# Machine learning the real discriminant locus

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

July 14, 2020

**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. For multidimensional parameter spaces, this article presents a novel sampling method which carefully samples the 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 neighbor and deep learning 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 the 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 Kuramoto model.

**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 [9, 18, 19, 20, 60]. In many of these applications, the coefficients of the equations depend upon one or more parameters

*Rochester Data Science Consortium, University of Rochester, Rochester, NY 14627 (edgar.bernal@rochester.edu).

†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 (dhagashbmehta@gmail.com).

§Department of Applied and Computational Mathematics and Statistics, University of Notre Dame, Notre Dame, IN 46556 (mregan9@nd.edu, `www.nd.edu/~mregan9`). 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. 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 a given value of the parameters.

Due to the ubiquity of the problem, 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 [28]. For example, the discriminant of

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

is $D = b^2 - 4ac$. Since real parameters and the number of real solutions are of most interest in applications, we take the real discriminant locus to be the boundary in the parameter space where the number of real solutions change. For the quadratic expression in Eq. (1), the solution set in $\mathbb{R}^3$ of $D = 0$ is the real discriminant locus which is the boundary between the region in $\mathbb{R}^3$ with $D > 0$ where $f = 0$ has two real solutions and the region in $\mathbb{R}^3$ with $D < 0$ where $f = 0$ has no real solutions.

Comprehensive Gröbner basis computations [64] can be used to symbolically compute the discriminant polynomial over the complex numbers whose solution set is called the classical 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 [6, 33, 42, 49, 65], Brouwer degree [15], and polyhedral methods [11, 29] 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, 9, 60] have been employed in which one tracks the solution structure as the parameters are varied continuously. Several computational packages such as AUTO [25] and MATCONT [24] employ such techniques for parameterized differential equations. Rather than directly run into the real discriminant locus, a perturbed sweeping approach was presented in [34]. 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 a selected sample points in the parameter space [7, 14, 31, 41, 52].

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 [9, 60] to help guide the approximation of the decision boundaries.

Although there has been a collection of papers, e.g., [22, 44, 45, 46, 47, 48, 57], attempting to solve polynomial equations and improving algorithms using neural networks, the approach closest to the present work is [55]. This method 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. Their results indicate that the ability of an artificial neural network to generalize on the test sets are comparable to their performance on the training sets. They employed several algorithms to train the

network and concluded that when the polynomial has high degree, the choice of training program impacts the performance of the network.

Other applications of deep networks for analyzing polynomial equations include [4, 5] which investigate 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 [22]. Finally, [13] 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 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 loci 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. Design a real homotopy method that utilizes an approximation of the real discriminant locus to track only real 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 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. Finally, a conclusion is provided in Section 7.

## 2 Numerical algebraic geometry

The following provides a short description of two topics in numerical algebraic geometry, namely parameter homotopies and pseudowitness point sets, that will be used to learn the real discriminant locus. More details regarding numerical algebraic geometry are provided in [9, 60].

### 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 $p^* \in \mathbb{C}^k$, $f(x;p^*) = 0$ has finitely many solutions 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 $x$. See [36] and the references therein for reducing overdetermined parameterized systems to well-constrained parameterized systems adhering to Eq. (2). With this setup, the real parameter space $\mathbb{R}^k$ contains open subsets where the number of real solutions to $f(x;p) = 0$ is constant and 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* $\mathbb{C}$. *As mentioned in the Introduction, the real parameter space* $\mathbb{R}^3$ *contains two open subsets:* $D > 0$ *and* $D < 0$ *where* $D = b^2 - 4ac$ *in which the number of real solutions is constant, namely* 2 *and* 0. *The real discriminant locus is the set* $D = 0$ *in* $\mathbb{R}^3$.

The classical discriminant locus consists of the parameter points in $\mathbb{C}^p$ where $f(x;p) = 0$ does not have $d$ solutions. Since $f(x;p)$ is well-constrained, the classical discriminant locus is either empty or has codimension 1 in $\mathbb{C}^p$, i.e., a hypersurface. This will be exploited in Section 4 to generate sample points since the real discriminant locus is contained in the classical discriminant locus. The following example illustrates this relationship and shows that the real discriminant locus could be of smaller dimension in $\mathbb{R}^p$.

