The R-maTrIx Net

Shailesh Lal

shaileshlal at bisma dot cn

29 May 2023

BIMSA

SL, Suvajit Majumder, Evgeny Sobko 2304.07247

Artificial Intelligence

> *Artificial intelligence is the science of making machines do things that would require intelligence if done by humans.*
>
> *Marvin Minsky*

In Physics:

- phase classification of condensed matter systems
- construction of string vacua for phenomenology/cosmology
- classifying signals in colliders

Neural Networks are an important framework for these analyses.

Focus exclusively on them here: "AI $\Leftrightarrow$ Neural Network"

Why AI in Physics?

- A novel methodology of thinking about physical systems
- Conventional: provide an algorithm to generate solutions
- AI: provide data about what a solution is and let the AI infer how to arrive at it.
- Refinement: provide constraints on the solution instead.
- In many cases, outperforms conventional methods:
  - Stockfish vs AlphaZero
  - Computing metrics on Calabi Yau spaces
- Conversely: Use math phys tools to analyze Neural Nets

So what is a neural network?

Neural Networks

An Artificial Neuron

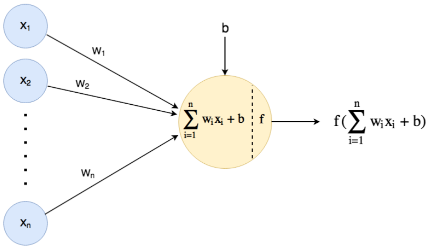


Figure: A single neuron: $\mathcal{O} = f\left(\vec{w} \cdot \vec{x} + b\right)$

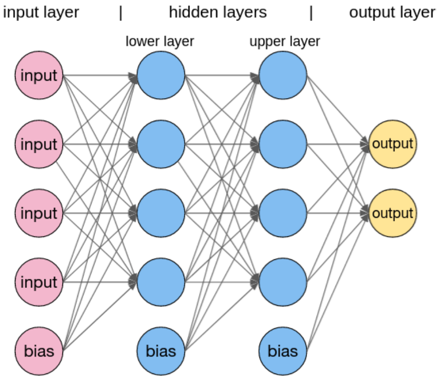
w and b are tunable parameters: weights & biases

f is a typically non-linear function: activation function

The activation allows neural nets to learn non-linear functions.

Neural Networks

A neural network is a set of neurons arranged in layers.



The output of each layer is fed to the next layer.

This yields a hierarchical structure:

layers process previous layer's output into the next layer's input.

Neural Networks

This neural network is a series of functional transformations.

- Take the inputs x and construct

$$a_k^{(1)} = w_{kj}^{(1)} x_j + b_j^{(1)}\,, \quad z_k^{(1)} = h^{(1)}\left(a_k^{(1)}\right)\,.$$

  The $z_k^{(1)}$ are the outputs of layer 1.

- Similarly for layer 2

$$a_k^{(2)} = w_{kj}^{(2)} z_j^{(1)} + b_j^{(2)}\,, \quad z_k^{(2)} = h^{(2)}\left(a_k^{(2)}\right)\,.$$

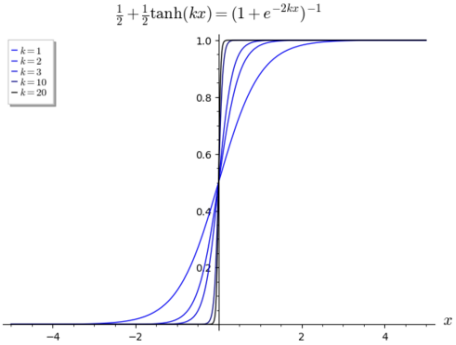
- Iterate this structure over all layers and collect the output

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

By tuning w, b we can change the output function.

Universal Approximation

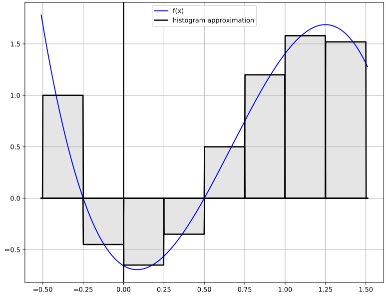
Extreme example: the non-linearity is a step function.



