Neural Networks and Discriminant Loci of Parameterized Polynomial Systems: First Examples

Wern Juin Gabriel Ong [1] Anna Seigal[2]

[1]Bowdoin College [2]Harvard University

Learning Discriminants

Consider a polynomial system with 0-dimensional solution set. The discriminant divides the parameter space into open subsets on which the number of real solutions is constant. We study the discriminant as the *decision boundary* of a classifier that, given a point of the parameter space, returns the number of real roots.

Learning the Discriminant

1. Generate training data: points of the parameter space labeled by the number of real solutions.
2. Train a neural network to classify the number of real solutions.
3. Sample points from the decision boundary of the neural network.
4. Interpolate polynomials of varying degrees through the sampled points.
5. Choose the polynomial of the lowest degree where the error starts to plateau.

Learning the Quadratic Discriminant

A quadratic polynomial $f(x) = x^2 + ax + b$ has discriminant $b = 0.25a^2$. Our method recovers $b = (2.49949210 \times 10^{-1})a^2 - (1.82254776 \times 10^{-6})a + (1.47289991 \times 10)^{-5}$, by interpolating a degree two polynomial through points on the decision boundary.

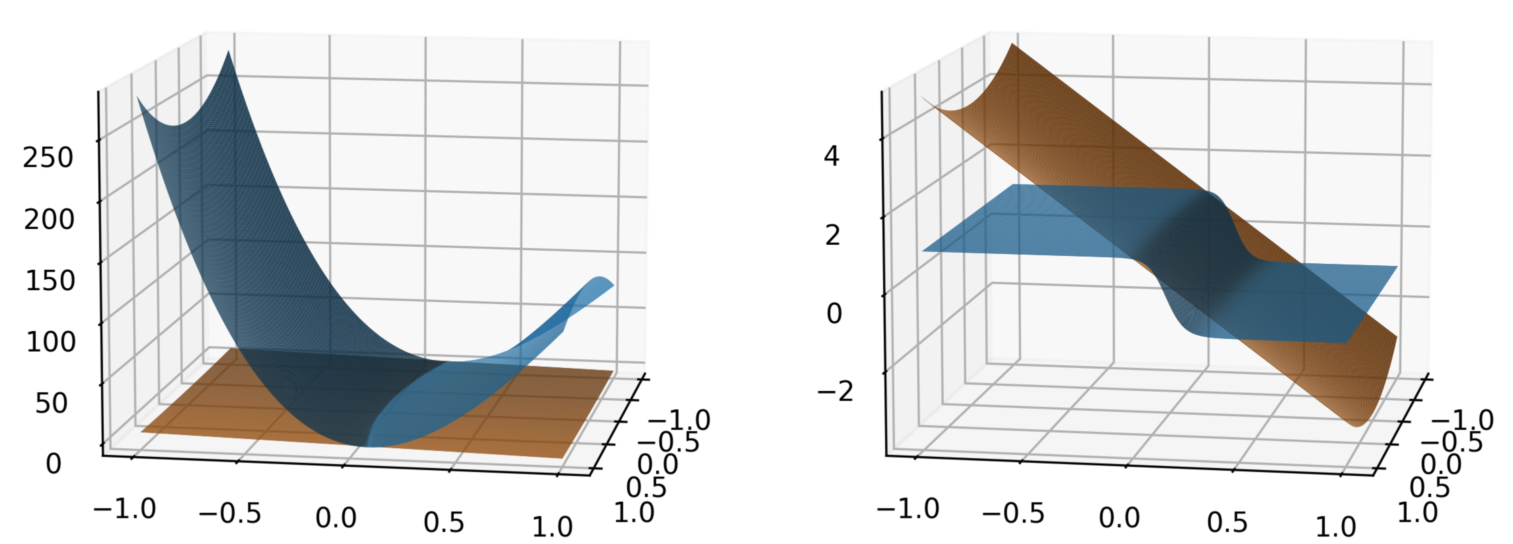

Figure 1. While both the discriminant are defined by the vanishing loci of polynomials $F(a, b)$, our plots display $z = F(a, b)$. **(Left)** The quadratic discriminant $z = 0.25a^2 - b$ (orange) and boundary approximator for a semi-quadratic network $z = F(a, b)$ (blue). **(Right)** The quadratic discriminant $z = 0.25a^2 - b$ (orange) and boundary approximator for a quadratic-`SoftMax` network (blue), with `SoftMax` activation in the final layer and quadratic activations elsewhere. Note that our (approximate) discriminant curves are defined by the intersection of these surfaces with $z = 0$ and partitions the a, b-parameter space into two cells where the number of real solutions is constant.

