Geometry of neural networks

Marta Panizzut

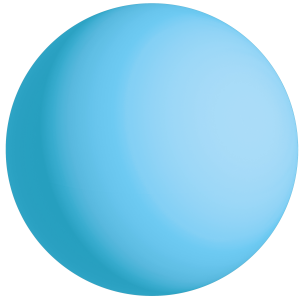
Math and Machine Learning Seminars
UiT – April 24, 2026

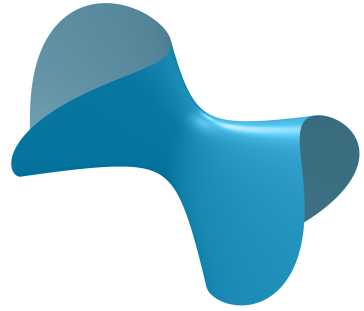
UiT Norges arktiske universitet

LIE-STØRMER  CENTER

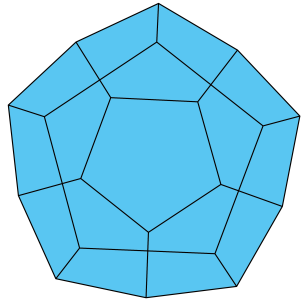
What do I do?

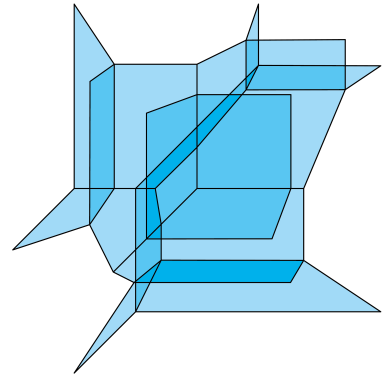
Algebraic geometry:





Discrete and polyhedral geometry:





Computational algebra: Symbolic computations.

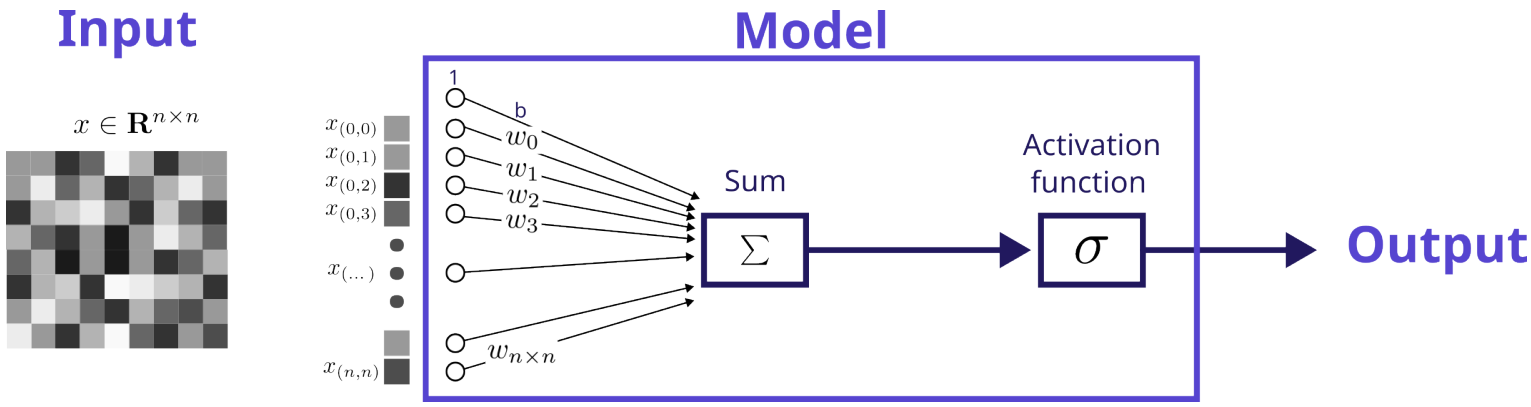
Why am I here?

