Welcome to

BS6207

2021

Lee Hwee Kuan

An Innovative Way to Learn and Teach Deep Learning

- We all learn together - sharing sessions
- Students are expected to *learn most techniques online by themselves*
- What is the use of this course?
  - This course provides concepts and understandings that cannot be easily self-learned
  - This course provides the real life experience for doing deep learning
  - This course teaches you how to think in different ways
- **Always** bring your laptop, we do coding in class
- If you want to get my attention, call me by my first name "Hwee Kuan", you can call me "Dr Lee" or "Prof Lee" if you want me to ignore you

Sharing sessions

If we categorize people into those with technical skills and those who can explain their ideas, there are 4 combinations.

1. Those who have no skills & cannot explain their ideas
2. Those who have skills & cannot explain their ideas
3. Those who have no skills & can explain their ideas
4. Those who have skills & can explain their ideas

Sharing sessions

1.Those who have no skill & cannot explain their ideas

These people may be beggars

2.Those who have skills & cannot explain their ideas

These people are good servants

3.Those who have no skill & can explain their ideas

These people are the bosses

4.Those who have skills & can explain their ideas

These people are the masters

Sharing sessions

We aim to have students to present something in front of the class.

Presentations will be on:

1. Assignment solutions
2. Project solutions

Be sure to run through your presentation with a few peers. Acknowledge their help in your presentation.

I teach the whole package

- Communication skills
- Teamwork
- Decision making and scientific methods
- Technical skills

An Innovative Way to Learn and Teach Deep Learning

A previous student told me that she/he was attending my class to only learn about deep learning, not expecting to learn other soft skills and learn how to be a good person.

Historical notes on neural networks and deep learning

Historical notes

1943 **Warren McCulloch (neurophysiologist), Walter Pitts (mathematician)** Mathematical model of the brain

*McCulloch, Warren; Walter Pitts (1943). "A Logical Calculus of Ideas Immanent in Nervous Activity". Bulletin of Mathematical Biophysics. **5** (4): 115–133.*

1949 **Donald O. Hebb** Strengthening of connection between neurons

Hebb, D. O. (1949). The Organization of Behavior: A Neuropsychological Theory. New York: Wiley and Sons.

1959 **Bernard Widrow, Marcian Hoff** Single layer and multilayer neural nets. ADALINE and MADALINE

An adaptive "ADALINE" neuron using chemical "memistors"

1970 **Seppo Linnainmaa** While gradient descend algorithm dates back much earlier, Seppo contributed to the modern idea of back propagation

Linnainmaa, Seppo (1970). The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors. Master's Thesis (in Finnish), Univ. Helsinki, 6-7.

1989 **George Cybenko** Universal approximation theorem, sigmoid function

Cybenko, G. (1989) "Approximations by superpositions of sigmoidal functions", *Mathematics of Control, Signals, and Systems*, 2 (4), 303-314

1991 **Kurt Hornik** Universal approximation theorem, more general function

Kurt Hornik (1991) "Approximation Capabilities of Multilayer Feedforward Networks", Neural Networks, 4(2), 251–257

https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/History/history1.html

‘Contemporary’ history of neural nets

1974	**Paul Werbos**, Backpropagation
1980	**Kunihiko Fukushima**, Neocogitron which inspired Convolutional Neural Networks
1985	**Hilton & Sejnowski**, Boltzmann Machine
1986	**Paul Smolensky**, Harmonium, later known as Restricted Boltzmann Machine **Michael I. Jordan** Recurrent Neural Network
1990	**Yann LeCun**, LeNet - convolutional neural net
2006	**G. Hinton**, Deep Belieft Net, layer wise pretraining
2009	**Salakhutdinov & Hinton**, Deep Boltzmann Machines
2012	**N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdin**, Dropout

‘Contemporary’ history of neural nets

2014 **Ian Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio**, Generative Adversarial Networks

2015 **Kaiming He, *Xiangyu Zhang, Shaoqing Ren, Jian Sun***, Deep Residual Network

2015 **Nicolas Papernot et al**, Adversarial Deep Learning

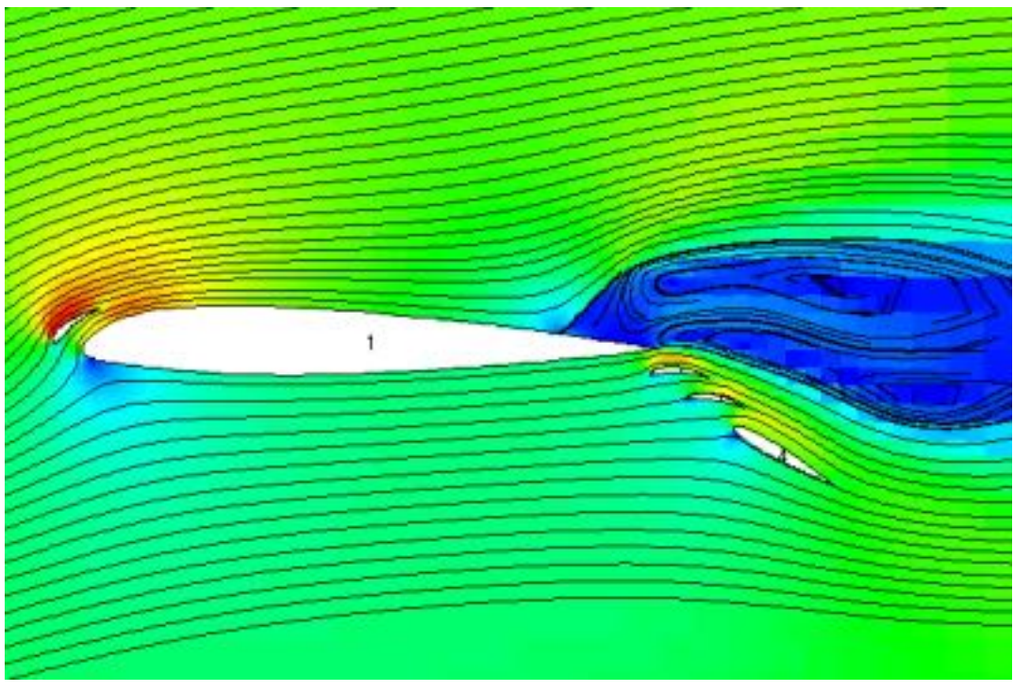
2016 **Shaoqing Ren et al**, Region Proposal Network

2017 **David Silver et al**, Alpha-Go Zero

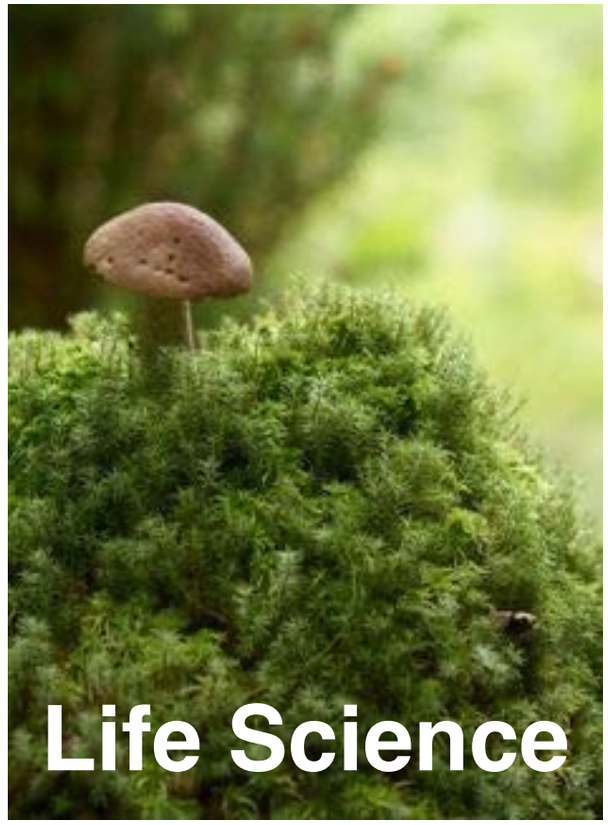
2018 **SMA Eslami et al**, Generative Query Network

Our world versus computer world

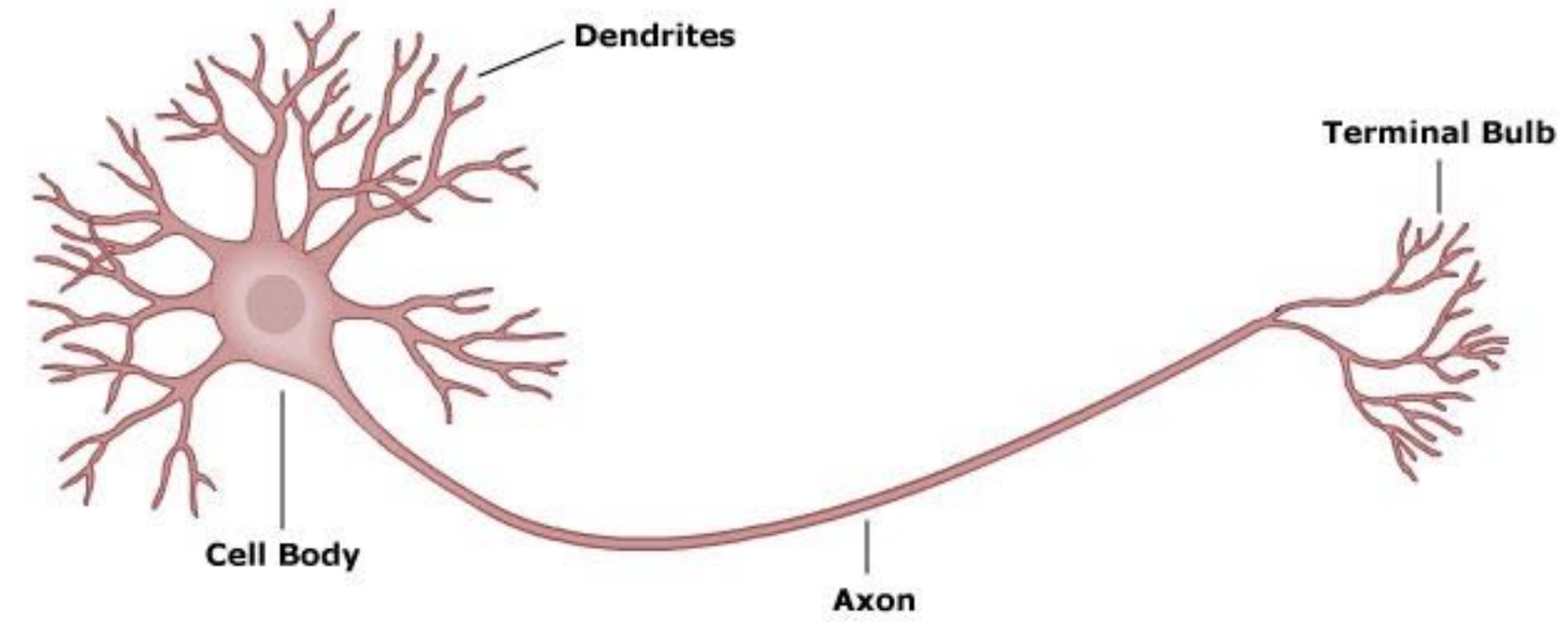
23684184 x 4729472 = 112013685070848



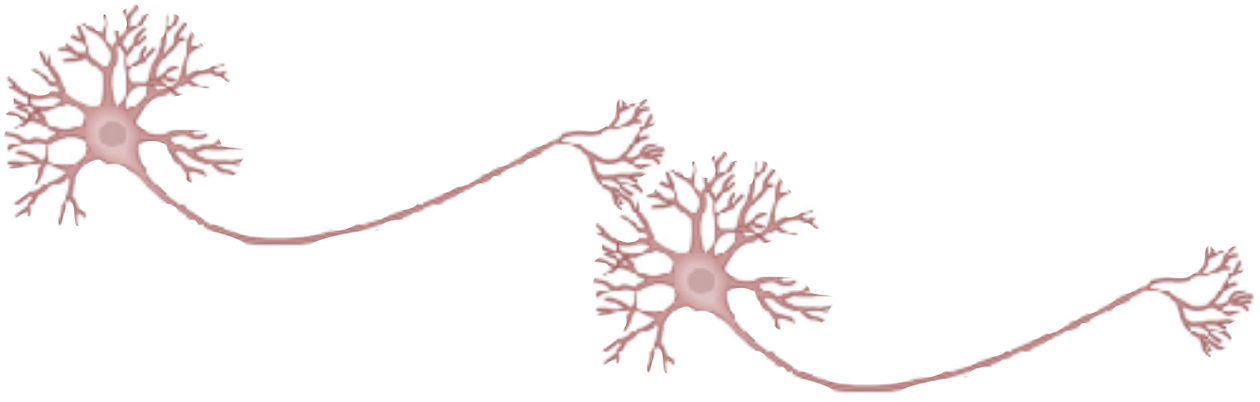
Our world versus computer world



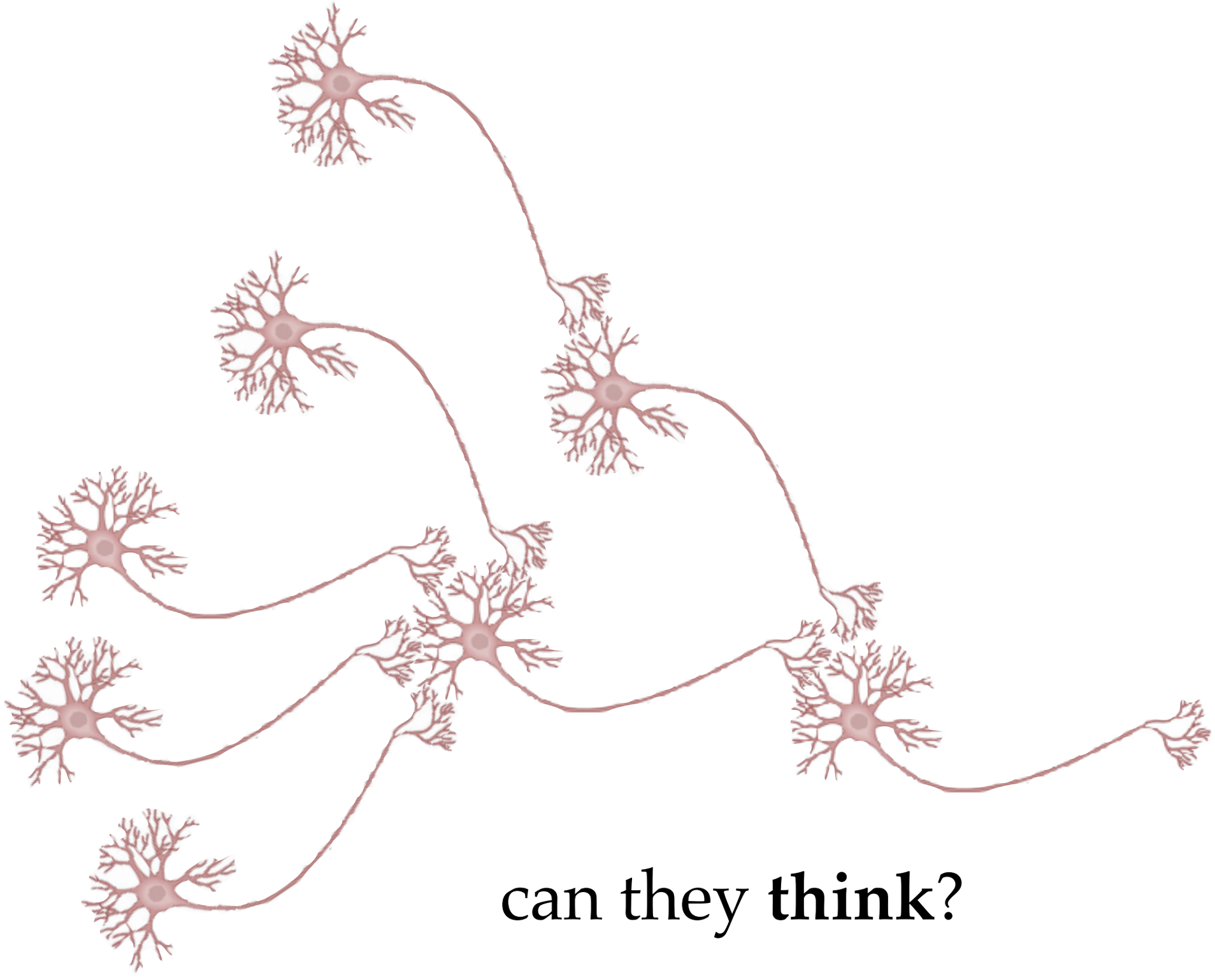
What makes Deep Leaning so good?



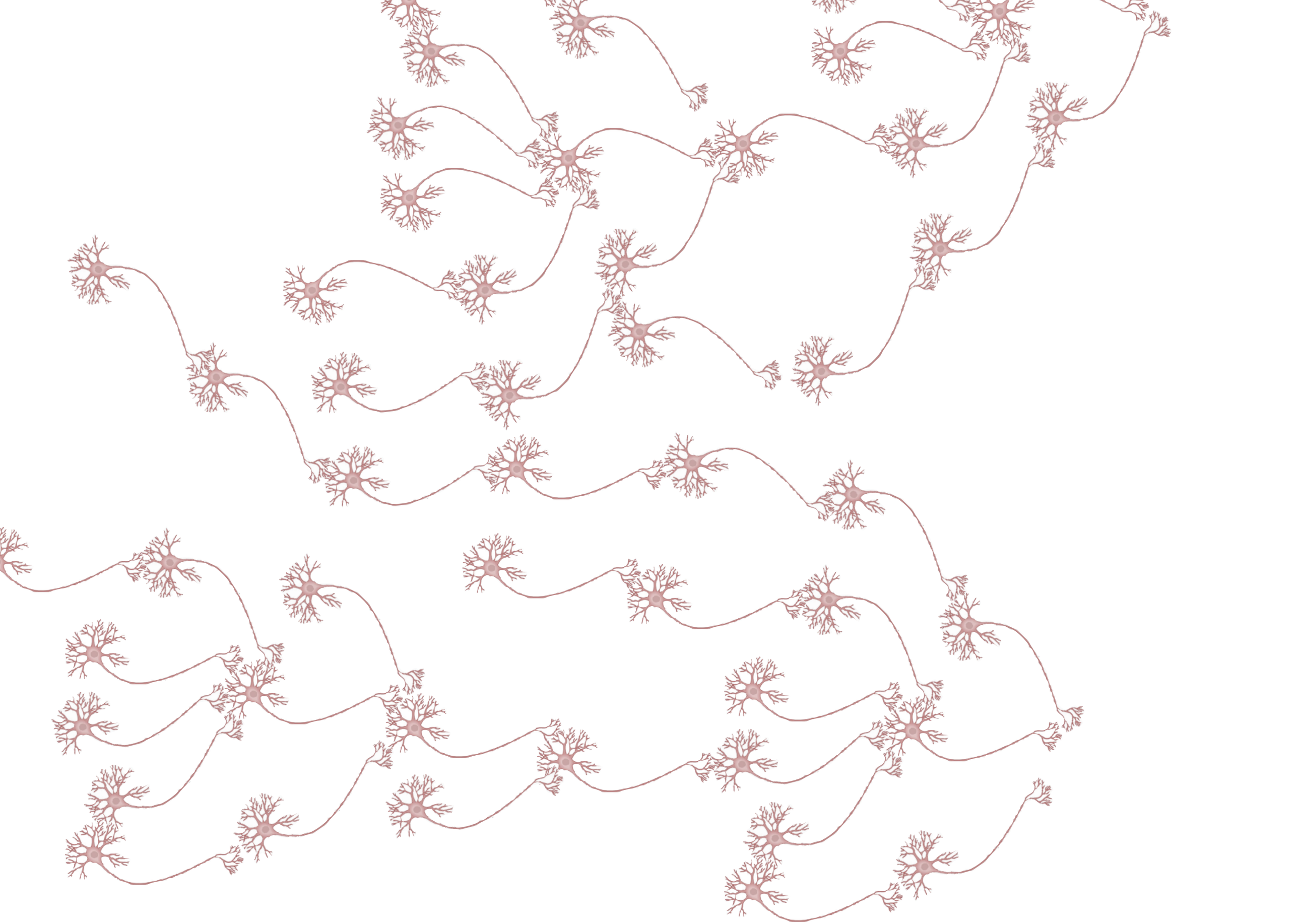
can this thing **think**?

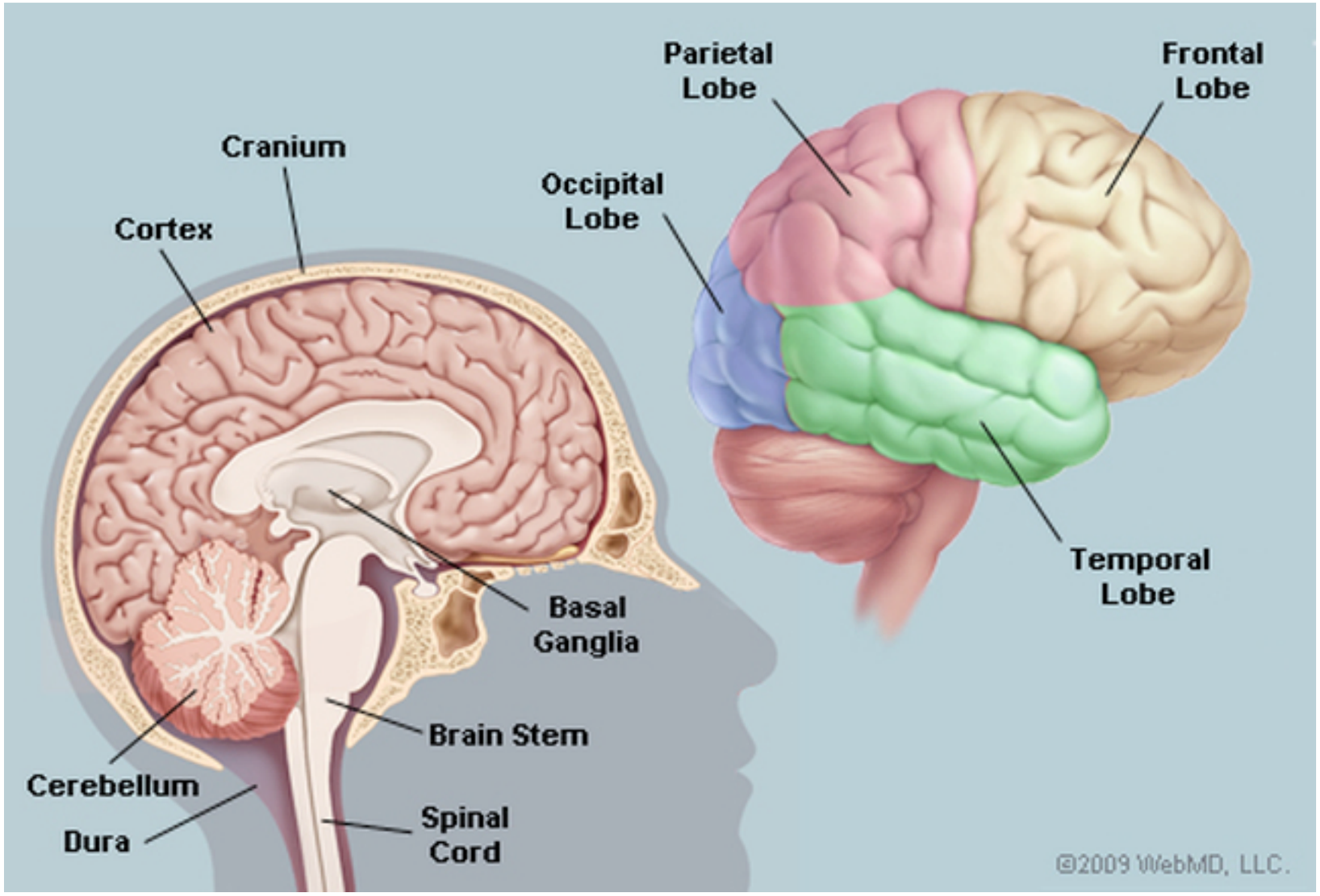


can they **think**?



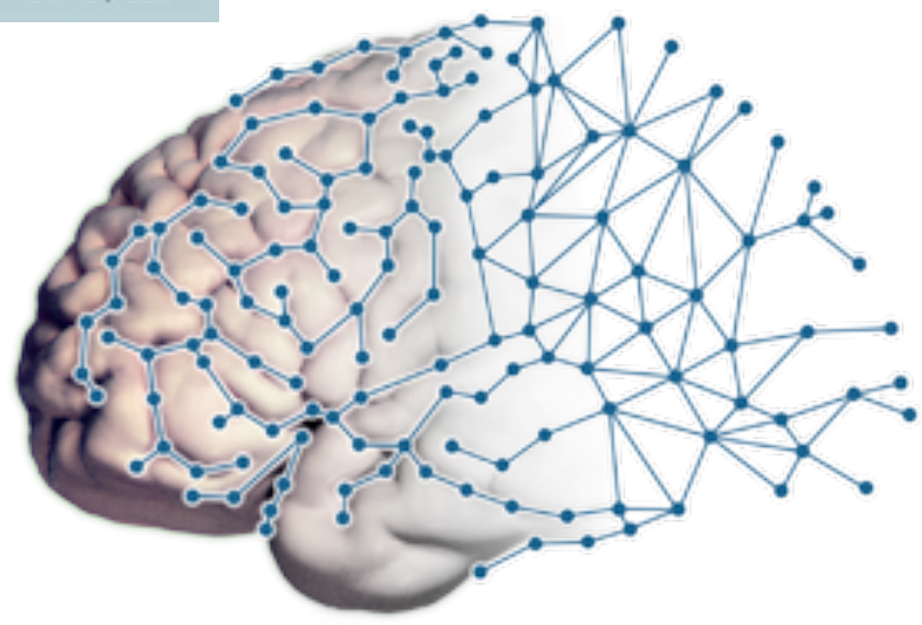
can they **think**?

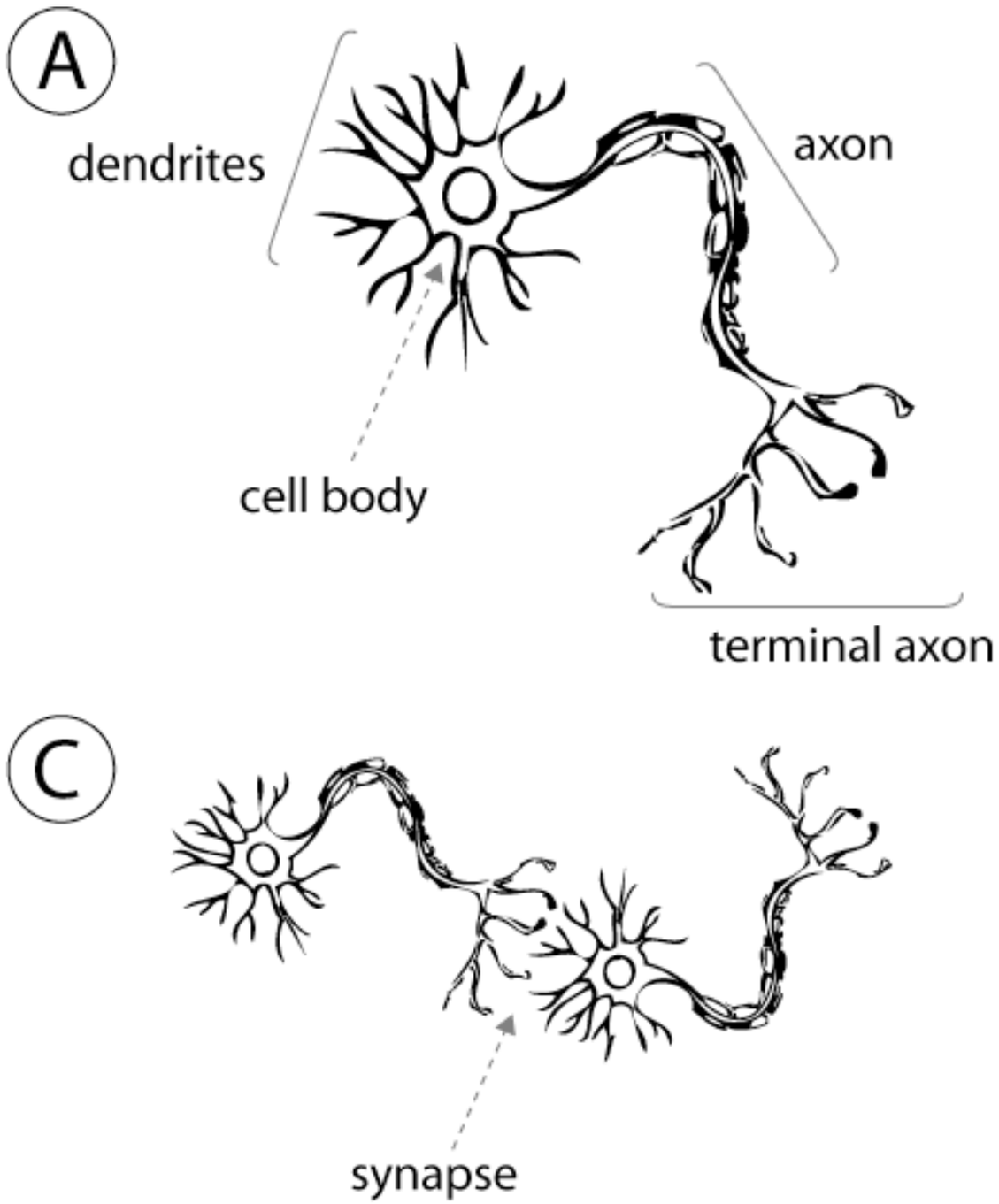


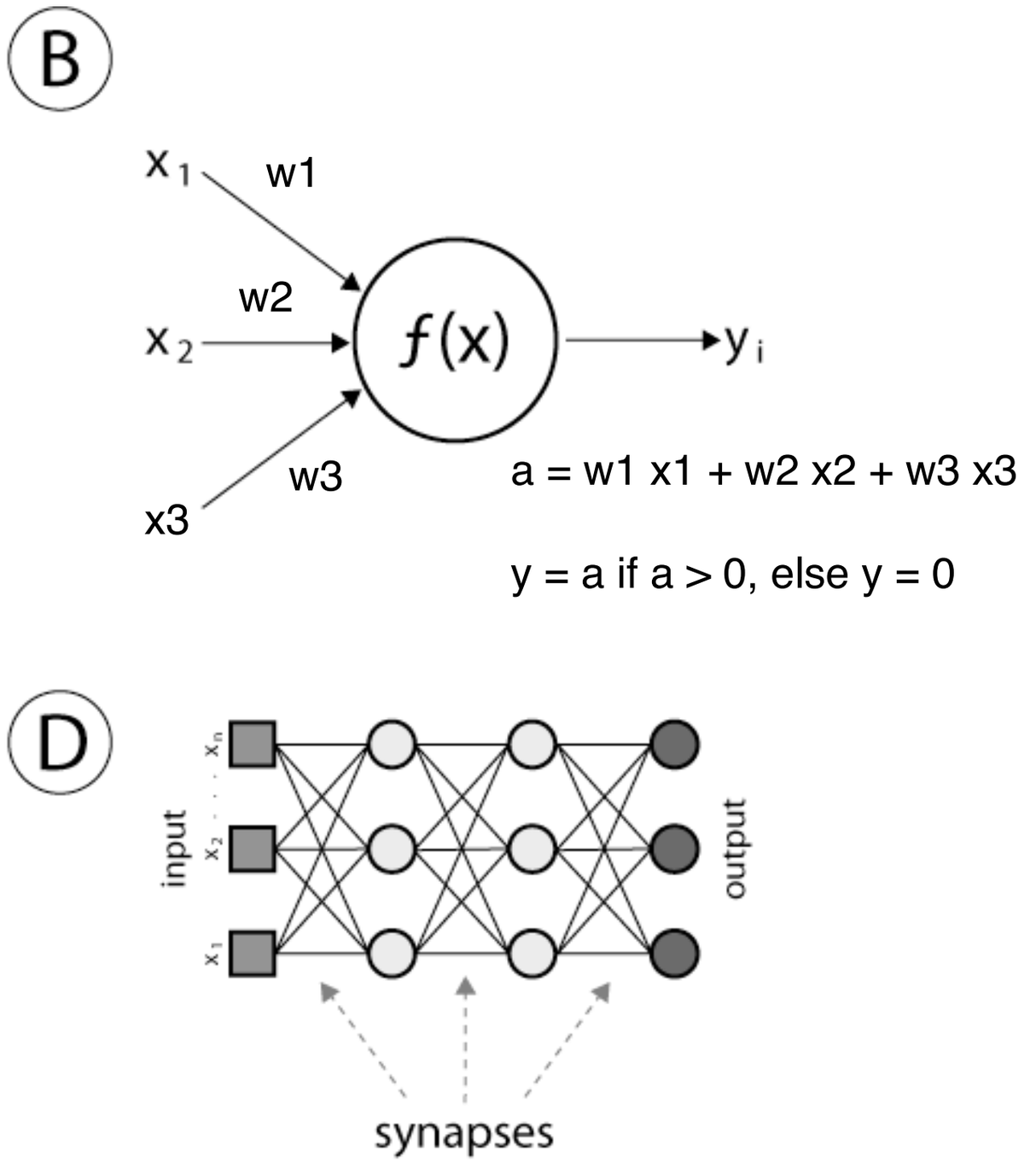


http://img.webmd.com/dtmcms/live/webmd/consumer_assets/site_images/articles/image_article_collections/anatomy_pages/brain2.jpg?resize=646px:*&output-quality=100

can it **think**?

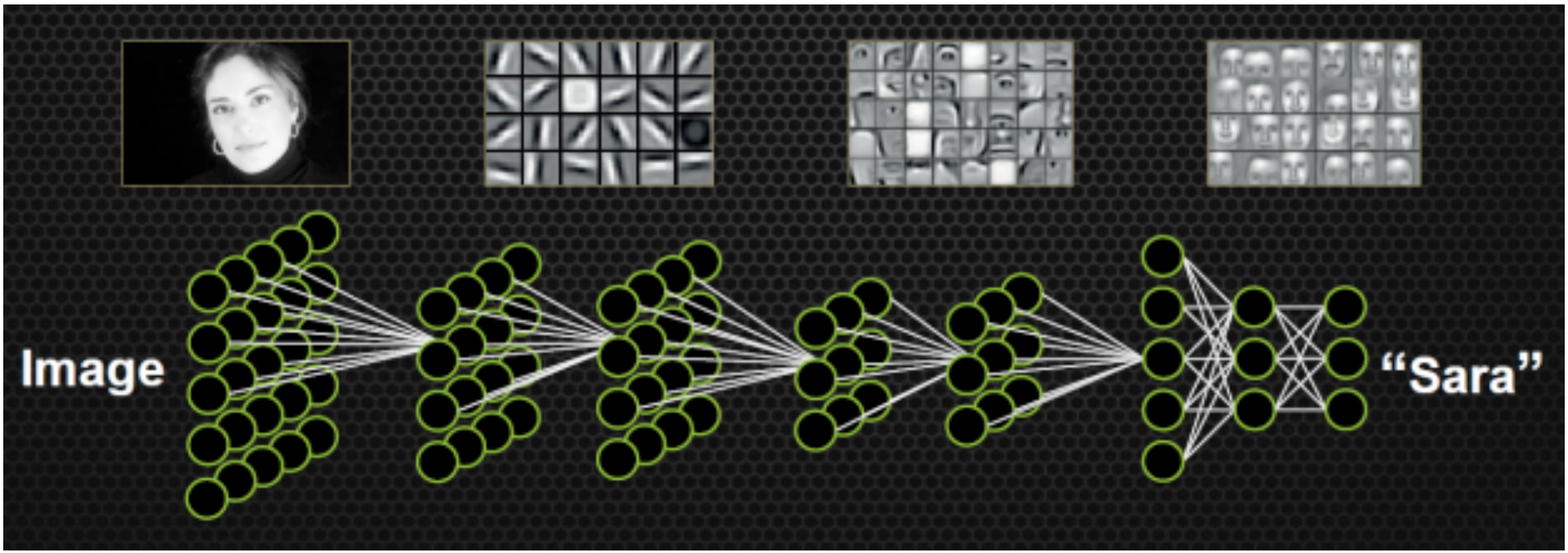




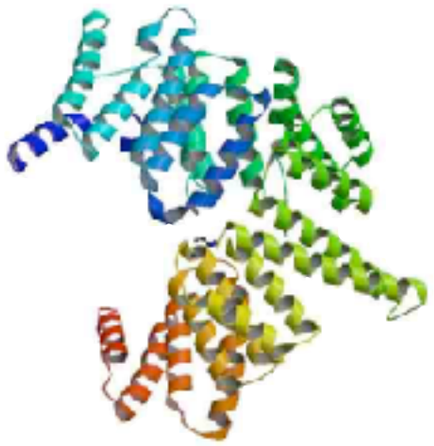


can this thing **think**?

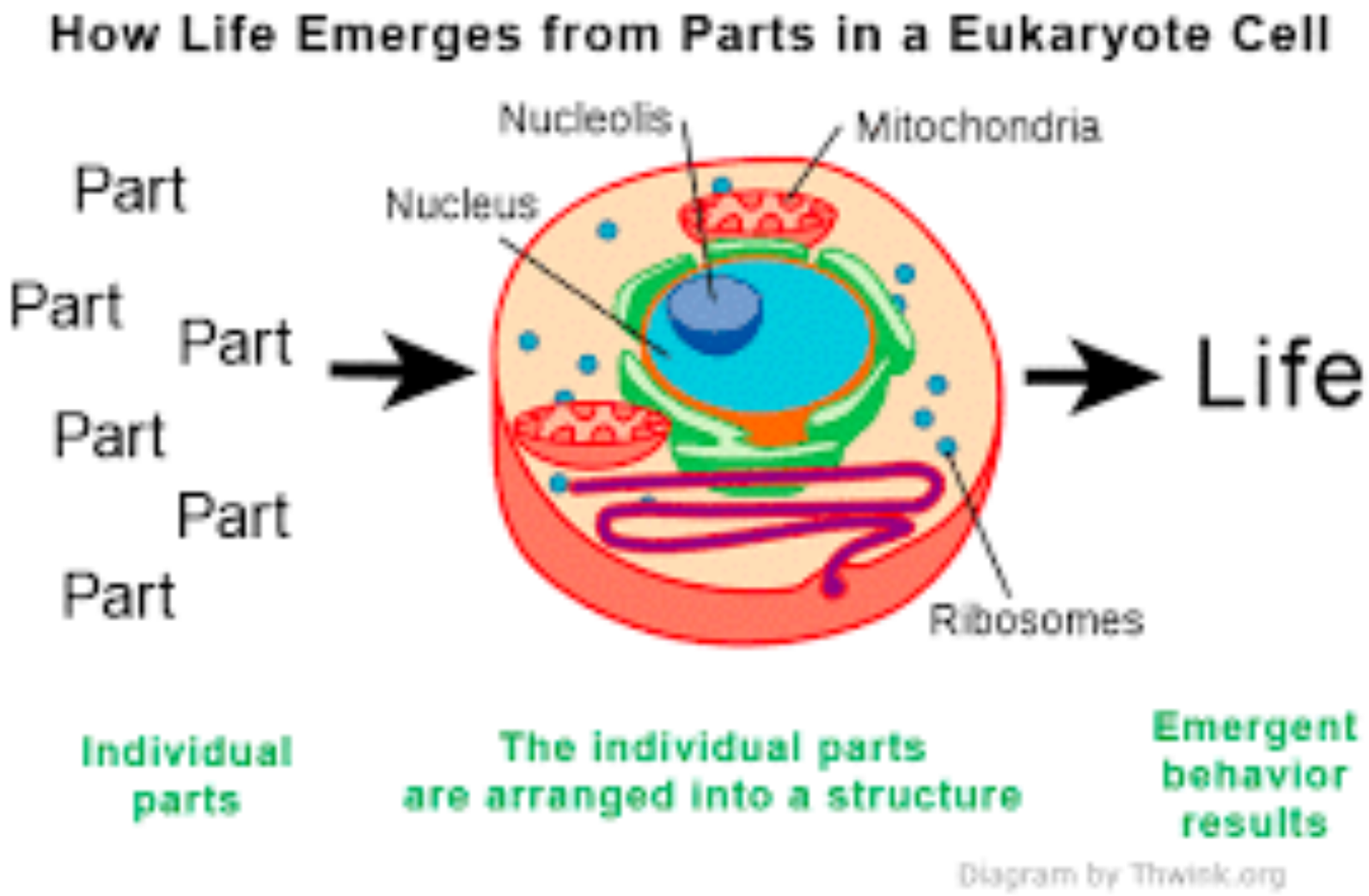
With millions of connections, it started to “think”



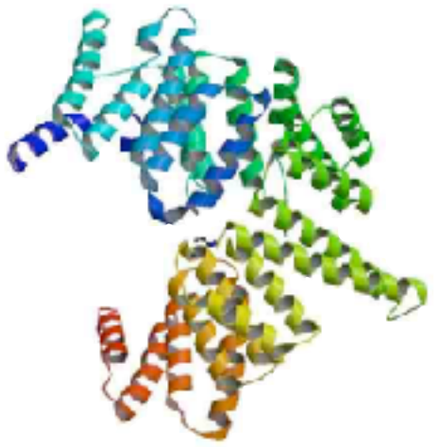
How do we get from molecules (proteins) to cell and then to life?



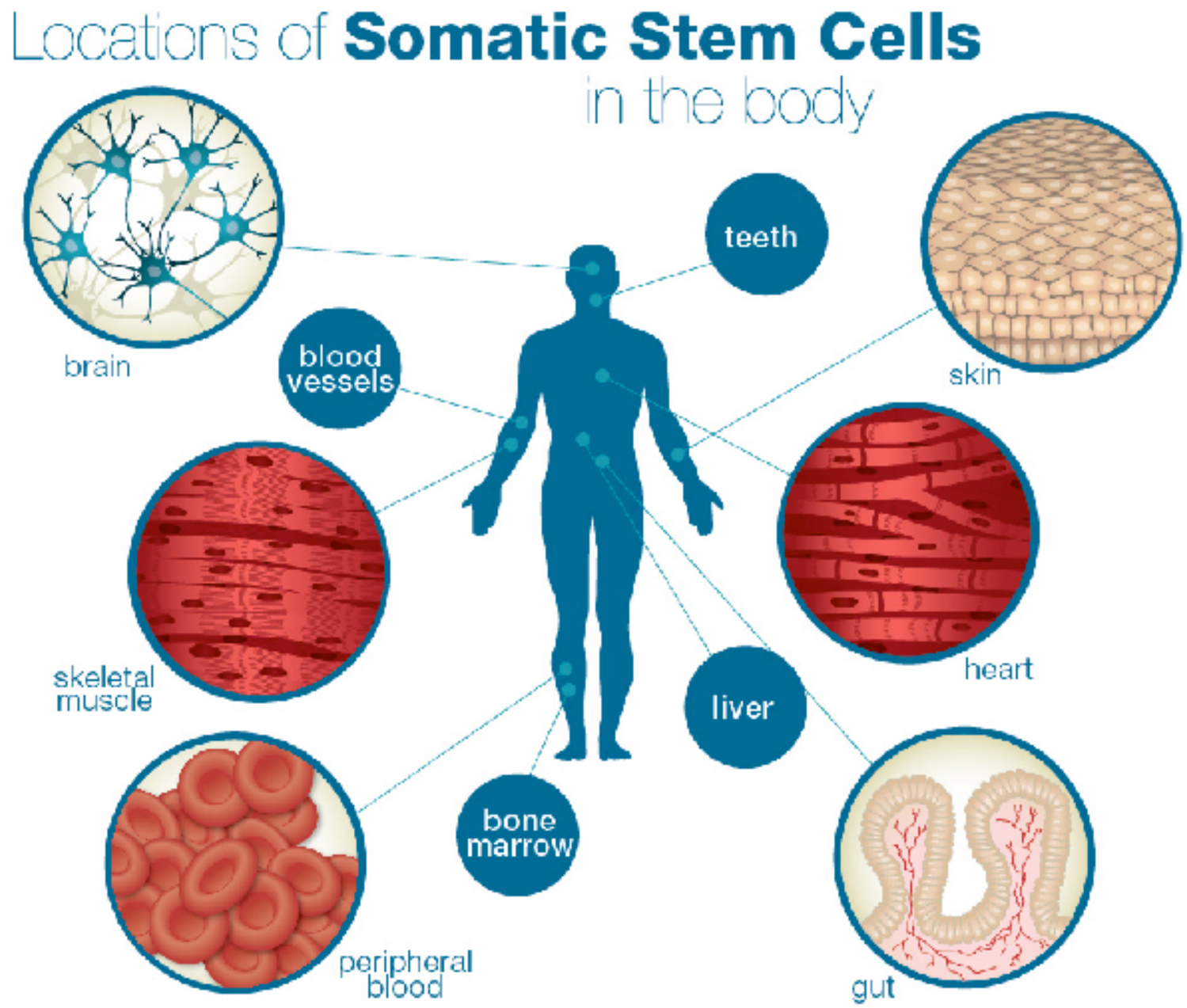
proteins



How do we get from molecules (proteins) to cell and then to life?

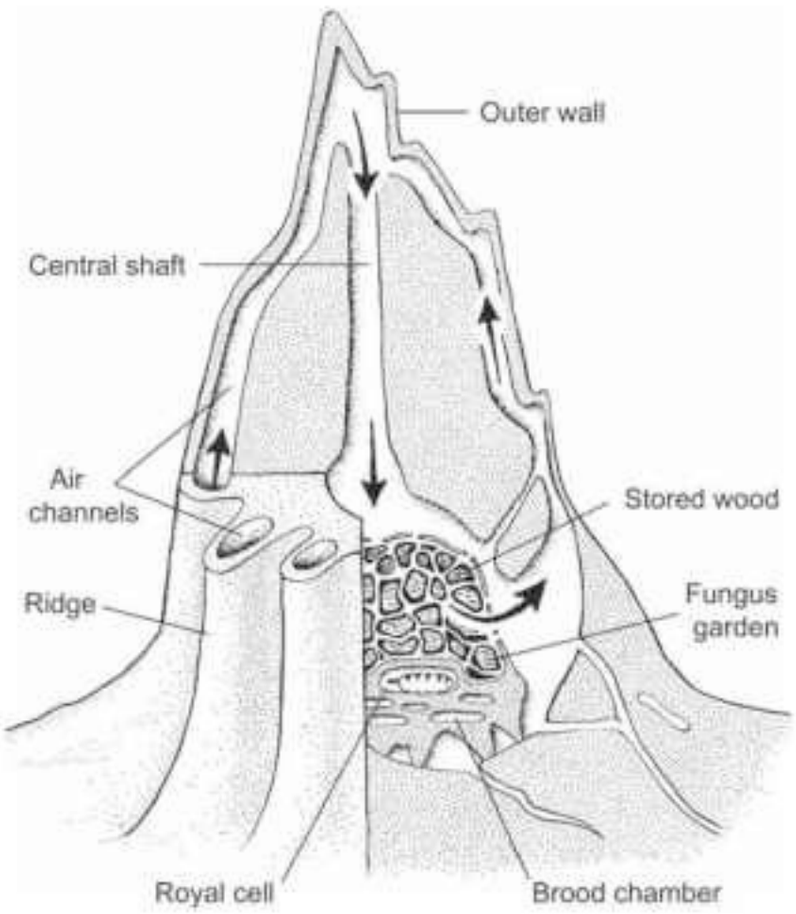


proteins



http://learn.genetics.utah.edu/content/stemcells/quickref/somaticstemcells.jpg

Life exhibits complexity



https://www.pinterest.com/sattele/termite-mound/

http://img.ev.mu/images/reportages/188/520x342/09.jpg

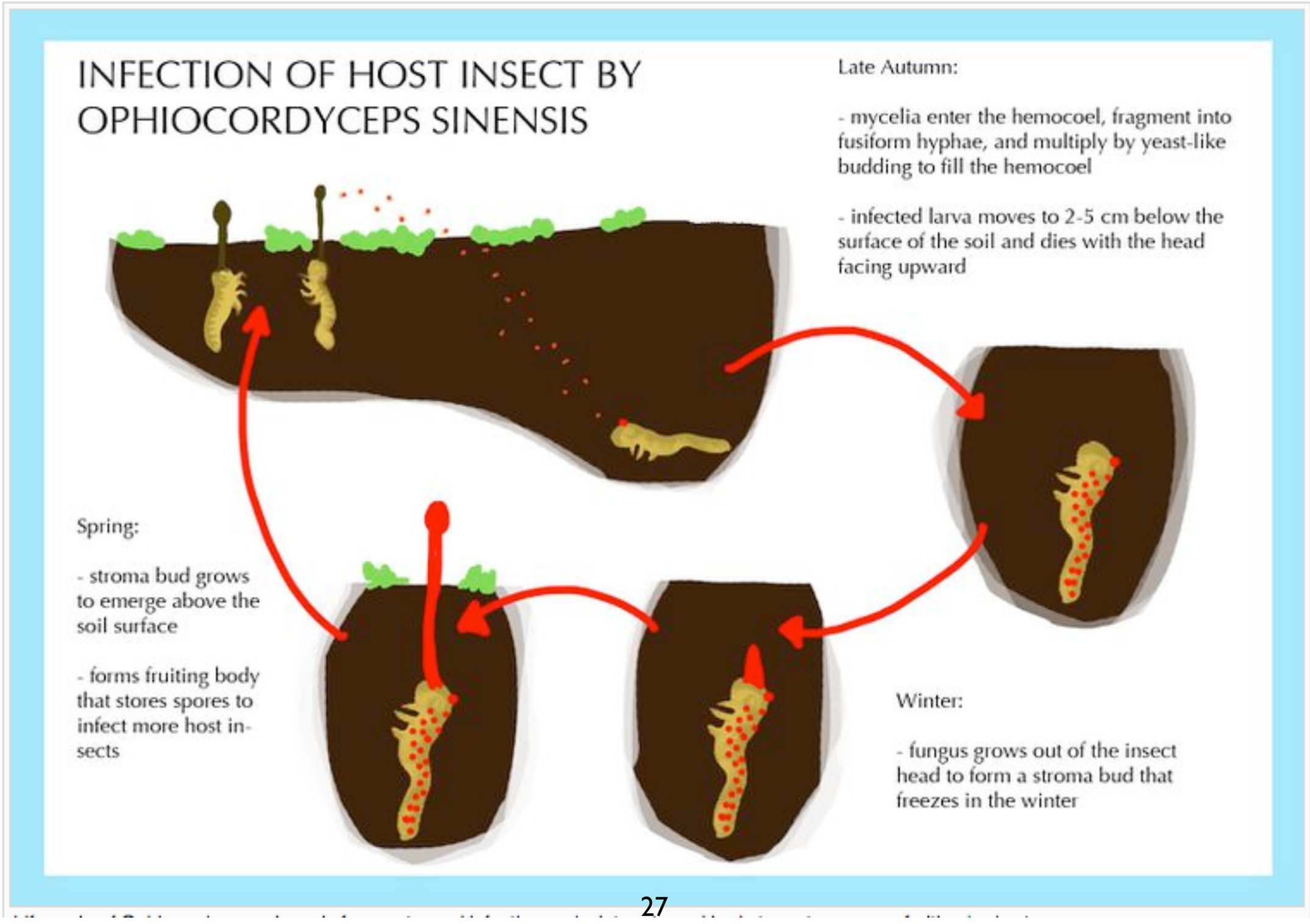
How extreme can life become?
How extreme can emergent behaviour become?

very extreme indeed

some life forms on earth evolved the ability to control the behaviour of another life form!

Ophiocordyceps sinensis
冬虫草

http://microbewiki.kenyon.edu/index.php/Ophiocordyceps_sinensis



https://microbewiki.kenyon.edu/index.php/Ophiocordyceps_unilateralis

How to make a zombie ant

Ophiocordyceps unilateralis, a fungus found in the tropical rainforests of Thailand, survives by controlling carpenter ants.

1. INFECTION

The foraging carpenter ant walks through an area of rainforest floor infested with microscopic spores dropped by a mature fungus. The spore excretes an enzyme that eats through the ant's exterior shell.

2. DEATH GRIP

After two days, the ant leaves its tree colony and climbs down to a spot where humidity and temperature are optimal for the fungus to grow. The ant crawls onto a stem or the underside of a leaf and bites into its main middle vein so it won't fall. Then it dies.

3. FUNGAL GROWTH

The fungus consumes the ant's internal organs, using its shell as a protective casing. The fungus' main stem, called a stroma, erupts from the back of the ant's head and grows

4. "KILLING ZONE"

The mature fungus releases spores from its stroma. The spores fall to the ground creating a 10-square-feet "killing zone" which will attack new ants.

Source: David P. Hughes,

Cordyceps

https://www.youtube.com/watch?v=XuKjBIBBAL8

How extreme can machines become?

are we able to make a machine that is as complex as the most primitive life known to us?

How to learn Deep Learning

If you only know how to drive, you can go far

If you have **money** and get a sport car,
you can go far very fast

You don’t have to know how to build a car

“Deep Learning” is free and they go fast

using them is as easy as driving a car

But there is a problem!!

…even if you have the data!

Deep Learning

To go to great places, you need to know where to go

No point going fast but go in circles
you need to know the route

You need to know if your car is ok or is breaking

The computer always give you

an output

is it correct?

Different levels of understanding Deep Learning

There are those who do not know what they are doing. Their computational results are unreasonable

There are those who know how to get some good results but cannot explain them

There are those who understand what is going on with their experiments. Able to explain their results

There are those who can combine different methods to create new things in Deep Learning

There are those who can fundamentally change Deep Learning research

Very basic

lets get everyone on the same level
sorry if this seems too simple to some of you

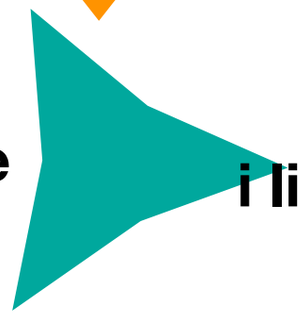
Lets play a game. . .

i like this picture

i like this picture

i like this picture

i don’t like this picture



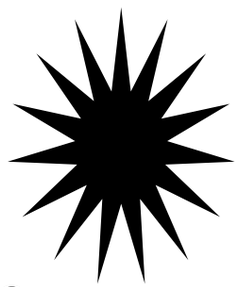
i like this picture

i like this picture

i don’t like this picture

i don’t like this picture

i like this picture

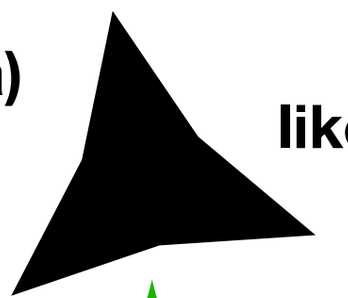


i don’t like this picture

i don’t like this picture

Testing set

(a)

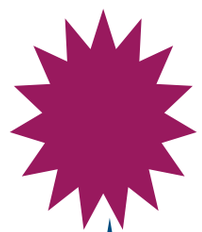


like or dislike?

(b)

like or dislike?

(c)



like or dislike?

(d)

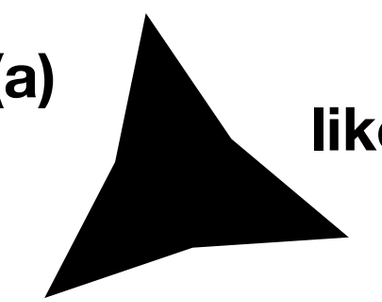
like or dislike?

Training set

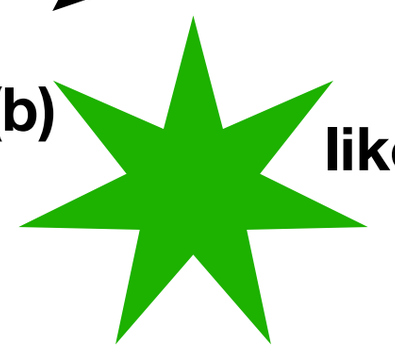
Like						
Dislike						

Testing set

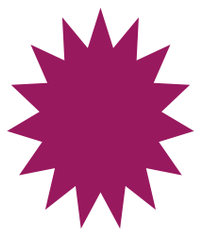
(a) like or dislike?



(b) like or dislike?



(c) like or dislike?

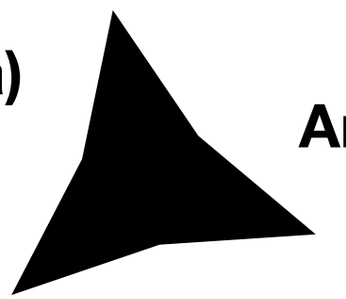


(d) like or dislike?

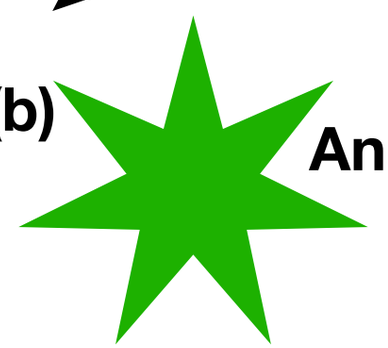
Training set

Like						
Dislike						

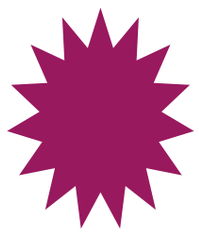
Testing set

(a) 

Answer: Like

(b) 

Answer: Dislike

(c) 

Answer: Dislike

(d) 

Answer: Like

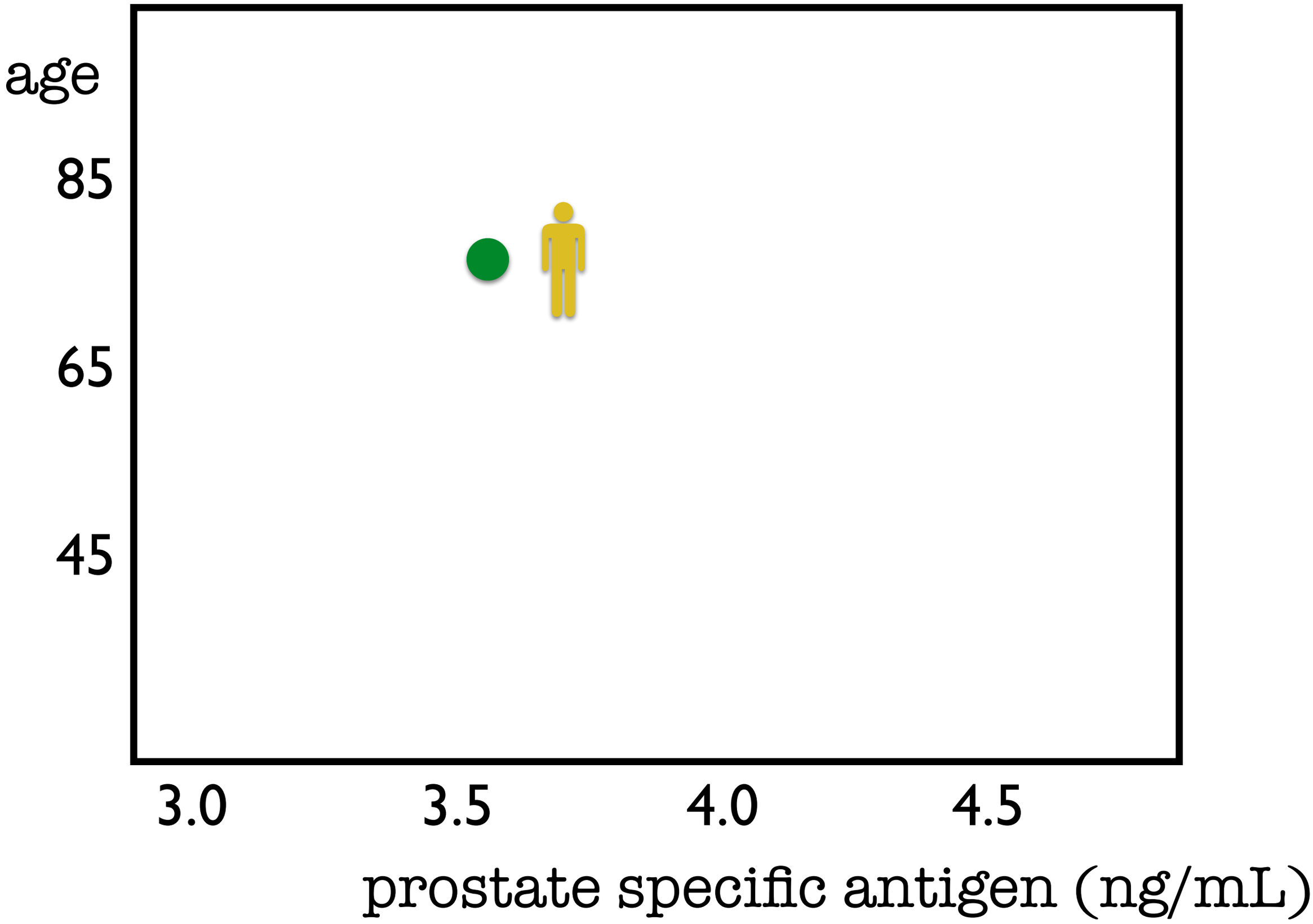
Case of prostate cancer diagnosis

Age	Prostate Specific Antigen (a blood test reading)	Got Prostate Cancer
59	4.9 ng/mL	Yes
72	3.9 ng/mL	Yes
45	6.0 ng/mL	Yes
47	3.2 ng/mL	No
39	3.9 ng/mL	No
89	3.5 ng/mL	Yes
61	5.5 ng/mL	Yes
62	2.1 ng/mL	No
49	3.4 ng/mL	No
95	3.1 ng/mL	Yes
67	4.3 ng/mL	Yes
49	3.8 ng/mL	?
58	4.3 ng/mL	?
88	4.1 ng/mL	?
31	2.1 ng/mL	?

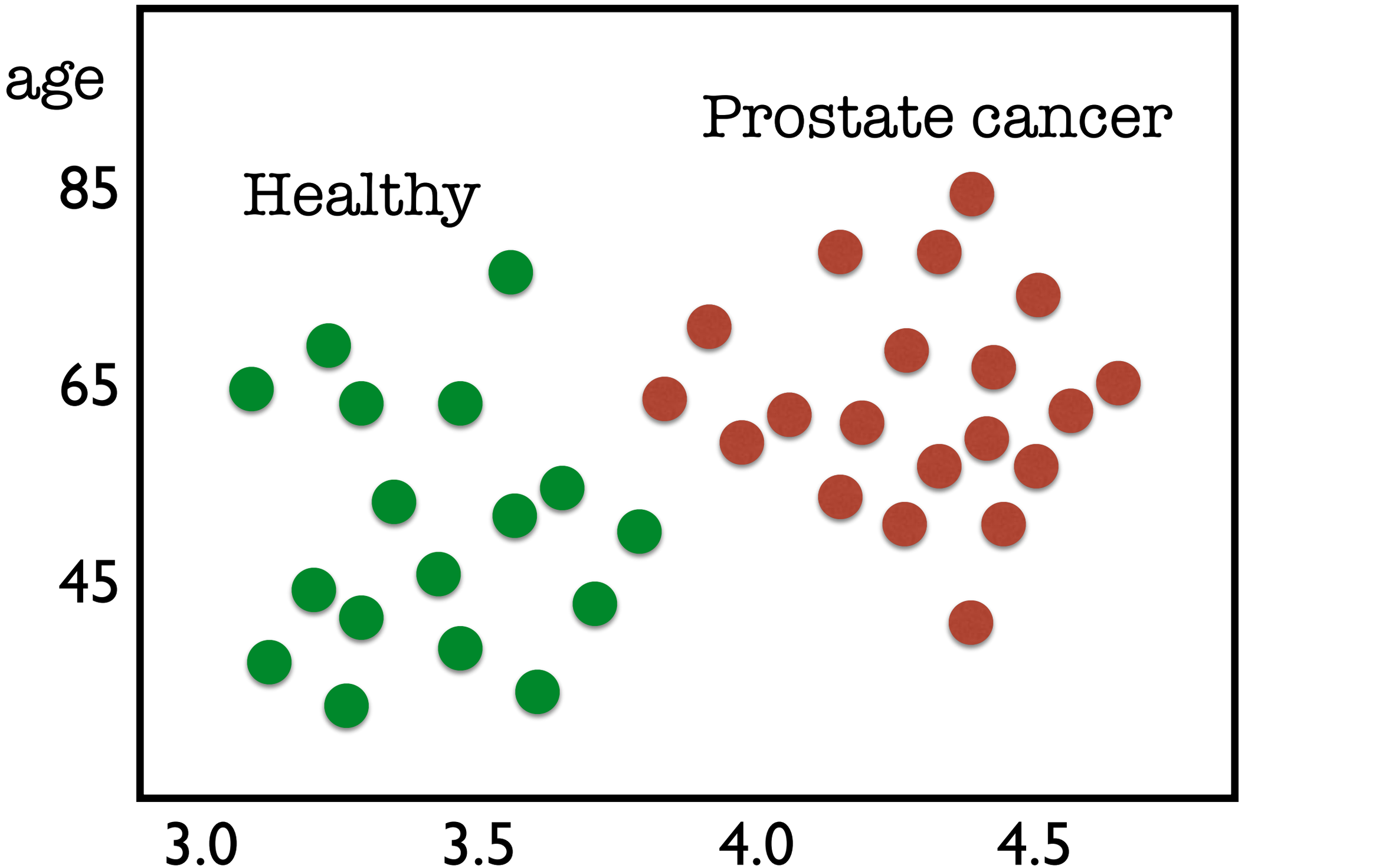
Age	Prostate Specific Antigen (a blood test reading)	Got Prostate Cancer
45	6.0 ng/mL	Yes
61	5.5 ng/mL	Yes
59	4.9 ng/mL	Yes
67	4.3 ng/mL	Yes
72	3.9 ng/mL	Yes
89	3.5 ng/mL	Yes
95	3.1 ng/mL	Yes
39	3.9 ng/mL	No
49	3.4 ng/mL	No
47	3.2 ng/mL	No
62	2.1 ng/mL	No
49	3.8 ng/mL	?
58	4.3 ng/mL	?
88	4.1 ng/mL	?
31	2.1 ng/mL	?

Age	Prostate Specific Antigen (a blood test reading)	Got Prostate Cancer
45	6.0 ng/mL	Yes
61	5.5 ng/mL	Yes
59	4.9 ng/mL	Yes
67	4.3 ng/mL	Yes
72	3.9 ng/mL	Yes
89	3.5 ng/mL	Yes
95	3.1 ng/mL	Yes
39	3.9 ng/mL	No
49	3.4 ng/mL	No
47	3.2 ng/mL	No
62	2.1 ng/mL	No
49	3.8 ng/mL	No?
58	4.3 ng/mL	borderline?
88	4.1 ng/mL	Yes?
31	2.1 ng/mL	No?