**Example 2.** *Consider the following parameterized polynomial system from [37, Ex 2.1]*

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

*which generically has* $d = 4$ *solutions. In fact,* $p_1^2 + p_2^2 = 0$ *in* $\mathbb{C}^2$ *is the classical discriminant locus. Thus, the classical discriminant locus is a curve in* $\mathbb{C}^2$, *but 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 indeed* $\{(0,0)\}$.

Given $p \in \mathbb{C}^k$ outside of the classical discriminant locus, the goal is to compute the solutions to $f(x;p) = 0$ which will be accomplished using a parameter homotopy [54]. Suppose that one knows $p^* \in \mathbb{C}^k$ and a set $S \subset \mathbb{C}^n$ consisting of the $d$ solutions to $f(x;p^*) = 0$. Therefore, one has the parameter homotopy

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

where $t \in [0,1]$, $\gamma \in \mathbb{C}$, and

$$\tau(t) = \frac{\gamma t}{1 + (\gamma - 1)t}.$$

In particular, $H(x,1) = f(x;p^*) = 0$ has known solutions $S$ 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 $t \in [0,1]$ that connects $p^*$ to $p$ avoids the classical discriminant locus so that $H(x,t) = 0$ defines $d$ solution paths for $t \in [0,1]$ which start at the $d$ points in $S$ and end at the $d$ solutions of $f(x;p) = 0$. The paths can be traversed using a variety of numerical methods [9] and a certified count on both the number of real and nonreal solutions can be obtained using [40].

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 discriminant locus in Section 3 and sampling scheme in Section 4 is only concerned with the codimension 1 boundaries in $\mathbb{R}^k$.

In order to utilize a parameter homotopy, one first needs to compute the solution set $S$ to $f(x;p^*)=0$. This can be accomplished by treating the polynomial system $f(x;p^*)$ as a member of another parameterized family and performing a parameter homotopy in that family. Classical examples include constructing a parameterized family based on the degrees of $f_i$, the multihomogeneous structure of $f$, and the monomial structure of $f$, e.g., see [60, Chap. 8].

### 2.2 Pseudowitness point sets

The key to the sampling method in Section 4 is to utilize domain knowledge from the classical discriminant locus to select sample points to guide the learning of the real discriminant locus. Rather than computing a polynomial defining the classical discriminant locus, which can often be a computationally challenging problem, the method in Section 4 computes a pseudowitness point set [38, 39] by intersecting with a real line.

For $f(x;p)$ as in Eq. (2), consider the system

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

Suppose that $V\subset\mathbb{C}^{n+k}$ is the solution set of $F(x,p)=0$, $\pi(x,p)=p$, and $\mathcal{L}\subset\mathbb{C}^k$ is a general line. Then, the pseudowitness point set for $V$ with respect to the projection map $\pi$ and line $\mathcal{L}$ is $\pi(V)\cap\mathcal{L}$. In fact, the degree of the classical discriminant locus is the number of points in $\pi(V)\cap\mathcal{L}$. Moreover, by using a parameter homotopy, one can deform the line $\mathcal{L}$ to compute a pseudowitness point set for other lines in $\mathbb{C}^k$ providing a method for sampling points on the classical discriminant locus.

Instead of $\det J_x f(x;p)$, one may also use a null space approach $J_x f(x;p)\cdot w$ for $w\in\mathbb{P}^{n-1}$ [8].

## 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. This creates labels for training data which, with machine learning techniques, can be used to 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. The following describes using two machine learning techniques, namely $K$-nearest neighbors and and feedforward neural networks (more popularly known as deep 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 [27, 62]. 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.

**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 [17, 58] 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 $\ell$ is the cardinality of the training set [63]. 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 [23, 51], a type of active learning [2, 59]. This enables us to ameliorate the impact of the trade-off between the number of samples stored and algorithmic performance.

Another approach for improving performance is to perform preliminary computations on the training data. One such method is to train a neural network as discussed next.

### 3.2 Deep networks

Backed by the universal approximation theorem [21, 43], deep learning techniques [10, 50] 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 [21, 43]. The layers of compositions of functions manifests the multilayer structure in a deep 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. Since overfitting based on noise in the training data typically results in poor generalization abilities of the network to classify unseen data and is usually associated with excessive network capacity, regularization techniques are often implemented [30]. Commonly used regularization techniques include $L_1$ (Lasso) and $L_2$ (Ridge) regularization, dropout, and early stopping [10, 12].

