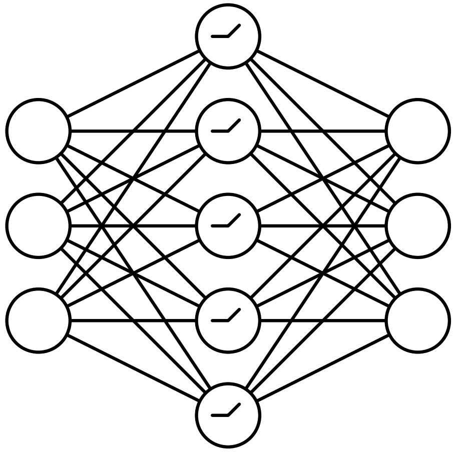
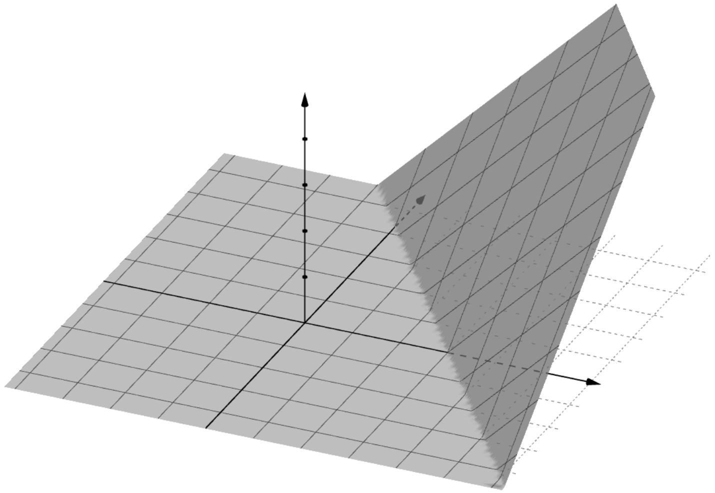
Training Fully Connected Neural Networks is $\exists\mathbb{R}$-Complete

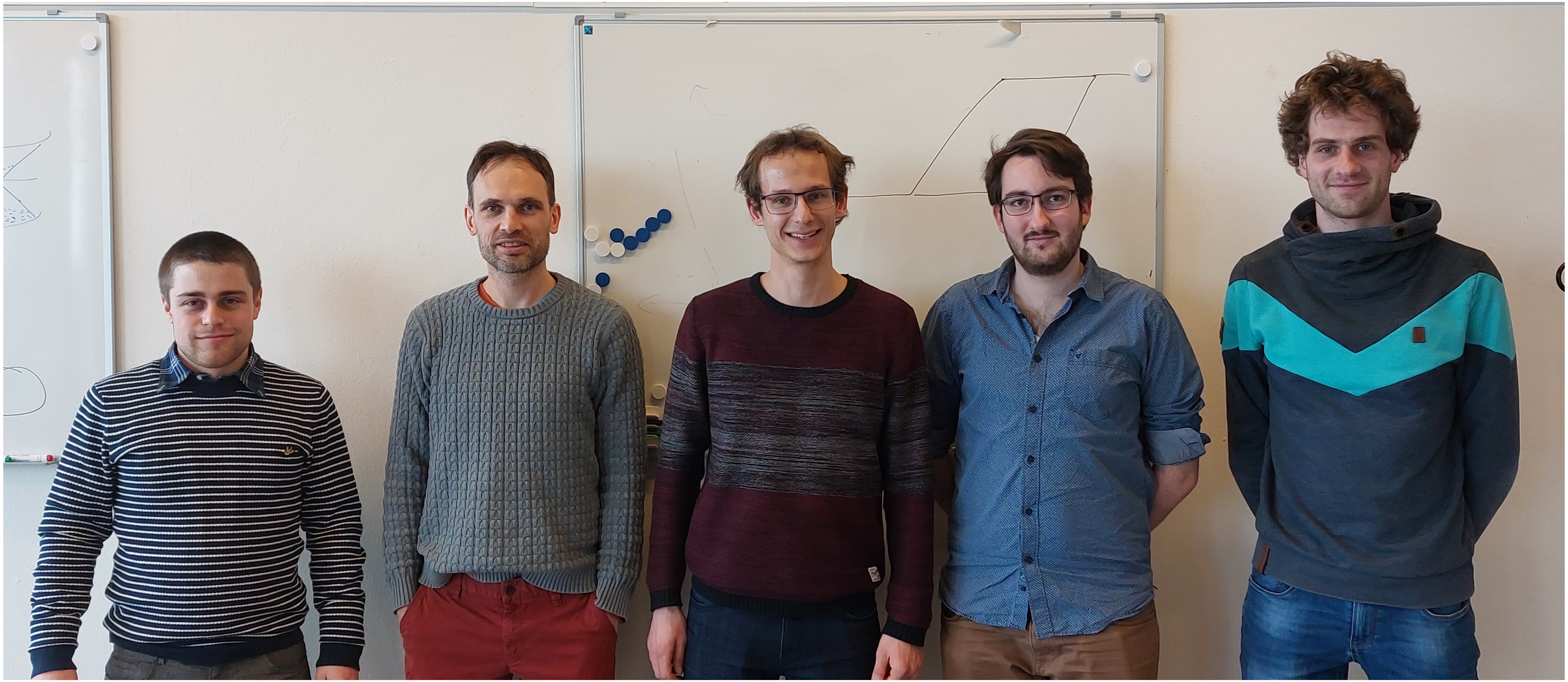
Daniel Bertschinger, Christoph Hertrich, **Paul Jungeblut**, Tillmann Miltzow, Simon Weber





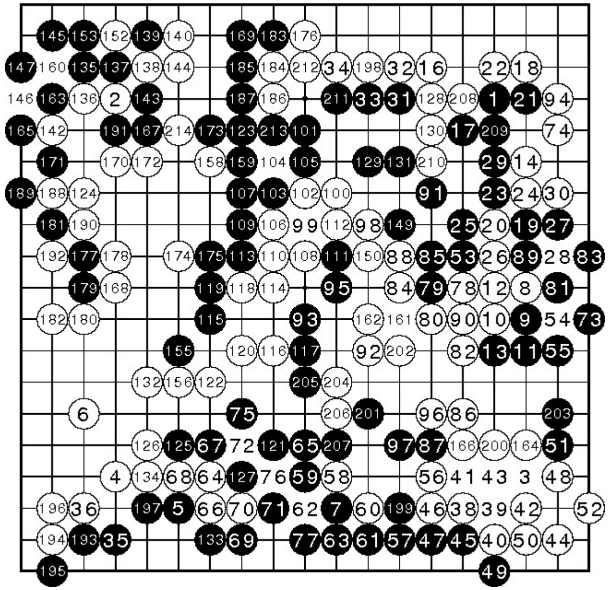
$$\exists X_1, \ldots, X_n \in \mathbb{R} \ : \ \Phi(X_1, \ldots, X_n)$$

Daniel Till Christoph Simon Paul



Motivation

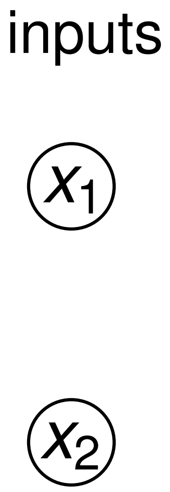
Neural Networks: The most successful tool in artificial intelligence.



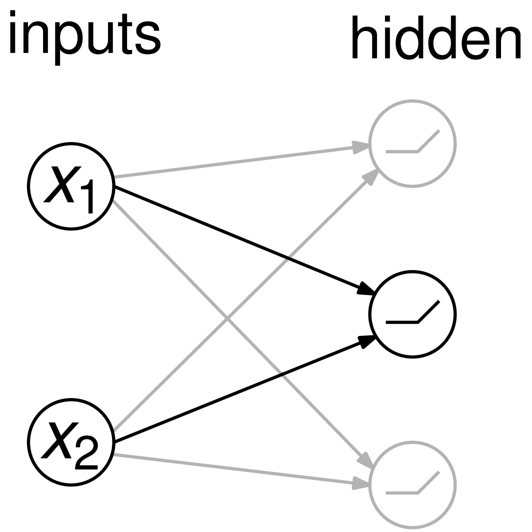
AlphaGo vs. Lee Sedol, 2016

photorealistic image generation
(StyleGAN, 2019)

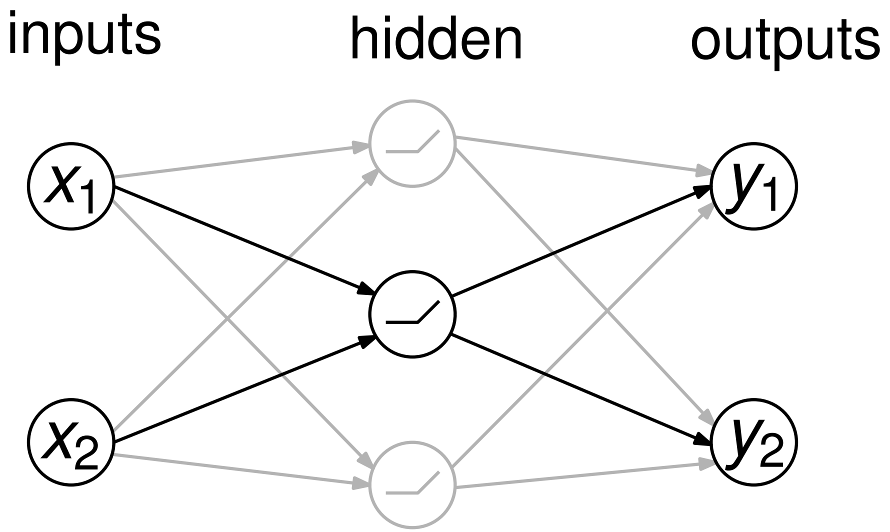
Neural Networks



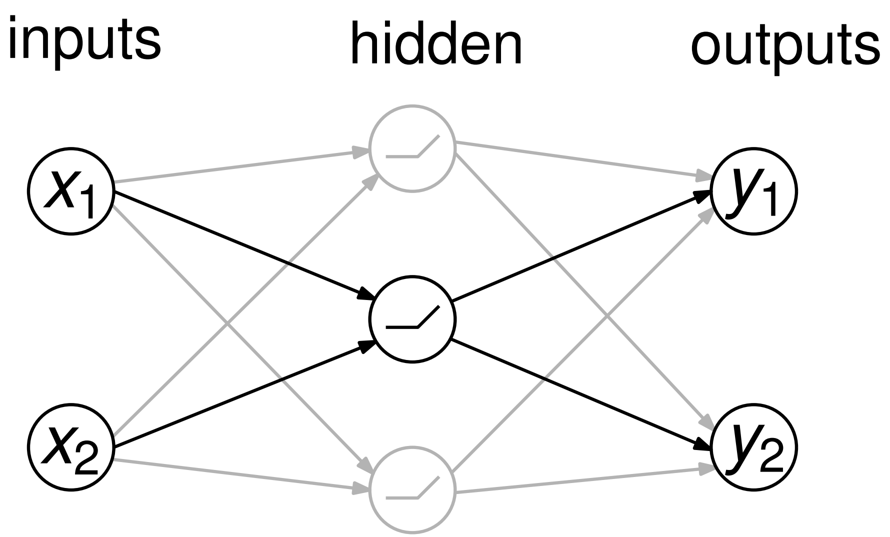
Neural Networks



Neural Networks

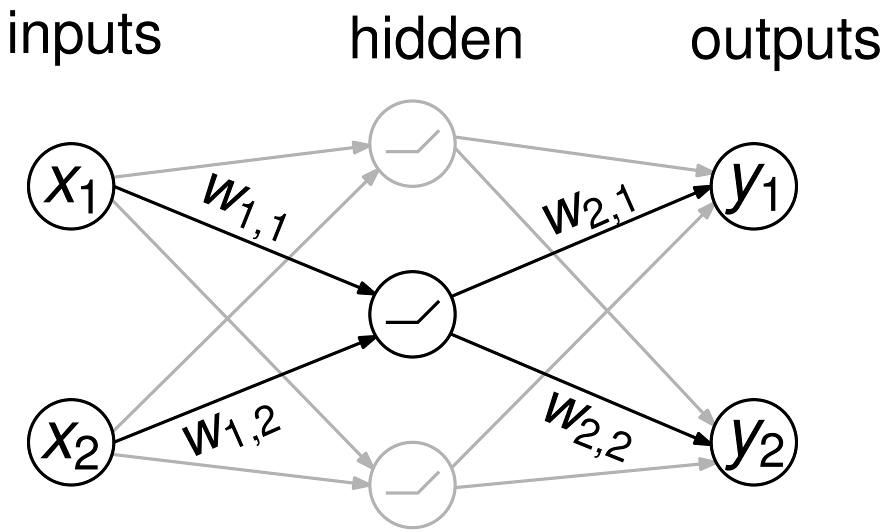


Neural Networks



Architecture: directed acyclic graph
(vertices = neurons)

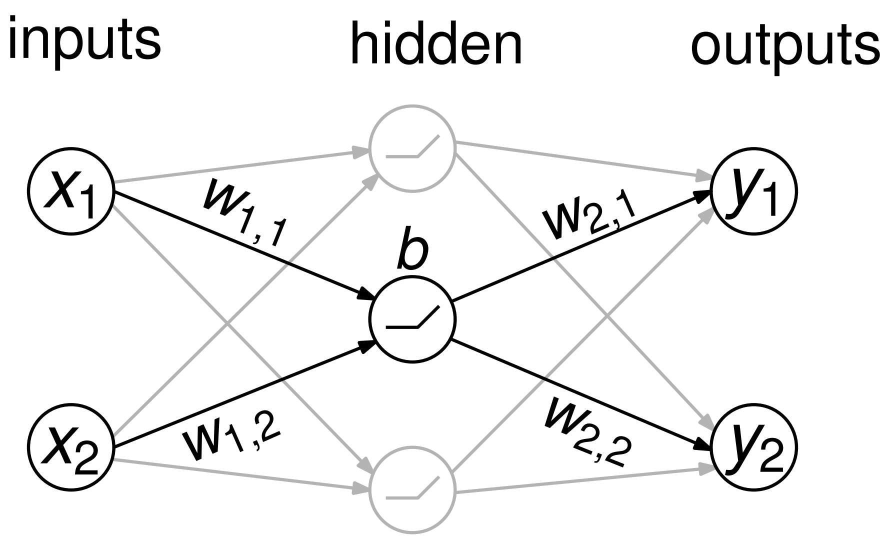
Neural Networks



Architecture: directed acyclic graph
(vertices = neurons)

Weights: on edges

Neural Networks

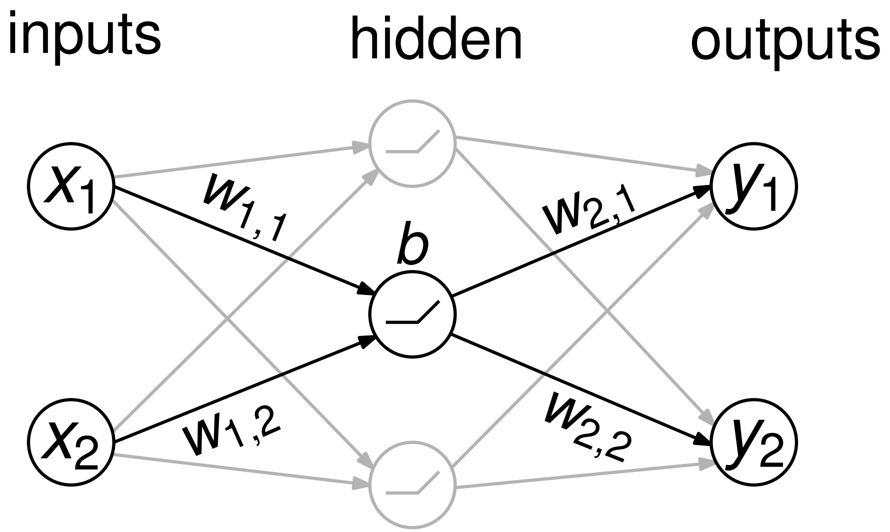


Architecture: directed acyclic graph
(vertices = neurons)

Weights: on edges

Biases: on hidden neurons

Neural Networks

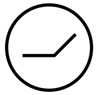


Architecture: directed acyclic graph

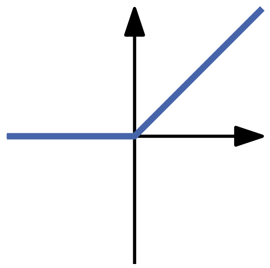
(vertices = neurons)

Weights: on edges

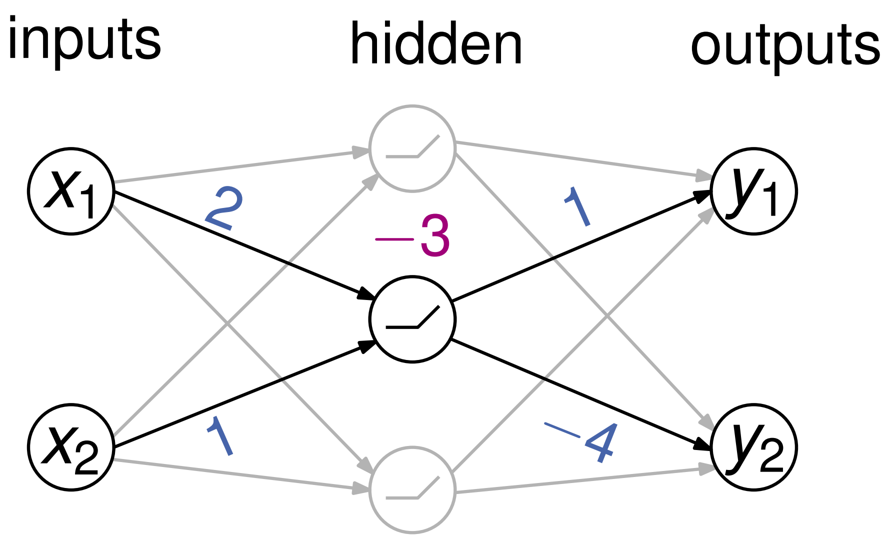
Biases: on hidden neurons

Activation Function:  = ReLU

$$\mathrm{ReLU} : \mathbb{R} \to \mathbb{R}$$
$$x \mapsto \max\{0, x\}$$



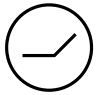
Neural Networks



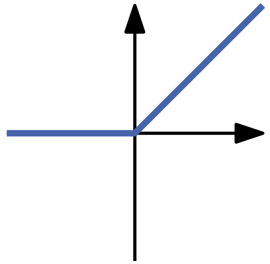
Architecture: directed acyclic graph
(vertices = neurons)

Weights: on edges

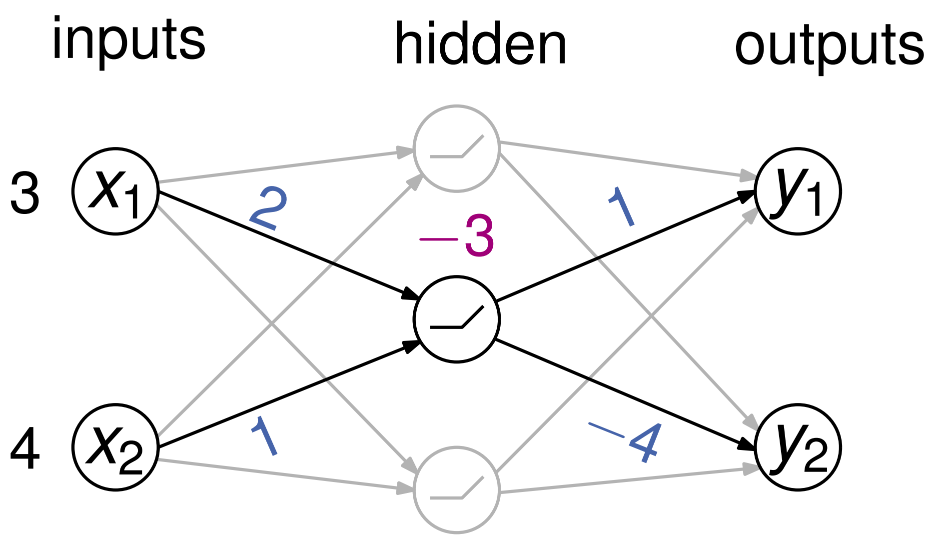
Biases: on hidden neurons

Activation Function:  = ReLU

$$\text{ReLU} : \mathbb{R} \to \mathbb{R}$$
$$x \mapsto \max\{0, x\}$$



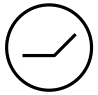
Neural Networks



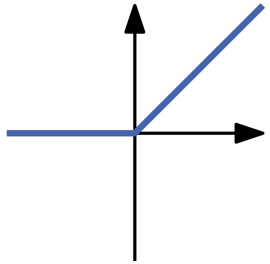
Architecture: directed acyclic graph
(vertices = neurons)

