博士論文

Learning Algebraic Varieties under Noise
（ノイズ下での代数多様体学習）

48-177405
計良　宥志

指導教員　長谷川禎彦
情報理工学系研究科　電子情報学専攻
東京大学

The University of Tokyo

Doctoral Thesis

Learning Algebraic Varieties under Noise

Author:
Hiroshi Kera

Supervisor:
Dr. Yoshihiko Hasegawa

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

in the

Information & Communication Engineering
Graduate School of Information Science Technology

January 29, 2020

Abstract

Hiroshi Kera

Learning Algebraic Varieties under Noise

How can we retrieve the relations between variables from noisy observation? This has been a fundamental problem in various fields. It has been shown that if data lie on am algebraic variety, which is described by a set of polynomials, we can enjoy several strong theoretical statements in supervised classification, semi-supervised learning, and clustering.

In the present thesis, we address the problem of computing a polynomial system from data. In particular, we would like to obtain a *nice* polynomial system from *noisy* data. In the last decade, this problem has been addressed in the context of the basis construction of the approximate vanishing. However, most existing algorithms are heavily dependent on the use of the monomial order, which is problematic prior information in various applications. Without using the monomial order, existing algorithms encounter various theoretical issues, resulting in a low quality of the output polynomial system. The main contribution of the present thesis is to realize a monomial-order-free basis construction algorithm that overcomes the following issues which the existing algorithms suffer from.

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 monomials in g are all small).

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. For example, a basis polynomial gh is redundant if g is included in the basis set. Determining the redundancy usually needs exponentially costly symbolic procedures and is also unreliable in our approximate setting.

Inconsistency of the basis set with respect to input transformation—Given translated or scaled data points, the output of the basis construction can drastically change in terms of the number of polynomials and their nonlinearity.

For the first issue, we design a monomial-order-free algorithm with normalization, which is proven valid, stable, and optimal. We consider coefficient normalization and gradient normalization. In particular, the gradient normalization is the first to realize the polynomial-time normalization in monomial-order-free basis construction. For the second issue, we reveal that the gradient of basis polynomials reflects the symbolic relation between polynomials. With this result, a basis reduction method is proposed to remove redundant basis polynomials. The last issue is resolved by the gradient normalization. We prove that with the gradient normalization, one can realize a sort of invariance of the basis set with respect to input transformation.

With these results, we establish an efficient basis construction framework that is accompanied by a rich theoretical foundation even without using the monomial order.

Acknowledgements

本研究を進めるにあたり，多くの方からご指導をいただきました．特に指導教員の長谷川禎彦先生に最大の感謝を贈ります．博士課程の 3 年間にわたり継続的で丁寧な指導を受け，多くのことを学びました．特に，常に最先端の研究を志す情熱に多大な影響を受けました．

同期として共に博士課程の 3 年間を過ごすことができたQuoc Hoan Tranさんにも感謝します．分野は違えど刺激を受け，時に協力し合い楽しく研究生活を送ることができました．また研究室秘書の犬束奈穂子さんには予算処理の面で多大なるサポートをいただきました．その他の研究室のメンバーにも研究室での楽しい時間を一緒に過ごしてくれたことに感謝します．伊庭斉志研究室の友人および後輩にも同様の感謝を送ります．

この博士論文にまとめた内容は 3 年間の難しい時間を乗り越えた末の成果です．ここまで辿り着くことができたのはひとえに，長谷川先生の親身なご指導，研究室の面々と学内外の友人や両親の誠実な支えのおかげです．ここに挙げきるることのできなかったその他の方々にもあわせて感謝を述べます．ありがとうございました．

Contents

List of Figures

List of Tables

Chapter 1

Preliminaries

1.1 Definitions and Notations

Definition 1 (Polynomial ring)**.** *The polynomial ring* $\mathbb{K}[x_1, x_2, ..., x_n]$ *in* n *variables* $\{x_i\}$ *over a field* K *is the set of* n*-variate polynomials. In particular, the polynomial ring over real is denote by* $\mathcal{P}_n = \mathbb{R}[x_1, x_2, ..., x_n]$.

Definition 2 (Vanishing Ideal)**.** *Given a set of* n*-dimensional points* $X \subset \mathbb{R}^n$*, the vanishing ideal of* X *is a set of* n*-variate polynomials that take zero values, (*i.e.*, vanish) at all points in* X*. Formally,*

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

Definition 3 (Evaluation vector)**.** *Given a set of points* $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_{|X|}\} \subset \mathbb{R}^n$*, where* $|\cdot|$ *denotes the cardinality of a set, the evaluation vector of polynomial* $h \in \mathcal{P}_n$ *is defined as:*

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

For a set of l *polynomials* $H = \{h_1, h_2, \ldots, h_l\}$*, its evaluation matrix is* $H(X) = (h_1(X)\ h_2(X)\ \cdots\ h_l(X)) \in \mathbb{R}^{|X| \times l}$.

Definition 4 (ϵ-vanishing polynomial)**.** *A polynomial* g *is an* ϵ*-vanishing polynomial for a set of points* X *when the evaluation vector for* X *satisfies* $\|g(X)\| \leq \epsilon$*, where* $\|\cdot\|$ *denotes the Euclidean norm. Polynomials that do not satisfy this condition is called* ϵ*-nonvanishing polynomials.*

Definition 5 (Ideal generated by a polynomial set)**.** *Given a set of polynomials $G \subset \mathcal{P}_n$, the following set of polynomials is an ideal generated by G.*

$$\langle G \rangle = \left\{ \sum_{g \in G} h_g g \mid h_g \in \mathcal{P}_n \right\}.$$

Definition 6 (A polynomial set spanned by a polynomial set)**.** *Given a set of polynomials $F \subset \mathcal{P}_n$, the following set of polynomials are spanned by F.*

$$\mathrm{span}(F) = \left\{ \sum_{f \in F} a_f f \mid a_f \in \mathbb{R} \right\}.$$

1.2 From Polynomials to Vectors

As Definition 2 indicates, the evaluation values of polynomials at given set of points X are of our interest. Hence, a polynomial h can be represented by its evaluation vector $h(X)$. As a consequence, the sum and product of polynomials (say, h_1, h_2) become linear-algebraic operations: $h_1 + h_2$ is mapped to $h_1(X) + h_2(X)$ and $h_1 h_2$ is mapped to $h_1(X) \odot h_2(X)$, where $\odot$ denotes the entry-wise product.

In particular, the weighted sum of polynomials plays an important role in the basis construction. Let us consider a set of polynomials $H = \{h_1, h_2, ..., h_{|H|}\}$. Then, the a weighted sum $\sum_{i=1}^{|X|} w_i h_i$ by $w_i \in \mathbb{R}$ is mapped to $\sum_{i=1}^{|X|} w_i h_i(X)$. In matrix–vector form,

$$(H\boldsymbol{w})(X) = H(X)\boldsymbol{w},$$

where $\boldsymbol{w} = (w_1, w_2, ..., w_{|H|})^\top$, and the product of a polynomial set H and vector $\boldsymbol{w}$ is defined as $H\boldsymbol{w} = \sum_{i=1}^{|H|} w_i h_i$. In words, this relation means that the evaluation vector of a weighted sum of polynomials equals to the weighted sum of evaluation vectors of polynomials. Similarly, the product of a polynomial set H and a matrix $W \in \mathbb{R}^{|H| \times s}$ is defined as $HW := \{H\boldsymbol{w}_1, H\boldsymbol{w}_2, ..., H\boldsymbol{w}_s\}$, where $\boldsymbol{w}_i$ is the i-th column vector of W. With this notation, the following holds:

$$(HW)(X) = H(X)W.$$

In the basis construction, one has to be aware that each operation can be interpreted in two ways—operations of vectors (matrices) and operations of polynomials. A good example is $H(X)W$ shown above; it is a multiplication of matrices,

but at the same time, it is a weighted sum of polynomials.

1.3 Basis Set of Vanishing Ideals

A vanishing ideal $\mathcal{I}(X) \neq \{0\}$ is an infinite-size polynomial set. This is obvious from the fact that given $g \in \mathcal{I}(X)$, it is $gh \in \mathcal{I}(X)$ for any polynomial $h \in \mathcal{P}_n$. Nevertheless, the Hilbert basis theorem tells us that any vanishing ideal can be generate by a finite set of vanishing polynomials. More formally, for any $\mathcal{I}(X)$, there is a finite polynomial set—a basis set—G that generates $\mathcal{I}(X)$; that is,

$$\mathcal{I}(X) = \langle G \rangle = \left\{ \sum_{g \in G} h_g g \mid h_g \in \mathcal{P}_n \right\}.$$

As a linear subspace does not, a vanishing ideal does not have a unique basis set. There are various types of basis sets. Among others, the Gröbner basis and border basis have extensively studied in computer algebra. We will now introduce these two types of bases and then present a relaxed requirement on which we focus.

1.3.1 The Gröbner bases

The most fundamental bases are the Gröbner bases [Buc65], which are used to address numerous important problems in computer algebra such as solving polynomial equations, the ideal membership problems, and cylindrical algebraic decomposition [CLO92; Jir95].

One of the powerful properties of the Gröbner bases is the unique normal form after a reduction as follows. Suppose we have a polynomial f and a set of polynomial B. One can reduce f to a reduced form r as

$$f = \sum_{g \in B} h_g g + r,$$

where $h_g \in \mathcal{P}_n$. Intuitively, r is a reminder after the polynomial divisions of f by polynomials in B. For example, when $f = x^3 + x + 1$ and $B = \{x + 1\}$, then $r = 2x + 1$ because $f = (x^2 - x)(x + 1) + 2x + 1$. However, in the multivariate case, the reduction becomes nontrivial.

- Suppose we divide f by $g \in B$. Then, which monomial should be divided first? For example, if $f = x^2y + y^3$ and $g = x + y$, should we first divide x^2y by x or x^2y by y (or in other way)?

- If f can be divisible by both $g_1, g_2 \in B$. Then, which division should we perform first, division of f by g_1 or by g_2?

It is known that different choices can result in different results.

The first issue is resolved by fixing a monomial order σ. For example, the Degree Lexicographical order defines the order as $1 \prec y \prec x \prec y^2 \prec xy \prec \cdots$, where monomials of larger degree are "larger" (e.g., $x \prec y^2$) and for monomials of the same degree, those of larger degree with respect to x is "larger" (e.g., $xy^2 \prec x^2y$). Once the monomial order is fixed, terms in each polynomial are sorted by the monomial order σ in descending order. The first term of f is called the leading term, and the polynomial division of f by g is performed by first comparing the leading terms of f and g. Therefore, if we adopt the Degree Lexicographical order, the answer to the first issue is that x^2y of f should be divided by x of g first; that is, $f = (xy)g + (-xy^2 + y^3)$. The reminder $-x^2y^2 + y^3$ can be further divided by g. At each division, the leading monomial of the reminder is strictly smaller than that of g, and thus the reduction terminates in finite divisions.

The second issue is elegantly resolved by the theory of the Gröbner bases. If B is a Gröbner basis with respect to a monomial order, the form of r is unique up to a constant factor. If B is not such a basis, than one can transform B into a Gröbner basis by using some algorithms. The definition of the Gröbner basis is as follows.

Definition 7 (Gröbner basis [CLO92])**.** *Fix a monomial order. A finite subset $G = \{g_1, g_2, ..., g_t\}$ of an ideal I is said to be a Gröbner basis (or standard basis) if*

$$\langle \mathrm{LT}(g_1), \mathrm{LT}(g_2), ..., \mathrm{LT}(g_t) \rangle = \langle \mathrm{LT}(I) \rangle,$$

where $\mathrm{LT}(g_i)$ denotes the leading term of g_i) and $\mathrm{LT}(I)$ denotes the set of leading terms of polynomials in I.

In particular, we are interested in the zero-dimensional ideals. In this case, the Gröbner basis has the following property.

Remark 1. *Let G be a Gröbner basis of a zero-dimensional ideal $I \subset \mathcal{P}_n$ with respect to a monomial order σ. The residual classes of monomials $\mathrm{LT}^{\mathrm{c}}(I)$, which is a set of monomials that are not contained in $\mathrm{LT}^{\mathrm{c}}(I)$, form a basis set of vector space $\mathcal{P}_n/I$.*

In words, for any polynomial $h \in \mathcal{P}_n$, its reduced form r after the reduction by G can be represented as a linear combination of monomials in $\mathrm{LT}^{\mathrm{c}}(I)$, i.e.,

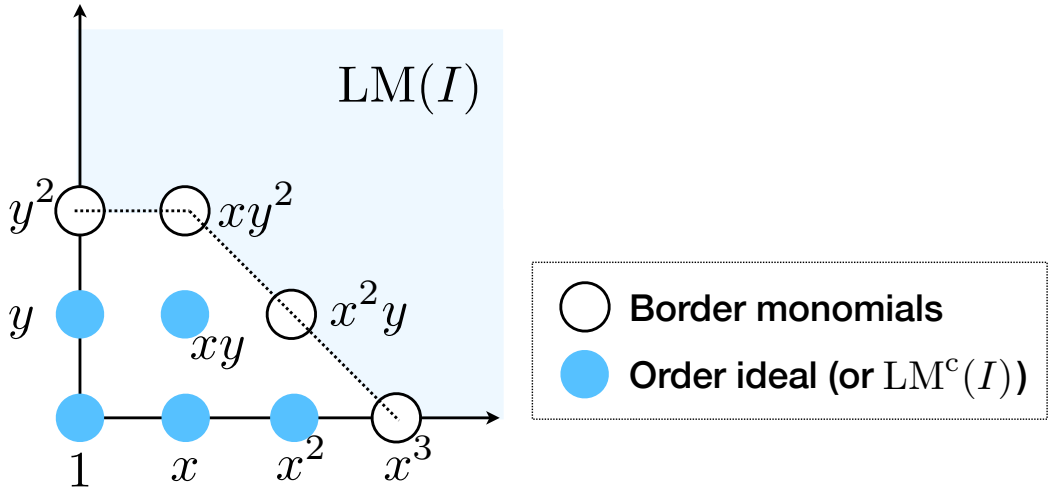

FIGURE 1.1: Illustration of border monomials (empty circles) and order ideal (filled circles in blue) for two-variate case. The order ideal corresponds to $LM^{\mathrm{c}}(I)$, where LM$(\cdot)$ denotes the leading monomial.

$r \in \mathrm{span}(\mathrm{LT}^{\mathrm{c}}(I))$. This fact is evident because for any polynomial $h \in \mathcal{P}_n$, it can be reduced to r, which consists of monomials in $\mathrm{LT}^{\mathrm{c}}(I)$. Note that $\mathrm{LT}^{\mathrm{c}}(I)$ is a finite set when I is a zero-dimensional ideal. By definition of the zero-dimensional ideal, the variety $V(I)$ consists of a finite number of points (say, X). If $\mathrm{LT}^{\mathrm{c}}(I)$ is not a finite set, this implies that any nonzero polynomial $f \in \mathrm{span}(\mathrm{LT}^{\mathrm{c}}(I))$ does not vanish for X[1]. However, the evaluation vector $m(X) \in \mathbb{R}^{|X|}$ of a monomial $m \in \mathrm{LT}^{\mathrm{c}}(I)$ is of finite dimension and this implies that for any $|X|+1$ monomials in $\mathrm{LT}^{\mathrm{c}}(I)$, there is a combination vector by which the evaluation vector of the linear combination of these monomials results in the zero vector.

Basis construction algorithms of zero-dimensional ideals exploit the finiteness of $\mathrm{LT}^{\mathrm{c}}(I)$. Conceptually, given a finite set of points X, these algorithms check monomials from smaller to larger with respect to a fixed monomial order, and form a set of monomial so that the evaluation vectors of these monomials are linearly independent. When the algorithm terminates (in noise-free case), the size of the monomial set is $|X|$.

1.3.2 Border bases

Despite its theoretical elegance and usefulness of the Gröbner bases, it has been known that the computation of the Gröbner bases is numerically unstable. Small difference in coefficients can change the Gröbner bases drastically [Ste04; RA10; Fas10]. This becomes problematic in floating-point computations and also in the

[1] If f vanishes for X, this means that f is divisible by G, which contradicts the fact f does not contain monomials of $\mathrm{LT}(I)$

setting of our interest—computing the approximate basis set of vanishing ideals from noisy data. For more stable computation, border bases have been extensively studied recently, which is robust for s small change in coefficients [Ste04; RA10; Fas10].

The main focus of border bases is zero-dimensional ideals, which form varieties that consists of a finite number of points. In this case, border bases can be regarded as a generalization of the Gröbner bases. Thus, as in the Gröbner bases, the reduction of a polynomial by border basis with respect to a monomial order is unique up to a constant factor. One of the differences between border bases and the Gröbner bases is that a border basis consists of more basis polynomials than the corresponding Gröbner basis. Such redundancy of basis sets results in the robust behavior for small changes in coefficients.

Definition 8 (Order Ideal)**.** *A finite set of monomials $\mathcal{O} \subset \mathbb{M}^n$ is called an order ideal if $t \in \mathcal{O}$ implies that $\mathcal{O}$ contains all monomials dividing t.*

Definition 9 (Border)**.** *The border $\partial\mathcal{O}$ of an order ideal $\mathcal{O}$ is defined as*

$$\partial\mathcal{O} = \cup_{i=1}^{n} x_i\mathcal{O} - \mathcal{O},$$

where x_i is the i-th variable and $x_i\mathcal{O}$ is a set of monomials that are products across x_i and monomials in $\mathcal{O}$.

Definition 10 ($\mathcal{O}$-border basis)**.** *A set of polynomial $G = \{g_1, g_2, ..., g_{|G|}\}$ is called $\mathcal{O}$-border basis if it satisfies the followings.*

- *$\forall i$, g_i can be represented as $g_i - b_i \in \mathrm{Span}(\mathcal{O})$.*
- *The residue classes of the elements of $\mathcal{O}$ forms a basis set of a vector space $\mathcal{P}_n/\mathcal{I}$.*

When G only satisfies the first condition, G is called $\mathcal{O}$-border prebasis.

In words, the first property states that g_i is a linear combination of one border term b_i and monomials in $\mathcal{O}$. The second property is the same as Remark 1. In Fig. 1.1, we provide an example of order ideal and border in the two-variate case.

1.3.3 Defining basis sets without monomial orders

As we have seen, the Gröbner basis is defined using a monomial order and the border base is defined using an order ideal. Thus, we need to assume some prior structure in basis polynomials to compute these basis sets. It is common for many applications that such prior information is not available. Although some works

use methods that rely on monomial orders in applications, they heuristically select the monomial orders [LS04; KI16].

Now, we present a relaxed requirement for the basis set, which are not based on the monomial order nor the order ideal.

Definition 11 (Requirements for the basis set)**.** *Let $G = \bigcup_t G_t$ be a polynomial set, where G_t is a subset of G of the degree-t polynomials. We require G to satisfy the following condition.*

- *For any t, any degree-t vanishing polynomial $g \in \mathcal{I}(X)$ can be generated by $G^t = \bigcup_{\tau=0}^{t} G_\tau$, i.e., $g \in \langle G^t \rangle$.*

Remark 2. *When a polynomial set G satisfies the condition in Definition 11, it readily follows that any vanishing polynomial $g \in \mathcal{I}(X)$ can be generated by G, i.e., $g \in \langle G \rangle$.*

Definition 11 implies that G consists of lower-degree polynomials as possible. For example, let us consider a vanishing ideal $\mathcal{I}(X)$ and its basis set $G = \{g_1, g_2\}$, where $\deg(g_1) = \deg(g_2) - 1 = t$. Suppose that t is the lower degree of the vanishing polynomials in $\mathcal{I}(X)$. Thus, G contains a vanishing polynomial g_1 of the lowest degree and any vanishing polynomial of degree t can be generated from g_1. On the other hand, we also can consider another basis set $G' = \{g'_1, g'_2\}$, where $g'_1 = g_1 + g_2$ and $g'_2 = 2g_1 + g_2$. This also satisfies $\mathcal{I}(X) = \langle G' \rangle$. However, the subset of G' of degree-t polynomials is an empty set; thus, it cannot generate vanishing polynomials of degree t.

In many applications, the polynomials of lower degrees reflects more basic structure of the algebraic varieties. Furthermore, in noisy case, the lower-degree polynomials are less sensitive to noise. For these reasons, the requirement in Definition 11 has an significant importance. The Gröbner bases and border bases also satisfy this requirement when monomial orders that are degree-graded are used. It is common to assume the degree-graded structure for the basis construction. In fact, all the basis construction algorithms that are introduced in this thesis, such as the BM algorithm, the AVI algorithm, VCA, AVICA, and so forth, are designed to compute from lower to higher degree polynomials. This assumption is also valid for many applications—given two polynomials of the same fidelity to data points, the lower-degree one is preferred according to the Occam's razor.

In [Liv+13], it is proven that the output of VCA in the noise-free case satisfies Definition 11. However, the importance of this requirement is not discussed. We

emphasize that this requirement does not assume special structure such as the monomial order and the order ideal. On the other hand, as a consequence of the relaxation, the output basis set do not sustain the properties of the Gröbner bases and border bases; the reduction of a polynomial of the basis set does not results in a unique form. However, such operations that exploit the Gröbner bases and border bases are no longer of our interest because we are focusing on the noisy setting. We receive a set of perturbed points and compute an approximate basis set that consists of approximate vanishing polynomials. The reduction of a polynomial by a set of approximate vanishing polynomials accumulates errors along the divisions and the evaluations of the target polynomial can drastically different before and after the reduction.

Chapter 2

Existing Basis Construction Algorithms of Vanishing Ideals

2.1 Overview

In computer algebra, it is common to receive a set of *polynomials* as input and compute a Gröbner basis (or a border basis). In this case, the complexity of computing a Gröbner basis is known to be doubly exponential [CLO92]. A fascinating result of the BM algorithm is that, by taking a set of *points* as input, it works with a polynomial time complexity with respect to the number of variables and points. In the BM algorithm, monomials are handled from lower to higher degrees and vanishing polynomials are generated by linearly combining monomials. Although the number of monomials grow exponentially according to degree and the number of variables, the BM algorithm exploit a monomial order to restrict the number of monomials; one only has to handle a set of monomials of a polynomial-order size. There are a few monomial-based algorithms that work without monomial orders. For example, Sauer et al. [Sau07] proposed an algorithm to compute approximate H bases, which consist of homogeneous polynomials. Hashemi et al. [HKP19] proposed a method to compute border bases for all possible monomial orders. Unfortunately, none of these algorithms work in polynomial time because the number of monomials are not restricted.

How does the basis construction algorithms in machine learning circumvent the awkward monomial order while enjoying a low computational complexity? Briefly speaking, these algorithms generate vanishing polynomials by linearly combining *polynomials* instead of combining *monomials* and the number of polynomials grows in a polynomial order. Thus, the shift from monomial-based approach to polynomial-based approach results in a new paradigm of the basis construction, i.e., the polynomial-time monomial-order-free basis construction.

Unfortunately, as a consequence of discarding the monomial order, these algorithms lost many advantages properties, resulting in the degradation of the quality of the basis set. We will discuss this issue in the next chapter.

In this chapter, we introduce three basis construction algorithms of approximate vanishing ideals. The first algorithm uses a monomial order and the rests do not. These are all polynomial-time algorithms that receive a set of points and output a basis set of its vanishing ideal. These algorithms have a hyperparameter ϵ of error tolerance and when $\epsilon = 0$, they output basis sets of exact vanishing ideals. A short summary for these algorithms are as follows.

Approximate Buchberger–Möller (ABM) algorithm [Lim13] The ABM algorithm belongs to the line of algorithms which compute border bases from noise-free or noisy points. It is based on milestone algorithms, the Buchberger–Möller (BM) algorithm [MB82] and its extension to compute approximate border bases from noisy points [Hel+09]. There are several algorithms that computes approximate border bases, we introduce the ABM algorithm because consists of more simpler procedures than others. The ABM algorithm requires a monomial order and error tolerance parameter that is a prior information of the extent of noise in input points.

Vanishing Component Analysis (VCA; [Liv+13]) Different from the previous two algorithms, VCA is proposed in machine learning. The design of VCA is more favorable for machine learning. VCA can deal with noisy points as the ABM algorithm, but does not require the monomial order, and can be implemented as the matrix–vector computation. VCA is the most commonly used in various applications such as classification and blind signal separation. However, as a consequence of discarding the monomial order, VCA does not sustain various theoretical properties that computer-algebraic algorithms hold.

Approximate Vanishing Ideal Component Analysis (AVICA; [KKT14]) AVICA is also proposed in machine learning. AVICA has similar advantages to VCA; namely, it can deal with noisy points without monomial order in the matrix–vector computation. The notable point of AVICA is that it uses kernel functions for the basis construction. In [Liv+13], it is proven that the the basis set of vanishing ideal cannot be obtained by the kernel trick, which is a common technique to use kernel functions in machine learning. AVICA computes the value of kernel functions not across data points but between data points and

randomly sampled points instead. Although AVICA provides a few computational advantages and new theoretical aspects, the output is non-deterministic (due to the randomly sampled points) and more hyperparameters needs to be controlled.

2.2 The Approximate Buchberger–Möller algorithm

2.2.1 Procedures

The input of the ABM algorithm is a set of points $X \subset \mathbb{R}^n$, the error tolerance $\epsilon \geq 0$, and a monomial order σ. The output are the basis set $\mathcal{B}$ and an order ideal $\mathcal{O}$ is updated. By initializing $\mathcal{B} = \{\}$ and $\mathcal{O} = 1$, the following steps are performed for degree $t = 1, 2, ...$

Step 1: Generate a set of new monomials L A set L of new monomials of degree t are generazed as follows.

$$L = \left\{ b \mid b \in \bigcup_{i=1}^{n} x_i \mathcal{O} - \mathcal{O}, \deg(b) = t \right\}.$$

If L is the empty set, then terminate with output $\mathcal{B}$ and $\mathcal{O}$.

Step 2: Check the monomials in L **one by one** Select a least monomial $b \in L$ with respect to monomial order σ and remove b from L. Solve the following eigenvalue problem.

$$\begin{pmatrix} b(X) & O(X) \end{pmatrix}^{\top} \begin{pmatrix} b(X) & O(X) \end{pmatrix} \boldsymbol{v}_{\min} = \lambda_{\min} \boldsymbol{v}_{\min}, \tag{2.1}$$

where $\lambda_{\min}$ is the smallest eigenvalue and $\boldsymbol{v}_{\min}$ is the corresponding eigenvector.

Step 3: Update basis sets If $\sqrt{\lambda_{\min}} \leq \epsilon$, then append a polynomial $g = (\{b\} \cup O)\boldsymbol{v}_{\min}$ to $\mathcal{B}$. Otherwise, append b to $\mathcal{O}$. Go back to Step 2 if L is not the empty set.

2.2.2 Details

At Step 1, we generate degree-t border monomials of $\mathcal{O}$. The key step of the basis construction is Step 2. Over the basis construction, we handle monomials from lower to higher degrees, and for each degree, from smaller to large monomials according to a monomial order σ. A set of monomial $\mathcal{O}$ is updated so that it always has the full-rank evaluation matrix $\mathcal{O}(X)$. To be precise, O is maintained so that the smallest eigenvalue of $O(X)$ is always larger than ϵ. A new monomial b is appended to O only if the new $O(X)$ keeps this condition. Intuitively, b is appended to $\mathcal{O}$ if $b(X)$ is sufficiently linearly independent to the evaluation vectors of monomials in $\mathcal{O}$. When $\epsilon = 0$, it is actually checking the linear independency of the evaluation vectors. If $b(X)$ is (almost) linearly dependent on the column vectors of $O(X)$, it implies that we can obtain an (approximate) vanishing polynomial; that is, with the vector $\boldsymbol{v}_{\min} = (v_1, v_2, ..., v_{|O|+1})$, we have a polynomial g as follows.

$$g := v_1 b + v_2 o_1, v_3 o_2, ..., v_{|O|+1} o_s,$$

where $\mathcal{O} = \{o_1, o_2, ..., o_s\}$. The extent of vanishing of g for X equals to the square root of the smallest eigenvalue, i.e., $\|g(X)\| = \sqrt{\lambda_{\min}}$ because

$$\|g(X)\|^2 = \boldsymbol{v}_{\min}^\top \begin{pmatrix} b(X) & O(X) \end{pmatrix}^\top \begin{pmatrix} b(X) & O(X) \end{pmatrix} \boldsymbol{v}_{\min} = \lambda_{\min}$$

The output of the ABM algorithm has the following properties.

