Machine learning the real discriminant locus

Edgar A. Bernal* Jonathan D. Hauenstein† Dhagash Mehta‡
Margaret H. Regan§ Tingting Tang¶

August 8, 2022

Abstract

Parameterized systems of polynomial equations arise in many applications in science and engineering with the real solutions describing, for example, equilibria of a dynamical system, linkages satisfying design constraints, and scene reconstruction in computer vision. Since different parameter values can have a different number of real solutions, the parameter space is decomposed into regions whose boundary forms the real discriminant locus. This article views locating the real discriminant locus as a supervised classification problem in machine learning where the goal is to determine classification boundaries over the parameter space, with the classes being the number of real solutions. This article presents a novel sampling method which carefully samples a multidimensional parameter space. At each sample point, homotopy continuation is used to obtain the number of real solutions to the corresponding polynomial system. Machine learning techniques including nearest neighbors, support vector classifiers, and neural networks are used to efficiently approximate the real discriminant locus. One application of having learned the real discriminant locus is to develop a real homotopy method that only tracks real solution paths unlike traditional methods which track all complex solution paths. Examples show that the proposed approach can efficiently approximate complicated solution boundaries such as those arising from the equilibria of the $N = 4$ Kuramoto model which was previously intractable using traditional methods.

Keywords. Discriminant locus, machine learning, deep learning, numerical algebraic geometry

1 Introduction

Systems of polynomial equations are a collection of multivariate nonlinear equations in which each equation is a multivariate polynomial. Such systems arise naturally in many areas of science and engineering ranging from chemistry, particle physics, string theory, mathematical biology, phylogenetics, control theory, robotics, power systems, and computer vision [11, 21, 22, 23, 69]. In many of these applications, the coefficients of the equations depend upon one or more parameters

*FLX AI, Rochester, NY 14607 (edgar.bernal@flxai.com).

†Department of Applied and Computational Mathematics and Statistics, University of Notre Dame, Notre Dame, IN 46556 (hauenstein@nd.edu, `www.nd.edu/~jhauenst`). This author was supported in part by NSF grant CCF-1812746 and ONR N00014-16-1-2722.

‡The Vanguard Group, Malvern, PA 19355 (dhagashbmehta@gmail.com).

§Department of Mathematics, Duke University, Durham, NC 27708 (mregan9@math.duke.edu, `www.margaretregan.com`). This author was supported in part by Schmitt Leadership Fellowship in Science and Engineering and NSF grant CCF-1812746.

¶Department of Mathematics and Statistics, San Diego State University, Imperial Valley, CA, 92231.

yielding parameterized systems of polynomial equations. Both the solutions and the number of real solutions are functions of the parameters. Investigating the solution structure as a function of the parameters is typically more difficult than solving the system for given values of the parameters.

Due to its ubiquity, many methods have been proposed to characterize the solution structure over the parameter space. Classically, the discriminant describes the boundary between regions where the solution structure changes [31]. For example, the discriminant of

$$f(x; b, c) = x^2 + bx + c \tag{1}$$

is $D = b^2 - 4c$. Since real parameters and the number of real solutions are of most interest in applications describing, for example, equilibria of dynamical systems and scene reconstruction in computer vision, this paper focuses on the boundary between regions where the number of real solutions change. Thus, from this perspective, we concentrate on the real discriminant locus associated with counting the number of real solutions.

Definition 1. *Given a parameteric polynomial system $f(x; p) : \mathbb{R}^n \times \mathbb{R}^k \to \mathbb{R}^n$, its real discriminant locus is the boundary in the parameter space where the number of real solutions changes.*

For Eq. (1), the solution set in $\mathbb{R}^2$ of $D = 0$ is the real discriminant locus, which forms the boundary between the region in $\mathbb{R}^2$ with $D > 0$ where $f = 0$ has two real solutions and the region in $\mathbb{R}^2$ with $D < 0$ where $f = 0$ has no real solutions. In addition to just the number of real solutions, one could also be interested in additional structure related to the real solutions, e.g., geometric properties, which would then define a corresponding real discriminant locus. See [53] for more details.

Comprehensive Gröbner basis computations [74] can be used to symbolically compute the discriminant polynomial over the complex numbers whose solution set, i.e., algebraic variety, is called the complex discriminant locus. Since the discriminant actually defines the boundaries over the complex parameter space, one can develop specialized methods over the real numbers. Some examples include cylindrical algebraic decomposition [7, 36, 46, 53, 75], Brouwer degree [18], and polyhedral methods [13, 32] which have been utilized for modest size systems. However, computational methods which depend upon Gröbner basis or other symbolic computations severely suffer from exponential complexity.

To mitigate these issues, global symbolic methods can be replaced by local, numerical approximations to determine the discriminant locus. For larger systems, numerical methods based on a form of homotopy continuation [3, 11, 69] have been employed in which one tracks the solution structure as the parameters are varied continuously. Several computational packages such as AUTO [28] and MATCONT [27] employ such techniques for parameterized differential equations. Rather than directly run into the real discriminant locus, a perturbed sweeping approach was presented in [37]. In particular, when all complex solutions over a general parameter point can be computed, homotopy continuation provides an approach to obtain global information about the solutions which, for example, can be used to obtain the number of real solutions at selected sample points in the parameter space [8, 17, 34, 45, 58].

Our method aims to numerically approximate the real discriminant locus by viewing this as a classification problem in machine learning. The problem of approximating the real discriminant locus is posed as a problem of approximating the decision boundaries that separate data points according to their labels, where the input features are the parameters of the given parametric system and the target labels are the number of real solutions at the corresponding parameter values. Given a parameter point, homotopy continuation can be used to generate the labels, i.e., compute the number of real solutions. A novel approach for selecting sample points is developed by leveraging domain

knowledge obtained from numerical algebraic geometry [11, 69] to help guide the approximation of the decision boundaries.

Although there has been a collection of papers, such as [25, 48, 49, 50, 51, 52, 66], attempting to solve polynomial equations and improving algorithms using neural networks, the approach closest to the present work is [62]. That work uses a feed-forward neural network with one hidden layer to predict the number of real solutions for univariate polynomials. All the sample points, including training data and testing data are combinations of integer coefficients in the parameter space. The results indicate that the ability of an artificial neural network to generalize on the test sets is comparable to its performance on the training sets. Employing several algorithms to train the network for high degree polynomials showed that the choice of training program impacts the performance of the network.

Other applications of deep networks for analyzing polynomial equations include [4, 5] which investigated the effectiveness of deep networks to learn a target function that is a low degree polynomial. A neural network which is used as a pre-training method for finding the number of real solutions and then used to design a neural network-like model to compute real solutions to univariate polynomial in parallel is provided in [25]. Finally, [16] employed machine learning algorithms to learn the geometry and topology of the complex solution set of systems of polynomial equations.

This manuscript provides a novel approach to analyzing the real solution structure of parameterized polynomial equations which are multivariate and depend upon many parameters. The specific contributions are as follows: (1) Transform the problem of computing the real discriminant locus of parameterized polynomial equations into a supervised classification problem in order to use machine learning constructs such as nearest neighbor, support vector classifiers, and deep learning techniques; (2) Devise a novel sampling technique that leverages domain knowledge from numerical algebraic geometry which can be thought of as a static active learning implementation where the desired training set is determined in advance; (3) Show that machine learning techniques can quickly approximate the real discriminant locus even when they contain cusps and other singular regions which, in turn, provides a decomposition of the parameter space into regions where the number of real solutions remains constant; (4) Demonstrate that the proposed method was able to break a ceiling in computational algebraic geometry as the new approach is able to analyze the $N = 4$ Kuramoto model which was previously intractable using traditional methods; (5) Design a real homotopy method that utilizes an approximation of the real discriminant locus to track only real solution paths and computes only real solutions, thus improving efficiency.

The rest of the paper is organized as follows. Section 2 provides background information on numerical algebraic geometry, homotopy continuation, and parameterized systems. Similarly, Section 3 provides an overview of the machine learning techniques utilized: nearest neighbor, support vector classifiers, and deep learning. The novel sampling scheme that leverages domain knowledge from numerical algebraic geometry is presented in Section 4. Section 5 applies the proposed approach to several examples. A real homotopy method is outlined in Section 6, which utilizes the learned real discriminant locus to track only real solution paths. The paper concludes in Section 7.

