Conditional Gradients for the Approximately Vanishing Ideal

Elias Wirth
TU Berlin

Sebastian Pokutta
TU Berlin

Abstract

The vanishing ideal of a set of points $X \subseteq \mathbb{R}^n$ is the set of polynomials that evaluate to 0 over all points $\mathbf{x} \in X$ and admits an efficient representation by a finite set of polynomials called generators. To accommodate the noise in the data set, we introduce the *Conditional Gradients Approximately Vanishing Ideal algorithm* (`CGAVI`) for the construction of the set of generators of the approximately vanishing ideal. The constructed set of generators captures polynomial structures in data and gives rise to a feature map that can, for example, be used in combination with a linear classifier for supervised learning. In `CGAVI`, we construct the set of generators by solving specific instances of (constrained) convex optimization problems with the *Pairwise Frank-Wolfe algorithm* (`PFW`). Among other things, the constructed generators inherit the LASSO generalization bound and not only vanish on the training but also on out-sample data. Moreover, `CGAVI` admits a compact representation of the approximately vanishing ideal by constructing few generators with sparse coefficient vectors.

1 Introduction

The accuracy of classification algorithms relies on the quality of the available features. Naturally, feature transformations are an important area of research in the field of machine learning. In this paper, we focus on feature transformations for a subsequently applied linear kernel *Support Vector Machine* (`SVM`) (Suykens and Vandewalle, 1999), an algorithm that achieves high accuracy only if the features are such that the different classes are linearly separable.

Proceedings of the 25[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151.

Our approach is based on the idea that a set $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subseteq \mathbb{R}^n$ can be succinctly described by the set of algebraic equations satisfied by each point $\mathbf{x} \in X$. Put differently, we seek polynomials $g_1, \ldots, g_k \in \mathcal{P}$ such that $g_i(\mathbf{x}) = 0$ for all $\mathbf{x} \in X$ and $i \in \{1, \ldots, k\}$, where $\mathcal{P}$ is the polynomial ring in n variables. An obvious candidate for a succinct description of X is the *vanishing ideal*[1] of X,

$$\mathcal{I}_X = \{f \in \mathcal{P} \mid f(\mathbf{x}) = 0 \text{ for all } \mathbf{x} \in X\}.$$

Even though $\mathcal{I}_X$ contains an infinite number of vanishing polynomials, by Hilbert's basis theorem (Cox et al., 2013), there exists a finite number of *generators* of $\mathcal{I}_X$, $g_1, \ldots, g_k \in \mathcal{I}_X$ with $k \in \mathbb{N}$, such that for any $f \in \mathcal{I}_X$, there exist $h_1, \ldots, h_k \in \mathcal{P}$ such that

$$f = \sum_{i=1}^{k} g_i h_i.$$

The generators have all the points in X as a common root, implying that the set of generators captures the nonlinear structure of the data set X and represents the vanishing ideal $\mathcal{I}_X$. The construction of this set of generators has received a lot of attention (Heldt et al., 2009; Livni et al., 2013; Limbeck, 2013; Iraji and Chitsaz, 2017).

The set of generators can be used to transform the features of X such that linear separability is achieved in the transformed feature space. For example, consider the binary classification task of deciding whether a tumor is cancerous or benign and suppose that the sets $X_c \subseteq X$ and $X_b \subseteq X$ correspond to data points of cancerous and benign tumors, respectively. Then, generators $g_1, \ldots, g_k$ of $\mathcal{I}_{X_c}$ provide a succinct characterization of elements in the class of cancerous tumors and vanish over X_c. For points $\mathbf{x} \in X_b$, however, we expect that for some $i \in \{1, \ldots, k\}$, the polynomial g_i does not vanish over $\mathbf{x}$, that is, $g_i(\mathbf{x}) \neq 0$. Similarly, we construct a second set of generators of $\mathcal{I}_{X_b}$, representing the benign tumors. Then, evaluating both

[1] A set of polynomials $\mathcal{I} \subseteq \mathcal{P}$ is called an *ideal* if it is a subgroup with respect to addition and for $f \in \mathcal{I}$ and $g \in \mathcal{P}$, we have $f \cdot g \in \mathcal{I}$.

sets of generators over the entire data set $X = X_c \dot{\cup} X_b$ represents the data set X mapped into a new feature space in which the two classes are (hopefully) linearly separable. We further develop this idea in Section 7.

To accommodate the noise in the data set, we construct generators g of the *approximately vanishing ideal* instead of the vanishing ideal, where the approximately vanishing ideal contains polynomials that almost vanish over X, that is, polynomials g such that $g(\mathbf{x}) \approx 0$ instead of $g(\mathbf{x}) = 0$ for all $\mathbf{x} \in X$.

In this paper, we introduce the *Conditional Gradients Approximately Vanishing Ideal algorithm* (`CGAVI`), which takes as input a set of points $X \subseteq \mathbb{R}^n$ and constructs a set of generators of the approximately vanishing ideal over X. The novelty of our approach is that `CGAVI` constructs the generators of the approximately vanishing ideal using an oracle (which solves a (constrained) convex optimization problem), in our case, via the *Pairwise Frank-Wolfe algorithm* (`PFW`) (Guélat and Marcotte, 1986; Lacoste-Julien and Jaggi, 2015), whereas related methods such as the *Approximate Vanishing Ideal algorithm* (`AVI`) (Heldt et al., 2009) and *Vanishing Component Analysis* (`VCA`) (Livni et al., 2013) rely on *Singular Value Decompositions* (`SVD`) to construct generators. Our approach admits the following properties:

1. **Generalization bounds:** To the best of our knowledge, `CGAVI` is the first algorithm that constructs generators that provably vanish approximately on out-sample data.
2. **Compact transformation:** `CGAVI` constructs a small number of generators with sparse coefficient vectors.
3. **Blueprint:** For the implementation of the oracle in `CGAVI`, `PFW` can be replaced by any other solver of (constrained) convex optimization problems. Thus, our paper gives rise to an entire family of procedures for the construction of generators of the approximately vanishing ideal.
4. **Empirical results:** For the combined approach of constructing generators to transform features for a linear kernel `SVM`, generators constructed with `CGAVI` lead to test set classification errors and evaluation times comparable to or better than with related methods such as `AVI` and `VCA`.

2 Related works

2.1 Approximately vanishing ideal

The first algorithm for constructing generators of the vanishing ideal was the *Buchberger-Möller algorithm* (Möller and Buchberger, 1982). Its high susceptibility to noise was first addressed with the introduction of `AVI` in Heldt et al. (2009), see also Fassino (2010); Limbeck (2013). `AVI` is similar to `CGAVI` in that both require a term ordering and construct generators as linear combinations of monomials. The term ordering requirement is the reason why `AVI` and `CGAVI` can produce different outputs depending on the order of the features of the data set, an undesirable property in practice. The currently most prevalent, and contrary to `AVI` and `CGAVI`, polynomial-based approach for constructing generators of the approximately vanishing ideal is `VCA`, introduced by Livni et al. (2013) and improved by Zhang (2018). `VCA` constructs generators not as linear combinations of monomials but of polynomials, that is, `VCA` constructed generators are, in some sense, polynomials whose terms are other polynomials. Like most polynomial-based approaches, `VCA` does not require an ordering of terms and the algorithm has been exploited in hand posture recognition, principal variety analysis for nonlinear data modeling, solution selection using genetic programming, and independent signal estimation for blind source separation tasks (Zhao and Song, 2014; Iraji and Chitsaz, 2017; Kera and Iba, 2016; Wang and Ohtsuki, 2018). Variants of `VCA` have also been studied in connection with kernels (Király et al., 2014; Hou et al., 2016). Despite `VCA`'s prevalence, foregoing the term ordering of monomial-based approaches also gives rise to major disadvantages. `VCA` constructs more generators than `AVI` or `CGAVI`, `VCA`'s generators are non-sparse in their polynomial representation, and `VCA` is highly susceptible to the *spurious vanishing problem* (Kera and Iba, 2016; Kera and Hasegawa, 2019, 2020): Polynomials with small coefficient vector entries that vanish over X still get added to the set of generators even though they do not hold any structurally useful information of the data, and, conversely, polynomials that describe the data well get rejected as non-vanishing due to the size of their (large) coefficient vector entries.

2.2 Conditional Gradients

The core difference between `CGAVI`, `AVI`, and `VCA` is the replacement of the `SVD` steps in the latter two methods with `PFW` calls to construct generators. `PFW` is a variant of the *Frank-Wolfe* (Frank and Wolfe, 1956) or *Conditional Gradients* (Levitin and Polyak, 1966) algorithm. Conditional Gradients methods are a family of algorithms which appear as building blocks in a variety of scenarios in machine learning, e.g., structured prediction (Jaggi and Sulovskỳ, 2010; Giesen et al., 2012; Harchaoui et al., 2012; Freund et al., 2017; Garber et al., 2018), optimal transport (Courty et al., 2016; Paty and Cuturi, 2019; Luise et al., 2019), and video co-localization (Joulin et al., 2014; Bojanowski et al.,

2015; Peyre et al., 2017; Miech et al., 2018). They have also been extensively studied theoretically, with many algorithmic variations (Garber and Meshi, 2016; Braun et al., 2017; Bashiri and Zhang, 2017; Braun et al., 2019; Combettes and Pokutta, 2020; Carderera and Pokutta, 2020) and accelerated convergence regimes (Garber and Hazan, 2013; Lacoste-Julien and Jaggi, 2013; Garber and Hazan, 2015; Pena and Rodriguez, 2018; Gutman and Pena, 2018). Furthermore, Frank-Wolfe algorithms enjoy many appealing properties (Jaggi, 2013); they are easy to implement, projection-free, do not require affine pre-conditioners (Kerdreux et al., 2021), and variants, e.g., `PFW`, offer a simple trade-off between optimization accuracy and sparsity of iterates. All these properties make them appealing algorithmic procedures for practitioners that work at scale. Although Frank-Wolfe algorithms have been considered in polynomial regression, particle filtering, or as pruning methods of infinite RBMS (Blondel et al., 2017; Bach et al., 2012; Ping et al., 2016), their favorable properties have not been exploited in the context of approximately vanishing ideals.

3 Preliminaries

Throughout, let $k, m, n \in \mathbb{N}$. For $n \in \mathbb{N}$, let $[n] := \{1, \ldots, n\}$. We denote vectors in bold and let $\mathbf{0}, \mathbf{1} \in \mathbb{R}^n$ denote the 0- and 1-vector, respectively. Throughout, we use capital calligraphic letters to denote sets of polynomials and denote the sets of terms (or monomials) and polynomials in n variables by $\mathcal{T}$ and $\mathcal{P}$, respectively. Given a polynomial $f \in \mathcal{P}$, let $\deg(f)$ denote its *degree*. We denote the set of polynomials in n variables of and up to degree d by $\mathcal{P}_d$ and $\mathcal{P}_{\leq d}$, respectively. Given a set of polynomials $\mathcal{G} \subseteq \mathcal{P}$, let $\mathcal{G}_d := \mathcal{G} \cap \mathcal{P}_d$ and $\mathcal{G}_{\leq d} := \mathcal{G} \cap \mathcal{P}_{\leq d}$. Given a set of polynomials $\mathcal{G} = \{g_1, \ldots, g_k\} \subseteq \mathcal{P}$ and vector $\mathbf{x} \in \mathbb{R}^n$, define the *evaluation vector* of $\mathcal{G}$ over $\mathbf{x}$ as

$$\mathcal{G}(\mathbf{x}) := (g_1(\mathbf{x}), \ldots, g_k(\mathbf{x}))^\intercal \in \mathbb{R}^k.$$

Throughout, let $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subseteq \mathbb{R}^n$ be a data set consisting of m n-dimensional feature vectors. Given a polynomial $f \in \mathcal{P}$ and a set of polynomials $\mathcal{G} = \{g_1, \ldots, g_k\} \subseteq \mathcal{P}$, define the *evaluation vector* of f and the *evaluation matrix* of $\mathcal{G}$ over X as

$$f(X) := (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_m))^\intercal \in \mathbb{R}^m$$

and

$$\mathcal{G}(X) := (g_1(X), \ldots, g_k(X)) \in \mathbb{R}^{m,k},$$

respectively. Further, define the *mean squared error* of f over X as

$$\operatorname{MSE}(f, X) := \frac{1}{m} \|f(X)\|_2^2.$$

Recall that we address noise in the data set by constructing approximately vanishing polynomials, which we define via the mean squared error.

Definition 3.1 (Approximately vanishing polynomial)**.** Let $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subseteq \mathbb{R}^n$ and $\psi \geq 0$. A polynomial $f \in \mathcal{P}$ is *ψ-approximately vanishing* (over X) if $\operatorname{MSE}(f, X) \leq \psi$.

