Abstract

The vanishing ideal of a set of points \( X = \{x_1, \ldots, x_m\} \subseteq \mathbb{R}^n \) is the set of polynomials that evaluate to 0 over all points \( x \in X \) and admits an efficient representation by a finite subset of generators. In practice, to accommodate noise in the data, algorithms that construct generators of the approximate vanishing ideal are widely studied but their computational complexities remain expensive. In this paper, we scale up the oracle approximate vanishing ideal algorithm (OAVI), the only generator-constructing algorithm with known learning guarantees. We prove that the computational complexity of OAVI is not superlinear, as previously claimed, but linear in the number of samples \( m \). In addition, we propose two modifications that accelerate OAVI’s training time: Our analysis reveals that replacing the pairwise conditional gradients algorithm, one of the solvers used in OAVI, with the faster blended pairwise conditional gradients algorithm leads to an exponential speed-up in the number of features \( n \). Finally, using a new inverse Hessian boosting approach, intermediate convex optimization problems can be solved almost instantly, improving OAVI’s training time by multiple orders of magnitude in a variety of numerical experiments.

1. Introduction

High-quality features are essential for the success of machine-learning algorithms (Guyon and Elisseeff, 2003) and as a consequence, feature transformation and selection algorithms are an important area of research (Kusiak, 2001; Van Der Maaten et al., 2009; Abdi and Williams, 2010; Paul et al., 2021; Manikandan and Abirami, 2021; Carderera et al., 2021). A recently popularized technique for extracting nonlinear features from data is the concept of the vanishing ideal (Heldt et al., 2009; Livni et al., 2013), which lies at the intersection of machine learning and computer algebra. Unlike conventional machine learning, which relies on a manifold assumption, vanishing ideal computations are based on an algebraic set assumption, for which powerful theoretical guarantees are known (Vidal et al., 2005; Livni et al., 2013; Globerson et al., 2017). The core concept of vanishing ideal computations is that any data set \( X = \{x_1, \ldots, x_m\} \subseteq \mathbb{R}^n \) can be described by its vanishing ideal,

\[
I_X = \{g \in \mathcal{P} \mid g(x) = 0 \text{ for all } x \in X\},
\]

where \( \mathcal{P} \) is the polynomial ring over \( \mathbb{R} \) in \( n \) variables. Despite \( I_X \) containing infinitely many polynomials, there exists a finite number of generators of \( I_X \), \( g_1, \ldots, g_k \in I_X \) with \( k \in \mathbb{N} \), such that any polynomial \( h \in I_X \)

1. A set \( X \subseteq \mathbb{R}^n \) is algebraic if it is the set of common roots of a finite set of polynomials.
can be written as
\[ h = \sum_{i=1}^{k} g_i h_i, \]
where \( h_i \in \mathcal{P} \) for all \( i \in \{1, \ldots, k\} \) (Cox et al., 2013). Thus, the generators share any sample \( \mathbf{x} \in X \) as a common root, capture the nonlinear structure of the data, and succinctly represent the vanishing ideal \( I_X \).

Due to noise in empirical data, we are interested in constructing generators of the *approximate vanishing ideal*, the ideal generated by the set of polynomials that approximately evaluate to 0 for all \( \mathbf{x} \in X \) and whose leading term coefficient is 1, see Definition 2.2. For classification tasks, constructed generators can, for example, be used to transform the features of the data set \( X \subseteq \mathbb{R}^n \) such that the data becomes linearly separable (Livni et al., 2013) and training a linear kernel *support vector machine* (SVM) (Suykens and Vandewalle, 1999) on the feature-transformed data results in excellent classification accuracy. Various algorithms for the construction of generators of the approximate vanishing ideal exist (Heldt et al., 2009; Fassino, 2010; Limbeck, 2013; Livni et al., 2013; Iraji and Chitsaz, 2017; Kera and Hasegawa, 2020; Kera, 2022), but among them, the *oracle approximate vanishing ideal algorithm* (OAVI) (Wirth and Pokutta, 2022) is the only one capable of constructing sparse generators and admitting learning guarantees. More specifically, CGAVI (OAVI with Frank-Wolfe algorithms (Frank and Wolfe, 1956), a.k.a. *conditional gradients algorithms* (CG) (Levitin and Polyak, 1966) as a solver) exploits the sparsity-inducing properties of CG to construct sparse generators and, thus, a robust and interpretable corresponding feature transformation. Furthermore, generators constructed with CGAVI vanish on out-sample data and the combined approach of transforming features with CGAVI for a subsequently applied linear kernel SVM inherits the margin bound of the SVM (Wirth and Pokutta, 2022).

Despite OAVI’s various appealing properties, the computational complexities of vanishing ideal algorithms for the construction of generators of the approximate vanishing ideal are superlinear in the number of samples \( m \). With training times that increase at least cubically with \( m \), vanishing ideal algorithms have yet to be applied to large-scale machine-learning problems.

### 1.1 Contributions

In this paper, we improve and study the scalability of OAVI.

**Linear computational complexity in \( m \).** Up until now, the analysis of computational complexities of approximate vanishing ideal algorithms assumed that generators need to vanish exactly, which gave an overly pessimistic estimation of the computational cost. For OAVI, we exploit that generators only have to vanish approximately and prove that the computational complexity of OAVI is not superlinear but linear in the number of samples \( m \) and polynomial in the number of features \( n \).

**Solver improvements.** OAVI repeatedly calls a solver of quadratic convex optimization problems to construct generators. By replacing the *pairwise conditional gradients algorithm* (PCG) (Lacoste-Julien and Jaggi, 2015) with the faster *blended pairwise conditional gradients algorithm* (BPCG) (Tsuji et al., 2022), we improve the dependence of the time complexity of OAVI on the number of features \( n \) in the data set by an exponential factor.

**Inverse Hessian boosting (IHB).** OAVI solves a series of quadratic convex optimization problems that differ only slightly and we can efficiently maintain and update the inverse of the corresponding Hessians. *Inverse Hessian boosting* (IHB) then refers to the procedure of passing a starting vector, computed with inverse Hessian information, close to the optimal solution to the convex solver used in OAVI. Empirically, IHB speeds up the training time of OAVI by multiple orders of magnitude.

**Large-scale numerical experiments.** We perform numerical experiments on data sets of up to two million samples, highlighting that OAVI is an excellent large-scale feature transformation method.

### 1.2 Related work

The *Buchberger-Möller algorithm* was the first method for constructing generators of the vanishing ideal (Möller and Buchberger, 1982). Its high susceptibility to noise was addressed by Heldt et al. (2009) with the *approximate vanishing ideal algorithm* (AVI), see also Fassino (2010); Limbeck (2013). The latter introduced two algorithms that construct generators term by term instead of degree-wise such as AVI, the *approximate Buchberger-Möller algorithm* (ABM) and the *border bases approximate Buchberger-Möller algorithm*. The
aforementioned algorithms are monomial-aware, that is, they require an explicit ordering of terms and construct generators as linear combinations of monomials. However, monomial-awareness is an unattractive property: Changing the order of the features changes the outputs of the algorithms. Monomial-agnostic approaches such as vanishing component analysis (VCA) (Livni et al., 2013) do not suffer from this shortcoming, as they construct generators as linear combinations of polynomials. VCA found success in hand posture recognition, solution selection using genetic programming, principal variety analysis for nonlinear data modeling, and independent signal estimation for blind source separation (Zhao and Song, 2014; Kera and Iiba, 2016; Iraji and Chitsaz, 2017; Wang and Ohitsu, 2018). The disadvantage of foregoing the term ordering is that VCA sometimes constructs multiple orders of magnitude more generators than monomial-aware algorithms (Wirth and Pokutta, 2022). Furthermore, VCA is susceptible to the spurious vanishing problem: Polynomials that do not capture the nonlinear structure of the data but whose coefficient vector entries are small become generators, and, conversely, polynomials that capture the data well but whose coefficient vector entries are large get treated as non-vanishing. The problem was partially addressed by Kera and Hasegawa (2019; 2020; 2021).

