by its behavior for given points—it is reasonable to consider g as "redundant" for practical use.

Lastly, we describe how to use Conjecture 1 to remove redundant polynomials. Given g and $G^{\deg(g)-1}$, we solve the following least squares problem for each $\boldsymbol{x} \in X$:

$$\min_{\boldsymbol{v} \in \mathbb{R}^{|G^{\deg(g)-1}|}} \|\nabla g(\boldsymbol{x}) - \boldsymbol{v}^\top \nabla G^{\deg(g)-1}(\boldsymbol{x})\|, \quad (7)$$

where $\nabla G^{\deg(g)-1}(\boldsymbol{x}) \in \mathbb{R}^{|G^{\deg(g)-1}| \times n}$ is a matrix that stacks $\nabla g'(\boldsymbol{x})$ for $g' \in G^{\deg(g)-1}$ in each row (note that $\nabla g(\boldsymbol{x}) \in \mathbb{R}^{1 \times n}$). This problem has a closed-form solution $\boldsymbol{v}^\top = \nabla g(\boldsymbol{x}) \nabla G^{\deg(g)-1}(\boldsymbol{x})^\dagger$. If the residual error is zero for all the points in X, then g is removed as a redundant polynomial. In the approximately vanishing case ($\epsilon > 0$), we set a threshold for the residual error. The procedure above can be performed during or after basis construction. When the basis construction is not normalized using $\mathfrak{n}_{\mathrm{g}}$, it is also necessary to check the linear dependency of the gradient within G_t (see the supplementary material for details).

Compute the gradient without differentiation

In our setting, exact gradients for input points can be computed without differentiation. Recall that at degree t, `Step 3` of SBC computes linear combinations of the candidate polynomials in C_t. Noting that C_t is generated from the linear combinations of C_t^{pre} and F^{t-1}, any $h \in \mathrm{span}(C_t)$ can be described as

$$h = \sum_{c \in C_t^{\mathrm{pre}}} u_c c + \sum_{f \in F^{t-1}} v_f f,$$

where $u_c, v_f \in \mathbb{R}$. Note that $c \in C_t^{\mathrm{pre}}$ is a product of a polynomial in F_1 and a polynomial in F_{t-1}. Let $p_c \in F_1$ and $q_c \in F_{t-1}$ be such polynomials, i.e., $c = p_c q_c$. Using the product rule, the evaluation of $\partial h / \partial x_k$ for $\boldsymbol{x} \in X$ is then

$$\frac{\partial h}{\partial x_k}(\boldsymbol{x}) = \sum_{c \in C_t^{\mathrm{pre}}} u_c q_c(\boldsymbol{x}) \frac{\partial p_c}{\partial x_k}(\boldsymbol{x}) + \sum_{c \in C_t^{\mathrm{pre}}} u_c p_c(\boldsymbol{x}) \frac{\partial q_c}{\partial x_k}(\boldsymbol{x}) + \sum_{f \in F^{t-1}} v_f \frac{\partial f}{\partial x_k}(\boldsymbol{x}). \quad (8)$$

Note that $p_c(\boldsymbol{x})$, $q_c(\boldsymbol{x})$, $(\partial p_c / \partial x_k)(\boldsymbol{x})$, $(\partial q_c / \partial x_k)(\boldsymbol{x})$, and $(\partial f / \partial x_k)(\boldsymbol{x})$ have already been calculated in the previous iterations up to degree $t-1$. For degree $t = 1$, the gradients of the linear polynomials are the combination vectors $\boldsymbol{v}_i$ obtained in `Step 2`. Thus, $\nabla h(X)$ can be exactly calculated without differentiation using the results at lower degrees.