Recall the *spurious vanishing problem*. For $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subseteq \mathbb{R}^n$, any polynomial f with $\operatorname{MSE}(f, X) > \psi$ can be re-scaled to become ψ-approximately vanishing, regardless of its roots, by multiplying all coefficient vector entries of f with $\sqrt{\frac{\psi}{\operatorname{MSE}(f,X)}}$. To address this issue, we require an ordering of terms, in our case, the *degree-lexicographical ordering of terms (DegLex)* (Cox et al., 2013), denoted by $<_\sigma$. For example, $x <_\sigma y <_\sigma x^2 <_\sigma xy <_\sigma y^2 \ldots$. Throughout, for $\mathcal{O} = \{t_1, \ldots, t_k\}_\sigma \subseteq \mathcal{T}$, the subscript σ indicates that $t_1 <_\sigma \ldots <_\sigma t_k$. DegLex allows us to define the leading term (coefficient) of a polynomial.

Definition 3.2 (Leading term (coefficient))**.** Let $f = \sum_{i=1}^k c_i t_i \subseteq \mathcal{P}$. Define the *leading term* and *leading term coefficient* of f as $\operatorname{LT}(f) := \{t_i \mid t_i >_\sigma t_j \ \forall j \neq i\}$ and $\operatorname{LTC}(f) := \{c_i \mid t_i >_\sigma t_j \ \forall j \neq i\}$, respectively.

For $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subseteq \mathbb{R}^n$, a polynomial f with $\operatorname{LTC}(f) = 1$ that vanishes ψ-approximately over X is called *$(\psi, 1)$-approximately vanishing* (over X). Fixing the LTC of generators prevents rescaling, thus addressing the spurious vanishing problem. We now formally define the approximately vanishing ideal.

Definition 3.3 (Approximately vanishing ideal)**.** Let $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subseteq \mathbb{R}^n$ and $\psi \geq 0$. The *ψ-approximately vanishing ideal* (over X), $\mathcal{I}_X^\psi$, is the ideal generated by all $(\psi, 1)$-approximately vanishing polynomials over X.

Note that the definition above subsumes the definition of the vanishing ideal, i.e., for $\psi = 0$, $\mathcal{I}_X^0 = \mathcal{I}_X$ is the vanishing ideal. In that case, we write $\mathcal{I}_X$ instead of $\mathcal{I}_X^0$, omitting the superscript. In this paper, we introduce an algorithm addressing the following problem.

Problem 3.4 (Setting)**.** Let $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subseteq \mathbb{R}^n$ and $\psi \geq 0$. Construct a set of generators of $\mathcal{I}_X^\psi$.

4 Convex optimization

To solve Problem 3.4, we first address the subproblem of certifying (non-)existence of and constructing generators of the ψ-approximately vanishing ideal, $\mathcal{I}_X^\psi$, under the restriction that terms of generators are contained in a specific set of terms. As we show in this section, this subtask can be reduced to solving a convex optimization problem.

Algorithm 1 ORACLE

Input: $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subseteq \mathbb{R}^n$, $\mathcal{O} = \{t_1, \ldots, t_k\}_\sigma \subseteq \mathcal{T}$, $t \in \mathcal{T}$ with $t >_\sigma t_i$ for all $i \in [k]$, and $\varepsilon \geq 0$.
Output: A polynomial $g \in \mathcal{P}$ with $\operatorname{LT}(g) = t$, $\operatorname{LTC}(g) = 1$, other terms only in $\mathcal{O}$, and $\operatorname{MSE}(g, X) \leq \min_{\mathbf{v} \in \mathbb{R}^{|\mathcal{O}|}} \ell(\mathcal{O}(X), t(X))(\mathbf{v}) + \varepsilon$.

Let $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subseteq \mathbb{R}^n$, $\mathcal{O} = \{t_1, \ldots, t_k\}_\sigma \subseteq \mathcal{T}$, $t \in \mathcal{T}$ such that $t >_\sigma t_i$ for all $i \in [k]$, and $\psi \geq 0$. We present a method that constructs a polynomial g with leading term t and other terms only in $\mathcal{O}$ such that there exist $(\psi, 1)$-approximately vanishing polynomials with leading term t and other terms only in $\mathcal{O}$, if and only if g is one of them.

Suppose there exists a $(\psi, 1)$-approximately vanishing polynomial $f = \sum_{i=1}^k c_i t_i + t$ with $\operatorname{LT}(f) = t$ and other terms only in $\mathcal{O}$. Let $\mathbf{c} = (c_1, \ldots, c_k)^\intercal$. Then,

$$\begin{aligned} \psi \geq \operatorname{MSE}(f, X) &= \ell(\mathcal{O}(X), t(X))(\mathbf{c}) \\ &\geq \min_{\mathbf{v} \in \mathbb{R}^k} \ell(\mathcal{O}(X), t(X))(\mathbf{v}), \end{aligned} \tag{1}$$

where for $A \in \mathbb{R}^{m,n}$, $\mathbf{y} \in \mathbb{R}^m$, and $\mathbf{x} \in \mathbb{R}^n$,

$$\ell(A, \mathbf{y})(\mathbf{x}) := \frac{1}{m} \|A\mathbf{x} + \mathbf{y}\|_2^2 \tag{SL}$$

is the *squared loss*. The right-hand side of (1) is a convex optimization problem and in the argmin-version has the form

$$\mathbf{d} \in \operatorname{argmin}_{\mathbf{v} \in \mathbb{R}^k} \ell(\mathcal{O}(X), t(X))(\mathbf{v}). \tag{COP}$$

With $\mathbf{d}$, we now construct the polynomial $g = \sum_{i=1}^k d_i t_i + t$. Note that $\operatorname{LT}(g) = t$, $\operatorname{LTC}(g) = 1$, and

$$\begin{aligned} \min_{\mathbf{v} \in \mathbb{R}^k} \ell(\mathcal{O}(X), t(X))(\mathbf{v}) &= \frac{1}{m} \|\mathcal{O}(X)\mathbf{d} + t(X)\|_2^2 \\ &= \operatorname{MSE}(g, X). \end{aligned}$$

Thus, g also vanishes $(\psi, 1)$-approximately over X, its leading term is t, and all of its other terms are in $\mathcal{O}$, as required. We obtain the following result.

Theorem 4.1 (Certificate)**.** *Let $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subseteq \mathbb{R}^n$, $\psi \geq 0$, $\mathcal{O} = \{t_1, \ldots t_k\}_\sigma \subseteq \mathcal{T}$, $t \in \mathcal{T}$ such that $t >_\sigma t_i$ for all $i \in [k]$, and $\mathbf{d}$ as in (COP). There exists a $(\psi, 1)$-approximately vanishing polynomial f with $\operatorname{LT}(f) = t$ and other terms only in $\mathcal{O}$, if and only if $g = \sum_{i=1}^k d_i t_i + t$ vanishes $(\psi, 1)$-approximately.*

The discussion above not only constitutes a proof of Theorem 4.1 but also gives rise to an algorithmic blueprint, ORACLE, presented in Algorithm 1. We denote the output of running ORACLE with $X, \mathcal{O}, t,$ and ε by $g = \texttt{ORACLE}(X, \mathcal{O}, t, \varepsilon)$. In practice, any ε-accurate solver of (COP), e.g., gradient descent, can be used to implement ORACLE in two steps: First, solve problem (COP) to ε-accuracy with the solver yielding a vector $\mathbf{d} \in \mathbb{R}^{|\mathcal{O}|}$. Second, construct and return $g = \sum_{i=1}^{|\mathcal{O}|} d_i t_i + t$. We then say that the ε-accurate solver is used as ORACLE. If, for example, we implement ORACLE with gradient descent, we say that we use gradient descent as ORACLE.

In case that ORACLE is implemented with an accurate solver of (COP), that is, $\varepsilon = 0$, by Theorem 4.1, either $\operatorname{MSE}(g, X) > \psi$ and we have proof that no $(\psi, 1)$-approximately vanishing polynomial with leading term t and other terms only in $\mathcal{O}$ exists, or g vanishes $(\psi, 1)$-approximately with leading term t and other terms only in $\mathcal{O}$.[2]

Algorithm 2 OAVI

Input: $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subseteq \mathbb{R}^n$ and $\psi \geq \varepsilon \geq 0$.
Output: $\mathcal{G} \subseteq \mathcal{P}$ and $\mathcal{O} \subseteq \mathcal{T}$.

```
d ← 1
O = {t_1}_σ ← {1}_σ
G ← ∅
while ∂_d O = {u_1, ..., u_k}_σ ≠ ∅ do
    for i = 1, ..., k do
        g ← ORACLE(X, O, u_i, ε) ∈ P
        if MSE(g, X) ≤ ψ then
            G ← G ∪ {g}
        else
            O ← (O ∪ {u_i})_σ
        end if
    end for
    d ← d + 1
end while
```

5 The OAVI algorithm

In this section, we give a detailed description of the *Oracle Approximately Vanishing Ideal algorithm* (OAVI), which we refer to as CGAVI when the ORACLE used is a Conditional Gradients algorithm. OAVI is presented in Algorithm 2 and we analyze its properties in Section 6.

Input: Recall Problem 3.4. Our goal is to construct generators of the ψ-approximately vanishing ideal, $\mathcal{I}_X^\psi$, where ψ is a tolerance introduced to control how strongly polynomials vanish over the data $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subseteq \mathbb{R}^n$. During its execution, OAVI makes calls to ORACLE, the accuracy of which is controlled by the tolerance ε.

To avoid cumbersome notation, we motivate OAVI for

[2] Here, we focus on $\varepsilon = 0$ for ease of exposition. We discuss the case when $\varepsilon > 0$ in Theorem 6.2.

$\psi = \varepsilon = 0$, that is, we focus on the construction of generators of the vanishing ideal $\mathcal{I}_X$ assuming access to an accurate solver of (COP).

Initialization: We keep track of two sets: $\mathcal{O} \subseteq \mathcal{T}$ for the set of terms such that no generator exists with terms only in $\mathcal{O}$ and $\mathcal{G} \subseteq \mathcal{P}$ the set of generators. Since the constant 1 polynomial does not vanish, we initialize $\mathcal{O} = \{t_1\}_\sigma \leftarrow \{1\}_\sigma$ and $\mathcal{G} \leftarrow \emptyset$.

Line 4: After constructing degree $d-1$ generators, in a next step, given $\mathcal{O}_{\leq d-1}$ and $\mathcal{G}_{\leq d-1}$, `OAVI` constructs all vanishing polynomials of degree d. Checking for all $\binom{n}{d}$ monomials of degree d whether one of them is the leading term of a generator is impractical. Instead, note that there exists a set of generators $\mathcal{G}$ of $\mathcal{I}_X$ such that none of the terms of generators in $\mathcal{G}$ are divisible[3] by the leading terms of other generators in $\mathcal{G}$.

Lemma 5.1. *Let $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subseteq \mathbb{R}^n$. There exists a set of generators $\mathcal{G} \subseteq \mathcal{P}$ of $\mathcal{I}_X$ such that for $g, h \in \mathcal{G}$,$g \neq h$, $h = \sum_{i=1}^k c_i t_i \in \mathcal{G}$, $\mathrm{LT}(g) \nmid t_i$ for any $i \in [k]$.*

To construct degree d generators, we thus only consider terms $t \in \mathcal{T}_d$ such that for all $g \in \mathcal{G}_{\leq d-1}$, $\mathrm{LT}(g) \nmid t$. This is equivalent to requiring that all divisors of degree $\leq d-1$ of t are in $\mathcal{O}_{\leq d-1}$.

Definition 5.2 (Border)**.** Let $\mathcal{O} \subseteq \mathcal{T}_{d-1}$. The *(degree d) border* of $\mathcal{O}$ is defined as $\partial_d\mathcal{O} := \{u \in \mathcal{T}_d \colon t \in \mathcal{O}_{\leq d-1} \text{ for all } t \in \mathcal{T}_{\leq d-1} \text{ such that } t \mid u\}$.

In other words, we only consider degree d terms that are contained in the border, drastically reducing the number of redundant generators constructed.

While loop: Suppose that $\partial_d\mathcal{O} \neq \emptyset$, else `OAVI` terminates. For each term $u \in \partial_d\mathcal{O}$, starting with the smallest with respect to $<_\sigma$, we construct a polynomial g via a call to `ORACLE`. By Theorem 4.1, there exists a vanishing polynomial f with $\mathrm{LTC}(f) = 1$, $\mathrm{LT}(f) = u$, and other terms only in $\mathcal{O}$, if and only if g is a vanishing polynomial. If g vanishes, we append g to $\mathcal{G}$. If g does not vanish, we append $\mathrm{LT}(g) = u$ to $\mathcal{O}$.