2 Numerical algebraic geometry

The following describes parameter homotopies and pseudowitness point sets, which are two topics in numerical algebraic geometry that will be used to learn the real discriminant locus. See [11, 69] for more details regarding numerical algebraic geometry.

2.1 Parameter homotopies

For simplicity, we consider parameterized polynomial systems of the form

$$f(x;p) = f(x_1,\ldots,x_n;p_1,\ldots,p_k) = \begin{bmatrix} f_1(x_1,\ldots,x_n;p_1,\ldots,p_k) \\ \vdots \\ f_n(x_1,\ldots,x_n;p_1,\ldots,p_k) \end{bmatrix} \tag{2}$$

such that, for a generic $p^* \in \mathbb{C}^k$, the algebraic variety defined by $f(x;p^*) = 0$ consists of finitely many points in $\mathbb{C}^n$, say d, all of which are nonsingular. A solution x^* of $f(x;p^*) = 0$ is nonsingular if $J_x f(x^*;p^*)$ is invertible where $J_x f(x;p)$ is the Jacobian matrix of f with respect to x. This is the typical situation for well-constrained parameterized polynomial systems arising in science and engineering applications. For reducing overdetermined parameterized systems to well-constrained parameterized systems adhering to Eq. (2), see [39] and the references therein. One can also reduce to the nonsingular isolated case for parameterized systems which generically define a positive-dimensional algebraic variety using linear slicing and singular isolated solutions using deflation, e.g., see [44, 56].

A key consequence of this setup is that the real parameter space $\mathbb{R}^k$ contains open subsets where the number of real solutions to $f(x;p) = 0$ is constant. The boundaries of these open subsets form the *real discriminant locus*.

Example 1. *The parameterized quadratic described by Eq.* (1) *generically has* $d = 2$ *solutions in* $\mathbb{C}$. *For* $D = b^2 - 4c$, *the two open subsets defined by* $D > 0$ *and* $D < 0$ *in* $\mathbb{R}^2$ *have a constant number of real solutions, namely* 2 *and* 0. *The real discriminant locus is the algebraic variety defined by* $D = 0$ *in* $\mathbb{R}^2$.

The complex discriminant locus consists of the parameter points in $\mathbb{C}^p$ where $f(x;p) = 0$ does not have d nonsingular solutions. Since $f(x;p)$ is well-constrained, the complex discriminant locus is either empty or is a hypersurface in $\mathbb{C}^p$. Section 4 exploits this fact to generate sample points on the real discriminant locus, which is contained in the complex discriminant locus. The following illustrates this while showing that the real discriminant locus could have smaller dimension.

Example 2. *Consider the following from [40, Ex 2.1]:*

$$f(x;p) = \begin{bmatrix} x_1^2 - x_2^2 - p_1 \\ 2x_1x_2 - p_2 \end{bmatrix}.$$

The system of equations $f(x;p) = 0$ *generically has* $d = 4$ *solutions with* $p_1^2 + p_2^2 = 0$ *in* $\mathbb{C}^2$ *defining the complex discriminant locus. Thus, the complex discriminant locus is a curve in* $\mathbb{C}^2$, *while the only point in* $\mathbb{R}^2$ *on this complex hypersurface is the origin. Moreover, for all* $p \in \mathbb{R}^2 \setminus \{(0,0)\}$, $f(x;p) = 0$ *has* 2 *real solutions showing that the real discriminant locus in* $\mathbb{R}^2$ *is* $\{(0,0)\}$.

Given $p \in \mathbb{C}^k$ outside of the complex discriminant locus, the solutions to $f(x;p) = 0$ can be computed using a parameter homotopy [61]. In order to utilize a parameter homotopy, one first needs to know a parameter value $p^* \in \mathbb{C}^k$ and a set $S \subset \mathbb{C}^n$ consisting of the d solutions to $f(x;p^*) = 0$. Obtaining this starting information is called the *ab initio* phase. See [11, Chap. 6] for more details.

Ab initio phase The input for a parameter homotopy is a parameter value $p^* \in \mathbb{C}^k$ and the algebraic variety $S \subset \mathbb{C}^n$ defined by $f(x;p^*) = 0$ consisting of d points. One can compute S by using a classical linear homotopy. For example, if $e_i = \deg f_i$ and $g_i(x) = x_i^{e_i} - 1$, then

$$H(x,t) = (1-t)f(x;p^*) + \gamma t g(x) = 0$$

where $\gamma \in \mathbb{C}$ is generic is called a total degree homotopy consisting of $\prod_{i=1}^{n} e_i$ paths. Clearly, the solutions of $H(x,1) = g(x) = 0$ are trivial to compute providing the $\prod_{i=1}^{n} e_i$ start points. For $t \in (0,1]$, the solution paths defined by $H(x,t) = 0$ are smooth and can be traversed using a variety of numerical methods [10, 11]. The d solution paths which have a finite limit as t approaches 0 converge to the d solutions of $f(x;p^*) = 0$. By assumption on the structure of f, the other solution paths will diverge to infinity. To possibly reduce the number of paths that diverge, one can select a different structure for $g(x)$ such as based on the multihomogeneous structure of f or the monomial structure of f, e.g., see [69, Chap. 8].

Parameter homotopy phase With the *ab initio* phase complete, the "online" parameter homotopy phase can commence to solve $f(x;p) = 0$ for various $p \in \mathbb{C}^k$. One utilizes the parameter homotopy

$$H(x,t) = f(x; \tau(t) \cdot p^* + (1 - \tau(t)) \cdot p) = 0 \quad \text{where} \quad \tau(t) = \frac{\gamma t}{1 + (\gamma - 1)t} \tag{3}$$

such that $t \in [0,1]$ and $\gamma \in \mathbb{C}$. In particular, $H(x,1) = f(x;p^*) = 0$ has known solutions S, computed in the *ab initio* phase, and one aims to compute the solutions to $H(x,0) = f(x;p) = 0$. For generic values of the constant $\gamma \in \mathbb{C}$, the arc $\tau(t) \cdot p^* + (1 - \tau(t)) \cdot p$ for $t \in [0,1]$ connects p^* to p and avoids the complex discriminant locus. Thus, for $t \in [0,1]$, $H(x,t) = 0$ defines precisely d solution paths connecting the d points in S with the d solutions to $f(x;p) = 0$. As above, numerical methods can be used to track the paths and a certified count on the number of real and nonreal solutions can be obtained, e.g., see [43].

When $p \in \mathbb{R}^k$, the number of complex solutions d can be significantly larger than the number of real solutions to $f(x;p) = 0$. Thus, Section 6 considers a real parameter homotopy aiming to only track real solution paths by trying to stay within each open subset of the parameter space where the number of real solutions is constant. In particular, if the real discriminant locus has smaller dimension, such as in Ex. 2, this is beneficial since it becomes easier to avoid intersecting the real discriminant locus. Therefore, our learning of the real discriminant locus in Section 3 and sampling scheme in Section 4 is only concerned with the codimension 1 boundaries in $\mathbb{R}^k$.

2.2 Pseudowitness point sets

The key to the sampling method in Section 4 is to utilize domain knowledge from the complex discriminant locus to select sample points to guide the learning of the real discriminant locus. Rather than computing a polynomial defining the complex discriminant locus, which can often be a computationally challenging problem, the method in Section 4 computes a pseudowitness point set [41, 42] by intersecting the complex discriminant locus with a real line. This permits one to perform geometric computations on the complex discriminant locus without explicitly needing to compute its defining equation.