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 will not be affected by noise. Therefore, in order to closely approximate the underlying structure from data involving no noise, we deliberately chose not to regularize our models.

# 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., [7, 40, 56]. With the aim of approximating the real discriminant locus, i.e., the classification boundaries, the following method uses domain knowledge regarding the classical discriminant locus to provide 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 classical discriminant locus (see Section 2.2). The general framework of the sampling method starts with a randomly selected parameter value $p^* \in \mathbb{R}^k$, e.g., uniformly sampled in a compact subset $\Omega$ of the parameter space $\mathbb{R}^k$. For simplicity, we assume that $\Omega$ 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 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 classical discriminant locus is used to compute the real points in the corresponding pseudowitness point set along $\mathcal{L}^*$ inside of $\Omega$, 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 $\lambda_0$ 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 $\Omega$.

Along $\mathcal{L}^*$, the classical 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 are 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 are 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 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.

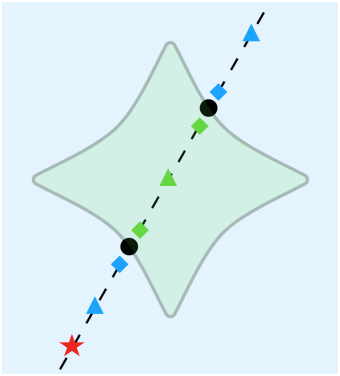


Figure 1: A visual representation of the sampling scheme, where the star represents the uniform random sample point, circles are points on the boundary, midpoints are marked by triangles, and near boundary points as diamonds. The points are color coded based on the number of real solutions.

## 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, 26, 61]. The data sets for training and testing for these examples were generated using the sampling scheme are summarized in Table 1.

| | 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 |

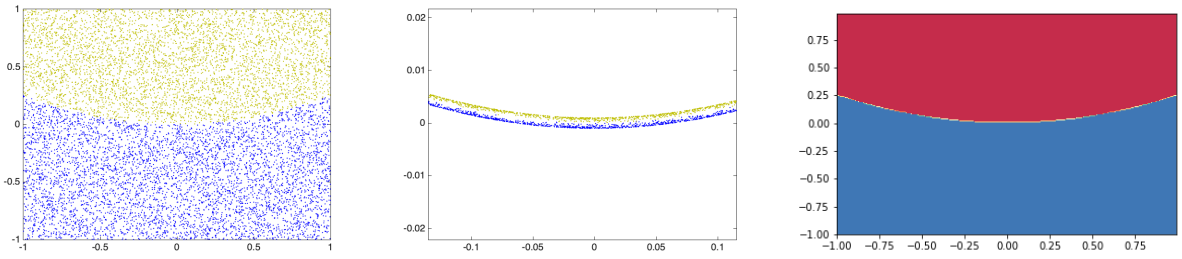
Table 1: Number of data points in each data set used for training/testing for each example.

With these data sets, the computational setup for using the one nearest neighbor (1-NN) classification was based on *KNeighborsClassifier* in `PyTorch` 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` which leveraged CUDA acceleration. Multi-layer, fully connected feedforward networks with ReLU activation functions [32] 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.

### 5.1 Quadratic

As a first example, 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.



(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 $f(x; b, c) = x^2 + bx + c$.

Table 2 summarizes the results of using various training and testing data sets with a one nearest neighbor (1-NN) classification method. This shows that including sample point data near the boundary for training produces classification results which are highly accurate. However, the accuracy of classifying points near the boundary severely declines when training with uniform data.

| | Testing Data | | |
|---|---|---|---|
| | Uniform | NearBoundary | NearBoundary+NearCenter |
| Training Data | Accuracy | Accuracy | Accuracy |
| Uniform | 1 | 0.5392 | 0.7678 |
| NearBoundary | 0.9999 | 1 | 1 |
| NearBoundary+NearCenter | 0.9999 | 1 | 1 |

Table 2: Performance of 1-NN method on the univariate quadratic $f(x; b, c) = x^2 + bx + c$.

A feedforward, fully connected neural network with three hidden layers each with 20 neurons was trained on the data from Figure 2(b). We employed tanh as the activation function for the neurons and used a 2-neuron softmax layer as the output layer – use of a single neuron layer with a sigmoid activation would have been equivalent. The network was trained to separate the data on either side of the discriminant locus into two different classes. A binary cross-entropy loss without regularization was used to train the network and implemented a variable learning rate scheme. The network was trained until it was able to correctly classify every sample in the training set. Once trained, testing data was used where each of the data points was fed to the neural 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.

