Vanishing Component Analysis

Roi Livni* ROI.LIVNI@MAIL.HUJI.AC.IL
Interdiscplinary Center for Neural Computation Edmond and Lily Safra Center for Brain Sciences, The Hebrew University of Jerusalem Givat Ram, Jerusalem 91904, Israel

David Lehavi*, Sagi Schein, Hila Nachlieli DAVID.LEHAVI, SAGI.SCHEIN, HILA.NACHLIELI@HP.COM
Hewlett-Packard Laboratories Israel Ltd. Technion City Haifa 32000 Israel

Shai Shalev-Shwartz, Amir Globerson SHAIS, GAMIR@CS.HUJI.AC.IL
Rachel and Selim Benin School of Computer Science and Engineering, The Hebrew University of Jerusalem Givat Ram, Jerusalem 91904, Israel

Abstract

The vanishing ideal of a set of points, $S \subset \mathbb{R}^n$, is the set of all polynomials that attain the value of zero on all the points in S. Such ideals can be compactly represented using a small set of polynomials known as *generators* of the ideal. Here we describe and analyze an efficient procedure that constructs a set of generators of a vanishing ideal. Our procedure is numerically stable, and can be used to find approximately vanishing polynomials. The resulting polynomials capture nonlinear structure in data, and can for example be used within supervised learning. Empirical comparison with kernel methods show that our method constructs more compact classifiers with comparable accuracy.

1. Introduction

Classification algorithms can only be as good as the features they work with. For example, in linear classification, high accuracy will only be obtained if our features are such that the classes are linearly separable. The problem of feature extraction has thus traditionally attracted considerable interest in the machine learning literature.

One conceptually simple approach to describing a set of points S is to find a set of equations that each data point $\boldsymbol{x} \in S$ should (approximately) satisfy. In other words, we seek a set of functions $f_1(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x})$ such that $f_i(\boldsymbol{x}) \approx 0$ for all i and $\boldsymbol{x} \in S$. If the set of points S belongs to a particular class (say the digit eight) then these functions may provide a succinct characterization of elements in the class. We can thus extract such functions for all classes and use them as features in classification.

Clearly the complexity of the functions $f_i(\boldsymbol{x})$ should be restricted so that the description is both compact and interpretable. A natural class which we focus on here is degree bounded polynomials. The goal of our work is to find a small set of such polynomials for a given set of points.

The set of *all* polynomials $f(\boldsymbol{x})$ that attain a value of zero on a set S is known as the vanishing ideal of S and is denoted by $I(S)$ (i.e., $f \in I(S)$ iff $f(\boldsymbol{x}) = 0 \ \forall \boldsymbol{x} \in S$). This set would qualify as a description of S. However, it contains an infinite number of polynomials and we would rather represent it with finitely many polynomials if possible. The first key observation is that if $f(\boldsymbol{x}) \in I(S)$ and $h(\boldsymbol{x})$ is any polynomial, then hf is also in $I(S)$. Thus, it is natural to ask whether there are $f_1(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x})$ such that any $g \in I(S)$ can be represented as $g(\boldsymbol{x}) = \sum_i f_i(\boldsymbol{x}) h_i(\boldsymbol{x})$ for some polynomials $h_i(\boldsymbol{x})$. Such a set of $f_i(\boldsymbol{x})$ is known as a *set of generators* of the ideal $I(S)$. Luckily, Hilbert's basis theorem (Cox et al., 2007) tells us that a finite set of generators exists for any ideal. A finite set of generators is an attractive mechanism for describing $I(S)$ since all elements in $I(S)$ can be derived from the set of generators. Thus, we turn our attention to finding such a finite set of generators, whose elements we call Vanishing Components.

Current machine learning approaches do not offer a solution to the above problem. One seemingly relevant

* These authors contributed equally.

Proceedings of the 30^{th} International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28.

approach is kernel PCA (Schölkopf et al., 1998). However, as we argue later, kernel PCA cannot be used to find vanishing components, since the kernel trick is inapplicable in this case. Linear PCA can be used to find vanishing components only in the case where these are linear, which is not expected to hold generally. The work that is closest in spirit to ours is the approximately vanishing ideal (AVI) algorithm in (Heldt et al., 2009). AVI requires an additional lexical order on the original features, and is also geared towards functions with a small number of monomials, which is often not the case.

Our vanishing component analysis (VCA) algorithm takes as input a set S and outputs a set of polynomials $V = \{f_1(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x})\}$. VCA has the following attractive properties:

- The set V generates $I(S)$.
- The algorithm is polynomial in $|S|$ and in the original dimension n. Furthermore, $f_i(\boldsymbol{x})$ can be evaluated in time polynomial in $|S|$ and n.
- The algorithm does not depend on any lexical ordering of the variables.

Thus, we achieve the goal of efficiently finding a generator set of $I(S)$ and obtain a compact description of S. In practice, due to noisy data, we might wish to find a set of polynomials that only *approximately* vanish over the the set. To address this issue, our algorithm is dependent on an ϵ-tolerance parameter, and we search for polynomials that approximately vanish on the set S. In section 7 we illustrate its use as a feature learning procedure for classification.

2. Preliminaries and Problem Setup

We begin with basic definitions.

