Neural Networks Solvers for the Yang-Baxter Equation

Shailesh Lal



# Neural Networks Solvers for the Yang-Baxter Equation

Shailesh Lal

shaileshlal at bimsa dot cn

7 March 2024

University of Nottingham

SL, Suvajit Majumder, Evgeny Sobko 2304.07247,. . .

# Introduction & Motivation

- Symmetry: a deep and profound role in physical systems.
- Constrains possible dynamics, physical parameters.
- e.g. momentum conservation iff translational invariance.
- An extreme example: infinitely many conserved quantities.
- Morally the 'simplest' theories, everything constrained.
- Question: completely determine the theory by symmetry?
- Example: The Conformal Bootstrap

# Introduction & Motivation

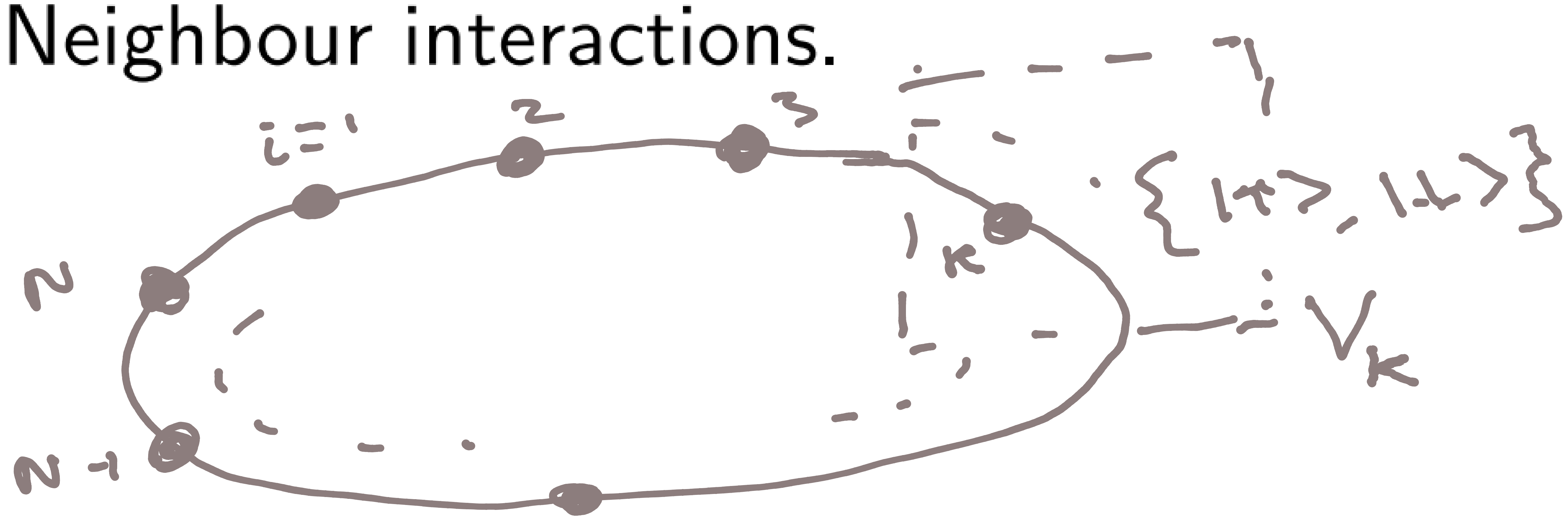
## Example: The Conformal Bootstrap

- Consider a theory with rotational invariance.
- Two observers must agree on the length of given vectors.
- Let us consider additional symmetries, e.g. scaling.
- Two observers need not agree on the length of a vector.
- must agree on the angle between two vectors.
- A non-trivial extension of this idea is conformal invariance.
- This symmetry is very restrictive, $\infty$ dimensional in 2d.
- unitarity + crossing symmetry + some minimal information: is the theory uniquely specified?

# Quantum Integrability

Consider an N site spin chain indexed by $k = 1, 2, \ldots N$

1. periodic boundary conditions $N + k \equiv k$.
2. Nearest-Neighbour interactions.




The Hilbert space: $V_1 \otimes V_2 \otimes \ldots \otimes V_N$ , $V_k \simeq \mathbb{C}^2$

Each $V_k$ is spanned by $\{|\uparrow\rangle, |\downarrow\rangle\}$.

The local spin operators at site k: $S_k^{x,y,z} = \frac{1}{2}\sigma_k^{x,y,z}$ .

# Quantum Integrability

The Hamiltonian is a sum of nearest-neighbour interactions

$$H = \sum_{k=1}^{N} \left( \sum_{\alpha=\{x,y,z\}} J^{\alpha} S_i^{\alpha} \otimes S_{i+1}^{\alpha} \right) .$$

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} , \; B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} , \; A \otimes B = \begin{pmatrix} a_{11} b_{11} & a_{11} b_{12} & a_{12} b_{11} & a_{12} b_{12} \\ a_{11} b_{21} & a_{11} b_{22} & a_{12} b_{21} & a_{12} b_{22} \\ a_{21} b_{11} & a_{21} b_{12} & a_{22} b_{11} & a_{22} b_{12} \\ a_{21} b_{21} & a_{21} b_{22} & a_{22} b_{21} & a_{22} b_{22} \end{pmatrix}$$

Then the nearest-neighbour Hamiltonian is

$$H = \begin{pmatrix} J_z & 0 & 0 & J_x - J_y \\ 0 & -J_z & J_x + J_y & 0 \\ 0 & J_x + J_y & -J_z & 0 \\ J_x - J_y & 0 & 0 & J_z \end{pmatrix} : \text{XYZ model} .$$

Also limits $J_x = J_y \neq J_z$: XXZ model and $J_x = J_y = J_z$ Heisenberg spin-chain

# Quantum Integrability

These systems are quantum integrable: infinite number of conserved charges.

Construction: define an R-matrix $R_{ij}(u_i, u_j)$ over $V_i \otimes V_j$

Further, take difference form: $R_{ij}(u_i, u_j) = R_{ij}(u_i - u_j)$.

If $V \simeq \mathbb{C}^2$ then 16 undetermined functions. Conditions:

- $R(0) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \qquad H = R^{-1} \frac{d}{du} R|_{u=0}$
- higher derivatives encode higher charges.

R-matrix also obeys an important consistency condition.

# Quantum Integrability

## Yang-Baxter Equation

$$\boxed{R_{12}(u_1, u_2)\, R_{13}(u_1, u_3)\, R_{23}(u_2, u_3) = R_{23}(u_2, u_3)\, R_{13}(u_1, u_3)\, R_{12}(u_1, u_2)}$$

This equation is over $V_1 \otimes V_2 \otimes V_3$.

Implicit shorthand: $R_{12}(u_1, u_2) \equiv R_{12}(u_1, u_2) \otimes \mathbb{I}$

If $V_k \simeq \mathbb{C}^2$ then 64 cubic functional equations

Therefore: an abstract definition of quantum integrability:

1. solve YBE + other consistency criterion
2. determines a qtm integrable spin chain with n.n. intxns.

# Constructing Integrable Systems

How can we construct integrable systems from scratch?

- Assume the $R$ matrix is holomorphic, local.
- The target Hamiltonian is known.
- Construct the $R$ matrix from the Yang-Baxter Equation?

Q: what if only constraints on Hamiltonian are given?

Q: find integrable systems nearby a given starting system?

Q: finding classes of integrable systems?

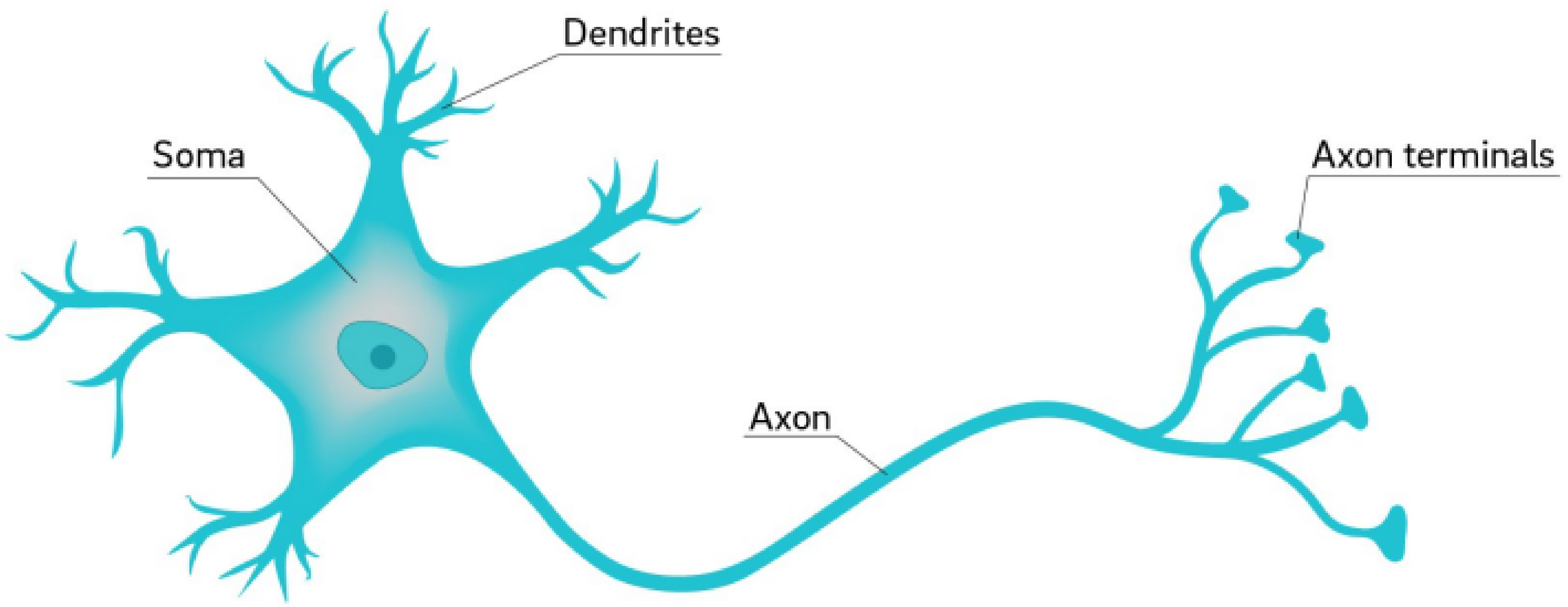
Neural Networks are promising tools for these questions.

- span a large function space
- constraints can be supplied using loss functions.

# Neural Networks

## Inspiration from biological neurons

Neuron




- dendrites receive inputs to neuron
- soma, or cell body, takes a weighted sum of inputs
- axon processes this into the output.

responsible for all our perceptions, decisions, . . .