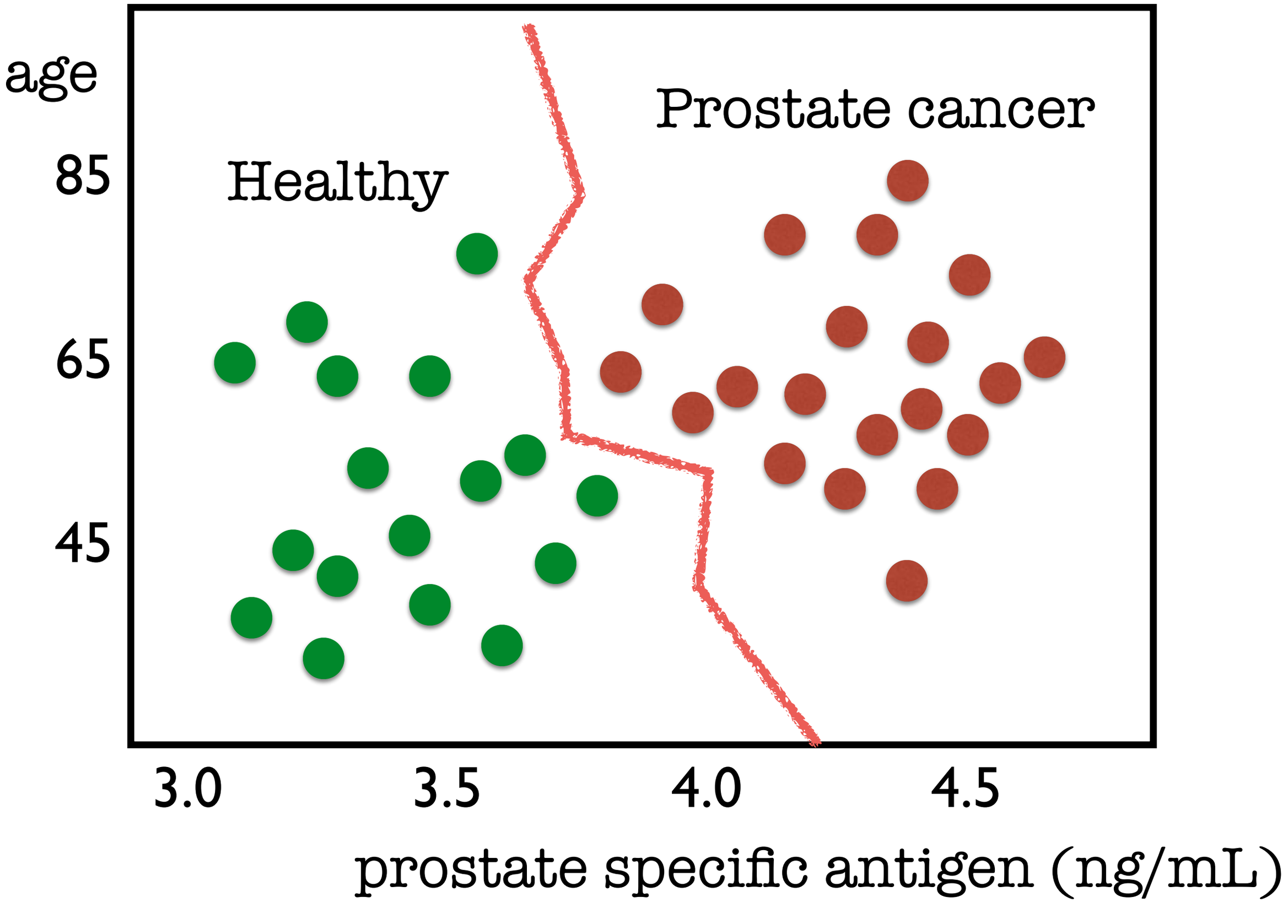
Prostate cancer prediction



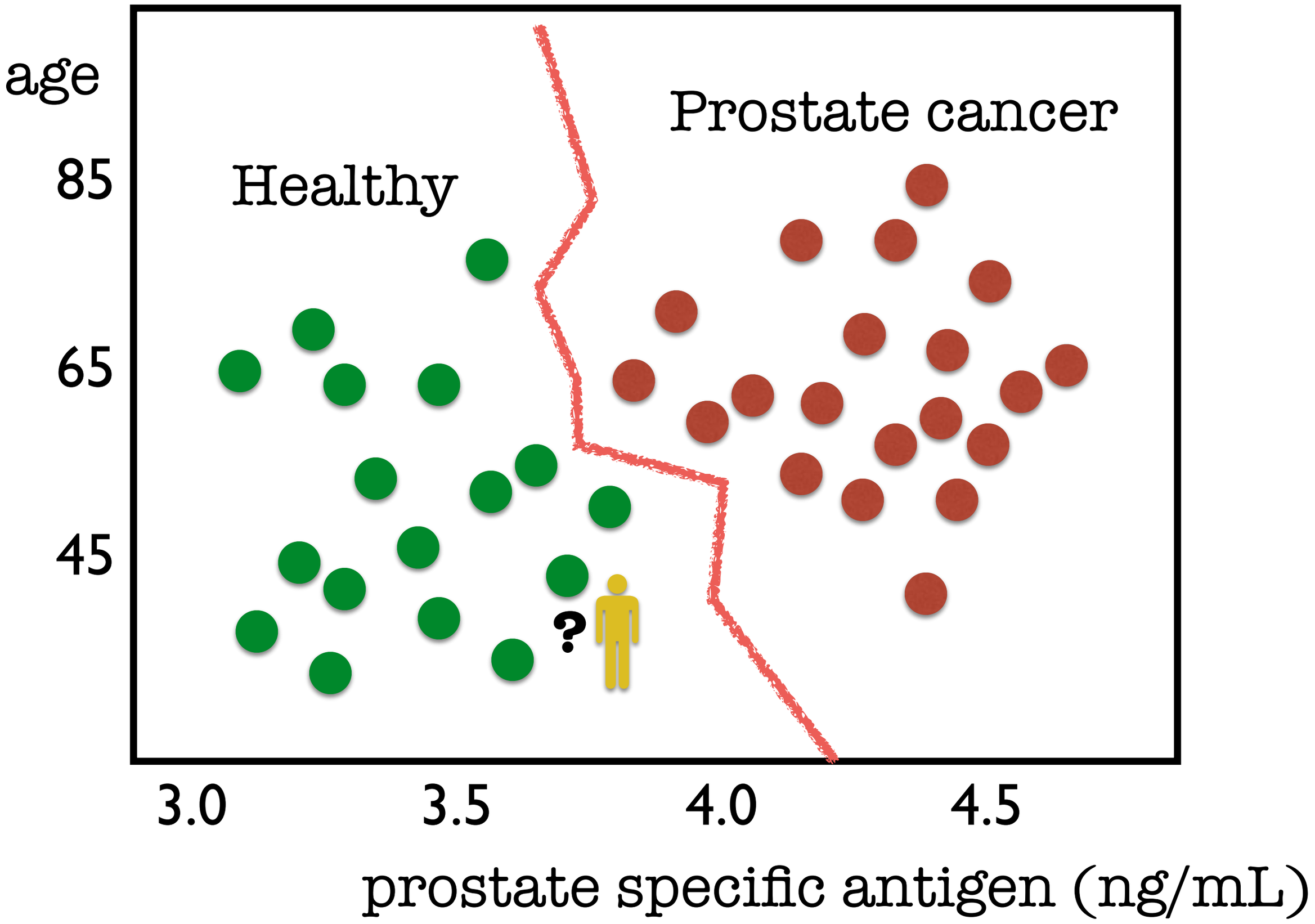
Prostate cancer prediction



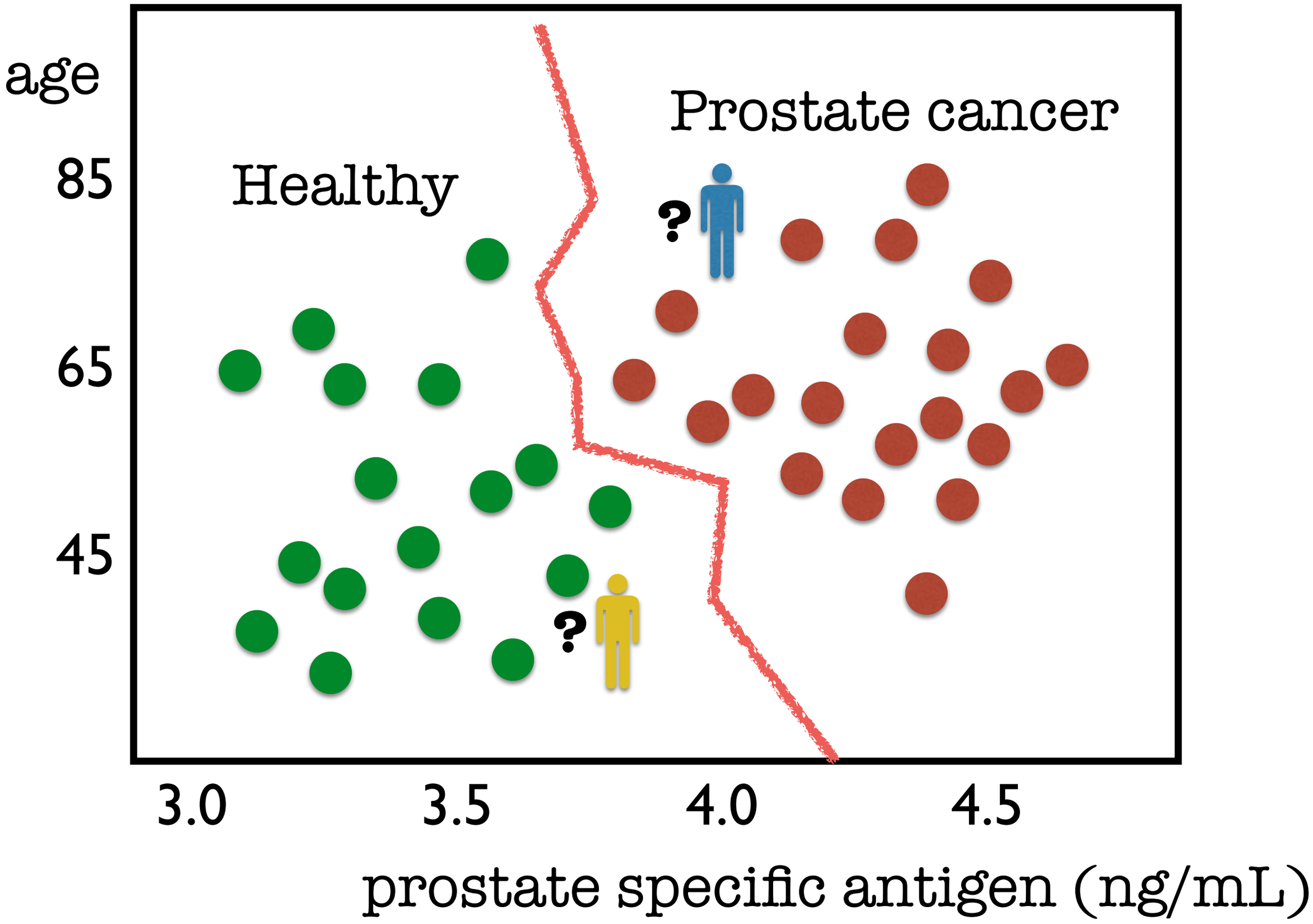
Prostate cancer prediction



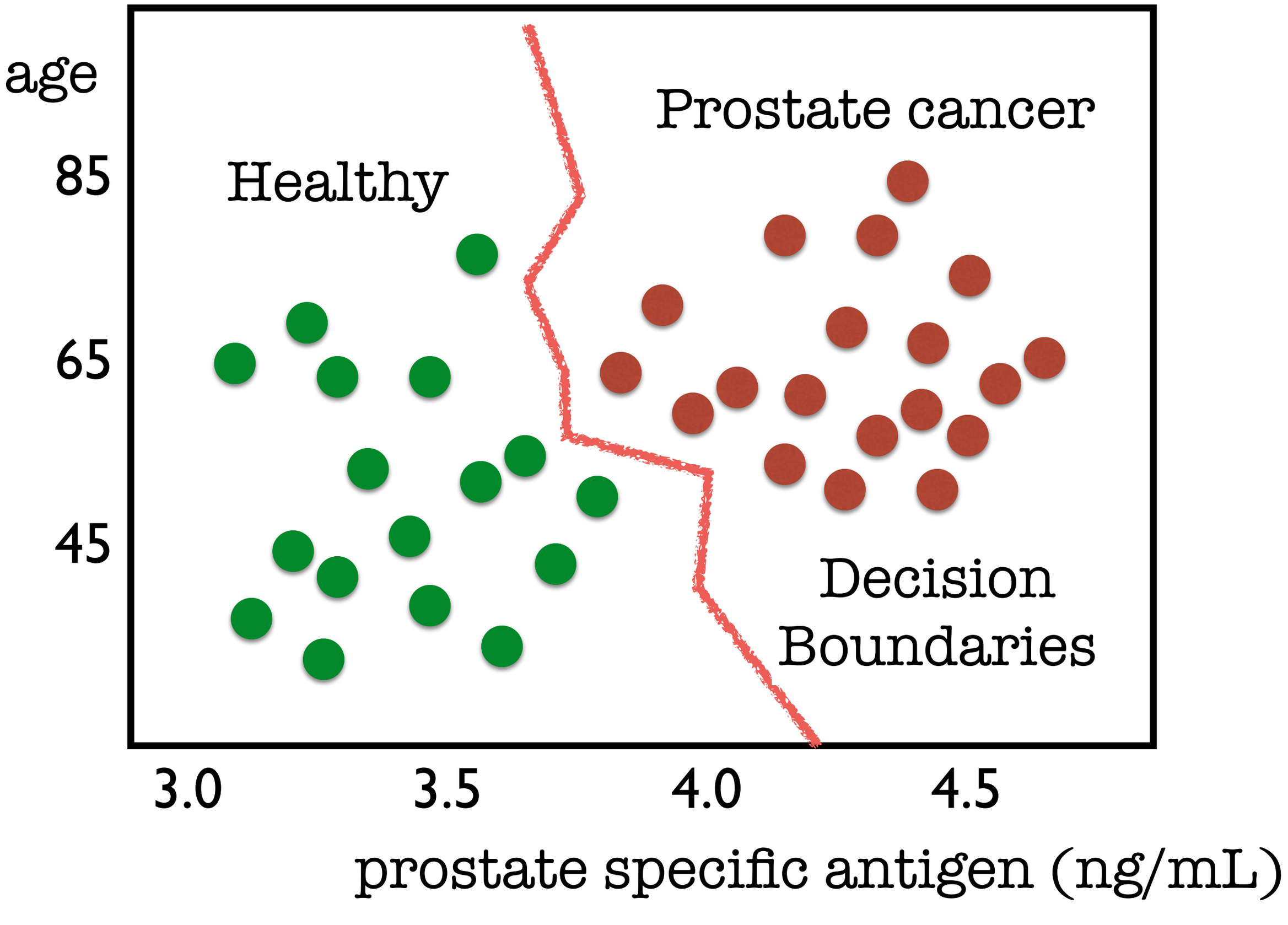
Prostate cancer prediction



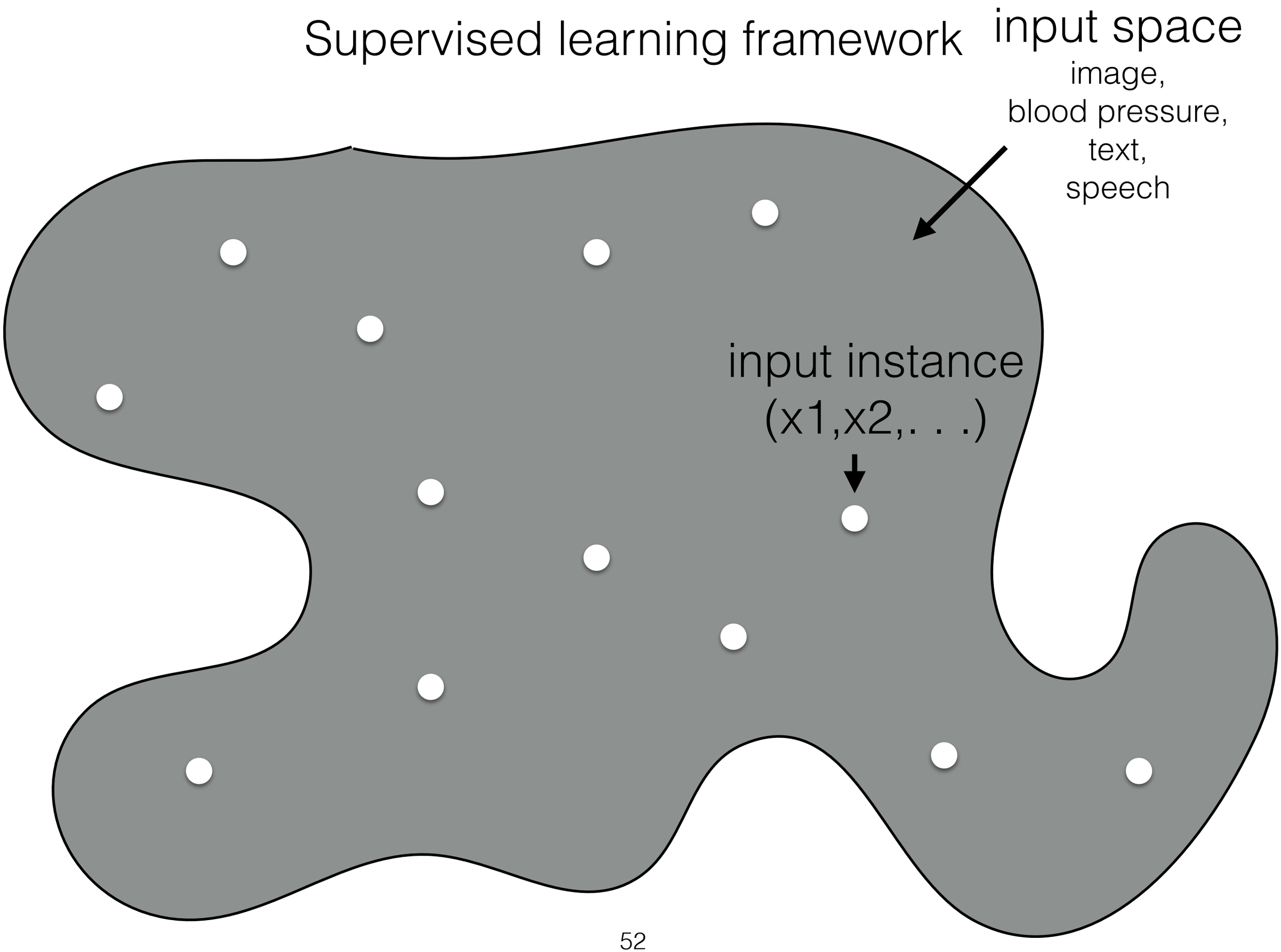
Prostate cancer prediction



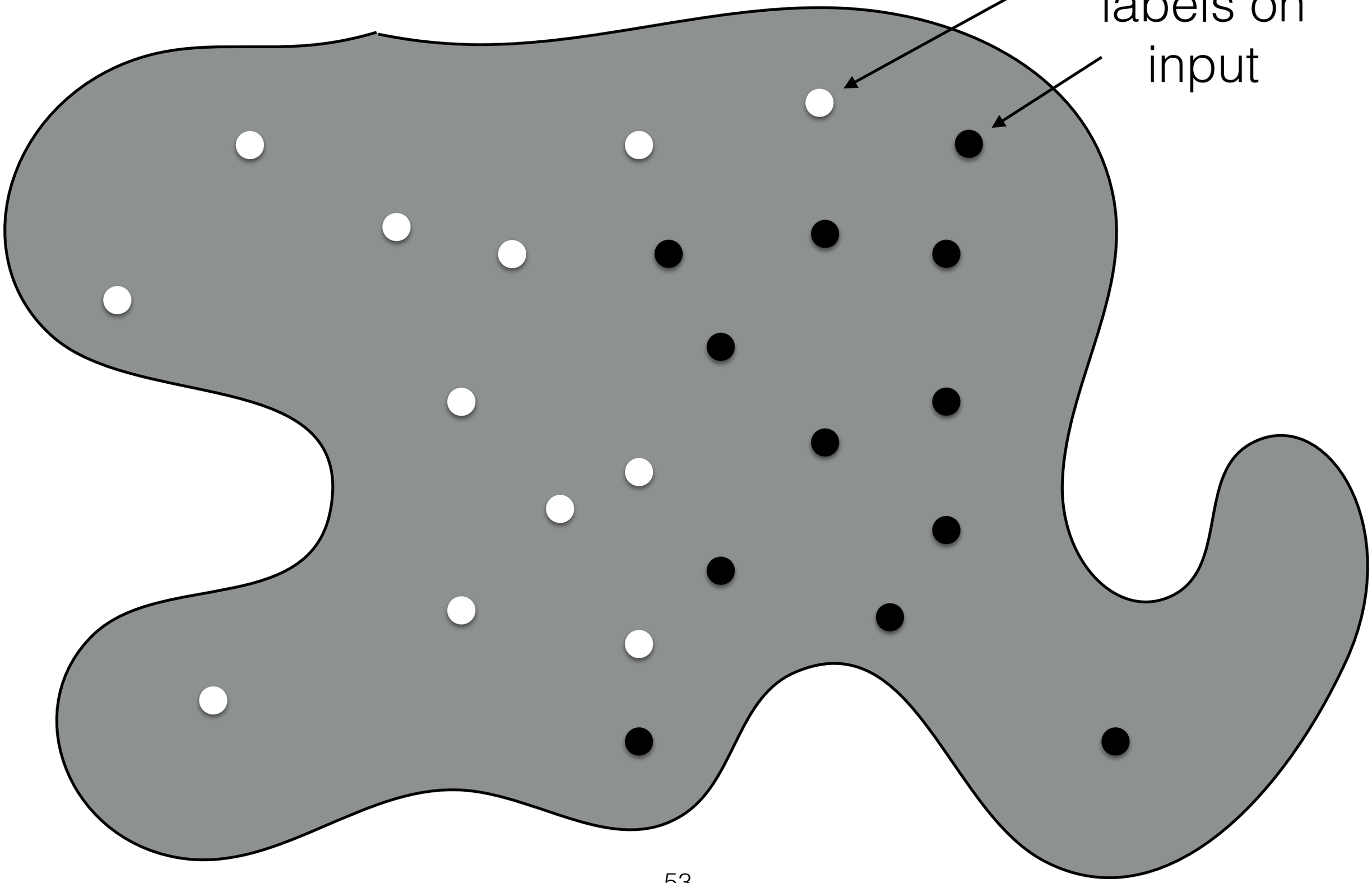
Prostate cancer prediction



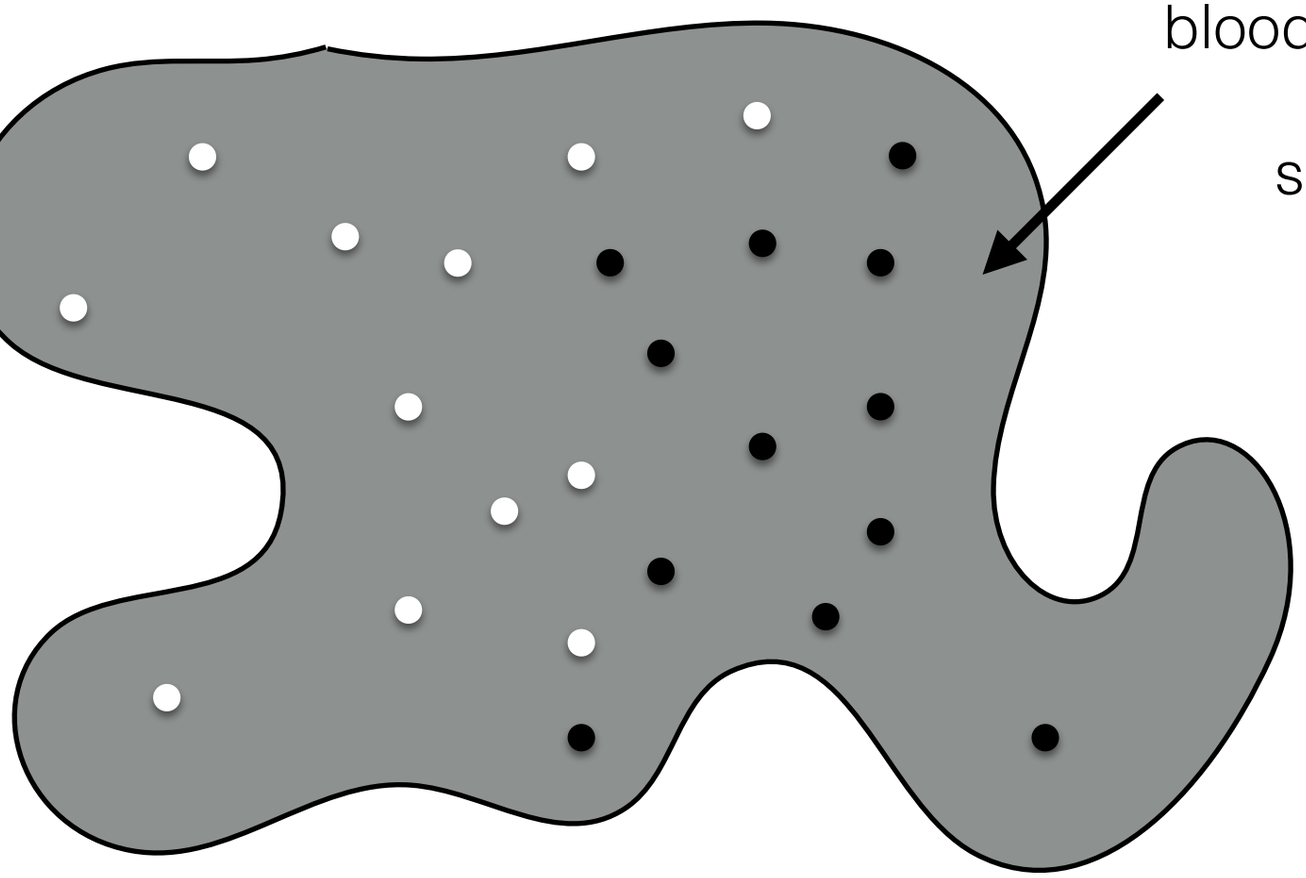
Supervised learning framework



Supervised learning framework



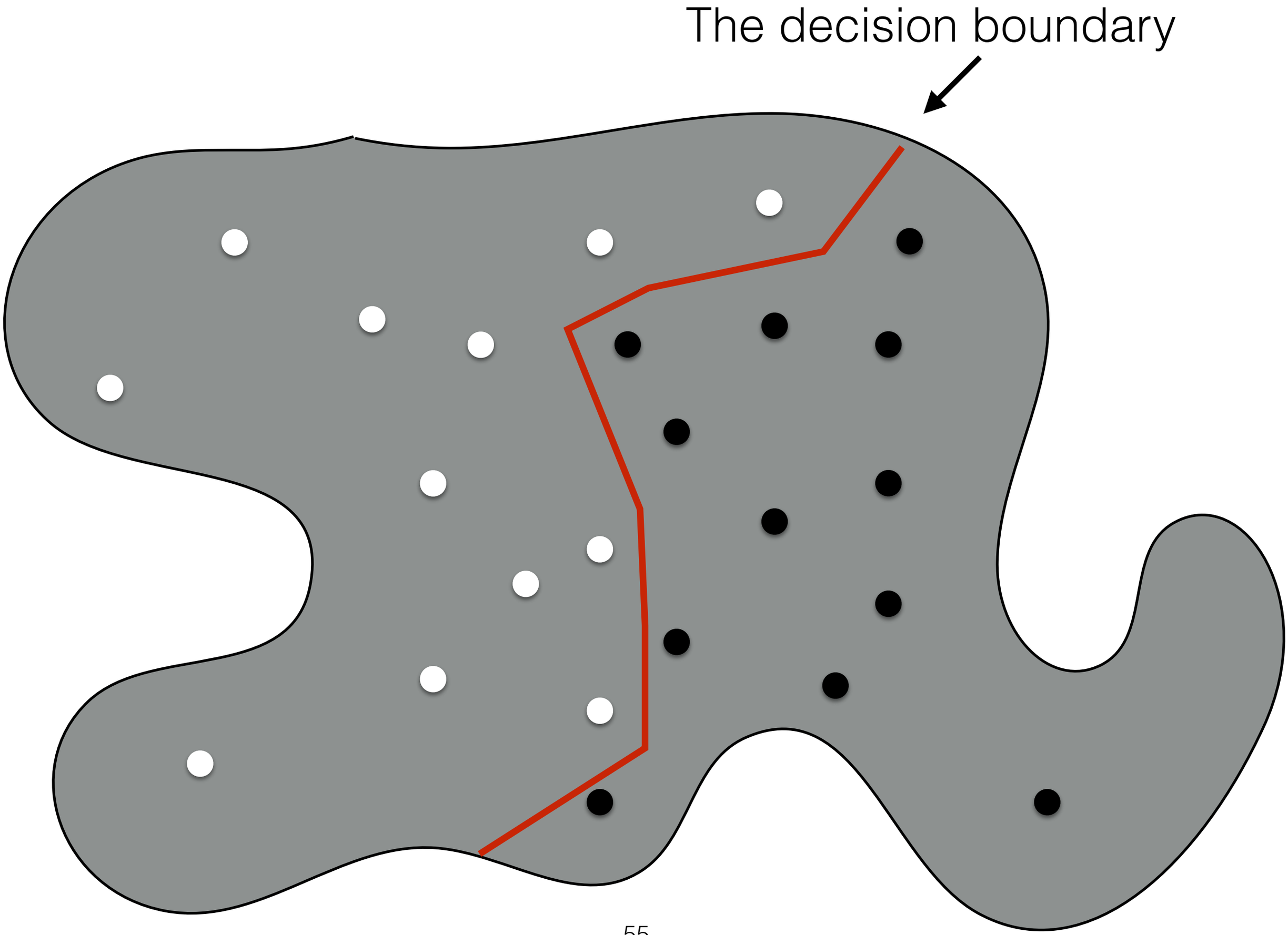
Supervised learning framework

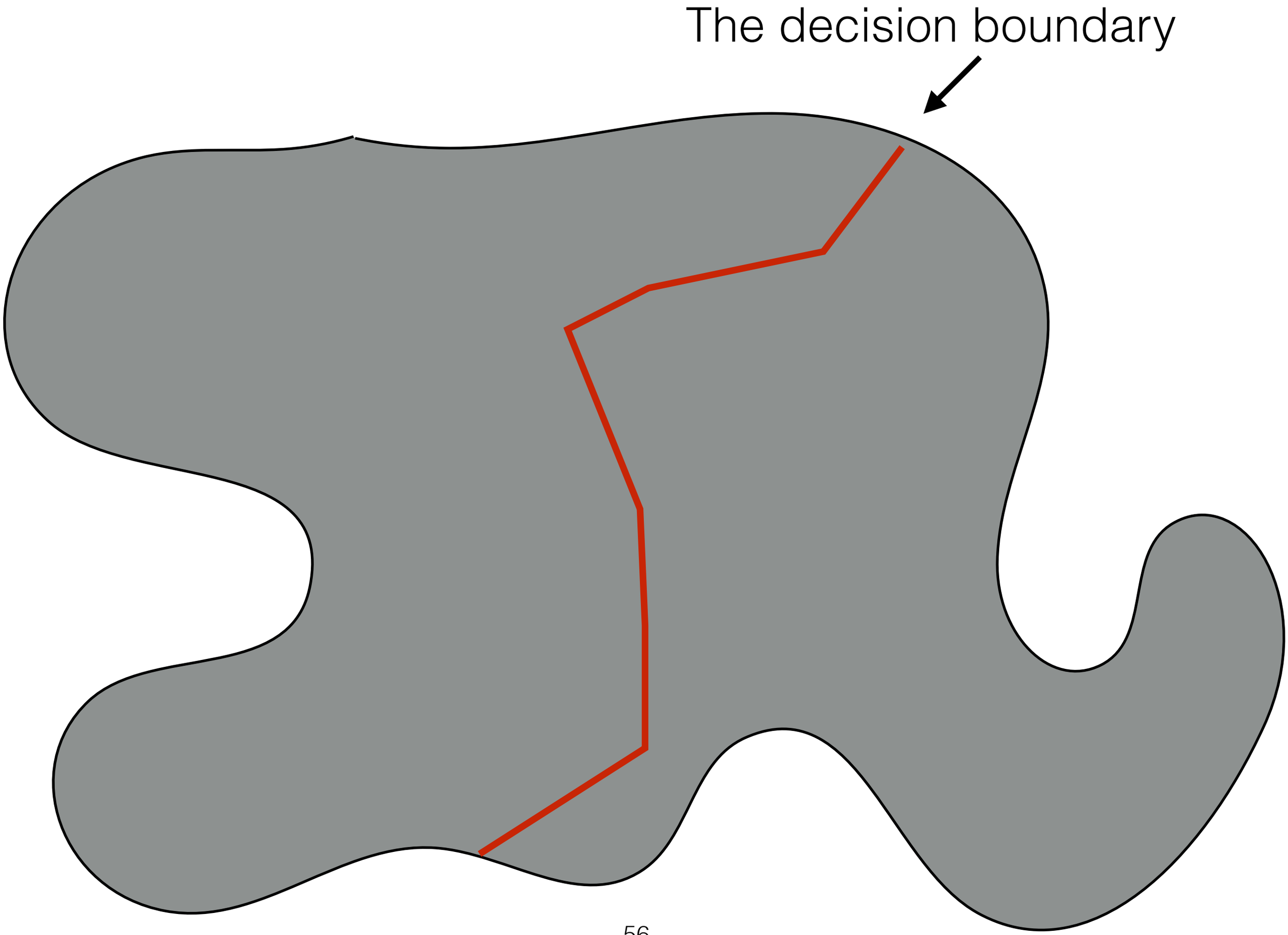


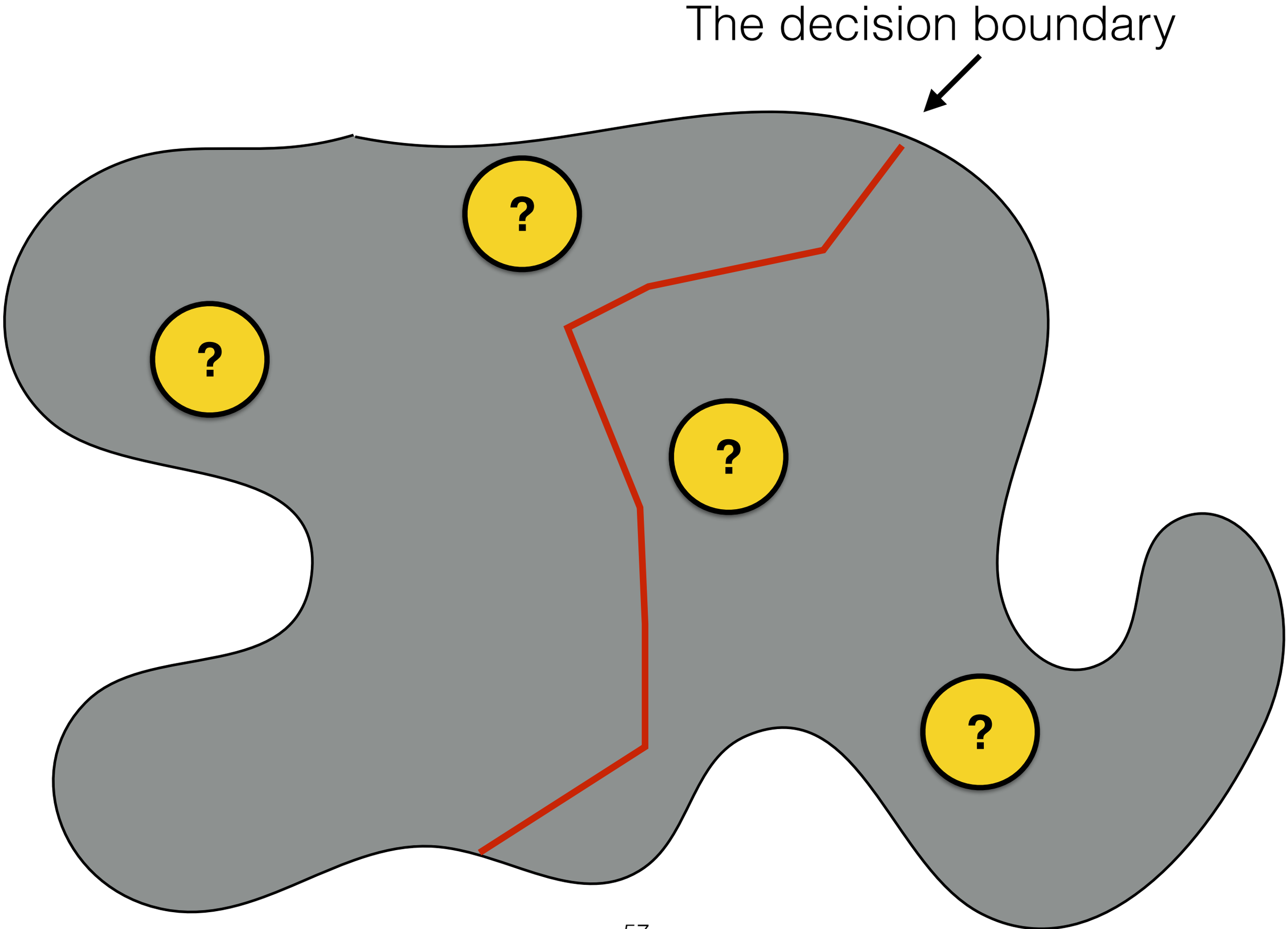
input:
a point
in input space

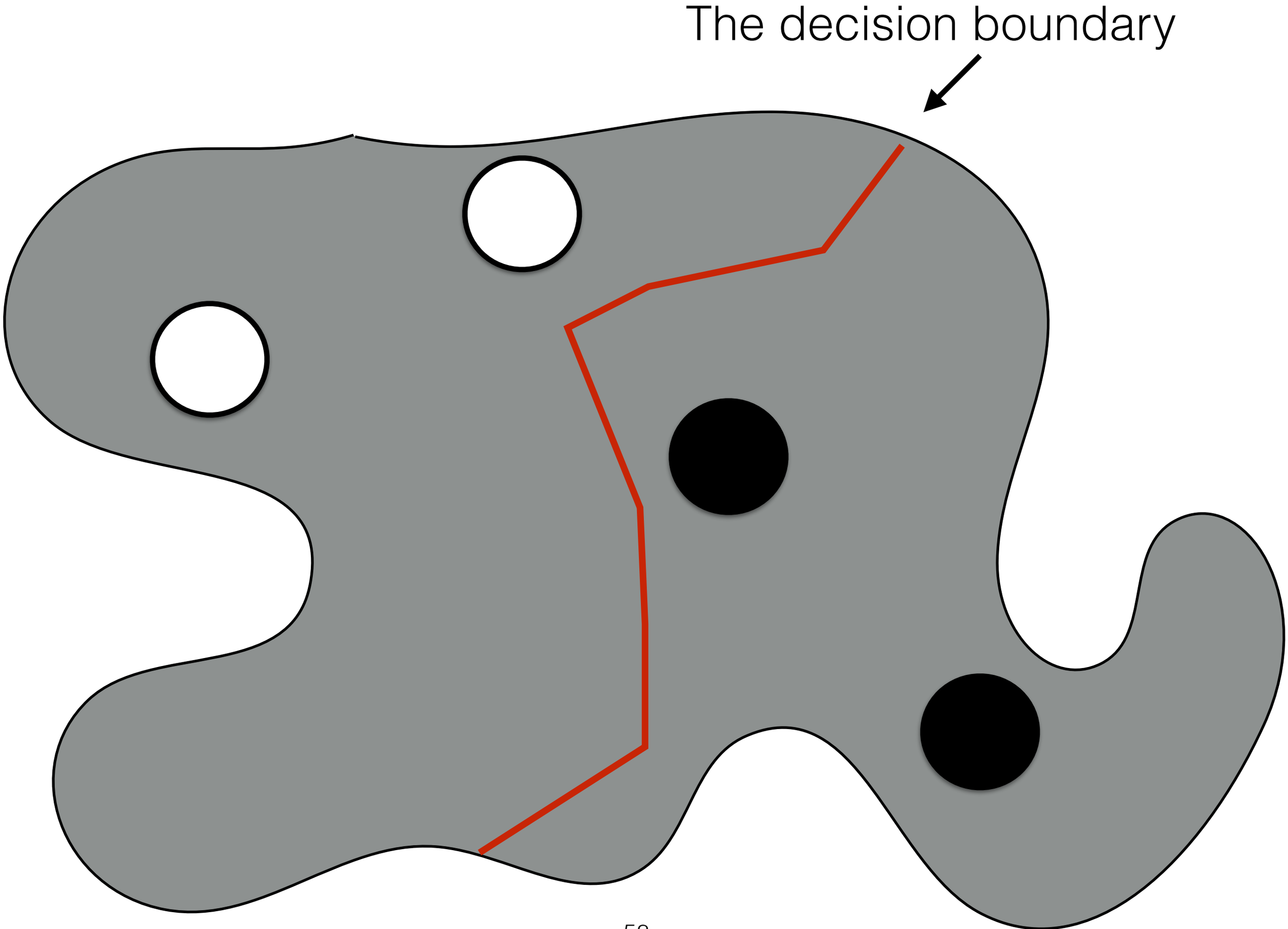
Machine

output:
black
or
white


The decision boundary


The decision boundary

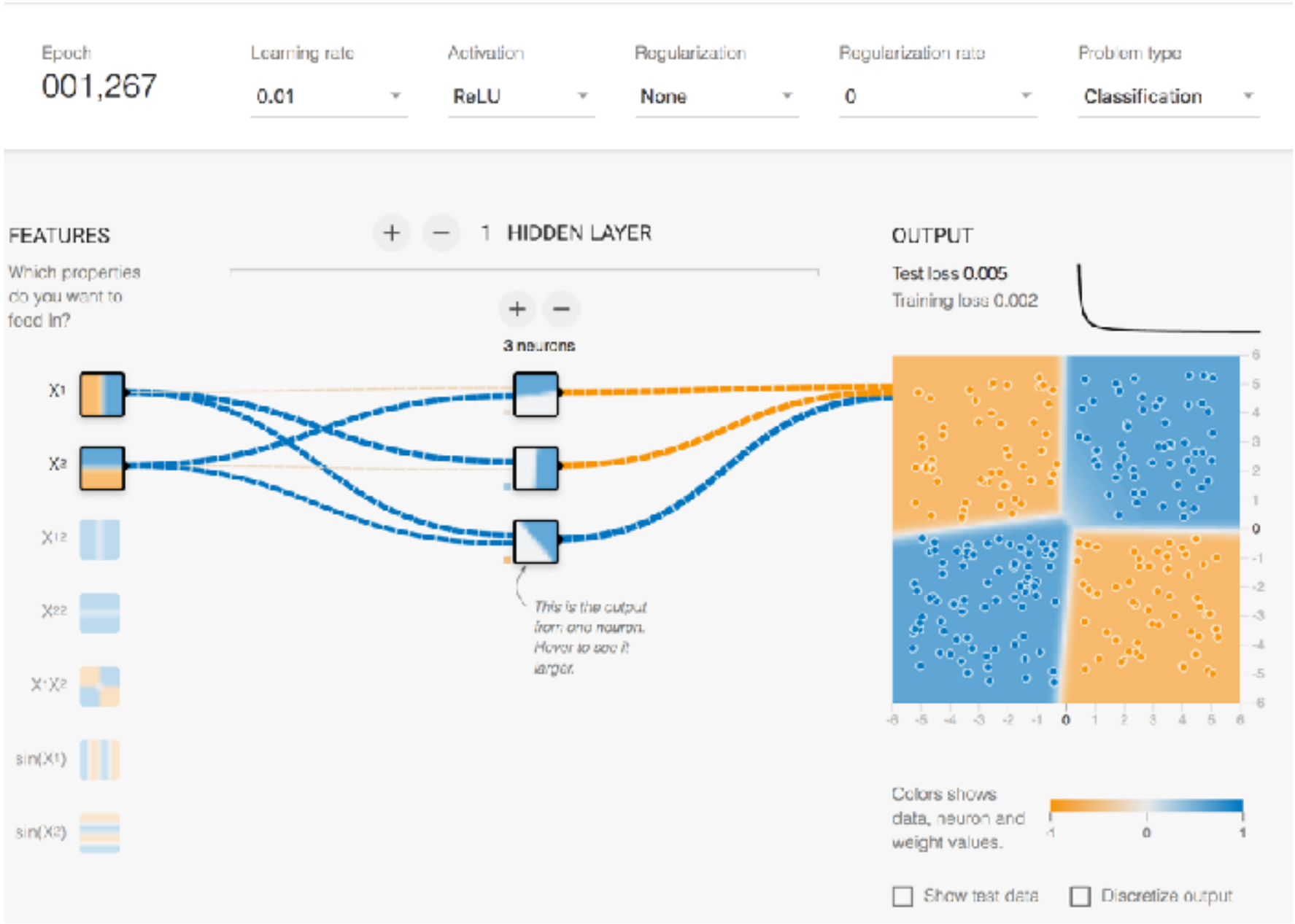

The decision boundary
?
?
?
?


The decision boundary

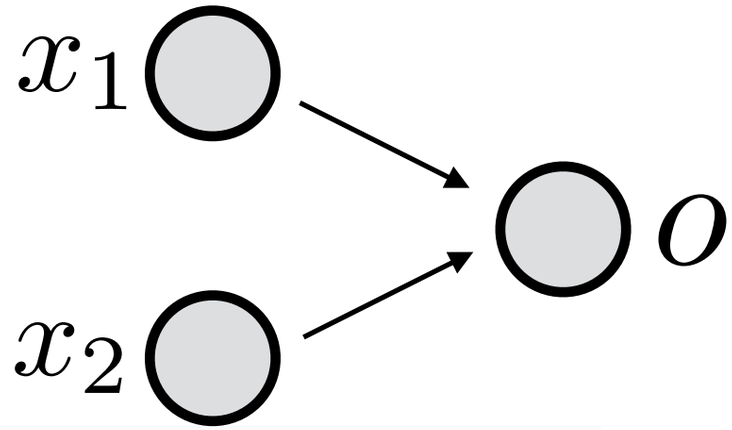
Animation @ playground
search for
“playground tensorflow"

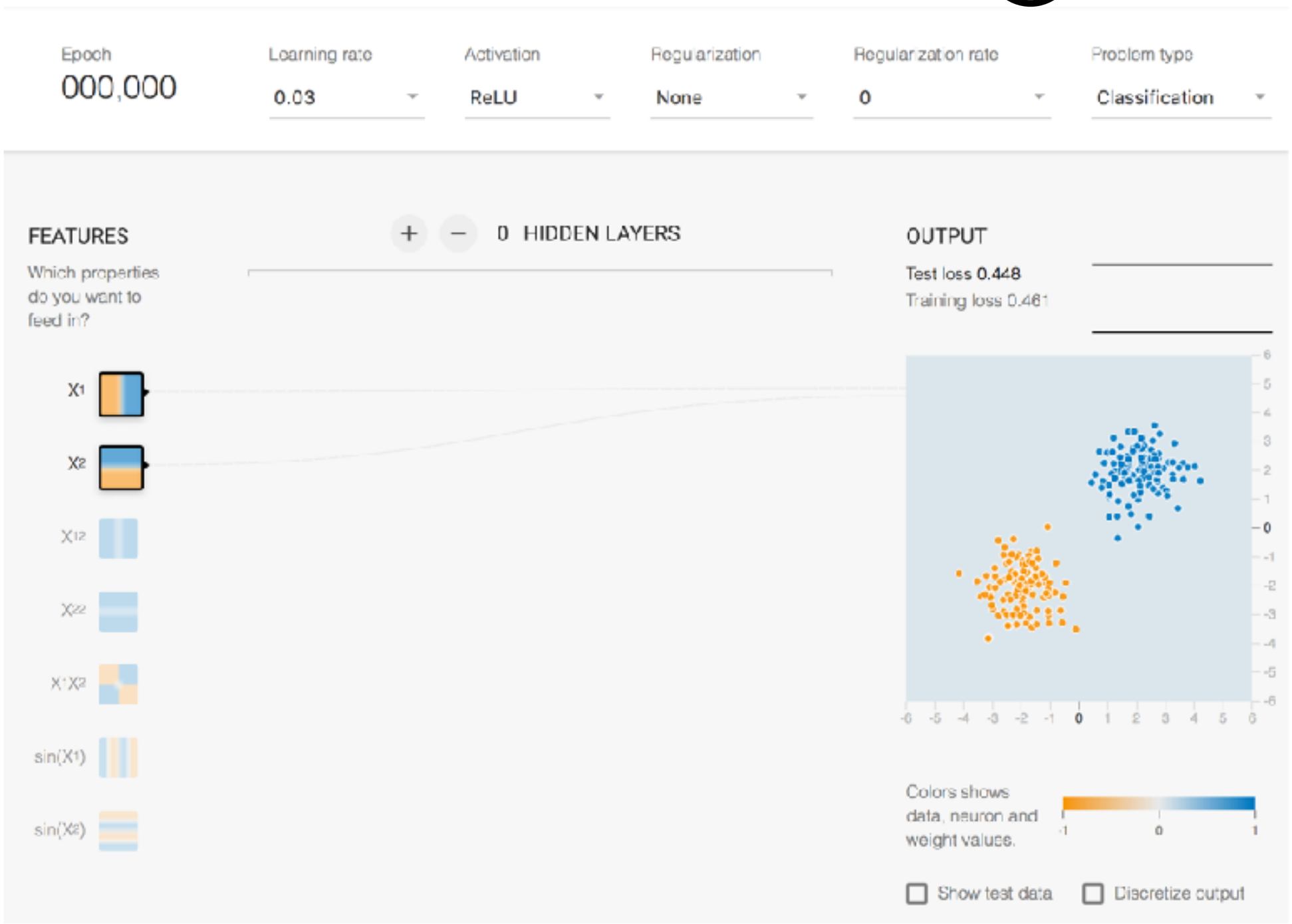
introduction

Different functionalities of applet

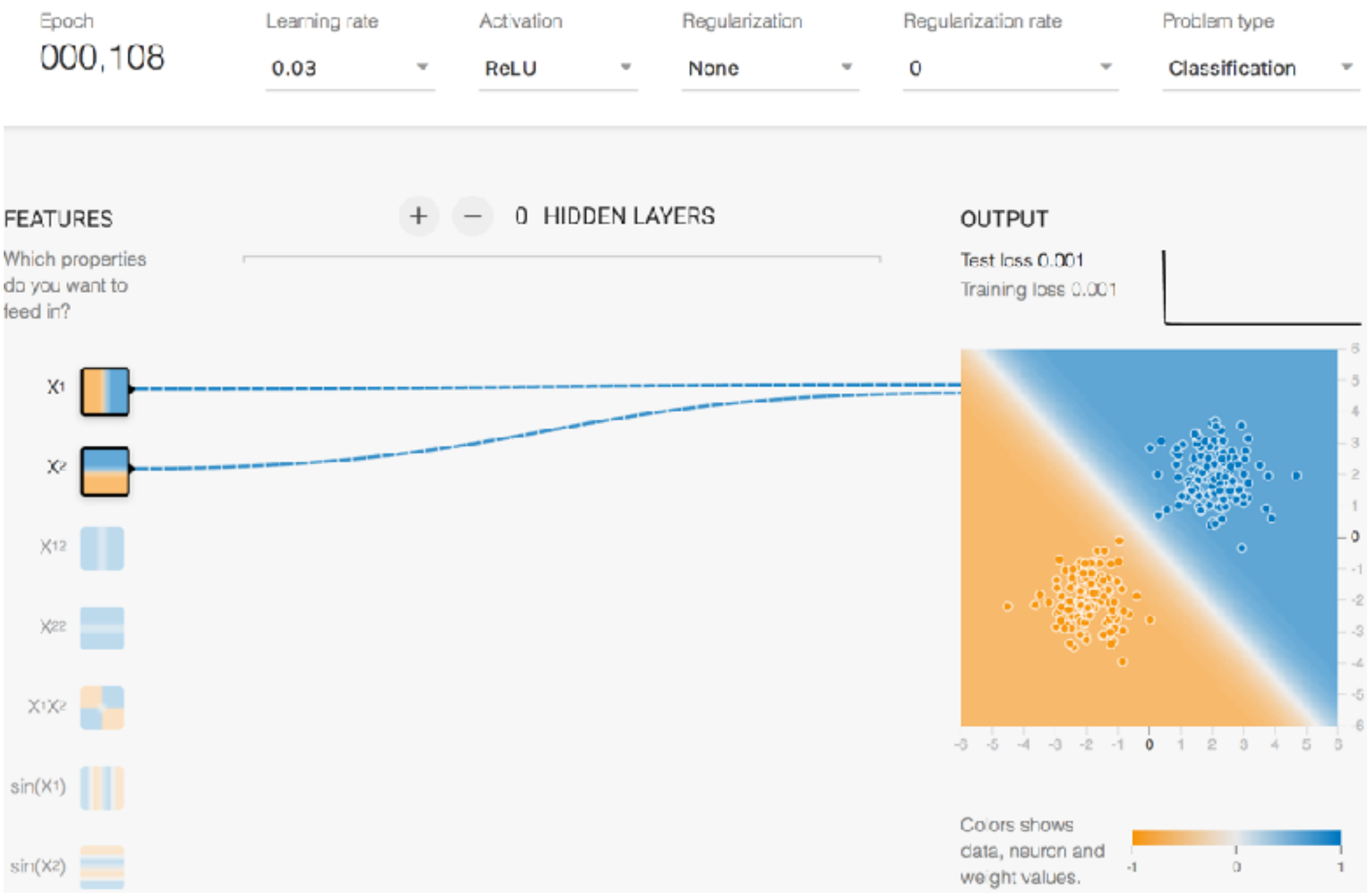


question

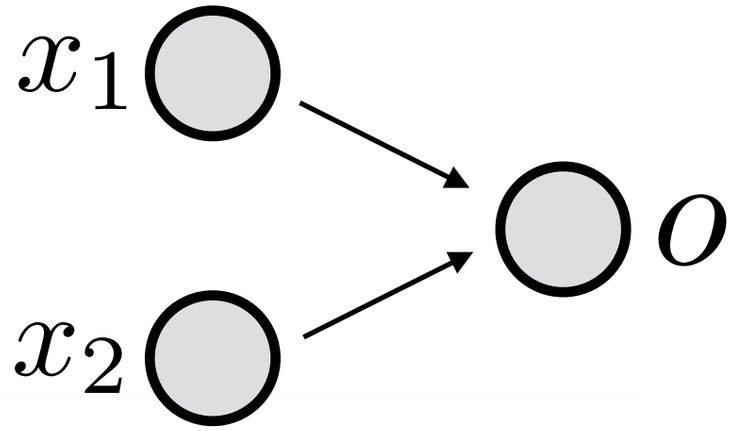

x_1
x_2
o

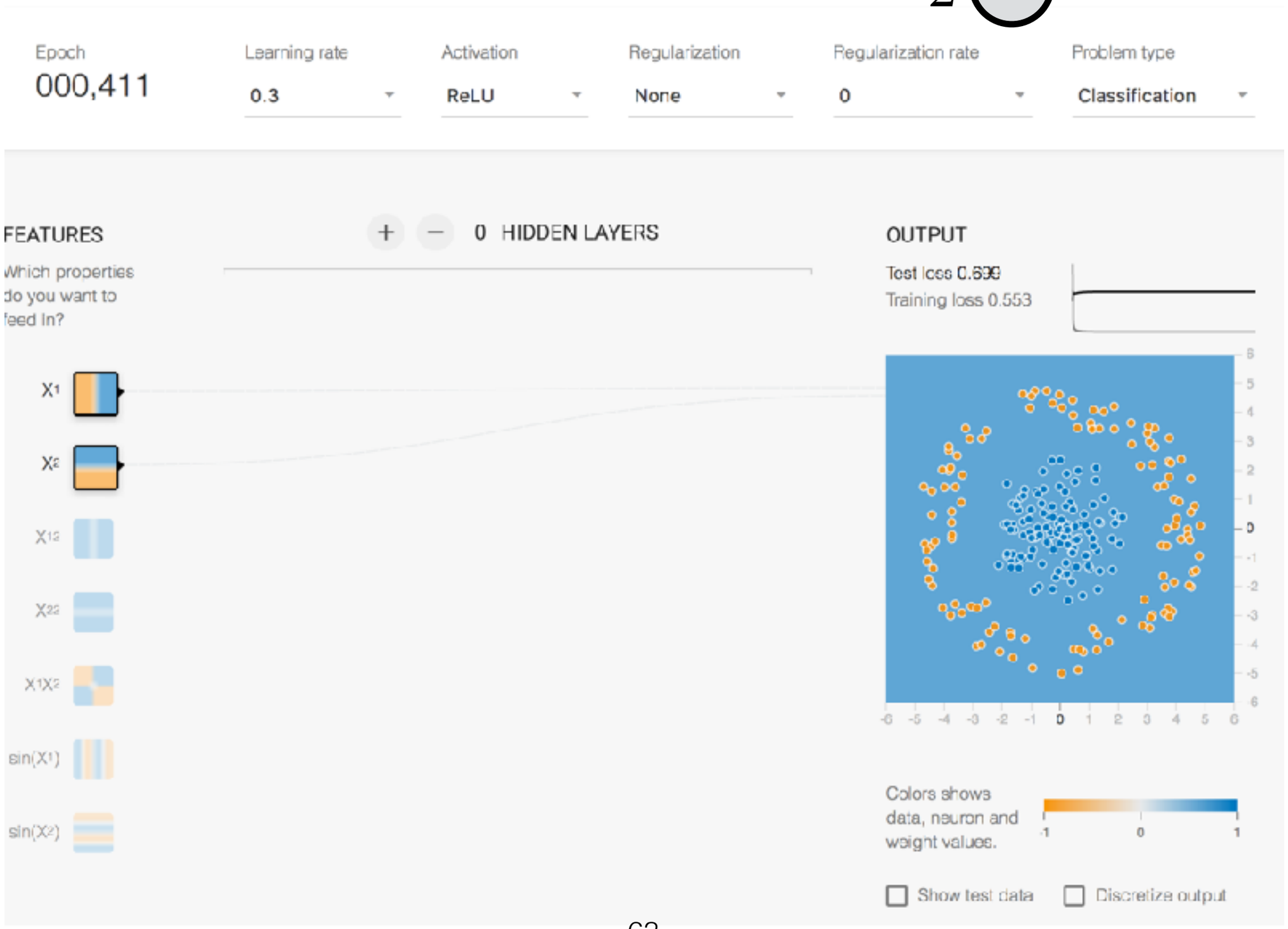

Epoch
000,000
Learning rate
0.03
Activation
ReLU
Regularization
None
Regularization rate
0
Problem type
Classification
FEATURES
Which properties do you want to feed in?
+ − 0 HIDDEN LAYERS
OUTPUT
Test loss 0.448
Training loss 0.461
X1
X2
X1²
X2²
X1X2
sin(X1)
sin(X2)
Colors shows data, neuron and weight values.
-1
0
1
Show test data
Discretize output

answer

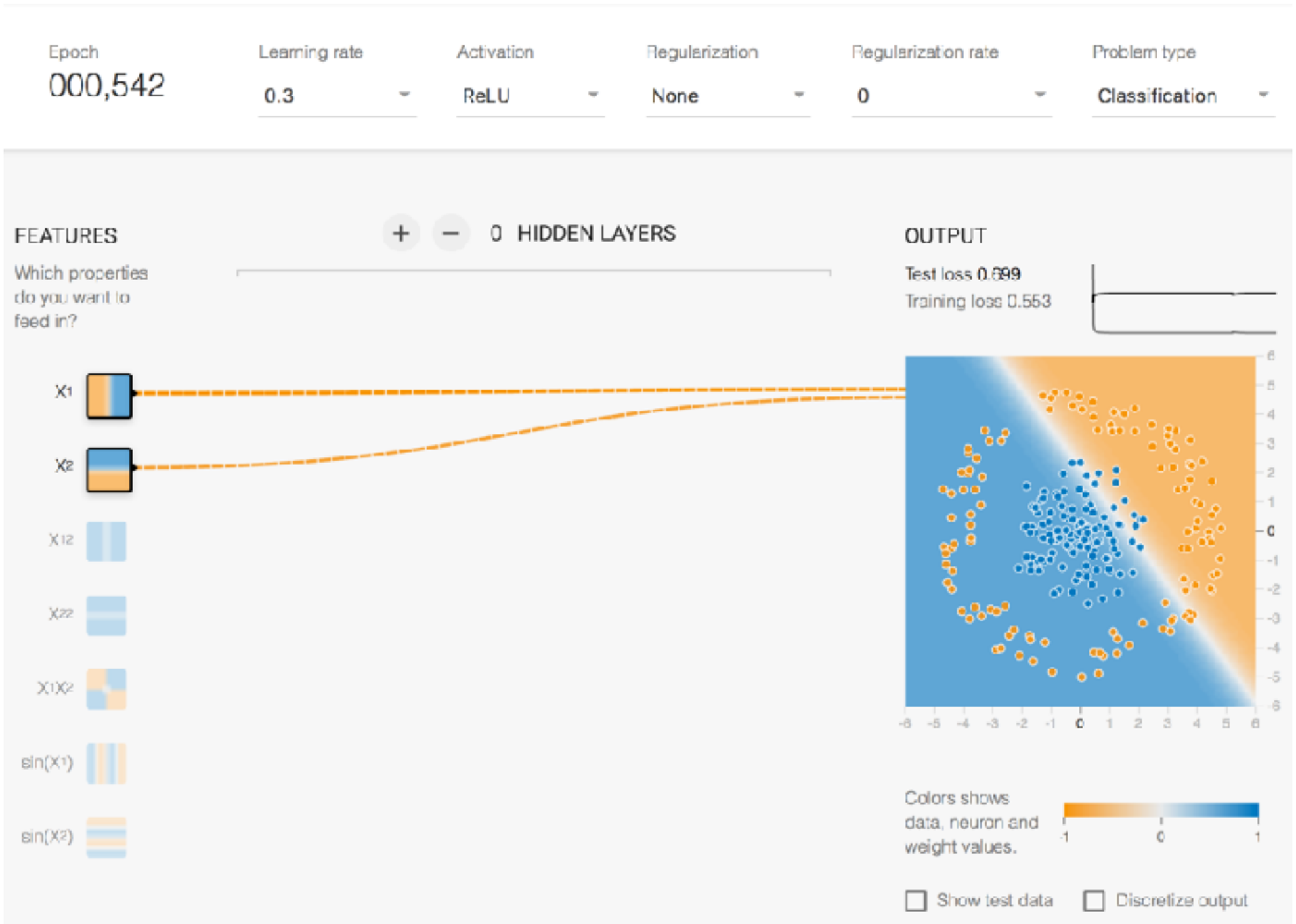

Epoch
000,108
Learning rate
0.03
Activation
ReLU
Regularization
None
Regularization rate
0
Problem type
Classification
FEATURES
Which properties do you want to feed in?
0 HIDDEN LAYERS
OUTPUT
Test loss 0.001
Training loss 0.001
X1
X2
X12
X22
X1X2
sin(X1)
sin(X2)
Colors shows data, neuron and weight values.

question

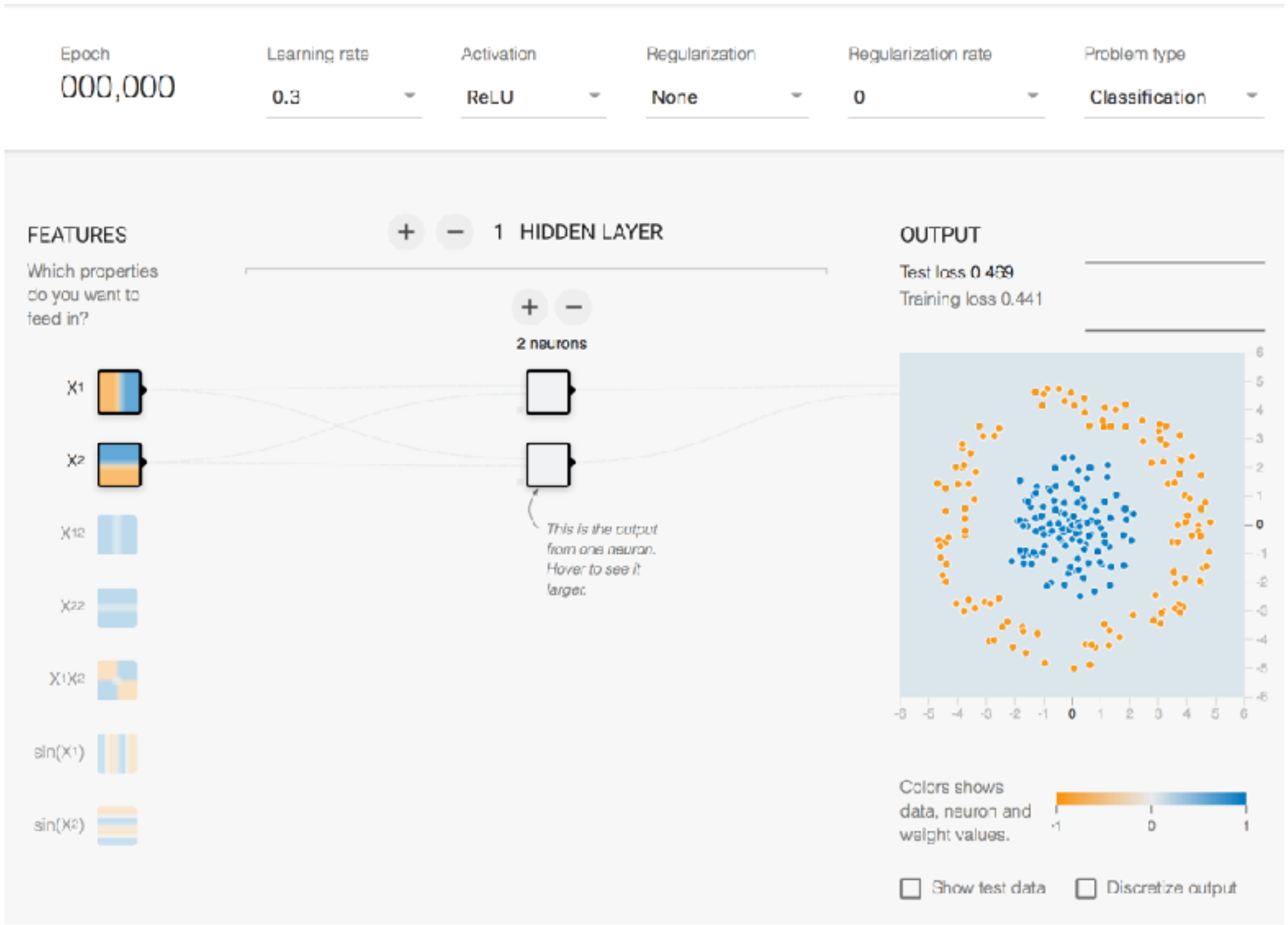

x_1
x_2
o


Epoch
000,411
Learning rate
0.3
Activation
ReLU
Regularization
None
Regularization rate
0
Problem type
Classification
FEATURES
Which properties do you want to feed in?
0 HIDDEN LAYERS
OUTPUT
Test loss 0.539
Training loss 0.553
X1
X2
X1²
X2²
X1X2
sin(X1)
sin(X2)
Colors shows data, neuron and weight values.
Show test data
Discretize output

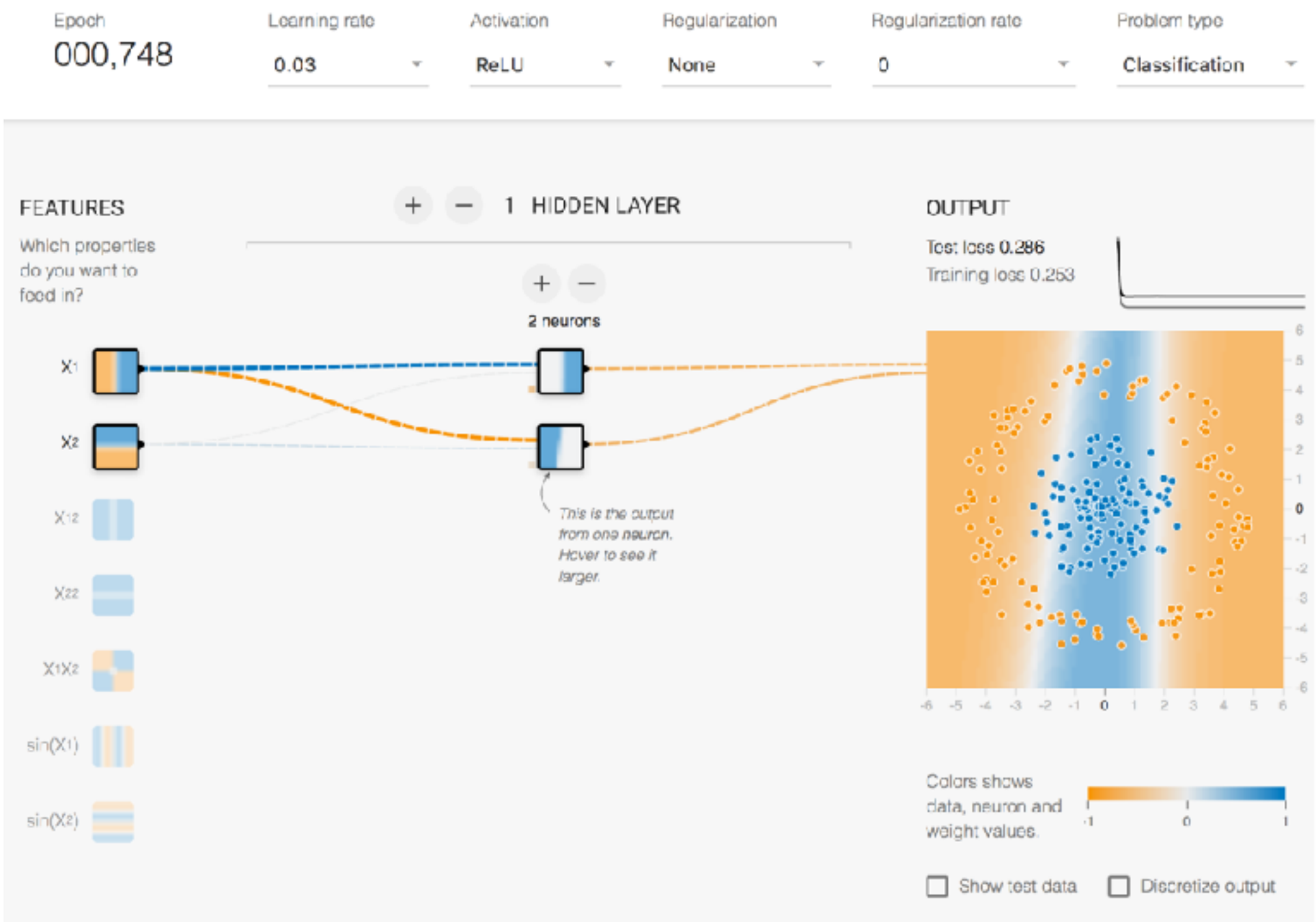
answer


Epoch
000,542
Learning rate
0.3
Activation
ReLU
Regularization
None
Regularization rate
0
Problem type
Classification
FEATURES
Which properties do you want to feed in?
0 HIDDEN LAYERS
OUTPUT
Test loss 0.699
Training loss 0.553
X1
X2
X12
X22
X1X2
sin(X1)
sin(X2)
Colors shows data, neuron and weight values.
Show test data
Discretize output

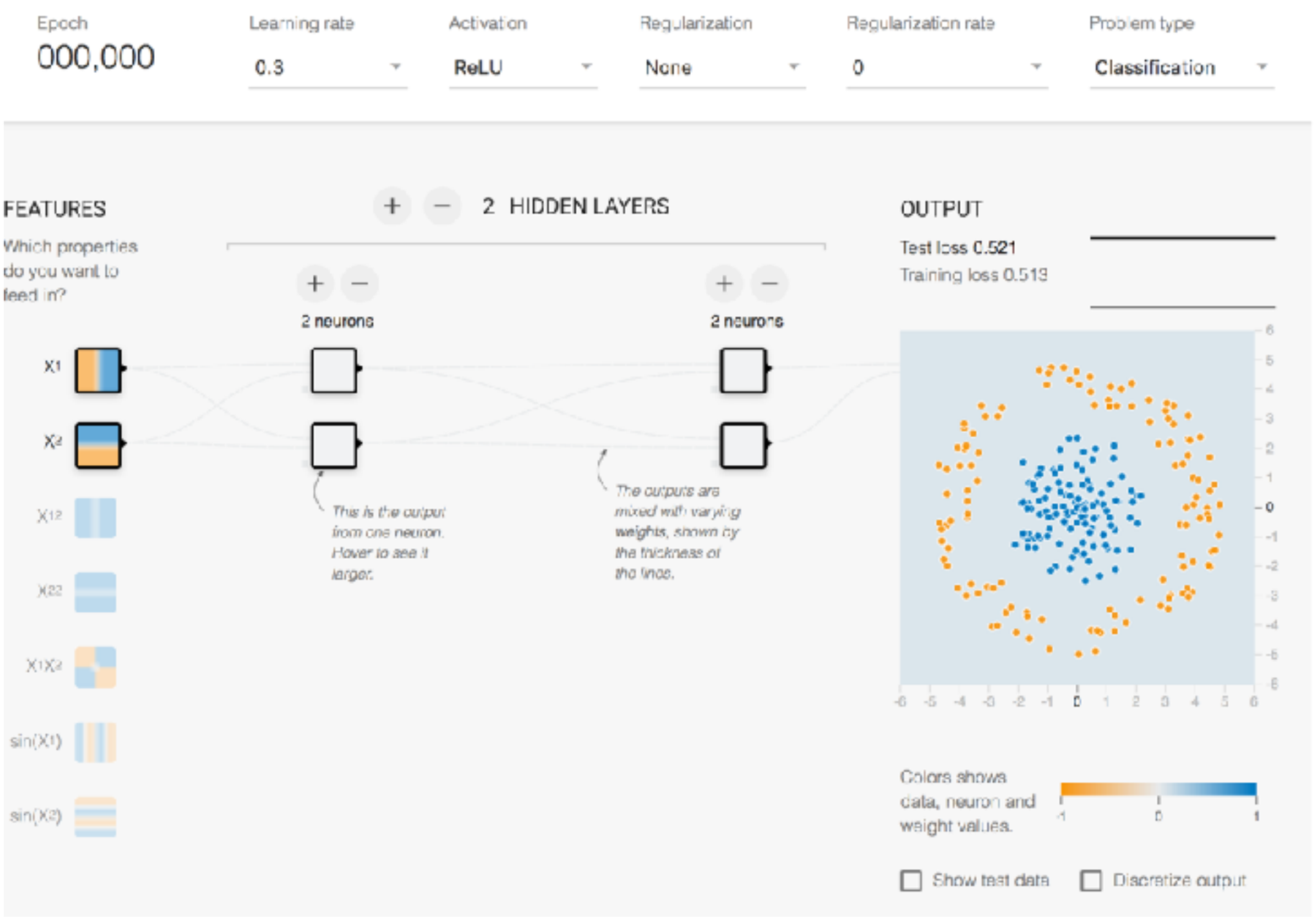
question


Epoch
000,000
Learning rate
0.3
Activation
ReLU
Regularization
None
Regularization rate
0
Problem type
Classification
FEATURES
Which properties do you want to feed in?
1 HIDDEN LAYER
2 neurons
X1
X2
X1²
X2²
X1X2
sin(X1)
sin(X2)
This is the output from one neuron. Hover to see it larger.
OUTPUT
Test loss 0.459
Training loss 0.441
Colors shows data, neuron and weight values.
-1
0
1
Show test data
Discretize output

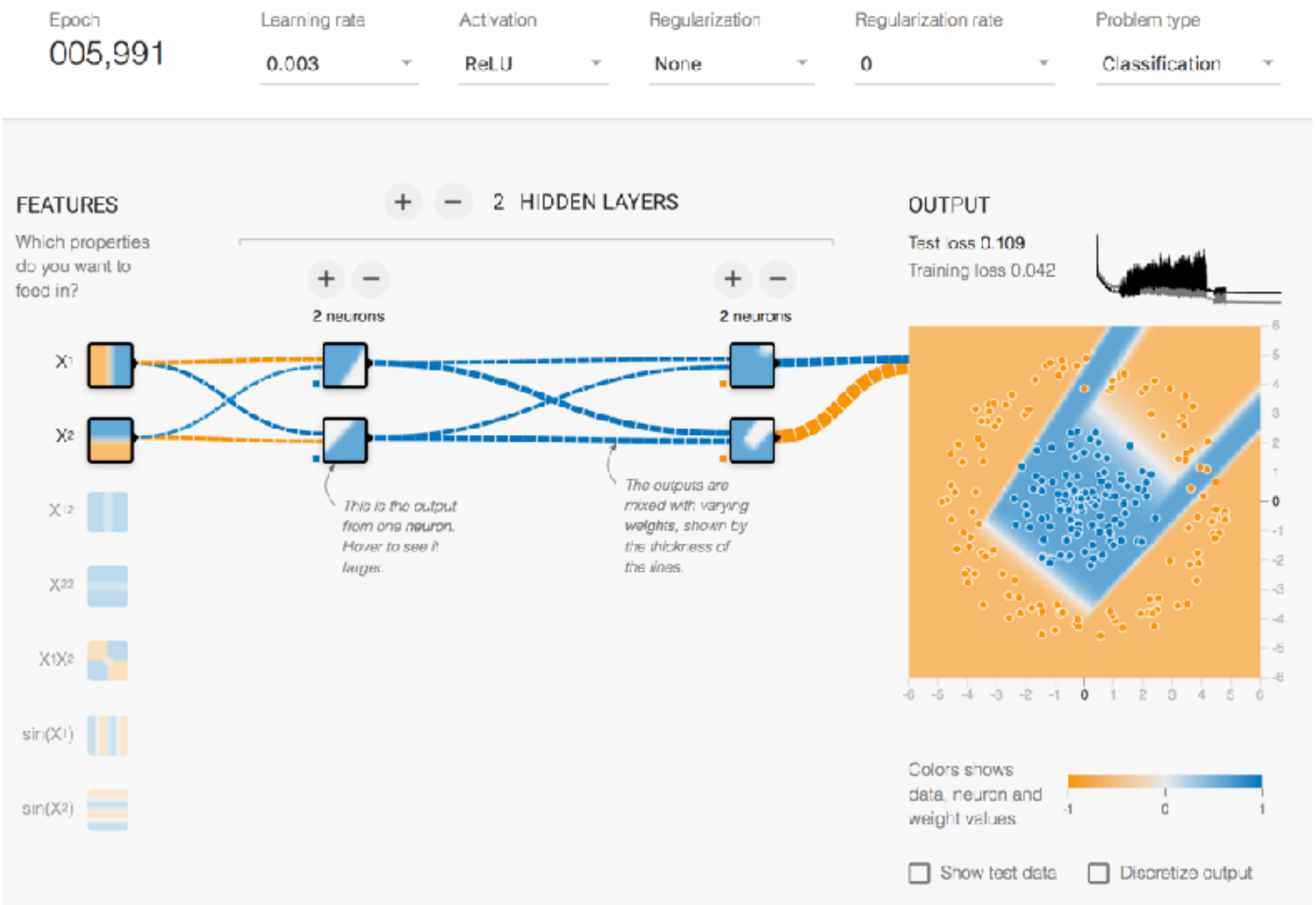
answer


Epoch
000,748
Learning rate
0.03
Activation
ReLU
Regularization
None
Regularization rate
0
Problem type
Classification
FEATURES
Which properties do you want to feed in?
1 HIDDEN LAYER
2 neurons
X1
X2
X1²
X2²
X1X2
sin(X1)
sin(X2)
This is the output from one neuron. Hover to see it larger.
OUTPUT
Test loss 0.286
Training loss 0.253
Colors shows data, neuron and weight values.
Show test data
Discretize output