Weights: on edges

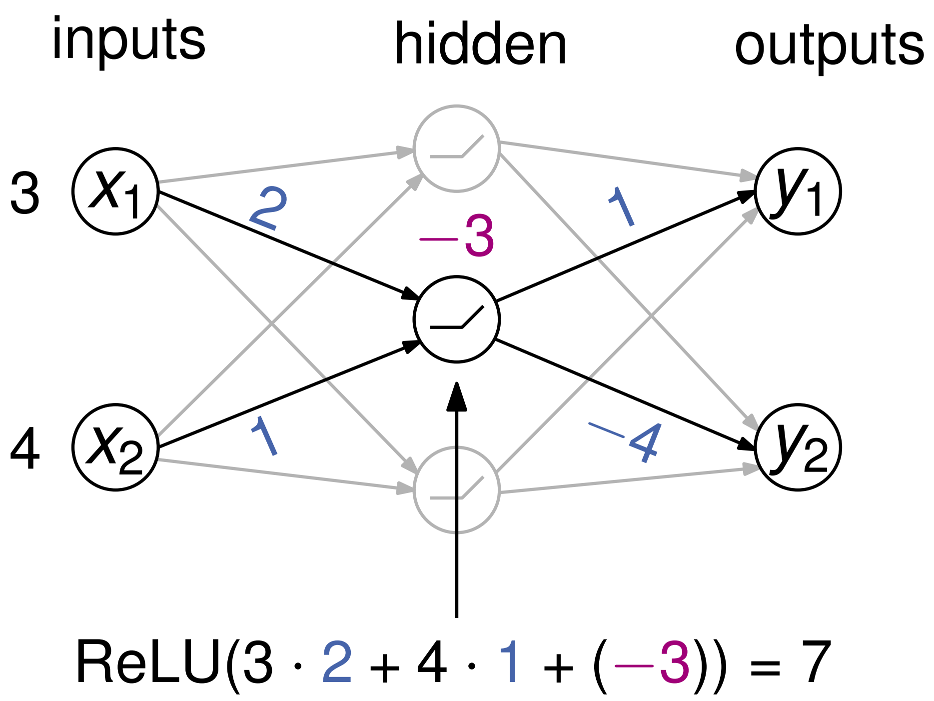
Biases: on hidden neurons

Activation Function:  = ReLU

$$\text{ReLU} : \mathbb{R} \to \mathbb{R}$$
$$x \mapsto \max\{0, x\}$$



Neural Networks



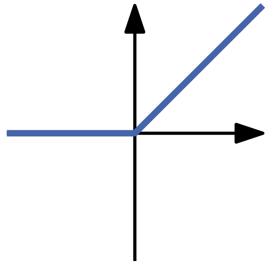
Architecture: directed acyclic graph
(vertices = neurons)

Weights: on edges

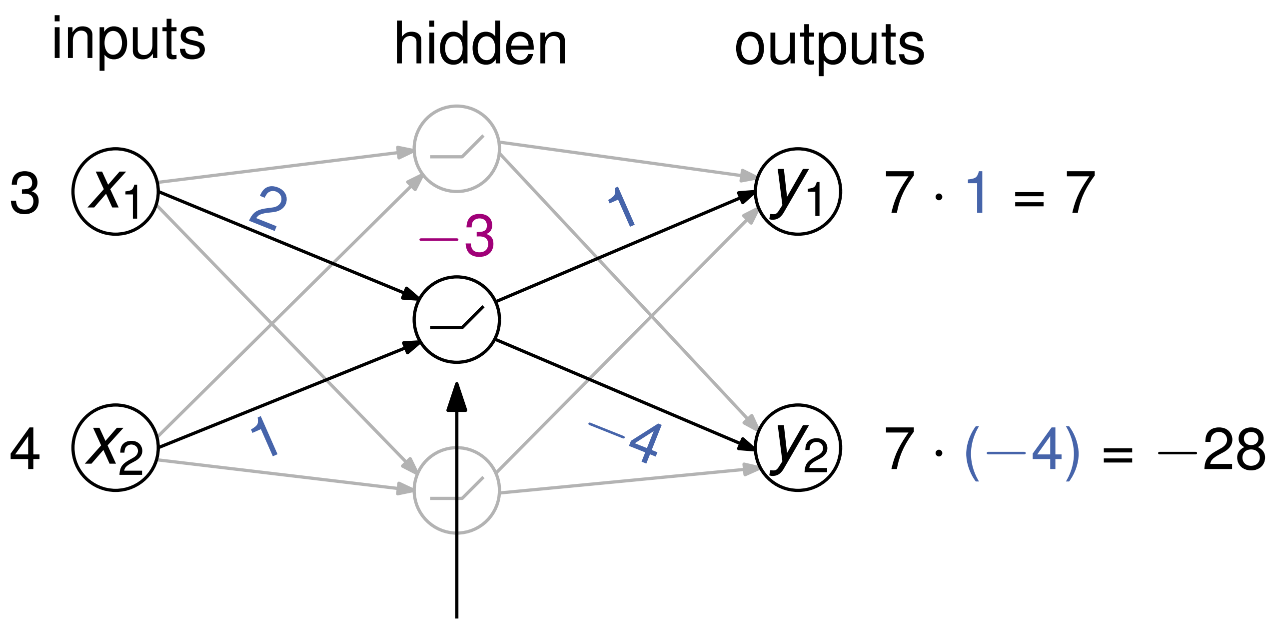
Biases: on hidden neurons

Activation Function: = ReLU

$$\text{ReLU} : \mathbb{R} \to \mathbb{R}$$
$$x \mapsto \max\{0, x\}$$



Neural Networks



$\mathrm{ReLU}(3 \cdot 2 + 4 \cdot 1 + (-3)) = 7$

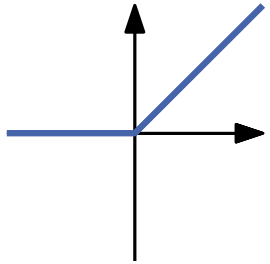
Architecture: directed acyclic graph
(vertices = neurons)

Weights: on edges

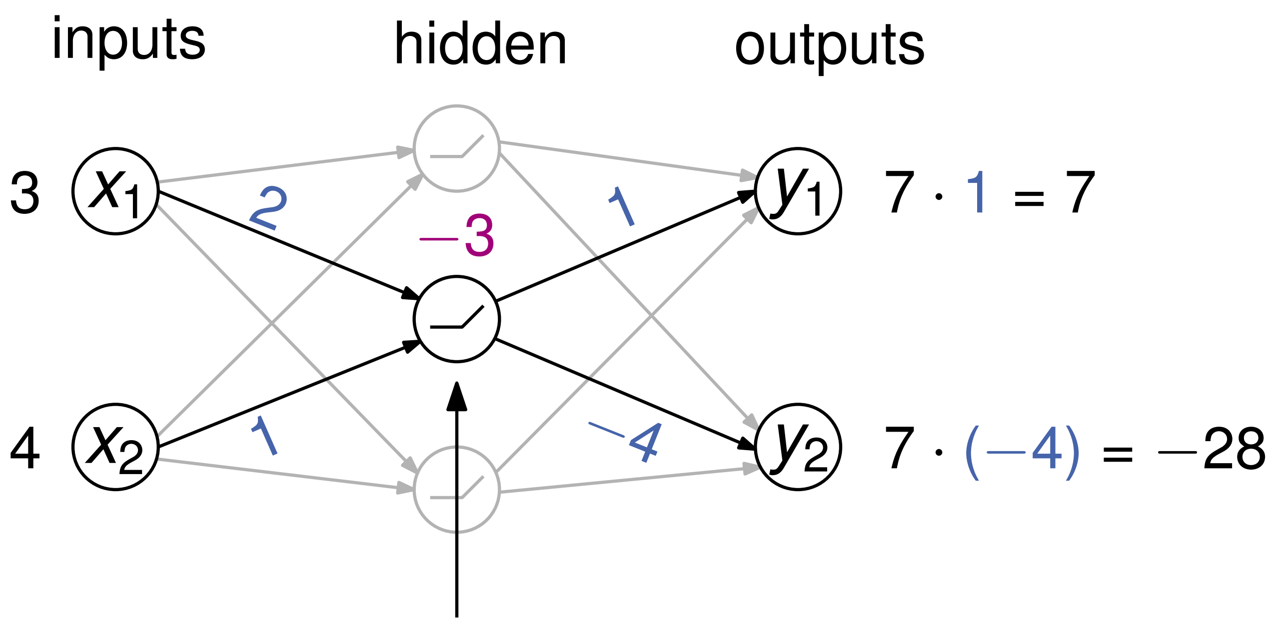
Biases: on hidden neurons

Activation Function: = ReLU

$$\mathrm{ReLU} : \mathbb{R} \to \mathbb{R}$$
$$x \mapsto \max\{0, x\}$$



Neural Networks



$\mathrm{ReLU}(3 \cdot 2 + 4 \cdot 1 + (-3)) = 7$

Neural network realizes a function:
$f(\cdot, \Theta) : \mathbb{R}^2 \to \mathbb{R}^2$
weights + biases parametrize f

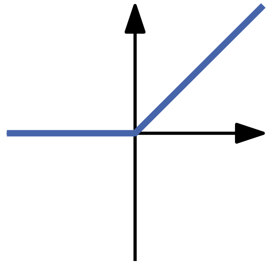
Architecture: directed acyclic graph
(vertices = neurons)

Weights: on edges

Biases: on hidden neurons

Activation Function: = ReLU

$\mathrm{ReLU} : \mathbb{R} \to \mathbb{R}$
$x \mapsto \max\{0, x\}$



Training Neural Networks

Question:

- The weights and biases Θ parametrize the function $f(\cdot, \Theta)$.
  $\rightsquigarrow$ What are *good* values for Θ?

Training Neural Networks

Question:

- The weights and biases Θ parametrize the function $f(\cdot, \Theta)$.
  $\rightsquigarrow$ What are *good* values for Θ?

Training Data:

List of points $(x_i; y_i) \in \mathbb{R}^2 \times \mathbb{R}^2$:

- $x_i \in \mathbb{R}^2$: input values
- $y_i \in \mathbb{R}^2$: *labels* = desired output values

Training Neural Networks

Question:

- The weights and biases Θ parametrize the function $f(\cdot, \Theta)$.
  $\rightsquigarrow$ What are *good* values for Θ?

Training Data:

List of points $(x_i; y_i) \in \mathbb{R}^2 \times \mathbb{R}^2$:

- $x_i \in \mathbb{R}^2$: input values
- $y_i \in \mathbb{R}^2$: *labels* = desired output values

Optimize: Choose Θ such that overall fitting error is minimal.
For all i: $\quad y_i \approx f(x_i, \Theta)$

Training Neural Networks

Question:

- The weights and biases Θ parametrize the function $f(\cdot, \Theta)$.
  $\rightsquigarrow$ What are *good* values for Θ?

Training Data:
List of points $(x_i; y_i) \in \mathbb{R}^2 \times \mathbb{R}^2$:

- $x_i \in \mathbb{R}^2$: input values
- $y_i \in \mathbb{R}^2$: *labels* = desired output values

Optimize: Choose Θ such that overall fitting error is minimal.
For all i: $\quad y_i \approx f(x_i, \Theta)$

Best case: $y_i = f(x_i, \Theta)$

Decision Problem

TRAIN-NN:

Input:

- network architecture
- n data points $(x_i; y_i)$

Question: Are there weights and biases Θ, such that:

$$y_i = f(x_i, \Theta) \quad \forall i \in \{1, \dots, n\}$$

Decision Problem

TRAIN-NN:

Input:

- network architecture
- n data points $(x_i; y_i)$

Question: Are there weights and biases Θ, such that:

$$y_i = f(x_i, \Theta) \quad \forall i \in \{1, \ldots, n\}$$

No optimization, just a yes/no-question.

Decision Problem

TRAIN-NN:

Input:

- network architecture
- n data points $(x_i; y_i)$

Question: Are there weights and biases Θ, such that:

$$y_i = f(x_i, \Theta) \quad \forall i \in \{1, \ldots, n\}$$

No optimization, just a yes/no-question.

A little more general:

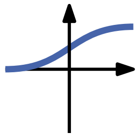
- a *cost function* $\mathrm{cost}(\cdot)$
- a *threshold* γ

$$\sum_{i=1}^{n} \mathrm{cost}(y_i, f(x_i, \Theta)) \leq \gamma?$$

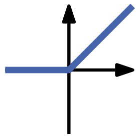
How hard can it be?

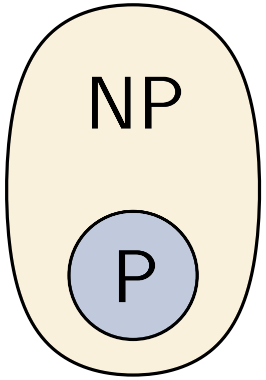
NP-**hard** in many settings:

- binary classification (Blum, Rivest 1992)
- sigmoid activation function (Jones 1997, . . .)



- single hidden neuron with ReLU (Geol et al. 2020)

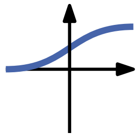




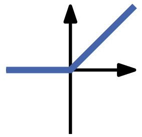
How hard can it be?

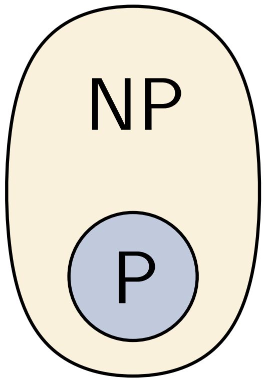
NP-**hard** in many settings:

- binary classification (Blum, Rivest 1992)
- sigmoid activation function (Jones 1997, . . .)



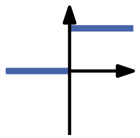
- single hidden neuron with ReLU (Geol et al. 2020)





NP-**membership** in simple settings:

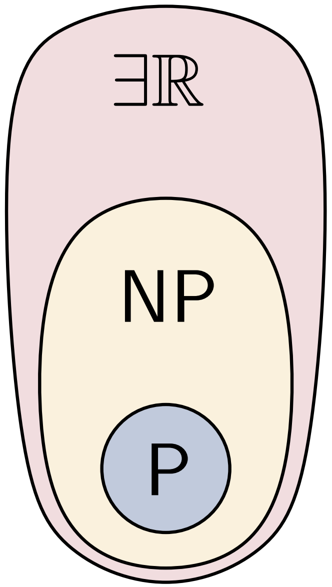
- single output neuron, one ReLU layer (Arora et al. 2016)
- step activation functions (Khalife, Basu 2022)



How hard can it be?

$\exists\mathbb{R}$-complete for:

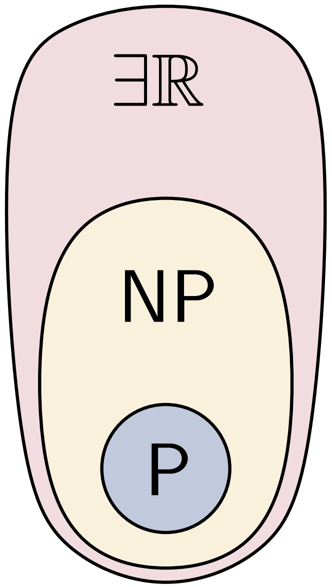
- one hidden layer, three outputs, identity activation function (Abrahamsen, Kleist, Miltzow 2021)



How hard can it be?

$\exists\mathbb{R}$-complete for:

- one hidden layer, three outputs, identity activation function (Abrahamsen, Kleist, Miltzow 2021)



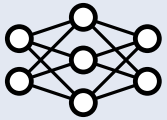
Their proof relies on *particularly difficult* to train network architectures.
$\rightsquigarrow$ This is not a practical setting.

Our Result

Theorem: Training neural networks is **$\exists\mathbb{R}$-complete**, for

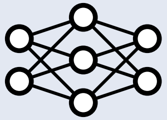
Our Result

Theorem: Training neural networks is **$\exists\mathbb{R}$-complete**, for

- exactly one hidden layer, 

Our Result

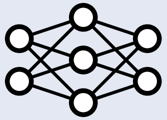
Theorem: Training neural networks is **$\exists\mathbb{R}$-complete**, for

- exactly one hidden layer, 
- two inputs, two outputs,

in NP for single output

Our Result

Theorem: Training neural networks is $\exists\mathbb{R}$**-complete**, for

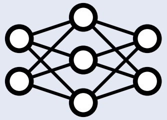
- exactly one hidden layer, 
- two inputs, two outputs,
- fully connected network architecture,

in NP for single output

often used (as a building block) in practical architectures

Our Result

Theorem: Training neural networks is **∃ℝ-complete**, for

- exactly one hidden layer, 
- two inputs, two outputs,
- fully connected network architecture,
- only 13 different labels,

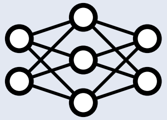
in NP for single output

often used (as a building block) in practical architectures

common in classification tasks

Our Result

Theorem: Training neural networks is **$\exists\mathbb{R}$-complete**, for

- exactly one hidden layer, 
- two inputs, two outputs,
- fully connected network architecture,
- only 13 different labels,
- (more or less) any training error γ,

in NP for single output

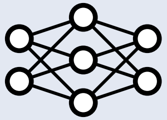
often used (as a building block) in practical architectures

common in classification tasks

We prove γ = 0. Add inconsistent training data for $\gamma > 0$.

Our Result

Theorem: Training neural networks is **$\exists\mathbb{R}$-complete**, for

- exactly one hidden layer, 
- two inputs, two outputs,
- fully connected network architecture,
- only 13 different labels,
- (more or less) any training error γ,
- ReLU activation function,

in NP for single output

often used (as a building block) in practical architectures

common in classification tasks

We prove $\gamma = 0$. Add inconsistent training data for $\gamma > 0$.

by far the most used in practice

Existential Theory of the Reals

Definition: (ETR)
EXISTENTIAL THEORY OF THE REALS:
All true sentences of the form

$$\exists X_1, \ldots, X_n \in \mathbb{R} : \varphi(X_1, \ldots, X_n).$$

φ = quantifier-free formula of
polynomial equations and inequalities

Existential Theory of the Reals

Definition: (ETR)
EXISTENTIAL THEORY OF THE REALS:
All true sentences of the form

$$\exists X_1, \ldots, X_n \in \mathbb{R} : \varphi(X_1, \ldots, X_n).$$

φ = quantifier-free formula of
polynomial equations and inequalities

↑ Solving systems of non-linear equations and inequalities.

Existential Theory of the Reals

Definition: (ETR)
EXISTENTIAL THEORY OF THE REALS:
All true sentences of the form

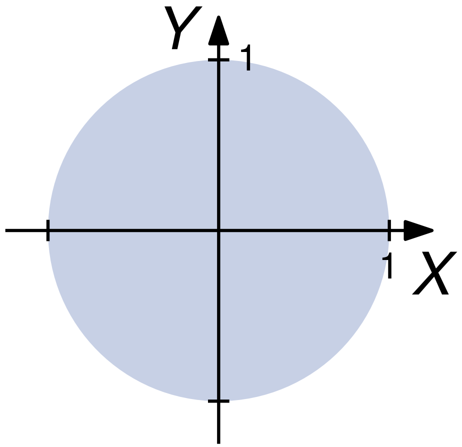
$$\exists X_1, \ldots, X_n \in \mathbb{R} : \varphi(X_1, \ldots, X_n).$$

φ = quantifier-free formula of
polynomial equations and inequalities

Solving systems of non-linear equations and inequalities.