Note: limit of the tanh function.

Universal Approximation

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



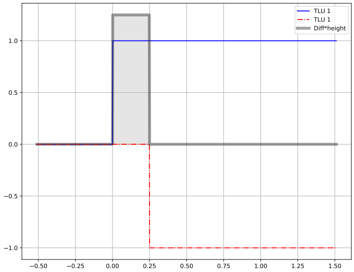
$$f(x) \approx f_0(x) = \bar{f}_j\,, \quad x \in [x_{j-1}, x_j]\,, \quad j = 1, 2, \ldots, N$$

Larger $N \Rightarrow$ better approximation.

Universal approximation

This can be realized through the step function

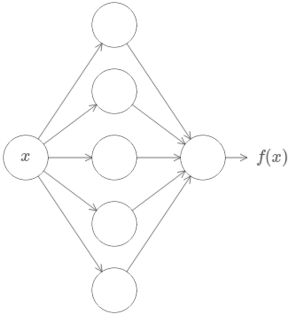
$$f_s(x) = \Theta(wx + b) \,, \qquad s = -\frac{b}{w} \,.$$



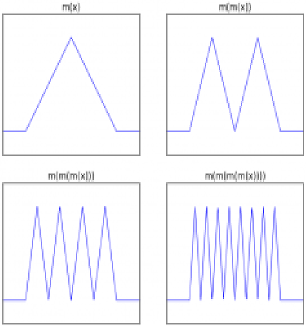
define: $f_{s_1,s_2,h} = h\left(f_{s_1} - f_{s_2}\right) = \begin{cases} h & : \quad x \in (s_1, s_2) \\ 0 & : \quad \text{otherwise} \end{cases}$

Universal Approximation

A single layer of neurons can approximate any function.



(a) Single Layer Network



(b) Sawtooth Function

Depth is crucial to learning heirarchical structures efficiently.

Training a Neural Network

The classic framework is supervised learning

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

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

A typical network has $\approx 10^6$ parameters. huge search space

- Suppose each parameter can take 10 values.
- N such parameters $\Rightarrow 10^N$ possible solutions.

The only way forward is to search locally for $\{w, b\}$.

Training a Neural Network

We search locally via Gradient Based Optimization

- Define a loss function $\mathcal{L}(w, b)$

$$\mathcal{L}(w, b) = \frac{1}{N}\sum_i (y_i - y_{NN}(x_i))^2 \,.$$

- Initialize $\{w, b\}$ at w_o
- Step iteratively in the direction of steepest descent

$$w_{j+1} - w_j \sim -\nabla_w \mathcal{L}(w_j) \,.$$

- Stop when $\mathcal{L}(w) \approx 0$.

Conceptually this is no different to fitting by least squares.

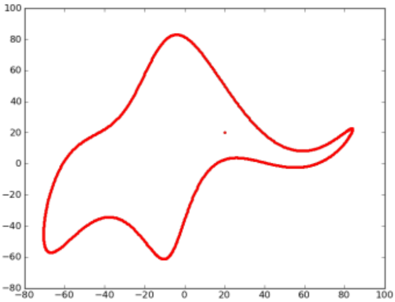
BUT

universal approximation lets us scan in a huge function space.

Overfitting

With four parameters I can fit an elephant, and with five I can make him wiggle his trunk.

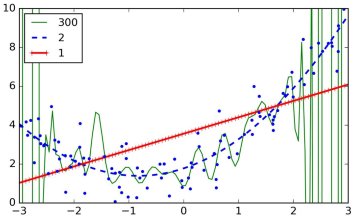
John von Neumann



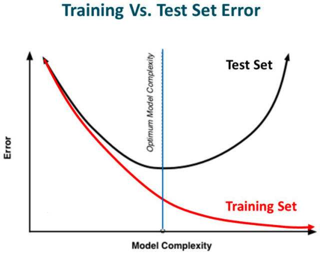
"Drawing an elephant with four complex parameters", Mayer, Khairy, Howard, AmJPhys 78, 648 (2010)

Overfitting

A more complicated model has more parameters, tends to learn spurious features in the data: Overfitting



(a) Polynomial fitting noisy data with degree 1, 2, 300.