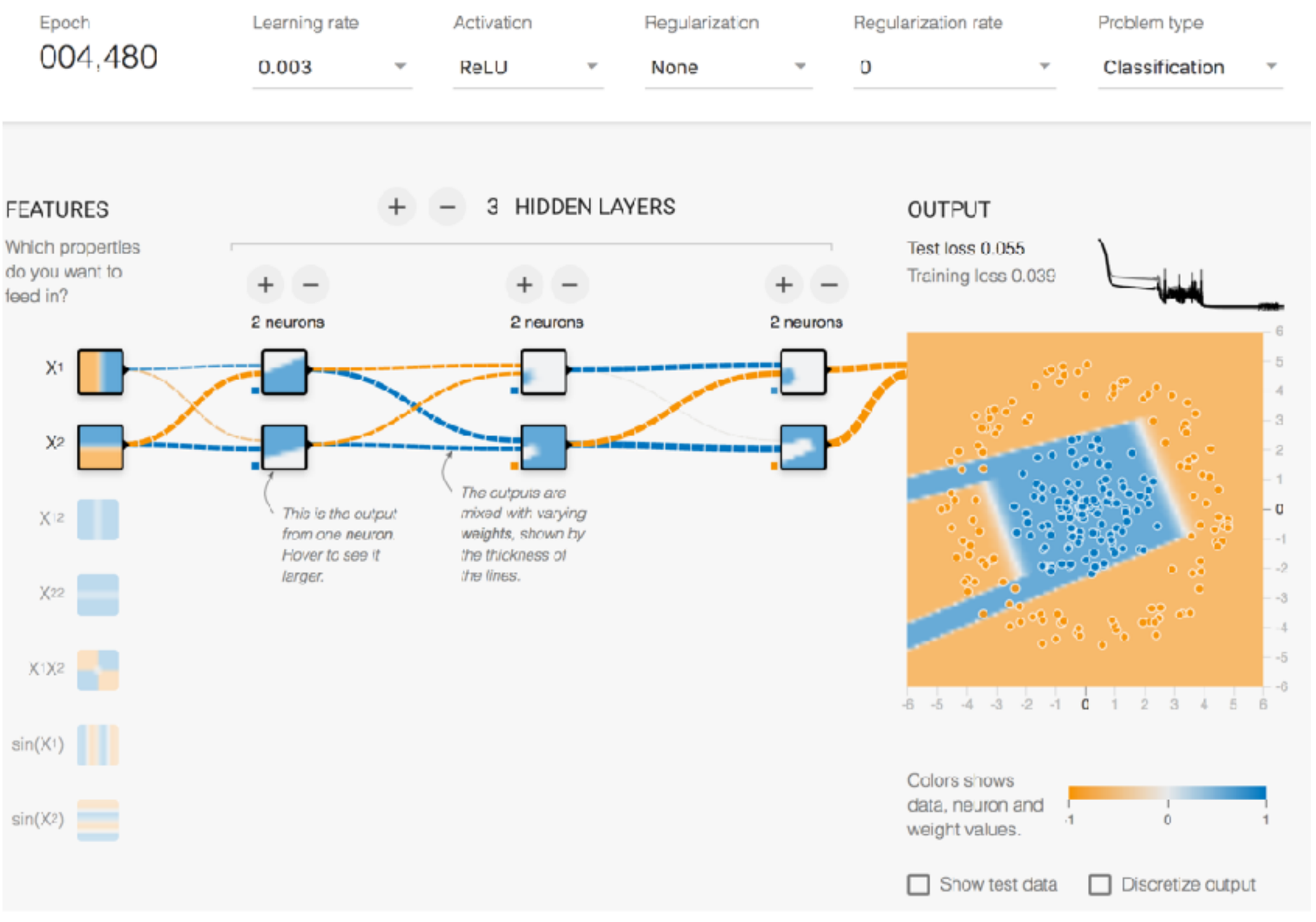
question


Epoch
000,000
Learning rate
0.3
Activation
ReLU
Regularization
None
Regularization rate
0
Problem type
Classification
FEATURES
Which properties do you want to feed in?
2 HIDDEN LAYERS
2 neurons
2 neurons
X1
X2
X1²
X2²
X1X2
sin(X1)
sin(X2)
This is the output from one neuron. Hover to see it larger.
The outputs are mixed with varying weights, shown by the thickness of the lines.
OUTPUT
Test loss 0.521
Training loss 0.513
Colors shows data, neuron and weight values.
Show test data
Discretize output

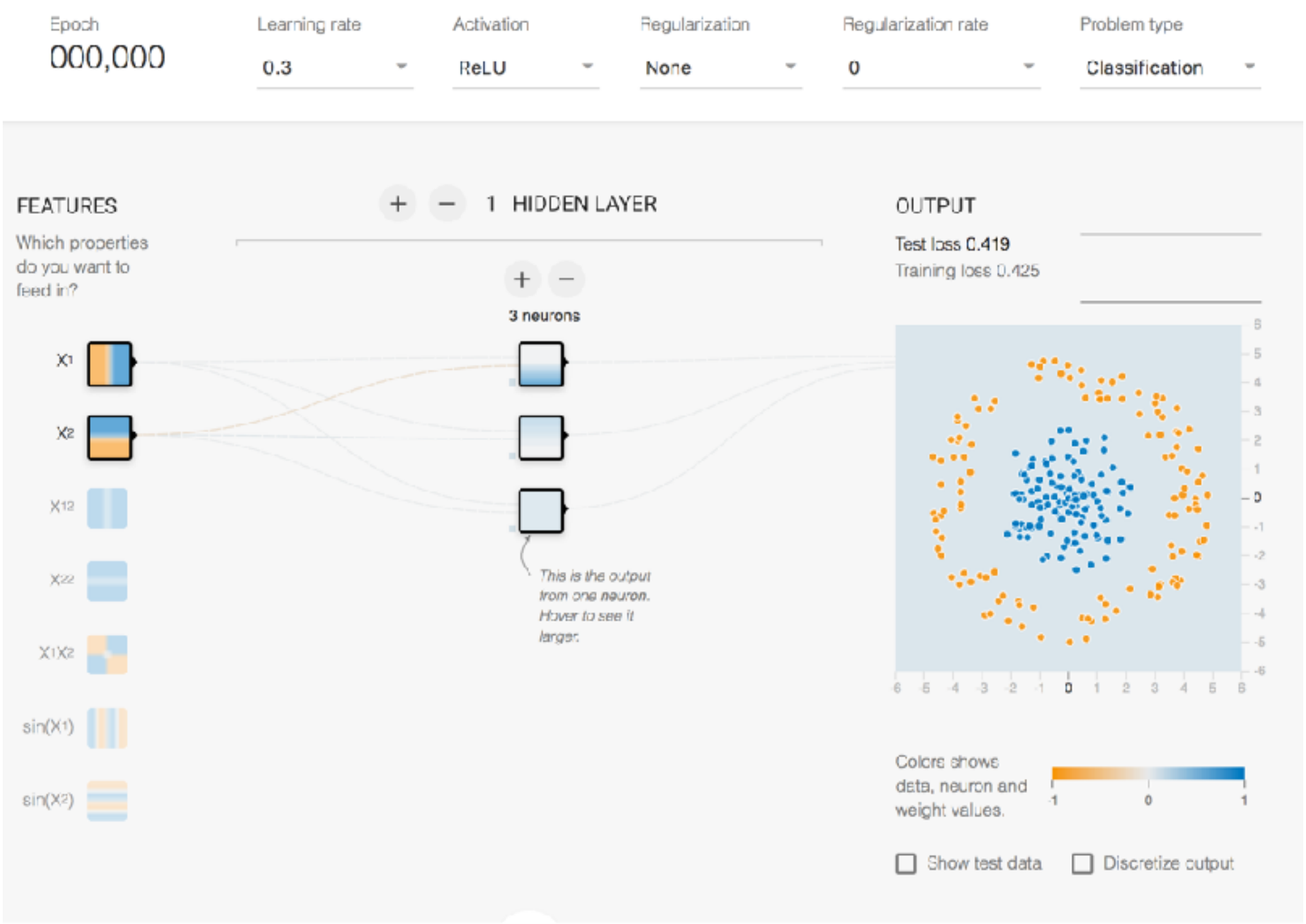
answer


Epoch
005,991
Learning rate
0.003
Activation
ReLU
Regularization
None
Regularization rate
0
Problem type
Classification
FEATURES
Which properties do you want to feed in?
2 HIDDEN LAYERS
2 neurons
2 neurons
X1
X2
X1²
X2²
X1X2
sin(X1)
sin(X2)
This is the output from one neuron. Hover to see it larger.
The outputs are mixed with varying weights, shown by the thickness of the lines.
OUTPUT
Test loss 0.109
Training loss 0.042
Colors shows data, neuron and weight values.
Show test data
Discretize output

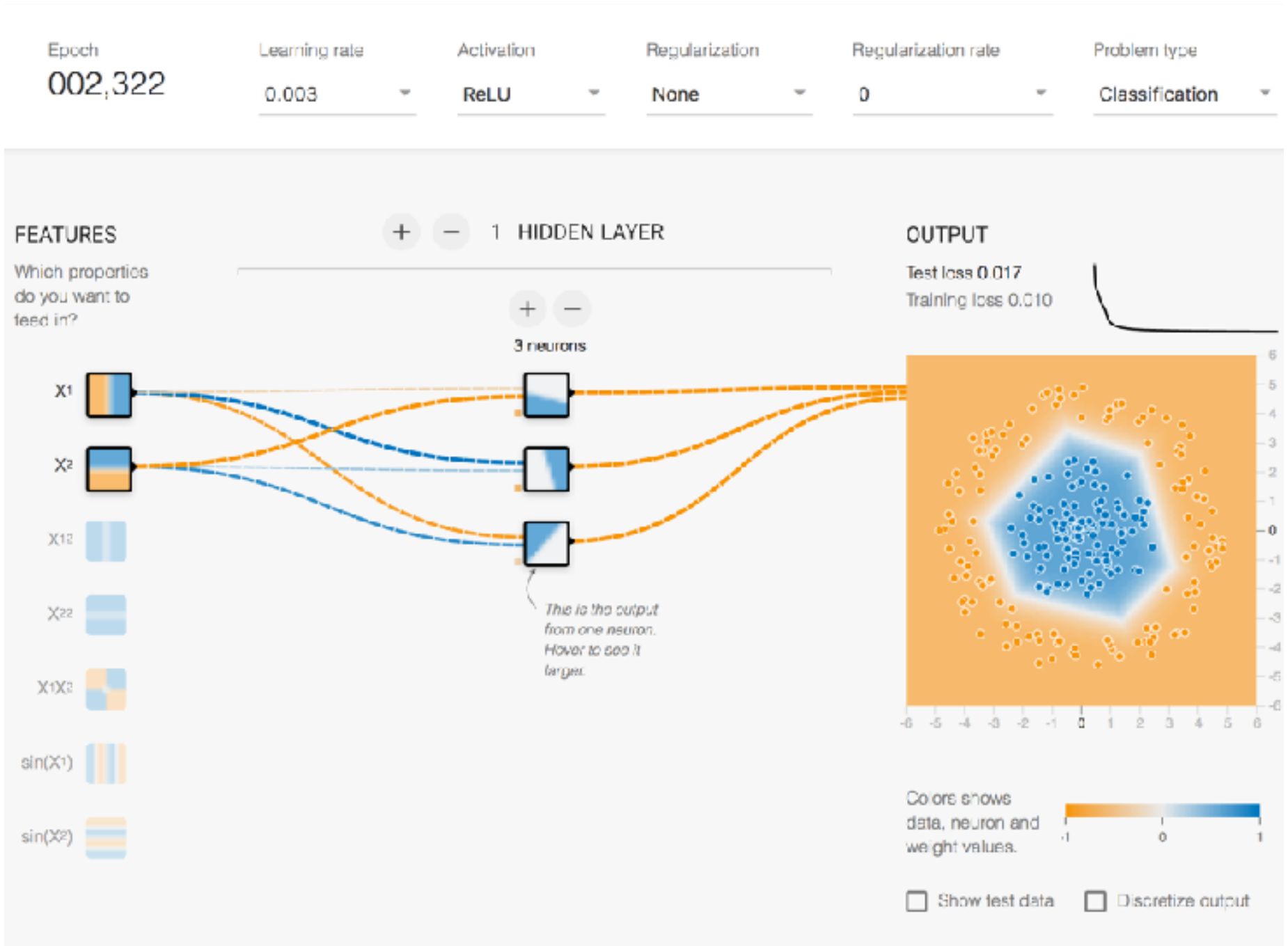
answer


Epoch
004,480
Learning rate
0.003
Activation
ReLU
Regularization
None
Regularization rate
0
Problem type
Classification
FEATURES
Which properties do you want to feed in?
3 HIDDEN LAYERS
2 neurons
2 neurons
2 neurons
X1
X2
X12
X22
X1X2
sin(X1)
sin(X2)
This is the output from one neuron. Hover to see it larger.
The outputs are mixed with varying weights, shown by the thickness of the lines.
OUTPUT
Test loss 0.055
Training loss 0.039
Colors shows data, neuron and weight values.
-1
0
1
Show test data
Discretize output

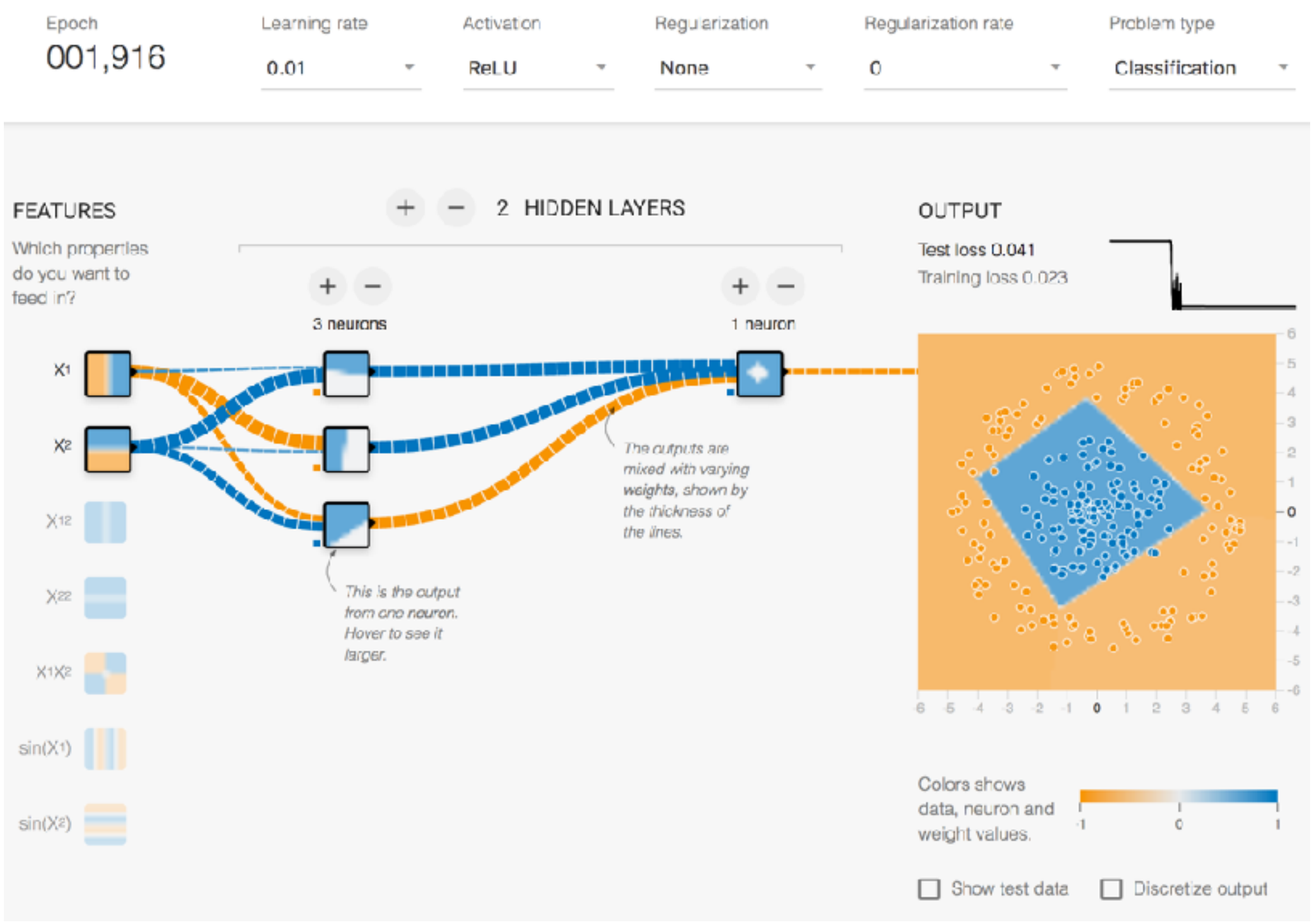
question


Epoch
000,000
Learning rate
0.3
Activation
ReLU
Regularization
None
Regularization rate
0
Problem type
Classification
FEATURES
Which properties do you want to feed in?
1 HIDDEN LAYER
3 neurons
X1
X2
X12
X22
X1X2
sin(X1)
sin(X2)
This is the output from one neuron. Hover to see it larger.
OUTPUT
Test loss 0.419
Training loss 0.425
Colors shows data, neuron and weight values.
-1
0
1
Show test data
Discretize output

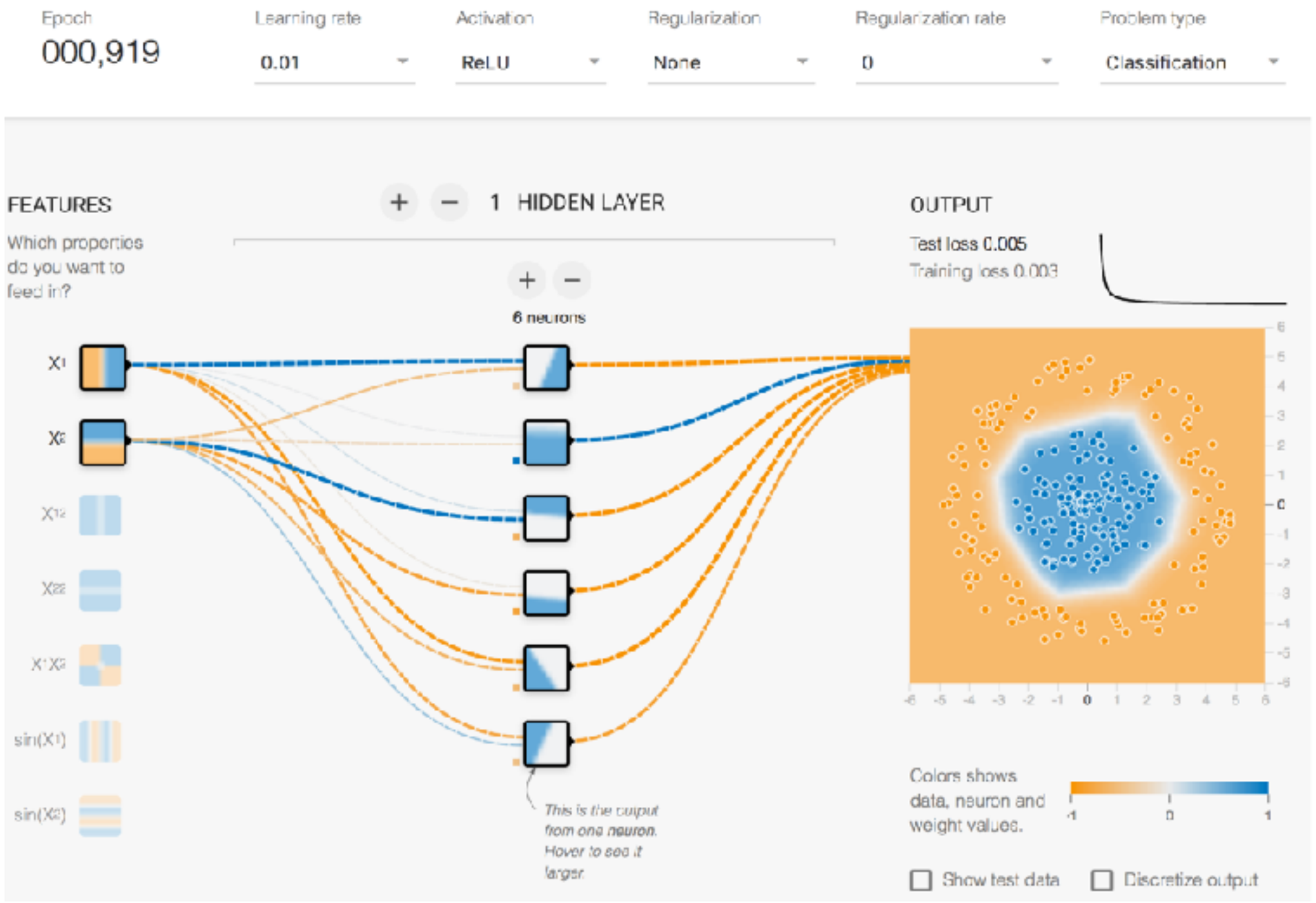
answer


Epoch
002,322
Learning rate
0.003
Activation
ReLU
Regularization
None
Regularization rate
0
Problem type
Classification
FEATURES
Which properties do you want to feed in?
1 HIDDEN LAYER
3 neurons
X1
X2
X12
X22
X1X2
sin(X1)
sin(X2)
This is the output from one neuron. Hover to see it larger.
OUTPUT
Test loss 0.017
Training loss 0.010
Colors shows data, neuron and weight values.
Show test data
Discretize output

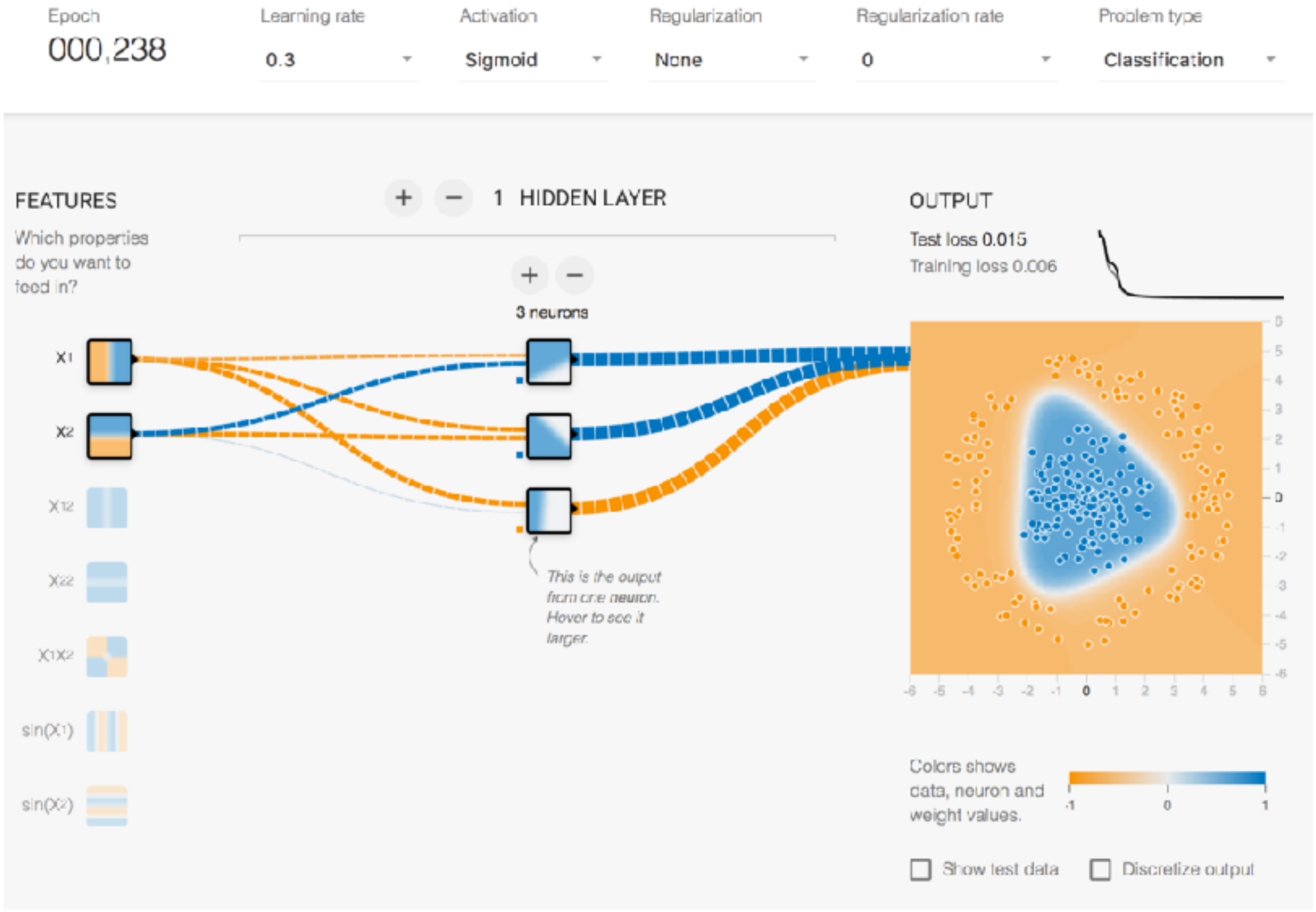
answer


Epoch
001,916
Learning rate
0.01
Activation
ReLU
Regularization
None
Regularization rate
0
Problem type
Classification
FEATURES
Which properties do you want to feed in?
2 HIDDEN LAYERS
3 neurons
1 neuron
X1
X2
X12
X22
X1X2
sin(X1)
sin(X2)
The outputs are mixed with varying weights, shown by the thickness of the lines.
This is the output from one neuron. Hover to see it larger.
OUTPUT
Test loss 0.041
Training loss 0.023
Colors shows data, neuron and weight values.
Show test data
Discretize output

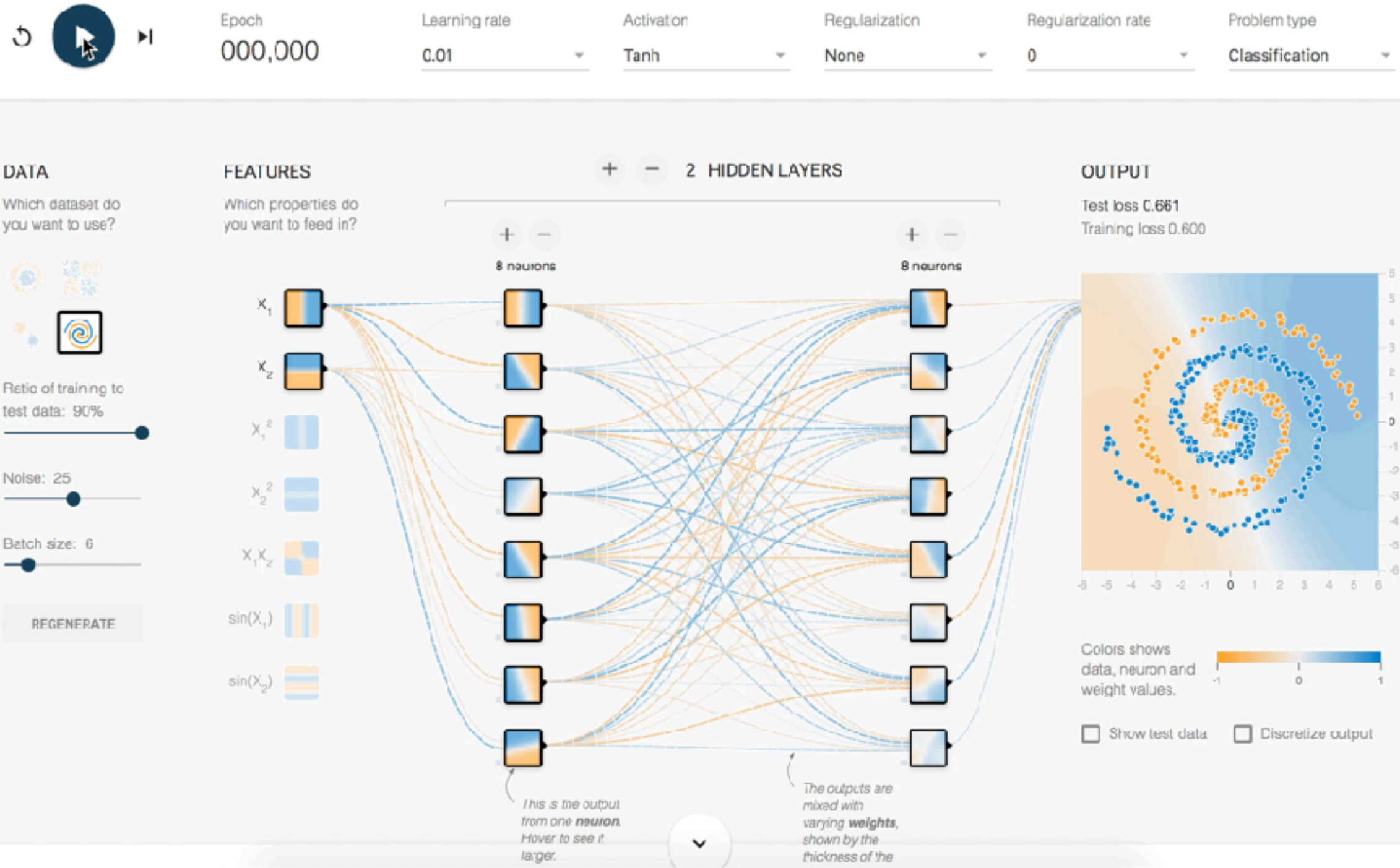
answer


Epoch
000,919
Learning rate
0.01
Activation
ReLU
Regularization
None
Regularization rate
0
Problem type
Classification
FEATURES
Which properties do you want to feed in?
1 HIDDEN LAYER
6 neurons
X1
X2
X1²
X2²
X1X2
sin(X1)
sin(X2)
This is the output from one neuron. Hover to see it larger.
OUTPUT
Test loss 0.005
Training loss 0.003
Colors shows data, neuron and weight values.
Show test data
Discretize output

answer - with sigmoid



question


Epoch
000,000
Learning rate
0.01
Activation
Tanh
Regularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 90%
Noise: 25
Batch size: 6
REGENERATE
FEATURES
Which properties do you want to feed in?
X1
X2
X1²
X2²
X1X2
sin(X1)
sin(X2)
2 HIDDEN LAYERS
8 neurons
8 neurons
This is the output from one neuron. Hover to see it larger.
The outputs are mixed with varying weights, shown by the thickness of the
OUTPUT
Test loss 0.661
Training loss 0.600
Colors shows data, neuron and weight values.
Show test data
Discretize output

Administrative Issues

Communications matters

Use **whatsapp** for communications. This is the fastest way to get response.

For private matters you can privately WhatsApp me.

Get course materials from:
https://web.bii.a-star.edu.sg/~leehk/index.html

I need one class monitor, Tianyi

How do we want to do lecture? 2 hours on Fridays and 1 hr office 12-1pm on Wednesdays

General grading and expectations

F/E There are those who do not know what they are doing. Their results are unreasonable

D/C There are those who know how to get some good results but cannot explain them.

B There are those who understand what is going on with their experiments. Able to explain their results.

A There are those who know enough to combine different methods and be creative in using Deep Learning

There are those who break the frontiers of Deep Learning research.

Grades

- Class participation: 20%
- Assignments: 40%
- Project: 40%

- No exams. No quiz

Class participation 20%

1% point for each question or comment asked in class

Class monitor will record

Assignments (10% * 4 = 40%)

- For each assignment
  - It has multiple sub-tasks, including coding tasks
  - You need to submit the answers, code and execution results. Everything in git hub, and one copy in NTULearn.
- All assignment report must have:
  - Algorithm description in plain text, supplemented by equations if needed (3-4 marks)
  - Code walk through, part-by-part of the code need to be explained (2-3 marks)
  - Plots and results (1-2 marks)
  - Explanation and interpretation of results (3-4 marks)

Final Project (40%)

- Projects will be
  - Submission deadline: pre-submission around week 7, final submission around week 10, to be confirmed
  - Report deadline: around week 11, to be confirmed
- Assessment is based on the report (20%) and presentation (20%)
  - All students will present their work in class
- One student per project submission. You are allowed discuss project solutions among students.
- No copying of source code, code everything yourself!

Final Project (40%)

Report format: Strictly limited to 6 pages (figures and text font size >10) + 1 page reference, **any longer report will be rejected**

Report template (40%)

- Problem definition
  - Highlights (new algorithms, insights from the experiments)
  - Dataset pre-processing description
  - Training and testing procedure
  - Experimental study
  - (clarity, model understanding, and highlights are important for the assessment)
- Presentation -> score (10%)

Start your project early

Do not miss the deadline

30% grade deductions for within 1 week late submissions
60% grade deductions for within 2 weeks late submissions
no grade given for beyond 2 weeks late submissions

To be fair to everyone, request for late submission without grade deductions for emergency cases are to be done before week 9, otherwise request will not be entertained

Questions?

5 minutes break

What we mean by “how do you compute”

It means code from scratch or write out all atomic operations on a document

How does pytorch compute average pooling?
You answer should be, giving a clear example:

Input

x1	x2	x3
x4	x5	x6
x7	x8	x9

output

y1	y2
Y3	Y4

Average pooling with kernel size 2x2, stride 1x1, 1 channel

y1 = (x1+x2+x4+x5)/4
y2 = (x2+x3+x5+x6)/4
y3 = (x4+x5+x7+x8)/4
y4 = (x5+x6+x8+x9)/4

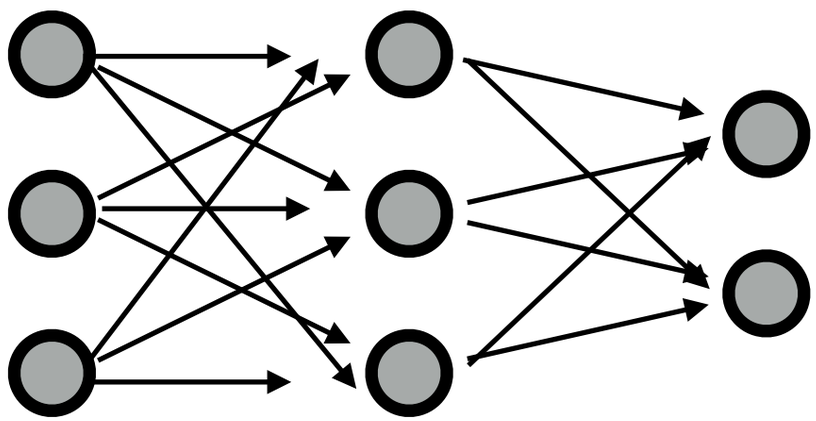
All students create one GitHub project for all assignments and project submission

You also submit one copy in NTULearn for records

Forward Propagation

The network and information flow

feed in data into first layer



final layer presents the output of network

transmitter receiver

transmitter receiver

Notation

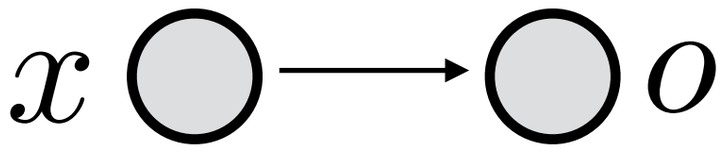
Let $x \in \mathbb{R}^d$ be the input space

Let $y \in \mathbb{R}$ or $y \in \mathbb{N}$ be the label

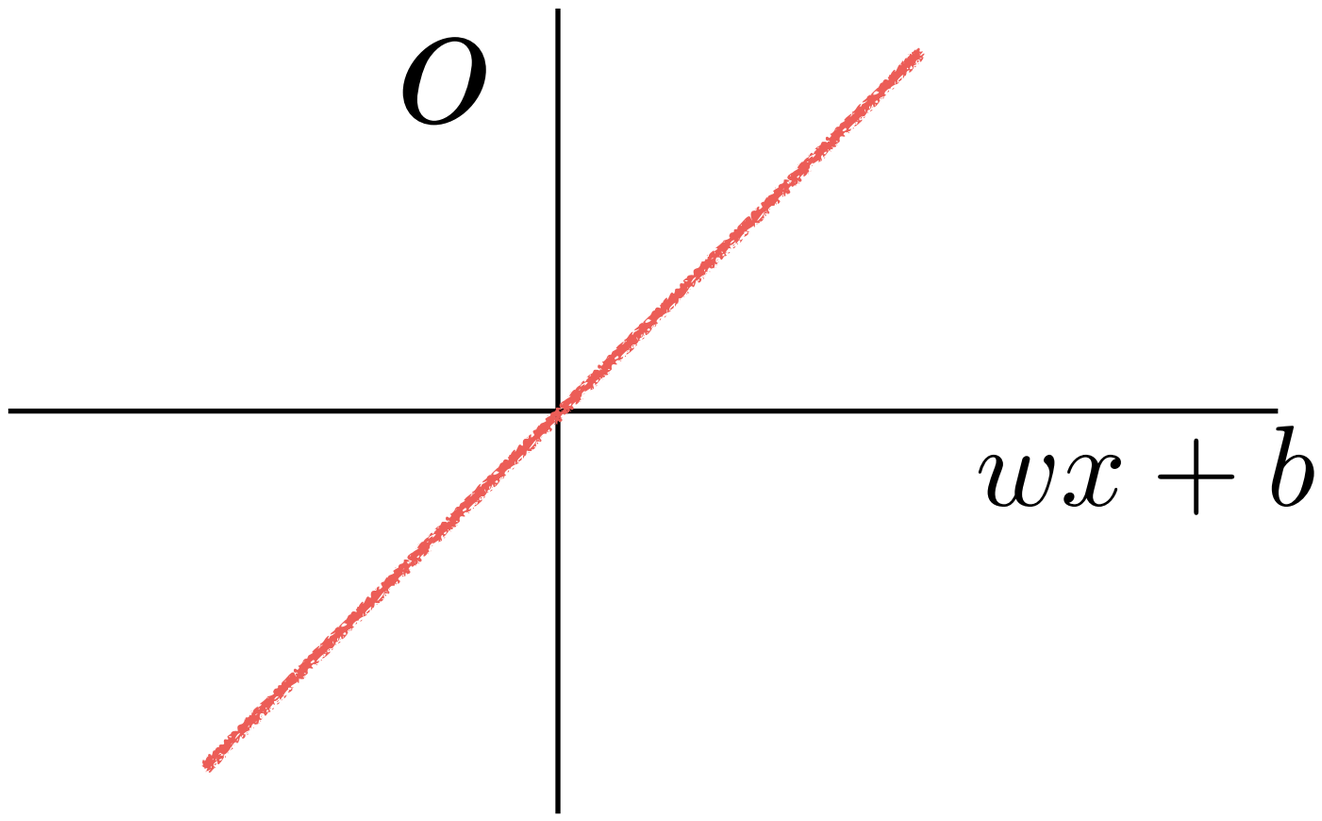
Let $o \in \mathbb{R}$ be the output of the neural network

Simplest perceptron - linear activation function

$x \in \mathbb{R}$

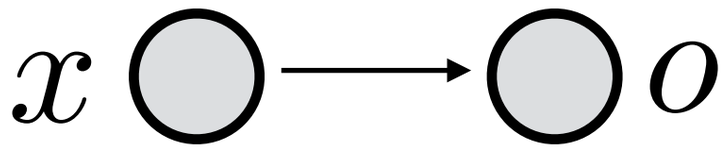


$$o = wx + b$$

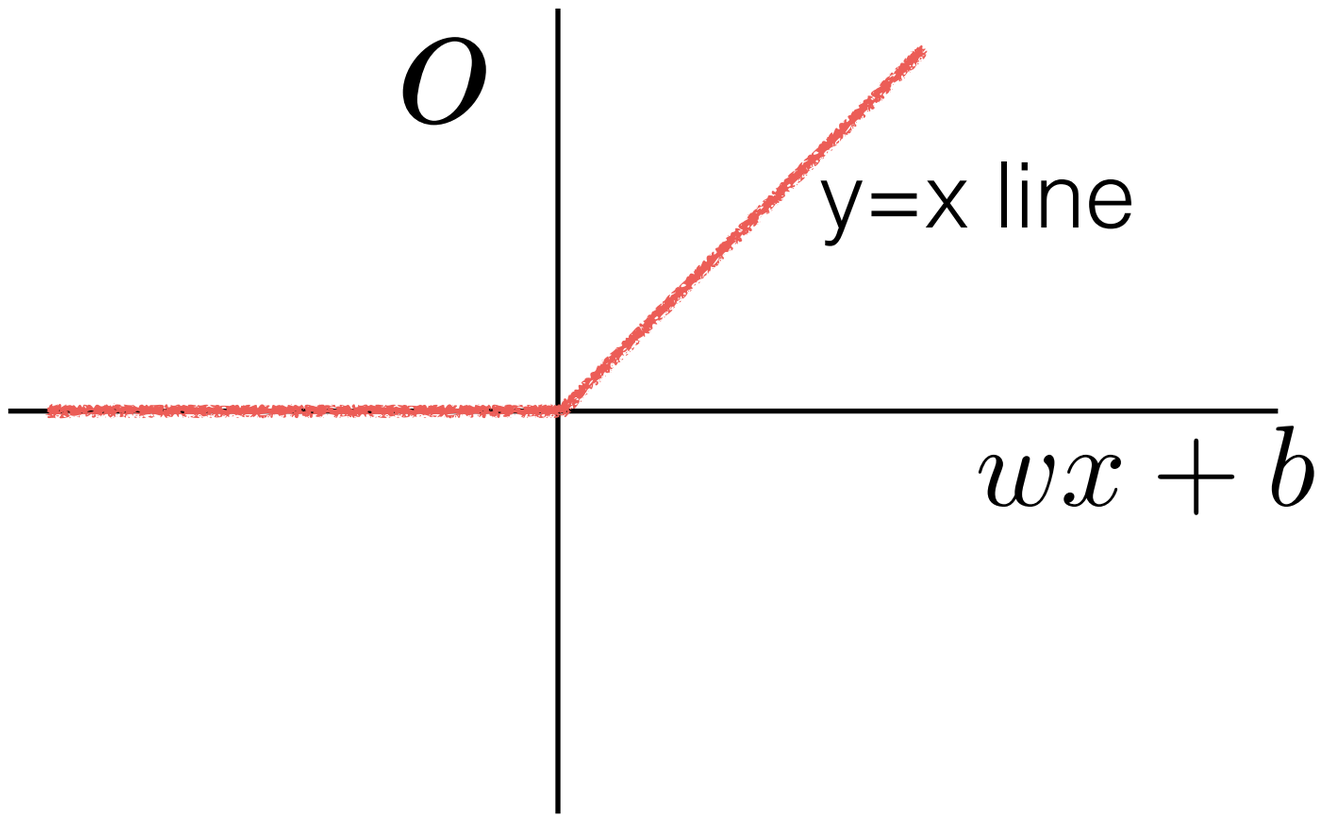


Simplest perceptron - rectilinear activation function

$x \in \mathbb{R}$

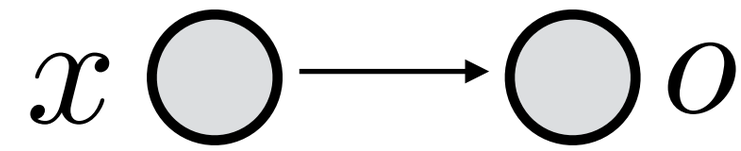


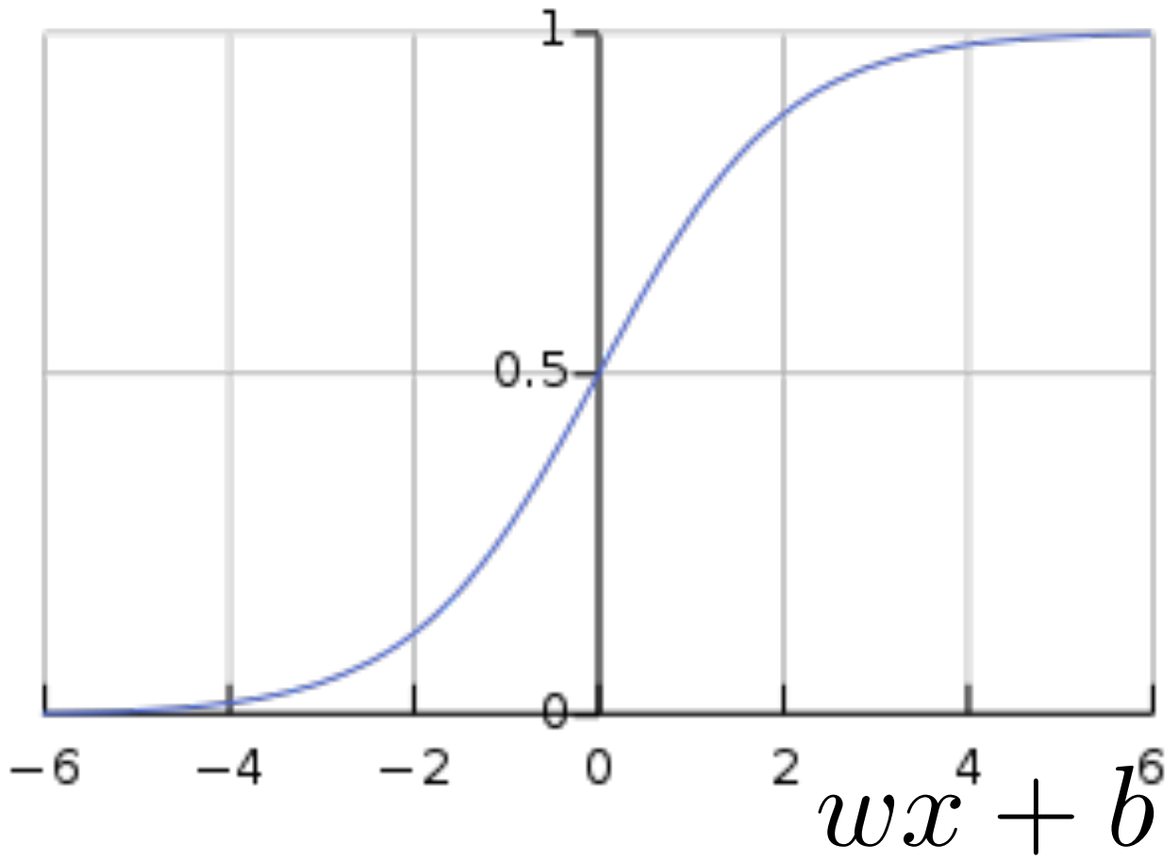
$$o = ReLu(wx + b)$$



Simplest perceptron - sigmoid activation function

$$o = \frac{1}{1 + \exp(-wx - b)}$$

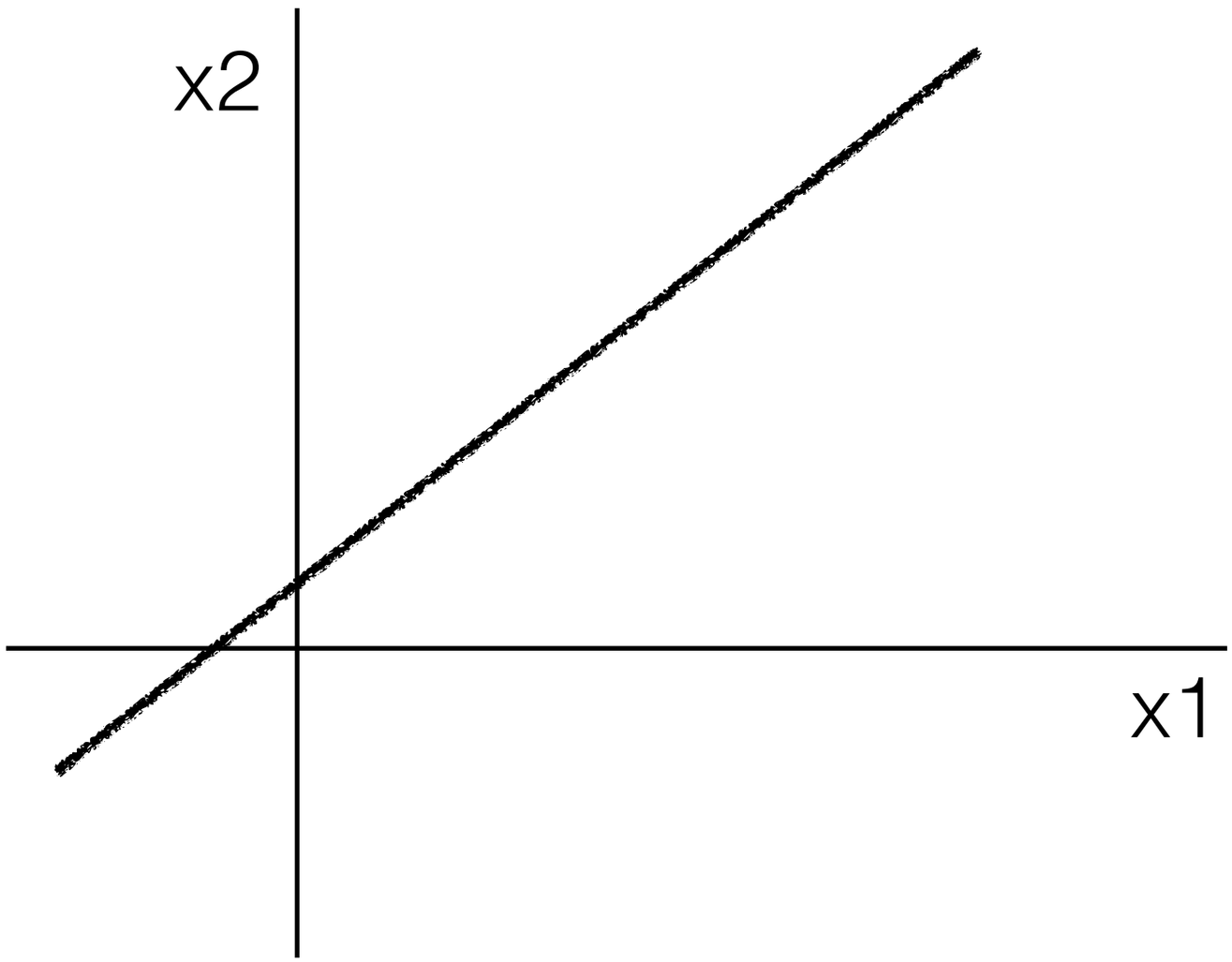




For more activation functions, check out

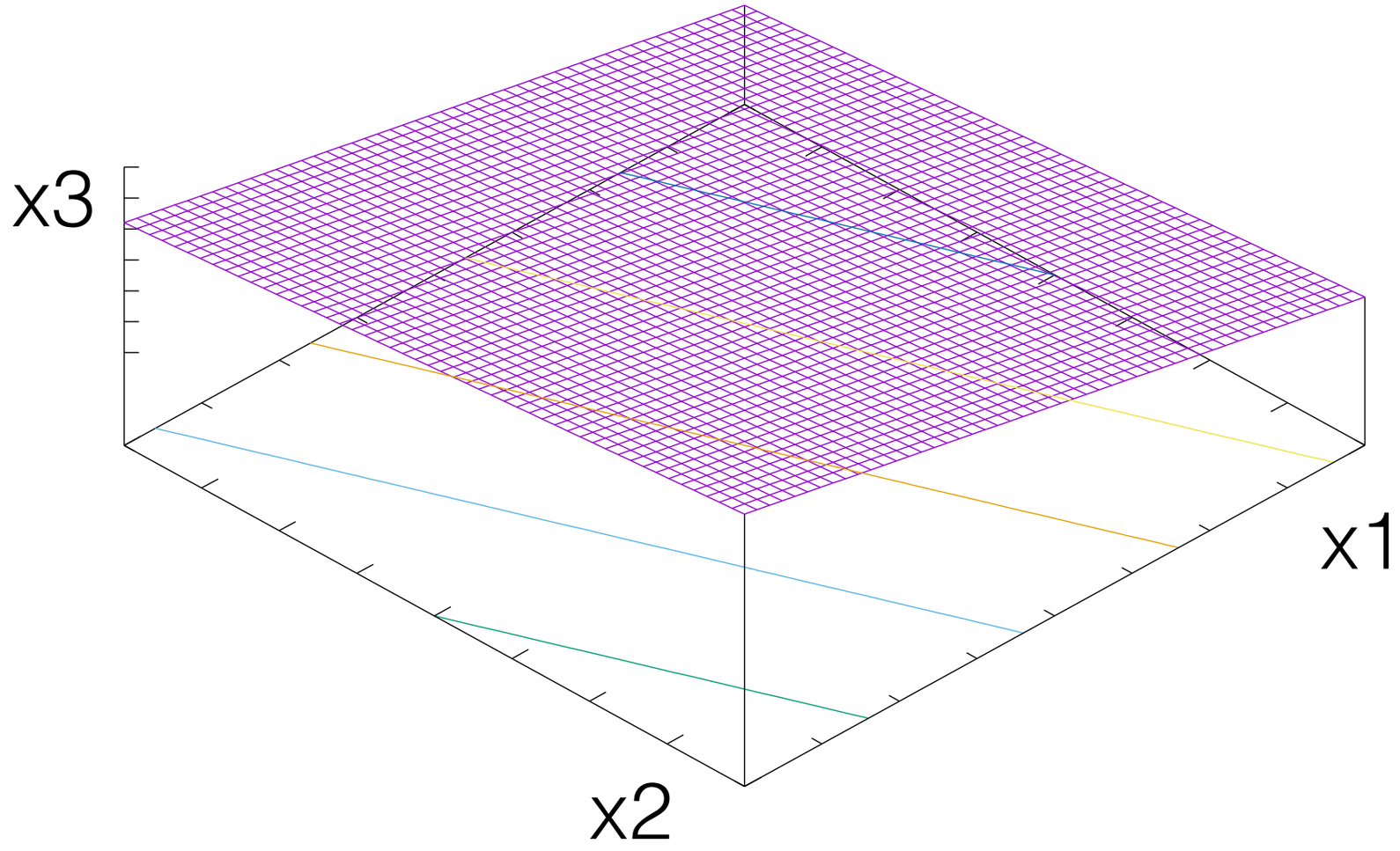
https://en.wikipedia.org/wiki/Activation_function

Equation of straight lines : x1 = m*x2 + c



$$0 = m1*x1 + m2*x2 + c$$

Equation of a plane :



$$0 = m1*x1 + m2*x2 + m3*x3 + c$$

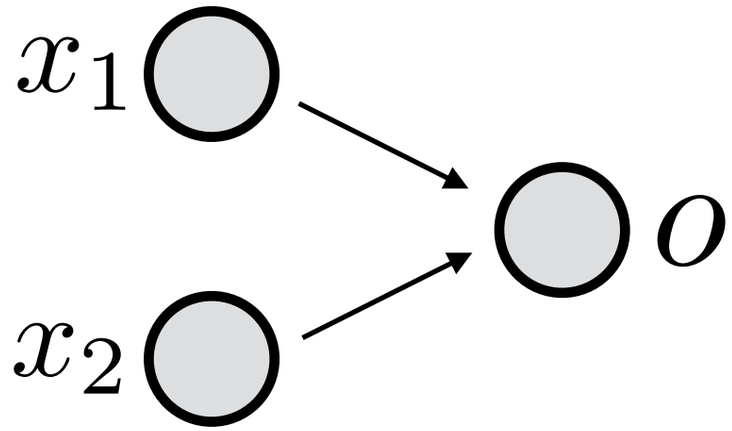
Equation of a hyper-plane :

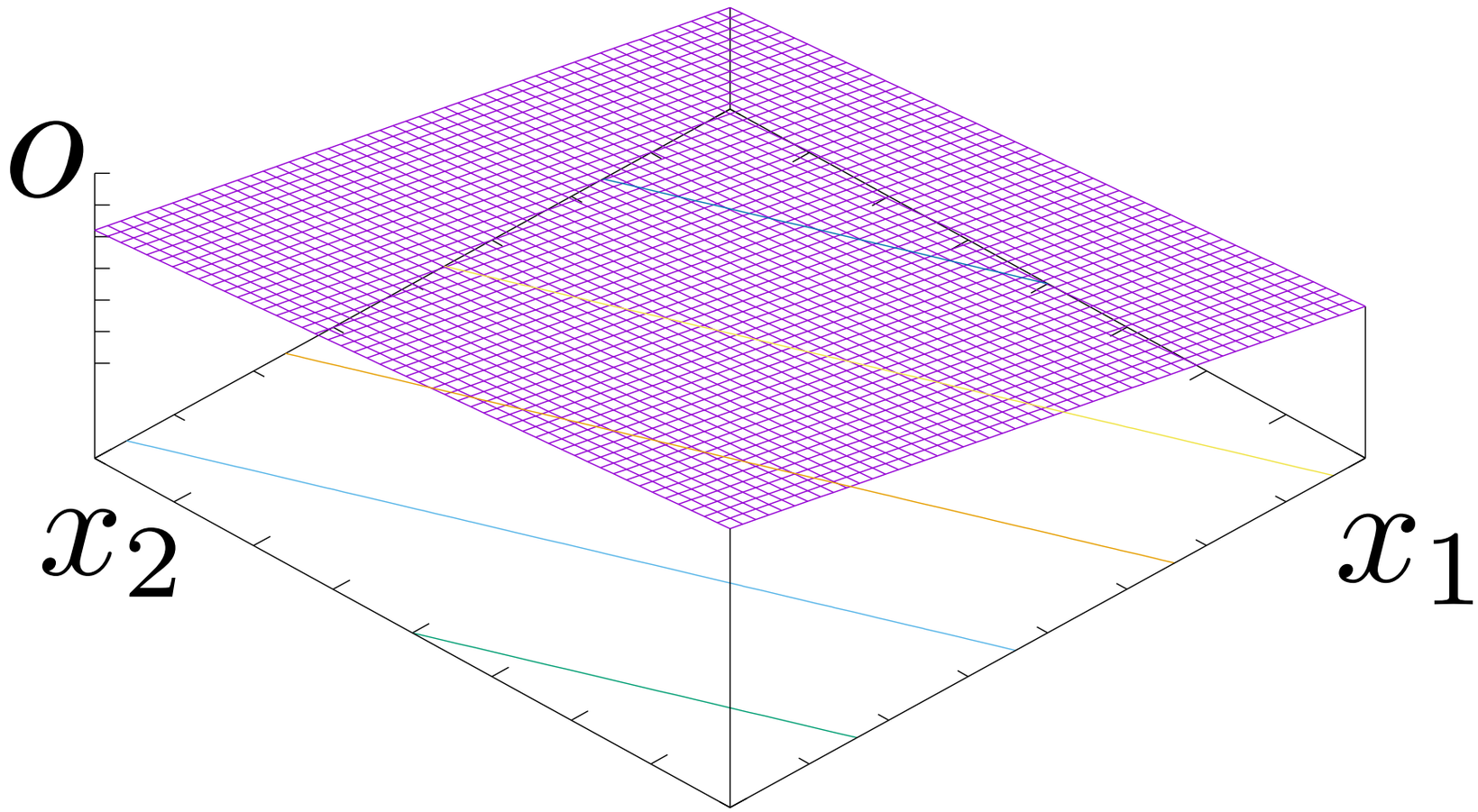
$$0 = m1*x1 + m2*x2 + m3*x3 + \ldots + md * xd + c$$

Next to simplest

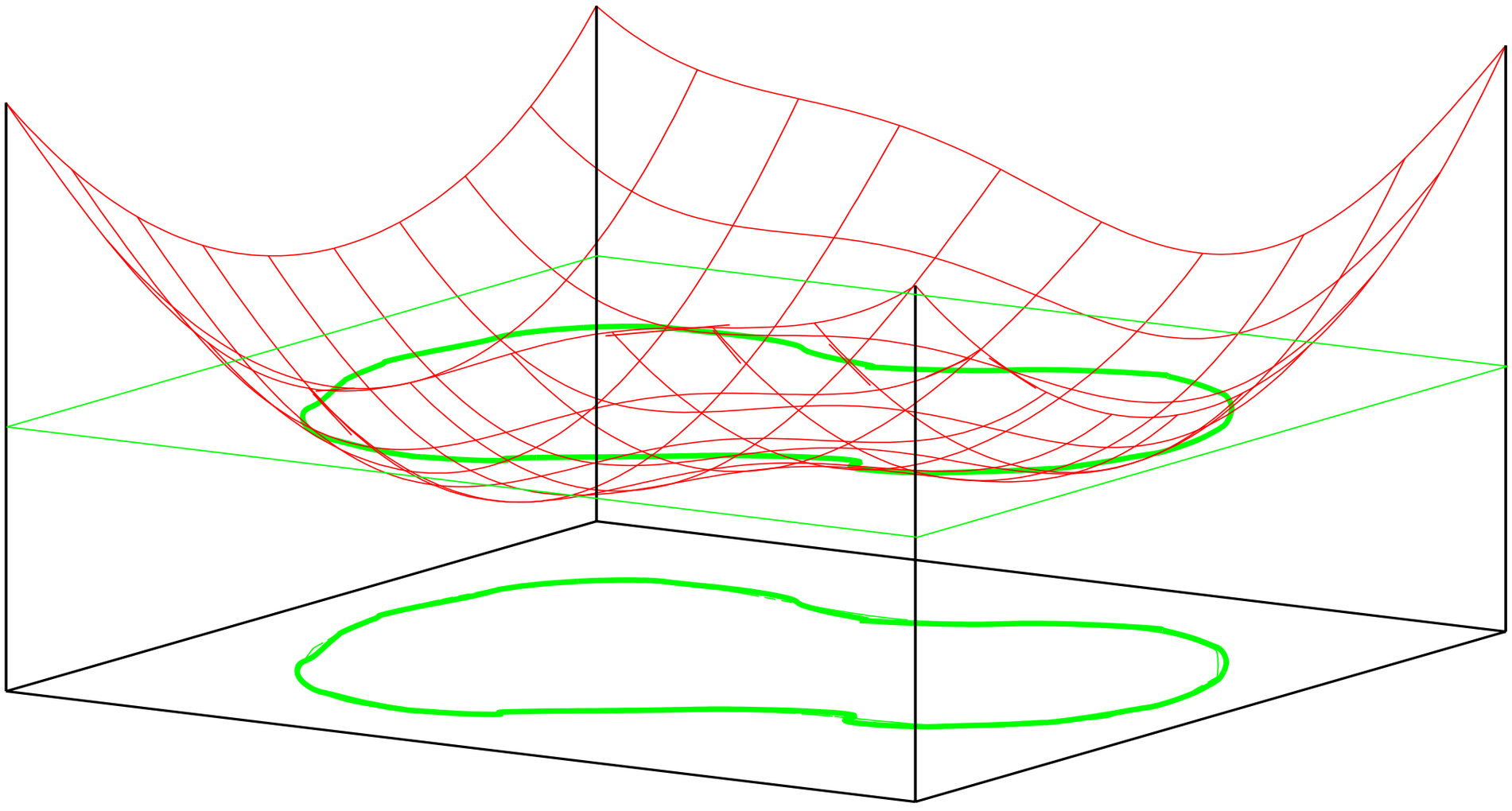
$$x = (x_1, x_2) \in \mathbb{R}^2$$

$$o = w_1 x_1 + w_2 x_2 + b$$





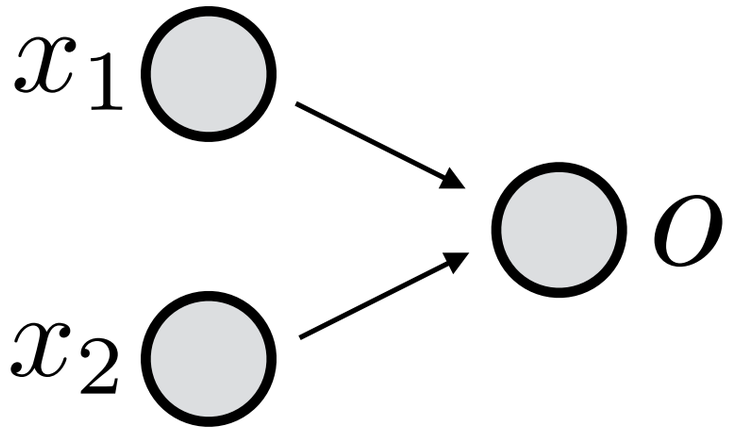
Concept of level sets

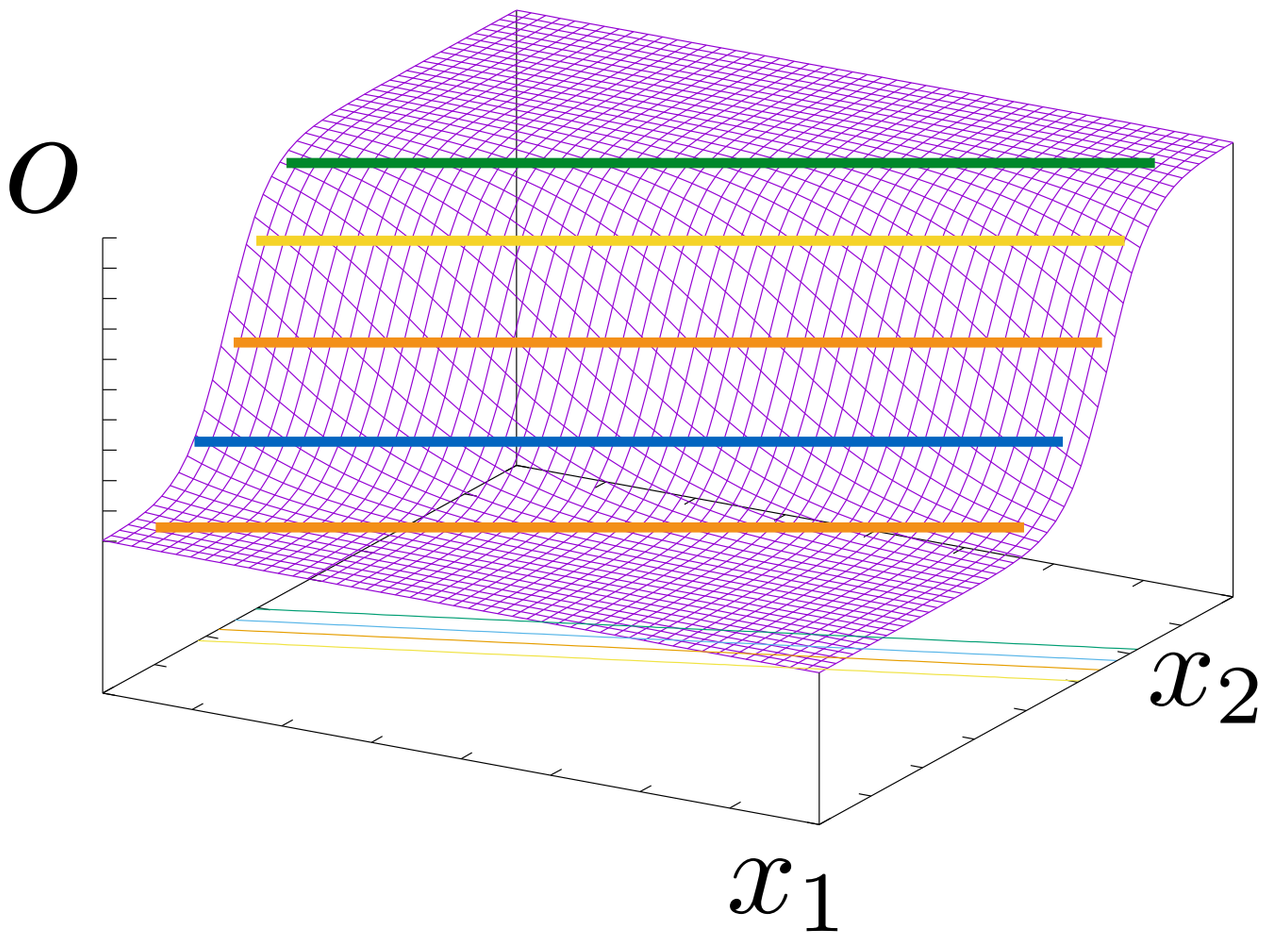


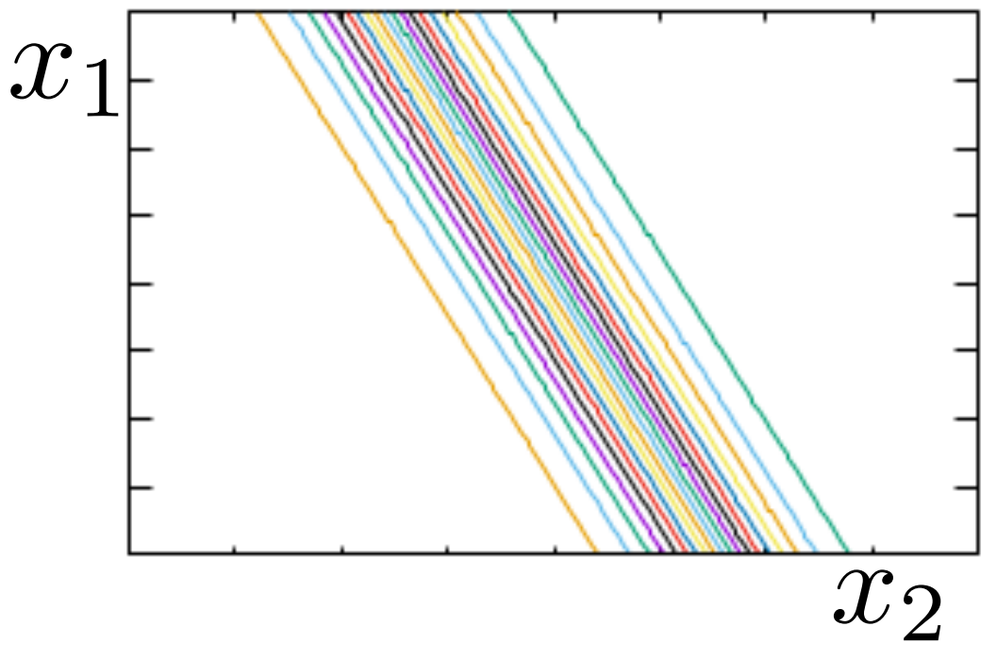
Next to simplest

$$x = (x_1, x_2) \in \mathbb{R}^2$$

$$o = \sigma(w_1 x_1 + w_2 x_2 + b)$$

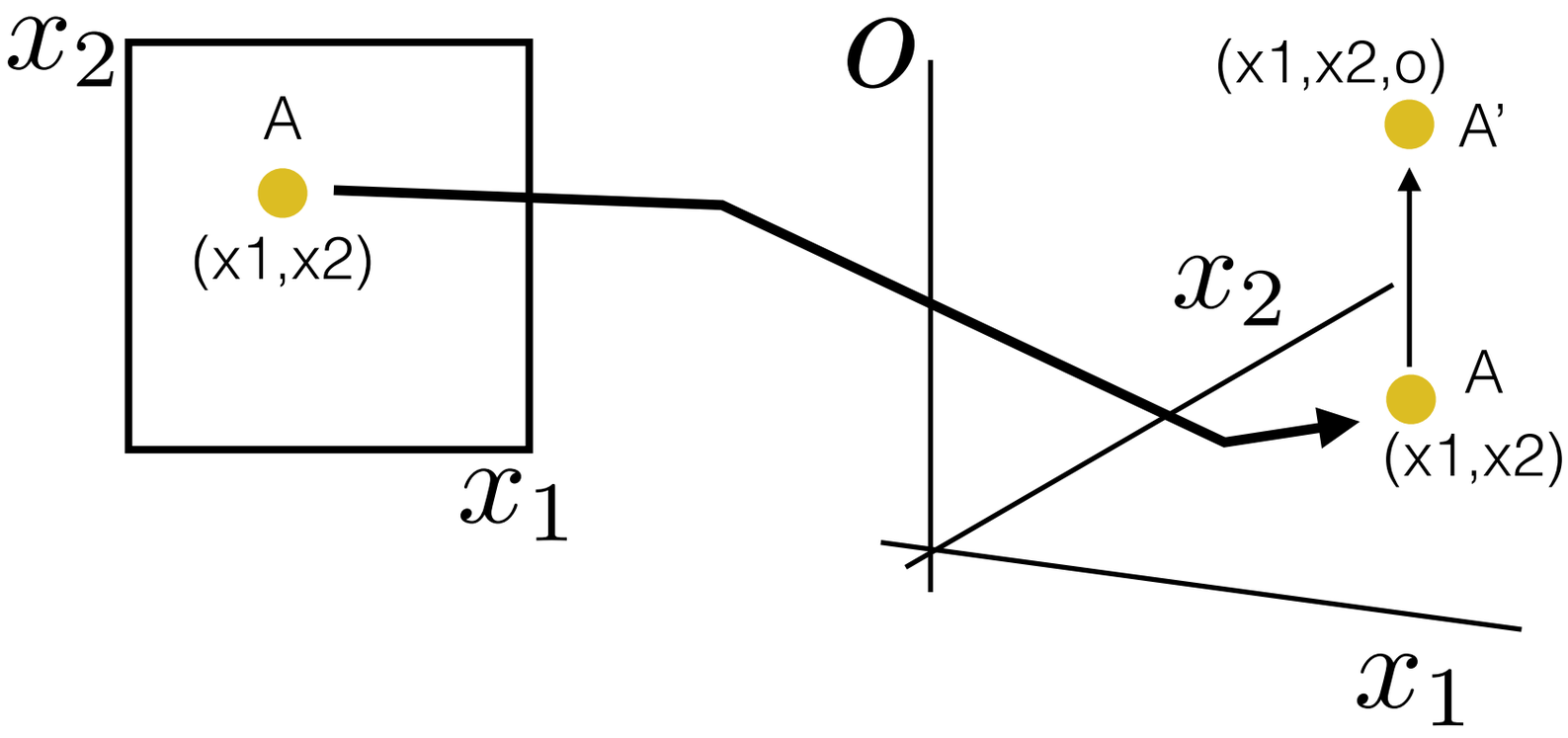


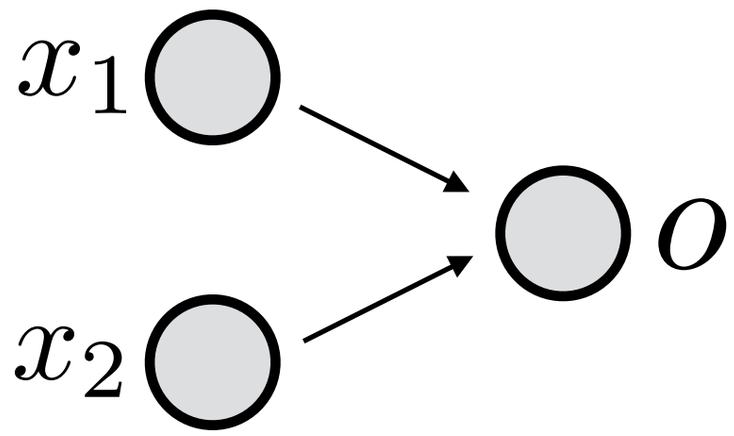




$$x = (x_1, x_2) \in \mathbb{R}^2$$

$$o = \sigma(w_1 x_1 + w_2 x_2 + b)$$

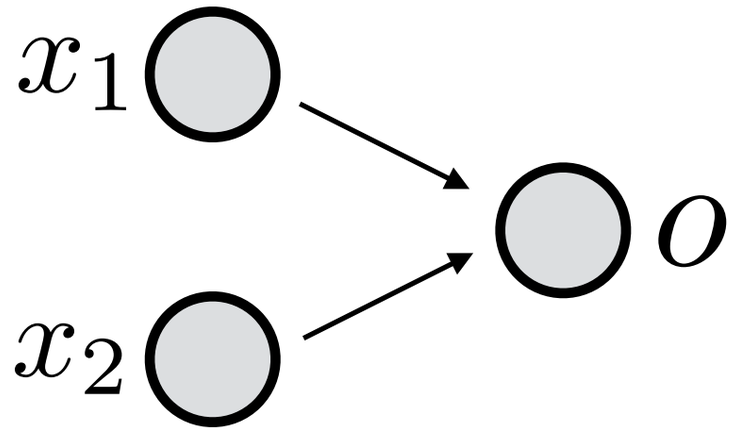


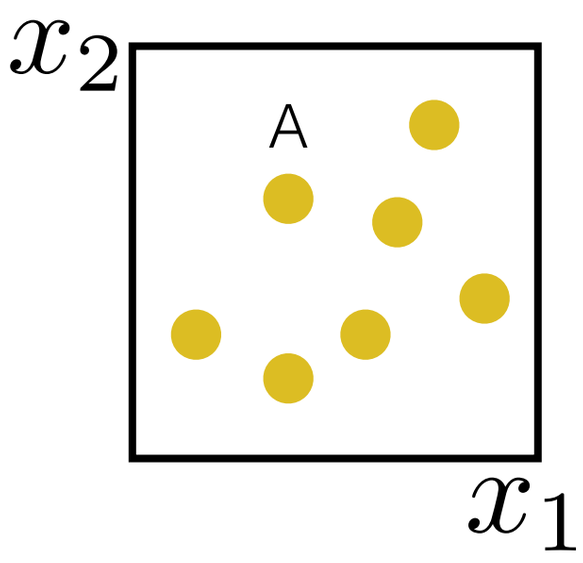


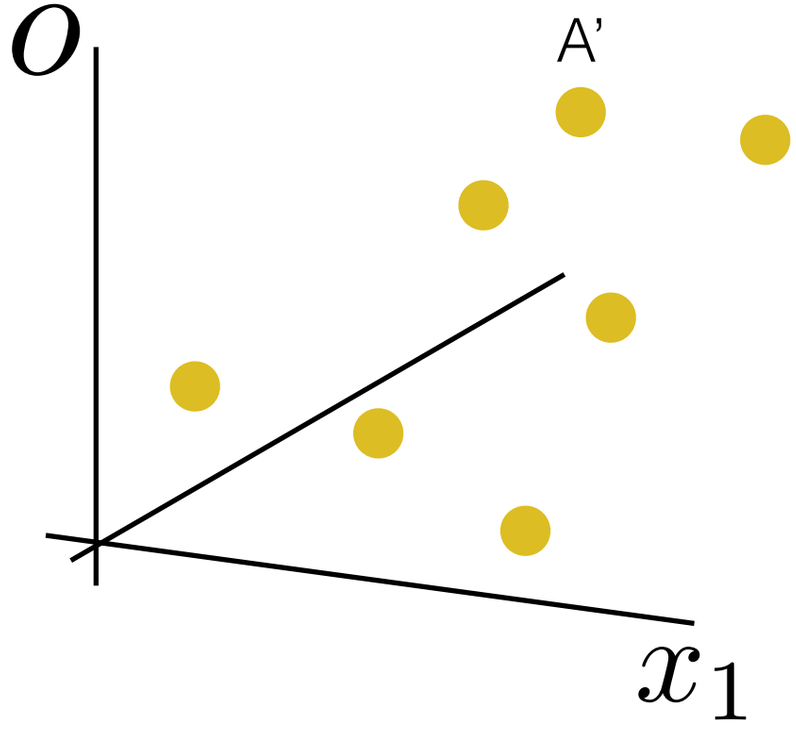
A' = (x1 , x2 , o)