- Zhang, Naitzat, Lim, Tropical Geometry of Deep Neural Networks, International Conference on Machine Learning (2018).
- Montúfar, Ren, Zhang, Sharp bounds for the number of regions of maxout networks and vertices of Minkowski sums. SIAM J. Appl. Algebra Geom. 6 (2022), no. 4, 618–649.
- Hertrich, Basu, Di Summa, Skutella, Towards lower bounds on the depth of ReLU neural networks. SIAM J. Discrete Math. 37 (2023), no. 2, 997–1029.
- Haase, Hertrich, Loho, Lower Bounds on the Depth of Integral ReLU Neural Networks via Lattice Polytopes. ICLR (2023)
- Brandenburg, Loho, Montúfar, The Real Tropical Geometry of Neural Networks. Transactions of Machine Learning Research (2024).
- Grillo, Hertrich, Loho, Depth-Bounds for Neural Networks via the Braid Arrangement. NeurIPS (2025)
- ...

- Survey: `https://arxiv.org/pdf/2305.00241`

Perceptron

Recall from Antonio's talk...



The **perceptron** produces

$$\hat{y} = \sigma(w^\top x + b)$$

where:

- $w \in \mathbb{R}^d$ are the weights
- $b \in \mathbb{R}$ is the bias
- σ is an activation function. Common examples include:

ReLU: $\sigma(x) = \max(0, x)$

Tanh: $\sigma(x) = \tanh(x)$

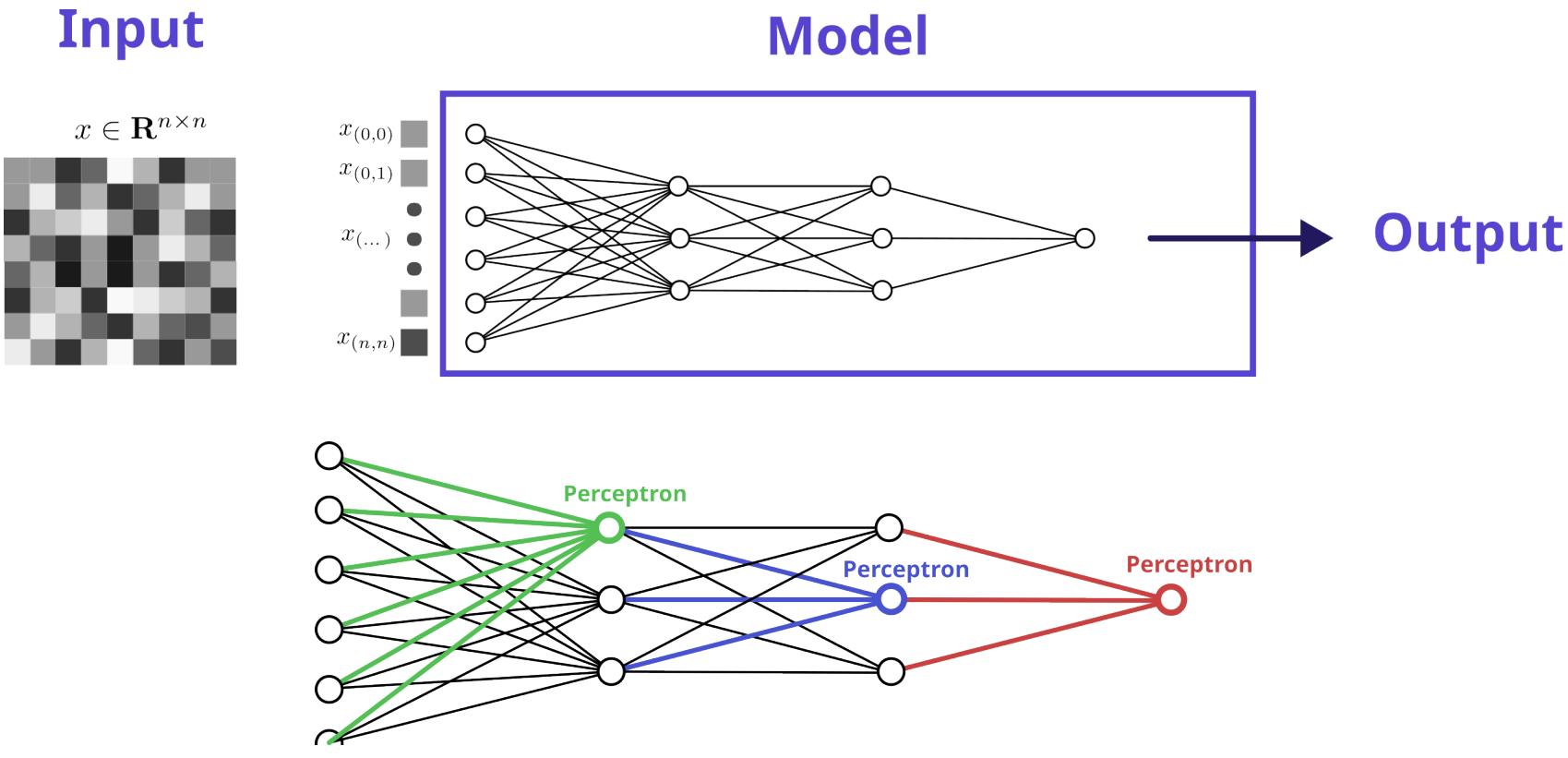
Sigmoid: $\sigma(x) = \frac{1}{1+e^{-x}}$

