Machine Learning & Algebraic Geometry

Vahid Shahverdi

March 2023

Disclaimer: This lecture note is not intended to be a comprehensive overview of algebraic geometry and machine learning. Rather, it represents my perspective on these areas based on my reading of various articles and papers. As such, it may not cover all aspects of these fields and some details may be oversimplified or omitted. I encourage the reader to explore these topics further and consult additional resources for a more complete understanding.

1 Introduction to Machine Learning

Have you ever wondered how your phone can recognize your face or how virtual assistants like Siri or Alexa can understand what you're saying? Machine learning is the technology behind these amazing feats. It's a subset of artificial intelligence that involves creating algorithms capable of making predictions or decisions based on data. By analyzing vast amounts of data, machine learning algorithms can identify patterns and relationships to produce accurate predictions or decisions.

There are three main types of machine learning techniques: supervised, unsupervised, and reinforcement learning. Supervised learning involves training an algorithm on labeled data to make predictions on new data. This technique is commonly used in applications such as spam detection or fraud detection. Unsupervised learning, on the other hand, involves identifying patterns and relationships within an unlabeled dataset. This technique is useful for clustering similar data points or identifying anomalies. Reinforcement learning is a type of machine learning where an agent learns to interact with an environment to maximize a reward signal. This technique is commonly used in game playing or robotics.

In this note, we'll be focusing on supervised learning and its potential relation to algebraic geometry.

1.1 Machine Learning as an Optimization Problem

Many machine learning tasks can be formulated as optimization problems. Here is a general framework:

Given a set of input-output pairs, $\{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$, where x_i is the input and y_i is the corresponding output, the goal is to find a function f_θ

that maps input x to an output y, where θ is a set of parameters belongs to a set $\mathcal{P}$ that the function f_θ depends on. The set of all such functions f_θ considered in this setting is called function space $\mathcal{M}$. The function f_θ is typically chosen to minimize a loss function $l(y, f_\theta(x))$, which measures the difference between the predicted output $f_\theta(x)$ and the true output y.

Thus, the machine learning task can be expressed as the following optimization problems:

1. Over function space: $\min_{f_\theta \in \mathcal{M}} l(y, f_\theta(x))$.
2. Over parameter space: $\min_{\theta \in \mathcal{P}} l(y, f_\theta(x))$.

In some cases, regularization terms may be added to the loss function to avoid overfitting and improve the generalization ability of the model. The optimization problem can then be modified to:

$$\min_{\theta \in \mathcal{P}} l(y, f_\theta(x)) + \lambda R(\theta),$$

where $R(\theta)$ is a regularization term and λ is a hyperparameter that controls the strength of regularization.

Example 1.1. Let us consider a parameter space $\mathcal{P}$ that is defined as $\mathbb{R}^2$. Further, suppose we have a function space $\mathcal{M}$ that encompasses linear functions $f_\theta : \mathbb{R}^2 \to \mathbb{R}$ that map each data point $z \in \mathbb{R}^2$ to $\theta^T \cdot z$. Our goal is to find f_θ (or θ) such that it minimizes the expression $\sum_{i=1}^n |f_\theta(z_i) - y_i|^2 = \sum_{i=1}^n |\theta^T \cdot z_i - y_i|^2 = ||\theta^T Z - Y||^2$, where n is the number of data points, z_i and y_i are the i-th input and output data points, respectively, and $||\cdot||^2$ denotes the Euclidean norm squared. We can derive a unique solution for θ^T when ZZ^T is invertible, and this solution is given by $YZ^T(ZZ^T)^{-1}$. Notably, this solution provides the minimizer of the above expression for θ^T. To illustrate the concept of binary classification via a linear function, we present an example in Figure 1.

1.2 From Linear to Non-linear via Kernel Method

The kernel method is a popular machine learning technique used for nonlinear classification and regression problems. In many real-world applications, the relationship between the input variables and the target variable is not linear. In such cases, traditional linear models like linear regression or logistic regression may not perform well.

The kernel method allows us to transform the input variables into a higher-dimensional feature space, where the relationship between the input variables and the target variable may be linear. This transformation is done using a kernel function, which is a mathematical function that measures the similarity between pairs of input data points.

By mapping the input data to a higher-dimensional feature space, the kernel method can effectively capture complex nonlinear relationships between the input variables and the target variable. Additionally, the kernel method can be

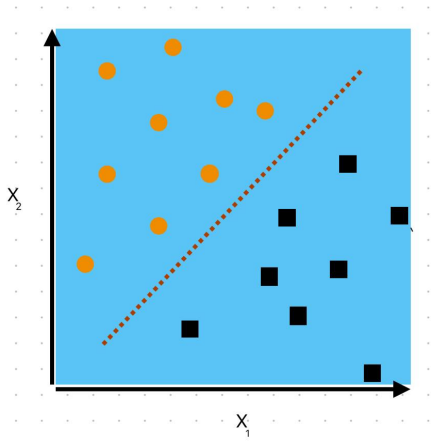

Figure 1: Linearly separable data presented in a plane

used with a variety of machine learning algorithms, including support vector machines (SVMs), ridge regression, and principal component analysis (PCA); see [SV08].