2. Preliminaries

Throughout, let $\ell, k, m, n \in \mathbb{N}$. We denote vectors in bold and let $\mathbf{0} \in \mathbb{R}^n$ denote the 0-vector. Sets of polynomials are denoted by capital calligraphic letters. We denote the set of terms (or monomials) and the polynomial ring over $\mathbb{R}$ in $n$ variables by $\mathcal{T}$ and $\mathcal{P}$, respectively. For $\tau \geq 0$, a polynomial $g = \sum_{i=1}^{k} c_i t_i \in \mathcal{P}$ with $c = (c_1, \ldots, c_k)^T$ is said to be $\tau$-bounded in the $\ell_1$-norm if the $\ell_1$-norm of its coefficient vector is bounded by $\tau$, that is, if $\|c\|_1 := \|c\|_1 \leq \tau$. Given a polynomial $g \in \mathcal{P}$, let $\deg(g)$ denote its degree. The sets of polynomials in $n$ variables of and up to degree $d \in \mathbb{N}$ are denoted by $\mathcal{P}_d$ and $\mathcal{P}_{\leq d}$, respectively. Similarly, for a set of polynomials $\mathcal{G} \subseteq \mathcal{P}$, let $\mathcal{G}_d = \mathcal{G} \cap \mathcal{P}_d$ and $\mathcal{G}_{\leq d} = \mathcal{G} \cap \mathcal{P}_{\leq d}$. We often assume that $X = \{x_1, \ldots, x_m\} \subseteq [0,1]^n$, a form that can be realized, for example, via min-max feature scaling. Given a polynomial $g \in \mathcal{P}$ and a set of polynomials $\mathcal{G} = \{g_1, \ldots, g_k\} \subseteq \mathcal{P}$, define the evaluation vector of $g$ and evaluation matrix of $\mathcal{G}$ over $X$ as $g(X) = (g(x_1), \ldots, g(x_m))^T \in \mathbb{R}^m$ and $\mathcal{G}(X) = (g_1(X), \ldots, g_k(X)) \in \mathbb{R}^{m \times k}$, respectively. Further, define the mean squared error of $g$ over $X$ as

$$\text{mse}(g, X) = \frac{1}{|X|} \|g(X)\|_2^2 = \frac{1}{m} \|g(X)\|_2^2.$$  

$\text{OAVI}$ sequentially processes terms according to a so-called term ordering, as is necessary for any monomial-aware algorithm. For ease of presentation, we restrict our analysis to the degree-lexicographical ordering of terms (DegLex) (Cox et al., 2013), denoted by $<_{\sigma}$. For example, given the terms $t, u, v \in \mathcal{T}_1$, DegLex works as follows:

$$1 <_{\sigma} t <_{\sigma} u <_{\sigma} v <_{\sigma} t^2 <_{\sigma} t \cdot u <_{\sigma} t \cdot v <_{\sigma} u^2 <_{\sigma} u \cdot v <_{\sigma} v^2 <_{\sigma} t^3 <_{\sigma} \ldots,$$

where 1 denotes the constant-1 term. Given a set of terms $\mathcal{O} = \{t_1, \ldots, t_k\}, \sigma \subseteq \mathcal{T}$, the subscript $\sigma$ indicates that $t_1 <_{\sigma} \ldots <_{\sigma} t_k$.

**Definition 2.1** (Leading term (coefficient)). Let $g = \sum_{i=1}^{k} c_i t_i \in \mathcal{P}$ with $c_i \in \mathbb{R}$ and $t_i \in \mathcal{T}$ for all $i \in \{1, \ldots, k\}$ and let $j \in \{1, \ldots, k\}$ such that $t_j >_{\sigma} t_i$ for all $i \in \{1, \ldots, k\} \setminus \{j\}$. Then, $t_j$ and $c_j$ are called leading term and leading term coefficient of $g$, denoted by $\text{lt}(g) = t_j$ and $\text{ltc}(g) = c_j$, respectively.

We thus define approximately vanishing polynomials via the mean squared error as follows.

**Definition 2.2** (Approximately vanishing polynomial). Let $X = \{x_1, \ldots, x_m\} \subseteq \mathbb{R}^n$, $\psi \geq 0$, and $\tau \geq 2$. A polynomial $g \in \mathcal{P}$ is $\psi$-approximately vanishing (over $X$) if $\text{mse}(g, X) \leq \psi$. If also $\text{ltc}(g) = 1$ and $\|g\|_1 \leq \tau$, then $g$ is called $(\psi, 1, \tau)$-approximately vanishing (over $X$).

In the definition above, we fix the leading term coefficient of polynomials to address the spurious vanishing problem, and the requirement that polynomials are $\tau$-bounded in the $\ell_1$-norm is necessary for the learning guarantees of $\text{OAVI}$ to hold.
Approximately vanishing generators of the approximate vanishing ideal. Finally, we introduce the generator-construction problem addressed by $OAVI$.

Oracle approximate vanishing ideal algorithm ($OAVI$)

**Input**: A data set $X = \{x_1, \ldots, x_m\} \subseteq \mathbb{R}^n$ and parameters $\psi \geq 0$ and $\tau \geq 2$.

**Output**: A set of polynomials $\mathcal{G} \subseteq \mathcal{P}$ and a set of monomials $O \subseteq \mathcal{T}$.

1. $d \leftarrow 1$, $O = \{t_1\}_\sigma \leftarrow \{\emptyset\}_\sigma$, $\mathcal{G} \leftarrow \emptyset$
2. While $\partial_d O = \{u_1, \ldots, u_k\}_\sigma \neq \emptyset$ do
   3. For $i = 1, \ldots, k$ do
      4. $t \leftarrow |O| \in \mathbb{N}$, $A \leftarrow O(X) \in \mathbb{R}^{m \times t}$, $b \leftarrow u_i(X) \in \mathbb{R}^m$, $P = \{y \in \mathbb{R}^t \mid \|y\|_1 \leq \tau - 1\}$
      6. $c \leftarrow \arg\min_{y \in \mathbb{R}_m^t} \frac{1}{m} \|Ay + b\|_2^2$ \> call to convex optimization oracle
      7. $g \leftarrow \sum_{j=1}^t c_j t_j + u_i$
      8. If $\text{mse}(g, X) \leq \psi$ then \> check whether $g$ vanishes
         9. $\mathcal{G} \leftarrow \mathcal{G} \cup \{g\}$
      10. Else $O = \{t_1, \ldots, t_{\ell+1}\}_\sigma \leftarrow (O \cup \{u_i\})_\sigma$
   11. End
   12. End
   13. $d \leftarrow d + 1$
14. End

**Definition 2.3** (Approximate vanishing ideal). Let $X = \{x_1, \ldots, x_m\} \subseteq \mathbb{R}^n$, $\psi \geq 0$, and $\tau \geq 2$. The $(\psi, \tau)$-approximate vanishing ideal (over $X$), $I_X^{\psi, \tau}$, is the ideal generated by all $(\psi, 1, \tau)$-approximately vanishing polynomials over $X$.

For $\psi = 0$ and $\tau = \infty$, it holds that $I_X^{0, \infty} = I_X$, that is, the approximate vanishing ideal becomes the vanishing ideal. Finally, we introduce the generator-construction problem addressed by $OAVI$.

**Problem 2.4** (Setting). Let $X = \{x_1, \ldots, x_m\} \subseteq \mathbb{R}^n$, $\psi \geq 0$, and $\tau \geq 2$. Construct a set of $(\psi, 1, \tau)$-approximately vanishing generators of $I_X^{\psi, \tau}$.

Recall that for $t, u \in \mathcal{T}$, $t$ divides (or is a divisor of) $u$, denoted by $t | u$, if there exists $v \in \mathcal{T}$ such that $t \cdot v = u$. If $t$ does not divide $u$, we write $t \nmid u$. $OAVI$ constructs generators of the approximate vanishing ideal of degree $d \in \mathbb{N}$ by checking whether terms of degree $d$ are leading terms of an approximately vanishing generator. As explained in Wirth and Pokutta (2022), $OAVI$ does not have to consider all terms of degree $d$ but only those contained in the subset defined below.

**Definition 2.5** (Border). Let $O \subseteq \mathcal{T}$. The (degree-$d$) border of $O$ is defined as

$$\partial_d O = \{u \in \mathcal{T}_d \mid t \in O_{d-1} \text{ for all } t \in \mathcal{T}_{d-1} \text{ such that } t | u\}.$$  

3. Oracle approximate vanishing ideal algorithm ($OAVI$)

In this section, we recall the oracle approximate vanishing ideal algorithm ($OAVI$) (Wirth and Pokutta, 2022) in Algorithm 1, a method for solving Problem 2.4.