Multi-Layer Perceptron

Recall from Antonio's talk...

A **multi-layer perceptron (MLP)** is obtained by composing functions which have perceptrons as components.

$$x \to h^{(1)} \to h^{(2)} \to \cdots \to \hat{y}$$



Multi-Layer Perceptron (MLP)

Recall from Antonio's talk...

Formally, an MLP is a function defined by the composition of L layers.

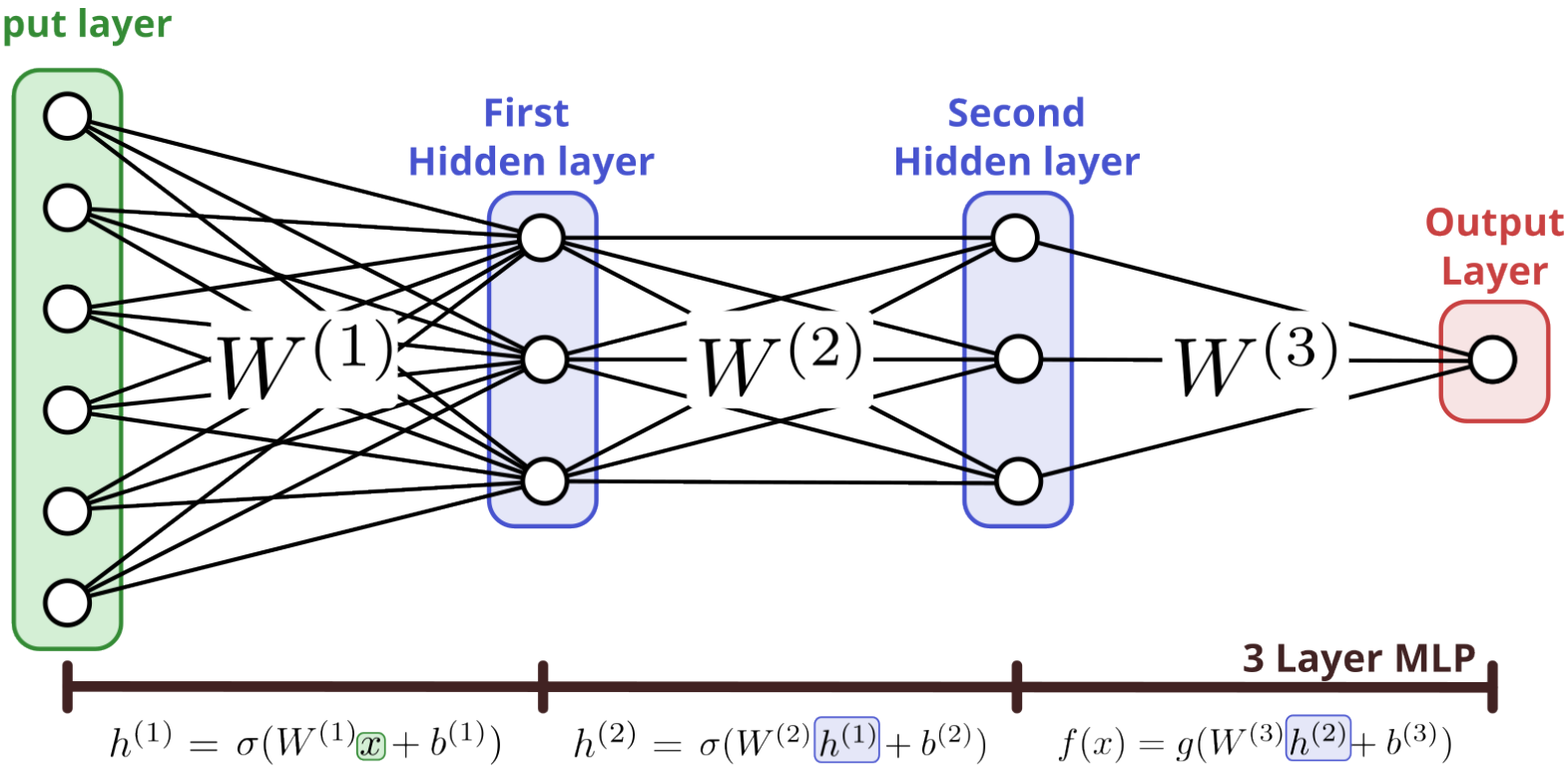
$$
\begin{aligned}
h^{(0)} &= x \\
h^{(\ell)} &= \sigma\big(W^{(\ell)} h^{(\ell-1)} + b^{(\ell)}\big), \qquad \ell = 1, \ldots, L-1 \\
f_\theta(x) &= W^{(L)} h^{(L-1)} + b^{(L)}
\end{aligned}
$$

