Administrative Issues

CS5242 mirror site:
https://web.bii.a-star.edu.sg/~leehk/cs5424.html

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

Assignments (10% * 4 = 40%)
Quiz (20%)
Final Project (40%)

Final Projects (40%)

- Projects will be held on Kaggle-in-class
  - Register using your nus email (https://inclass.kaggle.com/)
  - The project webpages will be announced once groups are finalized
  - No data disclosure
  - No additional data
  - Each group <=2 students
- Each group will be randomly assigned to
  - A computer vision project
  - A natural language processing project
- Assessment is based on the report (15%) and ranking (25%)
  - Top ranked groups of each project will be invited to do presentation at week 13.

Will announce in IVLE once Kaggle is ready

Final Project (40%)

- Report template (15%)
  - 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)
- Ranking -> score (25%)
  - Rank top 10%, score ~25 (max score)
  - Rank top 20%, score ~21
  - Rank top 40%, score ~17
  - Rank top 70%, score ~14
  - Rank others, score ~13
  - Prediction scores less than 10% better than random guess, score ~5

Final rank to grade distribution may be adjusted depending on overall class performance

Final projects timeline summary

Date	Event
27 August 18:00	Register your project group @ IVLE 2 or less students per group
04 November 18:00	Results submission deadline
05 November 18:00	Selection and notification of selected projects for presentation
11 November 18:00	Report deadline
12 November 18:00	Selected project group send slides to cs5242@comp.nus.edu.sg
12 November 18:00	Make appointment to meet Hwee Kuan to do presentation rehearsal
16 November 18:30	Project presentations by students

GPU Machine/Cluster

- NSCC (National Super-Computing Center)
  - Free for NUS students
  - 128 GPU nodes, each with a NVIDIA Tesla K40 (12 GB)
  - Follow the manual and test program on IVLE workbin/misc/gpu to get yourself familiar with it.
- Google Cloud Platform
  - 300 USD credit for new register
  - NVIDIA Tesla K80 (2x12GB) is available for Region Taiwan
- Amazon EC2 (g2.8xlarge)
  - 100 USD credit for students
  - NVIDIA Grid GPU, 1536 CUDA cores, 4GB memory
  - Available for Singapore region
- Tips:
  - STOP/TERMINATE the instance immediately after your program terminates
  - Check the usage status frequently to avoid high outstanding bills
  - Use Amazon or Google cloud platform for debugging and use NSCC for actual training and tuning

Lecture 1: Introduction

Lecture 2: Neural Network Basics

Lecture 3: Training the Neural Network

Lecture 4: Convolutional Neural Network

Lecture 5: Convolutional Neural Network

Lecture 6: Convolutional Neural Network

Lecture 7: Recurrent Neural Network

Lecture 8: Recurrent Neural Network

Lecture 9: Recurrent Neural Network

Lecture 10: Advanced Topic: Generative Adversarial Network

Lecture 11: Advanced Topic: One-shot learning

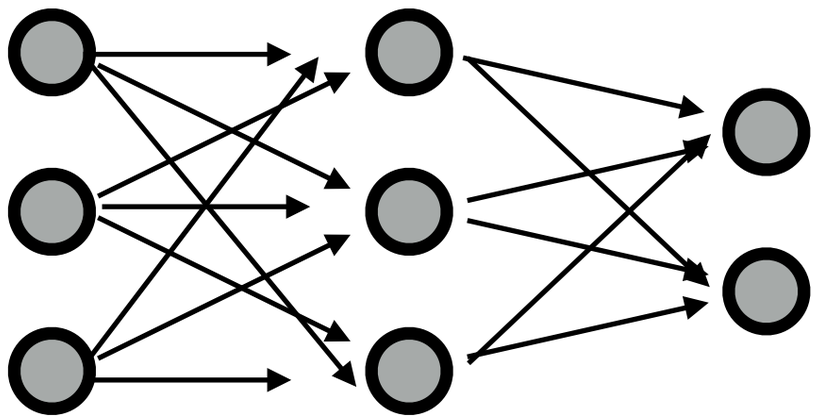
Lecture 12: Advanced Topic: Distributed training

Lecture 13: Project demonstrations

Forward Propagation

The network and information flow

feed in data into first layer



final layer presents the output of network

transmitter receiver

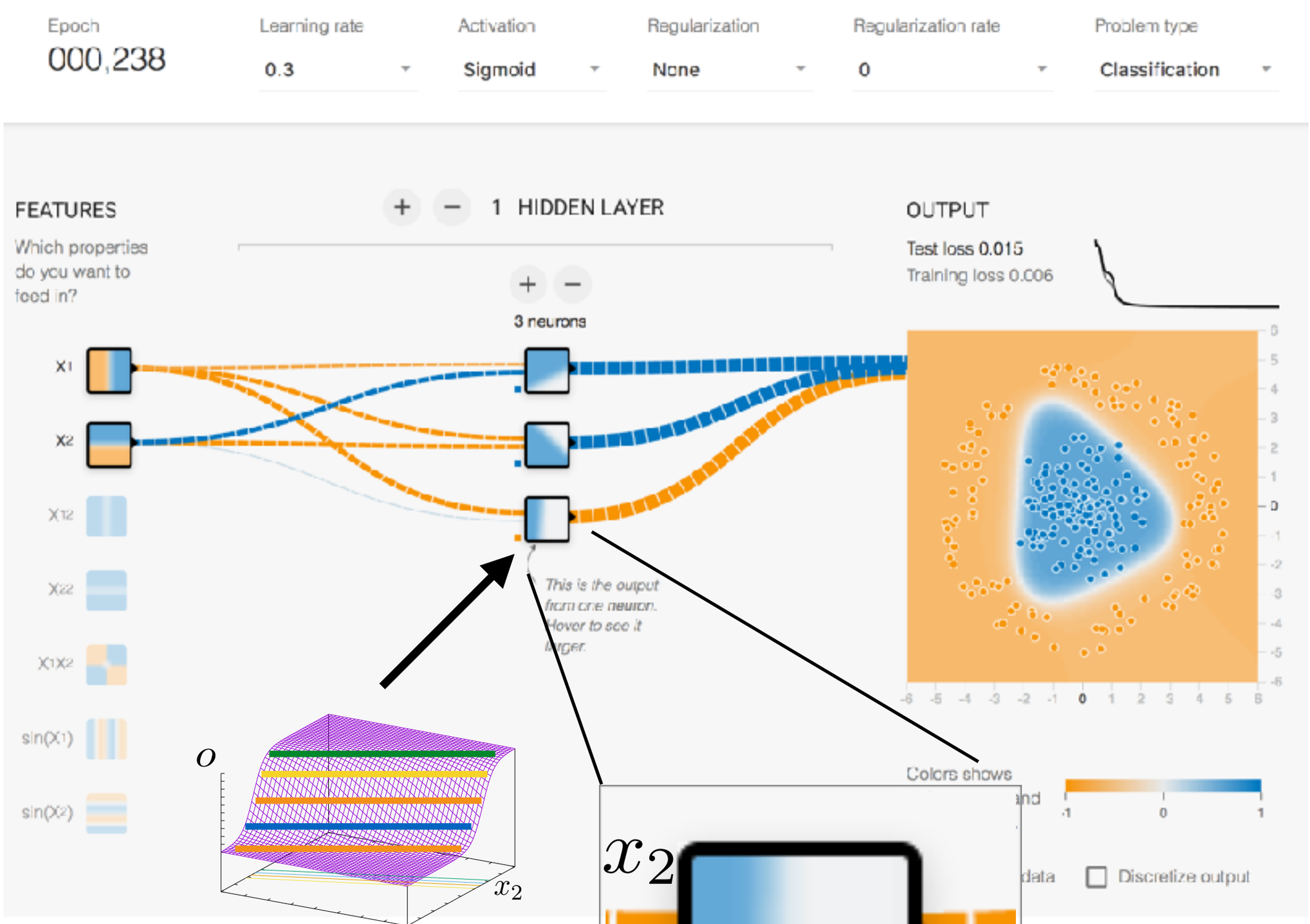
transmitter receiver

Notation

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

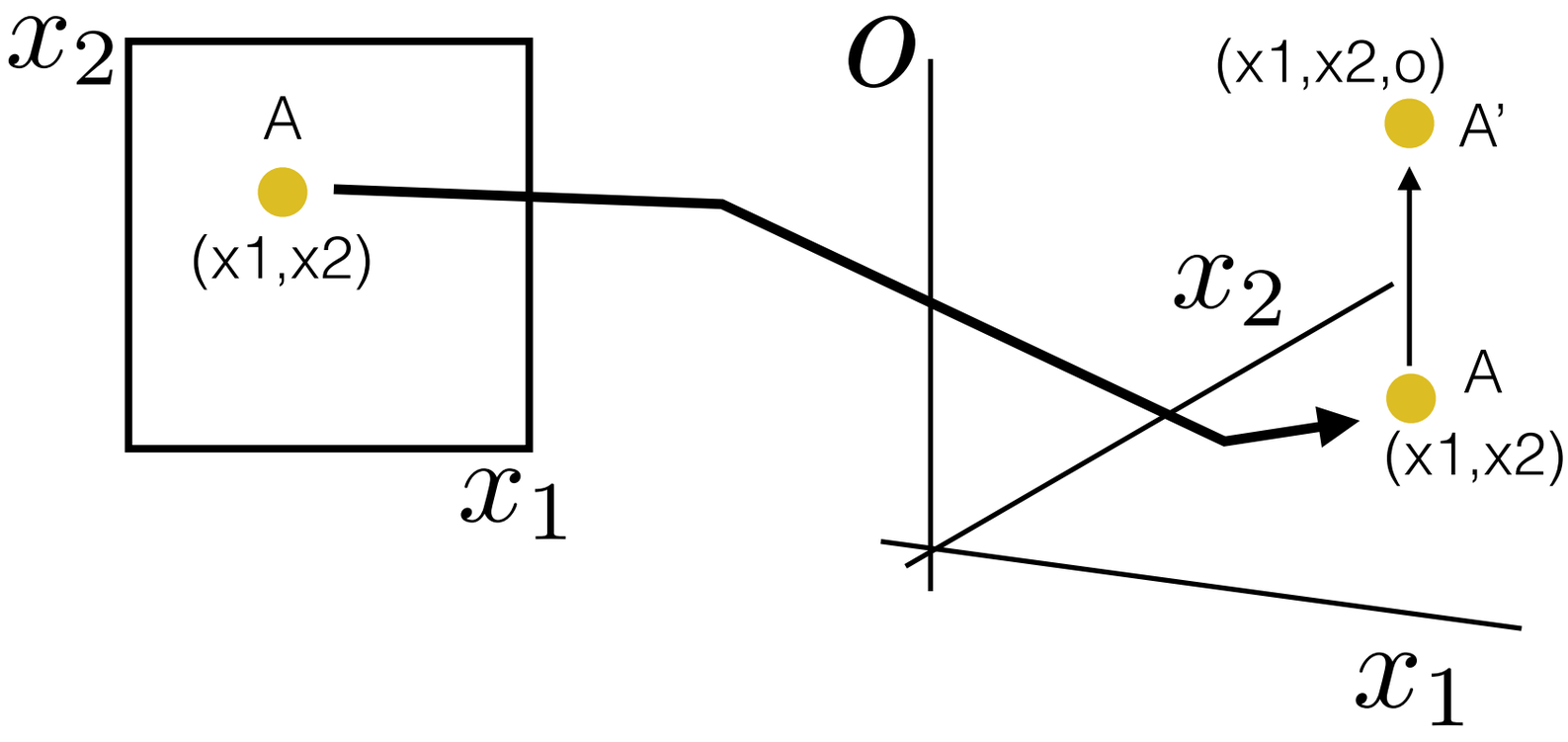
Let $y \in \mathbb{R}$ or $y \in \mathbb{N}$ be the label

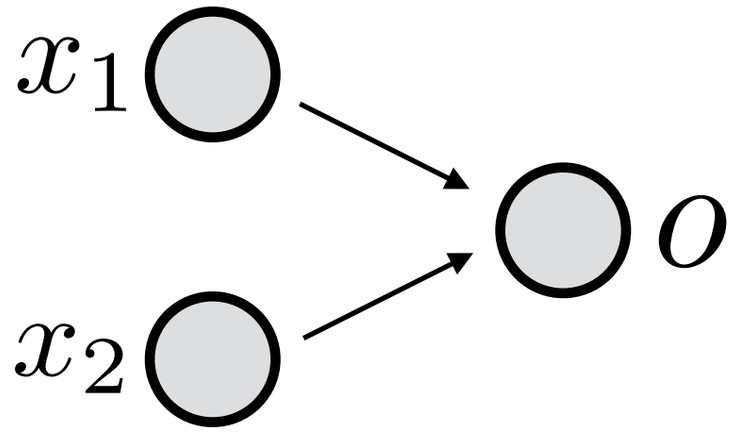
Let $o \in \mathbb{R}$ be the output of the neural network


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

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

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

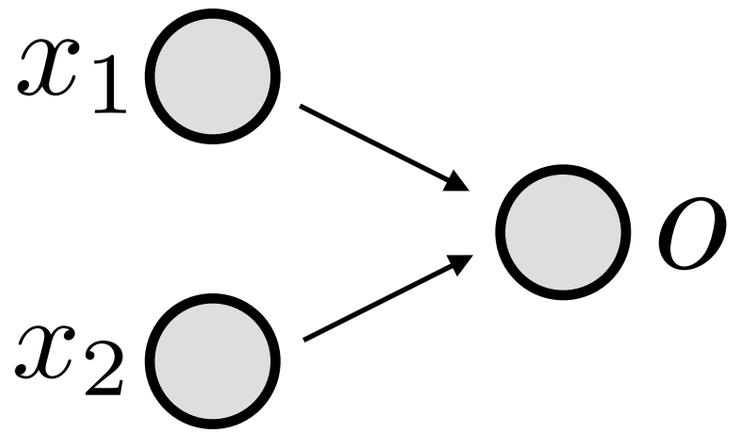


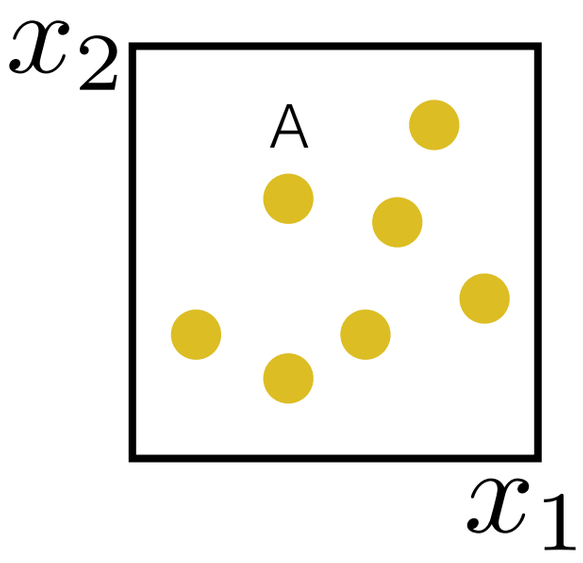


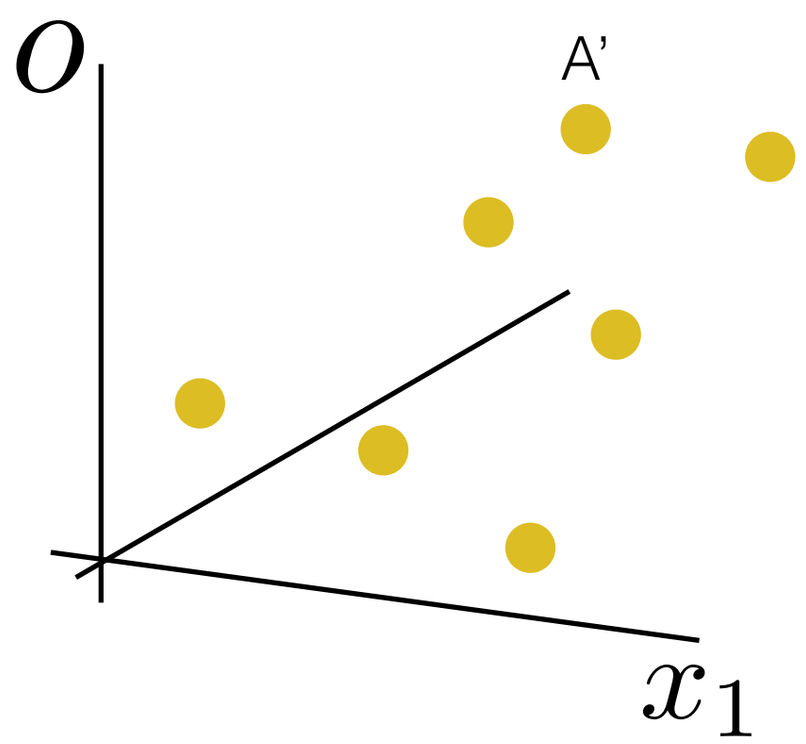
A' = (x1 , x2 , o)