A summary of the quantitative results with various training and testing data sets are presented in Table 3. This shows that the network was able to learn the real discriminant locus. Moreover, it behaves well even at regions of the parameter space that were not represented in the training

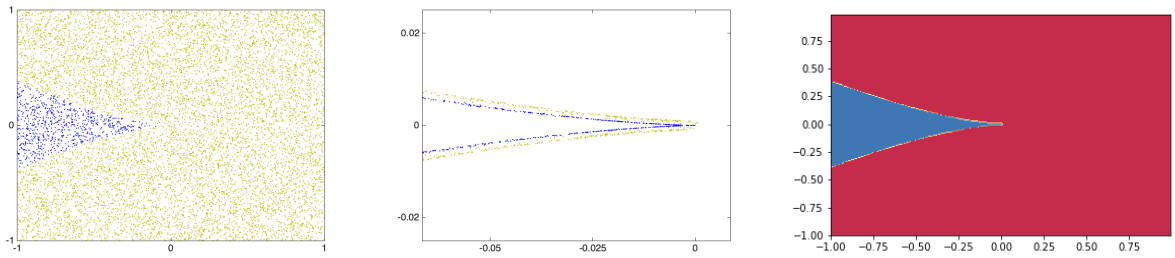
procedure, namely those located away from the boundary. Similar to the 1-NN results, the neural network behaves best when data near the boundary is used within the training data set.

| | Test Data | | |
|---|---|---|---|
| | Uniform | NearBoundary | NearBoundary+NearCenter |
| Training Data | Accuracy | Accuracy | Accuracy |
| Uniform | 1 | 0.9559 | 0.9779 |
| NearBoundary | 1 | 1 | 1 |
| NearBoundary+NearCenter | 1 | 1 | 1 |

Table 3: Performance of feedforward neural network on the univariate quadratic $f(x; b, c) = x^2+bx+c$.

## 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.




(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 $f(x; b, c) = x^3 + bx + c$.

A 1-NN classification method was used with various training and testing data sets, with results summarized in Table 4. As for the quadratic in Section 5.1, including sample point data near the boundary for training yields higher accuracy. In addition, when uniform data is used for training, the accuracy declines when boundary data is included in the testing data set.

A feedforward network with the same hyperparameters as that used in the quadratic from Section 5.1 was trained on the data from Figure 3(b) and illustrated in Figure 3(c). Table 5 presents the quantitative results with the various training and testing data sets. This shows that the network learned a good approximation of the real discriminant locus even in this challenging case where the boundary has a cusp. As before, the network response behaves well across the full parameter space. In addition, when near boundary data is included in the training data set, the model performs better across all testing data sets, similarly to the 1-NN method.

| | Test Data | | |
|---|---|---|---|
| | Uniform | NearBoundary | NearBoundary+NearCenter |
| Training Data | Accuracy | Accuracy | Accuracy |
| Uniform | 1 | 0.5259 | 0.7259 |
| NearBoundary | 0.9999 | 1 | 1 |
| NearBoundary+NearCenter | 0.9999 | 1 | 1 |

Table 4: Performance of 1-NN method on the univariate cubic $f(x; b, c) = x^3 + bx + c$.

| | Test Data | | |
|---|---|---|---|
| | Uniform | NearBoundary | NearBoundary+NearCenter |
| Training Data | Accuracy | Accuracy | Accuracy |
| Uniform | 1 | 0.7848 | 0.8689 |
| NearBoundary | 1 | 1 | 1 |
| NearBoundary+NearCenter | 1 | 1 | 1 |

Table 5: Performance of feedforward neural network on the univariate cubic $f(x; , b, c) = x^3 + bx + c$.