The parameters are:

$$\theta = \{W^{(\ell)}, b^{(\ell)}\}_{\ell=1}^{L}$$

- $W \in \mathbb{R}^{i \times j}$ are the weights
- $b \in \mathbb{R}^{k}$ are the bias
- σ is an activation function.

Multi-Layer Perceptron (MLP) Visual



$$f(x) = g(W^{(3)}\sigma(W^{(2)}\sigma(W^{(1)}x + b^{(1)}) + b^{(2)}) + b^{(3)})$$

"Feature map"

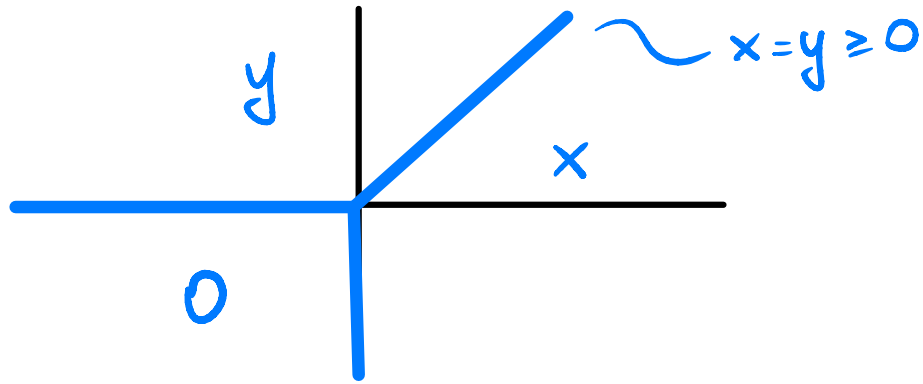
Simple classifier

Regions of MLPs

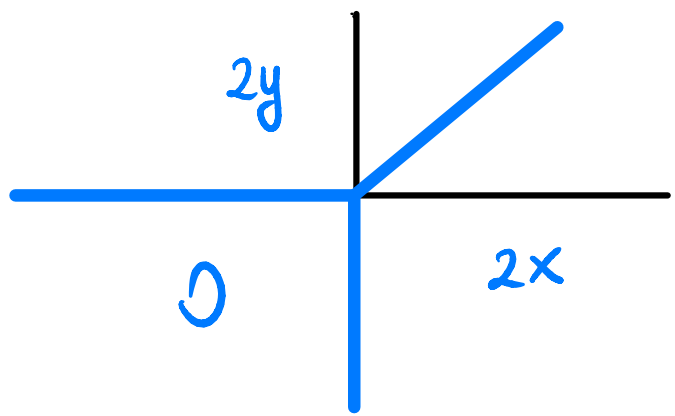
Suppose $f : \mathbb{R}^d \to \mathbb{R}^n$ is a MLP.
Question: We want to compute the regions in which the domain $\mathbb{R}^d$ gets partitioned depending on the output of f.