$$x = (x_1, x_2) \in \mathbb{R}^2$$

$$o = \sigma(w_1 x_1 + w_2 x_2 + b)$$

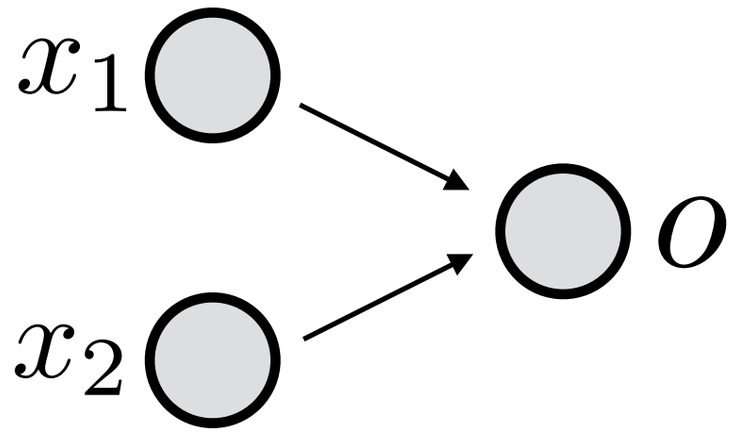


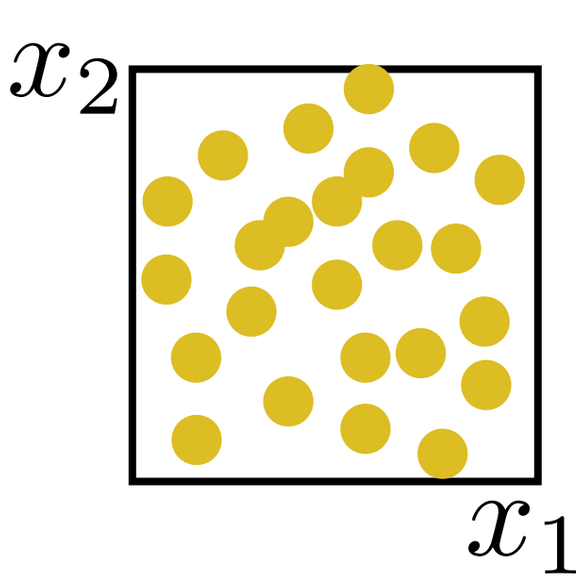


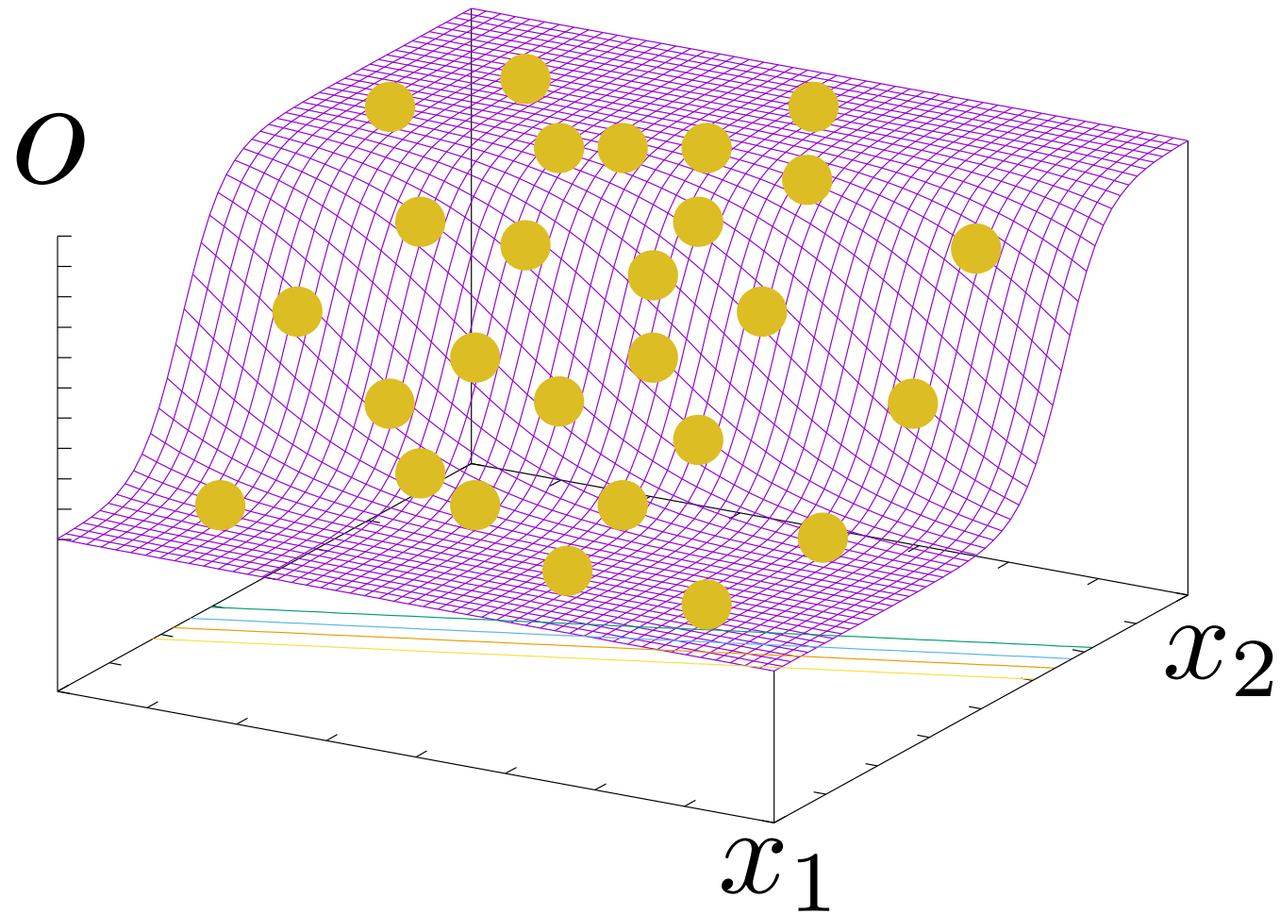


$$x = (x_1, x_2) \in \mathbb{R}^2$$

$$o = \sigma(w_1 x_1 + w_2 x_2 + b)$$



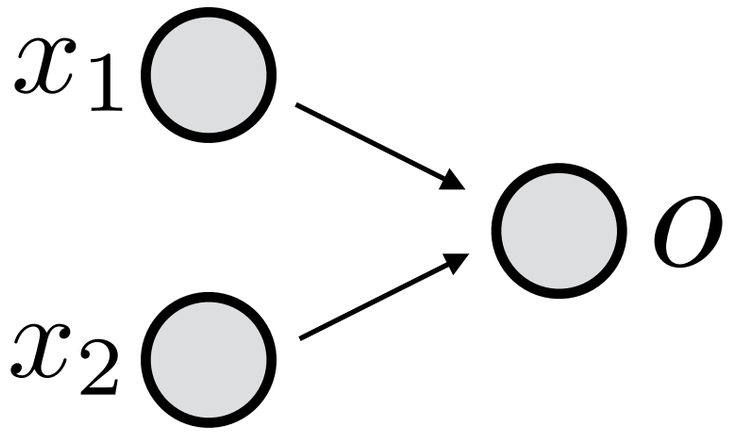


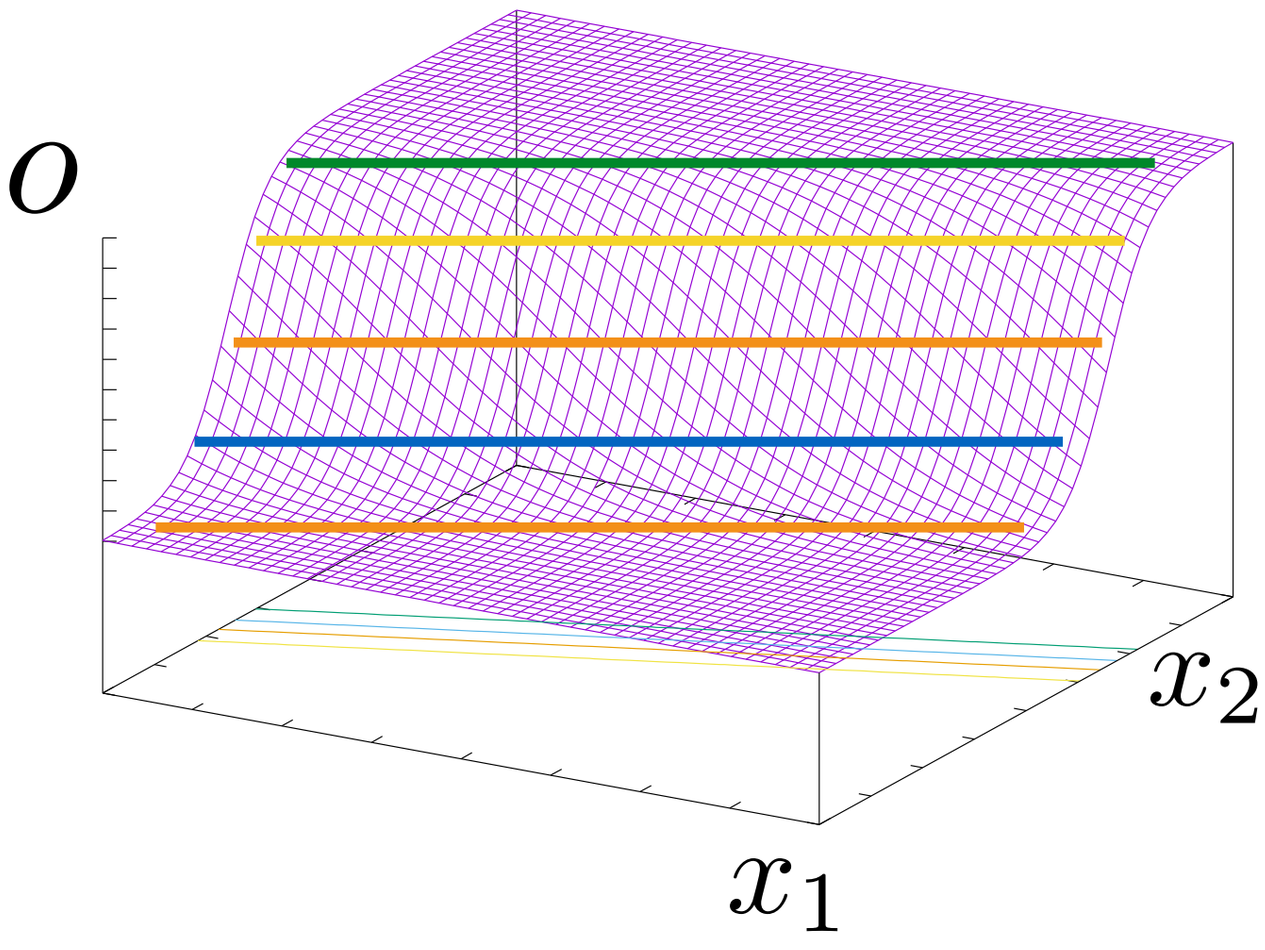


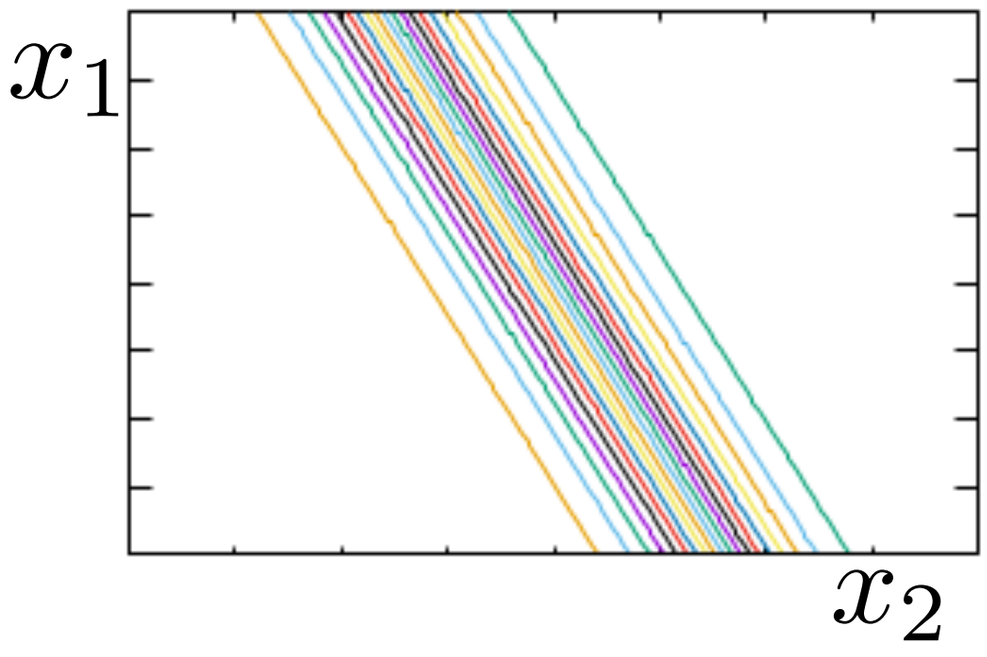
Next to simplest

$$x = (x_1, x_2) \in \mathbb{R}^2$$

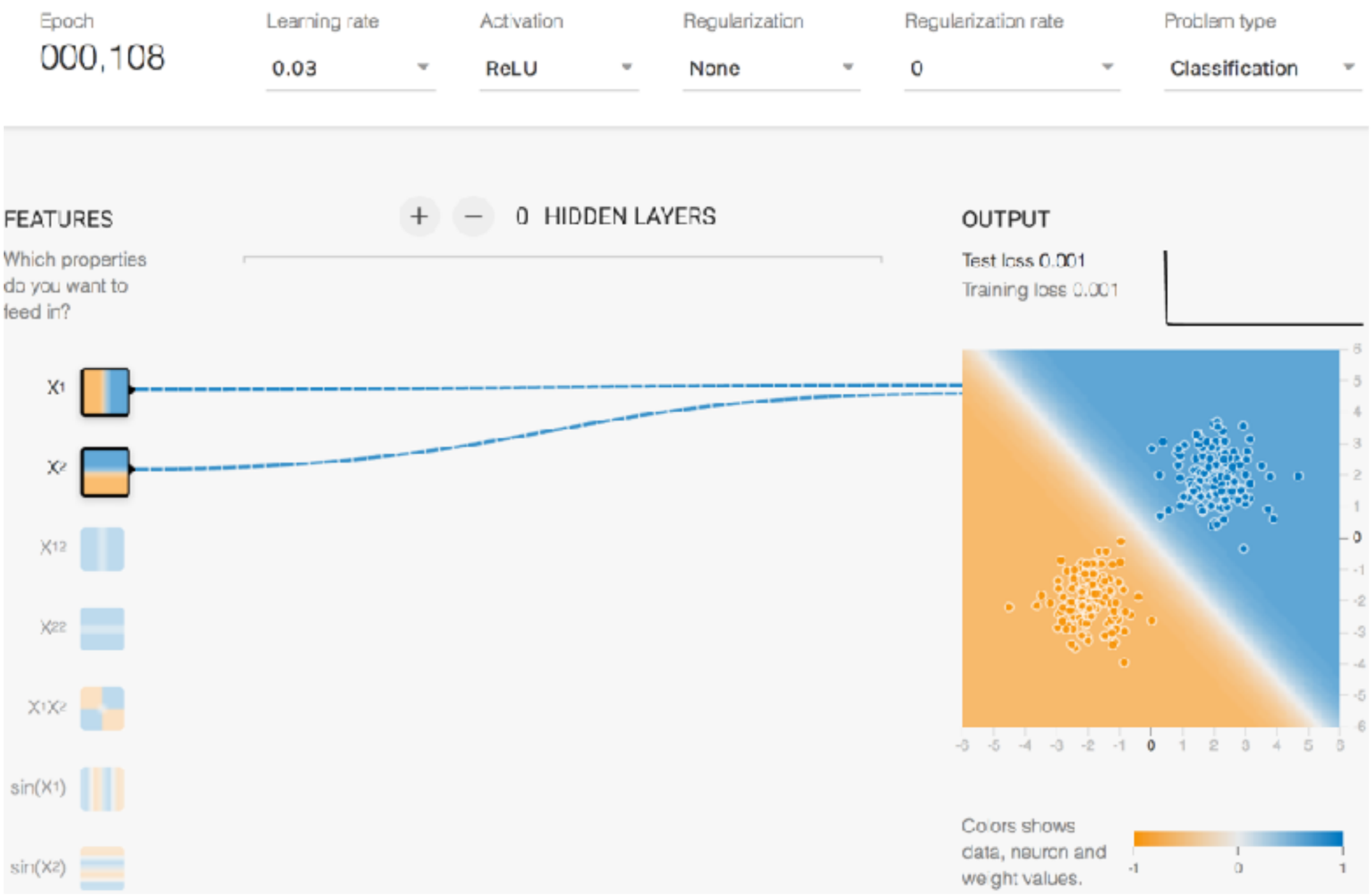
$$o = \sigma(w_1 x_1 + w_2 x_2 + b)$$



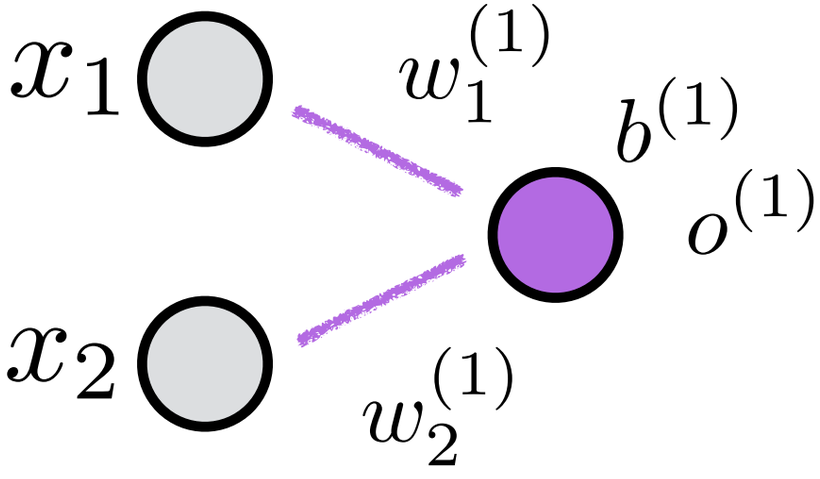




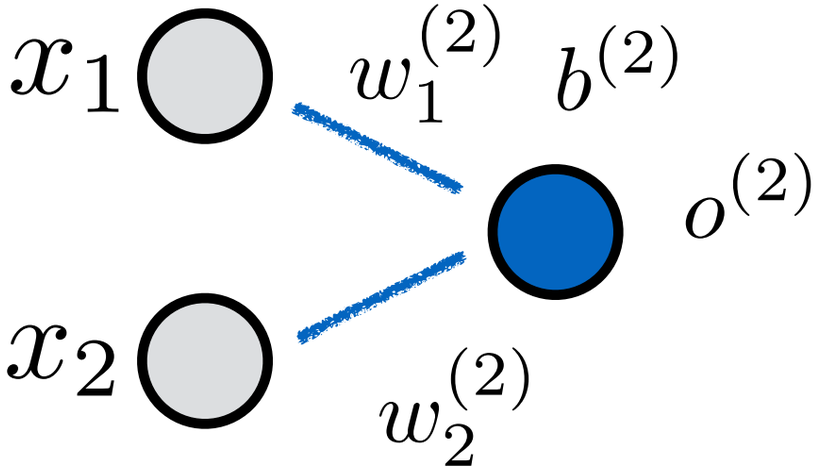
Playground can verify this, please try playground with different activation functions



Stacked up

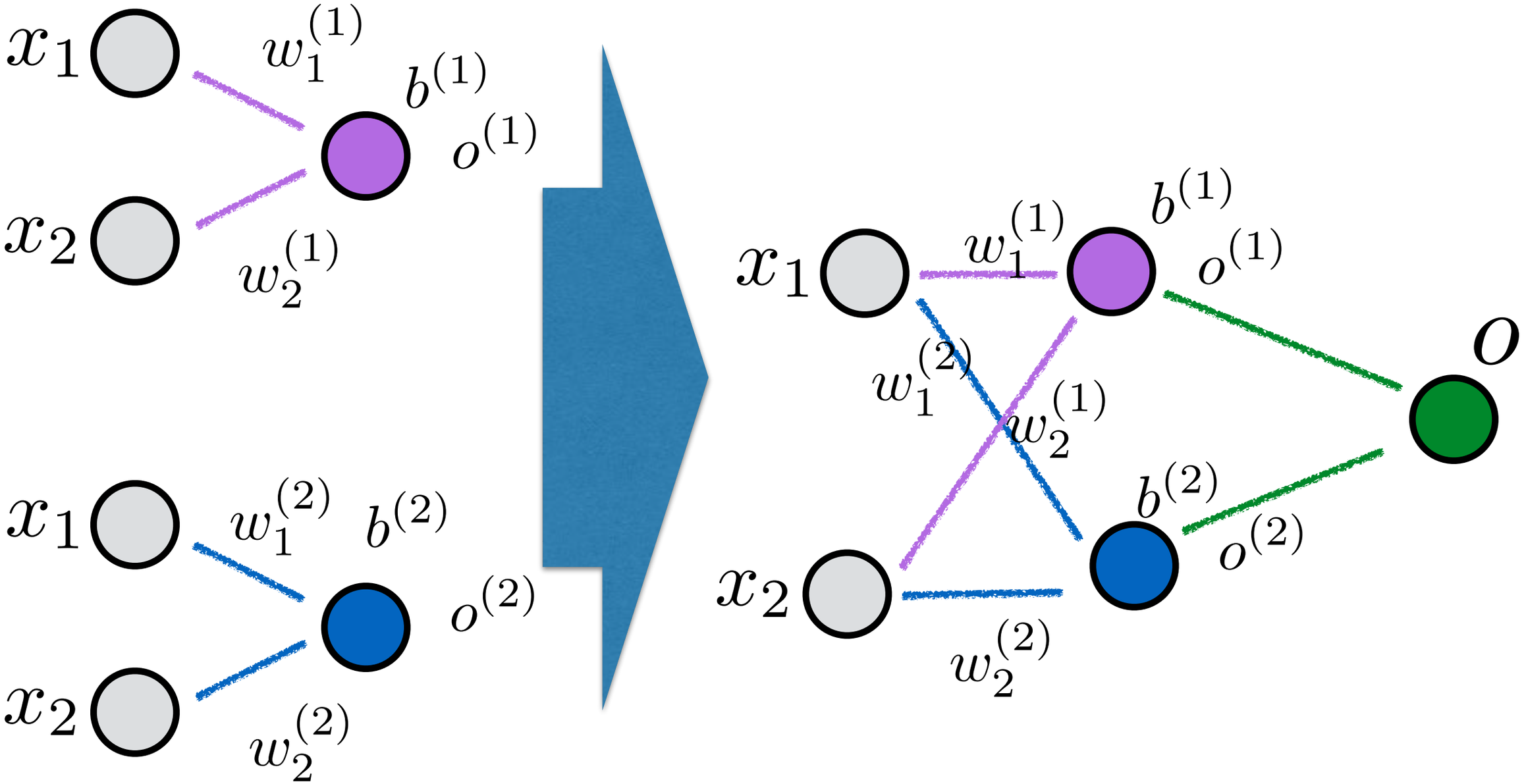


$$o^{(1)} = w_1^{(1)} x_1 + w_2^{(1)} x_2 + b^{(1)}$$

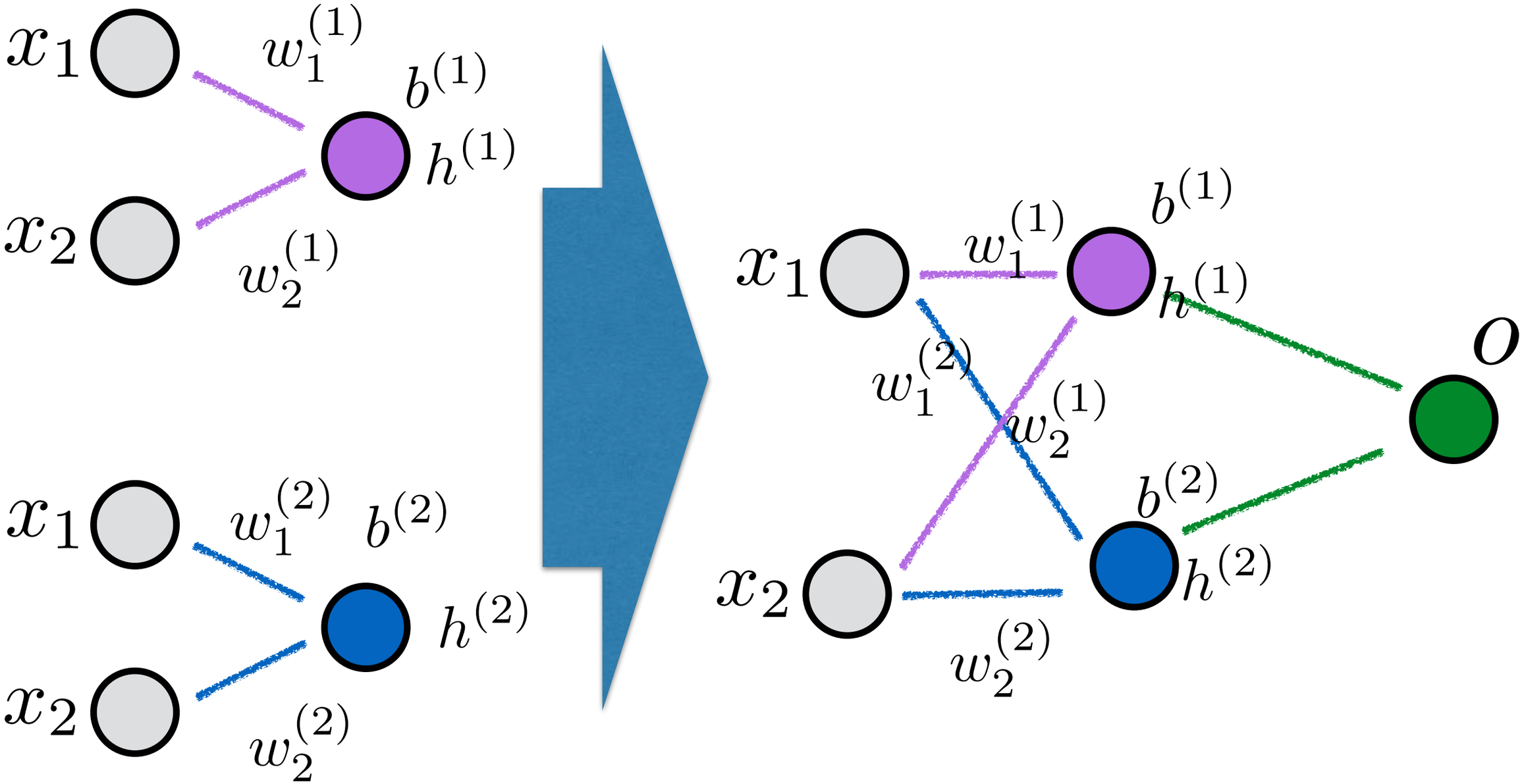


$$o^{(2)} = w_1^{(2)} x_1 + w_2^{(2)} x_2 + b^{(2)}$$

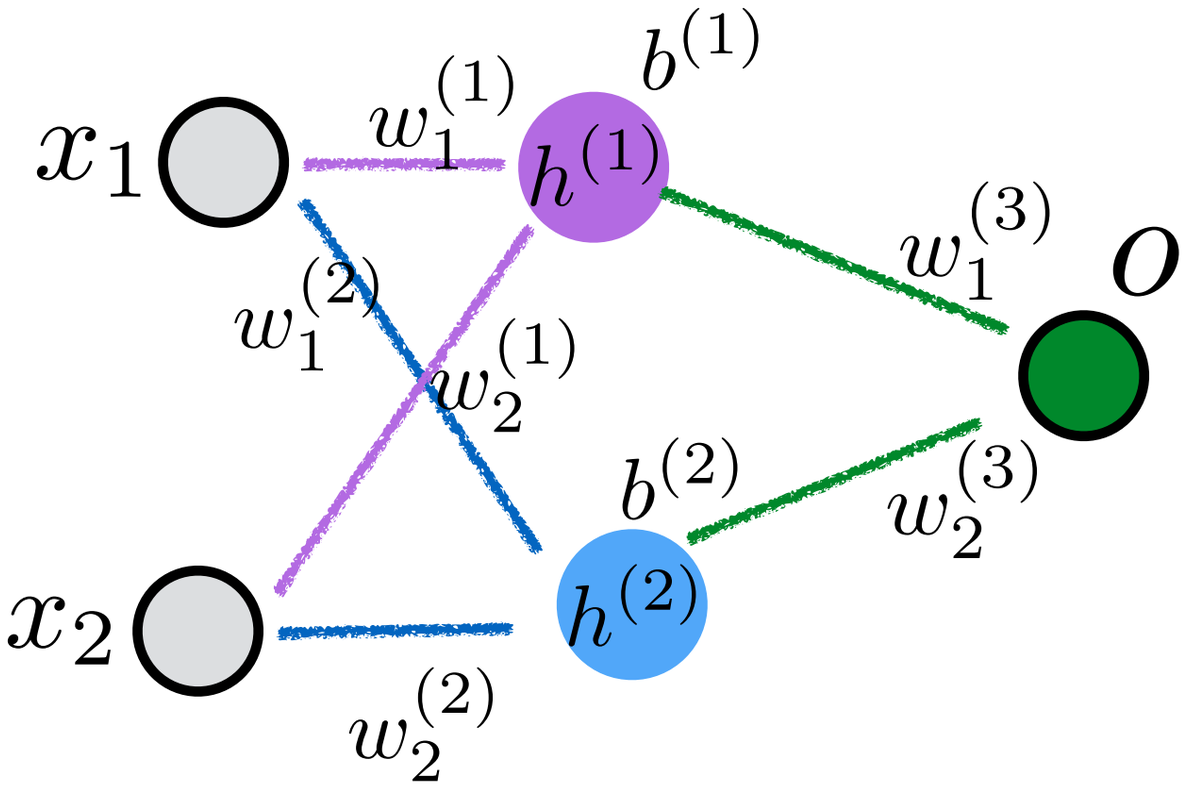
Stacked up


x_1
$w_1^{(1)}$
$b^{(1)}$
$o^{(1)}$
x_2
$w_2^{(1)}$
x_1
$w_1^{(2)}$
$b^{(2)}$
$o^{(2)}$
x_2
$w_2^{(2)}$
x_1
$w_1^{(1)}$
$b^{(1)}$
$o^{(1)}$
O
$w_1^{(2)}$
$w_2^{(1)}$
$b^{(2)}$
$o^{(2)}$
x_2
$w_2^{(2)}$

Stacked up


x_1
$w_1^{(1)}$
$b^{(1)}$
$h^{(1)}$
x_2
$w_2^{(1)}$
x_1
$w_1^{(2)}$
$b^{(2)}$
$h^{(2)}$
x_2
$w_2^{(2)}$
x_1
$w_1^{(1)}$
$b^{(1)}$
$h^{(1)}$
$w_1^{(2)}$
$w_2^{(1)}$
O
$b^{(2)}$
$h^{(2)}$
x_2
$w_2^{(2)}$

Stacked up

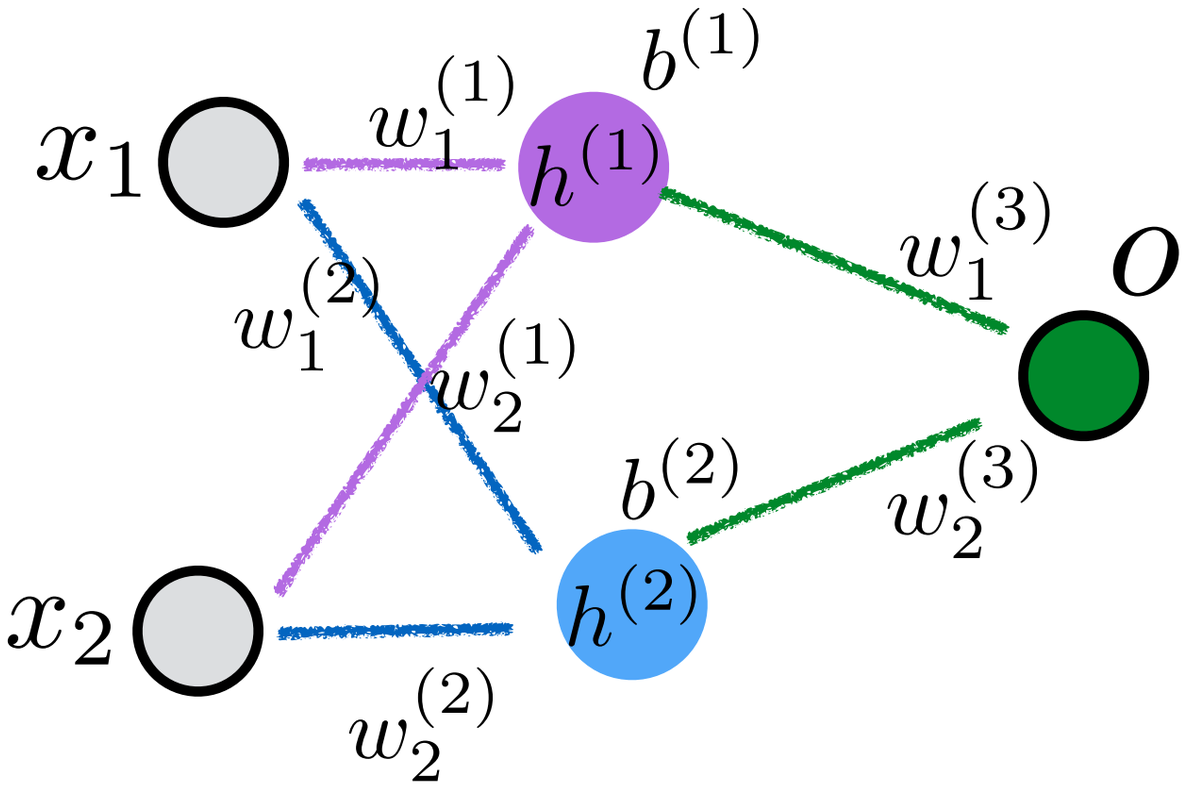


$$h^{(1)} = w_1^{(1)} x_1 + w_2^{(1)} x_2 + b^{(1)}$$

$$h^{(2)} = w_1^{(2)} x_1 + w_2^{(2)} x_2 + b^{(2)}$$

$$o = w_1^{(3)} h^{(1)} + w_2^{(3)} h^{(2)} + b^{(3)}$$

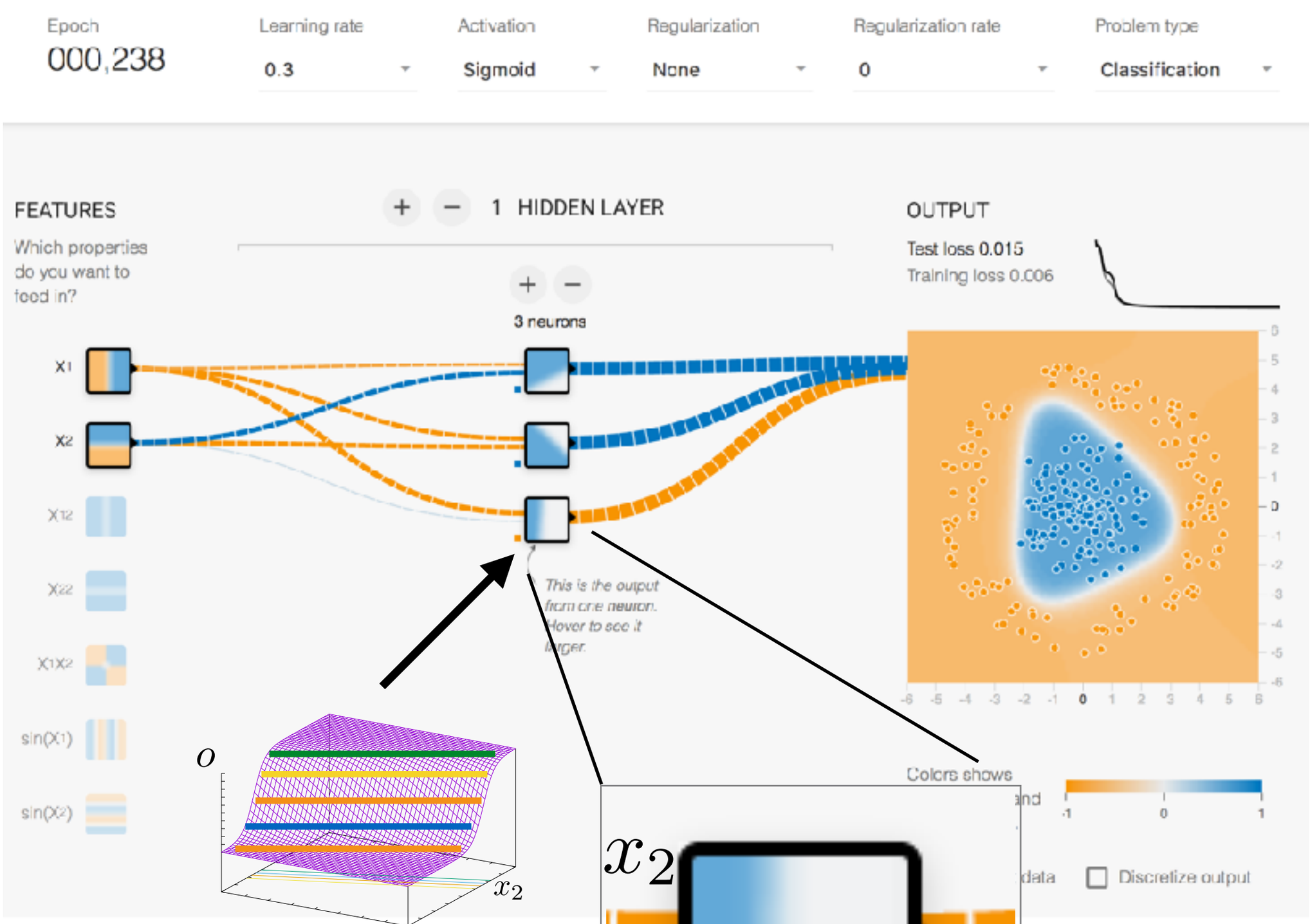
Stacked up with general activation function

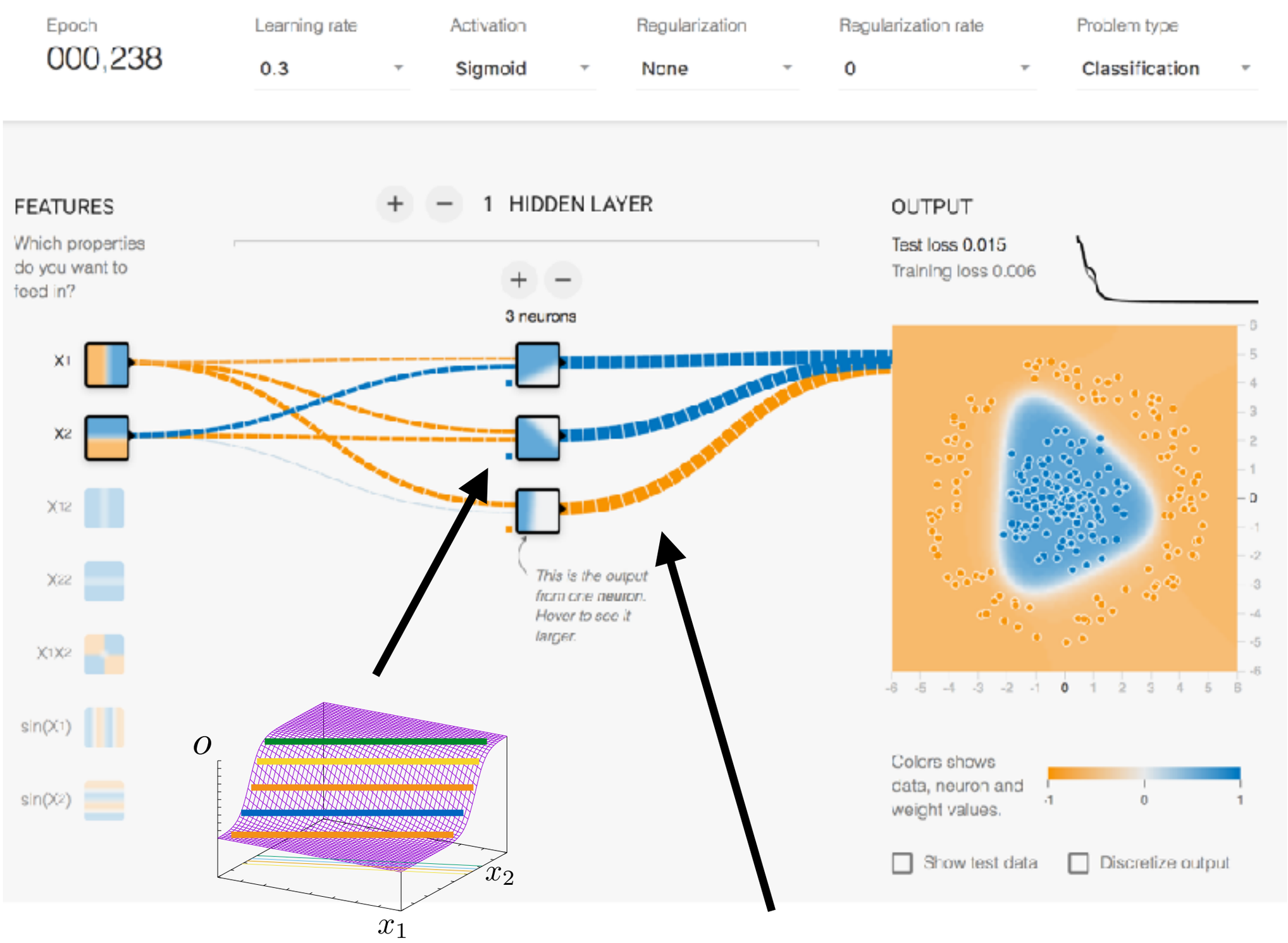


$$h^{(1)} = \sigma(w_1^{(1)} x_1 + w_2^{(1)} x_2 + b^{(1)})$$

$$h^{(2)} = \sigma(w_1^{(2)} x_1 + w_2^{(2)} x_2 + b^{(2)})$$

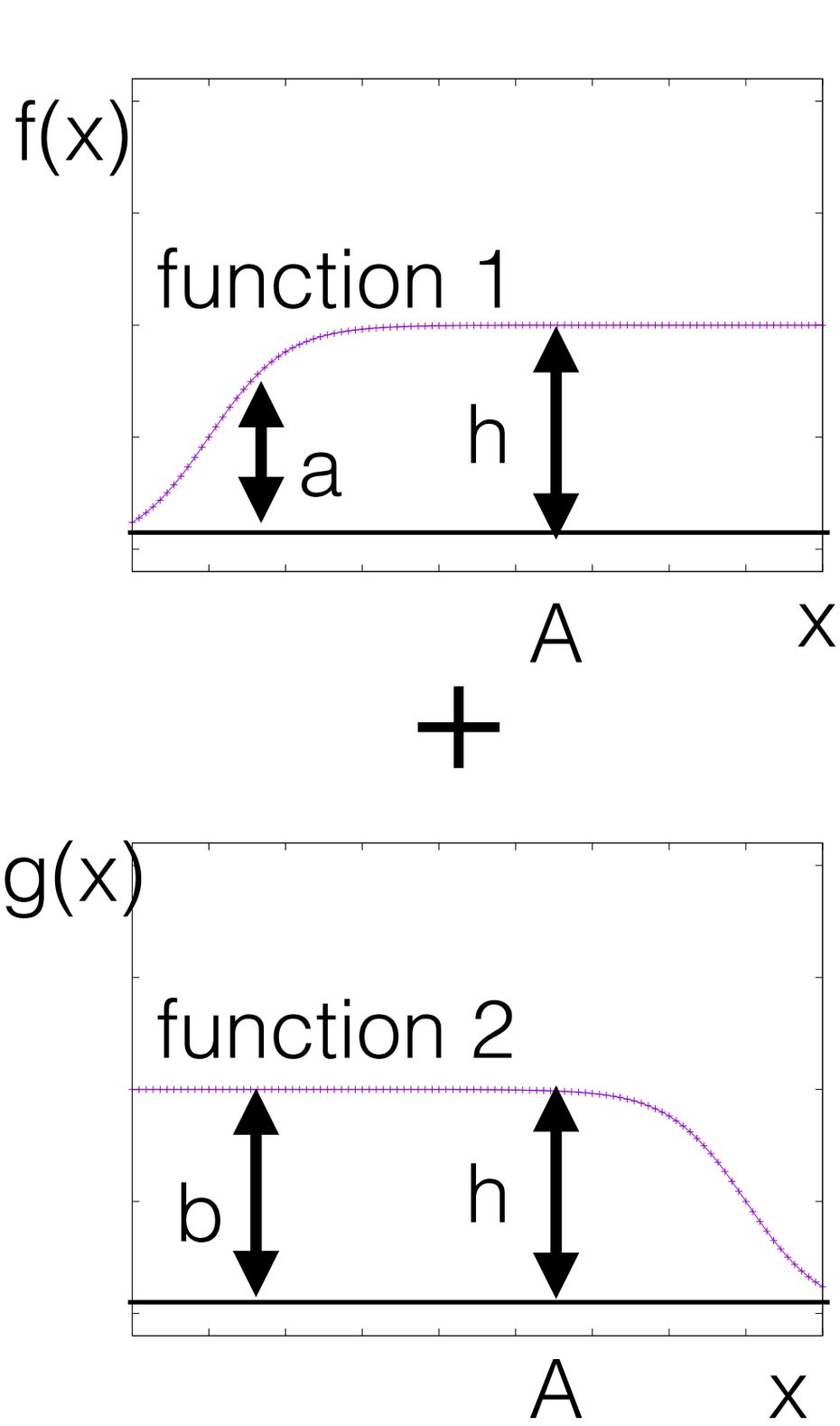
$$o = \sigma(w_1^{(3)} h^{(1)} + w_2^{(3)} h^{(2)} + b^{(3)})$$


Epoch
000,238
Learning rate
0.3
Activation
Sigmoid
Regularization
None
Regularization rate
0
Problem type
Classification
FEATURES
Which properties do you want to feed in?
1 HIDDEN LAYER
3 neurons
OUTPUT
Test loss 0.015
Training loss 0.006
X1
X2
X12
X22
X1X2
sin(X1)
sin(X2)
This is the output from one neuron. Hover to see it larger.
Colors shows
data
Discretize output
o
x_1
x_2
x_2
x_1

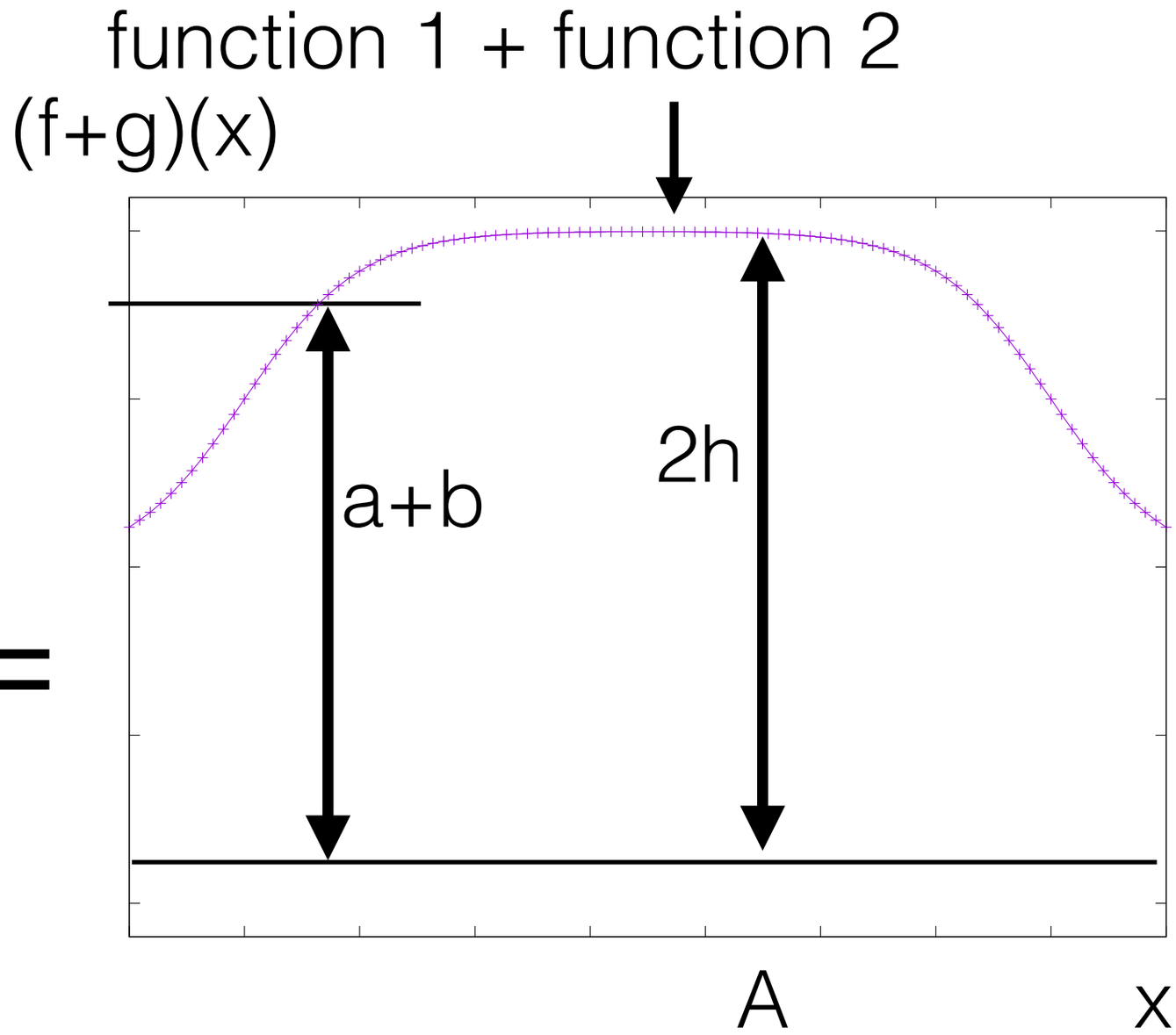


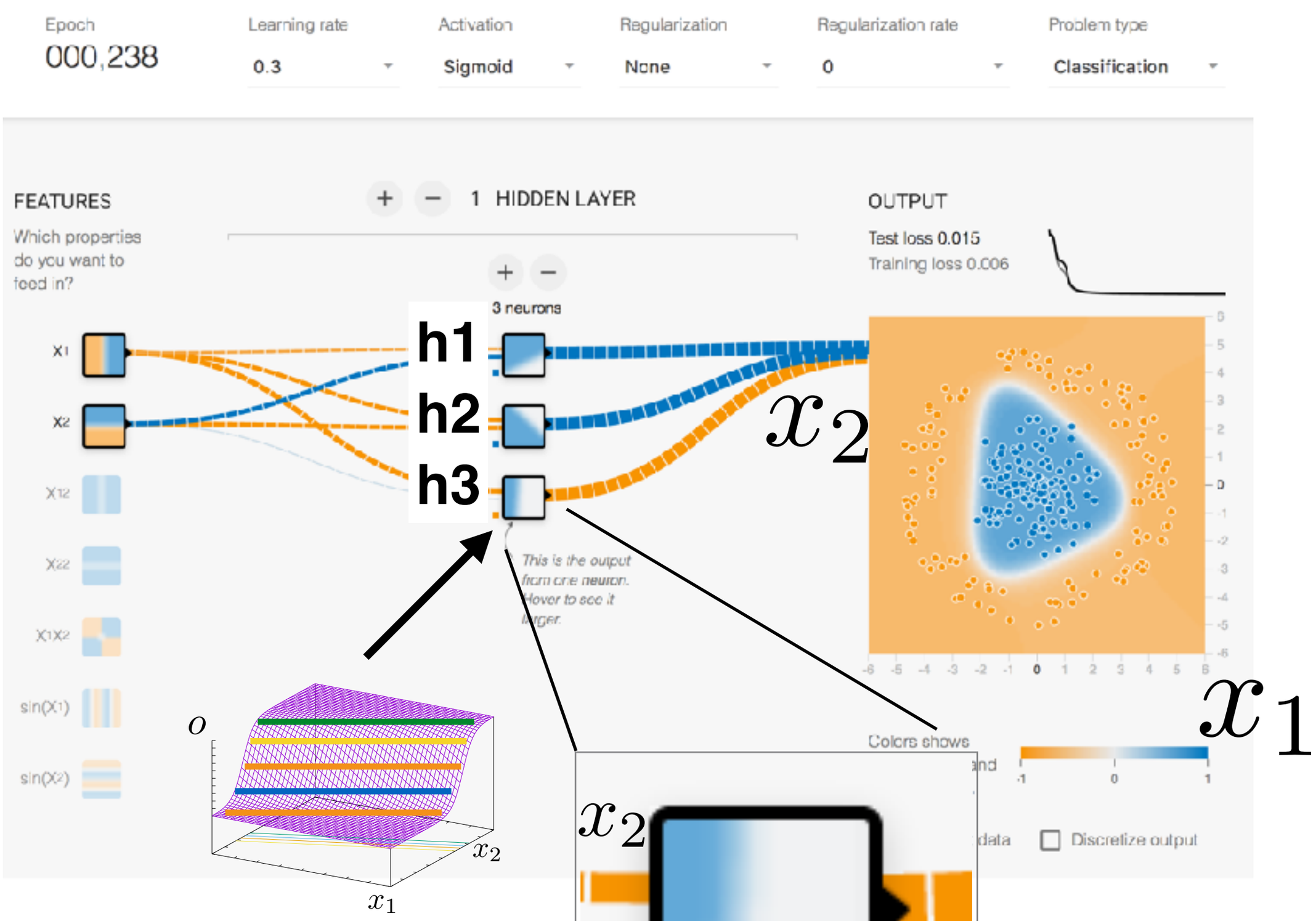
These three surfaces add to form a triangular decision boundary!

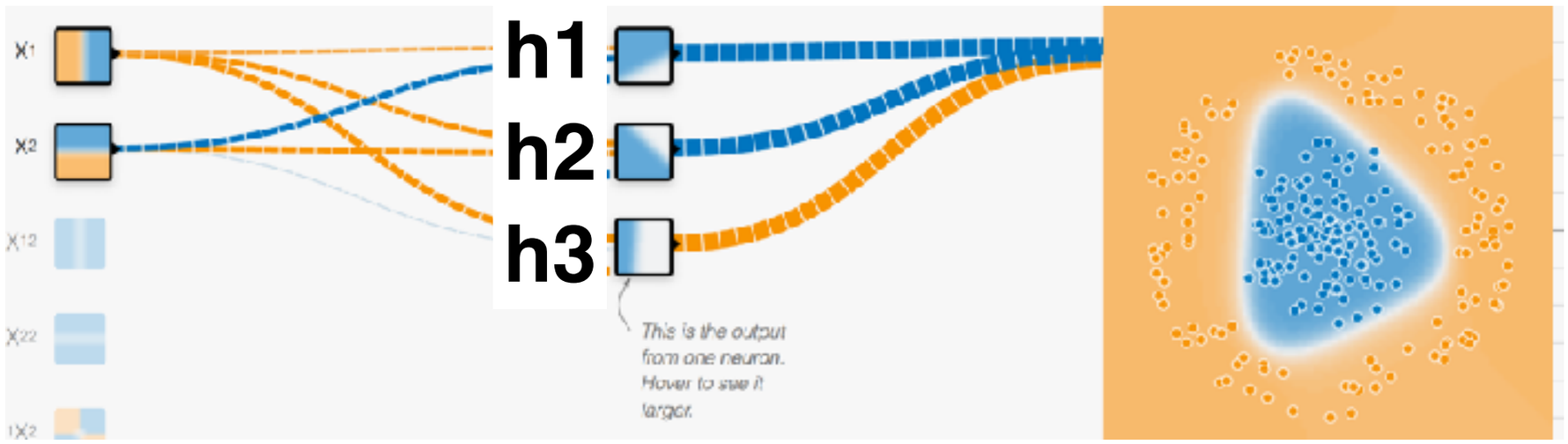
Adding functions



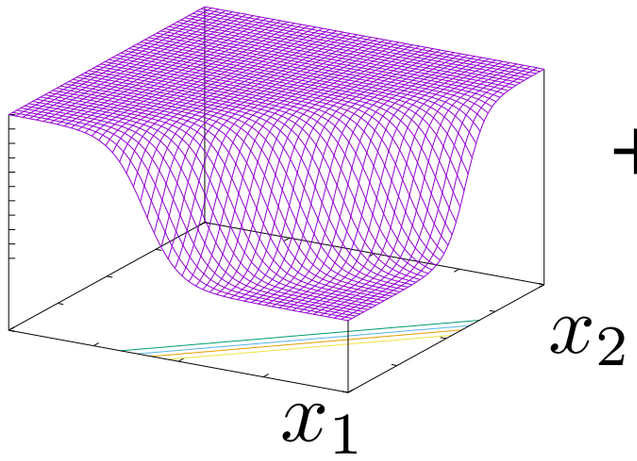
=




Epoch
000,238
Learning rate
0.3
Activation
Sigmoid
Regularization
None
Regularization rate
0
Problem type
Classification
FEATURES
Which properties do you want to feed in?
1 HIDDEN LAYER
3 neurons
OUTPUT
Test loss 0.015
Training loss 0.006
X1
X2
X12
X22
X1X2
sin(X1)
sin(X2)
h1
h2
h3
This is the output from one neuron. Hover to see it larger.
Colors shows
data
Discretize output
x_2
x_1
o
x_2
x_1
x_2
x_1

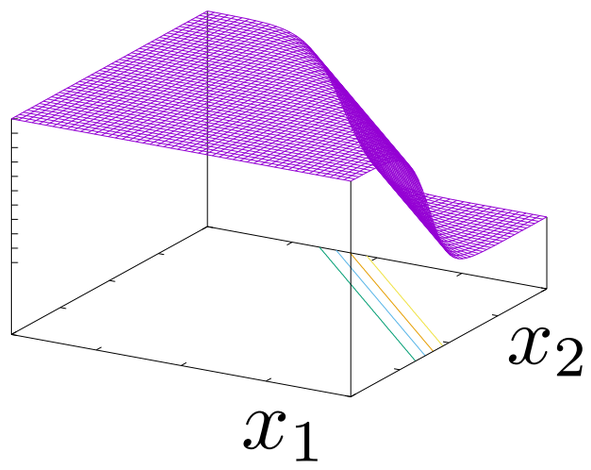


$h_1(x_1, x_2)$



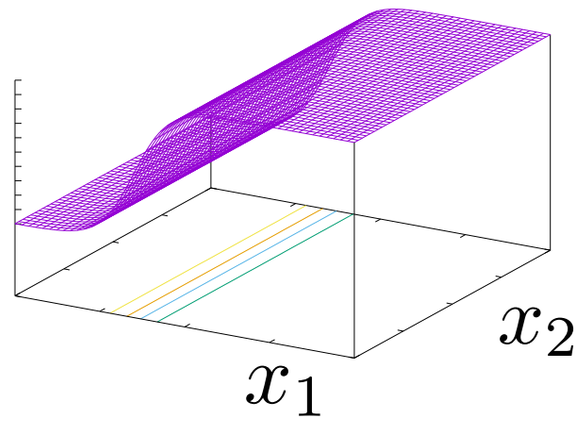
+

$h_2(x_1, x_2)$

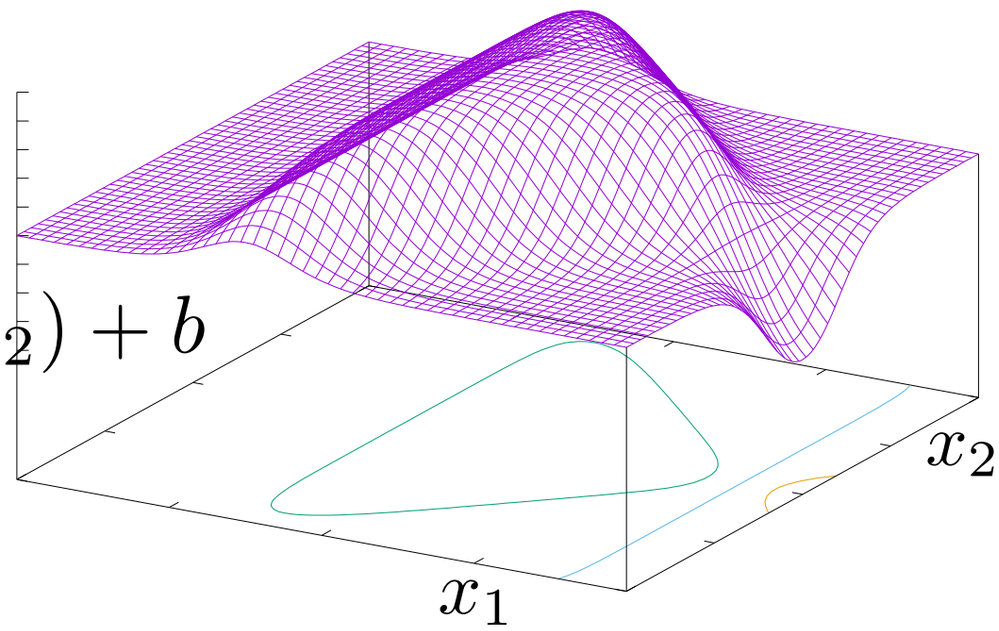


+

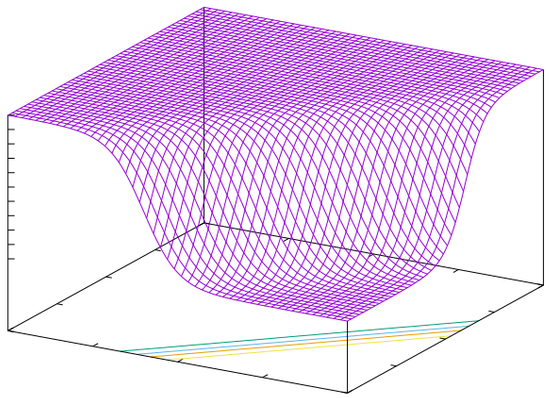
$-h_3(x_1, x_2)$

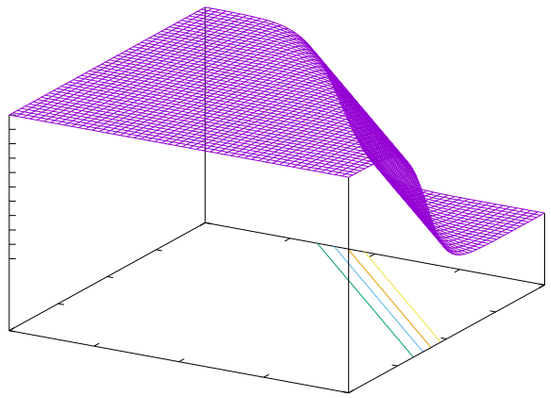


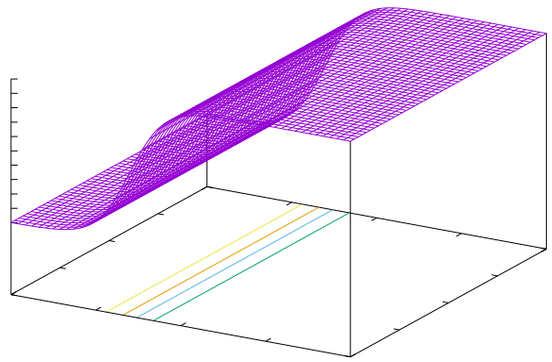
$w_1 h_1(x_1, x_2) + w_2 h_2(x_1, x_2) + w_3 h_3(x_1, x_2) + b$

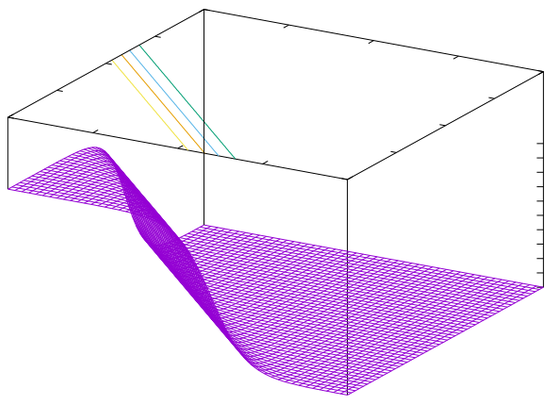


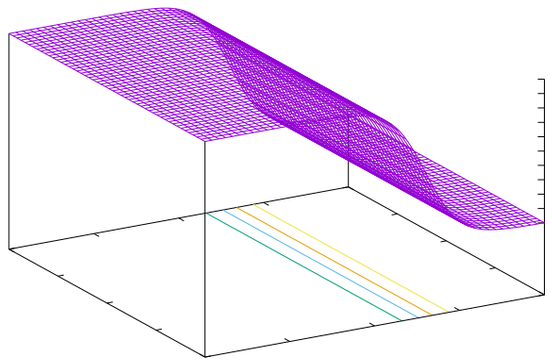
If you add up enough “step” surfaces, are you able to form any functions?