### 5.3 Kuramoto Model

To move beyond toy examples, we consider the Kuramoto model [1, 26, 61] which is a popular model to study synchronization phenomena observed in systems consisting of coupled oscillators. Such a system can be found in a wide variety of applications such as synchronizing fire-flies, rhythmic applause, biological neural networks, laser arrays, power grids, particle coordination, and spin glass models. For $N$ oscillators, the mathematical model is a system of ordinary differential equations:

$$\frac{d\theta_i}{dt} = \omega_i - \frac{1}{N}\sum_{j=1}^{N} \sin(\theta_i - \theta_j), \tag{4}$$

for $i = 1, \ldots, N$ where $\theta_i$ is the phase of the $i^{\text{th}}$ oscillator (at time $t$). The oscillators are coupled through the sine function. The parameters $\omega_i$ are the natural frequencies of the respective oscillators.

We aim to study the number of equilibria of the Kuramoto model as a function of the parameters $\omega_i$. The equilibria satisfy

$$\omega_i - \frac{1}{N}\sum_{j=1}^{N} \sin(\theta_i - \theta_j) = 0. \tag{5}$$

Since these equations are invariant under transformation $\theta_i \to \theta_i + \alpha$ for any $\alpha$, this continuous global symmetry gives rise to solution curves. In order to remove the symmetry and, in turn, restrict the solution space generically to isolated solutions, we fix $\theta_N = 0$. Since the sum of all of the equations in Eq. (5) yields

$$\omega_1 + \cdots + \omega_N = 0,$$

we can remove the $N^{\text{th}}$ equation from the system which is now parameterized by $\omega_1, \ldots, \omega_{N-1}$ with $\omega_N = -(\omega_1 + \cdots + \omega_{N-1})$. Finally, we can rewrite using trigonometric identities and replace each $\cos\theta_i$ and $\sin\theta_i$ by variables $c_i$ and $s_i$ which are constrained by the Pythagorean theorem. Hence, the final parameterized polynomial system under consideration is

$$F(c_1, s_1, \ldots, c_{N-1}, s_{N-1}; \omega_1, \ldots, \omega_{N-1}) = \begin{bmatrix} \omega_i - \frac{1}{N}\sum_{j=1}^{N}(s_i c_j - s_j c_i) \\ c_i^2 + s_i^2 - 1 \\ i = 1, \ldots, N-1 \end{bmatrix} = 0. \tag{6}$$

From Eq. (5), it is easy to observe that if $\omega_i \notin \left[-\frac{N-1}{N}, \frac{N-1}{N}\right]$, then Eq. (6) can have no real solutions. Hence, the parameter space is naturally restricted to a compact subset of $\mathbb{R}^{N-1}$. Moreover, 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, $F$ from Eq. (6) has $2^N - 2$ solutions [16, Thm. 4.3]. Thus, for $N = 3$, there can be a maximum of 6 isolated real solutions and there are parameters for any possible even number of solutions, e.g., see Figure 4(d). For $N = 4$, it was conjectured in [66] 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 [35, Thm. 8.1].

The following considers learning the real discriminant locus of the parameter space for $N = 3$ and $N = 4$. It is known that the classical discriminant locus has degree 12 and 48, respectively, and has many singularities.

#### 5.3.1 3-Oscillators

For $N = 3$, there are two parameters $(\omega_1, \omega_2) \in [-1, 1]^2$. A uniform sampling 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).

A 1-NN classification method was used with various training and testing data sets to learn the real discriminant locus with the results summarized in Table 6. As in the previous examples, including sample point data near the boundary for training yields higher accuracy when any testing data set is used.

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 do 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 6, the accuracy when the size of data sets is comparable does not drastically increase. 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 feedforward, fully connected neural network with five hidden layers each with 20 neurons was trained on the data from Figure 4(b) and shown in Figure 4(d). We employed ReLU activation function for the neurons 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. The network was trained to classify the parameter points according to the number of real solutions associated with the portion of the space they reside in. A multi-class cross-entropy loss without regularization was used to train the