Proposition 2. *Suppose we perform SBC for a set of points $X \in \mathbb{R}^n$. At the iteration for degree t, for any polynomial $h \in \mathrm{span}(C_t \cup F^{t-1})$ and any point $\boldsymbol{x} \in \mathbb{R}^n$, we can compute $\nabla h(\boldsymbol{x})$ without differentiation with a computational cost of $O(n|C_t|) = O(n\mathrm{rank}(X)|X|)$.*

This computational cost $O(n\mathrm{rank}(X)|X|)$ is quite acceptable, noting that generating C_t already needs $O(\mathrm{rank}(X)|X|)$ and solving Eq. (2) needs $O(|C_t|^3) = O(\mathrm{rank}(X)^3|X|^3)$. Moreover, in this analysis, we use a very rough relation $O(|F_t|) = |X|$, whereas $|F_t| \ll |X|$ in practice (see the supplementary material). Giving up the exact calculation, one can further reduce the runtime by restricting the variables and points to be taken into account. That is, a normalized component of a polynomial h can be $\widehat{\mathfrak{n}}_{\mathrm{g}}(h) = \nabla_\Omega h(Y)$, where $\Omega \subset \{1, 2, ..., n\}$, $Y \subset X$, and $\nabla_\Omega h = \{\partial h / \partial x_i \mid i \in \Omega\}$. For example, Ω can be the index set of variables that have large variance and Y as the centroids of clusters on X.

Results

We compare four basis construction algorithms, VCA, SBC with the coefficient normalization (SBC-$\mathfrak{n}_{\mathrm{c}}$), SBC-$\mathfrak{n}_{\mathrm{g}}$, and SBC-$\mathfrak{n}_{\mathrm{c}}$ with the basis reduction. All experiments were performed using Julia implementations on a desktop machine with an eight-core processor and 32 GB memory.

Basis reduction using the gradient

We confirm that redundant basis sets can be reduced by our basis reduction method. We consider the vanishing ideal of $X = \{(1,0), (0,1), (-1,0), (0,-1)\}$ in a noise-free setting, where the exact Gröbner basis and polynomial division can be computed to verify our reduction. As shown in Fig. 1, the VCA basis set consists of five vanishing polynomials and the SBC-$\mathfrak{n}_{\mathrm{g}}$ basis set consists of four vanishing polynomials. These basis sets share two polynomials, $g_1 = x^2 + y^2 - 1$ and $g_2 = xy$ (the constant scale is ignored). A simple calculation using the Gröbner basis of $\{g_1, g_2\}$ reveals that the other polynomials in each basis set can be generated by $\{g_1, g_2\}$. Using our basis reduction method, both basis sets were successfully reduced to $\{g_1, g_2\}$. Other examples and the noisy case can be found in the supplementary material.

Comparison of basis sets

We construct two datasets (D_1 and D_2, respectively) from two algebraic varieties: (i) triple concentric ellipses (radii $(\sqrt{2}, 1/\sqrt{2})$, $(2\sqrt{2}, 2/\sqrt{2})$, and $(3\sqrt{2}, 3/\sqrt{2})$) with $3\pi/4$ rotation and (ii) $\left\{x_1 x_3 - x_2^2, x_1^3 - x_2 x_3\right\}$. From each of them, 75 points and 100 points are randomly sampled. Five additional variables $y_i = k_i x_1 + (1 - k_i) x_2$ for $k_i \in \{0.0, 0.2, 0.5, 0.8, 1.0\}$ are added to the former and nine additional variables $y_i = k_i x_1 + l_i x_2 + (1 - k_i - l_i) x_3$ for $(k_i, l_i) \in \{0.2, 0.5, 0.8\}^2$ are added to the latter. Then, sampled points are mean-centralized and perturbed by additive Gaussian noise. The mean of the noise is set to zero, and the

