Artificial Neural Network 2431

An illustration of learning process

Sparisoma Viridi[1]

[1]Nuclear Physics and Biophysics Research Division, Faculty of Mathematics and Natural Sciences

[1]Institut Teknologi Bandung, Bandung 40132, Indonesia

20230410-v4| https://doi.org/10.5281/zenodo.7839422

Outline

• Architecture	3
• Activation functions	6
• Feedforward	10
• Backprogpagation	14
• TensorFlow playground	17
• Assignment	27

Architecture

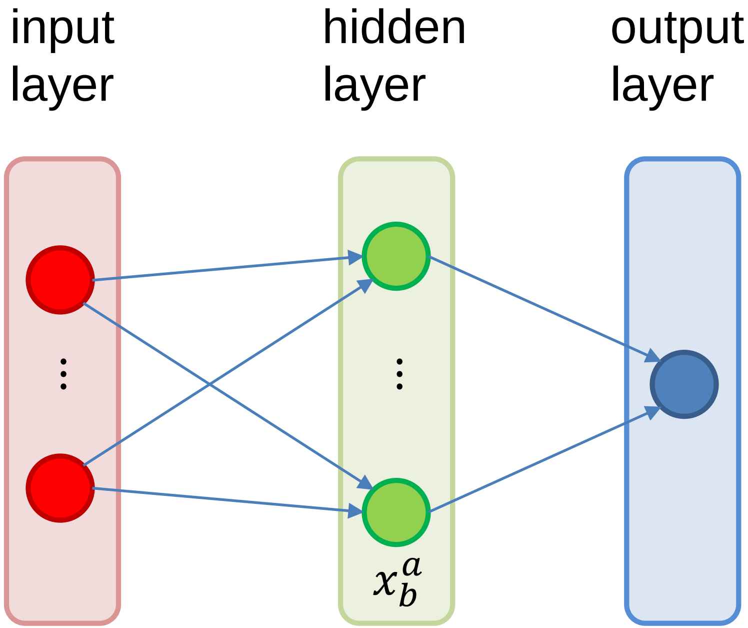
Neuron and layer types

Color

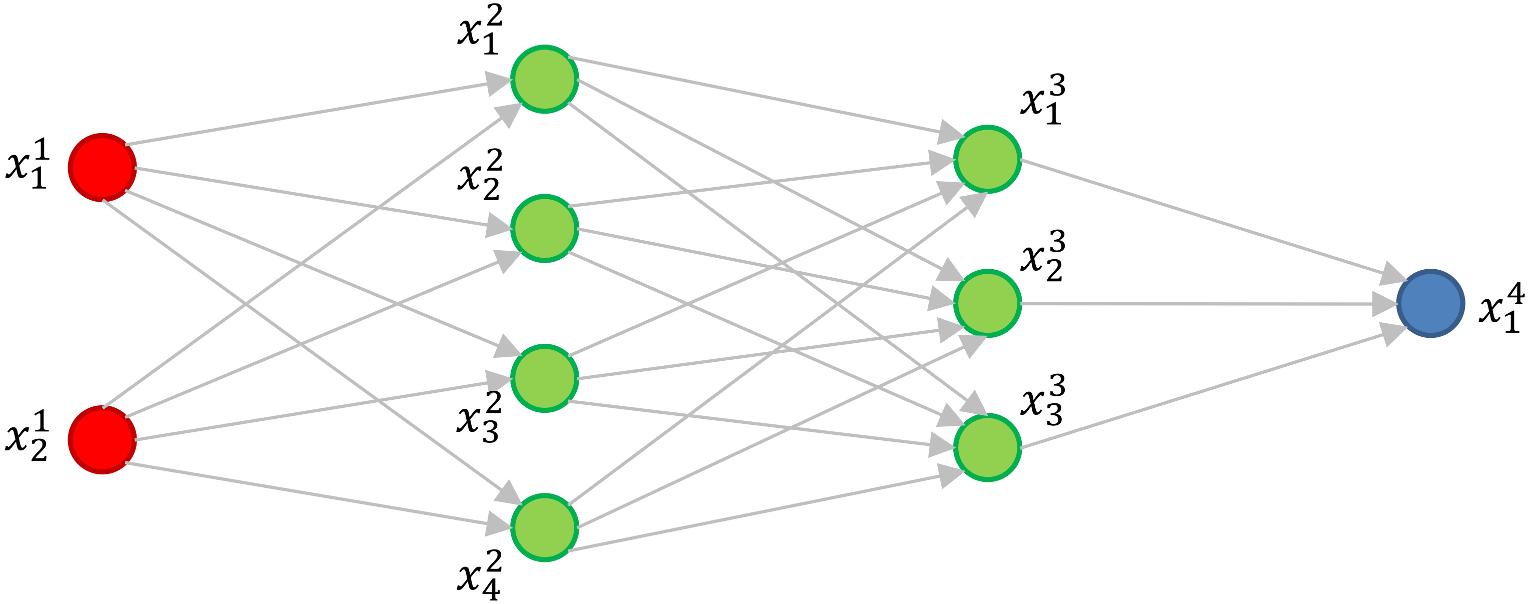
- 🔴 input neuron
- 🟢 hidden neuron
- 🔵 output neuron