In the mathematical setting, the kernel method involves mapping the input data points, represented as vectors x in a d-dimensional input space, to a higher-dimensional feature space $\mathcal{F}$ using a kernel function K.

The kernel function takes a pair of input vectors x and x', and computes the dot product of their corresponding feature representations in $\mathcal{F}$. Mathematically, the kernel function can be defined as:

$$K(x, x') = \langle \phi(x), \phi(x') \rangle$$

where $\phi(x)$ and $\phi(x')$ are the feature representations of x and x' in $\mathcal{F}$, respectively.

The choice of kernel function depends on the specific problem and the characteristics of the input data. Some common kernel functions include:

- Linear kernel: $K(x, x') = x^T x'$.
- Polynomial kernel: $K(x, x') = (x^T x' + c)^d$, where c is a constant and d is the degree of the polynomial.
- Gaussian (RBF) kernel: $K(x, x') = exp(-||x - x'||^2/(2\sigma^2))$, where σ is a parameter that controls the width of the Gaussian function.

Once the input data has been mapped to the feature space $\mathcal{F}$, a linear algorithm such as SVMs or ridge regression can be used to perform classification or regression. The decision boundary in $\mathcal{F}$ is a hyperplane, which corresponds to a nonlinear decision boundary in the original input space.

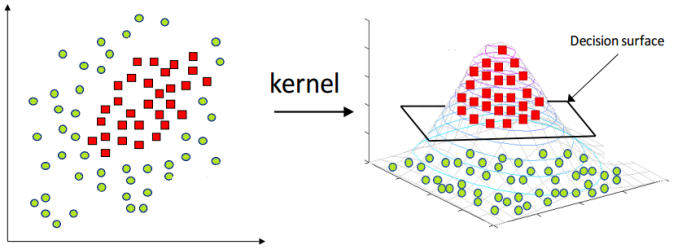

Figure 2: Binary classification of data points with non-linear boundary through kernel from [Gra23].

In practice, the feature representations $\phi(x)$ in $\mathcal{F}$ are often not explicitly computed, as the kernel function allows us to compute the dot product without ever explicitly representing the feature vectors. This is known as the "kernel trick" and allows the kernel method to be computationally efficient even in high-dimensional feature spaces.

Example 1.2. Given the labeled data displayed in Figure 2, employing the feature mapping $\phi : \mathbb{R}^2 \to \mathbb{R}^3$ that maps $z = (z_1, z_2)$ to $(z_1^2, z_1 z_2, z_2^2)$ enables us to achieve linear separability of the data.

While the kernel method has the capability to capture non-linearity in data, the selection of a suitable feature function can be a complex and daunting task, and in some cases, it may be impractical. To address this issue, a clever solution has been devised known as a neural network, which is specifically designed to automatically identify and capture non-linearity in data.

2 Neural Network

As a mathematician, you can think of a neural network as a function that takes in some input data, processes it through a series of interconnected layers, and produces an output. Each layer in the network is composed of nodes (also called neurons), which perform mathematical operations on the input data.

These mathematical operations typically involve multiplying the input data by weights (which the network learns through a process called training), adding a bias term, and passing the result through a non-linear activation function; see Figure 3. The activation function helps to introduce non-linearity into the network, which allows it to model complex relationships between inputs and outputs.

2.1 Feedforward Neural Network

Feedforward neural networks are a commonly used type of artificial neural network where information flows in only one direction, from input to output layers, without feedback loops, and are used in various applications such as image

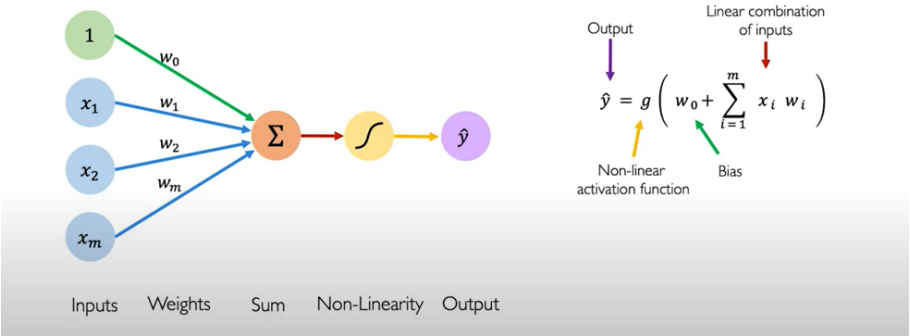

Figure 3: A single hidden layer neural network comprising of a single neuron from [Aji23].

recognition, speech recognition, natural language processing, and regression and classification problems; see [Saz06].