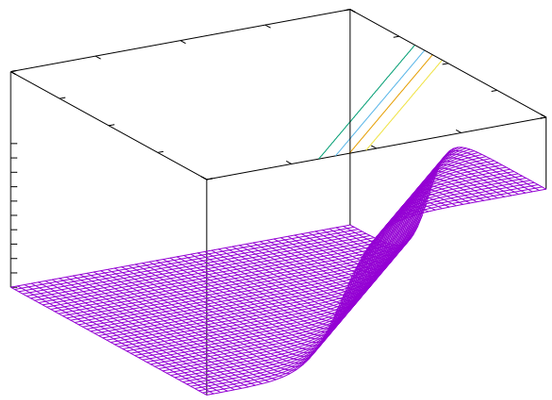


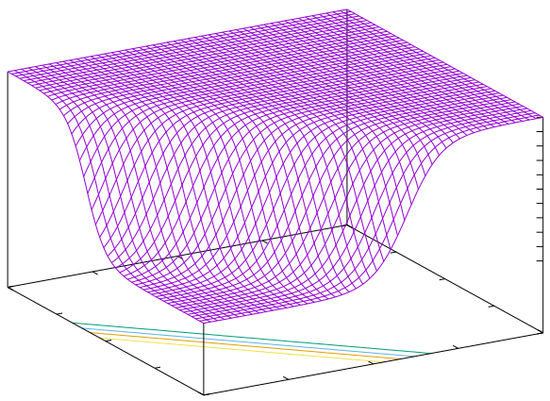


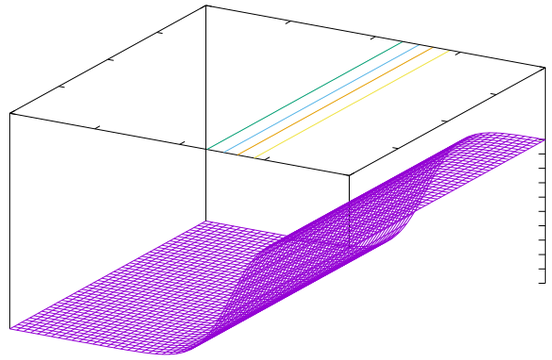


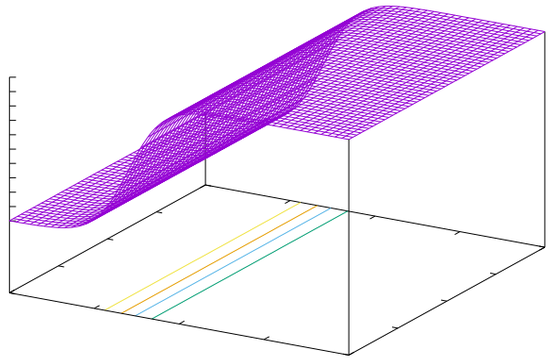








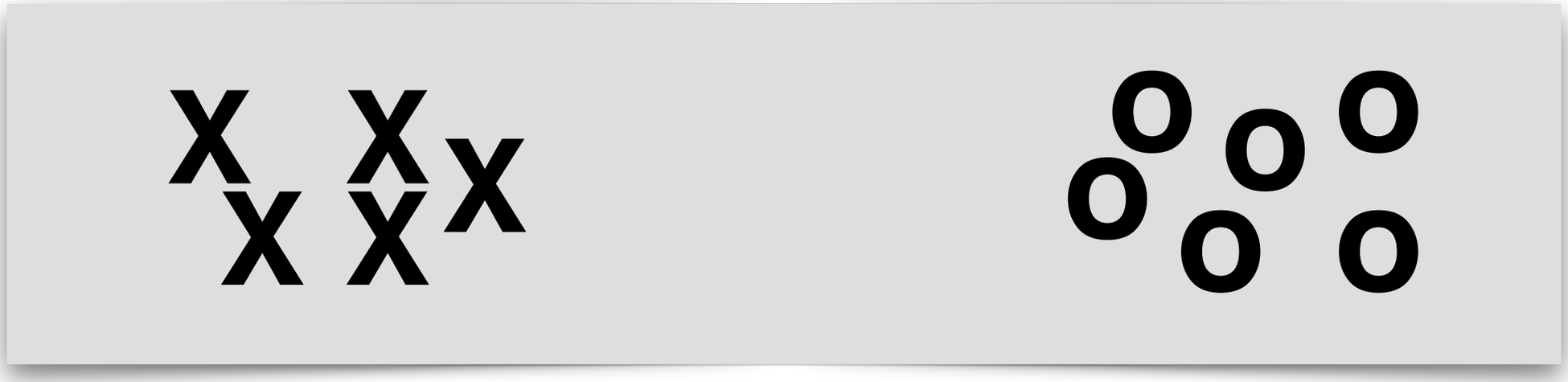




neural network fingers activities

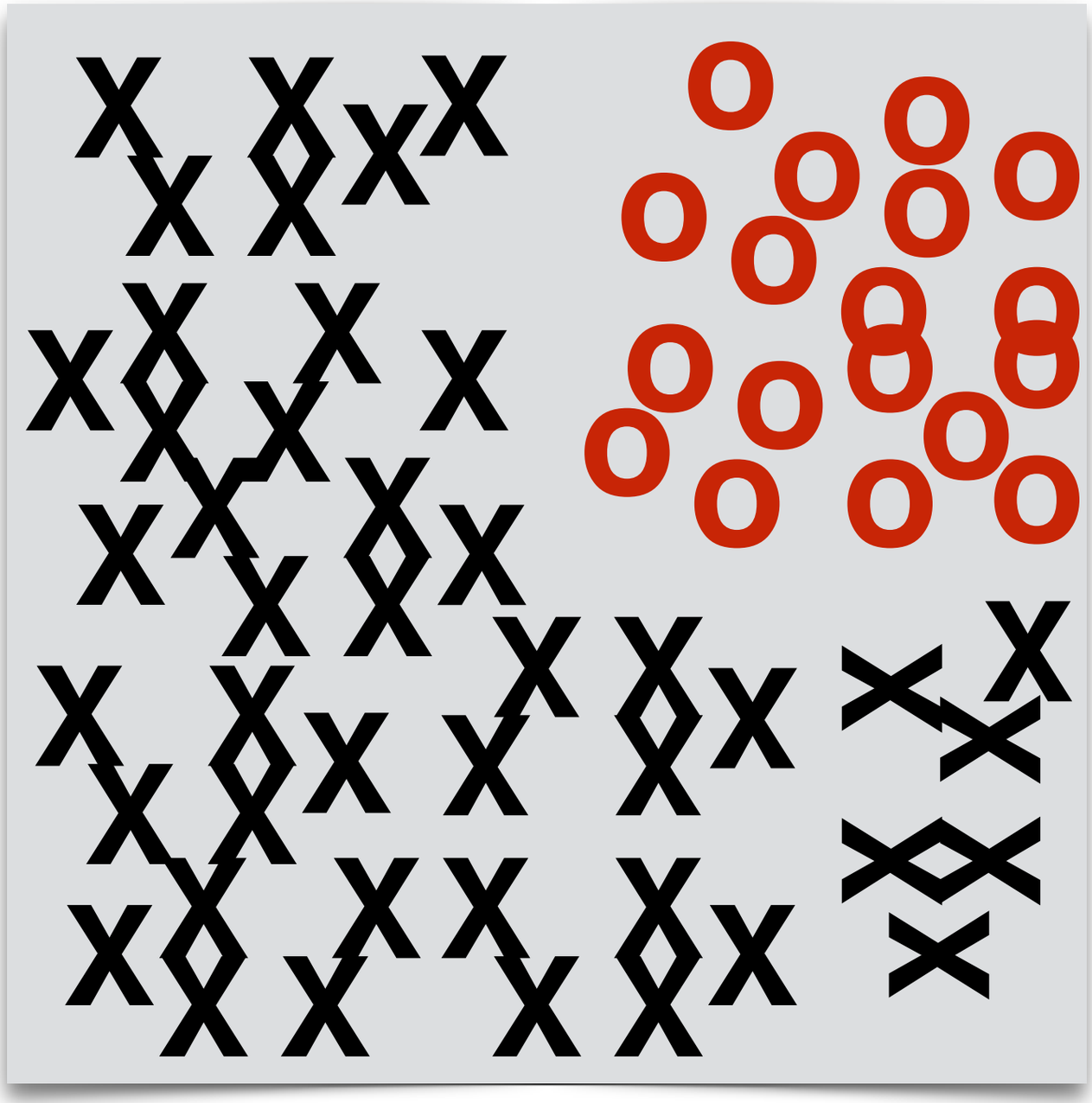
Activity one

Take out a piece of paper, draw patterns as shown



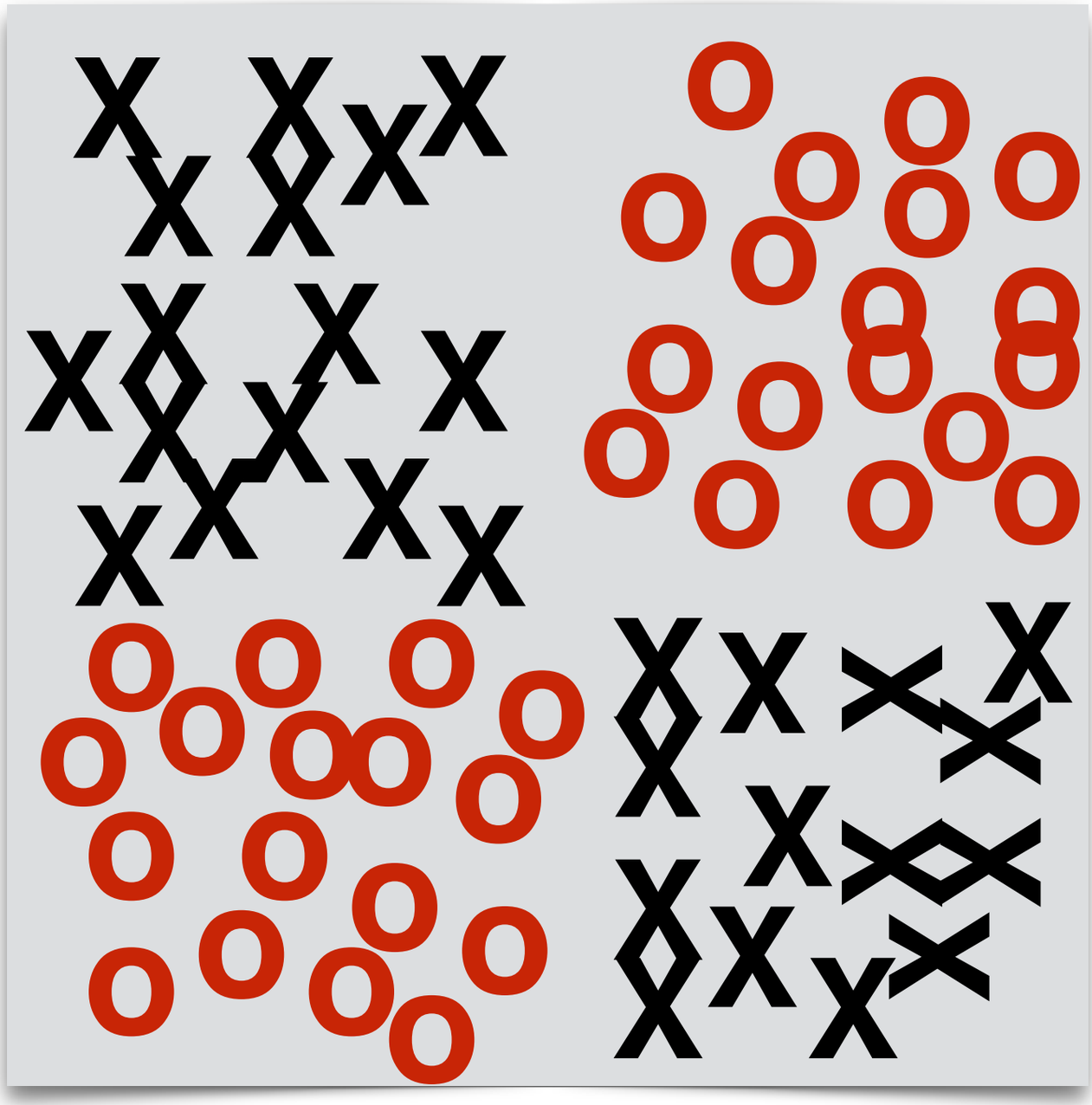
Task: Cut (tear) with one **straight line** to completely separate the “X” and “O”

Activity two



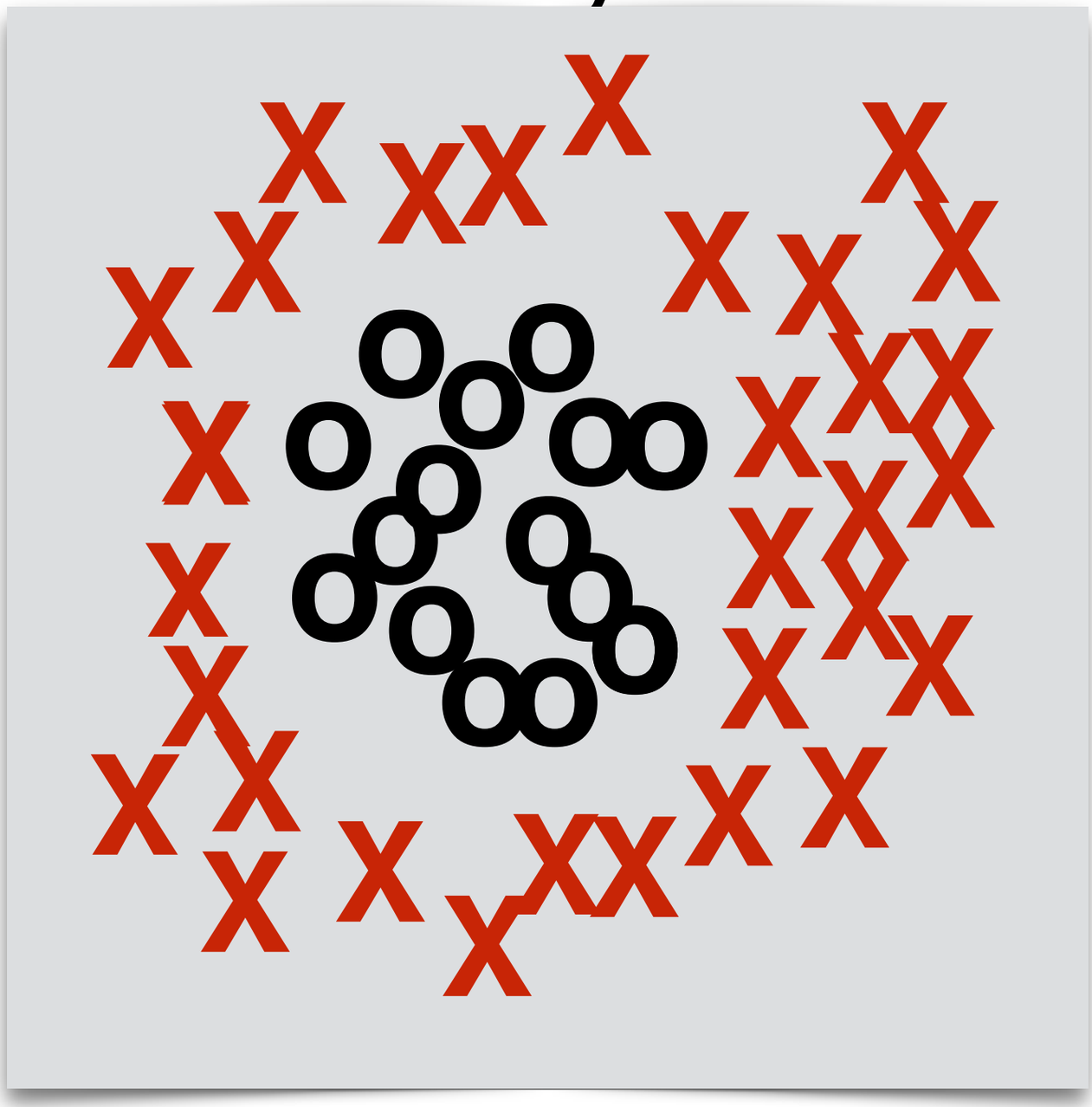
Cut (tear) with **one** straight line!

Activity three



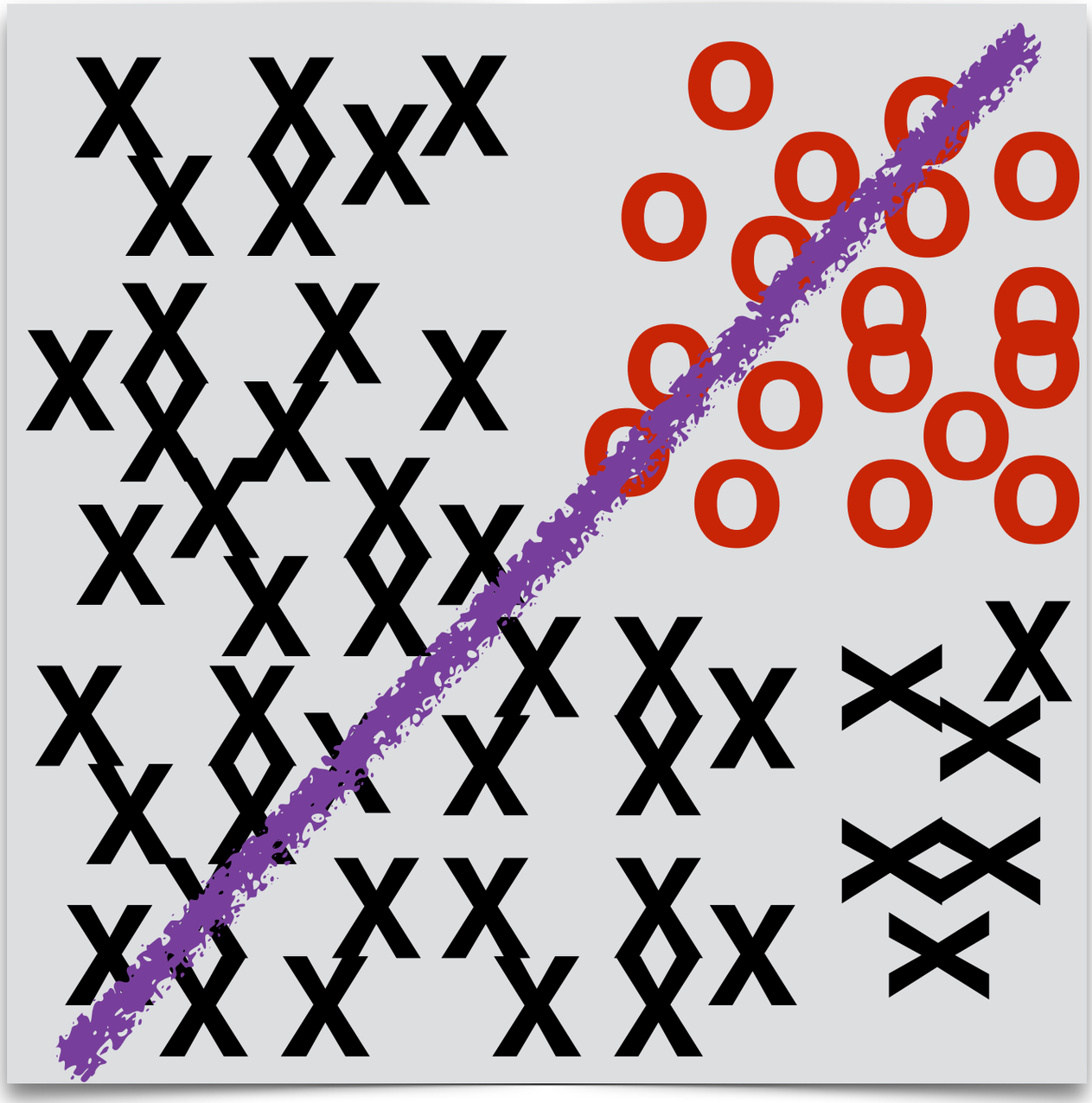
Cut (tear) with **one** straight line!

Activity four



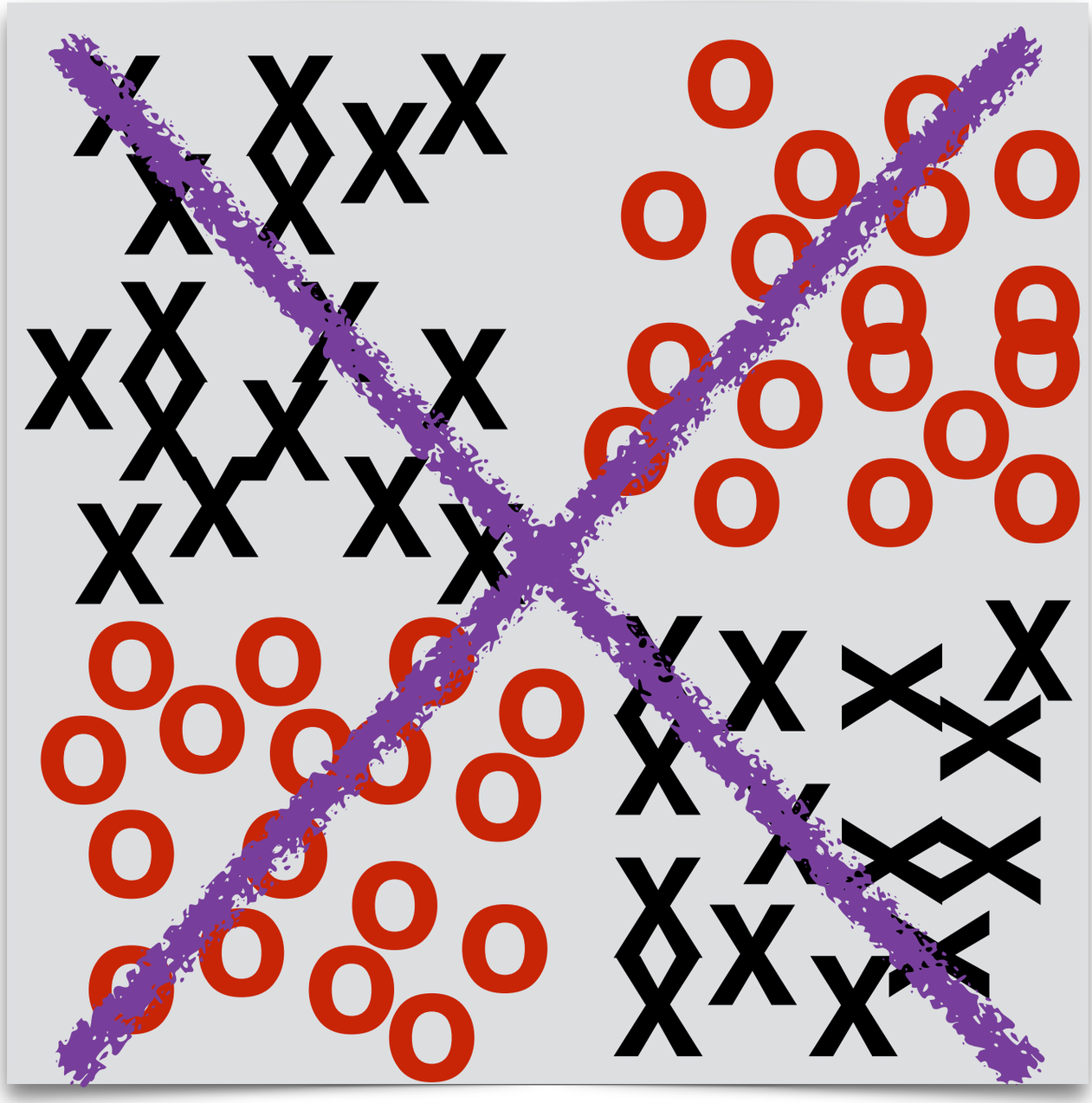
Cut (tear) with **one** straight line!

Activity two



Cut (tear) with **one** straight line!

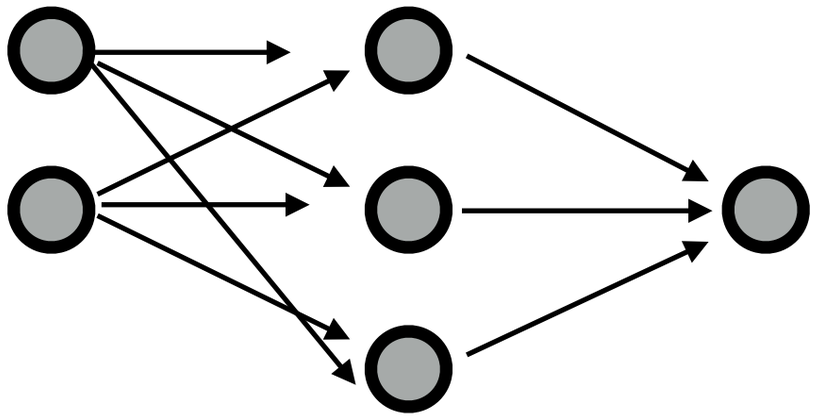
Activity three



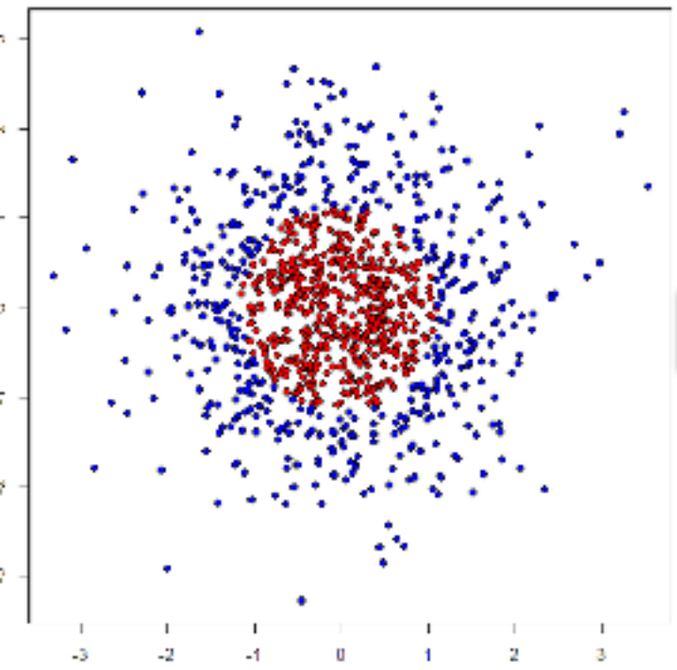
Cut (tear) with **one** straight line!

Manifold view of neural network

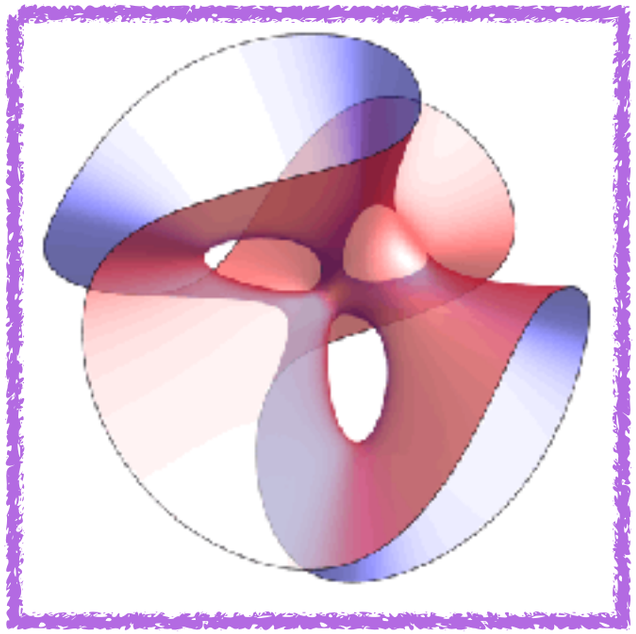
feed in data into first layer



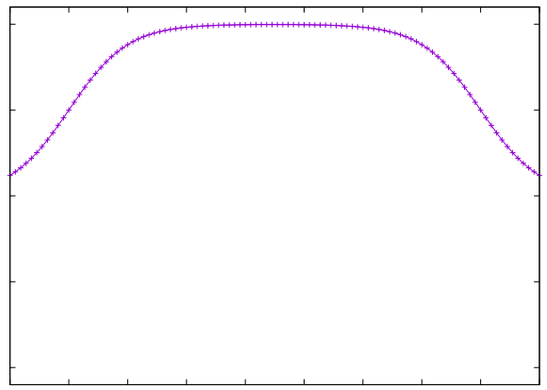
final layer presents the output of network



data



hidden layer

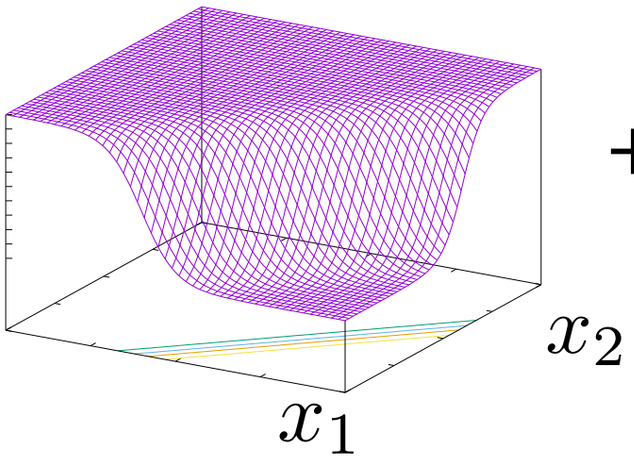


output

https://www.pinterest.com/pin/414260865705737484/

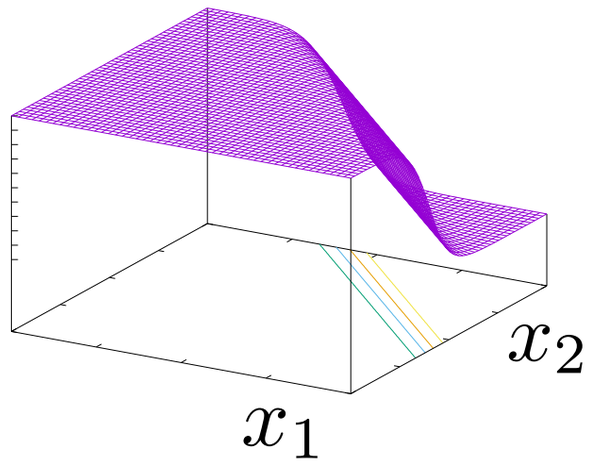
Function view of neural network

$h_1(x_1, x_2)$



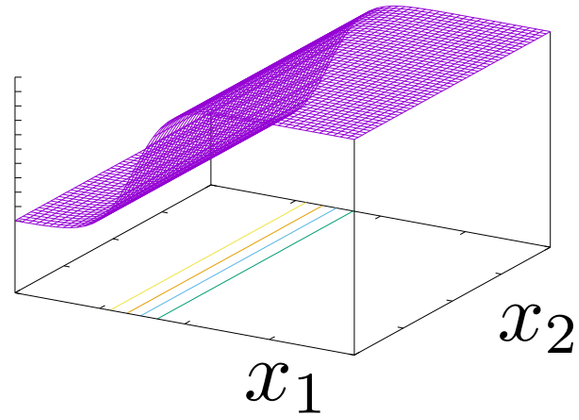
+

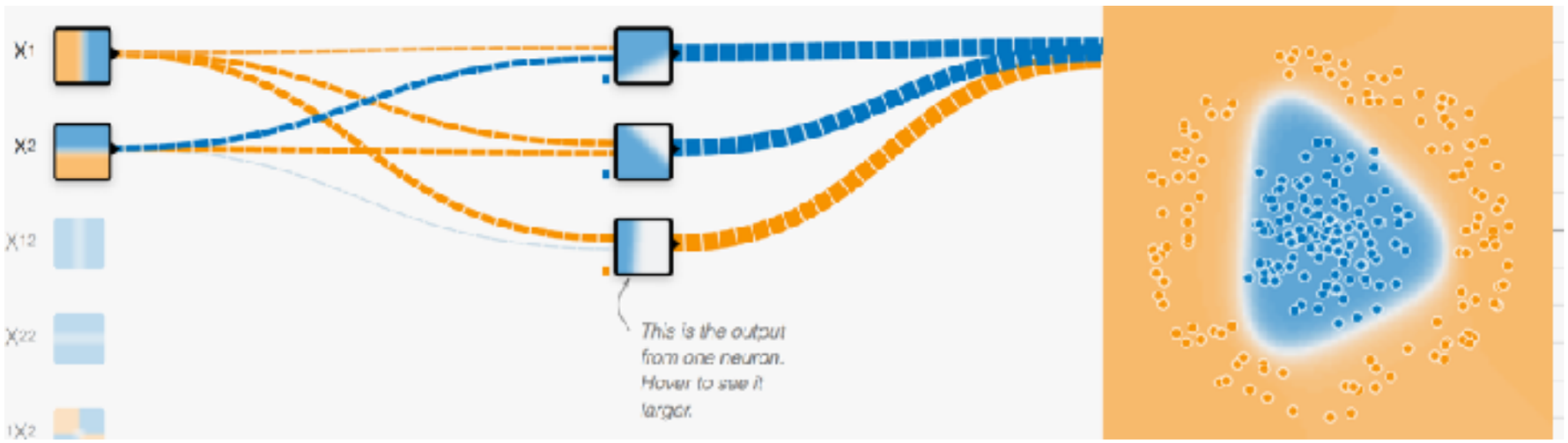
$h_2(x_1, x_2)$

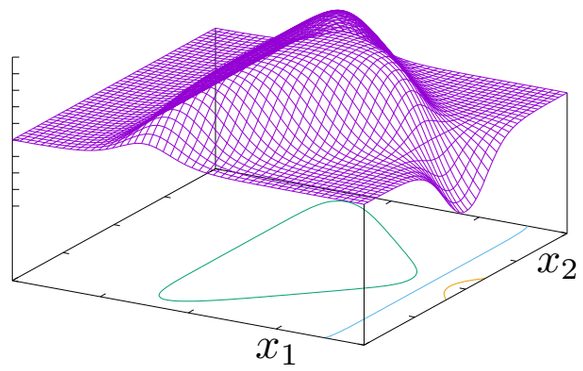


+

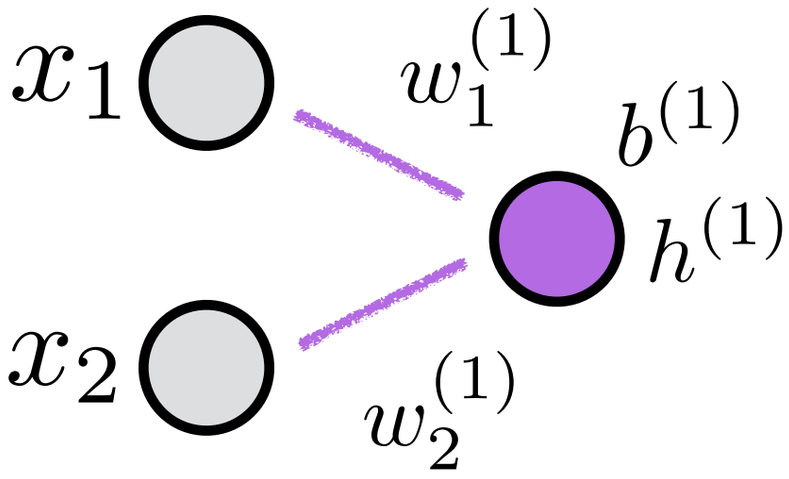
$-h_3(x_1, x_2)$

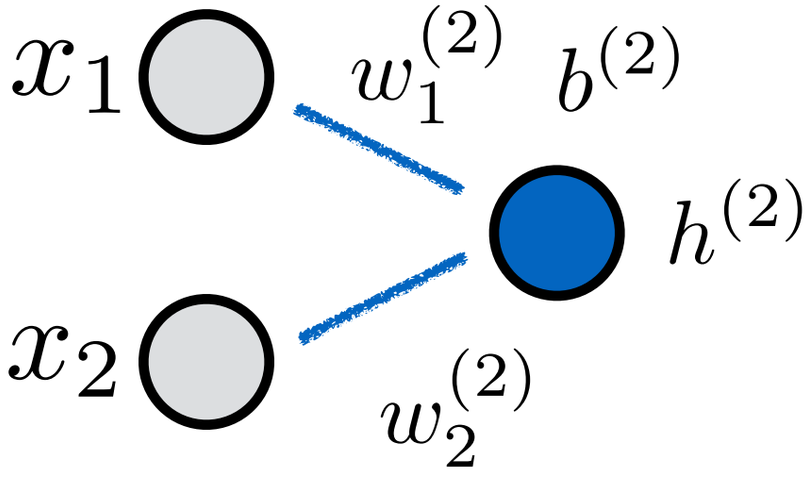


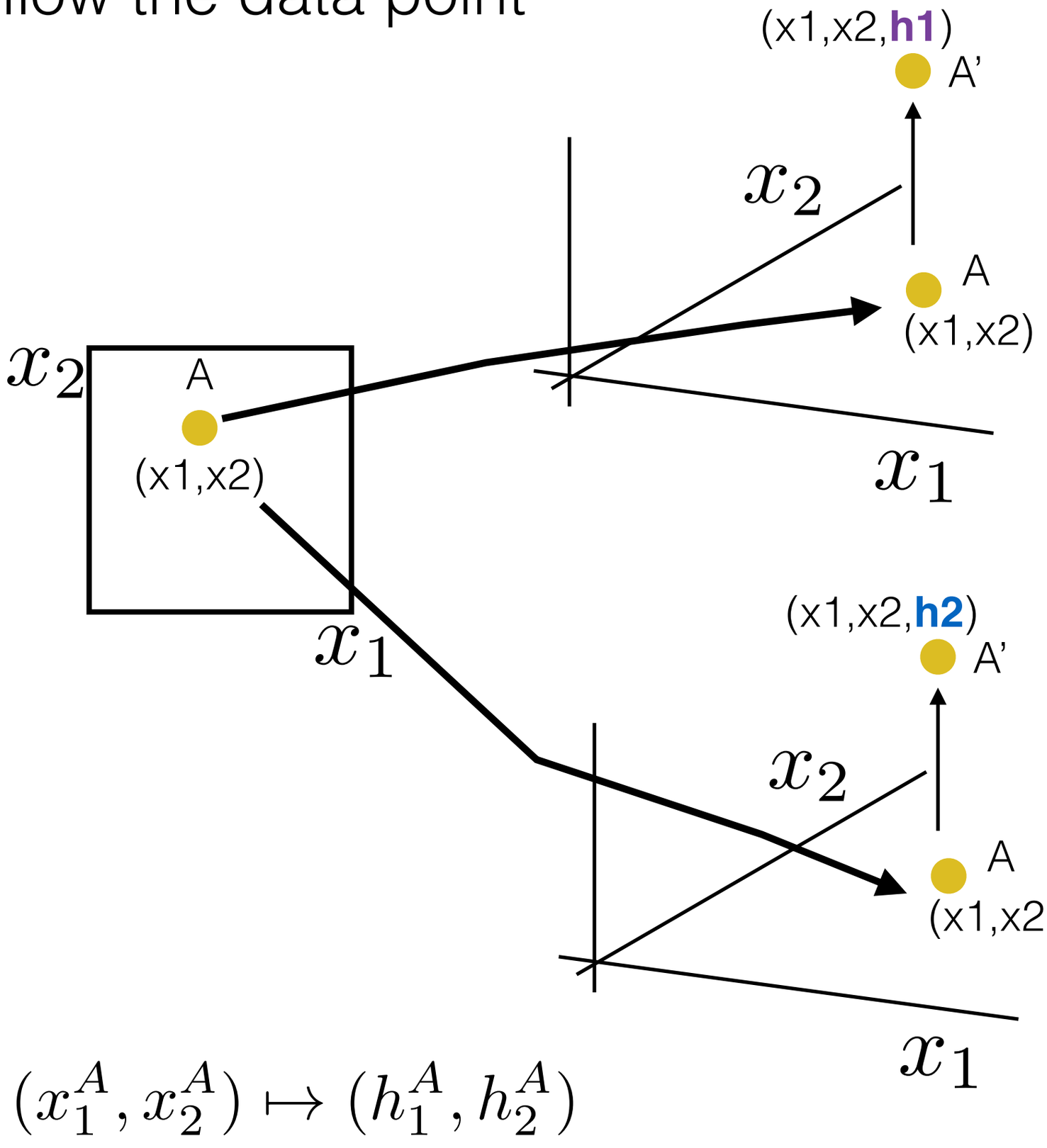




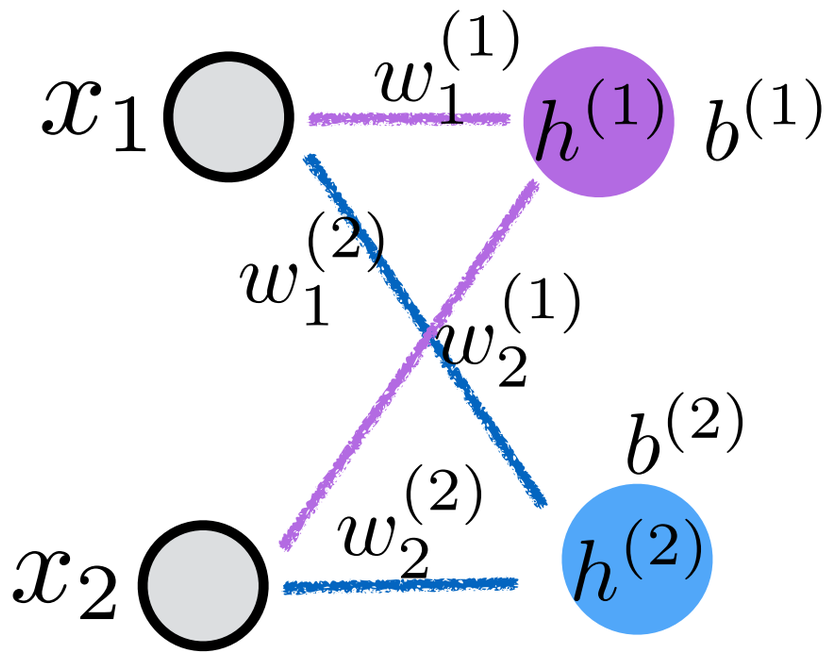
Follow the data point

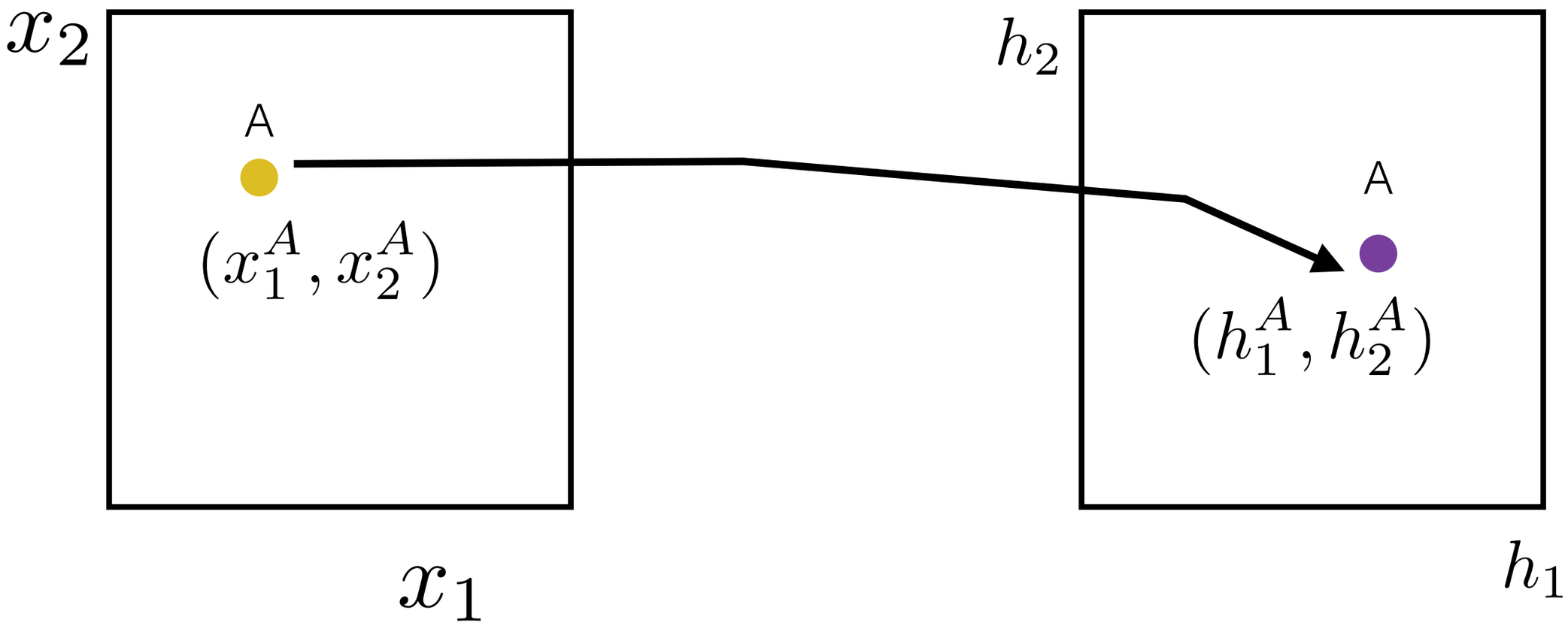




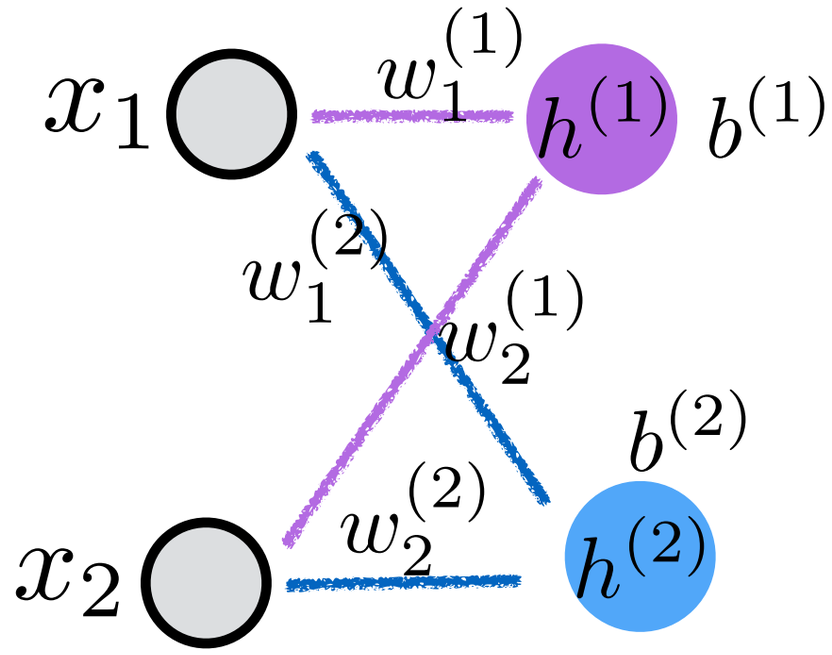


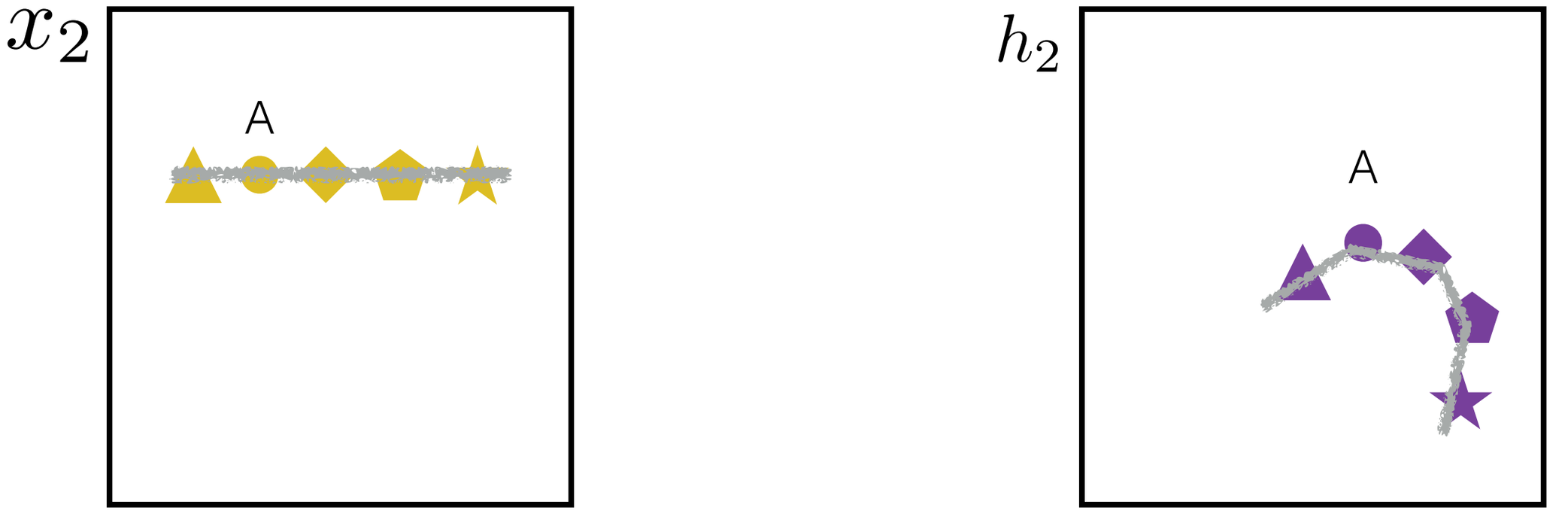
$$(x_1^A, x_2^A) \mapsto (h_1^A, h_2^A)$$



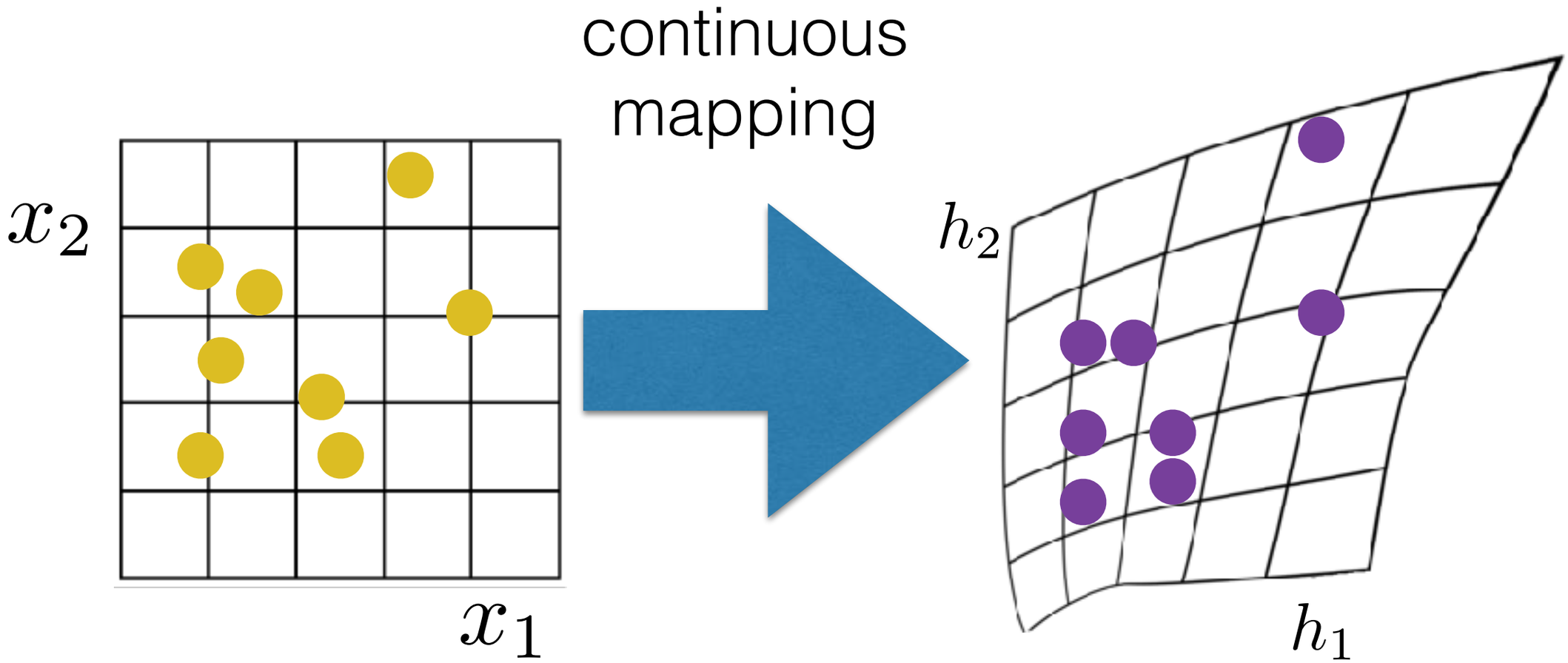


$$(x_1^A, x_2^A) \mapsto (h_1^A, h_2^A)$$

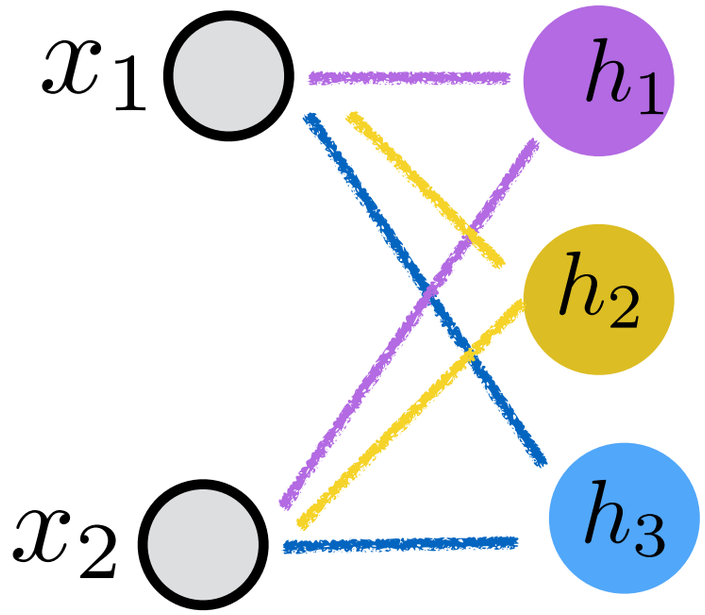


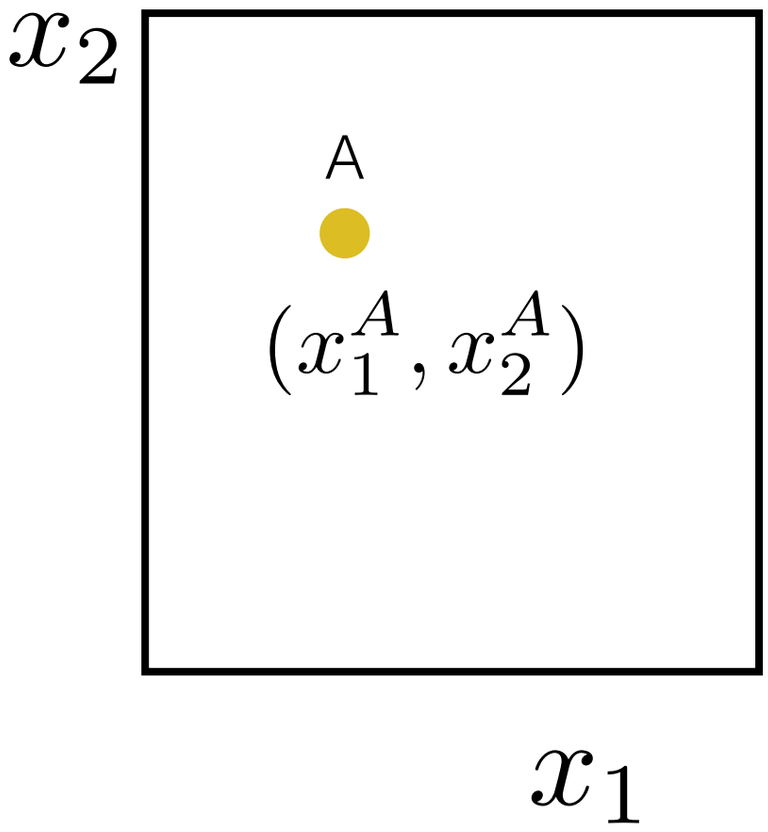


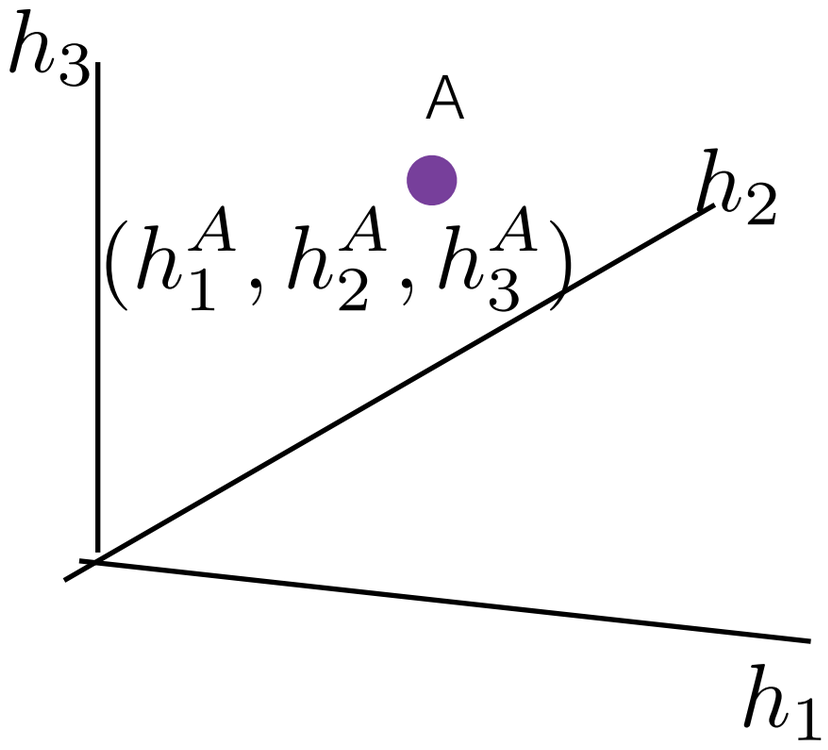
Neighbourhoods relationship is conserved

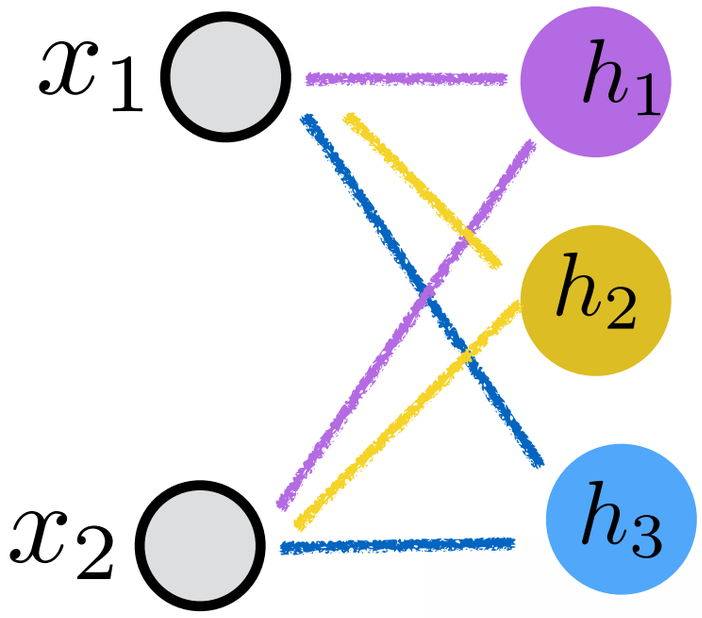


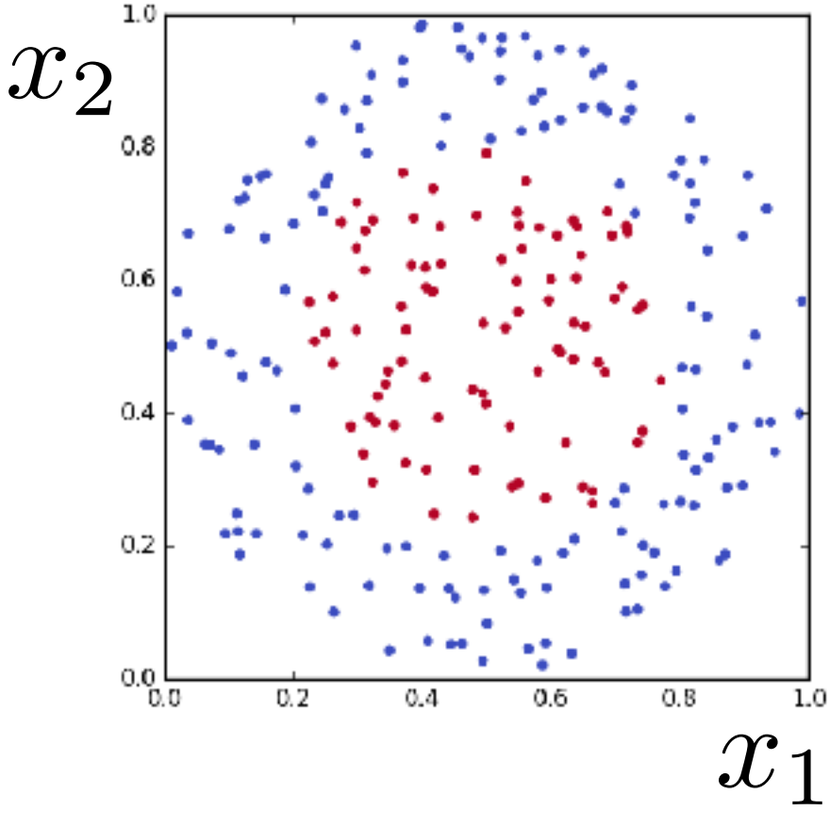
$$(x_1^A, x_2^A) \mapsto (h_1^A, h_2^A)$$

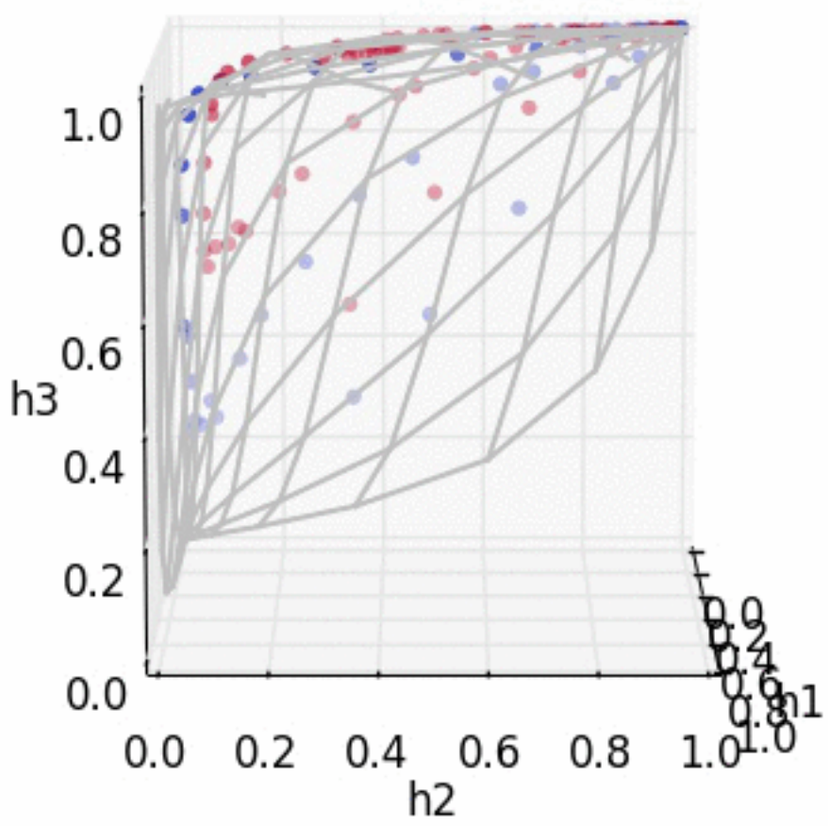

x_1
x_2
h_1
h_2
h_3

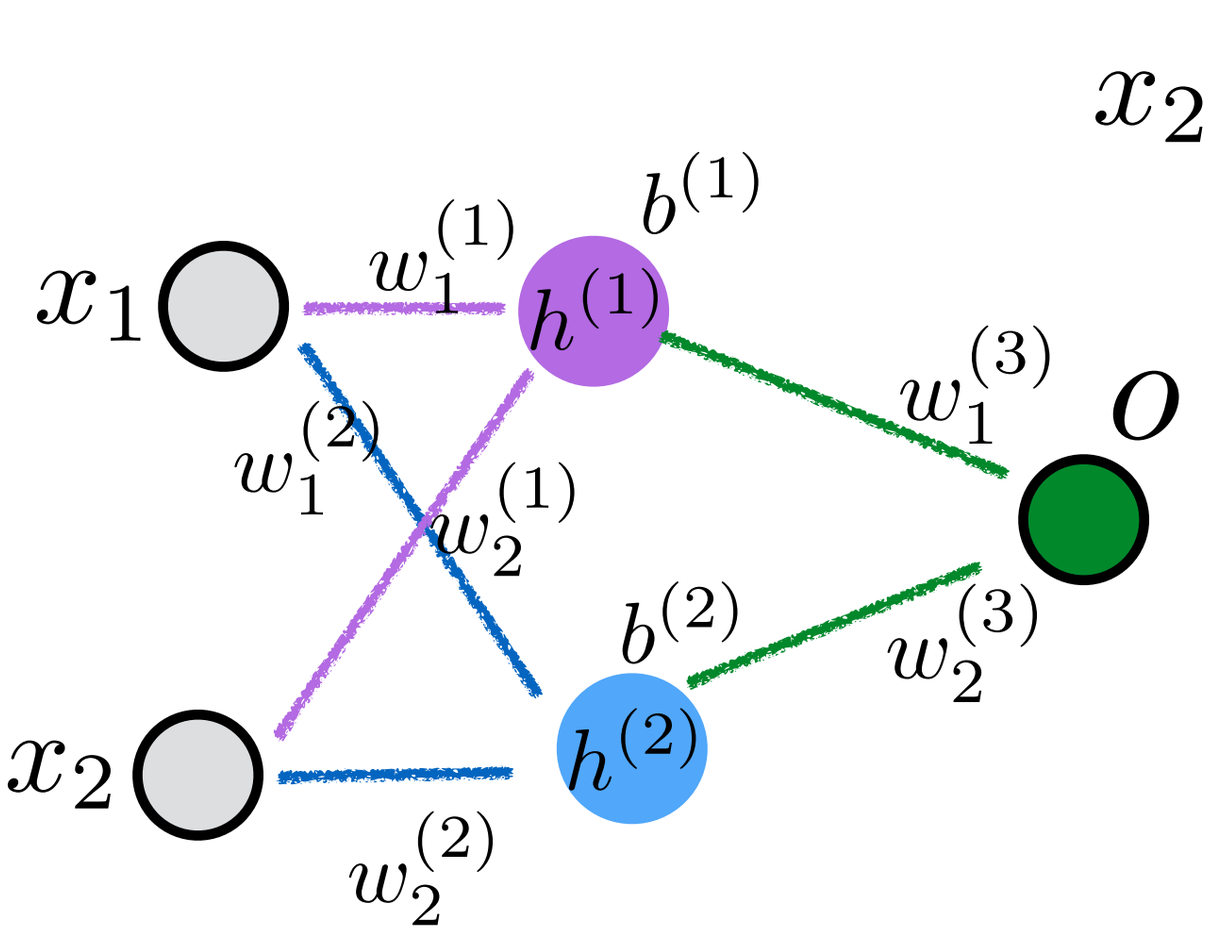

x_2
A
(x_1^A, x_2^A)
x_1

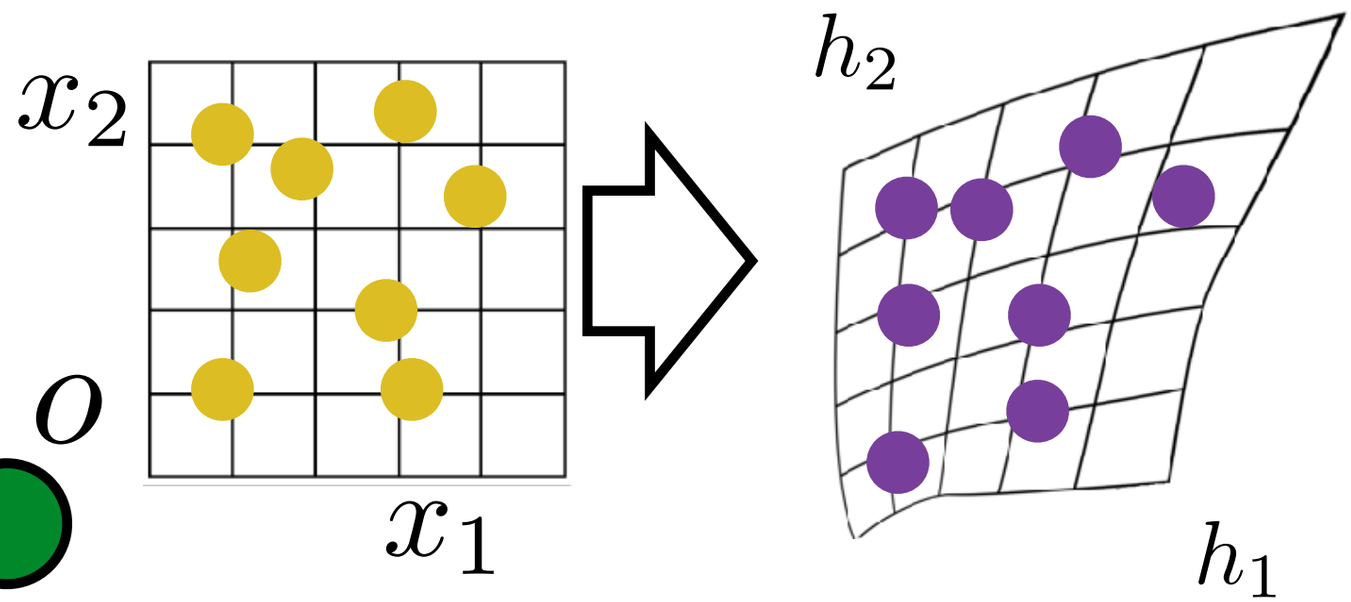

h_3
A
(h_1^A, h_2^A, h_3^A)
h_2
h_1

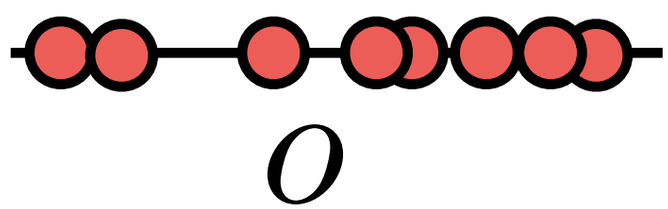

x_1
x_2
h_1
h_2
h_3


x_2
x_1
0.0
0.2
0.4
0.6
0.8
1.0


h3
h2
h1
0.0
0.2
0.4
0.6
0.8
1.0







$$h^{(1)} = \sigma(w_1^{(1)} x_1 + w_2^{(1)} x_2 + b^{(1)})$$

$$h^{(2)} = \sigma(w_1^{(2)} x_1 + w_2^{(2)} x_2 + b^{(2)})$$

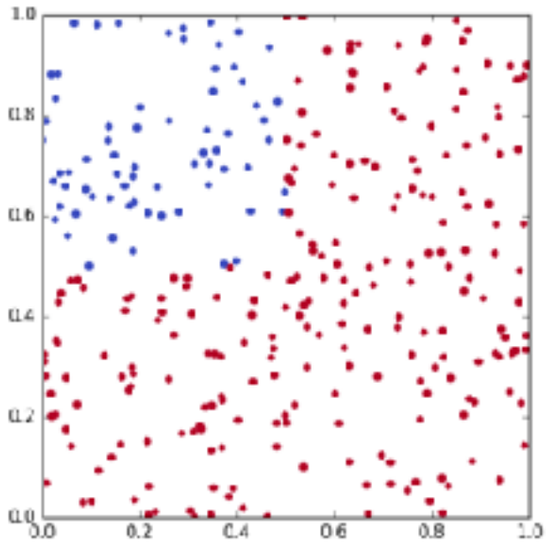
$$o = \sigma(w_1^{(3)} h^{(1)} + w_2^{(3)} h^{(2)} + b^{(3)})$$