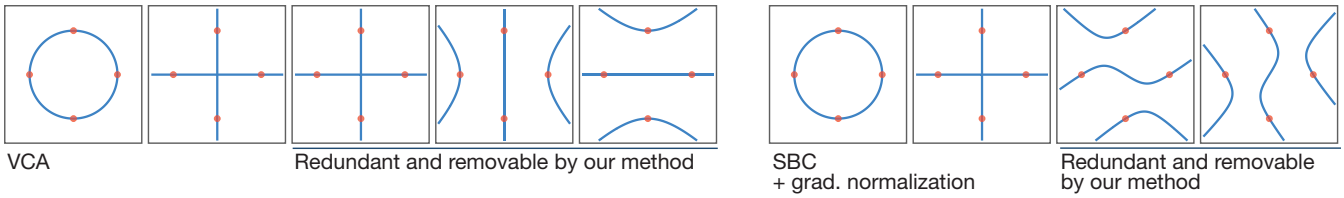

Figure 1: Sets of vanishing polynomials obtained by VCA (left panel) and by SBC-$\mathfrak{n}_{\mathrm{g}}$ (right panel). Both sets contain redundant basis polynomials (the last three in the left panel and the last two in the right panel), which can be efficiently removed by the proposed method based on Conjecture 1.

Table 1: Comparison of basis sets obtained by SBC with different normalization ($\mathfrak{n}_{\mathrm{c}}$ and $\mathfrak{n}_{\mathrm{g}}$). Here, $\mathfrak{n}$-*ratio* denotes the ratio of the largest norm to the smallest norm of the polynomials in the basis set with respect to $\mathfrak{n}$.

		# of bases	$\mathfrak{n}_{\mathrm{c}}$-ratio	$\mathfrak{n}_{\mathrm{g}}$-ratio	runtime (ms)
D_1	$\mathfrak{n}_{\mathrm{c}}$	41	1.00	12.2e+2	48.0e+1
	$\mathfrak{n}_{\mathrm{g}}$	30	46.6	1.00	**13.4**
D_2	$\mathfrak{n}_{\mathrm{c}}$	70	1.00	19.6e+2	17.5e+3
	$\mathfrak{n}_{\mathrm{g}}$	33	76.9	1.00	**11.4**

standard deviation is set to 5% of the average absolute value of the points. The parameter ϵ is selected so that (i) the number of linear vanishing polynomials in the basis set agrees with the number of additional variables y_i and (ii) except for these linear polynomials, the lowest degree (say, $d_{\min}$) of the polynomials agree with that of the Gröbner basis of the target variety and the number of degree-$d_{\min}$ polynomials in the basis set agrees with or exceeds that of the Gröbner basis. Refer to the supplementary material for details. As can be seen from Table 1, SBC-$\mathfrak{n}_{\mathrm{g}}$ runs substantially faster than SBC-$\mathfrak{n}_{\mathrm{c}}$ (about 10 times faster in D_1 and about 10^3 times faster in D_2). Here, $\mathfrak{n}$-ratio denotes the ratio of the largest to smallest norms of the polynomials in a basis set with respect to $\mathfrak{n}$. Hence, $\mathfrak{n}_{\mathrm{c}}$-ratio and $\mathfrak{n}_{\mathrm{g}}$-ratio are unity for SBC-$\mathfrak{n}_{\mathrm{c}}$ and SBC-$\mathfrak{n}_{\mathrm{g}}$, respectively. Here, VCA is not compared because a proper ϵ could not be found; if the correct number of linear vanishing polynomials were found by VCA, then the degree-$d_{\min}$ polynomials could not be found, and vice versa. This implies the importance of sidestepping the spurious vanishing problem by normalization.

Classification

We compared the basis sets obtained by different basis construction algorithms in the classification tasks. This experiment aims at observing the output of basis construction algorithms for data points not lying on an algebraic variety, and for ϵ that is tuned for a lower classification error. Following (Livni et al. 2013), the feature vector $\mathcal{F}(\boldsymbol{x})$ of a data point $\boldsymbol{x}$ was defined as

$$\mathcal{F}(\boldsymbol{x}) = \Big(\cdots, \underbrace{\left|g_1^{(i)}(\boldsymbol{x})\right|, \cdots, \left|g_{|G_i|}^{(i)}(\boldsymbol{x})\right|}_{G_i}, \cdots\Big)^{\top}, \qquad (9)$$

Table 2: Classification results. Here, *dim.* denotes the dimensionality of the extracted features, i.e., the length of $\mathcal{F}(\boldsymbol{x})$, and *br.* denotes the basis reduction. The result of the linear classifier (LC) is shown for reference. The results were averaged over ten independent runs.