A feedforward neural network is a family of functions $\mathcal{M}$ that every function can be written as

$$f : \mathbb{R}^{d_0} \to \mathbb{R}^{d_L}; \quad x \to (\alpha_L \circ \rho \circ \alpha_{L-1} \circ \rho \cdots \circ \rho \circ \alpha_1)(x), \tag{1}$$

where α_i are affine maps and ρ is an activation function.

Goal: The goal of this endeavor is to acquire a comprehensive understanding of the geometric characteristics of the function space $\mathcal{M}$, with the aim of effectively minimizing the discrepancy between the true output vector Y and the predicted output vector $f(X)$ through the solution of an optimization problem $\min_{f\in\mathcal{M}} ||f(X) - Y||$.

2.2 Universal Approximation Theorem

Although the structure of neural networks is inspired by the human brain, the reason "why" this structure works is governed by the following theorem:

Theorem 2.1. *The Universal Approximation Theorem: Let $F(x)$ be a continuous function defined on a compact set K in n-dimensional Euclidean space. Then, for any $\epsilon > 0$ and $\delta > 0$, there exists a single hidden layer neural network with finite number of hidden units, and a non-constant, bounded, and monotonically-increasing activation function $\rho(x)$, such that:*

$$|f(x) - F(x)| < \epsilon \quad \textit{for all } x \textit{ in } K$$

where $f(x)$ is the output of the neural network with weights and biases chosen appropriately, and $||.||$ denotes the Euclidean norm.

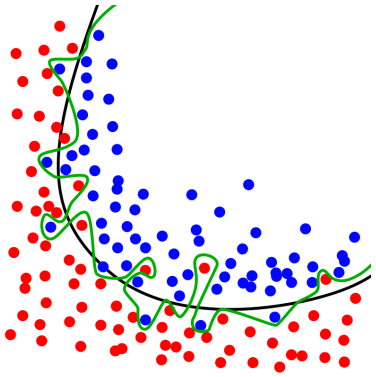

Figure 4: The model depicted by the green line is overfitted, whereas the model represented by the black line is regularized. Although the green line closely adheres to the training data, it is excessively reliant on that data, rendering it susceptible to higher error rates when evaluated on fresh, unseen data compared to the black line; image from [Ch23].

Several variations of this theorem have been developed to deal with different types of networks. For instance, in [LLPS93], it is demonstrated that this structure also applies to networks with multiple hidden layers and non-polynomial activation functions.

One additional contention posits that the ultimate objective of a neural network bears strong resemblance to interpolation, as referred in Theorem 2.1. In light of this, a pertinent inquiry would be the outcome of utilizing polynomial interpolation. To satisfactorily address this query, a comprehensive comprehension of overfitting is imperative.

Overfitting: Overfitting can cause a problem in machine learning where a model is trained too well on a particular dataset. Specifically, overfitting occurs when a model is too complex and begins to fit not only the underlying patterns in the data, but also the noise or random fluctuations in the data. As a result, an overfit model may perform well on the training data but perform poorly on new, unseen data; see Figure 4.

2.3 Geometric questions in the theory of machine learning

Many goals in the theory of machine learning with neural networks are of geometric nature, such as

1. *Expressivity* [GRK20] refers to understanding the set of functions that a given neural network can learn or appoximate, i.e., understanding the function space $\mathcal{M}$ and its geometric properties.

2. The study of the *loss landscape* [LXT+18] of a given neural network and

a given loss function refers to analyzing the critical points in function space and in parameter space, e.g., distinguishing local / global minima from saddle points, or investigating the sharpness / flatness around critical points. The loss landscape encodes the static properties of the optimization problem that do not depend on a choice of optimization algorithm such a gradient descent.

3. The *dynamic optimization properties* of a given network, loss, and optimization algorithm entail for instance the convergence behavior to critical points or understanding the curves traced by the optimization algorithm [NRT21].

In the upcoming section, we will exhibit how these goals can be approached with techniques from algebraic or tropical geometry.

3 Algebraic Geometry for Machine Learning

In the realm of deep learning, the choice of activation function is a critical component of neural network design. The choice of activation function has implications for the underlying mathematical structure of the optimization problem. For instance, if the activation function is polynomial, the neural network function in (1) is polynomial and the set $\mathcal{M}$ of all such functions is semi-algebraic. Hence, we can employ tools from (real) algebraic geometry and commutative algebra to study the questions in Section 2.3. If the activation function is rectified linear units (ReLU), the function space $\mathcal{M}$ consists of tropical rational maps [ZNL18], and tropical geometry can provide insights for the goals in Section 2.3.

3.1 Linear Neural Networks

The simplest choice of activation function is the identity. Such networks are called *linear neural networks*. There is a large body of theoretical work studying their behaviour, e.g., [NRT21, ACGH18, KMMT22, TFHV21, VZH22, LB18, BH89, Kaw16, CLC22, BRTW22] to name a few. Two of the earliest studies of the expressivity and loss landscape of linear neural networks from the algebro-geometric perspective are [TKB19] and [MCTH21].