Remark 3 ([Lim13])**.** *The output $(\mathcal{B}, \mathcal{O})$ of the ABM algorithm for input (X, ϵ) satisfies the followings.*

- *Any polynomial in $g \in \mathcal{B}$, the coefficient norm of g is unity and $\|g(X)\| \leq \epsilon$.*
- *The is no polynomial of the unit coefficient norm in* $\mathrm{span}(\mathcal{O})$ *that is ϵ-vanishing for X.*
- *If $\mathcal{O}$ is an order ideal of terms, then the set $\widetilde{\mathcal{B}} = \{g/\mathrm{LC}(g) \mid g \in \mathcal{B}\}$ is an $\mathcal{O}$-border prebasis, where* $\mathrm{LC}(g)$ *denotes the coefficient of the leading monomial of g.*
- *If $\mathcal{O}$ is an order ideal of terms, then the set $\widetilde{\mathcal{B}}$ is an δ-approximate border basis with $\delta = 2\|X\|_{\max}/\min_i |\gamma_i| + \nu/(\min_i |\gamma_i|)^2$, where $\|X\|_{\max}$ denotes the maximal abosolute coordinate in X and γ_i denotes the coefficient of the border monomial of $g_i \in \mathcal{B}$.*

- *If $\epsilon = 0$, then the algorithm produces the same results as the BM algorithm for border bases [].*

Refer to [Lim13] for details.

In general, the number of monomials grows exponentially with respect to degree and number of variables. However, in the ABM algorithm (to be precise, in the framework of the BM algorithm [MB82]), we only have to deal with much fewer number of monomials. More specifically, we only have to handle at most $|X|n$ monomials, where n is the number of variables).

Remark 4. *In the basis construction by the ABM algorithm, for any degree t, $|L| \leq |X|n$ at Step 1.*

Note that it is $|\mathcal{O}| \leq |X|$ because $\mathcal{O}$ is designed to have a full-rank evaluation matrix $\mathcal{O}(X) \in \mathbb{R}^{|X| \times |\mathcal{O}|}$ in the ABM algorithm. The equality holds when the algorithm runs with $\epsilon = 0$. From another point of view, this relates to the discussion in Section 1.3.3: when we consider an exact vanishing ideal for finite data points, the complementary monomial set of $\mathrm{LT}(\langle G \rangle)$ is a finite set and its size is $|X|$. Now, it can be readily proven based on the fact that $|\mathcal{O}| \leq |X|$ and L is a subset of $\bigcup_{i=1}^{n} x_i \mathcal{O}$ by construction.

2.3 Vanishing Component Analysis

2.3.1 Procedures

The input of VCA is a set of points $X \subset \mathbb{R}^n$ and the error tolerance $\epsilon \geq 0$. The output is a basis set G and a basis set F. By initializing $G_0 = \{\}$ and $F_0 = 1/\sqrt{|X|}$, the following steps are performed for degree $t = 1, 2, \ldots$. In the following, we use notations $G^t = \bigcup_{\tau=0}^{t} G_\tau$ and $F^t = \bigcup_{\tau=0}^{t} F_\tau$.

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.

$$C_t = C_t^{\mathrm{pre}} - F^{t-1}F^{t-1}(X)^{\dagger}C_t^{\mathrm{pre}}(X), \tag{2.2}$$

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

Step 2: Perform SVD We perform SVD on $C_t(X)$.

$$C_t(X) = UDV^{\top},$$

where $U \in \mathbb{R}^{|X|\times|X|}$ and $V \in \mathbb{R}^{|C_t|\times|C_t|}$ are orthonormal matrices and D is a rectangular diagonal matrix with singular values $\sigma_1, \sigma_2, ..., \sigma_{\min(|C_t|,|X|)}$ along its diagonal.

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

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

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

2.3.2 Details

At Step 1, the orthogonalization procedure (2.2) makes the column space of $C_t(X)$ orthogonal to that of $F^{t-1}(X)$. This aims at focusing on the subspace of $\mathbb{R}^{|X|}$ that cannot be spanned by the evaluation vectors of polynomials of degree less than t. Note that

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

At Step 3, a polynomial $C_t\boldsymbol{v}_i$ is classified as an ϵ-vanishing polynomial if $\sigma_i \leq \epsilon$ because σ_i equals the extent of vanishing of a polynomial $C_t\boldsymbol{v}_i$. In fact,

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

Note that for nonvanishing polynomials in F_t, the polynomials are rescaled by $1/\sigma_i$ so that the norm of the evaluation is equal to unity. This is introduced to

avoid overflow and underflow in the computation. As a consequence, polynomials in F is not necessarily ϵ-nonvanishing polynomials. If one wants F to be a set of ϵ-nonvanishing polynomials, it can be simply resolved. We keep $\widetilde{F}_t = \{C_t \boldsymbol{v}_i \mid \sigma_i > \epsilon\}$ for output and use F_t for the computation in higher degrees. When the algorithm terminates (say, at T), then output $\widetilde{F}^T$ instead of F^T.

Remark 5. *The VCA algorithm has the following properties [Liv+13].*

- *If VCA runs with $\epsilon = 0$, then G generates $\mathcal{I}(X)$*
- *If VCA runs with $\epsilon = 0$, any polynomial h can be written as $h = f + g$, where $f \in \mathrm{span}(F)$ and $g \in \langle G \rangle$.*
- *If VCA runs with $\epsilon = 0$, then for all t, any polynomial h of degree at most t can be written as $h = f + g$, where $f \in \mathrm{span}(F^t)$ and $g \in \langle G^t \rangle$.*
- $|F| \leq |X|$ *and* $|G| \leq |F|^2 \cdot \min(|F|, n)$

In particular, the third statement is the most fundamental. The first and second statements derive from the third statement. The basis set G also satisfies the requiement in Definition 11. Refer to [Liv+13] for more properties.

2.4 Approximate Vanishing Ideal Component Analysis

2.4.1 Procedures

The input of AVICA is a set of points $X \subset \mathbb{R}^n$, an error tolerance $\epsilon \geq 0$, a linear polynomial kernel $k(\boldsymbol{x}, \boldsymbol{y}) = \theta + \boldsymbol{x}^\top \boldsymbol{y}$, where θ is also an input hyperparameter, a probability distribution $\mathcal{D}$ for random sampling, the number of random samples N, and a degree T to terminate. The output is a basis set G and a basis set F.

As the initialization, we first obtain a set of random samples $Y = \{\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_N\}$ from the distribution $\mathcal{D}$. Then, generate a set of polynomials $K = \{k(\cdot, \boldsymbol{y}_1), k(\cdot, \boldsymbol{y}_2), ..., k(\cdot, \boldsymbol{y}_N)\}$. We denote the evaluation matrix of K by $K(X; Y) \in \mathbb{R}^{|X| \times N}$. We also set $K_0 = \{1\}$. Then, the following steps are repeated for degree $t = 1, 2, ..., T$.

Step 1: Generate a set of candidate polynomials We generate a candidate polynomials of degree t as follows.

$$K_t = K_{t-1} \odot K.$$

Step 2: Perform SVD We perform SVD on $K_t(X;Y)$.

$$K_t(X;Y) = UDV,$$

where $U \in \mathbb{R}^{|X|\times|X|}$ and $V \in \mathbb{R}^{N\times N}$ are orthonormal matrices and D is a rectangular diagonal matrix with singular values $\sigma_1, \sigma_2, ..., \sigma_{\min(|X|,N)}$ along its diagonal.

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

$$G_t = \{K_t\boldsymbol{v}_i \mid \sigma_i \leq \epsilon\theta^t\},$$
$$F_t = \{K_t\boldsymbol{v}_i \mid \sigma_i > \epsilon\theta^t\}.$$

Set $K_t = \widetilde{U}\widetilde{D}\widetilde{V}$, where $\widetilde{V}$ represents a matrix whose columns consist of those of V which are used to generate F_t ($\widetilde{U}$ and $\widetilde{V}$ are defined similarly).

2.4.2 Details

Remark 6. *Here, we present a slightly simplified version of AVICA to reduce the parameters. In the original algorithm, at Step 3, the machine precision $\epsilon_{\text{machine}}$ is considered and $G_t = \{K_t\boldsymbol{v}_i \mid \epsilon_{\text{machine}} \leq \sigma_i \leq \epsilon\theta^t\}$. In addition, it stores $q = \sigma_i\theta^t$ for each polynomial as its importance. This quantity q is used when one wants to sort obtained polynomials from more informative to less informative ones.*

The interesting idea of AVICA is that the vanishing polynomials are generated by using the polynomial kernel. In [Liv+13], it is proven that vanishing polynomials cannot be generated from the usual kernel trick. More specifically, let $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ be a reproducing kernel function (e.g., the polynomial kernel). We consider the Gram matrix $K(X,X) \in \mathbb{R}^{|X|\times|X|}$ for $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_{|X|}\}$, where the i-th entry of $K(X,X)$ is $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Then, the solution of the following problem

$$K(X,X)\boldsymbol{v} = \boldsymbol{0}, \tag{2.3}$$

generates a polynomial $g(\boldsymbol{x}) = \sum_{i=1}^{|X|} k(\boldsymbol{x}, \boldsymbol{x}_i)v_i$. Here, g consists of the linear combination of kernel functions that are evaluated for each point $\boldsymbol{x}_i$. This kind of technique is called the kernel trick and widely used in machine learning for extending linear methods to nonlinear methods. Typically, the right hand side of Eq. (2.3) is a nonzero vector, or $\boldsymbol{v}^\top K(X,X)\boldsymbol{v}$ is maximized instead.

Interestingly, in our case, the polynomial g generated by the kernel trick is the zero polynomial, although $\boldsymbol{v}$ is a nonzero vector in general. Intuitively, this is because the kernel trick only uses the relative relations between points but the coefficients of a vanishing polynomial relies not on the relative relations. A rough explanation is as follows. For instance, let us consider the case where X lies on a variety V and let us apply a rotation by an orthonormal matrix U to obtain X' and V'. Then, for any $\boldsymbol{x}_1, \boldsymbol{x}_2 \in X$ and the corresponding points $\boldsymbol{x}'_1, \boldsymbol{x}'_2 \in X'$, it is $(\boldsymbol{x}'_1)^\top \boldsymbol{x_2}' = (U\boldsymbol{x}_1)^\top (U\boldsymbol{x}_2) = \boldsymbol{x}_1^\top \boldsymbol{x}_2$. Therefore, $K(X, X)$ and $K(X', X')$ yields the same vanishing polynomial $g(\boldsymbol{x}) = \sum_{i=1}^{|X|} k(\boldsymbol{x}, \boldsymbol{x}_i) v_i$[1], which implies that the coefficient vector of g (note that it is not $\boldsymbol{v}$) is invariant to the rotation. However, it seems less likely that a vanishing polynomial g which is calculated for X is also vanishing for $X' = XU$ for arbitrary orthonormal matrix U—unless g is the zero polynomial.

The above discussion only provides an intuitive explanation of why usual kernel trick does not work for the calculation of the vanishing polynomials. Refer to [Liv+13] for the formal proof in the case of the reproducing kernel functions.

The key idea of AVICA to exploit kernel functions for computing vanishing polynomials is to consider $K(X, Y)$ instead of $K(X, X)$, where Y is a set of randomly sampled points. We solve the following problem,

$$K(X, Y)\boldsymbol{v} = \boldsymbol{0},$$

and generate a polynomial $g(\boldsymbol{x}) = \sum_{\boldsymbol{y} \in Y} k(\boldsymbol{x}, \boldsymbol{y}) v_{\boldsymbol{y}}$. Now, $\boldsymbol{y}$ is a randomly sampled point and in general, it does not lie on the variety to which X belongs. Therefore, g is represented from the outside of the variety.

Now, we explain the algorithm of AVICA. At the initial step, we calculate a set of polynomial $K = \{k(\cdot, \boldsymbol{y}_1), k(\cdot, \boldsymbol{y}_2), ..., k(\cdot, \boldsymbol{y}_N)\}$, where $k(\boldsymbol{x}, \boldsymbol{y}_i) = \theta + \boldsymbol{x}^\top \boldsymbol{y}$. Thus, for each $\boldsymbol{y}_i$, $k(\cdot, \boldsymbol{y}_i)$ is a linear polynomial. Both the ABM algorithm and VCA consider $L = C_1^{\text{pre}} = \{x_1, x_2, ..., x_n\}$ as a set of linear polynomials. Namely, if we randomly sample enough number of points, then the expressive power of K is equivalent to $L = C^{\text{pre}}$. Then, at Step 1, degree-t polynomials are generated from polynomials of degree $t-1$ and 1. At Steps 2 and 3 are almost identical to those of VCA.

The only concern is that weather the capacity of $K_t(X; Y)$ is enough large that we can discover all the degree-t nonvanishing and vanishing polynomials that are necessary for basis sets. In [KKT14], it is discussed that the capacity of

[1] We here focus on the polynomial kernel.

$K_t(X;Y)$ is sufficiently large when $|X|$ and $|Y|$ are larger than the dimension of the algebraic variety where X lie.

Different from the previous two algorithms, the output polynomial set G of AVICA at $\epsilon = 0$ forms a basis set of the vanishing ideal only when sufficiently many random points are sampled and the termination degree T is large enough (T is a hyperparameter).

Chapter 3

Monomial-Order-Free Basis Construction with Normalization

In this chapter, we introduce a monomial-order-free basis construction with normalization. We discuss fundamental issues that arises in the monomial-order-free basis construction. In particular, we tackle the following issues.

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 monomials in g are all small).

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[1]. Determining the redundancy usually needs exponentially costly symbolic procedures and is also unreliable in our approximate setting.

Inconsistency of the basis set with respect to input transformation—Given translated or scaled data points, the output of the basis construction can drastically change in terms of the number of polynomials and their nonlinearity.

The first and the second issues relate to the quality of the output basis set. A basis set is desired to be succinct description of the algebraic variety. However, if a basis set contains many polynomials close to the zero polynomial, this implies that the basis set is noisy; the symbolic relations represented by such spurious vanishing polynomials do no reflect the actual structure of input data points. Similarly, if the basis set contains many redundant basis polynomials, this means that the basis set is not compact enough; the same relations between variables are repeatedly described by different polynomials.

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

Both issues can be elegantly resolved by using a monomial order and/or symbolic operations. However, in many applications, proper monomial orders are unknown. Also, symbolic operations are common only in limited research areas and it is computationally costly. Therefore, we are interested in how to resolve the aforementioned fundamental issues in the numerical computation.

The third issue contradicts an intuition that the intrinsic structure of an algebraic variety does not change by a translation or scaling on data. We would like the basis sets to share the same number of polynomials at each degree. However, most basis construction algorithms including ABM, VCA, and AVICA, do not have the invariance for translation and scaling on data regardless how well the parameter ϵ is controlled. Note that VCA has a invariance for the translation on data but not for the scaling.

In this chapter, we first introduce a basis construction algorithm with normalization, which is a key tool of our analysis and discussion. Then, we discuss each of the three issues in detail and present approaches to resolve them.

3.1 Simple Basis Construction

We present the Simple Basis Construction (SBC) algorithm, which is a general framework of monomial-order-free basis construction algorithms with normalization. This is the first algorithm that takes normalization into account in the basis construction in polynomial time without using a monomial order. The SBC algorithm is designed based on VCA, and many monomial-order-free algorithms can be discussed using this framework.

3.1.1 Procedures

The input of SBC is a set of points $X \subset \mathbb{R}^n$ and the error tolerance $\epsilon \geq 0$. The output is a basis set G and a basis set F. By initializing $G_0 = \{\}$ and $F_0 = m$, where m is a nonzero constant polynomial, the following steps are performed for degree $t = 1, 2, ...$ In the following, we use notations $G^t = \bigcup_{\tau=0}^{t} G_\tau$ and $F^t = \bigcup_{\tau=0}^{t} F_\tau$.

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.

$$C_t = C_t^{\mathrm{pre}} - F^{t-1}F^{t-1}(X)^{\dagger}C_t^{\mathrm{pre}}(X), \tag{3.1}$$

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

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

$$C_t(X)^{\top}C_t(X)V = \mathfrak{N}(C_t)V\Lambda, \tag{3.2}$$

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 $F = F^t$.

3.1.2 Toy example

We demonstrate the computation of SBC-I, where $\mathfrak{N}(C_t) = I$ at each t. We will also encounter the spurious vanishing problem in this example. Let us consider SBC-I for $X = \{(1+\xi, 1), (1, 1+\xi), (-1+\xi, -1+\xi), (-1, -1)\}$, where $\xi = 0.1$. We use $\epsilon = \sqrt{4\xi^2} = 2\xi = 0.2$. The constant polynomial m at degree 0 is set to $m = 1$, and thus, $F_0 = \{f_1 := 1\}, G_0 = \{\}$. We use x and y for variables and the values are rounded for illustration while keeping the use of "=" notation. The coefficient vector of a polynomial h is denoted by $\mathfrak{n}_{\mathrm{c}}(h)$, e.g., $\mathfrak{n}_{\mathrm{c}}(1 - x + 2xy) = (1, -1, 0, 0, 2, 0)^{\top}$, and $\|\mathfrak{n}_{\mathrm{c}}(h)\|$ is called the coefficient norm of h.

Degree $t = 1$ At `Step 1`, the candidate polynomials are generated as follows.

$$\begin{aligned} C_1 &= \{x, y\} - \{1\}\frac{1}{4}\begin{pmatrix} 1 & 1 & 1 & 1 \end{pmatrix}\begin{pmatrix} 1.1 & 1 \\ 1 & 1.1 \\ -0.9 & -0.9 \\ -1 & -1 \end{pmatrix}, \\ &= \{x - 0.05, y - 0.05\}. \end{aligned}$$

The eigenvalues of $C_1(X)$ is $\{0.01, 8.01\}$, and the corresponding eigenvectors are $\boldsymbol{v}_1 = (-0.707, 0.707)^\top$ and $\boldsymbol{v}_2 = (0.707, 0.707)^\top$. According to the square roots of eigenvalues $\{\sqrt{0.01}, \sqrt{8.01}\} = \{0.1, 2.83\}$,

$$\begin{aligned} G_1 &= \{g_1 := C_1\boldsymbol{v}_1 = -0.707(x - 0.05) + 0.707(y - 0.05)\}, \\ F_1 &= \{f_2 := C_1\boldsymbol{v}_2 = 0.707(x - 0.05) + 0.707(y - 0.05)\}. \end{aligned}$$

Degree $t = 2$ The set of precandidate polynomials is $C_2^{\text{pre}} = \{f_2^2\} = \{(0.707(x - 0.05) + 0.707(y - 0.05))^2\}$, and thus, the set of candidate polynomials of degree 2 is

$$\begin{aligned} C_2 &= \{(0.707(x - 0.05) + 0.707(y - 0.05))^2\} \\ &\quad - \{1, f_2\}\begin{pmatrix} \mathbf{1}_4 & f_2(X) \end{pmatrix}^\dagger f_2^2(X), \\ &= \{0.5x^2 + xy + 0.5y^2 - 0.096x - 0.0963y - 2.00\}, \end{aligned}$$

where $\mathbf{1}_4 \in \mathbb{R}^4$ is the all-one vector. The eigenvalue and eigenvector of $C_2(X)$ are 0.78 and (1.0), respectively. Thus, $G_2 = \{\}$ and $F_2 = \{f_3 := 0.5x^2 + xy + 0.5y^2 - 0.096x - 0.0963y - 2.00\}$.

Degree $t = 3$ The set of precandidate polynomials is $C_3^{\text{pre}} = \{f_2 f_3\}$ and thus, the set of candidate polynomials of degree 3 is

$$\begin{aligned} C_3 &= \{f_2 f_3\} \\ &\quad - \{1, f_2, f_3\}\begin{pmatrix} \mathbf{1}_4 & f_2(X) & f_3(X) \end{pmatrix}^\dagger (f_2 f_3)(X), \\ &= \{0.354x^3 + 1.061x^2y + 1.061xy^2 + 0.354y^3 \\ &\quad + 0.601x^2 + 1.202xy + 0.601y^2 - 1.549x \\ &\quad - 1.549y - 2.671\}. \end{aligned}$$

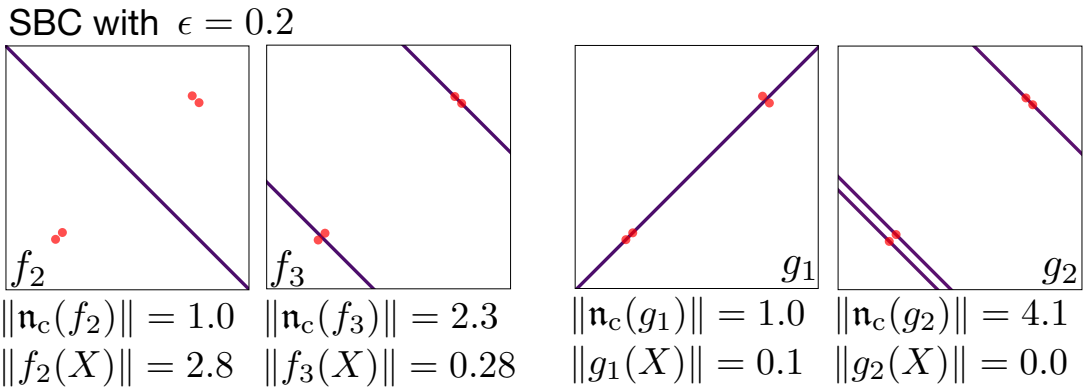

FIGURE 3.1: ϵ-nonvanishing polynomials (f_2 and f_3) and ϵ-vanishing polynomials (g_1 and g_2) obtained by the SBC algorithm for four points around $(-1, 1)$ and $(-1, -1)$ with $\epsilon = 0.2$. The coefficient vector of a polynomial h is denoted by $\mathfrak{n}_c(h)$. Here, f_3 is classified as a nonvanishing polynomial based on its extent of vanisng $\|f_3(X)\| = 0.28 > \epsilon$, which is overrated because of the relatively large coefficient norm $\|\mathfrak{n}_c(f_3)\| = 2.3$.

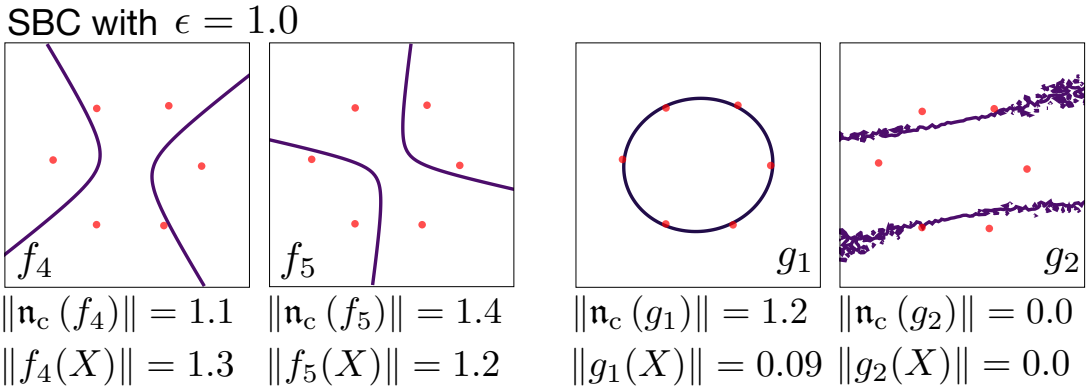

FIGURE 3.2: Some of the ϵ-nonvanishing polynomials (f_4 and f_5) and ϵ-vanishing polynomials (g_1 and g_2) obtained by the SBC algorithm for perturbed points X that are sampled from a unit circle (see the main text for detail). with $\epsilon = 1.0$. Here, g_2 is classified as an ϵ-vanishing polynomials based on its extent of vanishing $\|g_2(X)\| = 0.0 \leq \epsilon$, which is underrated because of its small coefficient norm $\|\mathfrak{n}_c(g_2)\| = 0.0$ (only approximately equal to zero but rounded off as 0.0).

The eigenvalue and eigenvector of $C_3(X)$ are 0 and (1.0), respectively. Thus, $G_3 = \{g_2\} := C_3$ and $F_2 = \{\}$, and the algorithm terminates with the outputs $G = \bigcup_{t=0}^{3} G_t = \{g_1, g_2\}$ and $F = \bigcup_{t=0}^{3} F_t = \{1, f_2, f_3\}$.

We provide contour plots of the obtained polynomials (except f_1) in Fig. 3.1. The coefficient norm and the extent of the vanishing of these polynomials are listed below the plots. Here, f_3 is classified as a nonvanishing polynomial based on $\|f_3(X)\| = 0.28 > \epsilon$, although the lines approximately pass through the points. This results from the large coefficient norm of f_3, i.e., $\|\mathfrak{n}_c(f_3)\| = 2.3$. In other words, f_3 is a spurious nonvanishing polynomial. In fact, once f_3 is normalized to $f_3' := f_3/\|\mathfrak{n}_c(f_3)\|$, it becomes an ϵ-vanishing polynomial because

of $\|f_3'(X)\| \approx 0.12 \leq \epsilon$. We observe the opposite case when we apply SBC to another set of points $X = \{(0.53, 0.87), (-0.49, 0.83), (-1.1, 0.1), (-0.5, -0.81), (-0.46, -0.83), (0.99, 0.02)\}$ with $\epsilon = 1.0$. This X is generated by perturbing points of $X_0 = \{(\cos(k\pi/3), \sin(k\pi/3))\}_{k=0,1,\ldots,5}$ on a unit circle with the zero-mean additive Gaussian noise (the standard deviation is 0.01). As shown in Fig. 3.2, g_2 is classified as an ϵ-vanishing polynomial, although the lines on the plot do not pass through the points at all. This is because the magnitude of the coefficients of g_2 is extremely small. In other words, g_2 is a spurious vanishing polynomial.

3.1.3 Analysis

We will now analyze the SBC algorithm. In particular, we discuss the following questions.

- What is the normalization matrix $\mathfrak{N}(C_t) \in \mathbb{R}^{|C_t| \times |C_t|}$ in Eq. (3.2)?
- What is the usefulness of the generalized eigenvalue problem Eq. (3.2)?
- Does the SBC algorithm output a basis set that satisfies the requirement of Definition 11?

The overview of the discussion is as follows. We first define a normalization mapping $\mathfrak{n} : \mathcal{P}_n \to \mathbb{R}^\ell$. With $\mathfrak{n}$, the (i, j)-th entry of $\mathfrak{N}(C_t)$ is defined as the inner product $\mathfrak{n}(c_i)^\top \mathfrak{n}(c_j)$ for $C_t = \{c_1, c_2, ..., c_{|C_t|}\}$. Solving a generalized eigenvalue problem Eq. (3.2) yields a polynomial h, which is normalized with respect to $\mathfrak{n}(h)$, i.e., $\|\mathfrak{n}(h)\| = 1$. For example, if coefficient vectors are used for normalization, i.e., if $\mathfrak{n} = \mathfrak{n}_\mathrm{c}$, then basis polynomials are normalized to have unit coefficient vectors. In this way, we can avoid including spurious vanishing polynomials (polynomials that are close to the zero polynomial) in the basis set. We will also show that the gradient normalization does not only resolve the spurious vanishing problem but also resolve the "inconsistency issue" (the second issue). We also prove that the output of the SBC algorithm satisfies the same properties of Remark 5 (and so as Definition 11.