Example:

$\varphi(X, Y) :\equiv X^2 + Y^2 \leq 1$



Existential Theory of the Reals

Definition: (ETR)
EXISTENTIAL THEORY OF THE REALS:
All true sentences of the form

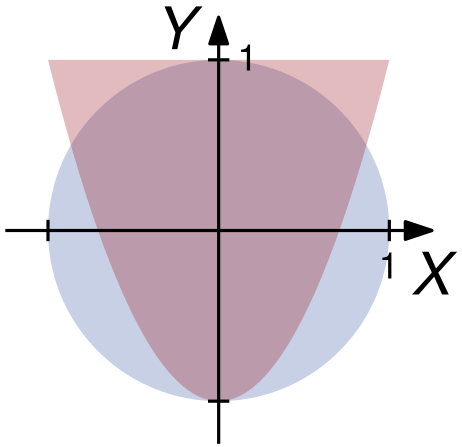
$$\exists X_1, \ldots, X_n \in \mathbb{R} : \varphi(X_1, \ldots, X_n).$$

φ = quantifier-free formula of
polynomial equations and inequalities

Solving systems of non-linear equations and inequalities.

Example:

$$\varphi(X, Y) :\equiv X^2 + Y^2 \leq 1 \quad \wedge \quad Y \geq 2X^2 - 1$$



Existential Theory of the Reals

Definition: (ETR)

EXISTENTIAL THEORY OF THE REALS:

All true sentences of the form

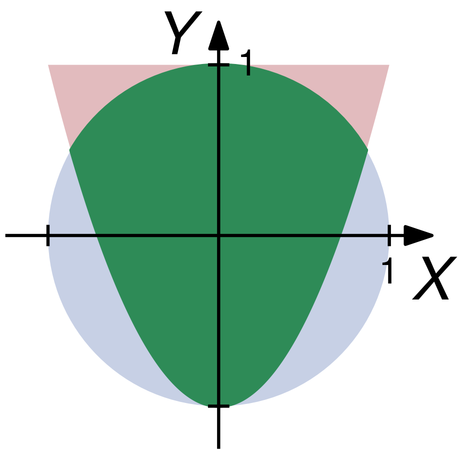
$$\exists X_1, \ldots, X_n \in \mathbb{R} : \varphi(X_1, \ldots, X_n).$$

φ = quantifier-free formula of polynomial equations and inequalities

Solving systems of non-linear equations and inequalities.

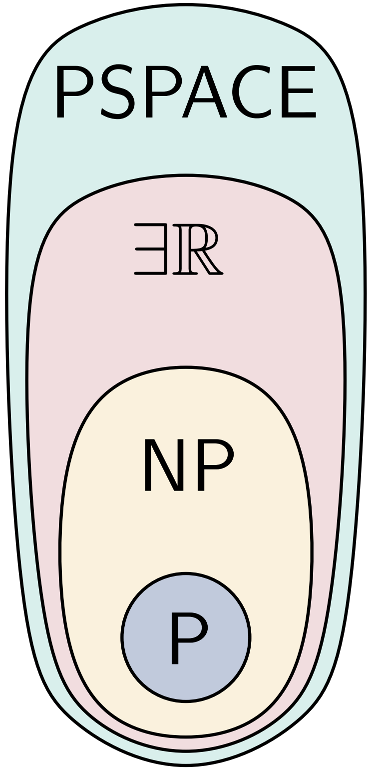
Example:

$$\varphi(X, Y) :\equiv X^2 + Y^2 \leq 1 \\ \wedge \quad Y \geq 2X^2 - 1$$



$\exists X, Y \in \mathbb{R} : \varphi(X, Y)$ is true

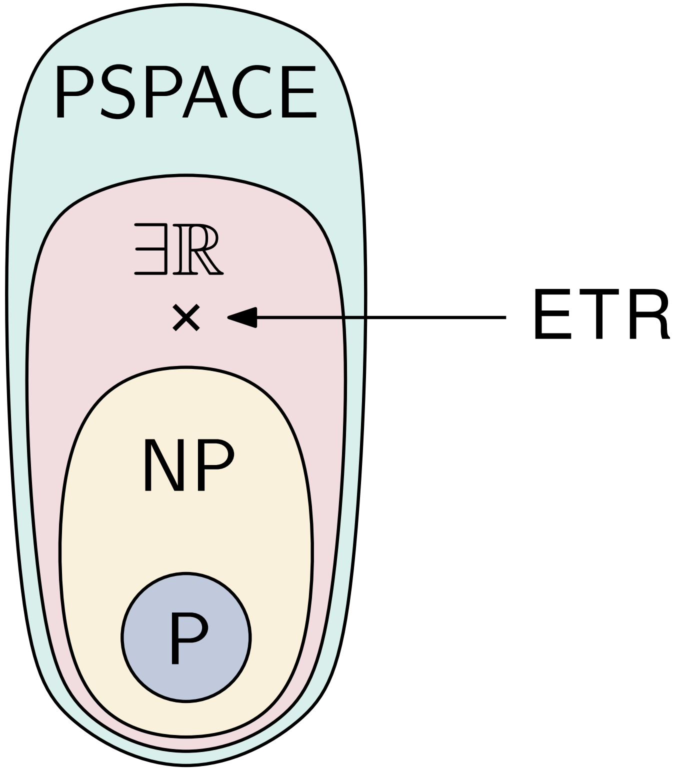
Complexity Class $\exists\mathbb{R}$



Base Problem: ETR

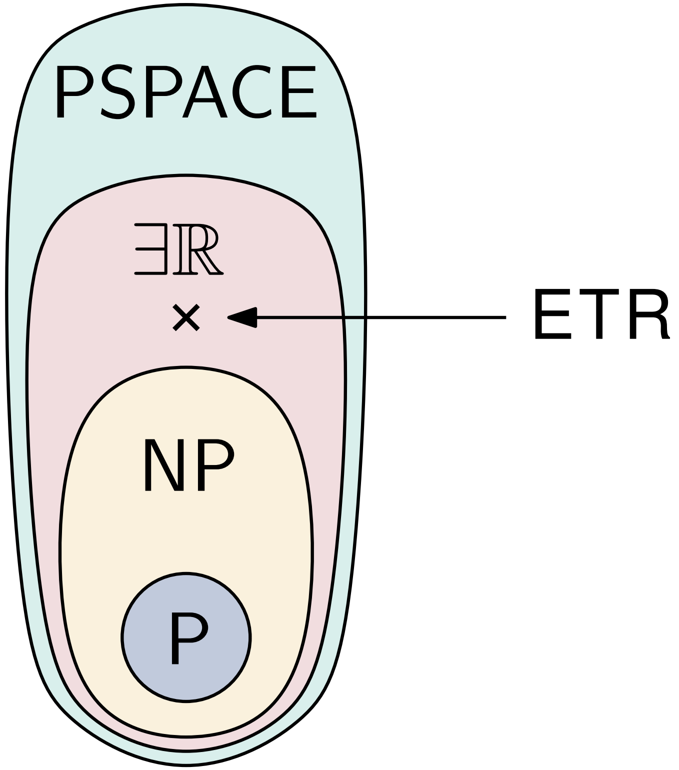
Decide whether $\exists X \in \mathbb{R}^n : \varphi(X)$ is true.

Complexity Class $\exists\mathbb{R}$



Base Problem: ETR
Decide whether $\exists X \in \mathbb{R}^n : \varphi(X)$ is true.

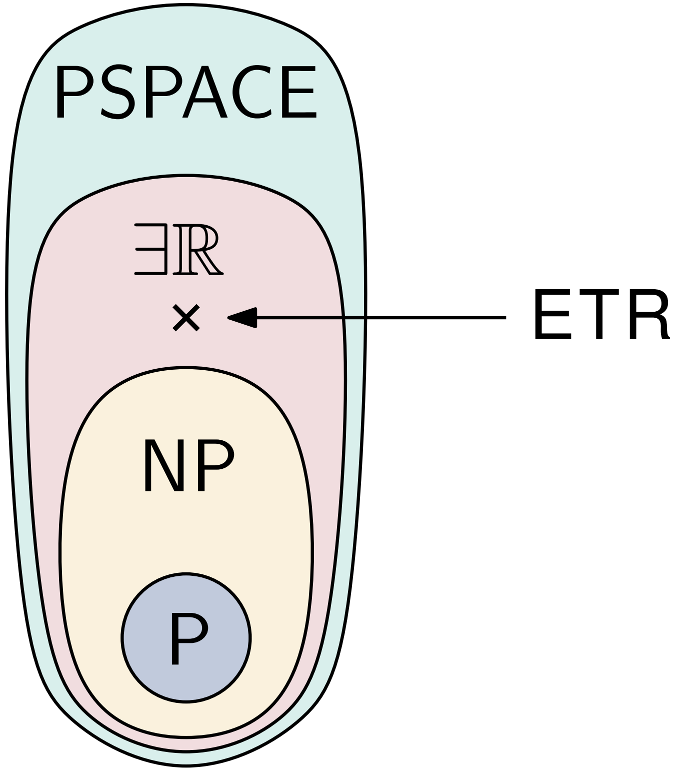
Complexity Class $\exists\mathbb{R}$



Base Problem: ETR
Decide whether $\exists X \in \mathbb{R}^n : \varphi(X)$ is true.

The **complexity class $\exists\mathbb{R}$** contains all problems that reduce to ETR.

Complexity Class $\exists\mathbb{R}$

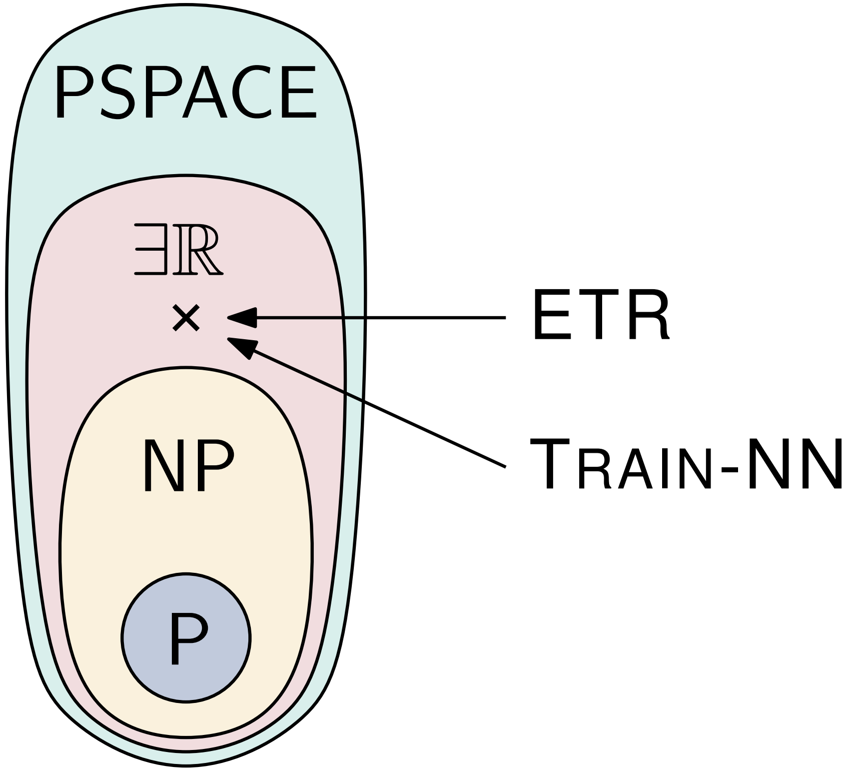


Base Problem: ETR
Decide whether $\exists X \in \mathbb{R}^n : \varphi(X)$ is true.

The **complexity class $\exists\mathbb{R}$** contains all problems that reduce to ETR.

$\exists\mathbb{R}$-complete $\Leftrightarrow$ equivalent to ETR
(under polynomial time transformations)

Complexity Class $\exists\mathbb{R}$

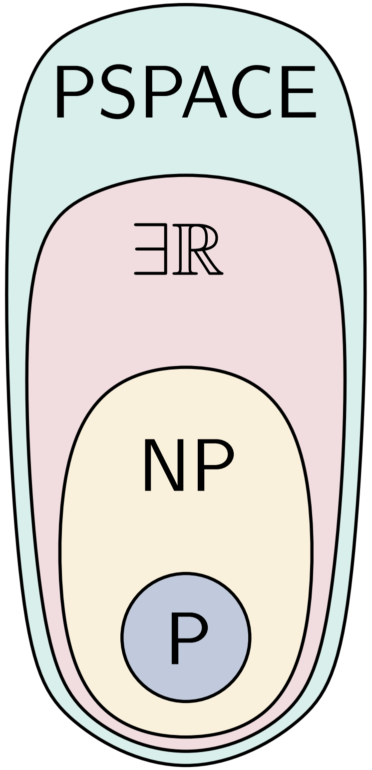


Base Problem: ETR
Decide whether $\exists X \in \mathbb{R}^n : \varphi(X)$ is true.

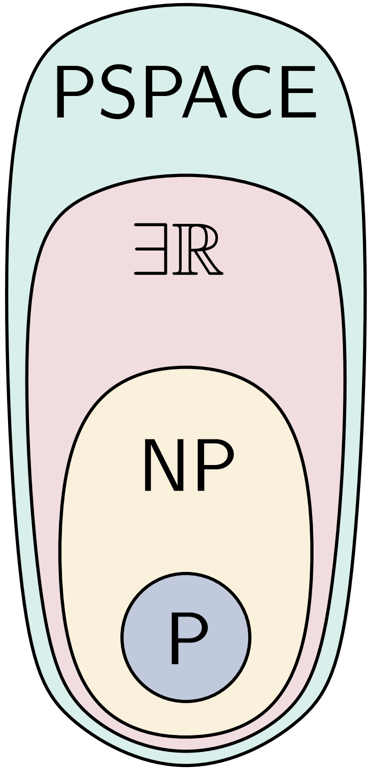
The **complexity class** $\exists\mathbb{R}$ contains all problems that reduce to ETR.

$\exists\mathbb{R}$-complete $\Leftrightarrow$ equivalent to ETR
(under polynomial time transformations)

Practical Implications



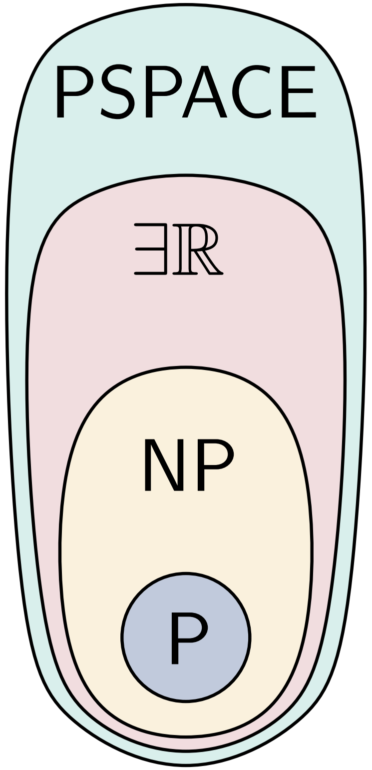
Practical Implications



Problems **in** P:

- Efficient algorithms in theory and practice.

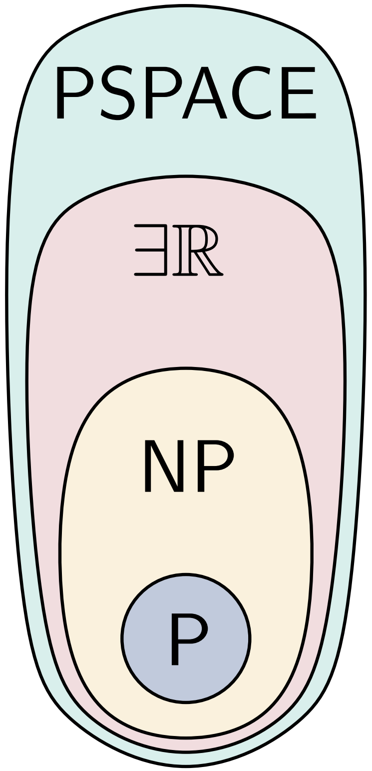
Practical Implications



NP-**complete** problems:

- No efficient algorithms in theory. (assuming NP $\neq$ P)
- Highly optimized off-the-shelf tools can solve large instance to optimality.

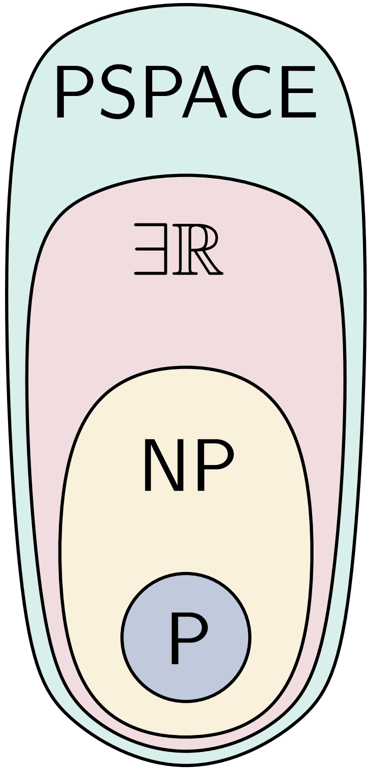
Practical Implications



∃ℝ-complete problems:

- Exponential time algorithms in theory.
  However, useless in practice.
- Gradient descent often works reasonably well.
  But: No guarantees on time and quality.

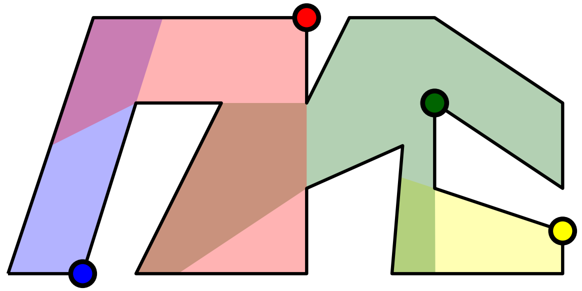
Practical Implications



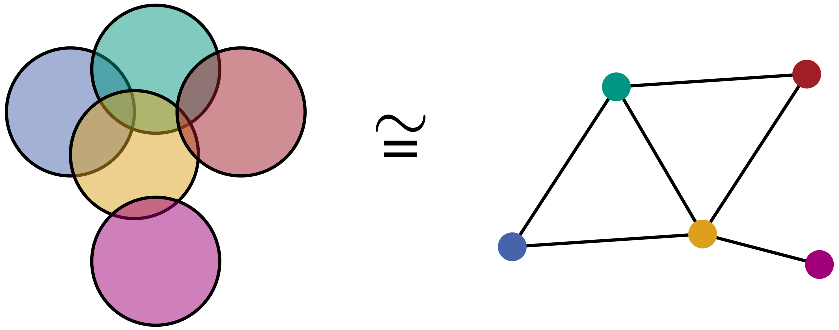
PSPACE-**complete** problems:

- No general purpose tools.
- P = NP = ∃ℝ = PSPACE is possible, but considered unlikely.

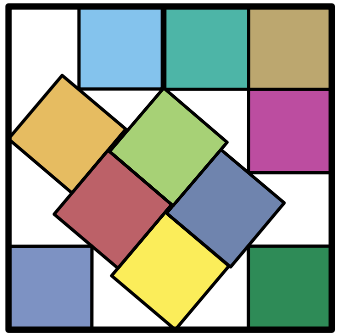
∃ℝ-Complete Problems



Art Gallery Problem



Recognition of Unit Disk Graphs



Packing

... and many more **geometric** problems

$\exists\mathbb{R}$-Membership

$\rightsquigarrow$ TRAIN-NN is at most as difficult as ETR

Goal: Express TRAIN-NN as an ETR formula.

∃ℝ-Membership

$\rightsquigarrow$ TRAIN-NN is at most as difficult as ETR

Goal: Express TRAIN-NN as an ETR formula.

$$\exists \underbrace{w_1, \ldots,}_{\text{weights}} \quad \underbrace{b_1, \ldots}_{\text{biases}} \in \mathbb{R} \; : \; \underbrace{y_1 = f(x_1, \Theta) \quad \wedge \quad \ldots \quad \wedge \quad y_n = f(x_n, \Theta)}_{\text{formula checking that training data is fit exactly}}$$

$\exists\mathbb{R}$-Hardness

$\rightsquigarrow$ TRAIN-NN is at lest as difficult as ETR

Express ETR formula as an instance of TRAIN-NN.

Step 1: Simplify formula.

ETR $\rightsquigarrow$ ETR-NN

Step 2: ETR-NN $\rightsquigarrow$ TRAIN-NN

$\exists\mathbb{R}$-Hardness

$\rightsquigarrow$ TRAIN-NN is at lest as difficult as ETR

Express ETR formula as an instance of TRAIN-NN.

Step 1: Simplify formula.