Definition 3.1. A linear network is a mapping $\Phi : \mathbb{R}^{d_\theta} \times \mathbb{R}^{d_x} \to \mathbb{R}^{d_y}$ that takes the form

$$\Phi(\theta, x) = W_L \cdots W_1 x, \quad \theta = (W_L, \ldots, W_1) \in \mathbb{R}^\theta, \tag{2}$$

where $W_i \in \mathbb{R}^{d_i} \times \mathbb{R}^{d_{i-1}}$, and thus $d_\theta = d_0 d_1 + \cdots + d_{L-1} d_L$.

Definition 3.2. Let (d_L, d_0) be a fixed size, and consider the vector space of matrices with this size, denoted by $M_{d_L \times d_0}$. We define the determinantal variety $\mathcal{M}_r$ to be a subset of $M_{d_L \times d_0}$ that contains matrices of rank at most r, where $r \leq \min(d_0, d_L)$.

In the linear setting, the function space $\mathcal{M}_\Phi$ consists of matrices $\overline{W}$ of size (d_L, d_0) that can be factorized as $W_L \cdots W_1$ according to Definition 3.1. It can be easily verified that $\mathcal{M}_\Phi = \mathcal{M}_r$ for $r = \min(d_0, \ldots, d_L)$.

Remark 3.3. The Eckart-Young Theorem tells us that the ED degree of the determinantal variety $\mathcal{M}_r$ is $\binom{m}{r}$, where $m = \min(d_0, d_L)$. Furthermore, it can be shown that the singular locus of $\mathcal{M}_r$ is precisely $\mathcal{M}_{r-1} \subset \mathcal{M}_r$.

The square loss: Given data $X \in \mathbb{R}^{d_0 \times N}$ and $Y \in \mathbb{R}^{d_L \times N}$ (where N is the number of data samples), and assuming that XX^T has full rank, we write the quadratic loss as:

$$\begin{aligned} l(w) = l_{X,Y}(W) \quad &= ||WX - Y||^2 = \langle Y, Y\rangle - 2\langle WX, Y\rangle + \langle WX, WX\rangle \\ &= const. - 2\langle W(XX^T), YX^T(XX^T)^{-1} + \langle WX, WX\rangle \\ &= const. - 2\langle W, U\rangle_{XX^T} + \langle W, W\rangle_{XX^T} \\ &= const. + ||W - U||^2_{XX^T}, \end{aligned}$$

where U is the unconstrained optimizer of this quadratic problem, over the set of all matrices:

$$U = \text{argmin}_{V \in \mathbb{R}^{d_L \times d_0}} ||VX - Y||^2 = YX^T(XX^T)^{-1}.$$

Therefore, the optimization over the function space can be obtained by

$$\min_{\overline{W} \in \mathcal{M}_r} ||\overline{W} - U||^2_{XX^T}.$$

Remark 3.4. Despite the potential challenges posed by the utilization of the weighted Frobenius norm $||.||_{XX^T}$, it is important to bear in mind that if our samples adhere to the independent and identically distributed (i.i.d.) Gaussian random variable criteria, then $\mathbb{E}[XX^T]$ would represent a scalar multiple of the identity matrix.

Loss landscape via the parameterization map: Based on Definition 3.1, the loss function L on the parameter space factors through the *parameterization map* μ as follows:

$$L : \mathbb{R}^{d_\theta} \xrightarrow{\mu} \mathcal{M}_\Phi \overset{l|_{\mathcal{M}_\Phi}}{\longrightarrow} \mathbb{R}$$

$$(W_L, \ldots, W_1) \xrightarrow{\mu} \overline{W} := W_L \cdots W_1 \overset{l|_{\mathcal{M}_\Phi}}{\longrightarrow} ||\overline{W}X - Y||^2.$$

A machine learning algorithm like gradient descent aims to find critical points of L in parameter space. However, the meaningful critical points (for instance, the best function explaining the data) live in the function space $\mathcal{M}$ and are critical points of $l|_{\mathcal{M}}$. Hence, [TKB19] distinguishes between *pure* critical points of L (those that actually come from critical points of $l|_{\mathcal{M}}$) and *spurious* critical points (that are only caused by the parametrization map μ); see Figure 5.

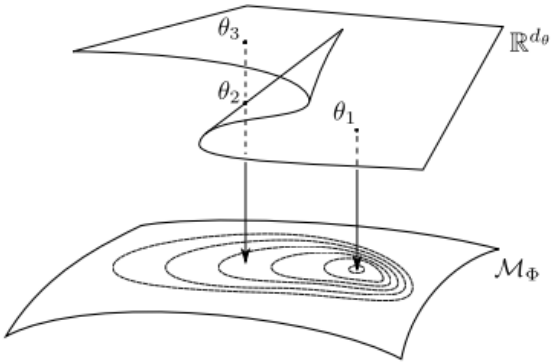

Figure 5: Pure and spurious critical points: θ_1 is a pure critical point, while θ_2 is a spurious critical point (the level curves on the manifold $\mathcal{M}_\Phi$ describe the landscape in functional space). Note that θ_3 is mapped to the same function as θ_2, but it is not a critical point; image from [TKB19].