Definition 1 (Monomials). *A function $f : \mathbb{R}^n \to \mathbb{R}$ is called a monomial if it is of the form $f(\boldsymbol{x}) = \prod_{i=1}^n x_i^{\alpha_i}$ where each α_i is a non-negative integer. We also use denote $\boldsymbol{x}^{\boldsymbol{\alpha}} = \prod_{i=1}^n x_i^{\alpha_i}$. The degree of the monomial is $\|\boldsymbol{\alpha}\|_1 = \sum_{i=1}^n \alpha_i$. We denote the set of monomials over n variables by $\mathcal{T}^n$, and the set of monomials of total degree up to d by $\mathcal{T}_d^n$.*

Definition 2 (Polynomials). *A function $f : \mathbb{R}^n \to \mathbb{R}$ is called a polynomial if it is a weighted sum of monomials. That is, $f(\boldsymbol{x}) = \sum_j \beta_j \boldsymbol{x}^{\boldsymbol{\alpha}^{(j)}}$, where each $\beta_j \in \mathbb{R}$ and each $\alpha_i^{(j)}$ is a non-negative integer. The degree of f is the maximal degree of its monomials.*

Definition 3 (The polynomial ring). *The polynomial ring in n variables over $\mathbb{R}$, denoted by $\mathbb{R}[x_1, \ldots, x_n]$, is the set of all polynomials in n variables over the reals of finite degree. The addition and multiplication operators over the ring are equivalent to addition and multiplication of functions. That is, if $h = f + g$ then for all $\boldsymbol{x}$, $h(\boldsymbol{x}) = f(\boldsymbol{x}) + g(\boldsymbol{x})$ and if $h = fg$ then for all $\boldsymbol{x}$, $h(\boldsymbol{x}) = f(\boldsymbol{x})g(\boldsymbol{x})$.*

Definition 4 (Ideal). *A set of polynomials I is an ideal if it is a sub-group with respect to addition (meaning that it is closed under addition and negation and contains the zero polynomial) and it "absorbs multiplication", meaning that for any $f \in I, g \in \mathbb{R}[x_1, \ldots, x_n]$ we have $fg \in I$.*

Definition 5 (Set of Generators). *Given an ideal I. A set of polynomials $\{f_1, \ldots, f_k\} \subseteq I$ is said to generate I, if $\forall f \in I$ there exist $g_1, \ldots, g_k \in \mathbb{R}[x_1, \ldots, x_n]$ such that $f = \sum_i g_i f_i$. In this case we denote the ideal by $I(F)$ where $F = \{f_1, \ldots, f_k\}$. Note that this should not be confused with $I(S)$.*

Definition 6 (Vanishing Ideal). *Given a set $S \subset \mathbb{R}^n$, the vanishing ideal of S is the set of polynomials that vanish on S. We denote it by $I(S)$ (it's easy to see that it's an ideal). That is, for all $\boldsymbol{x} \in S$ and $f \in I(S)$ we have $f(\boldsymbol{x}) = 0$.*

Definition 7 (Algebraic Set). *A set $V \subset \mathbb{R}^n$, is called an algebraic set if there is a finite set of polynomials $\{p_i\}_{i=1}^k$, such that V are the common roots of $\{p_i\}_{i=1}^k$.*

Our problem setup is as follows. We are given a sample S_m of points $\{\boldsymbol{x}^{(i)}\}_{i=1}^m$ where $\boldsymbol{x}^{(i)} \in \mathbb{R}^n$. Our goal is to find a set of generators of $I(S_m)$. As mentioned earlier this is desirable since it succinctly captures all polynomials that vanish on S_m.

To make the algorithm practical, we seek a method with a polynomial running time in m and n. Additionally, since real world data is noisy, we allow some tolerance by looking for polynomials that "almost" vanish on S_m.

In Section 4 we describe our Vanishing Component Analysis (VCA). Before presenting VCA, we first discuss an approach for finding a set of generators that is simple to understand but is exponential in the sample size m. We also show that the kernel trick cannot be used to overcome this difficulty.

3. A Simple but Impractical Approach

One approach to finding generators for $I(S_m)$ is to use a linear algebraic method. Assume for simplicity that we know there is a set of generators of $I(S_m)$ of maximal degree D. Now consider the set of monomials $\mathcal{T}_D^n$ and construct a matrix A of size $(m, |\mathcal{T}_D^n|)$ as follows: $A_{ij} = t_j(\boldsymbol{x}^{(i)})$, where $t_j(\boldsymbol{x})$ is the j^{th} monomial in $\mathcal{T}_D^n$.

Note that the number of columns in A is exponential in D. We now claim that the null space of A can be used to obtain a set of generators of $I(S_m)$.

Proposition 3.1. *Denote by V a set of vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k$ which are a basis of the null space of A. Namely, for all $i = 1, \ldots, k$ we have $A\boldsymbol{v}_i = \mathbf{0}$, and any vector $\boldsymbol{v}$ such that $A\boldsymbol{v} = \mathbf{0}$ can be written as a linear combination of the $\boldsymbol{v}_i$. Then the polynomials $f_i(\boldsymbol{x}) = \sum_{j=1}^{n} v_{ij} t_j(\boldsymbol{x})$ form a set of generators of $I(S_m)$.*

Proof. Clearly $f_i(\boldsymbol{x}^{(j)}) = 0$ for all sample points $\boldsymbol{x}^{(j)}$. Thus $f_i(\boldsymbol{x}) \in I(S_m)$. Next, we show that for any set of generators of $I(S_m)$ of maximal degree D, any polynomial in the set of generators can be obtained as a linear combination of f_i. Consider such a polynomial $g(\boldsymbol{x})$. By our assumption it is of max degree D, and hence can be written as a linear combination of the monomials in $\mathcal{T}_D^n$. Denote the vector of the corresponding coefficients by $\boldsymbol{z} \in \mathbb{R}^{|\mathcal{T}_d^n|}$. Then $A\boldsymbol{z} = \mathbf{0}$ and hence $\boldsymbol{z}$ is in the null space of A and is spanned by $\boldsymbol{v}_i$. Thus the polynomial $g(\boldsymbol{x})$ can be written as a linear combination of the $f_i(\boldsymbol{x})$ polynomials. Thus any polynomial in a set of generators of $I(S_m)$ can be (linearly) generated by $f_i(\boldsymbol{x})$ and we conclude that $f_i(\boldsymbol{x})$ are generators of $I(S_m)$. □

The above procedure achieves the goal of finding a set of generators of $I(S_m)$. However, it does so at a cost exponential in D, which is impractical even for modest D values. Furthermore, the value of D, the maximum degree of the generator set, may be $O(m)$.[1] Thus the cost of the above algorithm can be exponential in m. Next, we show why the standard use of the kernel trick *cannot* be used to overcome this difficulty.