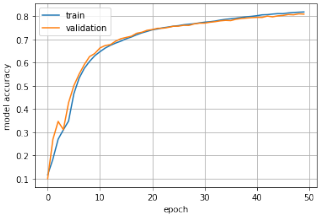
(b) Overfitting vs Model Complexity

Conversely, tends to perform poorly when evaluated on unseen data: Test data.

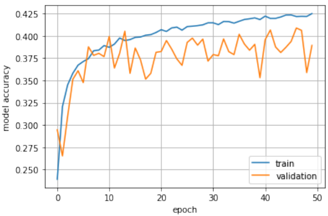
Tackling Overfitting

The trained model should perform equally well on unseen data.

- Partition data into disjoint Train/Validation/Test sets
- Train the model by optimizing parameters on the Train set
- Check the model performance is comparable on the Val set



(a) This is okay



(b) Overfitting

Finally, evaluate trained model on test set.

Today

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

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

- Use neural networks to scan across function space.
- Determine the target function by gradient descent.
- It is helpful to regard loss functions in a new light.

Rethinking Loss Functions

To get intuition about the effect of a loss function we consider what would happen at its optimum.

The mean square error cost function:

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

Minimized at:

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

Training this network is equivalent to searching for functions that obey the constraints

$$f(x_n, w) = y_n \quad \forall \quad n\,.$$

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

Siamese Neural Networks

Origins: Signature verification for Bank cheques

- $\exists$ hundreds of thousands of classes (i.e. customers)
- A signature has to be identified to the correct user

Problems

- Many classes, only 1-2 example signatures for each class.
- New customers always being added. Retrain classifier?

Instead train Siamese Network

- compare signature on record with signature given
- if the two signatures are similar, process the cheque

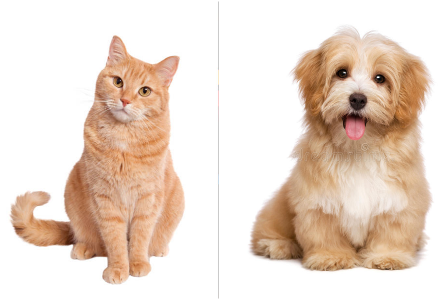
$\therefore$ a constraint on the prediction function

Siamese Nets for Signature Verification Bromley, Bentz, Bottou, Guyon, LeCun, Moore, Säckinger, Shah

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.

Fei-Fei Li

Machine Learning Similarity

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

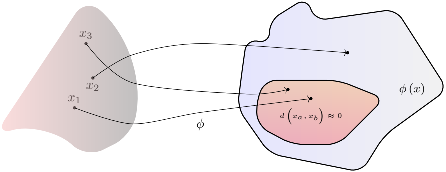
In essence the toddler learns the equivalence relations

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

This is now known to be an exceptionally robust framework.

Siamese Neural Networks

A machine learning framework to quantify similarity.



- Learn a map from Data 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 $\mathbb{R}^d$

Machine Learning Similarity

Recasting classification as Similarity

- Classification: organize N elements into k classes
- Similarity: Let $x_1 \in \mathcal{C}_1$ and $x_2 \in \mathcal{C}_2$
  - $x_1 \sim x_2$ if $\mathcal{C}_1 = \mathcal{C}_2$
  - $x_1 \nsim x_2$ if $\mathcal{C}_1 \neq \mathcal{C}_2$
- Recasting Data for similarity: create pairs (x_1, x_2)
  - Label (x_1, x_2) with $y = 1$ if $x_1 \sim x_2$
  - Label (x_1, x_2) with $y = 0$ if $x_1 \nsim x_2$
- Hence, organize ${}^{N}C_2$ elements into two classes

$\Rightarrow$ Data looks much bigger!

Contrastive Losses

The loss function for Siamese Nets encodes similarity

It is evaluated on pairs of 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 Loss function is the contrastive loss

$$E(w) = \sum_{(x_1,x_2)} y\, d_w^2(x_1, x_2) + (1-y) \max\left(1 - d_w^2(x_1, x_2), 0\right)$$

Here

$$d_w^2(x_1, x_2) = \left(\phi_w(x_1) - \phi_w(x_2)\right) \cdot^2 ,$$

'·' is the usual dot product.