# Neural Networks

## An Artificial Neuron

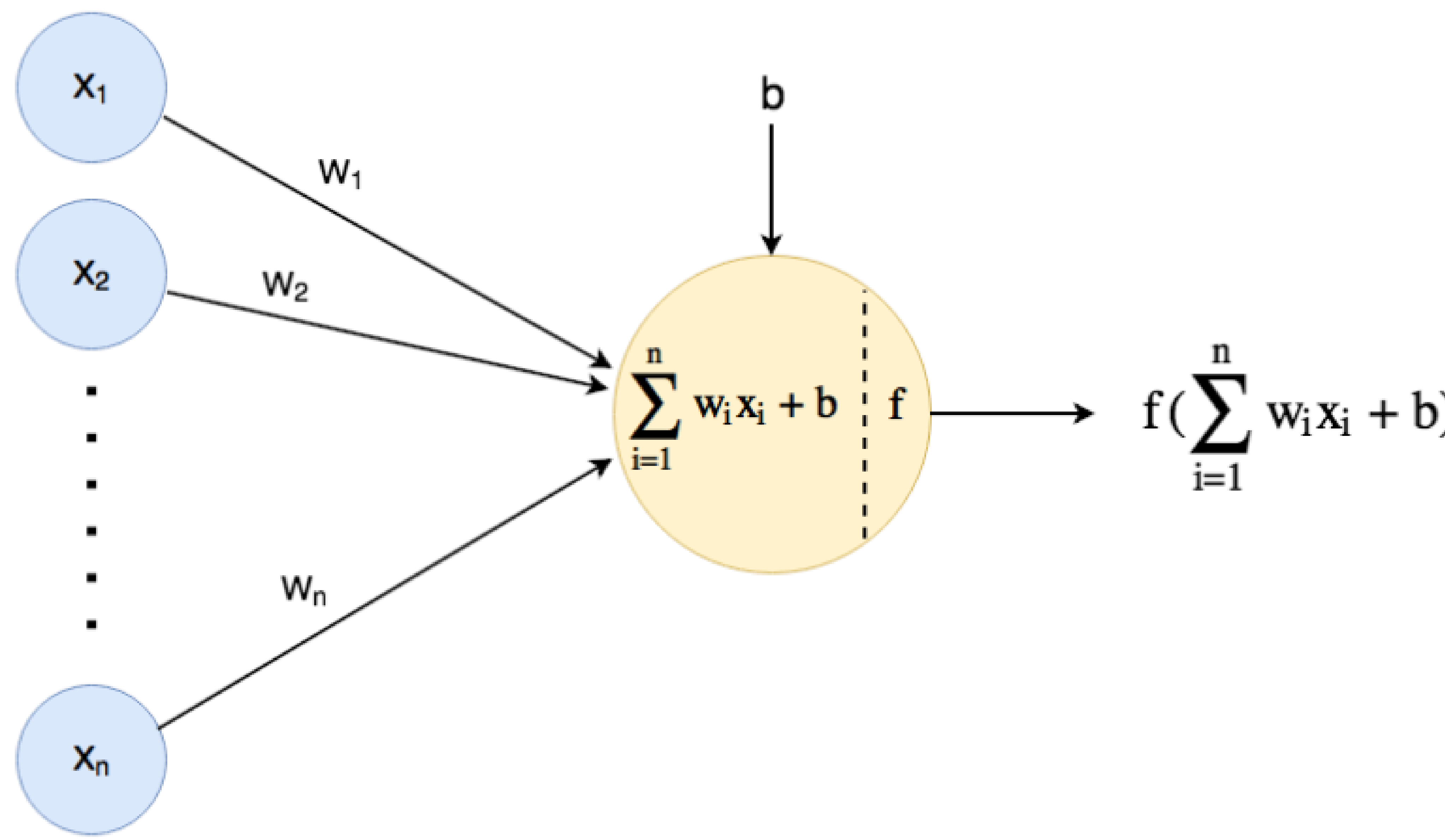


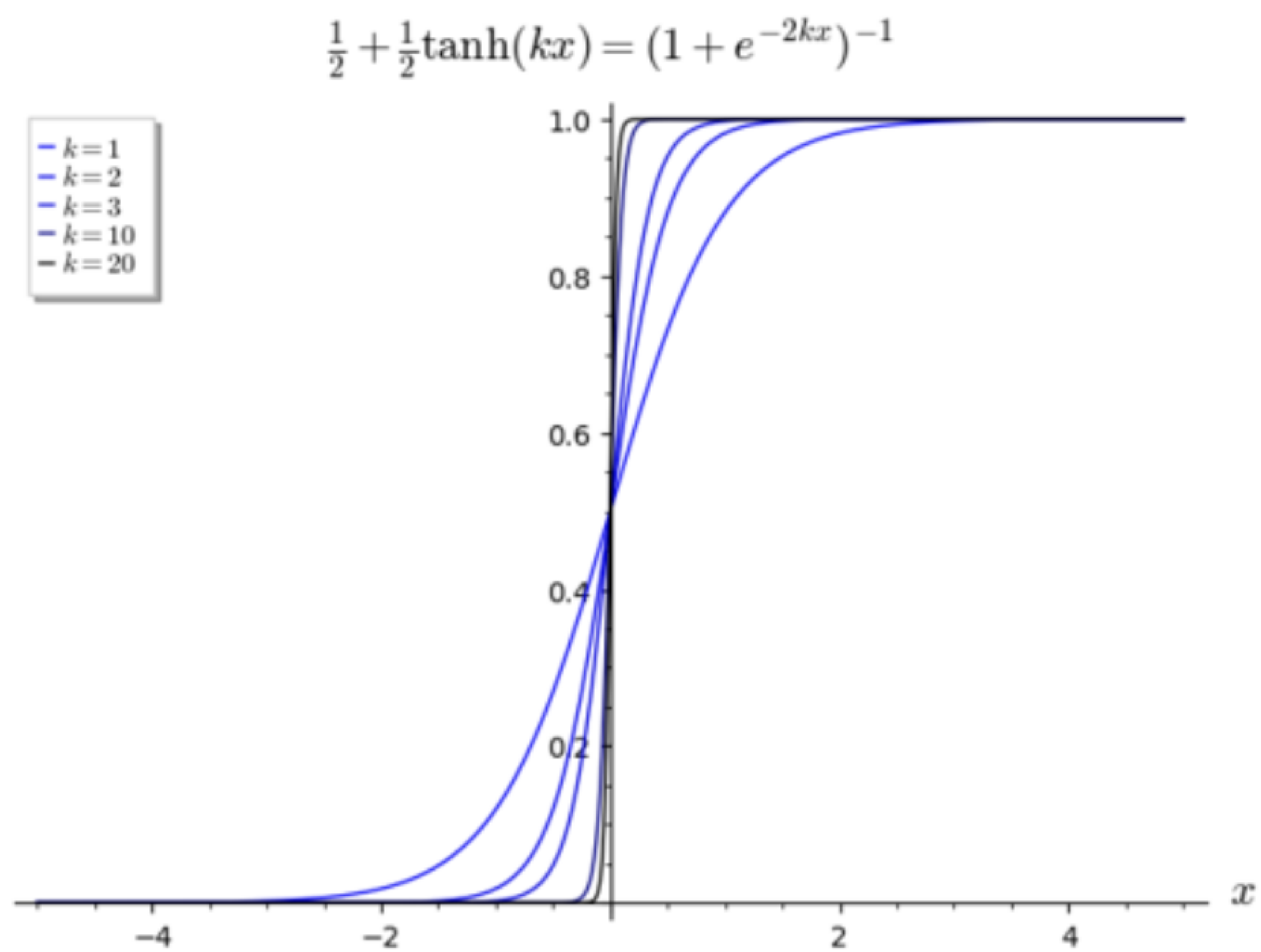

Figure: A single neuron: $\mathcal{O} = \mathbf{f}\left(\vec{w} \cdot \vec{x} + b\right)$

$w$ and $b$ are tunable parameters: weights & biases

**f** is a typically non-linear function: activation function

# The Perceptron

Extreme example: **f** is a step function.




Notes:

1. limit of the tanh function.
2. fires when inputs cross a threshold: property of neurons.

# Neural Networks

Consider $k$ such neurons acting on the inputs $x$:

$$z_k = w_{kj} x_j + b_k \, , \quad a_k = \mathbf{f}\left(a_k\right) \, .$$

Also, since hindsight is 20/20, redefine

$$x_k \to a_k^{(0)} \, , \quad z_k \to z_k^{(1)} \, , \quad a_k \to a_k^{(1)} \, .$$

Therefore we rewrite

$$z_k^{(\ell)} = w_{kj}^{(\ell)} a_j^{(\ell-1)} + b_k^{(\ell)} \, , \quad a_k^{(\ell)} = \mathbf{f}\left(z_k^{(\ell)}\right) \, ,$$

where $\ell = 1$. With hindsight, call this the first layer of neurons.

# Neural Networks

Higher layers: iterate this transformation.

$$z_k^{(\ell)} = w_{kj}^{(\ell)} a_j^{(\ell-1)} + b_k^{(\ell)}\,, \quad a_k^{(\ell)} = \mathbf{f}\left(z_k^{(\ell)}\right)\,,$$

for $\ell = 1, \ldots, L$.

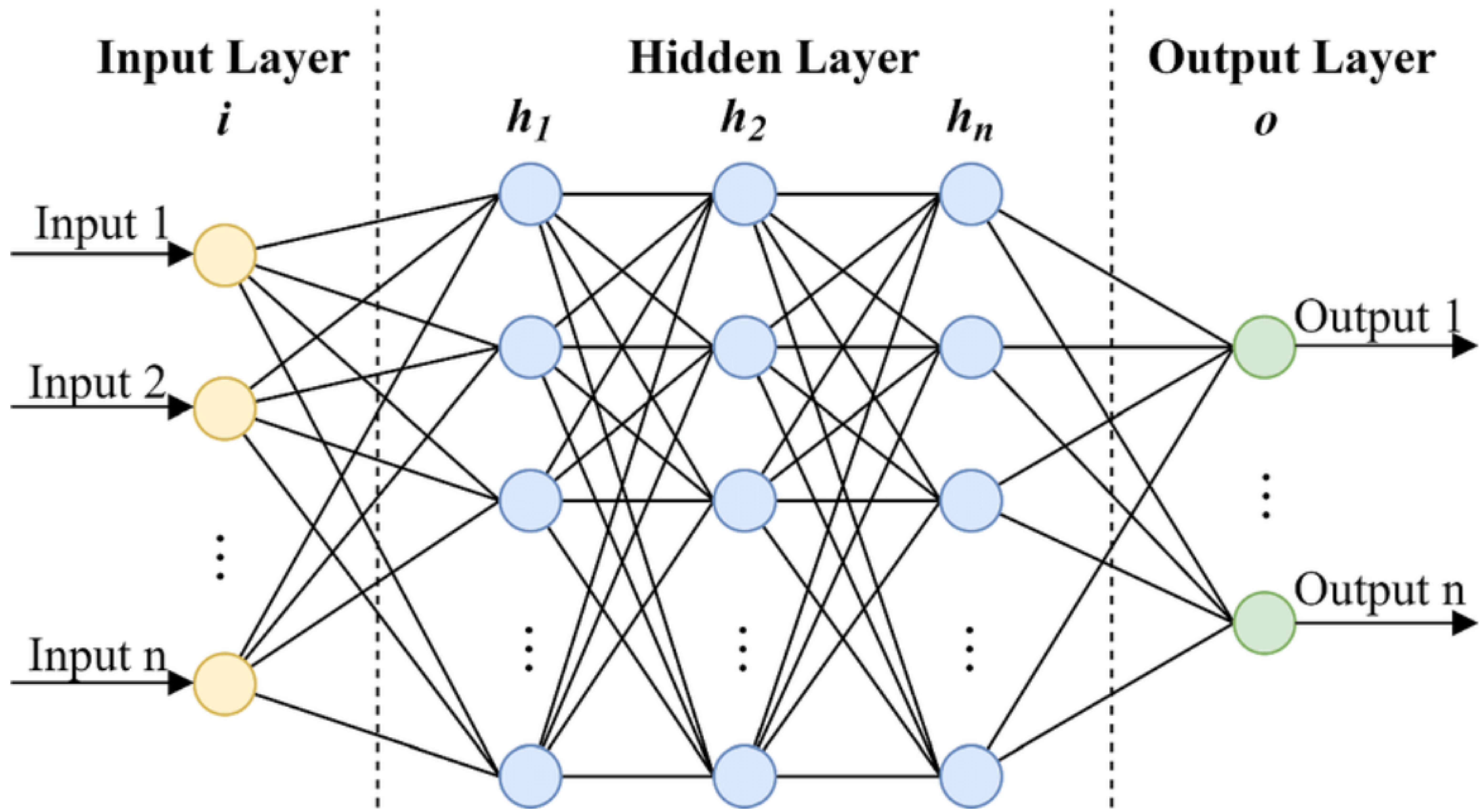
- The simplest modern neural network.
- Multi-layer Perceptron.
- Other topologies are also possible, e.g. residual learning.

hierarchical structure:

- Output of previous layer is the input of the current layer.
- Output of current layer is the input of the next layer.