3.1. Kernels Can't Help!

The kernel trick is an elegant method for avoiding working in high dimensional feature spaces explicitly. For example it can be used to perform non-linear PCA, and to find non linear polynomial separators using kernel SVM (e.g., see Scholkopf and Smola, 2001). It may seem like the kernel trick can be used to find vanishing components without explicitly calculating all monomials in $\mathcal{T}_d^n$. However, perhaps surprisingly, this is not the case, as we argue next.

We begin by recalling the kernel trick idea for the polynomial case, as used for kernel PCA (KPCA). The goal in KPCA is to perform PCA in the (exponentially large) vector of monomials in $\boldsymbol{x}$ of degree d, and find the projection on these components. To do this efficiently, one considers the kernel function $k(\boldsymbol{x}, \boldsymbol{y}) = (1 + \boldsymbol{x} \cdot \boldsymbol{y})^d$. Then it can be shown that if the principal components correspond to non-zero eigenvalues (as they always do) the projection on the j^{th} principal component is given by a polynomial of the form $\sum_i \alpha_{ij} k(\boldsymbol{x}, \boldsymbol{x}^{(i)})$, where α_{ij} are eigenvectors of the kernel matrix. Following a similar rationale one might posit that the vanishing polynomials are also of this form. However, as the following result shows, the vanishing polynomials *cannot* be expressed in this fashion.

Theorem 3.2. *Let k be a reproducing kernel and $f \in span(k(\cdot, \boldsymbol{x}^{(i)}))$ such that f vanishes on all $\boldsymbol{x}^{(i)}$. Then f is the zero function.*

Proof. Given $\boldsymbol{x}$, define $\tilde{K} := \begin{bmatrix} c & \boldsymbol{v}^T \\ \boldsymbol{v} & K \end{bmatrix}$, where $K_{i,j} := k(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)})$, $c := k(\boldsymbol{x}, \boldsymbol{x})$ and $\boldsymbol{v}_i := k(\boldsymbol{x}^{(i)}, \boldsymbol{x})$. Under these notations, we need to prove that for every $\boldsymbol{\alpha}$ in the null space of K, we have $\boldsymbol{\alpha}^T \boldsymbol{v} = 0$.

The reproducing property ensures that $\tilde{K}$ is a positive semidefinite matrix. The Schur complement of c in $\tilde{K}$ is defined as $A = K - \frac{1}{c}\boldsymbol{v}\boldsymbol{v}^T$. It is known that if $\tilde{K}$ is positive semidefinite then, $c = 0$ implies $\boldsymbol{v} = 0$, and if $c > 0$ then A must be positive semidefinite (Boyd and Vandenberghe, 2004). But for any $\boldsymbol{\alpha}$ in the null space of K we will have $|\boldsymbol{\alpha}^T \boldsymbol{v}|^2 = -c\boldsymbol{\alpha}^T A \boldsymbol{\alpha} \leq 0$. and thus $\boldsymbol{\alpha}^T \boldsymbol{v} = 0$. □

The above theorem says that we cannot use the kernel trick (i.e., use the kernel matrix instead of the explicit monomial vector) to find the vanishing components.

4. The VCA algorithm

Recall that our goal is to find a set of generators for $I(S_m)$. Since we are dealing with noisy data, it is unreasonable to seek generators that exactly vanish on S_m, and in our VCA procedure we use a tolerance parameter to allow generators to approximately vanish. In what follows, we give a step by step description of the algorithm and its rationale. The procedure itself is described in Figures 1 and 2 and its properties are analyzed in Section 5.

We can think of each polynomial f both in the usual sense, i.e. as a function from $\mathbb{R}^n$ to $\mathbb{R}$, but also as a

[1] As an example where $D = m$, consider the following: Let p be a polynomial of degree m in $\mathbb{R}$, and let $\{r_1, \ldots, r_m\}$ be its real roots. Choose some random unit vector $\boldsymbol{v}$ in $\mathbb{R}^n$ and let $S_m = \{r_1\boldsymbol{v}, r_2\boldsymbol{v}, \ldots, r_m\boldsymbol{v}\}$. $I(S_m)$ is the ideal generated by the polynomials $\{p(\boldsymbol{v}^T\boldsymbol{x}), \boldsymbol{u}_1^T\boldsymbol{x}, \ldots, \boldsymbol{u}_{n-1}^T\boldsymbol{x}\}$, where $\boldsymbol{v} \perp \boldsymbol{u}_1, \ldots, \boldsymbol{u}_{n-1}$. There is no alternative generator of $I(S_m)$ that does not contain a polynomial of total degree at least m.

vector in $\mathbb{R}^m$ containing the evaluations of f on the sample S_m, namely, $f(S_m) = \left[f(\boldsymbol{x}^{(1)}), \ldots, f(\boldsymbol{x}^{(m)})\right] \in \mathbb{R}^m$. A polynomial f vanishes on S_m if and only if $f(S_m) = \mathbf{0}$.