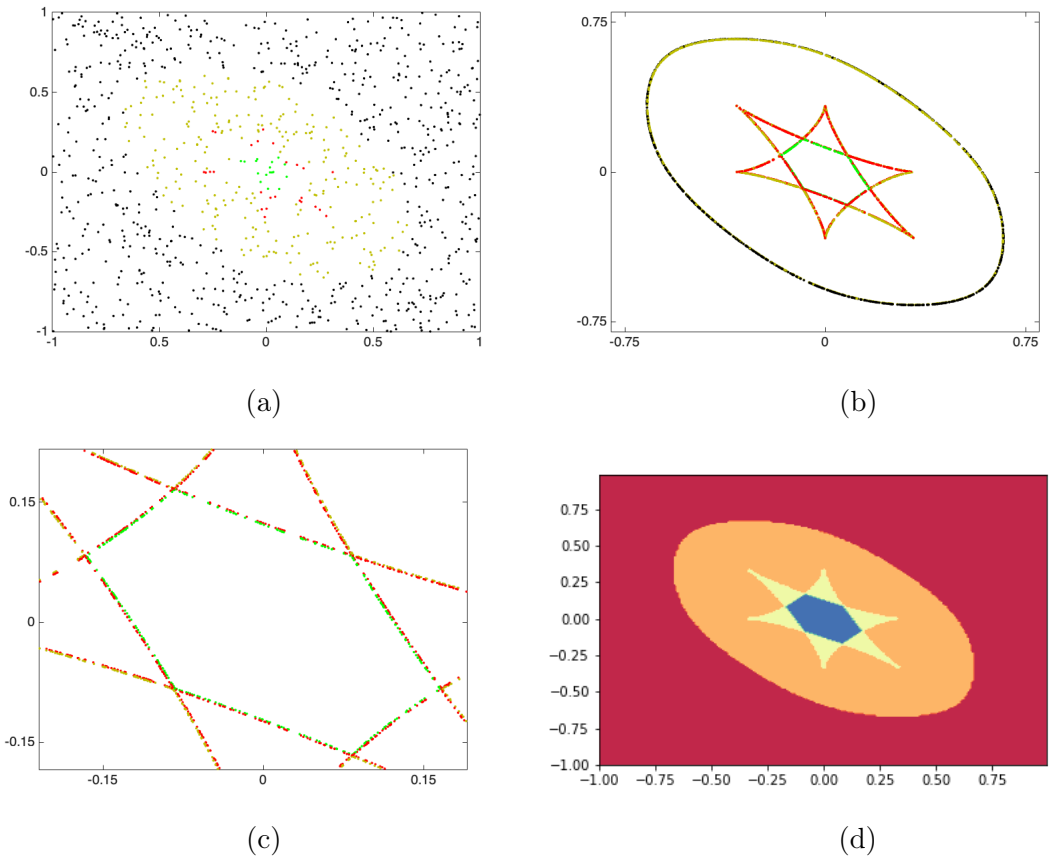



Figure 4: For the 3-Oscillator Kuramoto model, (a) uniformly selected parameter values, (b) data perturbed from the boundary, (c) a zoomed view of data perturbed from the boundary, and (d) decision boundary from neural network trained on data from (b).

| | Test Data | | |
|---|---|---|---|
| | Uniform | NearBoundary | NearBoundary+NearCenter |
| Training Data | Accuracy | Accuracy | Accuracy |
| Uniform | 1 | 0.4921 | 0.6525 |
| Uniform (large) | 1 | 0.5306 | 0.6973 |
| NearBoundary | 1 | 1 | 0.9985 |
| NearBoundary+NearCenter | 1 | 1 | 1 |

Table 6: Performance of 1-NN method on the 3-Oscillator Kuramoto model.

network with a variable learning rate scheme. The network was trained until it correctly classified every sample in the training set. For various testing data sets, each sample point was fed to the neural network and the classification decision recorded. Table 7 shows that including data points near the boundary in the training data set decidedly improves the accuracy across all testing data sets.

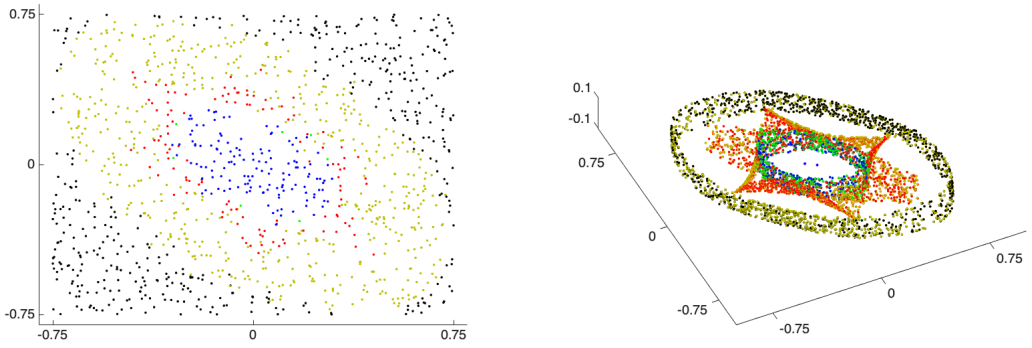
| | Test Data | | |
|---|---|---|---|
| | Uniform | NearBoundary | NearBoundary+NearCenter |
| Training Data | Accuracy | Accuracy | Accuracy |
| Uniform | 1 | 0.5027 | 0.6554 |
| NearBoundary | 1 | 1 | 0.9861 |
| NearBoundary+NearCenter | 1 | 1 | 1 |

Table 7: Performance of feedforward network on the 3-Oscillator Kuramoto model.