Proposition 3.5. *[TKB19, Proposition 6] Let $\theta \in Crit(L)$ be such that $d\mu(\theta)$ has maximal rank. Then θ is a pure critical point, i.e., $\mu(\theta) \in Crit(l|_{\mathcal{M}_r})$. Moreover, θ is a minimum (resp., saddle, maximum) for L if and only if $\mu(\theta)$ is a minimum (resp., saddle, maximum) for $l|_{\mathcal{M}_r}$.*

Proposition 3.6. *[TKB19, Proposition 7] If $\theta \in Crit(L)$ with $\operatorname{rank}(\overline{W}) = e \leq r$, then $\overline{W} \in \operatorname{Crit}(l|_{\mathcal{M}_e})$. In other words, if $\operatorname{rank}(\overline{W}) < r$, then $\theta \in Crit(L)$ implies that $\overline{W}$ is critical point for the restriction of l to a smaller determinantal variety $\mathcal{M}_e$ (which is in the singular locus of the functional space $\mathcal{M}_r$).*

Proposition 3.7. *[TKB19] Let l be smooth and convex. Then L has a non-global local minimum if and only if $l|_{\mathcal{M}}$ has a non-global local minimum.*

The latter result immediately implies classical results on the loss landscape of linear networks, such as that all local minima of L are global if either l is the quadratic loss [BH89, Kaw16], or if l is an arbitrary smooth convex functional and $r = \min(d_0, d_L)$ [LB18]. Note that the latter condition $r = \min(d_0, d_L)$ is equivalent to that the function space $\mathcal{M}$ is equal to its ambient vector space $M_{d_L \times d_0}$.

Under suitable conditions, gradient descent converges to "nice" critical points of L [NRT21]. An important ingredient in that analysis are the *algebraic invariants of gradient flow*. Although the curves traced out by gradient flow are generically not algebraic, the following algebraic map is constant under gradient flow:

$$(W_L, \ldots, W_1) \mapsto (W_L^T W_L - W_{L-1} W_{L-1}^T, \ldots, W_2^T W_2 - W_1 W_1^T);$$

see [CLC22, Lemma 2.3] or [ACGH18, BRTW22].

The discussion so far was restricted to *fully-connected / dense* linear networks where every neuron in a layer is connected to all neurons in the next layer and all weights are independent of each other. There are other types of architectures that play an important role in practice, e.g., convolutional networks. A first geometric study on the expressivity, loss landscape, and algebraic invariants of gradient flow in linear convolutional networks is [KMMT22].

3.2 Neural Networks with Polynomial Activation functions

According to the Stone-Weierstrass theorem, any function can be approximated by a polynomial over a compact set. A Polynomial Neural Network (PNN) applies polynomial approximations of activation functions in a feedforward neural network. One immediate implication is that the function space $\mathcal{M}$ becomes a semi-algebraic set, which can be studied using tools from algebraic geometry. A first study of polynomial networks from the algebro-geometric perspective is [KTB19]. In particular, they estimate the dimension of the function space as a measure of the expressivity of polynomial networks and provide conditions for when the function space is a linear vector space.

3.3 Neural Networks with ReLU Activation functions

The ReLU activation function is piecewise linear and thus the functions in the function space $\mathcal{M}$ of a ReLU neural network are piecewise linear as well. Hence, the expressivity of a ReLU network is largely governed by the number (and shape) of the linear regions of the functions in $\mathcal{M}$. It was shown in [ZNL18] that ReLU neural networks parametrize tropical rational maps. That tropical perspective has enabled the establishment of sharp bounds on the number of linear regions of the functions parametrized by ReLU networks [MRZ22]. The topological and geometric shape of the linear decision regions was studied in terms of bent hyperplane arrangements in [GL22]. A geometric study of the dynamic optimization properties of ReLU networks, including algebraic invariants of gradient flow, can be found in [WTP$^+$19].

4 Machine Learning for Algebraic Geometry

Thus far, we have observed that algebraic geometry can provide insights into the workings of machine learning. Furthermore, in this section, we will explore how machine learning can be a potent tool in algebraic geometry.

There is a growing body of research exploring the use of machine learning techniques in algebraic geometry and related areas. For instance, in [HKS22], they use a practical approach and train deep neural networks to predict the complexity of Gauss-Manin connections for a pencil of hypersurfaces. In [BHM$^+$20],

a machine learning-based method is proposed for approximating the real discriminant locus of parameterized polynomial systems, with applications in equilibria of dynamical systems and scene reconstruction. Similarly, [BLOS22] and [BHH$^+$22] demonstrate the effectiveness of machine learning techniques in predicting the number of real circles tangent to three conics respectively the geometric properties of Hilbert series. In [DLQ22], new machine learning methods are presented for efficiently computing numerical Calabi-Yau metrics. Lastly, [CHLM21] shows that machine learning can significantly speed up the computation of tensor products and branching rules of irreducible representations of Lie algebras, which are important for analyzing symmetry in physical systems.