3.1 Algorithm overview

$OAVI$ takes as input a data set set $X = \{x_1, \ldots, x_m\} \subseteq \mathbb{R}^n$, a vanishing parameter $\psi \geq 0$, and a tolerance $\tau \geq 2$ such that the constructed generators are $\tau$-bounded in the $l_1$-norm. From a high-level perspective, $OAVI$ constructs a finite set of $(\psi, 1, \tau)$-approximately vanishing generators of the $(\psi, \tau)$-approximate vanishing ideal $I_X^{\psi, \tau}$ by solving a series of constrained convex optimization problems. $OAVI$ tracks the set of terms $O \subseteq \mathcal{T}$ such that there does not exist a $(\psi, 1, \tau)$-approximately vanishing generator of $I_X^{\psi, \tau}$ with terms only in $O$ and the set of generators $\mathcal{G} \subseteq \mathcal{P}$ of $I_X^{\psi, \tau}$. For every degree $d \geq 1$, $OAVI$ computes the border $\partial_d O$ in Line 2.
Then, in Lines 3–12, for every term $u \in \partial_d \mathcal{O}$, OAVI determines whether there exists a $(\psi, 1, \tau)$-approximately vanishing generator $g$ of $I_X^{\psi, \tau}$ with $\tilde{t}(g) = u$ and other terms only in $\mathcal{O}$ via oracle access to a solver of the constrained convex optimization problem in Line 5. If such a $g$ exists, it gets appended to $\mathcal{G}$ in Line 8. Otherwise, the term $u$ gets appended to $\mathcal{O}$ in Line 10. OAVI terminates when a degree $d \in \mathbb{N}$ is reached such that $\partial_d \mathcal{O} = \emptyset$. For ease of exposition, unless noted otherwise, we ignore that the optimization problem in Line 5 of OAVI is addressed only up to a certain accuracy $\varepsilon > 0$. Throughout, we denote the output of OAVI by $\mathcal{G}$ and $\mathcal{O}$, that is, $(\mathcal{G}, \mathcal{O}) = \text{OAVI}(X, \psi, \tau)$. We replace the first letter in OAVI with the abbreviation corresponding to the solver used to solve the optimization problem in Line 5. For example, OAVI with solver CG is referred to as CGAVI.

### 3.2 Preprocessing with OAVI in a classification pipeline

OAVI can be used as a feature transformation method for a subsequently applied linear kernel support vector machine (SVM). Let $X \subseteq \mathbb{R}^n$ and $Y = \{1, \ldots, k\}$ be an input and output space, respectively. We are given a training sample $\mathcal{S} = \{(x_1, y_1), \ldots, (x_m, y_m)\} \in (\mathcal{X} \times \mathcal{Y})^m$ drawn i.i.d. from some unknown distribution $\mathcal{D}$. Our task is to determine a hypothesis $h : \mathcal{X} \to \mathcal{Y}$ with small generalization error $\mathbb{P}(x, y) = \mathbb{P}(h(x) \neq y)$. For each class $i \in \{1, \ldots, k\}$, let $X^i = \{x_j \in X \mid y_j = i\} \subseteq X = \{x_1, \ldots, x_m\}$ denote the subset of samples corresponding to class $i$ and construct a set of $(\psi, 1, \tau)$-approximately vanishing generators $\mathcal{G}_i$ of the $(\psi, \tau)$-approximate vanishing ideal $I_{X^i}^{\psi, \tau}$ via OAVI, that is, $(\mathcal{G}_i, \mathcal{O}_i) = \text{OAVI}(X^i, \psi, \tau)$. We combine the sets of generators corresponding to the different classes to obtain $\mathcal{G} = \{g_1, \ldots, g_{|\mathcal{G}|}\} = \bigcup_{i=1}^k \mathcal{G}_i$, which encapsulates the feature transformation we are about to apply to data set $X$. As proposed in Livni et al. (2013), we transform the training sample $X$ via the feature transformation

$$x_j \mapsto \tilde{x}_j = ([g_1(x_j)], \ldots, [g_{|\mathcal{G}|}(x_j)])^T \in \mathbb{R}^{|\mathcal{G}|}$$

(FT) for $j = 1, \ldots, m$. The motivation behind this feature transformation is that a polynomial $g \in \mathcal{G}_i$ vanishes approximately over all $x \in X^i$ and (hopefully) attains values that are far from zero over points $x \in X \setminus X^i$. We then train a linear kernel SVM on the feature-transformed data $\tilde{\mathcal{S}} = \{\tilde{x}_1, \ldots, (\tilde{x}_m, y_m)\}$ with $\ell_1$-regularization to promote sparsity. If $\psi = 0, \tau = \infty$, and the underlying classes of $\mathcal{S}$ belong to disjoint algebraic sets, then the different classes become linearly separable in the feature space corresponding to transformation (FT) and perfect classification accuracy is achieved with the linear kernel SVM (Livni et al., 2013).

### 3.3 Solving the optimization problem in Line 5 of OAVI

Setting $\tau = \infty$, the optimization problem in Line 5 becomes unconstrained and can, for example, be addressed with accelerated gradient descent (AGD) (Nesterov, 1983). If, however, $\tau < \infty$, OAVI satisfies two generalization bounds. Indeed, assuming that the data is of the form $X \subseteq [0, 1]^n$, the generators in $\mathcal{G}$ are guaranteed to also vanish on out-sample data. Furthermore, the combined approach of using generators obtained via OAVI to preprocess the data for a subsequently applied linear kernel SVM inherits the margin bound of the SVM (Wirth and Pokutta, 2022).

### 4. OAVI at scale

Here, we address the main shortcoming of vanishing ideal algorithms: time, space, and evaluation complexities that depend superlinearly on $m$, where $m$ is the number of samples in the data set $X = \{x_1, \ldots, x_m\} \subseteq \mathbb{R}^n$. Recall the computational complexity of OAVI for $\tau = \infty$.

**Theorem 4.1** (Complexity (Wirth and Pokutta, 2022)). Let $X = \{x_1, \ldots, x_m\} \subseteq \mathbb{R}^n$, $\psi \geq 0$, $\tau = \infty$, and $(\mathcal{G}, \mathcal{O}) = \text{OAVI}(X, \psi, \tau)$. Let $T_{\text{ORACLE}}$ and $S_{\text{ORACLE}}$ be the time and space complexities required to solve the convex optimization problem in Line 5 of OAVI, respectively. In the real number model, the time and space complexities of OAVI are $O((|\mathcal{G}| + |\mathcal{O}|)^2 + (|\mathcal{G}| + |\mathcal{O}|)T_{\text{ORACLE}})$ and $O((|\mathcal{G}| + |\mathcal{O}|)^2 + S_{\text{ORACLE}})$, respectively. The evaluation vectors of all polynomials in $\mathcal{G}$ over a set $Z = \{z_1, \ldots, z_q\} \subseteq \mathbb{R}^n$ can be computed in time $O((|\mathcal{G}| + |\mathcal{O}|)^2 q)$. 


Under mild assumptions, Wirth and Pokutta (2022) proved that $O(|\mathcal{G}| + |\mathcal{O}|) = O(mn)$, implying that OAVI’s computational complexity is superlinear in $m$. In this work, we improve OAVI’s computational complexity both with a tighter theoretical analysis and algorithmic modifications.

4.1 Number-of-samples-agnostic bound on $|\mathcal{G}| + |\mathcal{O}|$

For $\psi > 0$, OAVI, ABM, AVI, and VCA construct approximately vanishing generators of the $(\psi, \tau)$-approximate vanishing ideal. Despite being designed for the approximate setting ($\psi > 0$), so far, the analysis of these algorithms was only conducted for the exact setting ($\psi = 0$). Below, we exploit that $\psi > 0$ in OAVI and obtain the first number-of-samples-agnostic bound on the size of a generator-constructing algorithm’s output. The result below also highlights the connection between the size of OAVI’s output $|\mathcal{G}| + |\mathcal{O}|$ and the extent of vanishing $\psi$: Large $\psi$ implies small $|\mathcal{G}| + |\mathcal{O}|$ and small $\psi$ implies large $|\mathcal{G}| + |\mathcal{O}|$.