### 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 8 summarizes the results when using a 1-NN classification method which are consistent with previous experiments.

In our experiment using the neural network method, it is apparent that the 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 has been shown empirically [53] that the number of local minima in the optimization landscape of a neural network decreases as stronger regularization is enforced.




(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.

| Training Data | Test Data | | |
|---|---|---|---|
| | Uniform | NearBoundary | NearBoundary+NearCenter |
| | Accuracy | Accuracy | Accuracy |
| Uniform | 1 | 0.463 | 0.5911 |
| NearBoundary | 0.9782 | 1 | 0.9901 |
| NearBoundary+NearCenter | 0.9869 | 1 | 1 |

Table 8: Performance of 1-NN method on the 4-Oscillator Kuramoto model.

# 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 $f(x; p^*) = 0$, 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

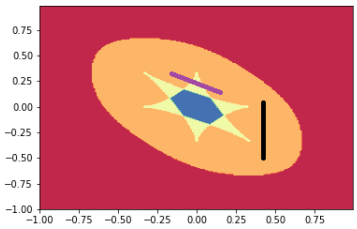



Figure 6: Illustration of two segments added to Figure 4(d), one (black) which is guaranteed to succeed while the other (purple) may fail since it intersects the real discriminant locus.

using a classical parameter homotopy with this machine learning assisted real parameter homotopy. Table 9 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.

| 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 9: The average computation time for finding all real roots for the 3-Osillator Kuramoto model using a machine learning assisted real parameter homotopy method.

### 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 complex solution paths using a classical parameter homotopy was 3.40 seconds. Table 10 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 8.

## 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

| 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 10: The average computation time for finding all real roots for the 4-Osillator Kuramoto model using a machine learning assisted real parameter homotopy method.

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. This is important since singularities often arise on real discriminant loci 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 are useful when the parameter space is not sampled very densely.

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 $\epsilon$ 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.

Finally, as an application of learning the real discriminant locus, we developed a real parameter homotopy method that tracks on real solution paths in Section 6. 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 easily extensible to other nonlinear functions such as rational, exponential, logarithmic, trigonometric, and piecewise functions.

## Acknowledgement

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.

## 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.

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

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

[8] 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.

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

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

[11] 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.

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

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

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

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

[16] 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.

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

[18] D. A. Cox. *Applications of Polynomial Systems*. CBMS Regional Conference Series in Mathematics. Conference Board of the Mathematical Sciences, 2020.

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

[20] 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.

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

[22] 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.

[23] 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.

[24] 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.

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

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

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

[28] 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.

[29] 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.

[30] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[31] 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.

[32] 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.

[33] W. Hanan, D. Mehta, G. Moroz, and S. Pouryahya. Stability and bifurcation analysis of coupled fitzhugh-nagumo oscillators. *arXiv preprint arXiv:1001.5420*, 2010.

[34] 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.

[35] K. Harris, J. D. Hauenstein, and A. Szanto. Smooth points on semi-algebraic sets. arXiv: 2002.04707, 2020.

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

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

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

[39] 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.

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

[41] 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.

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

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

[44] 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.

[45] 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.

[46] 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.

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

[48] Z. Huang, M. England, D. Wilson, J. H. Davenport, and L. C. Paulson. Using machine learning to improve cylindrical algebraic decomposition. *arXiv preprint arXiv:1804.10520*, 2018.

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

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

[51] 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.

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

[53] D. Mehta, X. Zhao, E. A. Bernal, and D. J. Wales. The loss surface of xor artificial neural networks. *Preprint*, 2018.

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

[55] 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.

[56] 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.

[57] 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.

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

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

[60] 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.

[61] 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.

[62] 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.

[63] 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.

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

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

[66] 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.