		VCA	SBC			LC
			$\mathfrak{n}_{\mathrm{c}}$	$\mathfrak{n}_{\mathrm{g}}$	$\mathfrak{n}_{\mathrm{g}}$+br.	
Iris	dim.	80.0	44.7	148	**24.4**	4
	error	0.04	**0.03**	0.04	0.08	0.17
Vowel	dim.	4744	3144	3033	**254**	13
	error	0.44	**0.33**	0.45	0.40	0.67
Vehicle	dim.	8205	6197	5223	**260**	18
	error	0.18	0.22	**0.16**	0.25	0.28

where $G_i = \{g_1^{(i)}, ..., g_{|G_i|}^{(i)}\}$ is the basis set computed for the data points of the i-th class. Because of its construction, the G_i part of $\mathcal{F}(\boldsymbol{x})$ is expected to take small values if $\boldsymbol{x}$ belongs to the i-th class. We trained ℓ_2-regularized logistic regression with a one-versus-the-rest strategy using LIBLINEAR (Fan et al. 2008). We used three small standard datasets (Iris, Vowel, and Vehicle) from the UCI dataset repository (Lichman 2013). Parameter ϵ was selected by 3-fold cross-validation. Because Iris and Vehicle do not have prespecified training and test sets, we randomly split each dataset into a training set (60%) and test set (40%), which were mean-centralized and normalized so that the mean norm of data points is equal to one. The result is summarized in Table 2. Both SBC-$\mathfrak{n}_{\mathrm{c}}$ and SBC-$\mathfrak{n}_{\mathrm{g}}$ achieved a classification error that is comparable or lower than that of VCA with a much lower dimensionality of feature vectors. In particular, the basis reduction drastically reduces the dimensionality of the feature with a slight change in error. Interestingly, the classification error of VCA is mostly comparable with that of other methods despite many spurious vanishing polynomials and redundant polynomials. We consider this is because these polynomials have little effect on the training of a classifier; spurious vanishing polynomials just extend the feature vector with entries that are close to zero, and redundant basis polynomials behaves like a "copy" of other non-redundant basis polynomials. It is interesting to construct the feature vector using discriminative information between classes using discriminative basis construction algorithms, e.g., (Király, Kreuzer, and Theran 2014; Hou, Nie, and Tao 2016). One can consider normalization

and basis reduction for these algorithms, but this is beyond the scope of this paper.

Conclusion

In this paper, we proposed to exploit the gradient of polynomials in the monomial-order-free basis construction of the approximate vanishing ideal. The gradient allows us to access some of the symbolic structure of polynomials and symbolic relations between polynomials. As a consequence, we overcome several theoretical issues in existing monomial-order-free algorithms in a numeraical manner. Specifically, the spurious vanishing problem is resolved in polynomial time complexity for the first time; translation and scaling on the input data lead to consistent changes in the basis set while maintaining the same number polynomials of and same nonlinearity; and redundant basis polynomials are removed from the basis set. These results are achieved efficiently because the gradient of the polynomials at input data points can be exactly computed without differentiation. We believe that this work opens up a new path for monomial-order-free algorithms, which have been theoretically difficult to handle. Future work includes proof of Conjecture 1. It would also be interesting to replace the coefficient normalization of basis construction algorithms in computer algebra with our gradient normalization.

Acknowledgement

This work was supported by JSPS KAKENHI Grant Number 17J07510.

References

Cox, D.; Little, J.; and O'shea, D. 1992. *Ideals, varieties, and algorithms*, volume 3. Springer.

Fan, R.-E.; Chang, K.-W.; Hsieh, C.-J.; Wang, X.-R.; and Lin, C.-J. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research* 9:1871–1874.