ETR $\rightsquigarrow$ ETR-NN

- Values: $\exists X, \ldots \in [-1, 1] : \varphi(X)$
- Constraints:

$$X + Y = Z$$

$$XY + X + Y = 0 \qquad \text{(nonlinear)}$$

$$X \geq 0$$

$$X = 1$$

Step 2: ETR-NN $\rightsquigarrow$ TRAIN-NN

$\exists\mathbb{R}$-Hardness

$\rightsquigarrow$ TRAIN-NN is at lest as difficult as ETR

Express ETR formula as an instance of TRAIN-NN.

Step 1: Simplify formula.
ETR $\rightsquigarrow$ ETR-NN

- Values: $\exists X, \ldots \in [-1, 1] : \varphi(X)$
- Constraints:

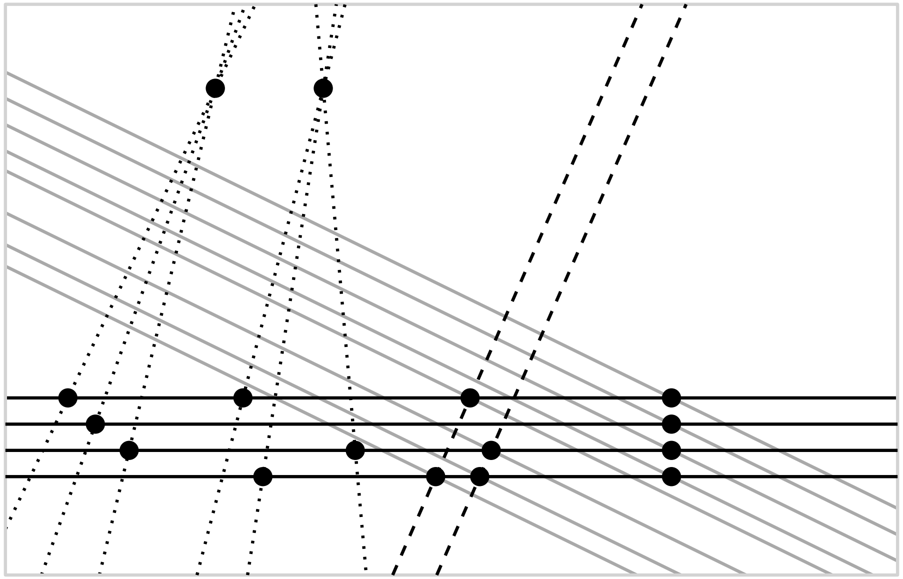
$$X + Y = Z$$

$$XY + X + Y = 0 \qquad \text{(nonlinear)}$$

$$X \geq 0$$

$$X = 1$$

Step 2: ETR-NN $\rightsquigarrow$ TRAIN-NN



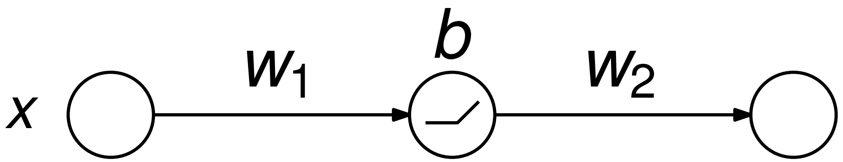
geometric construction

Geometry I

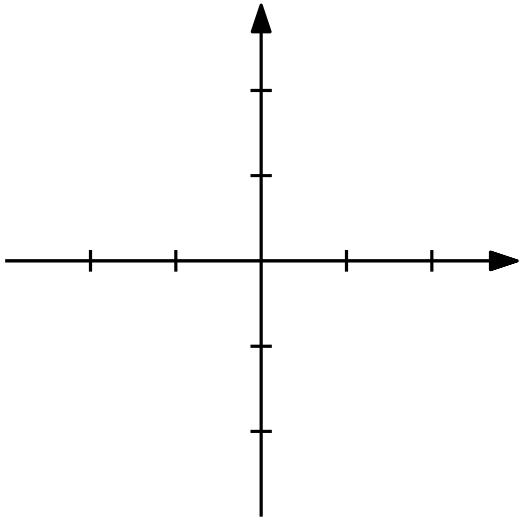
Recall: Neural network realizes a function $f(\cdot, \Theta)$.
How does is look like?

Geometry I

Recall: Neural network realizes a function $f(\cdot, \Theta)$.
How does is look like?



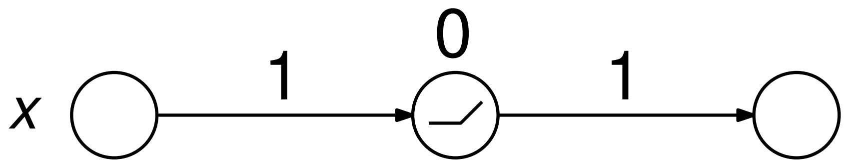
$\rightsquigarrow$



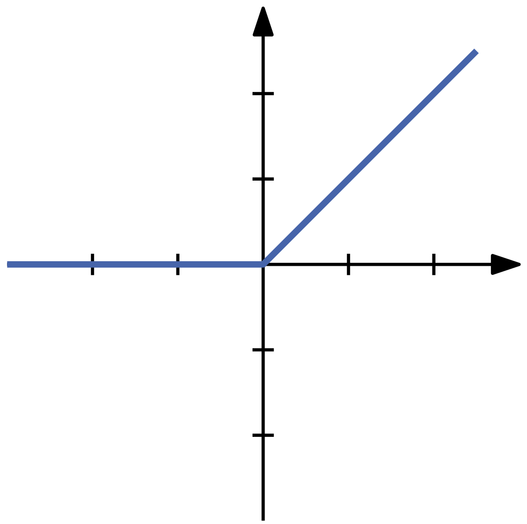
$$f(\cdot, \Theta) : \mathbb{R} \to \mathbb{R}$$
$$x \mapsto \mathrm{ReLU}(w_1 x + b) \cdot w_2$$

Geometry I

Recall: Neural network realizes a function $f(\cdot, \Theta)$.
How does is look like?



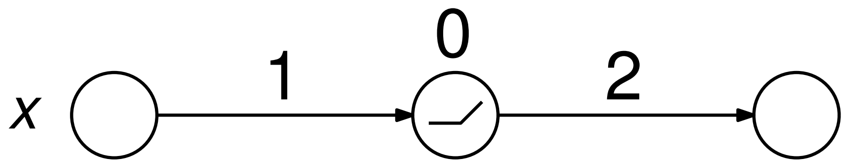
$\rightsquigarrow$



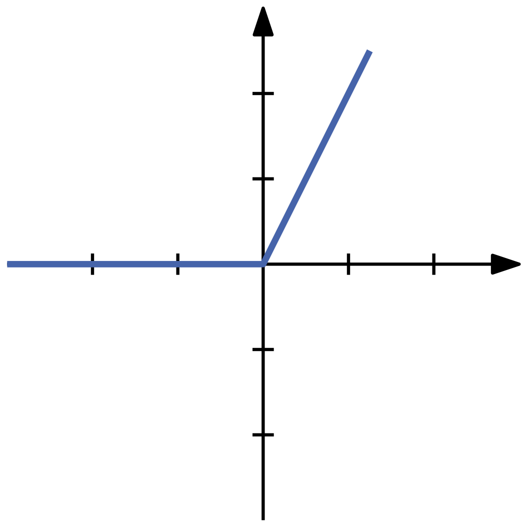
$$f(\cdot, \Theta) : \mathbb{R} \to \mathbb{R}$$
$$x \mapsto \mathrm{ReLU}(1x + 0) \cdot 1$$

Geometry I

Recall: Neural network realizes a function $f(\cdot, \Theta)$.
How does is look like?



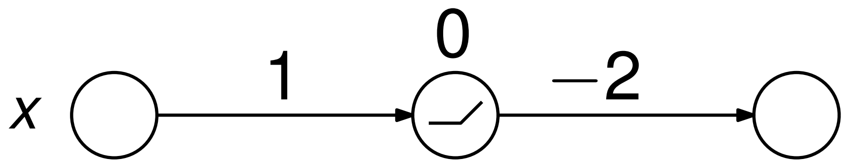
$\rightsquigarrow$



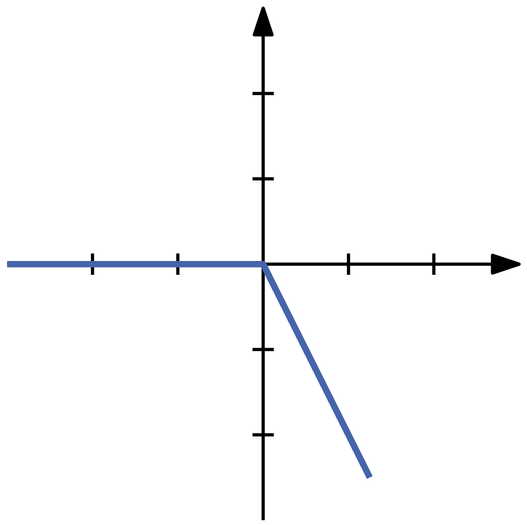
$$f(\cdot, \Theta) : \mathbb{R} \to \mathbb{R}$$
$$x \mapsto \mathrm{ReLU}(1x + 0) \cdot 2$$

Geometry I

Recall: Neural network realizes a function $f(\cdot, \Theta)$.
How does is look like?



$\rightsquigarrow$

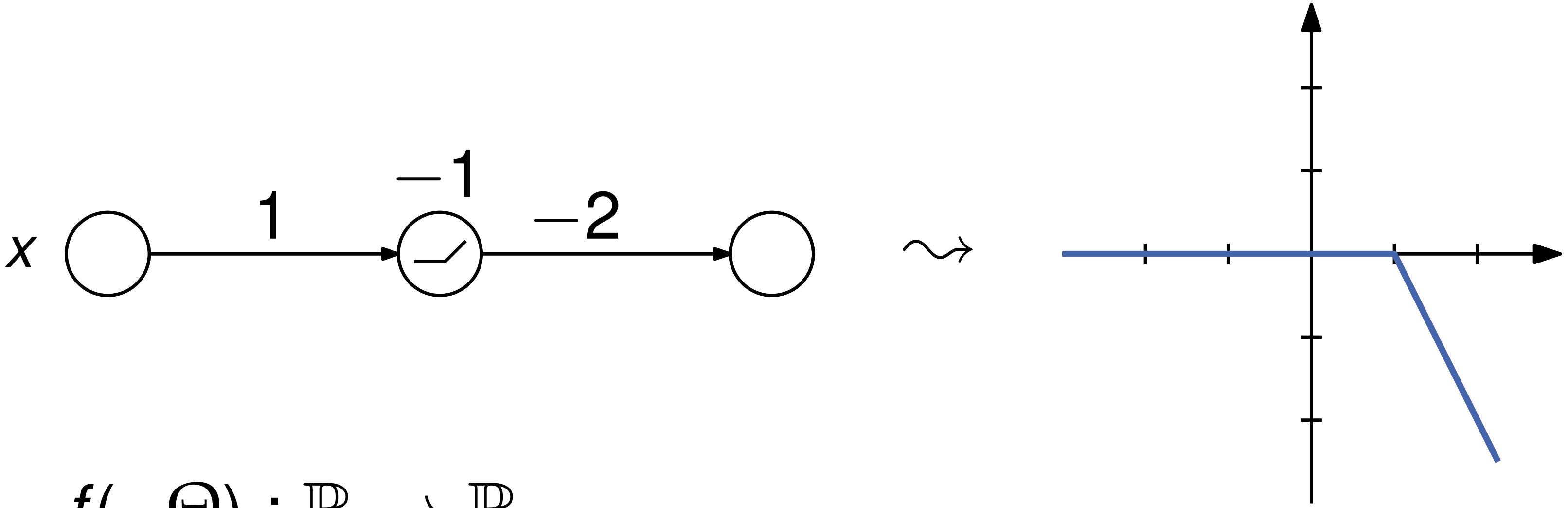


$$f(\cdot, \Theta) : \mathbb{R} \to \mathbb{R}$$
$$x \mapsto \mathrm{ReLU}(1x + 0) \cdot (-2)$$

Training Fully Connected Neural Networks is ∃ℝ-Complete
Daniel Bertschinger, Christoph Hertrich, **Paul Jungeblut**, Tillmann Miltzow, Simon Weber

Geometry I

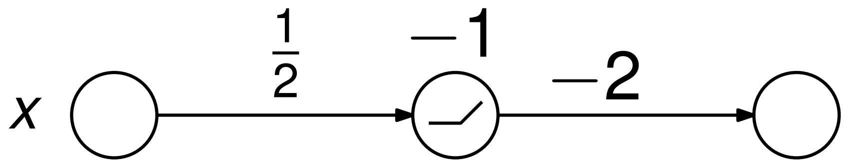
Recall: Neural network realizes a function $f(\cdot, \Theta)$.
How does is look like?



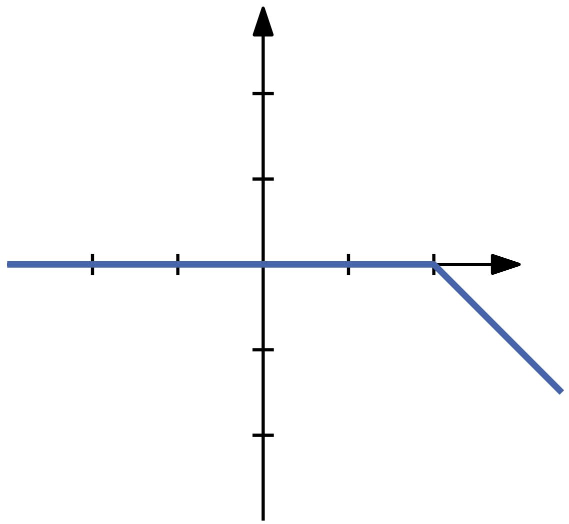
$$f(\cdot, \Theta) : \mathbb{R} \to \mathbb{R}$$
$$x \mapsto \mathrm{ReLU}(1x - 1) \cdot (-2)$$

Geometry I

Recall: Neural network realizes a function $f(\cdot, \Theta)$.
How does is look like?



$\rightsquigarrow$

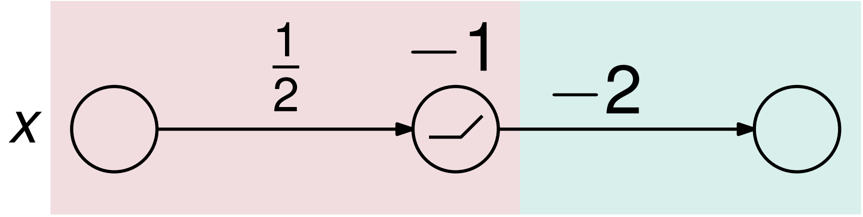


$$f(\cdot, \Theta) : \mathbb{R} \to \mathbb{R}$$
$$x \mapsto \mathrm{ReLU}(\tfrac{1}{2}x - 1) \cdot (-2)$$

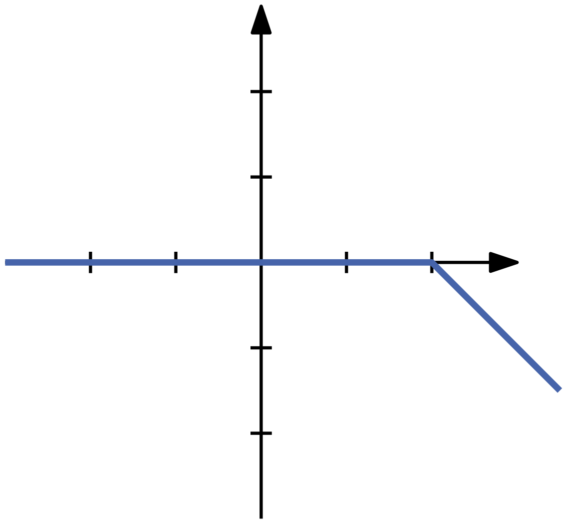
Geometry I

Recall: Neural network realizes a function $f(\cdot, \Theta)$.
How does is look like?

$f(\cdot, \Theta)$ is *continuous* and *piecewise linear*.



$\rightsquigarrow$



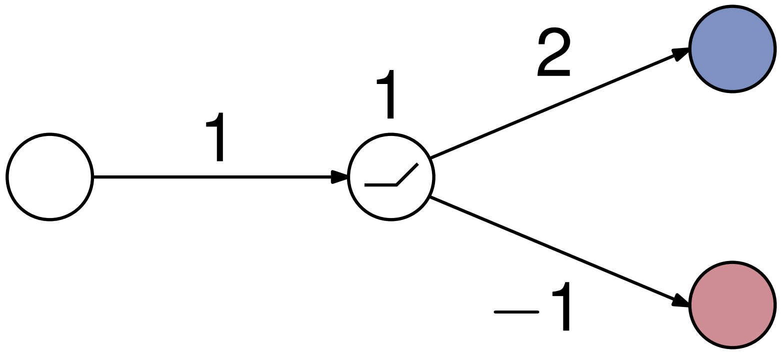
Breakpoint is determined only by first weight and bias.

$$f(\cdot, \Theta) : \mathbb{R} \to \mathbb{R}$$
$$x \mapsto \text{ReLU}(\tfrac{1}{2}x - 1) \cdot (-2)$$

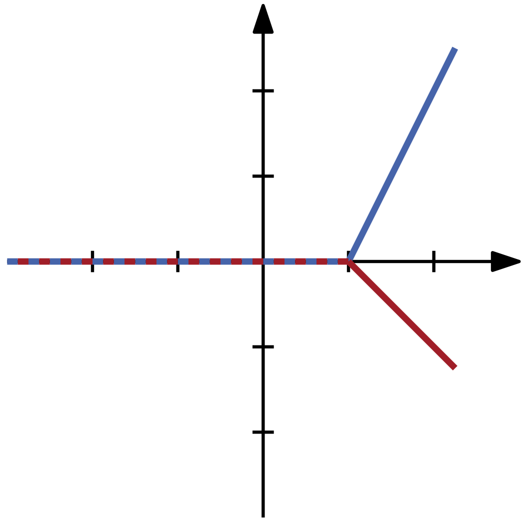
Second weight only for scaling.

Geometry II

Question: Two outputs?

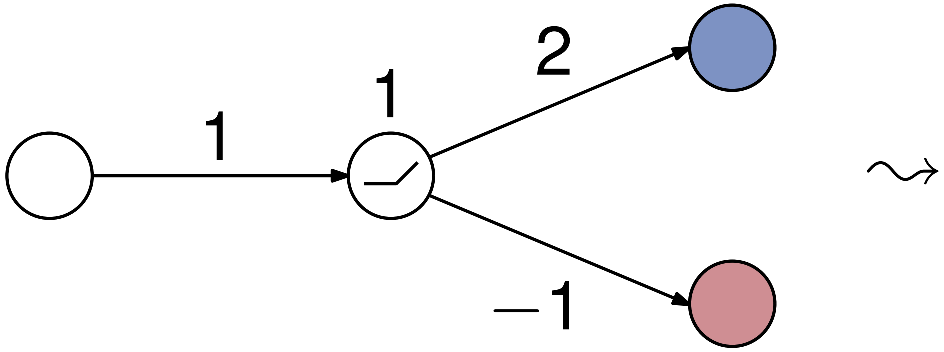


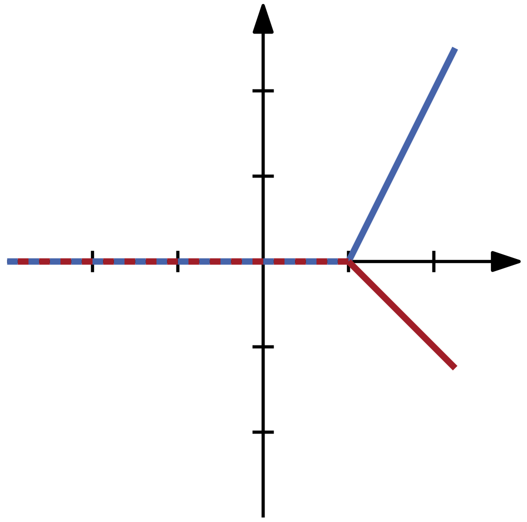
$\rightsquigarrow$



Geometry II

Question: Two outputs?



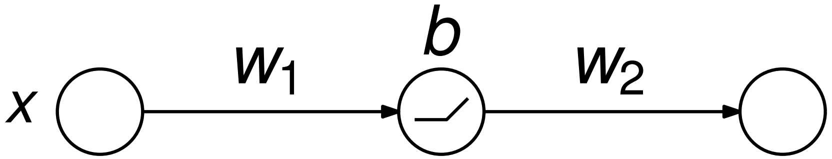


Separate functions, one per output.