Notation

- x_b^a is b-th neuron in a-th layer

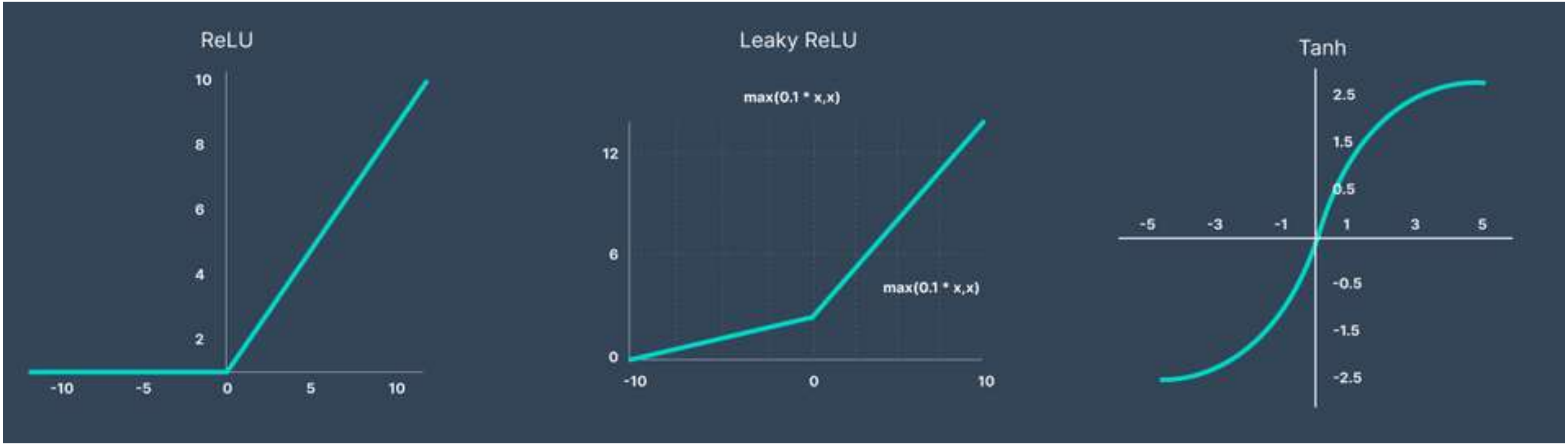


2-4-3-1 network



Peter Bartus Leonard Meijer, “Neural Network Applications in Device and Subcircuit Modelling for Circuit Simulation”, PhD Thesis, Technische Universiteit Eindhoven, Eindhoven, Netherlands, May 1996, p 10, url https://doi.org/10.6100/IR459139.

Activation functions

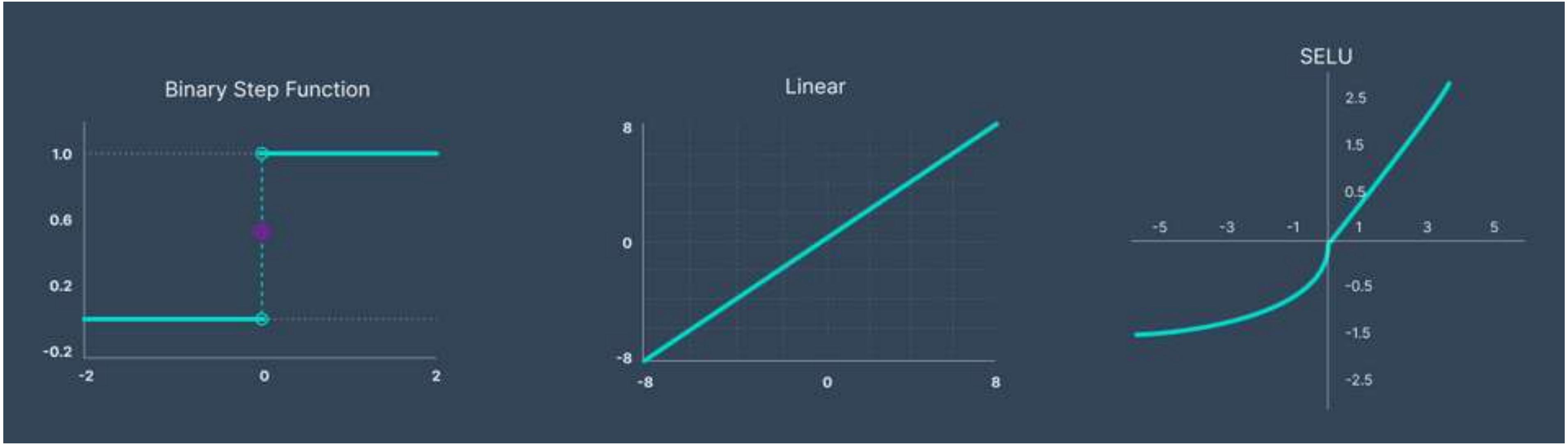


$$g(x) = \max(0, x)$$

$$g(x) = \max(0.1x, x)$$

$$g(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Pragati Baheti, “Activation Functions in Neural Networks [12 Types & Use Cases]”, V7Labs, 2 Mar 2023, url https://www.v7labs.com/blog/neural-networks-activation-functions [20230418].

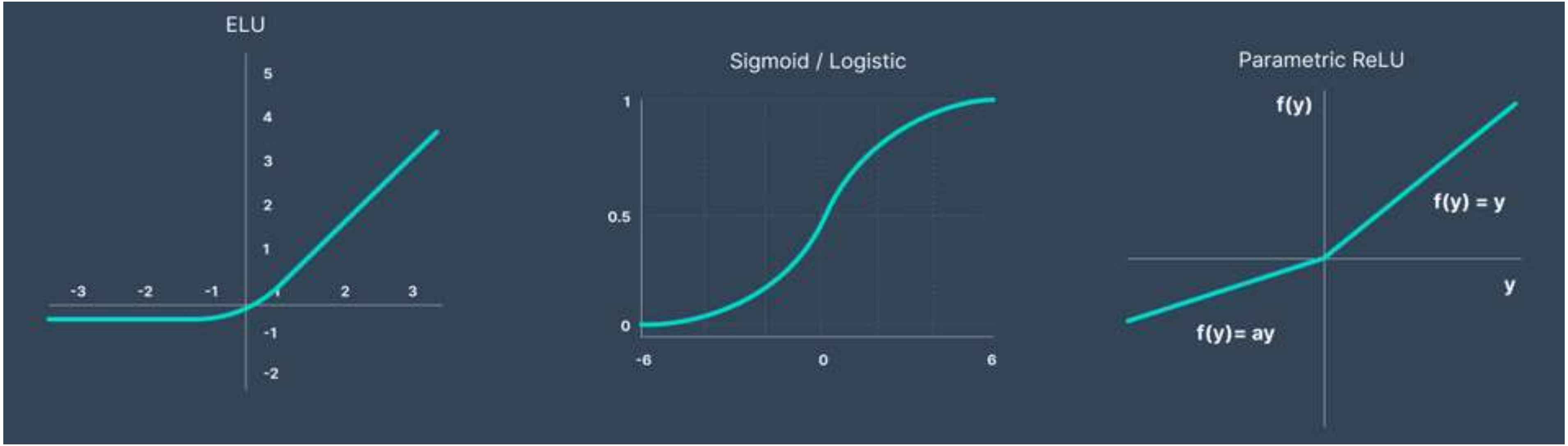


$$g(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$$

$$g(x) = x$$

$$g(x) = \lambda \begin{cases} \alpha(e^{x} - 1), & x < 0 \\ x, & x \geq 0 \end{cases}$$

Pragati Baheti, “Activation Functions in Neural Networks [12 Types & Use Cases]”, V7Labs, 2 Mar 2023, url https://www.v7labs.com/blog/neural-networks-activation-functions [20230418].



$$g(x) = \begin{cases} \alpha(e^x - 1), & x < 0 \\ x, & x \geq 0 \end{cases}$$

$$g(x) = \frac{1}{1 + e^{-x}}$$

$$g(x) = \max(ax, x)$$

Pragati Baheti, “Activation Functions in Neural Networks [12 Types & Use Cases]”, V7Labs, 2 Mar 2023, url https://www.v7labs.com/blog/neural-networks-activation-functions [20230418].