Normalization mapping

A polynomial g is exactly vanishing for X when $\|g(X)\| = 0$. This implies that X is a subset of roots of g. Since a scaling does not change the roots of a polynomial, kg $(k \neq 0)$ is also vanishing for X. However, this is not the case for the approximate vanishing. For example, g is ϵ-*non*vanishing for X, i.e.,

$\|g(X)\| > \epsilon$, it can turn into ϵ-vanishing by scaling kg for small k. More precisely, any g turns into approximate vanishing for X by using sufficiently small k.

To sidestep this issue, it is necessary to normalize g before evaluation as follows.

$$\widetilde{g} = \frac{g}{\|\mathfrak{n}(g)\|},$$

where $\mathfrak{n} : \mathcal{P}_n \to \mathbb{R}^\ell$ is a normalization mapping. For example, we can consider the coefficient normalization with $\mathfrak{n} = \mathfrak{n}_{\mathrm{c}}$, where $\mathfrak{n}_{\mathrm{c}}(g)$ is the coefficient vector of g. For any $k \neq 0$, we obtain the same $\widetilde{g}$ and thus, we can measure the approximate vanishing in a consistent way.

Such a normalization is an essential part of the basis construction of the approximate vanishing ideal. In fact, the computer-algebraic basis construction algorithms adopt the coefficient normalization. They construct a polynomial from a linear combination of distinct monomials by a unit combination vector (recall $(b \cup \mathcal{O})\boldsymbol{v}_{\min}$ at Step 3 in the ABM algorithm). Although the number of monomials grow exponentially according to degree and the number of variable, the linear combination can be performed efficiently because it is only necessary to handle not all but only a limited number of monomials by exploiting the monomial order. On the other hand, monomial-order-free algorithms such as VCA and AVICA does not takes normalization into account because it is unknown how to optimally and efficiently introduce a normalization into the basis construction. As we discuss later, the SBC algorithm provides a simple framework to introduce a normalization in optimal fashion.

Now, we present the conditions that we impose on the normalization mapping.

Definition 12 (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.

The third condition implies that $\|\mathfrak{n}(h)\|$ takes the zero value only for polynomials that are unnecessary for the basis set. For example, it is evident that the coefficient normalization mapping $\mathfrak{n}_{\mathrm{c}}$ satisfies the first two conditions. Since $\|\mathfrak{n}_{\mathrm{c}}(h)\| = 0$ if and only if h is the zero polynomial, the third condition is also satisfied. Later, we consider the coefficient normalization mapping $\mathfrak{n}_{\mathrm{g}}$. Although $\|\mathfrak{n}_{\mathrm{g}}(h)\|$ can take the zero value for a nonzero polynomial h, we will show that $\mathfrak{n}_{\mathrm{g}}$ can be used in the SBC algorithm because $\|\mathfrak{n}_{\mathrm{g}}(h)\| \neq 0$ if h is necessary for the basis set.

The validity of the SBC algorithm

With a valid normalization mapping, the output of the SBC algorithm satisfies the same properties listed in Remark 5 as VCA.

Theorem 1. *Let $\mathfrak{n}$ be a valid normalization mapping for the SBC algorithm. When SBC runs with $\mathfrak{n}$ and $\epsilon = 0$ 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 $g' \in \langle G^t \rangle$.*

Proof. The SBC algorithm with $\mathfrak{N}(C_t) = I$ (SBC-I) is identical to VCA up to a constant factor in basis polynomials. Specifically, VCA set $F_0 = \{m\} = \{1/\sqrt{|X|}\}$, and normalize polynomials $f \in F_t$ by $\|f(X)\|$ at each degree. Thus, from Theorem 5.2 in [Liv+13], which shows that VCA satisfies Theorem 1, we can conclude that SBC-I also satisfies Theorem 1.

Next, we prove the claim by induction with respect to degree t for general $\mathfrak{n}$ (SBC-$\mathfrak{n}$). We compares two processes (SBC-I and SBC-$\mathfrak{n}$) and . We refer to the former and the latter, respectively, as SBC-I and SBC-$\mathfrak{n}$. If we use symbols such as G_t and F_t in SBC-I, we put tildes on the corresponding symbols such as $\widetilde{G}_t$ and $\widetilde{F}_t$ in SBC-$\mathfrak{n}$.

Let F_t and G_t be the basis sets of nonvanishing polynomials and vanishing polynomials, respectively, obtained at degree t iteration in SBC-I. We know that

collecting F_t and G_t yields complete basis sets for both nonvanishing and vanishing polynomials. Here, we prove the claim by comparing $\widetilde{F}_t$ and $\widetilde{G}_t$ with F_t and G_t. Specifically, we show $\mathrm{span}(\widetilde{F}_t) = \mathrm{span}(F_t)$ and $\langle \widetilde{G}^t \rangle = \langle G^t \rangle$. Note that $\mathrm{span}(\widetilde{F}_t) \subset \mathrm{span}(F_t)$ and $\langle \widetilde{G}^t \rangle \subset \langle G^t \rangle$ are obvious because $\widetilde{F}_t$ and $\widetilde{G}_t$ are generated by assigning additional constraints of normalization on the generation of F_t and G_t. Thus, our goal is to prove the reverse inclusion $\mathrm{span}(\widetilde{F}_t) \supset \mathrm{span}(F_t)$ and $\langle \widetilde{G}^t \rangle \supset \langle G^t \rangle$.

At $t = 1$, it is obvious that $\mathrm{span}(F_1) = \mathrm{span}(\widetilde{F}_1)$ and $\langle G^1 \rangle = \langle \widetilde{G}^1 \rangle$. We assume $\mathrm{span}(F_t) = \mathrm{span}(\widetilde{F}_t)$ and $\langle G^t \rangle = \langle \widetilde{G}^t \rangle$ for all $t \leq \tau$. Then, we can show $\mathrm{span}(C^{\mathrm{pre}}_{\tau+1}) = \mathrm{span}(\widetilde{C}^{\mathrm{pre}}_{\tau+1})$ and $\mathrm{span}(C_{\tau+1}) = \mathrm{span}(\widetilde{C}_{\tau+1})$. In fact, it is $pq \in \mathrm{span}(\widetilde{C}^{\mathrm{pre}}_{\tau+1})$ for any $pq \in C^{\mathrm{pre}}_{\tau+1}$, where $p \in F_1$ and $q \in F_\tau$, because $p \in \mathrm{span}(\widetilde{F}_1)$ and $q \in \mathrm{span}(\widetilde{F}_\tau)$, and vice versa. The orthogonalization Eq. (1) projects $\mathrm{span}(C^{\mathrm{pre}}_{\tau+1})$ to subspace $\mathrm{span}(C_{\tau+1})$, which are orthogonal to $\mathrm{span}(F^\tau)$ in terms of the evaluation at points, i.e., $\mathrm{span}(C_{\tau+1}(X)) \perp \mathrm{span}(F^\tau(X))$. From $\mathrm{span}(C^{\mathrm{pre}}_{\tau+1}) = \mathrm{span}(\widetilde{C}^{\mathrm{pre}}_{\tau+1})$ and $\mathrm{span}(F^\tau(X)) = \mathrm{span}(\widetilde{F}^\tau(X))$, the orthogonalization projects $C^{\mathrm{pre}}_{\tau+1}$ and $\widetilde{C}^{\mathrm{pre}}_{\tau+1}$ into the same subspace, i.e., $\mathrm{span}(C_{\tau+1}) = \mathrm{span}(\widetilde{C}_{\tau+1})$.

Next, we show $\mathrm{span}(F_{\tau+1}) = \mathrm{span}(\widetilde{F}_{\tau+1})$ by showing $\mathrm{span}(F_{\tau+1}) \subset \mathrm{span}(\widetilde{F}_{\tau+1})$. Let $f \in \mathrm{span}(F_{\tau+1})$ be an nonvanishing polynomial. As shown above, $f \in \mathrm{span}(C_{\tau+1}) = \mathrm{span}(\widetilde{C}_{\tau+1})$. By construction, $f \in \mathrm{span}(\widetilde{F}_{\tau+1})$ unless $\mathfrak{n}(f) = \mathbf{0}$. On the other hand if $\mathfrak{n}(f) = \mathbf{0}$, then f need not be included in basis sets because of the third requirement of $\mathfrak{n}$. Therefore, $\mathrm{span}(F_{\tau+1}) = \mathrm{span}(\widetilde{F}_{\tau+1})$.

We show $\langle G^{\tau+1} \rangle = \langle \widetilde{G}^{\tau+1} \rangle$ in a similar way. Let $g \in \mathrm{span}(G_{\tau+1})$ be a vanishing polynomial. Then, $g \in \mathrm{span}(\widetilde{G}_{\tau+1})$ unless $\mathfrak{n}(g) = \mathbf{0}$. From the third requirement for $\mathfrak{n}$, if $\mathfrak{n}(g) = 0$ implies g need not be included in $G_{\tau+1}$. Therefore, $\langle \widetilde{G}^t \rangle \supset \langle G^t \rangle$ and thus $\langle \widetilde{G}^t \rangle = \langle G^t \rangle$. □

The optimality of the normalization scheme in the SBC algorithm

The SBC algorithm generates normalized polynomials by solving the generalized eigenvalue problems. On the other hand, one can consider a simple two-step approach. Namely, we first generate a polynomial g and then normalize it as $g/\|\mathfrak{n}(g)\|$. However, this two-step approach does not provide optimal vanishing polynomials; there can be normalized polynomials that are more vanishing for data points. Solving Eq. (3.2) is equivalent to minimizing the extent of vanishing under the normalization constraint in one step. Recall that in Step 3 of the SBC algorithm, a new polynomial g is generated by linearly combining candidate

polynomials in $C_t = \{c_1, c_2, ..., c_{|C_t|}\}$. This can be formulated as

$$g = \sum_{i=1}^{|C_t|} v_i c_i = C_t \boldsymbol{v},$$

where $\boldsymbol{v} = (v_1, v_2, ..., v_{|C_t|})^\top$ is a combination vector to be sought. From the first condition of Definition 12,

$$\mathfrak{n}(g) = \sum_{i=1}^{|C_t|} v_i \mathfrak{n}(c_i) = \mathfrak{n}(C_t)\boldsymbol{v},$$

where we use a slight abuse of notation such that $\mathfrak{n}(C_t)$ is a matrix whose i-th column is $\mathfrak{n}(c_i)$. Namely, $\mathfrak{n}(g)$ is calculated from $\mathfrak{n}(C_t)$ in the same way g is calculated from C_t. Let us consider to generate $g_k = C_t \boldsymbol{v}_k$ from $\boldsymbol{v}_k$ for $k = 1, 2, ..., r$. We minimize the square sum of the extent of vanishing under the constraint that g_i is normalized and orthogonal to each other with respect to $\mathfrak{n}$.

$$\min_{\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_r \in \mathbb{R}^{|C_t|}} \sum_k \|g_k(X)\|^2, \quad \text{s.t. } \forall k, l,\ \mathfrak{n}(g_k)^\top \mathfrak{n}(g_l) = \delta_{kl}, \tag{3.3}$$

where δ_{kl} is the Kronecker delta. Note that

$$\sum_k \|g_k(X)\|^2 = \mathrm{Tr}(V^\top C_t(X)^\top C_t(X) V),$$

where $V = (\boldsymbol{v}_1\ \boldsymbol{v}_2\ \cdots\ \boldsymbol{v}_r)$ and $\mathrm{Tr}(\cdot)$ is the trace operator. Using the first and the second conditions of Definition 12,

$$\begin{aligned}
\mathfrak{n}(g_k)^\top \mathfrak{n}(g_l) &= \mathfrak{n}(C_t \boldsymbol{v}_k)^\top \mathfrak{n}(C_t \boldsymbol{v}_l) \\
&= \mathfrak{n}\Big(\sum_i c_i v_k^{(i)}\Big)^\top \mathfrak{n}\Big(\sum_j c_j v_l^{(j)}\Big), \\
&= \sum_{i,j} \mathfrak{n}(c_i)^\top \mathfrak{n}(c_j) v_k^{(i)} v_l^{(j)}, \\
&= \boldsymbol{v}_k^\top \mathfrak{N}(C_t) \boldsymbol{v}_l.
\end{aligned}$$

Thus, it is the constraint is summarized to $V^\top \mathfrak{N}(C_t) V = I$. Therefore, the problem Eq. (3.3) can be reformulated as

$$\min_{V \in \mathbb{R}^{|C_t| \times r}} \mathrm{Tr}\left(V^\top C_t(X)^\top C_t(X) V\right), \quad \text{s.t. } V^\top \mathfrak{N}(C_t) V = I. \tag{3.4}$$

The following theorem states the optimality of the normalization scheme of the SBC algorithm.

Theorem 2. *Let r be an integer such that $1 \leq r \leq \mathrm{rank}(\mathfrak{N}(C_t))$. The r generalized eigenvectors $\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_r$ of (3.2), which correspond to the r-smallest generalized eigenvalues, generate polynomials $C_t\boldsymbol{v}_1, C_t\boldsymbol{v}_2, ..., C_t\boldsymbol{v}_r$, whose square sum of the extent of vanishing achieves the minimum under the orthonormal constraint on the normalization components of polynomials.*

Proof. It is known that the column vectors of the optimal V of the problem Eq. (3.4) are generalized eigenvectors corresponding to the r-smallest generalized eigenvalues of $(C_t(X)^\top C_t(X), \mathfrak{N}(C_t))$. The following proof is based on [VMS16].

Introducing a Lagrange multiplier $\widetilde{\Lambda} \in \mathbb{R}^{|C_t|\times|C_t|}$, we have

$$\mathcal{L} = \frac{1}{2}\mathrm{Tr}\left[V^\top C_t(X)^\top C_t(X)V\right] + \frac{1}{2}\mathrm{Tr}\left[(I - V^\top \mathfrak{N}(C_t)V)\widetilde{\Lambda}\right].$$

Note that here $\widetilde{\Lambda}$ is symmetric due to the symmetric constraint, but not diagonal in general. By differentiating $\mathcal{L}$ with V and setting it to zero,

$$\left.\frac{\partial L}{\partial V}\right|_{V=\widetilde{V}} = C_t(X)^\top C_t(X)\widetilde{V} - \mathfrak{N}(C_t)\widetilde{V}\widetilde{\Lambda} = 0.$$

Thus, we obtain

$$C_t(X)^\top C_t(X)\widetilde{V} = \mathfrak{N}(C_t)\widetilde{V}\widetilde{\Lambda}.$$

Note that this is not yet a generalized eigenvalue problem because $\widetilde{\Lambda}$ is not diagonal. Because $\widetilde{\Lambda}$ is symmetric, it has an eigenvalue decomposition $\widetilde{\Lambda} = R\Lambda R^\top$ for some orthonormal matrix R and diagonal matrix Λ. Thus, we obtain the generalized eigenvalue problem (3.2) by $V = \widetilde{V}R$. The cost function is

$$\mathrm{Tr}\left[V^\top C_t(X)^\top C_t(X)V\right] = \mathrm{Tr}\left[V\mathfrak{N}(C_t)V\Lambda\right] = \mathrm{Tr}\left[\Lambda\right],$$

which means that the r smallest generalized eigenvalues minimize the cost. □

Stable computation of the generalized eigenvectors

It is known that in practice, we need to solve the following problem instead of (3.2) for numerical stability.

$$C_t(X)^\top C_t(X)V = (\mathfrak{N}(C_t) + \alpha I)V\Lambda, \tag{3.5}$$

where α is a small positive constant. Such α is typically set to a small multiple of the average eigenvalue of $\mathfrak{N}(C_t)$, i.e., $\mathrm{Tr}(\mathfrak{N}(C_t))/|C_t|$ [Fri89]. It can be shown that such an addition by αI only gives a slight change both on the extent of vanishing and the normalization of obtained polynomials.

Theorem 3. *Let $\{\boldsymbol{v}_1^\alpha, \boldsymbol{v}_2^\alpha, \ldots, \boldsymbol{v}_{|C_t|}^\alpha\}$ be the generalized eigenvectors of (3.5) for $\alpha \geq 0$. Both the extent of vanishing and the norm of $C_t\boldsymbol{v}_k^\alpha$ differ only by $O(\alpha)$ from those of $C_t\boldsymbol{v}_i^0$. Specifically,*

$$\begin{aligned}
\lambda_k^0 - \lambda_k^\alpha &= \frac{\lambda_k^0}{1+\alpha\|\boldsymbol{v}_k^0\|^2}, \\
-\alpha\|\boldsymbol{v}_k^0\|^2\frac{\lambda_k^0}{\lambda_{\min}^0} + O(\alpha^2) &\leq \sqrt{(\boldsymbol{v}_k^0)^\top\mathfrak{N}(C_t)\boldsymbol{v}_k^0} - \sqrt{(\boldsymbol{v}_k^\alpha)^\top\mathfrak{N}(C_t)\boldsymbol{v}_k^\alpha}, \\
&\leq -\alpha\|\boldsymbol{v}_k^0\|^2\frac{\lambda_k^0}{\lambda_{\max}^0} + O(\alpha^2),
\end{aligned}$$

where $\lambda_{\min}^0$ and $\lambda_{\max}^0$ are the smallest and the largest eigenvalues of λ_k^0 for $k = 1, ..., |C_t|$, respectively.

The proof of Theorem 3 relies on the following lemma.

Lemma 1. *Suppose square matrices $A, B \in \mathbb{R}^{n\times n}$ are symmetric and positive-semidefinite, and* nullspace$(A) \supset$ nullspace(B). *Let us consider a perturbed generalized eigenvalue problem $A\boldsymbol{v}_k^\alpha = \lambda_k^\alpha(B+\alpha I)\boldsymbol{v}_k^\alpha$ for a small nonnegative constant α, $(k = 1, ..., \mathrm{rank}(B))$. Then*

$$\begin{aligned}
\lambda_k^0 - \lambda_k^\alpha &= \frac{\lambda_k^0}{1+\alpha\|\boldsymbol{v}_k^0\|^2}, \\
-\alpha\|\boldsymbol{v}_k^0\|^2\frac{\lambda_k^0}{\lambda_{\min}^0} + O(\alpha^2) \leq \|W\boldsymbol{v}_k^0\| - \|W\boldsymbol{v}_k^\alpha\| &\leq -\alpha\lambda_k^0\|\boldsymbol{v}_k^0\|^2\frac{\lambda_k^0}{\lambda_{\max}^0} + O(\alpha^2),
\end{aligned}$$

where $\lambda_{\min}^0$ and $\lambda_{\max}^0$ are the smallest and the largest generalized eigenvalue among $\lambda_1^0, ..., \lambda_{\mathrm{rank}(B)}^0$, respectively.

Proof. A symmetric and positive-semidefinite matrix B has orthonormal eigenvectors $\boldsymbol{u}_1, ..., \boldsymbol{u}_n$, where the first rank$(B)$ eigenvectors span the column space of B and the rest span the nullspace. Note that the generalized eigenvectors $\boldsymbol{v}_k^0, (k = 1, ..., \mathrm{rank}(B))$ are mutually linearly independent due to $(\boldsymbol{v}_k^0)^\top B\boldsymbol{v}_l^0 = \delta_{kl}$.

Hence, $\{\boldsymbol{v}_1^0, ..., \boldsymbol{v}_{\mathrm{rank}(B)}^0, \boldsymbol{u}_{\mathrm{rank}(B)+1}, ..., \boldsymbol{u}_n\}$ becomes a complete basis of $\mathbb{R}^n$. Therefore, for any k,

$$\boldsymbol{v}_k^\alpha = \sum_{i=1}^{\mathrm{rank}(B)} a_i^\alpha \boldsymbol{v}_i^0 + \sum_{i=\mathrm{rank}(B)+1}^{n} a_i^\alpha \boldsymbol{u}_i,$$

for some $a_1^\alpha, ..., a_n^\alpha$. Substituting the above expression into $A\boldsymbol{v}_k^\alpha = \lambda_k^\alpha (B+\alpha I)\boldsymbol{v}_k^\alpha$, we have

$$A \sum_{i=1}^{\mathrm{rank}(B)} a_i^\alpha \boldsymbol{v}_i^0 = \lambda_k^\alpha B \sum_{i=1}^{\mathrm{rank}(B)} a_i^\alpha \boldsymbol{v}_i^0 + \alpha\lambda_k^\alpha \left(\sum_{i=1}^{\mathrm{rank}(B)} a_i^\alpha \boldsymbol{v}_i^0 + \sum_{i=\mathrm{rank}(B)+1}^{n} a_i^\alpha \boldsymbol{u}_i \right),$$

where we used $A\boldsymbol{u}_i = 0$ for $i > \mathrm{rank}(B)$ because $\mathrm{nullspace}(A) \supset \mathrm{nullspace}(B)$. Multiplying both sides by $(\boldsymbol{v}_k^0)^\top$ from the left,

$$a_k^\alpha \lambda_k^0 = \lambda_k^\alpha a_k^\alpha + \alpha\lambda_k^\alpha a_k^\alpha \|\boldsymbol{v}_k^0\|^2,$$

where $(\boldsymbol{v}_k^0)^\top A \boldsymbol{v}_i^0 = \lambda_k^0 (\boldsymbol{v}_k^0)^\top B \boldsymbol{v}_i^0 = \lambda_k \delta_{ik}$ is used. When α is sufficiently small, a_k^α is nonzero (almost 1). Thus, dividing by a_k^α, we obtain

$$\lambda_k^\alpha = \frac{\lambda_k^0}{1 + \alpha\|\boldsymbol{v}_k^0\|^2} = \lambda_k^0 + O(\alpha) > 0. \tag{3.6}$$

Next, by simple calculations,

$$\begin{aligned}
\lambda_k^\alpha &= (\boldsymbol{v}_k^\alpha)^\top A \boldsymbol{v}_k^\alpha, \\
&= \left(\sum_{i=1}^{\mathrm{rank}(B)} a_i^\alpha \boldsymbol{v}_i^0 \right)^\top A \sum_{i=1}^{\mathrm{rank}(B)} a_i^\alpha \boldsymbol{v}_i^0, \\
&= \sum_{i=1}^{\mathrm{rank}(B)} (a_i^\alpha)^2 \lambda_i^0, \\
&= \lambda_k^0 + \sum_{i=1}^{\mathrm{rank}(B)} \left((a_i^\alpha)^2 - \delta_{ik} \right) \lambda_i^0.
\end{aligned}$$

Note that $\lambda_k^\alpha = \lambda_k^0 - \alpha\lambda_k^0\|\boldsymbol{v}_k^0\|^2 + O(\alpha^2)$ from (3.1.3). Therefore,

$$\sum_{i=1}^{\mathrm{rank}(B)} \left((a_i^\alpha)^2 - \delta_{ik} \right) \lambda_i^0 = -\alpha\lambda_k^0\|\boldsymbol{v}_k^0\|^2 + O(\alpha^2).$$

For sufficiently small α, the right-hand side is negative because $\alpha\lambda_k^0\|\boldsymbol{v}_k^0\|^2 > 0$ and $O(\alpha^2) \approx 0$, and thus, the left-hand side is also negative. Hence,

$$-\alpha\|\boldsymbol{v}_k^0\|^2\frac{\lambda_k^0}{\lambda_{\min}^0} + O(\alpha^2) \leq \sum_{i=1}^{\mathrm{rank}(B)} ((a_i^\alpha)^2 - \delta_{ik}) \leq -\alpha\lambda_k^0\|\boldsymbol{v}_k^0\|^2\frac{\lambda_k^0}{\lambda_{\max}^0} + O(\alpha^2).$$

□

Proof of Theorem 3. To simplify the notations, let $A = C_t(X)^\top C_t(X)$ and $B = \mathfrak{N}(C_t)$. Let us consider

$$A\boldsymbol{v}_k^\alpha = \lambda_k^\alpha(B + \alpha I)\boldsymbol{v}_k^\alpha, \tag{3.7}$$

where αI is a small perturbation on B and λ_k^α is the perturbed k-th generalized eigenvalue. We cannot directly apply the standard matrix perturbation theory, which assumes positive-definite B and describes $\boldsymbol{v}_k^\alpha$ by a linear combination of unperturbed generalized eigenvectors. In our case, B is positive-semidefinite, and thus there are only $\mathrm{rank}(B)$ generalized eigenvectors. Hence, the generalized eigenvectors do not form a complete basis of $\mathbb{R}^{|C_t|}$, and $\boldsymbol{v}_k^\alpha$ cannot always be described by these generalized eigenvectors. Fortunately, the theorem above holds using the fact $\mathrm{nullspace}(A) \supset \mathrm{nullspace}(B)$, where $\mathrm{nullspace}(\cdot)$ denotes the nullspace of a given matrix. This relation holds because any vector $\boldsymbol{v} \in \mathrm{nullspace}(B)$ implies the zero polynomial according to the third requirement for $\mathfrak{n}$. From Lemma 1, we conclude that the claim holds. □

Other propositions

Proposition 1. *At $t = 1$, the orthogonalization Eq. (2.2) is equivalent to mean centralization.*

Proof. We use the notations of the SBC algorithm in Section 3.1.1. Let us consider the evaluation matrices at the orthogonalization Eq. (2.2) as follows.

$$\begin{aligned} C_t(X) &= C_t^{\mathrm{pre}}(X) - F^{t-1}F^{t-1}(X)^\dagger C_t^{\mathrm{pre}}(X), \\ &= (I - F^{t-1}(X)F^{t-1}(X)^\dagger)C_t(X). \end{aligned}$$

Note that $P = I - \frac{1}{\sqrt{N}}\mathbf{1}\mathbf{1}^\top \in \mathbb{R}^{N\times N}$, where $\mathbf{1} \in \mathbb{R}^N$ is the all-one vector, is a matrix such that PM is column-wise mean-centralized for any matrix $M \in \mathbb{R}^{N\times D}$; that is, the mean of each column vector of PM is zero. We now show

$F^{t-1}(X)F^{t-1}(X)^\dagger = \frac{1}{N}\mathbf{1}\mathbf{1}^\top \in \mathbb{R}^{N\times N}$.

$$\begin{aligned} F^{t-1}(X)F^{t-1}(X)^\dagger &= (m\mathbf{1})(m\mathbf{1})^\dagger, \\ &= (m\mathbf{1})\left(\frac{1}{m^2N}m\mathbf{1}^\top\right), \\ &= \frac{1}{N}\mathbf{1}\mathbf{1}^\top. \end{aligned}$$

For the second equality, we used a relation $\boldsymbol{a}^\dagger = \boldsymbol{a}^\top/\|\boldsymbol{a}\|$. □

3.2 Spurious Vanishing Problem

In the basis construction, we evaluate a polynomial (say, g) for given set of points X. When g is approximately vanishing for X, two possibilities exist: (i) the roots of g are closes to the points in X or (ii) g is close to the zero polynomials, which takes the zero values everywhere. Needless to say, we would like to discover approximate vanishing polynomials of case (i) because such polynomials reflect the structure of data points.

The spurious vanishing problem is that polynomials of case (ii)—spurious vanishing polynomials—are classified as approximate vanishing polynomials and appended to the basis set. Similarly, the spurious nonvanishing problem is that polynomials of case (i) are classified as nonvanishing polynomials when they have a large coefficient norm. For simplicity, we use the term "spurious vanishing problem" for both of the problems throughout the thesis. Let us see the following example.

Example 1. *Let us consider polynomials* $g = 0.01x^2$ *and* $g' = x^2 - 1$. *Then, for a set of points* $X = \{(1.1), (-1.1)\} \subset \mathbb{R}^1$, g *is* 0.15*-vanishing but* g' *is not.*

In this example, g is more (approximately) vanishing for X than g' even though the roots of g is further than g' in the Euclidean distance. If we use $\epsilon = 0.15$, then g is a spurious vanishing polynomial and g' is a spurious nonvanishing polynomial. Obviously, this is because of the small coefficient of g. The coefficient norm of g and g' is 0.01 and $\sqrt{2}$, respectively. If we normalize g and g' by their coefficient norms to $\widetilde{g} = g/0.01 = x^2$ and $\widetilde{g'} = g'/\sqrt{2} = (x^2 - 1)/\sqrt{2}$, then $\widetilde{g'}$ becomes more vanishing than $\widetilde{g}$.