To motivate the construction, let us first recall the case of the most simple polynomials — linear functions. Suppose we would like to find a set V of linear functions such that for each $f \in V$ and $\boldsymbol{x}^{(i)} \in S_m$ we have $f(\boldsymbol{x}^{(i)}) = 0$. Each linear function is described by a vector $\boldsymbol{\beta} \in \mathbb{R}^{n+1}$, such that $f(\boldsymbol{x}) = \beta_0 + \sum_{j=1}^n \beta_j x_j$. We can rewrite the linear function as a combination of base polynomials. Indeed, let f_0 be the constant polynomial, $f_0(\boldsymbol{x}) = 1/\sqrt{m}$ for all $\boldsymbol{x}$. Let $C_1 = \{f_1, \ldots, f_n\}$ be a set of polynomials, where for all i, $f_i(\boldsymbol{x}) = x_i$. Now, we can rewrite any linear function as a linear combination of polynomials from $C_1 \cup \{f_0\}$. That is, each linear function is of the form:

$$f(\boldsymbol{x}) = \beta_0 + \sum_{i=1}^n \beta_i x_i = \sum_{i=0}^n \beta_i f_i(\boldsymbol{x}) \ .$$

It follows that for any such polynomial we have $f(S_m) = \sum_{i=0}^n \beta_i f_i(S_m)$. Therefore, a linear function vanishes on S_m if and only if $f(S_m) = \mathbf{0} \in \mathbb{R}^m$. This amounts to requiring that the vector $\boldsymbol{\beta}$ would be in the null space of the $m \times (n+1)$ matrix $A_1 = [f_0(S_m), \ldots, f_n(S_m)]$.

To find the null space of A_1, we can follow the Gram-Schmidt procedure. As we show later, using the Singular Value Decomposition is preferable, since it provides us with a stable method for finding an approximate null space. However, for the sake of clarity, let us first describe the Gram-Schmidt approach.

We maintain two sets: V for the vanishing polynomials and F for the non-vanishing polynomials. We use the notation $F(S_m) = \{f(S_m) : f \in F\} \subset \mathbb{R}^m$ to denote the vectors in $\mathbb{R}^m$ corresponding to evaluations of non-vanishing polynomials in F on S_m. We will construct F such that $F(S_m)$ is a set of orthonormal vectors in $\mathbb{R}^m$.

Since f_0 is clearly non-vanishing we can initialize $F = \{f_0\}$ and $V = \emptyset$. Now, at round t, consider the remainder of $f_t(S_m)$ after projecting on the orthonormal set $F(S_m)$. That is, $r_t(S_m) = f_t(S_m) - \sum_{f \in F} \langle f_t(S_m), f(S_m) \rangle \, f(S_m)$. Note that $r_t(S_m)$ is the evaluation of the polynomial $r_t(\boldsymbol{x}) = f_t(\boldsymbol{x}) - \sum_{f \in F} \langle f_t(S_m), f(S_m) \rangle \, f(\boldsymbol{x})$, on S_m.

Now, if $r_t(S_m)$ is the zero vector, then r_t is vanishing on S_m, so we update: $V \leftarrow V \cup \{r_t\}$. Otherwise, we update $F \leftarrow F \cup \{r_t / \|r_t(S_m)\|\}$, where the normalization ensures that all the vectors in $F(S_m)$ are of unit norm. At the end of this process, F contains a set of linear polynomials which are non-vanishing on S_m and V contains a set of linear polynomials that vanish on S_m. Furthermore, $F(S_m)$ is an orthonormal basis of the range of A_1. Let us call F_1 and V_1 the values of F and V after dealing with polynomials of degree 1.

Next, consider polynomials of degree 2. Consider the set of polynomials $C_2 = \{f_{i,j}\}_{i,j=1}^n$, where for all i, j, $f_{i,j}(\boldsymbol{x}) = x_i x_j$. Each polynomial of degree 2 takes the form $f(\boldsymbol{x}) = \sum_{i=0}^n \beta_i f_i(\boldsymbol{x}) + \sum_{i,j=1}^n \beta_{i,j} f_{i,j}(\boldsymbol{x})$.

As before, we can find vanishing 2^{nd} order polynomials via the null space of the matrix: $A_2 = [A_1, f_{1,1}(S_m), \ldots, f_{n,n}(S_m)]$. To find the null space of the matrix A_2, we could simply continue the Gram-Schmidt procedure we have already performed for the columns of A_1. However, we now need to consider n^2 columns, and as the degree goes up the number of columns increases exponentially. To overcome this obstacle, we rely on the underlying structure of the vanishing ideal, and in particular its absorbedness property.

Take some f of degree 2. Then, $f = \sum_i g_i h_i$, where g_i, h_i are all of degree at most 1. Without loss of generality, assume that for $i \leq i_1$ we have that both g_i and h_i are non vanishing on S_m, and that for $i > i_1$ either g_i or h_i vanishes. It follows that for all $i > i_1$ we have that the polynomial $g_i h_i$ vanishes. Therefore, the polynomial $\hat{f} = \sum_{i \leq i_1} g_i h_i$ satisfies $\hat{f}(S_m) = f(S_m)$. In other words, f vanishes on S_m if and only if $\hat{f}$ vanishes on S_m. Since, by our construction, $f - \hat{f}$ is generated by V, it suffices to deal with $\hat{f}(S_m)$. Using our construction of F_1, we know that for all $i \leq i_1$, both g_i and h_i are in the span of F_1. Denoting $F_1 = \{p_1, \ldots, p_k\}$, we can write

$$\hat{f} = \sum_{i \leq i_1} \sum_j \alpha_j^i p_j \sum_s \beta_s^i p_s = \sum_{j,s} p_j p_s \left(\sum_{i \leq i_1} \alpha_j^i \beta_s^i \right)$$