When all d solution paths remain finite when performing a parameter homotopy using Eq. (3) for every $p \in \mathbb{C}^k$, then one can compute a pseudowitness point set for the complex discriminant locus as follows. For $f(x;p)$ as in Eq. (2), consider the system

$$F(x,p) = \begin{bmatrix} f(x;p) \\ \det J_x f(x;p) \end{bmatrix}.$$

Let $V \subset \mathbb{C}^{n+k}$ be the algebraic variety defined by $F(x,p) = 0$, $\pi(x,p) = p$ be the projection map onto the parameters, and $\mathcal{L} \subset \mathbb{C}^k$ be a general line. Then, the pseudowitness point set for V

with respect to the projection map π and line $\mathcal{L}$ is $\pi(V) \cap \mathcal{L}$. The number of points in $\pi(V) \cap \mathcal{L}$ is the degree of the complex discriminant locus. One can treat the coefficients of the line $\mathcal{L}$ as parameters and utilize a parameter homotopy to deform the line $\mathcal{L}$ to compute a pseudowitness point set corresponding to other lines in $\mathbb{C}^k$. Hence, this provides a method for sampling points on the complex discriminant locus. Note that a null space approach $J_x f(x;p) \cdot w$ for $w \in \mathbb{P}^{n-1}$ may be used instead of $\det J_x f(x;p)$ [9].

If some solution paths of a parameter homotopy diverge to infinity, then one can projectivize the variables x to compactify the fiber over each parameter point p, e.g., see [42, § 3], and then proceed as above.

3 Machine learning

Parameter homotopies discussed in Section 2.1 provide a means for counting the number of real solutions corresponding to a given parameter value. Indeed, there are other options such as using Hermite matrices [6, 54]. Machine learning techniques can use the number of real solutions as labels and make predictions about previously unseen parameter points. This setup follows a supervised learning paradigm in machine learning since the labels are known for training data. Moreover, approximating the real discriminant locus is equivalent to approximating the decision boundaries between different classes. Since Section 6 applies the learned boundaries to construct a real parameter homotopy which requires knowing the real solutions rather than just the number of them, we utilize parameter homotopies in our computations.

The following describes the leveraged machine learning techniques, namely k-nearest neighbors (k-NN), support vector classifiers (SVC) and feedforward neural networks.

3.1 k-nearest neighbors

The underlying premise of a nearest neighbor classification algorithm is that the class to which a previously unseen data sample belongs can be inferred from the class to which the most similar samples in the training set belong. In our context, similarity will be measured in the form of the Euclidean distance using k samples in the training set nearest to the test sample thereby yielding the k-nearest neighbors. The label assigned to the previously unseen data sample is simply the class to which to the majority of the k-nearest neighbors belong.

In addition to being easy to implement, a 1-nearest neighbor classification algorithm has desirable properties to our problem. In particular, the Bayes error rate is the lowest misclassification rate achievable by any classifier on the associated data [30, 72]. Since the labels are deterministic and the classes do not overlap for our problem, the Bayes error rate is equal to 0. This is summarized in the following.

Theorem 1. *Provided the parameter space is sampled densely enough, no other classifier will outperform a 1-nearest neighbor classification algorithm for determining the number of real solutions associated with a given parameter point.*

Proof. The result follows from the fact that, as the number of training samples tends to infinity, the error rate of any given classifier is at worst its Bayes error rate [20, 67] with the best possible error rate attainable by any classifier being 0. Since, in this case, the Bayes error rate is indeed 0 due to the non-overlapping nature of the classes, no other classifier can possibly improve upon the asymptotic behavior of the 1-nearest neighbor classifier. □

Clearly, Theorem 1 has significant practical limitations since both the complexity and the storage requirements of naive implementations, i.e., non-tree-based methods, for a 1-nearest neighbor classification algorithm are $\mathcal{O}(k\ell)$ when the parameter space is $\mathbb{R}^k$ and ℓ is the cardinality of the training set [73]. Therefore, implementing a truly optimal version would be unfeasible. One approach to partially overcome these strict computational requirements is by implementing a sampling technique that utilizes domain knowledge as described in Section 4 which can be viewed as a form of selective sampling [26, 57], a type of active learning [2, 68]. This enables us to ameliorate the impact of the trade-off between the number of samples stored and algorithmic performance.

The techniques used in k-nearest neighbor algorithms belong to memory-based classification methods as they require the entire training set to be stored. The next two subsections discuss sparse kernel methods, which classify new inputs based on computations performed on a subset of the training set, followed by parametric methods, which learn a set of parametric classification rules based on the training data and perform decision-making based on the learned rules only without referring back to training samples.

3.2 Support vector classifiers

Perhaps the most popular instance of so-called kernel methods are support vector classifiers [14]. The term kernel refers to a (typically) nonlinear mapping that is effected on the training data points, and classification is performed in the resulting nonlinear space. Kernel mapping has both computational and capacity-related advantages since it enables reasoning in a high-dimensional space (usually higher-dimensional than the original feature space) without explicitly computing the high-dimensional representation of data points. Rather, only inner products between kernel representations of the samples are involved [71]. Inter-class boundaries for SVCs are computed by maximizing the gap between the samples in the different classes.

In real-life scenarios, where it may be difficult to find a representation space in which classes are separable, overfitting less representative samples in the training data, in particular those that cross inter-class boundaries, typically results in poor generalization abilities of the network to classify unseen data. Since this is usually associated with excessive network capacity, regularization techniques are often implemented [33]. Commonly used regularization techniques include L_1 (Lasso) and L_2 (Ridge) regularization, dropout, and early stopping [12, 15]. We adopt a strategy that goes against this widely accepted principle. The reason for this is that we know *a priori* that the training data originated from counting the number of real solutions to a parameterized system of polynomial equations, which can be certifiably computed as discussed in Section 2.1. The benefit of knowing the provenance of the data is the awareness that the data in question is separable. Therefore, in order to closely approximate the underlying structure of the training data, which closely follows the decision boundaries without inter-class overlap, we deliberately minimize the degree of regularization in our models. In the context of SVCs, this is achieved in practice by choosing a large inverse regularization parameter c.

3.3 Neural networks

Backed by the universal approximation theorem [24, 47], deep learning techniques [12, 55] have garnered significant popularity in recent times based on success in a wide array of applications. In particular, the feedforward neural network, i.e., a multi-layer structure of compositions of activation functions, has been shown to be a universal approximator for any mildly-constrained target function provided that the network parameters (or weights) and the multilayer structure

are chosen appropriately [24, 47]. The layers of compositions of functions manifest the multilayer structure in a network where the depth refers to the number of composition levels.

A practical way to obtain a sensible model and its corresponding weights is to start with a large architecture (as a rule of thumb, as many weights as the number of training data points) and apply an optimization routine, e.g., stochastic gradient descent method, to achieve numerical values of the weights which best approximate the underlying function. As motivated above, we completely forego the use of regularization techniques in the training process of our neural networks.

4 Sampling method

Given a compact subset of the parameter space $\mathbb{R}^k$, one approach to generate sample points is to randomly, e.g., uniformly, select a parameter value and use a parameter homotopy to count the number of real solutions. Such an approach has been applied to a variety of problems, e.g., [8, 43, 63]. Aiming to approximate the real discriminant locus (classification boundaries), the following uses domain knowledge via the complex discriminant locus to find sample points near the boundaries to guide the learning of the boundaries.

For a parameterized polynomial system $f(x;p)$ as in Eq. (2), we start with parameter homotopies for solving $f = 0$ (see Section 2.1) and computing a pseudowitness point set for the complex discriminant locus (see Section 2.2). The sampling method starts with a randomly selected parameter value $p^* \in \mathbb{R}^k$, e.g., uniformly sampled in a compact subset Ω of the parameter space $\mathbb{R}^k$. For simplicity, we assume that Ω is a rectangular box. The parameter homotopy for $f = 0$ is used to count the number of real solutions to $f(x;p^*) = 0$ thereby obtaining the label for p^*.