To summarize, we need to normalize polynomials by some scale before comparing the approximate vanishing. Most basis construction algorithms using a monomial order (e.g., the ABM algorithm) use the coefficient norm as a scale; polynomials are generated to have a unit coefficient norm. Such monomial-order-dependent algorithms can efficiently take coefficient normalization into account. This is because polynomials are generated by linear combinations of distinct monomials by unit vectors and the number of monomials to consider is narrowed down from exponential to a polynomial order. On the other hand, most monomial-order-free algorithms generate polynomials (e.g., VCA and AVICA) by linear combinations of *polynomials*. As a consequence, we need a huge computational cost to calculate the coefficient norm of polynomials. Therefore, we need to answer the following questions to overcome the spurious vanishing problem.

- Which scale for normalization should we use for the basis construction? In particular, which scale is efficient to calculate?
- How can be introduce the normalization into the monomial-order-free basis construction? Is there any simple method that can be introduce into various algorithms?

We will present a simple monomial-order-free algorithm, which generate polynomials under normalization in a simple and optimal fashion. This algorithm is general enough that various monomial-order-free algorithms can be discussed on this framework. As specific scales for normalization, we discuss coefficient normalization and gradient normalization. In particular, the latter can be computed much efficiently.

3.2.1 Coefficient normalization

Coefficient normalization is an intuitive normalization, which normalizes a polynomial h by $\mathfrak{n}_{\mathrm{c}}(h)$ to $h/\|\mathfrak{n}_{\mathrm{c}}(h)\|$.

In fact, it has been commonly used by computer-algebraic basis construction algorithms such as the ABM algorithm. However, without using a monomial order, coefficient normalization becomes computationally costly. There are two sources that cause the cost.

Polynomial expansion—We need to expand polynomials to obtain their coefficient vectors because in the SBC algorithm, polynomials are in the nested sum-product form of polynomials due to the repetition of `Step1` and `Step3` along the degree. In general, such an expansion is computationally expensive because one has to manipulate exponentially many monomials from the expansion in the worst case.

Exponentially long coefficient vectors—Even after the polynomial expansion, the obtained coefficient vectors of polynomials are exponentially long in general. Specifically, a degree-t n-variate polynomial has a coefficient vector of length $\binom{n+t}{n}$.

We present two methods for each of the two challenges above. For the former challenge, the iterative nature of the basis construction is exploited. The precomputation can also help us calculate coefficient vectors. For the latter challenge, we present a coefficient truncation method that enables the coefficient normalization to work much faster while giving up the exact calculation of coefficients.

Circumventing polynomial expansion

The main idea for circumventing polynomial expansion is to hold coefficient vectors of polynomials separately and update these vectors by applying to them the equivalent transformations that are applied to the corresponding polynomials. For example, let us consider a weighted sum $af + bg$ of two polynomials f and g by weights $a, b \in \mathbb{R}$. Then, the coefficient vector of $af + bg$ is also a weighted sum $\mathfrak{n}_{\mathrm{c}}(af + bg) = a\mathfrak{n}_{\mathrm{c}}(f) + b\mathfrak{n}_{\mathrm{c}}(g)$. In contrast to the weighted sum case, it is not easy to calculate the coefficient vector of the product of polynomials, e.g., $\mathfrak{n}_{\mathrm{c}}(fg)$. We encounter such a case at `Step1`, where the precandidate polynomials are generated from the multiplication across linear polynomials and nonlinear polynomials. We will now deal with this problem.

Let us consider n-variate polynomials. Let $M_n^t = \binom{n+t-1}{t}$ and $M_n^{\leq t} = \binom{n+t}{t}$ be the number of n-variate monomials of degree t and of degree up to t, respectively. For simple description, we assume that monomials and coefficients are indexed in the degree-lexicographic order. For instance, in the two-variate case, the degree-lexicographic order is $1, x, y, x^2, xy, y^2, x^3, ...$, and so forth ($x, y$ are variables). We will refer to "the i-th monomial" according to this ordering. Now, we consider a matrix that extends a coefficient vector of a degree-t polynomial to that of a degree-$(t+1)$ polynomial after multiplication by a linear polynomial.

Remark 7. *Given a linear polynomial p, there is a matrix $R_p^{\leq t} \in \mathbb{R}^{M_n^{\leq t+1} \times M_n^{\leq t}}$ such that*

$$\mathfrak{n}_{\mathrm{c}}(pq) = R_p^{\leq t} \mathfrak{n}_{\mathrm{c}}(q),$$

for any polynomial q of degree t.

The existence of such matrix $R_p^{\leq t}$ will soon become evident (see [VYS05] for the case of homogeneous polynomials). Suppose a linear polynomial p is described by $p = \sum_{i=0}^{n} b_k x_k$, where $b_k \in \mathbb{R}$ are coefficients and $x_1, ..., x_n$ are variables. For convenience, we use a notation $x_0 = 1$. Then, $R_p^{\leq t}$ can be described as

$$R_p^{\leq t} = \sum_{k=0}^{n} b_k R_{x_k}^{\leq t},$$

because as observed above, the coefficient vector of the weighted sum of polynomials is the weighted sum of their coefficient vectors. Now, the existence of $R_{x_k}^{\leq t}$ is evident. In fact, the (i,j)-th entry of $R_{x_k}^{\leq t}$ takes value one if the i-th monomial becomes the j-th monomial by the multiplication with x_k, and otherwise the (i,j)-th entry of $R_{x_k}^{\leq t}$ is zero value. Note that $R_{x_k}^{\leq t}$ is not dependent on input data (except the number of variables), and thus, we can compute these matrices in advance. Different monomials are mapped to different monomials after multiplied by x_k. Thus, each column of $R_{x_k}^{\leq t}$ has exactly one nonzero entry (and it is 1), implying $R_{x_k}^{\leq t}$ is a sparse matrix with only $M_n^{\leq t}$ entries, which can be efficiently handled. Moreover, we can represent $R_{x_k}^{\leq t}$ in a block diagonal matrix for $1 \leq k \leq n$,

$$R_{x_k}^{\leq t} = \begin{pmatrix} R_{x_k}^{\leq t-1} & O \\ O & R_{x_i}^{t} \end{pmatrix},$$

where O is the zero matrix, and $R_{x_k}^t \in \mathbb{R}^{M_n^{t+1} \times M_n^t}$ is a submatrix of $R_{x_k}^{\leq t}$ that corresponds to the mapping from degree-t monomials to degree-$(t+1)$ monomials.

For $k = 0$, $R_{x_0}^{\leq t} \in \mathbb{R}^{M_n^{\leq t+1} \times M_n^{\leq t}}$ is a rectangular diagonal matrix with value one along its diagonal.

In summary, in the basis construction, we first hold the coefficient vectors of F_1 besides polynomials (or their evaluation vectors). Then, for each $p \in F_1$, we linearly combine precomputed $R_{x_k}^{\leq 1}$ to obtain $R_p^{\leq 1}$. Using these matrices, we can obtain the coefficient vectors of C_2^{pre}. We then extend $R_p^{\leq 1}$ to $R_p^{\leq 2}$ by appending R_p^2, which is a linear combination of the precomputed $R_{x_k}^2$ to obtain C_3^{pre}. In this way, we can directly manipulate coefficient vectors without performing costly polynomial expansions.

In addition to its less computational cost, there is another practical advantage in the approach that skips the polynomial expansion: it can work with the fast numerical implementation of basis construction. In the numerical implementation, a polynomial is expressed by its evaluation vector instead of a symbolic entity (for example, see the code of [Liv+13] provided in the first author's web page). Because we only know an evaluation vector of a polynomial in the numerical implementation, the "polynomial" cannot be expanded because its symbolic form is unknown. Numerical implementations work much faster because in practice, symbolic operations are much slower than the same number of numerical operations (matrix–vector operations). Also, it is slow to evaluate symbolic entities, although many evaluations are necessary to obtain evaluation vectors of polynomials.

Coefficient truncation for acceleration

We here describe the coefficient truncation method to deal with significantly long coefficient vectors. Coefficient vectors are truncated based on the importance of the corresponding monomials. In particular, at each degree t, we only keep degree-t monomials that have large coefficients in the degree-t nonvanishing polynomials $F_t = \{f_1, f_2, ..., f_s\}$. Although this strategy is simple, the coefficient truncation method has an interesting contrast to a monomial-based algorithm as will be further discussed.

The specific procedures of the coefficient truncation are as follows. Let $\mathfrak{n}_{\mathrm{c}}^t : \mathcal{P}_n \to M_n^t$ be a mapping that gives the coefficient vector corresponding to degree-t monomials of the given polynomial; thus, $\mathfrak{n}_{\mathrm{c}}^t(f_i)$ is a subvector of $\mathfrak{n}_{\mathrm{c}}(f_i)$. With the same abuse of notation of $\mathfrak{n}_{\mathrm{c}}(F_t)$, we define $\mathfrak{n}_{\mathrm{c}}^t(F_t)$ as a matrix whose i-th column is $\mathfrak{n}_{\mathrm{c}}^t(f_i)$. Note that the j-th row of $\mathfrak{n}_{\mathrm{c}}^t(F_t)$ corresponds to the coefficients of the j-th degree-t monomial across polynomials of F_t. Let Δ_j be the norm of the j-th row of $\mathfrak{n}_{\mathrm{c}}^t(F)$. Then, setting a threshold parameter θ, we select monomials

individually from larger Δ_j as long as the following holds,

$$\sum_{j \in \mathcal{B}_t} \Delta_j^2 \leq \theta^2, \tag{3.8}$$

where $\mathcal{B}_t$ is the index set of selected degree-t monomials. We also truncate $R_{x_k}^t$ of the previous section to size $M_n^{t+1} \times |\mathcal{B}_t|$, which becomes a sparse matrix with $|\mathcal{B}_t|$ nonzero entries. Because the coefficient norm is always underestimated from the truncated coefficient vectors, we need to scale the norm of the truncated coefficient vectors according to the truncation rate. To this end, we calculate a rate γ_t, which is a ratio of the root square sum of the preserved coefficients to the root square sum of the full coefficients; that is,

$$\gamma_t = \sqrt{\sum_{j \in \mathcal{B}_t} \Delta_j^2 \,/ \|\mathfrak{n}_{\mathrm{c}}^t(F)\|^2}.$$

The product $\prod_{\tau=1}^t \gamma_\tau$ approximates the truncation rate up to degree t. The normalization matrix for `Step2′` is set to

$$\widetilde{\mathfrak{n}}_{\mathrm{c}}(C_t)^\top \widetilde{\mathfrak{n}}_{\mathrm{c}}(C_t) \left(\prod_{\tau=1}^t \gamma_\tau \right)^{-1}, \tag{3.9}$$

where $\widetilde{\mathfrak{n}}_{\mathrm{c}}(C_t)$ is a matrix whose column vectors are consist of the truncated coefficient vectors of C_t.

The coefficient truncation is similar to a monomial-based algorithm, approximate Buchberger–Möller algorithm (ABM algorithm; [Lim13]). This algorithm proceeds from lower to higher degree monomials, while updating a set of "important" monomials $\mathcal{O}$ (called an order ideal), which corresponds to the basis set F of nonvanishing polynomials of the SBC algorithm. Given a new monomial b, if the evaluation vector of b cannot be well approximated by a linear combination of monomials in $\mathcal{O}$, the ABM algorithm assorts b into $\mathcal{O}$. More specifically, if $b(X) \approx \sum_{m \in \mathcal{O}} c_m m(X)$ for some coefficients $\{c_m\}_{m \in \mathcal{O}}$, then $b - \sum_{m \in \mathcal{O}} c_m m$ is an approximate vanishing polynomial and b is discarded; otherwise b is appended to $\mathcal{O}$. Importantly, monomials divisible by b (i.e., multiples of b) need not be considered at a higher degree, which reduces the number of monomials to handle. It is shown that $|\mathcal{O}| \leq |X|$; thus, the number of monomials to handle does not explode.

The coefficient truncation is distinct from the strategy of ABM algorithm in that it is fully data driven, whereas ABM algorithm relies on a specific monomial

TABLE 3.1: Summary of Example 2

Setting	$m_1 \prec m_2$, $m_1(X) = km_2(X)$	
	Nonvanishing basis	Most important monomial
ABM	m_1	m_1
SBC-$\mathfrak{n}_c$	$\frac{k}{\sqrt{1+k^2}}m_1 + \frac{1}{\sqrt{1+k^2}}m_2$	$\begin{cases} m_1 & (k < 1) \\ m_2 & (k > 1) \end{cases}$

TABLE 3.2: Summary of Example 3

Setting	$m_1 \prec m_2 \prec m_3$, $m_1(X) = km_2(X)$, $m_i(X)^\top m_3(X) = 0, (i = 1, 2)$	
	Nonvanishing basis	Top-2 important monomials
ABM	m_1, m_3	m_1, m_3
SBC-$\mathfrak{n}_c$	$\frac{k}{\sqrt{1+k^2}}m_1 + \frac{1}{\sqrt{1+k^2}}m_2, m_3$	$m_3, \begin{cases} m_1 & (k < 1) \\ m_2 & (k > 1) \end{cases}$

ordering. Now, we provide two examples to highlight the difference in their strategies (also see Tables 3.1 and 3.2).

Example 2. *Let us consider to decide the more "important" monomial from m_1 and m_2 of the same degree, where $m_1(X) = km_2(X) \neq \mathbf{0}$ for a constant k and $m_1 \prec m_2$ for some monomial order. The ABM strategy selects m_1 as the more important monomial because of $m_1 \prec m_2$, whereas the coefficient truncation selects m_1 when $k > 1$ and m_2 when $k < 1$.*

The coefficient truncation gives higher importance on m_1 than m_2 as follows. By solving a generalized eigenvalue problem, the following nonvanishing polynomial is obtained.

$$\frac{k}{\sqrt{1+k^2}}m_1 + \frac{1}{\sqrt{1+k^2}}m_2. \tag{3.10}$$

According to the coefficients of each monomial, m_1 is considered more important when $k > 1$, and m_2 is considered more important when $k < 1$. This result is quite natural because, for example, $k > 1$ implies that m_1 is more nonvanishing than m_2 because $\|m_1(X)\| = k\|m_2(X)\|$. In this way, the truncation strategy that keeps monomials with larger coefficients is fully data-driven. In contrast, due to the predefined monomial order $m_1 \prec m_2$, the ABM strategy consistently selects m_1, regardless of k.

Next, we introduce an additional monomial m_3.

Example 3. *In addition to m_1 and m_2 in Example 2, let us consider a monomial*

m_3, where the degree of m_3 is the same as that of m_1 and m_2, the evaluation vector $m_3(X)$ that is orthogonal to $m_1(X)$ and $m_2(X)$, and $m_1 \prec m_2 \prec m_3$. The ABM strategy selects m_1 and m_3 as the most important and the next most important monomial, respectively, due to the monomial order and the relation of evaluation vectors. On the other hand, the proposed strategy considers m_3 as the most important and m_1 or m_2 as the next most important based on k.

The coefficient truncation first selects m_3 and then m_1 is as follows. By solving a generalized eigenvalue problem, we obtain m_3 and (3.10) as nonvanishing polynomials. Based on the coefficients of the monomials, m_3 is considered the most important in this strategy. From the previous discussion, the second most important monomial is m_1 if $k > 1$, otherwise m_2. We emphasize that the magnitude of the coefficient of m_3 is larger than that of m_1 and m_2 because the coefficient norm of nonvanishing polynomials are normalized and m_3 completely takes the coefficient norm 1, whereas m_1 and m_2 share the coefficient norm 1 by $k/\sqrt{1+k^2}$ and $1/\sqrt{1+k^2}$ due to their mutually linearly dependent evaluation vectors. This result implies that the coefficient truncation gives a priority to monomials that have unique evaluation vectors, such as m_3. The monomials that have similar evaluation vectors to others, such as m_1 and m_2, tend to have moderate coefficients. Hence, the coefficient truncation takes monomials from those with unique evaluation vectors to those with less unique evaluation vectors.

As a consequence of truncating coefficient vectors, coefficient normalization matrix $\mathfrak{N}(C_t) = \mathfrak{n}_{\mathrm{c}}(C_t)^\top \mathfrak{n}_{\mathrm{c}}(C_t)$ in (3.2) is replaced with the one computed from the truncated coefficient vectors (3.9). The coefficient norm of the obtained polynomials is no longer equal but only close to unity. However, we can still calculate the exact evaluation of polynomials by keeping their evaluation vectors and coefficient vectors separate. Thus, the generalized eigenvalues $\{\lambda_i\}_{i=1,2,\ldots,|C_t|}$ at `Step2′` maintain the exact value of the square extent of vanishing.

It is difficult to estimate the error caused by the coefficient truncation because basis construction proceeds iteratively, and error gets accumulated. The following theorem gives a theoretical lower bound of the coefficient truncation without any loss.

Theorem 4. *For an exact calculation of coefficients, we need at least $|F_t|$ monomials for each degree t.*

Proof. Evaluation vectors of nonvanishing polynomials are mutually orthogonal. They form a basis that spans the subspace of $\mathbb{R}^{|X|}$ for a set of data points X, and appending new nonvanishing polynomials gradually completes the basis.

By construction, $\mathrm{span}(F_t(X))$ is a subspace of $\mathrm{rank}(|F_t|)$ that is orthogonal to $\mathrm{span}(F^{t-1}(X))$. A degree-$t$ polynomial is the sum of a linear combination of degree-t monomials and a polynomial of degree less than t. Thus, we need $|F_t|$ or more degree-t monomials to obtain F_t whose evaluation vectors are mutually orthogonal and orthogonal to $\mathrm{span}(F^{t-1}(X))$, which concludes that the claim holds true. □

Theorem 4 states the minimal number of monomials required to perform an exact calculation of coefficient vectors. The equality holds when the evaluation vectors of monomials are always orthogonal until the termination. Since this is too optimistic in practice, we propose to keep $O(|F_t|)$ coefficients at each degree t. Suppose the basis construction terminates at $t = T$. Then, the length of coefficient vectors at $T-1$ is $O(|F^{T-1}|)$. The matrix used to calculate the coefficient vectors of C_T is $O(|F^{T-1}|)$-sparse. The number of new monomials in this step is $O(|F^{T-1}|n)$. It is known $|F^T| \leq |X|$ because the evaluation vectors of F^T (approximately) spans the $\mathbb{R}^{|X|}$. Therefore, the coefficient truncation yields coefficient vectors in polynomial-order length $O(n|X|)$. As a consequence, computing $\mathfrak{n}_{\mathrm{c}}(C_t)^\top \mathfrak{n}_{\mathrm{c}}(C_t)$ in (3.2) costs $O(|C_t|^2 \cdot n|X|) = O(n^3|X|^3)$. This is acceptable when one considers the cost of solving (3.2) is also $O(|C_t|^2) = O(n^3|X|^3)$.

An unhelpful approach for the coefficient norm estimation.

We can consider another approach for approximating coefficient norm of polynomials: how about calculating the norm of the evaluation vector of a polynomial at randomly sampled points to infer the coefficient norm? Unfortunately, this strategy does not work.

Let $\mathcal{V}_n^t(\cdot)$ be the Veronese map, which gives the evaluations of n-variate monomials of degree up to t. For instance, $\mathcal{V}_2^2(\boldsymbol{x}) = (1, x_1, x_2, x_1^2, x_1x_2, x_2^2) \in \mathbb{R}^{1\times 6}$. For a set of points X, we define $\mathcal{V}_n^t(X) \in \mathbb{R}^{|X|\times M_n^{\leq t}}$ as a matrix whose i-th row is the Veronese map of the i-th point. Now, let us consider a polynomial $g = C_t\boldsymbol{v}$ and its evaluation for randomly sampled points Y.

$$\begin{aligned}\|g(Y)\|^2 &= \|C_t(Y)\boldsymbol{v}\|^2, \\ &= \|\mathcal{V}_n^t(Y)\mathfrak{n}_{\mathrm{c}}(C_t)\boldsymbol{v}\|^2, \\ &= \boldsymbol{v}^\top \mathfrak{n}_{\mathrm{c}}(C_t)^\top \mathcal{V}_n^t(Y)^\top \mathcal{V}_n^t(Y)\mathfrak{n}_{\mathrm{c}}(C_t)\boldsymbol{v}.\end{aligned}$$

Note that $\mathfrak{n}_{\mathrm{c}}(C_t)\boldsymbol{v}$ is the coefficient vector of g. Thus, if $\mathcal{V}_n^t(Y)^\top \mathcal{V}_n^t(Y) = I$, then we can estimate the coefficient norm of g from the random evaluation vector $g(Y)$.

TABLE 3.3: Statistics of basis sets of nonvanishing polynomials computed by SBC-I and SBC-$\mathfrak{n}_c$ with coefficient truncation thresholds $\theta = 0.0, 0.5, 0.9, 1.0$. The basis sets of degree up to ten are considered in the calculation of the statistics for fair comparison. One can see that (i) the nonvanishing polynomials obtained by SBC-I have significantly large coefficient norms, while those found by SBC-$\mathfrak{n}_c$ ($\theta = 1.0$) have unit coefficient norms; (ii) even with the coefficient truncation ($\theta = 0.0, 0.5, 0.9$), the coefficient norms are close to unity despite of the drastically shortened coefficient vectors; and (iii) at the same time, runtimes and used memories are reduced by approximately 2–20 times.

		SBC-$\mathfrak{n}_c$				SBC-I
	θ	0.0	0.5	0.9	1.0	
$D2^+$	Length of coeff. vec.	21	21	233	19427	19448
	Mean coeff. norm	2.11	2.11	3.00	1.00	1.19e+4
	Min / Max coeff. norm	0.60 / 4.46	0.60 / 4.46	1.00 / 6.21	1.00 / 1.00	1.00 / 2.33
	Runtime [msec]	1.50e+1	1.52e+1	1.62e+2	4.58e+1	1.57
	Memory [MB]	2.46e+1	2.46e+1	2.72e+1	8.18e+1	2.42
$D3^+$	Length of coeff. vec.	38	38	4207	646625	646646
	Mean coeff. norm	1.79	1.79	2.12	1.00	1.08e+8
	Min / Max coeff. norm	0.54 / 5.30	0.54 / 5.30	0.88 / 3.89	1.00 / 1.00	1.00 / 1.00e
	Runtime [msec]	8.17e+2	8.31e+2	3.61e+3	1.79e+4	4.47e+1
	Memory [MB]	1.10e+3	1.1e+3	4.01e+3	1.17e+4	3.54

However, this cannot be achieved. For instance, when $Y = \{(y_1, y_2)^\top\} \subset \mathbb{R}^2$,

$$\mathcal{V}_2^2(Y)^\top \mathcal{V}_2^2(Y) = \begin{pmatrix} 1 & y_1 \cdots & y_2^2 \end{pmatrix}^\top \begin{pmatrix} 1 & y_1 \cdots & y_2^2 \end{pmatrix},$$
$$= \begin{pmatrix} 1 & & & & & \\ & y_1^2 & & & & \\ & & \ddots & & y_1^2 y_2^2 & \\ & & & y_1^2 y_2^2 & & \\ & & & & \ddots & \end{pmatrix},$$

which has $y_1^2 y_2^2$ both in diagonal and off-diagonal entries. Therefore, $\mathcal{V}_2^2(\boldsymbol{x})^\top \mathcal{V}_2^2(\boldsymbol{x})$ will not be the identity matrix regardless of the sampled points.

3.2.2 Demonstration of the spurious vanishing problem in numerical experiments

Here, we compare VCA, SBC without any normalization (i.e., SBC with `Step2`; SBC-I), and SBC with the coefficient normalization (SBC-$\mathfrak{n}_c$). In the first experiment, we show that VCA and SBC-I encounter severe spurious vanishing problem even in simple datasets, whereas SBC-$\mathfrak{n}_c$ does not. In the second experiment, we compare these methods in classification tasks. All experiments were

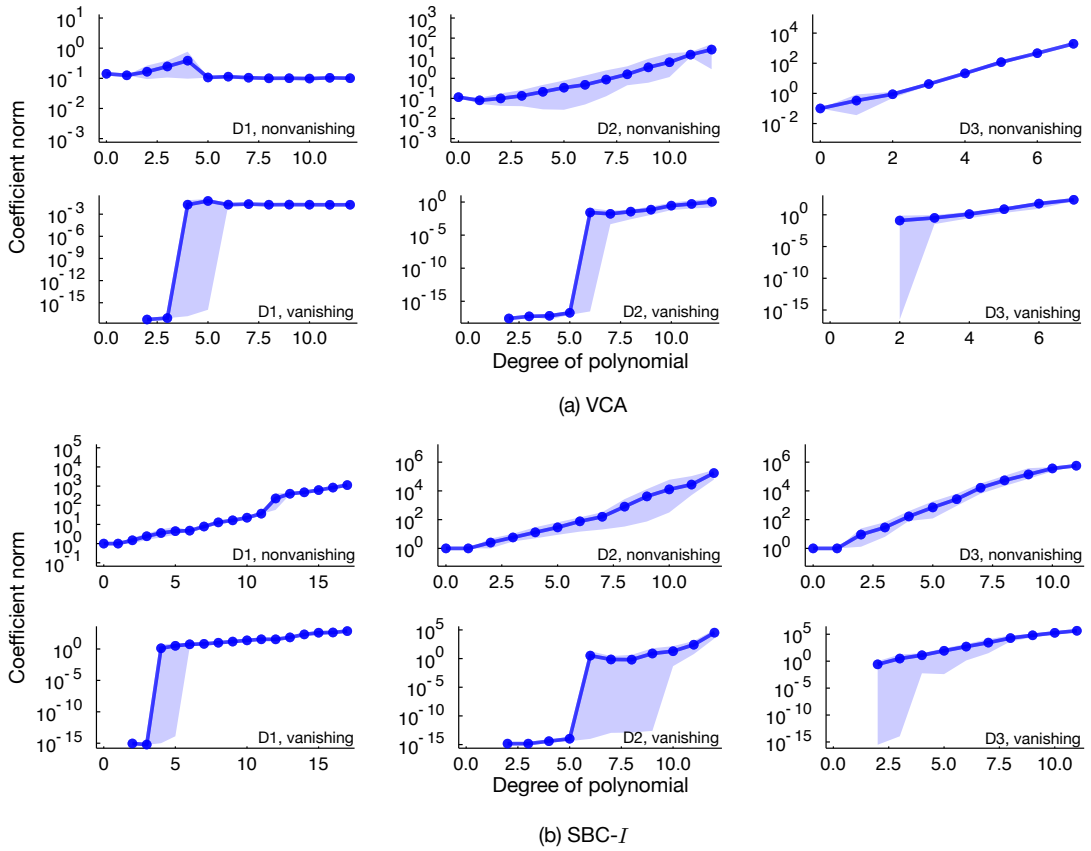

FIGURE 3.3: Coefficient norm of nonvanishing polynomials (upper row of each panel) and vanishing polynomials (lower row of each panel) by (a) VCA and (b) SBC-I for three datasets (each column for each dataset). The mean coefficient norms of each degree are linked by solid lines. The range from the smallest to the largest coefficient norms is represented by shades. The coefficient norm is considerably different even at a degree, and the average coefficient norm increases sharply over degree.

performed using Julia implementation on a desktop machine with an eight-core processor and a 32 GB memory. We emphasize that the proposed methods (coefficient normalization with the generalized eigenvalue problem and the coefficient truncation) can be easily unified with other basis construction methods because these methods are all based on the SBC framework. However, these methods are less commonly used than VCA, and they need more hyperparameters to control, which makes the analysis unnecessarily complicated.

Analysis of coefficient norm and the extent of vanishing with Simple Datasets

We perform basis construction by VCA, SBC-I, and SBC-$\mathfrak{n}_c$. The coefficient norm and the extent of vanishing of obtained polynomials are respectively compared between three methods. We also compare SBC-$\mathfrak{n}_c$ with and without the coefficient truncation.