All functions have the same breakpoint!

Geometry III

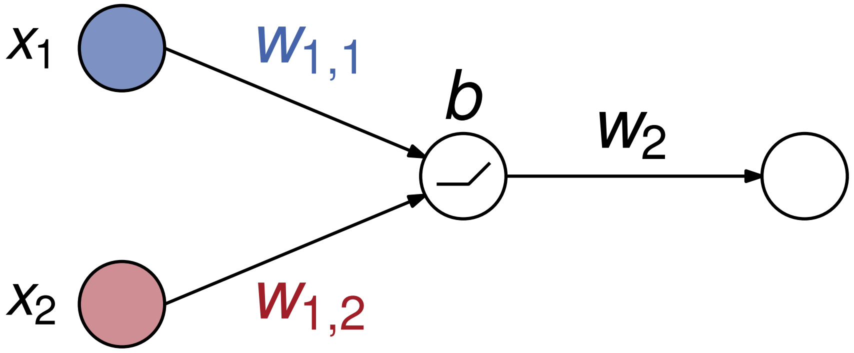
Question: Two inputs?



$$f(\cdot, \Theta) : \mathbb{R} \to \mathbb{R}$$

$$x \mapsto \text{ReLU}(w_1 x + b) \cdot w_2$$

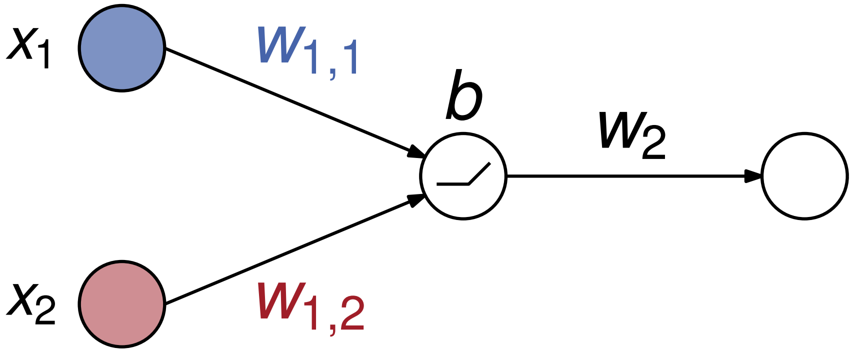
Geometry III



$$f(\cdot, \Theta) : \mathbb{R}^2 \to \mathbb{R}$$

$$x \mapsto \mathrm{ReLU}(w_{1,1} x_1 + w_{1,2} x_2 + b) \cdot w_2$$

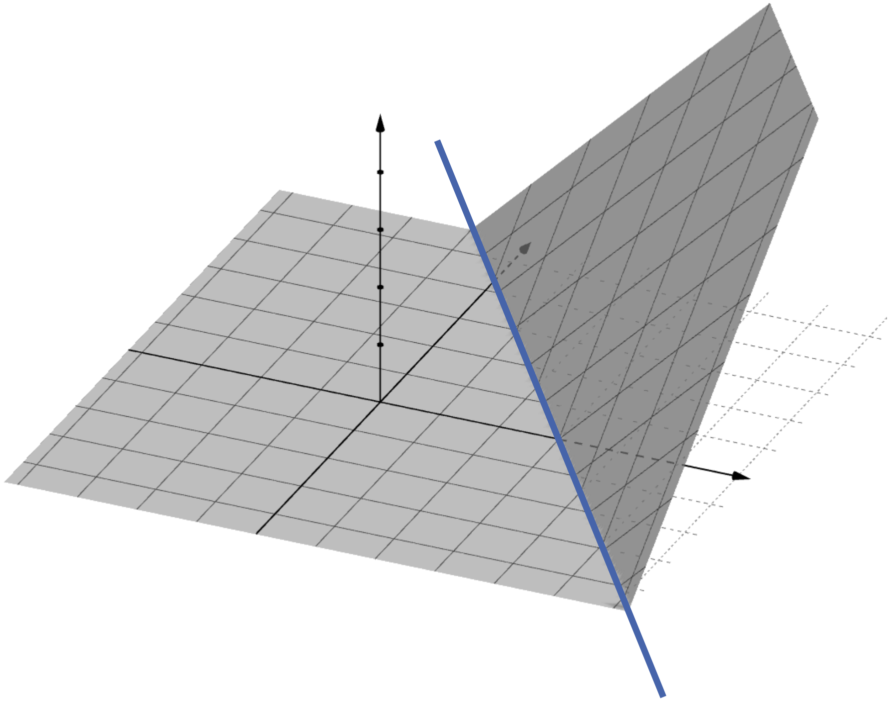
Geometry III



$$f(\cdot, \Theta) : \mathbb{R}^2 \to \mathbb{R}$$

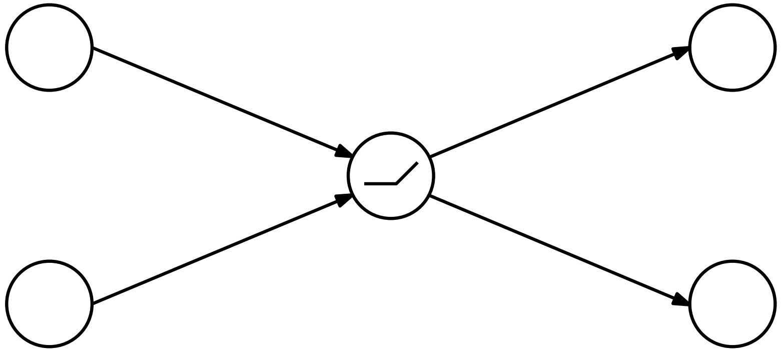
$$x \mapsto \mathrm{ReLU}(w_{1,1}x_1 + w_{1,2}x_2 + b) \cdot w_2$$

breakpoint $\rightsquigarrow$ breakline



$$w_{1,1}x_1 + w_{1,2}x_2 + b = 0$$

Geometry IV

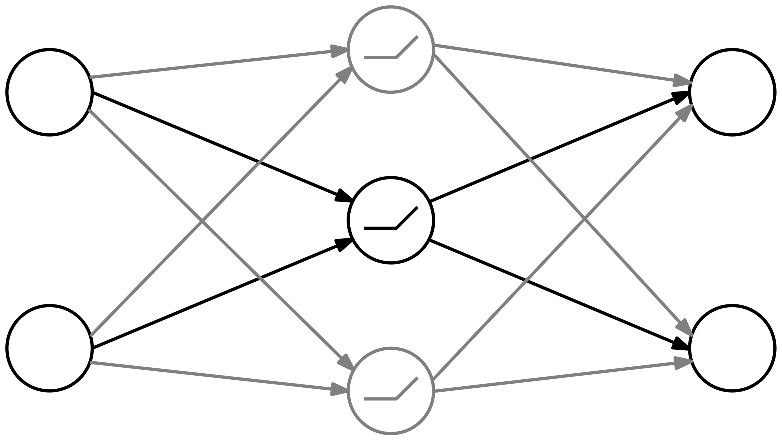


$f_1(\cdot, \Theta) : \mathbb{R}^2 \to \mathbb{R}$

$f_2(\cdot, \Theta) : \mathbb{R}^2 \to \mathbb{R}$

$$f(\cdot, \Theta) : \mathbb{R}^2 \to \mathbb{R}^2$$
$$x \mapsto (f_1(x, \Theta), f_2(x, \Theta))$$

Geometry IV



$f_1(\cdot, \Theta) : \mathbb{R}^2 \to \mathbb{R}$

$f_2(\cdot, \Theta) : \mathbb{R}^2 \to \mathbb{R}$

$$f(\cdot, \Theta) : \mathbb{R}^2 \to \mathbb{R}^2$$
$$x \mapsto (f_1(x, \Theta), f_2(x, \Theta))$$

More hidden neurons:

- Each ReLU neuron contributes exactly one breakline.
- $f(\cdot, \Theta)$ is the sum of all individual continuous piecewise linear functions.
- Same breaklines in f_1 and f_2.

Encoding ETR as a Neural Network

Goal: ETR-NN $\rightsquigarrow$ TRAIN-NN

Encoding ETR as a Neural Network

Goal: ETR-NN $\rightsquigarrow$ TRAIN-NN

Given: variables
constraints

Find: data points
integer m

Such that: formula true $\iff$ trainable with m ReLUs

Daniel Bertschinger, Christoph Hertrich, **Paul Jungeblut**, Tillmann Miltzow, Simon Weber

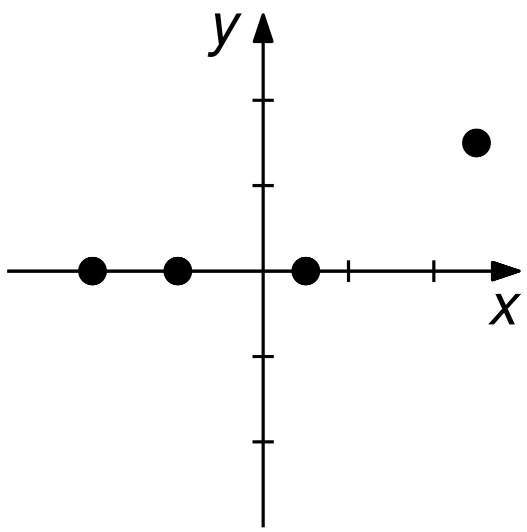
Encoding ETR as a Neural Network

Goal: ETR-NN $\rightsquigarrow$ TRAIN-NN

Given: variables
constraints

Find: data points
integer m

Such that: formula true $\iff$ trainable with m ReLUs



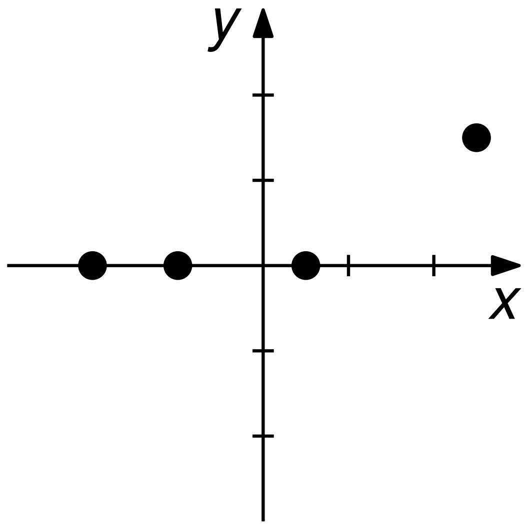
Encoding ETR as a Neural Network

Goal: ETR-NN $\rightsquigarrow$ TRAIN-NN

Given: variables
constraints

Find: data points
integer m

Such that: formula true $\iff$ trainable with m ReLUs



not collinear $\rightsquigarrow$ at least one ReLU

Recall: #ReLUs = #breakpoints

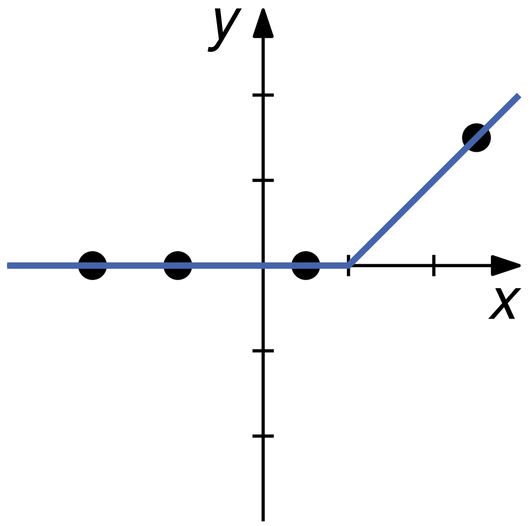
Encoding ETR as a Neural Network

Goal: ETR-NN $\rightsquigarrow$ Train-NN

Given: variables
constraints

Find: data points
integer m

Such that: formula true $\iff$ trainable with m ReLUs



not collinear $\rightsquigarrow$ at least one ReLU

Possible with 1 ReLU.

Recall: #ReLUs = #breakpoints

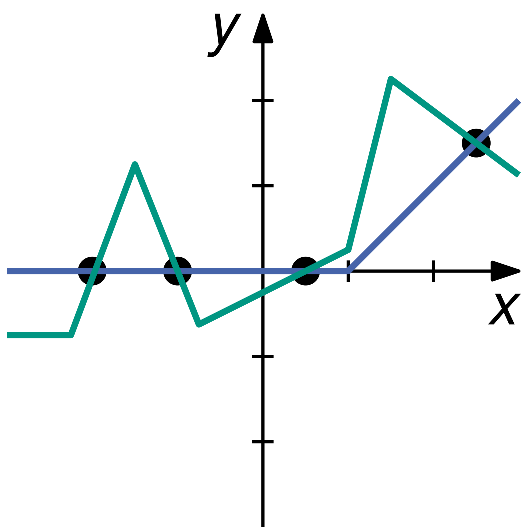
Encoding ETR as a Neural Network

Goal: ETR-NN $\rightsquigarrow$ TRAIN-NN

Given: variables
constraints

Find: data points
integer m

Such that: formula true $\iff$ trainable with m ReLUs



not collinear $\rightsquigarrow$ at least one ReLU

Possible with 1 ReLU.

Possible with more ReLUs.

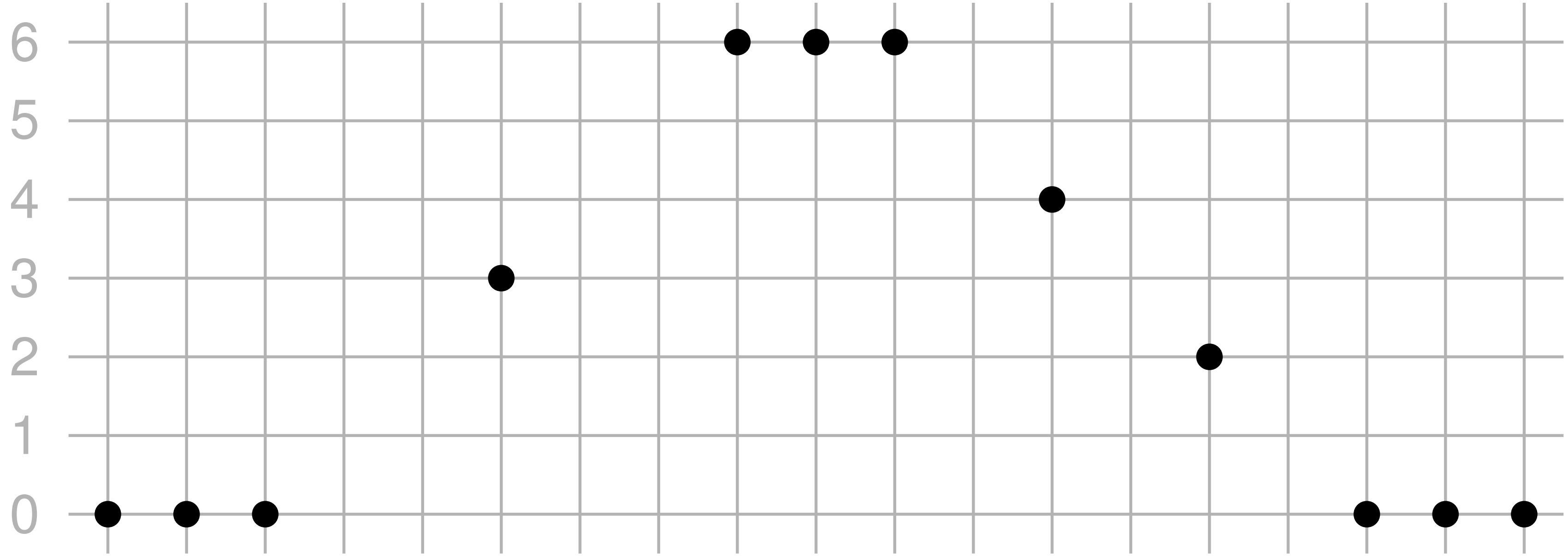
Recall: #ReLUs = #breakpoints

Encoding Variables

Task: Encode a value $X \in [-1, 1]$.

Encoding Variables

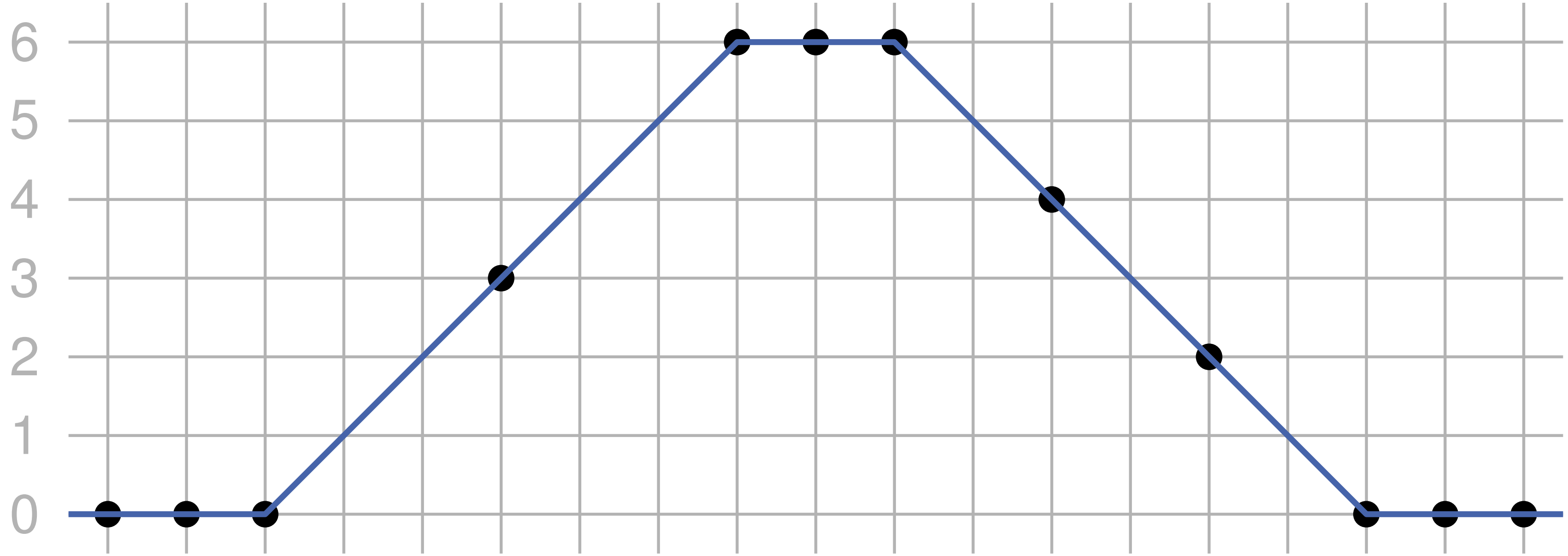
Task: Encode a value $X \in [-1, 1]$.