**Theorem 4.2** (Bound on $|\mathcal{G}| + |\mathcal{O}|$). Let $X = \{x_1, \ldots, x_m\} \subseteq [0, 1]^n$, $\psi \in [0, 1]$, $D = \lceil -\log(\psi)/\log(4) \rceil$, $\tau \geq (3/2)^D$, and $(\mathcal{G}, \mathcal{O}) = \text{OAVI}(X, \psi, \tau)$. Then, OAVI terminates after having constructed generators of degree $D$. Thus, $|\mathcal{G}| + |\mathcal{O}| \leq (D + n)^D$.

**Proof.** Let $X = \{x_1, \ldots, x_m\} \subseteq [0, 1]^n$ and let $t_1, \ldots, t_n \in \mathcal{T}_1$ be the degree-1 monomials. Suppose that during OAVI’s execution, for some degree $d \in \mathbb{N}$, OAVI checks whether the term $u = \prod_{j=1}^n t_j^{\alpha_j} \in \partial_d \mathcal{O}$, where $\sum_{j=1}^n \alpha_j = d$ and $\alpha_j \in \mathbb{N}$ for all $j \in \{1, \ldots, n\}$, is the leading term of a $(\psi, 1, \tau)$-approximately vanishing generator with non-leading terms only in $\mathcal{O}$. Let

$$h = \prod_{j=1}^n \left( t_j - \frac{1}{2} \right)^{\alpha_j},$$

where 1 denotes the constant-1 monomial, that is, $1(x) = 1$ for all $x \in \mathbb{R}^n$. Note that $\|h\|_1 \leq (\frac{3}{4})^d \leq \tau$, guaranteeing that $h$ can be constructed in Lines 5–6 of OAVI. Since $u \in \partial_d \mathcal{O}$, for any term $t \in \mathcal{T}$ such that $t \not\equiv u$ and $t \not\equiv u$, it holds that $t \not\in \mathcal{O}$ and thus, $h$ is a polynomial with $\text{ltc}(h) = 1$, $\text{lt}(h) = u$, and other terms only in $\mathcal{O}$. Under the assumption that the convex optimization oracle in OAVI is accurate, $\text{mse}(h, X) \geq \text{mse}(g, X)$, where $g$ is the polynomial constructed during Lines 5 and 6 of OAVI. Hence, proving that $h$ vanishes approximately implies that $g$ vanishes approximately. Note that for all $x \in X$ and $t \in \mathcal{T}_1$, it holds that $t(x) \in [0, 1]$, and, thus, $|t(x) - \frac{1}{2}| \leq \frac{1}{2}$. We obtain

$$\text{mse}(g, X) \leq \text{mse}(h, X)$$

$$= \frac{1}{m} \|h(X)\|_2^2$$

$$= \frac{1}{m} \sum_{x \in X} \left( \prod_{j=1}^n \left( t_j(x) - \frac{1}{2} \right)^{\alpha_j} \right)^2$$

$$\leq \max_{x \in X} \sum_{j=1}^n \left| t_j(x) - \frac{1}{2} \right|^{2\alpha_j}$$

$$\leq 4^{-d}.$$

Since $\text{mse}(g, X) \leq 4^{-d} \leq \psi$ is satisfied for $d \geq D := \left\lceil \frac{-\log(\psi)}{\log(4)} \right\rceil$, OAVI terminates after reaching degree $D$. Thus, at the end of OAVI’s execution, $|\mathcal{G}| + |\mathcal{O}| \leq (D + n)^D \leq (D + n)^D$. \hfill $\Box$

When $T_{\text{Oracle}}$ and $S_{\text{Oracle}}$ are linear in $m$ and polynomial in $|\mathcal{G}| + |\mathcal{O}|$, the time and space complexities of OAVI are linear in $m$ and polynomial in $n$. Furthermore, the evaluation complexity of OAVI is independent of $m$ and polynomial in $n$. 

6
the real number model, the time and space complexities of \( Wirth \) and Pokutta (2022) solved the optimization problem in Line 5 of \( IHB \) by multiple orders of magnitude by exploiting the structure of the optimization problems solved in \( OAVI \).
Corollary 4.4 (Complexity). Let $X = \{x_1, \ldots, x_m\} \subseteq \mathbb{R}^n$, $\psi \geq 0$, $\tau \geq 2$ large enough such that (INF) never holds, and $(\mathcal{G}, \mathcal{O}) = \text{CAVI-IHB}(X, \psi, \tau)$. Assuming CG terminates after a constant number of iterations due to IHB, the time and space complexities of CVAI-IHB are $O((|\mathcal{G}| + |\mathcal{O}|)^2 m + (|\mathcal{G}| + |\mathcal{O}|)^2)$ and $O((|\mathcal{G}| + |\mathcal{O}|) m + (|\mathcal{G}| + |\mathcal{O}|)^2)$, respectively.

In Appendix B.2, we introduce weak inverse Hessian boosting (WIHB), a variant of IHB that speeds up CGAVI variants while preserving sparsity-inducing properties. In Figure 2, see Section 5.1 for details,
Table 1: Overview of data sets. All data sets are binary classification data sets and are retrieved from the UCI Machine Learning Repository (Dua and Graff, 2017) and additional references are provided.

| Data set | Full name | # of samples | # of features |
|----------|-----------|--------------|---------------|
| bank     | banknote authentication | 1,372        | 4             |
| credit   | default of credit cards (Yeh and Lien, 2009) | 30,000       | 22            |
| htru     | HTRU2 (Lyon et al., 2016) | 17,898       | 8             |
| skin     | skin (Bhatt and Dhall, 2010) | 245,057      | 3             |
| spam     | spambase | 4,601        | 57            |
| synthetic | synthetic, see Appendix D | 2,000,000    | 3             |

Table 2: Hyperparameter ranges for numerical experiments.

| Hyperparameters | Values |
|-----------------|--------|
| $\psi$          | 0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001 |
| Regularization coefficient for SVMs | 0.1, 1, 10 |
| Degree for polynomial kernel SVM | 1, 2, 3, 4 |

we observe that CGAVI-IHB is faster than BPCGAVI-WIHB, which is faster than BPCGAVI. In Figure 3, see Section 5.2 for details, we compare the training times of OAVI-IHB to ABM and VCA. In Figure 3d, we observe that OAVI-IHB’s training time scales better than ABM’s and VCA’s.

5. Numerical experiments

Unless noted otherwise, the setup for the numerical experiments applies to all experiments in the paper. Experiments are implemented in Python and performed on an Nvidia GeForce RTX 3080 GPU with 10GB RAM and an Intel Core i7 11700K 8x CPU at 3.60GHz with 64 GB RAM. Our code is publicly available on GitHub.

We implement OAVI as in Algorithm 1 with convex solvers CG, PCG, BPCG, and AGD and refer to the resulting algorithms as CGAVI, PCGAVI, BPCGAVI, and AGDAVI, respectively. Solvers are run for up to 10,000 iterations. For the CG variants, we set $\tau = 1,000$. The CG variants are run up to accuracy $\epsilon = 0.01 \cdot \psi$ and terminated early when less than $0.0001 \cdot \psi$ progress is made in the difference between function values, when the coefficient vector of a generator is constructed, or if we have a guarantee that no coefficient vector of a generator can be constructed. AGD is terminated early if less than $0.0001 \cdot \psi$ progress is made in the difference between function values for 20 iterations in a row or the coefficient vector of a generator is constructed. OAVI implemented with WIHB or IHB is referred to as OAVI-WIHB or OAVI-IHB, respectively. We implement ABM as in Limbeck (2013) but instead of applying the singular value decomposition (SVD) to the matrix corresponding to $A = O(X)$ in OAVI, we apply the SVD to $A^\top A$ in case this leads to a faster training time and we employ the border as in Definition 2.5. We implement VCA as in Livni et al. (2013) but instead of applying the SVD to the matrix corresponding to $A = O(X)$ in OAVI, we apply the SVD to $A^\top A$ in case this leads to a faster training time. We implement a polynomial kernel SVM with a one-versus-rest approach using the scikit-learn software package (Pedregosa et al., 2011) and run the polynomial kernel SVM with $\ell_2$-regularization up to tolerance $10^{-3}$ or for up to 10,000 iterations. For OAVI*, ABM*, and VCA*, the hyperparameters are the vanishing tolerance $\psi$ and the $\ell_1$-regularization coefficient of the linear kernel SVM. For the polynomial kernel SVM, the hyperparameters are the degree and the $\ell_2$-regularization coefficient.