TABLE 3.4: Classification results by VCA and SBC-$\mathfrak{n}_c$ in three datasets (Iris, Vowel, and Vehicle). SBC-$\mathfrak{n}_c$ achieved comparable or even lower errors than VCA with significantly shorter feature vectors, which implies VCA basis sets contain many spurious vanishing polynomials. The results are averaged over ten independent runs.

		SBC-$\mathfrak{n}_c$			VCA
	θ	0.5	0.9	1.0	
Ir.	Error	0.03	0.05	0.06	0.05
	Length of $\mathcal{F}(\cdot)$	8.20e+1	9.29e+1	6.35e+1	1.51e+2
	Runtime [msec]	1.01e+1	1.10e+1	8.66	6.90
	Memory [MB]	5.61	6.50	4.91	4.74
Vo.	Error	0.45	0.50	0.34	0.45
	Length of $\mathcal{F}(\cdot)$	3.29e+3	3.24e+3	3.12+3	4.74e+3
	Runtime [msec]	5.47e+3	5.85e+3	4.89e+3	7.20e+3
	Memory [MB]	5.24e+2	5.71e+2	6.88e+2	3.97e+2
Ve.	Error	0.26	0.25	0.20	0.19
	Length of $\mathcal{F}(\cdot)$	5.57e+3	5.78e+3	5.25e+3	8.26e+3
	Runtime [msec]	2.95e+4	3.89e+4	4.69e+4	1.31e+4
	Memory [MB]	4.23e+3	4.24e+3	4.38e+3	1.01e+3

Datasets and parameters We use three algebraic varieties: (D1) double concentric circles (radii 1 and 2), (D2) 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 (D3) $\{xz - y^2, x^3 - yz\}$. We randomly sampled 50, 70, and 100 points from these algebraic varieties, respectively. We further consider two datasets by adding variables to D2 and D3. (D2$^+$) five additional variables $y_i = k_i x_1 + (1-k_i)x_2$ for $k_i \in \{0.0, 0.2, 0.5, 0.8, 1.0\}$, where x_1 and x_2 are the variables of D2. (D3$^+$) nine additional variables $y_i = k_i x_1 + l_i x_2 + (1 - k_i - l_i)x_3$ for $(k_i, l_i) \in \{0.2, 0.5, 0.8\}^2$, where x_1, x_2, and x_3 are the variables of D3. Each dataset is mean-centralized and then perturbed by the additive Gaussian noise. The mean of the noise is set to zero, and the standard deviation is set to 5% of the average absolute value of the points.

For each dataset and method, the threshold ϵ is selected as follows. First, we compute a Gröbner basis G of the algebraic variety of the dataset. Suppose G contains $M_t, M_{t+1}, \cdots, M_T$ polynomials at degree $t, t+1, ..., T$, respectively, where t and T are the lowest degree and highest degree of polynomials in G, respectively. Then, ϵ is selected so that the target basis construction yields a basis G' whose lowest-degree polynomial is degree t, and $|G'_\tau| \geq M_\tau$ for $\tau \geq t$. To be more precise, we first search the range of such thresholds, and set ϵ to the mean of that range.

Results In Fig. 3.3, the coefficient norm of nonvanishing polynomials (upper row of each panel) and that of vanishing polynomials (bottom row of each panel),

which are obtained by VCA and SBC-I, are plotted along the degree. The mean values are represented by solid lines and dots, and the range from minimum to maximum is represented by shades. As can be seen from the figure, the mean coefficient norm tends to sharply grow along the degree (note that the vertical axes are in the logarithm scale) for both methods. Even within a degree, there can be a huge gap as in degree-5 VCA vanishing polynomials of D1, degree-6 SBC-I vanishing polynomials of D2 (bottom middle panel), and so on. These results imply that some vanishing (or nonvanishing) polynomials might be vanishing (or nonvanishing) merely due to their small (or large) coefficients; such polynomials might become nonvanishing (or vanishing) polynomials once these polynomials are normalized to have a unit coefficient norm. In fact, this is corroborated by the result shown in Fig. 3.4(a,b). The extent of vanishing (blue dots and solid lines) is contrasted against the rescaled extent of vanishing (red dots and dashed lines), which is calculated by rescaling the extent of vanishing using the coefficient norm of polynomials at post-processing so that polynomials have a unit coefficient norm. After the rescaling, some nonvanishing polynomials show the extent of vanishing below the threshold (gray dotted line) and some vanishing polynomials show the extent of vanishing above the threshold. For example, degree-5 VCA vanishing polynomials become nonvanishing polynomials after the rescaling; degree-10 SBC-I nonvanishing polynomials become vanishing polynomials after the rescaling. The variance of the extent of vanishing at each degree also changes drastically. For example, the rescaling degree-5 VCA vanishing polynomials show large variance of the extent of vanishing, but the rescaling reveals that the actual extent of vanishing is almost identical to these polynomials. The reverse is also observed as in degree-5 VCA nonvanishing polynomials. Note that both VCA and SBC-I required expensive calculations for this post-processing (rescaling), because usually, these methods cannot access the coefficient norm of polynomials (especially in the numerical implementation).

In contrast, as shown in Fig. 3.4(c), the extent of vanishing of polynomials from SBC-$\mathfrak{n}_c$ are consistent before and after the normalization, which is simply because the polynomials are generated under the coefficient normalization. Moreover, we can see that under coefficient normalization, the extent of vanishing shows considerably lower variance for both nonvanishing and vanishing polynomials. In other words, VCA and SBC-I overestimate (or underestimate) the extent of vanishing due to the bloat in the coefficient norm.

Next, we evaluate SBC-$\mathfrak{n}_c$ with the coefficient truncation. The result is summarized in Table 3.3. We change the truncation threshold θ in (3.8) from 0.0

to 1.0. Following Theorem 4, we keep at least $|F_t|$ coefficients at each degree regardless of θ. Thus, $\theta = 0.0$ corresponds to the case where we keep exactly $|F_t|$ coefficients for each degree. $\theta = 1.0$ corresponds to SBC-$\mathfrak{n}_c$ without the coefficient truncation. Here, we analyze the nonvanishing polynomials in terms of the length of coefficient vectors, the actual coefficient norm (mean, minimum, and maximum), the runtime of basis construction, and the memory used during the basis construction. To measure these statistics consistently across methods and parameters, the basis construction is terminated at degree 10 even if the termination condition is not satisfied. We also show the same statistics of SBC-I. As for VCA, we cannot find proper parameter ϵ so that the degree and number of basis polynomials are similar to the Gröbner basis. VCA rescales each nonvanishing polynomial by the norm of its evaluation vector during the basis construction. We consider that this rescaling can lead to more spurious vanishing polynomials, resulting in too early termination, or lead to more spurious nonvanishing polynomials, resulting in fewer vanishing polynomials than those of the Gröbner basis at each degree. Because the computation of VCA and SBC-I is quite similar, it is enough only to consider SBC-I for measuring runtime and memory.

As can be seen in Table 3.3, with $\theta = 0.9$, the truncated coefficient vectors are approximately 100 times shorter. Nevertheless, the mean, minimum, and maximum of the coefficient norm are still moderately close to unity, respectively, for both datasets. This means that only about 1% of monomials and coefficients have a significant contribution to the basis polynomials. Even in the extreme case ($\theta = 0.0$), the coefficient norm of polynomials still lies in the moderate range, while the coefficient vectors are significantly shortened (less than 0.1%). By the coefficient truncation, the runtime and memory for SBC-$\mathfrak{n}_c$ is reduced. For example, at $\theta = 0.9$, the runtime and memory of SBC-$\mathfrak{n}_c$ is reduced by around 3 for both datasets; at $\theta = 0.5$, the runtime is reduced by 20 times for $D3^+$. SBC-I remains faster than SBC-$\mathfrak{n}_c$ even with $\theta = 0.0$. However, the coefficient norm of SBC-I significantly varies across polynomials (e.g., 10^{10} gap between minimum and maximum for $D3^+$). In other words, the fast calculation of SBC-I is a consequence of allowing the basis construction to encounter the spurious vanishing problem.

Again, note that coefficient vectors are typically not accessible for VCA and SBC-I in the numerical implementation. Thus, one cannot normalize nor discard polynomials by weighing their coefficient norms, as done in the analysis. For the above analysis, we calculated the coefficient vectors for VCA and SBC-I in the same manner as in SBC-$\mathfrak{n}_c$, which takes the additional cost. The runtime

was measured by independently running VCA and SBC-I without the coefficient calculation.

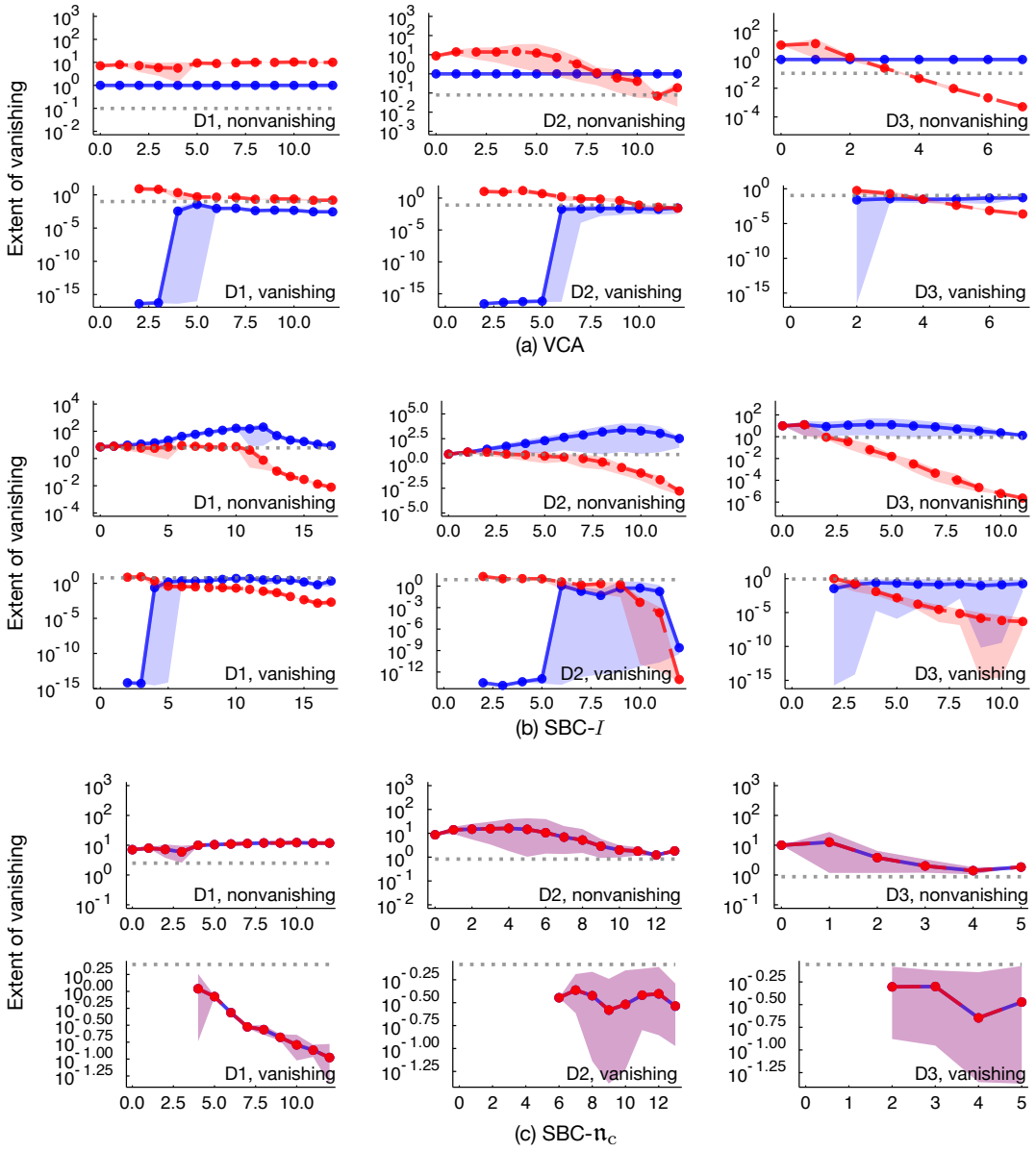

FIGURE 3.4: The extent of vanishing of polynomials obtained by (a) VCA and (b) SBC-I, and (c) SBC-$\mathfrak{n}_c$ for three datasets (each column for each dataset). In each panel, the upper and lower rows show the result for nonvanishing and vanishing polynomials, respectively. The mean extent of vanishing at each degree is linked by dots and solid lines (blue), and the mean extent of vanishing of polynomials whose coefficient norm is normalized to unity by a post-processing is linked by dots and dashed lines (red). The range from the smallest to the largest extent of vanishing is represented by shades (red and blue). Dotted lines (gray) represent threshold ϵ.

3.2.3 Gradient normalization

The spurious vanishing polynomials are closed to the zero polynomial. The coefficient normalization in the previous section measure the closeness by the coefficient norm; namely, polynomials that consists of small magnitude of coefficients are considered close to the zero polynomials. Here, we shift our thinking and consider gradient normalization, where the closeness to the zero polynomial is measured by how "flat" polynomials are. 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

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

If h is close to the zero polynomial, $\nabla h(X) \approx O$, which implies that h is "flat" at all the points of X. We will show that the following mapping is a valid normalization mapping for the SBC algorithm.

$$\mathfrak{n}_{\mathrm{g}}(h; X) = \mathrm{vec}(\nabla h(X)) \in \mathbb{R}^{|X|n}, \tag{3.11}$$

where $\mathrm{vec}(\cdot)$ denotes the vectorization of a given matrix. Under the gradient normalization, the gradient norm $\|\mathfrak{n}_{\mathrm{g}}(h; X)\|$ of h for X is unity. The gradient normalization has the following advantages.

Polynomial time complexity—The normalization vector $\mathfrak{n}_{\mathrm{g}}(h; X)$ is of length $|X|n$, where n is the number of variables. This is a noticeable difference from the coefficient normalization vector, the length of which grows exponentially according to n and the degree of a polynomial.

Exact gradient calculation without differentiation—By exploiting the iterative nature of the basis construction, exact normalization vectors can be computed without performing a differentiation. Thus, we do not have to resort to symbolic differentiation nor the finite difference approximation.

Bounding the approximate vanishing based on the noise level—For each point $\boldsymbol{x} \in X$ of given dataset X, we can consider the "noise-free" point $\boldsymbol{x}^* = \boldsymbol{x} - \boldsymbol{n}$, where $\boldsymbol{n}$ is the perturbation. A basis polynomial approximately vanishes for $\boldsymbol{x}$ but does it also approximately vanish for $\boldsymbol{x}^*$, too? The basis construction with gradient normalization answer the question affirmatively when the noise is small.

Consistent basis sets for input transformation—Simple translation and scaling can drastically change the basis set—this inconsistency issue is overcome

by using the gradient normalization as we will discuss in Section 3.4.

Based on a classical basis construction algorithm for noise-free points [MB82], 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 [Sau07; HKP19], 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 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 [FT13], 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.

Validity of the gradient normalization.

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. (3.2) only provides polynomials with the nonzero gradient norm. Is it sufficient for basis construction to only collect such polynomials? We can answer this question affirmatively. We first prove the following lemma.

Lemma 2. *Any $g \in \mathcal{P}_n$ of degree at least one can be represented as*

$$g = \sum_{k=1}^{n} h_k \frac{\partial g}{\partial x_k} + r,$$

where $h_k, r \in \mathcal{P}_n$ and $\deg_k(r) < \deg_k(g)$ for $k = 1, 2, ..., n$. Here, $\deg_k(\cdot)$ denotes the degree of a given polynomial with respect to the k-th variable x_k.

Proof. We provide a constructive proof. For simplicity of notation, we use t_k such that $t_k + 1 = \deg_k(g)$. If $\deg_1(g) < t_1 + 1$, we set $h_1 = 0$ and proceed to

$k = 2$. Otherwise, we rearrange g according to the degree of x_1 as follows.

$$g = x_1^{t_1+1} g_1^{(0)} + x_1^{t_1} g_1^{(1)} + \cdots + g_1^{(t_1+1)},$$

where $g_1^{(\tau)}$ denotes an $(n-1)$-variate polynomial of degree at most τ that does not contain x_1, $(\tau = 0, 1, ..., t_1 + 1)$. Then,

$$\frac{\partial g}{\partial x_1} = (t_1 + 1) x_1^t g_1^{(0)} + t_1 x_1^{t_1-1} g_1^{(1)} + \cdots + 0.$$

By setting $h_1 = x_1/(t_1 + 1)$,

$$g = h_1 \frac{\partial g}{\partial x_1} + \frac{r_1}{t_1 + 1},$$

where $r_1 = x_1^{t_1} g_1^{(1)} + 2x_1^{t_1-1} g_1^{(2)} \cdots + (t_1 + 1) g_1^{(t_1+1)}$. Note that $\deg_1(r_1) \leq t$ and $\deg_l(r_1) \leq t_l + 1$ for $l \neq 1$. Next, we perform the same procedures for $k = 2$ and r_1; if $\deg_2(r_1) < t_2 + 1$ then set $h_2 = 0$ and $r_2 = r_1$, and proceeds to $k = 3$; otherwise, rearrange r_1 according to the degree of x_2 as

$$r_1 = x_2^{t_2+1} r_2^{(0)} + x_2^{t_2} r_2^{(1)} + \cdots + r_2^{(t_2+1)},$$

where $r_2^{(\tau)}$ denotes an $(n-1)$-variate polynomial of degree at most τ that does not contain x_2, $(\tau = 0, 1, ..., t_2 + 1)$. Again, setting $h_2 = x_2/(t_2 + 1)$, we obtain

$$g = h_1 \frac{\partial g}{\partial x_1} + h_2 \frac{\partial g}{\partial x_2} + r_2,$$

where $r_2 = x_2^{t_2} r_2^{(1)} + 2x_2^{t_2-1} r_2^{(2)} \cdots + (t_2 + 1) r_2^{(t_2+1)}$. Note that $\deg_1(r_2) \leq t_1$, $\deg_2(r_2) \leq t_2$ and $\deg_l(r_2) \leq t_l + 1$ for $l \neq 1, 2$. Repeating this procedure until $k = n$, then $r := r_n$ satisfies $\deg_l(r) \leq t_l$ for all l. □

With Lemma 2, we can prove the following two lemmas, which states that the gradient norm of a polynomial equals the zero value only when the polynomial is not necessary for the basis set.

Lemma 3. *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 $g \in \langle G^t \rangle$.*

Proof. The degree of g and $\partial g/\partial x_k$ are $t+1$ and at most t, respectively. g can be represented as

$$g = \sum_{k=1}^{n} h_k \frac{\partial g}{\partial x_k} + r,$$

where h_k and r are polynomials. From Lemma 2, h_k can be selected so that the degree of r is at most t. By evaluating this for X, we obtain $r(X) = \mathbf{0}$. Since G^t can generate any vanishing polynomial of degree at most t, $r \in \langle G^t \rangle$. Also, $\partial g/\partial x_k \in \langle G^t \rangle$ for $k = 1, 2, ..., n$. From the absorption property of the ideal, $g \in \langle G^t \rangle$. □

Lemma 4. *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 $f(X) \in \mathrm{span}(F^t(X))$.*

Proof. Since f and $\partial f/\partial x_k$ are polynomials of degree $t+1$ and degree at most t, respectively, there are polynomials h and r such that

$$f = \sum_{k=1}^{n} h_k \frac{\partial f}{\partial x_k} + r,$$

where the degree of h_k and r are polynomials. From Lemma 2, h_k can be selected so that the degree of r is at most t. By evaluating this for X, we obtain $r(X) = f(X)$. Since column space of $F^t(X)$ spans evaluation vectors of any polynomial of degree at most t, $r(X) \in \mathrm{span}(F^t(X))$. □

Remark 8. *Even without gradient normalization, any nonvanishing polynomial $f \in F_{t+1}$ has a nonzero gradient norm. Due to the orthogonalization Eq. (2.2), C_{t+1} consists of polynomials such that $\mathrm{span}(C_{t+1}(X))$ is orthogonal to $\mathrm{span}(F^t(X))$. Thus, for any nonvanishing polynomial $f \in \mathrm{span}(C_{t+1})$, it is $f(X) \notin \mathrm{span}(F^t(X))$ and thus, $\|\mathfrak{n}_{\mathrm{g}}(f; X)\| \neq 0$.*

Lemmas 3 and 4 imply that we do not need polynomials with a zero gradient norm 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 the SBC algorithm. Now, we can readily show that $\mathfrak{n}_{\mathrm{g}}$ is a valid normalization mapping for SBC.

Theorem 5. *The mapping* $\mathfrak{n}_{\mathrm{g}}$ *of Eq. (3.11) is a valid normalization mapping for the SBC algorithm.*

Proof. It is trivial that the first two requirements are satisfied. As for the third requirement, Lemmas 3 and 4 state that we do not need to use polynomials with zero gradient norm for the basis sets of both vanishing polynomials and nonvanishing polynomials. □

Exact gradient computation 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

$$\begin{aligned}\frac{\partial h}{\partial x_k}(\boldsymbol{x}) &= \sum_{c \in C_t^{\mathrm{pre}}} u_c \frac{\partial (p_c q_c)}{\partial x_k}(\boldsymbol{x}) + \sum_{f \in F^{t-1}} v_f \frac{\partial f}{\partial x_k}(\boldsymbol{x}), \\ &= \sum_{c \in C_t^{\mathrm{pre}}} u_c q_c(\boldsymbol{x}) \frac{\partial p_c}{\partial x_k}(\boldsymbol{x}) + \sum_{c \in C_t^{\mathrm{pre}}} u_c p_c(\boldsymbol{x}) \frac{\partial q_c}{\partial x_k}(\boldsymbol{x}) + \sum_{f \in F^{t-1}} v_f \frac{\partial f}{\partial x_k}(\boldsymbol{x}).\end{aligned} \tag{3.12}$$

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* $O(n|C_t|) = O(n\mathrm{rank}(X)|X|)$.

Proof. Equation (3.12) shows that the evaluation of a partial derivative $(\partial h/\partial x_k)(\boldsymbol{x})$ is the sum of $|C_t^{\mathrm{pre}}| + |F^{t-1}|$ terms (note $|C_t| = |C_t^{\mathrm{pre}}|$). Hence, $\nabla h(\boldsymbol{x})$ requires $O(n(|C_t| + |F^{t-1}|))$. Note that for an final output F of SBC $|F| \leq |X|$ holds. This is because $F(X)$ is full-rank thanks to the orthogonalization in Eq. (1), and $\mathrm{rank}(F^t(X)) \leq |X|$ because $\mathrm{span}(F(X)) \subseteq \mathrm{R}^{|X|}$ (the equalities hold at $\epsilon = 0$), where $\mathrm{rank}(\cdot)$ denotes the matrix rank of given matrix. Hence, $O(|F^{t-1}|) = O(|X|)$. Also, by its construction, $|F_1| \leq \mathrm{rank}(X)$. Therefore, $|C_t| = |F_1||F_{t-1}| \leq \mathrm{rank}(X)|X|$, and thus, $O(|C_t|) = O(\mathrm{rank}(X)|X|)$. Thus, $O(n(|C_t| + |F^{t-1}|)) = O(n|C_t|) = O(n \cdot \mathrm{rank}(X)|X|)$. □

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. (3.2) 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. 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.

Bound on the approximate vanishing based on the noise level

Let X_0 be the "noise-free dataset" of given dataset X; that is, for each $\boldsymbol{x} \in X$, there is $\boldsymbol{x}_0 \in X_0$ such that $\boldsymbol{x} = \boldsymbol{x}_0 + \boldsymbol{n}$, where $\boldsymbol{n}$ is the perturbation. In practice, one can only have access to a set of perturbed data points X and run a basis construction algorithm to obtain the approximate vanishing polynomials G for X. Implicitly, it is assumed that any approximate vanishing polynomial $g \in G$ approximately vanishes for the points in X_0. However, this is not the case when one uses coefficient normalization. For simplicity, we analyze the behavior of the exact vanishing polynomials for X.

Example 4. *Let us consider a polynomial $g = (x - p)x^5/\sqrt{1 + p^2}$, which is an exact vanishing polynomial for a set of single one-dimensional point $X = \{p\} \subset \mathbb{R}$. Let $p^* = p - \delta$ be a noise-free point of p, where δ is the noise. Note that g is normalized by its coefficient norm. Then, it is $g(p^*) = \delta p^5/\sqrt{1 + p^2} = O(\delta p^3)$. Therefore, the effect of noise can be arbitrarily large by increasing the value of p.*

On the other hand, if we normalize g by its gradient in Example 4 (say, $\widetilde{g}$), the value of p has no effect on the evaluation of $\widetilde{g}$ at p^* is only determined by δ.

Example 5. *Let us consider the same setting as in Example 4. If g is normalized by its gradient, i.e., $\widetilde{g} := g/(dg/dx)(p) = (x-p)$, then its evaluation at p^* is $\widetilde{g}(p^*) = \delta/5 = O(\delta)$.*

This difference between coefficient normalization and gradient normalization arises from the fact that the latter is an data-dependent normalization. Although Examples 4 and 5 cannot be directly generalized to multi-variate and many-point cases, we can still prove a similar statement. The following proposition argues that when noise is small enough, the extent of vanishing at noise-free points are also small or more specifically, bounded by the largest noise.

Proposition 3. *Let $X \subset \mathbb{R}^n$ be a set of points which are generated by perturbing $X^* \subset \mathbb{R}^n$. We denotes the corresponding noisy point in X and noise-free point in X^* by $\boldsymbol{x}$ and $\boldsymbol{x}^*$, respectively. Let g be any exact vanishing polynomial for X. If g is normalized with respect its gradient at X, then it is $\|g(X^*)\| \leq \|\boldsymbol{n}_{\max}\| + O(\|\boldsymbol{n}_{\max}\|^2)$, where $\boldsymbol{n}_{\max} = \max_{\boldsymbol{x}\in X} \|\boldsymbol{x}^* - \boldsymbol{x}\|$.*

Proof. We consider a set of noisy points $X = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_{|X|}\}$ and noise vectors $\boldsymbol{n}_i = \boldsymbol{x}^* - \boldsymbol{x}$ $(i = 1, 2, ..., |X|)$. By the Taylor expansion,

$$\begin{aligned}
\|g(X^*)\|^2 &= \sum_{i=1}^{|X|} g(\boldsymbol{x}_i + \boldsymbol{n}_i)^2, \\
&= \sum_{i=1}^{|X|} \left(g(\boldsymbol{x}_i) + \nabla g(\boldsymbol{x}_i)\boldsymbol{n}_i + O(\|\boldsymbol{n}_i\|^2)\right)^2, \\
&= \sum_{i=1}^{|X|} (\nabla g(\boldsymbol{x}_i)\boldsymbol{n}_i)^2 + O(\|\boldsymbol{n}_i\|^4).
\end{aligned}$$

Since g is normalized with respect to its gradient,

$$\begin{aligned}
\sum_{i=0}^{|X|} (\nabla g(\boldsymbol{x}_i)\boldsymbol{n}_i)^2 &\leq \sum_{i=0}^{|X|} \|\nabla g(\boldsymbol{x}_i)\|^2 \|\boldsymbol{n}_i\|^2, \\
&= (\max_i \|\boldsymbol{n}_i\|^2) \sum_{i=0}^{|X|} \|\nabla g(\boldsymbol{x}_i)\|^2, \\
&= \max_i \|\boldsymbol{n}_i\|^2,
\end{aligned}$$

TABLE 3.5: Comparison of basis sets obtained by SBC with different normalization ($\mathfrak{n}_c$ and $\mathfrak{n}_g$). 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	$\mathfrak{n}_c$-ratio	$\mathfrak{n}_g$-ratio	runtime (ms)
D_1	$\mathfrak{n}_c$	41	1.00	12.2e+2	48.0e+1
	$\mathfrak{n}_g$	30	46.6	1.00	**13.4**
D_2	$\mathfrak{n}_c$	70	1.00	19.6e+2	17.5e+3
	$\mathfrak{n}_g$	33	76.9	1.00	**11.4**

where we used the Cauchy–Schwarz inequality for the second inequality. Therefore, we can conclude

$$\begin{aligned}\|g(X^*)\| &\leq \sqrt{\max_i \|\boldsymbol{n}_i\|^2 + O(\max_i \|\boldsymbol{n}_i\|^4)}, \\ &= \max_i \|\boldsymbol{n}_i\| + O(\max_i \|\boldsymbol{n}_i\|^2).\end{aligned}$$

□

3.2.4 Comparison of basis sets in numerical experiments

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

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) $\{x_1x_3 - x_2^2, x_1^3 - x_2x_3\}$. From each of them, 75 points and 100 points are randomly sampled. Five additional variables $y_i = k_ix_1 + (1 - k_i)x_2$ for $k_i \in \{0.0, 0.2, 0.5, 0.8, 1.0\}$ are added to the former and nine additional variables $y_i = k_ix_1 + l_ix_2 + (1 - k_i - l_i)x_3$ 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 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.