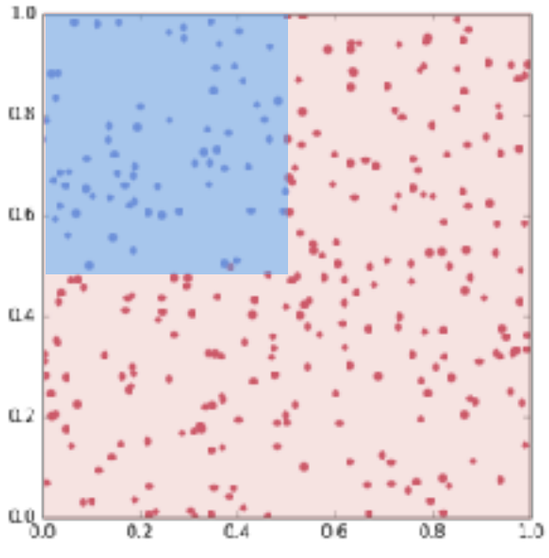
Some real life examples

courtesy of Connie Kou Khor Li

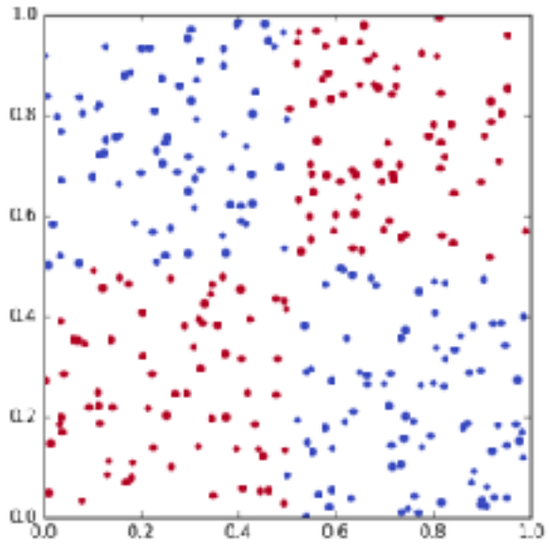
TA for this course years ago

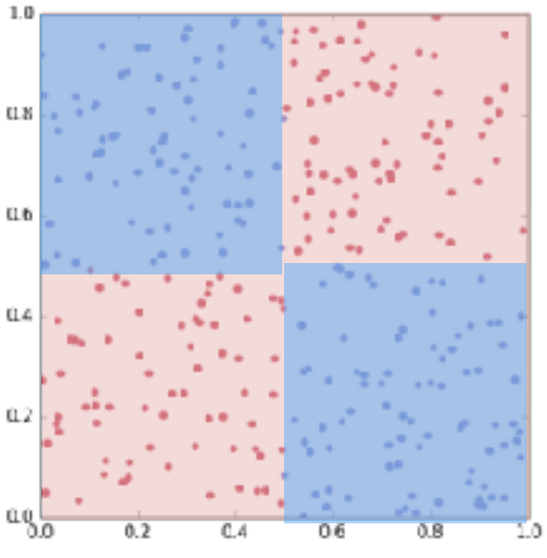
The Angle Data



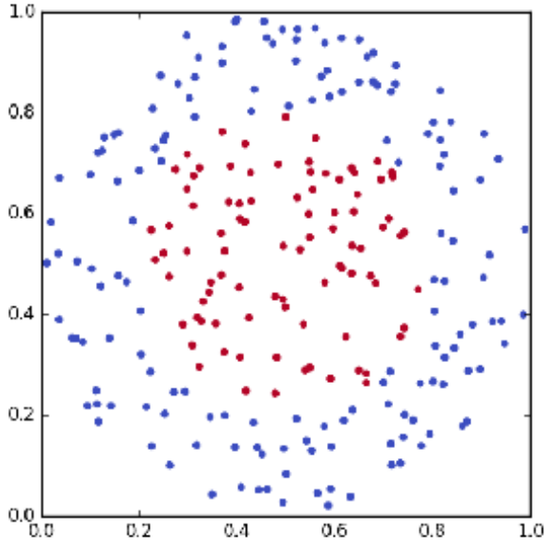


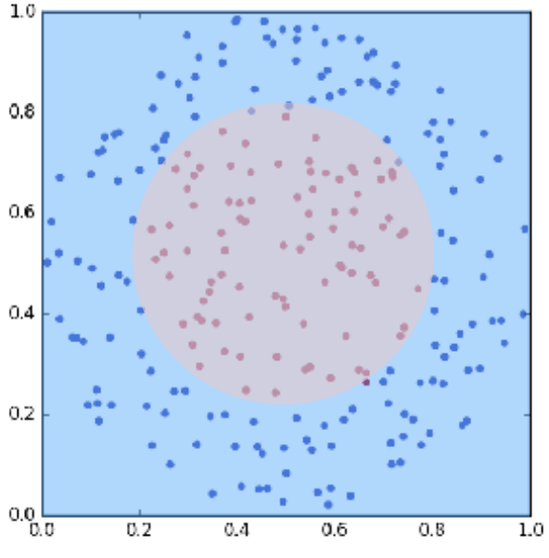
The XOR Data



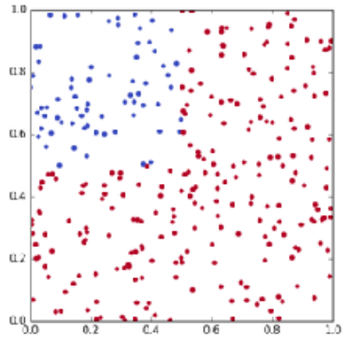


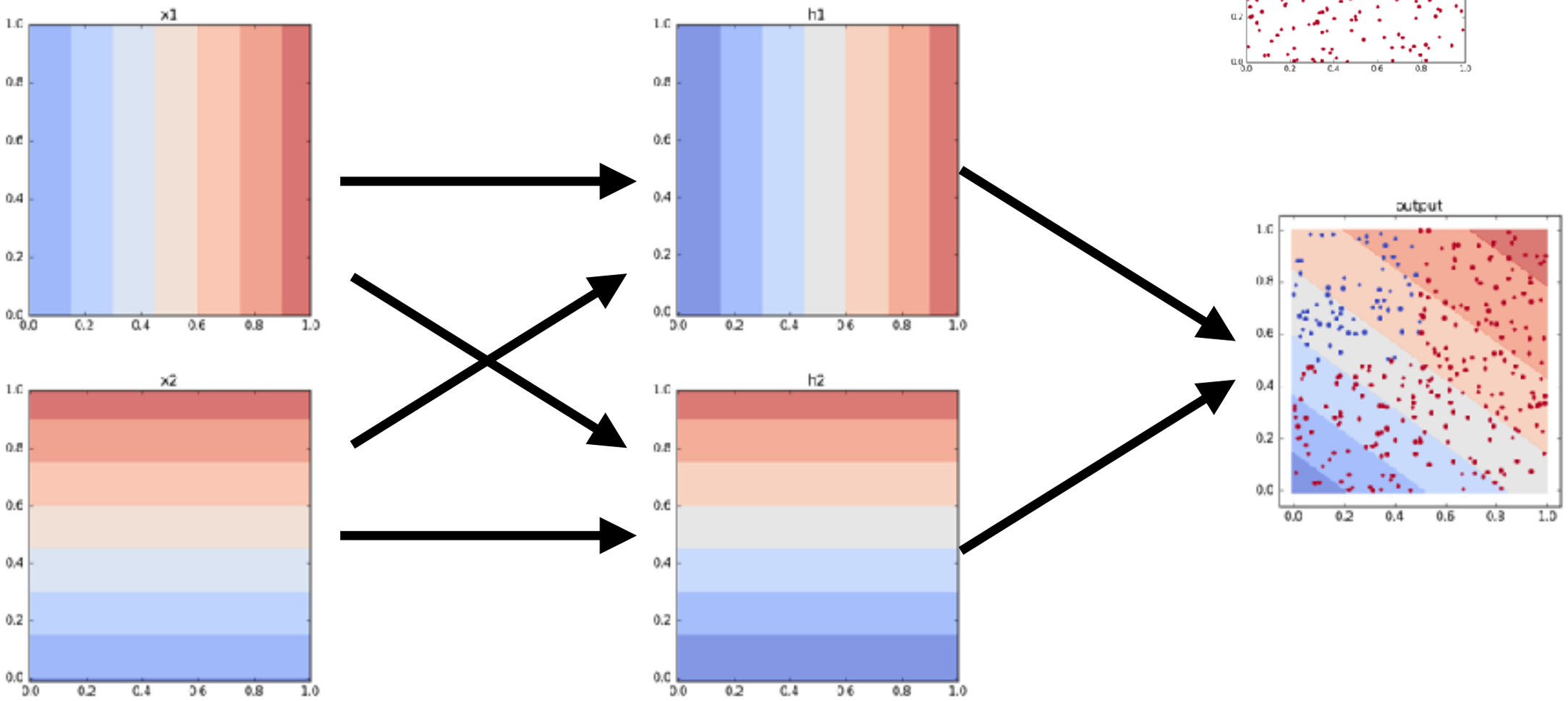
The Ring Data

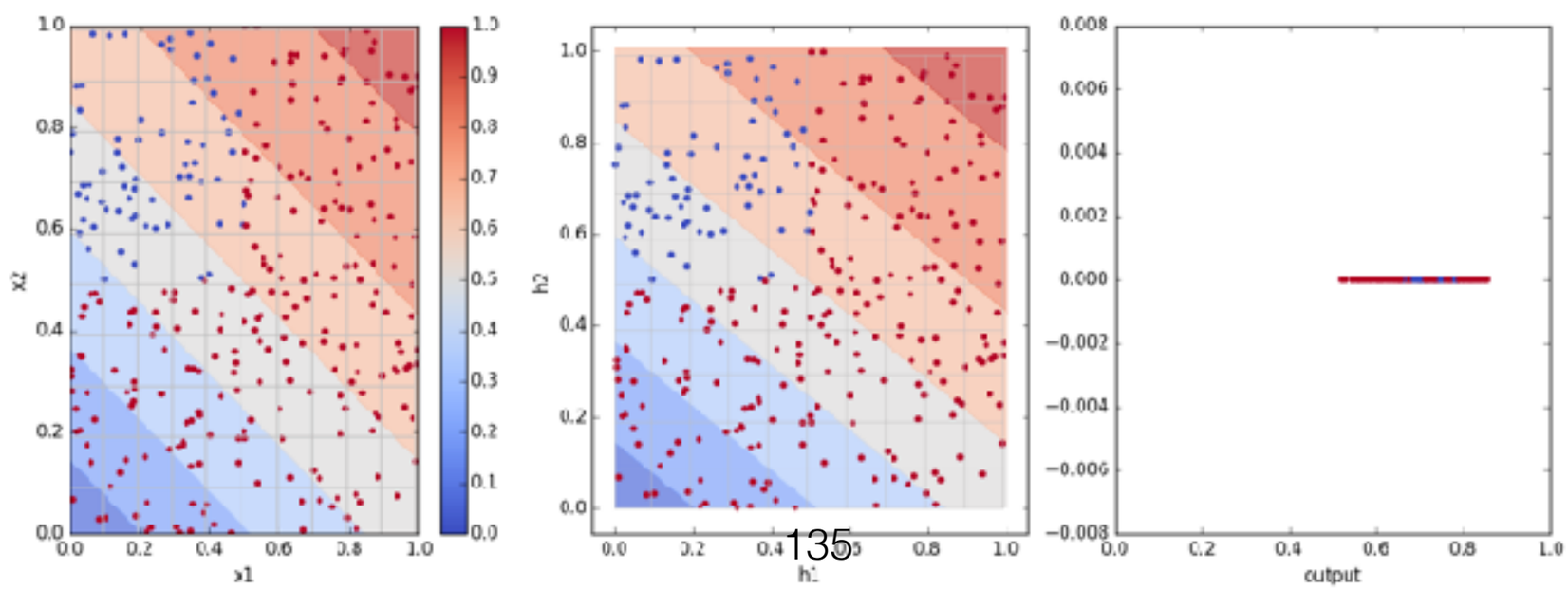




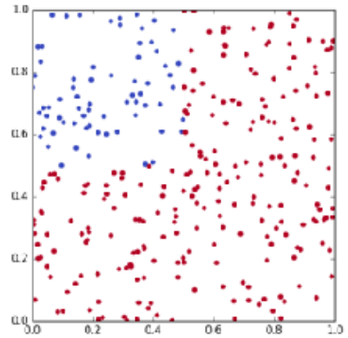
The Angle Data - ReLu

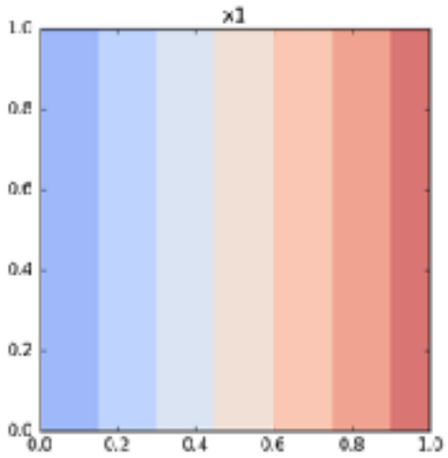


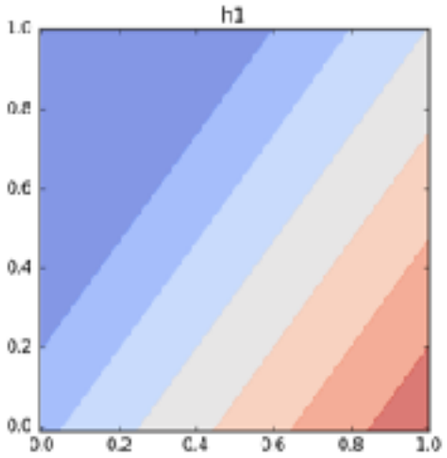


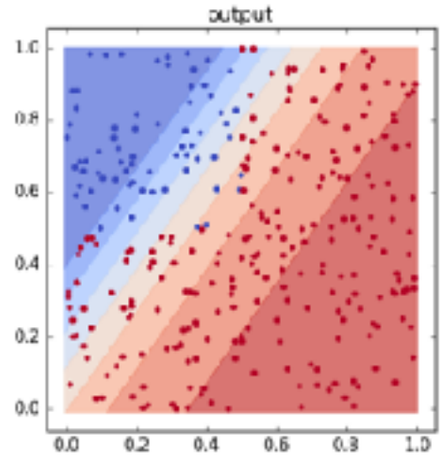


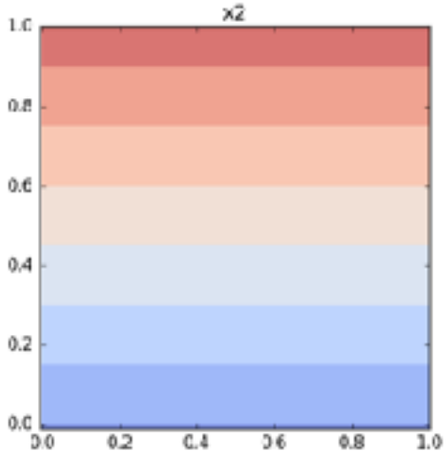
The Angle Data - ReLu

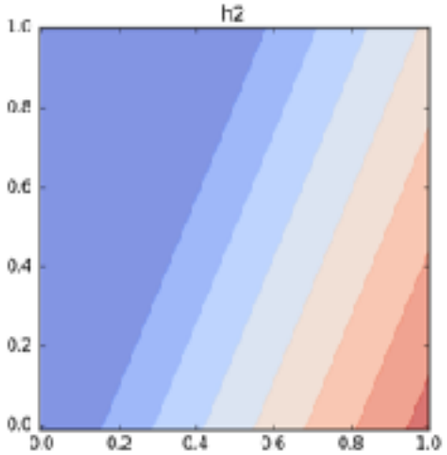


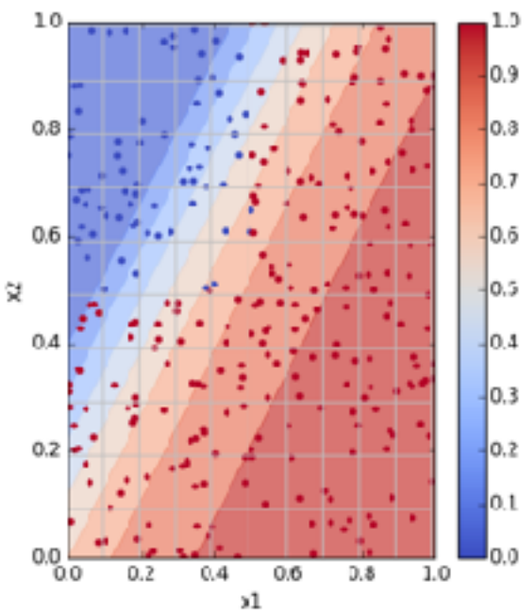


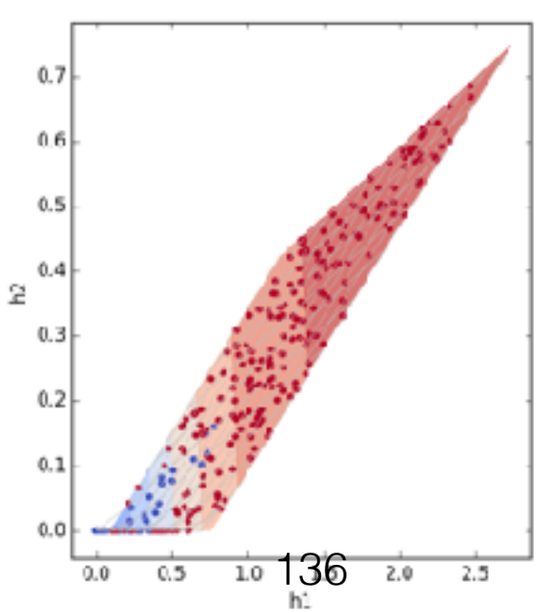


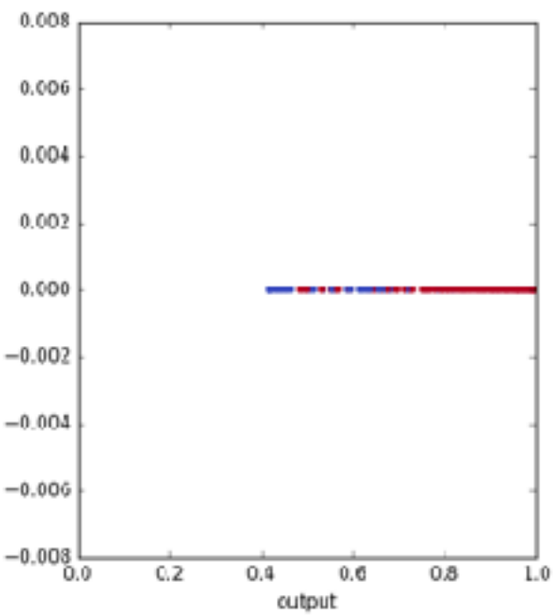




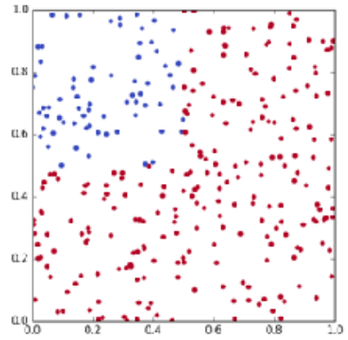


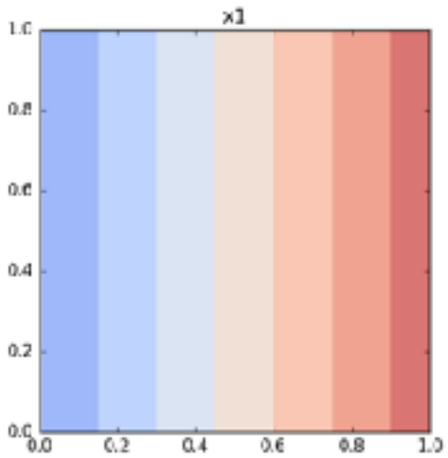


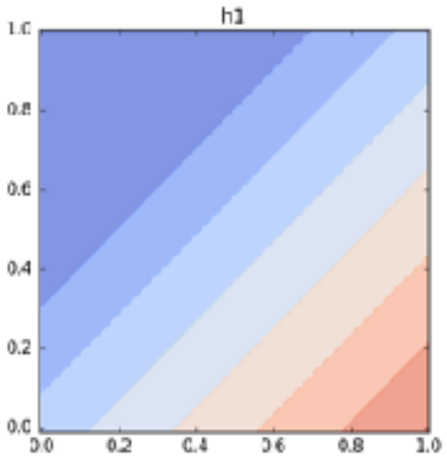


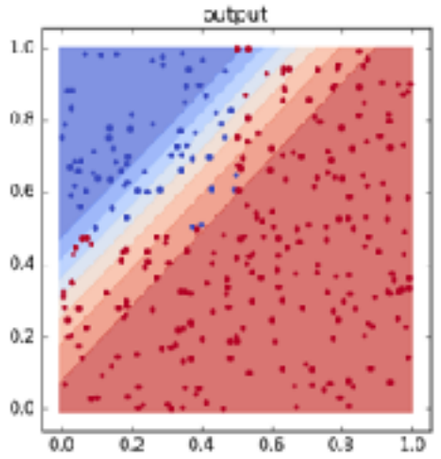


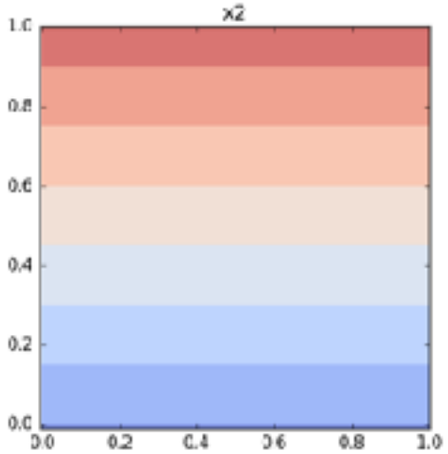
The Angle Data - ReLu

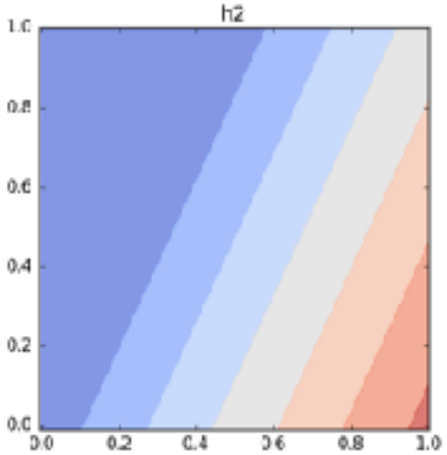


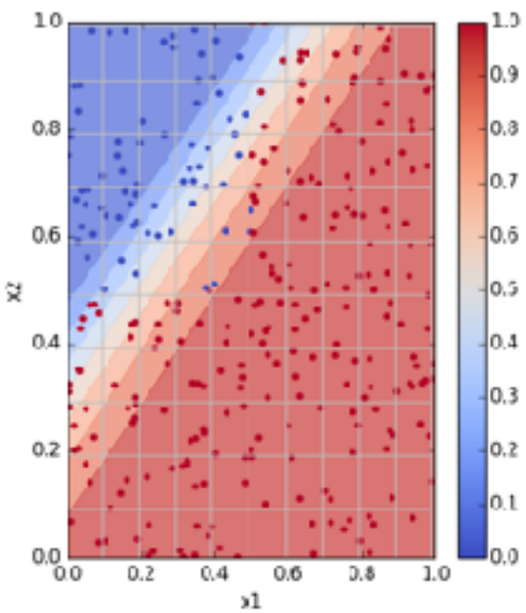


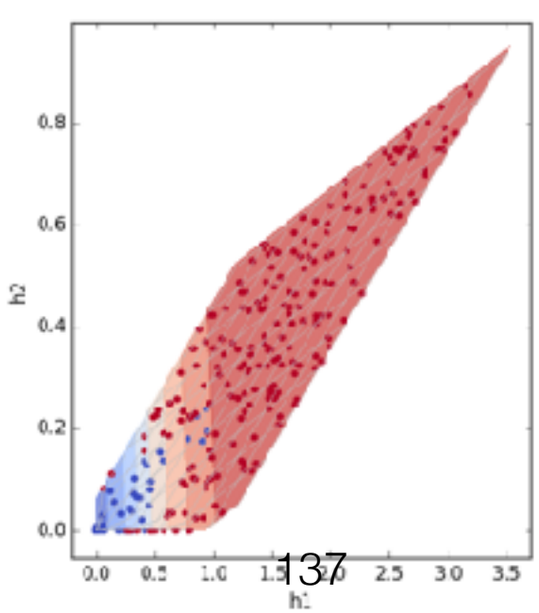


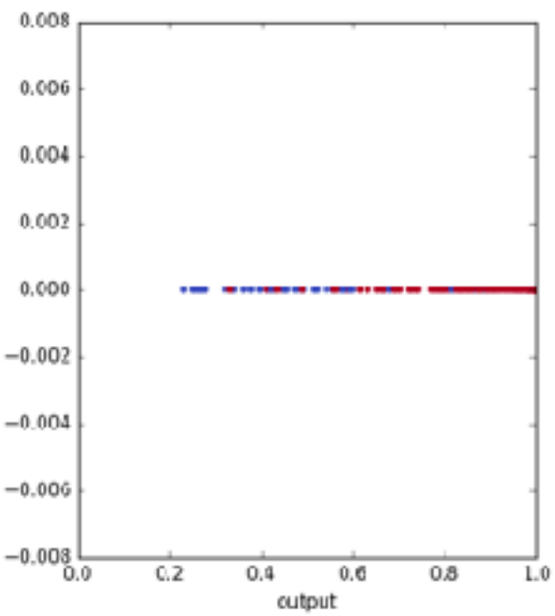




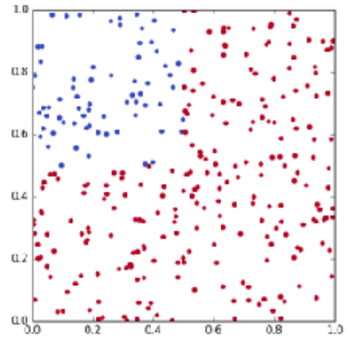


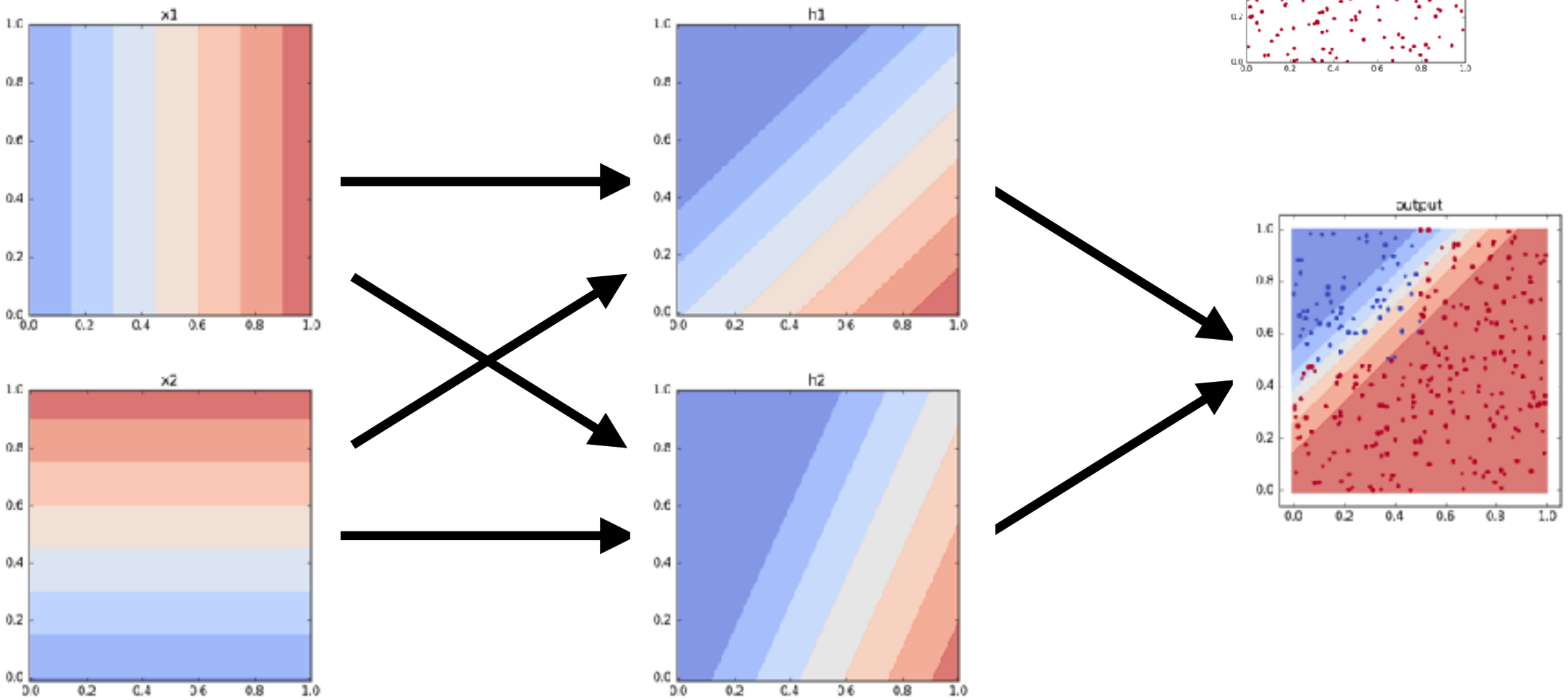


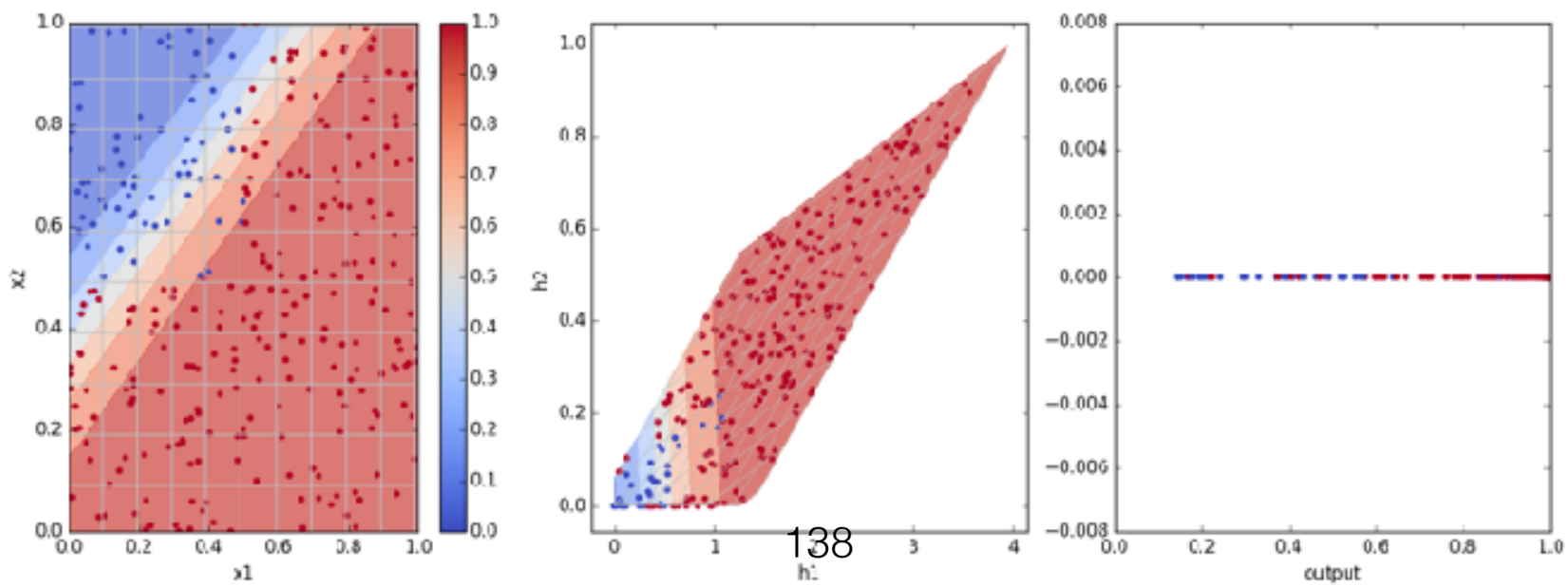




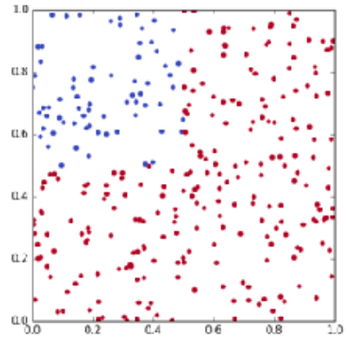
The Angle Data - ReLu

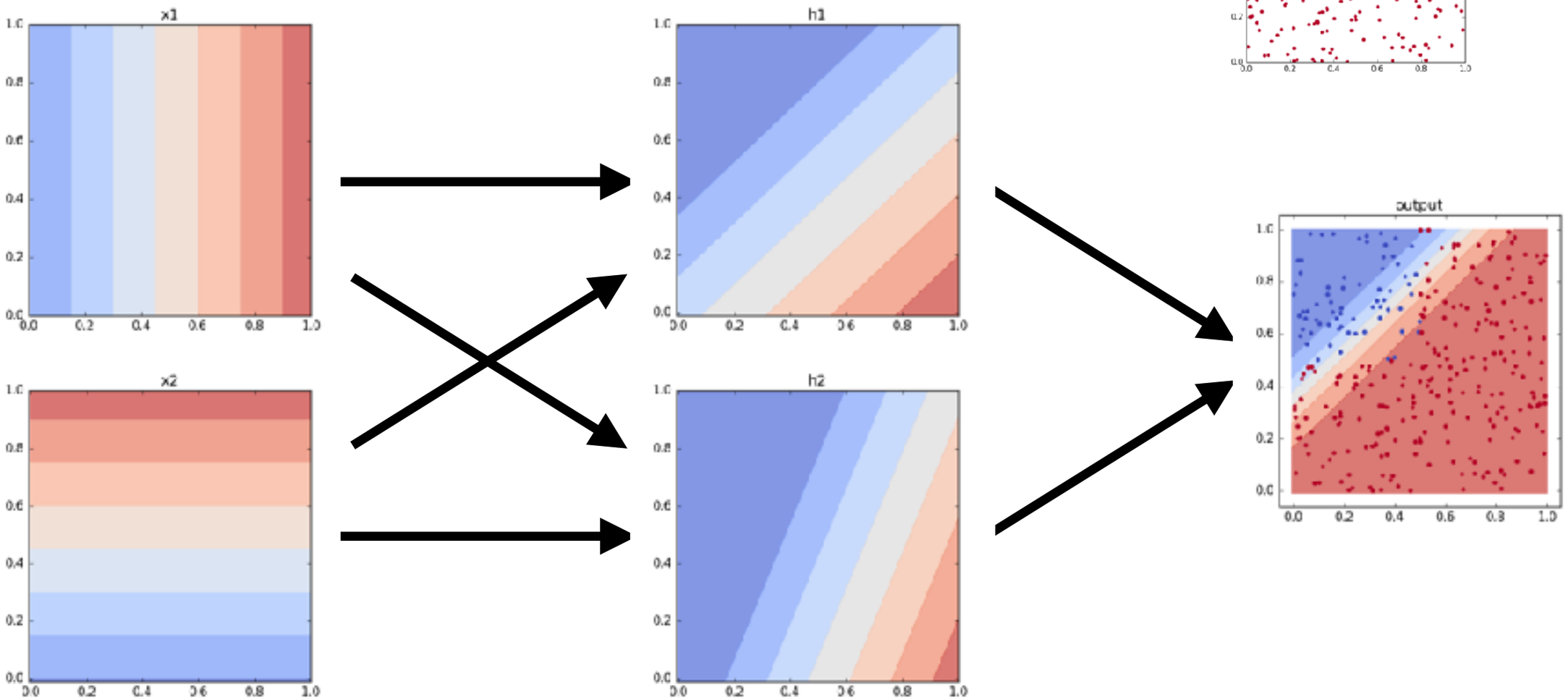


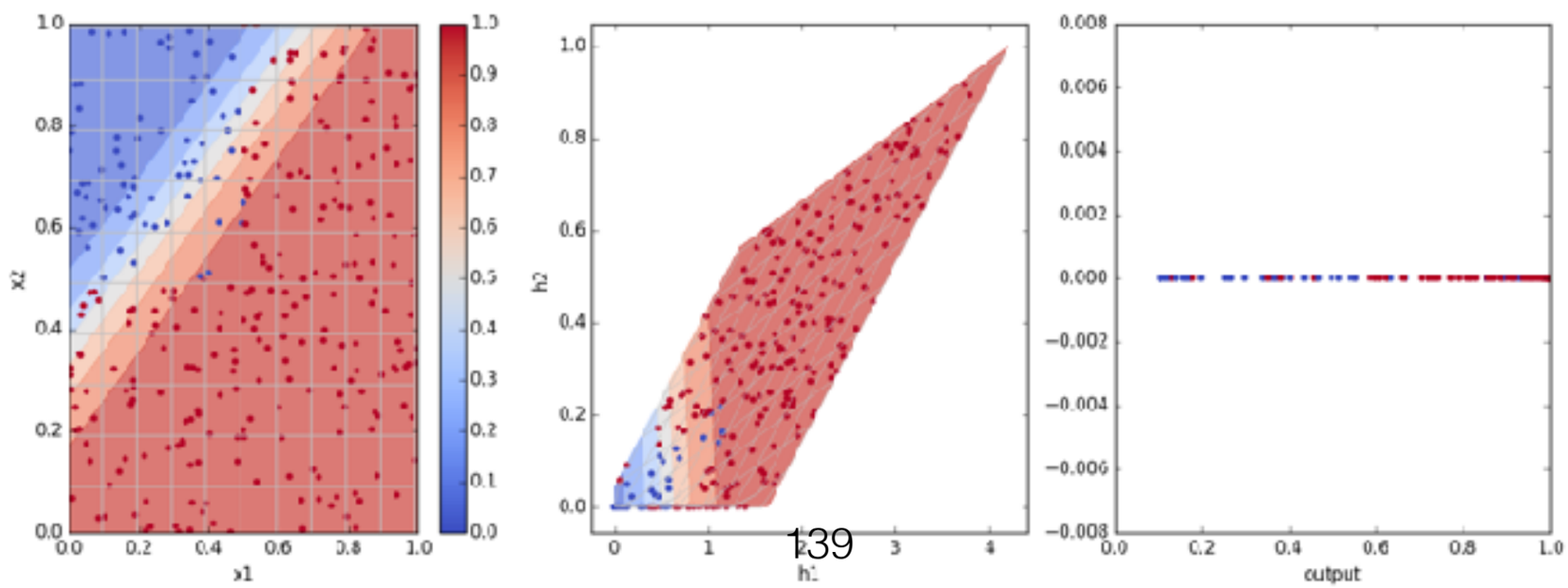




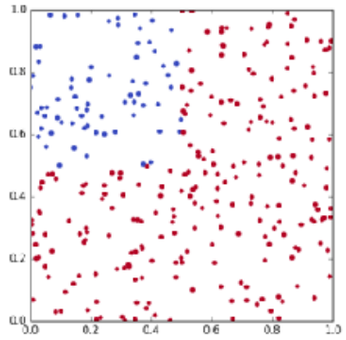
The Angle Data - ReLu

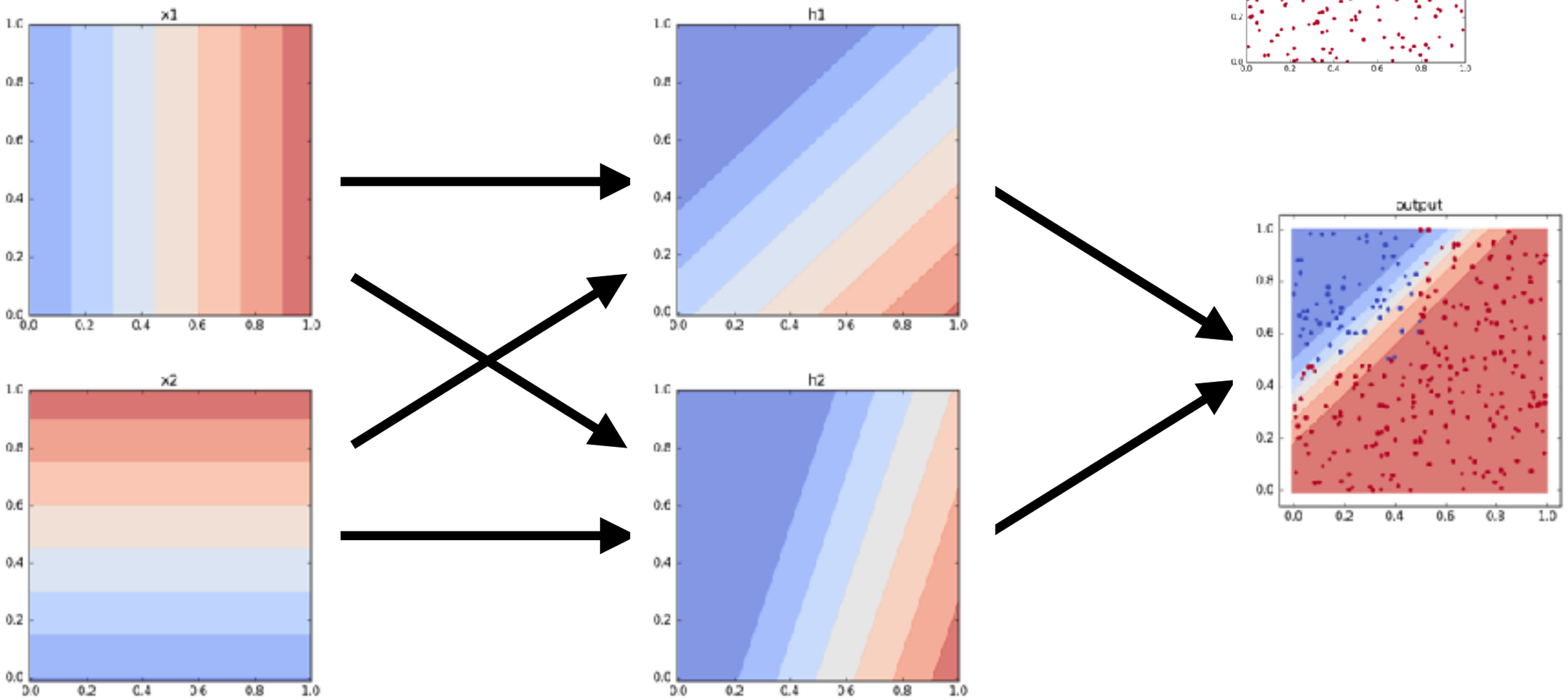


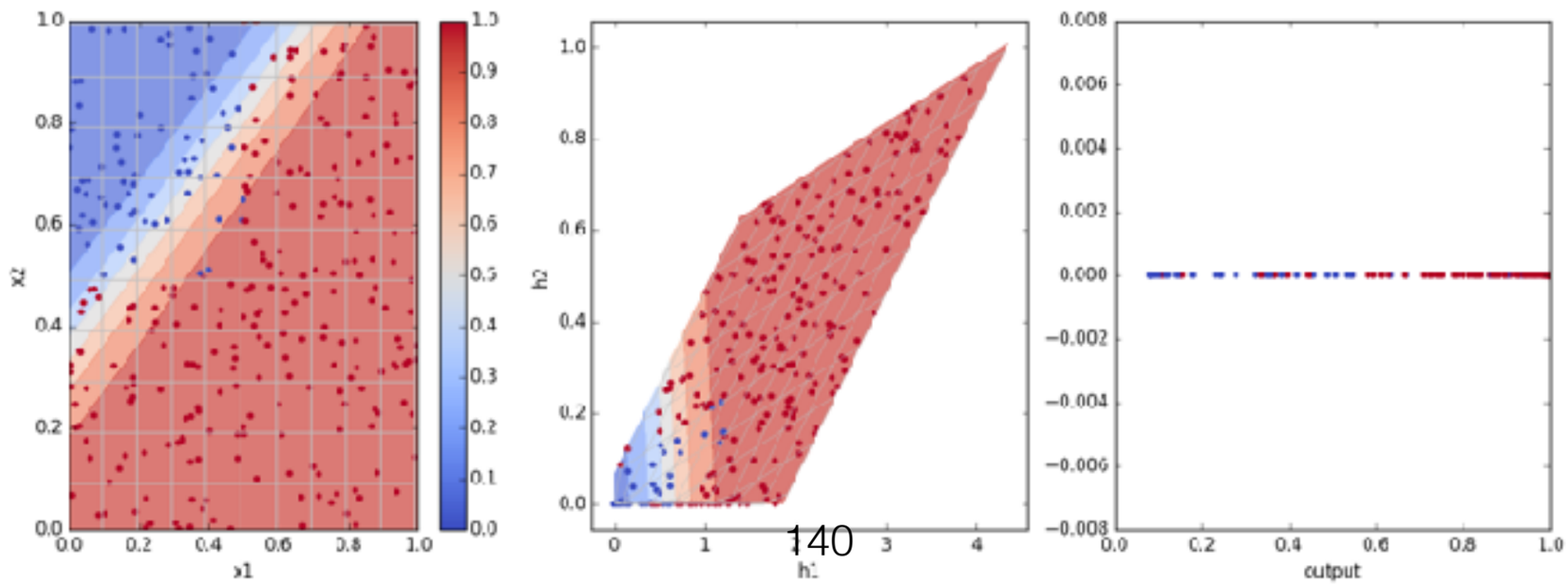




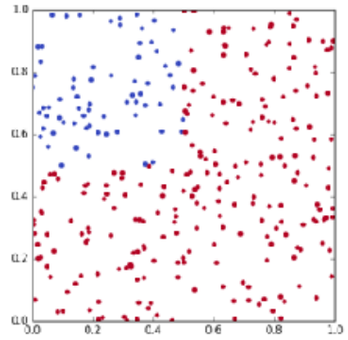
The Angle Data - ReLu

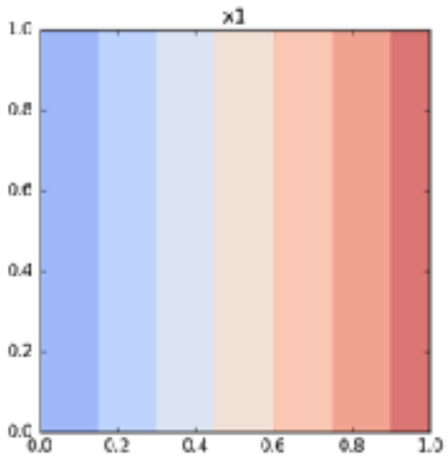


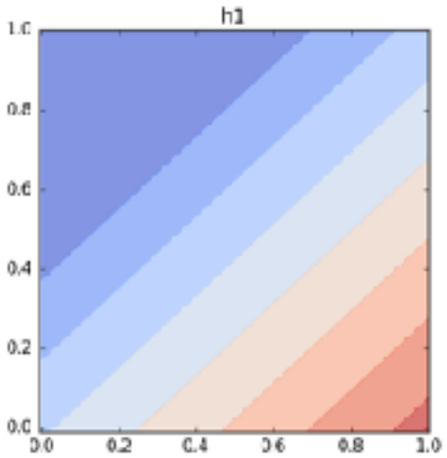


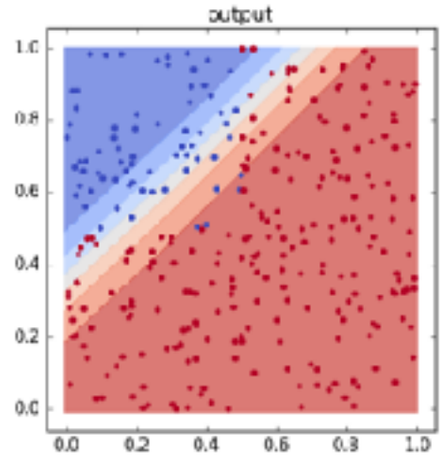


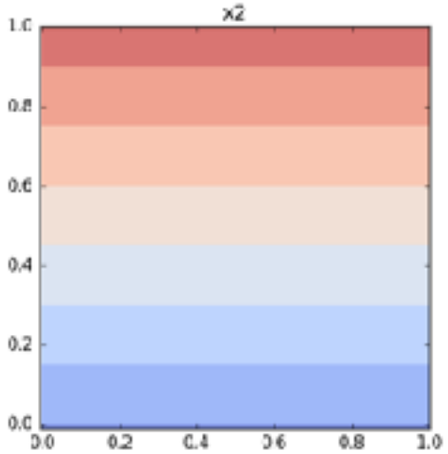
The Angle Data - ReLu

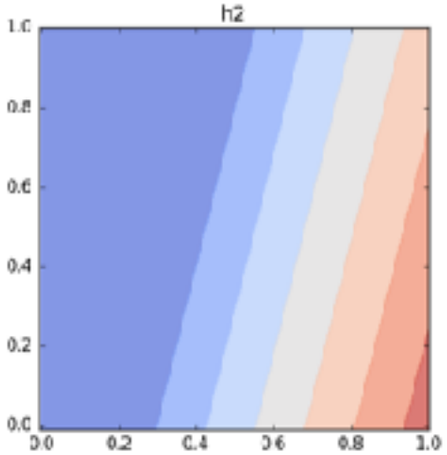


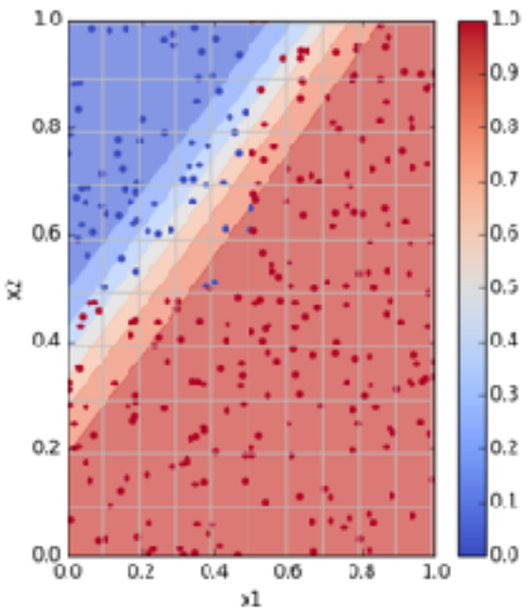


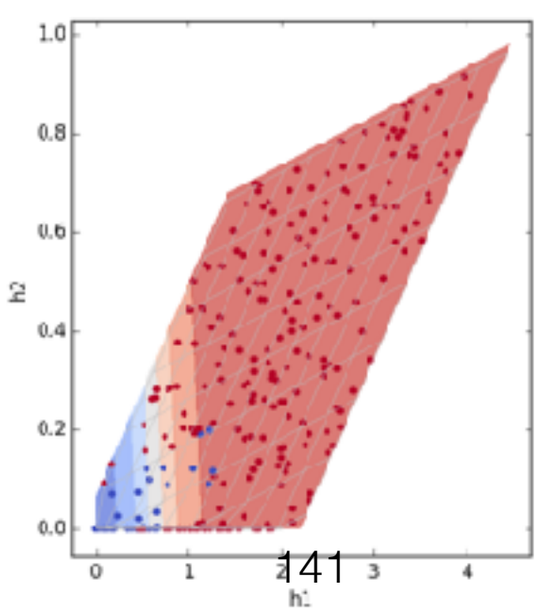


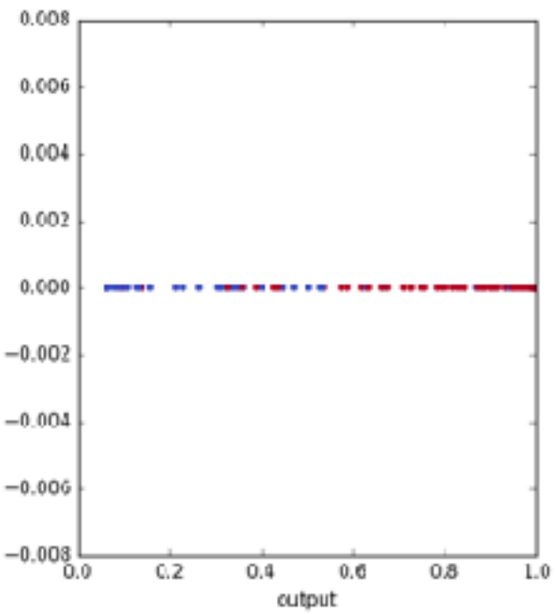




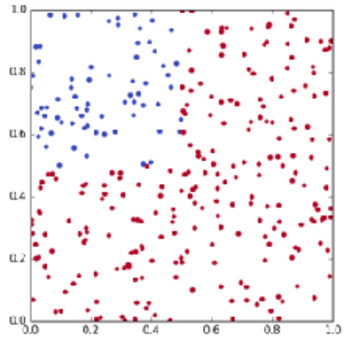


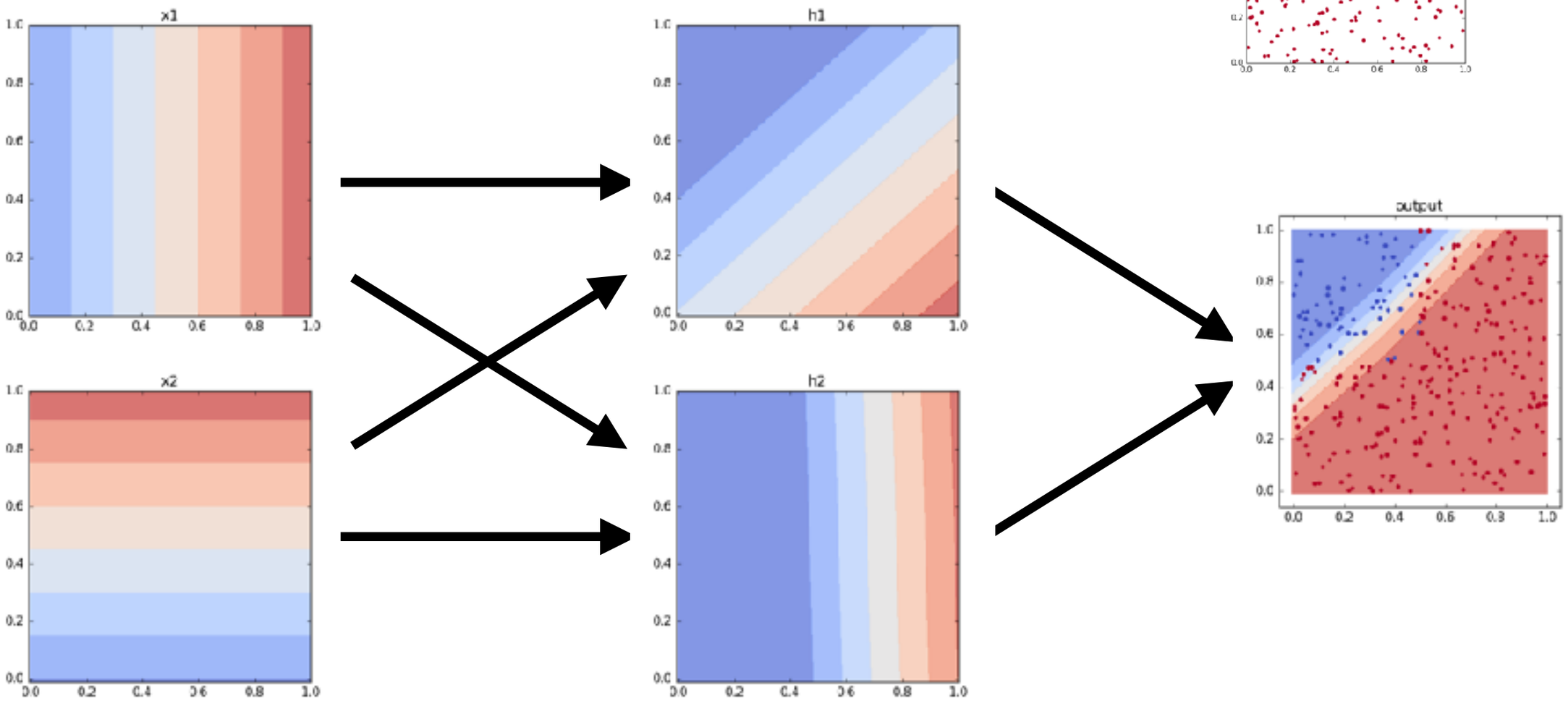


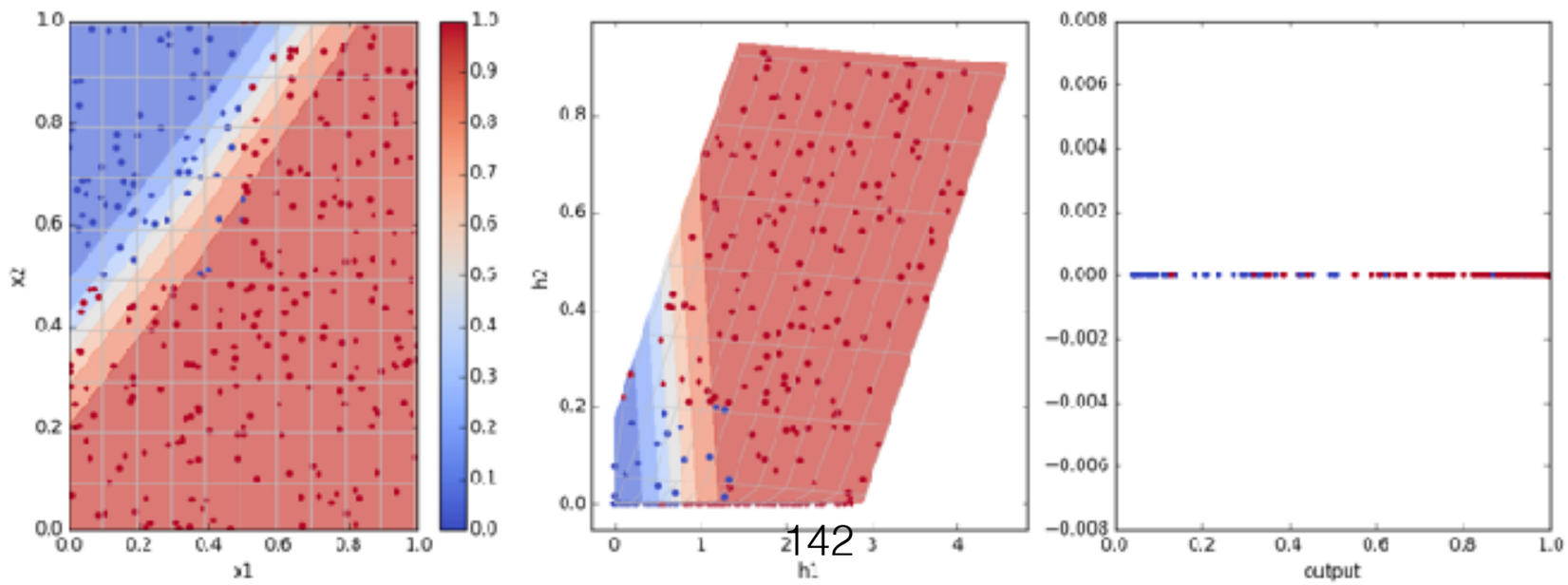




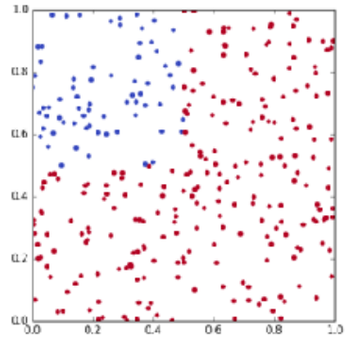
The Angle Data - ReLu

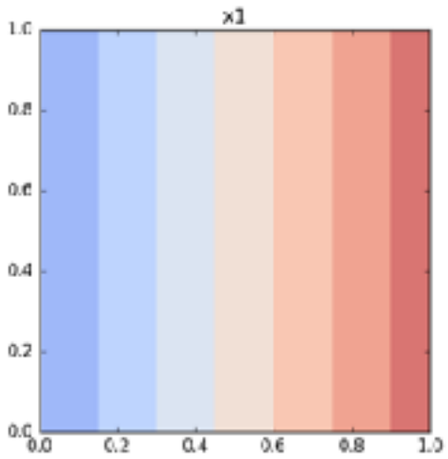


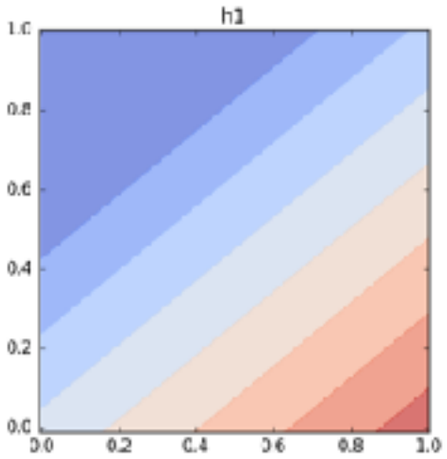


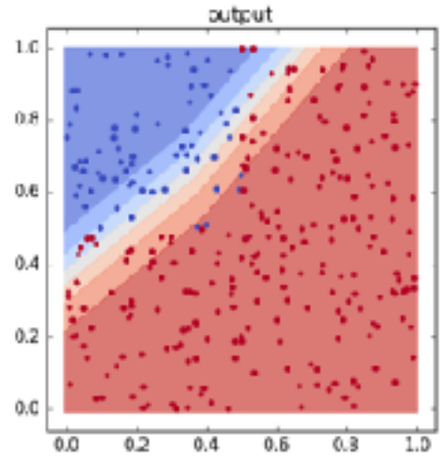


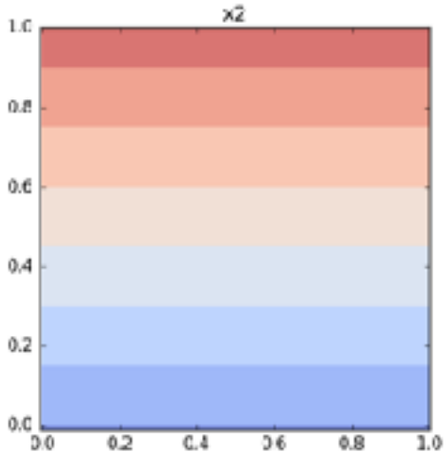
The Angle Data - ReLu

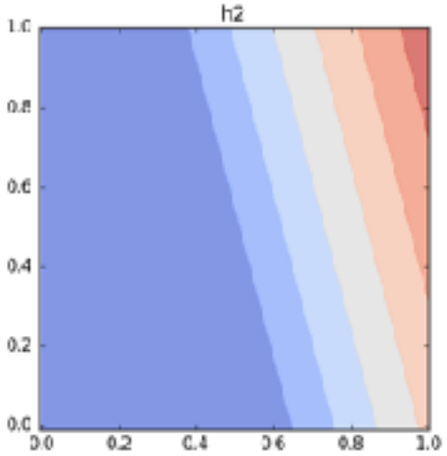


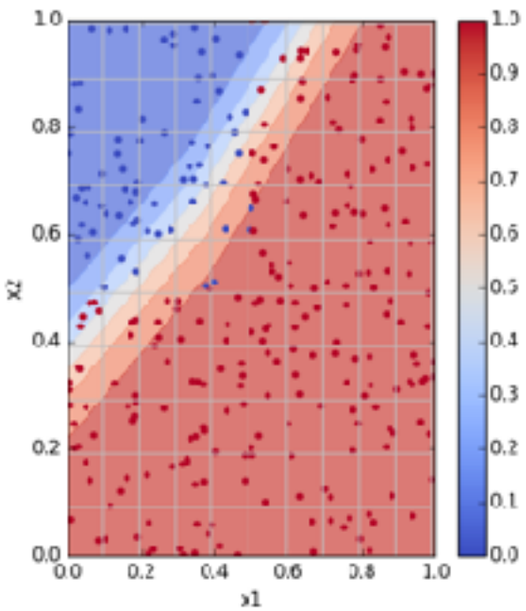


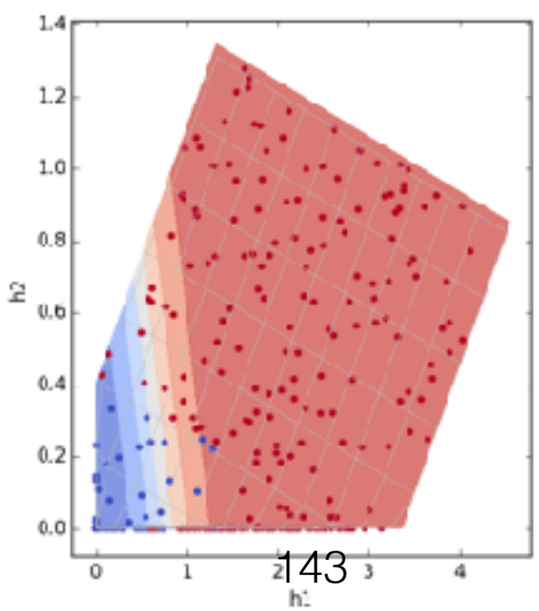


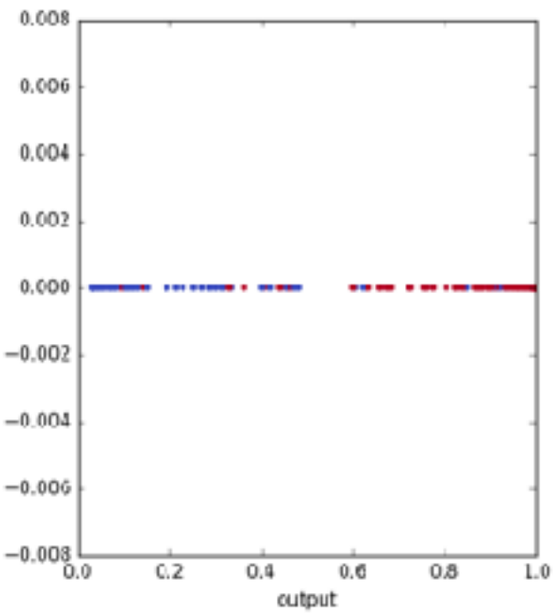