It follows that $\hat{f}$ is in the span of $F_1 \times F_1$ and thus to construct F_2 and V_2, it suffices to find the null space and range on the set of candidate polynomials in $F_1 \times F_1$. Formally, let us redefine C_2 to be the set

$$C_2 = \{f = gh : g, h \in F_1\} \ ,$$

and let $C_2(S_m) = \{f(S_m) : f \in C_2\} \subset \mathbb{R}^m$ be the corresponding set of vectors obtained by evaluating all polynomials in C_2 on S_m. We will construct F_2 and V_2 by continuing the Gram-Schmidt process on the candidate vectors in $C_2(S_m)$, as follows. Choose a polynomial $f \in C_2$ and let $f(S_m)$ be the corresponding vector in $C_2(S_m)$. The remainder of f after projecting it on the current set F is the polynomial r defined by $r(\boldsymbol{x}) = f(\boldsymbol{x}) - \sum_{g \in F} \langle f(S_m), g(S_m) \rangle g(\boldsymbol{x})$. If

VCA

parameters:
Tolerance ϵ ; "FindRangeNull" procedure
initialize:
$F = \{f(\cdot) = 1/\sqrt{m}\}$, $V = \emptyset$
$C_1 = \{f_1, \ldots, f_n\}$ where $f_i(\boldsymbol{x}) = x_i$
$(F_1, V_1) = \text{FindRangeNull}(F, C_1, S_m, \epsilon)$
$F = F \cup F_1,\ V = V \cup V_1$
for $t = 2, 3 \ldots$
$C_t = \{gh : g \in F_{t-1}, h \in F_1\}$
if $C_t = \emptyset$
break
$(F_t, V_t) = \text{FindRangeNull}(F, C_t, S_m, \epsilon)$
$F = F \cup F_t,\ V = V \cup V_t$
output: F, V

Figure 1. Our Vanishing Component Analysis (VCA) algorithm for finding a generator set for $I(S_m)$.

$r(S_m) = \mathbf{0}$, we add it to V_2 and to V. Otherwise, we add $r/\|r(S_m)\|$ to F_2 and to F.

This process continues to higher degrees. At iteration t, we construct the set of candidate polynomials to be $F_{t-1} \times F_1$. Then, we construct F_t and V_t (and update F and V accordingly) by performing the Gram-Schmidt procedure on this set of candidate polynomials.

The VCA procedure is described in Figures 1 and 2. We make one crucial modification to the above description. When constructing the new polynomials in F_t and V_t out of the candidates in C_t, we use the Singular Valued Decomposition procedure to find an *approximate* null space instead of performing the Gram-Schmidt procedure. This is obtained by calling a sub-procedure $(F_t, V_t) = \text{FindRangeNull}(F, C_t, S_m, \epsilon)$, which receives the current set F, a list of candidates, C_t, the set of examples S_m, and a tolerance parameter ϵ. When $\epsilon = 0$, the procedure can be implemented by initializing a Gram-Schmidt procedure with the orthonormal basis $F(S_m)$, and then continuing performing Gram-Schmidt orthonormalization on the candidate vectors $f(S_m)$ with $f \in C_t$. When $\epsilon > 0$, we rely on the SVD procedure, as described below. As we will formally prove in the next section, if one sets $\epsilon = 0$, then VCA is guaranteed to construct a set of generators of the vanishing ideal.

We next describe how to implement the FindRangeNull procedure. It is known that finding an orthonormal basis using the Gram-Schmidt method can be numerically instable, and a preferred approach is by using the Singular Value Decomposition (SVD) approach.

FindRangeNull (using SVD)

input: F, C, S_m, ϵ
denote $k = |C|$ and $C = \{f_1, \ldots, f_k\}$
for $t = 1, \ldots, k$
let $\tilde{f}_i = f_i - \sum_{g \in F} \langle f_i(S_m), g(S_m) \rangle g$
let $A = [\tilde{f}_1(S_m), \ldots, \tilde{f}_k(S_m)] \in \mathbb{R}^{m,k}$
decompose $A = LDU^\top$ using SVD
for $i = 1, \ldots, k$
let $g_i = \sum_{j=1}^{k} U_{j,i} \tilde{f}_j$
output:
$F_1 = \{g_i / \|g_i(S_m)\| : D_{i,j} > \epsilon\}$
$V_1 = \{g_i : D_{i,j} \leq \epsilon\}$

Figure 2. The implementation of the FindRangeNull function in Figure 1.

First, given a candidate set $C = \{f_1, \ldots, f_k\}$, and a current set of polynomials F, we calculate $\tilde{f}_1, \ldots, \tilde{f}_k$ to be the remainder of the polynomials in C after projecting them on F. Next, we use SVD to find the (approximate) range and null space of the matrix $A = [\tilde{f}_1(S_m), \ldots, \tilde{f}_k(S_m)]$, by calculating the right singular vectors of A. Finally, we define the corresponding polynomials based on the right singular vectors.

5. Analysis

In this section we analyze the efficiency and correctness of the VCA procedure. We use the notation $F^t = \cup_{i \leq t} F_t$ and $V^t = \cup_{i \leq t} V_t$. Our first theorem addresses with the efficiency of the VCA procedure.

Theorem 5.1. *If VCA is run with any ϵ then:*

1. *VCA stops after at most $t \leq m + 1$ iterations.*
2. *All polynomials in V, F are of degree at most m.*
3. *$|F| \leq m$ and $|V| \leq |F|^2 \cdot \min\{|F|, n\}$.*
4. *It is possible to evaluate all the polynomials in F in time $O(|F|^2)$, and polynomials in V in tim $O(|F|^2 + |F| \cdot |V|)$.*

Proof.