4.1 A Research Problem

To this end, I would like to present a potential research question dealing with possibility of learning topology of hypersurfaces with machine learning. The rest of this section briefly outlines the motivation and context of this problem.

Hilbert's Sixteenth Problem (H16): This problem, presented in a contemporary formulation, entails studying the number, shape, and position of the components comprising a smooth algebraic hypersurface of degree d in n-dimensional real projective space. An illustration of this problem can be found in the case where n=2, where Harnack's inequality sets the upper bound at $\frac{(d-1)(d-2)}{2} + 1$ for the number of connected components in a smooth real algebraic curve of degree d in the plane.

Space of polynomials and its discriminant: We define $\mathscr{P}_{n,d}$ as the collection of real homogeneous polynomials of degree d, namely $\mathbb{R}[x_0, \ldots, x_n]_{(d)}$. The real discriminant $\mathscr{S}_{n,d}$ refers to the subset of $\mathscr{P}_{n,d}$ consisting of polynomials whose zero sets possess real singular points.

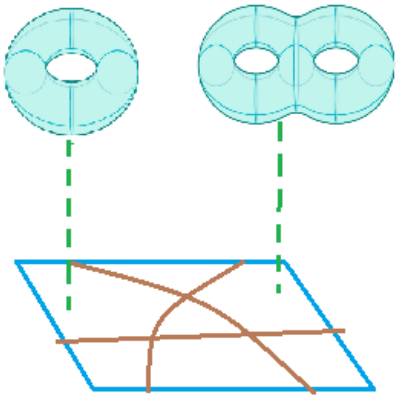

Figure 6: The zero sets of two polynomials from different chambers may exhibit distinct topological properties, this is not the case for polynomials selected from within the same chamber.

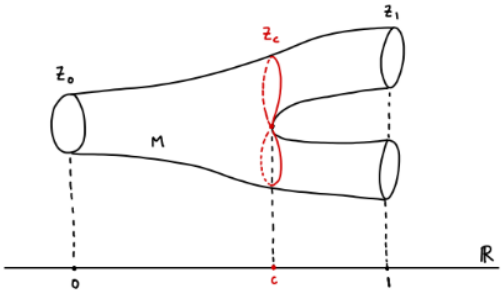

Figure 7: A cobordism M between Z_0 and Z_1. The red curve occurs while passing through the discriminant; image from [Ant23].

It can be inferred that $\mathscr{P}_{n,d} \setminus \mathscr{S}_{n,d}$ comprises a finite number of connected regions, known as *chambers*, represented as C_τ. Additionally, Thom's Isotopy Lemma guarantees that the zero sets of any two polynomials p_1, p_2 in $\mathscr{P}_{n,d}$ within the same chamber exhibit identical topological characteristics; see Figure 6 and 7.

Question: Is it possible to employ machine learning techniques to learn the topology of hypersurfaces?

After conducting a series of experiments, we have arrived at the preliminary conclusion that the answer to the above question is likely affirmative; also see [MPTV06]. Our approach involves generating a large number of random samples

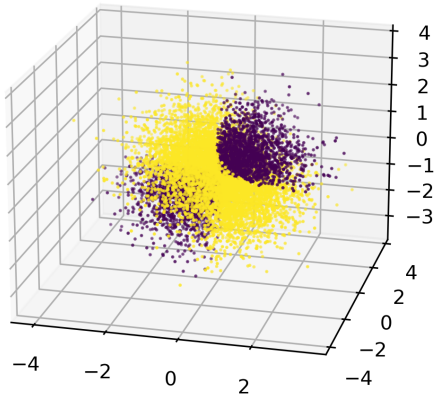

Figure 8: Labeling of random samples of degree 2 polynomials based on the number of real roots.

in $\mathscr{P}_{n,d}$ and utilizing tools in topological data analysis to compute/approximate the betti numbers. Subsequently, we constructed a neural network to discern the boundary decision between various chambers for some fixed d and $n \leq 2$. Here are some of our specific findings for the scenario in which n is equal to 1: Our procedure involved the utilization of polynomials $\sum_{i=0}^{d} \xi_i x^i$, whereby ξ_i denotes independent Gaussian random variables having a zero mean and unit variance. These polynomials were assigned labels based on the quantity of their actual roots; see Figure 8. Subsequently, we proceeded to train a neural network utilizing a dataset consisting of 20000 to 30000 samples of polynomials with degrees 2, 3, and 6. The outcomes of these experiments have been depicted in the form of confusion matrices, as illustrated in Figure 9, 10, and 11.