Fassino, C., and Torrente, M.-L. 2013. Simple varieties for limited precision points. *Theoretical Computer Science* 479:174–186.

Fassino, C. 2010. Almost vanishing polynomials for sets of limited precision points. *Journal of Symbolic Computation* 45:19–37.

Globerson, A.; Livni, R.; and Shalev-Shwartz, S. 2017. Effective semisupervised learning on manifolds. In Kale, S., and Shamir, O., eds., *Proceedings of the 2017 Conference on Learning Theory (COLT)*, 978–1003. PMLR.

Hashemi, A.; Kreuzer, M.; and Pourkhajouei, S. 2019. Computing all border bases for ideals of points. *Journal of Algebra and Its Applications* 18(06):1950102.

Heldt, D.; Kreuzer, M.; Pokutta, S.; and Poulisse, H. 2009. Approximate computation of zero-dimensional polynomial ideals. *Journal of Symbolic Computation* 44:1566–1591.

Hou, C.; Nie, F.; and Tao, D. 2016. Discriminative vanishing component analysis. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI)*, 1666–1672. AAAI Press.

Iraji, R., and Chitsaz, H. 2017. Principal variety analysis. In *Proceedings of the 1st Annual Conference on Robot Learning*, 97–108. PMLR.

Kehrein, A., and Kreuzer, M. 2006. Computing border bases. *Journal of Pure and Applied Algebra* 205(2):279–295.

Kera, H., and Hasegawa, Y. 2018. Approximate vanishing ideal via data knotting. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI)*, 3399–3406. AAAI Press.

Kera, H., and Hasegawa, Y. 2019. Spurious vanishing problem in approximate vanishing ideal. *arXiv preprint arXiv:1901.08798*.

Kera, H., and Iba, H. 2016. Vanishing ideal genetic programming. In *Proceedings of the 2016 IEEE Congress on Evolutionary Computation (CEC)*, 5018–5025. IEEE.

Király, F. J.; Kreuzer, M.; and Theran, L. 2014. Dual-to-kernel learning with ideals. *arXiv preprint arXiv:1402.0099*.

Li, Q., and Wang, Z. 2017. Riemannian submanifold tracking on low-rank algebraic variety. In *the Thirty-First AAAI Conference on Artificial Intelligence*, 2196–2202. AAAI Press.

Lichman, M. 2013. UCI machine learning repository.

Limbeck, J. 2013. *Computation of approximate border bases and applications*. Ph.D. Dissertation, Passau, Universität Passau.

Livni, R.; Lehavi, D.; Schein, S.; Nachliely, H.; Shalev-Shwartz, S.; and Globerson, A. 2013. Vanishing component analysis. In *Proceedings of the Thirteenth International Conference on Machine Learning (ICML)*, 597–605. PMLR.

Möller, H. M., and Buchberger, B. 1982. The construction of multivariate polynomials with preassigned zeros. In *Computer Algebra. EUROCAM 1982. Lecture Notes in Computer Science*, 24–31. Springer Berlin Heidelberg.

Ongie, G.; Willett, R.; Nowak, R. D.; and Balzano, L. 2017. Algebraic variety models for high-rank matrix completion. In *Proceedings of the Thirth-Forth International Conference on Machine Learning (ICML)*, 2691–2700. PMLR.

Sauer, T. 2007. Approximate varieties, approximate ideals and dimension reduction. *Numerical Algorithms* 45(1):295–313.

Vidal, R.; Yi Ma; and Sastry, S. 2005. Generalized principal component analysis (GPCA). *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27(12):1945–1959.

Wang, L., and Ohtsuki, T. 2018. Nonlinear blind source separation unifying vanishing component analysis and temporal structure. *IEEE Access* 6:42837–42850.

Zhao, Y.-G., and Song, Z. 2014. Hand posture recognition using approximate vanishing ideal generators. In *Proceedings of the 2014 IEEE International Conference on Image Processing (ICIP)*, 1525–1529. IEEE.