Feedforward

Neuron in a layer as vector

- From figure of 2-4-3-1 network there are following vectors representing each layers

$$\overrightarrow{x_1} = \begin{bmatrix} x_1^1 \\ x_2^1 \end{bmatrix} \qquad \overrightarrow{x_2} = \begin{bmatrix} x_1^2 \\ x_2^2 \\ x_3^2 \\ x_4^2 \end{bmatrix} \qquad \overrightarrow{x_3} = \begin{bmatrix} x_1^3 \\ x_2^3 \\ x_3^3 \end{bmatrix} \qquad \overrightarrow{x_4} = \begin{bmatrix} x_1^4 \end{bmatrix}$$

Artem Oppermann, “Activation Functions in Deep Learning: Sigmoid, tanh, ReLU”, 14 Oct 2021, url https://artemoppermann.com/activation-functions-in-deep-learning-sigmoid-tanh-relu/ [20230418].

Weight as matrix

- Weights connected two layers from previous network can be formulated as follow

$$W_{21} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} \quad W_{21} = \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \end{bmatrix} \quad W_{32} = \begin{bmatrix} w_{11} & w_{12} & w_{13} \end{bmatrix}$$

Artem Oppermann, “Activation Functions in Deep Learning: Sigmoid, tanh, ReLU”, 14 Oct 2021, url https://artemoppermann.com/activation-functions-in-deep-learning-sigmoid-tanh-relu/ [20230418].

Information propagation

- Between two successive layers

$$\vec{x}_2 = g(W_{21}\vec{x}_1)$$

$$\vec{x}_3 = g(W_{32}\vec{x}_2)$$

$$\vec{x}_4 = g(W_{43}\vec{x}_3)$$

with g is activation function

- Final result $\vec{x}_4$ then compared to observed value $\vec{y}$

Backpropagation

Error and weight change

- It can be defined as follow

$$\varepsilon = |\vec{x}_4 - \vec{y}|^2$$

- Then, change of weight

$$\Delta w_{ij} = -\eta \frac{\partial \varepsilon}{\partial w_{ij}}$$

with η is learning rate

Weight modification

- Using previous formulation, weight is updated using

$$w_{ij}(\text{new}) = w_{ij}(\text{old}) + \Delta w_{ij}$$

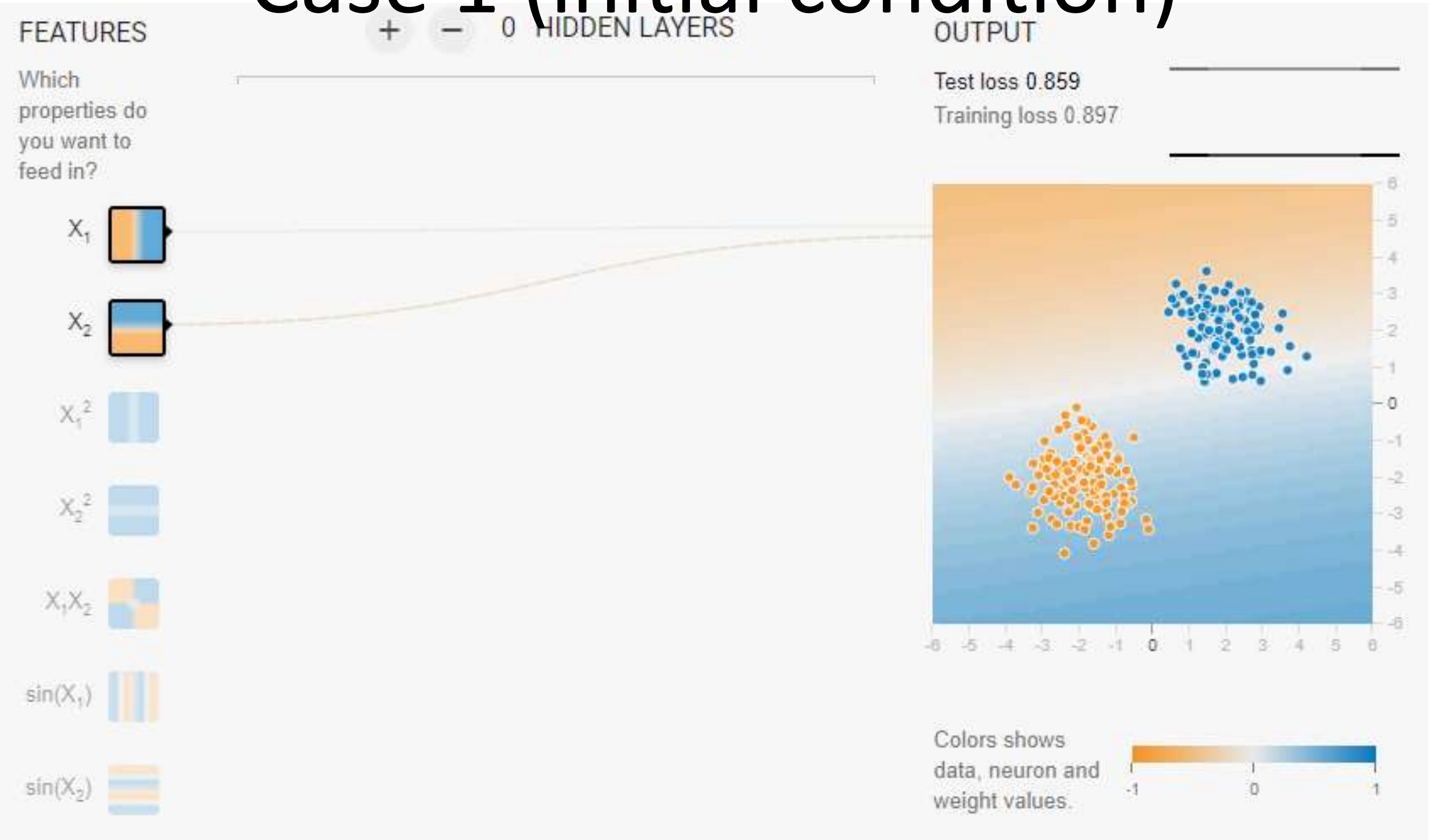
- The process continues until certain expected value of total ε

TensorFlow playground

A neural network playground

- url https://playground.tensorflow.org/
- It can help in visualizing how an ANN works
- Let’s try the classification this time

Case 1 (initial condition)



FEATURES

Which properties do you want to feed in?