Contrastive Losses

Intuition for the Contrastive Loss $\Rightarrow$ When is it zero?

- $y = 0 \quad \Rightarrow d_w^2\left(x_1, x_2\right) > 1$
- $y = 1 \quad \Rightarrow d_w^2\left(x_1, x_2\right) = 0$

Hence ϕ_w must be such that

- If $x_1 \sim x_2$, i.e. $y = 1$, ϕ_w maps them close together
- If $x_1 \nsim x_2$, i.e. $y = 0$, ϕ_w maps them far apart

As a result, under this map

- Dataset breaks up into disjoint clusters
- Each cluster is made up of similar data

Importantly we can infer properties of new data by seeing which cluster it is mapped to.

Summary of Neural Networks

- Neural networks are a powerful framework capable of expressing complex relationships in data.
- Universal Approximation properties: tuning neural network parameters $\Rightarrow$ 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.

Quantum Integrability

Consider a periodic spin chain on 3 sites. The Hilbert space is

$$\mathbb{V} = V_1 \otimes V_2 \otimes V_3\,, \qquad V_i \sim V = \mathbb{C}^2\,.$$

The Hamiltonian is a sum of nearest-neighbour interactions

$$H = \sum_{i=1}^{3} H_{i,i+1}\,.$$

For example

$$H = \sum_{i=1}^{3} \sum_{\alpha} J^\alpha S_i^\alpha S_{i+1}^\alpha\,,$$

$\alpha = \{x, y, z\}$ and S_i^α are Pauli matrices.

This is the XYZ model. When $J_x = J_y$ we get the XXZ model.

Quantum Integrability

These systems are characterized by an R-matrix $R(u)$

- holomorphic in u
- $R(0) = P$, the permutation matrix.
- $P \frac{d}{du} R(u)|_{u=0} = H$, the Hamiltonian.
- Higher charges $\mathbb{Q}_3, \mathbb{Q}_4, \ldots$ also encoded in $R(u)$
- $\{H, \mathbb{Q}_3, \mathbb{Q}_4, \ldots\}$ all commute with each other.
- Hence, an infinite number of conserved charges.

The R-matrix solves the Yang-Baxter equation

$$R_{ij}(u-v)R_{ik}(u)R_{jk}(v) = R_{jk}(v)R_{ik}(u)R_{ij}(u-v)$$

Questions

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.

Quantum Integrability

As a concrete example, consider a family of integrable models

$$H = \begin{pmatrix} a_1 & 0 & 0 & 0 \\ 0 & b_1 & c_1 & 0 \\ 0 & c_2 & b_2 & 0 \\ 0 & 0 & 0 & a_2 \end{pmatrix} \leftrightarrow \quad R(u) = \begin{pmatrix} R_{00} & 0 & 0 & 0 \\ 0 & R_{11} & R_{12} & 0 \\ 0 & R_{21} & R_{22} & 0 \\ 0 & 0 & 0 & R_{33} \end{pmatrix}.$$

where $a_2 = a_1$ or $a_2 = b_1 + b_2 - a_1$ and $a, b, c \in \mathbb{C}$.

Structurally

$$R_{00,33} \sim e^{bu}(\cosh u + \sinh u)$$
$$R_{11,22} \sim e^{bu} \sinh u$$
$$R_{12,21} \sim e^{bu}$$

de Leeuw, Pribytok, Ryan

We will experiment on this system to showcase our approach.

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 Solver

Strategy: Learning holomorphic functions training with real u.

Restrict to the real interval $u \in \Omega = (-1, 1)$

Choose a holomorphic activation function, e.g. tanh

Decompose: $R_{ij}(u) = a_{ij}(u) + i\, b_{ij}(u)$

By construction R_{ij} will be holomorphic.