Observe that in the case of `SoftMax` activation in the final layer **Figure 1 (Right)**, it is difficult to sample points as described in **step 3** as the root-finding algorithm does not converge unless it starts close to the discriminant curve.

Polynomial Neural Networks

A polynomial neural network $\mathbb{R}^{d_0} \to \mathbb{R}^{d_1} \to \cdots \to \mathbb{R}^{d_k}$ has **depth** k and **width** $d_1 = \cdots = d_{k-1}$, where each $\mathbb{R}^{d_{i-1}} \to \mathbb{R}^{d_i}$ is a function $\sigma_i(W\mathbf{x} + b)$ where $x \in \mathbb{R}^{d_{i-1}}, W \in \mathbb{R}^{d_i \times d_{i-1}}, b \in \mathbb{R}^{d_i}$ and σ_i is a pointwise m-th power function (ie. $(x_i)_{i \in [n]} \mapsto (x_i^m)_{i \in [n]}$). We consider polynomial neural networks where σ_i is either the pointwise square function or the identity function. A **quadratic neural network** has σ_i the pointwise squaring function for all i while a **semi-quadratic neural network** has σ_1 the pointwise squaring function and $\sigma_i = \mathrm{id}$ for $i > 1$.

The Discriminant and (Semi)-Quadratic Neural Networks

In **step 3**, we recover points on the decision boundary. We expect that we will be able to recover the discriminant when it lies in the function class of the boundary approximator, which we define to be the function class of the polynomial $\mathbb{R}^2 \to \mathbb{R}$ defined by taking the inner product of the network output in with $(1, -1)$.

Theorem (Discriminants in Function Classes of Polynomial Neural Networks)

The containment of the quadratic discriminant in the function class of the boundary approximator for a neural network with a given architecture is as follows:

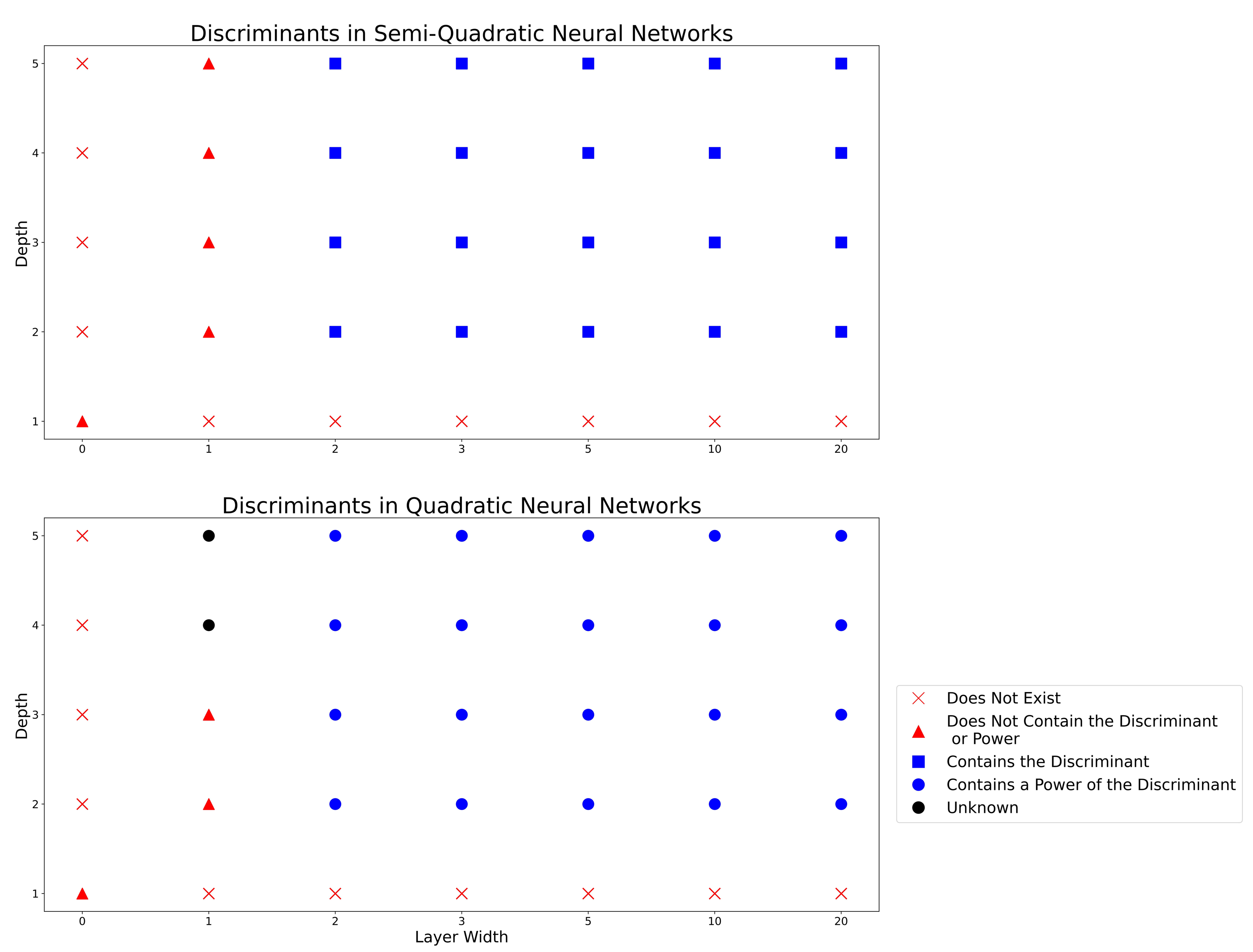

Figure 2. Containment of the quadratic discriminant in quadratic and semi-quadratic neural networks.

Width 0 Depth 1 Quadratic Networks

A width 0 depth 1 quadratic network is determined by six parameters, 4 weights and 2 biases:

$$W = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}, b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

determines the plane conic curve $(w_{11}a + w_{12}b + b_1)^2 - (w_{21}a + w_{22}b + b_2)^2$. The map from the set of the parameters to the set of coefficients of the plane conic is given by

$$(w_{11}, w_{12}, w_{21}, w_{22}, b_1, b_2) \mapsto (w_{11}^2 - w_{21}^2, 2w_{11}w_{12} - 2w_{21}w_{22}, w_{12}^2 - w_{22}^2, 2w_{11}b_1 - 2w_{21}b_2, 2w_{12}b_1 - 2w_{22}b_2, b_1^2 - b_2^2)$$

with finite (possibly empty) fibers.

Proposition (Fiber Over The Discriminant)

The fiber over the discriminant is empty, there do not exist parameters W, b such that $(1, -1)^T \mathrm{Sq}\,(W\mathbf{x} + b) = 0.25a^2 - b$, however the boundary approximator converges to a point with a flat loss landscape.