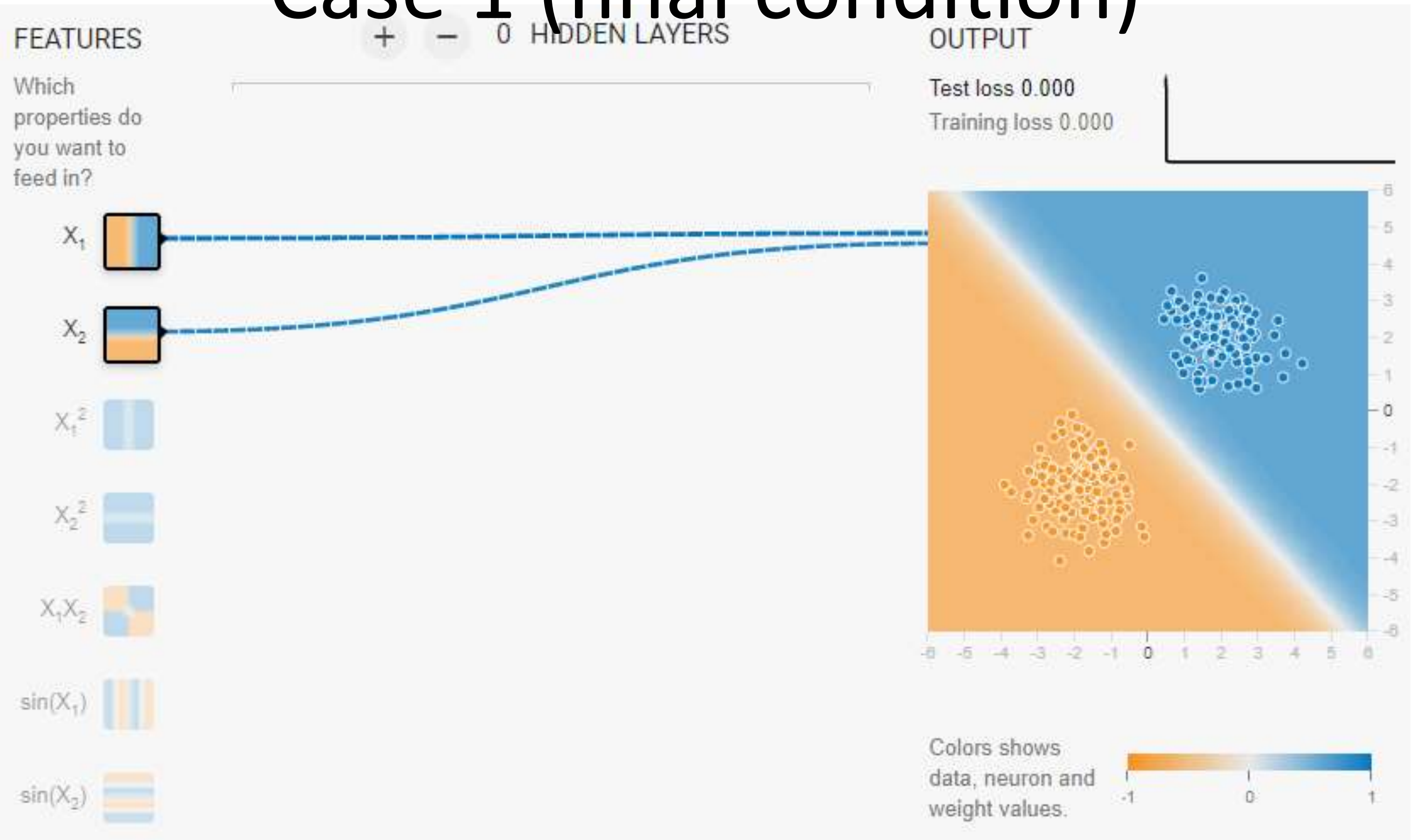
0 HIDDEN LAYERS

OUTPUT

Test loss 0.859
Training loss 0.897

Colors shows data, neuron and weight values.

Case 1 (final condition)



FEATURES

Which properties do you want to feed in?