# Neural Networks

This graph represents the above sequence of transformations.




The output $a^{(L)}$ is

$$y_k = f_{\{w,b\}}(x) ,$$

and can be changed for given $x$ by tuning $w$ and $b$.

# Universal Approximation

**Why are Neural Networks so appealing?**

A classic problem: Supervised Learning

- start with a set of known input/output pairs $\{(x_i, y_i)\}$
- determine $\{w, b\}$ of the neural network so that

$$y_{NN}(x_i) \approx y_i \quad \forall \quad x_i .$$

**Claim: neural networks are universal approximators**

i.e. this data can always be modeled by *some* neural network.

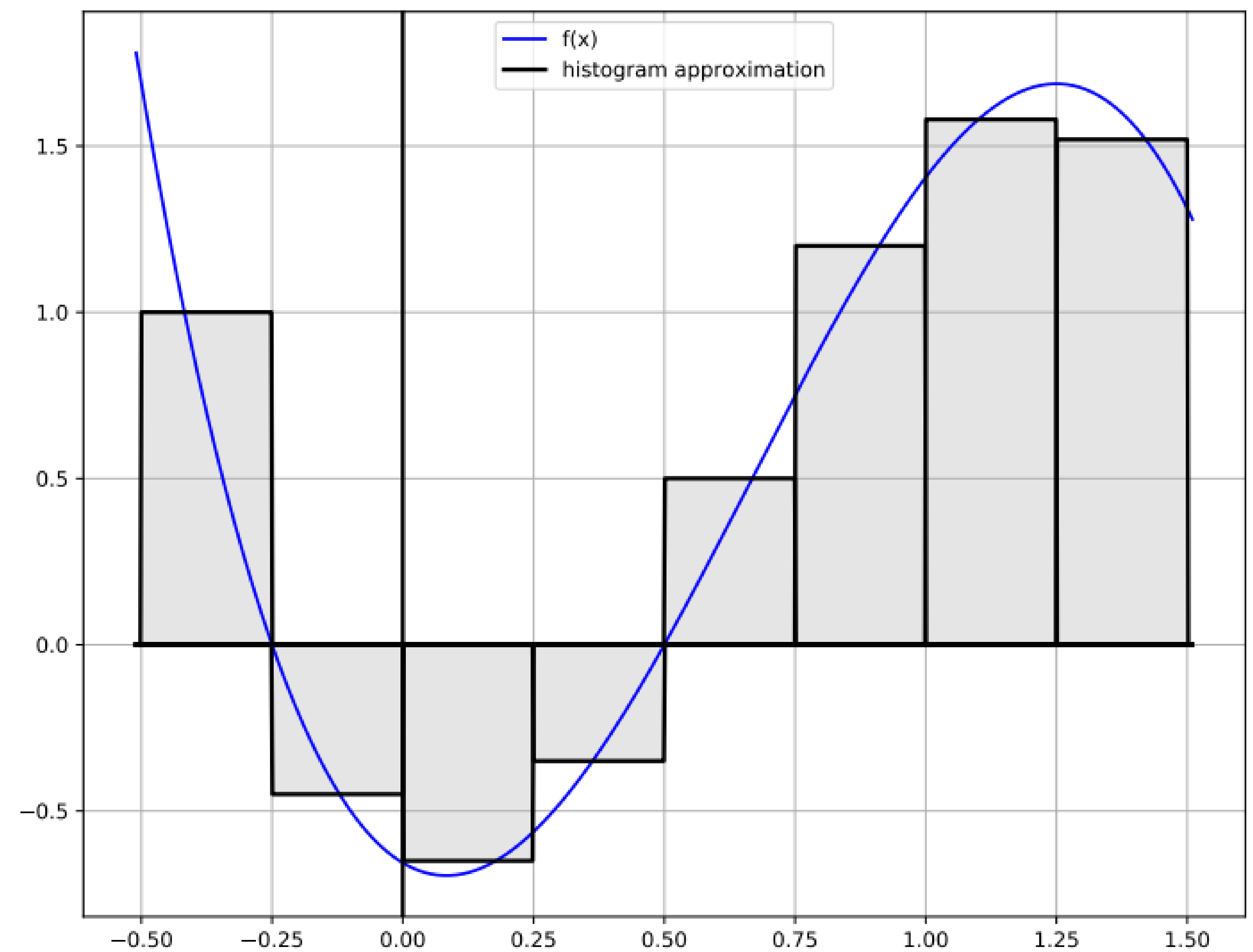
This is for two reasons: width and depth.

- $\infty-$width neural nets are universal approxtrs.
- depth is needed to learn not memorize.

We start with the first statement.

# Universal Approximation

Idea: approximate target function by a histogram of $N$ bins.



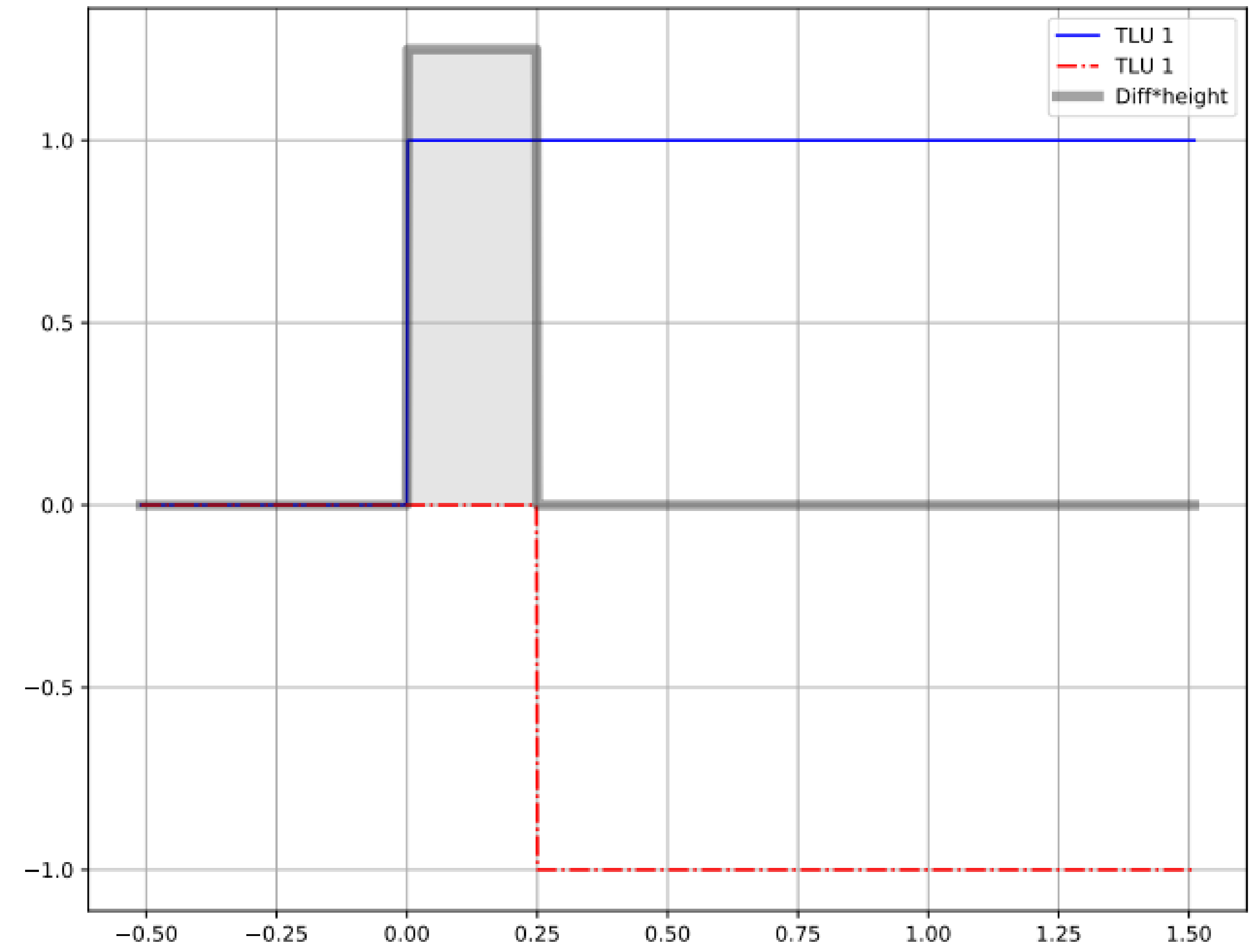

$$f(x) \approx f_0(x) = \bar{f}_j\,, \quad x \in [x_{j-1}, x_j]\ , \quad j = 1, 2, \ldots, N$$

Larger $N \Rightarrow$ better approximation.