The key addition in the sampling scheme is to then select a random direction v^* uniformly in $\mathbb{S}^{k-1}$, the unit sphere in $\mathbb{R}^k$. Let $\mathcal{L}^* \subset \mathbb{C}^k$ be the line parameterized by $p^* + \lambda \cdot v^*$ for $\lambda \in \mathbb{C}$. Then, the parameter homotopy for computing a pseudowitness point set for the complex discriminant locus is used to compute the real points in the corresponding pseudowitness point set along $\mathcal{L}^*$ inside of Ω, say $p_1 = p^* + \lambda_1 \cdot v^*, \ldots, p_\ell = p^* + \lambda_\ell \cdot v^*$. Without loss of generality, we can assume $\lambda_1 < \lambda_2 < \cdots < \lambda_\ell$. Compute λ_0 and $\lambda_{\ell+1}$ such that $\lambda_0 < \lambda_1 < \lambda_\ell < \lambda_{\ell+1}$ where $p_0 = p^* + \lambda_0 \cdot v$ and $p_{\ell+1} = p^* + \lambda_{\ell+1} \cdot v$ are the intersection points of $\mathcal{L}^*$ with the boundary of Ω. We note that if one has access to the complex discriminant polynomial D, e.g., using [38, 54, 74], then an alternative to compute $\lambda_1, \ldots, \lambda_\ell$ would be via computing real roots of the univariate polynomial $D(p^* + \lambda \cdot v^*)$.

Along $\mathcal{L}^*$, the complex discriminant locus yields that the number of real solutions is constant on the intervals (p_i, p_{i+1}) contained in $\mathcal{L}^*$ for $i = 0, \ldots, \ell$. Hence, the next step is to determine the number of real solutions associated with each interval (p_i, p_{i+1}). This is accomplished by selecting the midpoint of each interval, namely $m_i = p^* + (\lambda_i + \delta_i/2) \cdot v$ for $i = 0, \ldots, \ell$ and $\delta_i = \lambda_{i+1} - \lambda_i$. The parameter homotopy for $f = 0$ is used to count the number of real solutions of $f(x;m_i) = 0$.

Our sampling scheme takes the midpoints m_i of each interval, which we call "near center" points in the corresponding cell. We add "near boundary" points as follows. Given $\alpha > 0$, the near boundary points are $b_{i,f} = p^* + (\lambda_i + \Delta_i^f) \cdot v$ and $b_{i,b} = p^* + (\lambda_i - \Delta_i^b) \cdot v$ for $i = 1, \ldots, \ell$ where $\Delta_i^f = \min\{\alpha, \delta_i/20\}$ and $\Delta_i^b = \min\{\alpha, \delta_{i-1}/20\}$. Since $b_{i,f} \in (p_i, p_{i+1})$ and $b_{i,b} \in (p_{i-1}, p_i)$, the number of real solutions of $f(x;b_{i,f}) = 0$ and $f(x;b_{i,b}) = 0$ are known from the computation above.

The aim of the near center points is to provide a parameter point sufficiently in the interior of the region in $\mathbb{R}^k$ with the same number of real solutions. The aim of the near boundary points is to help learn the boundary by providing points on either side of the boundary. Of course, one could also explicitly force the learned boundary to pass through the sampled boundary points. However, they are not utilized in Section 5 since the near boundary points provide both interior points of the

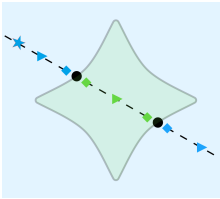


Figure 1: Visual representation of the sampling scheme where the star is a uniform random sample point, circles are points on the boundary, triangles are midpoints, and diamonds are near boundary points. The points are color coded based on the number of real solutions.

corresponding regions as well as guide the learning of the boundary.

In total, our sampling scheme utilized in Section 5 provides three different types of data points: uniform points, near center points, and near boundary points. Figure 1 provides an illustration of these point categories based on a selected uniformly selected sample point (star) along a randomly selected line $\mathcal{L}^*$ (dotted). The boundary points (circles), near center points (triangles), and near boundary points (diamonds) are also shown.

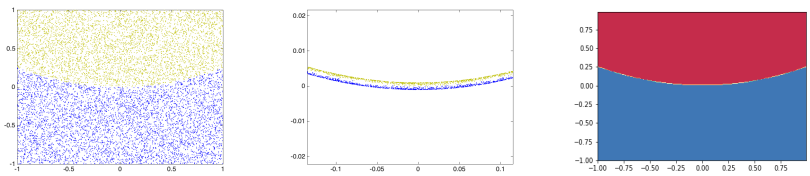
5 Computational setup and results

The sampling method in Section 4 utilizes domain knowledge about the location of the boundary to provide carefully chosen sample points to guide the learning of the boundary which is demonstrated in the following four examples: two warm-up examples utilizing a quadratic and cubic followed by two examples involving the Kuramoto model [1, 29, 70]. The data sets for training and testing for these examples were generated using the sampling scheme and are summarized in Table 1. The computational time for developing the data sets was approximately 6 hours and 5 days for the 3-oscillator and 4-oscillator Kuramoto models, respectively, when computed using a single core of a 2.4 GHz AMD Opteron processor.

	Quadratic	Cubic	Kuramoto $N = 3$	Kuramoto $N = 4$
Uniform	10,000	10,000	972	8,995
Uniform (large)	—	—	8000	—
NearBoundary	12,934	12,022	5,192	54,040
NearBoundary+NearCenter	25,860	22,036	8,440	78,823

Table 1: Number of data points in each data set used for training/testing for each example.

With these data sets, the computational setup for using the one nearest neighbor (1-NN) and SVC-based classification was based on *KNeighborsClassifier* and *svm.SVC* in `SciKit-Learn` [65], respectively, and performed on a laptop with a 2.50 GHz Intel processor and 12 GB RAM. Additionally, a feedforward network was utilized with computations performed on a laptop with a six-core Intel i7 2.60 GHz processor, 32 GB RAM, and Nvidia Quadro P2000 GPU with 4 GB of video RAM. The code was implemented in `PyTorch` [64] leveraging CUDA acceleration. Multi-layer, fully connected feedforward networks with ReLU activation functions [35] were used. A loss function based on multi-class cross-entropy without regularization was optimized during the learning process utilizing an adaptive learning rate scheme.



(a) (b) (c)

Figure 2: (a) Uniform random sampled data, (b) near boundary data, and (c) decision boundary from neural network trained on data from (b) for $f(x; b, c) = x^2 + bx + c$. The blue region has 2 real solutions while the gold and red regions have 0 real solutions.

We note that the problems in this paper use relatively small data sets, the largest being 13.7 MB for the 4-oscillator Kuramoto model. In all instances, the classification of the test data points using the 1-NN models, SVCs, and feedforward neural networks had a computational time of under 1 second. However, the time to train the SVCs and feedforward neural networks took much longer, ranging from seconds to hours and minutes to days, respectively. Due to this computational expense of training, the 1-NN methods were found to be more efficient for the examples covered here.

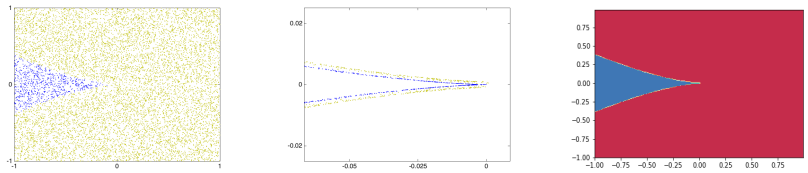
5.1 Quadratic

Consider the quadratic $f(x; b, c) = x^2 + bx + c = 0$ with parameters b and c. This toy system provides a demonstration of the method restricting the parameter space to $[-1, 1]^2$. Of course, the boundary between f having 2 real solutions and 0 real solutions is defined by $b^2 - 4c = 0$. Figure 2(a) plots uniformly selected data in $[-1, 1]^2$ with Figure 2(b) showing the near boundary data.

Table 2 summarizes the performance of the different classifiers on various training and testing data sets. The results in Table 2 were obtained with a traditional 1-NN classifier, a two-class SVC classifier with a radial basis function (RBF) kernel and inverse regularization coefficient $c = 10^6$, and a feedforward, fully connected neural network with three hidden layers each with 20 neurons. We employed tanh as the activation function for the neurons and used a 2-neuron softmax layer as the output layer. A binary cross-entropy loss without regularization was used to train the network and implemented a variable learning rate scheme. Once trained, testing data was used where each of the data points was fed to the network and the classification decision recorded. Figure 2(c) illustrates the decision boundary learned with training data shown in Figure 2(b). This plot was obtained by densely and uniformly sampling the parameter region $[-1, 1]^2$, feeding the resulting samples to the trained network, and color-coding the response of the network for each of the input values in the densely sampled region. The results indicate that including sample data points near the boundary for training produces highly accurate classification results. As points near the boundary are to be classified, the performance declines when training with uniform data particularly with the 1-NN classifier. We attribute the relative robustness of the SVC and neural network classifiers to the sampling method via inductive bias [60] as both of those classifiers tend to learn continuous classification boundaries. In contrast, the 1-NN classifier tends to inherently overfit the training data which often results in boundaries that are not smooth.