+ − 0 HIDDEN LAYERS

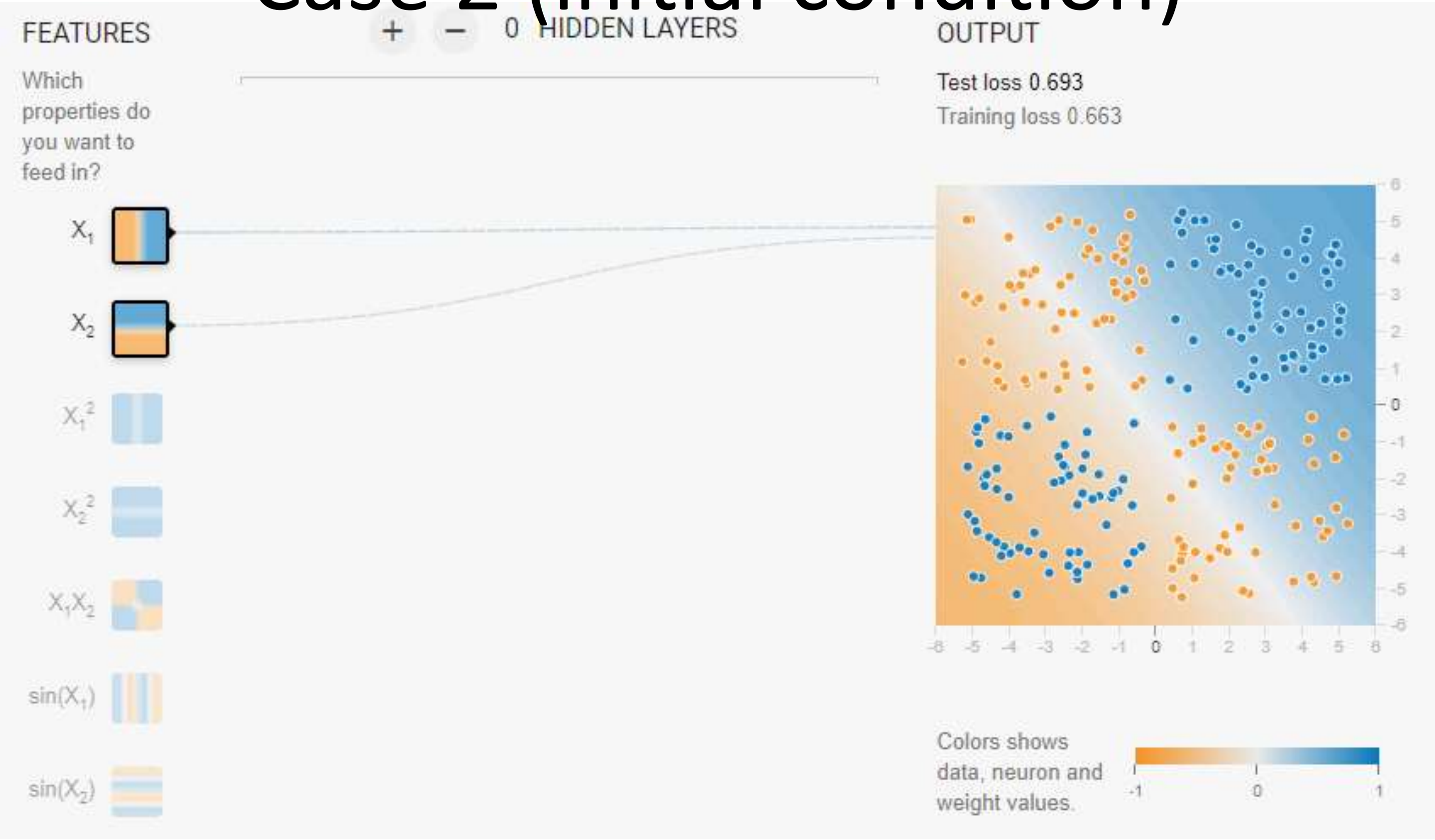
OUTPUT

Test loss 0.000

Training loss 0.000

Colors shows data, neuron and weight values.

Case 2 (initial condition)



FEATURES

Which properties do you want to feed in?

+ − 0 HIDDEN LAYERS

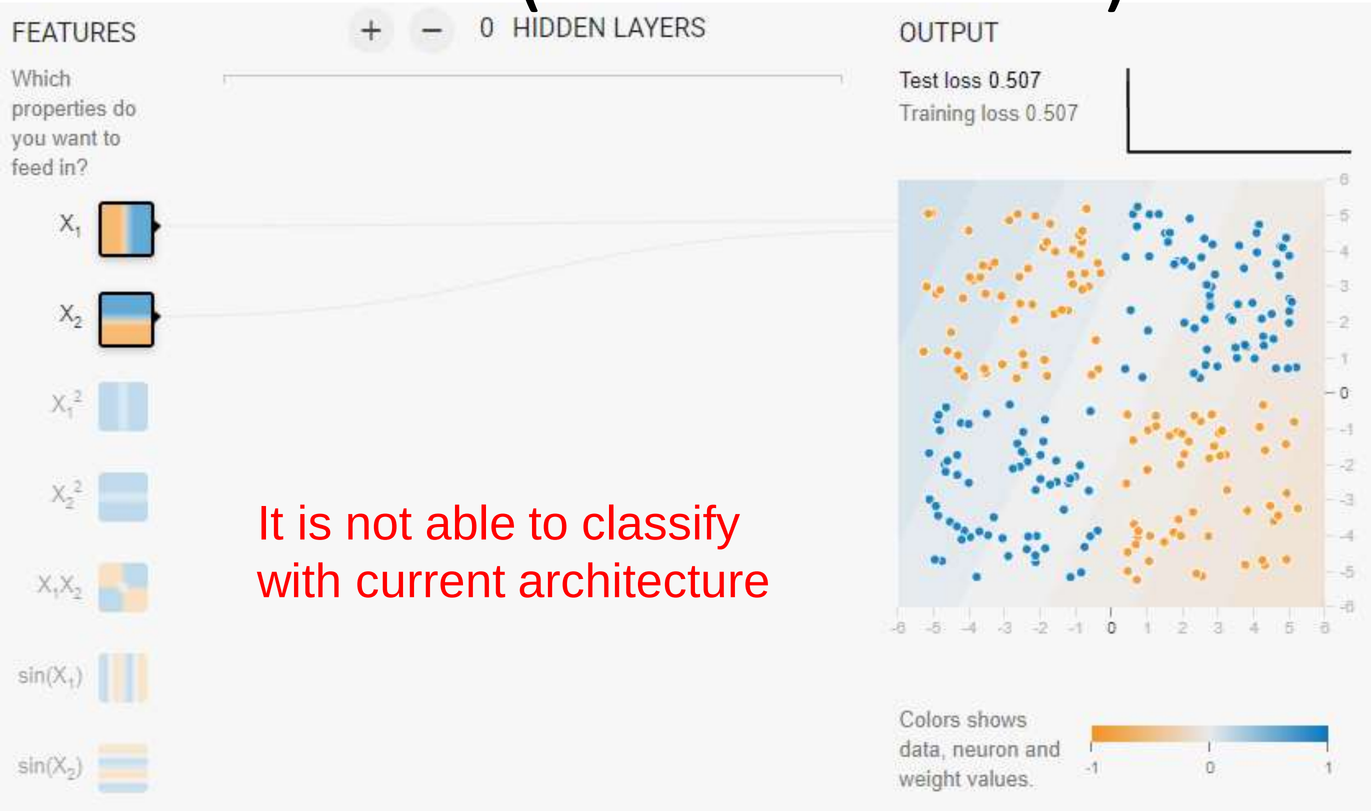
OUTPUT

Test loss 0.693

Training loss 0.663

Colors shows data, neuron and weight values.

Case 2 (final condition)



FEATURES

Which properties do you want to feed in?