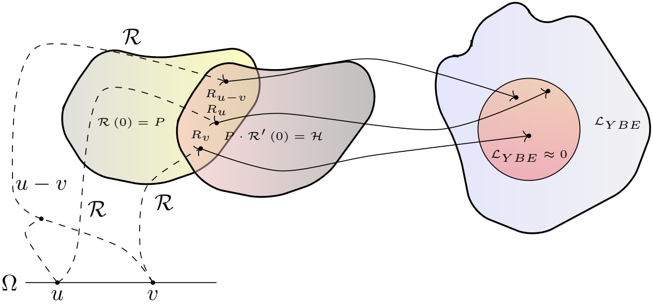
Note: a_{ij} and b_{ij} are $Re(R_{ij})$ and $Im(R_{ij})$ on the real line only.

Each a_{ij} and b_{ij} is modeled by an individual neural network.

These neural networks $\Rightarrow$ R matrix and loss functions imposed.

A Neural Network Solver

Overall, the structure looks like:



This is strongly reminiscent of the Siamese Network.

Constraining the Hamiltonian

There are two ways of constraining the Hamiltonian

1. by value: $a_1 = 0.8\,, \quad \mathcal{L} = |a_1 - 0.8|\ .$
2. by condition: $a_1 = a_2\,, \quad \mathcal{L} = |a_1 - a_2|\ .$

Note: it is also possible to repel away by imposing $a_1 \neq 0.8\,,$

However: in practice, difficult to control. Used sparingly.

We can also impose additional constraints such as Hermiticity.

$$\mathcal{L} = \sum_{ij} \left| H_{ij} - H^{\dagger}_{ij} \right|$$

Training with an XYZ Target

The XYZ model has two parameters η, m.

We set $\eta = \pi/3$ and $m = 0.6$

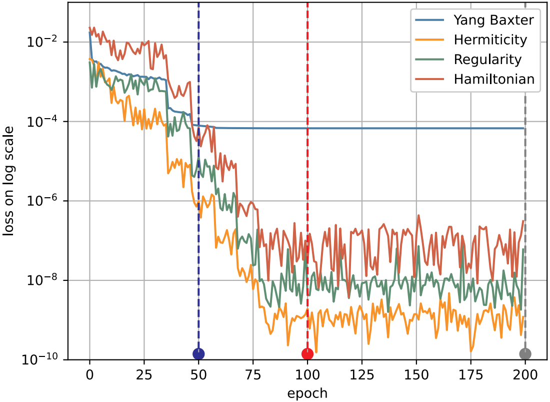
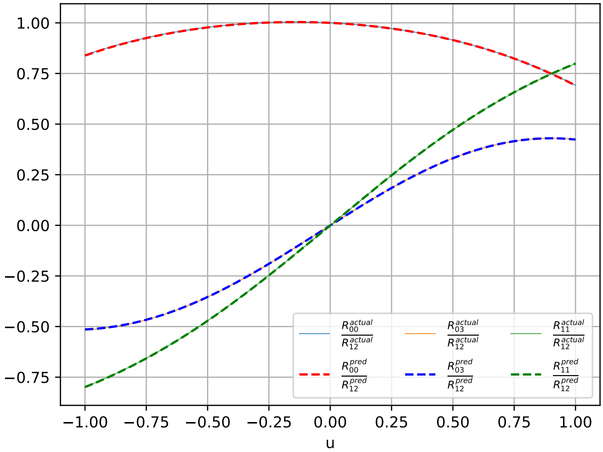


Figure: Evolution of the loss functions

Training with an XYZ Target

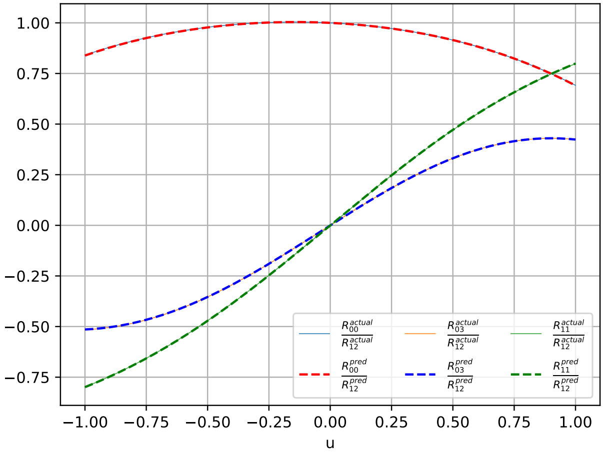
Comparing with the analytic results



There is a precise match.