We preprocess with OAVI, ABM, and VCA for a linear kernel SVM as in Section 3.2 and refer to the combined approaches as OAVI+, ABM+, and VCA+, respectively. The linear kernel SVM is implemented using the scikit-learn software package and run with $\ell_1$-penalized squared hinge loss up to tolerance $10^{-4}$ or for up to 10,000 iterations. For OAVI+, ABM+, and VCA+, the hyperparameters are the vanishing tolerance $\psi$ and the $\ell_1$-regularization coefficient of the linear kernel SVM. For the polynomial kernel SVM, the hyperparameters are the degree and the $\ell_2$-regularization coefficient. Table 1 and 2 contain overviews of the data sets and hyperparameter values, respectively.
5.1 Experiment: speeding up CGAVI

We compare the training times of PCGAVI, BPCGAVI, BPCGAVI-WIHB, and CGAVI-IHB on the data sets bank, htru, skin, and synthetic.

Setup. For a single run, we randomly split the data set into subsets of varying sizes. Then, for fixed $\psi = 0.005$, we run the generator-constructing algorithms on subsets of the full data set of varying sizes and plot the training times, which are the times required to run PCGAVI, BPCGAVI, BPCGAVI-WIHB, and CGAVI-IHB once for each class. The results are averaged over ten random runs and standard deviations are shaded in Figures 1 and 2.

Results. The results are presented in Figures 1 and 2. In Figure 1, we observe that the training times for BPCGAVI are often shorter than for PCGAVI, except for the skin data set. In Figure 2, we observe that IHB not only leads to theoretically better training times but also speeds up training in practice: CGAVI-IHB is always faster than BPCGAVI. Furthermore, WIHB is indeed the best of both worlds: BPCGAVI-WIHB is always faster than BPCGAVI but preserves the sparsity-inducing properties of BPCG. The latter can also be seen in Table 3, where BPCGAVI-WIHB is the only algorithmic approach that constructs sparse generators.

5.2 Experiment: scalability comparison

We compare the training times of CGAVI-IHB, AGD-IHB, ABM, and VCA on the data sets bank, htru, skin, and synthetic.

Setup. For a single run, on at most 10,000 samples of the data set, we tune the hyperparameters of generator-constructing algorithms OAVI, ABM, and VCA and a subsequently applied linear kernel SVM using threefold cross-validation. Then, using the determined hyperparameters, we run only the generator-constructing algorithms on subsets of the full data set of varying sizes and plot the training times, which are the times required to run CGAVI-IHB, AGD-IHB, ABM, and VCA once for each class. The results are averaged over ten random runs and standard deviations are shaded in Figure 3.

Results. The results are presented in Figure 3. For small data sets, we observe that ABM and VCA are faster than OAVI. However, when the number of samples in the data set is large, as in the synthetic data set, OAVI can be trained faster than ABM and VCA. AGDAVI-IHB is slower than CGAVI-IHB because AGD cannot use the Frank-Wolfe gap as an early termination criterion to quickly decide that a solution close enough to the optimum is reached.
We compare the performance of CGAVI-IHB*, BPCGAVI-WIHB*, AGDAVI-IHB*, ABM*, VCA*, and polynomial kernel SVM on the data sets credit, htru, skin, and spam.

**Setup.** We tune the hyperparameters on the training data using threefold cross-validation. We retrain on the entire training data set using the best combination of hyperparameters and evaluate the classification error on the test set and the hyperparameter optimization time. For the generator-constructing methods, we also compare $|G| + |O|$, where $|G| = \sum_i |G^i|$, $|O| = \sum_i |O^i|$, and $(G^i, O^i)$ is the output of applying a generator-constructing algorithm to samples belonging to class $i$ and the sparsity of the feature transformation $\mathcal{G} = \bigcup_i \mathcal{G}^i$, which is defined as

$$\text{spar}(\mathcal{G}) = \left(\sum_{g \in \mathcal{G}} g_z\right) / \left(\sum_{g \in \mathcal{G}} g_e\right) \in [0, 1],$$

where for a polynomial $g = \sum_{j=1}^{k} c_j t^j + t$ with $\text{lt}(g) = t$, $g_e = k$ and $g_z = |\{c_j = 0 | j \in \{1, \ldots, k\}\}|$, that is, $g_e$ and $g_z$ are the number of non-leading and the number of zero coefficient vector entries of $g$, respectively. Results are averaged over ten random 60%/40% train/test partitions.

**Results.** The results are presented in Table 3. OAVI* admits excellent test-set classification accuracy. BPCGAVI-WIHB*, in particular, admits the best test-set classification error on all data sets but one. Hyperparameter tuning for OAVI* is often slightly slower than for ABM* and VCA*. Since BPCGAVI-WIHB* does not employ IHB, the approach is always slower than CGAVI-IHB* and AGDAVI-IHB*. For all data sets but credit and skin, the hyperparameter optimization time for the SVM is shorter than for the other approaches. On skin, since the training time of the polynomial kernel SVM is superlinear in $m$, the hyperparameter training for the polynomial kernel SVM is slower than for the other approaches. For data sets with few features $n$, the magnitude of $|G| + |O|$ is often the smallest for VCA*. However, for spam, a data set with $n = 57$, as already
pointed out by Kera and Hasegawa (2019) as the spurious vanishing problem, we observe VCA’s tendency to create unnecessary generators. Finally, only BPCGAVI-WIHB constructs sparse feature transformations, potentially explaining the excellent test-set classification accuracy of the approach.

In conclusion, the numerical results justify considering OAVI as the generator-constructing algorithm of choice, as it is the only approximate vanishing ideal algorithm with known learning guarantees and, in practice, performs better than or similar to related approaches.
Acknowledgements

We would like to thank Gabor Braun for providing us with the main arguments for the proof of the lower bound on the pyramidal width of the $\ell_1$-ball. This research was partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany’s Excellence Strategy – The Berlin Mathematics Research Center MATH* (EXC-2046/1, project ID 390685689, BMS Stipend) and JST, ACT-X Grant Number JPMJAX200F, Japan.

References

Abdi, H. and Williams, L. J. (2010). Principal component analysis. Wiley Interdisciplinary Reviews: Computational Statistics, 2(4):433–459.

Bartlett, M. S. (1951). An inverse matrix adjustment arising in discriminant analysis. The Annals of Mathematical Statistics, 22(1):107–111.

Bhatt, R. and Dhall, A. (2010). Skin segmentation dataset. UCI Machine Learning Repository.

Carderera, A., Pokutta, S., Schütte, C., and Weiser, M. (2021). Cindy: Conditional gradient-based identification of non-linear dynamics–noise-robust recovery. arXiv preprint arXiv:2101.02630.

Cox, D., Little, J., and O’Shea, 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.

Galántai, A. (2000). The theory of newton’s method. Journal of Computational and Applied Mathematics, 124(1-2):25–44.

Globerson, A., Livni, R., and Shalev-Shwartz, S. (2017). Effective semisupervised learning on manifolds. In Proceedings of the Conference on Learning Theory, pages 978–1003.

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. Journal of Machine Learning Research, 3(Mar):1157–1182.

Hazan, E., Agarwal, A., and Kale, S. (2007). Logarithmic regret algorithms for online convex optimization. Machine Learning, 69(2):169–192.

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.

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

Kera, H. (2022). Border basis computation with gradient-weighted normalization. In Proceedings of the International Symposium on Symbolic and Algebraic Computation, ISSAC’22, pages 225–234.

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 Conference on Artificial Intelligence, number 04, pages 4428–4435.
Kera, H. and Hasegawa, Y. (2021). Monomial-agnostic computation of vanishing ideals. arXiv preprint arXiv:2101.00243.