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

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

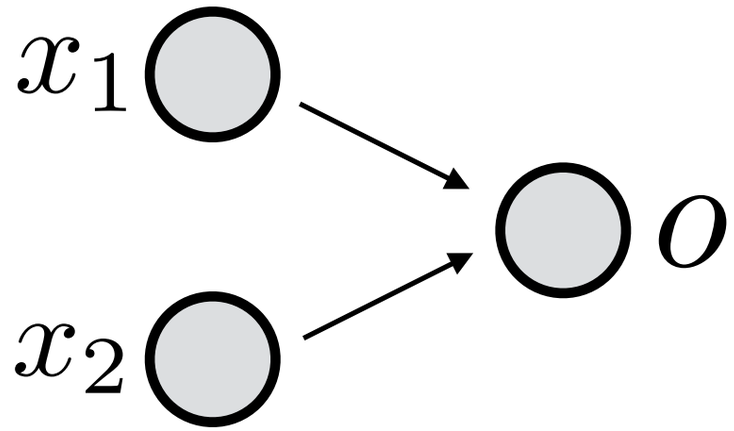


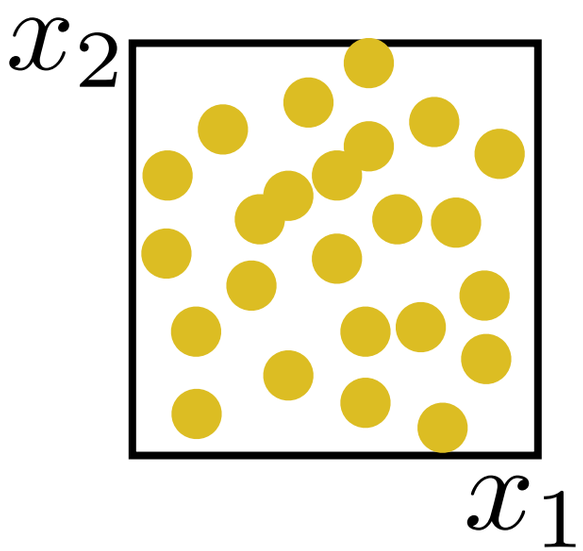


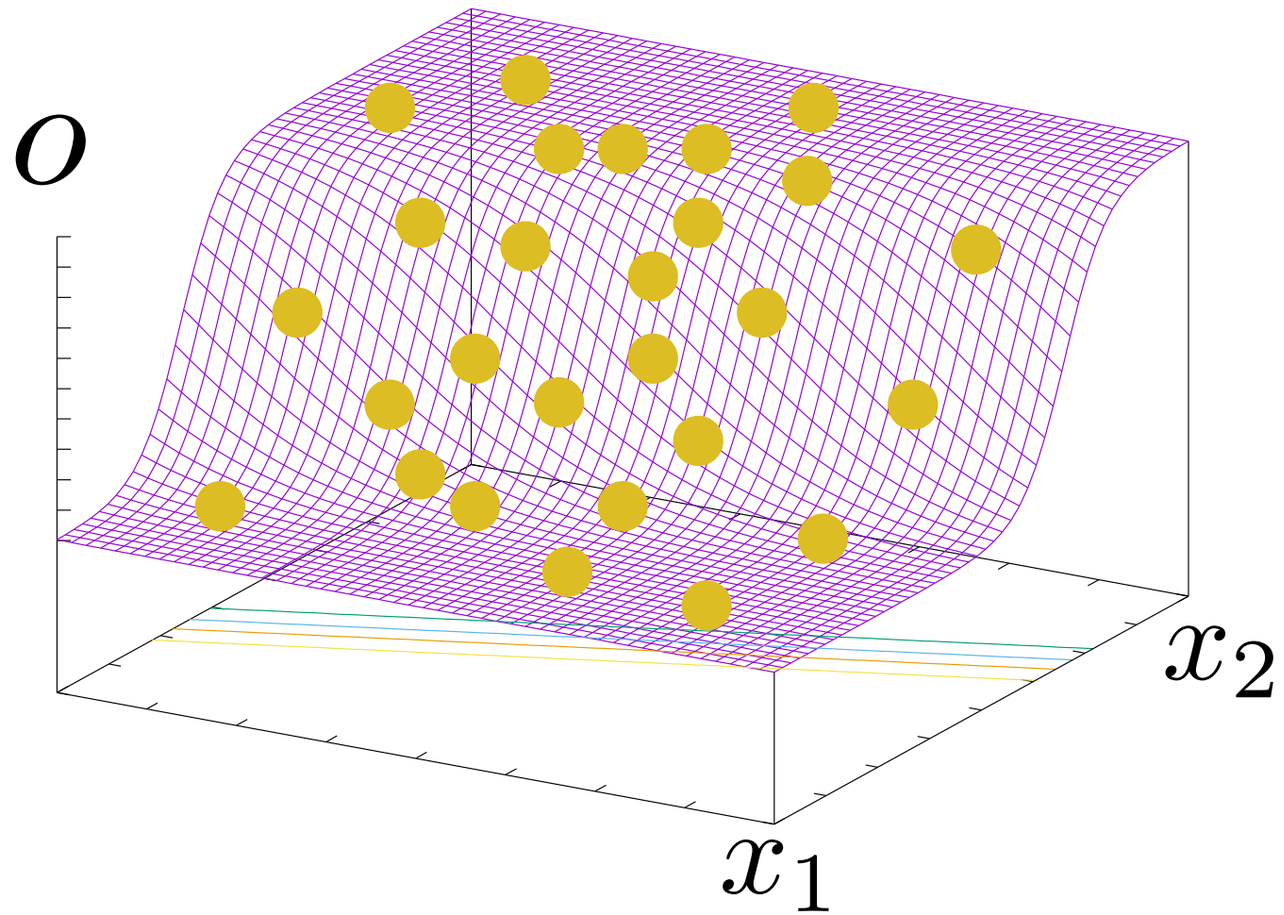


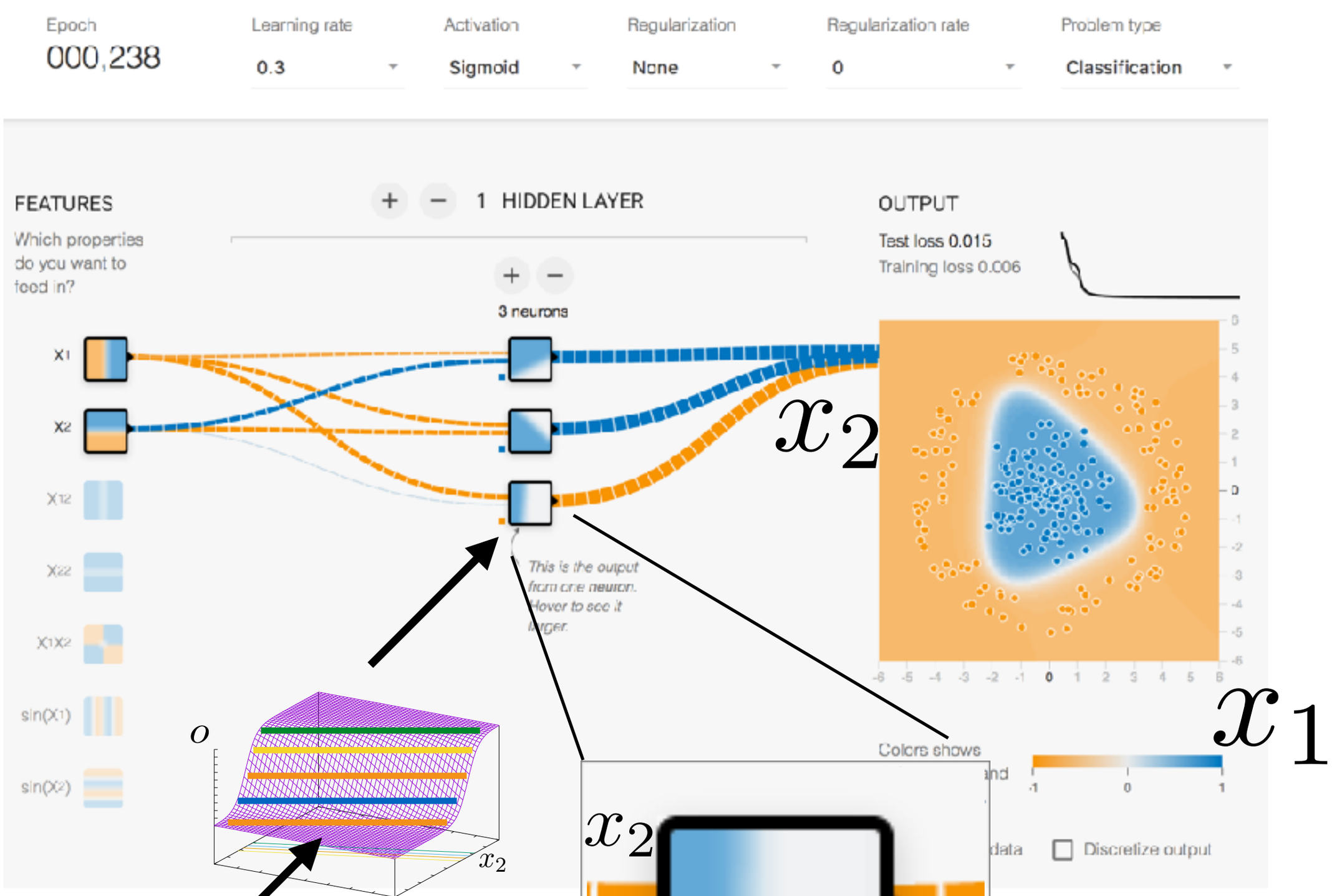
$$x = (x_1, x_2) \in \mathbb{R}^2$$

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



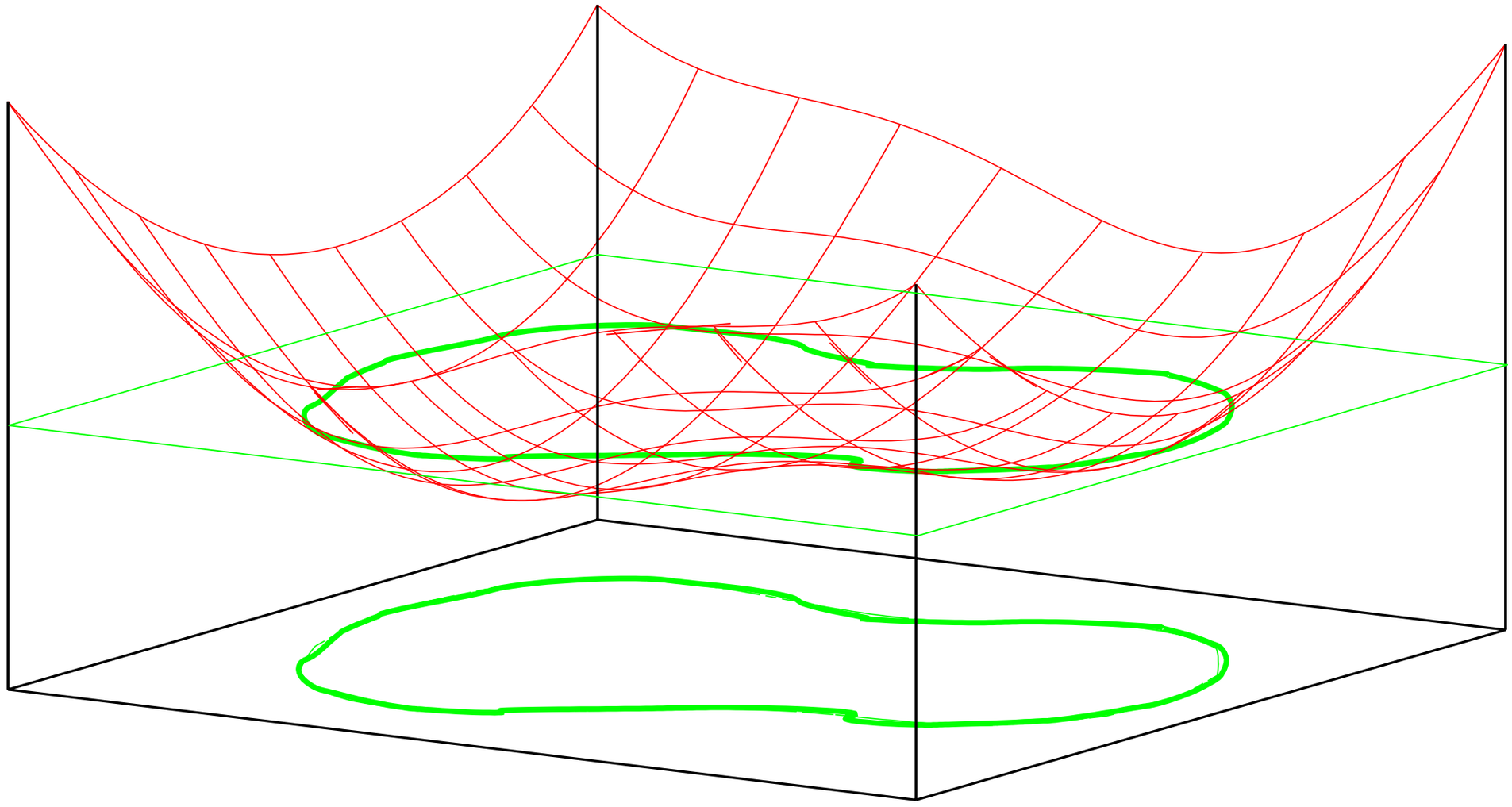






level sets form straight lines

Concept of level sets



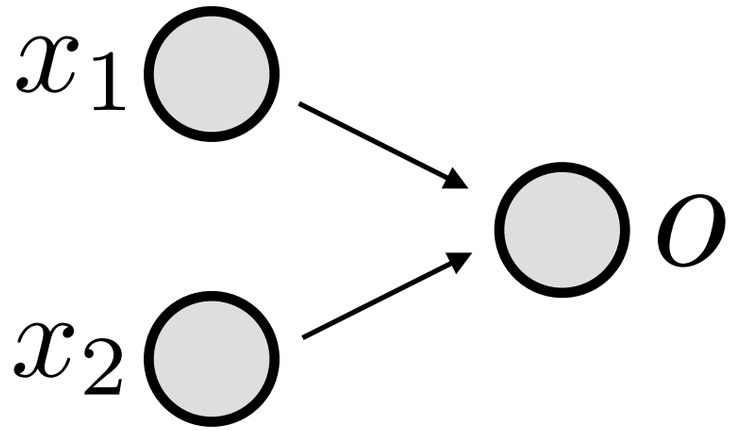
Next to simplest

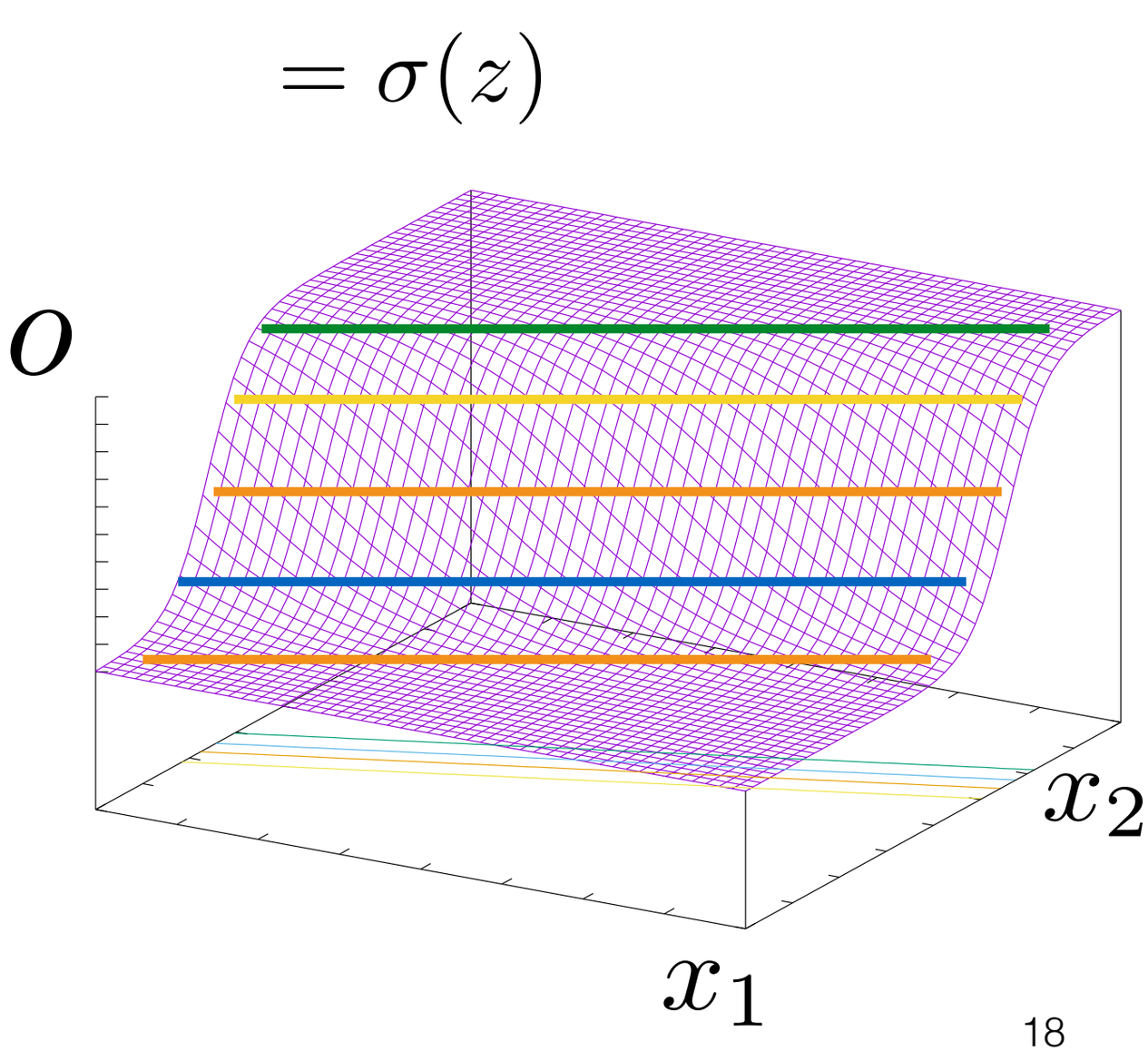
$$x = (x_1, x_2) \in \mathbb{R}^2$$

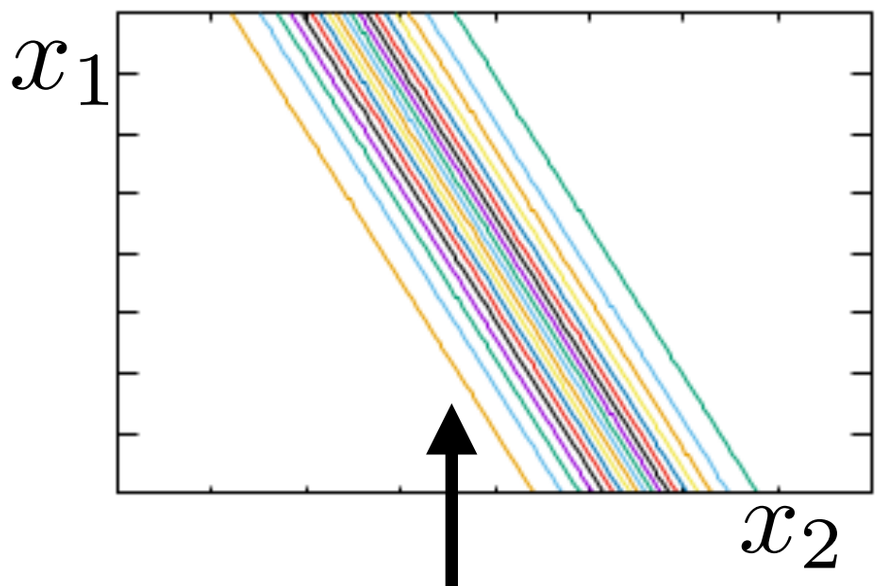
$$z = w_1 x_1 + w_2 x_2 + b$$

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

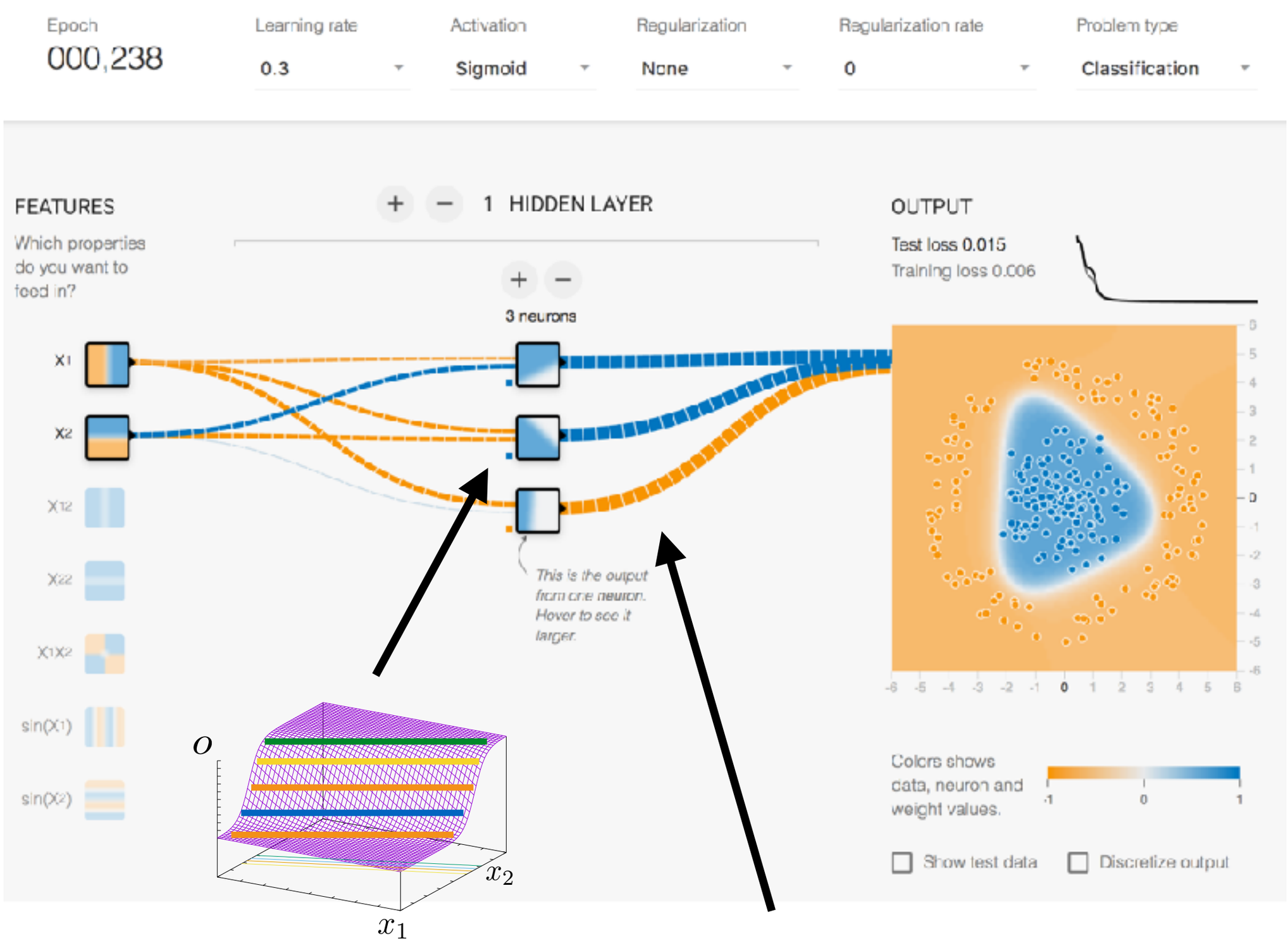
$$= \sigma(z)$$





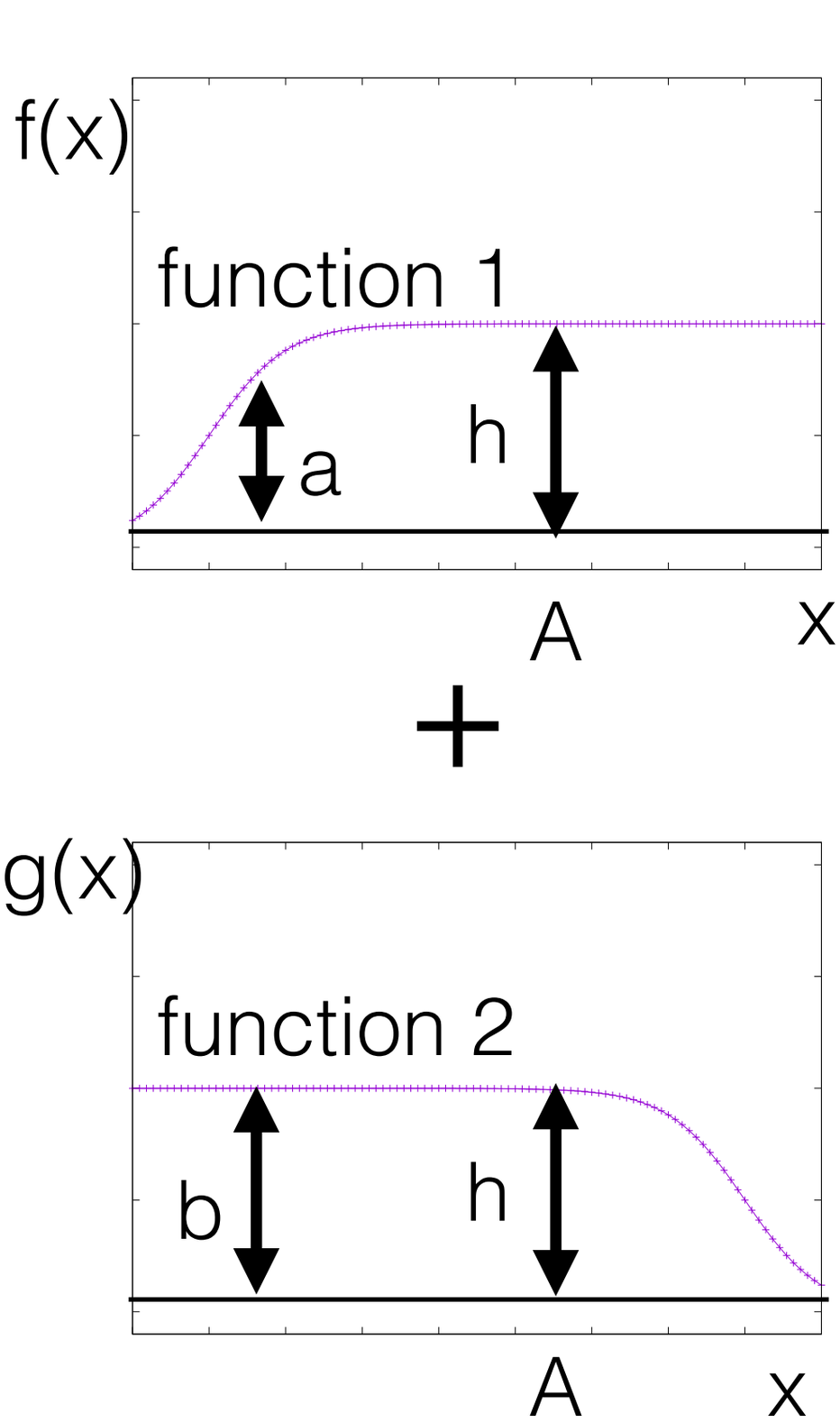


lines of constant z

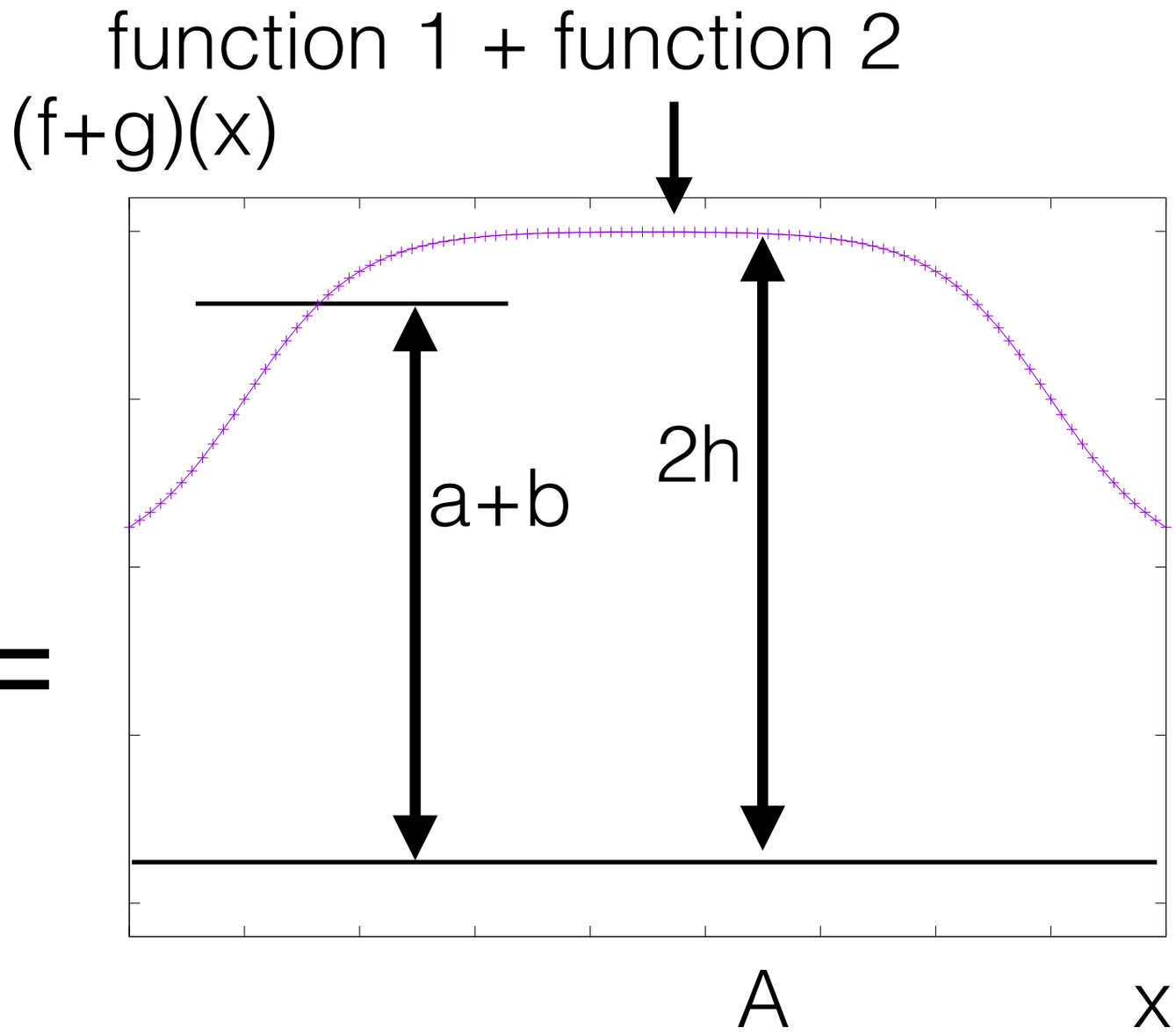


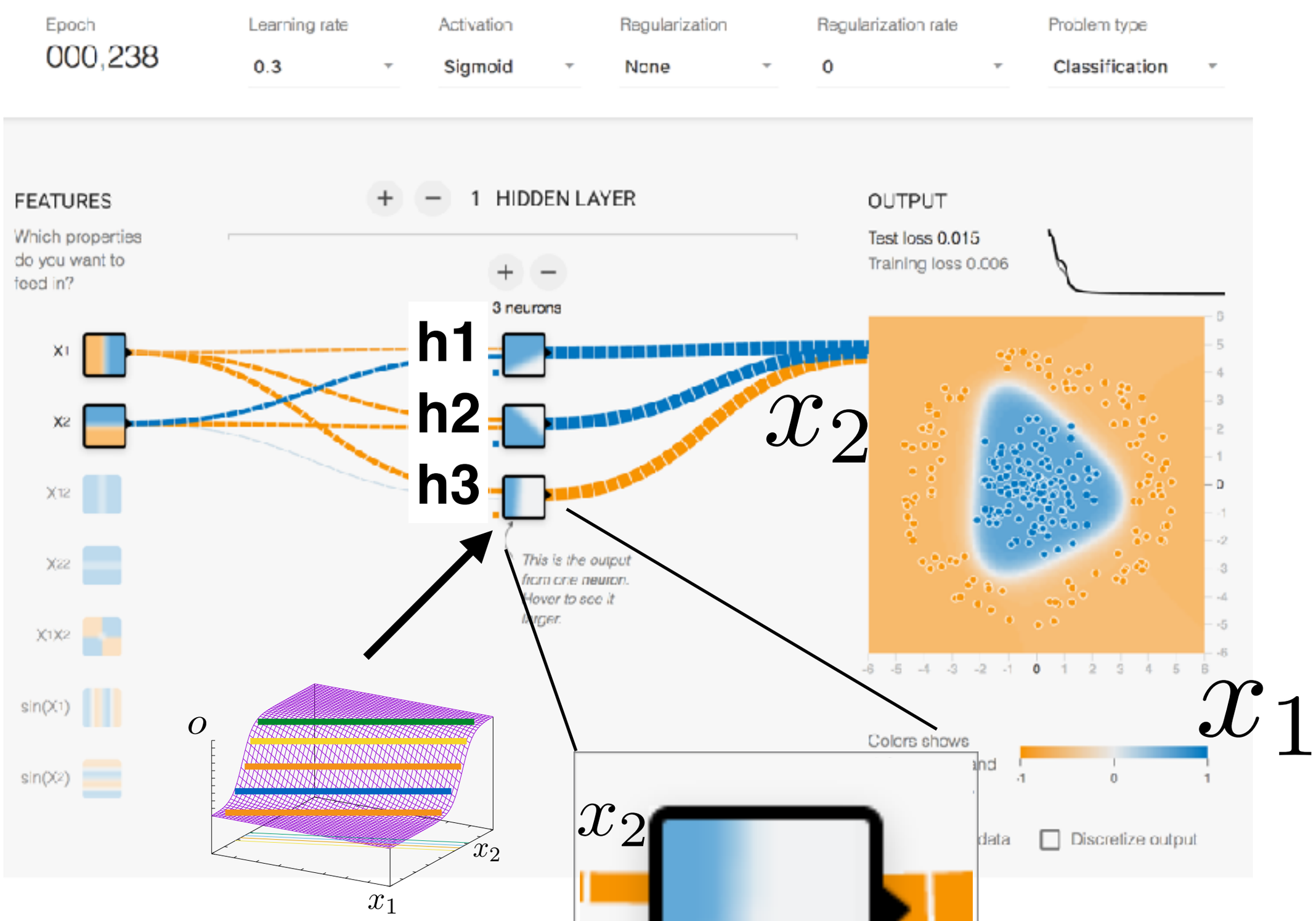
These three surfaces add to form a triangular decision boundary!

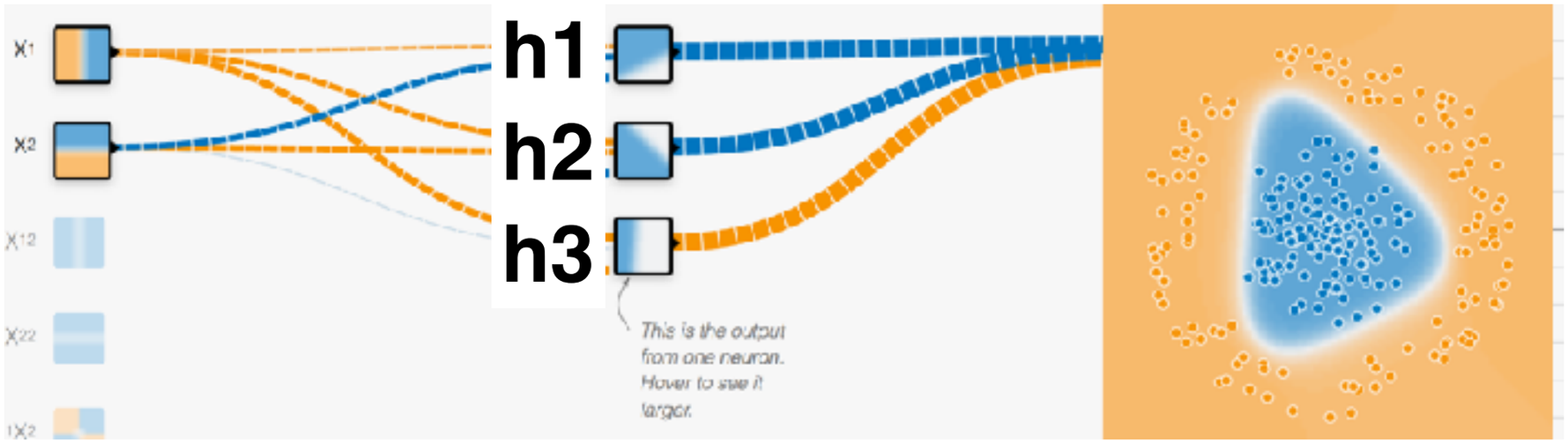
Adding functions



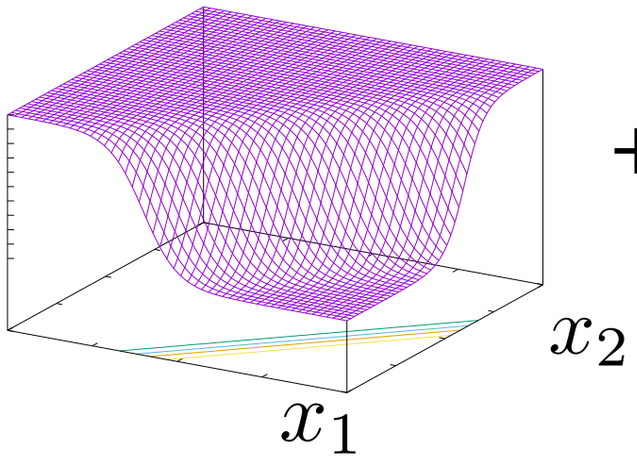
=



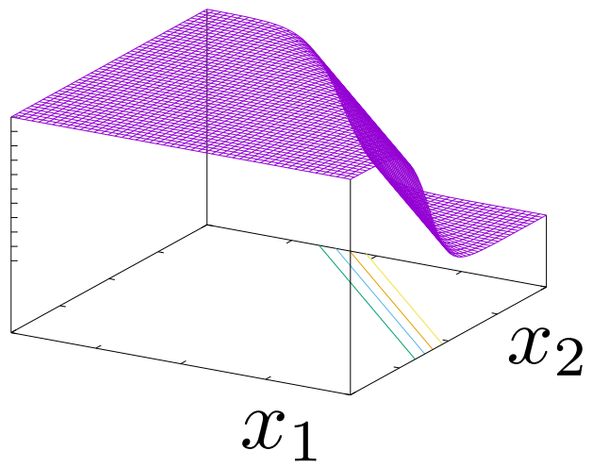

Epoch
000,238
Learning rate
0.3
Activation
Sigmoid
Regularization
None
Regularization rate
0
Problem type
Classification
FEATURES
Which properties do you want to feed in?
1 HIDDEN LAYER
3 neurons
OUTPUT
Test loss 0.015
Training loss 0.006
X1
X2
X12
X22
X1X2
sin(X1)
sin(X2)
h1
h2
h3
This is the output 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



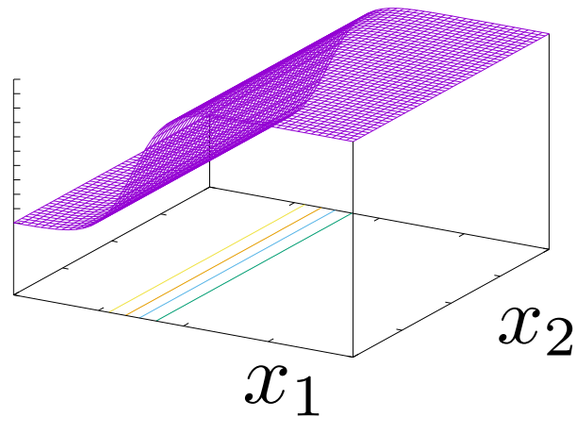
$h_1(x_1, x_2)$



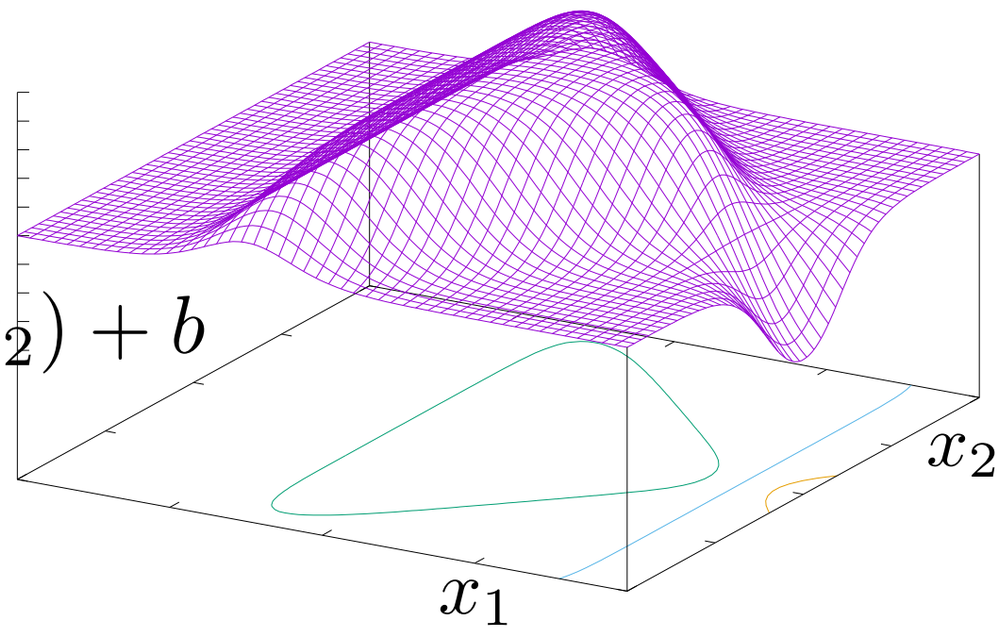
+ $h_2(x_1, x_2)$



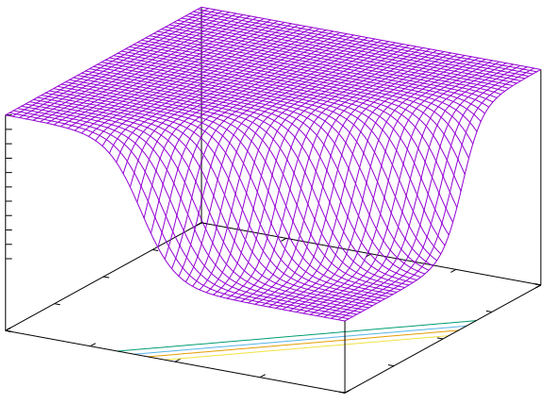
+ $-h_3(x_1, x_2)$

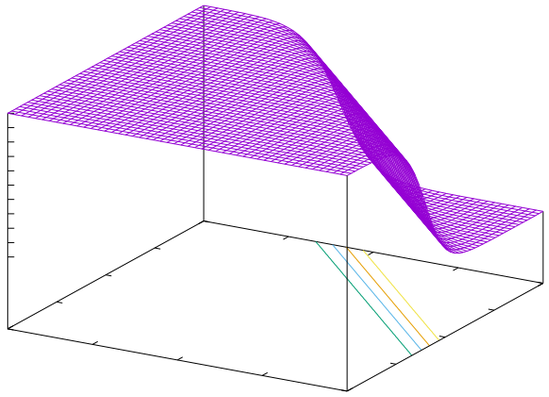


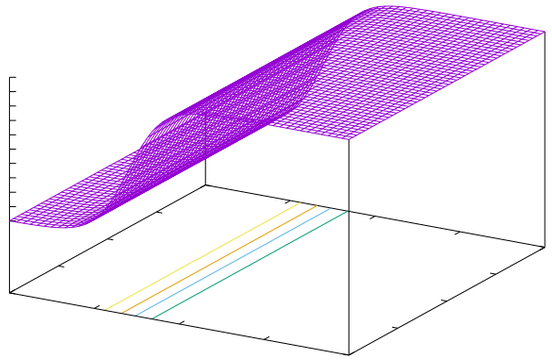
$$w_1 h_1(x_1, x_2) + w_2 h_2(x_1, x_2) + w_3 h_3(x_1, x_2) + b$$

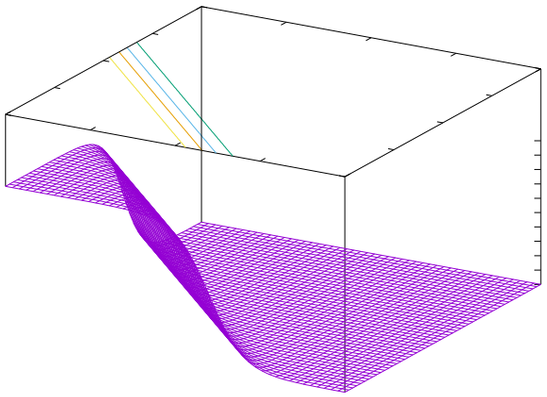


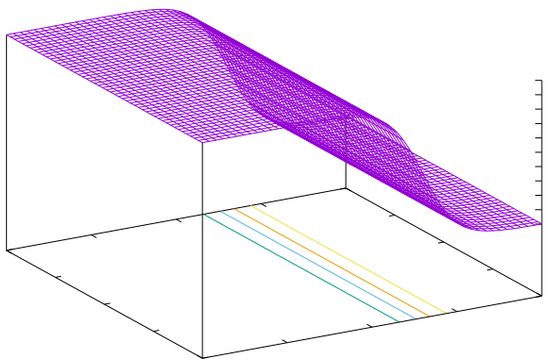
If you add up enough “step” surfaces, are you able to form any functions?