Examples:

$$f : \mathbb{R}^2 \to \mathbb{R}, \quad f = \max\{x, y, 0\}$$



$$f : \mathbb{R}^2 \to \mathbb{R}, \quad f = \max\{2x,\ 2y,\ x,\ y,\ x+y, 0\}$$



Example

$$h^{(1)} : \mathbb{R}^2 \to \mathbb{R}^5, \quad h^{(2)} : \mathbb{R}^5 \to \mathbb{R}, \quad f = h^{(2)} \circ h^{(1)} : \mathbb{R}^2 \to \mathbb{R}$$

$$y = h^{(1)}(x_1, x_2) = \max\left\{ \begin{bmatrix} 1 & 1 \\ 1 & 3 \\ 1 & 2 \\ 4 & 1 \\ 3 & 2 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 2 \\ 0 \\ 2 \end{bmatrix}, 0 \right\}$$

$$h^{(2)}(y) = \max\left\{ [1\ 2\ 1\ 1\ 3] \cdot [y_1\ y_2\ y_3\ y_4\ y_5]^T, 0 \right\}$$

Let's first re-write $h^{(1)}$ and $h^{(2)}$:

$$y = h^{(1)}(x_1, x_2) = \max\left\{ \begin{bmatrix} 1 + x_1 + x_2 \\ 1 + x_1 + 3x_2 \\ 2 + x_1 + 2x_2 \\ 0 + 4x_1 + x_2 \\ 2 + 3x_1 + 2x_2 \end{bmatrix}, 0 \right\}$$

$y_1 = \max\{1 + x_1 + x_2, 0\}$

$y_2 = \max\{1 + x_1 + 3x_2, 0\}$

$y_3 = \ldots$

$$h^{(2)}(y) = \max\{y_1 + 2y_2 + y_3 + y_4 + 3y_5, 0\}$$

Now we need to substitute y in h_2. How can we do this symbolically?

Tropical algebra

$(\mathbb{R} \cup \{-\infty\}, \oplus, \odot)$ with the operations $a \oplus b = \max\{a, b\}$ and $a \odot b = a + b$ and is **max-plus algebra** or **tropical semiring**.

$$y = h^{(1)}(x_1, x_2) = \max\left\{ \begin{bmatrix} 1 + x_1 + x_2 \\ 1 + x_1 + 3x_2 \\ 2 + x_1 + 2x_2 \\ 0 + 4x_1 + x_2 + 0 \\ 2 + 3x_1 + 2x_2 \end{bmatrix}, 0 \right\} = \begin{bmatrix} y_1 = 1 \odot x_1 \odot x_2 \oplus 0 \\ y_2 = 1 \odot x_1 \odot x_2^3 \oplus 0 \\ 2 \odot x_1 \odot x_2^2 \oplus 0 \\ x_1^4 \odot x_2 \oplus 0 \\ 2 \odot x_1^3 \odot x_2^2 \oplus 0 \end{bmatrix}$$