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

Kusiak, A. (2001). Feature transformation methods in data mining. IEEE Transactions on Electronics Packaging Manufacturing, 24(3):214–221.

Lacoste-Julien, S. and Jaggi, M. (2015). On the global linear convergence of Frank-Wolfe optimization variants. In Proceedings of 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 Proceedings of the International Conference on Machine Learning, pages 597–605. PMLR.

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.

Manikandan, G. and Abirami, S. (2021). Feature selection is important: state-of-the-art methods and application domains of feature selection on high-dimensional data. In Applications in Ubiquitous Computing, pages 177–196. Springer.

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 \(O(1/k^2)\). In Doklady an USSR, volume 269, pages 543–547.

Paul, D., Jain, A., Saha, S., and Mathew, J. (2021). Multi-objective pso based online feature selection for multi-label classification. Knowledge-Based Systems, 222:106966.

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. Journal of Machine Learning Research, 12:2825–2830.

Pena, J. and Rodriguez, D. (2019). Polytope conditioning and linear convergence of the frank-wolfe algorithm. Mathematics of Operations Research, 44(1):1–18.

Sherman, J. and Morrison, W. J. (1950). Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. The Annals of Mathematical Statistics, 21(1):124–127.

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

Tsuji, K. K., Tanaka, K., and Pokutta, S. (2022). Pairwise conditional gradients without swap steps and sparser kernel herding. In Proceedings of the International Conference on Machine Learning, pages 21864–21883. PMLR.

Van Der Maaten, L., Postma, E., Van den Herik, J., et al. (2009). Dimensionality reduction: a comparative. Journal of Machine Learning Research, 10(66-71):13.

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

14
Wang, L. and Ohtsuki, T. (2018). Nonlinear blind source separation unifying vanishing component analysis and temporal structure. *IEEE Access*, 6:42837–42850.

Wirth, E. and Pokutta, S. (2022). Conditional gradients for the approximate vanishing ideal. *arXiv preprint arXiv:2202.03349*.

Yeh, I.-C. and Lien, C.-H. (2009). The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications*, 36(2):2473–2480.

Zhao, Y.-G. and Song, Z. (2014). Hand posture recognition using approximate vanishing ideal generators. In *Proceedings of the International Conference on Image Processing*, pages 1525–1529. IEEE.
Algorithm 2: Blended pairwise conditional gradients algorithm (BPCG)

Input : A smooth and convex function \( f \), a starting vertex \( y_0 \in \text{vert}(P) \).
Output : A point \( y_T \in P \).

1. \( S^{(0)} \leftarrow \{y_0\} \)
2. \( \lambda_{y_0}^{(0)} \leftarrow 1 \)
3. \( \Lambda_{y_0}^{(0)} \leftarrow 0 \) for \( v \in \text{vert}(P) \setminus \{y_0\} \)
4. for \( t = 0, \ldots, T - 1 \) do
5.     \[ a_t \leftarrow \arg\max_{v \in S^{(t)}} \langle \nabla f(y_t), v \rangle \]
6.     \[ q_t \leftarrow \arg\min_{v \in S^{(t)}} \langle \nabla f(y_t), v \rangle \]
7.     \[ w_t \leftarrow \arg\min_{v \in \text{vert}(P)} \langle \nabla f(y_t), v \rangle \]
8.     if \( \langle \nabla f(y_t), w_t - y_t \rangle \geq \langle \nabla f(y_t), q_t - a_t \rangle \) then
9.         \[ d_t \leftarrow q_t - a_t \]
10. \[ \gamma_t \leftarrow \arg\min_{\gamma \in [0, \lambda_w^{(0)}]} f(y_t + \gamma d_t) \]
11. \[ \lambda_v^{(t+1)} \leftarrow \lambda_v^{(t)} \] for \( v \in \text{vert}(P) \setminus \{a_t, q_t\} \)
12. \[ \lambda_{a_t}^{(t+1)} \leftarrow \lambda_{a_t}^{(t)} - \gamma_t \]
13. \[ \lambda_{q_t}^{(t+1)} \leftarrow \lambda_{q_t}^{(t)} + \gamma_t \]
14. else
15.     \[ d_t \leftarrow w_t - y_t \]
16. \[ \gamma_t \leftarrow \arg\min_{\gamma \in [0, 1]} f(y_t + \gamma d_t) \]
17. \[ \lambda_v^{(t+1)} \leftarrow (1 - \gamma_t) \lambda_v^{(t)} \] for \( v \in \text{vert}(P) \setminus \{w_t\} \)
18. \[ \lambda_{w_t}^{(t+1)} \leftarrow (1 - \gamma_t) \lambda_{w_t}^{(t)} + \gamma_t \]
19. end
20. \( S^{(t+1)} \leftarrow \{v \in \text{vert}(P) \mid \lambda_v^{(t+1)} > 0\} \)
21. \( y_{t+1} \leftarrow y_t + \gamma_t d_t \)
22. end

Appendix A. Blended pairwise conditional gradients algorithm (BPCG)

The blended pairwise conditional gradients algorithm (BPCG) (Tsuji et al., 2022) is presented in Algorithm 2.

Appendix B. Additional information on IHB

We discuss the computational complexity of IHB in Appendix B.1 and WIHB in Appendix B.2.

B.1 Computational complexity of IHB

The main cost of IHB is the inversion of the matrix \( A^T A \in \mathbb{R}^{f \times f} \), which generally requires \( O(f^3) \) elementary operations. Since OAVI solves a series of quadratic convex optimization problems that differ from each other only slightly, we can maintain and update \((A^T A)^{-1}\) using \( O(f^2) \) instead of \( O(f^3) \) elementary operations.

Theorem B.1 (IHB update cost). Let \( A \in \mathbb{R}^{m \times f} \), \( A^T A \in \mathbb{R}^{f \times f} \), \( (A^T A)^{-1} \), and \( b \in \mathbb{R}^f \) be given. In case \( \|b\|_2 > 0 \) and \( b^T A (A^T A)^{-1} A^T b \neq \|b\|_2^2 \),

\[
\hat{A} = (A, b) \in \mathbb{R}^{m \times (f+1)}, \quad \hat{A}^T \hat{A} \in \mathbb{R}^{(f+1) \times (f+1)}, \quad \hat{A}^T \hat{A}^{-1} \in \mathbb{R}^{(f+1) \times (f+1)} \tag{B.1}
\]

can be constructed in \( O(f m + \ell^2) \) elementary operations.

Proof. Let \( B = A^T A \in \mathbb{R}^{f \times f} \), \( \hat{B} = \hat{A}^T \hat{A} \in \mathbb{R}^{(f+1) \times (f+1)} \), \( N = B^{-1} \in \mathbb{R}^{f \times f} \), and \( \hat{N} = \hat{B}^{-1} = (\hat{A}^T \hat{A})^{-1} \in \mathbb{R}^{(f+1) \times (f+1)} \).
In \( O(m f) \) elementary operations, we can construct \( A^T b \in \mathbb{R}^f \), \( b^T b = \|b\|_2^2 \in \mathbb{R} \), and \( \hat{A} = (A, b) \in \mathbb{R}^{m \times (f+1)} \) and
We can then compute
\[
\tilde{B} = \tilde{A}^\top \tilde{A} = \left( \begin{bmatrix} B & A^\top b \end{bmatrix} b^\top A \left\| b \right\|^2_2 \right) \in \mathbb{R}^{(\ell+1) \times (\ell+1)}.
\]