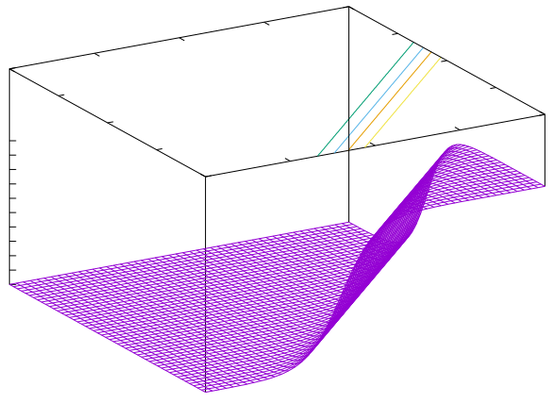


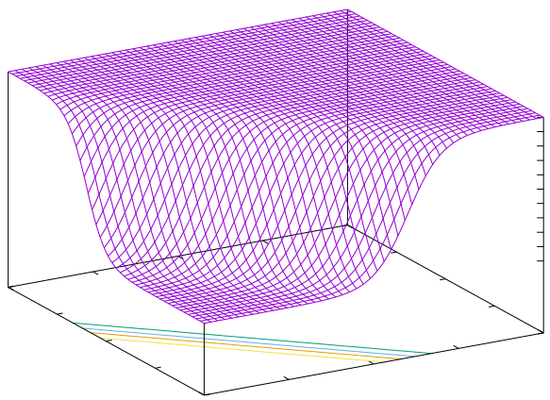


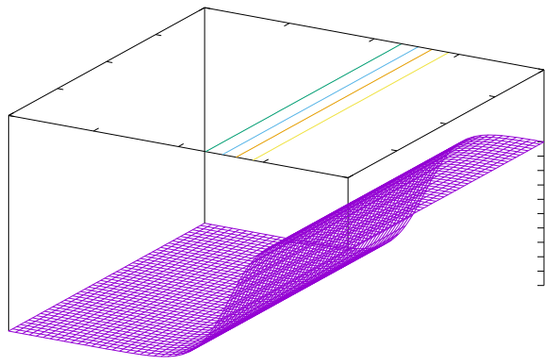


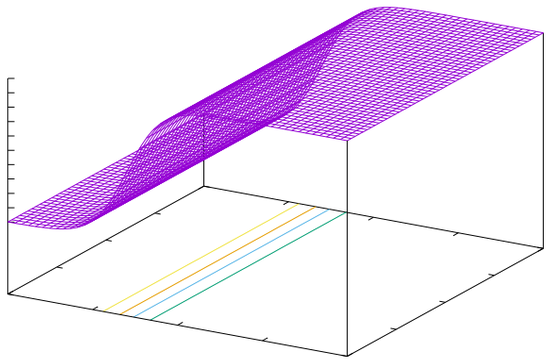








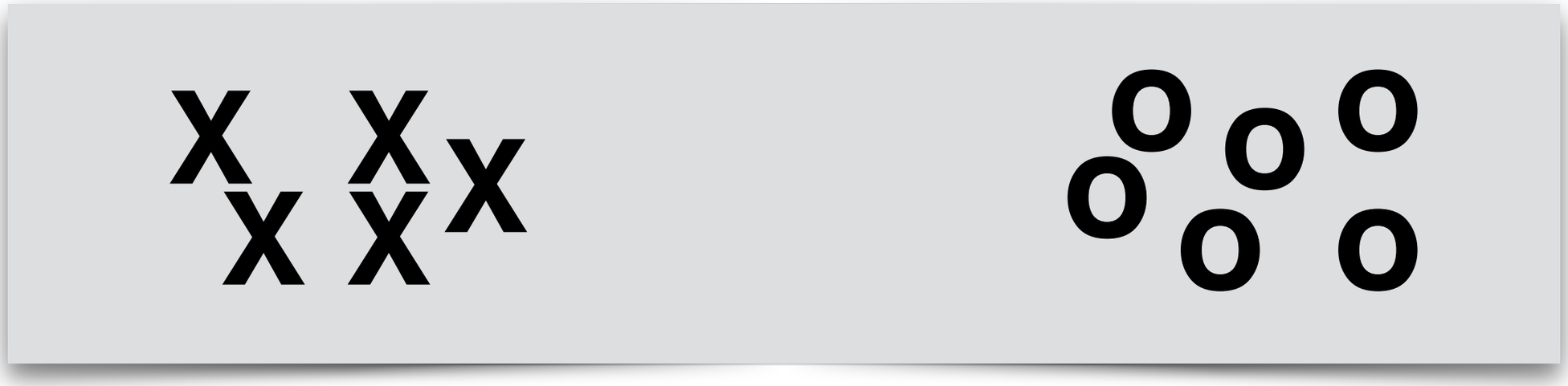




neural network fingers activities

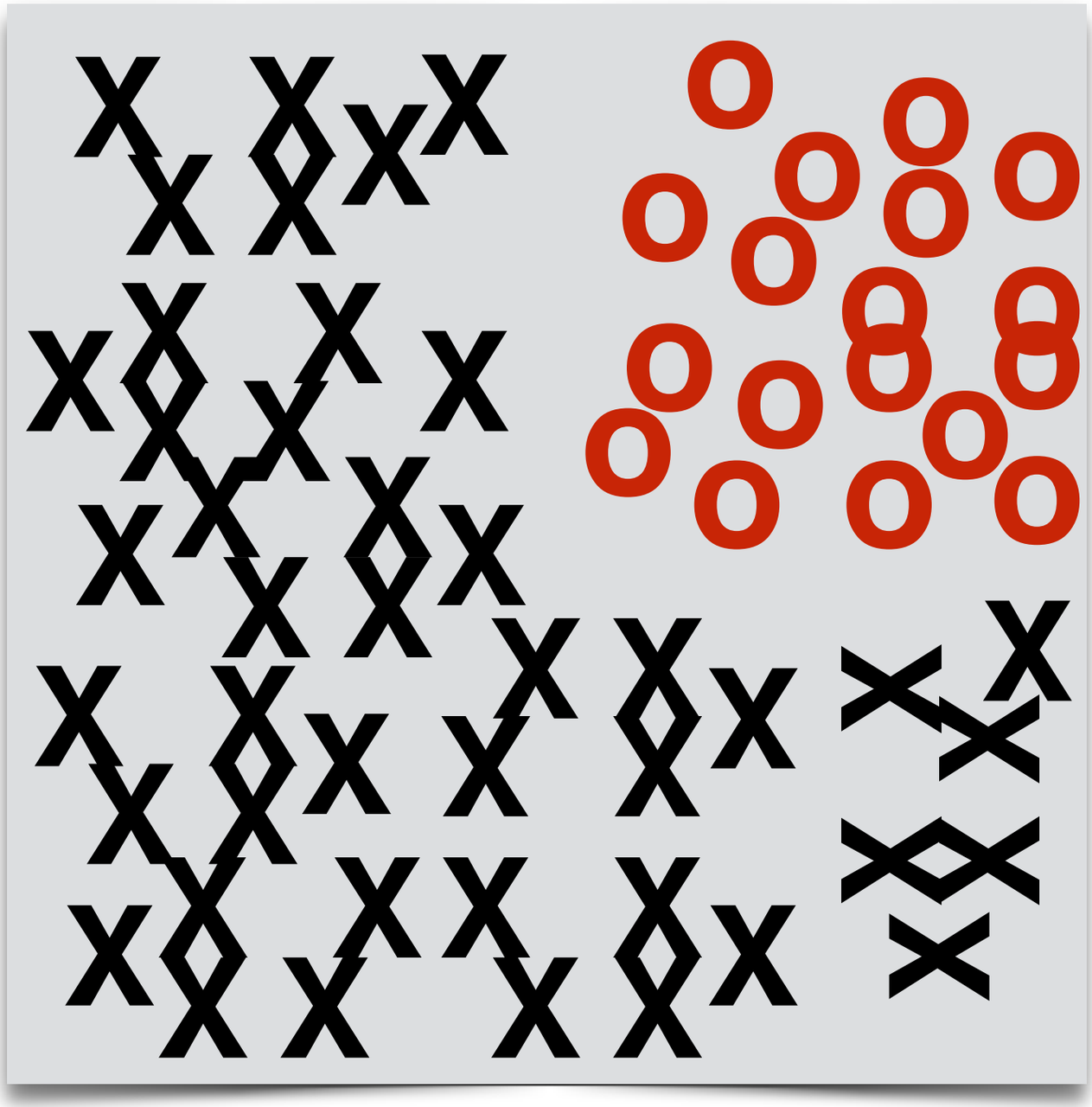
Activity one

Take out a piece of paper, draw patterns as shown



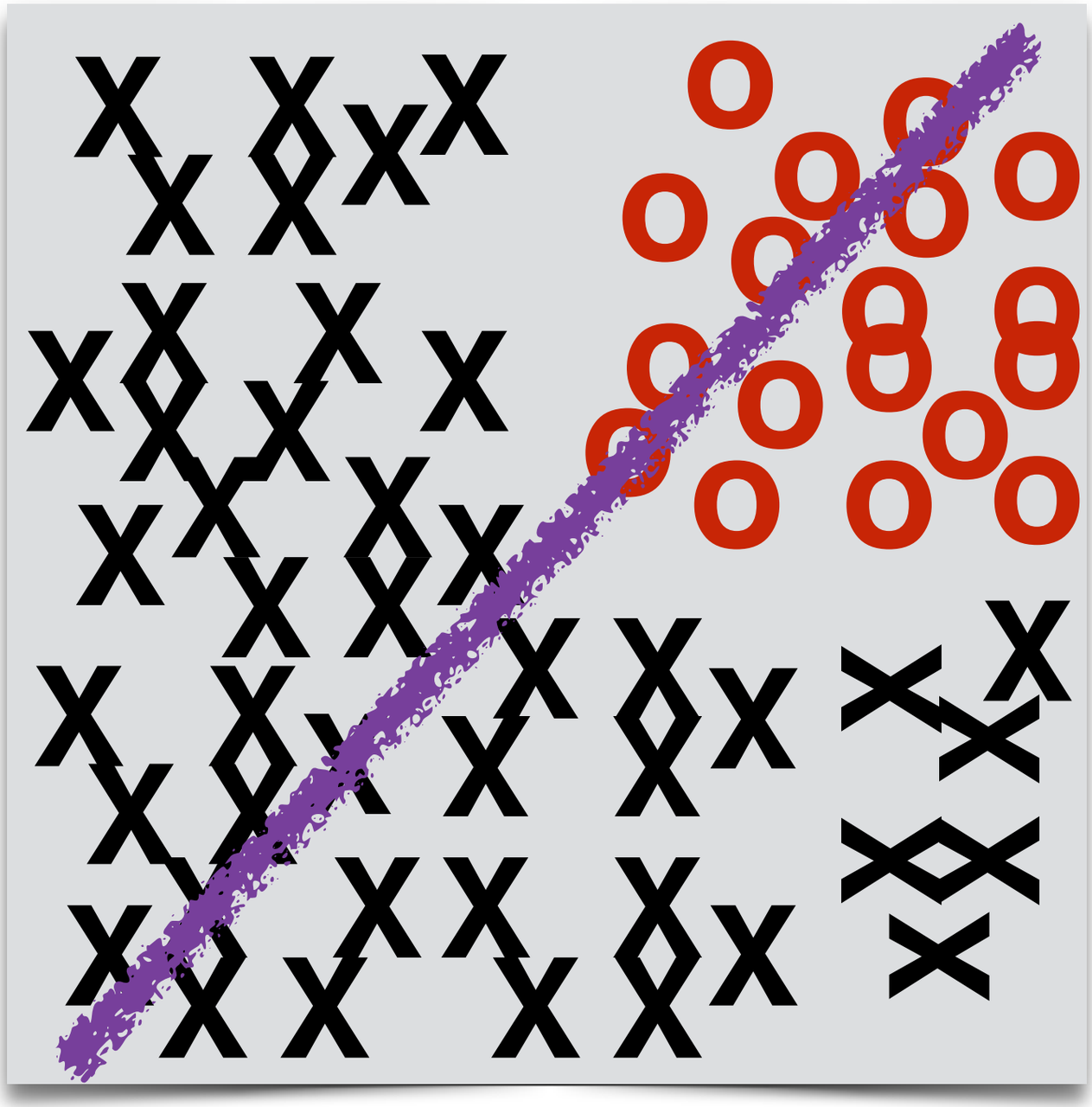
Task: Cut (tear) with one **straight line** to completely separate the “X” and “O”

Activity two



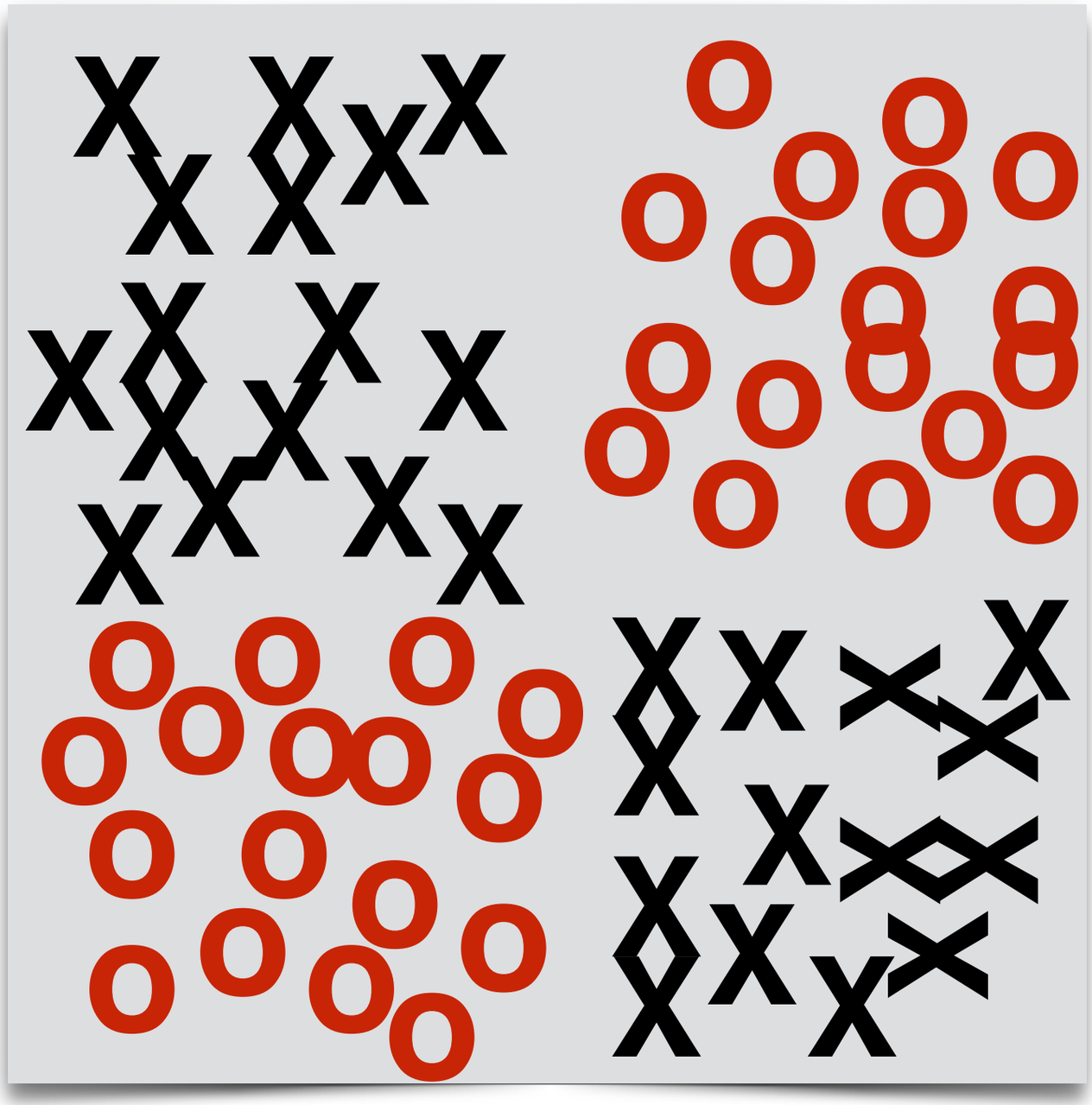
Cut (tear) with **one** straight line!

Activity two



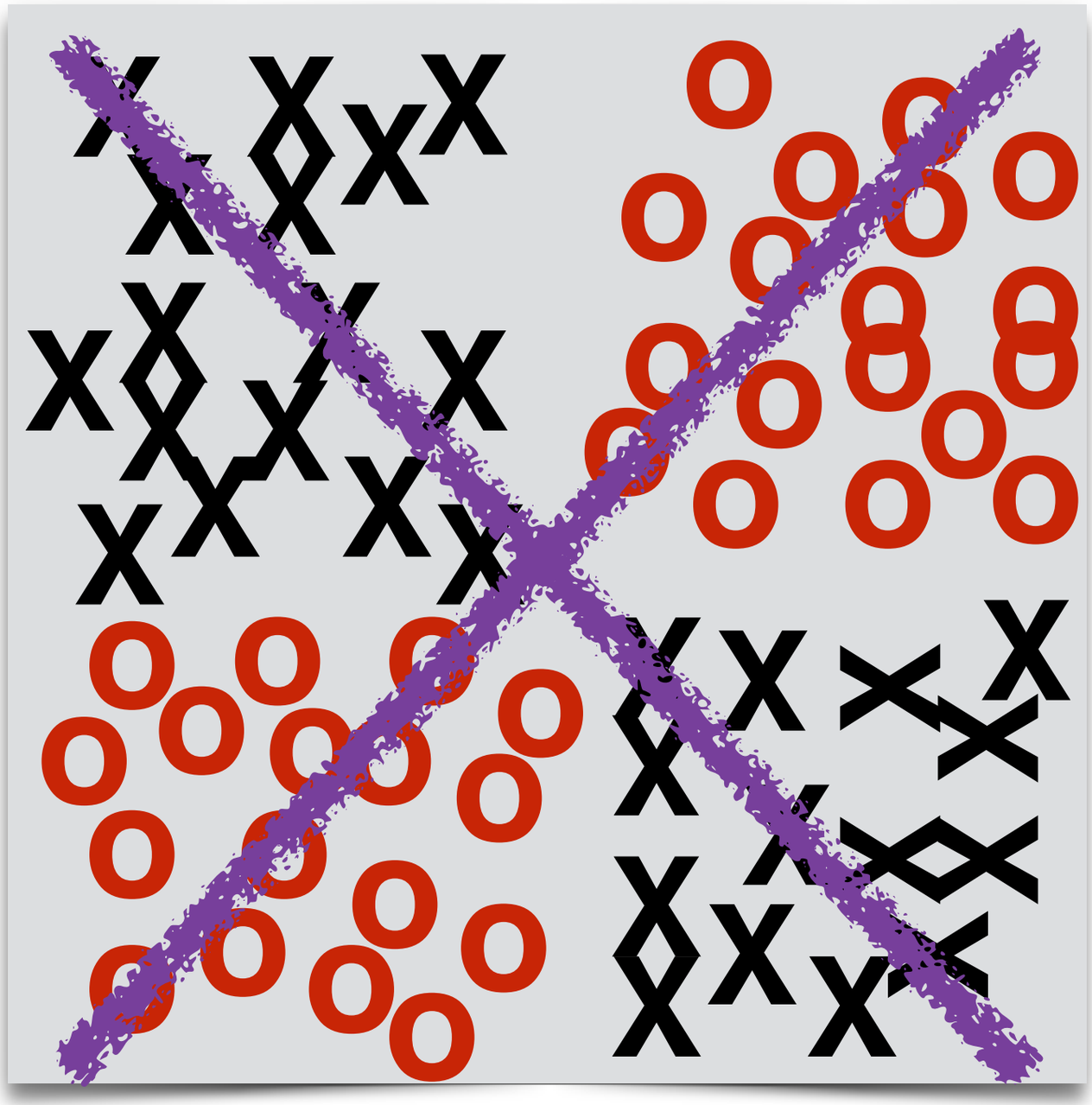
Cut (tear) with **one** straight line!

Activity three



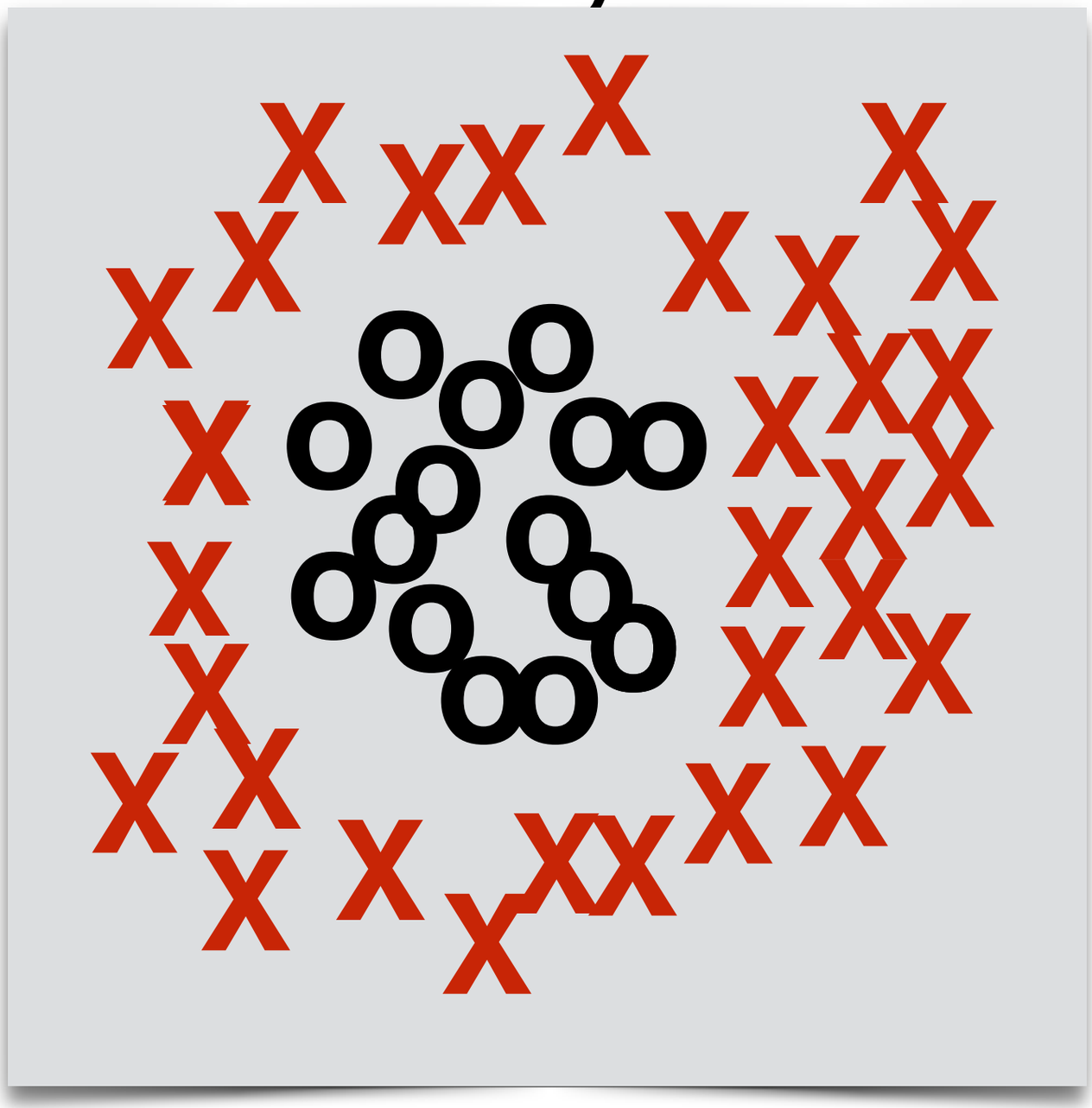
Cut (tear) with **one** straight line!

Activity three



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

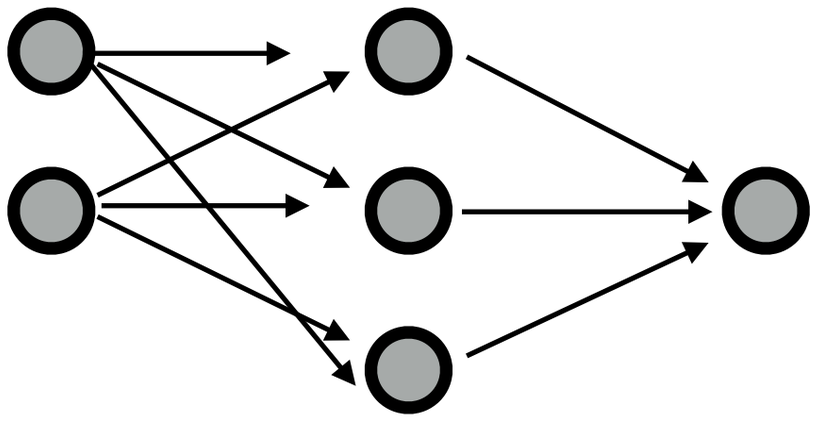
Activity four



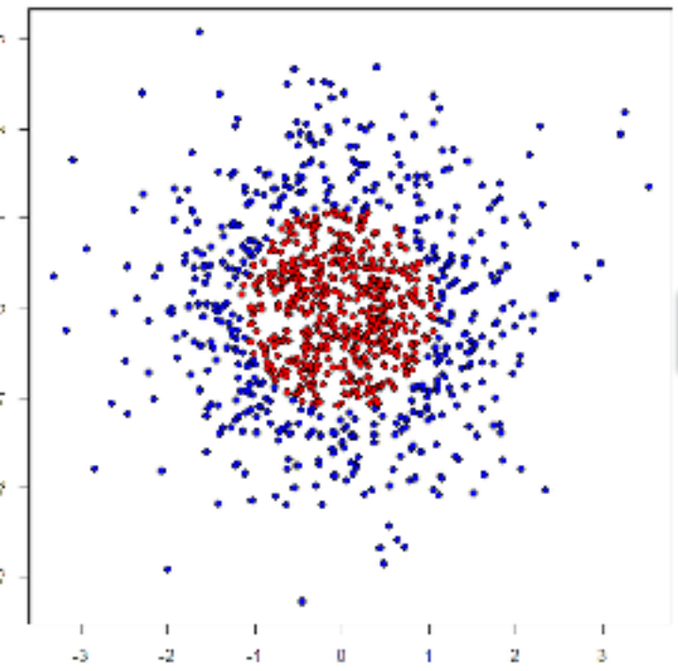
Cut (tear) with **one** straight line!

Manifold view of neural network

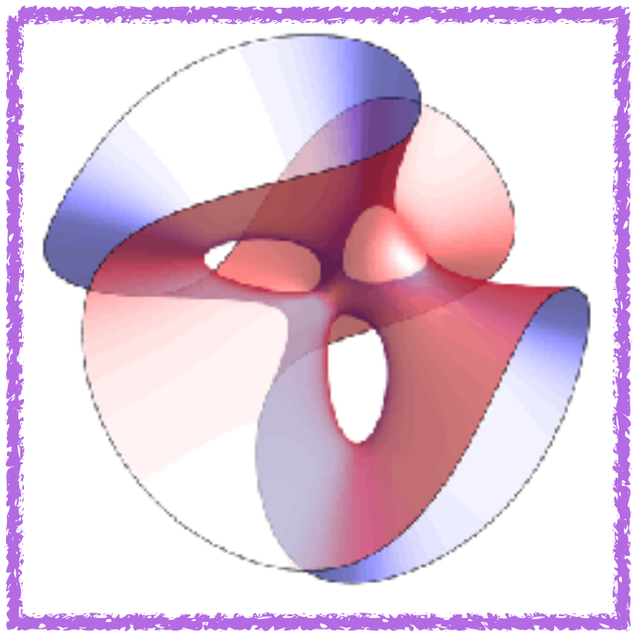
feed in data into first layer



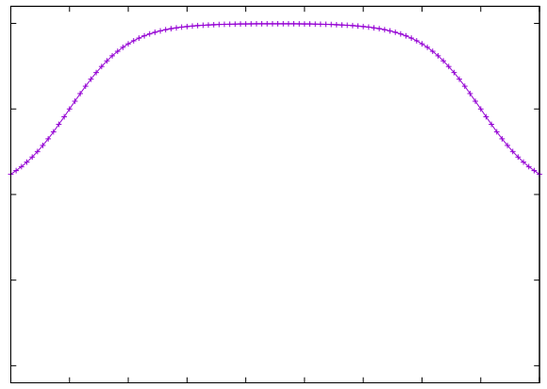
final layer presents the output of network