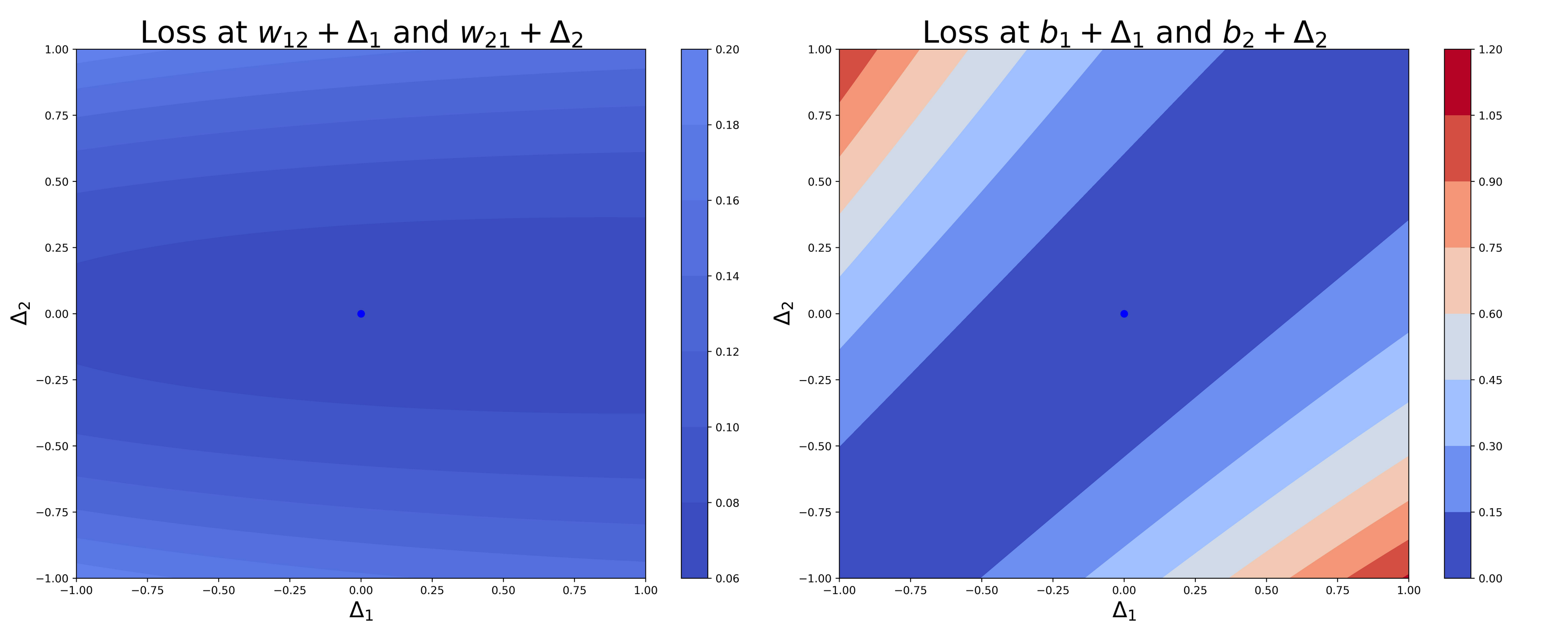

Figure 3. Loss landscape of width 0 quadratic network with respect to pairs of parameters w_{12}, w_{21} and b_1, b_2. Observe that the low error indicates that our quadratic network converges to a point of good fit.

References

[1] Paul Breiding and Sascha Timme. HomotopyContinuation.jl: A package for homotopy continuation in Julia, 2018.

[2] Paul Breiding, Türkü Özlüm Çelik, Timothy Duff, Alexander Heaton, Aida Maraj, Anna-Laura Sattelberger, Lorenzo Venturello, and Oğuzhan Yürük. Nonlinear algebra and applications, 2021.

[3] James H. Davenport and Matthew England. Need polynomial systems be doubly-exponential? In *Mathematical Software – ICMS 2016*, pages 157–164. Springer International Publishing, 2016.

[4] Jan Draisma, Emil Horobet, Giorgio Ottaviani, Bernd Sturmfels, and Rekha R. Thomas. The euclidean distance degree of an algebraic variety. *Foundations of Computational Mathematics*, 16:99–149, 2016.

[5] Alexander Esterov. The discriminant of a system of equations. *Advances in Mathematics*, 245:534–572, 2013.

[6] I.M. Gelfand, M. Kapranov, and A. Zelevinsky. *Discriminants, Resultants, and Multidimensional Determinants*. Modern Birkhäuser Classics. Birkhäuser Boston, 2009.

[7] Charles R. Harris, K. Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585:357–362, 2020.

[8] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[9] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

[10] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 9.5)*, 2022.