We can then compute \( \tilde{N} = \tilde{B}^{-1} = (\tilde{A}^\top \tilde{A})^{-1} \in \mathbb{R}^{(\ell+1) \times (\ell+1)} \) in additional \( O(\ell^2) \) elementary operations. We write
\[
\tilde{N} = \begin{pmatrix} \tilde{N}_1 & \tilde{n}_2 \\ \tilde{n}_2^\top & \tilde{n}_3 \end{pmatrix} \in \mathbb{R}^{(\ell+1) \times (\ell+1)},
\]
where \( \tilde{N}_1 \in \mathbb{R}^{\ell \times \ell} \), \( \tilde{n}_2 \in \mathbb{R}^\ell \), and \( \tilde{n}_3 \in \mathbb{R} \). Then, it has to hold that
\[
\tilde{B} \tilde{N} = \begin{pmatrix} B & A^\top b \\ b^\top A \left\| b \right\|^2_2 \end{pmatrix} \begin{pmatrix} \tilde{N}_1 & \tilde{n}_2 \\ \tilde{n}_2^\top & \tilde{n}_3 \end{pmatrix} = I \in \mathbb{R}^{(\ell+1) \times (\ell+1)},
\]
where \( I \) is the identity matrix. Note that \( b^\top A \tilde{n}_2 + \left\| b \right\|^2_2 \tilde{n}_3 = 1 \). Thus,
\[
\tilde{n}_3 = \frac{1 - b^\top A \tilde{n}_2}{\left\| b \right\|^2_2} \in \mathbb{R}, \tag{B.2}
\]
which is well-defined due to the assumption \( \left\| b \right\| > 0 \). Since \( b^\top A \) is already computed, once \( \tilde{n}_2 \) is computed, the computation of \( \tilde{n}_3 \) requires only additional \( O(\ell^2) \) elementary operations. Similarly, we have that
\[
B \tilde{n}_2 + A^\top b \tilde{n}_3 = 0 \in \mathbb{R}^\ell. \tag{B.3}
\]
Plugging (B.2) into (B.3), we obtain \( (B - A^\top bb^\top A) \tilde{n}_2 = -A^\top b / \left\| b \right\|^2_2 \). Thus,
\[
\tilde{n}_2 = -\left( B - A^\top bb^\top A / \left\| b \right\|^2_2 \right)^{-1} A^\top b / \left\| b \right\|^2_2.
\]
The existence of \( (B - A^\top bb^\top A)^{-1} \) follows from the Sherman-Morrison formula (Sherman and Morrison, 1950; Bartlett, 1951) and the assumption that \( b^\top A (A^\top A)^{-1} A^\top b \neq \left\| b \right\|^2_2 \). Then, again using the Sherman-Morrison formula,
\[
\tilde{n}_2 = -\left( B^{-1} + \frac{B^{-1} A^\top b b^\top A B^{-1}} {\left\| b \right\|^2_2 - b^\top A b} \right) A^\top b / \left\| b \right\|^2_2,
\]
which can be computed using additional \( O(\ell^2) \) elementary operations. Finally, we construct \( \tilde{N}_1 \), which is determined by \( B \tilde{N}_1 + A^\top \tilde{n}_2 \tilde{n}_2^\top = I \in \mathbb{R}^{\ell \times \ell} \). Thus, \( \tilde{N}_1 = B^{-1} - B^{-1} A^\top b \tilde{n}_2 \tilde{n}_2^\top \in \mathbb{R}^{\ell \times \ell} \), which can be computed in additional \( O(\ell^2) \) elementary operations since \( B^{-1} A^\top b \) is already computed. In summary, we require \( O(\ell m + \ell^2) \) elementary operations. Note that even if we do not compute \( (A^\top A)^{-1} \), we still require \( O(\ell m + \ell^2) \) elementary operations.

In the remark below, we discuss the literature related to IHB.

**Remark B.2** (Work related to IHB). The proof of Theorem B.1 is similar to the proof that the inverse of the Hessian can be updated efficiently in the online Newton algorithm (Hazan et al., 2007). Both proofs rely on the Sherman-Morrison formula (Sherman and Morrison, 1950; Bartlett, 1951). However, in our setting, the updates occur column-wise instead of row-wise. Updating \( (A^\top A)^{-1} \) column-wise for generator-construction algorithms was already discussed in Limbeck (2013) using QR decompositions without addressing the numerical instability of inverse matrix updates.

Next, we prove that Theorem B.1 implies the improved computational complexity of CGAVI-IHB in Corollary 4.4.
Proof of Corollary 4.4. Suppose that CGAVI-IHB is currently executing Lines 5–11 of Algorithm 1 for a particular term $u_i \in \partial_2O$ and recall the associated notations $\ell = |O|$, $A = O(X) \in \mathbb{R}^{m \times \ell}$, $b = u_i(X)$, $f(y) = \frac{1}{m} \|Ay + b\|_2^2$, and $P = \{y \in \mathbb{R}^{\ell} \mid \|y\|_1 \leq \tau - 1\}$. Then, the optimization problem in Line 5 of Algorithm 1 takes the form
\[
c \in \arg\min_{y \in P} f(y).
\]
We first prove that the violation of any of the two assumptions of Theorem B.1 implies the existence of a $(\psi, 1, \tau)$-approximately vanishing generator $g$. We prove this claim by treating the two assumptions separately:

1. If $\|b\|_2 = 0$, it holds that $\text{mse}(u_i, X) = 0$. By the assumption that $\tau \geq 2$ is large enough to guarantee that (INF) never holds, CGAVI-IHB constructs a $(\psi, 1, \tau)$-approximately vanishing generator in Line 6.

2. If $b^\top A (A^\top A)^{-1} A^\top b = \|b\|_2^2$, it holds that $f(-(A^\top A)^{-1} A^\top b) = 0$. Thus, by the assumption that $\tau \geq 2$ is large enough to guarantee that (INF) never holds, CGAVI-IHB constructs a $(\psi, 1, \tau)$-approximately vanishing generator in Line 6.

Since an update of $(A^\top A)^{-1}$ is necessary only if Line 10 is executed, that is, when there does not exist a $(\psi, 1, \tau)$-approximately vanishing generator $g$, we never have to update $(A^\top A)^{-1}$ when the assumptions of Theorem B.1 are violated. Thus, by Theorem B.1, we can always update $A$, $A^\top A$, and $(A^\top A)^{-1}$ in $O((m + \ell^2) \log((|G| + |O|)m + (|G| + |O|)^2))$ elementary operations using space $O((m + \ell^2) \log((|G| + |O|)m + (|G| + |O|)^2))$. Then, since we run CG for a constant number of iterations, the time and space complexities of Lines 5–11 are $T_{\text{oracle}} = O((|G| + |O|)m + (|G| + |O|)^2)$ and $S_{\text{oracle}} = O((|G| + |O|)m + (|G| + |O|)^2)$, respectively. \qed

Note that the time and space complexities in Corollary 4.4 also hold for AGDAVI-IHB with $\tau = \infty$.

B.2 Weak inverse Hessian boosting (WIHB)

So far, we have introduced IHB to drastically speed-up training of AGDAVI and CGAVI. A drawback of using CGAVI-IHB is that the initialization of CG variants with a non-sparse initial vector such as $y_0 = (A^\top A)^{-1} A^\top b$ leads to the construction of generally non-sparse generators. In this section, we explain how to combine the speed-up of IHB with the sparsity-inducing properties of CG variants, referring to the resulting technique as weak inverse Hessian boosting (WIHB). Specifically, we present WIHB with BPCGAVI (OAVI with solver BPCG), referred to as BPCGAVI-WIHB in Algorithm 3.

The high-level idea of BPCGAVI-WIHB is to use IHB and vanilla CG to quickly check whether a $(\psi, 1, \tau)$-approximately vanishing generator exists. If it does, we then use BPCG to try and construct a $(\psi, 1, \tau)$-approximately vanishing generator that is also sparse.