Training Data	Testing Data: Uniform	Testing Data: NearBoundary	Testing Data: NearBoundary+NearCenter
Uniform	1 / 1 / 1	0.5392 / 0.9542 / 0.9559	0.7678 / 0.9770 / 0.9779
NearBoundary	0.9999 / 1 / 1	1 / 1 / 1	1 / 1 / 1
NearBoundary +NearCenter	0.9999 / 1 / 1	1 / 1 / 1	1 / 1 / 1

Table 2: Accuracy rate for various testing data sets using various training data sets for the 1-NN method / SVC / feedforward neural network on $f(x; b, c) = x^2 + bx + c$.



(a) (b) (c)

Figure 3: (a) Uniform random sampled data, (b) near boundary data near the cusp, and (c) decision boundary from neural network trained on data from (b) for $f(x; b, c) = x^3 + bx + c$. The blue region has 3 real solutions while the gold and red regions have 1 real solution.

Training Data	Test Data: Uniform	Test Data: NearBoundary	Test Data: NearBoundary+NearCenter
Uniform	1 / 1 / 1	0.5259 / 0.8614 / 0.7848	0.7259 / 0.9116 / 0.8689
NearBoundary	0.9999 / 1 / 1	1 / 0.9931 / 1	1 / 0.9951 / 1
NearBoundary +NearCenter	0.9999 / 1 / 1	1 / 1 / 1	1 / 0.9912 / 1

Table 3: Accuracy rate for various testing data sets using various training data sets for the 1-NN method / SVC / feedforward neural network on $f(x; b, c) = x^3 + bx + c$.

5.2 Cubic

Since the real discriminant locus for the quadratic in Section 5.1 was smooth, we increase the degree to have a cusp on the boundary. In particular, we consider the cubic $f(x; b, c) = x^3 + bx + c = 0$. The boundary between f having 3 real solutions and 1 real solution is defined by $4b^3 + 27c^2 = 0$ which has a cusp at the origin. Figure 3(a) plots uniformly selected data in $[-1, 1]^2$ with Figure 2(b) showing the near boundary data zoomed in near the cusp.

Table 3 summarizes the results obtained by the same classifiers used in Section 5.1. As previously observed, including sample point data near the boundary for training yields higher accuracy. When uniform data is used for training, the accuracy declines when boundary data is included in the testing data set which is particularly evident for the 1-NN classifier. Unlike competing methods, the SVC fails to fully separate the training data due to the somewhat limited capacity of the method which, in turn, bolsters generalization capabilities in the uniformly sampled data case. Figure 3(c) illustrates the boundary learned by the neural network trained on the data from Figure 3(b).

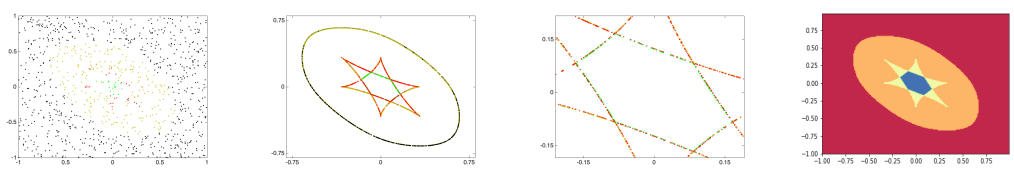
5.3 Kuramoto Model

The Kuramoto model [1, 29, 70] is a popular model to study synchronization phenomena observed in systems consisting of N coupled oscillators. We aim to learn the number of equilibria as a function of the parameters $\omega \in \mathbb{R}^N$ which are the natural frequencies of the N oscillators. The system can be simplified by noting that the sum of the natural frequencies must be zero to have equilibria and has rotational symmetry. The resulting parameterized polynomial system has $2(N-1)$ polynomials and variables with $N-1$ parameters:

$$F(c_1, s_1, \ldots, c_{N-1}, s_{N-1}; \omega_1, \ldots, \omega_{N-1}) = \begin{bmatrix} \omega_i - \frac{1}{N}\sum_{j=1}^{N}(s_i c_j - s_j c_i) & i = 1, \ldots, N-1 \\ c_i^2 + s_i^2 - 1 & \end{bmatrix} = 0. \tag{4}$$

Moreover, if $\omega_i \notin \left[-\frac{N-1}{N}, \frac{N-1}{N}\right]$, then Eq. (4) can have no real solutions so that the parameter space is naturally restricted to a compact subset of $\mathbb{R}^{N-1}$. Furthermore, the number of real solutions is invariant under permutations of the parameters. In particular, we do not label the axes in Figures 4 and 5 since equivalent pictures hold for any labelling.

For generic parameter values, Eq. (4) has $2^N - 2$ solutions [19, Thm. 4.3]. There are a maximum of 6 isolated real solutions for $N = 3$ and there are parameters for any possible even number of solutions, e.g., see Figure 4(d). For $N = 4$, it was conjectured in [76] to have a maximum of 10 isolated real solutions by scanning over of a grid of the parameter space. This conjecture was proven to be correct in [38, Thm. 8.1]. However, a complete characterization of the parameter space based on the number of real solutions for the $N = 4$ case has proved to be a particularly difficult problem for traditional methods such as comprehensive Gröbner basis, cylindrical algebraic decomposition, and homotopy continuation. We note that the complex discriminant locus for the $N = 3$ and $N = 4$ cases is a curve of degree 12 and surface of degree 48, respectively, and both have singularities. In particular, the $N = 4$ Kuramoto model example highlights how the proposed method can be used to understand the parameter space based on the number of real solutions even when the real discriminant locus contains a positive-dimensional set of singularities in a reasonable time frame.



(a) (b) (c) (d)

Figure 4: For the 3-oscillator Kuramoto model: (a) uniformly selected parameter values, (b) data perturbed from the boundary, (c) a zoomed view of data perturbed from the boundary, and (d) decision boundary from neural network trained on data from (b). For (d), the regions are colored based on the number of real solutions: red = 0, orange = 2, yellow = 4, and blue = 6. These same regions are part of (a), (b), and (c) with slightly different colors.

Training Data	Test Data		
	Uniform	NearBoundary	NearBoundary+NearCenter
Uniform	1 / 0.9856 / 1	0.4921 / 0.4962 / 0.5027	0.6525 / 0.6381 / 0.6554
Uniform (large) using 1-NN	1	0.5306	0.6973
NearBoundary	1 / 0.9804 / 1	1 / 0.7867 / 1	0.9985 / 0.8551 / 0.9861
NearBoundary +NearCenter	1 / 0.8533 / 1	1 / 0.7736 / 1	1 / 0.9928 / 1

Table 4: Accuracy rate for various testing data sets using various training data sets for the 1-NN method / SVC / feedforward neural network on the 3-oscillator Kuramoto model.