data



hidden layer

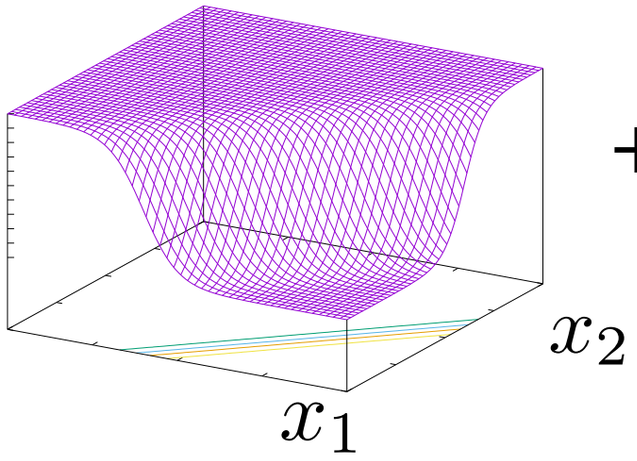


output

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

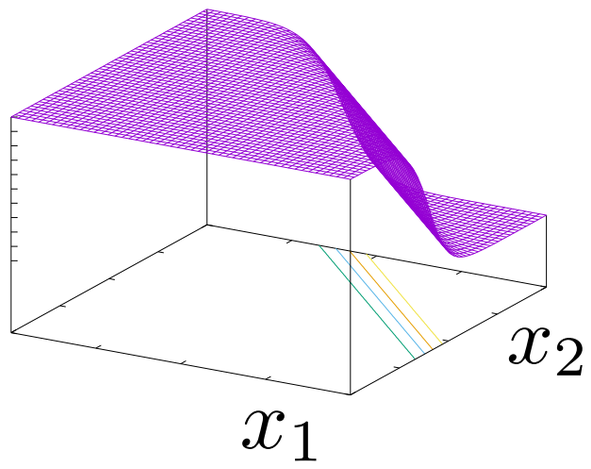
Function view of neural network

$h_1(x_1, x_2)$



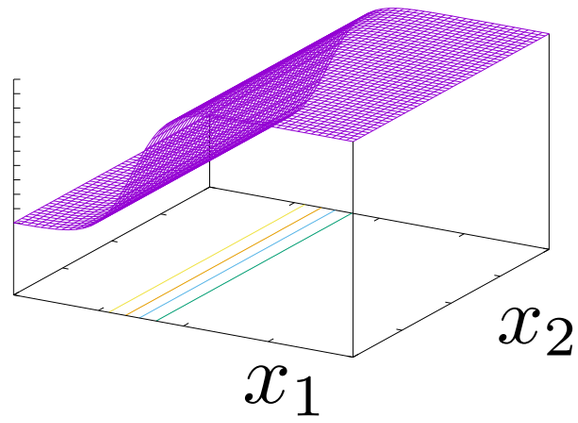
+

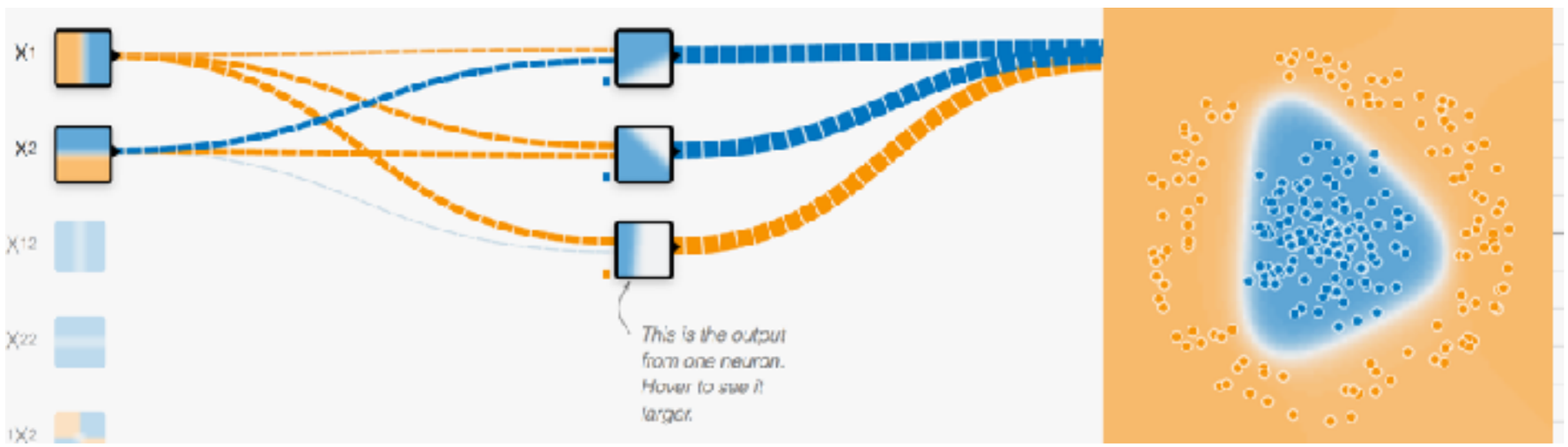
$h_2(x_1, x_2)$

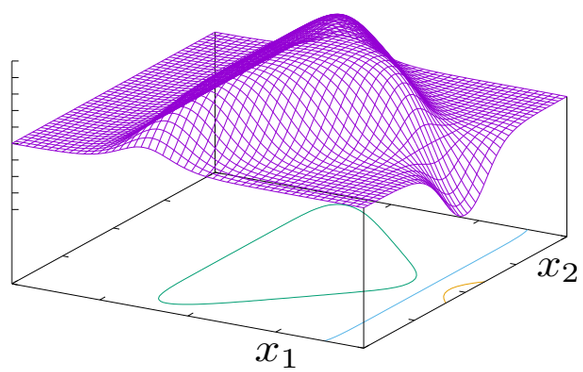


+

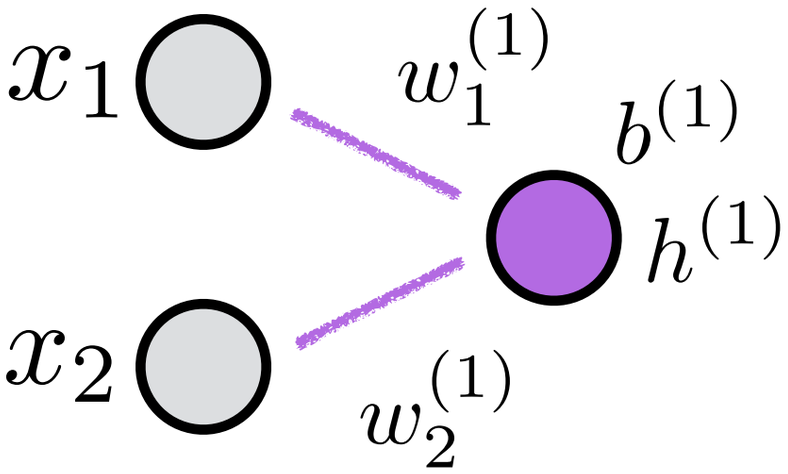
$-h_3(x_1, x_2)$

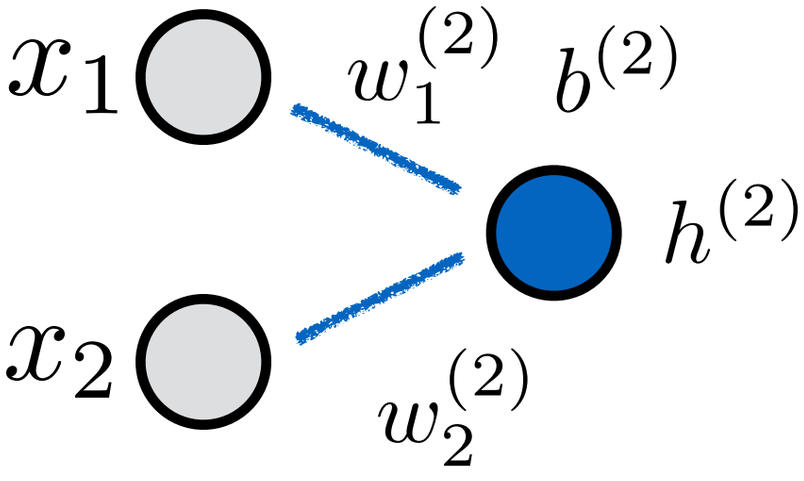


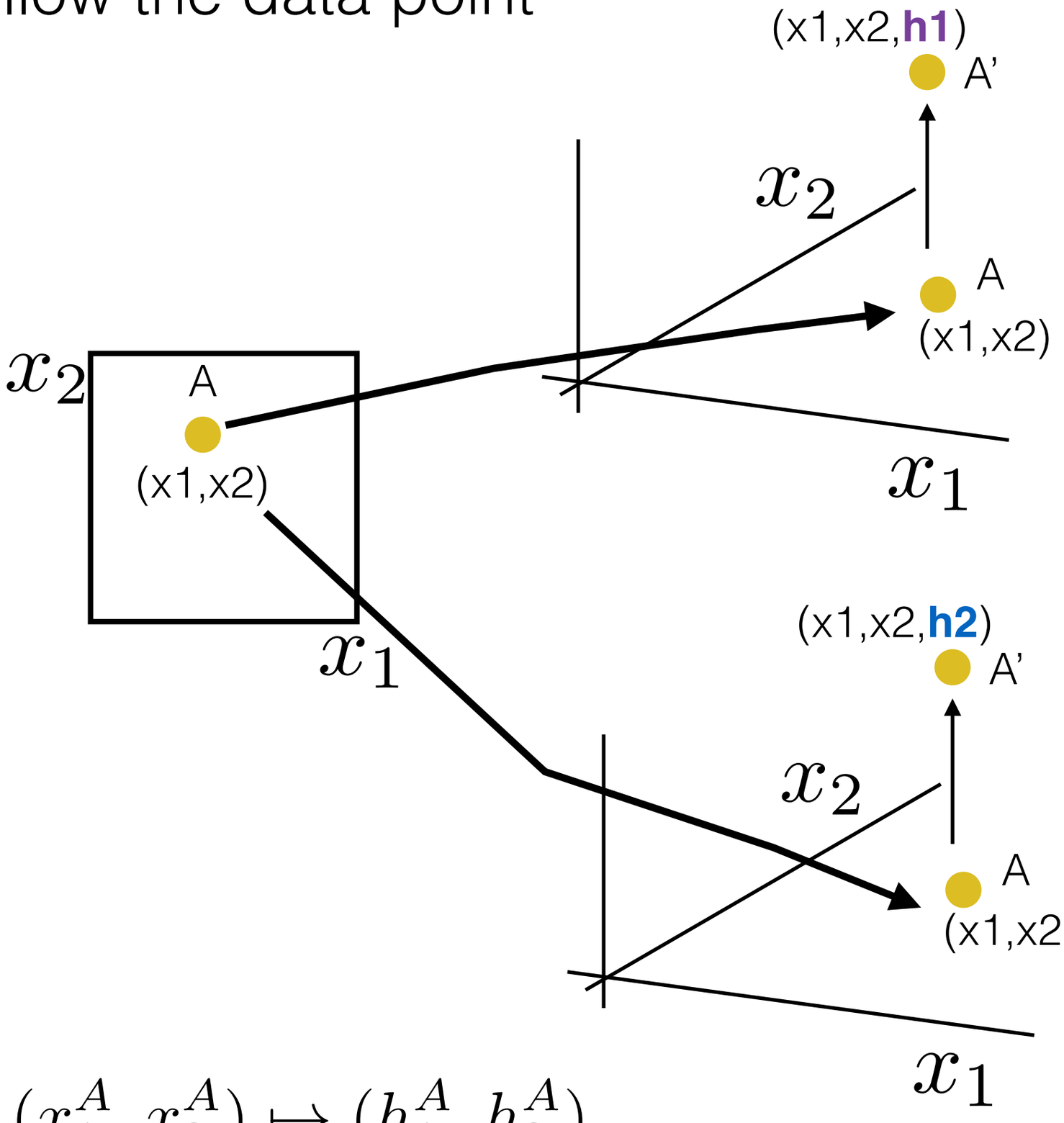




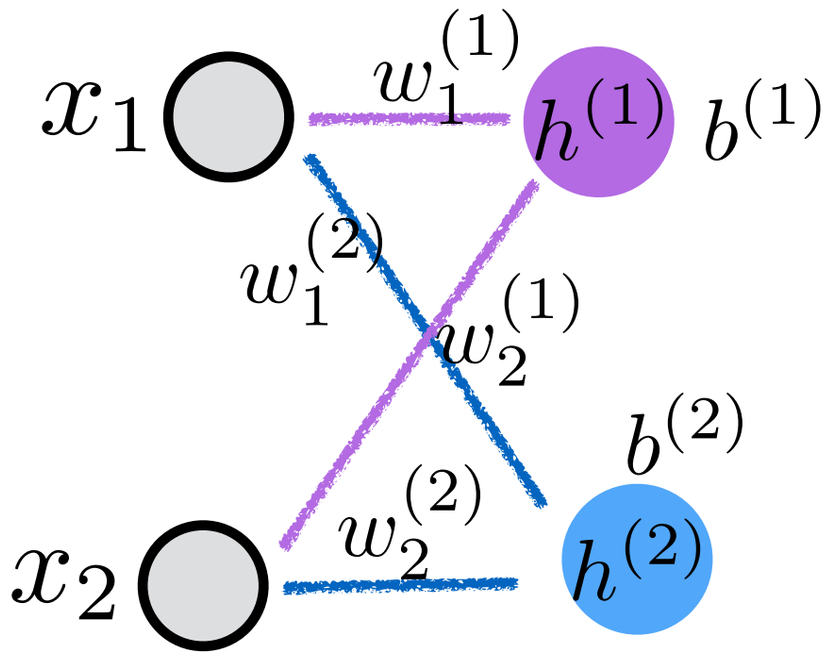
Follow the data point

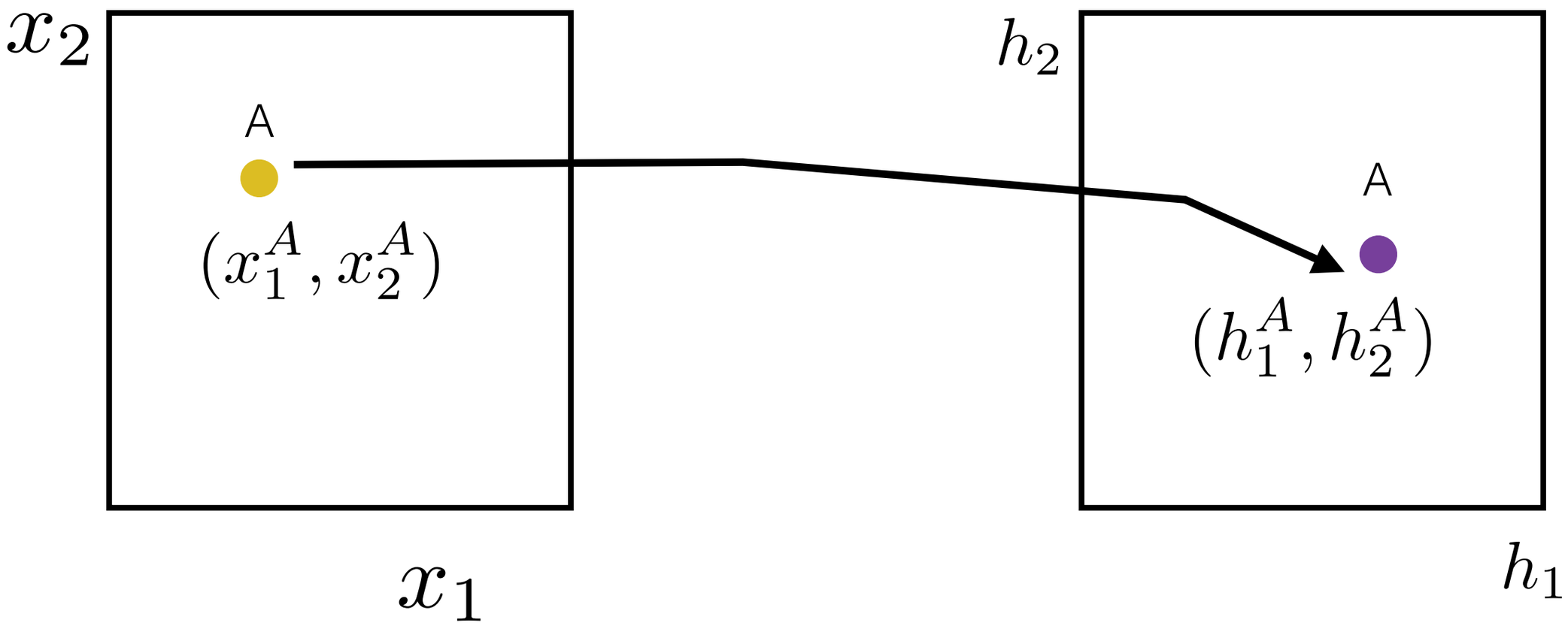




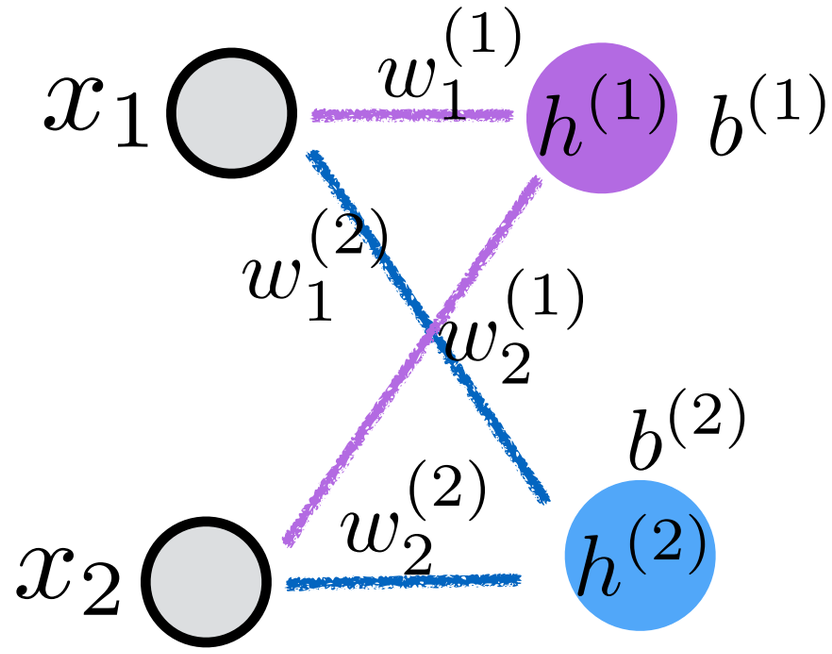


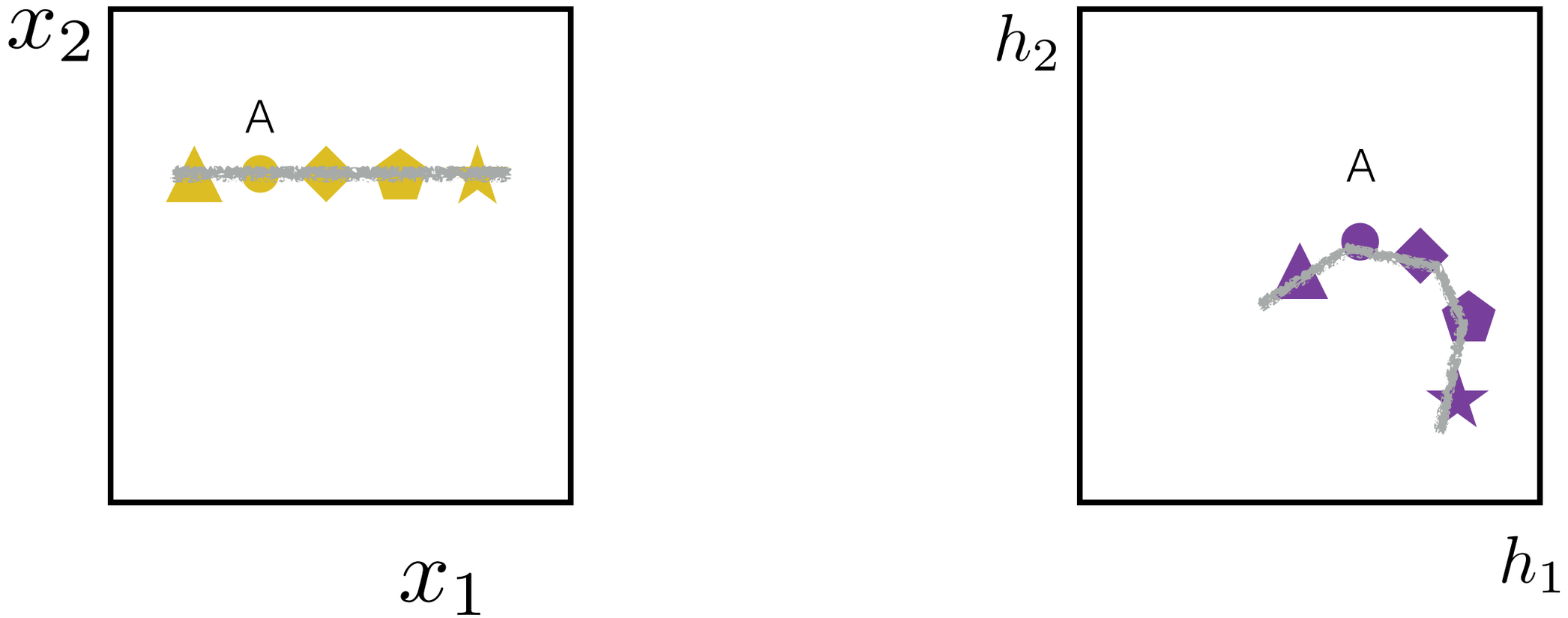
$$(x_1^A, x_2^A) \mapsto (h_1^A, h_2^A)$$



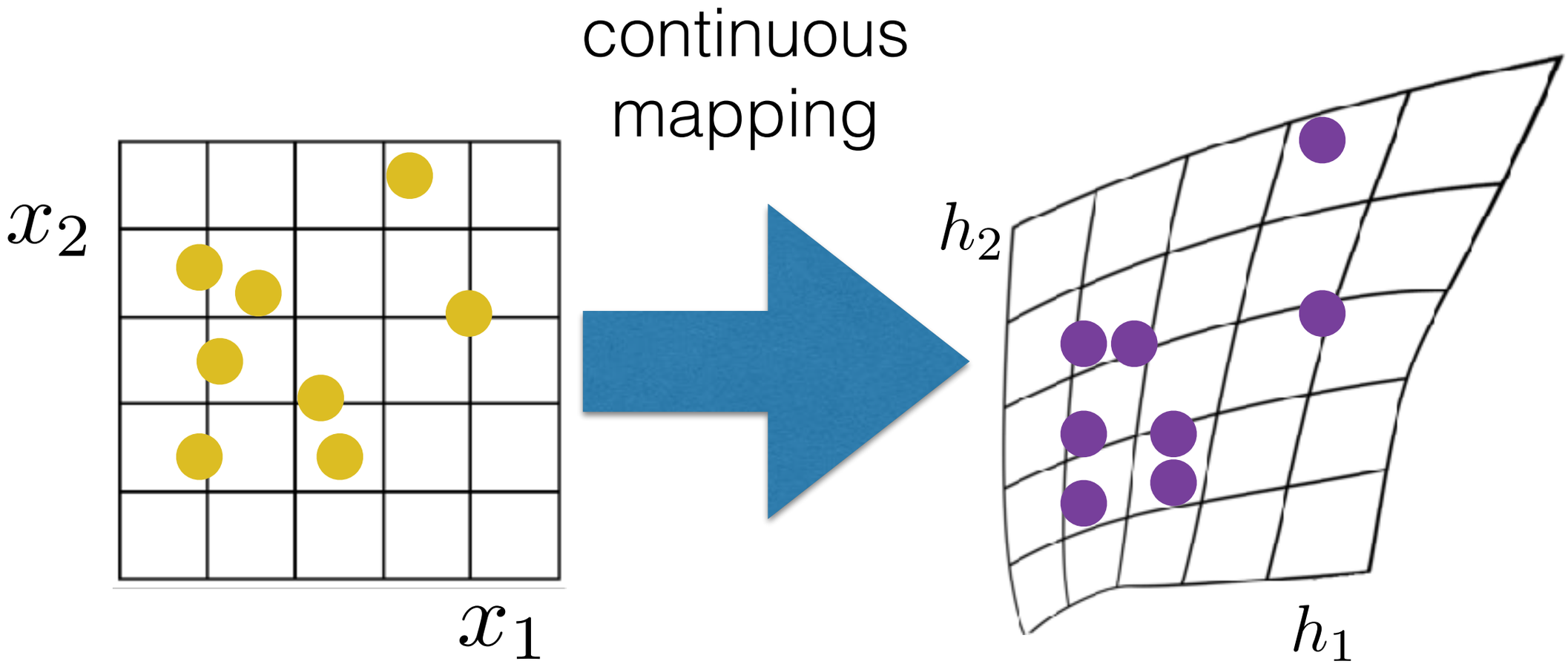


$$(x_1^A, x_2^A) \mapsto (h_1^A, h_2^A)$$

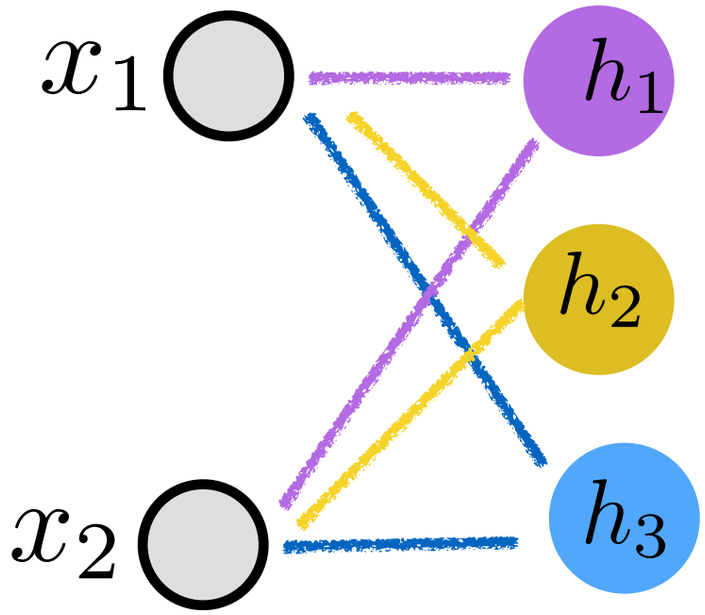


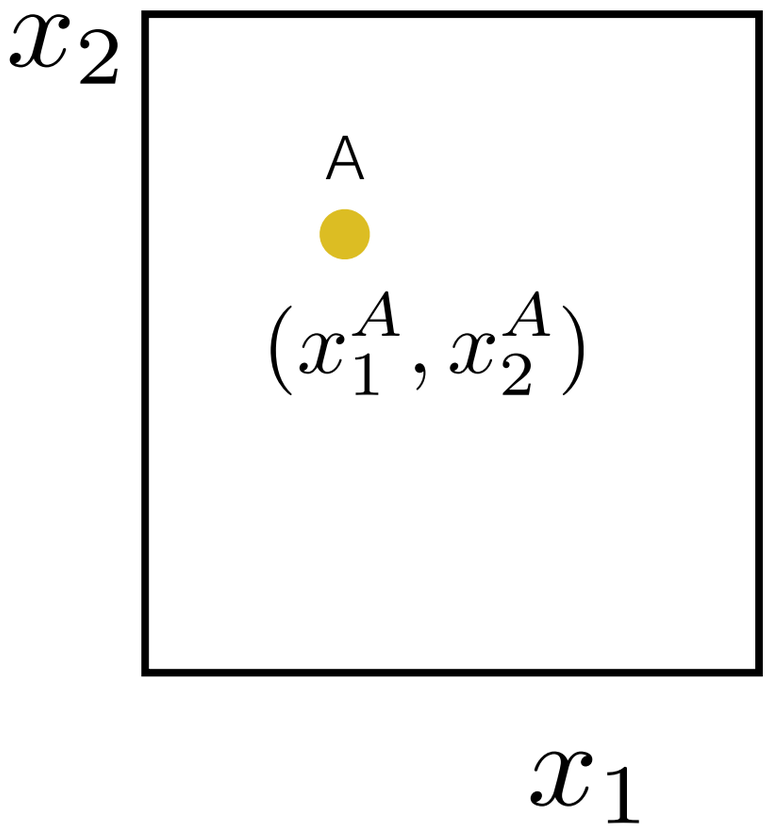


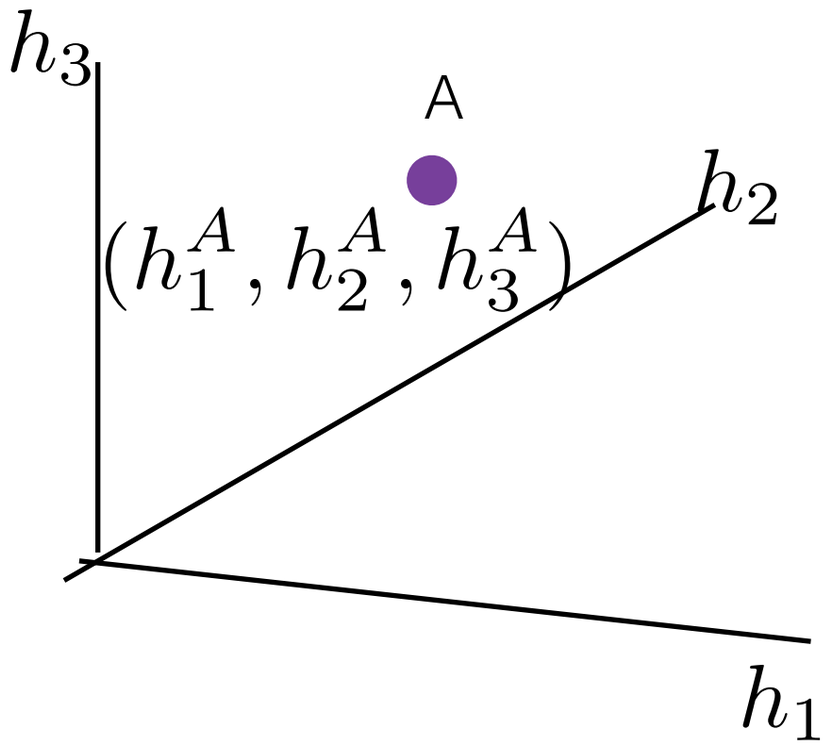
Neighbourhood relationship is conserved

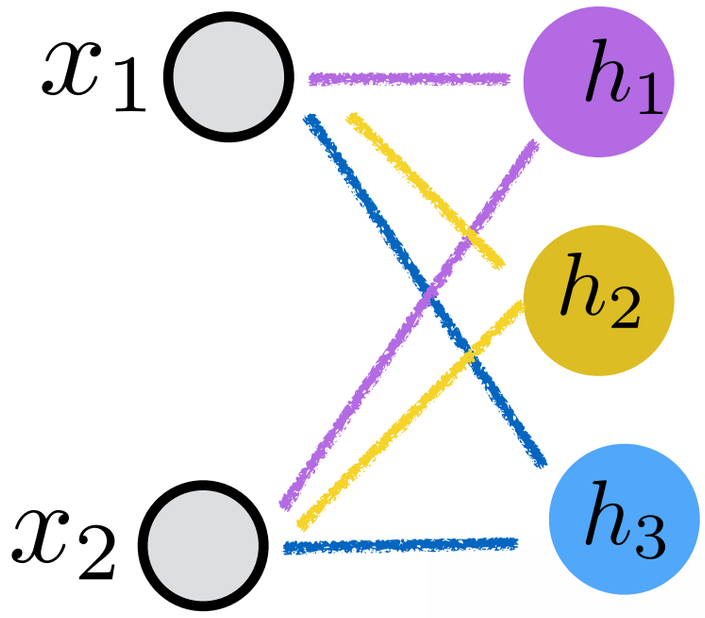


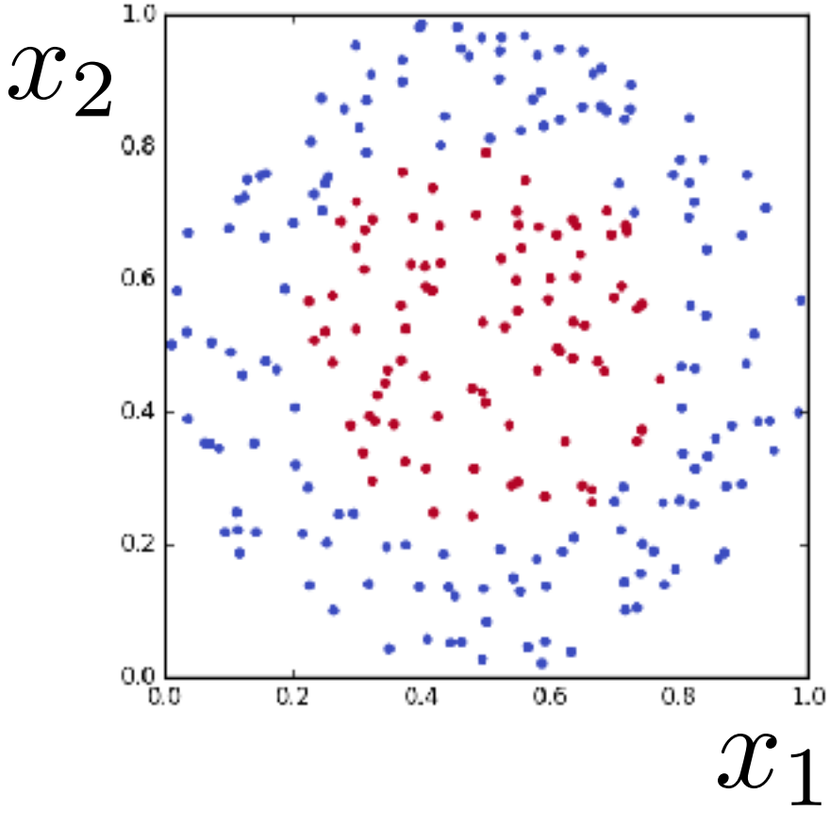
$$(x_1^A, x_2^A) \mapsto (h_1^A, h_2^A)$$