We proceed by giving a detailed overview of BPCGAVI-WIHB. In Line 5 of BPCGAVI-WIHB, we construct $y_0$. Then, in Line 6, we solve $c \in \arg\min_{y \in P} \frac{1}{m} \|Ay + b\|_2^2$ up to tolerance $\epsilon$ using CG with starting vector $y_0 = (A^\top A)^{-1} A^\top b$. Since $y_0$ is a well-educated guess for the solution to the constrained optimization problem $\arg\min_{y \in P} \frac{1}{m} \|Ay + b\|_2^2$ in Line 6, CG often runs for only very few iterations. The drawback of using the non-sparse $y_0$ as a starting vector is that $c$ constructed in Line 6 and $g$ constructed in Line 7 are generally non-sparse. We alleviate the issue of non-sparsity of $g$ in Lines 8–18. In Line 8, we first check whether $g$ is a $(\psi, 1, \tau)$-approximately vanishing generator of $X$. If $g$ does not vanish approximately, we know that there does not exist an approximately vanishing generator with leading term $u_i$ and we append $u_i$ to $O$ in Line 17. If, however, $\text{mse}(g, X) \leq \psi$, we solve the constrained convex optimization problem in Line 6 again in Line 9 up to tolerance $\epsilon$ using BPCG and a vertex of the $\ell_1$-ball $P$ as starting vector. This has two consequences:

1. The vector $d$ constructed in Line 9 tends to be sparse, as corroborated by the results in Table 3.
2. The execution of Line 9 tends to take longer than the execution of Line 6 since BPCG’s starting vector is not necessarily close in Euclidean distance to the optimal solution.
Algorithm 3: Blended pairwise conditional gradients approximate vanishing ideal algorithm with weak inverse Hessian boosting (BPCGAVI-WIHB)

| Input | A data set $X = \{x_1, \ldots, x_m\} \subseteq \mathbb{R}^m$ and parameters $\psi \geq \epsilon \geq 0$ and $\tau \geq 2$. |
|-------|---------------------------------------------------------------|
| Output| A set of polynomials $G \subseteq \mathcal{P}$ and a set of monomials $O \subseteq \mathcal{T}$. |

1. $d \leftarrow 1$, $O = \{t_1\}_\sigma \leftarrow \{1\}_\sigma$, $G \leftarrow 0$
2. while $\partial_d O = \{u_1, \ldots, u_k\}_\sigma \neq 0$ do
   1. for $i = 1, \ldots, k$ do
      1. $\ell \leftarrow |O| \in \mathbb{N}$, $A \leftarrow O(X) \in \mathbb{R}^{m \times \ell}$, $b \leftarrow u_i(X) \in \mathbb{R}^m$, $P = \{y \in \mathbb{R}^\ell \mid \|y\|_1 \leq \tau - 1\}$
      2. $y_0 \leftarrow (A^T A)^{-1} A^T b \in \mathbb{R}^\ell$
      3. Solve $c \in \arg\min_{y \in P} \frac{1}{m} \|Ay + b\|_2^2$ up to tolerance $\epsilon$ using CG with starting vector $y_0$.
      4. $g \leftarrow \sum_{j=1}^\ell c_j t_j + u_i$  \> a non-sparse polynomial
      5. if $\text{mse}(g, X) \leq \psi$ then \> check whether $g$ vanishes
         1. Solve $d \in \arg\min_{x \in P} \frac{1}{m} \|Ax + b\|_2^2$ up to tolerance $\epsilon$ using BPCG and a vertex of $P$ as starting vector.
         2. $h \leftarrow \sum_{j=1}^\ell d_j t_j + u_i$ \> a sparse polynomial
         3. if $\text{mse}(h, X) \leq \psi$ then \> check whether $h$ vanishes
            1. $G \leftarrow G \cup \{h\}$
         else
            1. $G \leftarrow G \cup \{g\}$
      else
         1. $O = \{t_1, \ldots, t_{\ell+1}\}_\sigma \leftarrow (O \cup \{u_i\})_\sigma$
   end
3. $d \leftarrow d + 1$
end

Then, in Line 10, we construct the polynomial $h$, which tends to be sparse. If $h$ is a $(\psi, 1, \tau)$-approximately vanishing generator, we append the sparse $h$ to $G$ in Line 12. If it happens that $\text{mse}(g, X) \leq \psi < \text{mse}(h, X)$, we append the non-sparse $g$ to $G$ in Line 14. Following the discussion of Section 4.3, should (INF) ever hold, that is, $\|y_0\|_1 > \tau - 1$, we can no longer proceed with BPCGAVI-WIHB as we can no longer guarantee that the inverse of $A^T A$ exists. In that case, we proceed with vanilla BPCGAVI for all terms remaining in the current and upcoming borders.

The discussion above illustrates that BPCGAVI-WIHB solves $\arg\min_{y \in P} \frac{1}{m} \|Ay + b\|_2^2$ with BPCG $|G|$ times as opposed to BPCGAVI, which solves $\arg\min_{y \in P} \frac{1}{m} \|Ay + b\|_2^2$ with BPCG $|G| + |O| - 1$ times. Thus, BPCGAVI-WIHB combines the sparsity-inducing properties of BPCG with the speed-up of IHB without any of IHB’s drawbacks. The speed-up of BPCGAVI-WIHB compared to BPCGAVI is evident from the numerical experiments in Figure 2 and the results of Table 3 indicate that the sparsity-inducing properties of BPCG are successfully exploited.

Appendix C. Pyramidal width of the $\ell_1$-ball

Pena and Rodriguez (2019) showed that the pyramidal width $\omega$ of the $\ell_1$-ball of radius $\tau$, that is, $P = \{x \in \mathbb{R}^\ell \mid \|x\|_1 \leq \tau\} \subseteq \mathbb{R}^\ell$ is given by

$$\omega = \min_{F \in \text{faces}(P), 0 \subseteq F \subseteq P} \text{dist}(F, \text{conv}(\text{vert}(P) \setminus F)),$$

where, for a polytope $P \subseteq \mathbb{R}^\ell$, faces $(P)$ denotes the set of faces of $P$ and $\text{vert}(P)$ denotes the set of vertices of $P$, for a set $F \subseteq \mathbb{R}^\ell$, $\text{conv}(F)$ is the convex hull of $F$, and for two disjoint sets $F, G \subseteq \mathbb{R}^\ell$, $\text{dist}(F, G)$ is the Euclidean distance between $F$ and $G$. 
Lemma C.1 (Pyramidal width of the $\ell_1$-ball). The pyramidal width of the $\ell_1$-ball of radius $\tau$, that is, $P = \{ x \in \mathbb{R}^\ell \mid \|x\|_1 \leq \tau \} \subseteq \mathbb{R}^\ell$, is lower bounded by $\omega \geq \frac{\tau}{\sqrt{\ell-1}}$.

Proof. Let $e^{(i)} \in \mathbb{R}^\ell$ denote the $i$th unit vector. Note that a non-trivial face of $P = \text{conv}\{ \pm \tau e^{(1)}, \ldots, \pm \tau e^{(n)} \} \subseteq \mathbb{R}^\ell$ that is not $P$ itself cannot contain both $\tau e^{(i)}$ and $-\tau e^{(i)}$ for any $i \in \{1, \ldots, \ell\}$. Thus, due to symmetry, we can assume that any non-trivial face of $P$ that is not $P$ itself is of the form $F = \text{conv}(\tau e^{(1)}, \ldots, \tau e^{(k)})$ for some $k \in \{1, \ldots, \ell-1\}$. Then,

$$G := \text{conv}(\text{vert}(P) \setminus F) = \text{conv}(\{-\tau e^{(1)}, \ldots, -\tau e^{(k)}, \pm \tau e^{(k+1)}, \ldots, \pm \tau e^{(\ell)}\}).$$

We have that

$$\text{dist}(F, G) = \min_{u \in F, v \in G} \|u - v\|_2$$

$$= \min_{u \in F, v \in G} \sqrt{\sum_{i=1}^{k} (u_i - v_i)^2 + \sum_{j=k+1}^{\ell} v_j^2}$$

$$\geq \min_{u \in F, v \in G} \sqrt{\sum_{i=1}^{k} (u_i - v_i)^2} \quad \because \text{since } v_j^2 \geq 0 \text{ for all } j \in \{k+1, \ldots, \ell\}$$

$$\geq \min_{u \in F} \sqrt{\sum_{i=1}^{k} u_i^2} \quad \because \text{since } v_i \leq 0 \text{ for all } i \in \{1, \ldots, k\}$$

$$= \sqrt{\frac{\tau^2}{k}}.$$

Since $\sqrt{\frac{\tau^2}{k}}$ is minimized for $k \in \{1, \ldots, \ell-1\}$ as large as possible, it follows that $\omega \geq \frac{\tau}{\sqrt{\ell-1}}$. \hfill \Box

Appendix D. The synthetic data set

The synthetic data set consists of three features and contains two different classes. The samples $x$ that belong to the first class are generated such that they satisfy the equation

$$x_1^2 + 0.01x_2 + x_3^2 - 1 = 0$$

and samples that belong to the second class are generated such that they satisfy the equation

$$x_1^2 + x_3^2 - 1.3 = 0.$$

The samples are perturbed with additive Gaussian noise with mean 0 and standard deviation 0.05.