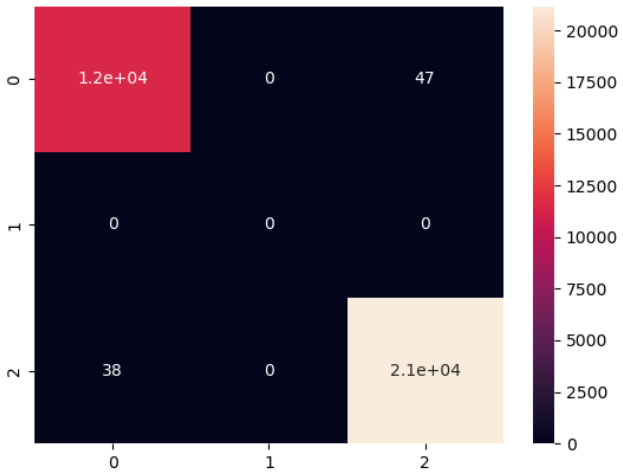

Figure 9: Confusion matrix for predicting the number of real roots for degree 2 polynomials

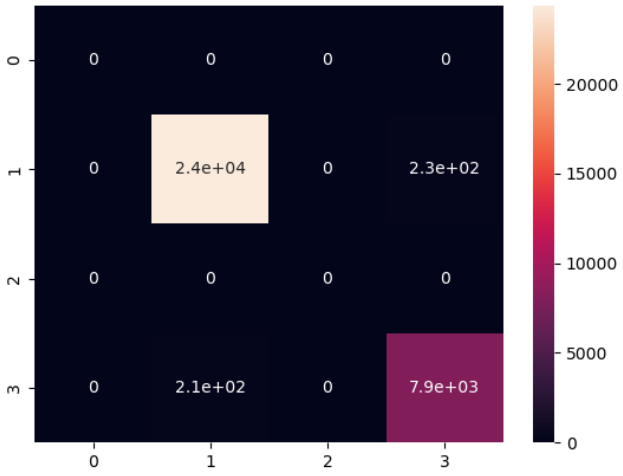

Figure 10: Confusion matrix for predicting the number of real roots for degree 3 polynomials

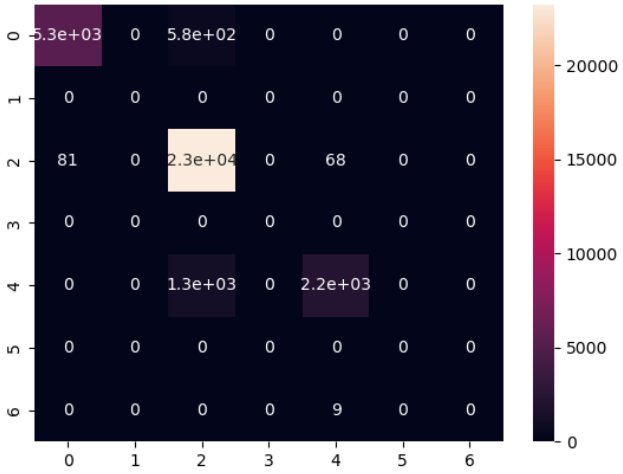

Figure 11: Confusion matrix for predicting the number of real roots for degree 2 polynomials

Acknowledgement

I would like to express my gratitude to Professor Kathlén Kohn for her valuable insights and ideas, as well as for her help in writing some parts of this lecture note. Her guidance and support have been instrumental in shaping my understanding of algebraic geometry and machine learning.

I would also like to thank my friend Björn Wehlin for his contributions in producing the results for my research question, using his expertise in Python programming. His assistance has been crucial in making this lecture note possible.

References

[Ch23] Chabacano. overfitting. `https://en.wikipedia.org/wiki/Overfitting`, 2023. [Online; accessed March 18, 2023].

[ACGH18] Sanjeev Arora, Nadav Cohen, Noah Golowich, and Wei Hu. A convergence analysis of gradient descent for deep linear neural networks. *arXiv preprint arXiv:1810.02281*, 2018.

[Aji23] Ajitesh Kumar. computation via activation. `https://vitalflux.com/perceptron-explained-using-python-example/`, 2023. [Online; accessed March 18, 2023].

[Ant23] Antonio Lerario. Cobordism. https://drive.google.com/file/d/1cwOm6S3M4FxtvUzVA9_-kFhBj2dx0WkX/view, 2023. [Online; accessed March 18, 2023].

[BH89] Pierre Baldi and Kurt Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, 2(1):53–58, 1989.

[BHH+22] Jiakang Bao, Yang-Hui He, Edward Hirst, Johannes Hofscheier, Alexander Kasprzyk, and Suvajit Majumder. Hilbert series, machine learning, and applications to physics. *Physics Letters B*, 827:136966, 2022.

[BHM+20] Edgar A Bernal, Jonathan D Hauenstein, Dhagash Mehta, Margaret H Regan, and Tingting Tang. Machine learning the real discriminant locus. *arXiv preprint arXiv:2006.14078*, 2020.

[BLOS22] Paul Breiding, Julia Lindberg, Wern Juin Gabriel Ong, and Linus Sommer. Real circles tangent to 3 conics. *arXiv preprint arXiv:2211.06876*, 2022.