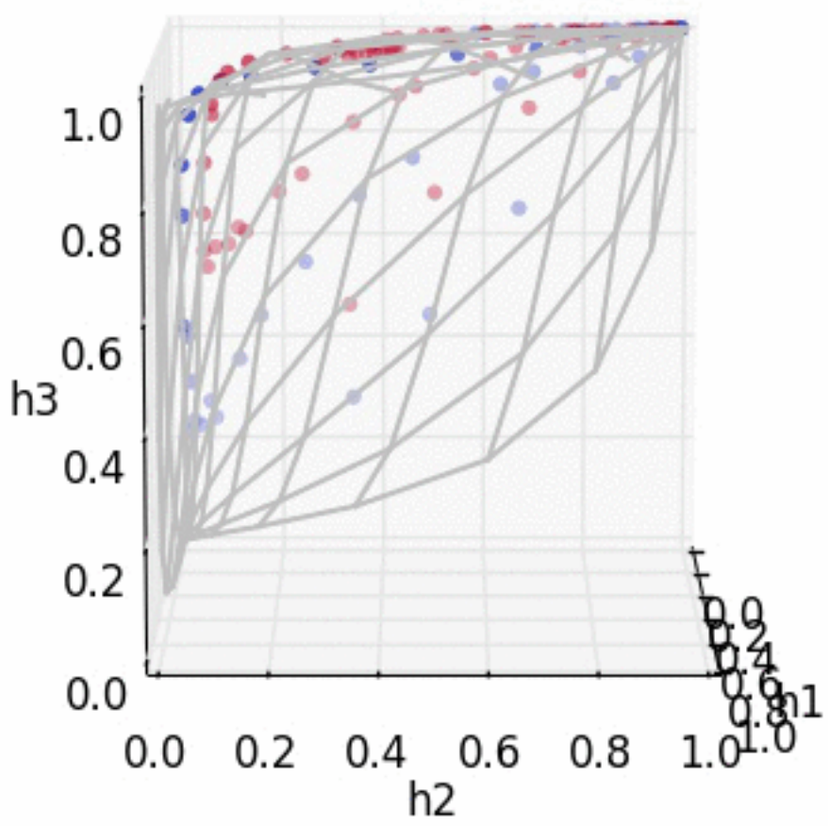

x_1
x_2
h_1
h_2
h_3

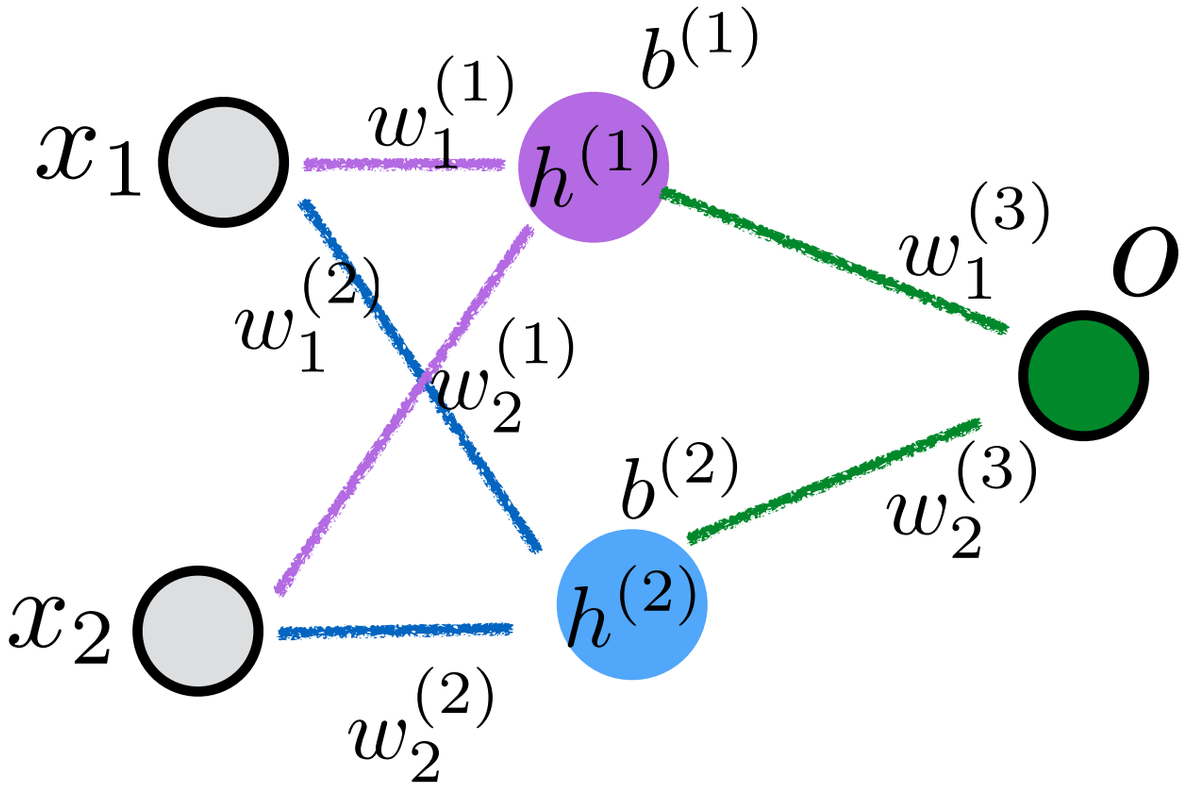

x_2
A
(x_1^A, x_2^A)
x_1

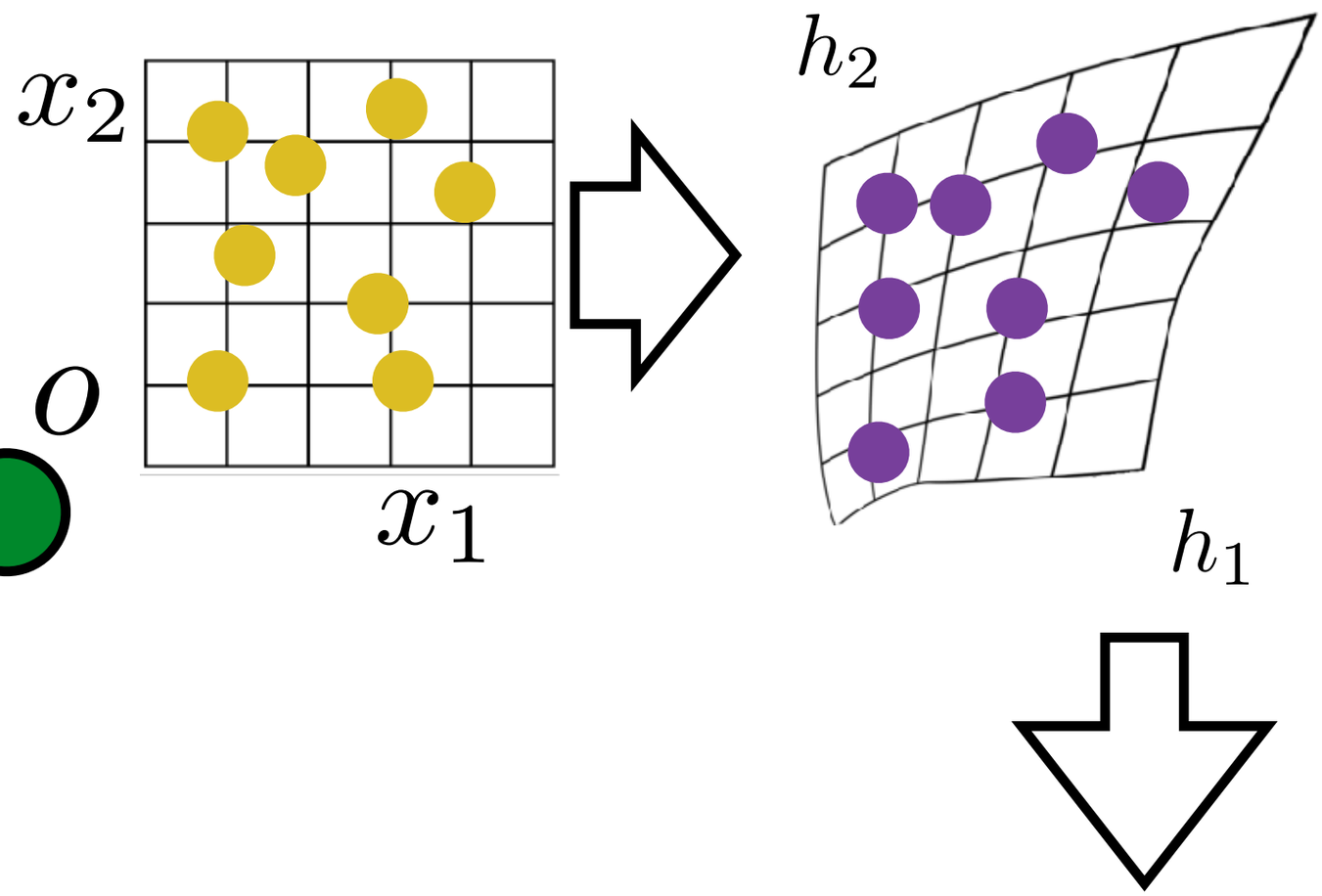

h_3
A
(h_1^A, h_2^A, h_3^A)
h_2
h_1


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





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

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

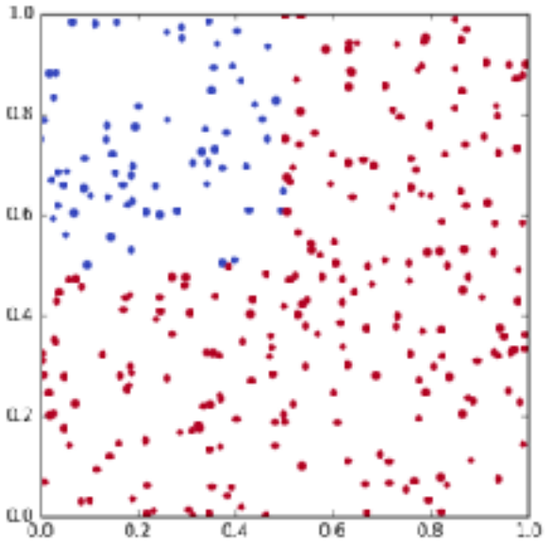
$$o = \sigma(w_1^{(3)} h^{(1)} + w_2^{(3)} h^{(2)} + b^{(3)})$$