+ − 0 HIDDEN LAYERS

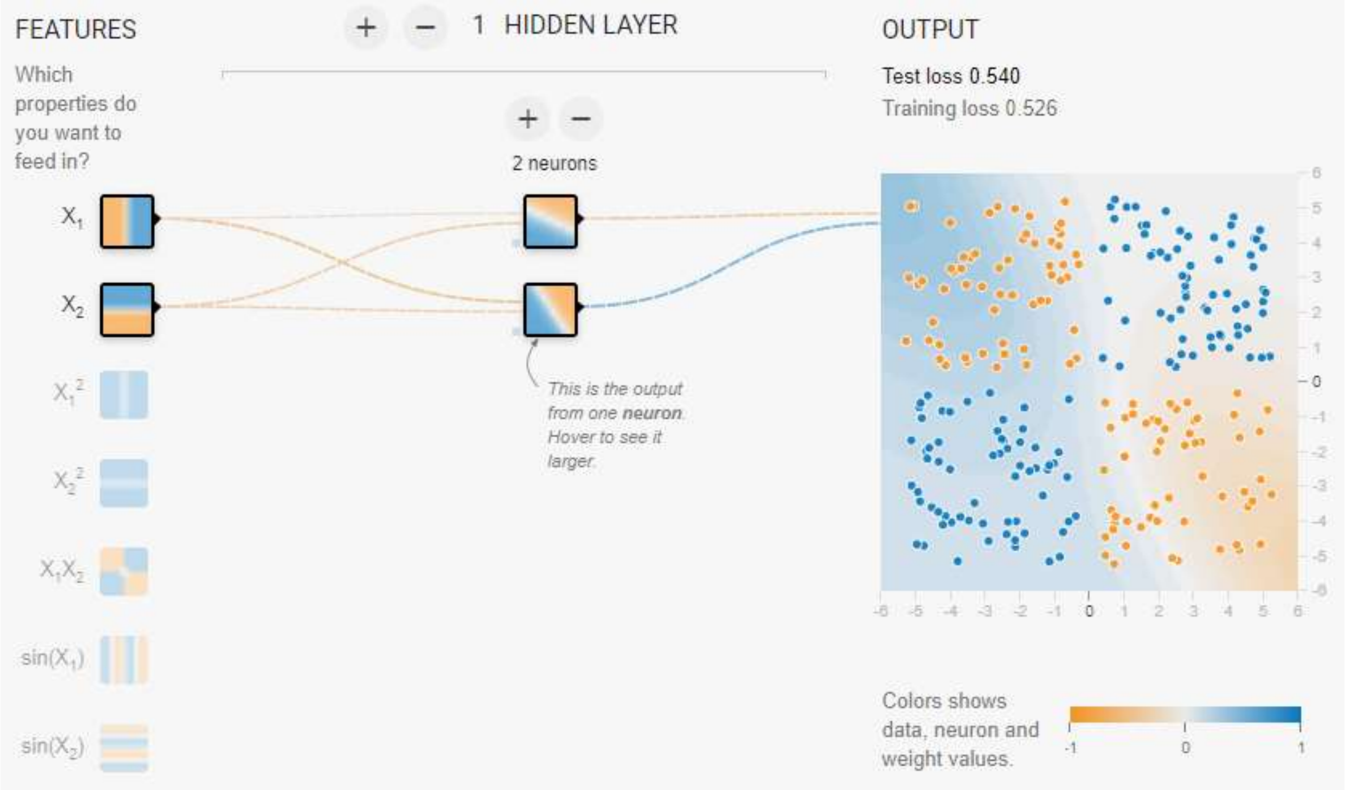
OUTPUT

Test loss 0.507
Training loss 0.507

Colors shows data, neuron and weight values.

It is not able to classify with current architecture

Case 2 (initial condition)



FEATURES

Which properties do you want to feed in?

+ − 1 HIDDEN LAYER

+ − 2 neurons

*This is the output from one **neuron**. Hover to see it larger.*

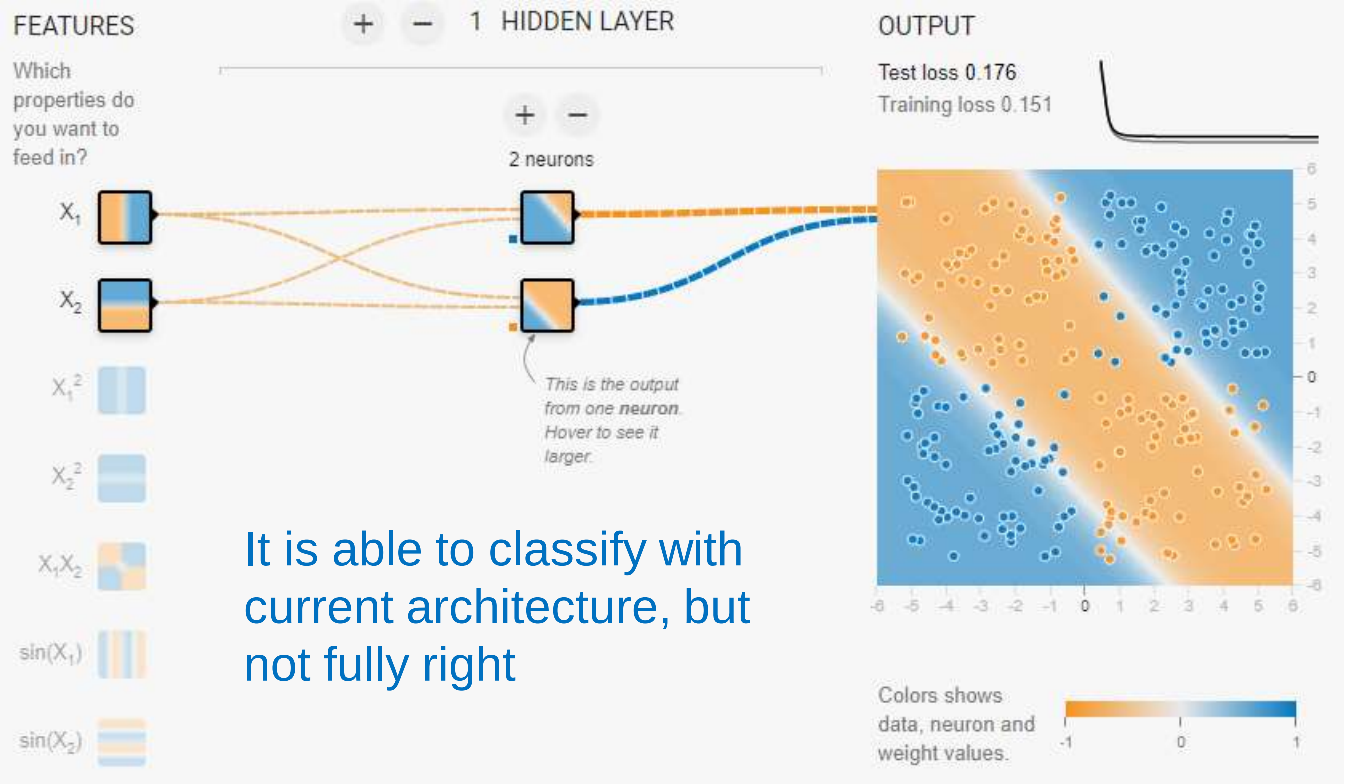
OUTPUT

Test loss 0.540

Training loss 0.526

Colors shows data, neuron and weight values.