Termination: For bounded data $X \subseteq [-1, 1]^n$[4], the border is guaranteed to become empty and the algorithm terminates, see Theorem 6.1.

Output: Throughout, we denote the output of `OAVI` by $\mathcal{G}$ and $\mathcal{O}$, that is, $(\mathcal{G}, \mathcal{O}) = \texttt{OAVI}(X, \psi, \varepsilon)$.

[3]Recall that for $t, u \in \mathcal{T}$, *t divides (or is a divisor of) u*, denoted $t \mid u$, if there exists $s \in \mathcal{T}$ such that $t \cdot s = u$. If t does not divide u, we write $t \nmid u$.

[4]Bounded data can, for example, be obtained via min-max feature scaling.

Remark 5.3 (`AVI` and `VCA`)**.** We use the notation from this paper to refer to the sets corresponding to $\mathcal{O}$, $\mathcal{G}$, and $\partial_d\mathcal{O}$ in `AVI` and `VCA` even though notation in Heldt et al. (2009) and Livni et al. (2013), respectively, may differ. For `AVI`, $\mathcal{O}$, $\mathcal{G}$, and $\partial_d\mathcal{O}$ are identical to the sets used in `OAVI`. `AVI` does, however, employ an `SVD` step to determine all generators with leading terms in the border at once. `VCA` employs a similar `SVD` step as `AVI` for generator construction. The main difference between `OAVI`, `AVI`, and `VCA` is that the latter is polynomial-based. Indeed, the set $\mathcal{O}$ in `VCA` does not contain monomials but polynomials that provably do not vanish approximately over the data. Thus, `VCA` does not require a term ordering. The disadvantage of avoiding the term ordering is that the border in `VCA` cannot be defined as in Definition 5.2 but is instead defined as $\partial_d\mathcal{O} := \{u \in \mathcal{T}_d \mid u = s \cdot t \text{ for } s \in \mathcal{O}_1 \text{ and } t \in \mathcal{O}_{d-1}, \}$, which can lead to the construction of unnecessary generators. For example, suppose that $\mathcal{O}_1 = \{t_1, \ldots, t_n\}$ and $\mathcal{O}_2 = \{t_1^2\}$. By construction, t_1t_j and t_it_j are leading terms of approximately vanishing polynomials for all $i, j \in \{2, \ldots, n\}$ and, by Lemma 5.1, there is no need to consider generators with leading terms $t_1^2t_2, \ldots t_1^2t_n$. This is exploited in the border of `OAVI` and `AVI`, for which $\partial_3\mathcal{O} = \{t_1^3\}$, but the border for `VCA` is $\partial_3\mathcal{O} = \{t_1^3, t_1^2t_2 \ldots, t_1^2t_n\}$, which can lead to the construction of $n-1$ unnecessary generators.

6 Analysis

We begin the analysis of `OAVI` by proving that the algorithm terminates.

Theorem 6.1 (Termination)**.** *Let $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subseteq [-1, 1]^n$, $\psi \geq \varepsilon \geq 0$, and $(\mathcal{G}, \mathcal{O}) = \texttt{OAVI}(X, \psi, \varepsilon)$. Then, `OAVI` terminates.*

Thus, for $\varepsilon = 0$, `OAVI` constructs a set of generators of the ψ-approximately vanishing ideal in finite time. Since, in practice, we employ solvers that are ε-accurate only for $\varepsilon > 0$, in the theorem below, we prove that when $\varepsilon > 0$, `OAVI` addresses Problem 3.4 up to tolerance ε.

Theorem 6.2 (Maximality)**.** *Let $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subseteq [-1, 1]^n$, $\psi \geq \varepsilon \geq 0$, and $(\mathcal{G}, \mathcal{O}) = \texttt{OAVI}(X, \psi, \varepsilon)$. Then, all $g \in \mathcal{G} \neq \emptyset$ are $(\psi, 1)$-approximately vanishing polynomials and there does not exist a $(\psi - \varepsilon, 1)$-approximately vanishing polynomial with terms only in $\mathcal{O}$.*

Thus, `OAVI` is guaranteed to construct all $(\psi - \varepsilon, 1)$-approximately vanishing polynomials but generators that vanish $(\lambda, 1)$-approximately, where $\lambda \in]\psi - \varepsilon, \psi]$, may not be detected by `OAVI`.

6.1 Computational complexity

For bounded data $X \subseteq [-1,1]^n$, OAVI returns bounded $|\mathcal{G}|$ and $|\mathcal{O}|$, a useful property for discussing time and space complexity.

Proposition 6.3. *Let $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subseteq [-1,1]^n$, $\psi \geq \varepsilon \geq 0$, and $(\mathcal{G}, \mathcal{O}) = \texttt{OAVI}(X, \psi, \varepsilon)$. Then, $|\mathcal{O}| \leq m$ and $|\mathcal{G}| \leq |\mathcal{O}| n \leq mn$.*

Proposition 6.3 is an essential tool for the analysis of the computational complexity of OAVI.

Theorem 6.4 (Computational Complexity)**.** *Let $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subseteq [-1,1]^n$, $\psi \geq \varepsilon \geq 0$, and $(\mathcal{G}, \mathcal{O}) = \texttt{OAVI}(X, \psi, \varepsilon)$. In the real number model, the time and space complexities of OAVI are $O(m^2n^2 + mnT_{\texttt{ORACLE}})$ and $O(m^2n + S_{\texttt{ORACLE}})$, where $T_{\texttt{ORACLE}}$ and $S_{\texttt{ORACLE}}$ are the time and space complexities of ORACLE, respectively.*

As we prove in the adaptation of Theorem 5.1 in Livni et al. (2013) to OAVI below, the evaluation complexity of OAVI's output is identical to that of related methods such as AVI and VCA.

Theorem 6.5 (Evaluation Complexity)**.** *Let $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subseteq [-1,1]^n$, $\psi \geq \varepsilon \geq 0$, and $(\mathcal{G}, \mathcal{O}) = \texttt{OAVI}(X, \psi, \varepsilon)$. In the real number model, we can compute the evaluation vectors of all monomials in $\mathcal{O}$ and polynomials in $\mathcal{G}$ over a set $Z = \{\mathbf{z}_1, \ldots, \mathbf{z}_q\} \subseteq [-1,1]^n$ in time $O(|\mathcal{O}|q)$ and $O(|\mathcal{G}||\mathcal{O}|q)$, respectively.*

To reduce the time required to evaluate the feature transformation on new data Z, one can reduce the number of terms in $\mathcal{O}$, construct fewer polynomials in $\mathcal{G}$, increase the sparsity of polynomials in $\mathcal{G}$, or reduce the number of samples in the set Z.

6.2 Generalization bounds

So far, the construction of generators for OAVI with ORACLE involves solving an unconstrained convex optimization problem. In this section, we replace (COP) with a constrained convex optimization problem and show that this replacement allows OAVI to create generators that approximately vanish not only over in-sample but also over out-sample data.

For $\tau \geq 2$, a polynomial $f = \sum_{i=1}^k c_i t_i \in \mathcal{P}$ with $\mathbf{c} = (c_1, \ldots, c_k)^\intercal$ is said to be *τ-bounded* in norm $\|\cdot\|$ if the norm of its coefficient vector is bounded by τ, that is, if $\|f\| := \|\mathbf{c}\| \leq \tau$. Replacing (COP) by

$$\mathbf{d} \in \operatorname{argmin}_{\mathbf{v} \in \mathbb{R}^k, \|\mathbf{v}\| \leq \tau - 1} \ell(\mathcal{O}(X), t(X))(\mathbf{v}) \quad \text{(CCOP)}$$

allows OAVI to create τ-bounded generators in $\|\cdot\|$. When $\|\cdot\| = \|\cdot\|_1$, (CCOP) is the bounded LASSO, for which generalization bounds are available, see, e.g., Mohri et al. (2018). Thus, τ-bounded generators constructed by OAVI also vanish on out-sample data.

Theorem 6.6 (Vanishing property)**.** *Let $\mathcal{X} \subseteq [-1,1]^n$, $\mathcal{Y} \subseteq [-1,1]$, $\mathcal{D}$ be a distribution over $\mathcal{X} \times \mathcal{Y}$, $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\} \in (\mathcal{X} \times \mathcal{Y})^m$ sampled according to $\mathcal{D}$, and $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$. Let $\psi \geq \varepsilon \geq 0$, $\tau \geq 2$, and $(\mathcal{G}, \mathcal{O}) = \texttt{OAVI}(X, \psi, \varepsilon)$. Suppose that for all $g \in \mathcal{G}$, we have $\|g\|_1 \leq \tau$. Then, for any $g \in \mathcal{G}$ and $\delta > 0$, with probability at least $1 - \delta$, we have*

$$\mathbb{E}_{(\mathbf{x},y) \backsim \mathcal{D}} \left[\mathrm{MSE}(g, \{\mathbf{x}\})\right] \leq \mathrm{MSE}(g, X) + 2\tau^2 \sqrt{\frac{2\log(2m)}{m}} + \tau^2 \sqrt{\frac{\log(\delta^{-1})}{2m}}.$$

Generators created by AVI and VCA are not guaranteed to be τ-bounded and, thus, in general, do not satisfy Theorem 6.6. On the other hand, generators constructed by OAVI with an ORACLE solving (CCOP) instead of (COP) are guaranteed to be τ-bounded in $\|\cdot\|_1$ due to the equivalence of norms in finite dimensions and, thus, vanish also on out-sample data.

We extend our analysis of generalization bounds in Appendix B. Specifically, in Theorem B.3, we present sufficient conditions under which feature transformation algorithms in combination with a linear kernel SVM inherit the margin bound of the SVM.

6.3 On approximately vanishing

In Definition 3.1, the mean squared error is used to define approximately vanishing polynomials but other options are possible. In AVI, for example, a polynomial $f \in \mathcal{P}$ vanishes ψ-approximately over $X \subseteq [-1,1]^n$ if $\|f(X)\|_2 \leq \psi$. However, definitions of vanishing approximately that are less similar to ours are imaginable. We could, for example, define vanishing ψ-approximately via the maximum loss instead, that is, f would be defined as vanishing ψ-approximately over X if $|f(\mathbf{x})| \leq \psi$ for all $\mathbf{x} \in X$. Minimizing the maximum loss instead of the squared loss in (COP) results in a convex-concave optimization problem that has to be addressed with specific solvers, for example via the method described in Shalev-Shwartz and Wexler (2016).

In this paper, we limit ourselves to adding an ℓ_2-regularization term $\frac{\lambda}{2}\|\mathbf{x}\|_2^2$ to (SL). Then, a polynomial $f = \sum_{i=1}^k c_i t_i + t \in \mathcal{P}$ with $\mathrm{LT}(f) = t$ vanishes $(\psi, 1)$-approximately over X if

$$\mathrm{MSE}(f, X) + \frac{\lambda}{2}\|(c_1, \ldots, c_k)^\intercal\|_2^2 \leq \psi.$$

Setting $\lambda > 0$ can prolong the running time of OAVI or make termination impossible altogether. Furthermore, $|\mathcal{G}|$ and $|\mathcal{O}|$ are no longer guaranteed to be bounded. In our practical experiments, however, these negative properties do not seem to be present, see Section 9.

7 Pipeline

Following the notation of Mohri et al. (2018), we present a detailed overview of the machine learning pipeline for classification problems using generators of the vanishing ideal to transform features for a linear kernel `SVM`.

Consider an input space $\mathcal{X} \subseteq [-1,1]^n$ and an output or target space $\mathcal{Y} = [k]$. We receive a training sample $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\} \in (\mathcal{X} \times \mathcal{Y})^m$ drawn *i.i.d.* from some unknown distribution $\mathcal{D}$. The problem is to determine a *hypothesis* $h\colon \mathcal{X} \to \mathcal{Y}$ with small *generalization error*

$$\mathbb{P}_{(\mathbf{x},y)\backsim\mathcal{D}}[h(\mathbf{x}) \neq y].$$

Let $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$. For each class $i \in [k]$, let $X^i \subseteq X$ denote the set of feature vectors corresponding to class i and construct a set of generators $\mathcal{G}^i = \{g_j^{(i)}\}_{j=1}^{|\mathcal{G}^i|}$ for the vanishing ideal $\mathcal{I}_{X^i}$. As proposed in Livni et al. (2013), we can then transform samples $\mathbf{x} \in X$ via the feature transformation

$$\mathbf{x} \mapsto \tilde{\mathbf{x}} = \left(\ldots, |g_1^{(i)}(\mathbf{x})|, \ldots, |g_{|\mathcal{G}^i|}^{(i)}(\mathbf{x})|, \ldots\right)^\intercal, \quad \text{(FT)}$$

where $g_j^{(i)}$ is the j-th vanishing polynomial corresponding to class i. In other words, $g_j^{(i)}$ vanishes over all $\mathbf{x} \in X^i$ and (hopefully) attains non-zero values over points $\mathbf{x} \in X \setminus X^i$. We then train a linear kernel `SVM` on the feature transformed data $\tilde{S} = \{(\tilde{\mathbf{x}}_1, y_1), \ldots, (\tilde{\mathbf{x}}_m, y_m)\}$ with ℓ_1-regularization to keep the number of used features as small as possible. As shown in Livni et al. (2013), if the underlying classes of S belong to disjoint *algebraic sets*[5], they become linearly separable in the feature space corresponding to transformation (FT). If we restrict the number and norm of the coefficient vectors of generators used to transform the features, e.g., by bounding the maximum degree of generators and, in the case of `OAVI`, replacing (COP) by (CCOP), this combination of feature transformation and `SVM` satisfies Theorem B.3, guaranteeing that test set accuracy improves as the number of training samples increases.