1. By the stopping rule of VCA, it will terminate at round t whenever F_{t-1} is empty (since this will imply $C_t = \emptyset$). Therefore, if we didn't stop at round t, then $|F^t| > t - 1$, and therefore we also have that $|F^t(S_m)| \geq t - 1$. On the other hand, by construction, $F^t(S_m)$ is a set of orthonormal vectors in $\mathbb{R}^m$, hence its size is at most m. It follows that $t - 1 \leq m$.

2. It follows immediately that all the polynomials in V and F are of degree at most m.

3. We already argued that $|F| \leq m$. Also, we only add polynomials to V in the first $|F|$ iterations. Furthermore, whenever we add a polynomial to V, it is constructed from a set of candidates, $C_t = F_{t-1} \times F_1$, whose size is at most $|F| \cdot \min\{|F|, n\}$. Therefore, $|V| \leq |F|^2 \cdot \min\{|F|, n\}$.

4. Let us enumerate the polynomials in $F = \{g_1, \ldots, g_{|F|}\}$ according to the order in which they were inserted into F. Since g_1 is the constant polynomial, it can be evaluated in time $O(1)$. Suppose we have evaluated $g_1, \ldots, g_{r-1}$. By construction, g_r can be written as a product of two polynomials from $g_1, \ldots, g_{r-1}$ minus a linear combination of $g_1, \ldots, g_{r-1}$. Therefore, it can be evaluated in time $O(r)$. It follows that we can evaluate all the polynomials in F in time $O(|F|^2)$. A similar argument shows that once we evaluated all polynomials in F, we can evaluate each polynomial in V in $O(|F|)$. □

We next prove that VCA finds a generator of $I(S_m)$.

Theorem 5.2. *If VCA is run with $\epsilon = 0$ its output satisfies:*

1. *V generates $I(S_m)$.*
2. *Any polynomial f can be written as $f = g + h$ where $g \in span(F)$ and $h \in I(S_m)$.*

The proof of the theorem relies on the following lemma.

Lemma 5.3. *If VCA is run with $\epsilon = 0$, then for all t, any polynomial f of degree at most t can be written as $f = g + h$ where $g \in span(F^t)$ and $h \in I(V^t)$.*

Proof. We prove this claim by induction on t. The base of the induction, i.e. $t = 1$, follows by standard results from linear algebra. So, from now on we assume that $t > 1$.

By our inductive assumption, for each $i < t$, any polynomial of degree at most i can be written as $f = g + h$ where $g \in \text{span}(F^i)$ and $h \in I(V^i)$. Now, consider a polynomial f of degree t. Clearly, if each monomial of f can be decomposed to a sum of a vector from $\text{span}(F^t)$ and a vector from $I(V^t)$, then the claim holds for t as well. Furthermore, by our inductive assumption, any monomial of f whose degree is strictly smaller than t can be decomposed as required. Therefore, from now on we can assume that f is a single monomial of degree exactly t, which we can write $f = pq$ where p is of degree $t-1$ and q is of degree 1.

Using our inductive assumption, we can write $p = p_F + p_V$ where $p_F \in \text{span}(F^{t-1})$ and $p_V \in I(V^{t-1})$. Similarly, $q = q_F + q_V$ where $q_F \in \text{span}(F^1)$ and $q_V \in I(V^1)$. Let us further rewrite $p_F = p_{F^{t-2}} + p_{F_{t-1}}$, where $p_{F^{t-2}} \in \text{span}(F^{t-2})$ and $p_{F_{t-1}} \in \text{span}(F_{t-1})$. Similarly, $q_F = q_{F^0} + q_{F_1}$, where $q_{F^0} \in \text{span}(F^0)$ and $q_{F_1} \in \text{span}(F_1)$. Therefore,

$$\begin{aligned} f = pq &= (p_{F^{t-2}} + p_{F_{t-1}} + p_V)(q_{F^0} + q_{F_1} + q_V) \\ &= p_{F_{t-1}} q_{F_1} + \underbrace{p_V q + p q_V}_{\in I(V^t)} + \underbrace{p_{F^{t-2}} q + p q_{F^0}}_{\text{of degree } < t} . \end{aligned}$$

Finally, since $p_{F_{t-1}} q_{F_1} \in \text{span}(F_{t-1} \times F_1)$, by our construction, it can be written as $a_F + a_V$ where $a_F \in \text{span}(F_t)$ and $a_V \in \text{span}(V_t)$, which concludes our inductive argument. □

Proof of Theorem 5.2. Equipped with Lemma 5.3, we only need to show that V generates the vanishing ideal of S_m. Take any vanishing polynomial f. By Lemma 5.3, we can rewrite $f = g + h$ where $g \in \text{span}(F)$ and $h \in I(V)$. Therefore, $\mathbf{0} = f(S_m) = g(S_m) + h(S_m) = g(S_m)$. Let us write $F = \{f_1, \ldots, f_k\}$. Since $g \in \text{span}(F)$ we can write $g = \sum_i \alpha_i f_i$. Therefore, $\mathbf{0} = g(S_m) = \sum_i \alpha_i f_i(S_m)$. But, by the construction of F, the set of vectors $\{f_1(S_m), \ldots, f_k(S_m)\}$ is orthonormal, hence all the α_i must be zero. It follows that g is the zero polynomial, thus $f = h \in I(V)$, which concludes our proof of the first part. The second part of Theorem 5.2 follows immediately from the first part which states that $I(V)$ is the vanishing ideal of S_m and Lemma 5.3. □

6. Related work

Finding a set of generators for an ideal is a classical problem in algebraic geometry. A key breakthrough in this field has been the introduction of Gröbner bases by Buchberger (2006). A Gröbner base is a set of generators of an ideal with particular properties that allow operations such as polynomial division with unique remainders. Later Möller and Buchberger also presented a method for finding a Gröbner base for the vanishing ideal $I(S_m)$ as we do here (Möller and Buchberger, 1982). However, their goal was to find a base that vanishes exactly at these points which is of little practical use.