# Universal approximation

This can be realized through the step function

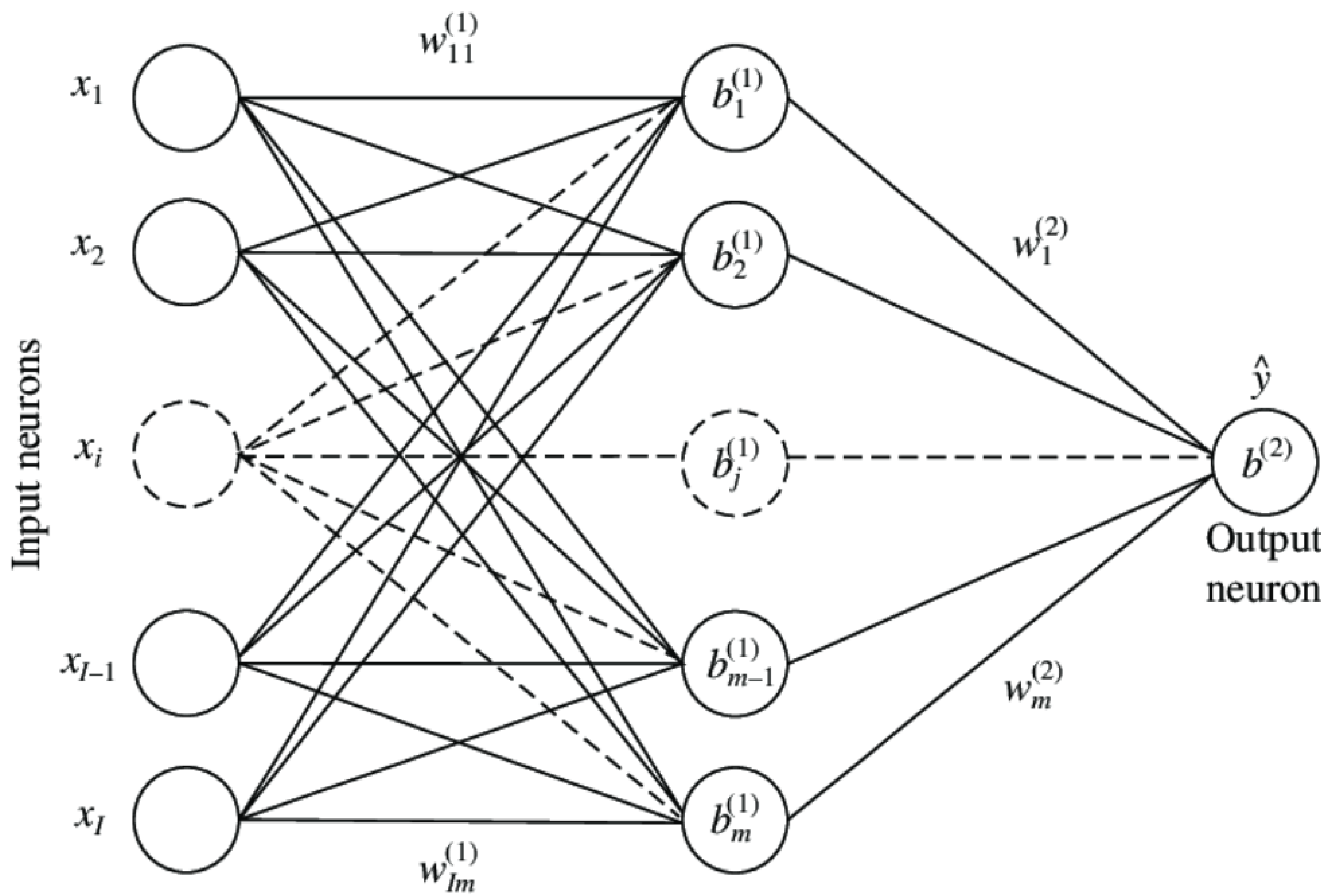
$$f_s(x) = \Theta(wx+b)\ , \qquad s = -\frac{b}{w}\ .$$




$$\text{define: } f_{s_1,s_2,h} = h\left(f_{s_1} - f_{s_2}\right) = \begin{cases} h & : \quad x \in (s_1, s_2) \\ 0 & : \quad \text{otherwise} \end{cases}$$

# Universal Approximation

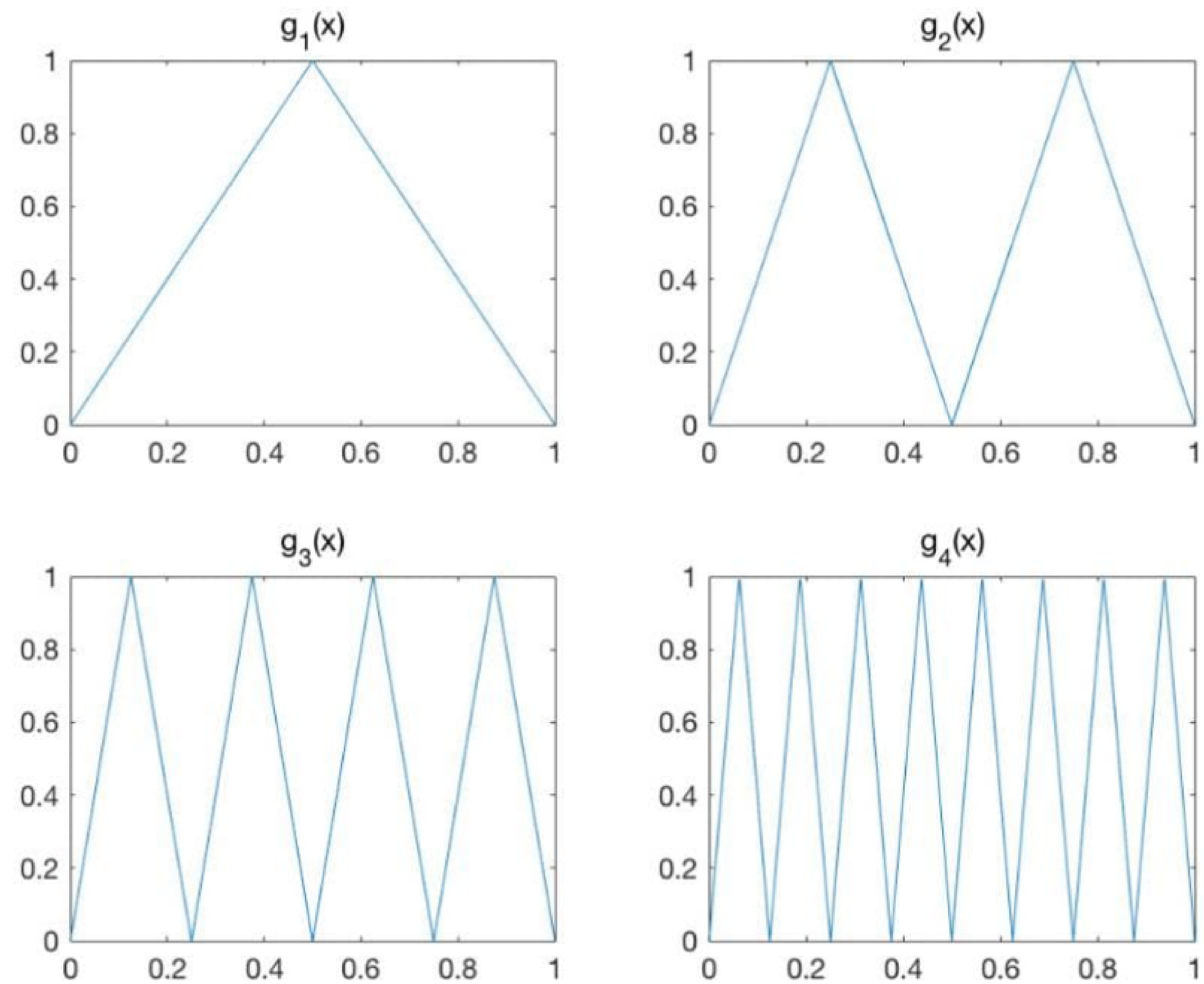
A single hidden layer of neurons yields a universal approximator.




why then do we need depth?

# Universal Approximation

Depth is crucial to learning heirarchical structures efficiently.




wide-net : 1,2,4,16 neurons.    deep-net : 1,2,3,4 neurons.

Also: learning not memorization.

# Today

- Use neural networks to tackle quantum integrability.
- What is the data? Generalize supervised learning.
- Provide the constraints that the solution must satisfy.

$$\text{e.g.}\quad R(u)\;:\quad f(R*R*R)=0\,.$$

- Neural nets scan across function space without fixing an explicit ansatz class, such as exponential or trig functions.
- Ansatz implicitly defined by activation, width and depth.
- Loss functions encode constraints on target function.

# Loss Functions

Original Idea: evaluate how good or bad is the predictor.

e.g. the mean square error loss function:

$$\mathcal{L}(w) = \sum_n \left(y_n - y_{NN}(x_n, w)\right)^2$$

If $\mathcal{L}(w)$ is small the corresponding predictions are good.

$\therefore$ minimize $\mathcal{L}(w)$ in $w \Rightarrow$ canonical supervised learning

conceptually not very different from ordinary least squares

Now: generalize how we think about loss functions.

# Rethinking Loss Functions

When are loss functions zero? consider mean square error

$$\mathcal{L}(w) = \sum_n \left(y_n - y_{NN}(x_n, w)\right)^2$$

Minimized at:

$$\mathcal{L}(w) = 0 \quad \Leftrightarrow \quad y_{NN}(x_n, w) = y_n \quad \forall \quad n$$

Minimizing $\mathcal{L} \Rightarrow$ searching for functions $y_{NN}$ such that

$$y_{NN}(x_n, w) = y_n \quad \forall \quad n\,.$$

Loss functions $\equiv$ constraints on the target function.

Loss functions are zero when constraints are satisfied.

# Machine Learning Similarity

Idea: Draw inspiration and analogy from how humans learn.

- a toddler learning animals needs only 1-2 example images

(a) Cats

(b) Dogs

- New images will be identified as dogs and cats depending on which image they are the most similar too.

Siamese Neural Networks

# Machine Learning Similarity

Hence identify dogs and cats by extrapolating from a tiny set of examples

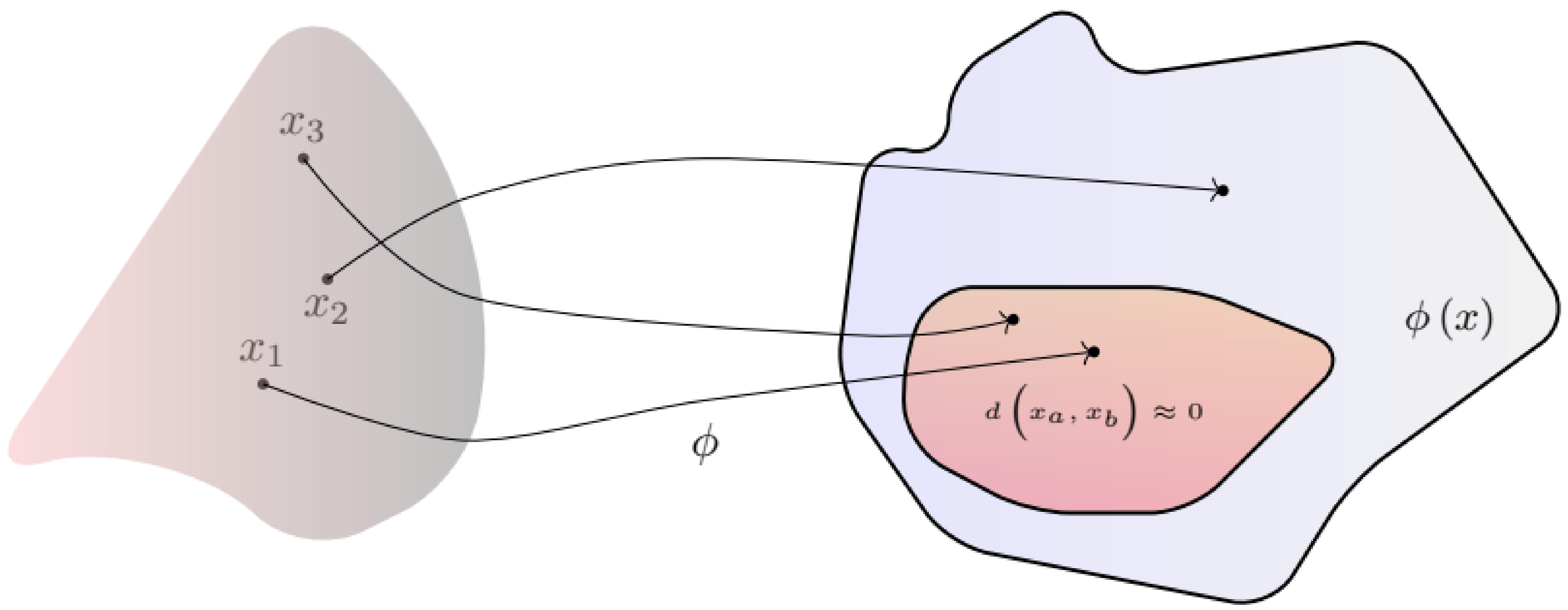
In essence the toddler learns the equivalence relations

$$\mathrm{I} \sim \mathrm{III}\,, \quad \mathrm{II} \sim \mathrm{IV}\,, \quad \text{but} \quad \mathrm{I}, \mathrm{III} \nsim \mathrm{II}, \mathrm{IV}\,.$$

This is now known to be an exceptionally robust framework.