TABLE 3.6: 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
			$\mathfrak{n}_c$	$\mathfrak{n}_g$	$\mathfrak{n}_g$+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

As can be seen from Table 3.5, SBC-$\mathfrak{n}_g$ runs substantially faster than SBC-$\mathfrak{n}_c$ (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, $\mathfrak{n}_c$-ratio and $\mathfrak{n}_g$-ratio are unity for SBC-$\mathfrak{n}_c$ and SBC-$\mathfrak{n}_g$, 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.

Detail of the experiment—We select ϵ for a basis construction algorithm $\mathcal{A}$ and a variety as follows. First, we compute the Gröbner basis of the variety with the degree-reverse-lexicographic order, which determines the number N of the lowest degree $d_{\min}$ of basis polynomials. We added M dummy variables to our datasets (five for D_1 and nine for D_2) and thus M approximate vanishing polynomials of degree 1 should be obtained, too. Using a linear search, we estimate the range (ϵ_1, ϵ_2) of ϵ with which $\mathcal{A}$ outputs a basis set of vanishing polynomials such that (i) M linear polynomials are contained, (ii) at least N polynomials of degree $d_{\min}$ are contained, and (iii) no nonlinear polynomials of degree less than $d_{\min}$ are contained. Finally, ϵ is set to $\epsilon = (\epsilon_1 + \epsilon_2)/2$. The condition (ii) does not require "exactly" N but "at least" N. This is because VCA and SBC do not calculate the Gröbner basis and thus there can be more than N polynomials of degree d_m for several reasons (e.g., the spurious vanishing problem and redundancy of the basis set).

3.2.5 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 [Liv+13], the feature vector $\mathcal{F}(\boldsymbol{x})$ of a data point $\boldsymbol{x}$ was defined as

$$\mathcal{F}(\boldsymbol{x}) = \Big(\cdots, \underbrace{\left|g_1^{(i)}(\boldsymbol{x})\right|, \cdots, \left|g_{|G_i|}^{(i)}(\boldsymbol{x})\right|}_{G_i}, \cdots\Big)^\top, \tag{3.13}$$

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+08]. We used three small standard datasets (Iris, Vowel, and Vehicle) from the UCI dataset repository [Lic13]. 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 3.6. 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., [KKT14; HNT16]. One can consider normalization and basis reduction for these algorithms, but this is beyond the scope of this paper.

3.3 Redundancy in the Basis Set

A basis set G of a vanishing polynomial $\mathcal{I}(X)$ can generate any polynomial of $\mathcal{I}(X)$; that is, $\forall g \in \mathcal{I}(X), g \in \langle G \rangle$. Therefore, if $g_1 \in \langle G - \{g_1\} \rangle$, then g_1 is a redundant polynomial in G. More precisely, 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 $\{h_{g'}\} \subset \mathcal{P}_n$,

$$g = \sum_{g' \in G^{\deg(g)-1}} h_{g'} g', \tag{3.14}$$

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 [CLO92]. 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 $\mathbf{0}$.

To summarize, we need to answer the following questions to resolve the redundancy in the basis set.

- Given a vanishing polynomial g and a set of vanishing polynomials G, how can we know weather g can be generated by G?
- Is there any method that can work in the approximate setting? Namely, is there any method to detect whether g can be approximately generated by G?

The brief answers are as follows. We exploit the gradient of polynomials at the points of the given set. We will show that the redundancy is reflected in the linear dependency of the gradients at each point. Since the linear dependency of a vector to vectors can be checked by solving a least square problem, this approach works even in the approximate setting.

3.3.1 The gradient of redundant basis polynomials

We will now show that the gradient reveals the symbolic relations across polynomials. For example, if a vanishing polynomials g_1 for X is a polynomial multiple of another vanishing polynomial g_2, i.e., $g_1 = g_2 h$ for some $h \in \mathcal{P}_n$, then, 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. This can be readily proven as follows. For any $\boldsymbol{x} \in X$,

$$\begin{aligned}\nabla g_1(\boldsymbol{x}) &= h(\boldsymbol{x})\nabla g_2(\boldsymbol{x}) + g_2(\boldsymbol{x})\nabla h(\boldsymbol{x}),\\ &= h(\boldsymbol{x})\nabla g_2(\boldsymbol{x}).\end{aligned}$$

For the last equality, we used $g_2(\boldsymbol{x}) = 0$.

We can show a more general statement: if a vanishing polynomial g for X can be generated by a set of vanishing polynomials G, then for any $\boldsymbol{x} \in X$, $g(\boldsymbol{x}) \in \text{span}(\nabla G(\boldsymbol{x}))$ Formally, we have the following conjecture.

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

$$\nabla g(\boldsymbol{x}) = \sum_{g' \in G^{\deg(g)-1}} \alpha_{g',\boldsymbol{x}} \nabla g'(\boldsymbol{x}), \tag{3.15}$$

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 $g'(\boldsymbol{x}) = 0$. From the assumption $g \in \langle G^{\deg(g)-1} \rangle$, we can represent g as $g = \sum_{g' \in G^{\deg(g)-1}} g' h_{g'}$, for some $\{h_{g'}\} \subset \mathcal{P}_n$. Thus,

$$\begin{aligned}\nabla g(\boldsymbol{x}) &= \sum_{g' \in G^{\deg(g)-1}} h_{g'}(\boldsymbol{x})\nabla g'(\boldsymbol{x}) + g'(\boldsymbol{x})\nabla h_{g'}(\boldsymbol{x}),\\ &= \sum_{g' \in G^{\deg(g)-1}} h_{g'}(\boldsymbol{x})\nabla g'(\boldsymbol{x}),\end{aligned}$$

where we used $g'(\boldsymbol{x}) = 0$ in the last equality.

Basis reduction method

We now present a method to remove redundant polynomials. Given g and $G^{\deg(g)-1}$ in Conjecture 1, we solve the following least squares problem for each

$\boldsymbol{x} \in X$:

$$\min_{\boldsymbol{v} \in \mathbb{R}^{|G^{\deg(g)-1}|}} \|\nabla g(\boldsymbol{x}) - \boldsymbol{v}^\top \nabla G^{\deg(g)-1}(\boldsymbol{x})\|, \tag{3.16}$$

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.

From the sufficiency of Conjecture 1 , we can remove all the redundant polynomials in the form of Eq. (3.14) from the basis set by checking whether or not Eq. (3.15) 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. (3.15) 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. (3.15) 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.

Remark 9. *When the SBC algorithm with $\mathfrak{n} \neq \mathfrak{n}_{\mathrm{g}}$, it is also necessary to check the linear dependency of the gradient within G_t because $\mathfrak{N}(G_t)$ may not be a full-rank matrix. In this case, we need an additional procedure in our basis reduction as follows: (i) compute the rank of $\mathfrak{N}(G_t)$ and (ii) remove $|G_t| - \mathrm{rank}(\mathfrak{N}(G_t))$ polynomials from G_t according to the extent of vanishing in ascending order (polynomials with the small extent of vanishing are removed first).*

3.3.2 Numerical experiments of the basis reduction

We demonstrate that redundant basis sets can be reduced by the basis reduction method. The threshold of the basis reduction is set to 10^{-9}. 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 the reduction. As shown in Fig. 3.5, 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 the basis reduction method, both basis sets were successfully reduced to $\{g_1, g_2\}$.

We obtain the same result for six-point dataset $X = \{\cos(k\pi/3), \sin(k\pi/3)\}_{k=0,1,\dots,5}$ as shown in Fig. 3.7. Redundant polynomials are successfully found by the basis reduction method. Again, the correctness is confirmed by computing the Gröbner basis.

Next, we apply the basis reduction to the perturbed datasets. The 4-point dataset and 6-point dataset are perturbed by additive Gaussian noise $\mathcal{N}(0, 0.05)$ and the basis sets are computed with $\epsilon = 0.05$. The results are shown in Figs. 3.6 and 3.8. Note that the polynomials that are considered redundant by the reduction are not exactly but only approximately generated from the lower-degree basis polynomials. Thus, it is no longer meaningful to compute the exact Gröbner basis to find the *exact* redundant basis polynomials. A few polynomials of obtained basis sets are considered redundant by the basis reduction, which seems reasonable according to the noise-free case.

Both the third basis polynomial in VCA basis set in Fig. 3.6 and the second basis polynomial in VCA basis set in Fig. 3.6 are drawn in clutter lines, which do not go through the points. This is because this polynomial is close to the zero function; in fact, the coefficient norm of these polynomials are approximately 1.0e-16.

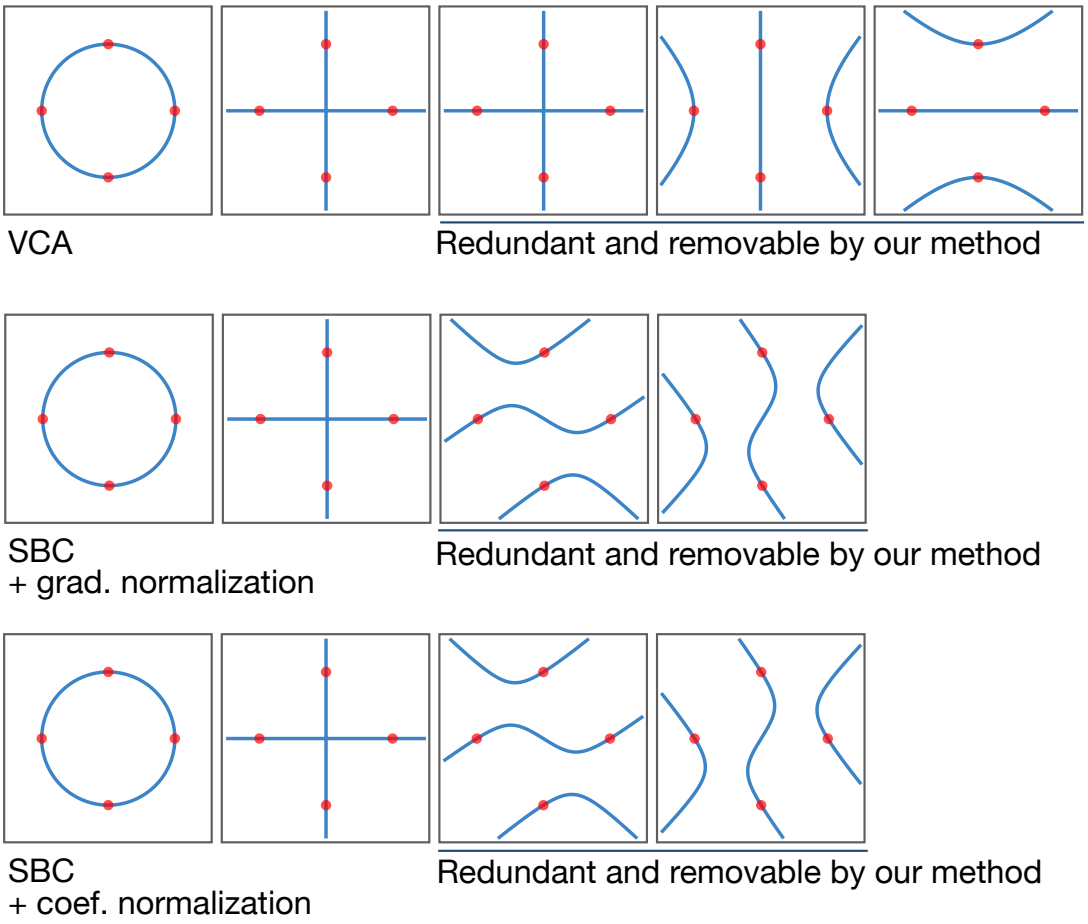

FIGURE 3.5: Sets of vanishing polynomials obtained by VCA (top row), by SBC-$\mathfrak{n}_g$ (middle row), and by SBC-$\mathfrak{n}_c$ (bottom row) for four-point dataset in a noise-free case. All sets contain redundant basis polynomials, which can be efficiently removed by the basis reduction.

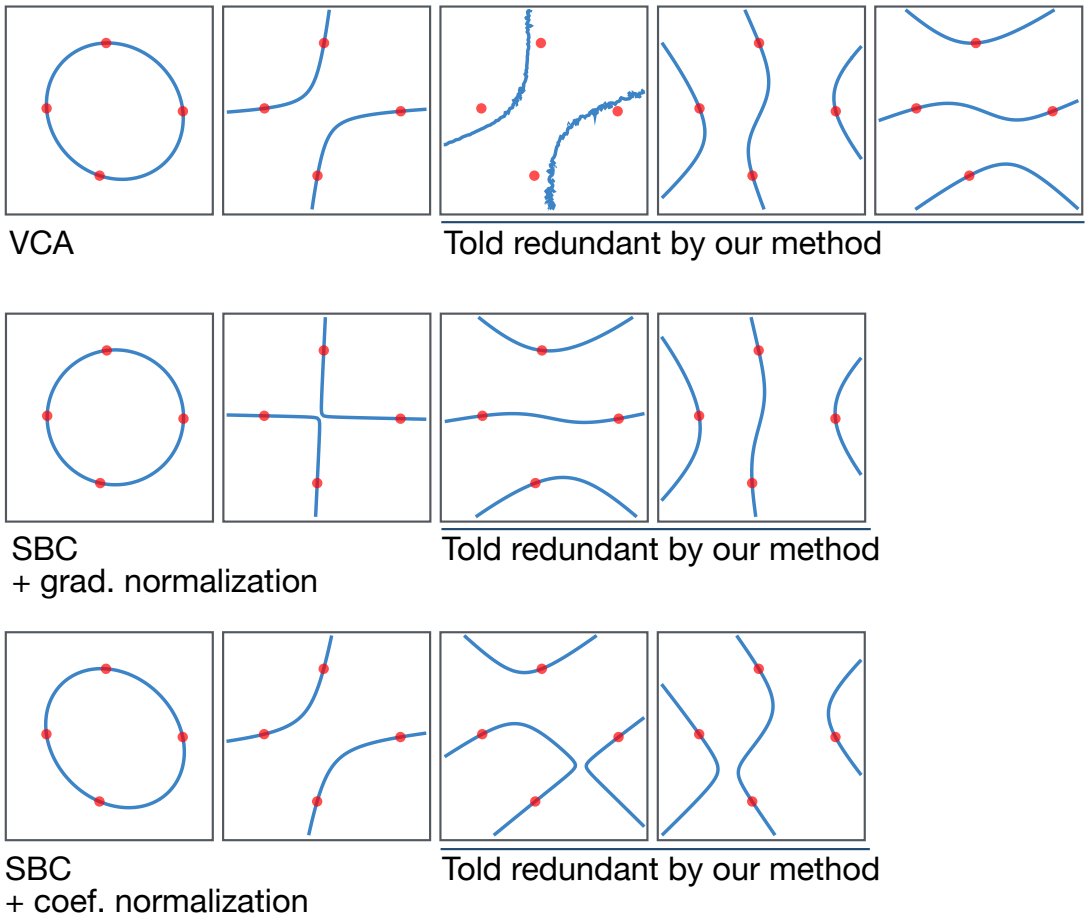

FIGURE 3.6: Sets of vanishing polynomials obtained by VCA (top row), by SBC-$\mathfrak{n}_g$ (middle row), and by SBC-$\mathfrak{n}_c$ (bottom row) for four-point dataset in a noisy case. All sets contain polynomials that are suggested as redundant by the basis reduction.

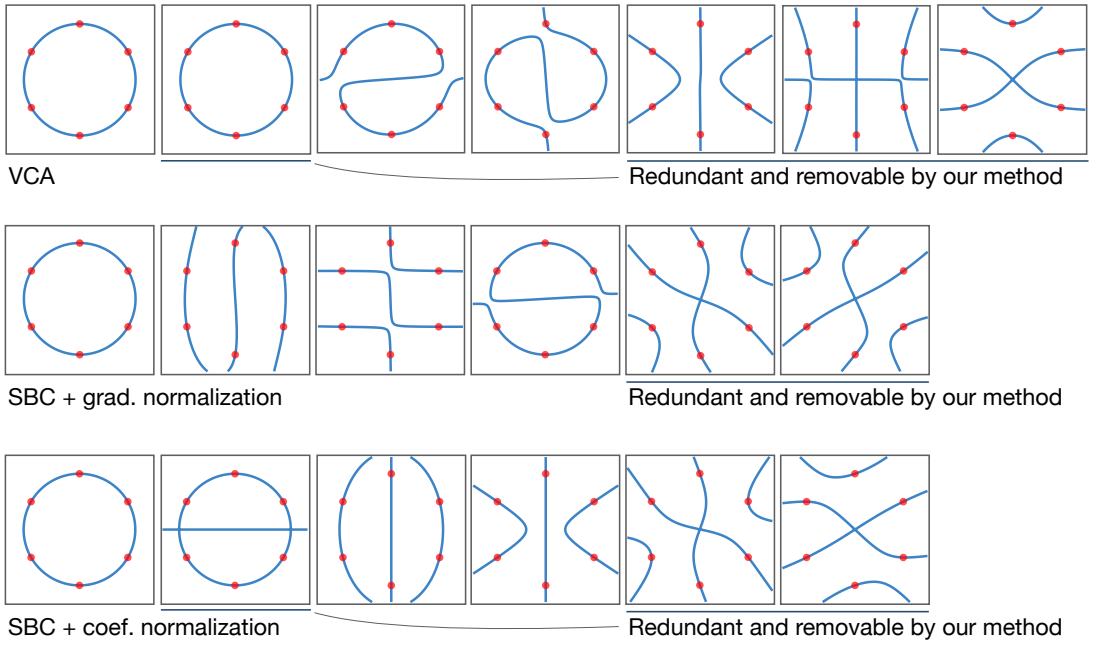

FIGURE 3.7: Sets of vanishing polynomials obtained by VCA (top row), by SBC-$\mathfrak{n}_g$ (middle row), and by SBC-$\mathfrak{n}_c$ (bottom row) for six-point dataset in a noise-free case. All sets contain redundant basis polynomials, which can be efficiently removed by the basis reduction.

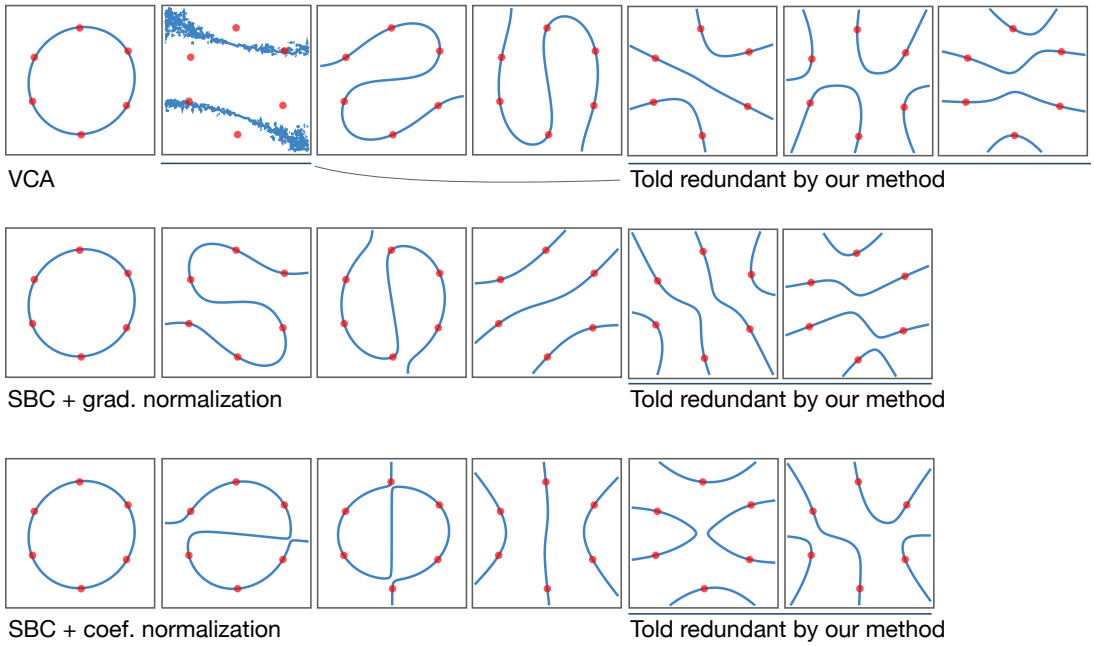

FIGURE 3.8: Sets of vanishing polynomials obtained by VCA (top row), by SBC-$\mathfrak{n}_g$ (middle row), and by SBC-$\mathfrak{n}_c$ (bottom row) for six-point dataset in a noisy case. All sets contain polynomials that are suggested as redundant by the basis reduction.

3.4 Inconsistency of the Basis Set on Translation and Scaling.

The goal of the basis construction is to reconstruct the algebraic variety on which the input set of points $X \subset \mathbb{R}^n$ lies. Let us consider the translation and scaling of X. Obviously, such transformation does not change the structure of the algebraic variety; the basis sets should consist of the same number of polynomials of the same degree before and after the translation and scaling. Therefore, we expect a basis construction algorithm to output consistent basis sets for X and transformed X.

Here, we present that the SBC algorithm with gradient normalization equips a sort of invariance for the input transformation. On the other hand, the SBC algorithm with coefficient normalization does not. This derives from the essential difference between gradient normalization and coefficient normalization—the former is data-dependent normalization whereas the latter is not.

Example 6. *Let us consider* $X = \{(0), (1)\}$ *and its translation* $\widetilde{X} = \{(l), (1+l)\}$, *where* $l \in \mathbb{R}$. *Then,* $g = x(x-1) + \epsilon$ *is an* ϵ*-vanishing polynomial for* X *and its natural counterpart for* $\widetilde{X}$ *is* $\widetilde{g} = (x-l)(x-l-1) + \epsilon$. *We expect both* g *and* $\widetilde{g}$ *are vanishing to the same extent even after normalization. This is the case for gradient-based normalization. Actually,* $(\mathrm{d}g/\mathrm{d}x)(X) = (\mathrm{d}\widetilde{g}/\mathrm{d}x)(\widetilde{X}) = (-1, 1)^\top$, *thus the gradient norm is the same. However, the coefficient norm of* $\widetilde{g}$ *is* $\sqrt{1 + (2l+1)^2 + (l^2+l)^2}$, *which varies according to* l.

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.

The goal of this subsection is to prove the following Proposition 4, which states a consistency of the SBC algorithm with the gradient norm (SBC-$\mathfrak{n}_{\mathrm{g}}$). It is worth noting that in contrast to Theorem 1 and Conjecture 1, this proposition holds even under noisy data points because it holds for arbitrary $\epsilon \geq 0$.

Proposition 4. *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 *is mean-centralized,* $X - \boldsymbol{b}$ *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 non-constant 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 to h (the definition will be introduced soon).*

The first statement argues the correspondence between G, $\widetilde{G}$, and $\widehat{G}$; if $g \in G$, there are corresponding polynomials $\widetilde{g}$ and $\widehat{g}$ of the same degree. Similarly, the second statement argues the correspondence between between F, $\widetilde{F}$, and $\widehat{F}$. Therefore, the basis construction shows a consistent behavior with translation and scaling of input.

The third statement argues how two corresponding polynomials h and $\widetilde{h}$ relate to each other at a translation of data. It argues that a translation of X results in the shift in variables of polynomials while the evaluation vectors are the same, i.e., $g(X) = \widetilde{g}(\widetilde{X})$. Note that although it is trivial that a translation of data can be handled by translated polynomials, it is not trivial that the *algorithm* outputs such polynomials.

The last statement argues how two corresponding polynomials h and $\widehat{h}$ relate to each other at a scaling of data. To elaborate the relation, we first introduce the following definition.

Definition 13 ((t, α)-degree-wise identical)**.** *Let $k \neq 0$ and let t be an integer. A polynomial $\widehat{h} \in \mathcal{P}_n$ is (t, α)-degree-wise identical to a polynomial $h \in \mathcal{P}_n$ if h and $\widehat{h}$ consist of the same terms up to scale, and any pair of the corresponding terms m of h and $\widehat{m}$ of $\widehat{h}$ satisfies $\widehat{m} = \alpha^{t - \deg(\widehat{m})} m$.*

Example 7. *$\widehat{h} = x^2y + 8y$ is $(3, 2)$-degree-wise identical to $h = x^2y + 2y$.*

In the last statement of Proposition 4, $\widehat{h}$ is $(1, \alpha)$-degree-wise identical to h. By definition, it indicates 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 $\widehat{m} = \alpha^{1-\tau} m$. Thus, as α increases, the magnitude of the coefficients of monomials decreases. In particular, the coefficients of higher-degree monomial decay more sharply. This

is quite natural because the evaluation value of highly nonlinear terms grows sharply as the input value increases.

As for the evaluation vector, the $(1, \alpha)$-degree-wise identicality of $\widehat{h}$ to h, where $\widehat{h}$ and h are both normalized with respect to $\mathfrak{n}_{\mathrm{g}}$, implies the following relation (cf. Lemma 7):

$$\widehat{h}(\alpha X) = \alpha h(X). \tag{3.17}$$

In words, the scaling by α on input X only *linearly* affects the evaluation vectors of the *nonlinear* polynomials of the basis sets. Thus, we only need linearly scaled threshold $|\alpha|\epsilon$ for αX. Without this property, a 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.

The relation Eq. (3.17) is proven as a direct result from the following lemma.

Lemma 5. *Let a polynomial $\widehat{h} \in \mathcal{P}_n$ be (t, α)-degree-wise identical to a polynomial $h \in \mathcal{P}_n$. Let $X \subset \mathbb{R}^n$ be a set of points. Then, $\widehat{h}(\alpha X) = \alpha^t h(X)$ and $\nabla\widehat{h}(\alpha X) = \alpha^{t-1}\nabla h(X)$, and thus,*

$$\frac{\widehat{h}(\alpha X)}{\|\mathfrak{n}_{\mathrm{g}}(\widehat{h}; \alpha X)\|} = \alpha \frac{h(X)}{\|\mathfrak{n}_{\mathrm{g}}(h; X)\|}.$$

Proof. Let m and $\widehat{m}$ be any corresponding terms between h and $\widehat{h}$, which satisfy $\deg(\widehat{m}) = \deg(m)$ and $\widehat{m} = \alpha^{t-\deg(\widehat{m})} m$. Then,

$$\begin{aligned}
\widehat{m}(\alpha X) &= \alpha^{t-\deg(\widehat{m})} m(\alpha X), \\
&= \alpha^{t-\deg(\widehat{m})} \alpha^{\deg(m)} m(X), \\
&= \alpha^t m(X).
\end{aligned}$$

Similarly, for any k,

$$\begin{aligned}
\frac{\partial \widehat{m}}{\partial x_k}(\alpha X) &= \alpha^{t-\deg(\widehat{m})} \frac{\partial m}{\partial x_k}(\alpha X), \\
&= \alpha^{t-\deg(\widehat{m})} \alpha^{\deg(m)-1} \frac{\partial m}{\partial x_k}(X), \\
&= \alpha^{t-1} \frac{\partial m}{\partial x_k}(X),
\end{aligned}$$

resulting in $\nabla\widehat{m}(\alpha X) = \alpha^{t-1}\nabla m(X)$. □

Now, we prove three lemmas (Lemmas 6, 7, and 8). Proposition 4 will be proven mainly from Lemma 7, where Lemma 7 is proven from Lemmas 6 and 8.