Case 2 (final condition)



Which properties do you want to feed in?

This is the output from one ***neuron****. Hover to see it larger.*

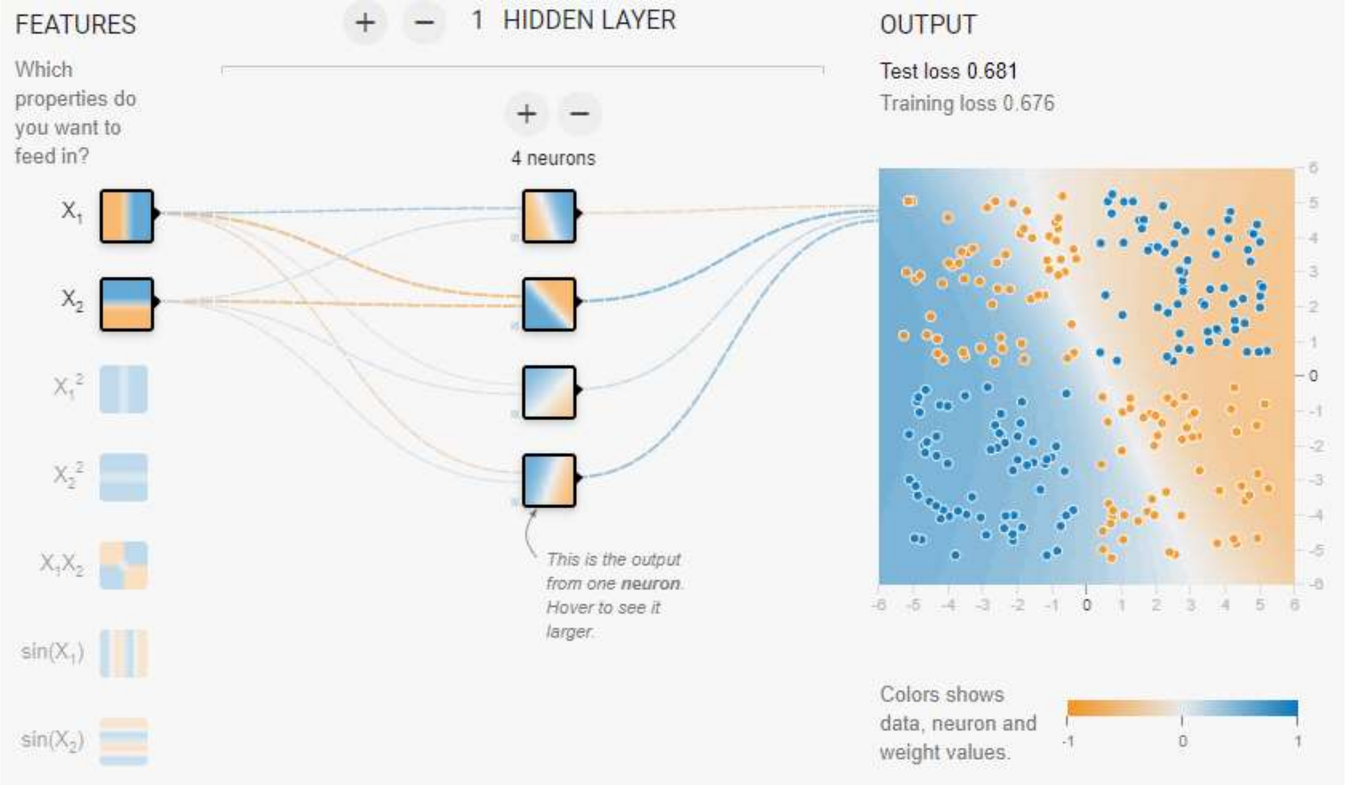
Test loss 0.176

Training loss 0.151

Colors shows data, neuron and weight values.

It is able to classify with current architecture, but not fully right

Case 2 (initial condition)



Which properties do you want to feed in?