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.

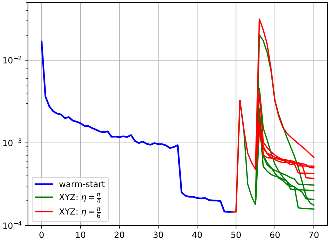
Framework:

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.

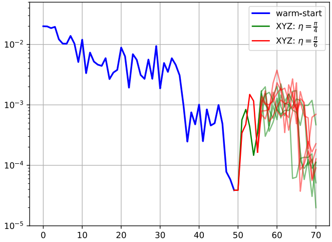
We find that we do converge to the correct XYZ models.

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 the Landscape

Experiment 2: Exploring 6-vertex models.

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

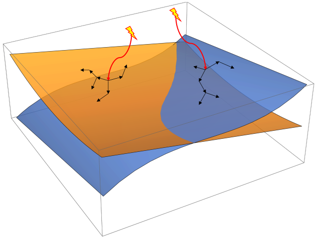
Fall into two classes:

1. $a_1 = a_2$
2. $a_1 + a_2 = b_1 + b_2$

Aim: discover these two classes.

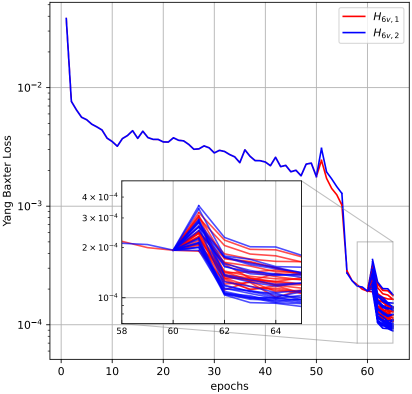
Exploring the Landscape

Strategy: Exploring 6-vertex models.



1. train to an integrable Hamiltonian.
2. repel away from this Hamiltonian slightly over 1 epoch.
3. train again, optimizing the Yang-Baxter loss and locality.
4. no target Hamiltonian is given.

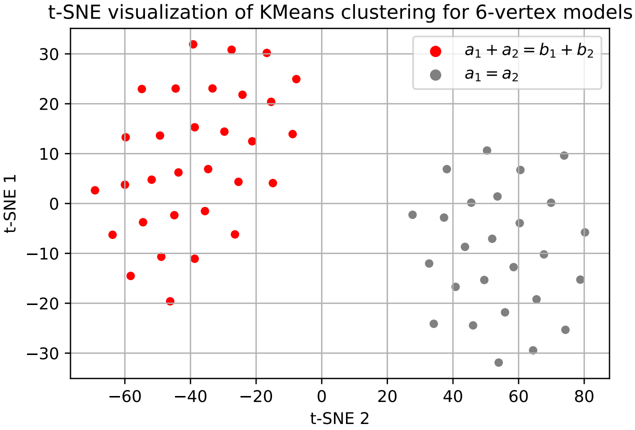
Exploring the Landscape



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

Exploring the Landscape

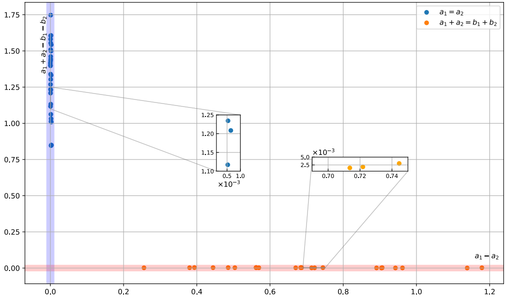
Visualizing the Learnt Hamiltonians



We find separation into two classes.

Exploring the Landscape

Visualizing the Learnt Hamiltonians



We find separation into two classes.

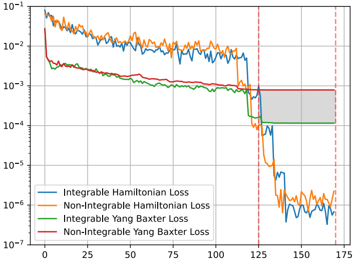
Summary

- 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 ...

Thank You

Integrable vs Non-Integrable

Loss functions



There is roughly an order of magnitude separation in Yang-Baxter loss.