Fit with 4 ReLUs:
$\rightsquigarrow$ 4 breakpoints

Encoding Variables

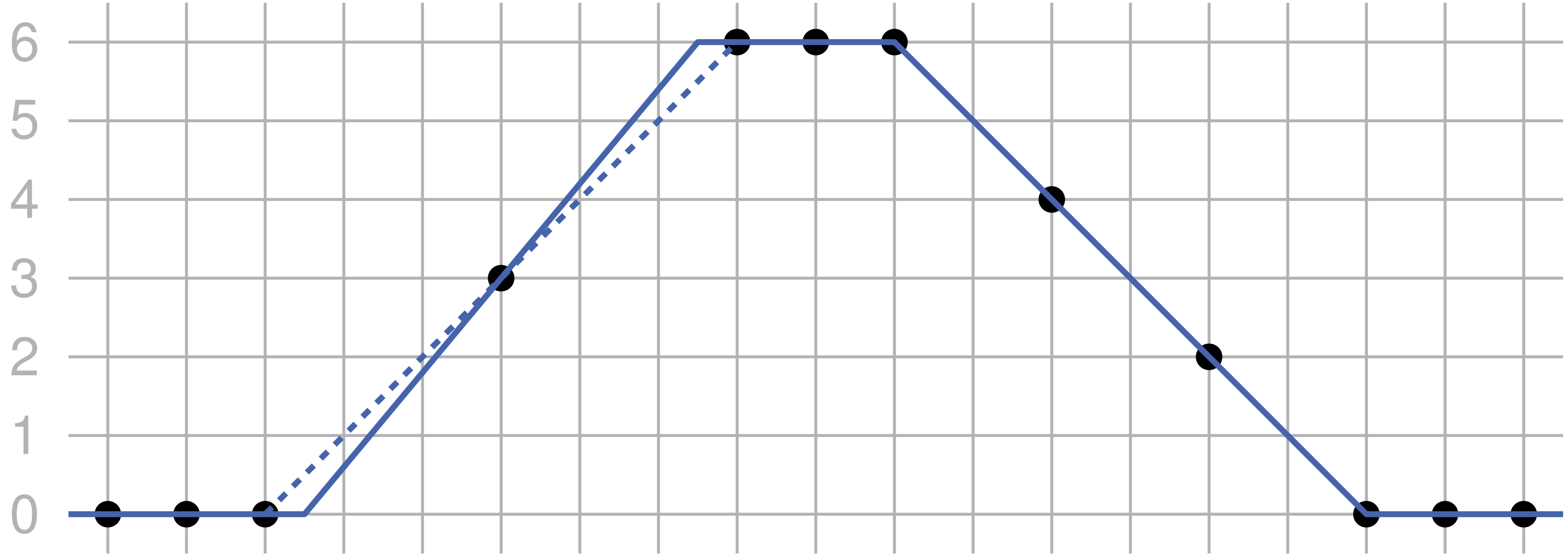
Task: Encode a value $X \in [-1, 1]$.



Fit with 4 ReLUs:
$\rightsquigarrow$ 4 breakpoints

Encoding Variables

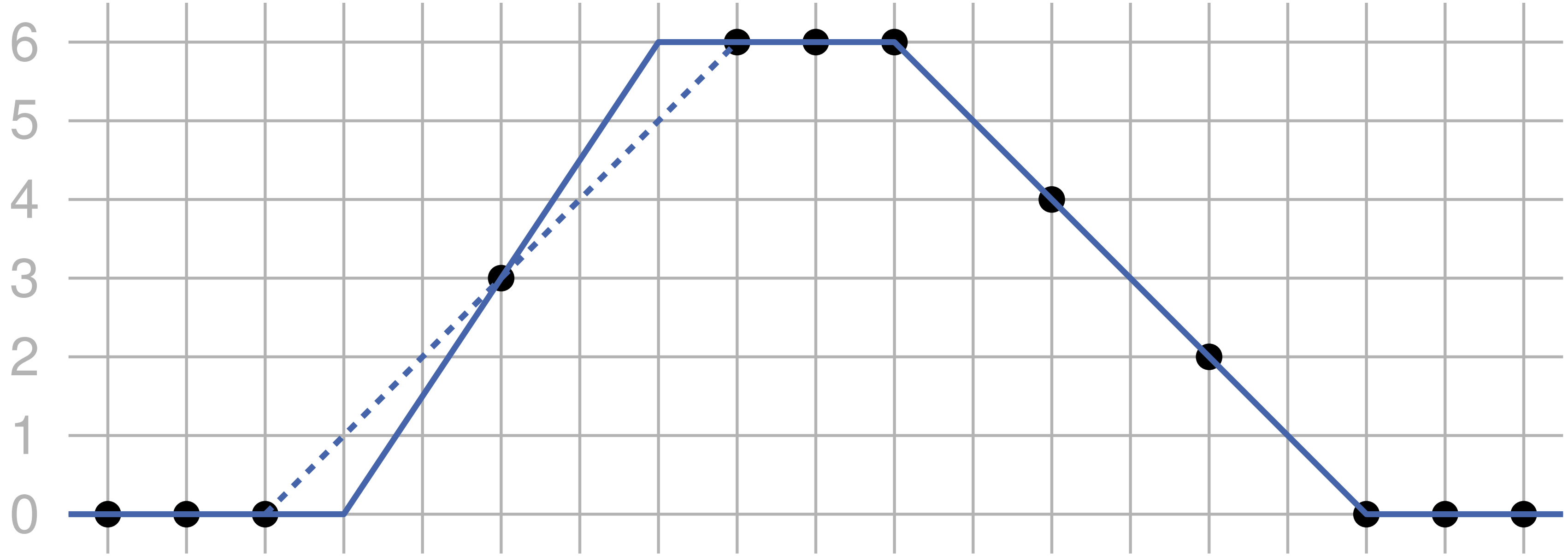
Task: Encode a value $X \in [-1, 1]$.



Fit with 4 ReLUs:
$\rightsquigarrow$ 4 breakpoints

Encoding Variables

Task: Encode a value $X \in [-1, 1]$.

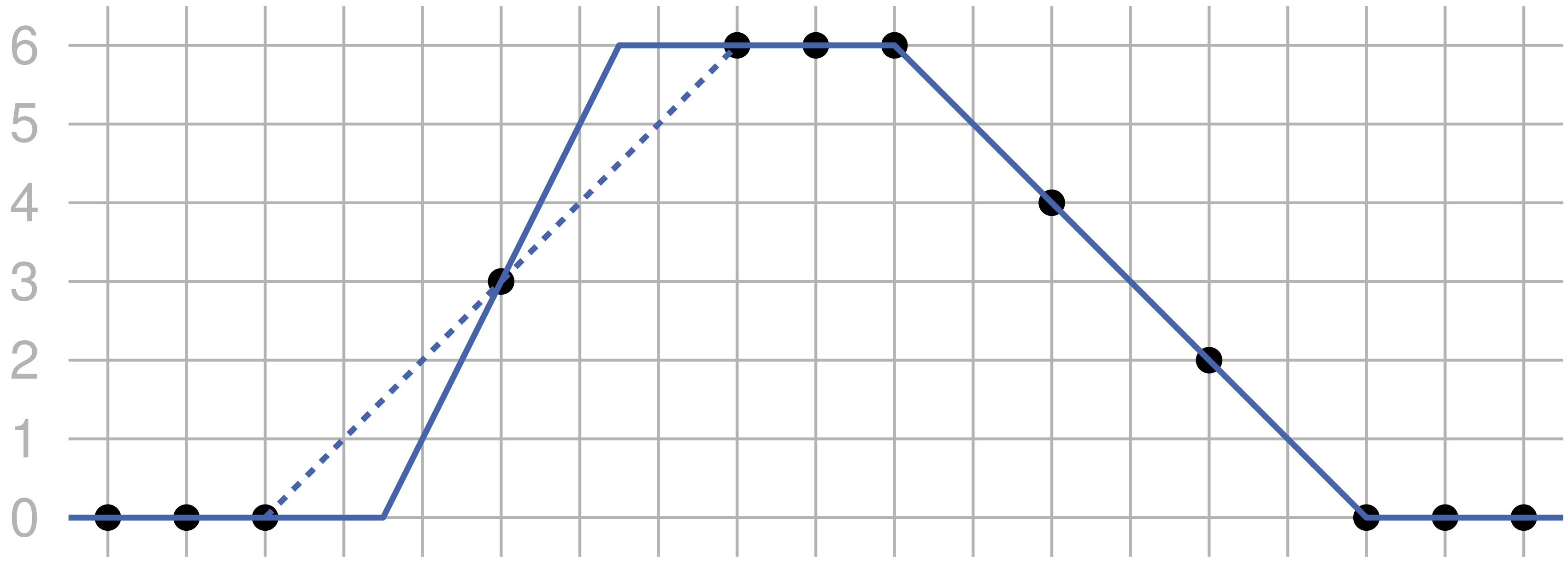


Fit with 4 ReLUs:
$\rightsquigarrow$ 4 breakpoints

Daniel Bertschinger, Christoph Hertrich, **Paul Jungeblut**, Tillmann Miltzow, Simon Weber

Encoding Variables

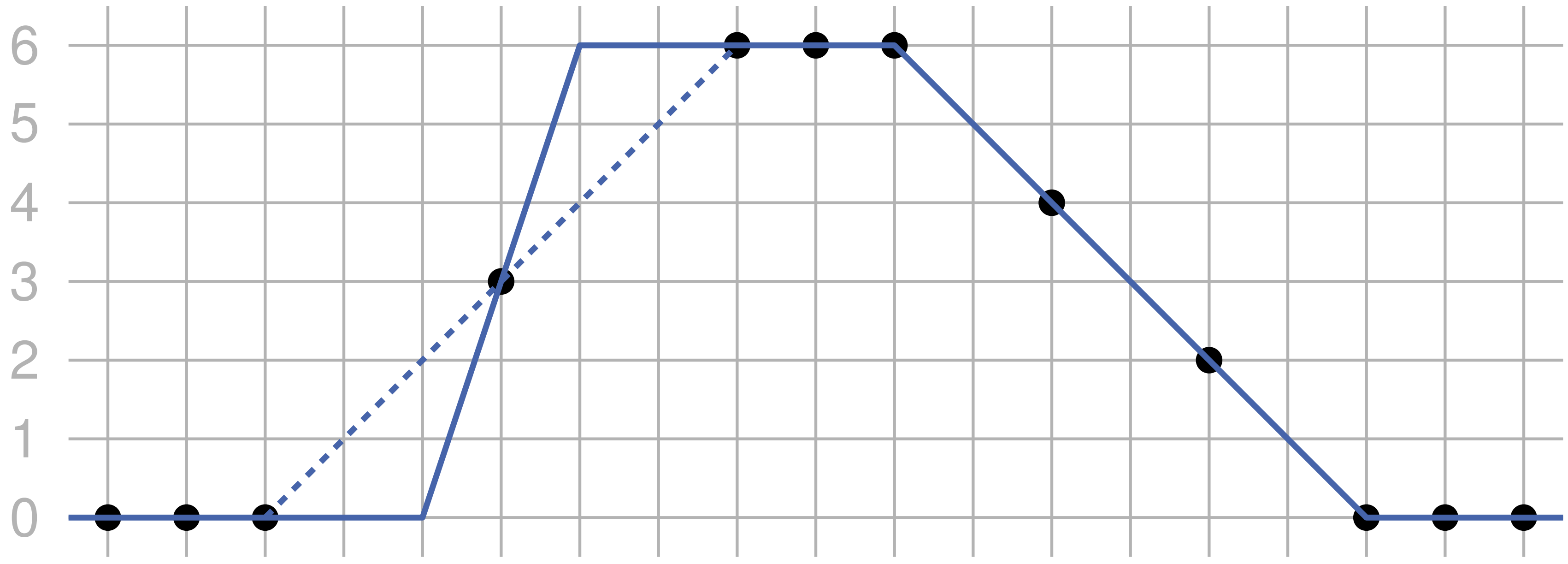
Task: Encode a value $X \in [-1, 1]$.



Fit with 4 ReLUs:
$\rightsquigarrow$ 4 breakpoints

Encoding Variables

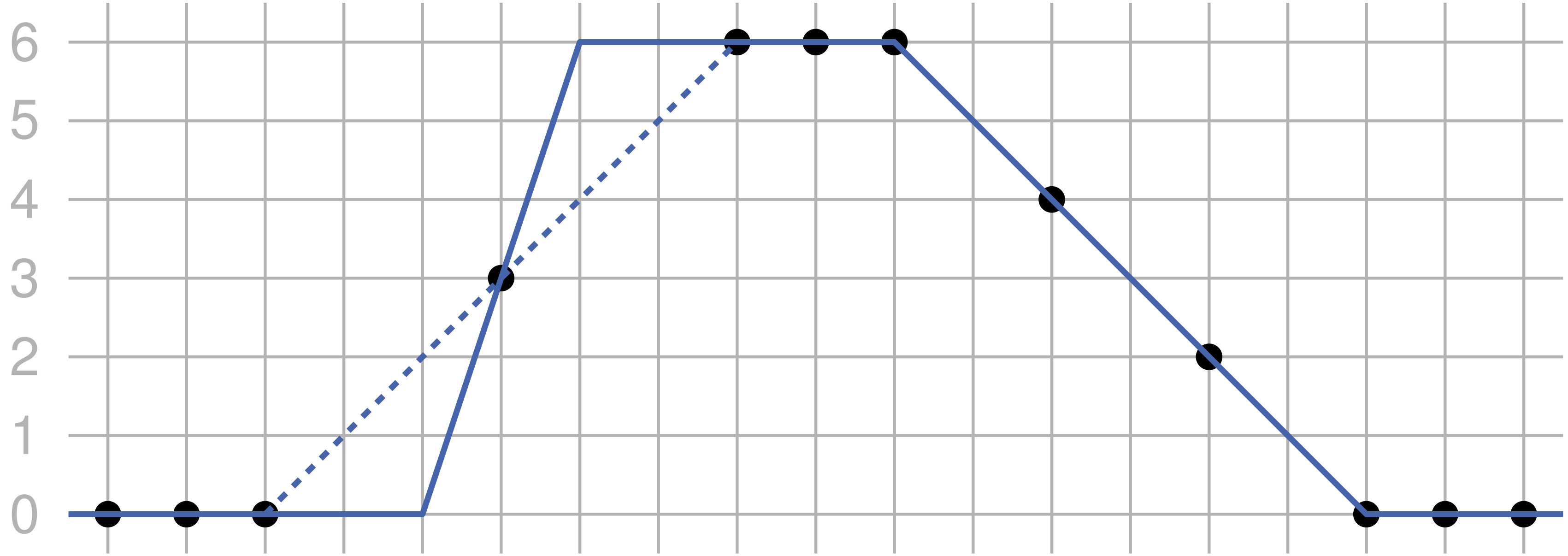
Task: Encode a value $X \in [-1, 1]$.



Fit with 4 ReLUs:
$\rightsquigarrow$ 4 breakpoints

Encoding Variables

Task: Encode a value $X \in [-1, 1]$.



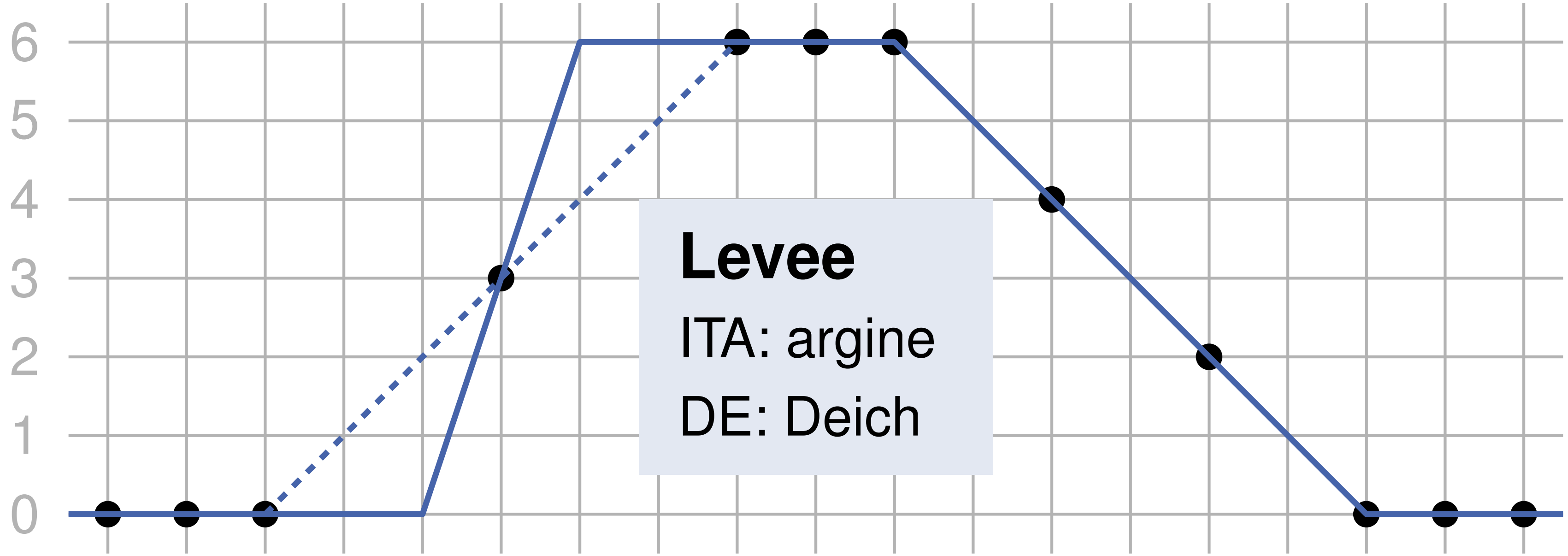
Fit with 4 ReLUs:
$\rightsquigarrow$ 4 breakpoints

Idea: The slope encodes the value.

Minimum slope is 1, we enforce a maximum slope of 3:
$\rightsquigarrow$ Interpret slopes in $[1, 3]$ as values in $[-1, 1]$.

Encoding Variables

Task: Encode a value $X \in [-1, 1]$.



Fit with 4 ReLUs:
$\rightsquigarrow$ 4 breakpoints

Idea: The slope encodes the value.

Minimum slope is 1, we enforce a maximum slope of 3:
$\rightsquigarrow$ Interpret slopes in $[1, 3]$ as values in $[-1, 1]$.

Linear Constraints

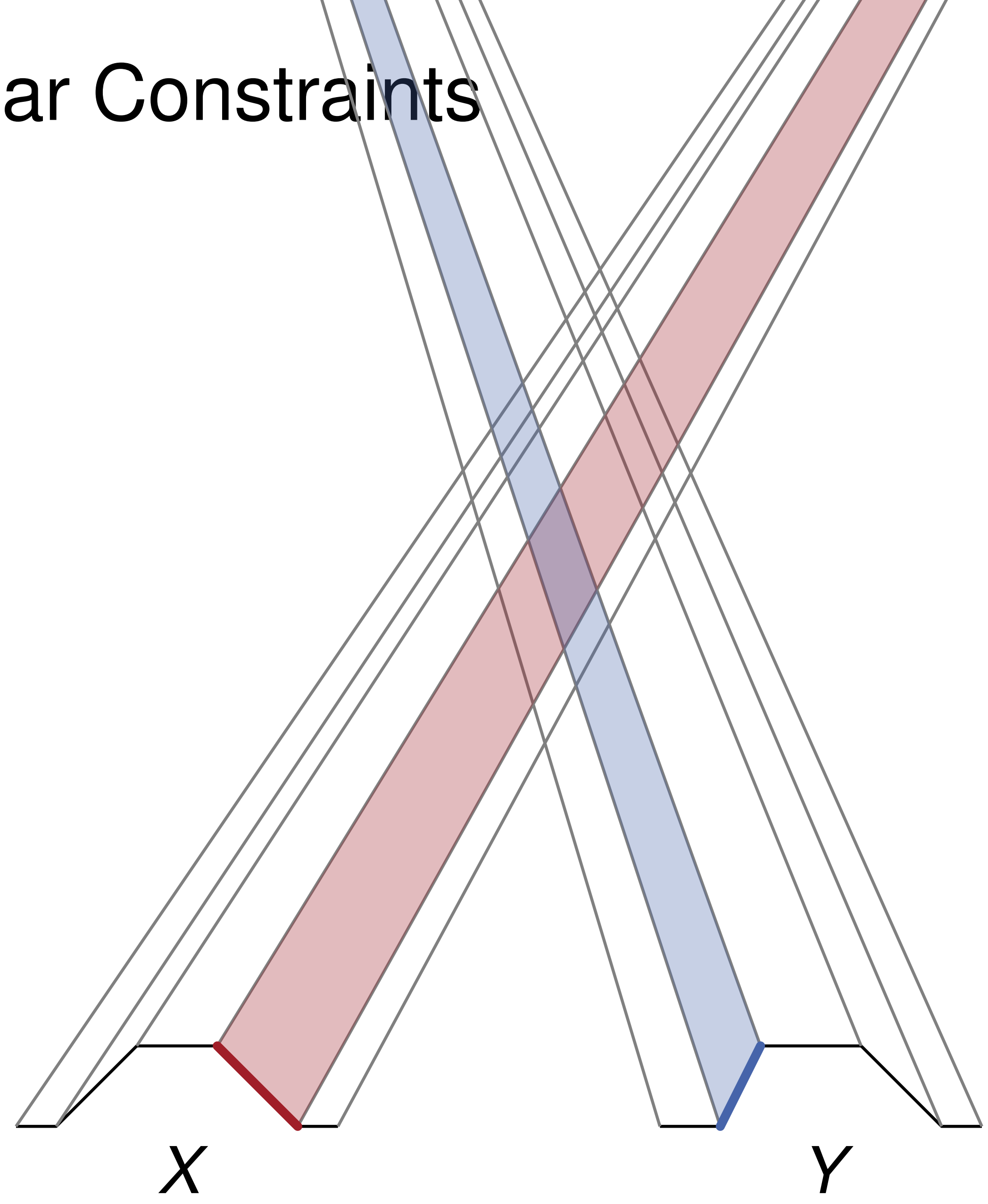
Question: How to encode constraints involving X and Y?