8 Putting the `CG` in `CGAVI`

Any algorithm guaranteeing an ε-accurate solution to (COP) or (CCOP) can be used to construct the polynomial g returned by `ORACLE`, allowing the practitioner to choose a solver with specific properties for any given task. Our goal is to construct a set of generators $\mathcal{G}$ consisting of few and sparse polynomials to obtain a compact representation of the approximately vanishing ideal. The former property is already achieved by restricting leading terms of generators to be in the border, see Section 5. The latter property, we address with our choice of solver for (COP) or (CCOP).

To do so, we formalize the notion of sparsity. Consider the execution of `OAVI` (or `AVI`) and suppose that, currently, $\mathcal{O} = \{t_1, \ldots, t_k\}$ and $g = \sum_{i=1}^k c_i t_i + t$ with $\mathrm{LT}(g) = t \notin \mathcal{O}$ gets appended to $\mathcal{G}$. Let the number of entries, the number of zero entries, and the number of non-zero entries in the coefficient vector of g be denoted by $g_e := k$, $g_z := |\{c_i = 0 \mid i \in [k]\}|$, and $g_n := g_e - g_z$, respectively.[6] We then define the *sparsity* of g as $\mathrm{SPAR}(g) := g_z / g_e \in [0,1]$. Larger $\mathrm{SPAR}(g)$ indicates a more thinly populated coefficient vector of g. Recall that for classification as in Section 7 we construct sets of vanishing polynomials $\mathcal{G}^i$ corresponding to classes i and then transform the samples via (FT) using all polynomials in $\mathcal{G} := \bigcup_i \mathcal{G}^i$. To measure sparsity of the feature transformation, define the *sparsity* of $\mathcal{G}$ as

$$\mathrm{SPAR}(\mathcal{G}) := \frac{\sum_{g\in\mathcal{G}} g_z}{\sum_{g\in\mathcal{G}} g_e} \in [0,1], \quad \text{(SPAR)}$$

which is the weighted average of the sparsity of all polynomials in $\mathcal{G}$ with respect to g_e.

In this paper, we focus on the implementation of `ORACLE` with the Pairwise Frank-Wolfe algorithm (`PFW`), presented in Algorithm 3 in Appendix C. The algorithm converges linearly when optimizing the squared loss over the ℓ_1-ball using line search for its step-size rule and the iterate returned by `PFW` tends to be sparse (Lacoste-Julien and Jaggi, 2015). Furthermore, when optimizing over a bounded region, the returned iterate is bounded and the corresponding generator constructed in `ORACLE` is bounded, which is necessary for the generalization bounds presented in this paper, Theorems 6.6 and B.3, to hold.

9 Numerical experiments

We use different algorithms that construct generators of the approximately vanishing ideal to transform features for a linear kernel `SVM` and compare them to each other and a polynomial kernel `SVM`.

Set-up. We apply different feature transformation algorithms for a linear kernel `SVM` using the set-up from Section 7 and a polynomial kernel `SVM` to classification problems. Let $\mathcal{G} := \bigcup_i \mathcal{G}^i$, where $\mathcal{G}^i$ is the set of generators constructed for class i as in Section 8. We compare the number of generators constructed, $\sum_i |\mathcal{G}^i|$,

[5] Recall that a set $U \in \mathbb{R}^n$ is *algebraic* if there exists a finite set of polynomials $\mathcal{U} = \{u_1, \ldots, u_k\} \subseteq \mathcal{P}$, such that U is the set of the common roots of $\mathcal{U}$.

[6] For `VCA`, define g_e, g_z, and g_n with respect to the representation of g as a linear combination of polynomials.

Table 1: Overview of the data sets used in the numerical experiments. Each row contains the data set used, its full name, the number of samples, the number of features, and the number of classes.

Data set	Full name	# samples	# features	# classes
bank	banknote authentication	1372	4	2
cancer	breast cancer Wisconsin	569	30	2
htru2	HTRU2 (Lyon et al., 2016)	17898	8	2
iris	iris	150	4	3
seeds	seeds	210	7	3
sonar	connectionist bench (sonar, mines vs. rocks)	208	60	2
spam	spambase	4601	57	2
voice	LSVT voice rehabilitation (Tsanas et al., 2013)	126	310	2
wine	wine	178	13	3

the sparsity of the feature transformation, (SPAR), the number of non-zero entries in the coefficient vectors of generators in $\mathcal{G}$, $\sum_{g\in\mathcal{G}} g_n$, the test time in seconds, Test time, and the classification error on the test set in %, Test error. The results are averaged over ten random 60%/40% train/test partitions.

Implementation. All experiments are performed on an NVIDIA TITAN RTX GPU with 24GB RAM and a 64-core AMD Ryzen Threadripper 3990X CPU at 3.30 GHz with 128 GB RAM. Our code is publicly available on GitHub. The algorithms are:
ℓ_1-CGAVI: We implement OAVI with PFW and ℓ_2-regularized squared loss as ORACLE for (CCOP) with $\varepsilon = \psi/2$ and radius $\tau = 50$[7] using line search. PFW is terminated when the polynomial corresponding to the constructed iterate is guaranteed to vanish ψ-approximately, if the Frank-Wolfe gap is smaller than ε, or if no significant progress is made during an iteration.
ℓ_2-CGAVI: The implementation is identical to ℓ_1-CGAVI, except that we replace PFW with vanilla Conditional Gradients and the ℓ_1- by the ℓ_2-ball.
AGDAVI: We implement *Accelerated Gradient Descent* (AGD) (Nesterov, 1983) with ℓ_2-regularized squared loss as ORACLE in OAVI. AGD is terminated when the polynomial corresponding to the constructed iterate is guaranteed to vanish ψ-approximately or if no significant progress is made over 5 iterations.
AVI: We implement AVI as in Heldt et al. (2009) with improved *Stabilized Reduced Row Echelon Form algorithm* as in Limbeck (2013).
VCA: We implement VCA as in Livni et al. (2013).
SVM: We implement a polynomial kernel multiclass SVM with one-versus-rest approach using the SCIKIT-LEARN software package (Pedregosa et al., 2011). We run the SVM up to tolerance 1e-3 with ℓ_2-regularization. For the SVM, $\sum_i |\mathcal{G}^i|$, (SPAR), and $\sum_{g\in\mathcal{G}} g_n$ do not exist.

All algorithms above are terminated after constructing degree 10 polynomials, even if the upcoming border is not empty. After applying the algorithms, except for SVM, we subsequently apply a linear kernel SVM as in Section 7. We implement the linear kernel SVM using the SCIKIT-LEARN software package and run it for up to 1000 iterations, up to tolerance 1e-4, with ℓ_1-penalized squared hinge loss, and one versus the rest approach.

The abbreviations for the feature transformation algorithms above also refer to the complete approach of feature transformation with subsequently applied linear kernel SVM. Further, OAVI refers to ℓ_1-CGAVI, ℓ_2-CGAVI, and AGDAVI.

Data sets. All data sets are retrieved from the UCI Machine Learning Repository (Dua and Graff, 2017). An overview is given in Table 1. For each data set, we apply min-max feature scaling into the range $[0, 1]$ as a preprocessing step.

Hyperparameters. We tune the hyperparameters using three-fold cross-validation. For all approaches except the polynomial kernel SVM, we tune the ℓ_1-regularization coefficient for the linear kernel SVM. For OAVI, we tune the vanishing parameter ψ and the ℓ_2-regularization parameter. For AVI, we tune the vanishing parameter ψ and the tolerance τ controlling the sparsity of the constructed generators. For VCA, we tune the vanishing parameter ψ. For the polynomial kernel SVM, we tune the ℓ_2-regularization parameter and the degree of the polynomial kernel.

Results.
$\sum_i |\mathcal{G}^i|$: As can be seen by Table 2, for all data sets except for bank and cancer, OAVI constructs fewer generators than AVI and VCA. For seeds, sonar, spam, voice, and wine, VCA constructs up to an order of magnitude more generators than OAVI due to employing a different border set as discussed in Remark 5.3. For most data sets, AVI also produces significantly more generators than OAVI but the reason for this is not yet understood.

[7] We observe that the radii of the feasibility regions do not affect the numerical results when chosen sufficiently large and that the effect of $\varepsilon \in]0, \psi[$ is small.

Table 2: We compare the number of generators constructed, $\sum_i |\mathcal{G}^i|$, the sparsity of the feature transformation, (SPAR), the number of non-zero entries in the coefficient vectors of generators in $\mathcal{G}$, $\sum_{g\in\mathcal{G}} g_n$, the test time in seconds, Test time, and the classification error on the test set in %, Test error. The results are averaged over ten random 60%/40% train/test partitions and the best results in each category are in bold.

Algorithms		Data sets								
		bank	cancer	htru2	iris	seeds	sonar	spam	voice	wine
$\sum_i \lvert\mathcal{G}^i\rvert$	ℓ_1-CGAVI	19	153	20	**13**	26	**345**	**118**	649	**79**
	ℓ_2-CGAVI	17	115	22	**13**	**24**	399	122	631	116
	AGDAVI	21	132	**18**	14	25	475	120	**622**	95
	AVI	15	288	101	22	47	1205	123	1204	201
	VCA	**14**	**104**	81	99	332	3350	2413	1789	505
(SPAR)	ℓ_1-CGAVI	**0.65**	**0.54**	**0.53**	**0.15**	**0.13**	**0.82**	**0.37**	**0.36**	**0.51**
	ℓ_2-CGAVI	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	AGDAVI	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00
	AVI	0.01	0.03	0.02	0.00	0.05	0.02	0.04	0.06	0.03
	VCA	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
$\sum_g g_n$	ℓ_1-CGAVI	**6.8e+1**	**5.2e+2**	**4.2e+1**	**3.0e+1**	9.9e+1	**1.2e+3**	**2.4e+2**	1.9e+3	**2.5e+2**
	ℓ_2-CGAVI	1.3e+2	1.3e+3	1.2e+2	4.0e+1	**7.8e+1**	9.5e+3	4.5e+2	2.5e+3	2.0e+3
	AGDAVI	2.2e+2	1.8e+3	7.3e+1	5.0e+1	1.3e+2	1.2e+4	4.1e+2	**1.8e+3**	1.6e+3
	AVI	7.1e+1	1.2e+4	2.4e+3	1.3e+2	3.8e+2	6.5e+4	3.7e+2	2.3e+4	4.6e+3
	VCA	1.7e+2	8.4e+3	4.7e+3	4.3e+3	3.8e+4	4.5e+6	4.0e+6	1.7e+6	9.0e+4
Test time	ℓ_1-CGAVI	2.2e-3	1.1e-3	1.8e-3	9.1e-4	1.1e-3	1.8e-3	1.3e-3	1.1e-3	1.9e-3
	ℓ_2-CGAVI	2.0e-3	9.1e-4	1.9e-3	1.1e-3	8.4e-4	2.1e-3	1.4e-3	1.1e-3	3.1e-3
	AGDAVI	2.6e-3	1.1e-3	**1.6e-3**	1.2e-3	1.2e-3	2.5e-3	1.5e-3	6.9e-4	2.5e-3
	AVI	1.2e-3	1.6e-3	3.7e-3	1.6e-3	1.7e-3	2.1e-3	**1.2e-3**	1.5e-3	2.0e-3
	VCA	7.6e-4	8.9e-4	2.3e-3	2.6e-3	3.2e-3	2.5e-3	8.5e-3	1.5e-3	2.4e-3
	SVM	**3.4e-4**	**3.8e-4**	8.3e-2	**1.6e-4**	**2.2e-4**	**3.2e-4**	2.8e-2	**2.6e-4**	**2.0e-4**
Test error	ℓ_1-CGAVI	0.09	3.42	**2.05**	4.33	5.95	20.95	**5.90**	19.80	2.08
	ℓ_2-CGAVI	0.05	3.29	2.09	4.17	5.48	19.88	6.08	**18.24**	2.64
	AGDAVI	0.09	3.38	2.10	4.33	6.43	**17.50**	5.92	21.76	**1.94**
	AVI	**0.00**	3.46	2.11	4.00	**4.76**	26.43	6.64	23.14	3.33
	VCA	**0.00**	5.44	2.15	4.17	5.71	31.90	7.13	29.02	3.06
	SVM	**0.00**	**2.72**	**2.05**	**3.17**	6.79	21.07	7.22	18.43	3.19

(SPAR): ℓ_1-CGAVI produces sparse, AVI less sparse, and other algorithms practically non-sparse feature transformations.