[BRTW22] Bubacarr Bah, Holger Rauhut, Ulrich Terstiege, and Michael Westdickenberg. Learning deep linear neural networks: Riemannian gradient flows and convergence to global minimizers. *Information and Inference: A Journal of the IMA*, 11(1):307–353, 2022.

[CHLM21] Heng-Yu Chen, Yang-Hui He, Shailesh Lal, and Suvajit Majumder. Machine learning lie structures & applications to physics. *Physics Letters B*, 817:136297, 2021.

[CLC22] Yacine Chitour, Zhenyu Liao, and Romain Couillet. A geometric approach of gradient descent algorithms in linear neural networks. *Mathematical Control and Related Fields*, pages 0–0, 2022.

[DLQ22] Michael Douglas, Subramanian Lakshminarasimhan, and Yidi Qi. Numerical calabi-yau metrics from holomorphic networks. In *Mathematical and Scientific Machine Learning*, pages 223–252. PMLR, 2022.

[GL22] J Elisenda Grigsby and Kathryn Lindsey. On transversality of bent hyperplane arrangements and the topological expressiveness of relu neural networks. *SIAM Journal on Applied Algebra and Geometry*, 6(2):216–242, 2022.

[Gra23] Grace Zhang. Kernel. https://medium.com/@zxr.nju/what-is-the-kernel-trick-why-is-it-important-98a98db0961d, 2023. [Online; accessed March 18, 2023].

[GRK20] Ingo Gühring, Mones Raslan, and Gitta Kutyniok. Expressivity of deep neural networks. *arXiv preprint arXiv:2007.04759*, 2020.

[HKS22] Kathryn Heal, Avinash Kulkarni, and Emre Can Sertöz. Deep learning gauss–manin connections. *Advances in Applied Clifford Algebras*, 32(2):24, 2022.

[Kaw16] Kenji Kawaguchi. Deep learning without poor local minima. *Advances in neural information processing systems*, 29, 2016.

[KMMT22] Kathlén Kohn, Thomas Merkh, Guido Montúfar, and Matthew Trager. Geometry of linear convolutional networks. *SIAM Journal on Applied Algebra and Geometry*, 6(3):368–406, 2022.

[KTB19] Joe Kileel, Matthew Trager, and Joan Bruna. On the expressive power of deep polynomial neural networks. *Advances in neural information processing systems*, 32, 2019.

[LB18] Thomas Laurent and James Brecht. Deep linear networks with arbitrary loss: All local minima are global. In *International conference on machine learning*, pages 2902–2907. PMLR, 2018.

[LLPS93] Moshe Leshno, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867, 1993.

[LXT+18] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31, 2018.

[MCTH21] Dhagash Mehta, Tianran Chen, Tingting Tang, and Jonathan D Hauenstein. The loss surface of deep linear networks viewed through the algebraic geometry lens. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5664–5680, 2021.

[MPTV06] Bernard Mourrain, Nicos G Pavlidis, Dimitris K Tasoulis, and Michael N Vrahatis. Determining the number of real roots of polynomials through neural networks. *Computers & Mathematics with Applications*, 51(3-4):527–536, 2006.

[MRZ22] Guido Montúfar, Yue Ren, and Leon Zhang. Sharp bounds for the number of regions of maxout networks and vertices of minkowski sums. *SIAM Journal on Applied Algebra and Geometry*, 6(4):618–649, 2022.

[NRT21] Gabin Maxime Nguegnang, Holger Rauhut, and Ulrich Terstiege. Convergence of gradient descent for learning linear neural networks. *arXiv preprint arXiv:2108.02040*, 2021.

[Saz06] Murat H Sazli. A brief review of feed-forward neural networks. *Communications Faculty of Sciences University of Ankara Series A2-A3 Physical Sciences and Engineering*, 50(01), 2006.

[SV08] Alex Smola and SVN Vishwanathan. Introduction to machine learning. *Cambridge University, UK*, 32(34):2008, 2008.

[TFHV21] Salma Tarmoun, Guilherme Franca, Benjamin D Haeffele, and Rene Vidal. Understanding the dynamics of gradient flow in overparameterized linear models. In *International Conference on Machine Learning*, pages 10153–10161. PMLR, 2021.

[TKB19] Matthew Trager, Kathlén Kohn, and Joan Bruna. Pure and spurious critical points: a geometric study of linear networks. *arXiv preprint arXiv:1910.01671*, 2019.

[VZH22] René Vidal, Zhihui Zhu, and Benjamin D Haeffele. Optimization landscape of neural networks. *Mathematical Aspects of Deep Learning*, page 200, 2022.

[WTP+19] Francis Williams, Matthew Trager, Daniele Panozzo, Claudio Silva, Denis Zorin, and Joan Bruna. Gradient dynamics of shallow univariate relu networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[ZNL18] Liwen Zhang, Gregory Naitzat, and Lek-Heng Lim. Tropical geometry of deep neural networks. In *International Conference on Machine Learning*, pages 5824–5832. PMLR, 2018.