4 neurons

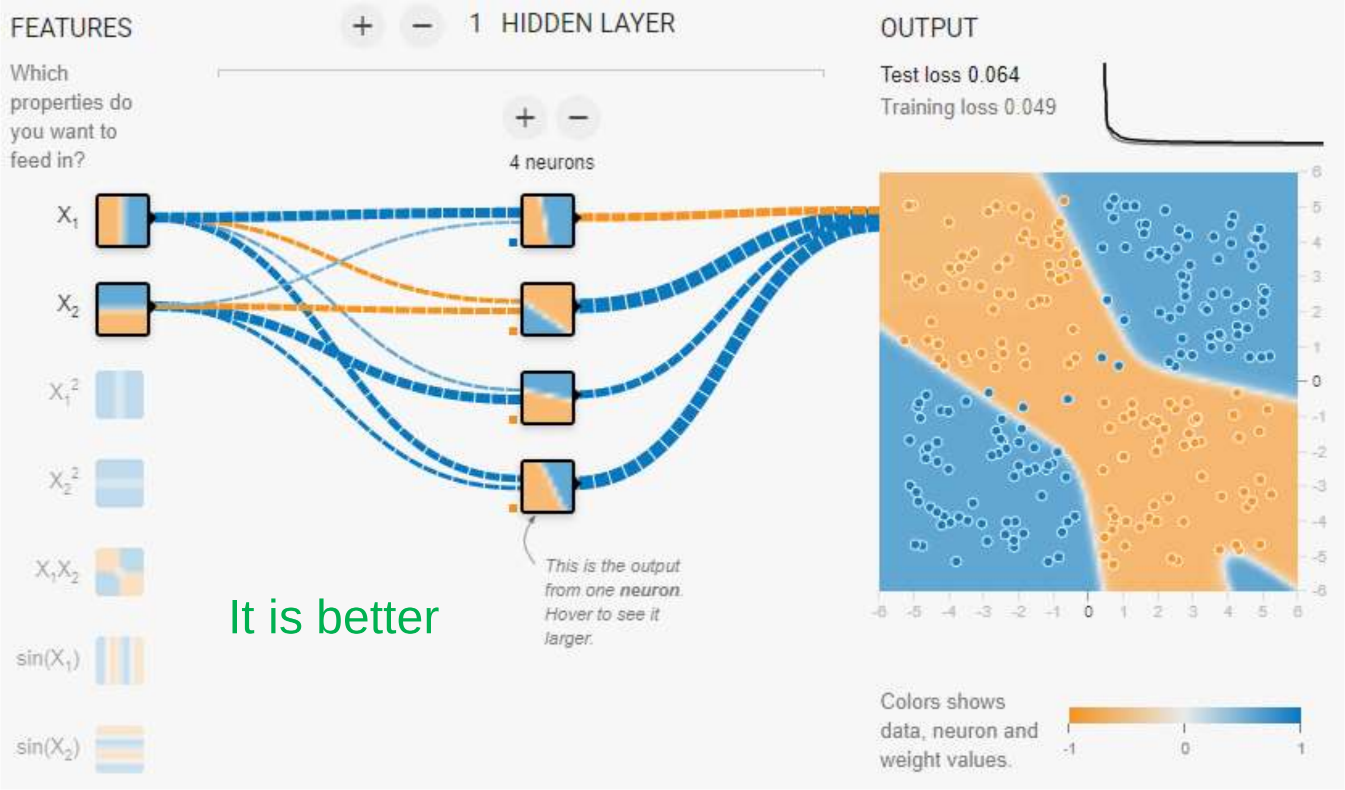
*This is the output from one **neuron**. Hover to see it larger.*

Test loss 0.681

Training loss 0.676

Colors shows data, neuron and weight values.

Case 2 (final condition)



FEATURES

Which properties do you want to feed in?

1 HIDDEN LAYER

4 neurons

OUTPUT

Test loss 0.064

Training loss 0.049

This is the output from one **neuron**. Hover to see it larger.

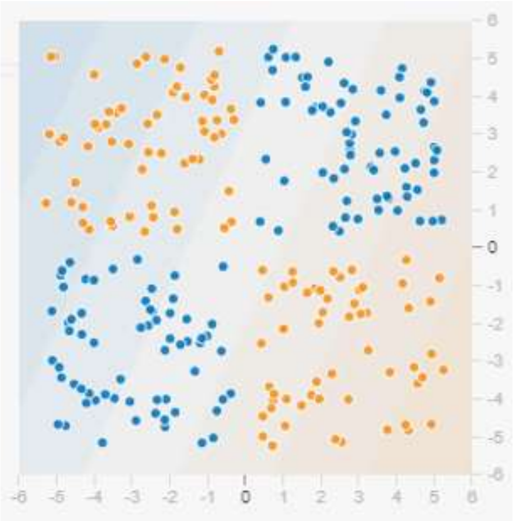
Colors shows data, neuron and weight values.

It is better

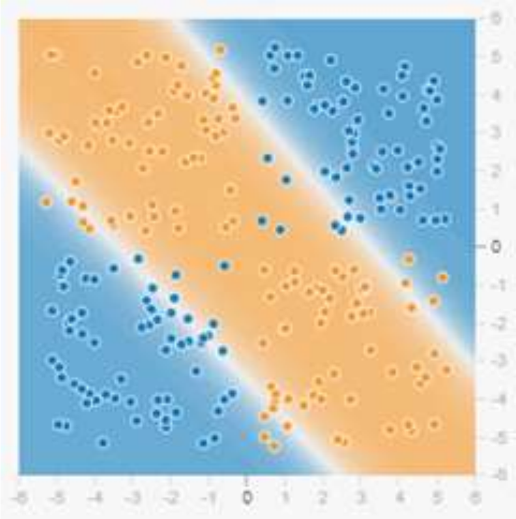
Assignment

Background: Case 2

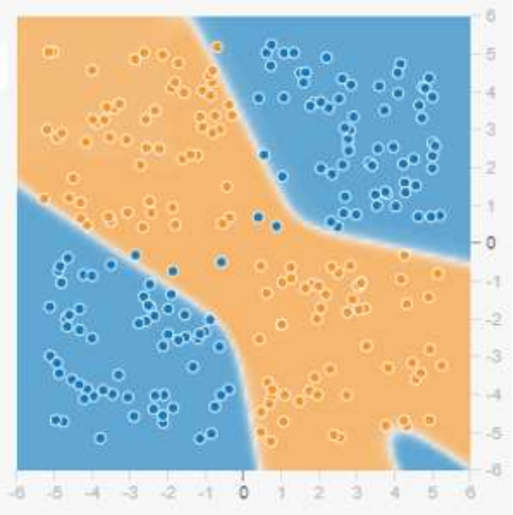
- Current results



2 - 1



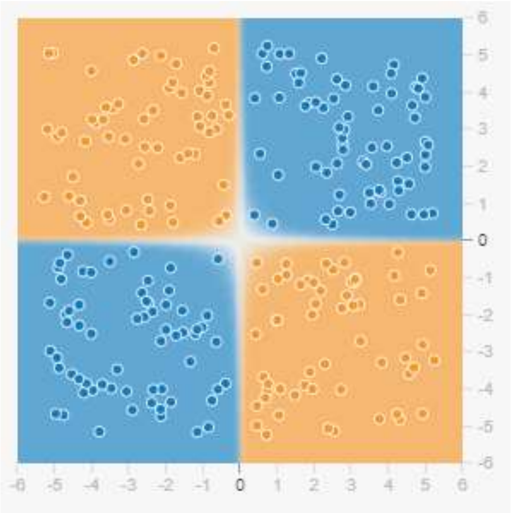
2 - 2 - 1



2 - 4 - 1

Problem: Architecture and features

- Target:



- Please provide:
  - Features? (x_1, x_2, ..)
  - Architecture? (e.g. 2-3-4-1, ..)
  - Learning rata?
  - Activation function?
  - Regularization?
  - Number of attemps?

Thank you

-