$\sum_{g\in\mathcal{G}} g_n$: For all data sets but seeds and voice, ℓ_1-CGAVI creates the feature transformation with the smallest number of non-zero entries in the coefficient vectors of generators in $\mathcal{G}$. ℓ_2-CGAVI, AGDAVI, and AVI produce similar amounts of non-zero entries. Since VCA constructs a lot of generators, VCA often constructs the most non-zero entries. For seeds, sonar, spam, and voice, VCA constructs multiple orders of magnitude more non-zero entries than other algorithms.

Test time: The test times for the SVM are the fastest for small data sets. For large data sets, that is, htru2 and spam, the SVM is slower than the other approaches by a factor of at least 3. The test times for the monomial-based algorithms ℓ_1-CGAVI, ℓ_2-CGAVI, AGDAVI, and AVI are very similar. Since ℓ_1-CGAVI represents the approximately vanishing ideal with few and sparse generators, for all data sets except for bank and cancer, ℓ_1-CGAVI enjoys either the second-fastest or third-fastest test time. Except for bank and cancer, VCA's test time is slow. Especially for spam, a large data set with a lot of features, VCA's test time is 5 times slower than the test times of the monomial-based approaches and the polynomial kernel SVM.

Test error: All six algorithms enjoy comparable error rates on the test set. For most data sets, VCA tends to perform slightly worse than the other algorithms, especially on cancer, sonar, spam, and voice. OAVI tends to outperform the SVM on more complex data sets, whereas on easier data sets, the SVM has better baseline results.

10 Discussion

Here we present an ORACLE-based approach for the construction of generators of the approximately vanishing ideal. OAVI constructs few and, in the case of ℓ_1-CGAVI, sparse generators. Most importantly, this compact representation of the approximately vanishing ideal does not lead to worse error rates on the test set but sometimes leads to superior generalization performance than AVI, VCA, and the polynomial kernel SVM. Under mild assumptions, we also derive two generalization bounds for CGAVI.

Acknowledgements

This research was partially funded by Deutsche Forschungsgemeinschaft (DFG) through the DFG Cluster of Excellence MATH+.

References

Bach, F., Lacoste-Julien, S., and Obozinski, G. (2012). On the equivalence between herding and conditional gradient algorithms. In *ICML 2012 International Conference on Machine Learning*.

Bashiri, M. A. and Zhang, X. (2017). Decomposition-invariant conditional gradient for general polytopes with line search. In *Advances in Neural Information Processing Systems*, pages 2687–2697.

Blondel, M., Niculae, V., Otsuka, T., and Ueda, N. (2017). Multi-output polynomial networks and factorization machines. In *Advances in Neural Information Processing Systems*, pages 3349–3359.

Bojanowski, P., Lajugie, R., Grave, E., Bach, F., Laptev, I., Ponce, J., and Schmid, C. (2015). Weakly-supervised alignment of video with text. In *Proceedings of the IEEE international conference on computer vision*, pages 4462–4470.

Braun, G., Pokutta, S., Tu, D., and Wright, S. (2019). Blended conditional gradients. In *International Conference on Machine Learning*, pages 735–743. PMLR.

Braun, G., Pokutta, S., and Zink, D. (2017). Lazifying conditional gradient algorithms. *Proceedings of ICML*.

Carderera, A. and Pokutta, S. (2020). Second-order conditional gradients. *arXiv preprint arXiv:2002.08907*.

Combettes, C. and Pokutta, S. (2020). Boosting frank-wolfe by chasing gradients. In *International Conference on Machine Learning*, pages 2111–2121. PMLR.

Courty, N., Flamary, R., Tuia, D., and Rakotomamonjy, A. (2016). Optimal transport for domain adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 39(9):1853–1865.

Cox, D., Little, J., and OShea, D. (2013). *Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra*. Springer Science & Business Media.

Dua, D. and Graff, C. (2017). Uci machine learning repository.

Fassino, C. (2010). Almost vanishing polynomials for sets of limited precision points. *Journal of symbolic computation*, 45(1):19–37.

Frank, M. and Wolfe, P. (1956). An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110.

Freund, R. M., Grigas, P., and Mazumder, R. (2017). An extended Frank-Wolfe method with "in-face" directions, and its application to low-rank matrix completion. *SIAM Journal on optimization*, 27(1):319–346.

Garber, D. and Hazan, E. (2013). A linearly convergent conditional gradient algorithm with applications to online and stochastic optimization. *arXiv preprint arXiv:1301.4666*.

Garber, D. and Hazan, E. (2015). Faster rates for the Frank-Wolfe method over strongly-convex sets. In *32nd International Conference on Machine Learning, ICML 2015*.

Garber, D. and Meshi, O. (2016). Linear-memory and decomposition-invariant linearly convergent conditional gradient algorithm for structured polytopes. *Advances in Neural Information Processing Systems*, 29:1001–1009.

Garber, D., Sabach, S., and Kaplan, A. (2018). Fast generalized conditional gradient method with applications to matrix recovery problems. *arXiv preprint arXiv:1802.05581*.

Giesen, J., Jaggi, M., and Laue, S. (2012). Optimizing over the growing spectrahedron. In *European Symposium on Algorithms*, pages 503–514. Springer.

Guélat, J. and Marcotte, P. (1986). Some comments on Wolfe's 'away step'. *Mathematical Programming*, 35(1):110–119.

Gutman, D. H. and Pena, J. F. (2018). The condition of a function relative to a polytope. *arXiv preprint arXiv:1802.00271*.

Harchaoui, Z., Douze, M., Paulin, M., Dudik, M., and Malick, J. (2012). Large-scale image classification with trace-norm regularization. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3386–3393. IEEE.

Heldt, D., Kreuzer, M., Pokutta, S., and Poulisse, H. (2009). Approximate computation of zero-dimensional polynomial ideals. *Journal of Symbolic Computation*, 44(11):1566–1591.

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

Iraji, R. and Chitsaz, H. (2017). Principal variety analysis. In *Conference on Robot Learning*, pages 97–108.

Jaggi, M. (2013). Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *Proceedings of*

the 30th international conference on machine learning, number CONF, pages 427–435.

Jaggi, M. and Sulovskỳ, M. (2010). A simple algorithm for nuclear norm regularized problems. In *ICML*.

Joulin, A., Tang, K., and Fei-Fei, L. (2014). Efficient image and video co-localization with Frank-Wolfe algorithm. In *European Conference on Computer Vision*, pages 253–268. Springer.

Kera, H. and Hasegawa, Y. (2019). Spurious vanishing problem in approximate vanishing ideal. *IEEE Access*, 7:178961–178976.

Kera, H. and Hasegawa, Y. (2020). Gradient boosts the approximate vanishing ideal. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4428–4435.

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

Kerdreux, T., Liu, L., Lacoste-Julien, S., and Scieur, D. (2021). Affine invariant analysis of frank-wolfe on strongly convex sets. In *International Conference on Machine Learning*, pages 5398–5408. PMLR.

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

Lacoste-Julien, S. and Jaggi, M. (2013). An affine invariant linear convergence analysis for Frank-Wolfe algorithms. *arXiv preprint arXiv:1312.7864*.

Lacoste-Julien, S. and Jaggi, M. (2015). On the global linear convergence of Frank-Wolfe optimization variants. In *Advances in neural information processing systems*, pages 496–504.

Levitin, E. S. and Polyak, B. T. (1966). Constrained minimization methods. *USSR Computational mathematics and mathematical physics*, 6(5):1–50.

Limbeck, J. (2013). Computation of approximate border bases and applications.

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

Luise, G., Salzo, S., Pontil, M., and Ciliberto, C. (2019). Sinkhorn barycenters with free support via Frank-Wolfe algorithm. In *Advances in Neural Information Processing Systems*, pages 9318–9329.

Lyon, R. J., Stappers, B., Cooper, S., Brooke, J. M., and Knowles, J. D. (2016). Fifty years of pulsar candidate selection: from simple filters to a new principled real-time classification approach. *Monthly Notices of the Royal Astronomical Society*, 459(1):1104–1123.

Miech, A., Laptev, I., and Sivic, J. (2018). Learning a text-video embedding from incomplete and heterogeneous data. *arXiv preprint arXiv:1804.02516*.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2018). *Foundations of machine learning*. MIT press.

Möller, H. M. and Buchberger, B. (1982). The construction of multivariate polynomials with preassigned zeros. In *European Computer Algebra Conference*, pages 24–31. Springer.

Nesterov, Y. (1983). A method for unconstrained convex minimization problem with the rate of convergence o (1/kˆ 2). In *Doklady an ussr*, volume 269, pages 543–547.

Paty, F.-P. and Cuturi, M. (2019). Subspace robust wasserstein distances. In *International Conference on Machine Learning*, pages 5072–5081. PMLR.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830.

Pena, J. and Rodriguez, D. (2018). Polytope conditioning and linear convergence of the Frank-Wolfe algorithm. *Mathematics of Operations Research*.

Peyre, J., Sivic, J., Laptev, I., and Schmid, C. (2017). Weakly-supervised learning of visual relations. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5179–5188.

Ping, W., Liu, Q., and Ihler, A. T. (2016). Learning infinite rbms with Frank-Wolfe. In *Advances in Neural Information Processing Systems*, pages 3063–3071.

Shalev-Shwartz, S. and Wexler, Y. (2016). Minimizing the maximal loss: How and why. In *International Conference on Machine Learning*, pages 793–801. PMLR.

Suykens, J. A. and Vandewalle, J. (1999). Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300.

Tsanas, A., Little, M. A., Fox, C., and Ramig, L. O. (2013). Objective automatic assessment of rehabilitative speech treatment in parkinson's disease. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 22(1):181–190.

Tsuji, K., Tanaka, K., and Pokutta, S. (2021). Sparser kernel herding with pairwise conditional gradients without swap steps. *arXiv preprint arXiv:2110.12650*.

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

Zhang, X. (2018). Improvement on the vanishing component analysis by grouping strategy. *EURASIP Journal on Wireless Communications and Networking*, 2018(1):111.

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

A Proofs

A.1 Proof of Lemma 5.1.

Proof of Lemma 5.1. By Hilbert's basis theorem (Cox et al., 2013), there exists a finite set of generators $\mathcal{G} \subseteq \mathcal{P}$ of $\mathcal{I}_X$. We modify $\mathcal{G}$ such that it satisfies the property described in the lemma while remaining a set of generators of $\mathcal{I}_X$. Let $h = \sum_{i=1}^k c_i t_i \in \mathcal{G}$ for some $k \in \mathbb{N}$ and let t_j, $j \in [k]$, be the $<_\sigma$-largest term which is divisible by the leading term of any generator in $\mathcal{G}$ other than h, that is, $t_j \geq_\sigma t$ for all $t \in \{t_i \colon i \in [k], \exists g \in \mathcal{G}, g \neq h, \text{ such that } \operatorname{LT}(g) \mid t_i\}$. Let $g \in \mathcal{G}$ such that $\operatorname{LT}(g) \mid t_j$. Since both h and g vanish, so does $\bar{h} = h - \frac{c_j}{\operatorname{LTC}(g)} g$. Further, either none of the terms in $\bar{h}$ are divisible by leading terms of generators in $\mathcal{G}$ other than h, or the new $<_\sigma$-largest term in $\bar{h}$ divisible by the leading term of some generator in $\mathcal{G}$ other than h is $<_\sigma$-smaller than t_j. Repeatedly applying this procedure results in a formulation $h = \sum_i \alpha_i g_i + \tilde{h}$, where $\alpha_i \in \mathbb{R}$ for all $i \in [k]$ and $\tilde{h} \in \mathcal{P}$ is a vanishing polynomial none of whose terms are divisible by the leading terms of generators in $\mathcal{G}$ other than h. We can thus replace h in $\mathcal{G}$ by $\tilde{h}$ and the updated $\mathcal{G}$ is still a set of generators. Performing this procedure for all generators in $\mathcal{G}$, from the generator with $<_\sigma$-smallest leading term to the generator with $<_\sigma$-largest leading term, transforms $\mathcal{G}$ into a set of generators with the property described in the lemma. □