# Machine Learning Similarity

Using machine learning to quantify similarity.




- Learn a map from input data $x$ to $\mathbb{R}^d$
- Points $x_a, x_p$ are similar $\Rightarrow$ close together in $\mathbb{R}^d$
- Points $x_a, x_n$ are dissimilar $\Rightarrow$ far apart in $\mathbb{R}^d$

$\therefore$ similarity encoded into geometry.

# Machine Learning Similarity

Prepare the data:

| Input: pairs $(x_1, x_2)$ | Output: y |
|---|---|
| $x_1 \sim x_2$ : $x_1$ similar to $x_2$ | 1 |
| $x_1 \nsim x_2$ : $x_1$ not similar to $x_2$ | 0 |

The neural network $\phi$ is trained to minimize

$$\mathcal{L} = \sum_{(x_1, x_2)} y\, d(x_1, x_2) + (1-y) \max(1 - d(x_1, x_2), 0)$$

Intuition: $\mathcal{L}$ is zero when

- $y = 1 \quad \Rightarrow d(x_1, x_2) = 0$, i.e. close together
- $y = 0 \quad \Rightarrow d(x_1, x_2) > 1$, i.e. far apart

i.e. a map to $\mathbb{R}^d$ such that distance encodes similarity.

# Summary

- neural networks $\Rightarrow$ exhaustive scan across functions.
- Loss functions encode properties of the target function.
- This can involve enumerating input output pairs.
- Equally well, more abstract properties can be encoded.
- not discussed: how to tune weights in practice.

# A Neural Network Solver

An $R$ matrix comprises of functions $R_{ij}(u)$

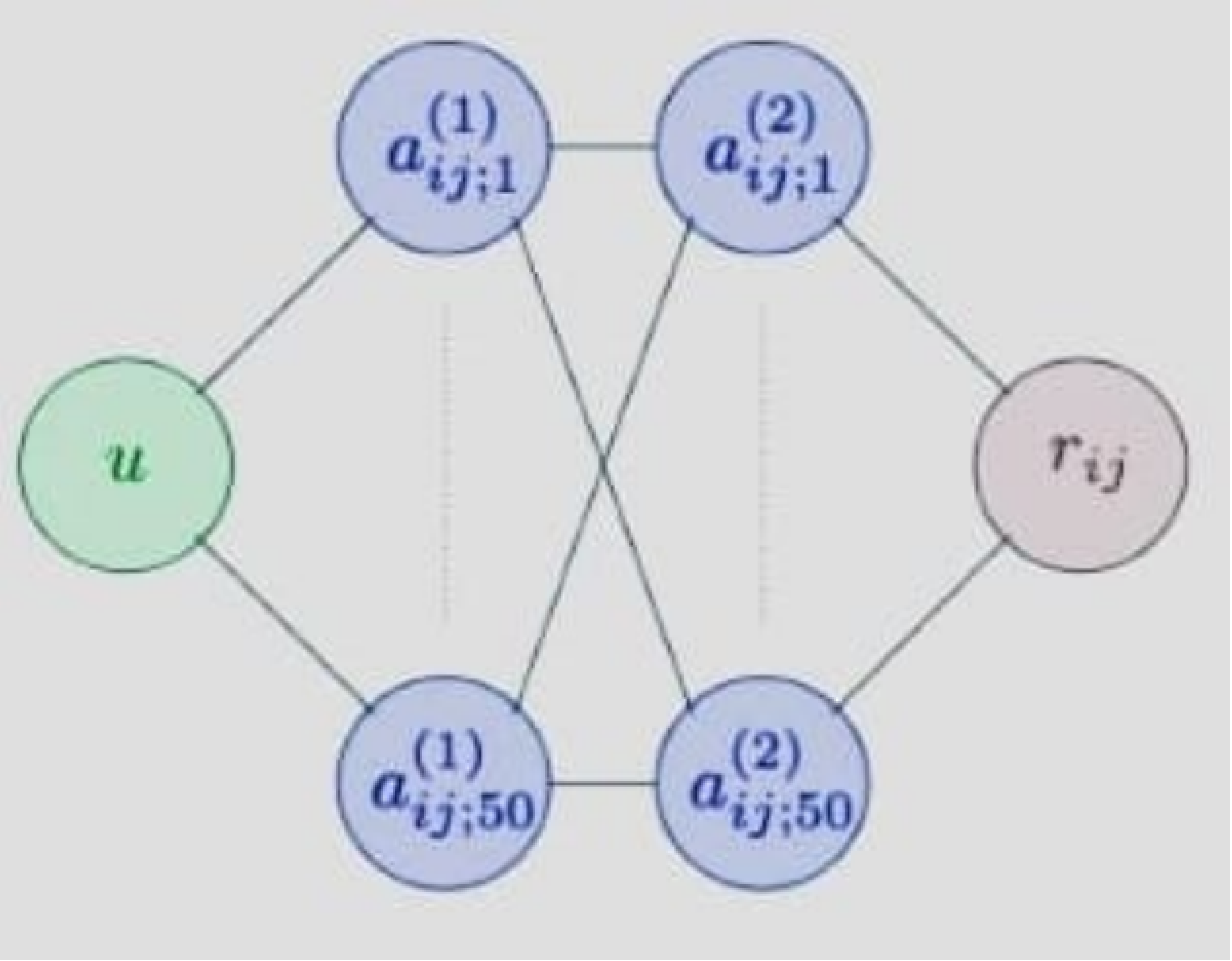
- holomorphic over the complex plane
- $[R_{ij}(0)] = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \equiv P :$ locality.
- $P \frac{d}{du} R(u)|_{u=0} = H :$ Hamiltonian.
- solves the Yang-Baxter equation.

We determine the functions $R_{ij}$ from these constraints.

- partly built into neural network architecture
- partly implemented by loss functions

# A Neural Network R-Matrix

Consider each scalar function $r_{ij}\left(u\right)$ in the $R-$matrix.




e.g. 16 non-zero entries in $\mathcal{R}$ $\Rightarrow$ 32 neural network functions.

functions entering $\mathcal{R}$ constrained through loss functions.

# Incorporating Holomorphy

Strategy: train on $u \in \Omega = (-1, 1)$

Choose a holomorphic activation function $h$ e.g.

$$\tanh(z)\,, \qquad \mathrm{swish}(z) = \frac{z}{1+e^{-z}}\,.$$

Then a function of the structure

$$\mathrm{a}(u) = \mathrm{h}\left(w_{jk} z_k + b_j\right)\,, \qquad w, b \in \mathbb{R}$$

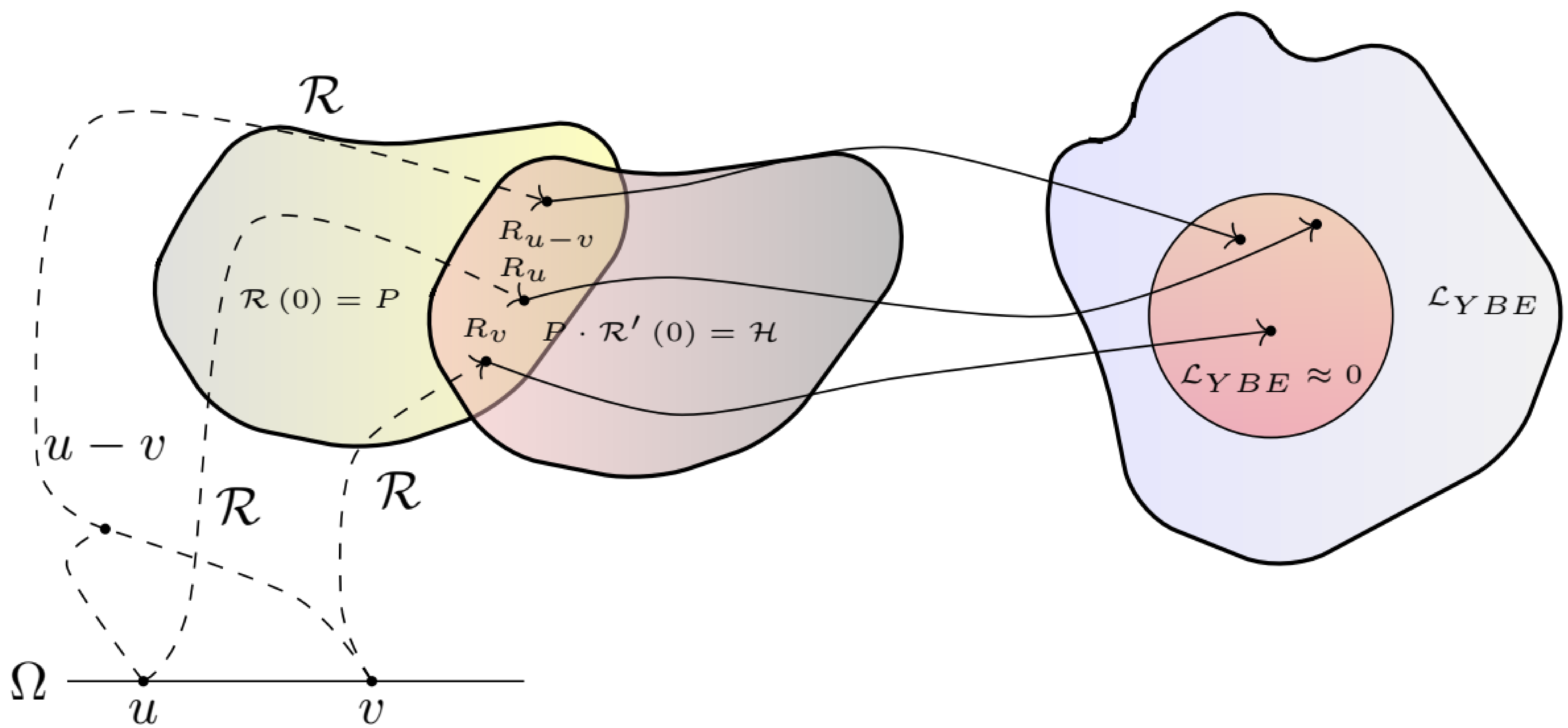
is holomorphic in u but real-valued for real inputs.

$$\mathrm{r}(u) = \mathrm{a}(u) + \mathrm{i}\,\mathrm{b}(u)$$

is holomorphic in u, without the reality constraint.