Linear Constraints

Question: How to encode constraints involving X and Y?

- impossible in one dimension

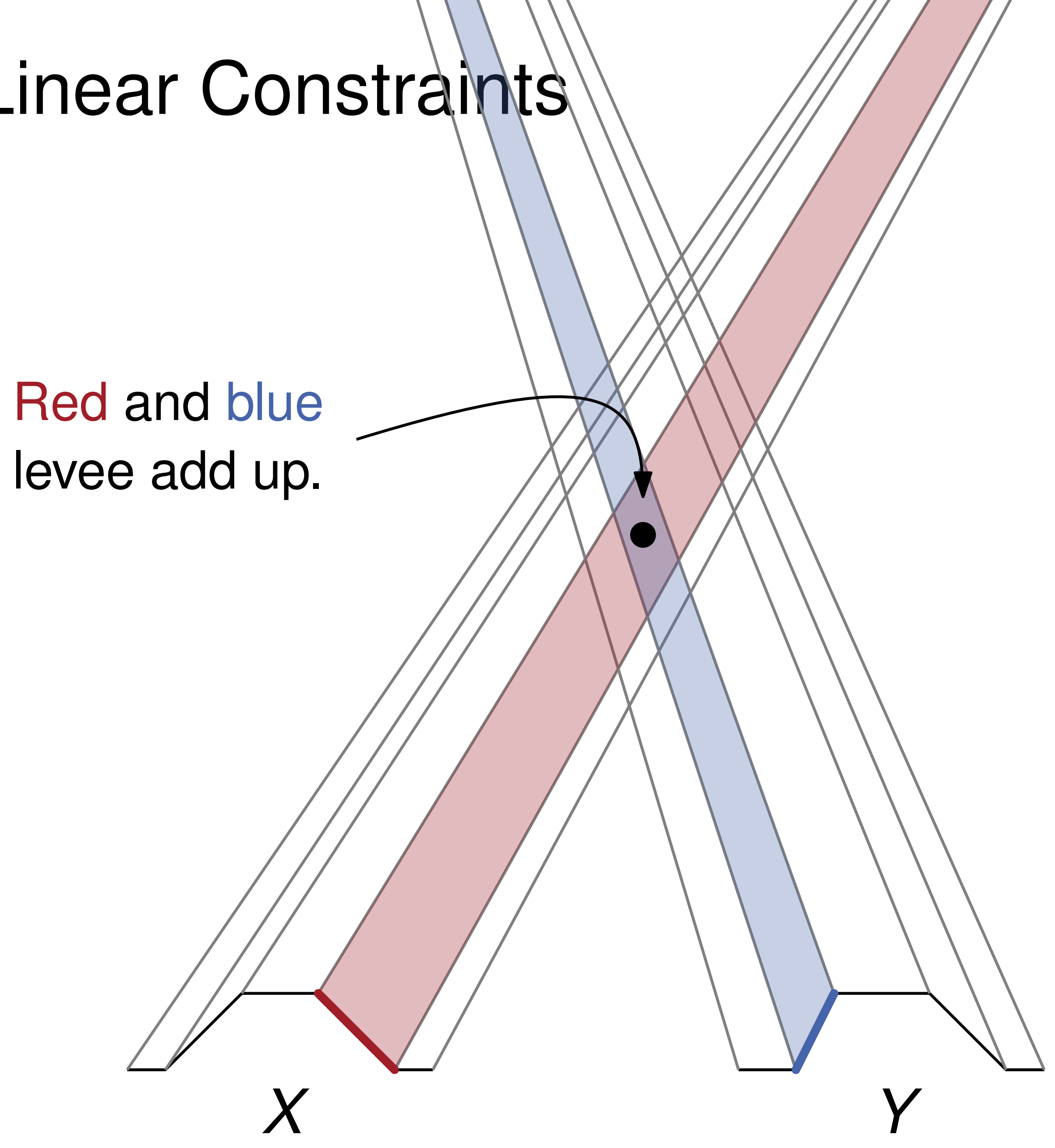
Linear Constraints



Question: How to encode constraints involving X and Y?

- impossible in one dimension
- levees intersect in two dimensions

Linear Constraints



Question: How to encode constraints involving X and Y?

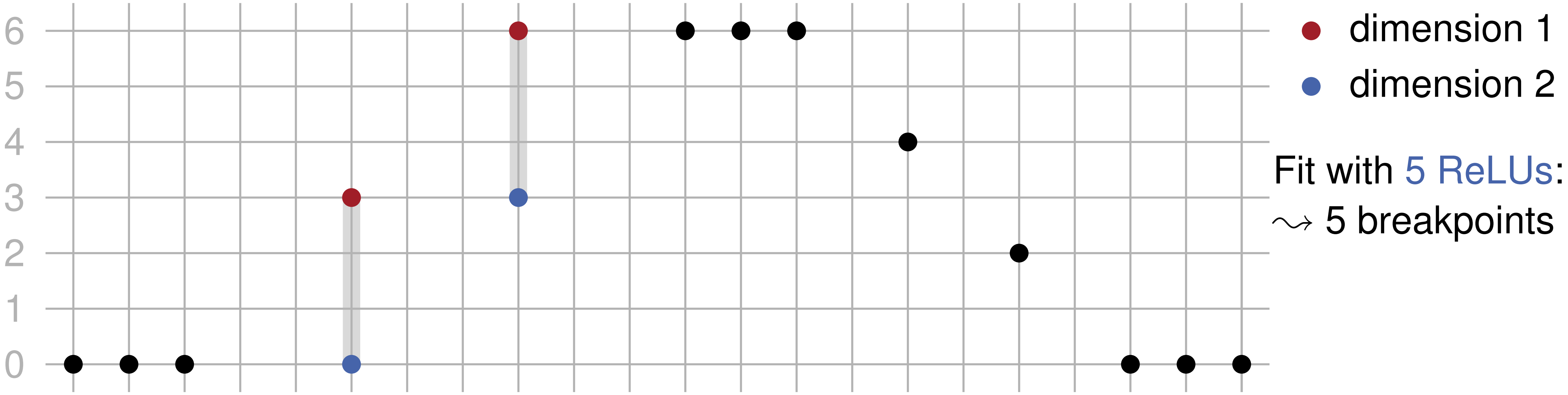
- impossible in one dimension
- levees intersect in two dimensions
- Add a data point in intersection to encode a linear constraint.

Nonlinear Constraint

Task: Encode a nonlinear relation.

Nonlinear Constraint

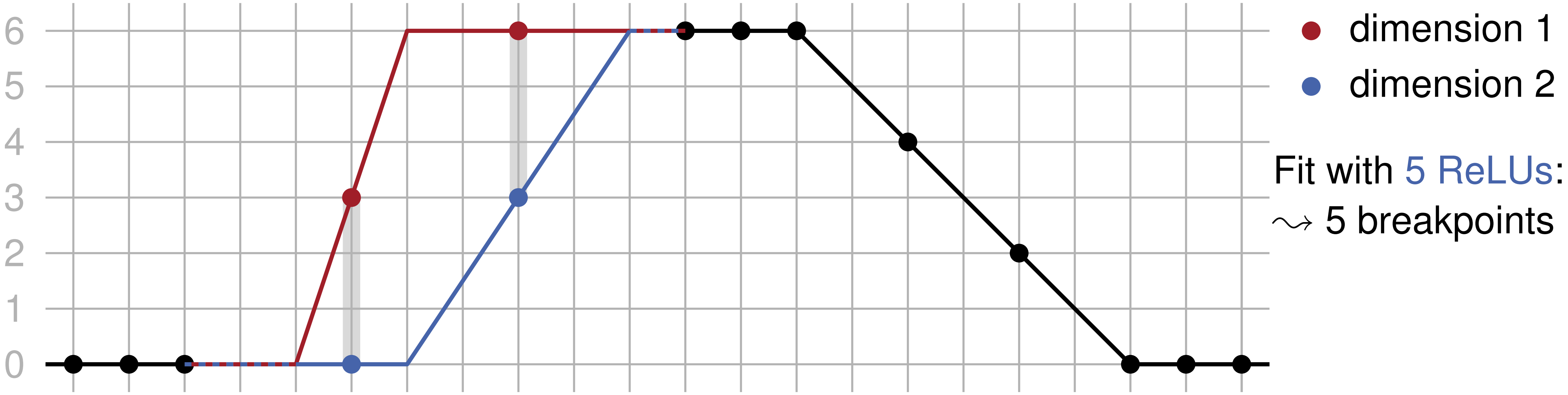
Task: Encode a nonlinear relation.



Fit with 5 ReLUs:
$\rightsquigarrow$ 5 breakpoints

Nonlinear Constraint

Task: Encode a nonlinear relation.

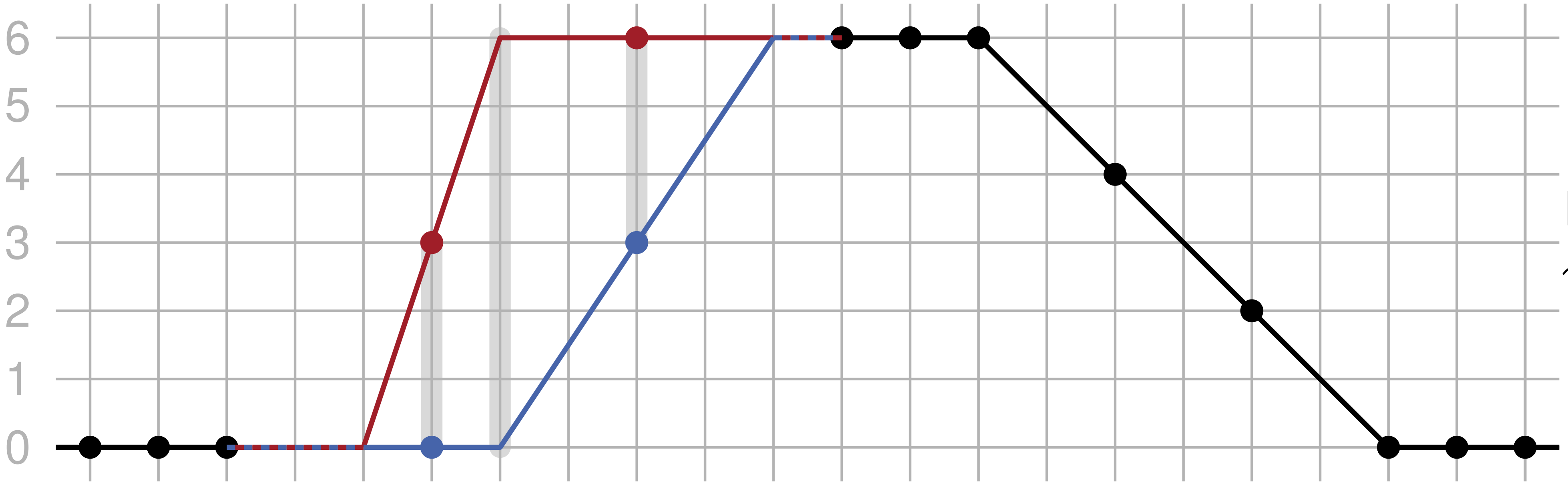


Fit with 5 ReLUs:
$\rightsquigarrow$ 5 breakpoints

Daniel Bertschinger, Christoph Hertrich, **Paul Jungeblut**, Tillmann Miltzow, Simon Weber

Nonlinear Constraint

Task: Encode a nonlinear relation.



dimension 1

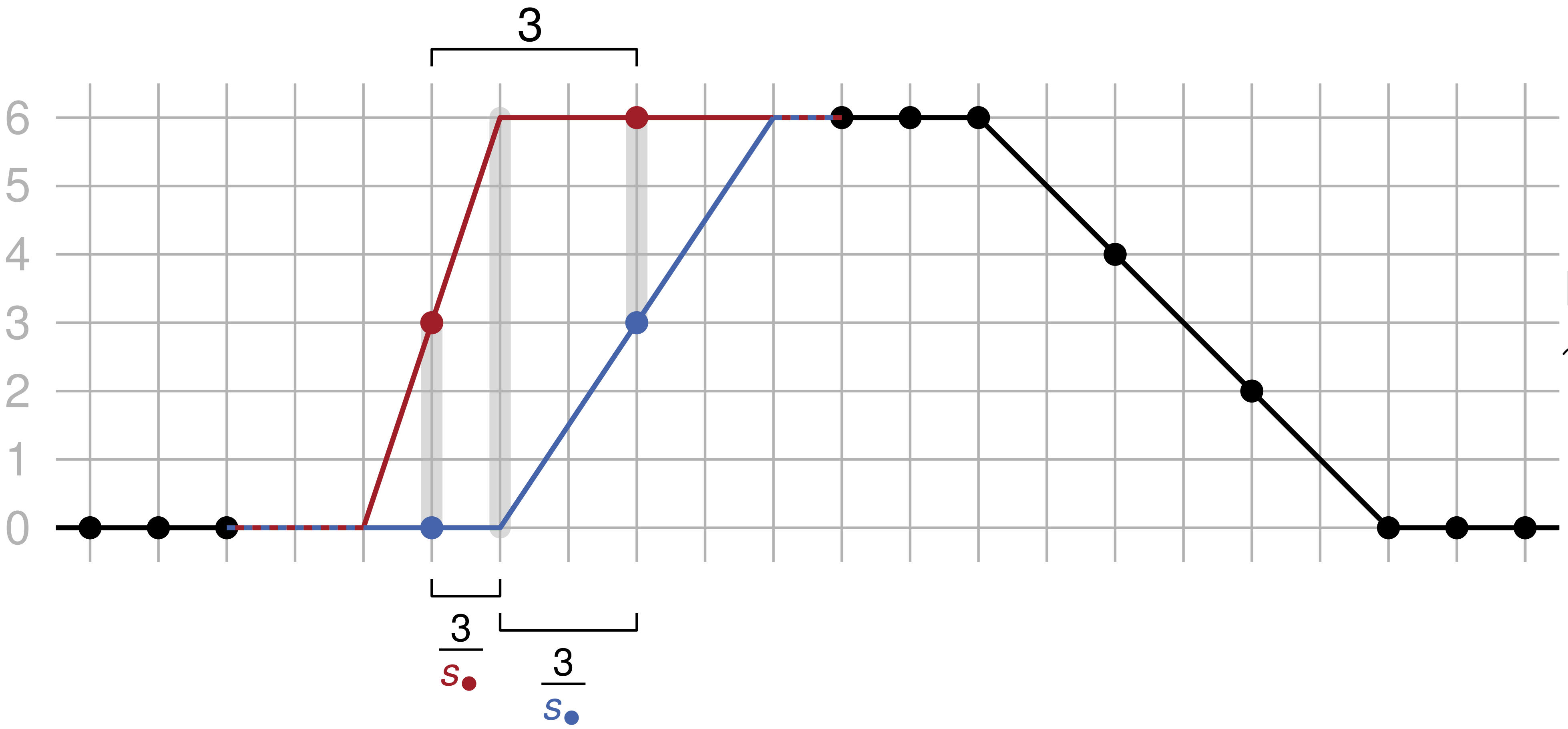
dimension 2

Fit with 5 ReLUs:
$\rightsquigarrow$ 5 breakpoints

Possible, but dimensions need to share one breakpoint.

Nonlinear Constraint

Task: Encode a nonlinear relation.



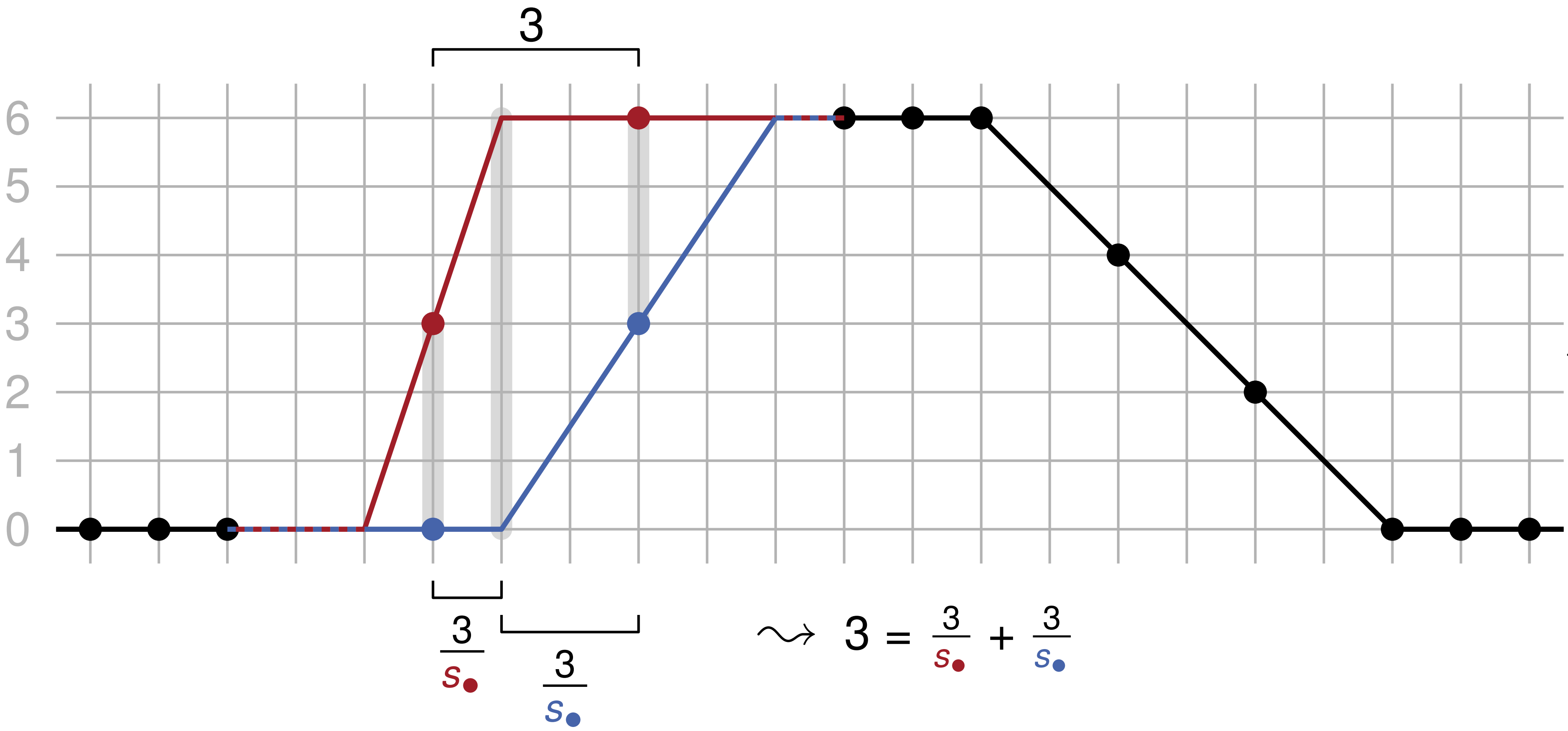
- dimension 1
- dimension 2

Fit with 5 ReLUs:
⇝ 5 breakpoints

Possible, but dimensions need to share one breakpoint.

Nonlinear Constraint

Task: Encode a nonlinear relation.