A.2 Proof of Theorem 6.1

Proof of Theorem 6.1. Let $i \in \mathbb{N}_{\geq 1}$. Since $X \subseteq [-1,1]^n$ is bounded, for any term $t \in \mathcal{T}_1$, we have that

$$\frac{1}{m} \left\| t^{i+2}(X) - t^i(X) \right\|_2^2 \leq 2(\max\{|t(\mathbf{x})| \colon \mathbf{x} \in X, |t(\mathbf{x})| < 1|\})^i.$$

As $i \to \infty$, the right-hand side tends to 0. Consider the polynomial $g := t^{i+2} - t^i$. Then, for any $\psi \geq \varepsilon \geq 0$, there exists $i^* \in \mathbb{N}$ such that

$$\operatorname{MSE}(g, X) = \frac{1}{m} \left\| t(X)^{i^*+2} - t(X)^{i^*} \right\|_2^2 \leq \varepsilon \leq \psi.$$

Thus, if no polynomial with leading term divisible by t is appended to $\mathcal{G}$ before reaching degree $i^* + 2$, `OAVI` appends a polynomial g with $\operatorname{LT}(g) = t^{i^*+2}$ to $\mathcal{G}$. The same observation holds for all other terms in $\mathcal{T}_1$ and, thus, there exists a degree $d^* \in \mathbb{N}$ such that for any $t \in \mathcal{T}_1$, there exists a polynomial $g \in \mathcal{G}$ with degree at most d^* and $t \mid \operatorname{LT}(g)$. Therefore, all terms in $\partial_{d^*n+1}\mathcal{O}$ are divisible by at least one leading term of a polynomial in $\mathcal{G}$ and $\partial_{d^*n+1}\mathcal{O} = \emptyset$, guaranteeing termination of `OAVI`. □

A.3 Proof of Theorem 6.2

Proof of Theorem 6.2. We prove the first statement. By Theorem 6.1, `OAVI` terminates with $\mathcal{G} \neq \emptyset$. Thus, by construction, at the end of `OAVI`'s execution, $\mathcal{G}$ contains $(\psi, 1)$-approximately vanishing polynomials. For the second statement, suppose towards a contradiction that there exists a $(\psi - \varepsilon, 1)$-approximately vanishing polynomial f with terms only in $\mathcal{O}$ and $\operatorname{LT}(f) = t \in \mathcal{O}$. Let $\mathcal{U} := \{u \in \mathcal{O} \mid t >_\sigma u\} = \{u_1, \ldots, u_k\}_\sigma$. At some point during its execution `OAVI` constructs a polynomial g with $\operatorname{LT}(g) = t$ and other terms only in $\mathcal{U}$. By ε-accuracy of `ORACLE`,

$$\operatorname{MSE}(g, X) \leq \min_{\mathbf{v} \in \mathbb{R}^k} \frac{1}{m} \left\| \mathcal{U}(X)\mathbf{v} + t(X) \right\|_2^2 + \varepsilon \leq \operatorname{MSE}(f, X) + \varepsilon \leq \psi - \varepsilon + \varepsilon \leq \psi,$$

a contradiction. □

A.4 Proof of Proposition 6.3

Proof of Proposition 6.3. Consider the execution of `OAVI` with X and $\psi \geq \varepsilon \geq 0$. Suppose that `OAVI` is currently inside the while loop corresponding to $\mathcal{O} = \{t_1, \ldots, t_m\}_\sigma$ and border $\partial_d\mathcal{O}$. Should this never occur, $|\mathcal{O}| \leq m$ is satisfied automatically. Let $\mathcal{R} \subseteq \mathcal{T}_d$ denote the remaining terms in $\partial_d\mathcal{O}$ for which `OAVI` has not yet checked whether they are leading terms of approximately vanishing polynomials. Since the m columns of $\mathcal{O}(X)$ are linearly independent, for any $u \in \mathcal{R}$, there exists $\mathbf{c} \in \mathbb{R}^m$ such that

$$\mathcal{O}(X)\mathbf{c} + u(X) = \mathbf{0}.$$

Hence, there exists a $(0, 1)$-approximately vanishing polynomial with leading term u and other terms only in $\mathcal{O}$. By ε-accuracy of ORACLE, $\psi \geq \varepsilon$, and Theorem 4.1, OAVI appends a $(\psi, 1)$-approximately vanishing polynomial with leading term u and other terms only in $\mathcal{O}$ to $\mathcal{G}$. Since this observation holds for any term in the remainder of the current border or the upcoming border, that is, $\mathcal{R}$ or $\partial_{d+1}\mathcal{O}$, respectively, no more terms get appended to $\mathcal{O}$. If no terms get appended to $\mathcal{O}$ for all terms in the border corresponding to a specific degree, then the border corresponding to the next higher degree is empty and OAVI terminates, proving the first statement.

For the second statement, suppose that OAVI terminates after performing the while loop corresponding to degree $D \in \mathbb{N}$. By construction, leading terms of polynomials in $\mathcal{G}$ are contained in $\bigcup_{d=1}^{D} \partial_d\mathcal{O}$. Hence, $|\mathcal{G}| \leq \left|\bigcup_{d=1}^{D} \partial_d\mathcal{O}\right| \leq |\mathcal{O}|n \leq mn$. □

A.5 Proof of Theorem 6.4

Proof of Theorem 6.4. By Theorem 6.1, OAVI is guaranteed to terminate. Suppose that OAVI terminates after performing the while loop corresponding to degree $D \in \mathbb{N}$. We first address the computational cost of OAVI, which is determined by two factors, the time required to construct the evaluation vectors of terms contained in any border, i.e., $\bigcup_{d=1}^{D} \partial_d\mathcal{O}$, and the time required to perform all the calls to ORACLE, which is exactly once for every term in $\bigcup_{d=1}^{D} \partial_d\mathcal{O}$.

We start with a brief overview of how the degree d border, $\partial_d\mathcal{O}$, is constructed in OAVI. First, we construct $\mathcal{C}_d = \{u \in \mathcal{T}_d \mid u = s \cdot t \text{ for } s \in \mathcal{O}_1 \text{ and } t \in \mathcal{O}_{d-1}\}$, a superset of $\partial_d\mathcal{O}$. Second, we determine $\mathcal{C}_d \setminus \partial_d\mathcal{O}$, that is, the set of terms $u \in \mathcal{C}_d$ such that there exists a polynomial $g \in \mathcal{G}_{\leq d-1}$ with $\mathrm{LT}(g) \mid u$. Third, we remove $\mathcal{C}_d \setminus \partial_d\mathcal{O}$ from $\mathcal{C}_d$ to obtain $\partial_d\mathcal{O}$. Fourth, we evaluate all terms in $\partial_d\mathcal{O}$.

We now show that it requires time $O(m^2n^2)$ to construct $\bigcup_{d=1}^{D} \partial_d\mathcal{O}$ and the evaluation vectors of terms therein. Note that the evaluation vectors of terms in $\partial_1\mathcal{O}$ are identical to the elements of input X of OAVI and thus require time $O(1)$ to construct. We thus consider any $d \in \{2, \ldots, D\}$ and suppose that $\mathcal{O}_i$, $\mathcal{G}_i$, and $\partial_i\mathcal{O}$ and the evaluation vectors of $\mathcal{O}_i$, $\mathcal{G}_i$, and $\partial_i\mathcal{O}$ are already determined for all $i \leq d-1$. First, assuming that multiplying two terms requires constant time, to construct $\mathcal{C}_d$, every term in $\mathcal{O}_{d-1}$ gets multiplied by all terms in $\mathcal{O}_1$ exactly once, requiring a total of $O(|\mathcal{O}_{d-1}|n)$ multiplications of terms. Second, assuming that it takes constant time to determine whether a term is divisible by another term, it requires time $O(|\mathcal{C}_d||\mathcal{G}_{\leq d-1}|) \leq O(|\mathcal{C}_d||\mathcal{G}|)$ to determine all terms in $\mathcal{C}_d$ that have to be deleted. Third, assuming that a term can be deleted in constant time, it takes $O(|\mathcal{C}_d|)$ time to perform all deletions of terms in $\mathcal{C}_d$. Fourth, evaluating a term $t \in \partial_d\mathcal{O}$ requires entry-wise multiplication of two m-dimensional evaluation vectors of terms in $\mathcal{O}_{\leq d-1}$, requiring time $O(m)$. Since we have to evaluate at most $|\mathcal{C}_d|$ terms, the evaluation step requires time $O(|\mathcal{C}_d|m)$. Thus, it requires time $O(|\mathcal{C}_d|(|\mathcal{G}| + 1 + m)) = O(|\mathcal{C}_d||\mathcal{G}|)$ to construct the evaluation vectors of terms in $\partial_d\mathcal{O}$, where equality follows due to Proposition 6.3. Thus, to construct $\bigcup_{d=1}^{D} \partial_d\mathcal{O}$ and the evaluation vectors of terms therein, it requires time $O(\sum_{d=1}^{D} |\mathcal{C}_d||\mathcal{G}|) = O(|\mathcal{C}||\mathcal{G}|) = O(m^2n^2)$, where the first equality holds because the sets $\mathcal{C}_d$ for $d \in [D]$ are pairwise disjoint and the second equality holds due to $O(|\mathcal{C}|) = O(|\mathcal{O}||\mathcal{O}_1|) = O(mn)$ and Proposition 6.3.

We next determine the time complexity of the ORACLE calls during OAVI's execution. ORACLE is called exactly once for every border term, i.e., $O(|\bigcup_{d=1}^{D} \partial_d\mathcal{O}|) = O(|\mathcal{G}| + |\mathcal{O}|) = O(mn)$ times, requiring time $O(mnT_{\texttt{ORACLE}})$. Thus, OAVI's total time complexity is $O(m^2n^2 + mnT_{\texttt{ORACLE}})$.

Throughout the algorithm's execution, we store $\mathcal{O}$, $\mathcal{G}$, and $\mathcal{C}$, and the evaluation vectors of terms therein. Assuming that we can store a term $t \in \mathcal{T}$ and the corresponding evaluation vector $t(X) \in \mathbb{R}^m$ in space $O(m)$, OAVI requires space $O((|\mathcal{C}|+|\mathcal{G}|+|\mathcal{O}|)m) = O(m^2n)$. To store the generators in $\mathcal{G}$, we further store $|\mathcal{G}|$ vectors of length $O(m)$. Since the space complexity of ORACLE affects the total space complexity only additively, the total space complexity is $O(m^2n + S_{\texttt{ORACLE}})$. □

A.6 Proof of Theorem 6.5

Proof of Theorem 6.5. The proof is an adaptation of the proof of Theorem 5.1 in Livni et al. (2013) to OAVI. Let $\mathcal{O} = \{t_1, \ldots, t_k\}_\sigma$ with $k \leq m$. Since $t_1 = 1$, we can evaluate it in time $O(1)$. Since the evaluation vectors of monomials in $\mathcal{O}_1$ over Z are identical to the data points $\mathbf{z} \in Z$, all degree 1 monomials can be evaluated in time $O(1)$. The evaluation of a term $t \in \mathcal{T}$ of degree greater than 1 over Z can be computed by multiplying the evaluation vectors of two q-dimensional monomials element-wise, requiring time $O(q)$. Hence, we can construct

the evaluation vectors of all monomials in $\mathcal{O}$ over Z in time $O(|\mathcal{O}|+|\mathcal{O}|q) = O(|\mathcal{O}|q)$. Since the evaluation vectors of leading terms of generators in $\mathcal{G}$ are element-wise multiplications of evaluation vectors in $\mathcal{O}$, we can construct all leading terms of generators in $\mathcal{G}$ in time $O(|\mathcal{G}|q + |\mathcal{O}|q) = O(|\mathcal{G}|q)$. After evaluating all monomials in $\mathcal{O}$ and $\mathcal{G}$, generators in $\mathcal{G}$ are linear combinations of at most $|\mathcal{O}| + 1$ evaluation vectors of terms. The computation of the linear combinations requires time $O(|\mathcal{G}||\mathcal{O}|q)$. Thus, the total evaluation complexity is $O(|\mathcal{G}||\mathcal{O}|q)$. □