Approximately vanishing ideals have been much less studied. The most relevant work on approximately vanishing ideals is the AVI algorithm (Heldt et al., 2009). AVI constructs an approximate border base for $I(S_m)$. A border base is also a generator set for $I(S_m)$, but with different properties from a Gröbner one. Unlike our approach, the AVI algorithm requires a lexical

ordering on the variables, and its output depends on this ordering . This is clearly undesirable since there is typically no a priori reason to order the variables in any way. Furthermore, AVI adds only monomial terms to the non-vanishing components F. Thus it will only generate compact descriptions of V when V contains few monomials. In contradistinction, VCA expands F using F_1, which can be non-sparse in the variables, and hence the resulting polynomials are non-sparse. Finally, AVI often generates redundant polynomials. For example, say we have two variables (x_1, x_2) and $x_1 = 0$ for all sample points. AVI will find the following vanishing components $x_1, x_1x_2, x_1x_2^2, \ldots, x_1x_2^{m-1}$, whereas VCA will find only the first one. In Section 8 we provide an example illustrating that these shortcomings of AVI result in superior generalization performance of VCA.

7. Application to Classification

In this section we show how VCA can be used as a feature learning procedure for classification. Consider a classification problem, where the goal is to learn a mapping from $\mathbb{R}^n$ to $\{1, \ldots, k\}$. To construct features for this task, we run VCA on the training sample for each class. The output of VCA is a set of polynomials which generate the vanishing ideal of each class. Let $\{p_1^l(\boldsymbol{x}), \ldots, p_{n_l}^l(\boldsymbol{x})\}$ be a set of generators of the vanishing ideal for class l. Then, for any training example in class l we have that $|p_j^l(\boldsymbol{x})|$ should be close to zero. In contrast, for points in other classes, we expect at least some of the polynomials not to vanish. If this is indeed the case, we can represent each point $\boldsymbol{x}$ using features of the form $|p_j^l(\boldsymbol{x})|$. As we formalize below, if the points of each class belong to different algebraic sets, then in the representation by the aforementioned features, the data becomes linearly separable.

Theorem 7.1. *Let $S = \{(x_i, y_i)\}_{i=1}^m$, be a set of labeled examples. Let S^l be the set of examples with $y_i = l$. Assume there are algebraic sets $\{U^l\}_{l=1}^k$ whose intersection is empty in pairs such that $S^l \subseteq U^l$. For each l, let $\{p_1^l(\boldsymbol{x}), \ldots, p_{n_l}^l(\boldsymbol{x})\}$ be a set of generators of the ideal $I(U^l)$. Then, S is linearly separable in the feature space:*

$$\boldsymbol{x} \mapsto (|p_1^1(\boldsymbol{x})|, \ldots, |p_{n_k}^k(\boldsymbol{x})|). \tag{1}$$

Proof. Choose some point $\boldsymbol{x}$, that belongs to class l, and hence $\boldsymbol{x} \in U^l$. Assume that $\boldsymbol{x}$ vanishes on all the polynomials in the set of generators of $I(U^j)$, for some $j \neq l$. Since U^j is defined as the common roots of some set of polynomials $\{h_i\}$ who are all in $I(U^j)$, it follows immediately that $\boldsymbol{x}$ is a common root of the $\{h_i\}$'s, hence $\boldsymbol{x} \in U^j$, but this contradicts the fact that $U^l \cap U^j = \emptyset$. Hence, for any j, there must be some p_i^j such that $|p_i^j(\boldsymbol{x})| > \epsilon$. On the other hand, for all i, $|p_i^l(\boldsymbol{x})| < \epsilon$. Therefore, the linear classifier that puts positive weights on the features corresponding to class l, and puts negative weights on the rest of the features, must separate between class l and the rest of the classes, which concludes our proof. □

Thus, we use VCA in classification as follows: calculate the polynomials p_j^i above and use the feature map in Equation 1 to generate a new training dataset from S. Train a linear classifier on the new dataset. Since we are doing linear classification we can use ℓ_1 regularization which has the added benefit of choosing a small number of features. Taking U^l to be the algebraic set S^l, theorem 7.1 states that the set S is linearly separable is the new feature space. In fact, it is enough to choose those polynomials that generate algebraic sets U^l whose intersection is empty.

8. Experiments

The following experiments use VCA as a feature extraction method, and use it in classification as described in Section 7 (we use VCA to refer to the overall classification approach). Linear classification (for VCA and Kernel SVM) is done using the LIBSVM (Chang and Lin, 2011) and LIBLINEAR (Fan et al., 2008) packages.

We compared the VCA approach to the popular Kernel SVM classification (KSVM) method with a polynomial kernel (Scholkopf and Smola, 2001), and to AVI feature extraction. In the experiments in Section 8.2 we measured both the accuracy of the learned classifier as well as the test runtime. We evaluate runtime using the number of operations required for prediction. For KSVM this corresponds to the number of support vectors. For VCA based classification, it is the number of operations needed to compute the vanishing components. Hyperparameters for all algorithms were tuned using cross validation. For KSVM the parameters were the polynomial degree, and the regularization coefficient. For VCA, it was the tolerance ϵ and the overall number of components, as well as the regularization coefficient for ℓ_1 regularized learning.