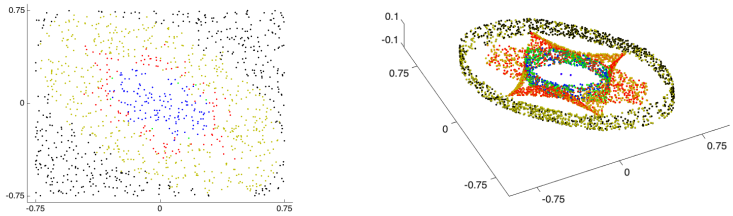
5.3.1 3-oscillators

For $N = 3$, we consider $(\omega_1, \omega_2) \in [-1, 1]^2$. The outcome of a uniform sampling process is provided in Figure 4(a), near boundary data points in Figure 4(b), and a zoomed in version of near boundary points in Figure 4(c). Table 4 includes results achieved by the different classifiers on this data. For the 1-NN classifier, and due to the increasing difference in size of the data sets, tests were completed to determine whether training with a much smaller data set and testing with a data set on the order of ten times larger impacted the accuracy results. To achieve this, the original uniform data set of approximately 1,000 data points as well as a uniform data set of 8,000 data points were used for training while data sets of approximately 1,000 (Uniform), 5,000 (NearBoundary), and 8,000 (NearBoundary+NearCenter) were used for testing. As summarized in Table 4, the accuracy does not drastically change when the size of data sets is comparable. Most importantly, it does not change the conclusion that including near boundary data in the training data set yields highest accuracy across all testing data sets. A multi-class SVC with RBF kernel and inverse regularization parameter $c = 10^{10}$ was implemented. A feedforward, fully connected neural network with five hidden layers each with 20 neurons was used. We employed the ReLU activation function for the neurons in the hidden layers and used a 4-neuron softmax layer as the output layer since this is a 4-class classification task corresponding to 0, 2, 4, and 6 real solutions. In this case, the limited capacity of SVCs prevented fully learning the decision boundary while the competing methods perform similarly. As before, performance is lackluster on boundary data in algorithms trained only using uniform data.

5.3.2 4-oscillators

Similar computations were performed on the 4-oscillator Kuramoto model, which has a three-dimensional parameter space. Following the theoretical bounds, we only considered sample points in $[-3/4, 3/4]^3$. Figure 5(a) shows a two-dimensional slice of the parameter space using uniformly selected points, while Figure 5(b) illustrates some of the near boundary data. Table 5 summarizes the results when using the classifiers from the previous section.

In our experiment, it became apparent that the neural network was unable to fully separate the data samples with correct labels for some of the near boundary points. We hypothesize that, although learning converged, it likely reached a local minimum in the optimization landscape. As the dimensionality of the data and the number of training data points grow, the complexity of the optimization landscape increases which makes it less likely to reach the global minimum or at least one that is truly optimal. This scenario is worsened by the absence of a regularization term where it is empirically known [59] that the number of local minima in the optimization landscape of a



(a) (b)

Figure 5: For the 4-oscillator Kuramoto model: (a) uniformly selected parameter values on a 2D slice, and (b) some of the data perturbed from the boundary colored based on the number of real solutions: black = 0, gold = 2, red = 4, green = 6, blue = 8, and magenta = 10.

	Test Data		
Training Data	Uniform	NearBoundary	NearBoundary+NearCenter
Uniform	1 / 0.9901	0.4630 / 0.5556	0.5911 / 0.6539
NearBoundary	0.9782 / 0.9730	1 / 0.7505	0.9901 / 0.8059
NearBoundary+NearCenter	0.9869 / 0.9901	1 / 0.7537	1 / 0.8190

Table 5: Accuracy rate for various testing data sets using various training data sets for the 1-NN method / SVC on the 4-oscillator Kuramoto model.

network decreases as stronger regularization is enforced.

6 Real parameter homotopy leveraging learned boundaries

The examples presented in Section 5 show that machine learning techniques coupled with the sampling scheme from Section 4 produce accurate results for classifying, i.e., predicting the number of real solutions, over the parameter space. Often in science and engineering applications, one is not only interested in the number of real solutions, but actually computing the real solutions. Typically, for these applications, the number of real solutions is significantly smaller than the number of complex solutions, so developing a parameter homotopy that only tracks real solution paths can drastically reduce the computational time. The key to developing such a real parameter homotopy is to track along a segment in the parameter space which does not intersect the real discriminant locus. Thus, after learning, one can develop a robust and efficient real parameter homotopy setup as follows that we demonstrate on the 3-oscillator and 4-oscillator Kuramoto model.

Given a real parameter $p \in \mathbb{R}^k$, the real parameter homotopy method uses the nearest neighbor method to select the closest parameter point p^* to p in the sampled (training) data set. Since the real solutions for $f(x; p^*) = 0$ have already been computed, one only tracks the solutions paths starting at real solutions for the homotopy defined by

$$H(x, t) = f(x; t \cdot p^* + (1 - t) \cdot p) = 0,$$

which is simply Eq. (3) with $\gamma = 1$. Therefore, if the line segment $[p^*, p]$ does not intersect the real discriminant locus, then there is a bijection between the real solutions of $f(x; p) = 0$ and

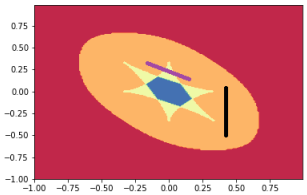


Figure 6: Illustration of two segments added to Figure 4(d), one (black) which is guaranteed to succeed while the other (purple) may fail since it intersects the real discriminant locus. See the caption of Figure 4 for a description of the color convention used.

Number of data points	Number of paths	Average time (in seconds)	Success rate
249	2	0.077	100%
26	4	0.081	100%
17	6	0.086	100%

Table 6: Average computation time for finding all real roots for the 3-oscillator Kuramoto model using a machine-learning-assisted real parameter homotopy method.

$f(x; p^*) = 0$, and every real solution path of $H = 0$ is smooth for $t \in [0, 1]$. Using sample points via the sampling scheme in Section 4 on either side of the boundary aims to increase the chance the segment between p and the nearest sample point p^* is contained in the same region and thus this real parameter homotopy method succeeds. Figure 6 is Figure 4(d) with two added segments. One segment (black) is within the same region so that the real parameter homotopy method would succeed. Although the other segment (purple) has endpoints with the same region, there is no guarantee of success since it intersects the real discriminant locus.

6.1 3-oscillators

As an illustration, consider the 3-oscillator Kuramoto model. Since, from Section 5.3, the generic number of complex solutions is 6, one of course can easily track all 6 complex solution paths using a classical parameter homotopy in Eq. (3). In our experiment, using a single core of a 2.4 GHz AMD Opteron Processor, this took on average 1.33 seconds. Nonetheless, we utilize this as a test case to show some improvement as well as analyzing the success rate which was determined by comparing using a classical parameter homotopy with this machine learning assisted real parameter homotopy. Table 6 shows that, on average, the real parameter homotopy took less than 0.1 seconds and was successful on every randomly selected parameter value tested. One reason for the order of magnitude reduction in computational time is that, by selecting the closest parameter value, the homotopy solution paths are much shorter and thus faster to track.

6.2 4-oscillators

Following a similar setup, we also applied the method to the 4-oscillator Kuramoto model. In this case, the generic number of complex solutions is 14, but the maximum number of real solutions is 10 showing that there will always be wasted computational effort when computing the real solutions using a classical parameter homotopy. In our experiment, the average time for tracking the 14

Number of data points	Number of paths	Average time (in seconds)	Success rate
30504	2	0.114	98.2%
17088	4	0.121	98.8%
9041	6	0.126	99.2%
4383	8	0.128	98.0%
345	10	0.132	100%

Table 7: The average computation time for finding all real roots for the 4-oscillator Kuramoto model using a machine learning assisted real parameter homotopy method.

complex solution paths using a classical parameter homotopy was 3.40 seconds. Table 7 summarizes the results that again show over an order of magnitude reduction in computational time with a success rate in accordance with the classification accuracy in Table 5.

7 Outlook and conclusions

This paper provides a novel viewpoint on the mathematical problem of identifying the boundaries, called the real discriminant locus, of the parameter space that separate the regions corresponding to different number of real solutions to a parameterized polynomial system. Although there is a discriminant polynomial which vanishes on the real discriminant locus, it can be difficult to compute, facilitating the need to numerically approximate it. Our approach is based on the correspondence between the real discriminant locus and decision boundaries of a supervised classification problem in machine learning. By utilizing domain knowledge from numerical algebraic geometry, we developed a sampling strategy for selecting points near the boundary to assist the machine learning techniques in providing an accurate approximation of the boundary. With a parameter homotopy, one is able to accurately label the data so that there is no noise in the data. Hence, no regularization techniques need to be utilized, which would have forced the algorithm to strictly learn only smooth boundaries, which is important since singularities often arise as illustrated in Section 5.