A.7 Proof of Theorem 6.6

Proof of Theorem 6.6. Let $\tau \geq 2$ and for $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ and $\psi \geq \varepsilon \geq 0$, let $(\mathcal{G}, \mathcal{O}) = \texttt{OAVI}(X, \psi, \varepsilon)$ and suppose that all $g \in \mathcal{G}$ are τ-bounded in $\|\cdot\|_1$. Consider any $(\psi, 1)$-approximately vanishing polynomial over X, $g \in \mathcal{G}$, with $\operatorname{LT}(g) = t$ and other terms only in $\mathcal{U} = \{u_1, \ldots, u_k\}_\sigma \subseteq \mathcal{O}$, where $k \leq m$. Recall that

$$\mathcal{U}(\mathbf{x}) = (u_1(\mathbf{x}), \ldots, u_k(\mathbf{x}))^\intercal \in \mathbb{R}^k$$

and let

$$\mathcal{H} := \{\mathbf{x} \in \mathcal{X} \mapsto \mathbf{w}^\intercal \mathcal{U}(\mathbf{x}) \colon \|\mathbf{w}\|_1 \leq \tau - 1\}\,.$$

The next part of the proof repeats the arguments used in Mohri et al. (2018, Theorem 11.15). We adapt the notation therein to our setting and show that the empirical Rademacher complexity of $\mathcal{H}$ is bounded. Let $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_k)^\intercal \in \{-1, 1\}^k$ be a vector of uniform random variables. Then, for the empirical Rademacher complexity $\hat{\mathcal{R}}_X(\mathcal{H})$, it holds that

$$\begin{aligned}
\hat{\mathcal{R}}_X(\mathcal{H}) &= \frac{1}{m}\mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{\|\mathbf{w}\|_1 \leq \tau - 1} \sum_{i=1}^m \sigma_i \mathbf{w}^\intercal \mathcal{U}(\mathbf{x}_i)\right] \\
&= \frac{\tau - 1}{m}\mathbb{E}_{\boldsymbol{\sigma}}\left[\left\|\sum_{i=1}^m \sigma_i \mathcal{U}(\mathbf{x}_i)\right\|_\infty\right] && \triangleright \text{ by definition of the dual norm} \\
&= \frac{\tau - 1}{m}\mathbb{E}_{\boldsymbol{\sigma}}\left[\max_{j \in [k]}\left|\sum_{i=1}^m \sigma_i \mathcal{U}(\mathbf{x}_i)_j\right|\right] && \triangleright \text{ by definition of } \|\cdot\|_\infty \\
&= \frac{\tau - 1}{m}\mathbb{E}_{\boldsymbol{\sigma}}\left[\max_{j \in [k]} \max_{s \in \{-1,1\}} s\sum_{i=1}^m \sigma_i \mathcal{U}(\mathbf{x}_i)_j\right] && \triangleright \text{ by definition of } \|\cdot\|_\infty \\
&= \frac{\tau - 1}{m}\mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{\mathbf{z} \in A} \sum_{i=1}^m \sigma_i z_i\right],
\end{aligned}$$

where $A = \{s(\mathcal{U}(\mathbf{x}_1)_j, \ldots, \mathcal{U}(\mathbf{x}_m)_j)^\intercal \colon j \in [k], s \in \{-1, 1\}\}$. For all $\mathbf{x} \in [-1, 1]^n$, we have $\|\mathcal{U}(\mathbf{x})\|_\infty \leq 1$. Thus, for any $\mathbf{z} \in A$, we have $\|\mathbf{z}\|_2 \leq \sqrt{m}$. By Mohri et al. (2018, Theorem 3.7), since A contains at most $2k \leq 2m$ elements, we have

$$\hat{\mathcal{R}}_X(\mathcal{H}) \leq \tau\sqrt{\frac{2\log(2m)}{m}}.$$

To conclude the proof, note that for all $h \in \mathcal{H}$ and $(\mathbf{x}, y) \in (\mathcal{X} \times \mathcal{Y})$, we have $|h(\mathbf{x}) + t(\mathbf{x})| \leq \tau - 1 + 1 = \tau$, and apply Mohri et al. (2018, Theorem 11.3) to the bound on $\hat{\mathcal{R}}_X(\mathcal{H})$, similarly to Mohri et al. (2018, Theorem 11.16). □

B Generalization bound for OAVI feature transformations for a linear kernel SVM

In this section, we derive a margin bound for the approach of using τ- and degree-bounded generators of the approximately vanishing ideal to transform features for a linear kernel SVM.

To do so, we recall some definitions from Mohri et al. (2018). Let $\mathcal{X} \subseteq [-1, 1]^n$ and $\mathcal{Y} = \{0, 1\}$ and recall that a hypothesis h is a mapping from $\mathcal{X}$ to $\mathcal{Y}$.

Definition B.1 (Margin loss function)**.** For any $\rho > 0$, the *ρ-margin loss* is the function $L_\rho \colon \mathbb{R} \times \mathbb{R} \to \mathbb{R}_{>0}$ defined for all $y, y' \in \mathbb{R}$ as

$$L_\rho(y, y') = \min\left\{1, \max\left\{0, 1 - \frac{yy'}{\rho}\right\}\right\}.$$

The margin loss functions is used to define the empirical margin loss, i.e., the error on the training set.

Definition B.2 (Empirical margin loss)**.** Let $\mathcal{X} \subseteq [-1,1]^n$, $\mathcal{Y} = \{0,1\}$, $\mathcal{D}$ be a distribution over $\mathcal{X} \times \mathcal{Y}$, $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\} \in (\mathcal{X} \times \mathcal{Y})^m$ be sampled according to $\mathcal{D}$, and h be a hypothesis. The *empirical margin loss* of h is defined as

$$\hat{R}_{S,\rho}(h) = \frac{1}{m} \sum_{i=1}^{m} L_\rho(h(\mathbf{x}_i), y_i).$$

In the theorem below, we present a margin bound for the approach of using τ- and degree-bounded generators of the approximately vanishing ideal to transform features for a linear kernel `SVM`.

Theorem B.3 (Margin bound for feature transformations for a linear kernel `SVM`)**.** *Let $\mathcal{X} \subseteq [-1,1]^n$, $\mathcal{Y} = \{0,1\}$, $\mathcal{D}$ be a distribution over $\mathcal{X} \times \mathcal{Y}$, $\mathcal{G} = \{g_1, \ldots, g_k\} \subseteq \mathcal{P}$ such that $|g(\mathbf{x}))| \leq \tau$ for all $g \in \mathcal{G}$ and $\mathbf{x} \in \mathcal{X}$, $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\} \in (\mathcal{X} \times \mathcal{Y})^m$ be sampled according to $\mathcal{D}$, and $\tilde{S} = \{(\mathcal{G}(\mathbf{x}), y) \mid (\mathbf{x}, y) \in S\}$. Let $K \colon [-\tau, \tau]^k \times [-\tau, \tau]^k \to \mathbb{R}$ denote the linear kernel and $\mathcal{H} = \{\mathbb{R}^k \ni \mathbf{v} \mapsto \mathbf{w}^\intercal \mathbf{v} \colon \|\mathbf{w}\|_2 \leq \Lambda\}$ for some $\Lambda \geq 0$. Fix $\rho > 0$. Then, for any $\delta > 0$, with probability at least $1 - \delta$, for any $h \in \mathcal{H}$, we have*

$$\mathbb{P}_{(\mathbf{x},y) \backsim \mathcal{D}}[h(\mathcal{G}(\mathbf{x})) \neq y] \leq \hat{R}_{\tilde{S},\rho}(h) + 2\sqrt{\frac{k\tau^2\Lambda^2}{\rho^2 m}} + \sqrt{\frac{\log(\delta^{-1})}{2m}}.$$

Proof. Let $\tilde{\mathcal{X}} = \{\mathcal{G}(\mathbf{x}) \mid \mathbf{x} \in \mathcal{X}\}$. Given a distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, let $\tilde{\mathcal{D}}$ denote an auxiliary distribution, samples $(\tilde{\mathbf{x}}, y) \backsim \tilde{\mathcal{D}}$ of which are obtained by drawing a sample $(\mathbf{x}, y) \backsim \mathcal{D}$ and letting $\tilde{\mathbf{x}} := \mathcal{G}(\mathbf{x})$. Then, for the linear kernel $K \colon \tilde{\mathcal{X}} \times \tilde{\mathcal{X}} \to \mathbb{R}$, we have

$$\sup_{\tilde{\mathbf{x}} \in \tilde{\mathcal{X}}} K(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}) = \langle \tilde{\mathbf{x}}, \tilde{\mathbf{x}} \rangle \leq k\tau^2.$$

We then apply the generalization bound for kernel-based classifiers (Mohri et al., 2018, Corollary 6.13) to data sampled according to $\tilde{\mathcal{D}}$, proving the theorem. □

For $\mathcal{X} \subseteq [-1,1]^n$ and any set of τ-bounded polynomials $\mathcal{G} \subseteq \mathcal{P}$ in norm $\|\cdot\|_1$, it holds that $|g(\mathbf{x})| \leq \tau$ for all $\mathbf{x} \in \mathcal{X}$ and $g \in \mathcal{G}$. Thus, using the notation of Section 7, if we apply a linear kernel `SVM` to the data set transformed according to the feature transformation (FT) corresponding to a set of τ-bounded generators in norm $\|\cdot\|_1$ constructed by `OAVI`, `AVI`, or `VCA`, Theorem B.3 is satisfied. However, the margin bound in Theorem B.3 relies on the number of polynomials in $\mathcal{G} = \bigcup_i \mathcal{G}^i$ to be $o(m)$, where $\mathcal{G}^i$ is the set of generators constructed for class i. This can be achieved by bounding the number of generators used to transform the features of the original data set, for example, via an upper bound on the maximum degree of generators.

To summarize, since generators constructed with `CGAVI` are τ-bounded in norm $\|\cdot\|_1$, whereas generators constructed with `AVI` or `VCA` are not necessarily τ-bounded in norm $\|\cdot\|_1$, Theorem B.3 applies to `CGAVI` when the algorithm is terminated after reaching a certain degree, but, in general, does not apply to `AVI` or `VCA`. Thus, if we apply a linear kernel `SVM` to the data set transformed according to the feature transformation (FT) corresponding to a set of generators of bounded degree constructed with `CGAVI`, increasing the number of training samples and decreasing the training error decreases the probability of misclassifying out-sample data.

C The `PFW` algorithm

The Pairwise Frank-Wolfe algorithm `PFW` is presented in Algorithm 3. Let $f \colon \mathbb{R}^n \to \mathbb{R}$ be a convex and smooth function and $\mathcal{A} = \{\mathbf{v}_1, \ldots, \mathbf{v}_k\} \subseteq \mathbb{R}^n$ a set of vectors. Then, `PFW` solves

$$\min_{\mathbf{x} \in \operatorname{conv}(\mathcal{A})} f(\mathbf{x}),$$

where $\operatorname{conv}(\mathcal{A})$ is the convex hull of $\mathcal{A}$. At iteration $t = 0, \ldots, T$, `PFW` writes the current iterate, $\mathbf{x}_t$, as a convex combination of elements of $\mathcal{A}$, that is, $\mathbf{x}_t = \sum_{\mathbf{v} \in \mathcal{A}} \lambda_{\mathbf{v}}^{(t)} \mathbf{v}$, where $\sum_{\mathbf{v} \in \mathcal{A}} \lambda_{\mathbf{v}}^{(t)} = 1$ and $\lambda_{\mathbf{v}}^{(t)} \in [0,1]$ for all $\mathbf{v} \in \mathcal{A}$. At iteration t, a vertex $\mathbf{v}$ whose corresponding weight $\lambda_{\mathbf{v}}^{(t)}$ is not zero is referred to as an *active vertex* and the set $S^{(t)} = \{\mathbf{v} \in \mathcal{A} \mid \lambda_{\mathbf{v}}^{(t)} > 0\}$ is referred to as the *active set* at iteration t. During each iteration, `PFW` determines

Algorithm 3 PFW

Input: A smooth and convex function f, a set of atoms $\mathcal{A} \subseteq \mathbb{R}^n$, a vertex $\mathbf{x}_0 \in \mathcal{A}$, and $T \in \mathbb{N}$.
Output: A point $\mathbf{x}_T \in \operatorname{conv}(\mathcal{A})$.