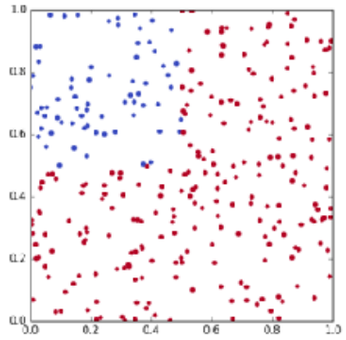


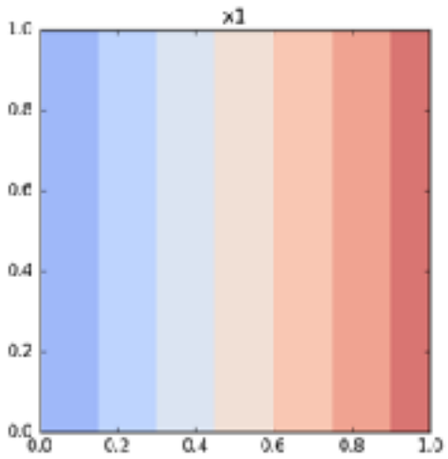


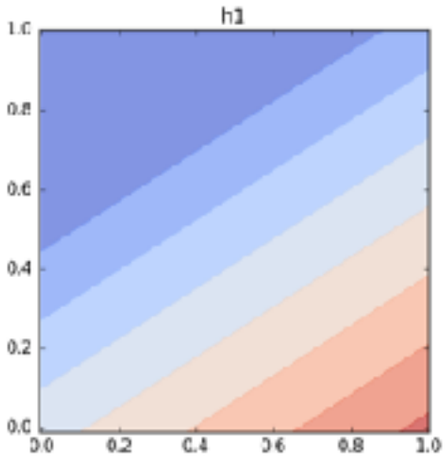


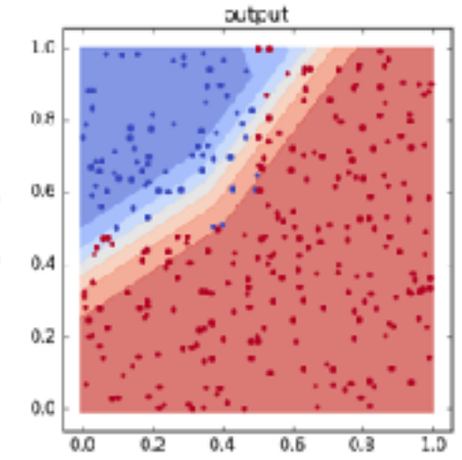


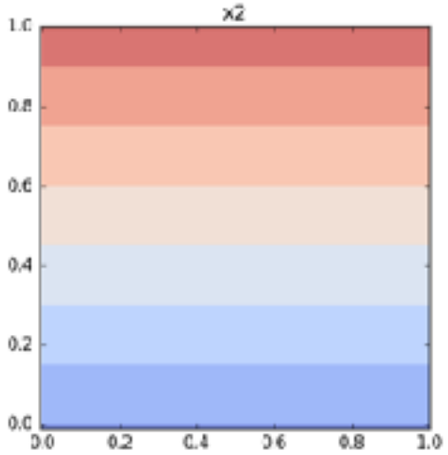
The Angle Data - ReLu

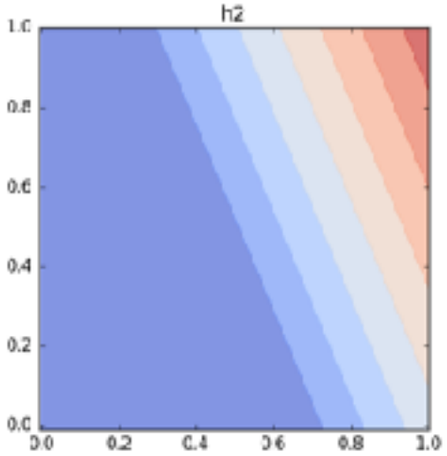


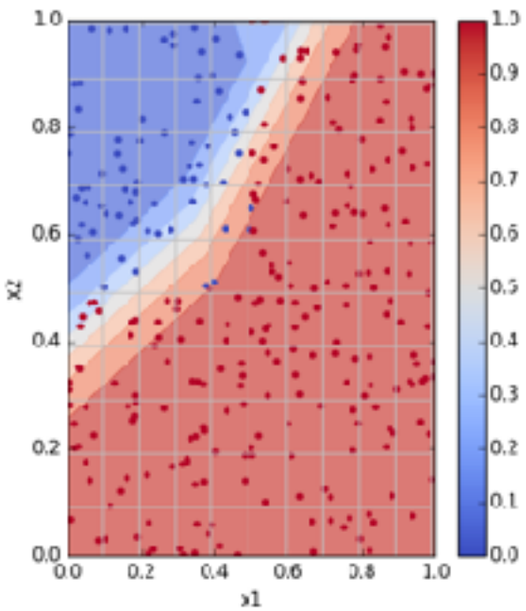


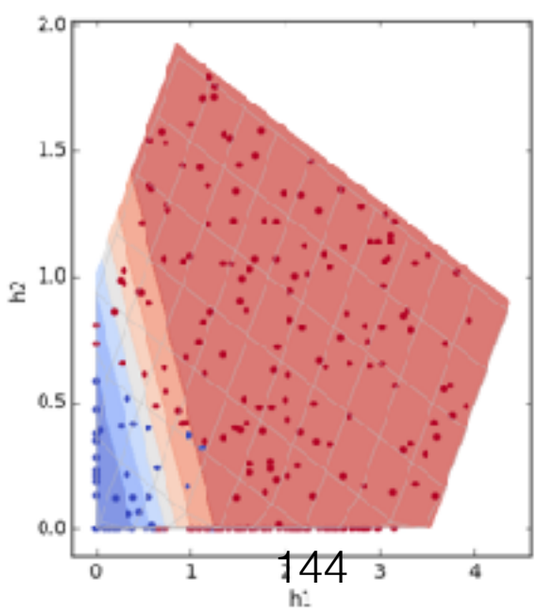


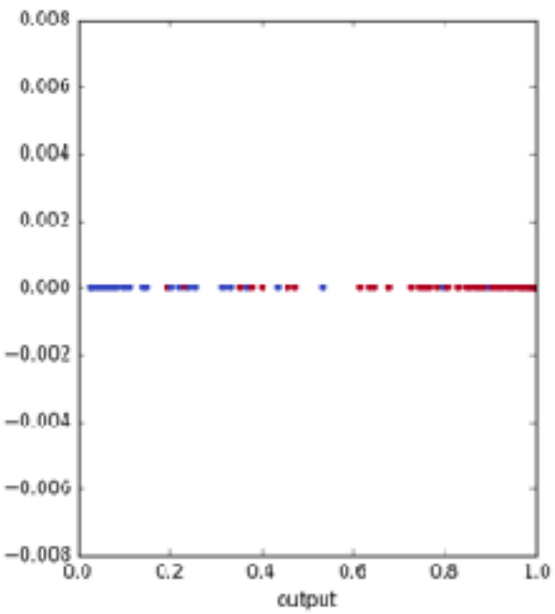




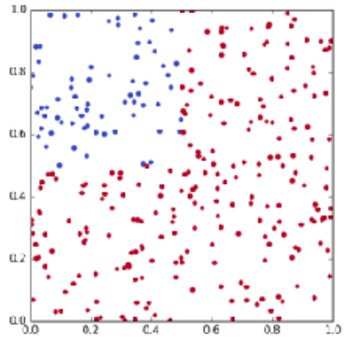


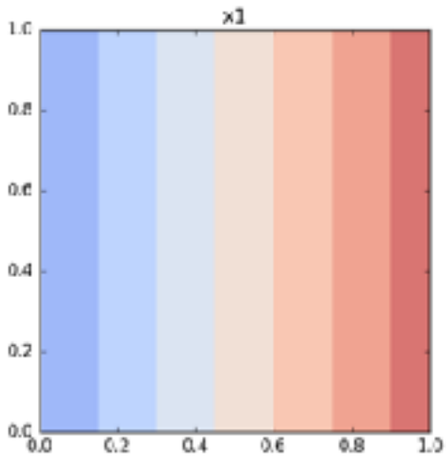


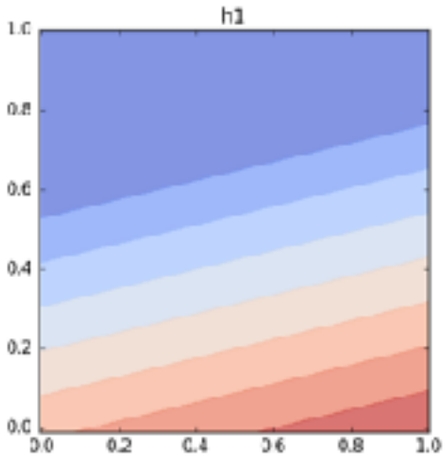


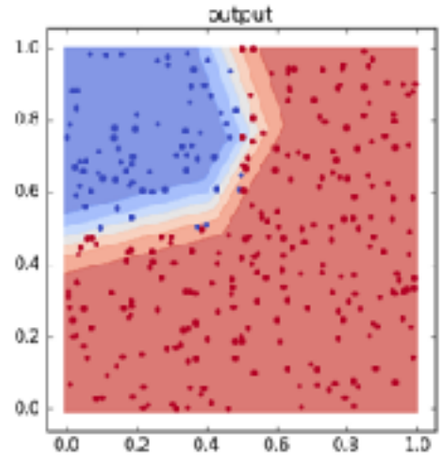


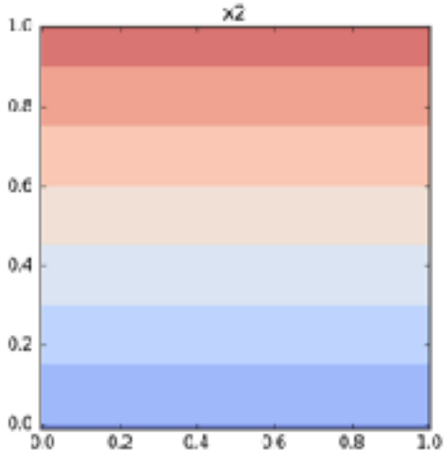
The Angle Data - ReLu

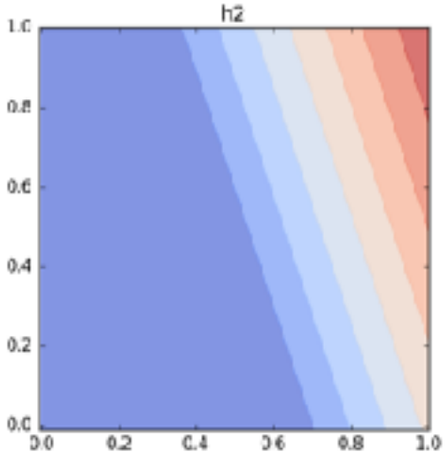


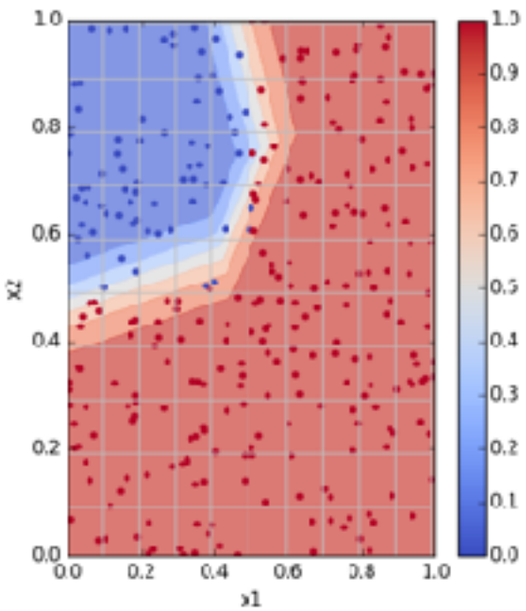


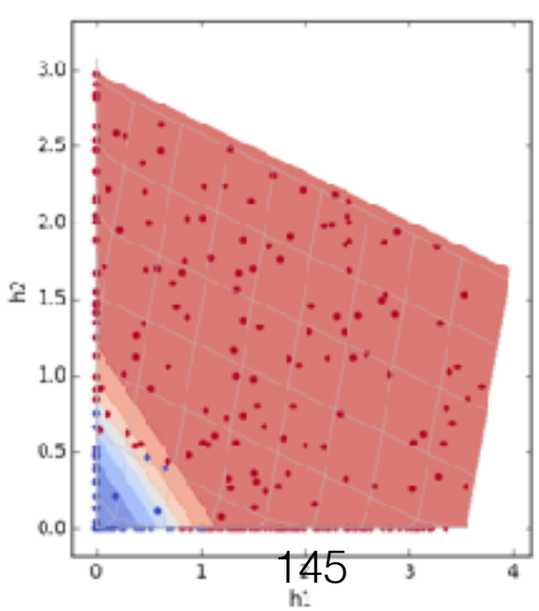


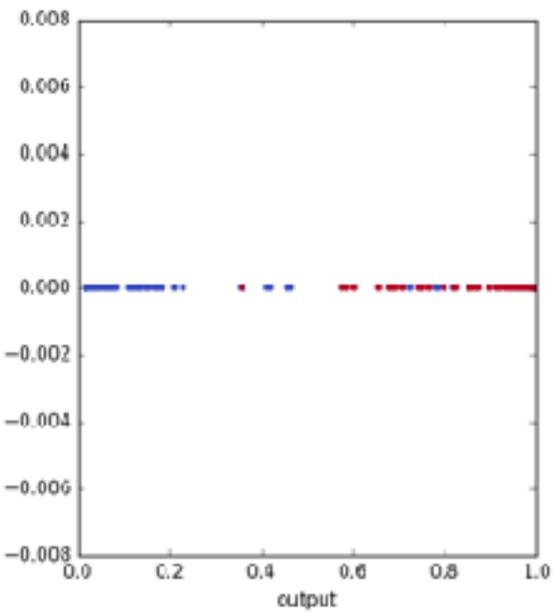






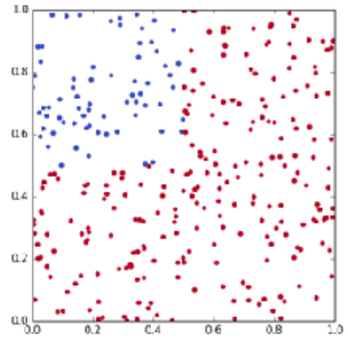


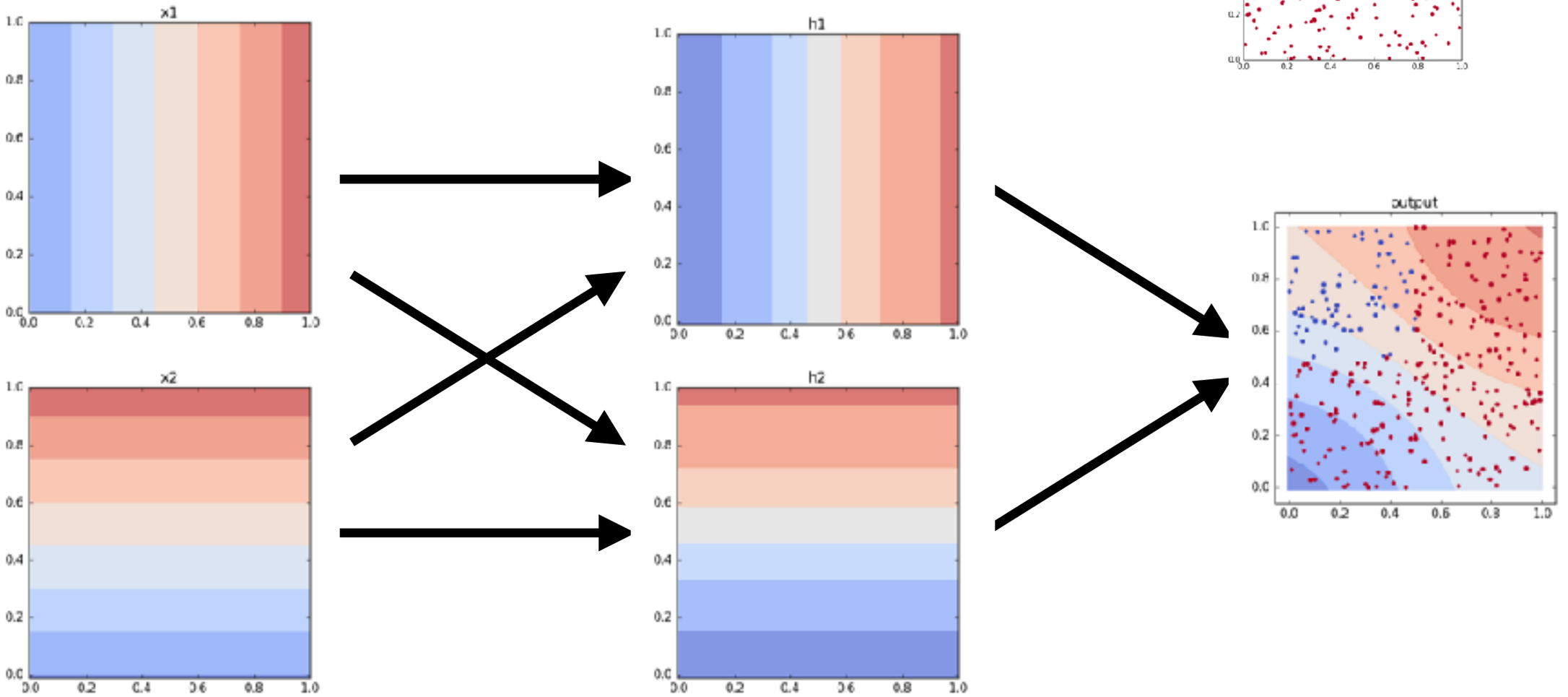


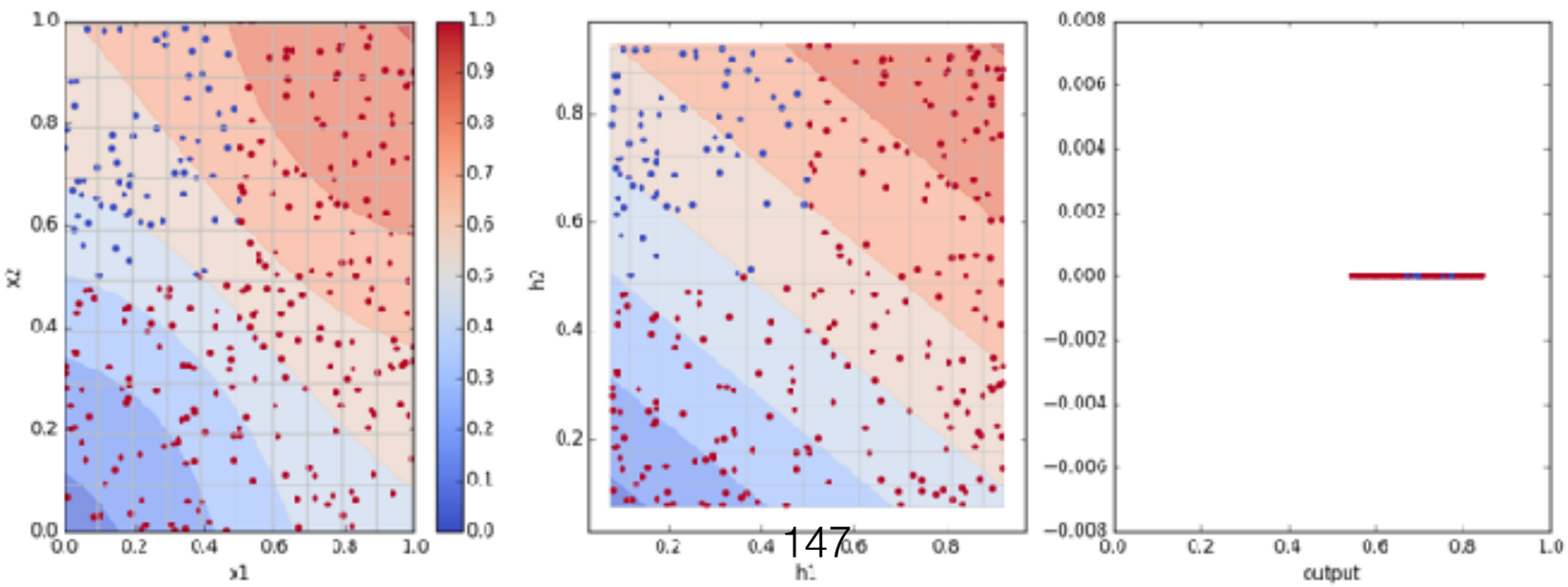


The Angle Data - Sigmoid

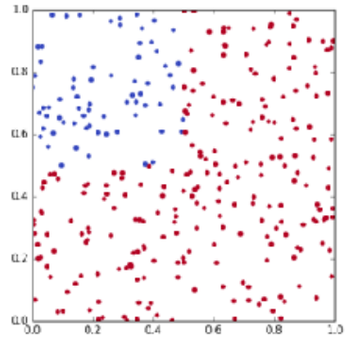
The Angle Data - Sigmoid

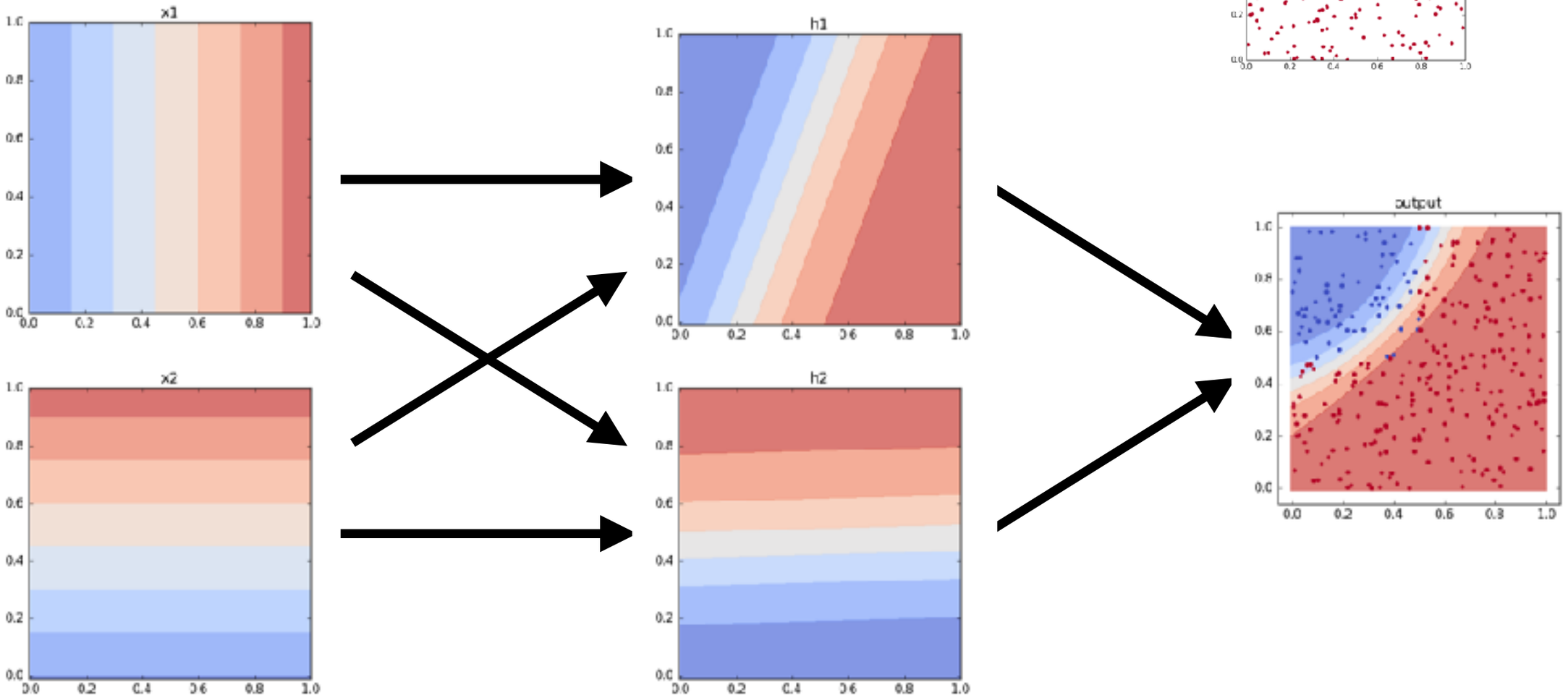


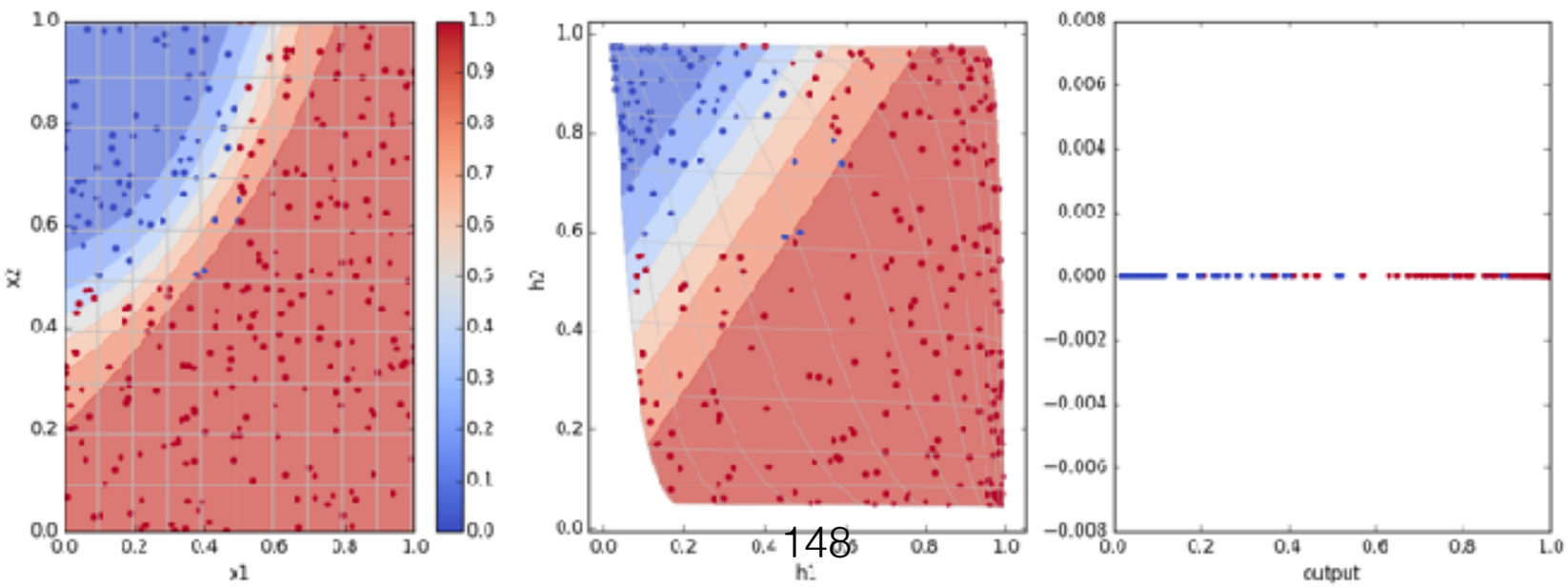




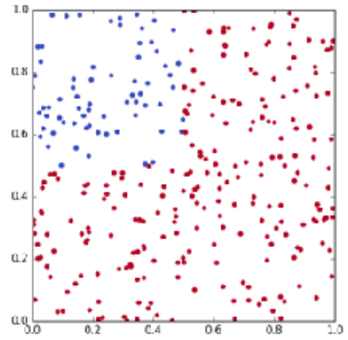
The Angle Data - Sigmoid

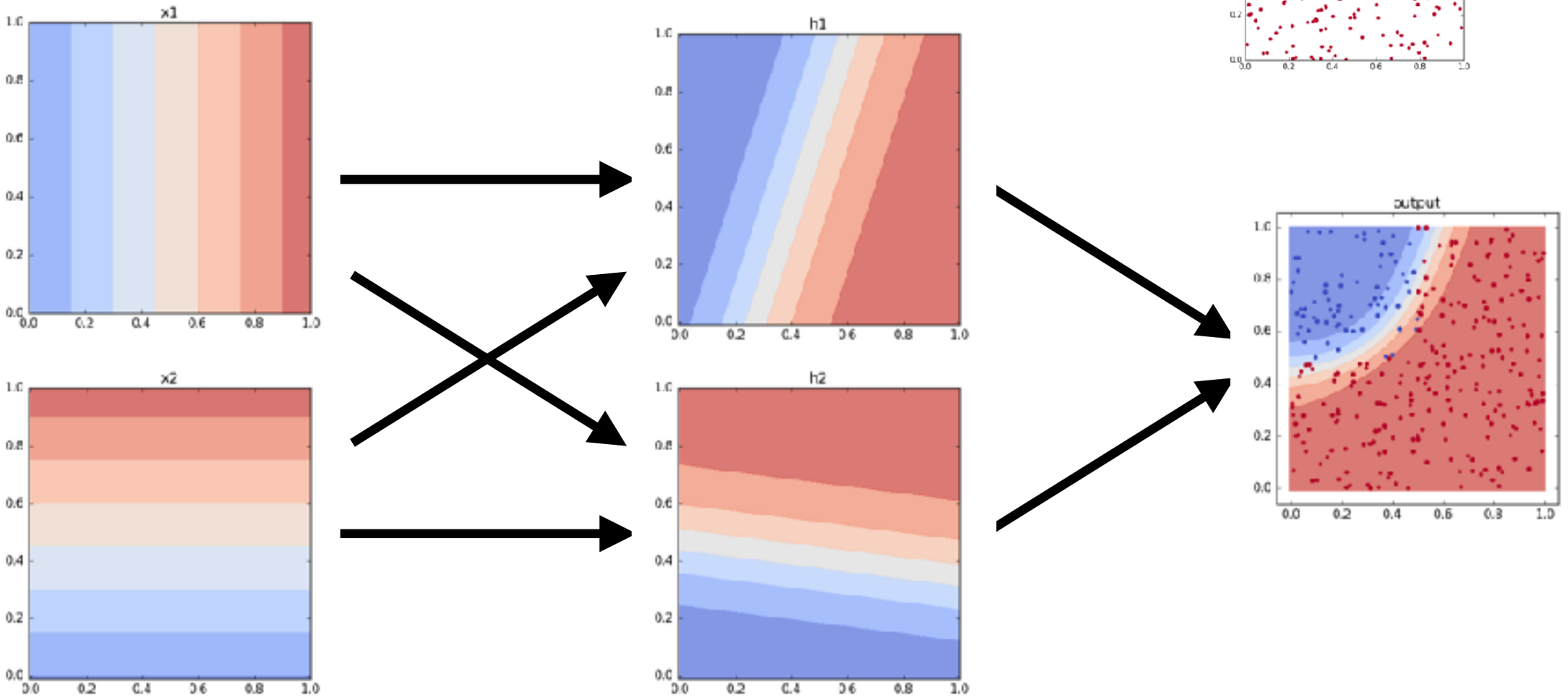


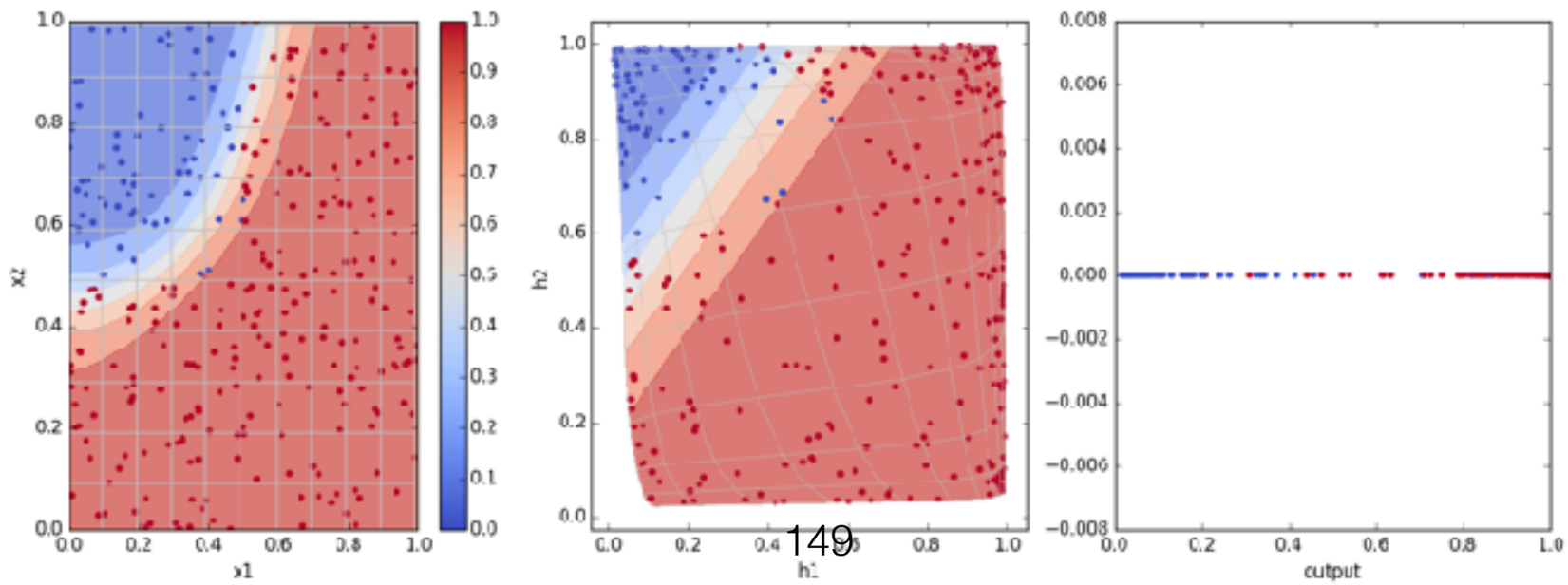




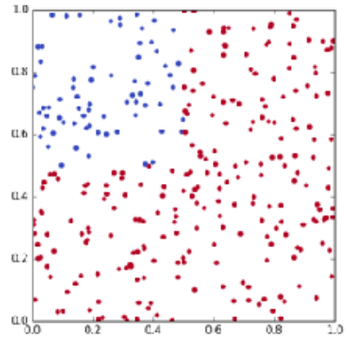
The Angle Data - Sigmoid

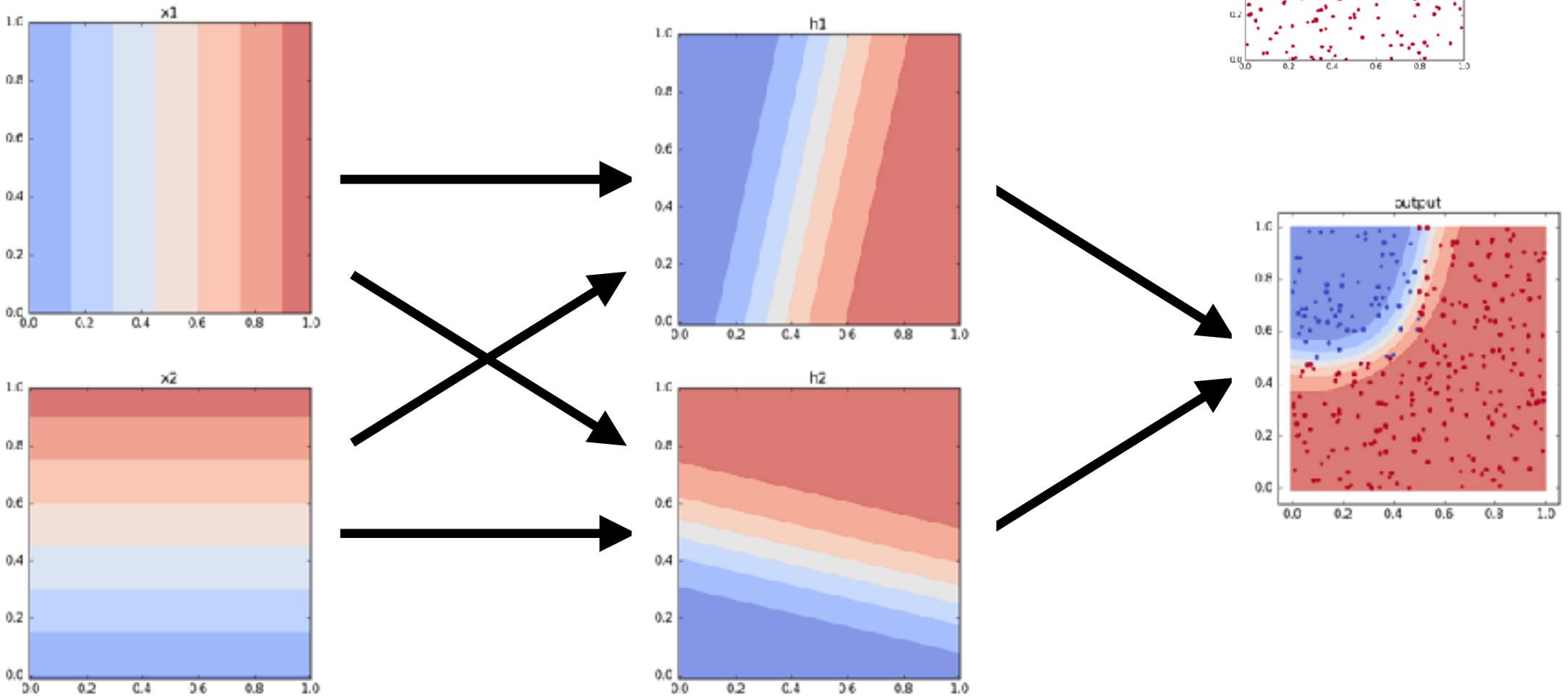


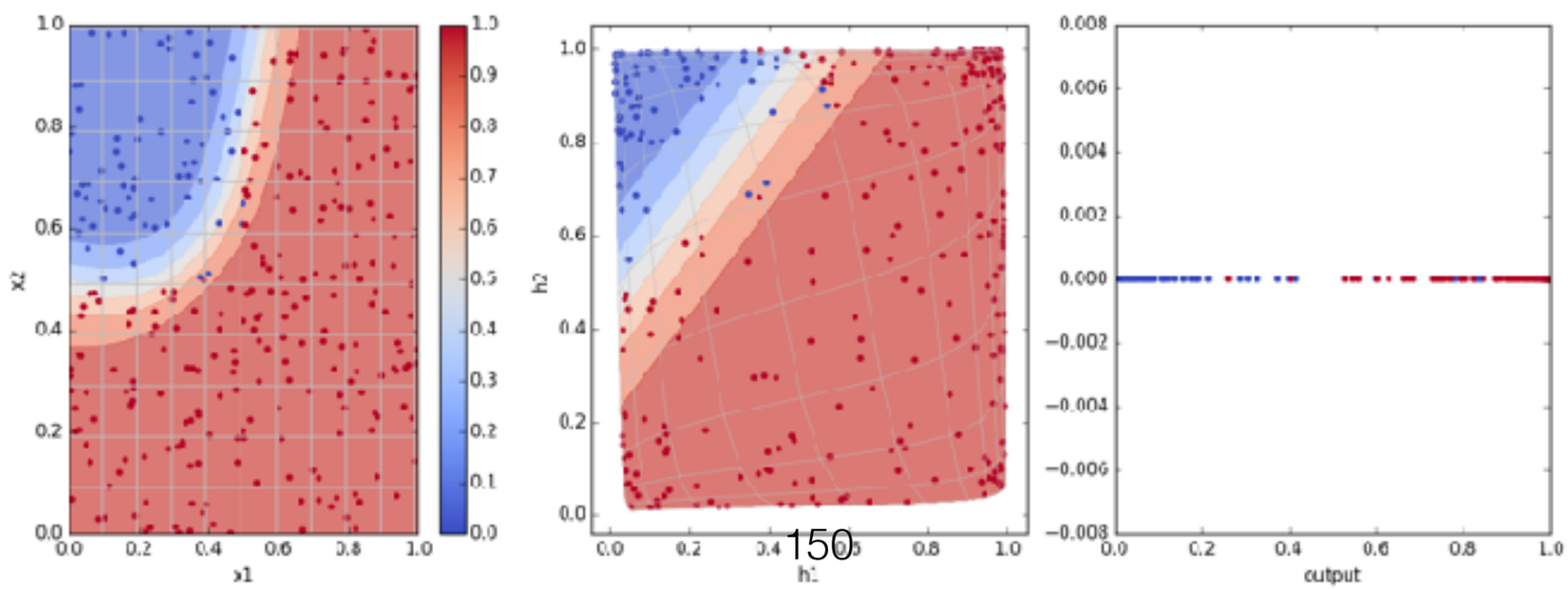




The Angle Data - Sigmoid

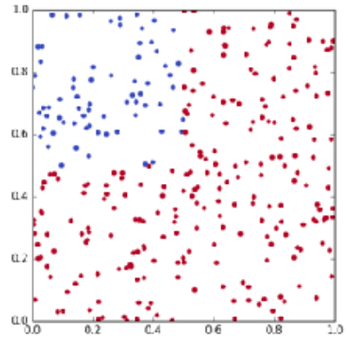


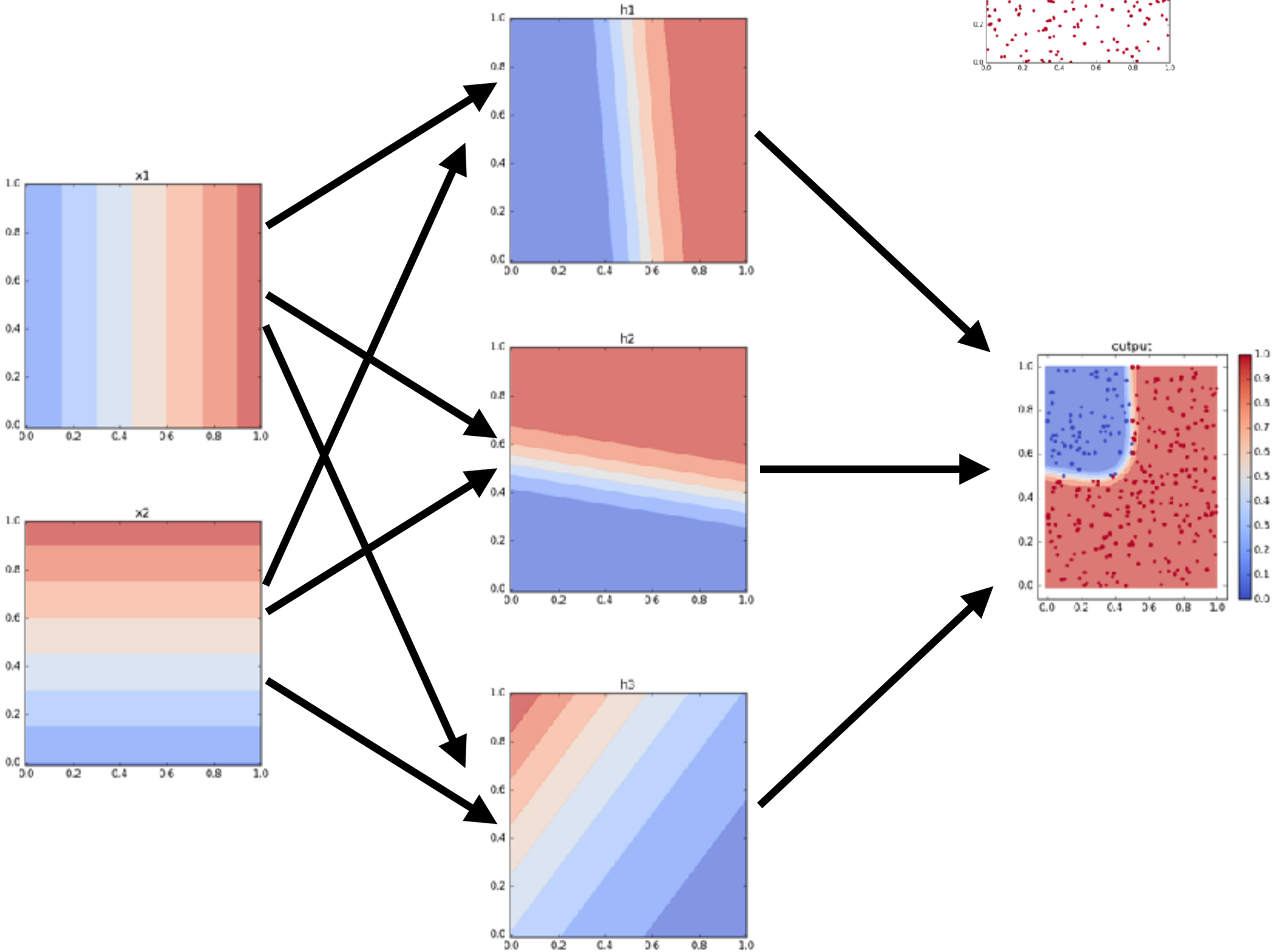




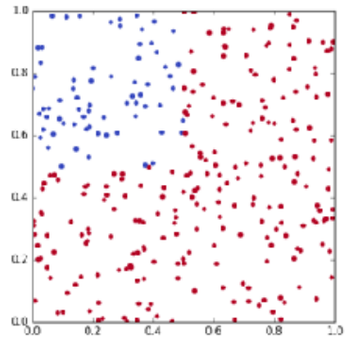
The Angle Data - Sigmoid 3 hidden nodes

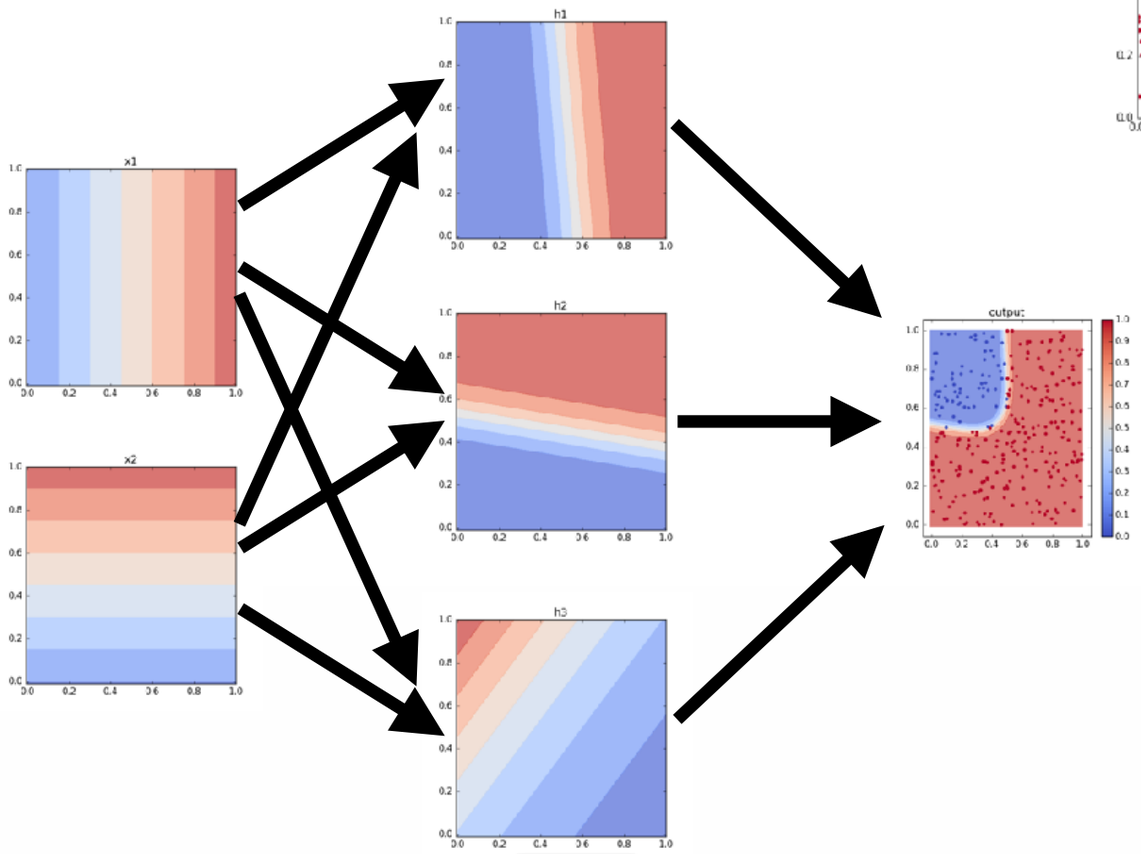
The Angle Data - Sigmoid

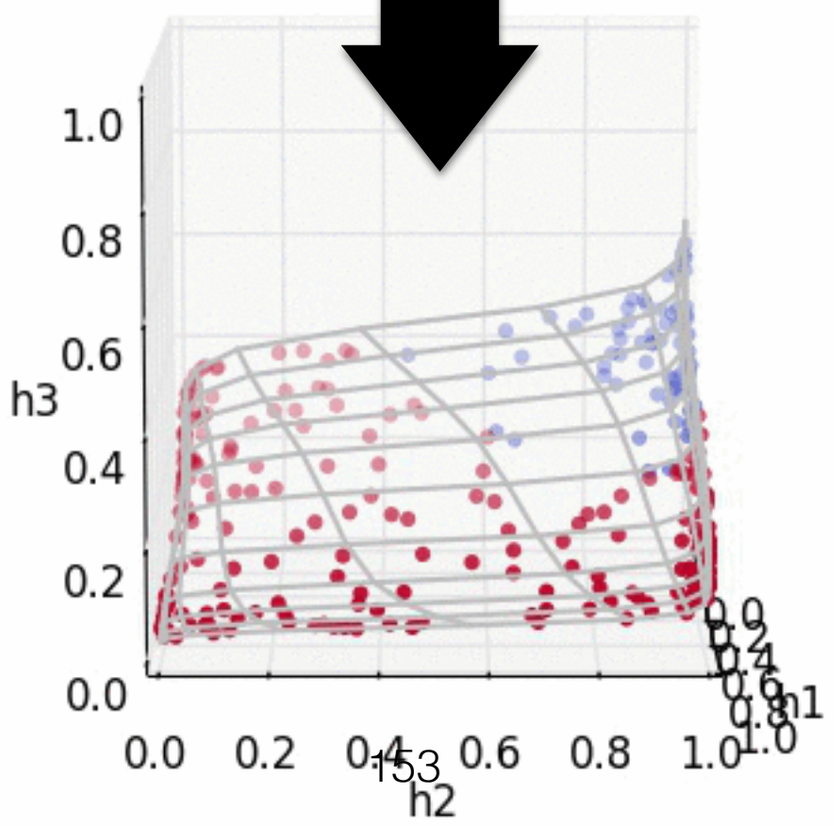




The Angle Data - Sigmoid

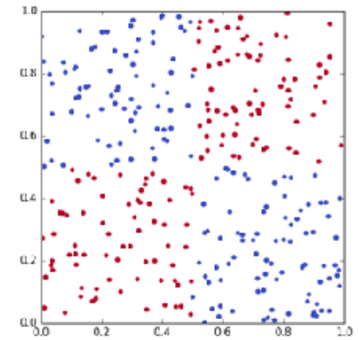


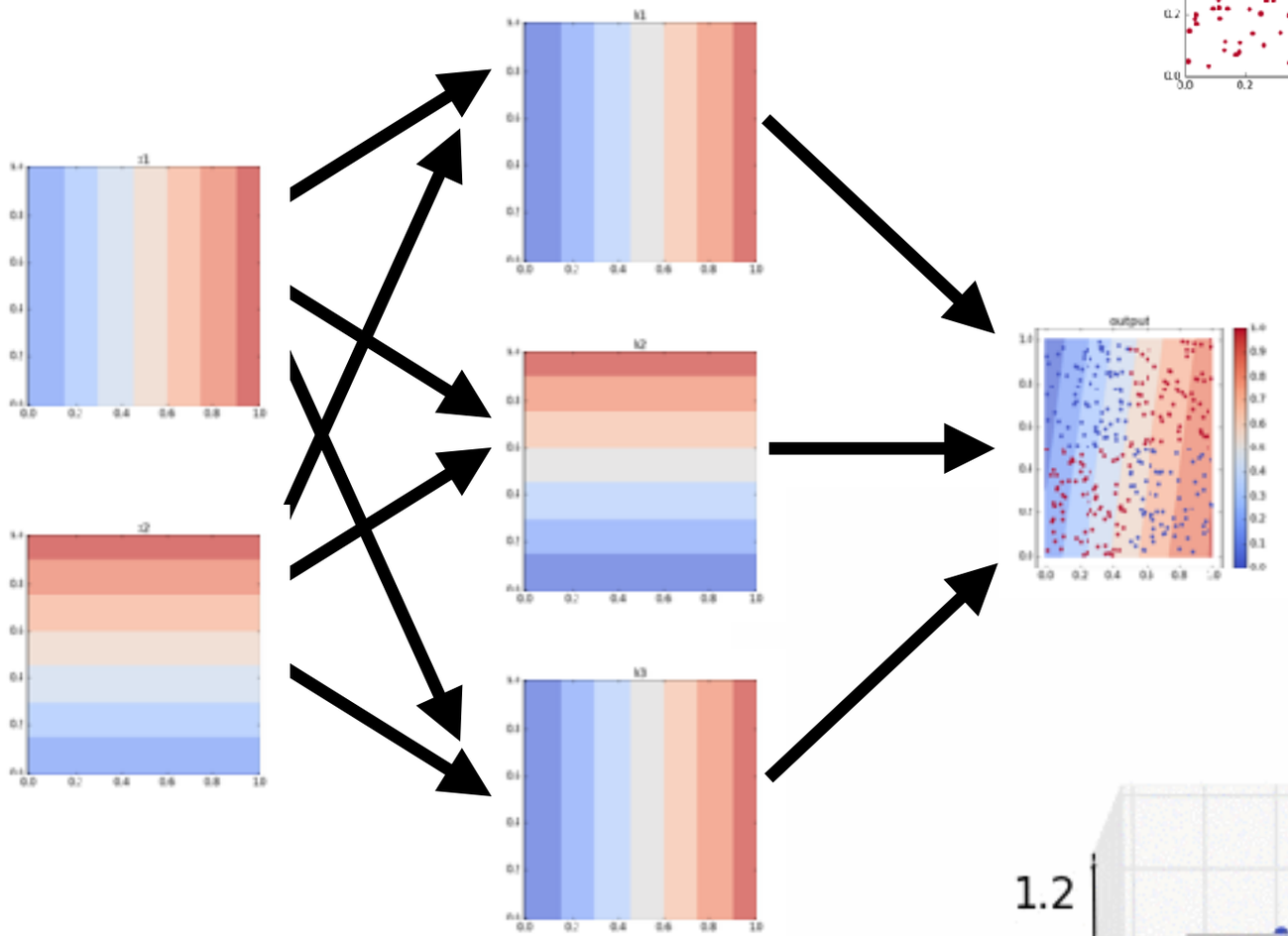


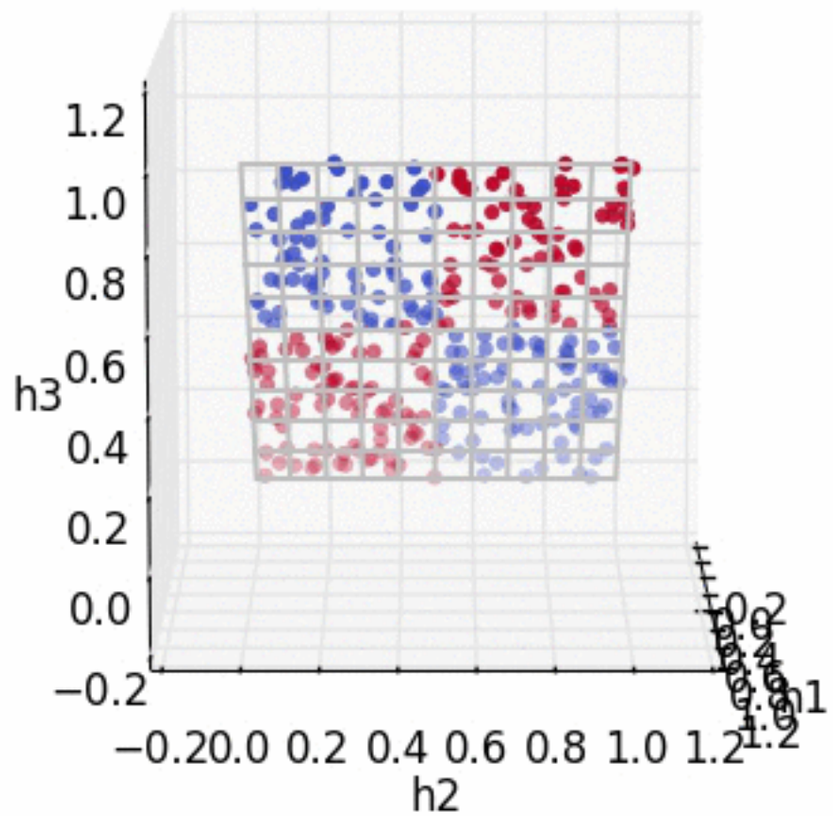


The XOR Data - ReLU

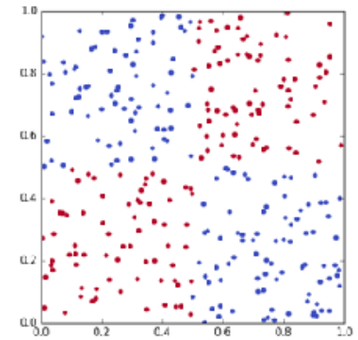
The XOR Data - ReLU

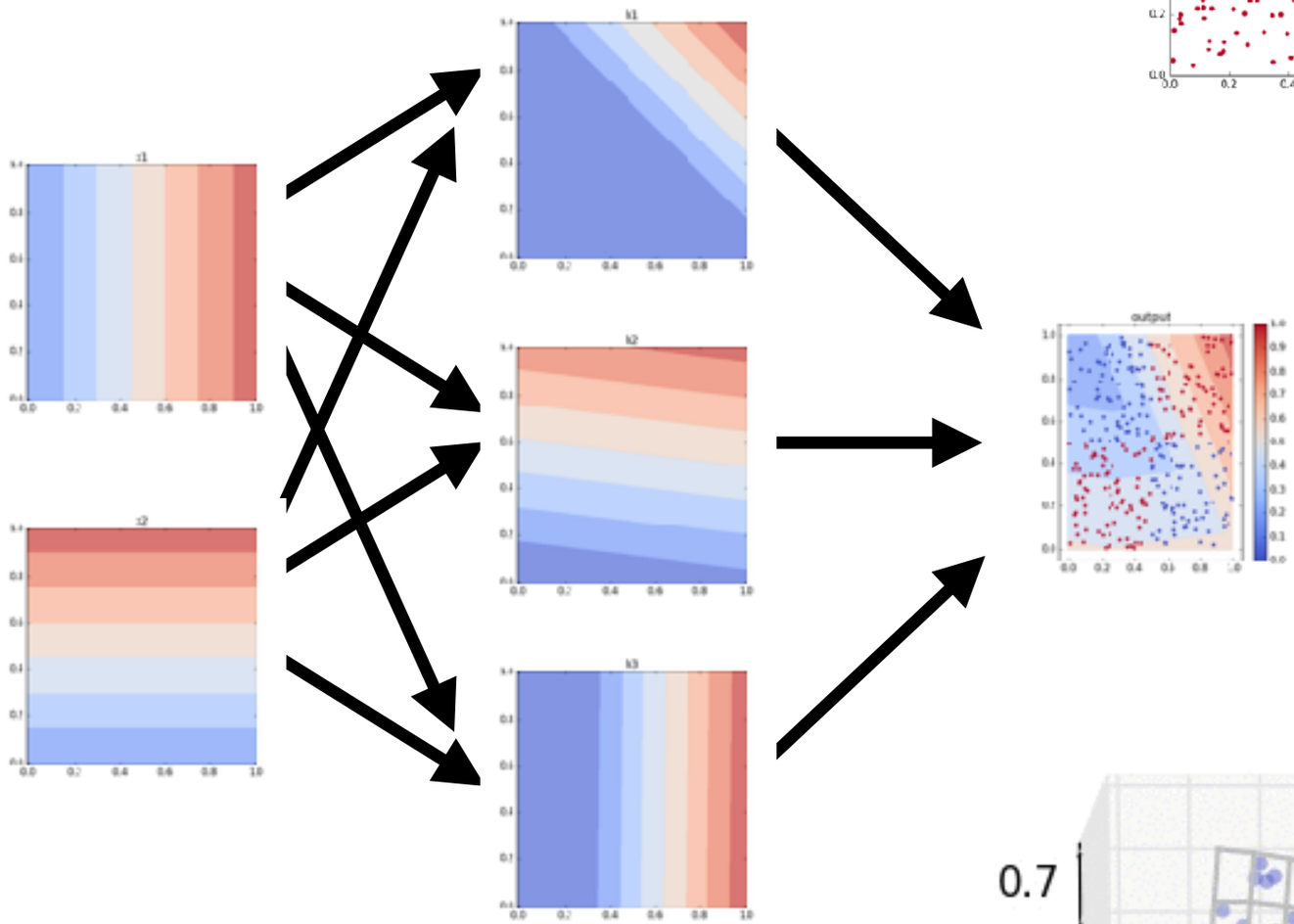


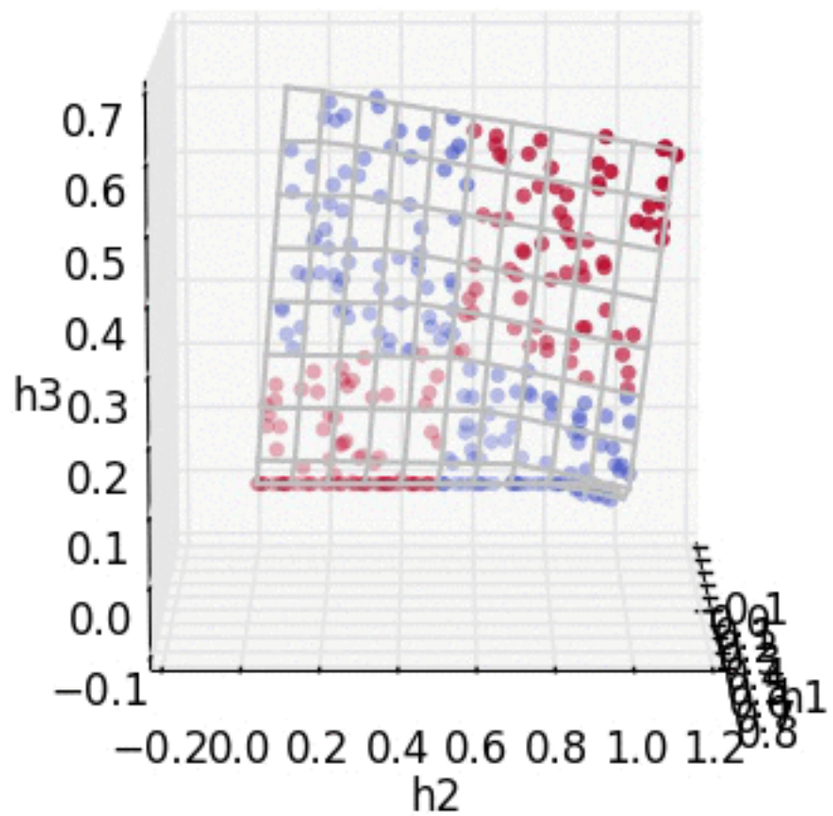




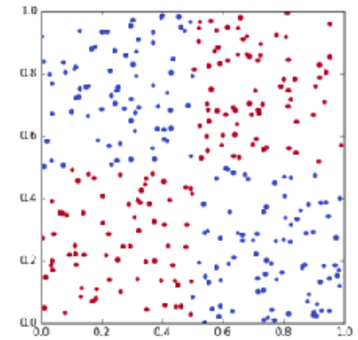
The XOR Data - ReLU

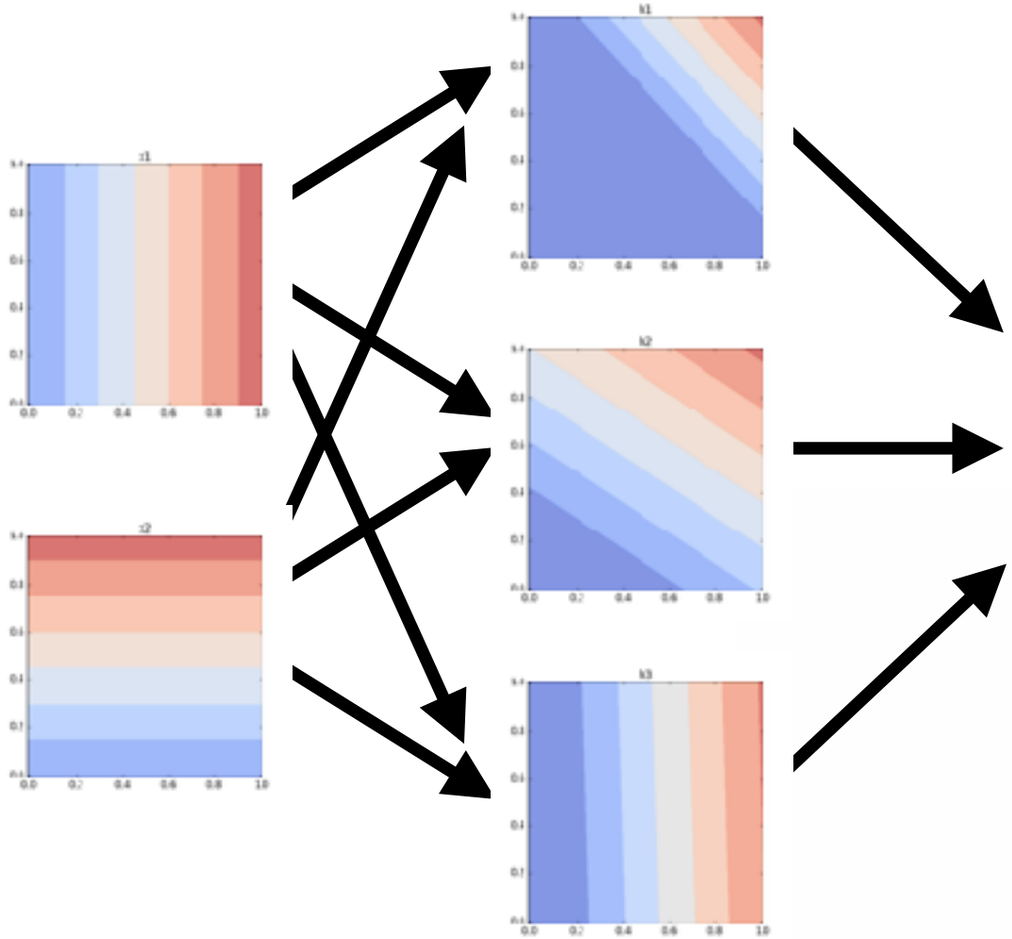


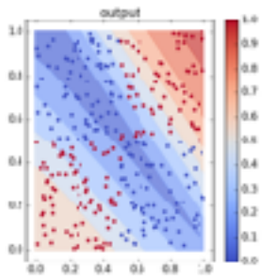


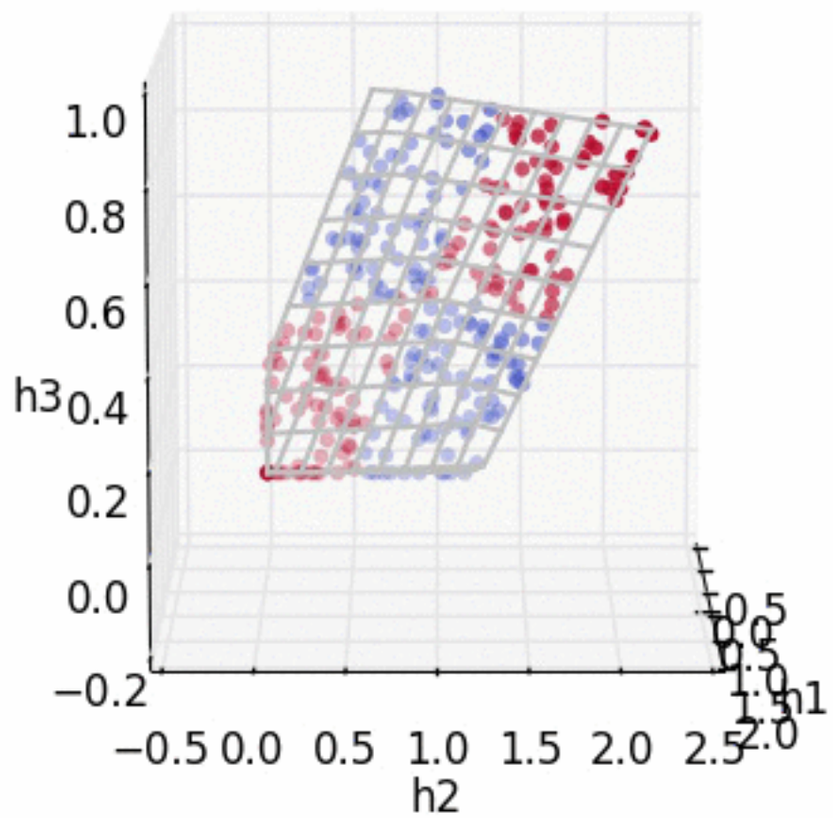


The XOR Data - ReLU

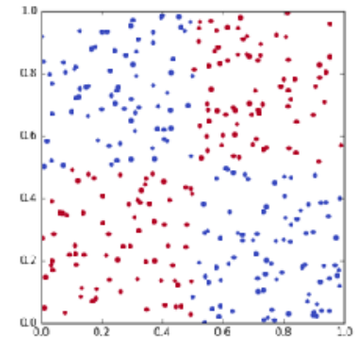


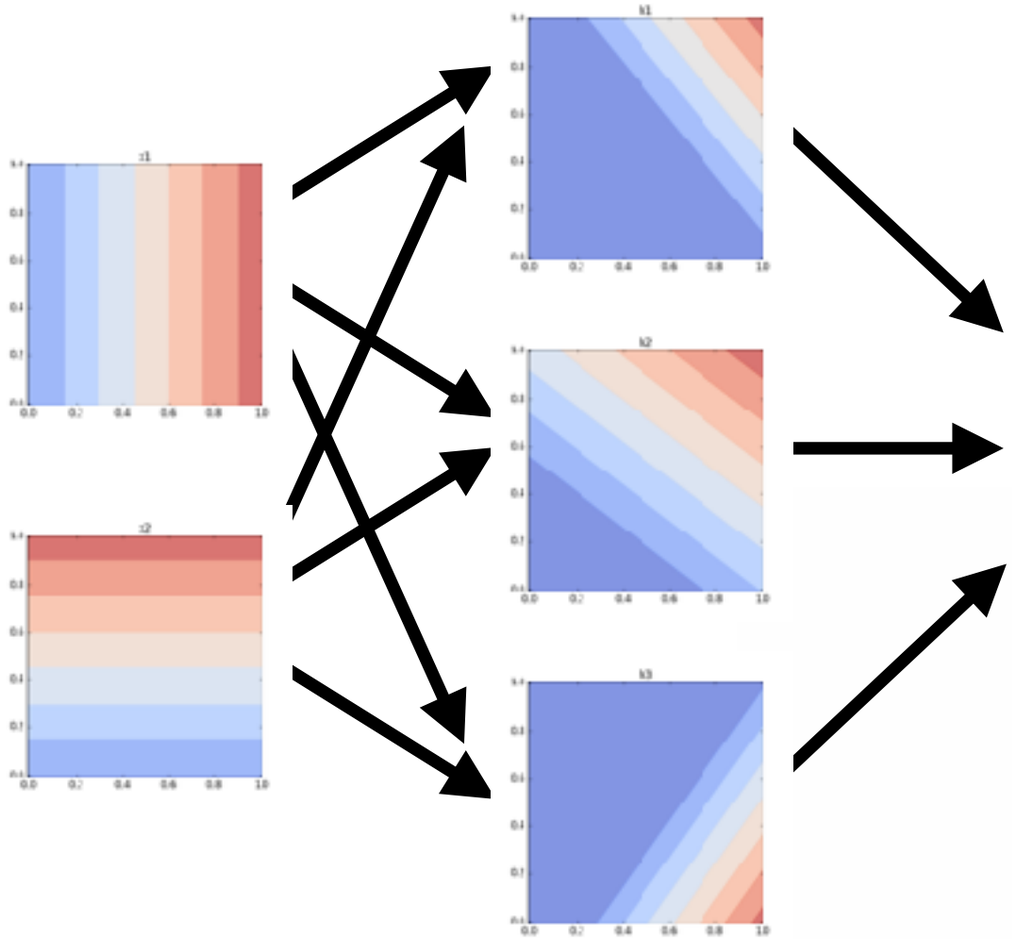


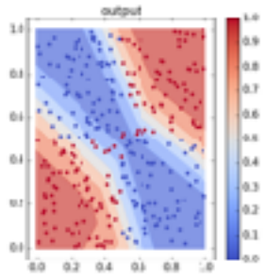


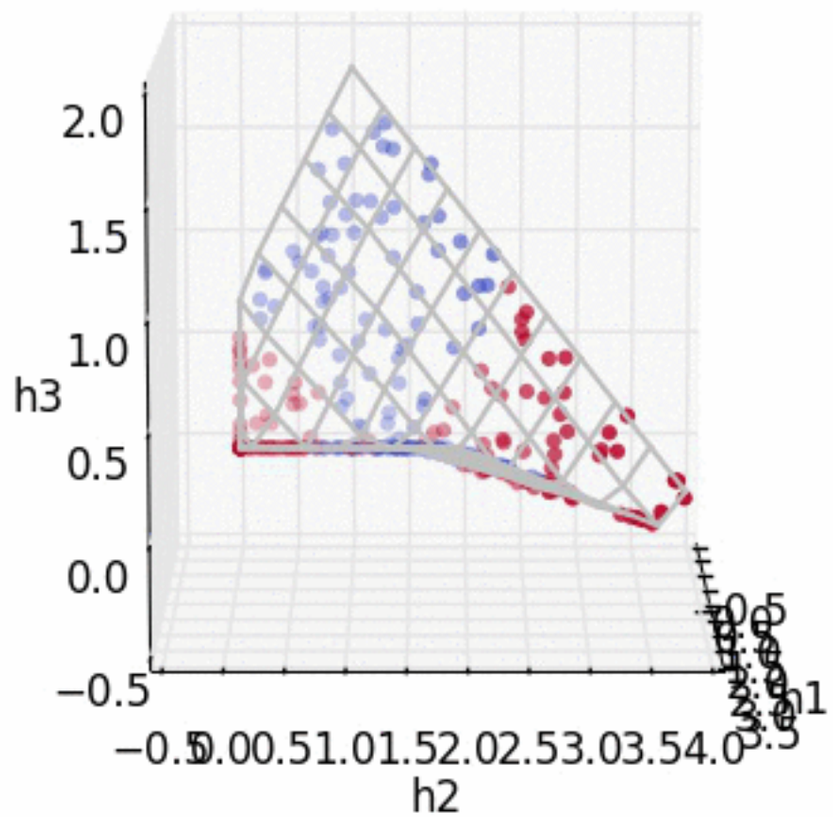


The XOR Data - ReLU

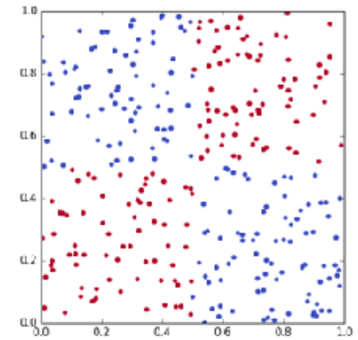


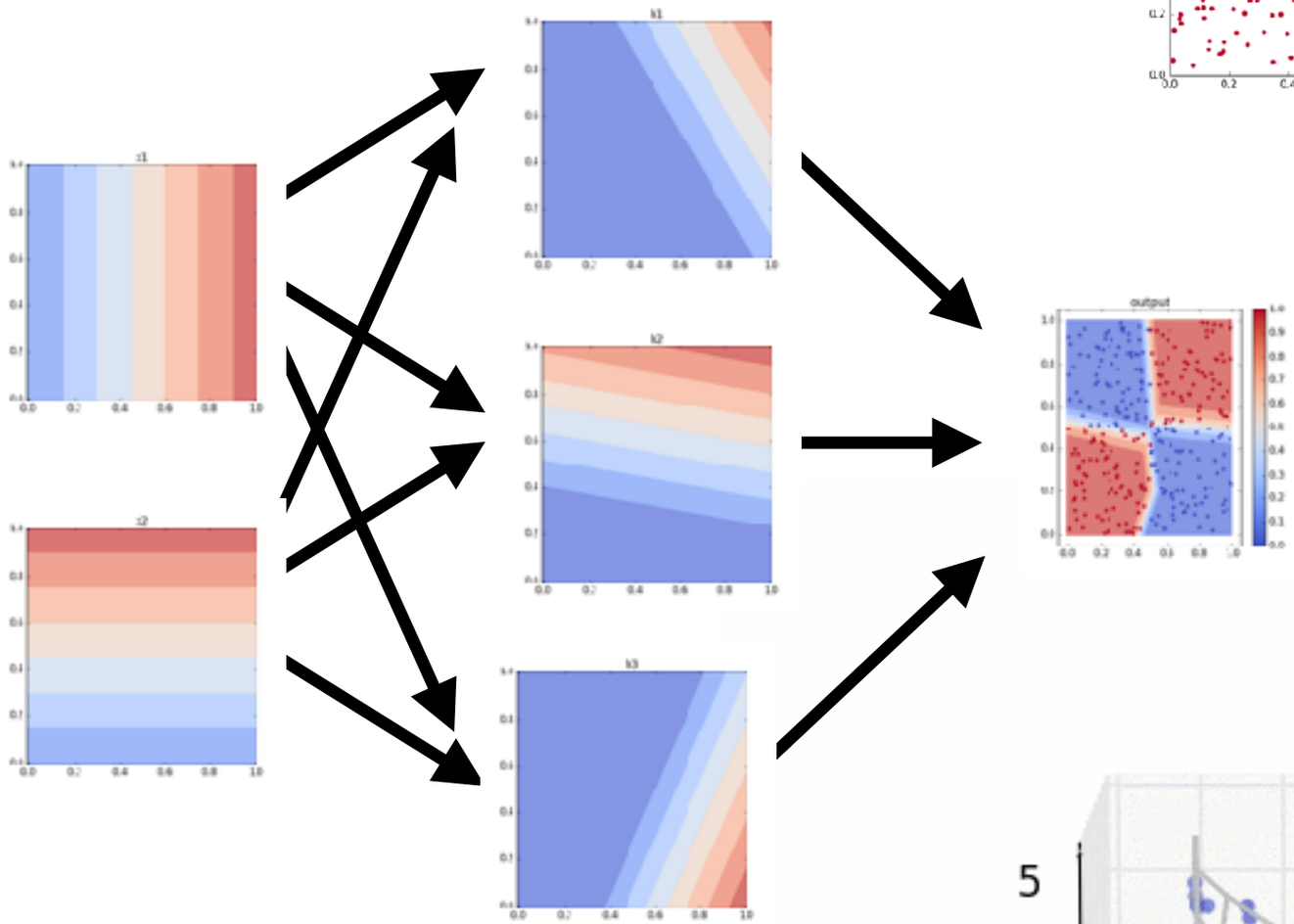


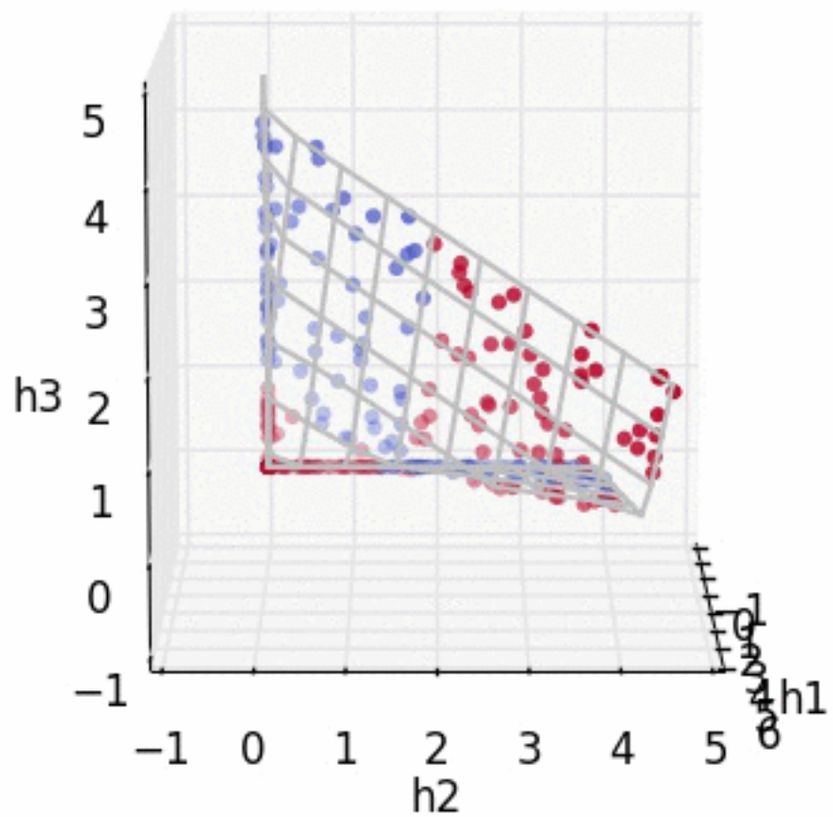




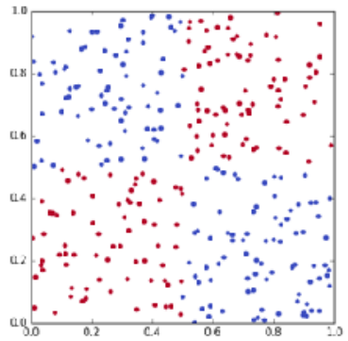
The XOR Data - ReLU

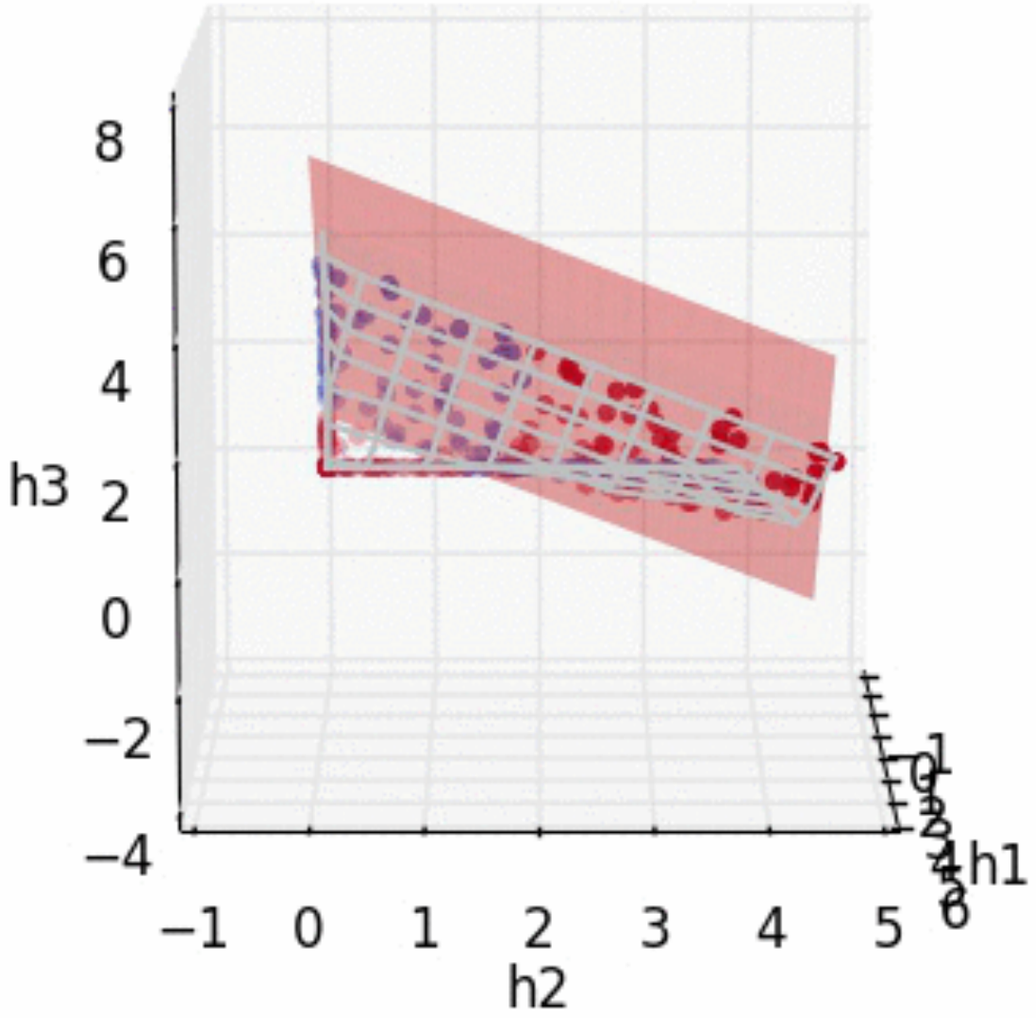






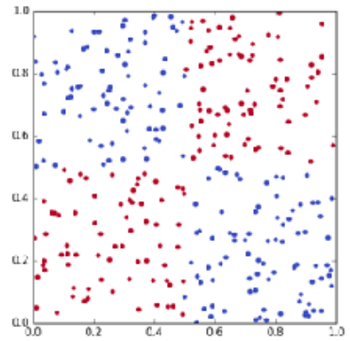