8.1. Toy Example

We begin with a toy example where the two classes correspond to algebraic manifolds and hence VCA is expected to work well. The first class is shaped as a flattened sphere, corresponding to the polynomial: $x_1^2 + 0.01x_2^2 + x_3^2 = 1$. The second is a cylinder, corresponding to the polynomial $x_1^2 + x_3^2 = 1.4$. The co-

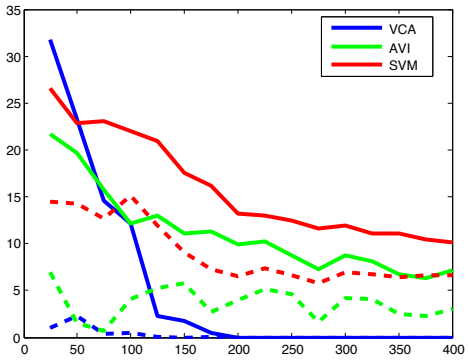

Figure 3. Toy dataset: The error percentage is depicted as a function of the sample size (solid lines are test errors, dashed lines are training errors).

efficients are needed to make the manifolds separable. We added Gaussian noise with $\sigma = 0.1$ to both sets, and embedded them in $\mathbb{R}^{10}$ using a random matrix with correlated columns. This results in two sets that are approximately vanishing on polynomials that are highly non sparse in their monomial representation. We compared VCA to KSVM and to feature extraction using AVI.

AVI chooses at each iteration a set of monomials that are non vanishing. These are analogous to the set F_d constructed by VCA. In AVI, the monomials are chosen by a lexicographical order (here we used *DegRevLex*). These monomials are then used to create new features and to represent the vanishing polynomials. When linear transformations are applied to the data, the representation of the vanishing polynomials in their monomial representation becomes less stable. This, in turn, affects the numerical stability of the algorithm. VCA on the other hand, represents each degree by the principle components of the former degrees, and this adds numerical stability even under linear embeddings.

Figure 3 shows test accuracy as a function of the training sample size. Since VCA finds the vanishing polynomial for each class, it can separate the data with only two features, and thus has a much better training curve than KSVM. VCA also outperforms AVI for most data sizes due to the stability reasons mentioned above.

8.2. Real Datasets

We next turn to experiments on real data of digit and character recognition tasks (obtained from the LIBSVM website). As mentioned earlier, we compare both the accuracy and the test runtime of the KSVM and VCA. Results in Table 1 show that while the accuracy of the methods is comparable, the runtime of VCA is up to two orders of magnitude better. This is a result of the compact representation that VCA achieves.

Data Set	Error Rate		Test Runtime	
	VCA	KSVM	VCA	KSVM
Pendigits (5996)	0.42	0.42	2.8e+003	9.6e+003
Letter (12000)	4.8	4.3	1.1e+003	7e+004
USPS (5833)	1.5	1.4	2.6e+003	3.8e+005
MNist (48000)	2.2	2	4e+03	3.1e+06

Table 1. Error rate in (%) and test runtime (in number of operations) for VCA and KernelSVM (training size in brackets). Results are averaged over 10 random 80%/20% train/test partitions.

9. Discussion

Algebraic geometry is a deep and fascinating field in mathematics, which deals with the structure of polynomial equations. Recently, methods in algebraic geometry and commutative algebra have been applied to study various problems in statistics. Specifically the field of "algebraic statistics" is concerned with statistical models that can be described via polynomials. For an introduction on the subject see for example (Gibilisco et al., 2009; Drton et al., 2008; Watanabe, 2009).

In the machine learning literature, Király et al. (2012) proposes a method for approximating an arbitrary system of polynomial equations by a system of a particular type. This method was applied to the problem of finding a linear projection that makes a set of random variable identically distributed.

However, to date, algebraic geometry has seen surprisingly few applications to the problem of classification and generative models. One reason is that many algebraic results and methods address *noise free* scenarios (e.g., what is the polynomial that *exactly* vanishes on a set of points). While there has been some recent interest in approximation approaches, these have not been applied to learning. Here we present an approach that is motivated by the notion of a polynomial ideal as a mechanism for compact description of a set of points. We show how to find a compact description of such an ideal, via a method that is also computationally stable. We believe such approaches have considerable potentials across machine learning, and could benefit from the deep known results in algebraic geometry, facilitating a synergistic interaction between the fields.

Acknowledgments: This research is supported by the HP Labs Innovation Research Program.

References

S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Bruno Buchberger. Bruno Buchberger's PhD thesis 1965: An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal. *Journal of Symbolic Computation*, 41 (3-4):475 – 511, 2006.

Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2: 27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

D.A. Cox, J. Little, and D. O'Shea. *Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra*, volume 10. Springer, 2007.

M. Drton, B. Sturmfels, and S. Sullivant. *Lectures on algebraic statistics*. Birkhauser, 2008.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874, June 2008. ISSN 1532-4435. URL `http://dl.acm.org/citation.cfm?id=1390681.1442794`.

P. Gibilisco, E. Riccomagno, M.P. Rogantin, and H.P. Wynn. *Algebraic and geometric methods in statistics*. Cambridge University Press, 2009.

Daniel Heldt, Martin Kreuzer, Sebastian Pokutta, and Hennie Poulisse. Approximate computation of zero-dimensional polynomial ideals. *J. Symb. Comput.*, 44(11):1566–1591, 2009.

F.J. Király, P. Buenau, J.S. Müller, D.A.J. Blythe, F. Meinecke, and K.R. Müller. Regression for sets of polynomial equations. *JMLR Workshop and Conference Proceedings*, 22:628–637, 2012.

H Möller and B Buchberger. The construction of multivariate polynomials with preassigned zeros. *Computer Algebra*, pages 24–31, 1982.

B. Schölkopf, A. Smola, and K.R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

Bernhard Scholkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001. ISBN 0262194759.

Sumio Watanabe. *Algebraic Geometry and Statistical Learning Theory*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, United Kingdom, 2009.