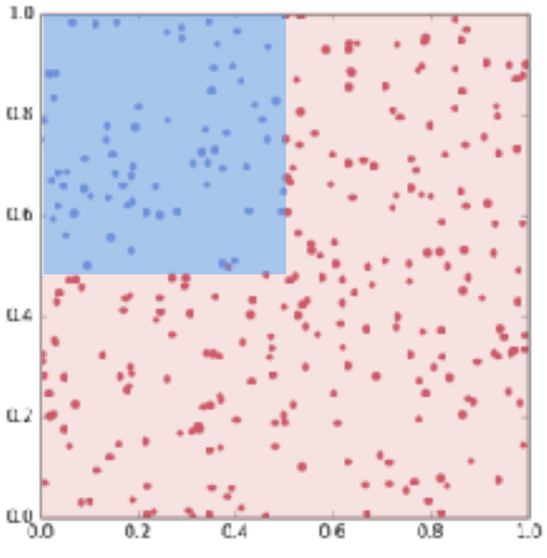
Some real life examples

courtesy of our TA Connie Kou

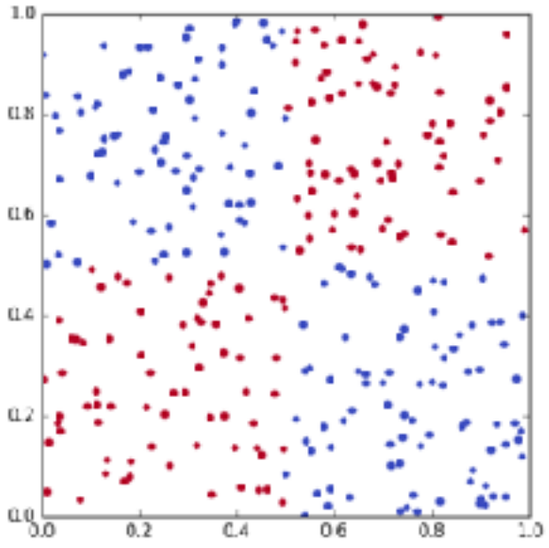
source code will be available online

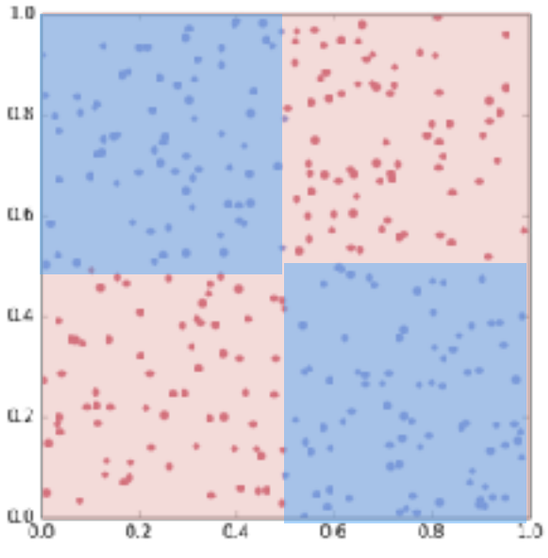
The Angle Data



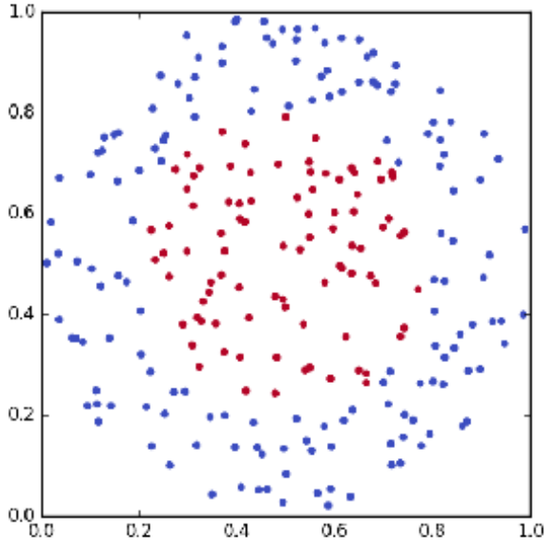


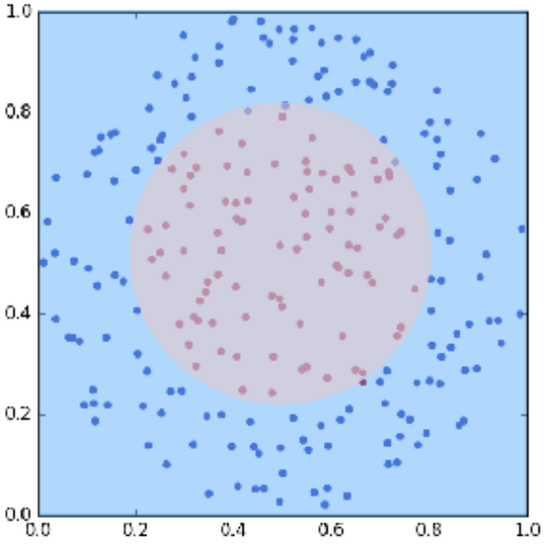
The XOR Data



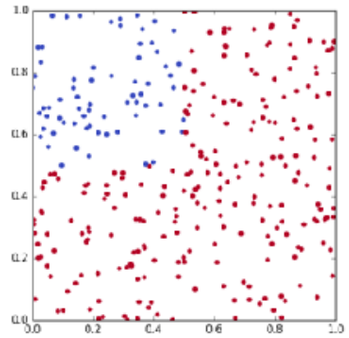


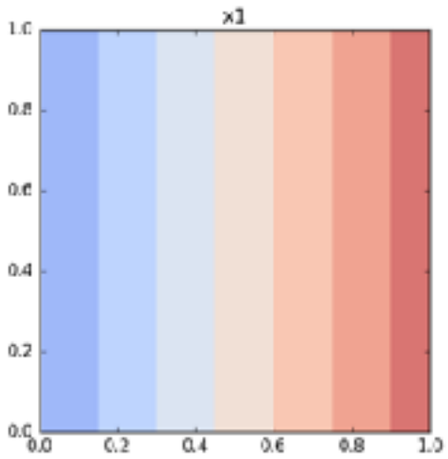
The Ring Data

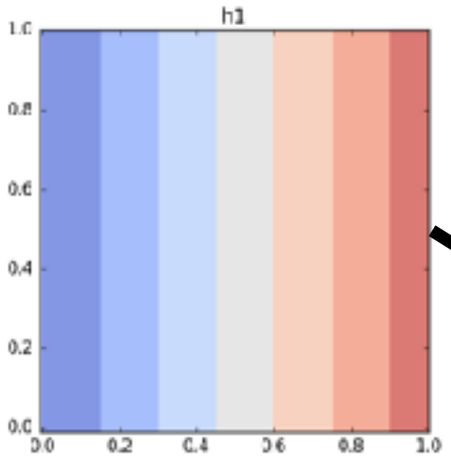


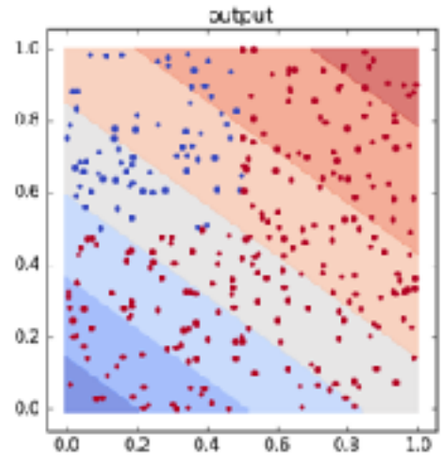


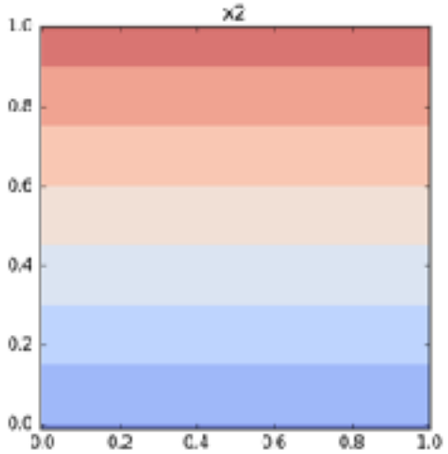
The Angle Data - ReLu

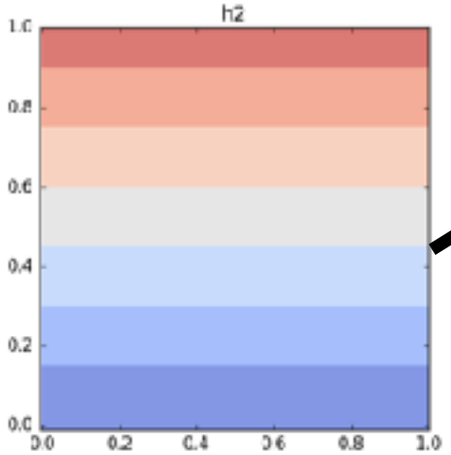


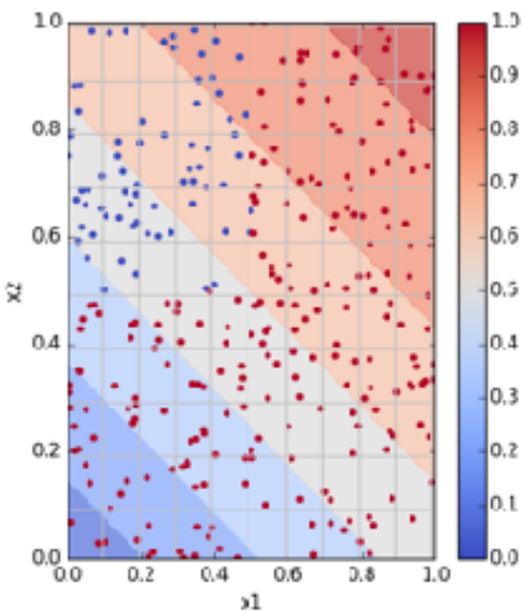


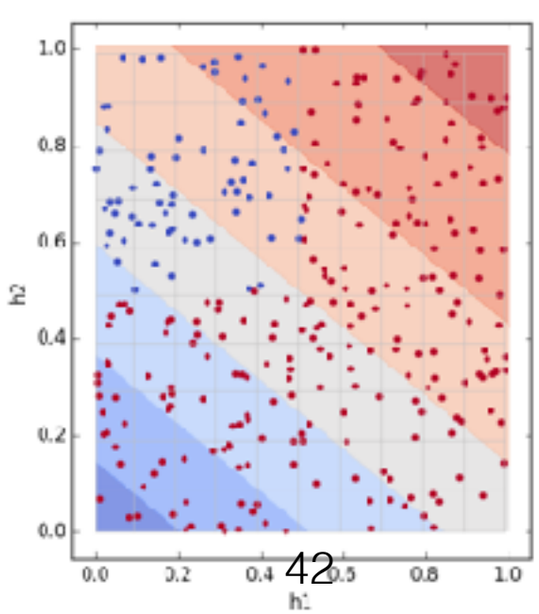


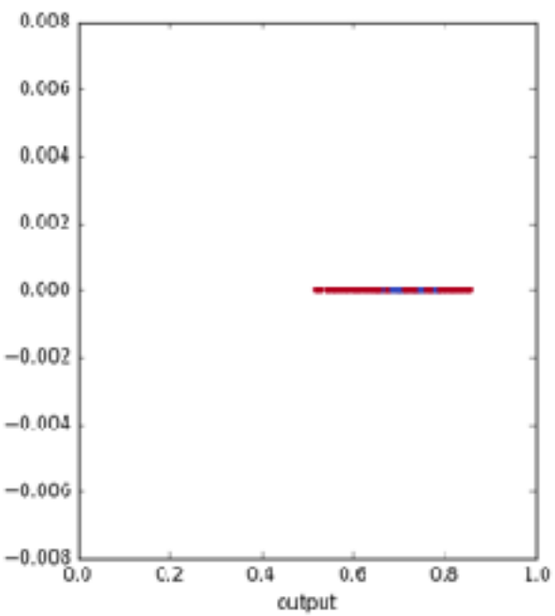




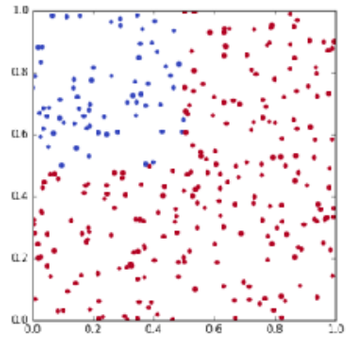


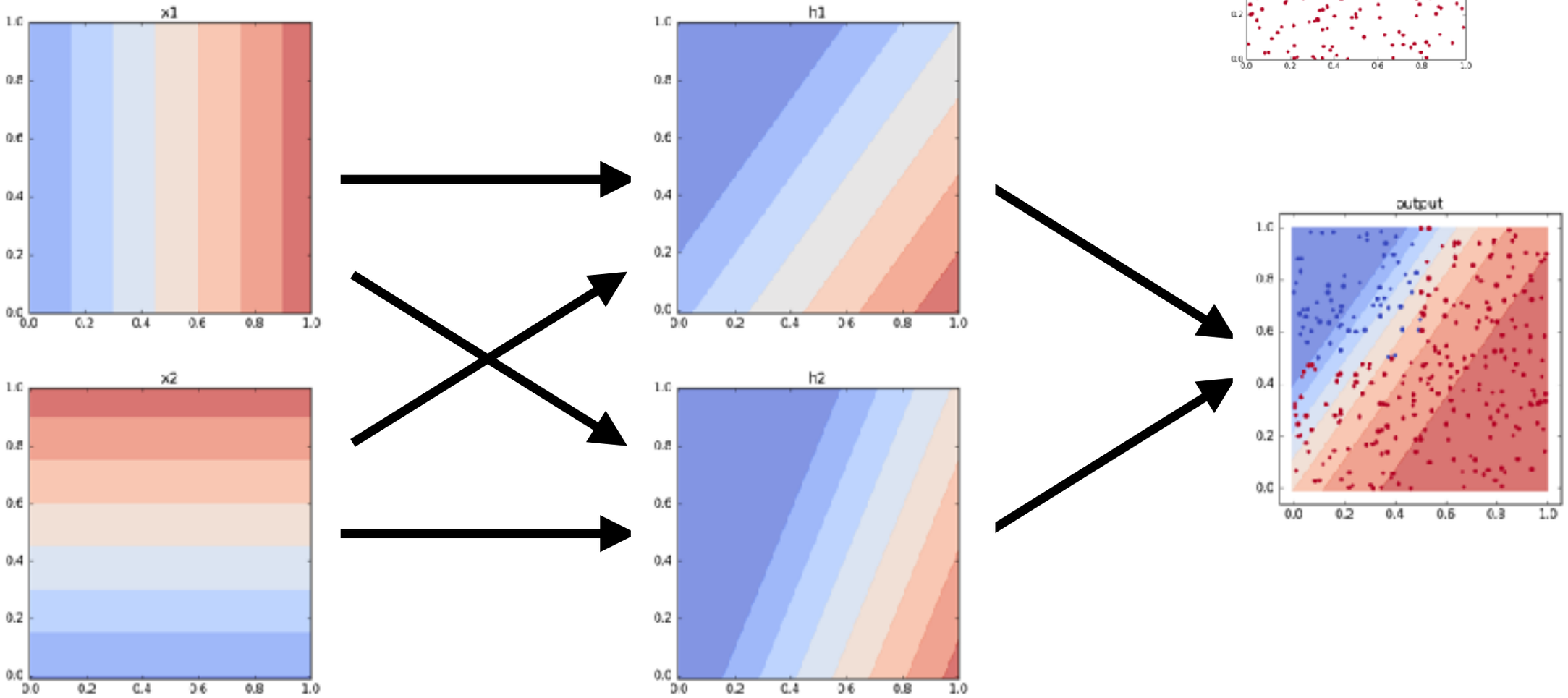


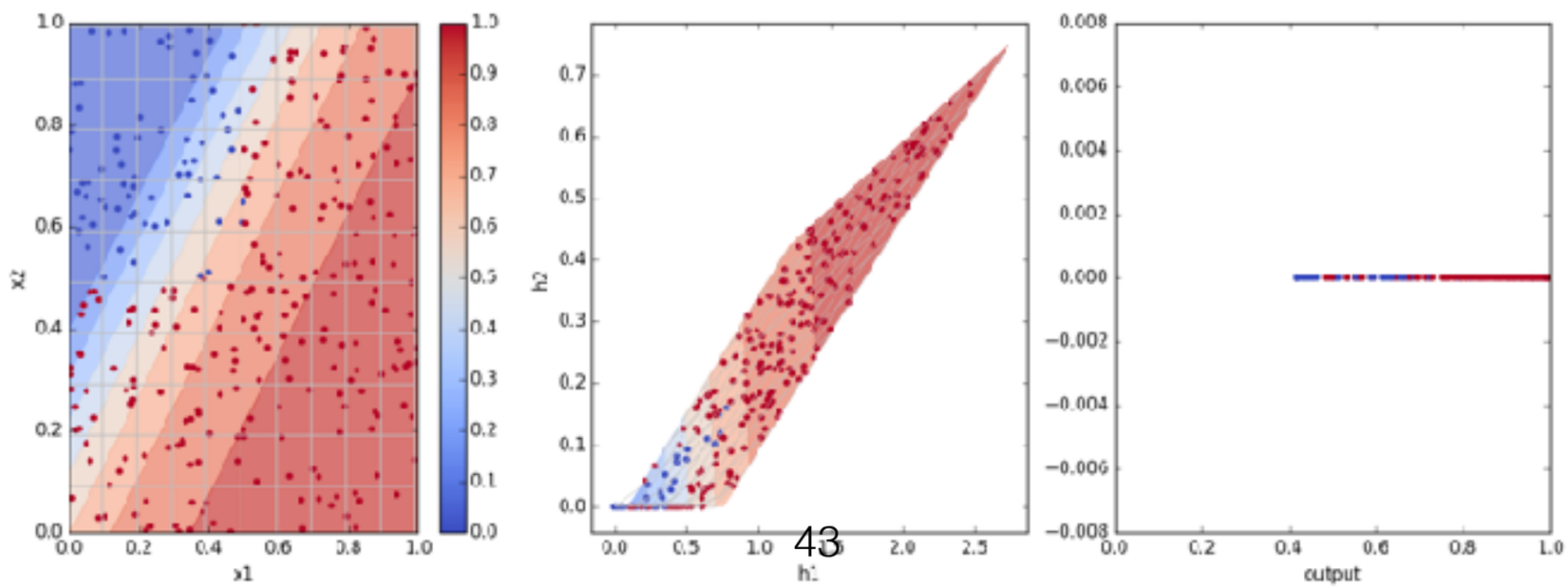




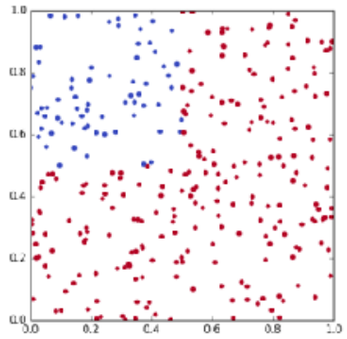
The Angle Data - ReLu

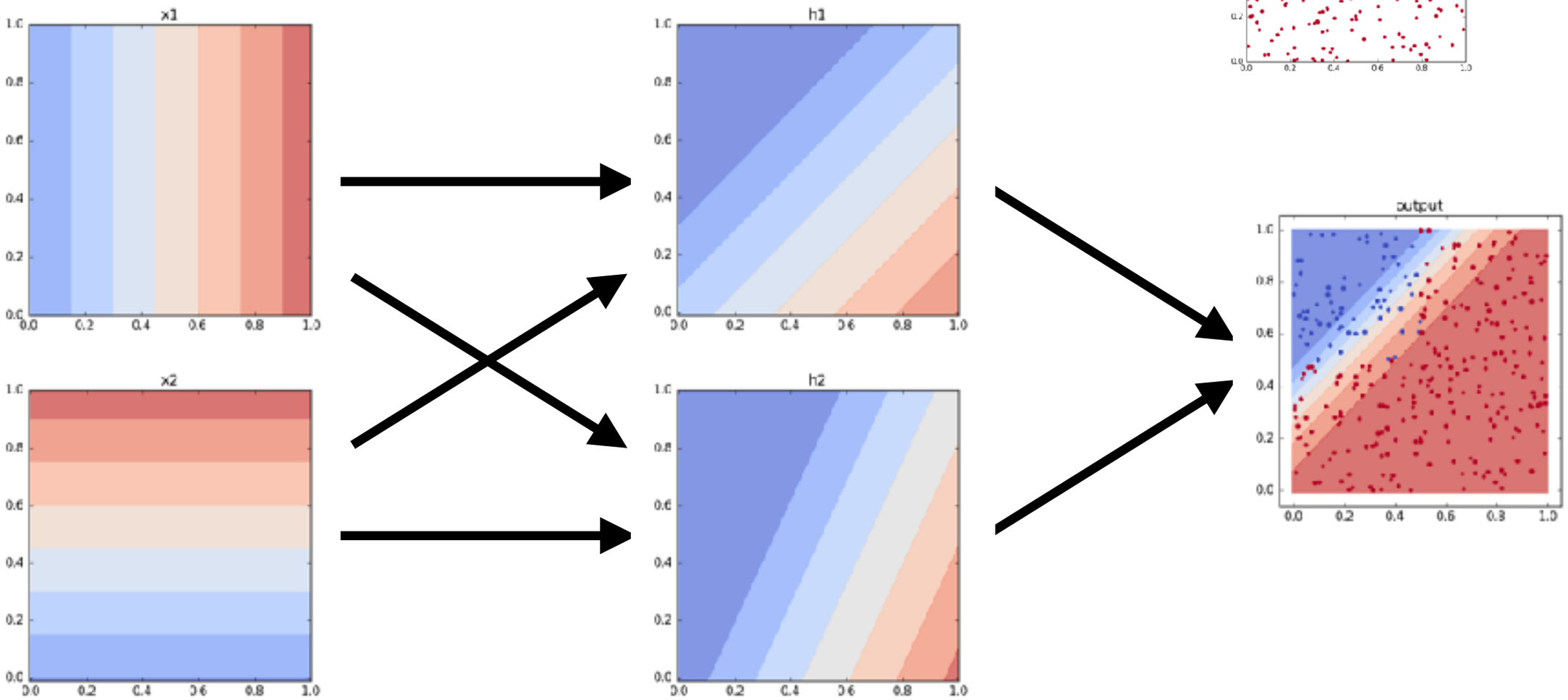


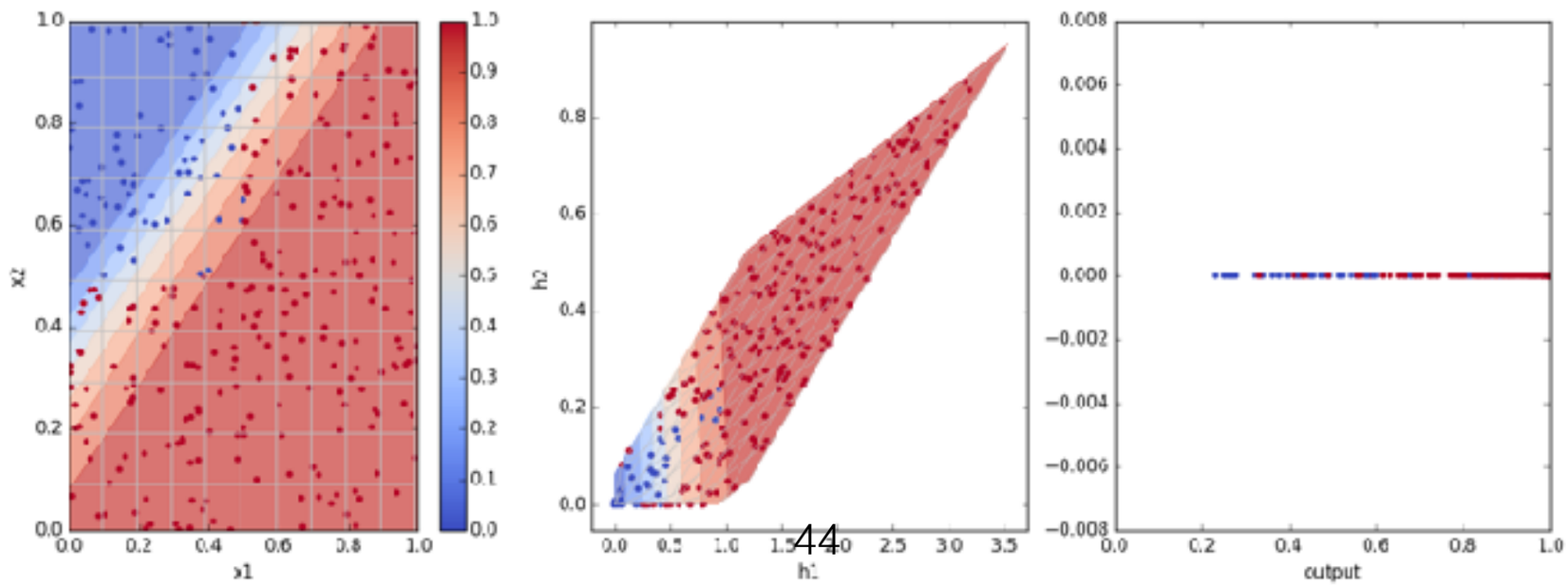




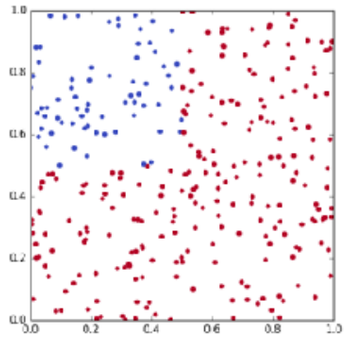
The Angle Data - ReLu

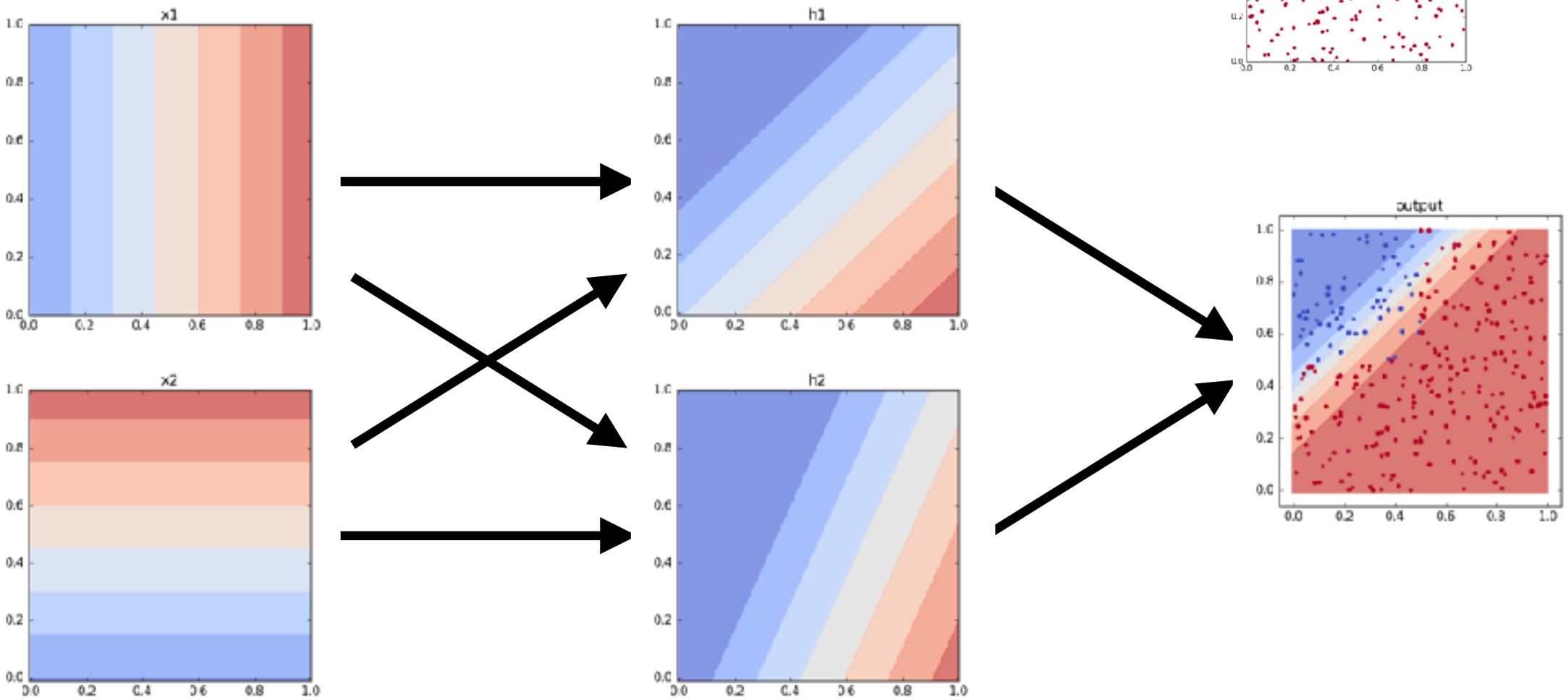


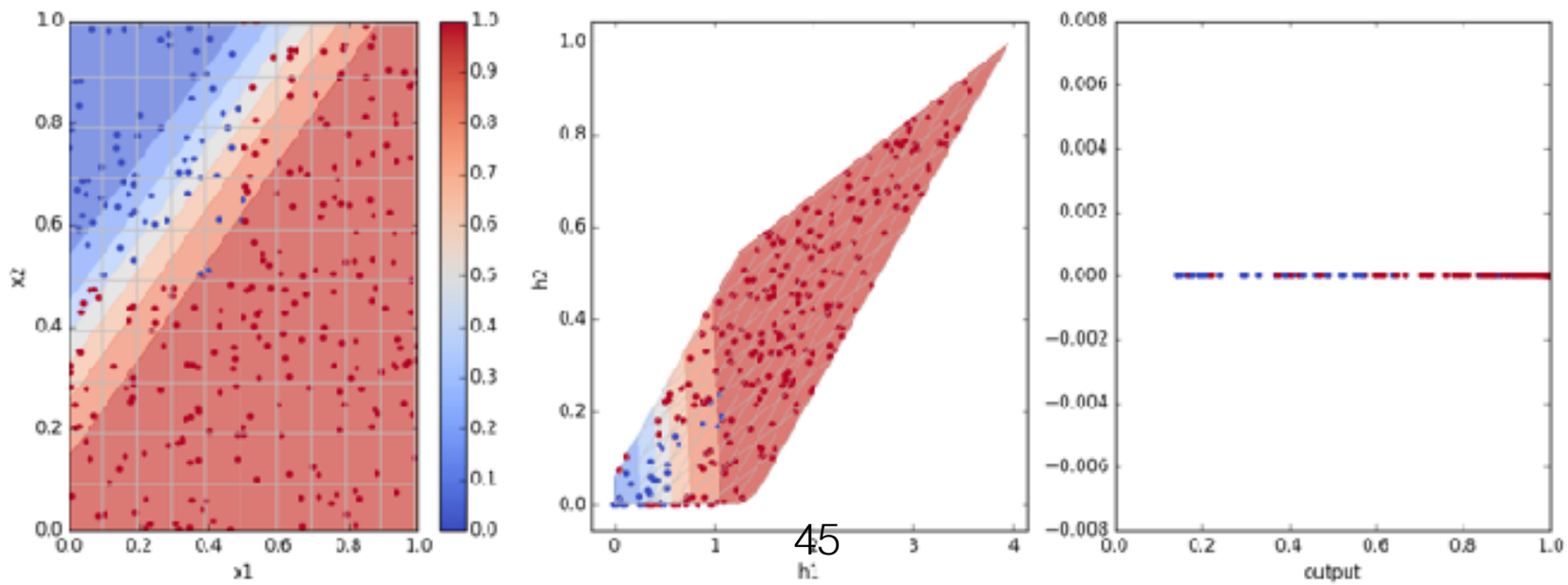




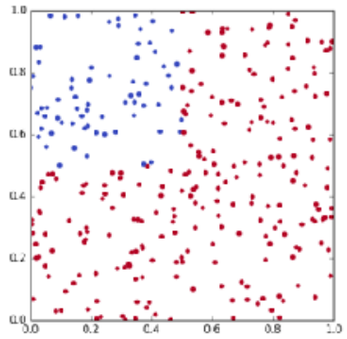
The Angle Data - ReLu