# A Neural Network R-Matrix

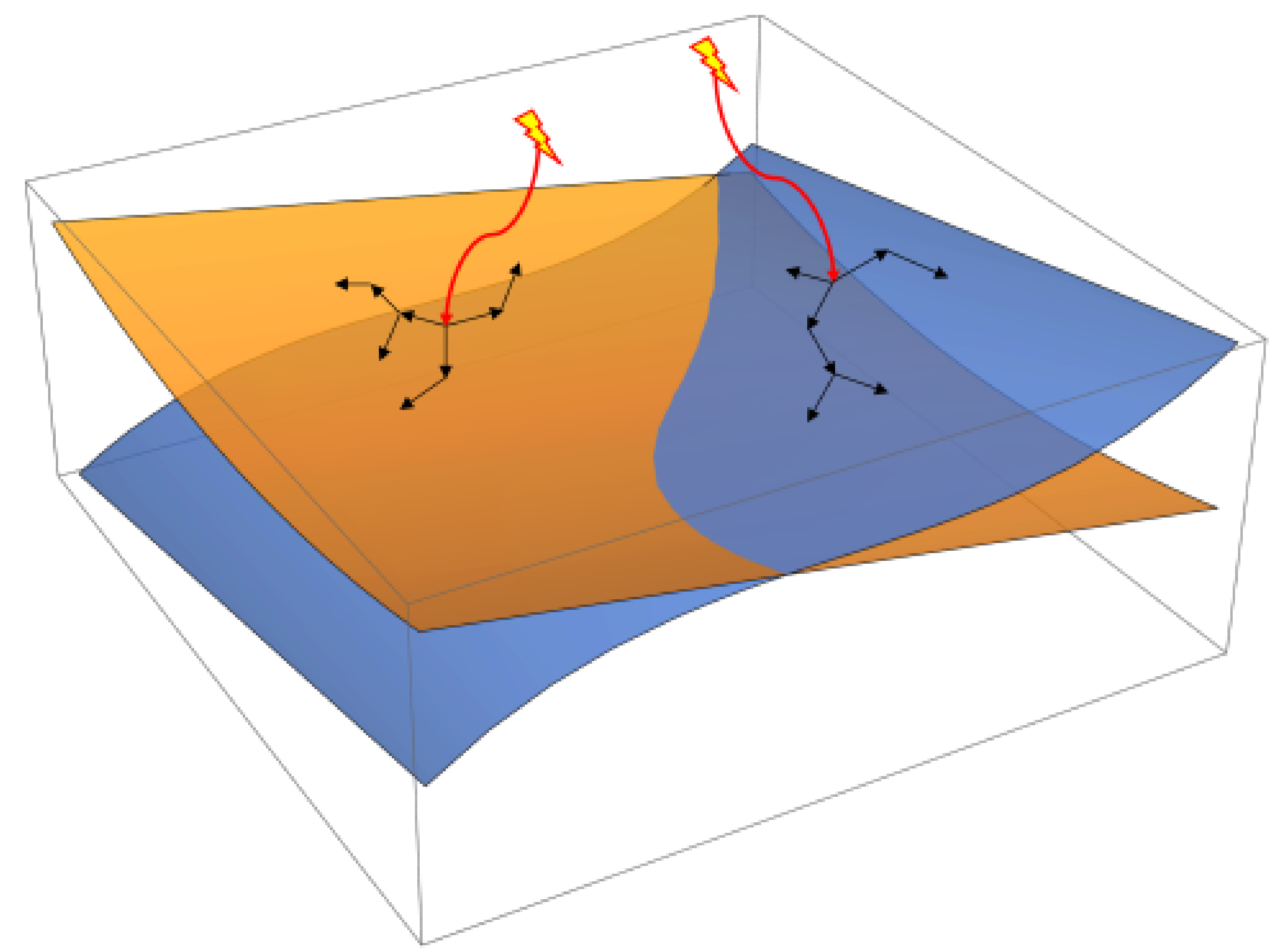
Overall, the structure looks like:




This is strongly reminiscent of the Siamese Network.

# The Loss Landscape

Another point of view: exploring function space.



- The box: R-matrices representable by the neural network.
- orange submanifold: YBE is obeyed
- blue submanifold: locality, form of Hamiltonian, . . . .
- idea: tune the network to scan intersections etc.

# A Quantum Integrable Playground

- physically distinct solutions completely known when $d = 2$.
- 'Typically' solutions are of the form

$$H = \begin{pmatrix} a_1 & 0 & 0 & d_1 \\ 0 & b_1 & c_1 & 0 \\ 0 & c_2 & b_2 & 0 \\ d_2 & 0 & 0 & a_2 \end{pmatrix}$$

- 6 additional 'outlier' classes also.
- R-matrix entries: can be exp, trig, Jacobi elliptic . . . .
- A good challenge for our proposed framework.

# The XYZ Model

Consider the XYZ Hamiltonian

$$H = \sum_{i\,\in\,\mathrm{sites}} J_x \sigma_i^x \sigma_{i+1}^x + J_y \sigma_i^z \sigma_{i+1}^z + J_z \sigma_i^z \sigma_{i+1}^z \,,$$

$\vec{\sigma}_i$ : Pauli matrices at site i .

The XYZ Hamiltonian admits the symmetric limits:

1. XXZ model: $J_x = J_y$
2. Heisenberg model: $J_x = J_y = J_z$ .

These are all integrable systems.

# The XYZ Model

The XYZ models are parametrized by real numbers $\eta$ and $m$:

$$J_x = 1 + \frac{\sqrt{m}}{2}\,\mathrm{sn}\,(2\eta|m)\,, \quad J_y = 1 - \frac{\sqrt{m}}{2}\,\mathrm{sn}\,(2\eta|m)\,, \quad J_z = \mathrm{cn}\,(2\eta|m)\,\mathrm{dn}\,(2\eta|m)$$

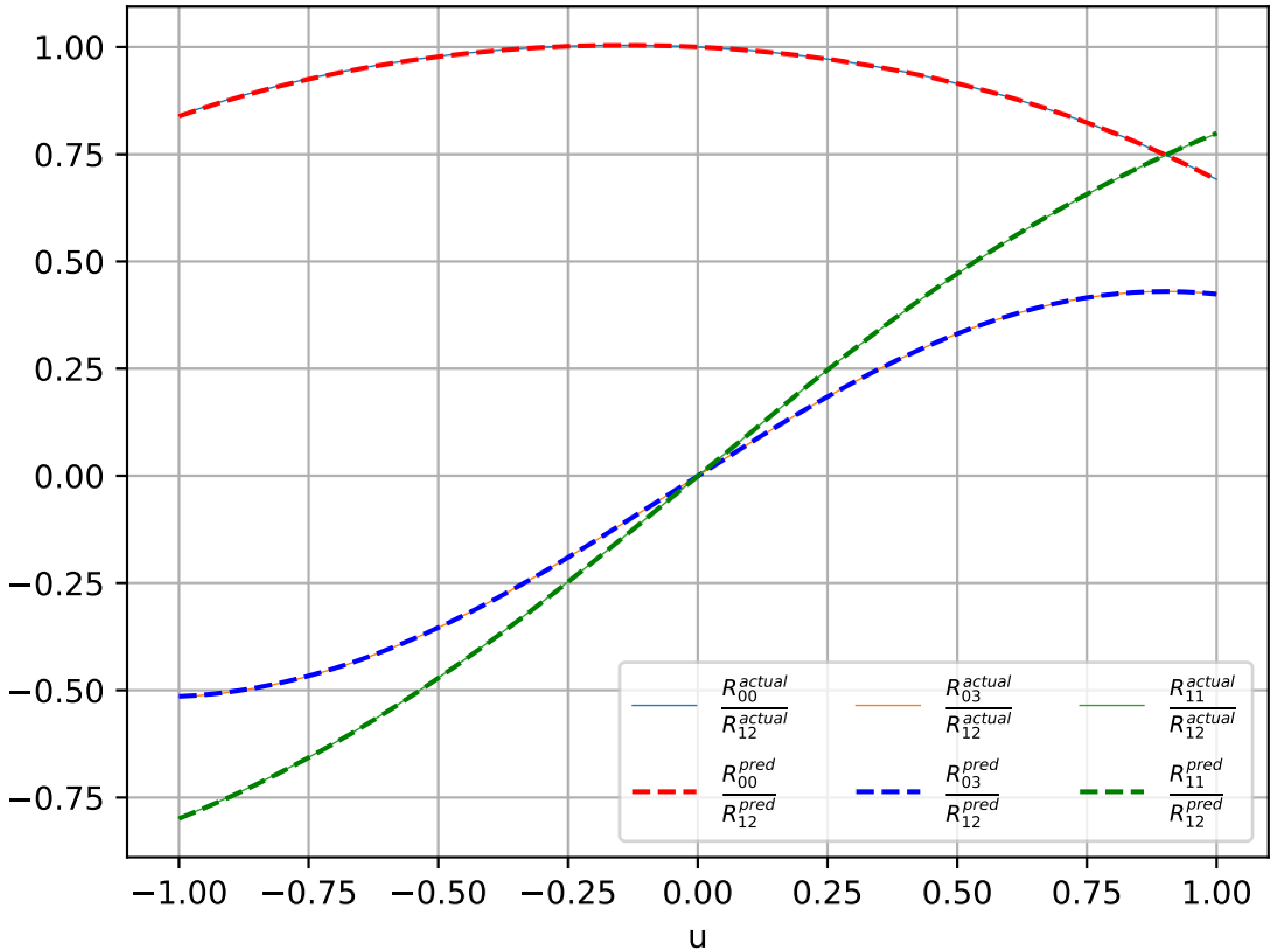
$$R(u) = \begin{pmatrix} a & 0 & 0 & d \\ 0 & b & c & 0 \\ 0 & c & b & 0 \\ d & 0 & 0 & a \end{pmatrix} \qquad \begin{aligned} a(u) &= \frac{\mathrm{sn}(2\eta + \omega u\,|\,m)}{\mathrm{sn}(2\eta\,|\,m)} \exp\left(-\frac{\mathrm{cn}(2\eta\,|\,m)\,\mathrm{dn}(2\eta\,|\,m)}{2\,\mathrm{sn}(2\eta\,|\,m)}\omega u\right)\,, \\ b(u) &= \frac{\mathrm{sn}(\omega u\,|\,m)}{\mathrm{sn}(2\eta\,|\,m)} \exp\left(-\frac{\mathrm{cn}(2\eta\,|\,m)\,\mathrm{dn}(2\eta\,|\,m)}{2\,\mathrm{sn}(2\eta\,|\,m)}\omega u\right)\,, \\ c(u) &= \exp\left(-\frac{\mathrm{cn}(2\eta\,|\,m)\,\mathrm{dn}(2\eta\,|\,m)}{2\,\mathrm{sn}(2\eta\,|\,m)}\omega u\right)\,, \\ d(u) &= \sqrt{m}\,\mathrm{sn}(\omega u\,|\,m)\,\mathrm{sn}(2\eta + \omega u\,|\,m) \exp\left(-\frac{\mathrm{cn}(2\eta\,|\,m)\,\mathrm{dn}(2\eta\,|\,m)}{2\,\mathrm{sn}(2\eta\,|\,m)}\omega u\right) \end{aligned}$$

We observe:

- Rich functional dependence in $u$.
- $m = 0 \quad \Rightarrow \quad J_x = J_y$ i.e. XXZ model.