One challenge with using deep networks to learn a real discriminant locus is how to properly select the number of layers and neurons within each layer needed to develop an accurate approximation. We utilized hyperparameter optimization to search for reasonable choices along with stochastic gradient descent methods to determine weights to fit the data. Another challenge is the presence of singularities which seem to make training more difficult for deep networks. Therefore, these type of problems provide a unique benchmarking opportunity for multi-class machine learning algorithms as the ground truth regarding both labels and classification boundaries can be explicitly computed for some examples, such as univariate polynomials as in Sections 5.1 and 5.2. We overcome some of these difficulties by developing a sampling scheme that produces significantly more points near the boundaries than in other areas of the parameter space so that one is able to quickly obtain an accurate approximation of the real discriminant locus.

When deep networks can take an inordinate amount of time to train, one can utilize local approximation methods such as k-nearest neighbor classification algorithm. In fact, as shown in Theorem 1, no classifier can outperform the 1-nearest neighbor classification algorithm provided that the parameter space is sampled densely enough. The examples in Section 5 show that deep networks can be useful and comparable to the 1-NN methods. However, the data confirms the effectiveness of the 1-NN methods, especially in the case of the $N = 4$ Kuramoto model when the deep network did

not converge to the global minimum.

Although our proposed sampling method can be viewed as active learning, one can also employ a more explicit active learning approach where an algorithm interactively queries the parameter space and samples more densely near singularities such as cusps and other difficult regions. One could also attempt to first construct an algorithm to remove ϵ neighborhoods surrounding all singularities, learn the remaining parameter space and real discriminant locus, and then take $\epsilon \to 0$. These approaches will be explored in the future.

The curse of dimensionality hampers most computational methods, including machine learning. In computational algebraic geometry, the actual dimension where problems become intractable is strictly problem specific. When identifying the real discriminant locus for parameterized polynomial systems, this brings in its own difficulties that are different from other applications such as those in computer vision and natural language processing where the curse of dimensionality may kick in at much larger dimensions. The previous best works [17, 37] proposed an approach based on homotopy continuation which could analyze the $N = 3$ Kuramoto model, while the $N = 4$ case was still out of reach. In the present work, we have now broken this ceiling with the combination of machine learning and homotopy continuation.

A real parameter homotopy method that tracks only real solution paths was developed in Section 6 as an application of learning the real discriminant locus. Even for relatively small problems, this method reduced the computational time by over an order of magnitude. After generating sample data "offline," this method is easy to implement in an "online" solver which could drastically improve the computation of real solutions. With proper adjustments, this method is extensible to other situations involving rational, exponential, logarithmic, trigonometric, and piecewise functions.

Acknowledgments

The paper is a result of exploratory and fundamental research, and statements made in it are Dhagash Mehta's and his co-authors' personal views which do not represent The Vanguard Group's views. The authors thank Martin Poláček for insightful comments. JDH was supported in part by NSF grant CCF-1812746 and ONR N00014-16-1-2722. MHR was supported in part by Schmitt Leadership Fellowship in Science and Engineering and NSF grant CCF-1812746.

References

[1] J. A. Acebrón, L. L. Bonilla, C. J. P. Vicente, F. Ritort, and R. Spigler. The Kuramoto model: A simple paradigm for synchronization phenomena. *Reviews of modern physics*, 77(1):137, 2005.

[2] C. C. Aggarwal, X. Kong, Q. Gu, J. Han, and P. S. Yu. Chapter 22 active learning: A survey, 2014.

[3] E. L. Allgower and K. Georg. *Numerical continuation methods: an introduction*, volume 13. Springer Science & Business Media, 2012.

[4] A. Andoni, R. Panigrahy, G. Valiant, and L. Zhang. Learning polynomials with neural networks. In *International Conference on Machine Learning*, pages 1908–1916, 2014.

[5] A. Andoni, R. Panigrahy, G. Valiant, and L. Zhang. Learning sparse polynomial functions. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 500–510. Society for Industrial and Applied Mathematics, 2014.

[6] T. Ayyildiz Akoglu and A. Szanto. Certified Hermite matrices from approximate roots - univariate case. In D. Slamanig, E. Tsigaridas, and Z. Zafeirakopoulos, editors, *Mathematical Aspects of Computer and Information Sciences*, pages 3–9, Cham, 2020. Springer International Publishing.

[7] S. Basu, R. Pollack, and M. F. Roy. *Algorithms in Real Algebraic Geometry.* Springer, 2003.

[8] D. J. Bates, D. A. Brake, and M. Niemerg. Paramotopy: Parameter homotopies in parallel. In *Mathematical Software – ICMS 2018*, pages 28–35, Cham, 2018. Springer International Publishing.

[9] D. J. Bates, J. D. Hauenstein, C. Peterson, and A. J. Sommese. *Numerical Decomposition of the Rank-Deficiency Set of a Matrix of Multivariate Polynomials*, pages 55–77. Springer Vienna, Vienna, 2010.

[10] D. J. Bates, J. D. Hauenstein, and A. J. Sommese. Efficient path tracking methods. *Numer. Algorithms*, 58(4):451–459, 2011.

[11] D. J. Bates, J. D. Hauenstein, A. J. Sommese, and C. W. Wampler. *Numerically solving polynomial systems with Bertini*, volume 25. SIAM, 2013.

[12] Y. Bengio, I. J. Goodfellow, and A. Courville. Deep learning. *MIT Press.*, 2015.

[13] F. Bihan, A. Dickenstein, and M. Giaroli. Lower bounds for positive roots and regions of multistationarity in chemical reaction networks. *arXiv preprint arXiv:1807.05157*, 2018.

[14] C. M. Bishop. *Pattern recognition and machine learning.* Springer, 2006.

[15] C. M. Bishop. *Pattern recognition and machine learning.* Springer Science+ Business Media, 2006.

[16] P. Breiding, S. Kališnik, B. Sturmfels, and M. Weinstein. Learning algebraic varieties from samples. *Revista Matemática Complutense*, 31(3):545–593, 2018.

[17] S. Chandra, D. Mehta, and A. Chakrabortty. Locating power flow solution space boundaries: A numerical polynomial homotopy approach. *arXiv preprint arXiv:1704.04792*, 2017.

[18] C. Conradi, E. Feliu, M. Mincheva, and C. Wiuf. Identifying parameter regions for multistationarity. *PLoS computational biology*, 13(10):e1005751, 2017.

[19] O. Coss, J. D. Hauenstein, H. Hong, and D. K. Molzahn. Locating and counting equilibria of the Kuramoto model with rank-one coupling. *SIAM Journal on Applied Algebra and Geometry*, 2(1):45–71, 2018.

[20] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, January 1967.

[21] D. A. Cox, J. Little, and D. O'Shea. *Using Algebraic Geometry.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1998.

[22] D. A. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra, 3/e (Undergraduate Texts in Mathematics).* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[23] D. A. Cox, with contributions from C. D'Andrea, A. Dickenstein, J. Hauenstein, H. Schenck, and J. Sidman. *Applications of Polynomial Systems*. CBMS Regional Conference Series in Mathematics. Conference Board of the Mathematical Sciences, 2020.

[24] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[25] M. Das and P. Seal. Polynomial real roots finding using feed forward neural network: a simple approach. *2012 National Conference on Computing and Communication Systems, IEEE*, Nov 21:1–4, 2012.

[26] S. Dasgupta. Consistency of nearest neighbor classification under selective sampling. In S. Mannor, N. Srebro, and R. C. Williamson, editors, *Proceedings of the 25th Annual Conference on Learning Theory*, volume 23 of *Proceedings of Machine Learning Research*, pages 18.1–18.15, Edinburgh, Scotland, 25–27 Jun 2012. PMLR.

[27] A. Dhooge, W. Govaerts, and Y. A. Kuznetsov. MATCONT: a MATLAB package for numerical bifurcation analysis of ODEs. *ACM Transactions on Mathematical Software (TOMS)*, 29(2):141–164, 2003.

[28] E. J. Doedel. Auto: A program for the automatic bifurcation analysis of autonomous systems. *Congr. Numer*, 30(265-284):25–93, 1981.

[29] F. Dörfler and F. Bullo. Synchronization in complex networks of phase oscillators: A survey. *Automatica*, 50(6):1539–1564, 2014.