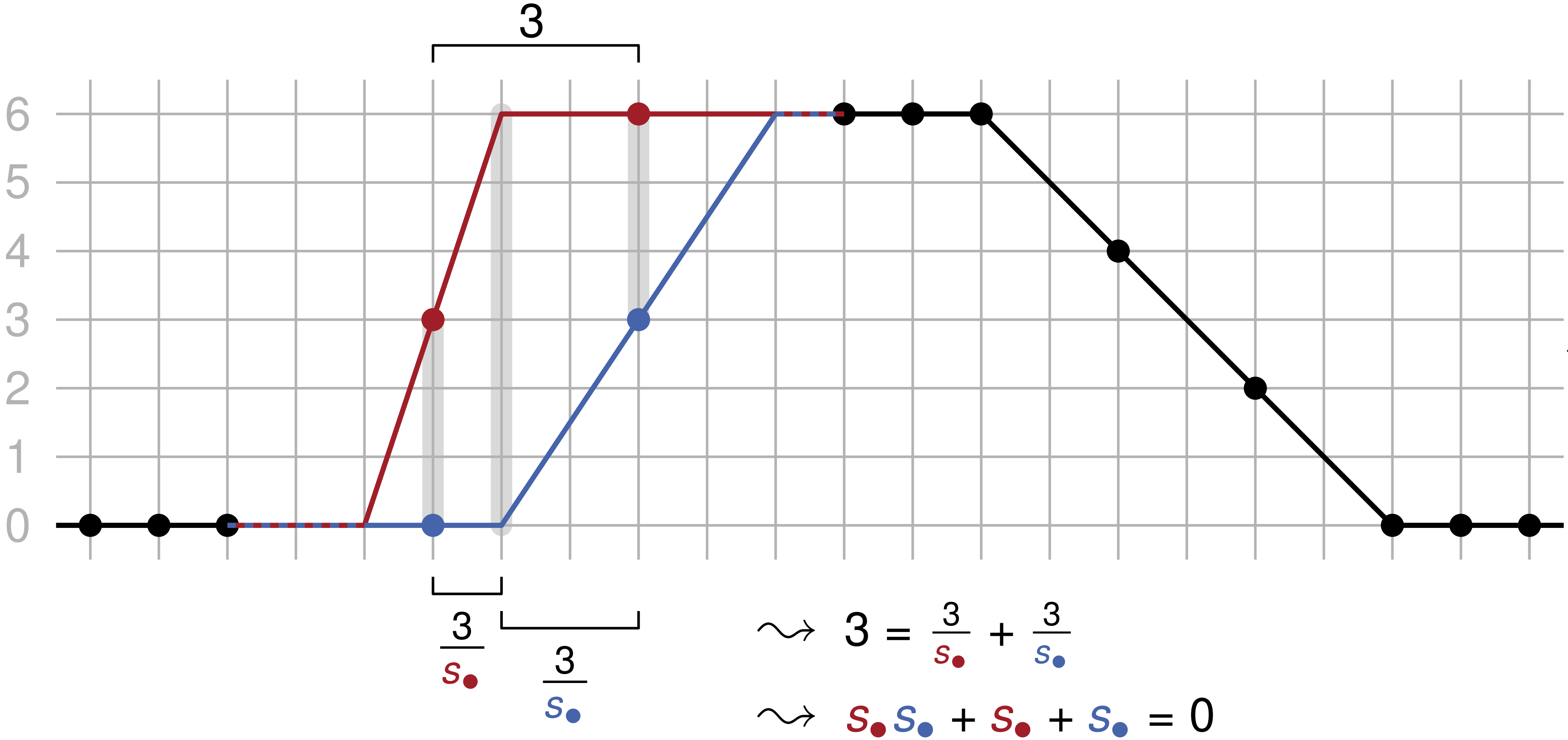
$\rightsquigarrow\ 3 = \frac{3}{s_{\color{red}\bullet}} + \frac{3}{s_{\color{blue}\bullet}}$

Fit with 5 ReLUs:
$\rightsquigarrow$ 5 breakpoints

Possible, but dimensions need to share one breakpoint.

Nonlinear Constraint

Task: Encode a nonlinear relation.



- ● dimension 1
- ● dimension 2

Fit with 5 ReLUs:
⇝ 5 breakpoints

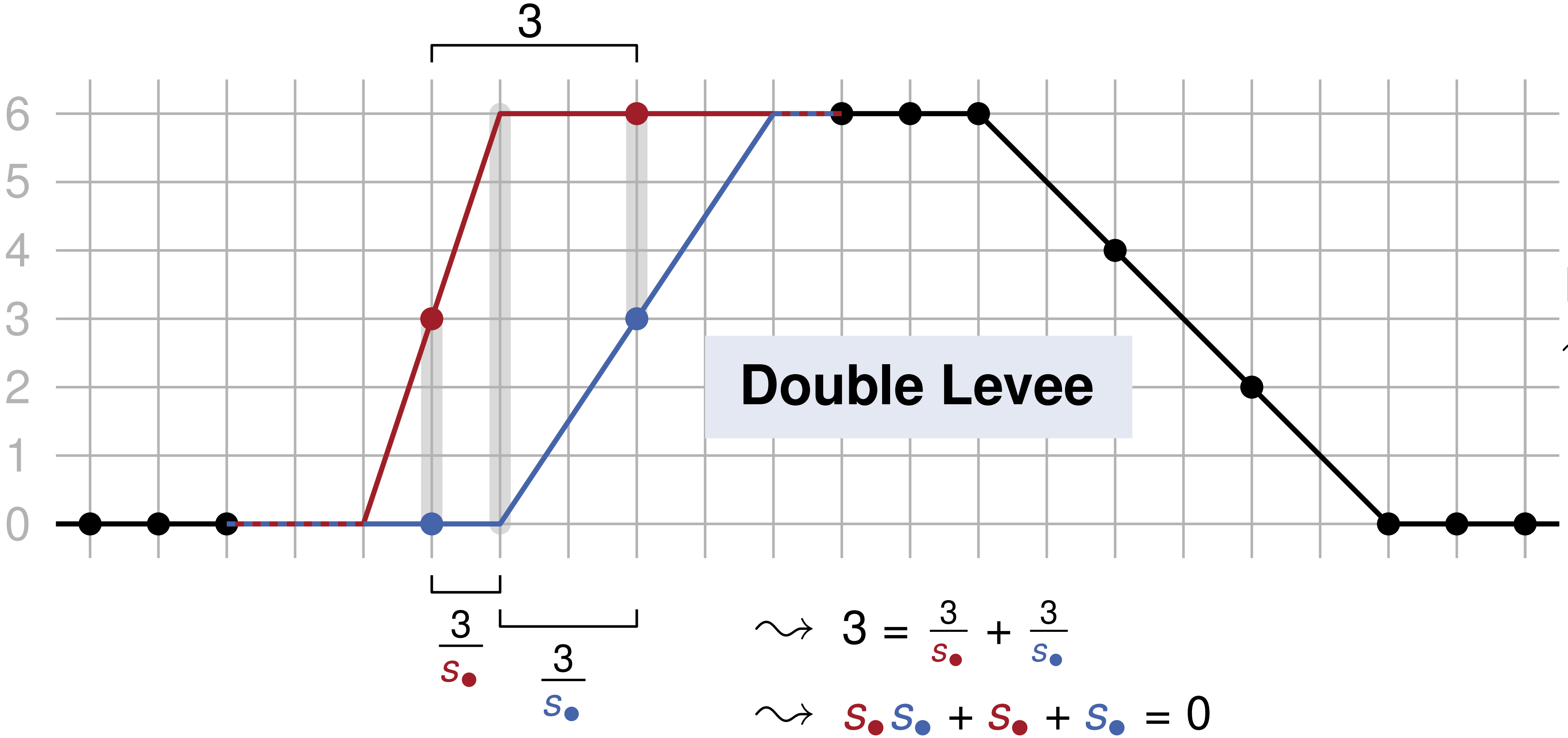
$\rightsquigarrow\ 3 = \frac{3}{s_{\bullet}} + \frac{3}{s_{\bullet}}$

$\rightsquigarrow\ s_{\bullet} s_{\bullet} + s_{\bullet} + s_{\bullet} = 0$

Possible, but dimensions need to share one breakpoint.

Nonlinear Constraint

Task: Encode a nonlinear relation.



$\rightsquigarrow\ 3 = \frac{3}{s_{\color{red}\bullet}} + \frac{3}{s_{\color{blue}\bullet}}$

$\rightsquigarrow\ s_{\color{red}\bullet} s_{\color{blue}\bullet} + s_{\color{red}\bullet} + s_{\color{blue}\bullet} = 0$

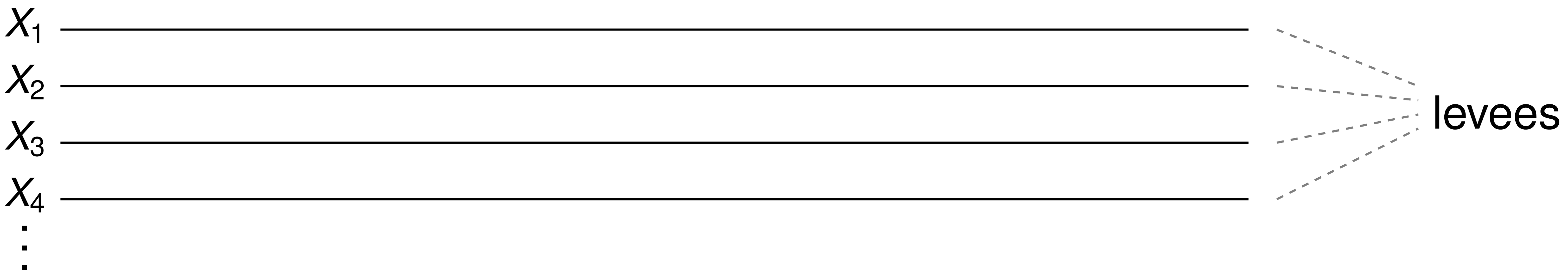
● dimension 1
● dimension 2

Fit with 5 ReLUs:
$\rightsquigarrow$ 5 breakpoints

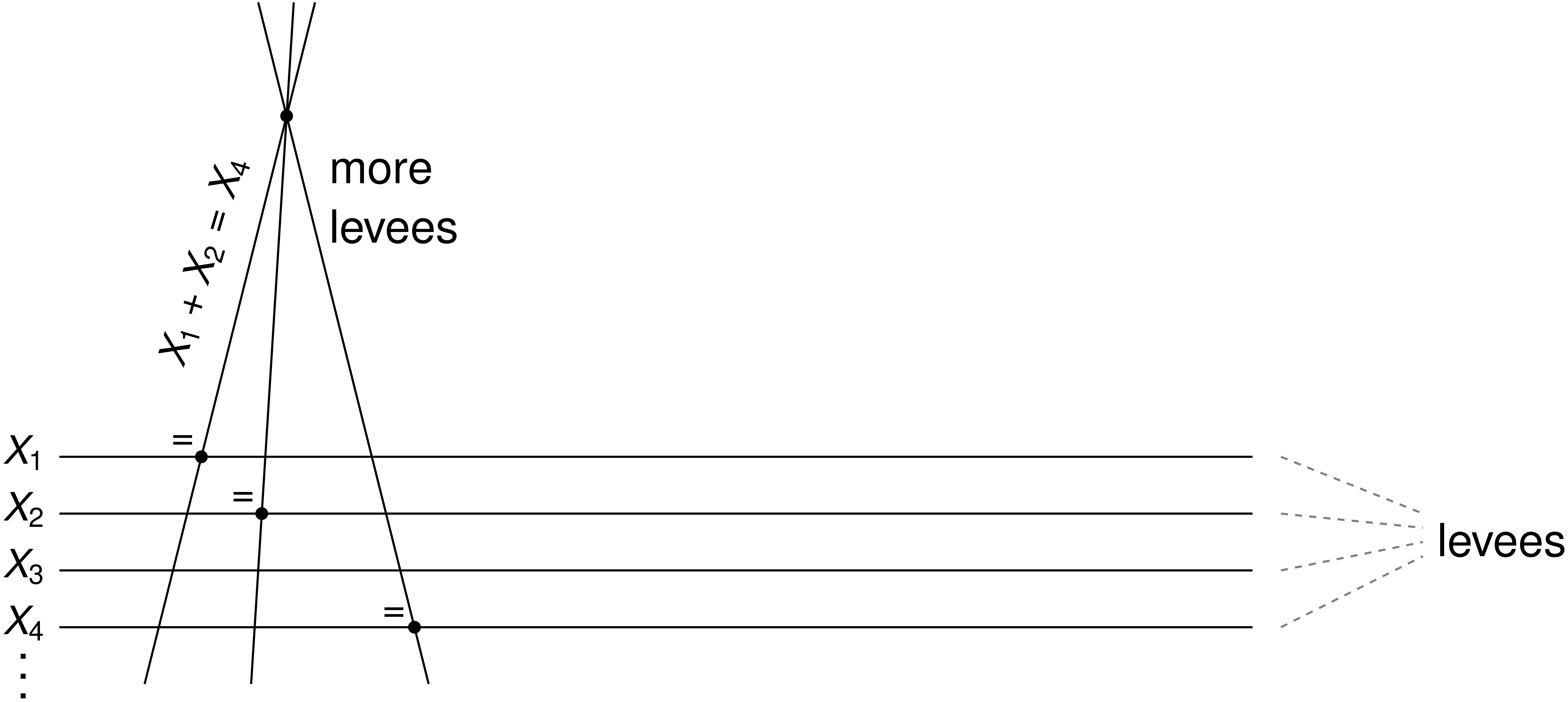
Possible, but dimensions need to share one breakpoint.

Training Fully Connected Neural Networks is ∃ℝ-Complete
Daniel Bertschinger, Christoph Hertrich, **Paul Jungeblut**, Tillmann Miltzow, Simon Weber

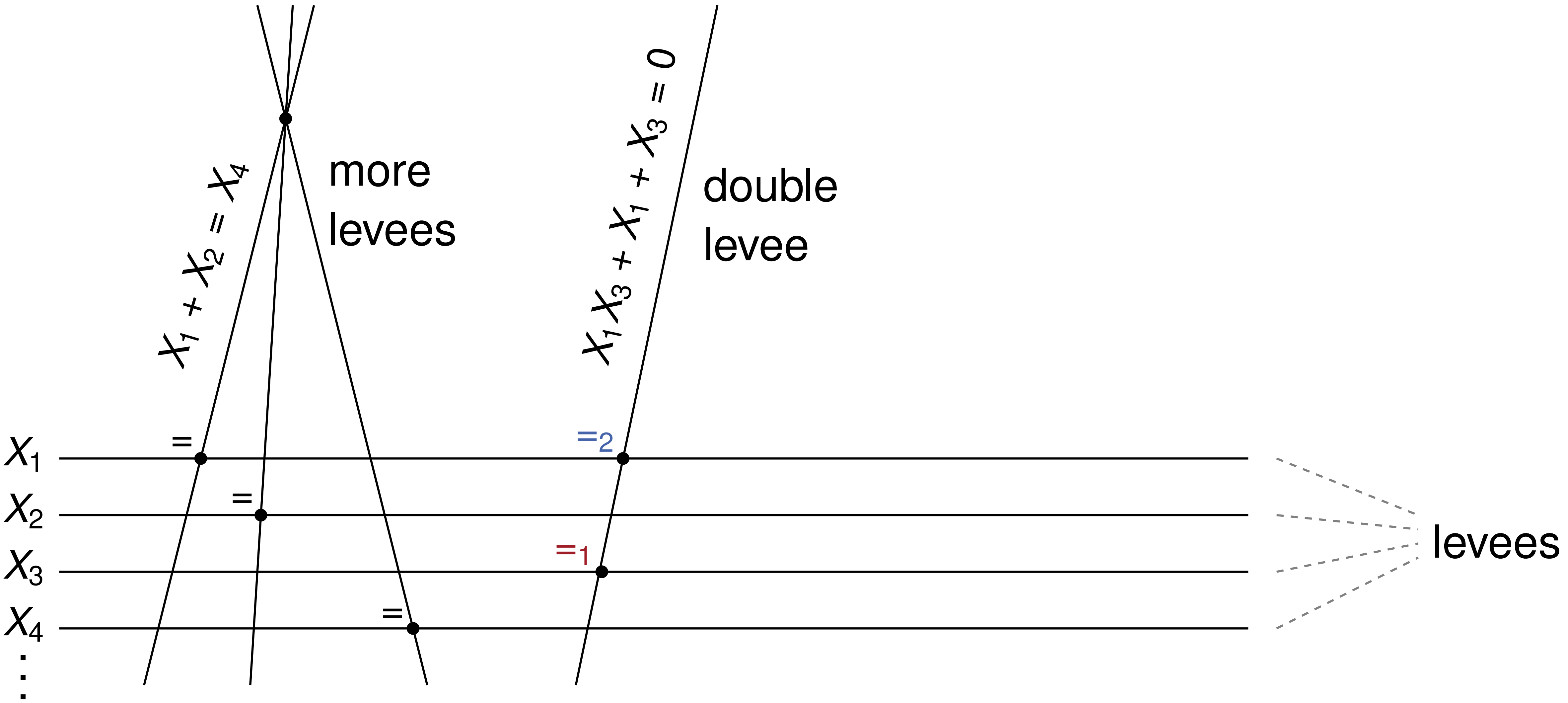
Top Down View



Top Down View

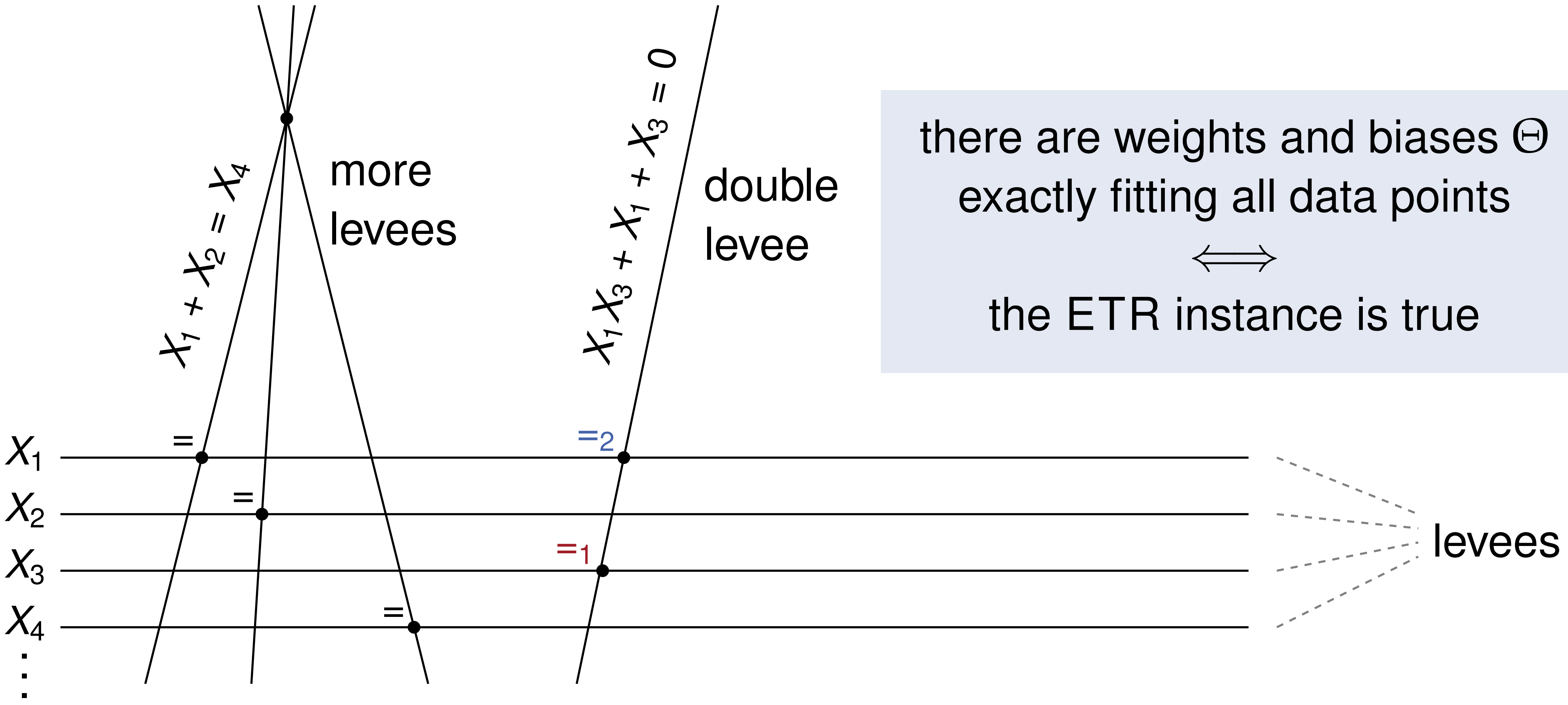


Top Down View



Daniel Bertschinger, Christoph Hertrich, **Paul Jungeblut**, Tillmann Miltzow, Simon Weber

Top Down View



there are weights and biases Θ exactly fitting all data points

$\Longleftrightarrow$

the ETR instance is true

Questions?

Questions?

Thank you!