Lemma 6. *Let us consider two sets of polynomials, $H = \{h_1, h_2, ..., h_s\} \subset \mathcal{P}_n$ and $\widehat{H} = \{\widehat{h}_1, \widehat{h}_2, ..., \widehat{h}_s\} \subset \mathcal{P}_n$, where $\widehat{h}_i$ is (t, α)-degree-wise identical to h for $i = 1, 2, ..., s$. Then, any pair of nonzero vectors $\widehat{\boldsymbol{w}}, \boldsymbol{w} \in \mathbb{R}^s$ such that $\widehat{\boldsymbol{w}} = \alpha^\tau \boldsymbol{w}$ yields a polynomial $\widehat{H}\boldsymbol{w}$ that is $(t + \tau, \alpha)$-degree-wise identical to $H\boldsymbol{w}$.*

Proof. The proof is trivial from Definition 13. □

Lemma 7. *Suppose we perform SBC-$\mathfrak{n}_\mathrm{g}$ for (X, ϵ) and for $(\alpha X, |\alpha|\epsilon)$, $(k \neq 0)$, and obtain (F^τ, G^τ) and $(\widehat{F}^\tau, \widehat{G}^\tau)$, respectively, up to degree $t = \tau$. Suppose that for all $t \leq \tau$, $F_t \cup G_t$ and $\widehat{F}_t \cup \widehat{G}_t$ have one-to-one correspondence such that for any corresponding pair $h \in F_t \cup G_t$ and $\widehat{h} \in \widehat{F}_t \cup \widehat{G}_t$, $\widehat{h}$ is degree-wise-$(1, \alpha)$ identical to h. Then, the same claim holds for $t = \tau + 1$.*

Proof. For $t = \tau + 1$, $C_{\tau+1}$ and $\widehat{C}_{\tau+1}$ are generated from (F_1, F_τ) and $(\widehat{F}_1, \widehat{F}_{\tau+1})$, respectively. From the one-to-one correspondence between F_t and $\widehat{F}_t$ for $t \leq \tau$, There is also one-to-one correspondence between $C^{\mathrm{pre}}_{\tau+1}$ and $\widehat{C}^{\mathrm{pre}}_{\tau+1}$. Moreover, $\widehat{c}^{\mathrm{pre}} \in \widehat{C}^{\mathrm{pre}}_{\tau+1}$ is $(2, k)$-degree-wise identical to the corresponding $c^{\mathrm{pre}} \in C^{\mathrm{pre}}_{\tau+1}$ due to the assumption. Suppose c^{pre} and $\widehat{c}^{\mathrm{pre}}$ becomes $c \in C_{\tau+1}$ and $\widehat{c} \in \widehat{C}_{\tau+1}$ after the orthogonalization Eq. (2.2). In can be shown that $\widehat{c}$ is $(2, k)$-degree-wise identical to c (cf. Lemma 8).

From Lemma 5, $\widehat{h}(\alpha X) = \alpha^2 h(X)$ and $\nabla\widehat{h}(\alpha X) = k\nabla h(X)$. Therefore, $\widehat{C}_{\tau+1}(\alpha\widehat{X}) = \alpha^2 C_{\tau+1}(X)$ and $\nabla\widehat{C}_\tau(\alpha\widehat{X}) = k\nabla C_{\tau+1}(X)$. Note that $\mathfrak{n}_\mathrm{g}(\widehat{C}_{\tau+1})(\alpha X) = \alpha^2\mathfrak{n}_\mathrm{g}(C_{\tau+1})(X)$. Thus, the generalized eigenvalue problem Eq. (3.2)

$$\widehat{C}_{\tau+1}(\alpha X)^\top \widehat{C}_{\tau+1}(\alpha X)\widehat{V} = \mathfrak{N}_\mathrm{g}(\widehat{C}_{\tau+1}; \alpha X)\widehat{V}\widehat{\Lambda},$$

where $\mathfrak{N}_\mathrm{g}$ is $\mathfrak{N}$ for the gradient-based normalization, is equivalent to

$$C_{\tau+1}(X)^\top C_{\tau+1}(X)\widehat{V} = \frac{1}{\alpha^2}\mathfrak{N}_\mathrm{g}(C_{\tau+1}; X)\widehat{V}\widehat{\Lambda},$$

which leads to $\widehat{\Lambda} = \alpha^2\Lambda$. Also,

$$\widehat{V}^\top \widehat{C}_{\tau+1}(\alpha X)^\top \widehat{C}_{\tau+1}(\alpha X)\widehat{V} = I,$$
$$\alpha^2 \widehat{V}^\top C_{\tau+1}(S)^\top C_{\tau+1}(X)\widehat{V} = I.$$

Comparing with $V^\top C_{\tau+1}(X)^\top C_{\tau+1}(X)V = I$, we obtain $\widehat{V} = \alpha^{-1}V$.

Let $\boldsymbol{v}_i$ and $\widehat{\boldsymbol{v}}_i$ be the i-th column of V and $\widehat{V}$, respectively. From $\widehat{\boldsymbol{v}}_i = \alpha^{-1}\boldsymbol{v}_i$ and Lemma 6, $\widehat{C}_1\widehat{\boldsymbol{v}}_i$ is $(1,\alpha)$-degree-wise identical to $C_{\tau+1}\boldsymbol{v}_i$. Hence, any polynomials $h \in F_{\tau+1} \cup G_{\tau+1}$ and $\widehat{h} \in \widehat{F}_{\tau+} \cup \widehat{G}_{\tau+1}$ satisfy $\widehat{h}(\alpha X) = \alpha h(X)$. This fact is also supported by $\widehat{\Lambda} = \alpha^2\Lambda$ (recall that the square root of the eigenvalues corresponds to the extent of vanishing). Note that polynomials in $\widehat{C}_{\tau+1}\widehat{V}$ are assorted into $\widehat{F}_{\tau+1}$ or $\widehat{G}_{\tau+1}$ by the threshold $k\epsilon$, which leads to the same classification as $F_{\tau+1}$ and $G_{\tau+1}$ by ϵ. Thus, the one-to-one correspondence is kept between $F_{\tau+1}$ and $\widehat{F}_{\tau+1}$ and also between $G_{\tau+1}$ and $\widehat{G}_{\tau+1}$. □

Lemma 8. *Consider the same setting in Lemma 7. Suppose* $h^{\mathrm{pre}} \in C^{\mathrm{pre}}_{\tau+1}$ *and* $\widehat{c}^{\mathrm{pre}} \in \widehat{C}^{\mathrm{pre}}_{\tau+1}$ *become* $c \in C_{\tau+1}$ *and* $\widehat{c} \in \widehat{C}_{\tau+1}$*, respectively, after the orthogonalization Eq. (2.2). Then,* $\widehat{c}$ *is* $(2,\alpha)$*-degree-wise identical to* c*.*

Proof. The entry-wise description of the orthogonalization Eq. (1) for $C_{\tau+1}$ and $\widehat{C}_{\tau+1}$ is respectively as follows.

$$
\begin{aligned}
c &= c^{\mathrm{pre}} - F^{\tau}F^{\tau}(X)^{\dagger}c^{\mathrm{pre}}(X),\\
\widehat{c} &= \widehat{c}^{\mathrm{pre}} - \widehat{F}^{\tau}\widehat{F}^{\tau}(\alpha X)^{\dagger}\widehat{c}^{\mathrm{pre}}(\alpha X).
\end{aligned}
$$

Let $\widehat{\boldsymbol{w}} = \widehat{F}^{\tau}(\alpha X)^{\dagger}\widehat{c}^{\mathrm{pre}}(\alpha X)$ and $\boldsymbol{w} = F^{\tau}(X)^{\dagger}c^{\mathrm{pre}}(X)$. We will now show $\widehat{\boldsymbol{w}} = k\boldsymbol{w}$. If this holds, each entry of $\widehat{F}^{\tau}\widehat{\boldsymbol{w}}$ becomes $(2,k)$-degree-wise identical to the corresponding entry of $F^{\tau}\boldsymbol{w}$. Thus, from Lemma 6, $\widehat{c}$ is $(2,\alpha)$-degree-wise identical to c.

First, note that the column vectors of $\widehat{F}^{\tau}(\alpha X)$ are mutually orthogonal by construction because the orthogonalization makes $\mathrm{span}(\widehat{F}_{t_1}(\alpha X))$ and $\mathrm{span}(\widehat{F}_{t_2}(\alpha X))$ mutually orthogonal for any $t_1 \neq t_2$, and the generalized eigenvalue decomposition makes the columns of $\widehat{F}_t(\alpha X)$ mutually orthogonal for $t \leq \tau$. Therefore,

$$
\begin{aligned}
\widehat{D} &:= \widehat{F}^{\tau}(\alpha X)^{\top}\widehat{F}^{\tau}(\alpha X),\\
&= \alpha^2 F^{\tau}(X)^{\top}F^{\tau}(X),\\
&=: \alpha^2 D,
\end{aligned}
$$

where both $\widehat{D}$ and D are diagonal matrices with positive entries in their diagonal. Hence, the pseudo-inverse becomes

$$\begin{aligned}\widehat{F}^{\tau}(\alpha X)^{\dagger} &= \widehat{D}^{-1}\widehat{F}^{\tau}(\alpha X)^{\top}, \\ &= (\alpha^{-2}D^{-1})(\alpha F^{\tau}(X)^{\top}), \\ &= \alpha^{-1}D^{-1}F^{\tau}(X)^{\top}, \\ &= \alpha^{-1}F^{\tau}(X)^{\dagger}.\end{aligned}$$

Therefore,

$$\begin{aligned}\widehat{\boldsymbol{w}} &= \widehat{F}^{\tau}(\alpha X)^{\dagger}\widehat{c}^{\mathrm{pre}}(\alpha X), \\ &= \alpha^{-1}F^{\tau}(X)^{\dagger}(\alpha^{2}c^{\mathrm{pre}}(X)), \\ &= \alpha F^{\tau}(X)^{\dagger}c^{\mathrm{pre}}(X), \\ &= \alpha\boldsymbol{w}.\end{aligned}$$

□

Now, Proposition 4 is proven as follows.

Proof of Proposition 4. In the proof, we consider the basis construction for (X, ϵ), $(X - \boldsymbol{b}, \epsilon)$, and $(\alpha X, |\alpha|\epsilon)$. We use notations such as H, $\widetilde{H}$, and $\widehat{H}$ for the corresponding symbols across three basis constructions.

First, we consider the case of the translation. At $t = 1$, it is $C_1^{\mathrm{pre}} = \widetilde{C}_1^{\mathrm{pre}}$. By Proposition 1, both $C_1(X)$ and $\widetilde{C}_1(X)$ are mean centralized. Thus, the effect of translation is canceled in terms of the evaluation matrices. Therefore, the succeeding basis construction proceeds in the same way. Symbolically, the mean centralization is translated in the shift of variables.

Next, we consider the case of the scaling. From the assumption, the mean vector of X and αX is the zero vector. Thus, it is $C_1 = \widehat{C}_1$ and $\alpha C_1(X) = \widehat{C}_1(\alpha X)$ because C_1 and $\widehat{C}_1$ consist of linear polynomials. It is also $\mathfrak{n}_{\mathrm{g}}(C_1; X) = \mathfrak{n}_{\mathrm{g}}(\widehat{C}_1; \alpha X)$ because the partial derivatives of linear polynomials are constant polynomials. Thus,

$$\begin{aligned}&\widehat{C}_1(\alpha X)^{\top}\widehat{C}_1(\alpha X)\widehat{V} = \mathfrak{N}_{\mathrm{g}}(\widehat{C}_1; \alpha X)\widehat{V}\widehat{\Lambda}, \\ \iff\ &C_1(X)^{\top}C_1(X)V = \frac{1}{\alpha^2}\mathfrak{N}_{\mathrm{g}}(C_1; X)V\Lambda,\end{aligned}$$

which implies $V = \widehat{V}$ and $\alpha^2\Lambda = \widehat{\Lambda}$. Therefore, $F_1 \cup G_1$ and $\widehat{F}_1 \cup \widehat{G}_1$ consist of the same polynomials and for any corresponding pair $h \in F_1 \cup G_1$ and $\widehat{h} \in \widehat{F}_1 \cup \widehat{G}_1$, where $h = \widehat{h}$ in this case, $\widehat{h}$ is (1,α)-degree-wise identical to h. The extent of vanishing of the polynomials in $\widehat{F}_1 \cup \widehat{G}_1$ are scaled by α from that of the polynomials in $F_1 \cup G_1$. Thus, by using $\widehat{\epsilon} = |\alpha|\epsilon$, $F_1 = \widehat{F}_1$ and $G_1 = \widehat{G}_1$. By induction and Lemma 7, we can conclude our proof. □

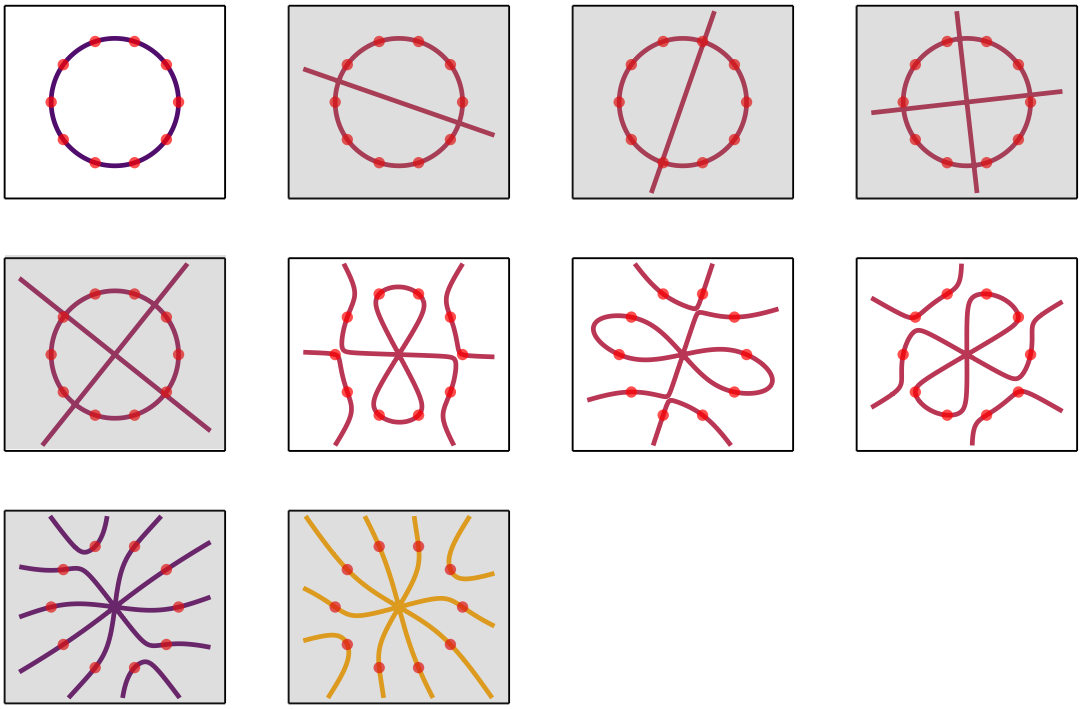

FIGURE 3.9: Vanishing polynomials obtained by SBC-$\mathfrak{n}_g$ with $\epsilon = 0$ for points sampled from a unit circle. Polynomials that are considered redundant by our basis reduction are grayed out.

3.5 Dimension Restriction

In this thesis, we discuss the basis construction algorithm for vanishing ideals of points. In particular, when $\epsilon = 0$, the obtained basis set (say, G) generates any polynomial that vanish for given set of points X. In other words, the algebraic variety that is represented by G is a (finite) set of points X. This is why vanishing ideals are referred to as zero-dimensional ideals in some literature. All basis construction algorithms that were introduced in this thesis aims at the basis construction of zero-dimensional ideals.

However, in many applications, the dimensionality of the target algebraic variety is not necessarily zero; data points can be sampled from a smooth algebraic hypersurface (the dimension of hypersurfaces is one). For example, as can be seen in Fig. 3.9, a basis set for the vanishing ideal of points that are sampled from a unit circle includes a unit circle; however, the basis set also includes highly nonlinear polynomials. Intuitively, such polynomials does not have much information on the target algebraic variety (in this case, a circle).

One typical approach is restricting the degree or number of basis polynomials. However, it is difficult to choose proper number for these parameters. Instead, we propose to restrict the dimension of the algebraic variety.

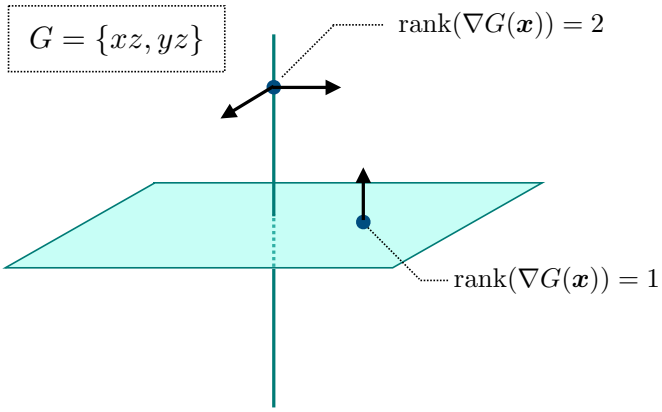

FIGURE 3.10: The union of a line and a plain, which is the zero set of $G = \{xz, yz\}$. The rank of $\nabla G(\boldsymbol{x})$ is two at the point on the line and one on the plain. This indicates the dimension of this algebraic variety is $3 - \min\{1, 2\} = 2$

Definition 14 (Dimension of an algebraic variety)**.** *Let $\mathcal{V} \subset \mathbb{R}^n$ is a algebraic variety, which is the zero set of a polynomial set G. Dimension $\dim(\mathcal{V})$ of $\mathcal{V}$ is defined as follows.*

$$\dim(\mathcal{V}) = n - \min_{\boldsymbol{x} \in \mathcal{V}} \mathrm{rank}(\nabla G(\boldsymbol{x})),$$

where $\nabla G(\boldsymbol{x}) \in \mathbb{R}^{n \times |G|}(\boldsymbol{x})$ is a matrix whose i-th column is the gradient (at $\boldsymbol{x}$) of the i-th polynomial of G. First, we define the dimension of an algebraic vairty.

We provide an example in Fig. 3.10. In this example, we consider the union of a line and a plain, which is the zero set of $G = \{xz, yz\}$. At the point on the vertical line, the rank of $\nabla G(\boldsymbol{x})$ is one and at the point on the plain, the rank of $\nabla G(\boldsymbol{x})$ is two. Therefore, the dimension of this variety is $3 - 1 = 2$.

In practice, we estimate the dimension of the algebraic variety from sample points $X \subset \mathbb{R}^n$. In particular, the following two numbers are of our interest.

$$d_{\max} = n - \min_{\boldsymbol{x} \in X} \mathrm{rank}(\nabla G(\boldsymbol{x})),$$
$$d_{\min} = n - \max_{\boldsymbol{x} \in X} \mathrm{rank}(\nabla G(\boldsymbol{x})).$$

Instead of the degree and number of basis polynomials, we propose to restrict these numbers. For example, restricting $d_{\max}$ corresponds to restricting the dimension of the algebraic variety to estimate. In the case of the union of a line and a plain in Fig. 3.10, we can impose $d_{\max} \leq 2$ on the basis construction. One can also consider $d_{\min} \geq 1$. In both cases, the output basis set becomes much smaller. The major advantage of using dimension rather than the degree

or number is that the range of dimension is evident. If one deals with data points in $\mathbb{R}^n$, then the dimension ranges from 1 to n. This is not the case for the degree restriction nor the number restriction; it is difficult to estimate the degree and number of basis polynomials without performing the basis construction.

Example 8. *In the case of Fig. 3.9, one can obtain the circle polynomial only by setting* $d_{\max} = 1$ *(or* $d_{\max} = 1$*).*

Note that with dimension restriction, the output set of vanishing polynomials does not form a basis set any longer as the restriction of the degree and number of polynomials does not; that is, the polynomial set only generate subset of the vanishing polynomials. It is an interesting future direction to consider general statements for the range of vanishing polynomials that can be generated by the imparfect basis set.

Chapter 4

Tradeoff between Algebraicity and Noise Tolerance

.

The central subject of this thesis is the basis construction of algebraic varieties *under noise*. In many applications, available data are exposed to noise for various reasons such as the measurement error. Computing not exact but approximate vanishing polynomials is a fundamental approach. However, this approximation destroys the sound algebraic structure of the vanishing ideal.

For example, in the left panel of Fig. 4.1, three curves only loosely intersect with each other around noisy data points (red dots). This indicates that the approximate basis set that is computed by the basis construction does not hold a strict algebraic structure any longer. Of course, we can realize the tight intersection by setting ϵ to zero or small value. However, this only results in a overfitting to noisy points; consequently, the obtained algebraic variety is far from the target algebraic variety of the noise-free data.

The goal of this chapter is to present a method to deal with this tradeoff—how can we avoid the overfitting to noise while preserving the algebraicity? This goal is illustrated in the right panel of Fig. 4.1, where three curves are tightly intersecting with each other at a few points (blue circles) as well as loosely intersecting around noisy points (red dots).

In this chapter, we address a new task that jointly discovers a set of polynomials and *summarized* data points (called data knots) from the input data. Specifically, given a set of data points X, we seek a set of polynomials G and data knots Z such that the polynomials in G loosely vanish for X and tightly vanish for Z.

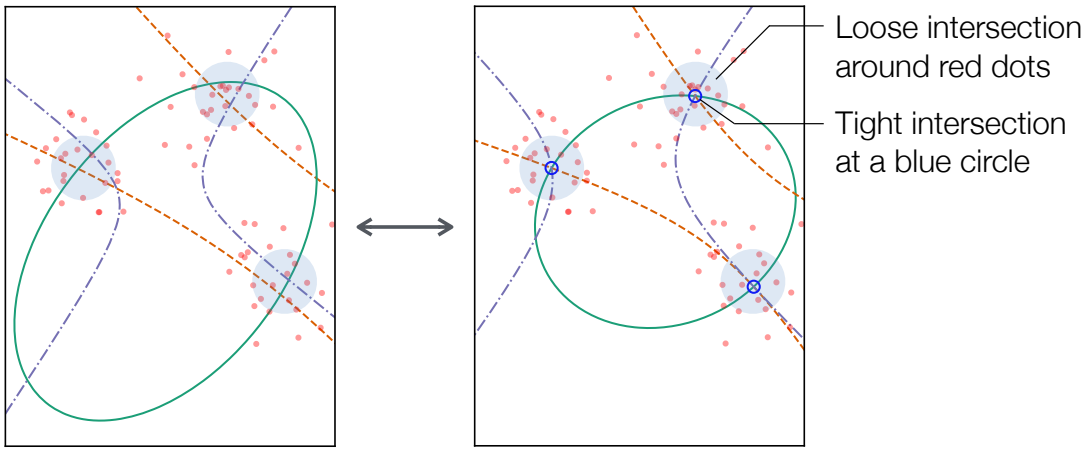

FIGURE 4.1: A tradeoff between the algebraicity and noise tolerance. (Left panel) three curves only loosely intersect with each other around noisy data points (red dots). (Right panel) three curves not only intersect with each other around noisy data points but also tightly intersect at blue circles.

Formally,

$$\forall g \in G,\ \|g(X)\| \leq \epsilon,\ \|g(Z)\| \leq \delta, \tag{4.1}$$

where ε and δ are given error tolerances that require g to be ε-vanishing for X and δ-vanishing for Z. It is assumed that δ is much smaller value than ε (or zero for exact vanish). In this way, we can achieve both two objectives (noise tolerance and algebraic structure) at once, whereas other methods only focus on the former and the classical methods only focus on the latter.

The method presented here consists of two components.

Double-scale basis construction for two datasets and two scales—For (X, ϵ) and (Z, δ'), vanishing polynomials satisfying Eq. (4.1) are constructed. Two problems are successively solved. First, by solving a generalized eigenvalue problem, we discover the subspace of polynomials that are ϵ-vanishing for X. From this subspace, we discover δ'-vanishing polynomials for Z by performing the Singular Value Decomposition (SVD).

Data knotting for pursuing exact vanishing—Given approximate vanishing polynomials and nonvanishing polynomials, tentative data knots Z is refined by gradient-based scheme. Gradient-based scheme updates Z to minimize the vanishing.

These components are alternatingly and repeatedly performed. One one hand, δ' gradually decreases from ϵ to δ and on the other hand, a set of data knots Z are updated.

4.1 Double-Scale Basis Construction

We now present the double-scale basis construction method, which generates polynomials that are ϵ-vanishing for X and δ-vanishing for Z from the candidate polynomials C_t. This method consists of two stages. First, a set of ϵ-vanishing polynomials for X are generated by solving the following problem.

$$C_t(X)^\top C_t(X)V = \mathfrak{N}(C_t; X)V\Lambda.$$

Let us split V into two matrices $\widetilde{V}$ and V_ϵ, where $\widetilde{V}$ and V_ϵ consist of the generalized eigenvectors corresponding to the generalized eigenvalues exceeding ϵ^2 and and not exceeding ϵ^2, respectively. Thus, C_tV_ϵ is a set of ϵ-vanishing polynomials for X. Then, we apply SVD to $C_t(Z)V_\epsilon$.

$$C_t(Z)V_\epsilon = UDW^\top,$$

where $U \in \mathbb{R}^{|Z|\times|Z|}$ and $W \in \mathbb{R}^{|C_t|\times|C_t|}$ are orthogonal matrices and D is a rectangular diagonal matrix with singular values along its diagonals. Similar to $\widetilde{V}$ and V_ϵ, we split W into $\widetilde{W}$ and W_δ, where $\widetilde{W}$ and W_δ consist of the right singular vectors corresponding to the singular values exceeding δ and and not exceeding δ, respectively.

Proposition 5. *Any polynomial $g \in C_tV_\epsilon W_{\delta'}$ is ϵ-vanishing for X and δ'-vanishing for Z. In addition, polynomials in $C_tV_\epsilon W_{\delta'}$ are normalized and mutually orthogonal with respect to $\mathfrak{n}(\,\cdot\,; X)$ i.e., $\|\mathfrak{n}(g; X)\| = 1$.*

We first prove the following lemma.

Lemma 9. *Let C be a set of polynomials and let $V = (\boldsymbol{v}_1\ \boldsymbol{v}_2\ \cdots\ \boldsymbol{v}_s) \in \mathbb{R}^{|X|\times s}$, where X is a set of points. Then, for any unit vector $\boldsymbol{w} \in \mathbb{R}^s$,*

$$\min_i \|(C\boldsymbol{v}_i)(X)\| \le \|(CV\boldsymbol{w})(X)\| \le \max_i \|(C\boldsymbol{v}_i)(X)\|.$$

Proof. Note that $\|(CV\boldsymbol{w})(X)\|^2 = \sum_{i=1}^s \|(C\boldsymbol{v}_i)(X)\|^2 w_i^2$, where w_i is the i-th entry of $\boldsymbol{w}$. Thus,

$$\min_i \|(C\boldsymbol{v}_i)(X)\|^2 \sum_{i=1}^s w_i^2 \le \sum_{i=1}^s \|(C\boldsymbol{v}_i)(X)\|^2 w_i^2 \le \max_i \|(C\boldsymbol{v}_i)(X)\|^2 \sum_{i=1}^s w_i^2.$$

With $\|\boldsymbol{w}\|^2 = 1$, we can conclude the proof. □

This lemma tells us that (i) the linear combination of ϵ-vanishing polynomials

by any unit vector yields an ϵ-vanishing polynomial and (ii) the linear combination of ϵ-nonvanishing polynomials by any unit vector yields an ϵ-nonvanishing polynomial.