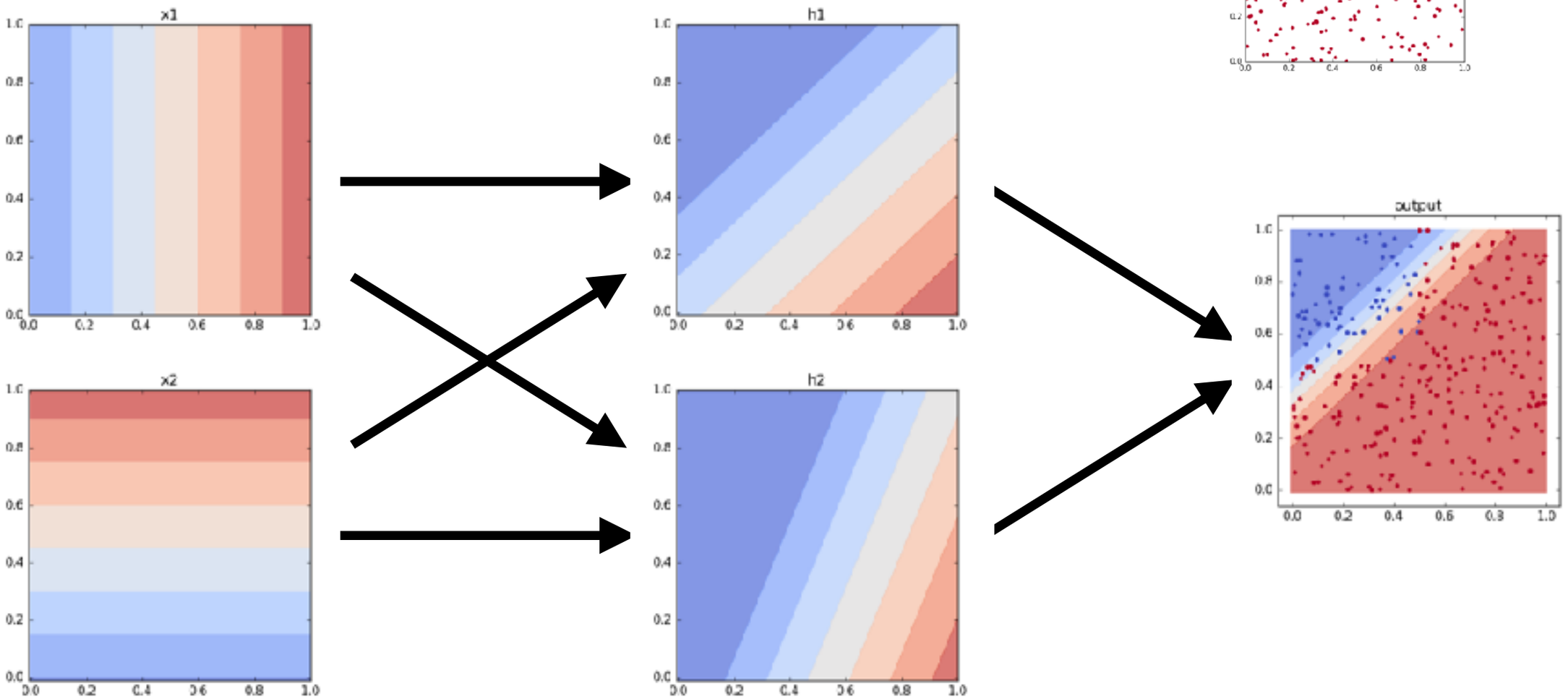


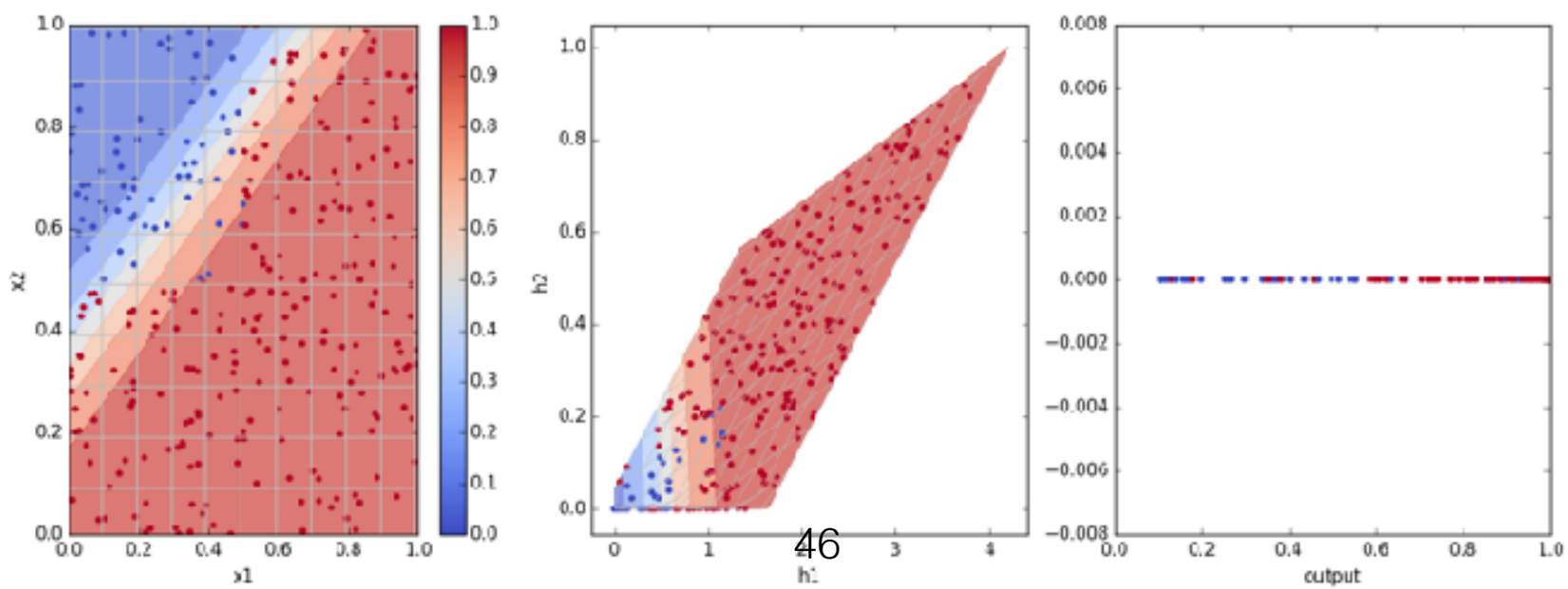




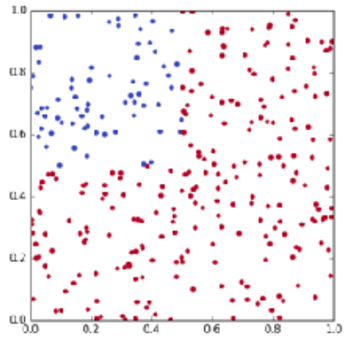
The Angle Data - ReLu

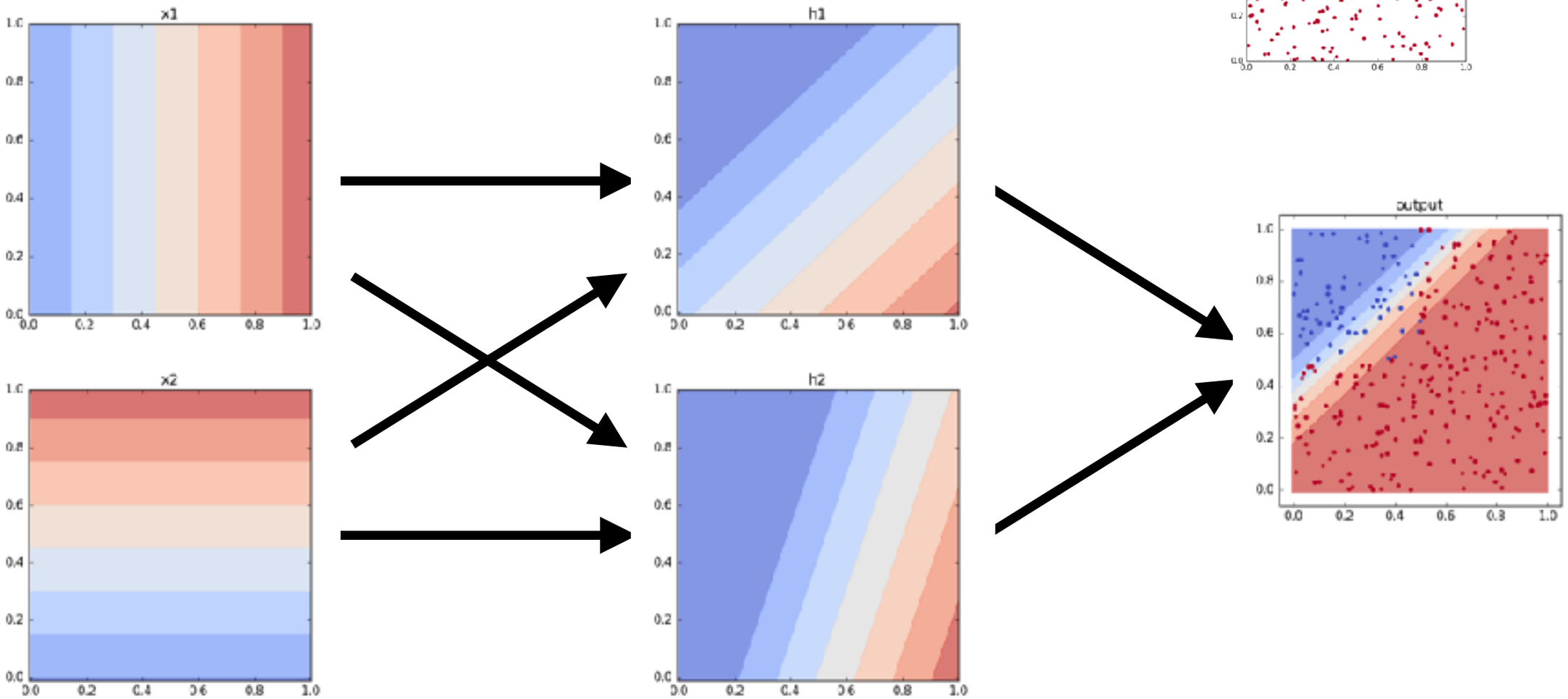


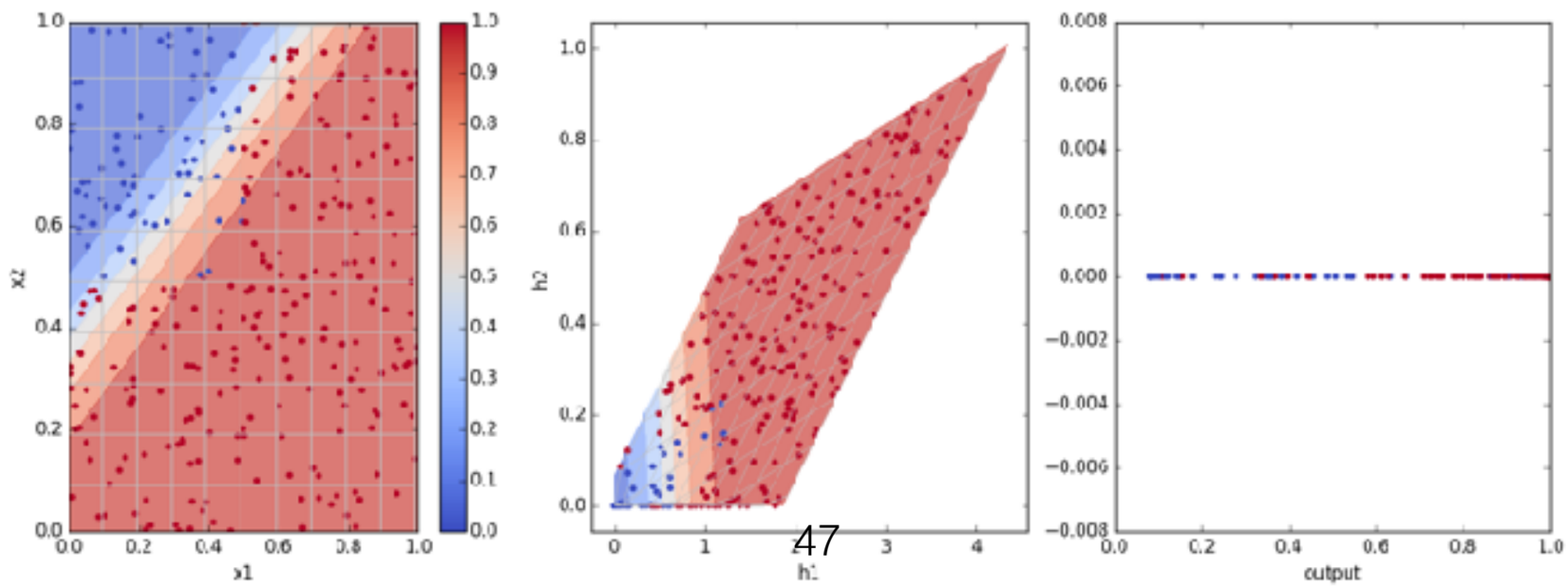




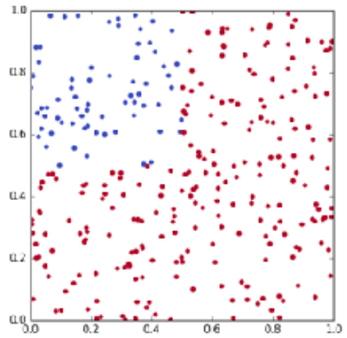
The Angle Data - ReLu

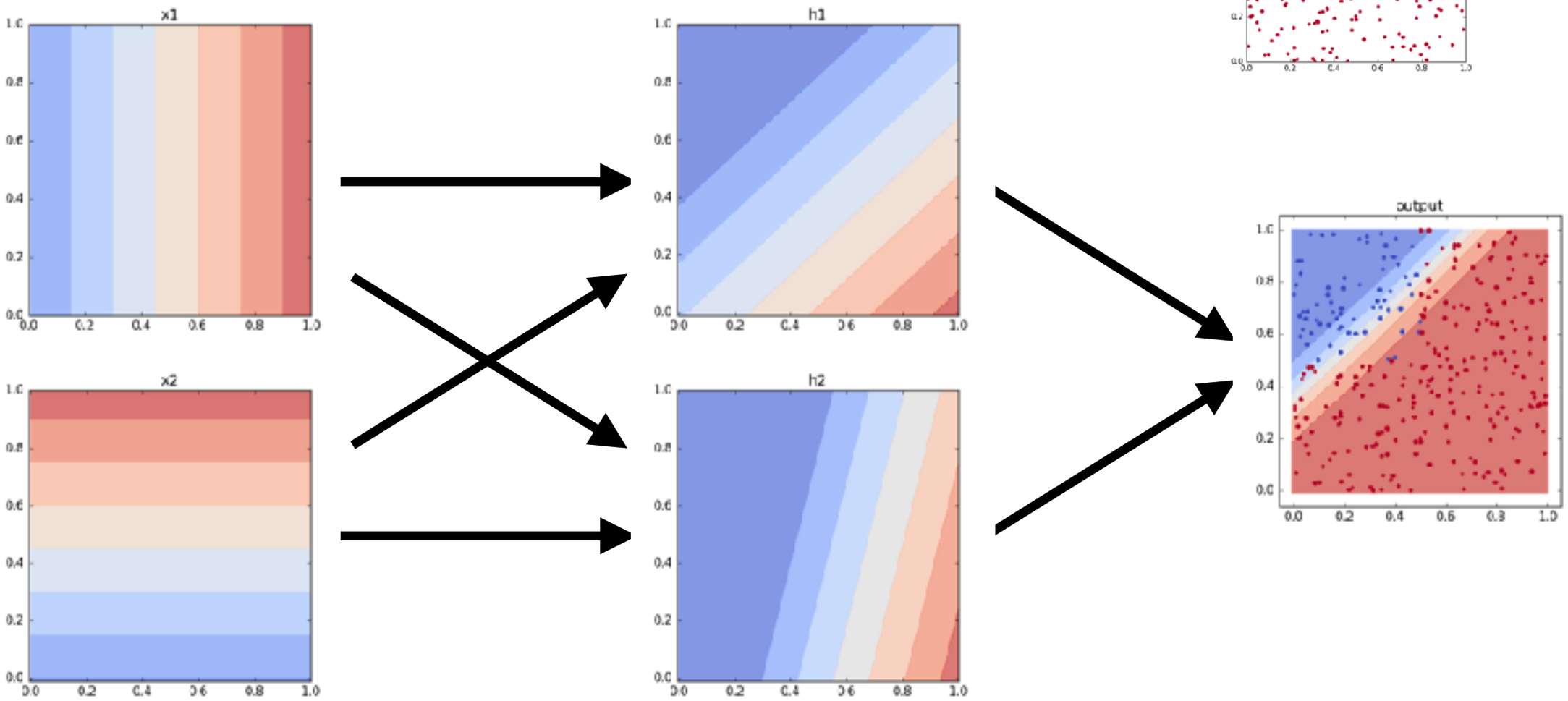


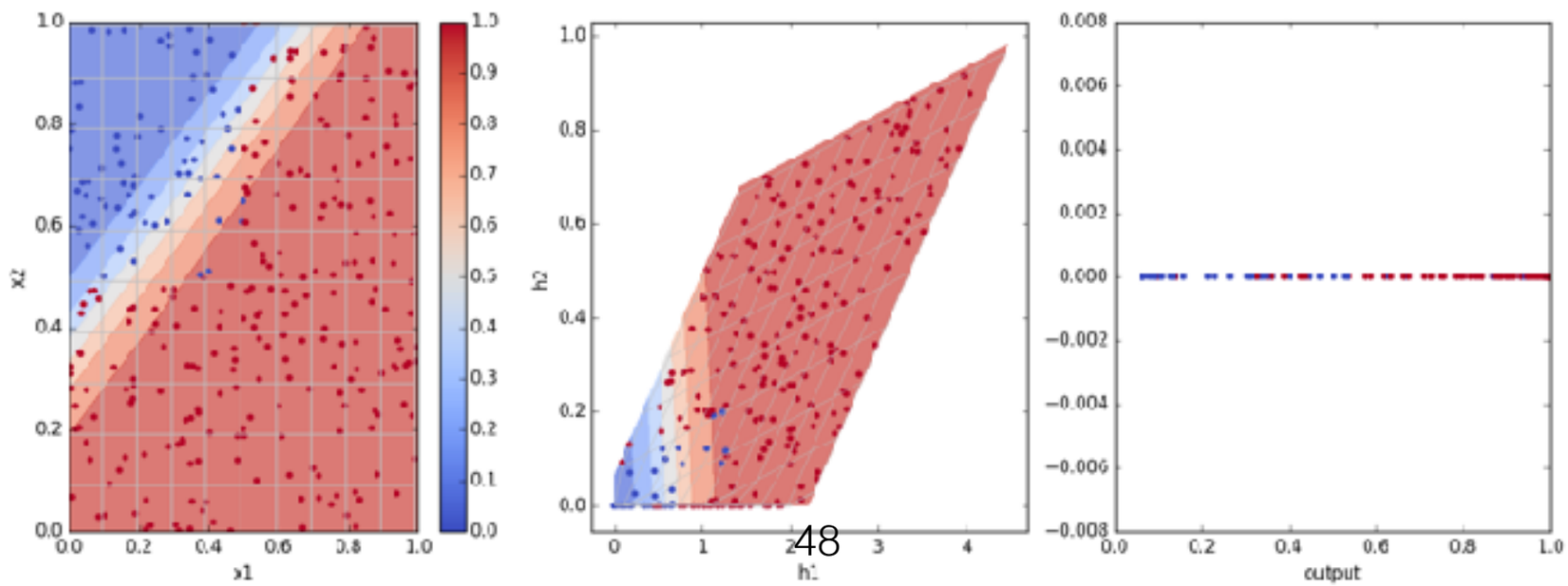




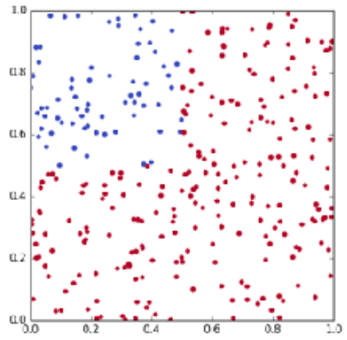
The Angle Data - ReLu

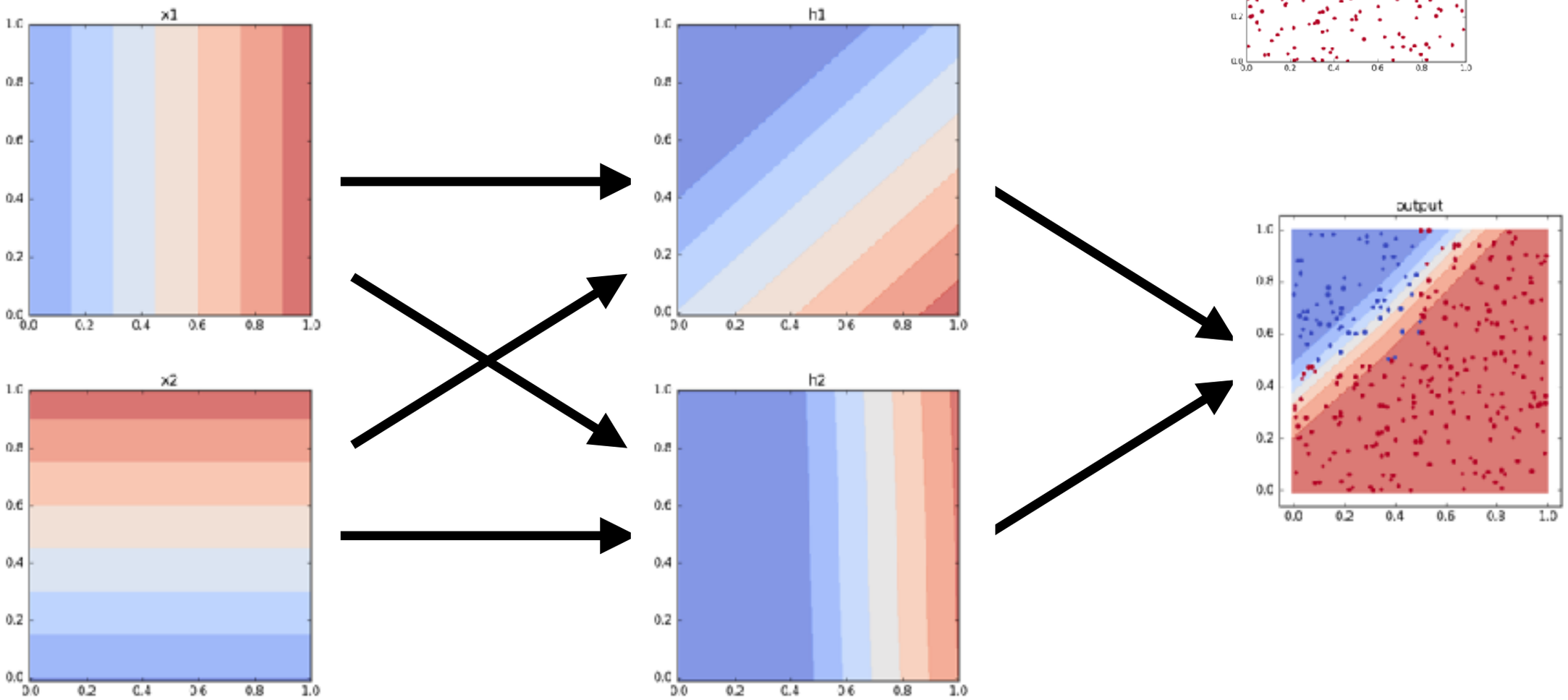


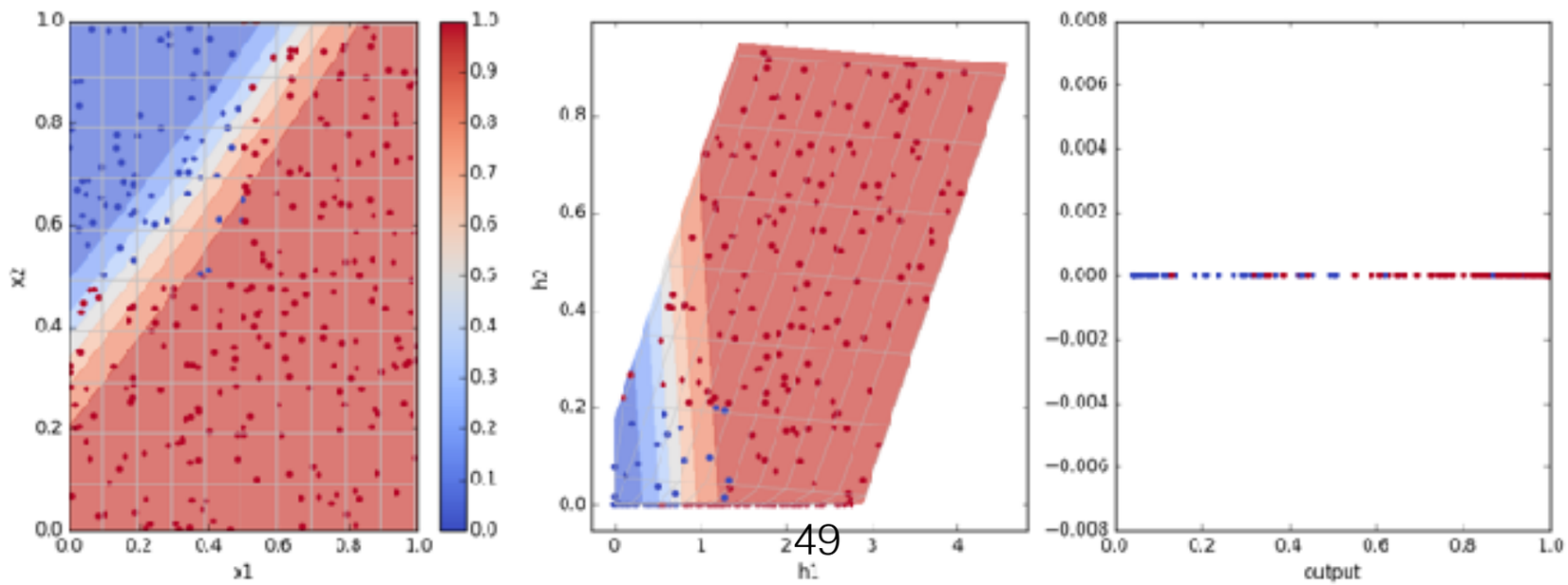




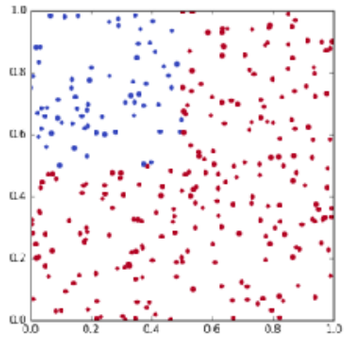
The Angle Data - ReLu

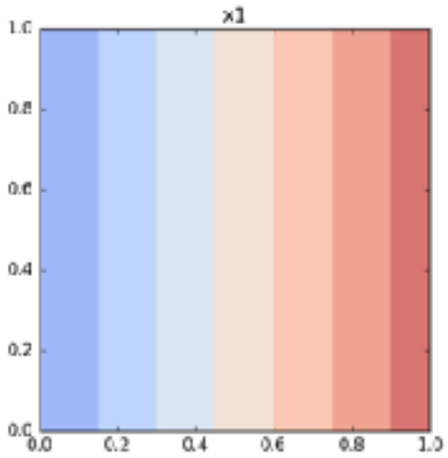


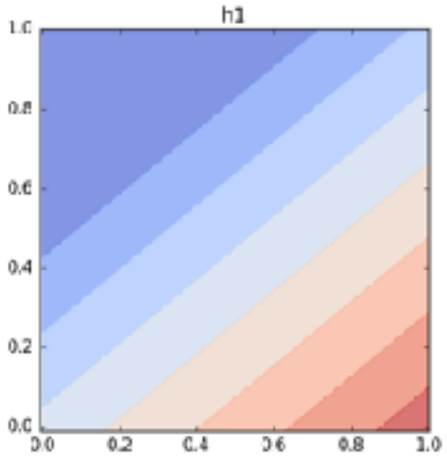


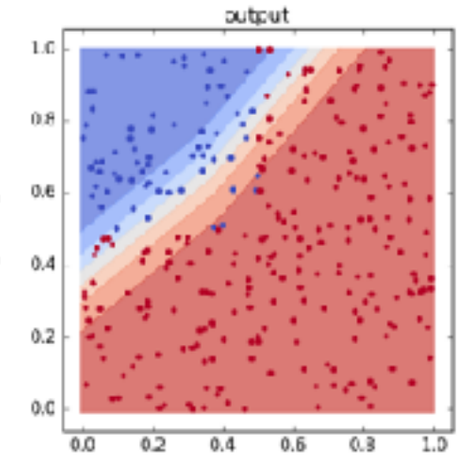


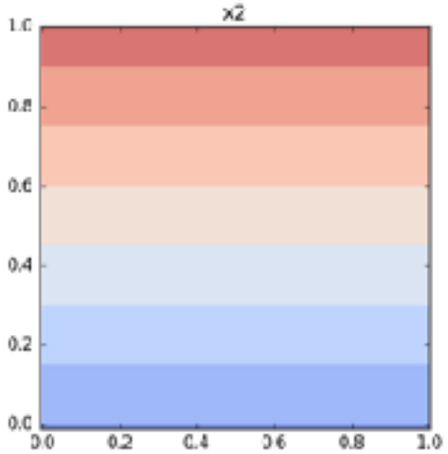
The Angle Data - ReLu

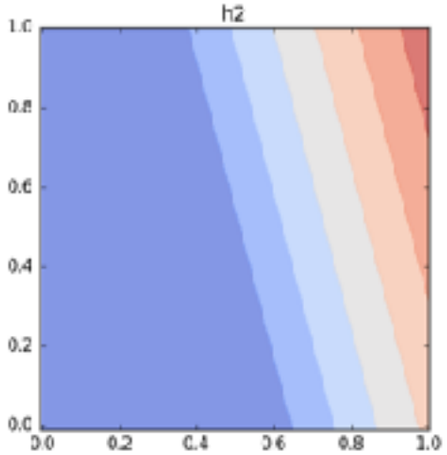


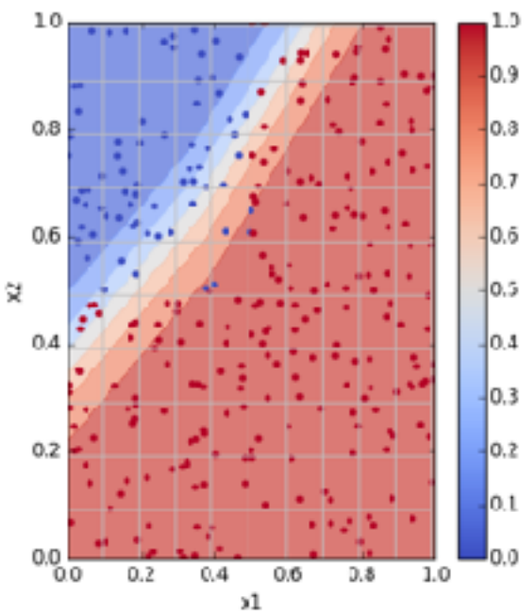


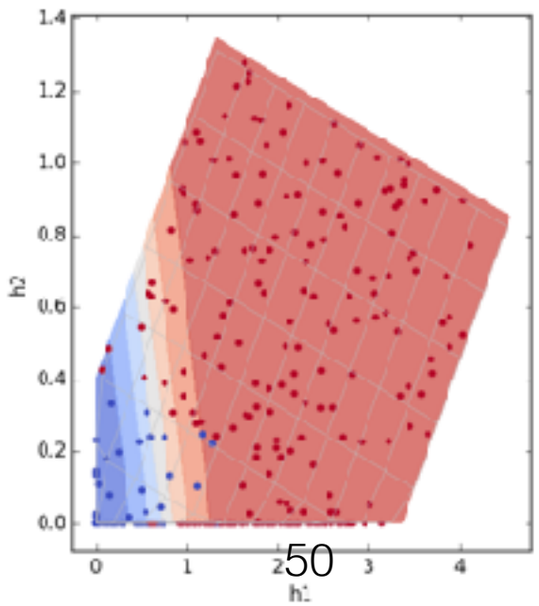


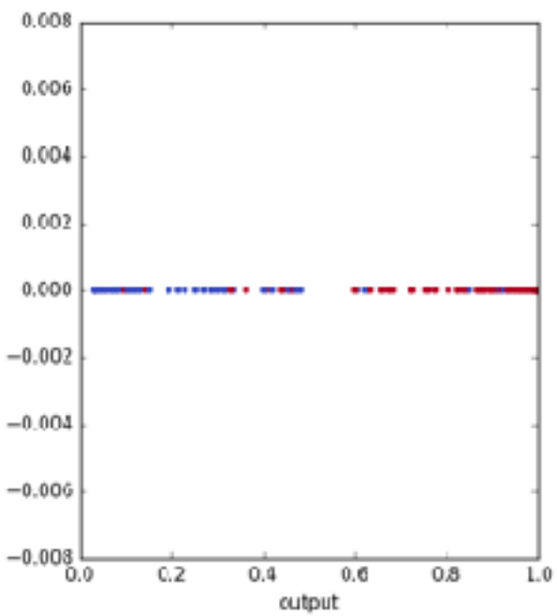




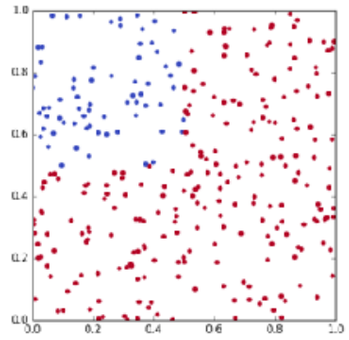


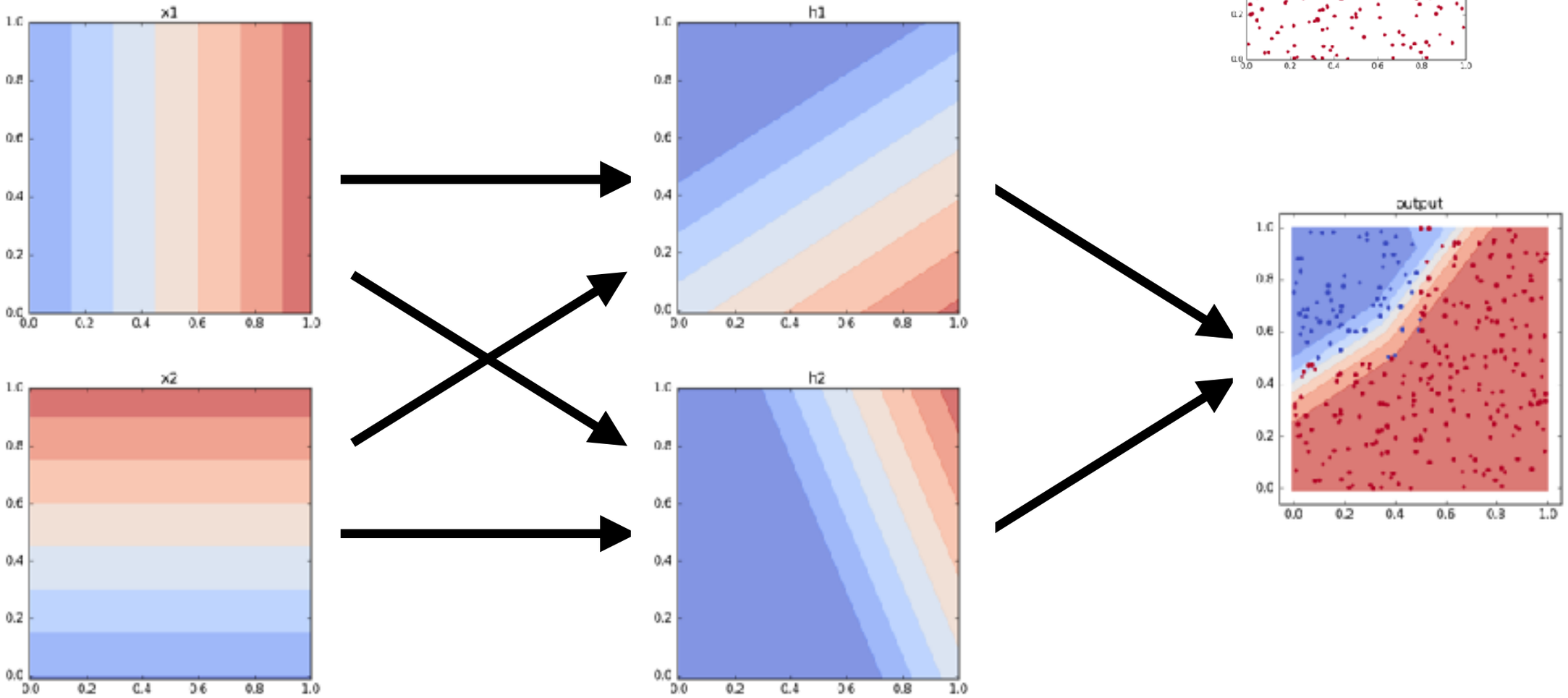


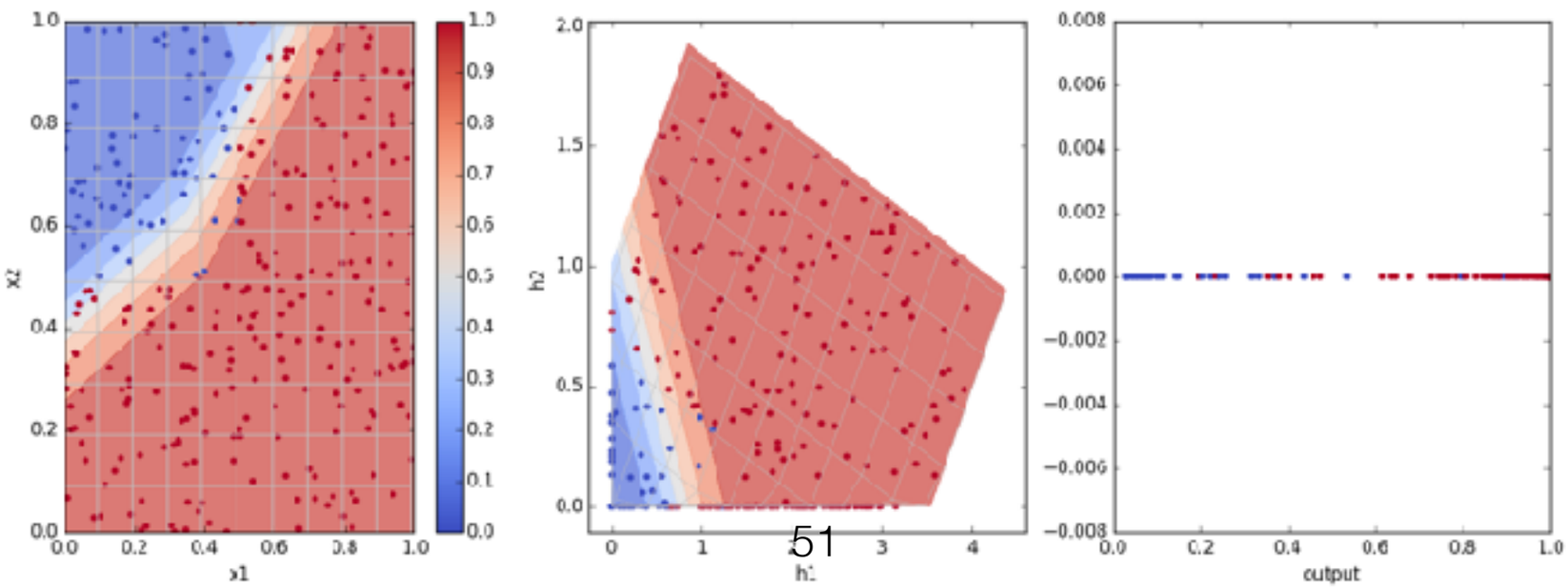




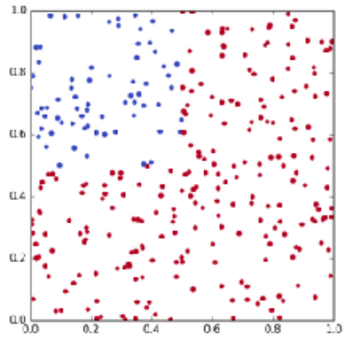
The Angle Data - ReLu

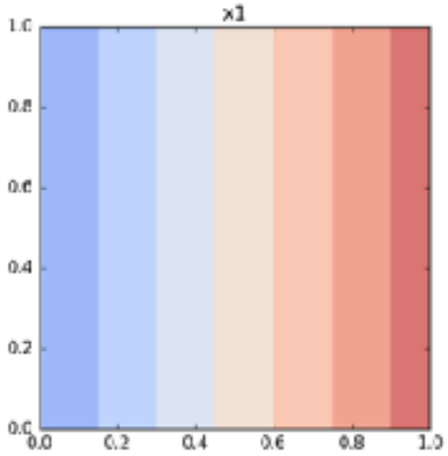


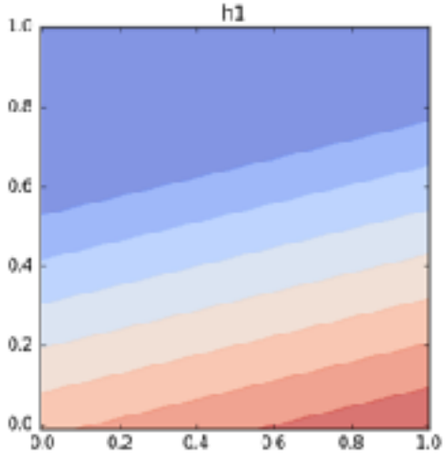