The XOR Data - ReLU

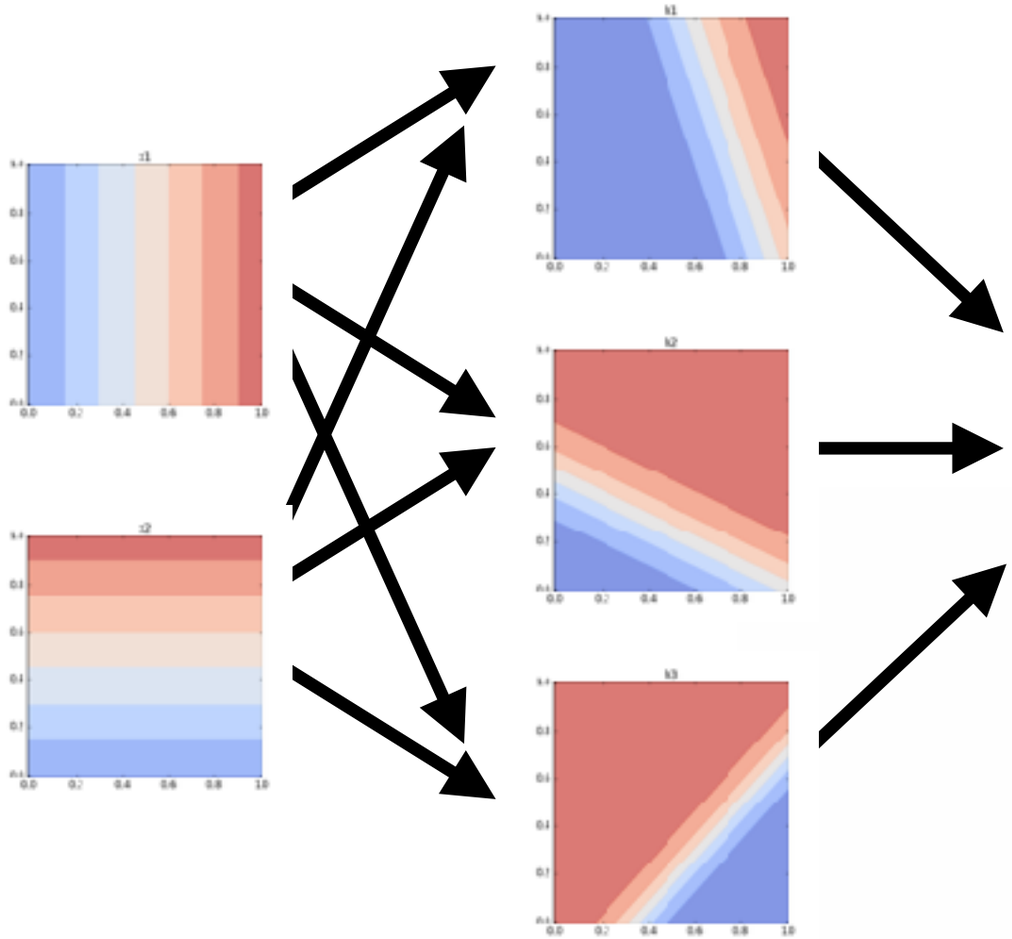


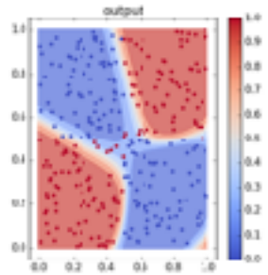


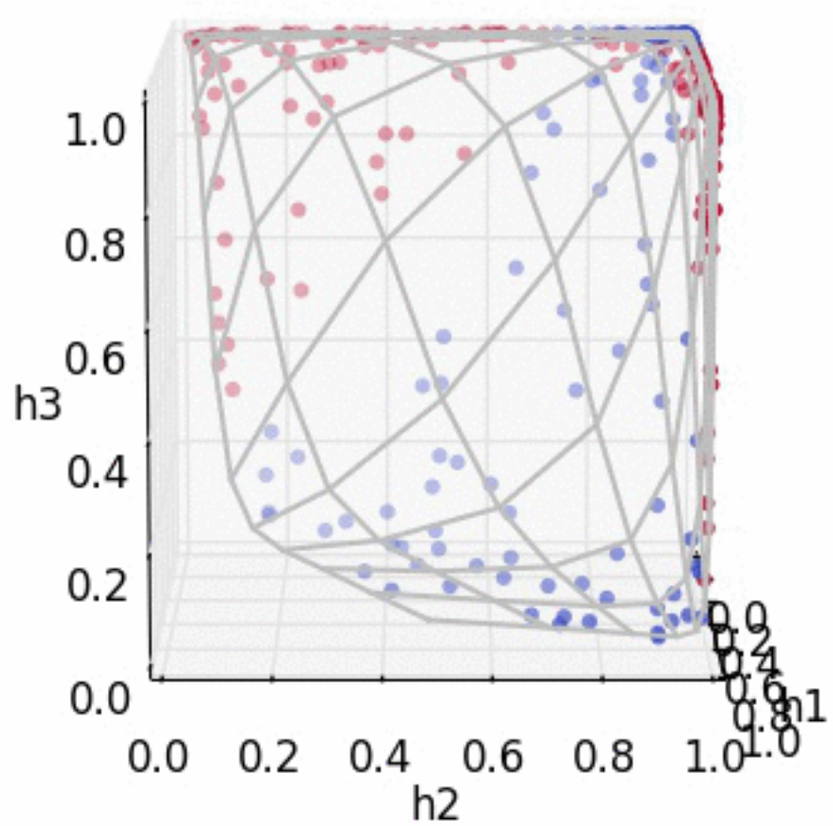
The XOR Data - Sigmoid

The XOR Data - Sigmoid

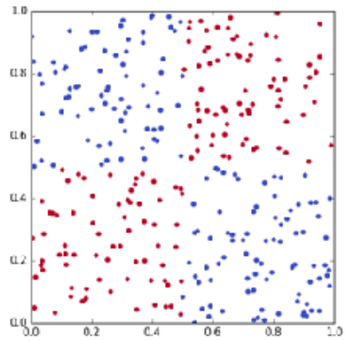


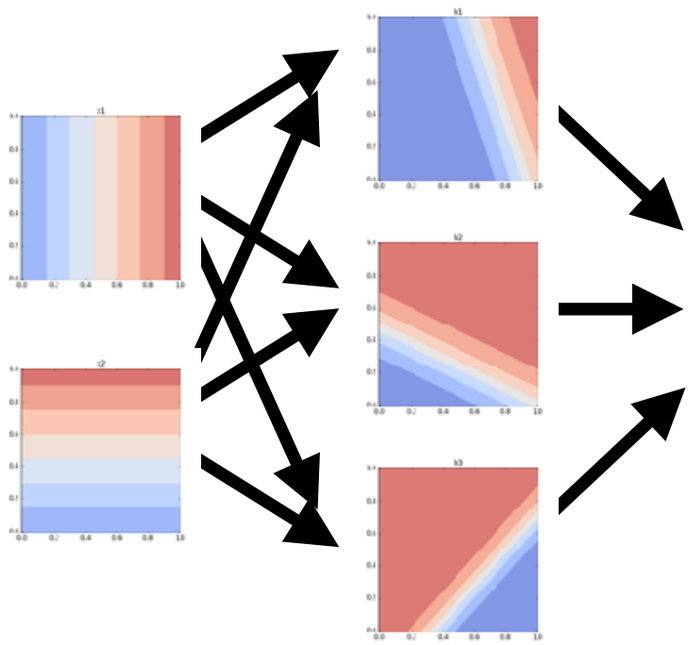


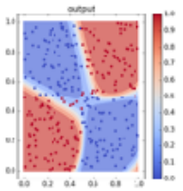


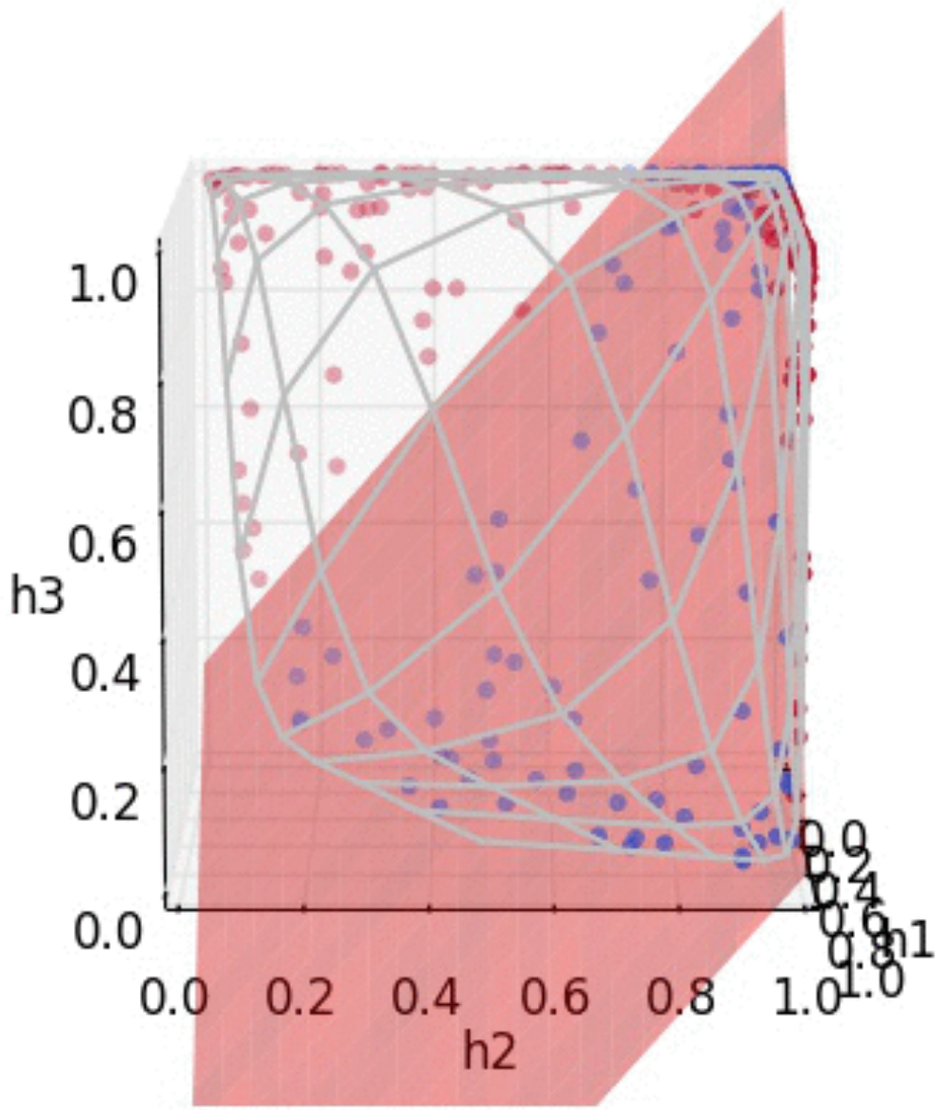


The XOR Data - Sigmoid





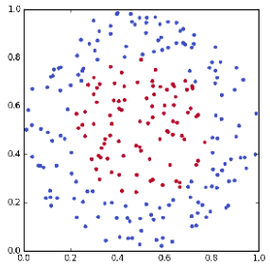


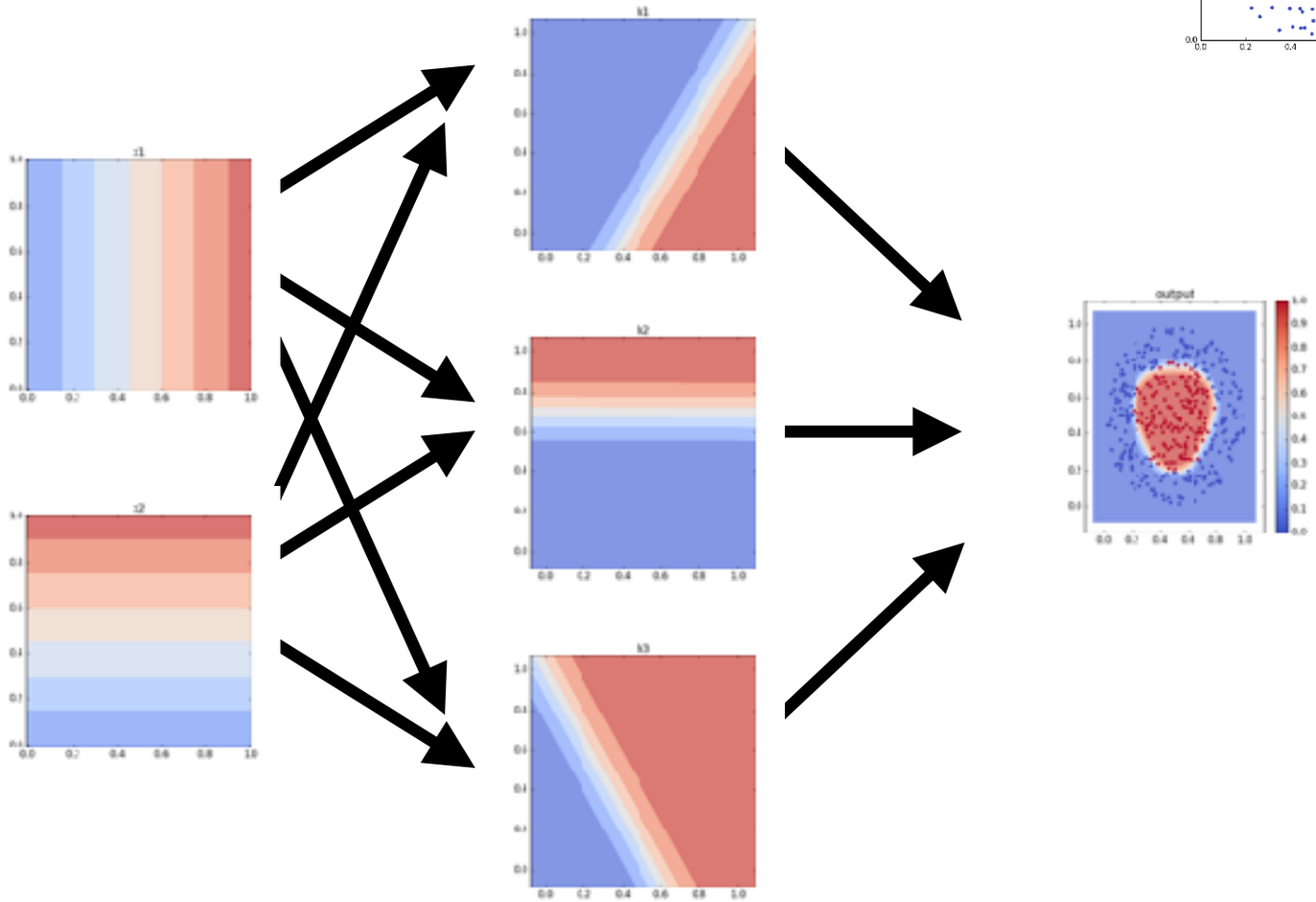


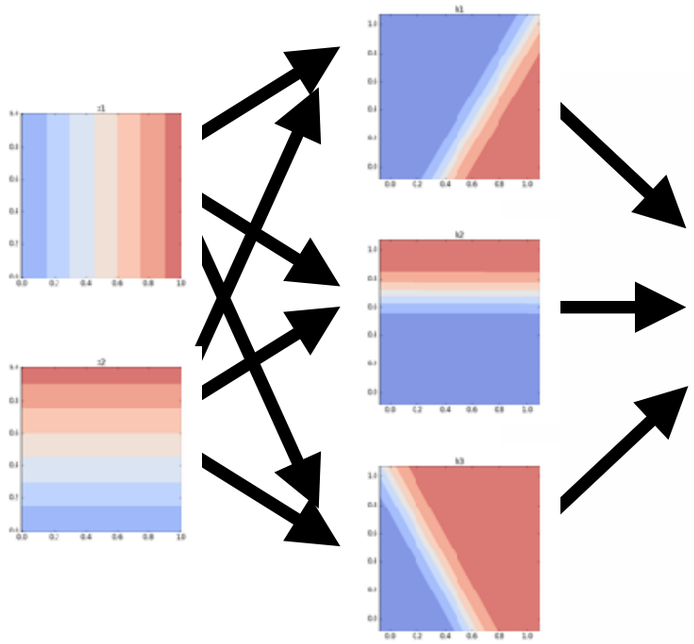
The Ring Data - Sigmoid

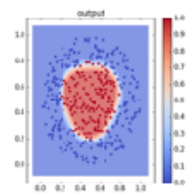
how the manifold is being fold?

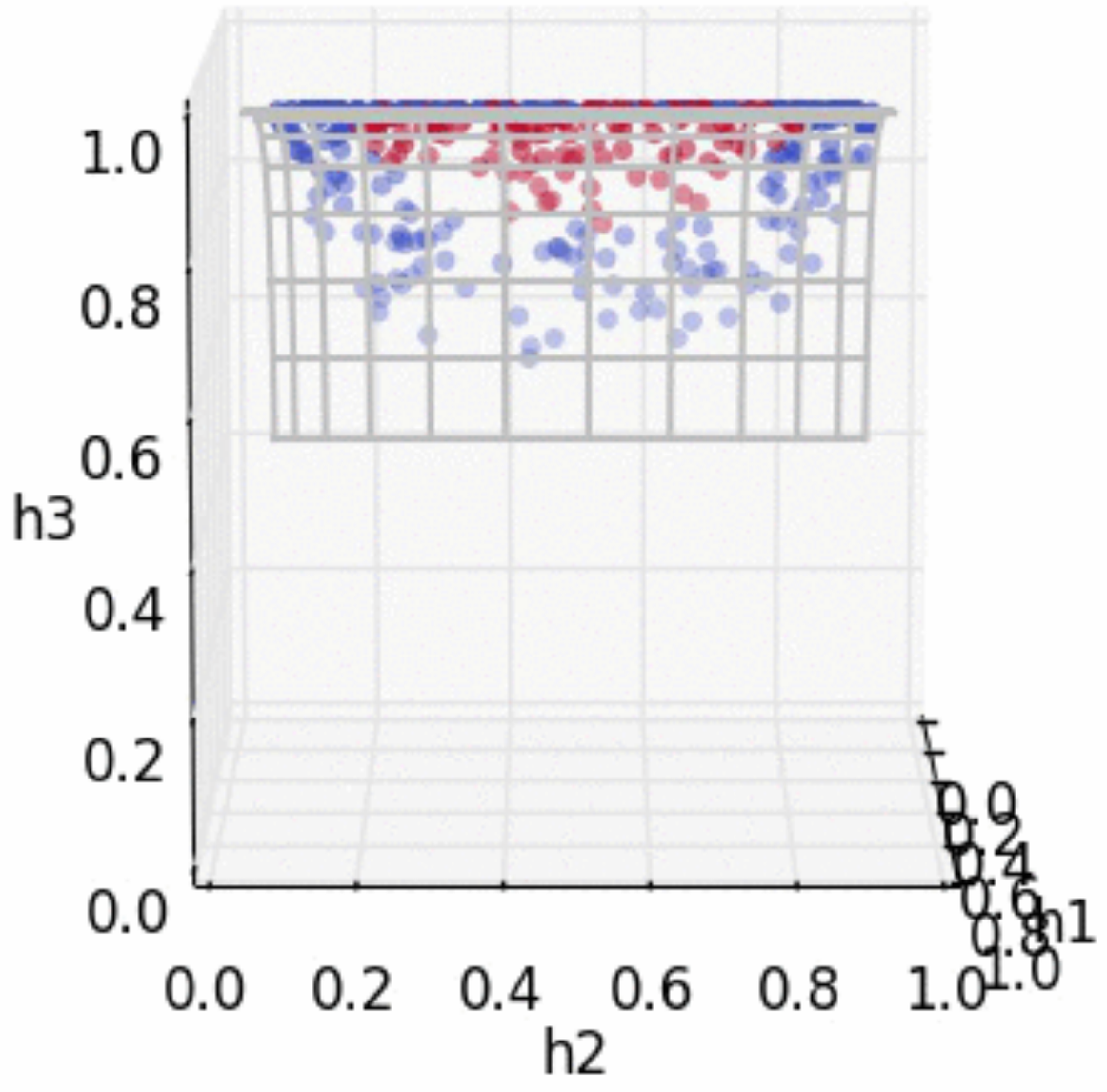
The Ring Data - Sigmoid

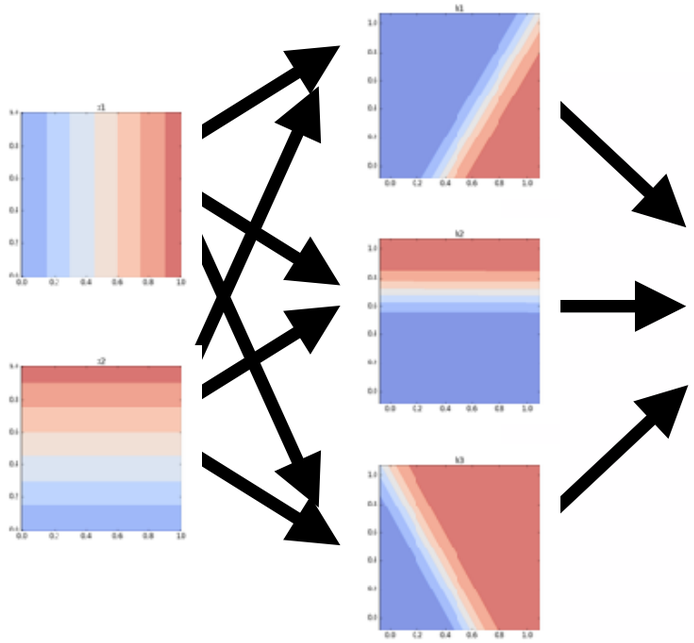


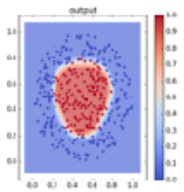


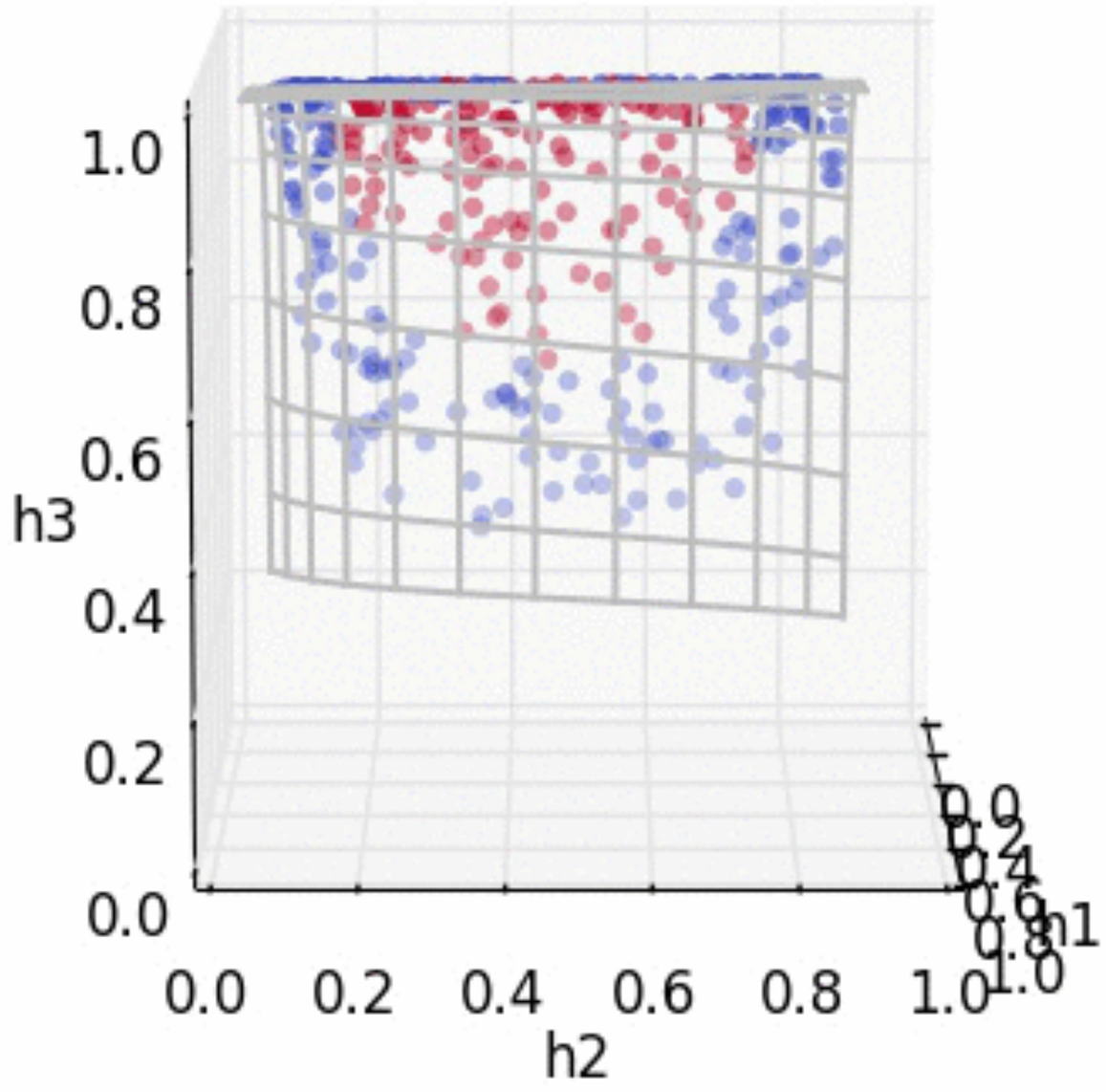

z1
z2
k1
k2
k3

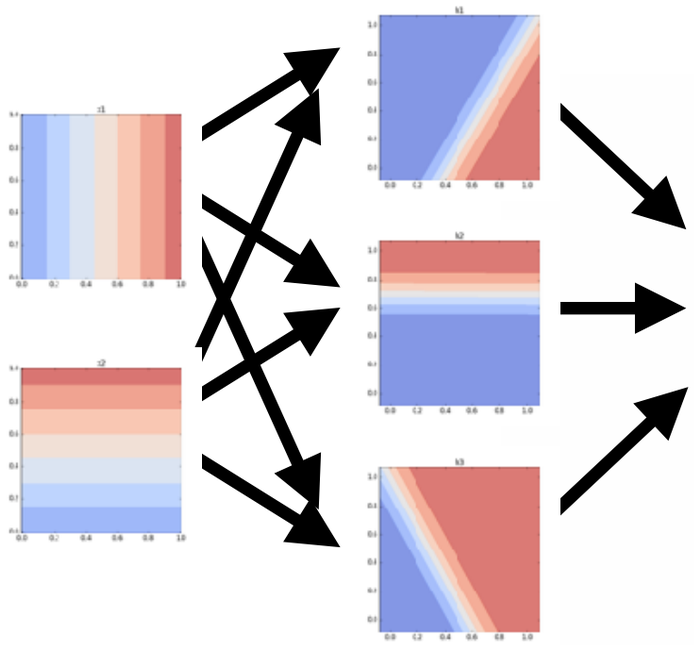

output

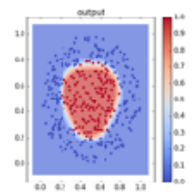

1.0
0.8
0.6
0.4
0.2
0.0
h3
0.0
0.2
0.4
0.6
0.8
1.0
h2
h1

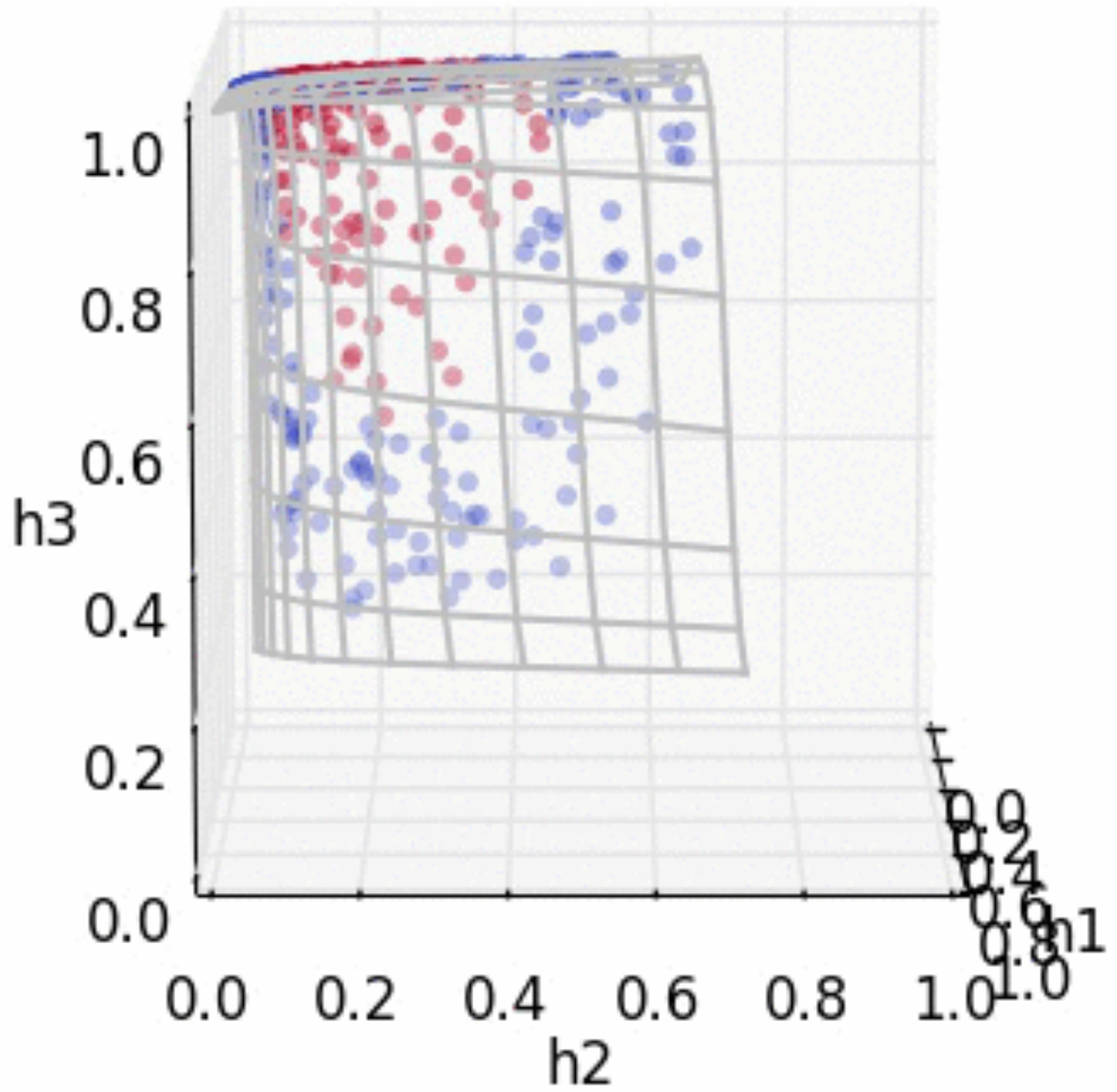

z1
z2
k1
k2
k3

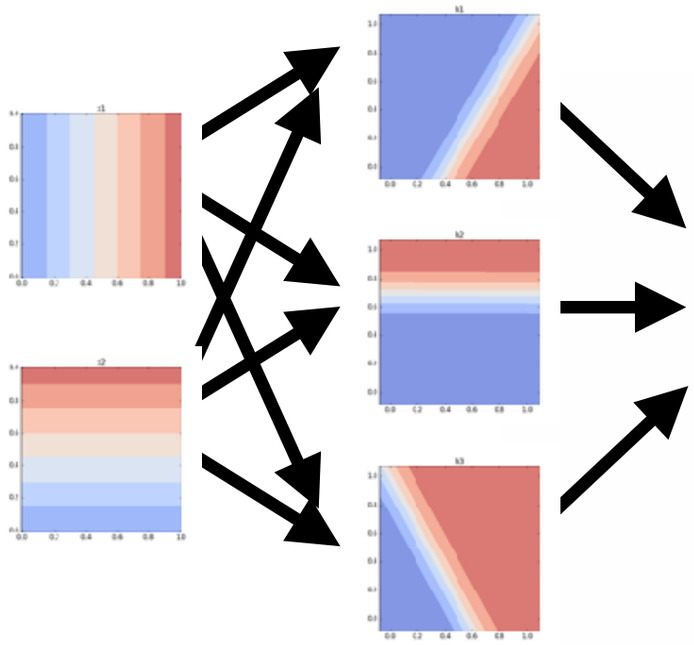

output

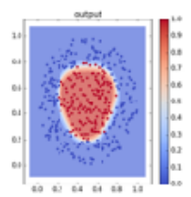

1.0
0.8
0.6
0.4
0.2
0.0
h3
0.0
0.2
0.4
0.6
0.8
1.0
h2
h1

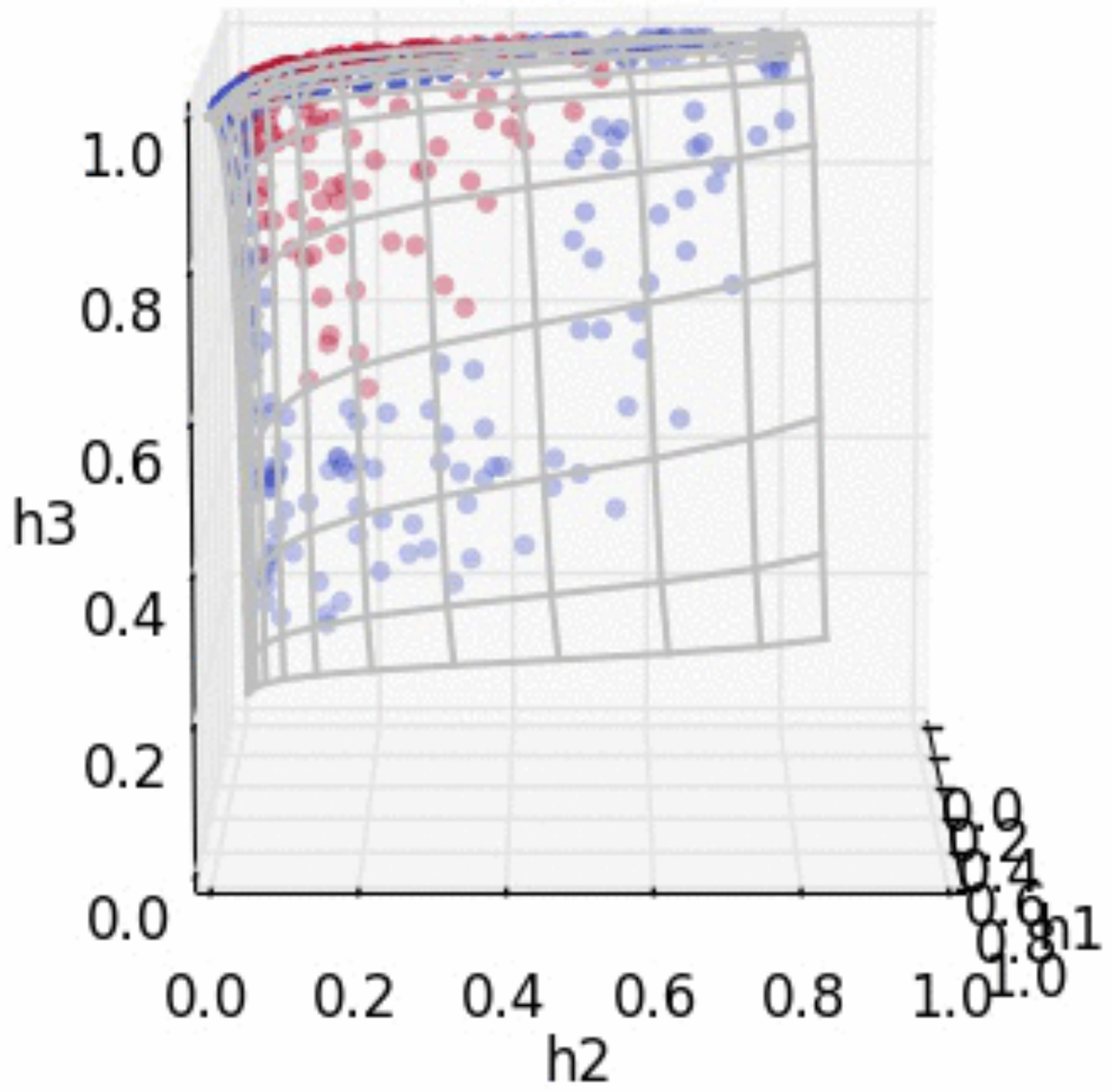

z1
z2
k1
k2
k3

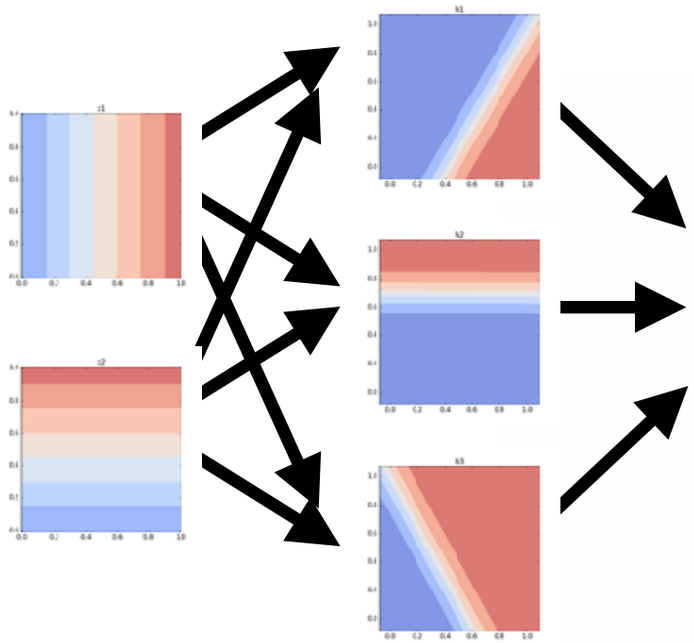

output

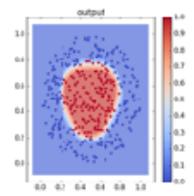

1.0
0.8
0.6
0.4
0.2
0.0
h3
0.0
0.2
0.4
0.6
0.8
1.0
h2
h1

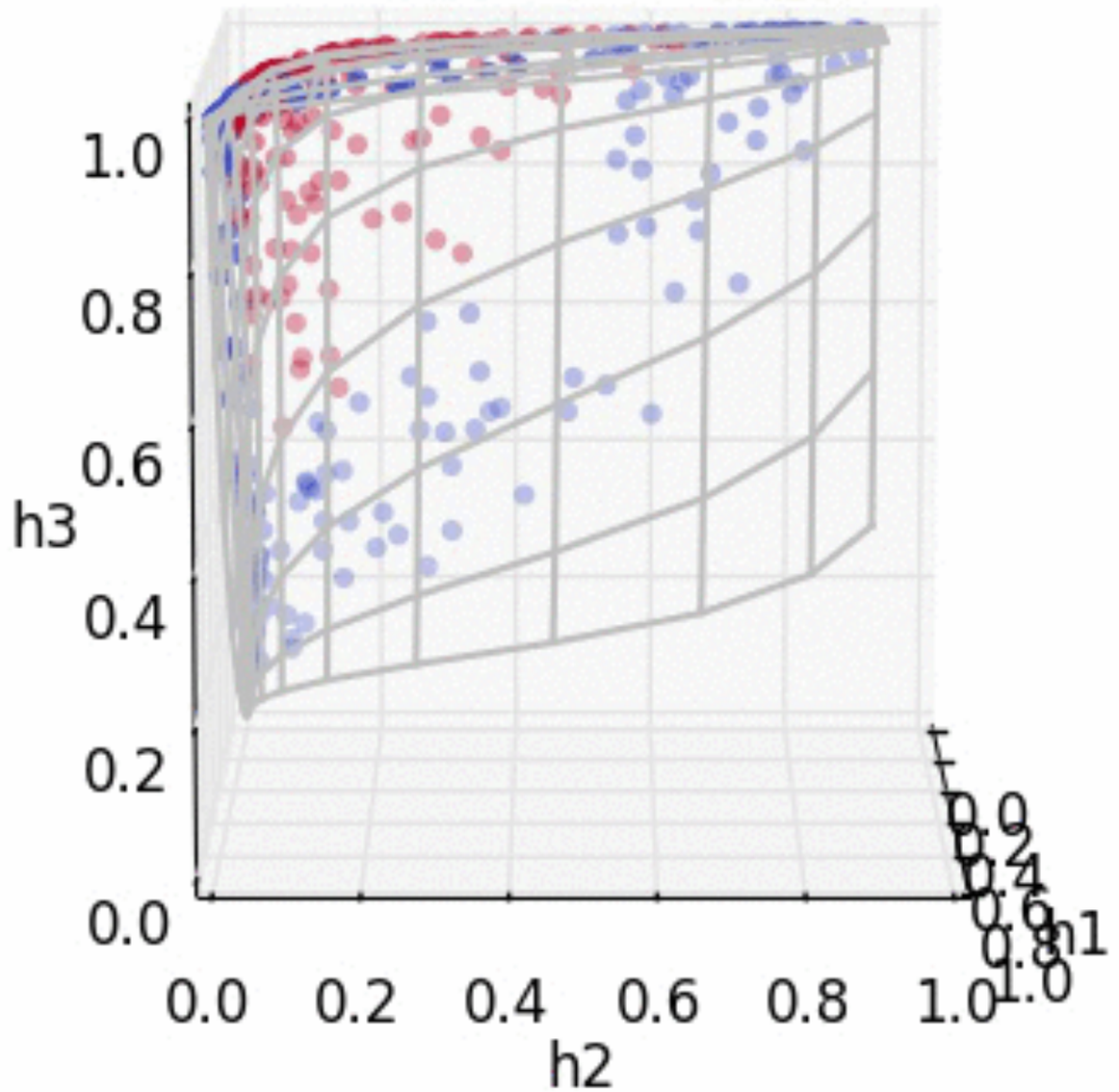

z1
z2
k1
k2
k3

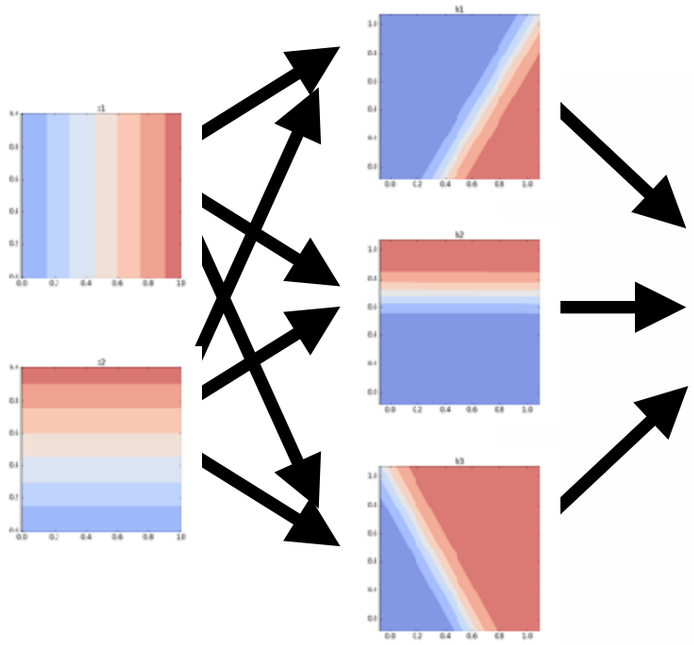

output

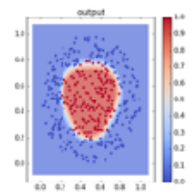

1.0
0.8
0.6
0.4
0.2
0.0
h3
0.0
0.2
0.4
0.6
0.8
1.0
h2
h1

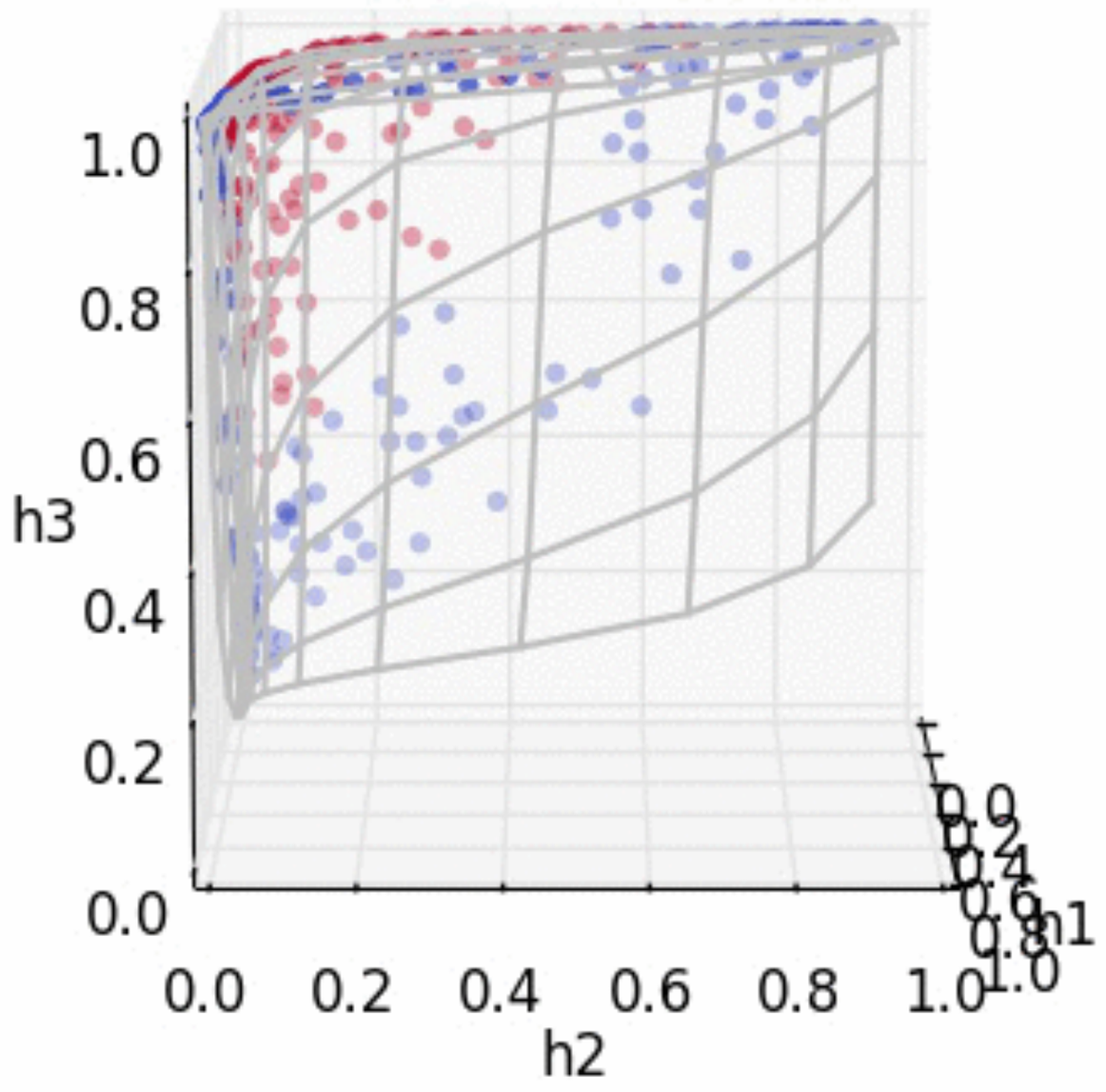

z1
z2
k1
k2
k3

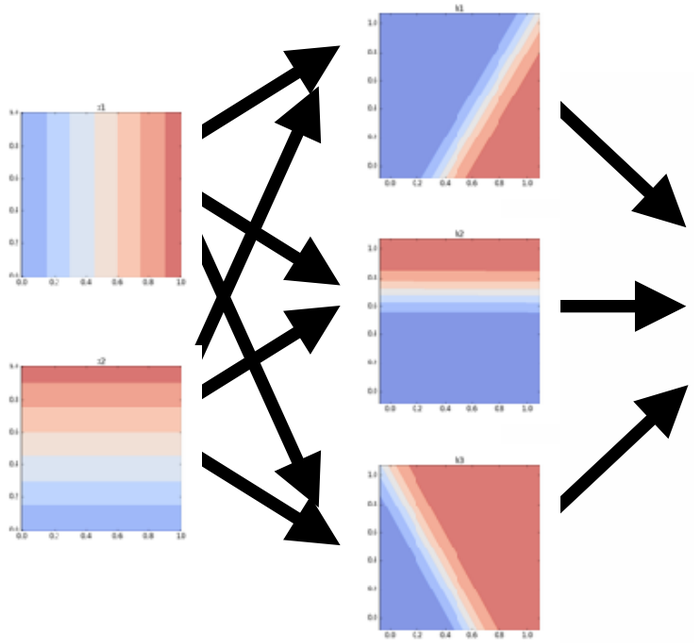

output

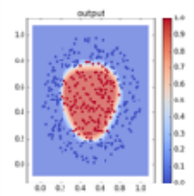

1.0
0.8
0.6
h3
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
h2
h1

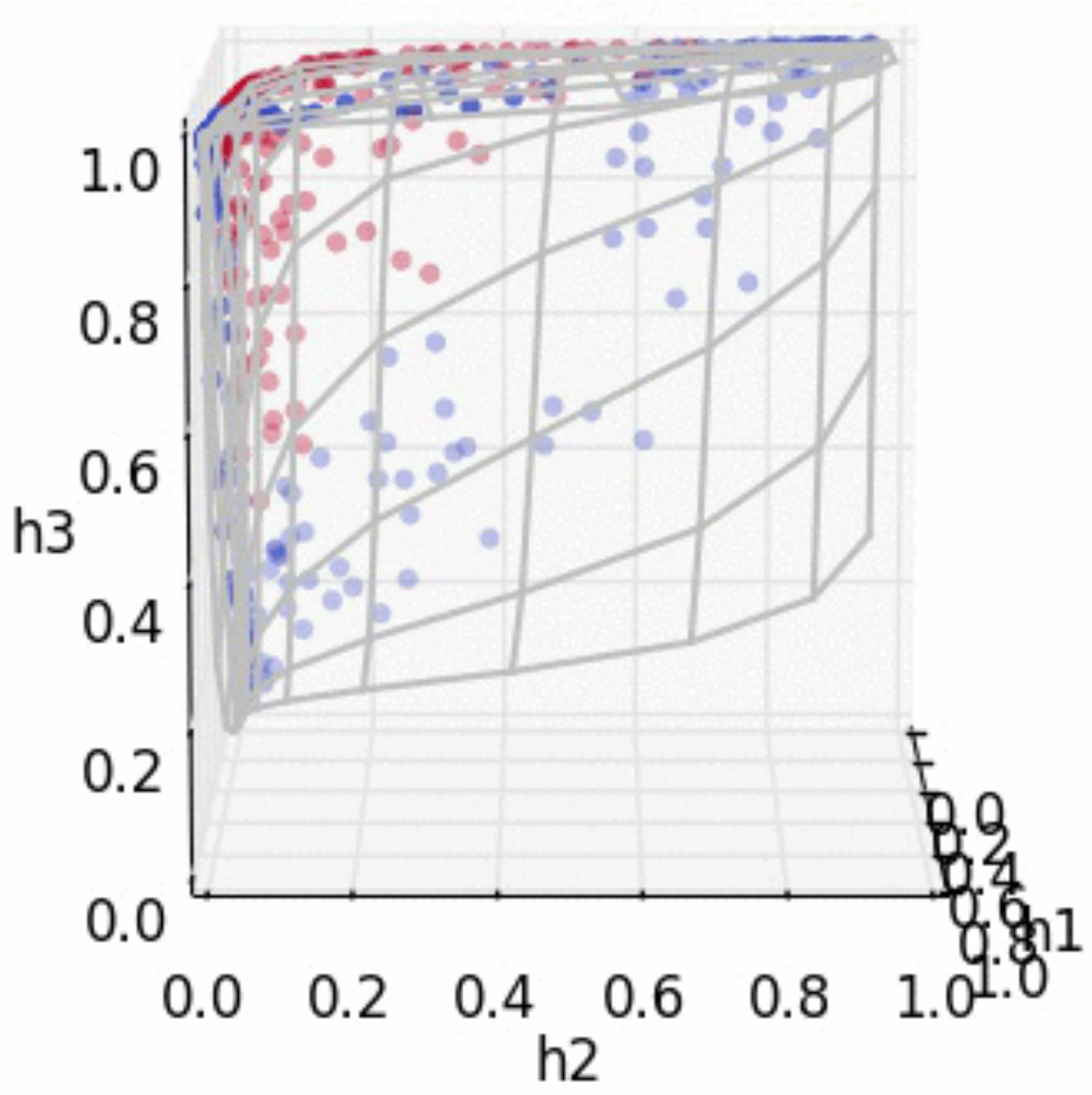

z1
z2
k1
k2
k3

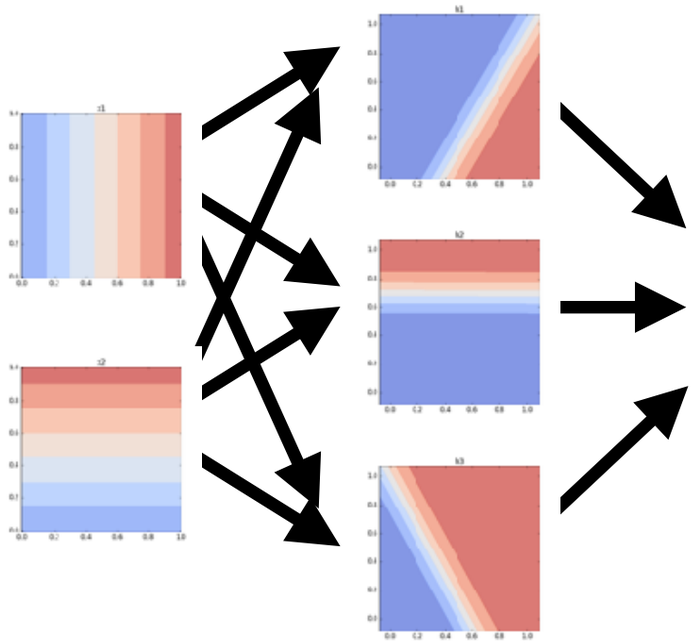

output

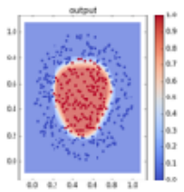

1.0
0.8
0.6
0.4
0.2
0.0
h3
0.0
0.2
0.4
0.6
0.8
1.0
h2
h1


z1
z2
k1
k2
k3

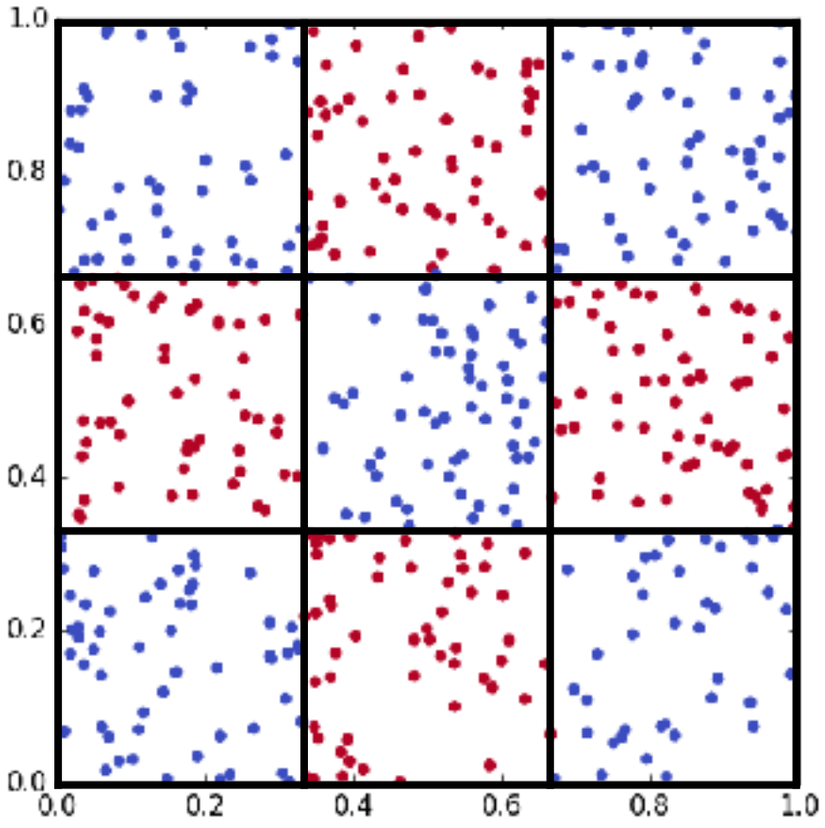

output

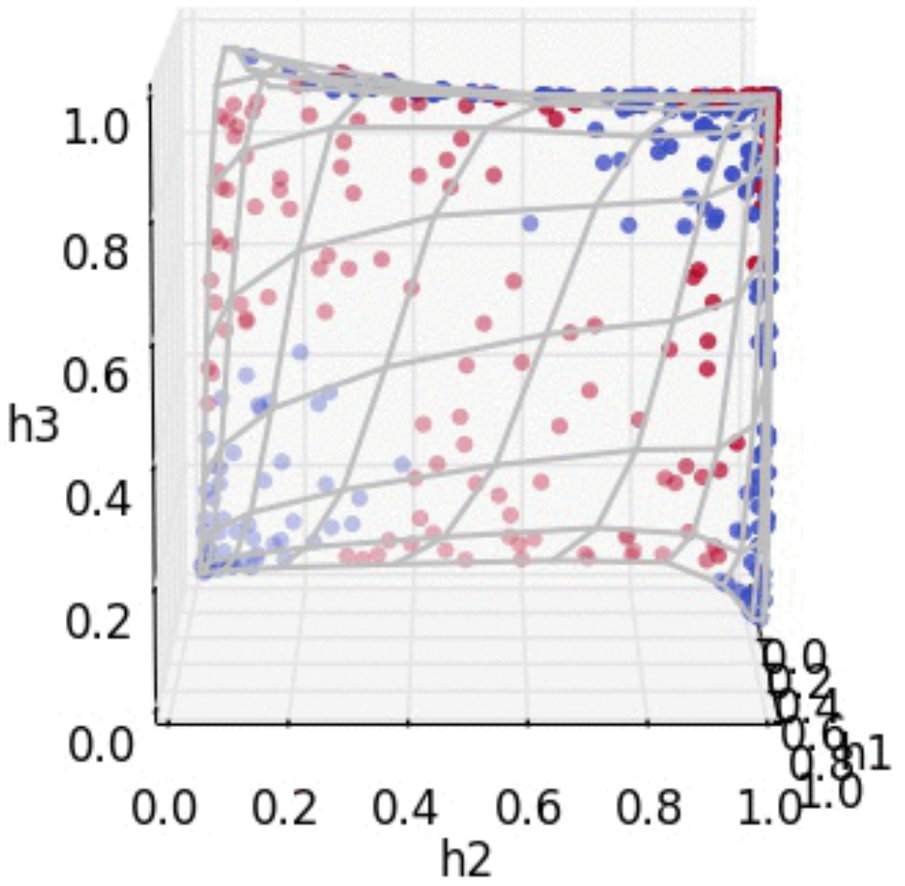

1.0
0.8
0.6
0.4
0.2
0.0
h3
0.0
0.2
0.4
0.6
0.8
1.0
h2
h1

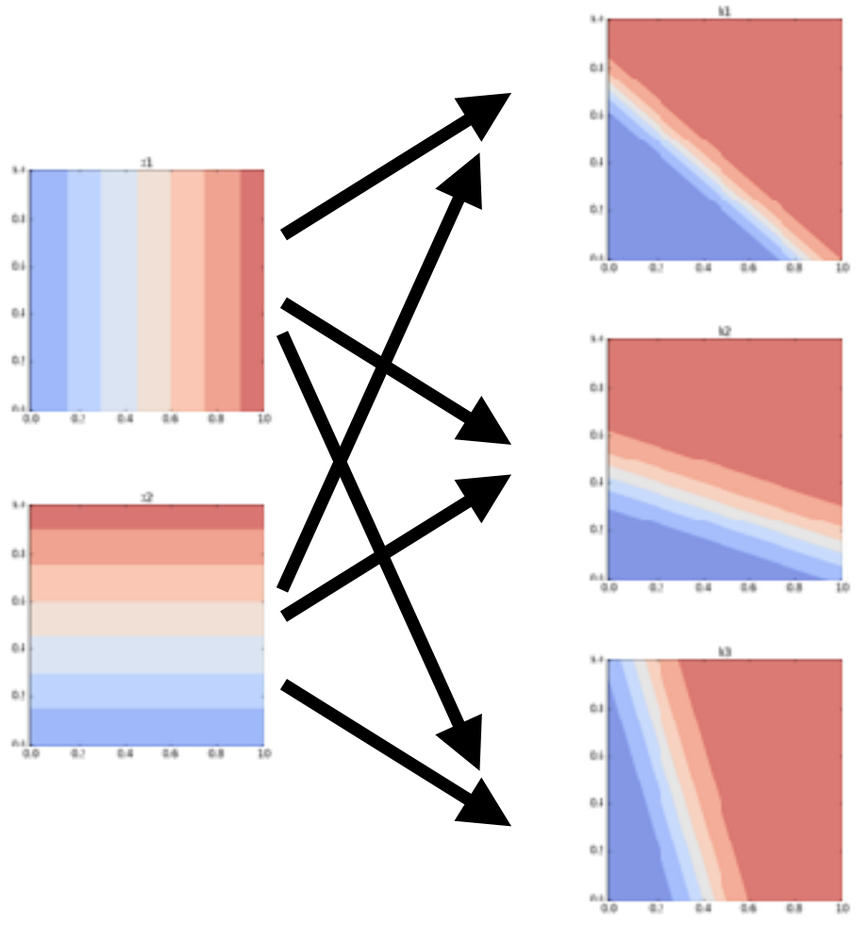

z1
z2
k1
k2
k3

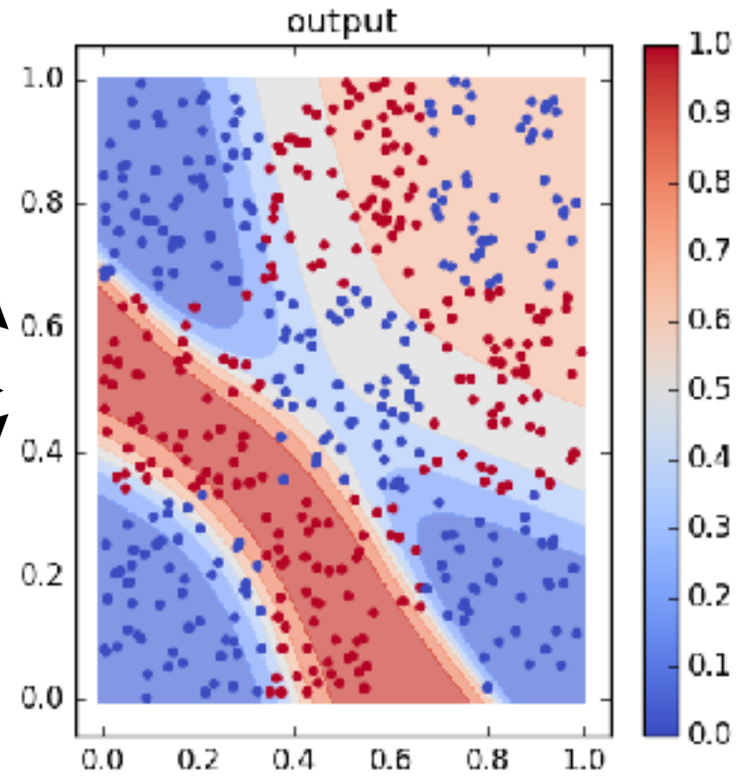

output

1.0
0.8
0.6
0.4
0.2
0.0
h3
0.0
0.2
0.4
0.6
0.8
1.0
h2
h1

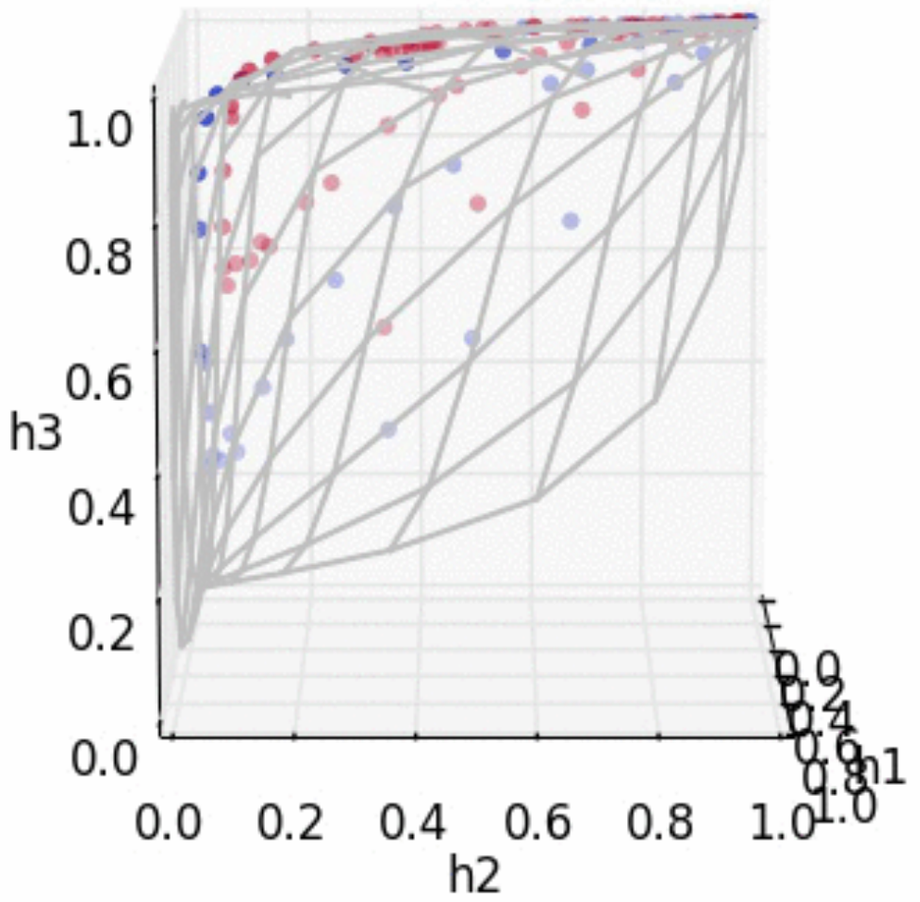

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

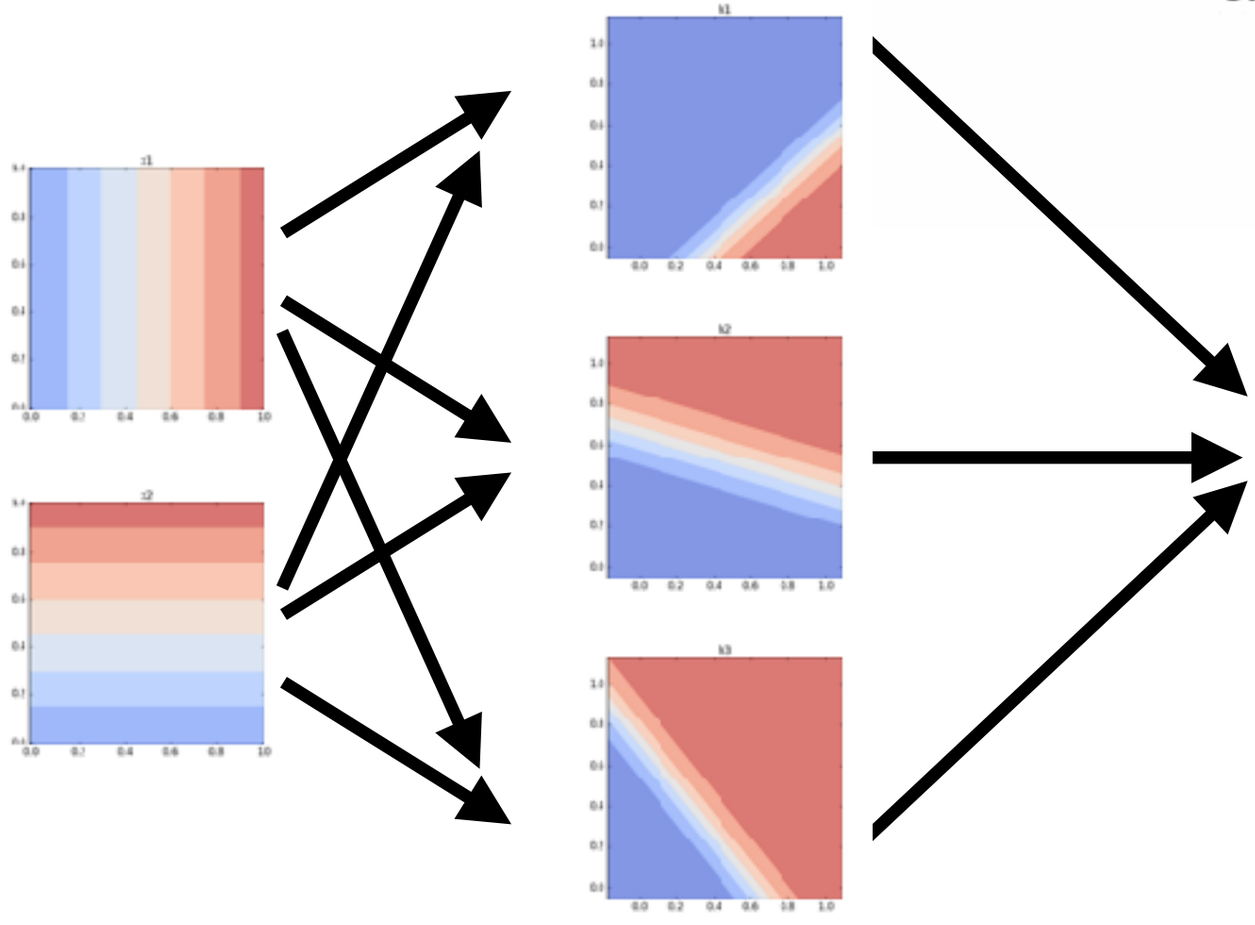

h3
h2
h1

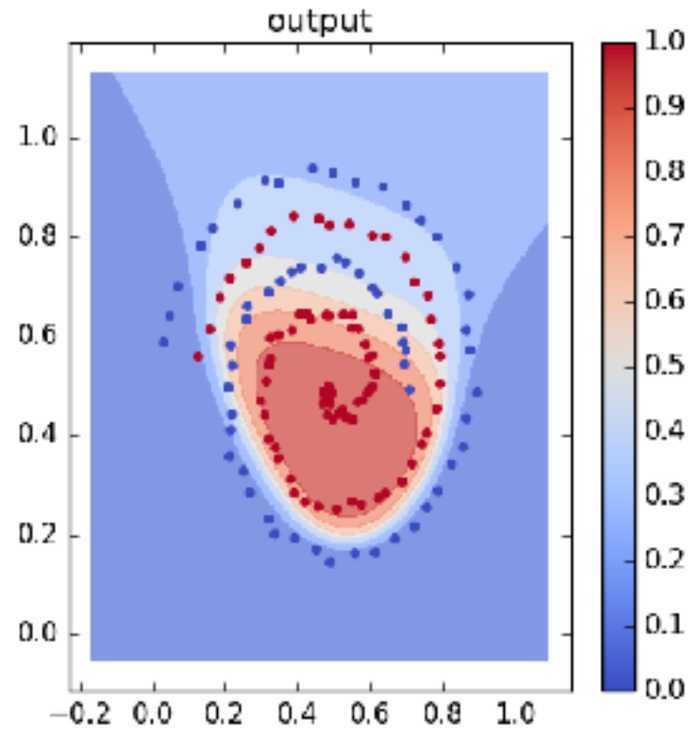



output
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0

Spiral


h3
h2
h1
0.0
0.2
0.4
0.6
0.8
1.0




output
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0

How about this network?