$$h^{(2)}(y) = y_1 \odot y_2^2 \odot y_3 \odot y_4 \odot y_5^3 \oplus 0$$

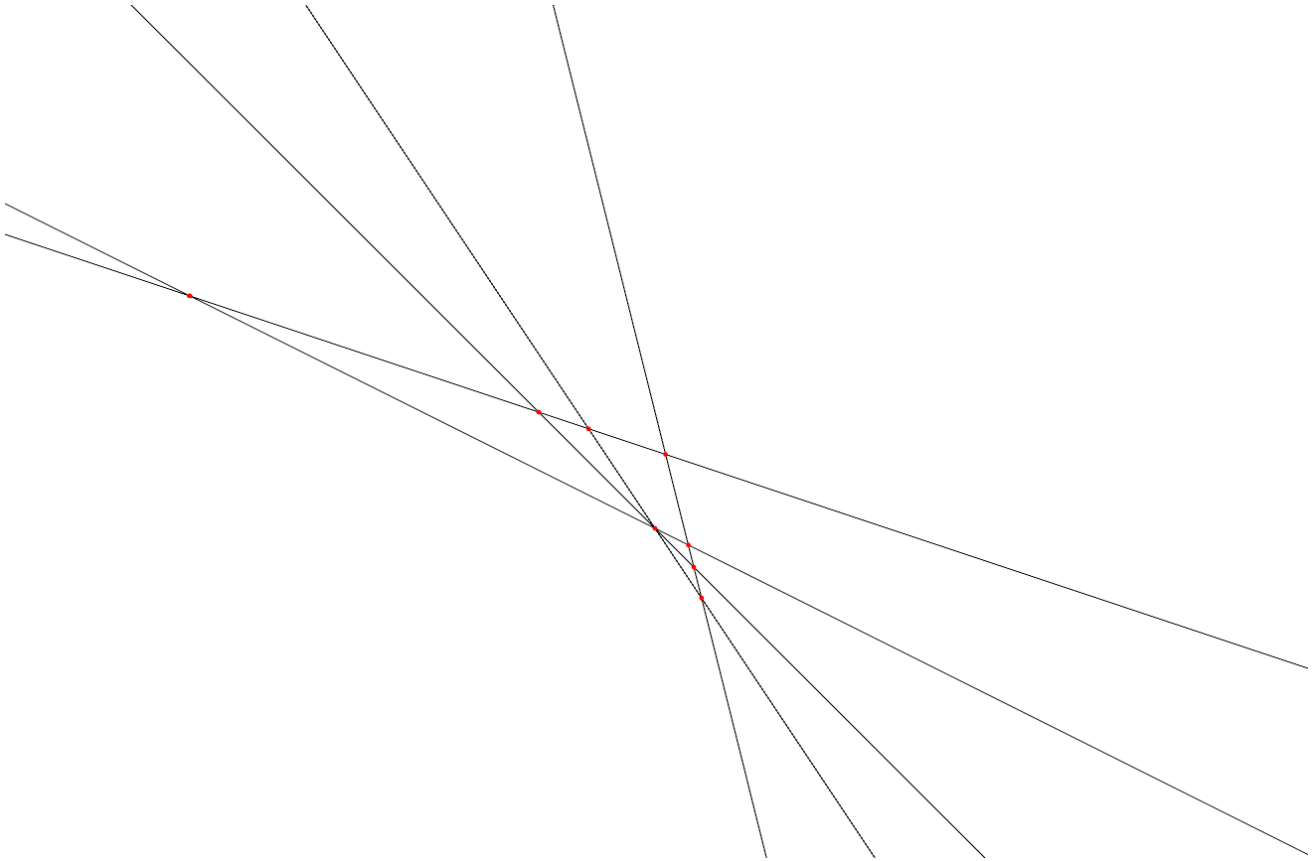
I can manipulate $h^{(1)}$ and $h^{(2)}$ as (tropical) polynomials.

$$\begin{aligned} f(x_1, x_2) &= 11 \odot x_1^{13} \odot x_2^{16} \oplus 10 \odot x_1^{12} \odot x_2^{15} \oplus 9 \odot x_1^{12} \odot x_2^{14} \oplus \cdots \oplus 0 \\ &= \max\{11 + 13x_1 + 16x_2, 10 + 12x_1 + 15x_2, \ldots, 0\} \end{aligned}$$

Discrete geometry

$$\begin{aligned} f(x_1, x_2) &= 11 \odot x_1^{13} \odot x_2^{16} \oplus 10 \odot x_1^{12} \odot x_2^{15} \oplus 9 \odot x_1^{12} \odot x_2^{14} \oplus \cdots \oplus 0 \\ &= \max\{11 + 13x_1 + 16x_2, 10 + 12x_1 + 15x_2, \ldots, 0\} \end{aligned}$$

So h is a **piecewise linear function**, and the regions will be **polyhedra**. The domain $\mathbb{R}^2$ gets subdivided as **polyhedral complex**... **Discrete geometry!**