# Training with an XYZ Target

Comparing with the analytic results




There is a precise match.

# Exploring the Landscape

Experiment 1: XYZ from XXZ.

The XXZ model is the $m = 0$ limit of the XYZ model.

Q: can we discover XYZ starting from XXZ.

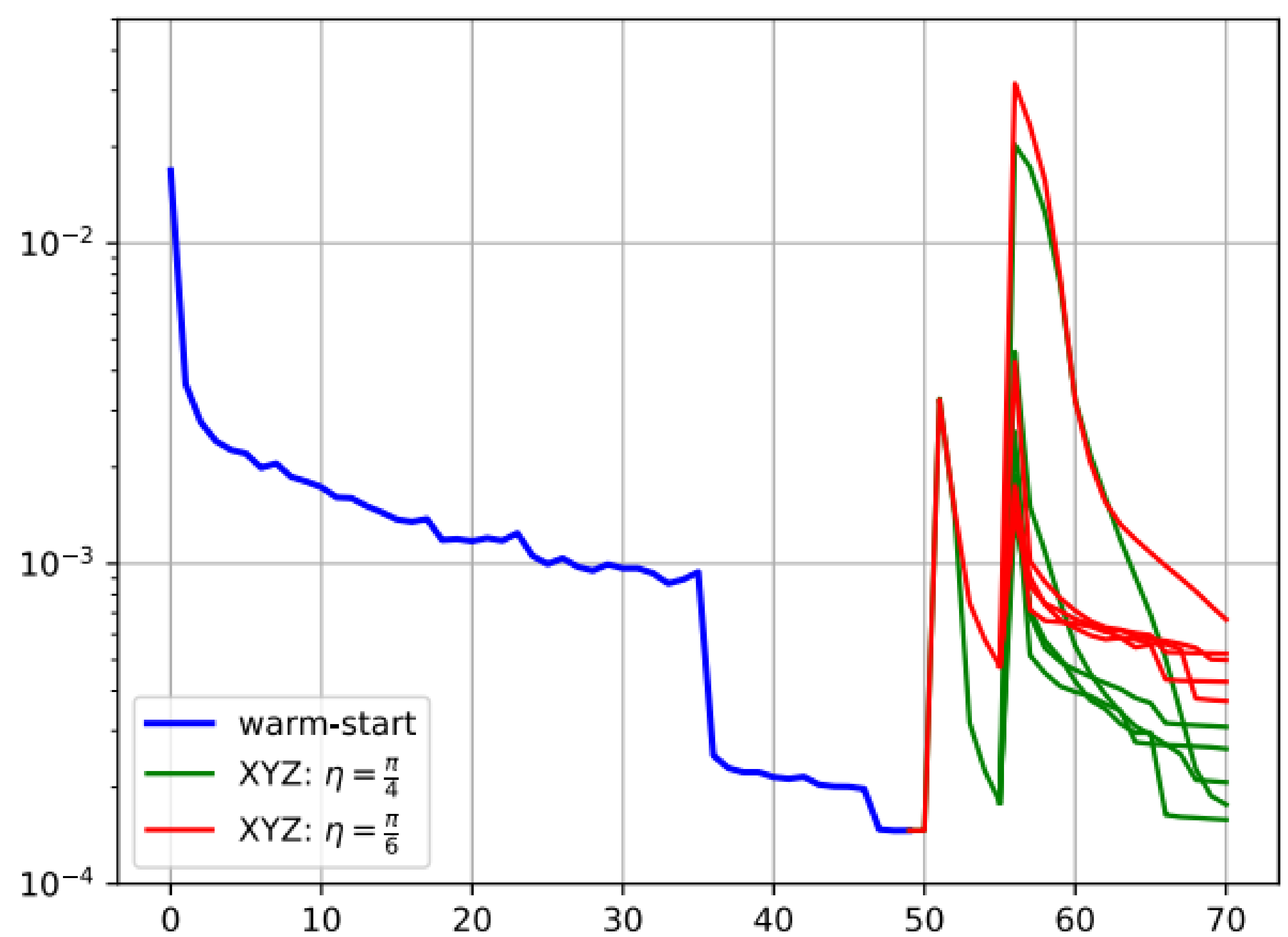
Framework:

1. Train the neural network with $\frac{\pi}{3}$ and $m = 0$ for 50 epochs.
2. Fine-tune to $\frac{\pi}{4}, \frac{\pi}{6}$ and $m = 0$ in 5 epochs.
3. This yields 3 XXZ models which are our starting points.
4. Randomly sample 5 non-zero values of $m$.
5. Train for 15 epochs with those target values.

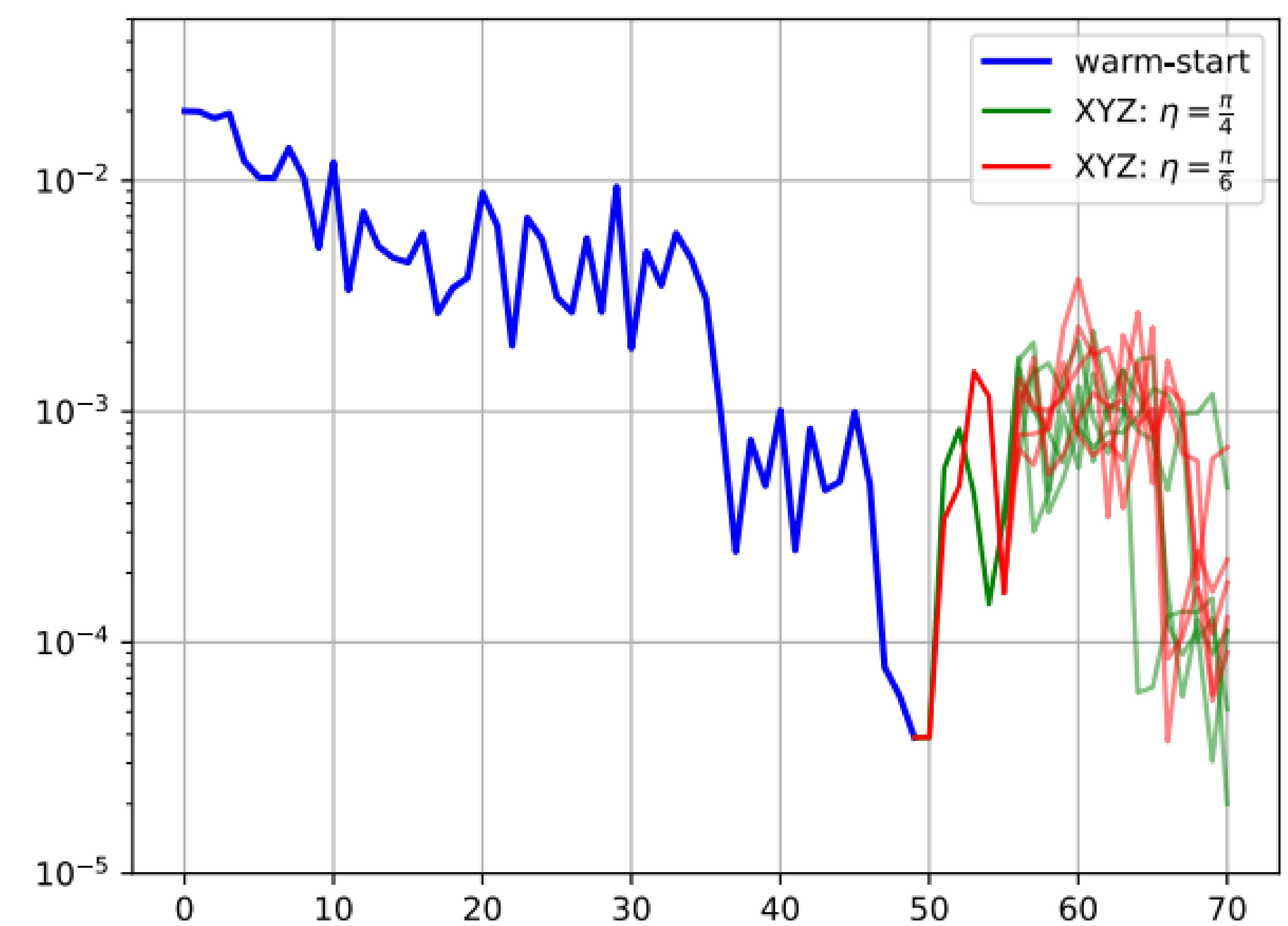
We find that we do converge to the correct XYZ models.

# Exploring the Landscape

Evolution of the Loss functions.




(a) Evolution of Yang-Baxter Loss




(b) Evolution of Hamiltonian Loss

The spikes correspond to resetting the target Hamiltonian.

# Exploring Model Families

Experiment 2: There exist integrable models

$$H = \begin{pmatrix} a_1 & 0 & 0 & 0 \\ 0 & b_1 & c_1 & 0 \\ 0 & c_2 & b_2 & 0 \\ 0 & 0 & 0 & a_2 \end{pmatrix}$$