Proof of Proposition 5. By construction, $C_t V_\epsilon$ is a set of ϵ-vanishing polynomials for X and these polynomials are normalized with respect to $\mathfrak{n}(\,\cdot\,; Z)$. By Lemma 9, $C_t V_\epsilon W$ is a set of ϵ-vanishing polynomials for X. In particular, by construction, $C_t V_\epsilon W_{\delta'}$ is a set of δ'-vanishing polynomials for Z. Note that

$$W^\top V^\top \mathfrak{N}(C_t; X) V W = W^\top W = I.$$

Thus, all the polynomials in $C_t V_\epsilon W$ are normalized and mutually orthogonal with respect to $\mathfrak{n}(\,\cdot\,; X)$. □

4.2 Data Knotting

Given a set of approximate vanishing polynomials $G^t = \bigcup_{\tau=0}^t G_\tau$ that are obtained up to degree t, data knotting refines a set of data knots from Z to Z^* so that these polynomials vanish more tightly for Z^*. The problem of our interest is in the following form.

$$\min_{Z^*} \sum_{g \in G^t} w_g \|g(Z^*)\|^2, \quad \text{s.t. } Z^* \in \mathcal{D}, \tag{4.2}$$

where $w_g \in \mathbb{R}$ is a weight for g and $\mathcal{D}$ is some constraint space for Z^*. Since δ' is not small enough before convergence, $\|g_i(Z)\|$ for tentative Z distributes moderately wide range across different g; some polynomials vanish relatively more tightly than others. The weights $\{w_g\}_{g \in G^t}$ are introduced to take these difference into account. Specifically, we defined w_i as follows:

$$w_i = \frac{1}{\|g_i(Z)\|^2}.$$

Consequently, data knots are updated to make well-vanishing polynomials more tightly vanishing for Z^* at the cost of deterioration of the extent of vanishing of other polynomials. One may consider that such survival-of-the-fittest principle may cause overfitting. However, this is less likely happen because the basis construction and data knotting are performed from lower to higher degree, and tentative Z at degree t has already refined for polynomials G^{t-1}.

Next, we discuss the constraint space $\mathcal{D}$ for Z^*. This constraint is expected to restrict the deviation from the original data points; otherwise, the optimal Z^* may converge to a single point. We split the constraint space $\mathcal{D}$ into $\{\mathcal{D}_k\}_{k=1,2,\dots,|Z|}$, where $\mathcal{D}_k$ is a constraint space for the k-th points z_k of Z, and

$$\mathcal{D}_k = \left\{ \widetilde{z}_k \mid \|F^t(\widetilde{z}_k) - F^t(z_k)\| \leq \delta' \cdot \sqrt{|F^t|} \right\}, \tag{4.3}$$

where $F^t = \cup_{i=0}^{t} F_t$ is a set of non-vanishing polynomials obtained up to degree t. Intuitively, the constraint $\mathcal{D}_k$ restrict z_k^* to be near from the start point z_k. The distance of points are the Euclidean norm of the nonlinear feature of points by projection $F^t(\cdot)$, and the distance is bounded to δ' normalized by the dimensionality of the feature space.

In the linear case, the data knotting can be optimally solved.

Proposition 6. *Let G_1 and F_1 be a set of ϵ-vanishing polynomials for Z and a set of ϵ-nonvanishing polynomials for Z, respectively, obtained by SBC-$\mathfrak{n}_{\mathrm{g}}$. Let Z be denoted by $Z = Z_c + \boldsymbol{m}$, where $\boldsymbol{m}$ is the mean of Z and Z_c is the mean-subtracted component of Z. Then, the optimal solution of Eq. (4.2) is*

$$Z^* = Z_c \widetilde{V} \widetilde{V}^\top + \boldsymbol{m},$$

where $\widetilde{V}$ is a matrix whose columns consist of the right singular vectors of Z_c corresponding to the singular values exceeding ϵ.

Proof. Let us consider SVD of $Z_c = UD(\widetilde{V}\ V_\epsilon)^\top$. Note that in SBC-$\mathfrak{n}_{\mathrm{g}}$, from Proposition 1, it is $C_1 = Z_c$. Thus, it is $G_1(Z) = Z_c V_\epsilon$ and $F_1(Z) = Z_c \widetilde{V}$. Since $\widetilde{V}^\top V_\epsilon = O$,

$$G_1(Z^*) = (Z^* - \boldsymbol{m}) V_\epsilon = Z_c \widetilde{V} \widetilde{V}^\top V_\epsilon = O.$$

Similarly, using $\widetilde{V}^\top \widetilde{V} = I$,

$$F_1(Z^*) = (Z^* - \boldsymbol{m}) \widetilde{V} = Z_c \widetilde{V} \widetilde{V}^\top \widetilde{V} = F_1(Z).$$

Therefore, we can achieve the minimum by $Z^* = Z_c \widetilde{V} \widetilde{V}^\top + \boldsymbol{m}$ in the constraint space Eq. (4.3) (note that $\|F_1(Z^*) - F_1(Z)\| = 0$). □

With such Z^* in Proposition 6, G_1 becomes a set of ϵ-vanishing polynomials for Z and 0-vanishing polynomials for Z^* and F_1 becomes a set of ϵ-nonvanishing polynomials for Z and 0-nonvanishing polynomials for Z^*.

In the nonlinear case, it is difficult to obtain the optimal Z^*. Thus, we resort to the gradient-based optimization. As already shown, the gradient of the polynomials can be efficiently obtained in the SBC algorithm.

4.3 Scale Control and Resetting Framework

We now present the overall algorithm, which is based on the SBC algorithm. Step 1 is same as that of the SBC algorithm. At Step 2, we alternatingly and repeatedly perform double-scale basis construction and data knotting. At Step 3, we select the next step from "terminate", "reset", or "continue". When "reset" is selected, then all the items but Z and δ' are initialized and the calculation restart from $t = 1$.

4.3.1 Procedures

The input of is a set of points $X \subset \mathbb{R}^n$, two error tolerances $\epsilon, \delta \geq 0$, and a cooling parameter $0 < \gamma < 1$. The output is basis sets (G, F) and the data knots Z. By initializing $G_0 = \{\}$, $F_0 = m$, $Z = X$, $\delta' = \delta$, where m is a nonzero constant polynomial, the following steps are performed for degree $t = 1, 2, ...$ In the following, we use notations $G^t = \bigcup_{\tau=0}^{t} G_\tau$ and $F^t = \bigcup_{\tau=0}^{t} F_\tau$.

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.

$$C_t = C_t^{\mathrm{pre}} - F^{t-1}F^{t-1}(X)^{\dagger}C_t^{\mathrm{pre}}(X), \tag{4.4}$$

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

Step 2: Basis construction of degree t Set $\eta = \delta'$. We repeat the following steps.

Step 2-1: Double-scale basis construction We first solve the following generalized eigenvalue problem:

$$C_t(X)^\top C_t(X)V = \mathfrak{N}(C_t; X)V\Lambda, \tag{4.5}$$

Then, we perform SVD.

$$C_t(Z)V_\epsilon = UDW^\top.$$

Then, we generate $G_t = C_t V_\epsilon W_\eta$ and $F_t = C_t\widetilde{V}$ (see Section 4.1 for the definition of $\widetilde{V}$, V_ϵ and W_η).

If G_t is an empty set, $\eta \leq \delta$, or the improvement of the extent of vanishing by data knotting is not significant. Then, we break the loop and go to Step 3; otherwise, go to Step 2-2.

Step 2-2: Data knotting With G^t and F^t, we update Z by the data knotting (see Section 4.2). Update η to the largest extent of vanishing of the polynomials in G^t. Then, go to Step 2.

Step 3: Decide the next step The we select the next step ("terminate", "reset", or "continue") by the following rules.

- If $|F_t| = 0$, $|G^t| \neq 0$, and for any $g \in G^t$, $\|g(Z)\| \leq \delta$, then the algorithm terminates with output $G = G^t$ and $F = F^t$.
- If $|F_t| = 0$ and $|G^t| = 0$, then we restart at Step 1 with $t = 1$, all the items are reset except Z and δ'. Update δ' with the largest extent of vanishing of the polynomials in G^t if it is smaller than δ'; otherwise, update δ' with $\delta'\gamma$.
- If $|F_t| \neq 0$, then continue to Step 1 of the next degree (t becomes $t + 1$).

Remark 10. *The output G is not necessarily a basis of the vanishing ideal for Z even when $\epsilon = 0$. This is because polynomials that are δ-vanishing on Z but ϵ-nonvanishing on X are excluded from G. This result is reasonable because the polynomials in G do not well approximate the original data. In some cases, however, we require a basis of the vanishing ideal. Such a basis can be generated by applying existing basis generation methods to small data knots Z, which is much less computationally costly than applying it to X.*

4.4 Related Work

Although the present work deals with a new problem of jointly computing vanishing polynomials and its data knots, several works are closely related to our work. The most straightforward approach to obtain exact vanishing ideal from noisy data is to summarize data by clustering-like approaches, and then compute exact vanishing polynomials. [AFT07] addressed a new task of thinning out the data points as a preprocessing for successive vanishing polynomial computation. Similar to clustering, their approach computes the empirical centroids of the input data. As thinning out and the vanishing-ideal computation are performed independently, the empirical centroids do not necessarily result in compact, lower-degree vanishing polynomials. For instance, three centroids of data points generated from a line are not necessarily linearly aligned; the exact vanishing polynomials for them become nonlinear. In contrast, our method conjointly performs the thinning out and vanishing-ideal computation in a single framework. Note that both methods by Abbott et al. (2007) and by us aim to reduce data points by summarizing nearby points, which are different from the clustering tasks that aim to group even distant points according to their categories.

Another two-step algorithm is Rational Recovery proposed by [Lim13], which first computes approximate vanishing polynomials for noisy data and then recovers the exact vanishing polynomials from them. However, their method results in noisier vanishing ideal than ours because polynomials from lower to higher degree are equally considered. Given that lower-degree polynomials are less overfitting to noisy data, our approach to pursuit exact vanishing from lower degree is more reasonable. Moreover, Rational Recovery method is only tailored for a special set of approximate vanishing polynomials (approximate border basis). Although border basis has rich theoretical foundation such as link to Gröbner basis and robustness for coefficient perturbation, other type of bases are preferred according to applications. For instance, VCA is more commonly used in machine learning, because of its rotation invariance on input data, monomial-order-free computation, and empirical performance for feature extraction in classification tasks. In contrast to Rational Recovery, our approach provides more generate framework which can be unified with standard SVD-based basis construction including approximate border basis construction methods.

Fassino (2010) proposed the Numerical Buchberger–Möler (NBM) algorithm that computes approximately vanishing polynomials for the vanishing ideal of input data X [Fas10]. NBM requires that each polynomial exactly vanishes on a set

of nearby data points X^ϵ (called an admissible perturbation) from X up to a hyperparameter ϵ. However, because the admissible perturbations can differ among the approximately vanishing polynomials, NBM does not generally output vanishing polynomials that vanish on the same points. In addition, NBM does not specify the admissible perturbations, but only checks the sufficient condition of the existence of such points.

4.5 Numerical Experiments

We provide graphical results of our method. We consider four simple datasets: (D1) 28 sample points from a unit circle, (D2) 48 sample points from a concentric circle (radii 1/2 and 1), and (D3, D4) 20 sample points from each of three blobs whose centers are $(0.6, 1.0), (-0.6, -1.0)$ and $(-0.4, 1.0)$. Sample points are perturbed by additive Gaussian noise. For D1, D2, and D3, the standard deviation is set to 10 % of the mean absolute value of data points. For D4, the standard deviation is set to 10 % of the mean absolute value of data points as in D3, but the first coordinate of the points in the blob centered by $(-0.6, -1.0)$ is perturbed by 40 % noise.

In Fig. 4.2, we provide the contour plots of output approximate vanishing polynomials for four datasets (each panel for each dataset). As shown in Fig 4.2(a), the output polynomials by SBC-$\mathfrak{n}_\mathrm{g}$ with data knotting are approximately vanishing for input data points (blue small dots) and at the same time these are tightly vanishing for the data knots (red larger dots), which are discovered along the calculation. On the other hand, without data knotting, SBC-$\mathfrak{n}_\mathrm{g}$ outputs more polynomials (Fig 4.2(b)) and these polynomials only loosely intersect with each other.

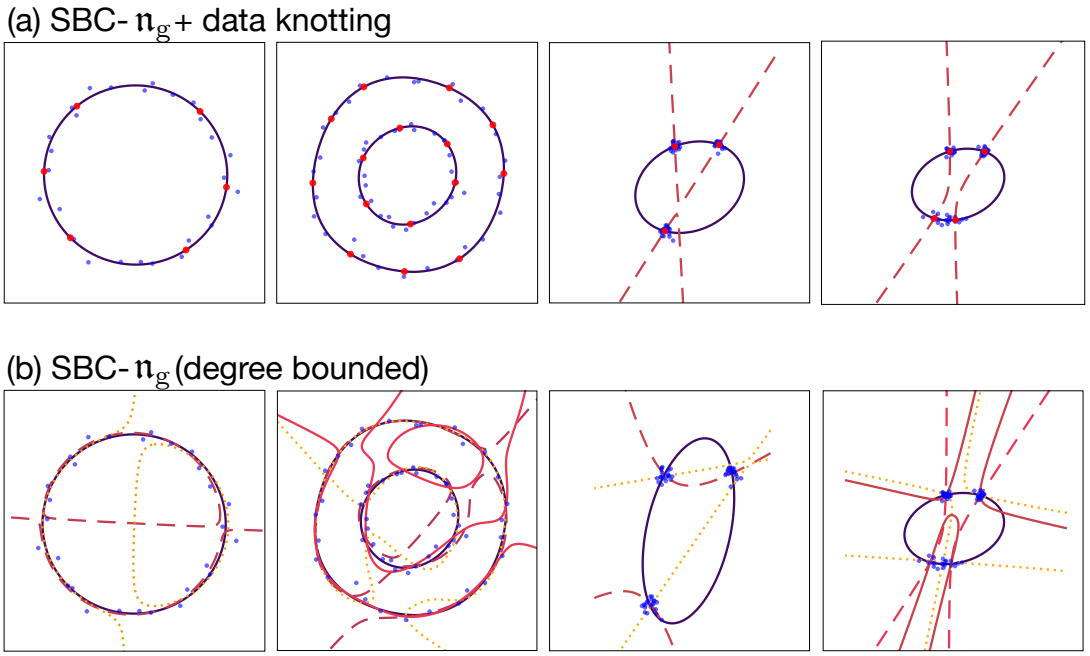

FIGURE 4.2: Contour plots of output approximate vanishing polynomials. (a) SBC-$\mathfrak{n}_g$ and data knotting result in a few approximate vanishing polynomials. These polynomials approximately vanish for data points (blue small dots) and tightly vanish for data knots (red larger dots). (b) SBC-$\mathfrak{n}_g$ without data knotting outputs many approximate vanishing polynomials, which only loosely intersect with each other. For visibility, only low-degree polynomials are shown.

Chapter 5

Conclusion

In this thesis, we addressed a task to retrieve an algebraic variety where given noisy data lie. We presented a basis construction framework, the SBC algorithm, that works efficiently and without any monomial order. Under this framework, we overcame various fundamental issues that the existing monomial-order-free basis construction algorithms encounter. In particular, the spurious vanishing problem, redundancy of the basis set, and the inconsistent behavior of the basis construction to an input transformation were discussed. The first two issues have not been efficiently resolved without using the monomial order, which is problematic in various applications. The third issue has not been discussed and achieved by any existing basis construction algorithms. The key tool in the present thesis was the gradient of polynomials at given points, which has rarely been considered in prior studies. Specifically, we showed that the gradient normalization vectors are the zero vector for unnecessary basis polynomials such as the zero polynomial and polynomials generated from basis polynomials of lower degrees. We believe that this thesis opens up a new path for monomial-order-free basis construction algorithms, which have been theoretically difficult to handle.

An important direction that is not well studied is the learning theory of the algebraic varieties with noisy data. Currently, we do not have a theory that tells us if the target algebraic varieties can be retrieved from noisy data in high probability. Intuitively, if we have enough number of points and if noise is sufficiently small, then the obtained polynomials are approximately vanishing not only for the given data but also for any point on the target algebraic variety. Assuming the noise model (e.g., the additive Gaussian noise), we might be able to formulate the relation between the extent of noise and the amount of the deviation from the target algebraic variety.

Appendix A

Minor lemmas

Here, we list several lemmas that are not used in the thesis but hopefully useful for further analysis.

Lemma 10. *During the the minimal basis construction with the orthogonal procedure, for $t \geq 1$, any polynomial h in C_{t+2}^{pre}, C_t, and F_t are orthogonal to a constant polynomial 1, i.e., $h(S)^\top 1(S) = 0$.*

Proof. Since $F_0 = \{1\}$, the orthogonalization procedure on $\mathrm{span}(C_t^{\mathrm{pre}}(S))$ always leads to $\mathrm{span}(C_t(S))$ that is orthogonal to $1(S)$. Since F_t is generated by the linear combinations of polynomials in $\mathrm{span}(C_t(S))$, $\mathrm{span}(F_t(S)) \subset \mathrm{span}(C_t(S))$[1].

We now show $\mathrm{span}(C_t^{\mathrm{pre}}(S))$ is orthogonal to $1(S)$ for $t \geq 3$. Noting that $\mathrm{span}(F_1)$ is orthogonal to $\mathrm{span}(F_t)$ for $t \geq 2$, any $pq \in C_t^{\mathrm{pre}}$, where $p \in F_1$ and $q \in F_t$, satisfies $(pq)(S)^\top 1(S) = (p(S) \circ q(S))^\top 1(S) = p(S)^\top q(S) = 0$. □

Without the orthogonalization procesure, the minial basis caonstruction algorithm can fail to find the basis set G.

Example 9. *Symbols with a tilde on its head denotes those related to the basis construction with the orthogonalization. Consider $S = \{(1,1),(1,-1),(-1,1),(-1,1)\}$. Since the mean point of S is (0,0), $F_1 = \widetilde{F}_1 = \{x, y\}$ and $G_1 = \widetilde{G}_1 = \{\}$. Thus, $C_2^{\mathrm{pre}} = \widetilde{C}_2^{\mathrm{pre}} = \{x^2, xy, y^2, yx\}$. By the orothogonalization, $\widetilde{C}_2 = \{x^2 - 1, xy, y^2 - 1, yx\}$. Note that $g = x^2 + y^2 - 1$ is a vanishing polynomial for S. However, g cannot be obtain from the linear combination of polynomials in $C_2 = C_2^{\mathrm{pre}}$, (performing SVD to $C_2(S)$ yields $G_2 = \{x^2 - y^2, 0\}$; constant factors are ignored). On the other hand, performing SVD to $\widetilde{C}_2(S)$ yields $\widetilde{G}_2 = \{x^2 + y^2 - 2, x^2 - y^2\}$ (constant factors are ignored).*

[1] To be precise, one can readily show $\mathrm{span}(F_t(S)) = \mathrm{span}(C_t(S))$.

Appendix B

Pseudo code of the SBC algorithm

Algorithm 1 Simple Basis Construction

Require: $X \subset \mathbb{R}^n, \epsilon \geq 0, m \neq 0, \mathfrak{n} : \mathcal{P}_n \to \mathbb{R}^\ell$
Ensure: G, F
$G_0, F_0 = \{\}, \{m\}$
$G, F = G_0, F_0$
for $t = 1, 2, ...$ **do**
  if $t \leq 1$ **then** ▷ `Step 1`
    $C_t = \{x_1, x_2, ..., x_n\}$
  else
    $C_t = \{pq \mid p \in F_1, q \in F_{t-1}\}$
  end if
  $C_t = C_t - FF(X)^\dagger C_t(X)$
  $\mathfrak{N}(C_t) =$ `NormlizationMatrix`$(C_t, \mathfrak{n})$
  $G_t, F_t =$ `BasisConstruction`$(C_t, \mathfrak{N}(C_t), X, \epsilon)$
  $G, F = G \cup G_t, F \cup F_t$
  if $F_t = \emptyset$ **then**
    terminate
  end if
end for

Algorithm 2 `NormalizationMatrix`

Require: $C_t, \mathfrak{n}$ ▷ $C_t = \{c_1, c_2, ..., c_{|C_t|}\}$
Ensure: $\mathfrak{N}(C_t)$
$\mathfrak{n}(C_t) = \begin{pmatrix} \mathfrak{n}(c_1) & \mathfrak{n}(c_2) & \cdots & \mathfrak{n}(c_{|C_t|}) \end{pmatrix}$
$\mathfrak{N}(C_t) = \mathfrak{n}(C_t)^\top \mathfrak{n}(C_t)$ ▷ For the unnormalized case, simply return $\mathfrak{N}(C_t) = I$.

Algorithm 3 `BasisConstruction`

Require: $C_t, \mathfrak{N}(C_t), X, \epsilon$
Ensure: G, F
$C_t(X)^\top C_t(X) V = \mathfrak{N}(C_t) V \Lambda$ ▷ `Step 2′`
▷ $\boldsymbol{v}_i$: the i^{th} column of V, λ_i: the i^{th} diagonal entry of Λ.
$G = \{C_t \boldsymbol{v}_i \mid \sqrt{\lambda_i} \geq \epsilon, i = 1, 2, ..., |C_t|\}$ ▷ `Step 3`
$F = \{C_t \boldsymbol{v}_i \mid \sqrt{\lambda_i} > \epsilon, i = 1, 2, ..., |C_t|\}$

Bibliography

[AFT07] John Abbott, Claudia Fassino, and Maria-Laura Torrente. "Thinning Out Redundant Empirical Data". In: *Mathematics in Computer Science* 1 (2007), pp. 375–392 (cit. on p. 84).

[Buc65] Bruno Buchberger. "An algorithm for finding a basis for the residue class ring of a zero-dimensional polynomial ideal". PhD thesis. University of Innsbruck, 1965 (cit. on p. 3).

[CLO92] David Cox, John Little, and Donal O'shea. *Ideals, varieties, and algorithms*. Vol. 3. Springer, 1992 (cit. on pp. 3, 4, 9, 60).

[Fan+08] Rong-En Fan et al. "LIBLINEAR: A Library for Large Linear Classification". In: *Journal of Machine Learning Research* 9 (2008), pp. 1871–1874 (cit. on p. 59).

[Fas10] Claudia Fassino. "Almost vanishing polynomials for sets of limited precision points". In: *Journal of Symbolic Computation* 45 (2010), pp. 19–37 (cit. on pp. 5, 6, 51, 84).

[FT13] Claudia Fassino and Maria-Laura Torrente. "Simple varieties for limited precision points". In: *Theoretical Computer Science* 479 (2013), pp. 174–186 (cit. on p. 51).

[Fri89] Jerome H. Friedman. "Regularized Discriminant Analysis". In: *Journal of the American Statistical Association* 84 (1989), pp. 165–175 (cit. on p. 30).

[HKP19] Amir Hashemi, Martin Kreuzer, and Samira Pourkhajouei. "Computing all border bases for ideals of points". In: *Journal of Algebra and Its Applications* 18.06 (2019), p. 1950102 (cit. on pp. 9, 51).

[Hel+09] Daniel Heldt et al. "Approximate computation of zero-dimensional polynomial ideals". In: *Journal of Symbolic Computation* 44 (2009), pp. 1566–1591 (cit. on p. 10).

[HNT16] Chenping Hou, Feiping Nie, and Dacheng Tao. "Discriminative Vanishing Component Analysis". In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI)*. AAAI Press, 2016, pp. 1666–1672 (cit. on p. 59).

[Jir95] Mats Jirstrand. *Cylindrical Algebraic Decomposition - an Introduction*. Tech. rep. 1807. Linköping University, The Institute of Technology, 1995, p. 38 (cit. on p. 3).

[KI16] Hiroshi Kera and Hitoshi Iba. "Vanishing ideal genetic programming". In: *Proceedings of the 2016 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2016, pp. 5018–5025 (cit. on p. 7).

[KKT14] Franz J Király, Martin Kreuzer, and Louis Theran. "Dual-to-kernel learning with ideals". In: *arXiv preprint arXiv:1402.0099* (2014) (cit. on pp. 10, 17, 59).

[LS04] Reinhard Laubenbacher and Brandilyn Stigler. "A computational algebra approach to the reverse engineering of gene regulatory networks". In: *Journal of Theoretical Biology* 229 (2004), pp. 523–537 (cit. on p. 7).

[Lic13] M. Lichman. *UCI Machine Learning Repository*. 2013. URL: http://archive.ics.uci.edu/ml (cit. on p. 59).

[Lim13] Jan Limbeck. "Computation of approximate border bases and applications". PhD thesis. Passau, Universität Passau, 2013 (cit. on pp. 10, 12, 13, 39, 84).

[Liv+13] Roi Livni et al. "Vanishing component analysis". In: *Proceedings of the Thirteenth International Conference on Machine Learning (ICML)*. PMLR, 2013, pp. 597–605 (cit. on pp. 7, 10, 15–17, 26, 38, 59).

[MB82] H. M. Möller and B. Buchberger. "The construction of multivariate polynomials with preassigned zeros". In: *Computer Algebra. EUROCAM 1982. Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 1982, pp. 24–31 (cit. on pp. 10, 13, 51).

[RA10] Lorenzo Robbiano and John Abbott. *Approximate Commutative Algebra*. Vol. 1. Springer, 2010 (cit. on pp. 5, 6).

[Sau07] Tomas Sauer. "Approximate varieties, approximate ideals and dimension reduction". In: *Numerical Algorithms* 45 (2007), pp. 295–313 (cit. on pp. 9, 51).

[Ste04] Hans J. Stetter. *Numerical Polynomial Algebra*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2004 (cit. on pp. 5, 6).

[VYS05] R. Vidal, Yi Ma, and S. Sastry. "Generalized principal component analysis (GPCA)". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27 (2005), pp. 1945–1959 (cit. on pp. 37, 51).

[VMS16] René Vidal, Yi Ma, and S. S. Sastry. *Generalized Principal Component Analysis*. 1st. Springer Publishing Company, Incorporated, 2016 (cit. on p. 29).

Publications

Internatial journals

1. Hiroshi Kera and Yoshihiko Hasegawa.
   "Spurious Vanishing Problem in Approximate Vanishing Ideals,"
   IEEE Access, vol. 7, pp.178961–178976, 2019

Internatial conferences (refereed)

2. Hiroshi Kera and Yoshihiko Hasegawa.
   "Gradient Boosts the Approximate Vanishing Ideal,"
   In *Proceedings of the Thirty-forth AAAI Conference on Artificial Intelligence (AAAI 2020)*, Feb. 2020 (to appear)

3. Hiroshi Kera and Yoshihiko Hasegawa.
   "Approximate Vanishing Ideal via Data Knotting,"
   In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI 2018)*, pp.3399–3406, Feb. 2018

Domestic conferences (non-refereed)

4. Hiroshi Kera and Yoshihiko Hasegawa.
   "Approximate Vanishing Ideal via Data Knotting,"
   The 17-th Forum on Information Technology 2018 (FIT2018), Sep. 2018 (Published Paper Session)

Others (refereed)

5. Hiroshi Kera and Yoshihiko Hasegawa.
   "Noise-Tolerant Algebraic Method for Reconstruction of Nonlinear Dynamical Systems,"
   Nonlinear Dynamics, vol. 85, pp.675-692, 2016

6. Hiroshi Kera and Hitoshi Iba.
   "Vanishing Ideal Genetic Programming,"

In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, pp.5018-5025, Jul. 2016