Code in OSCAR

```
julia> T = tropical_semiring(max)
Max tropical semiring

julia> T, (x,y) = polynomial_ring(T, [:x,:y])
(Multivariate polynomial ring in 2 variables over max tropical semiring, Abstrac
tAlgebra.Generic.MPoly{TropicalSemiringElem{typeof(max)}}[x, y])
```

```
julia> f = y1*y2^2*y3^3*y4*y5+0
(11)*x^13*y^16 + (10)*x^12*y^15 + (9)*x^12*y^14 + (10)*x^12*y^13 + (8)*x^11*y^13
 + (9)*x^10*y^14 + (11)*x^9*y^15 + (9)*x^11*y^12 + (8)*x^11*y^11 + (8)*x^9*y^13
+ (10)*x^8*y^14 + (9)*x^11*y^10 + (6)*x^10*y^11 + (7)*x^9*y^12 + (9)*x^8*y^13 +
(7)*x^10*y^10 + (8)*x^9*y^11 + (10)*x^8*y^12 + (8)*x^10*y^9 + (6)*x^8*y^11 + (8)
*x^7*y^12 + (9)*x^6*y^13 + (7)*x^10*y^8 + (4)*x^9*y^9 + (7)*x^8*y^10 + (9)*x^7*y
^11 + (5)*x^9*y^8 + (6)*x^8*y^9 + (8)*x^7*y^10 + (8)*x^5*y^12 + (6)*x^9*y^7 + (7
)*x^8*y^8 + (9)*x^7*y^9 + (6)*x^6*y^10 + (7)*x^5*y^11 + (5)*x^9*y^6 + (5)*x^7*y^
8 + (7)*x^6*y^9 + (8)*x^5*y^10 + (3)*x^8*y^6 + (6)*x^7*y^7 + (8)*x^6*y^8 + (6)*x
^4*y^10 + (4)*x^8*y^5 + (5)*x^7*y^6 + (7)*x^6*y^7 + (4)*x^5*y^8 + (7)*x^4*y^9 +
(3)*x^8*y^4 + (3)*x^6*y^6 + (5)*x^5*y^7 + (6)*x^4*y^8 + (4)*x^6*y^5 + (6)*x^5*y^
6 + (7)*x^4*y^7 + (4)*x^3*y^8 + (2)*x^7*y^3 + (3)*x^6*y^4 + (5)*x^5*y^5 + (5)*x^
3*y^7 + (1)*x^5*y^4 + (3)*x^4*y^5 + (6)*x^3*y^6 + (2)*x^5*y^3 + (4)*x^4*y^4 + (5
)*x^3*y^5 + (2)*x^2*y^6 + (1)*x^5*y^2 + (3)*x^4*y^3 + (3)*x^2*y^5 + (4)*x^2*y^4
+ x^4*y + (2)*x^3*y^2 + (3)*x^2*y^3 + (1)*x*y^3 + (2)*x*y^2 + (1)*x*y + (0)

julia> V = tropical_hypersurface(f)
Max tropical hypersurface

julia> visualize(V)
```

Tropical Geometry of Deep Neural Networks

Zhang, Naitzat, Lim '18:

- Geometric description of the decision boundary.
- Geometric complexity: The number of linear regions is bounded by $\mathcal{O}(n^{d(L-1)})$

Theorem: A feedforward neural network with weight matrices with integer-values, real-valued biases and activation functions $\max\{x, t\}$ is a function $h : \mathbb{R}^d \to \mathbb{R}^n$ whose coordinates are tropical rational functions of the input.

Theorem: A function $h : \mathbb{R}^d \to \mathbb{R}$ is a tropical rational function if and only if h is a feedforward neural network with weight matrices with integer-values, real-valued biases and activation functions $\max\{x, t\}$.

More on bounds of regions

Montúfar, Ren, Zhang '22:

- Results on the number of linear regions of shallow and deep feedforward neural networks with maxout units.

$$\sigma(x) = \max\{l_1(x),\ l_2(x), \ldots,\ l_d(x)\}, \quad l_i \text{ affine functions.}$$

- Use of bounds on the regions of hyperplane arrangements.

Lower Bounds on the Depth of ReLU Neural Networks

Hertrich, Basu, Di Summa, Skutella '23,
Haase, Hertrich, Loho '23:

- Universal approximation theorems: classes of functions which are **exactly representable** by neural networks with a certain architecture.
- Trade-off between *depth* (=number of layers) and *width* (= maximal dimension of vector spaces) of neural networks.

- Arora, Basu, Mianjy, and Mukherjee '18: a function is representable by a ReLU neural network if and only if it is continuous and piecewise linear, $\lceil \log_2(d+1) \rceil$ many hidden layers are sufficient to represent every CPWL function with d-dimensional input.
- Conjecture that this bound is sharp. Proofs of the conjecture in some cases.

Thank you!

Special thanks to Antonio for lending me some of his slides.