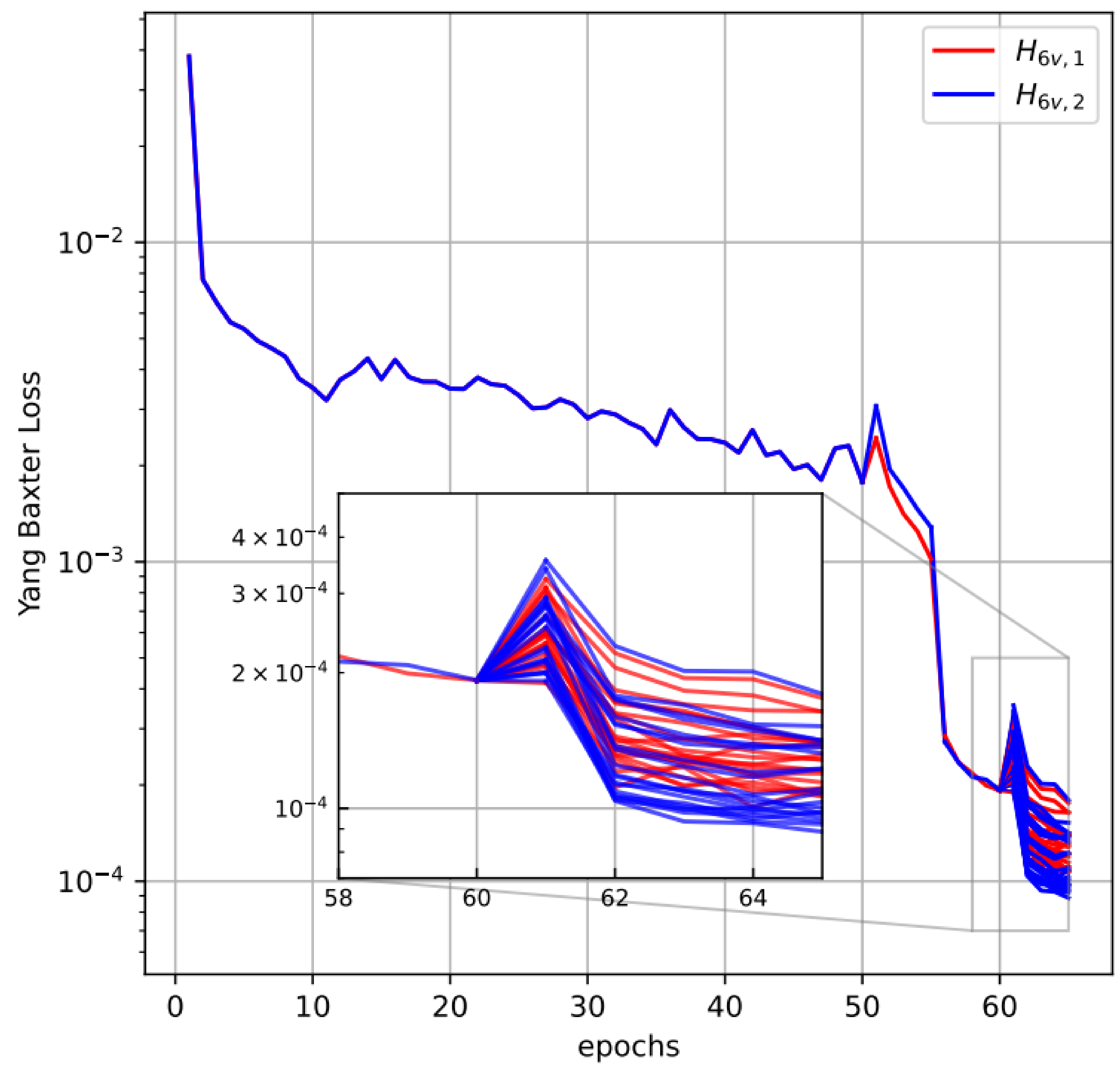
Two classes: $a_1 = a_2, \qquad a_1 + a_2 = b_1 + b_2$ .

Aim: discover these two classes.

1. train to an integrable Hamiltonian.
2. repel away from this Hamiltonian slightly.
3. train again, optimizing the Yang-Baxter loss and locality.
4. no target Hamiltonian is given.
5. idea: let network converge to *any* nearby Hamiltonian.

# Exploring the Landscape

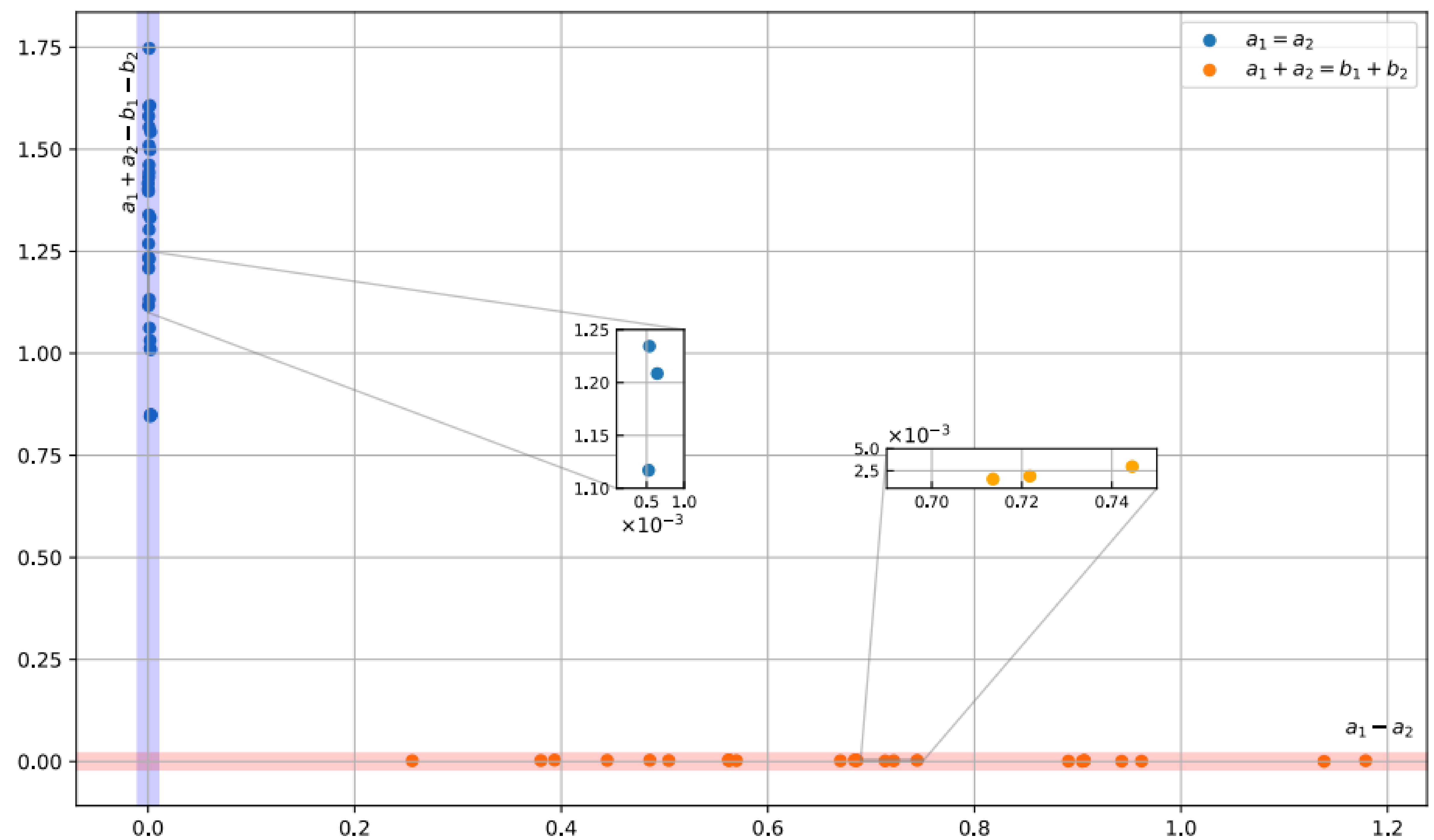
We repeat this exercise several times.




The rise in Yang-Baxter loss occurs when repulsion is turned on.

# Exploring the Landscape

## Visualizing the Learnt Hamiltonians



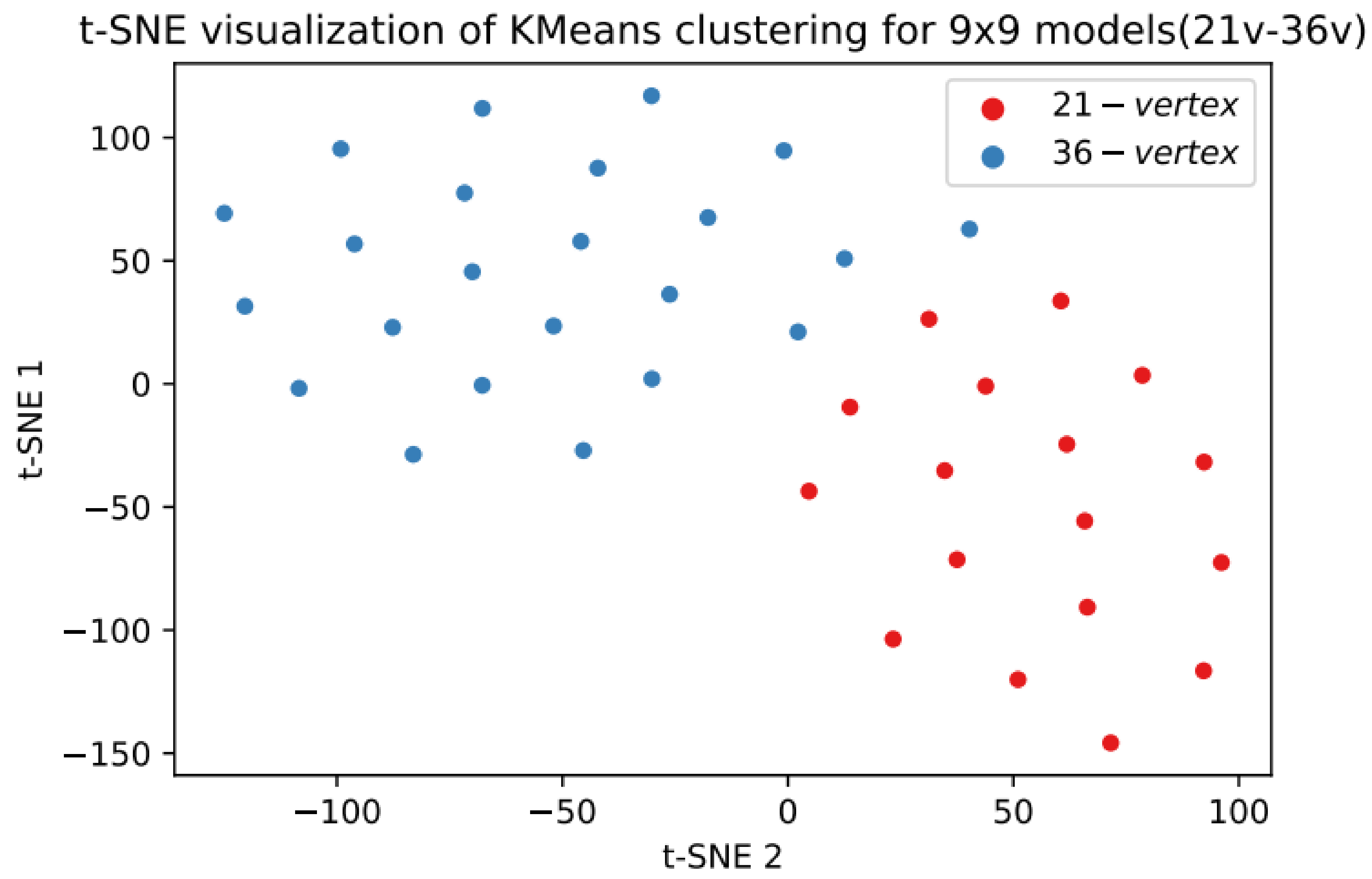

We find separation into two classes.

# Summary

- Neural Networks are universal approximators.
- Intuitive for finding functions that obey constraints.
- We can use them to solve the Yang Baxter equation.
- We can recover all 2d difference form solutions.
- Strategies for exploring the space of integrable theories.
- TODO: finding analytic solutions.
- TODO: finding new solutions (3d).
- TODO: non-difference form.
- and much more ...

# A 3d Out-Tro

- Now let the local Hilbert space at each site be $V \sim \mathbb{C}^3$.
- The $R-$ matrix is $9 \times 9$, but the same framework as above.
- Some preliminary results for 21 and 36 vertex models.




- In progress . . .