[30] K. Fukunaga. *Introduction to Statistical Pattern Recognition (2Nd Ed.)*. Academic Press Professional, Inc., San Diego, CA, USA, 1990.

[31] I. M. Gelfand, M. M. Kapranov, and A. V. Zelevinsky. *Discriminants, resultants and multidimensional determinants*. Modern Birkhäuser Classics. Birkhäuser Boston, Inc., Boston, MA, 2008. Reprint of the 1994 edition.

[32] M. Giaroli, F. Bihan, and A. Dickenstein. Regions of multistationarity in cascades of Goldbeter–Koshland loops. *Journal of mathematical biology*, 78(4):1115–1145, 2019.

[33] I. J. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[34] B. Greene, D. Kagan, A. Masoumi, D. Mehta, E. J. Weinberg, and X. Xiao. Tumbling through a landscape: Evidence of instabilities in high-dimensional moduli spaces. *Physical review d*, 88(2):026005, 2013.

[35] R. H. R. Hahnloser, H. S. Seung, and J.-J. Slotine. Permitted and forbidden sets in symmetric threshold-linear networks. *Neural Comput.*, 15(3):621–638, Mar. 2003.

[36] W. Hanan, D. Mehta, G. Moroz, and S. Pouryahya. Stability and bifurcation analysis of coupled Fitzhugh–Nagumo oscillators. *arXiv preprint arXiv:1001.5420*, 2010.

[37] H. A. Harrington, D. Mehta, H. M. Byrne, and J. D. Hauenstein. Decomposing the parameter space of biological networks via a numerical discriminant approach. In J. Gerhard and I. Kotsireas, editors, *Maple in Mathematics Education and Research*, pages 114–131, Cham, 2020. Springer International Publishing.

[38] K. Harris, J. D. Hauenstein, and A. Szanto. Smooth points on semi-algebraic sets. *ACM Commun. Comput. Algebra*, 54(3):105–108, 2021.

[39] J. D. Hauenstein and M. H. Regan. Adaptive strategies for solving parameterized systems using homotopy continuation. *Applied Mathematics and Computation*, 332:19–34, 2018.

[40] J. D. Hauenstein and M. H. Regan. Real monodromy action. *Applied Mathematics and Computation*, 373:124983, 2020.

[41] J. D. Hauenstein and A. J. Sommese. Witness sets of projections. *Applied Mathematics and Computation*, 217(7):3349–3354, 2010.

[42] J. D. Hauenstein and A. J. Sommese. Membership tests for images of algebraic sets by linear projections. *Applied Mathematics and Computation*, 219(12):6809–6818, 2013.

[43] J. D. Hauenstein and F. Sottile. Algorithm 921: AlphaCertified: Certifying solutions to polynomial systems. *ACM Trans. Math. Softw.*, 38(4), 2012.

[44] J. D. Hauenstein and C. W. Wampler. Isosingular sets and deflation. *Found. Comput. Math.*, 13(3):371–403, 2013.

[45] Y.-H. He, D. Mehta, M. Niemerg, M. Rummel, and A. Valeanu. Exploring the potential energy landscape over a large parameter-space. *Journal of High Energy Physics*, 2013(7):50, 2013.

[46] E. A. Hernandez-Vargas, D. Mehta, and R. H. Middleton. Towards modeling HIV long term behavior. *IFAC Proceedings Volumes*, 44(1):581–586, 2011.

[47] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[48] D.-S. Huang. Constrained learning algorithms for finding the roots of polynomials: A case study. In *2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering. TENCOM'02. Proceedings.*, volume 3, pages 1516–1520. IEEE, 2002.

[49] D.-S. Huang. A constructive approach for finding arbitrary roots of polynomials by neural networks. *IEEE Transactions on Neural Networks*, 15(2):477–491, 2004.

[50] D.-S. Huang and Z. Chi. Neural networks with problem decomposition for finding real roots of polynomials. In *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No. 01CH37222)*, page A25. IEEE, 2001.

[51] D.-S. Huang, H. H. S. Ip, and Z. Chi. A neural root finder of polynomials based on root moments. *Neural Computation*, 16(8):1721–1762, 2004.

[52] Z. Huang, M. England, D. Wilson, J. H. Davenport, and L. C. Paulson. Using machine learning to improve cylindrical algebraic decomposition. *Mathematics in Computer Science*, 13:1–28, 12 2019.

[53] D. Lazard and F. Rouillier. Solving parametric polynomial systems. *J. Symb. Comput.*, 42(6):636–667, 2007.

[54] H. Le and M. Safey El Din. Solving parametric systems of polynomial equations over the reals through Hermite matrices. *Journal of Symbolic Computation*, 112:25–61, 2022.

[55] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[56] A. Leykin, J. Verschelde, and A. Zhao. Newton's method with deflation for isolated singularities of polynomial systems. *Theoret. Comput. Sci.*, 359(1-3):111–122, 2006.

[57] M. Lindenbaum, S. Markovitch, and D. Rusakov. Selective sampling for nearest neighbor classifiers. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence*, AAAI '99/IAAI '99, pages 366–371, Menlo Park, CA, USA, 1999. American Association for Artificial Intelligence.

[58] D. Martinez-Pedrera, D. Mehta, M. Rummel, and A. Westphal. Finding all flux vacua in an explicit example. *JHEP*, 1306:110, 2013.

[59] D. Mehta, X. Zhao, E. A. Bernal, and D. J. Wales. The loss surface of XOR artificial neural networks. *Phys. Rev. E*, 97:052307, 2018.

[60] T. M. Mitchell. The need for biases in learning generalizations. Technical report, Rutgers University, New Brunswick, NJ, 1980.

[61] A. P. Morgan and A. J. Sommese. Coefficient-parameter polynomial continuation. *Applied Mathematics and Computation*, 29(2):123–160, 1989.

[62] B. Mourrain, N. G. Pavlidis, D. K. Tasoulis, and M. N. Vrahatis. Determining the number of real roots of polynomials through neural networks. *Computers & Mathematics with Applications*, 51(3-4):527–536, 2006.

[63] K.-M. Nam, B. M. Gyori, S. V. Amethyst, D. J. Bates, and J. Gunawardena. Robustness and parameter geography in post-translational modification systems. *PLOS Computational Biology*, 16(5):1–50, 2020.

[64] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[65] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[66] S. Perantonis, N. Ampazis, S. Varoufakis, and G. Antoniou. Constrained learning in neural networks: Application to stable factorization of 2-D polynomials. *Neural Processing Letters*, 7(1):5–14, 1998.

[67] B. D. Ripley and N. L. Hjort. *Pattern Recognition and Neural Networks*. Cambridge University Press, New York, NY, USA, 1st edition, 1995.

[68] B. Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[69] A. J. Sommese and C. W. Wampler. *The Numerical Solution of Systems of Polynomials Arising in Engineering and Science*. World Scientific Publishing, Hackensack, NJ, 2005.

[70] S. H. Strogatz. From Kuramoto to Crawford: exploring the onset of synchronization in populations of coupled oscillators. *Physica D: Nonlinear Phenomena*, 143(1-4):1–20, 2000.

[71] S. Theodoridis and K. Koutroumbas. *Pattern recognition; 4th ed.* Elsevier, Burlington, 2008.

[72] K. Tumer and J. Ghosh. Estimating the Bayes error rate through classifier combining. In *Proceedings of 13th International Conference on Pattern Recognition*, volume 2, pages 695–699 vol.2, Aug 1996.

[73] R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proceedings of the 24rd International Conference on Very Large Data Bases*, VLDB '98, pages 194–205, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[74] V. Weispfenning. Comprehensive Gröbner bases. *Journal of Symbolic Computation*, 14(1):1–29, 1992.

[75] B. Xia. Discoverer: a tool for solving semi-algebraic systems. *ACM Communications in Computer Algebra*, 41(3):102–103, 2007.

[76] X. Xin, T. Kikkawa, and Y. Liu. Analytical solutions of equilibrium points of the standard Kuramoto model: 3 and 4 oscillators. In *2016 American Control Conference (ACC)*, pages 2447–2452. IEEE, 2016.