1: $S^{(0)} \leftarrow \{\mathbf{x}_0\}$
2: $\lambda_{\mathbf{v}}^{(0)} \leftarrow 1$ for $\mathbf{v} = \mathbf{x}_0$ and 0 otherwise
3: **for** $t = 0, \ldots, T$ **do**
4: $\quad \mathbf{s}_t \leftarrow \operatorname{argmin}_{\mathbf{s} \in \mathcal{A}} \langle \nabla f(\mathbf{x}_t), \mathbf{s} \rangle$
5: $\quad \mathbf{d}_t^{FW} \leftarrow \mathbf{s}_t - \mathbf{x}_t$
6: $\quad \mathbf{v}_t \leftarrow \operatorname{argmax}_{\mathbf{v} \in S^{(t)}} \langle \nabla f(\mathbf{x}_t), \mathbf{v} \rangle$
7: $\quad \mathbf{d}_t^{AW} \leftarrow \mathbf{x}_t - \mathbf{v}_t$
8: $\quad \mathbf{d}_t \leftarrow \mathbf{d}_t^{FW} + \mathbf{d}_t^{AW}$
9: $\quad \gamma_t = \operatorname{argmin}_{\gamma \in [0, \lambda_{\mathbf{v}_t}]} f(\mathbf{x}_t + \gamma \mathbf{d}_t)$
10: $\quad \mathbf{x}_{t+1} \leftarrow \mathbf{x}_t + \gamma_t \mathbf{d}_t$
11: $\quad \lambda_{\mathbf{s}_t}^{(t+1)} \leftarrow \lambda_{\mathbf{s}_t}^{(t)} + \gamma_t$
12: $\quad \lambda_{\mathbf{v}_t}^{(t+1)} \leftarrow \lambda_{\mathbf{v}_t}^{(t)} - \gamma_t$
13: $\quad S^{(t+1)} \leftarrow \{\mathbf{v} \in \mathcal{A} \mid \lambda_{\mathbf{v}}^{(t+1)} > 0\}$
14: **end for**

two vertices requiring access to a first-order oracle and a linear minimization oracle. In Line 4, PFW determines the Frank-Wolfe vertex, which minimizes the scalar product with the gradient of f at iterate x_t. Taking a step of appropriate size towards the Frank-Wolfe vertex, in the Frank-Wolfe direction, reduces the objective function value. In Line 7, PFW determines the Away vertex in the active set, which maximizes the scalar product with the gradient of f at iterate x_t. Taking a step away from the Away vertex, in the Away direction, reduces the objective function value. In Line 8, PFW combines the Away direction and the Frank-Wolfe direction into the Pairwise direction and takes a step with optimal step size in the Pairwise direction, shifting weight from the Away vertex to the Frank Wolfe vertex. In each iteration, PFW thus only modifies two entries of the iterate $\mathbf{x}_t$, which is the main reason why PFW tends to return a sparse iterate $\mathbf{x}_T$. The iterate $\mathbf{x}_T$ returned by PFW satisfies $f(\mathbf{x}_T) - \min_{\mathbf{x} \in \operatorname{conv}(\mathcal{A})} f(\mathbf{x}) = O(e^{-T})$ (Lacoste-Julien and Jaggi, 2015) when optimizing the squared loss over the ℓ_1-ball. However, PFW's linear convergence rate is theoretically cumbersome to bound due to its dependence on various constants and the number of so-called swap steps when a vertex in the active set gets replaced by another vertex. To address this issue, one could replace PFW with a swap-step free version called *Blended Pairwise Conditional Gradients*, whose convergence rate constants are smaller than those of PFW (Tsuji et al., 2021). In practice, instead of computing the number of iterations to reach ε-accuracy, we use the Frank-Wolfe gap as a stopping criterion for PFW and we do not have to compute T.

When using PFW as ORACLE for OAVI, we implement ORACLE in Line 6 as follows: Run PFW with f, $\mathcal{A}$ the set of vertices of the ℓ_1-ball of radius $\tau - 1$, $\mathbf{x}_0 = (1, 0 \ldots, 0)^\intercal$, and $T \in \mathbb{N}$ such that PFW achieves ε-accuracy and obtain $\mathbf{x}_T$. Then, $(\mathbf{x}_T^\intercal, 1)^\intercal$ is the coefficient vector of the polynomial g returned by ORACLE.

D Additional numerical experiments

We use different algorithms that construct generators of the approximately vanishing ideal to transform features for a linear kernel SVM and compare them to each other and a polynomial kernel SVM.

As in Section 9, let $\mathcal{G} := \bigcup_i \mathcal{G}^i$, where $\mathcal{G}^i$ is the set of generators constructed for class i and let $\mathcal{O}^i$ denote the set of terms that are not leading terms of generators constructed for class i.[8] By Theorem 6.5, $(\mathcal{G}^i, \mathcal{O}^i) = \texttt{OAVI}(X, \psi, \varepsilon)$ for $X \subseteq [-1, 1]^n$ and $\psi \geq \varepsilon \geq 0$, the time required to evaluate a test set of q data points is $O(|\mathcal{G}^i||\mathcal{O}^i|q)$. It thus remains to compare $\sum_i |\mathcal{O}^i|$, the number of terms that are provably not leading terms of generators summed over all classes for the monomial-based algorithms, ℓ_1-CGAVI, ℓ_2-CGAVI, AGDAVI, and AVI, and the number of non-approximately vanishing polynomials constructed for the polynomial-based algorithm, VCA.

Set-up. We apply different feature transformation algorithms for a linear kernel SVM and a polynomial kernel SVM to classification problems using the set-up from Section 9. We compare the number of terms constructed,

[8] Recall that for VCA, $\mathcal{O}^i$ is a set of non-approximately vanishing polynomials instead.

Table 3: We compare the number of terms constructed, $\sum_i |\mathcal{O}^i|$, the maximum degree of generators constructed, Degree, the number of entries in the coefficient vectors of generators in $\mathcal{G}$, $\sum_{g\in\mathcal{G}} g_e$, the hyperparameter optimization time in seconds, Hyp. time, the training time in seconds, Train time, and the classification error on the training set in %, Train error. The results are averaged over ten random 60%/40% train/test partitions and the best results in each category are in bold.

Algorithms		Data sets								
		bank	cancer	htru2	iris	seeds	sonar	spam	voice	wine
$\sum_i \lvert\mathcal{O}^i\rvert$	ℓ_1-CGAVI	18	17	6	**5**	7	**34**	5	11	**20**
	ℓ_2-CGAVI	15	11	8	6	**5**	43	5	7	42
	AGDAVI	23	15	**5**	7	7	53	6	**3**	32
	AVI	8	37	42	12	17	71	**4**	38	46
	VCA	**4**	**8**	13	34	67	91	64	42	49
Degree	ℓ_1-CGAVI	4.30	2.30	2.20	**1.90**	1.70	3.10	1.90	2.50	2.90
	ℓ_2-CGAVI	3.80	1.80	2.60	2.00	**1.40**	3.50	2.30	2.50	4.80
	AGDAVI	5.20	2.10	**2.00**	2.20	2.00	4.30	2.70	**1.40**	3.10
	AVI	2.20	2.20	3.00	2.30	2.00	**2.40**	**1.40**	2.00	**2.20**
	VCA	**1.50**	**1.70**	2.20	3.60	3.60	3.10	2.00	2.20	3.00
$\sum_g g_e$	ℓ_1-CGAVI	1.6e+2	2.6e+3	9.0e+1	**3.6e+1**	1.3e+2	**6.1e+3**	**3.9e+2**	3.6e+3	**6.9e+2**
	ℓ_2-CGAVI	1.3e+2	**1.3e+3**	1.2e+2	4.0e+1	**7.8e+1**	9.5e+3	4.5e+2	2.5e+3	2.0e+3
	AGDAVI	2.2e+2	1.8e+3	**7.3e+1**	5.0e+1	1.3e+2	1.2e+4	4.1e+2	**1.8e+3**	1.6e+3
	AVI	**7.2e+1**	1.3e+4	2.4e+3	1.3e+2	4.1e+2	6.6e+4	4.1e+2	2.4e+4	4.8e+3
	VCA	1.7e+2	8.4e+3	4.7e+3	4.3e+3	3.8e+4	4.5e+6	4.0e+6	1.7e+6	9.0e+4
Hyp. time	ℓ_1-CGAVI	2.5e+2	6.4e+2	5.7e+2	4.9e+2	8.4e+2	1.1e+3	7.1e+2	2.3e+3	1.2e+3
	ℓ_2-CGAVI	1.9e+2	7.2e+2	5.7e+2	3.1e+2	5.8e+2	1.5e+3	6.6e+2	3.8e+3	9.8e+2
	AGDAVI	8.0e+1	2.7e+2	5.2e+2	1.1e+2	1.8e+2	4.9e+2	5.3e+2	1.1e+3	3.4e+2
	AVI	4.7e+1	4.8e+2	9.2e+2	3.4e+1	7.5e+1	5.7e+2	1.5e+3	1.1e+3	1.7e+2
	VCA	4.7e+1	5.7e+2	4.3e+2	4.4e+1	7.4e+1	4.8e+2	1.7e+3	4.6e+2	1.4e+2
	SVM	**2.3e-1**	**1.7e-**1	**9.9e+1**	**1.0e-1**	**1.1e-1**	**1.3e-1**	**1.3e+1**	**1.2e-1**	**1.0e-1**
Train time	ℓ_1-CGAVI	3.2e-1	9.9e-1	1.2e+0	5.9e-2	2.0e-1	2.1e+0	9.9e-1	2.5e+0	5.9e-1
	ℓ_2-CGAVI	2.1e-1	7.6e-1	1.4e+0	6.2e-2	9.9e-2	3.7e+0	9.9e-1	3.5e+0	1.2e+0
	AGDAVI	1.4e-1	2.9e-1	1.0e+0	4.0e-2	6.8e-2	1.5e+0	8.6e-1	7.9e-1	3.4e-1
	AVI	3.8e-2	4.2e-1	5.0e+0	4.8e-2	1.0e-1	1.6e+0	8.4e-1	1.3e+0	3.1e-1
	VCA	1.3e-2	4.7e-2	8.6e-1	8.3e-2	1.8e-1	1.0e+0	2.1e+0	5.1e-1	2.0e-1
	SVM	**2.5e-3**	**1.4e-3**	**1.1e+0**	**4.7e-4**	**7.6e-4**	**1.1e-3**	**2.3e-1**	**8.3e-4**	**6.1e-4**
Train error	ℓ_1-CGAVI	**0.00**	1.32	2.00	1.89	0.56	0.24	**4.74**	1.87	0.09
	ℓ_2-CGAVI	**0.00**	1.35	2.00	**1.22**	0.56	1.13	4.91	1.47	0.28
	AGDAVI	0.01	**1.26**	2.00	**1.22**	1.43	0.40	4.75	2.40	0.09
	AVI	**0.00**	1.29	1.99	1.67	1.03	0.65	5.34	4.53	0.09
	VCA	**0.00**	1.61	2.05	1.56	**0.32**	**0.00**	5.43	**0.00**	**0.00**
	SVM	**0.00**	1.61	**1.93**	1.44	1.83	3.31	6.53	1.87	0.85

$\sum_i |\mathcal{O}^i|$, the maximum degree of generators constructed, Degree, the number of entries in the coefficient vectors of generators in $\mathcal{G}$, $\sum_{g\in\mathcal{G}} g_e$, the hyperparameter optimization time in seconds, Hyp. time, the training time, that is, the time required to perform the retraining with the best hyperparameter combination, in seconds, Train time, and the classification error on the training set in %, Train error.

Implementation, data sets, and hyperparameters. Implementation, data sets, and hyperparameters are the same as in Section 9. For the polynomial kernel SVM, $\sum_i |\mathcal{O}^i|$, Degree, and $\sum_{g\in\mathcal{G}} g_e$ do not exist.

Results.
$\sum_i |\mathcal{O}^i|$: The three OAVI algorithms construct a similar number of terms that are not leading terms of generators for all data sets and for htru2, iris, seeds, sonar, voice, and wine, construct fewer terms that are not leading terms of generators than AVI and VCA.
Degree: Even though we use ℓ_2-regularized squared loss for OAVI, the maximum degrees of the generators constructed by OAVI are comparable to the maximum degrees of generators constructed by AVI and VCA.
$\sum_{g\in\mathcal{G}} g_e$: For all data sets but bank, one of the OAVI algorithms creates the feature transformation with the smallest number of entries in the coefficient vectors of generators in $\mathcal{G}$. For most data sets, VCA constructs the feature transformation with the largest number of entries in the coefficient vectors of generators in $\mathcal{G}$ and for seeds, sonar, spam, and voice, VCA constructs multiple orders of magnitude more entries in the coefficient vectors

of generators in $\mathcal{G}$ than the other algorithms.
Hyp. time: All algorithms except the `SVM` achieve comparable hyperparameter optimization times. For all data sets, tuning the hyperparameters for `AGDAVI` requires less time than tuning the hyperparameters for ℓ_1- and ℓ_2-`CGAVI`. For all data sets, the `SVM` requires the least amount of time for hyperparameter optimization.
Train time: All algorithms except the `SVM` achieve comparable training times. For all data sets, training `AGDAVI` requires less time than training ℓ_1- and ℓ_2-`CGAVI`. For five of the nine data sets, `VCA` can be trained faster than `OAVI` and `AVI`. For all data sets, the `SVM` requires the least amount of time for training.
Train error: All algorithms achieve comparable errors on the training data. However, it is noteworthy that `VCA` achieves perfect training accuracy on four of the nine data sets.