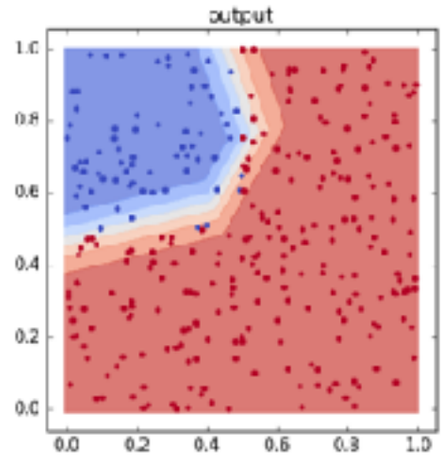


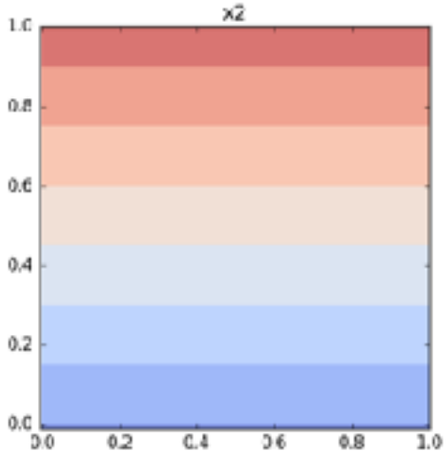
The Angle Data - ReLu

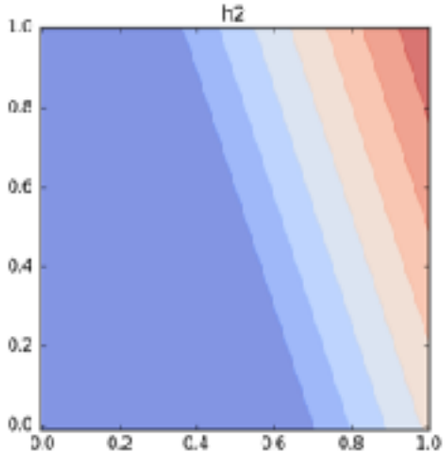


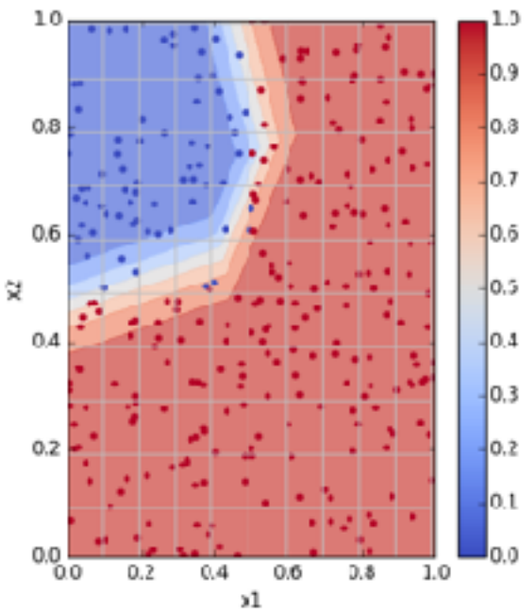


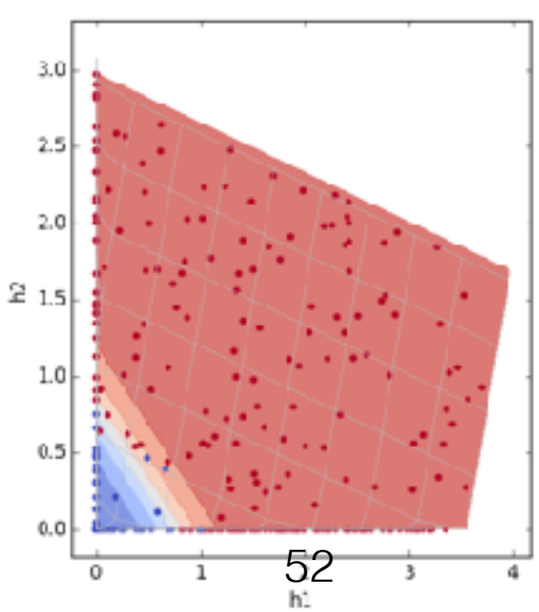


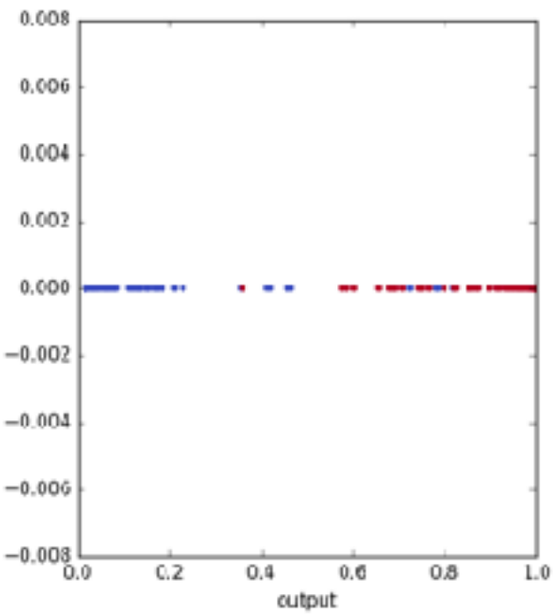






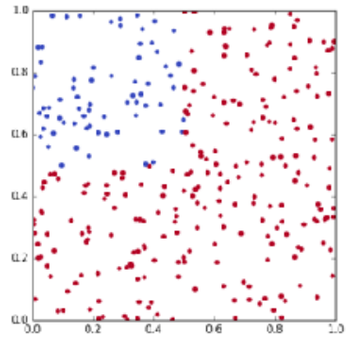


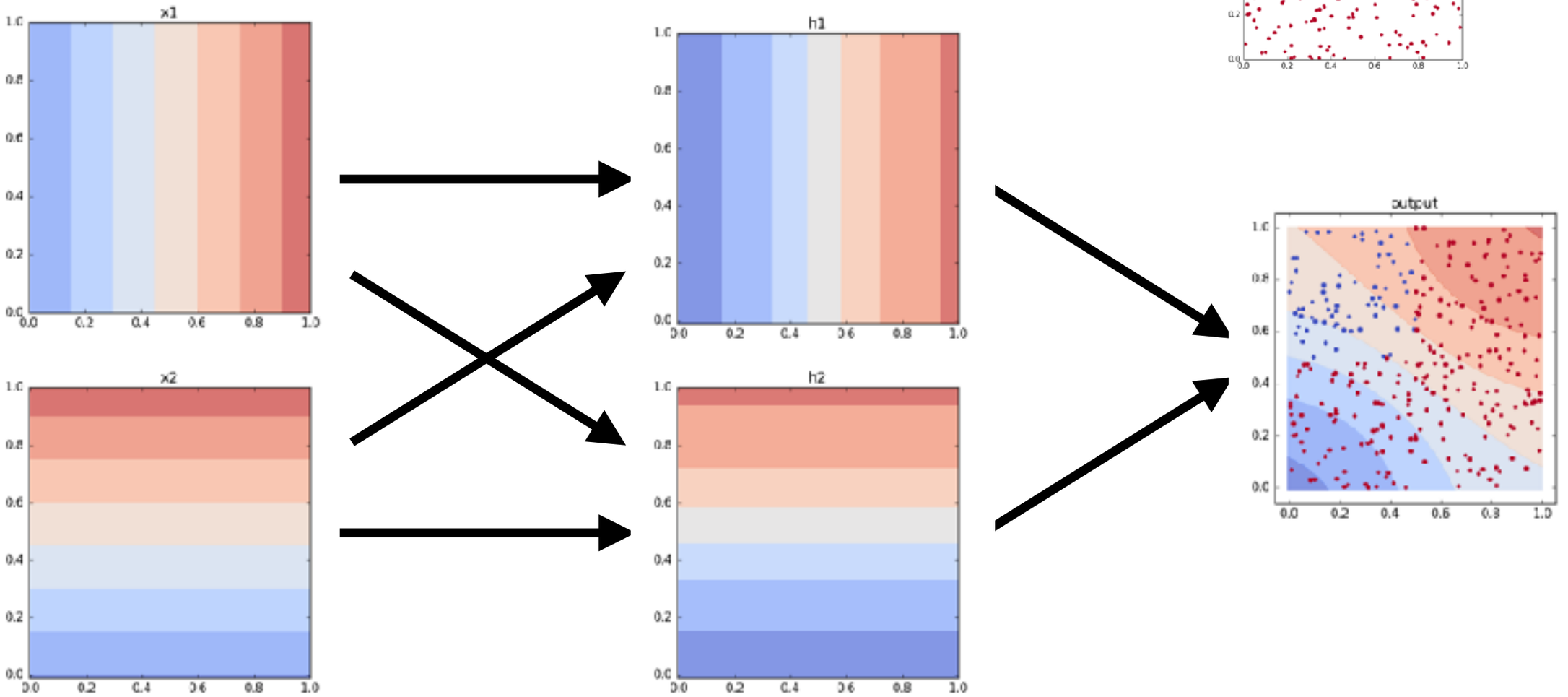


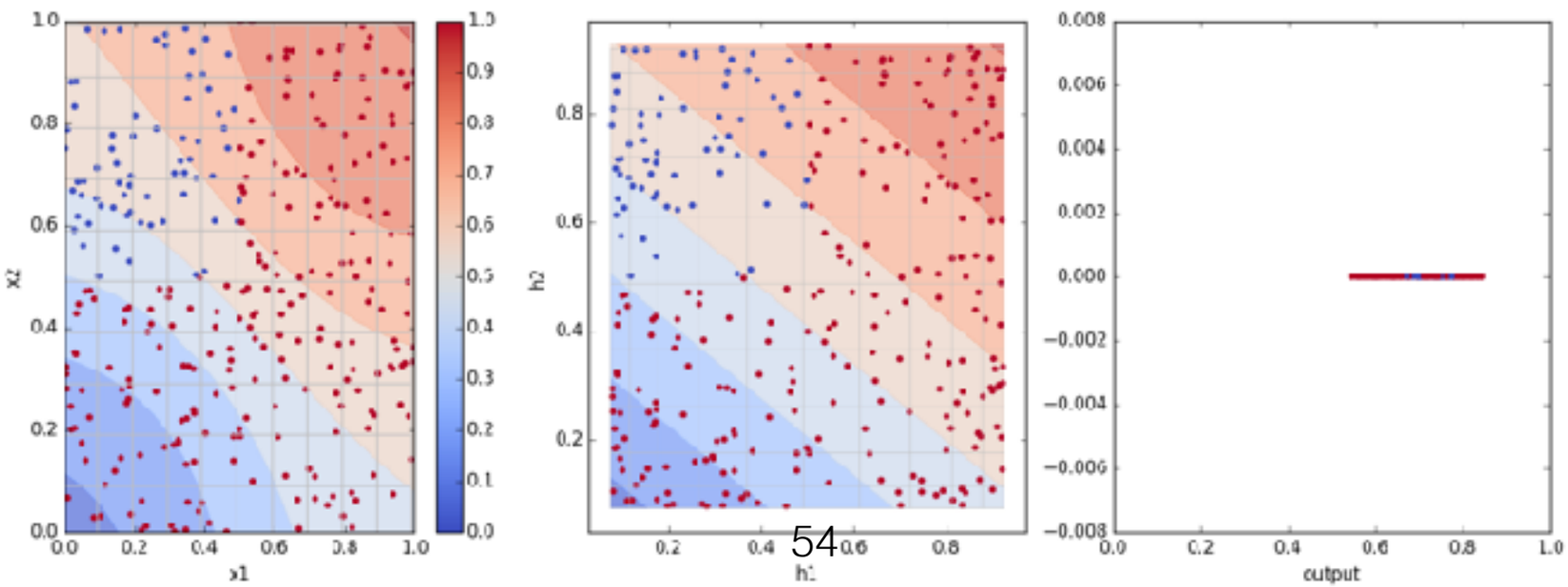


The Angle Data - Sigmoid

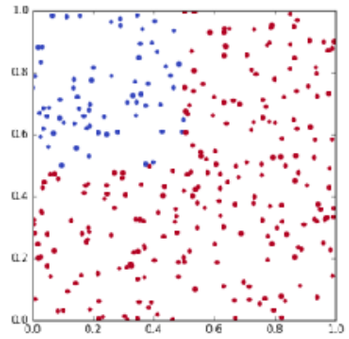
The Angle Data - Sigmoid

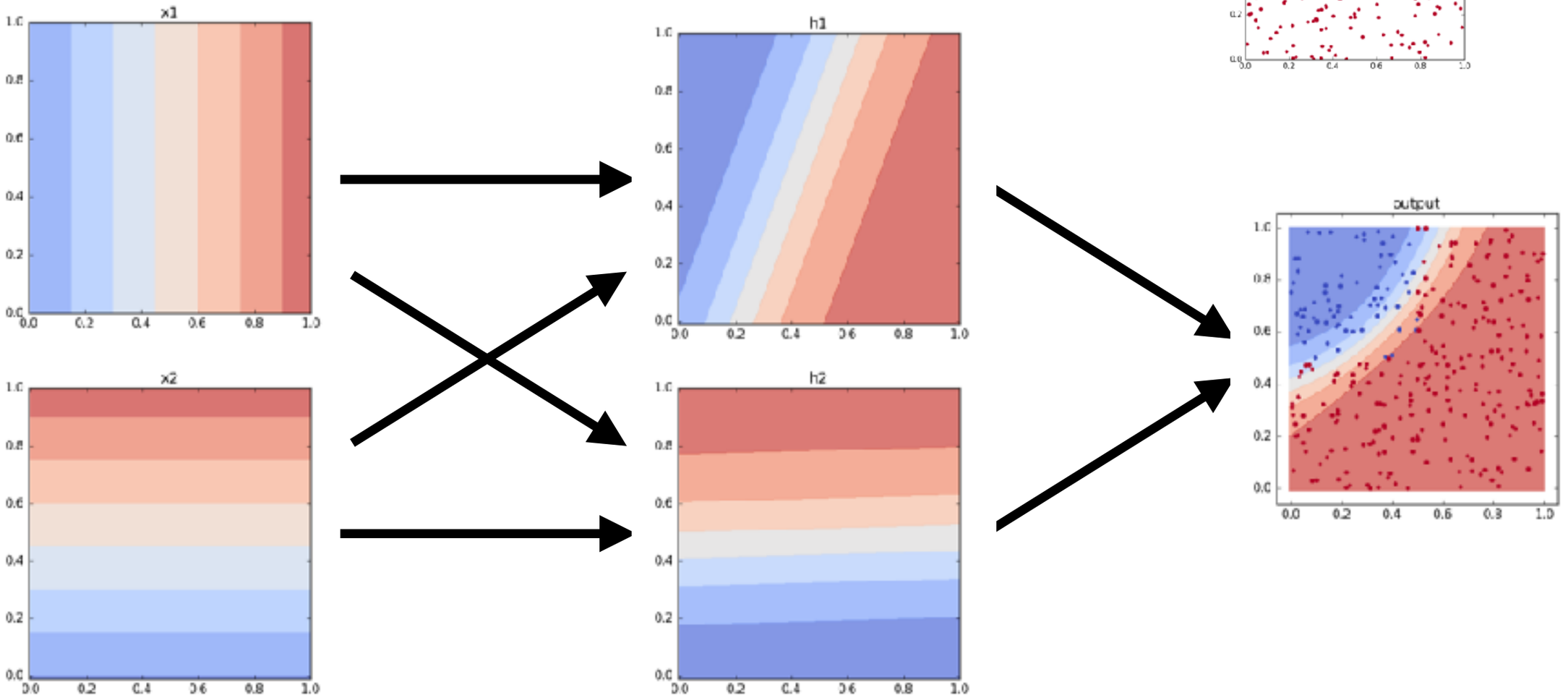


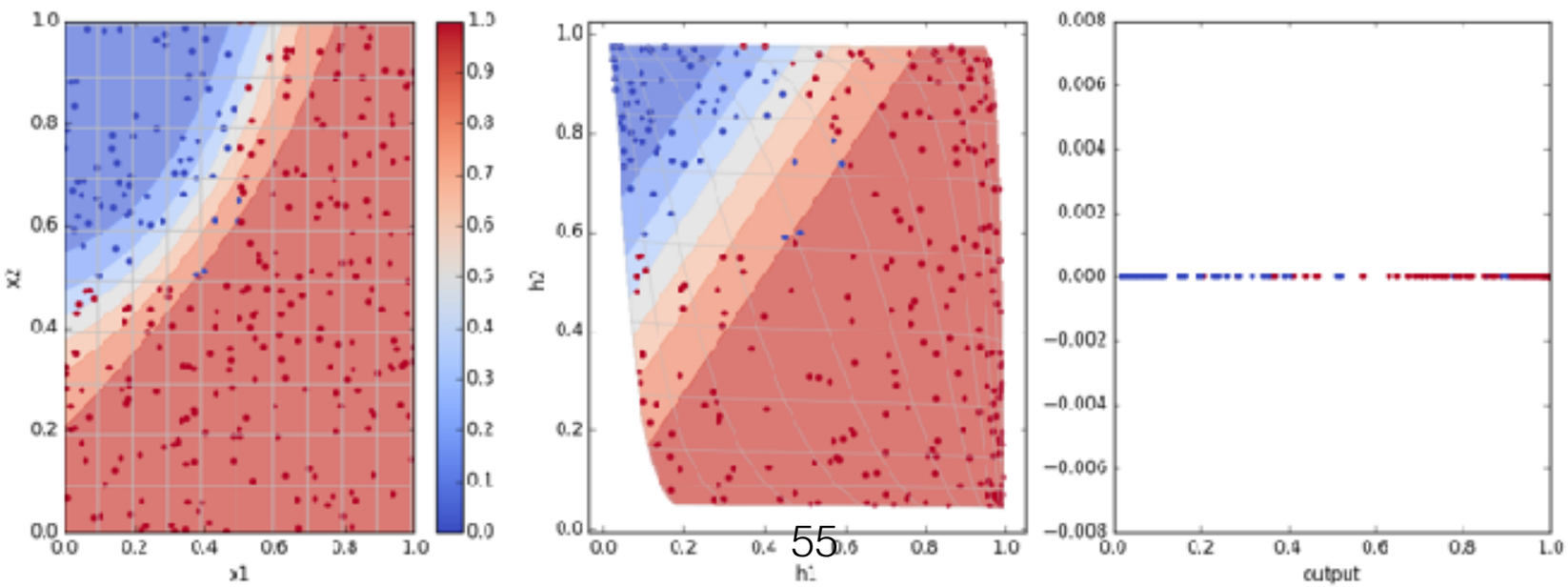




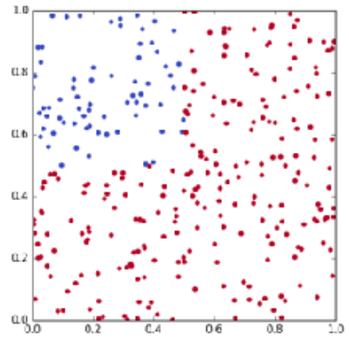
The Angle Data - Sigmoid

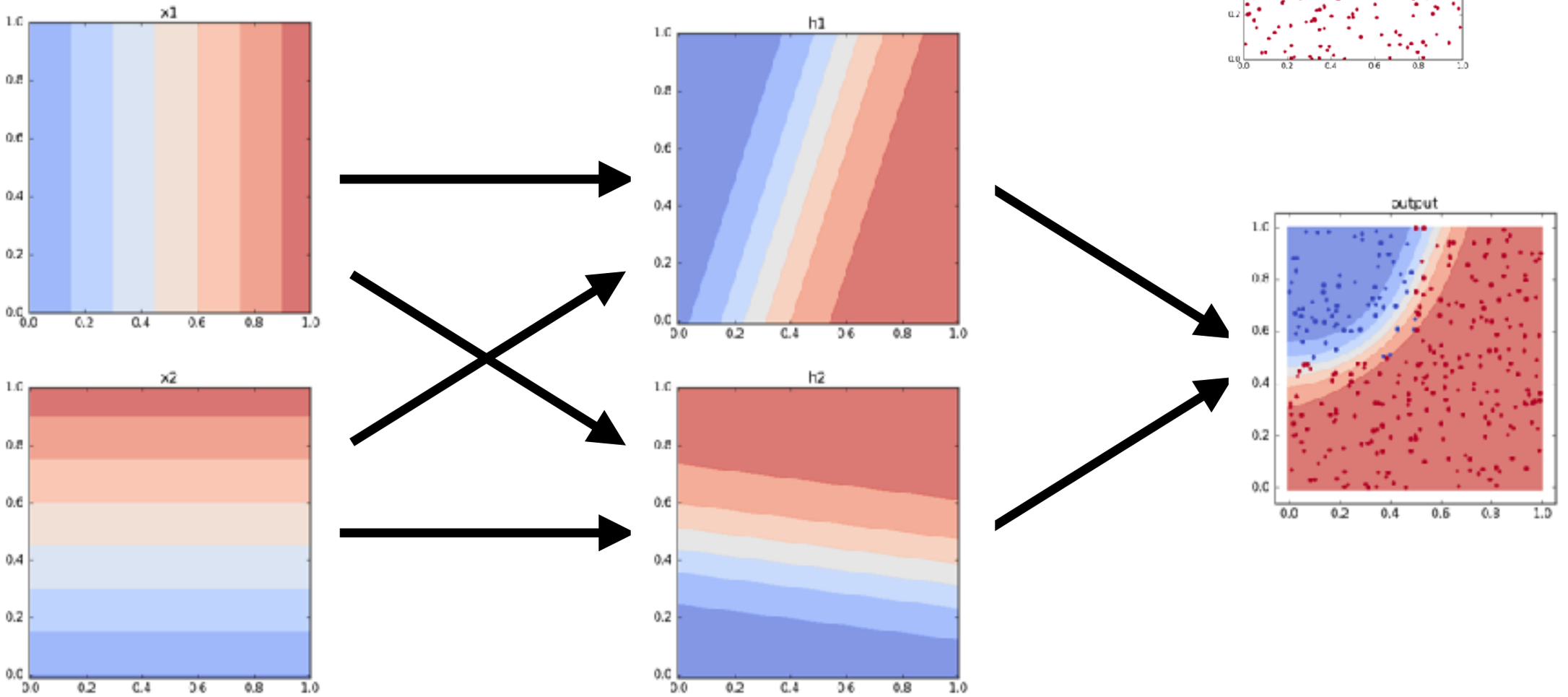


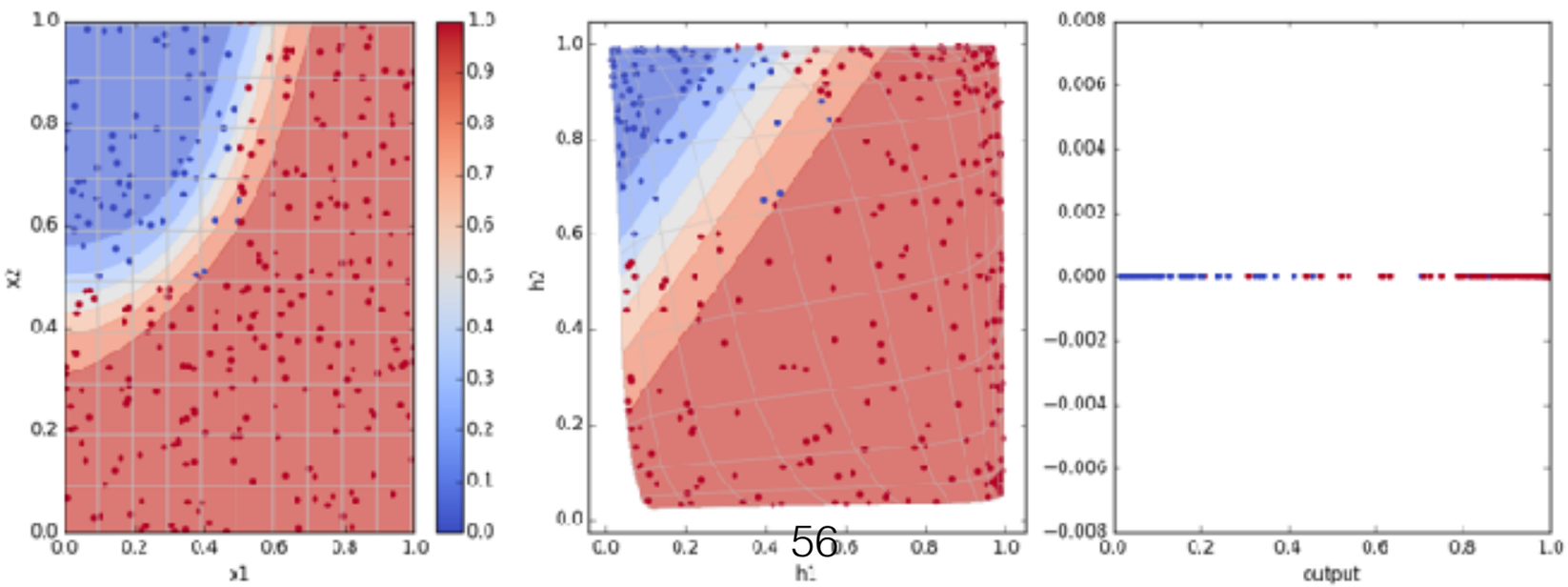




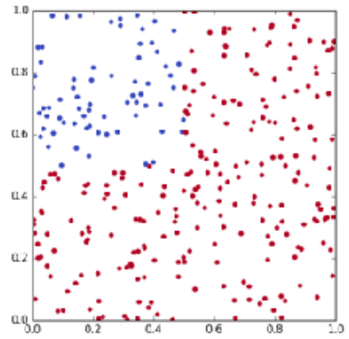
The Angle Data - Sigmoid

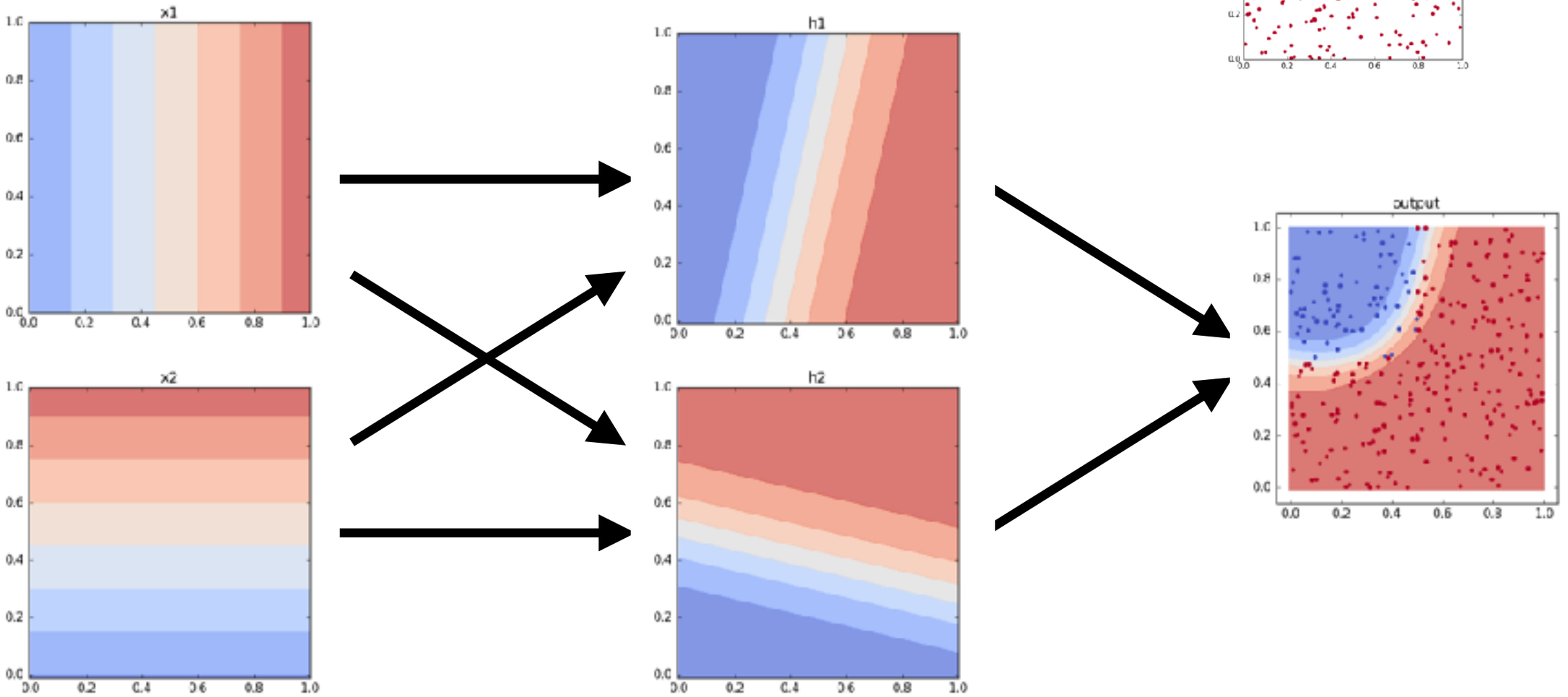


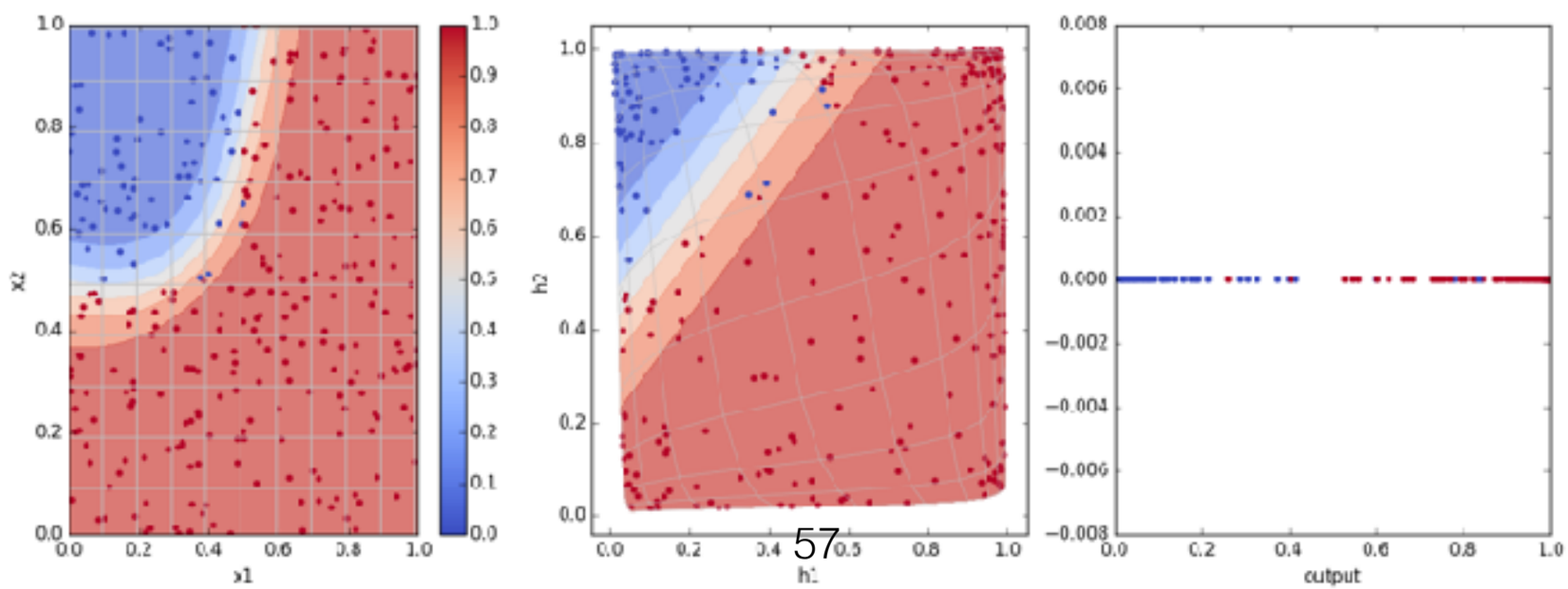




The Angle Data - Sigmoid

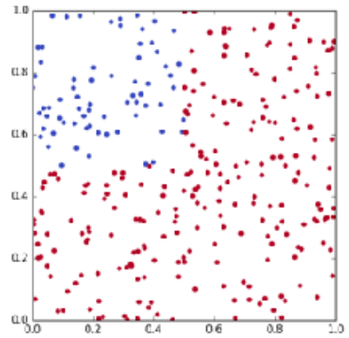


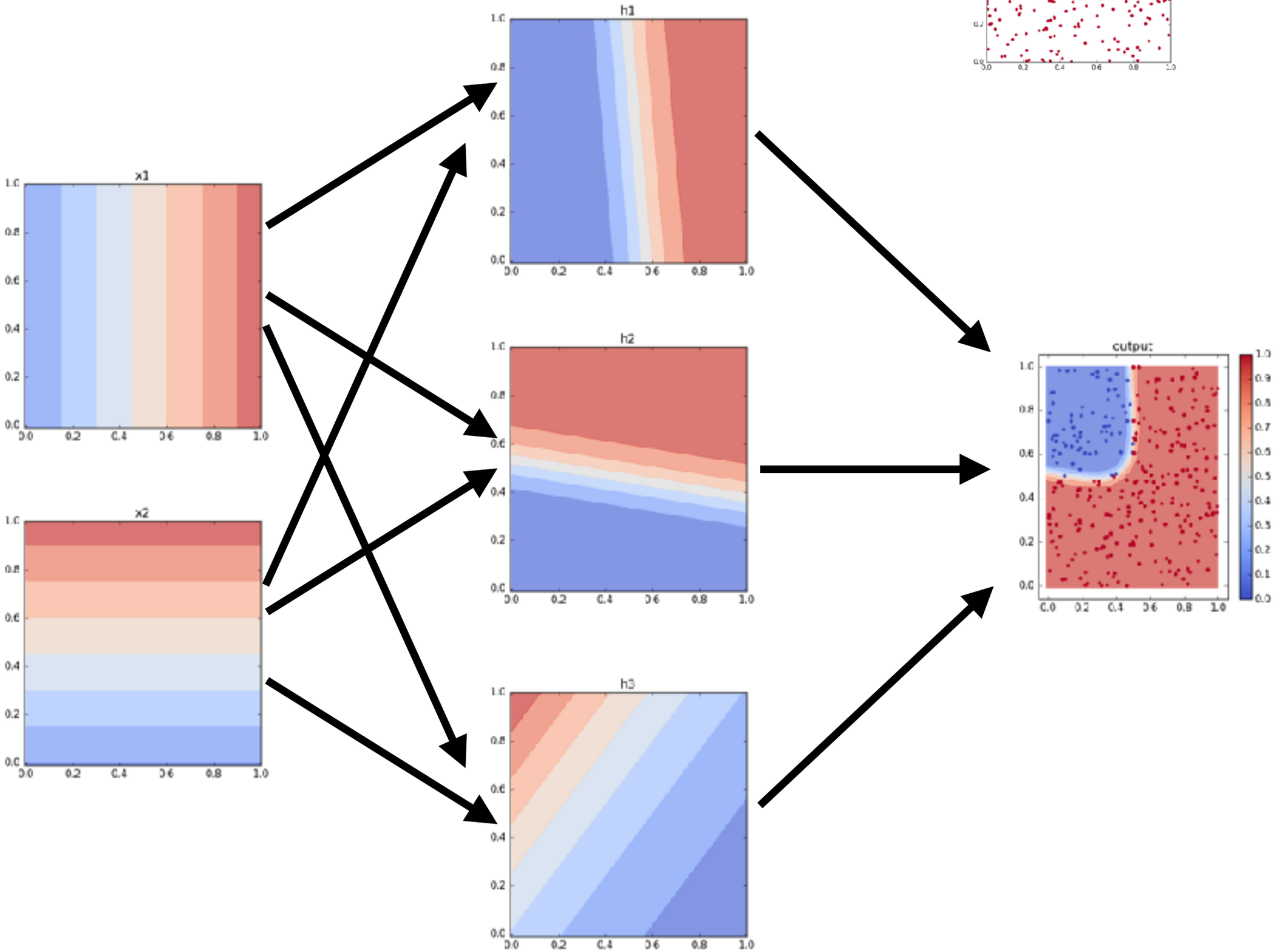




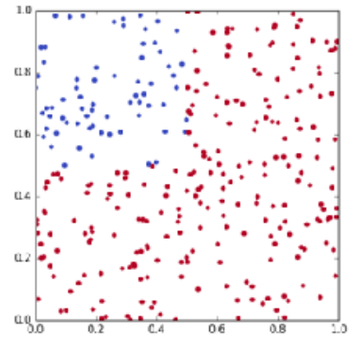
The Angle Data - Sigmoid 3 hidden nodes

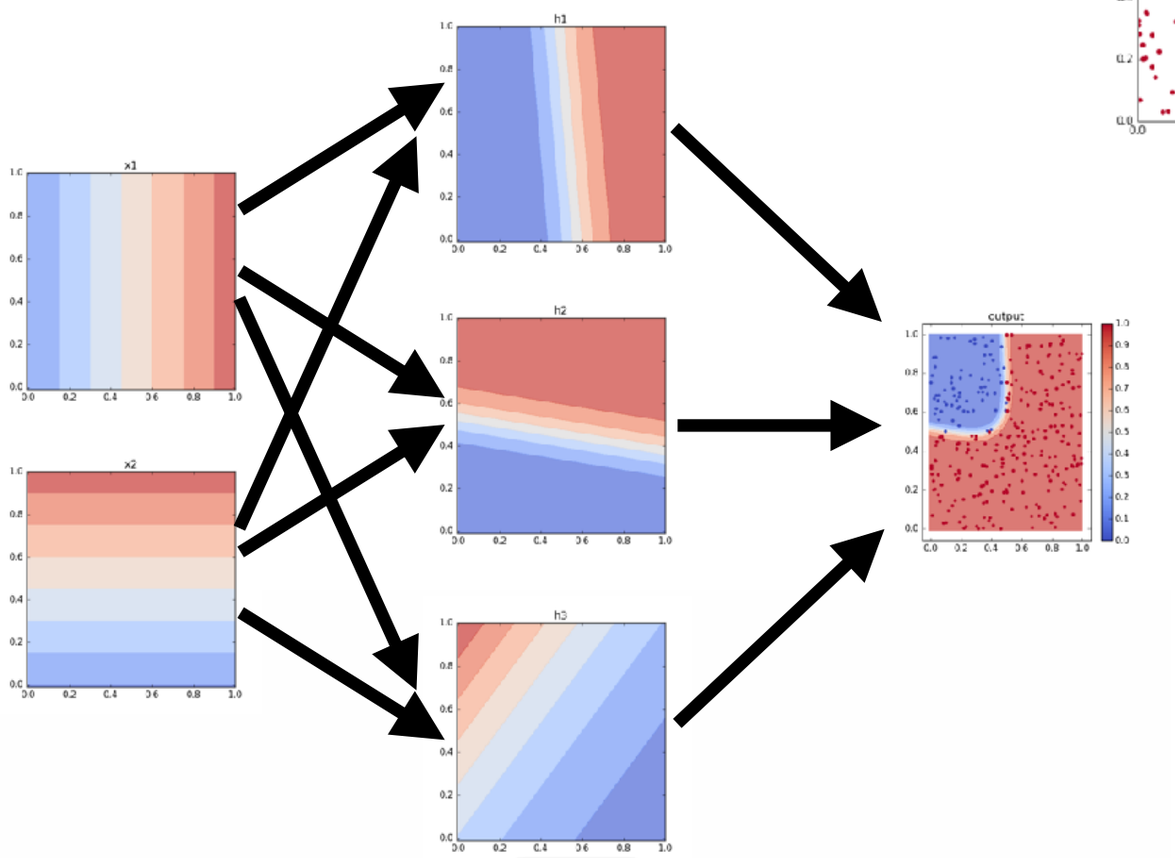
The Angle Data - Sigmoid

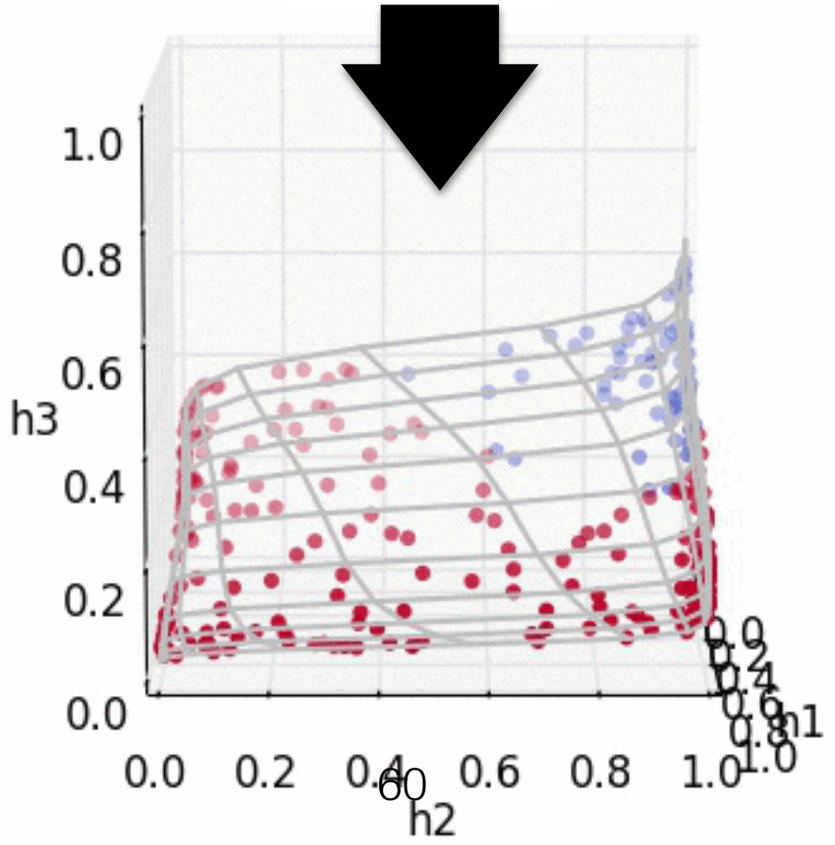




The Angle Data - Sigmoid

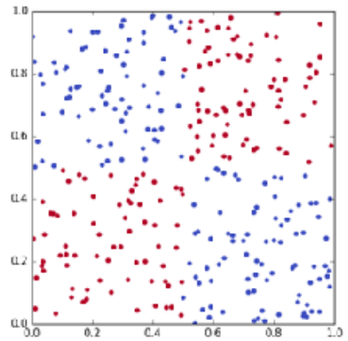


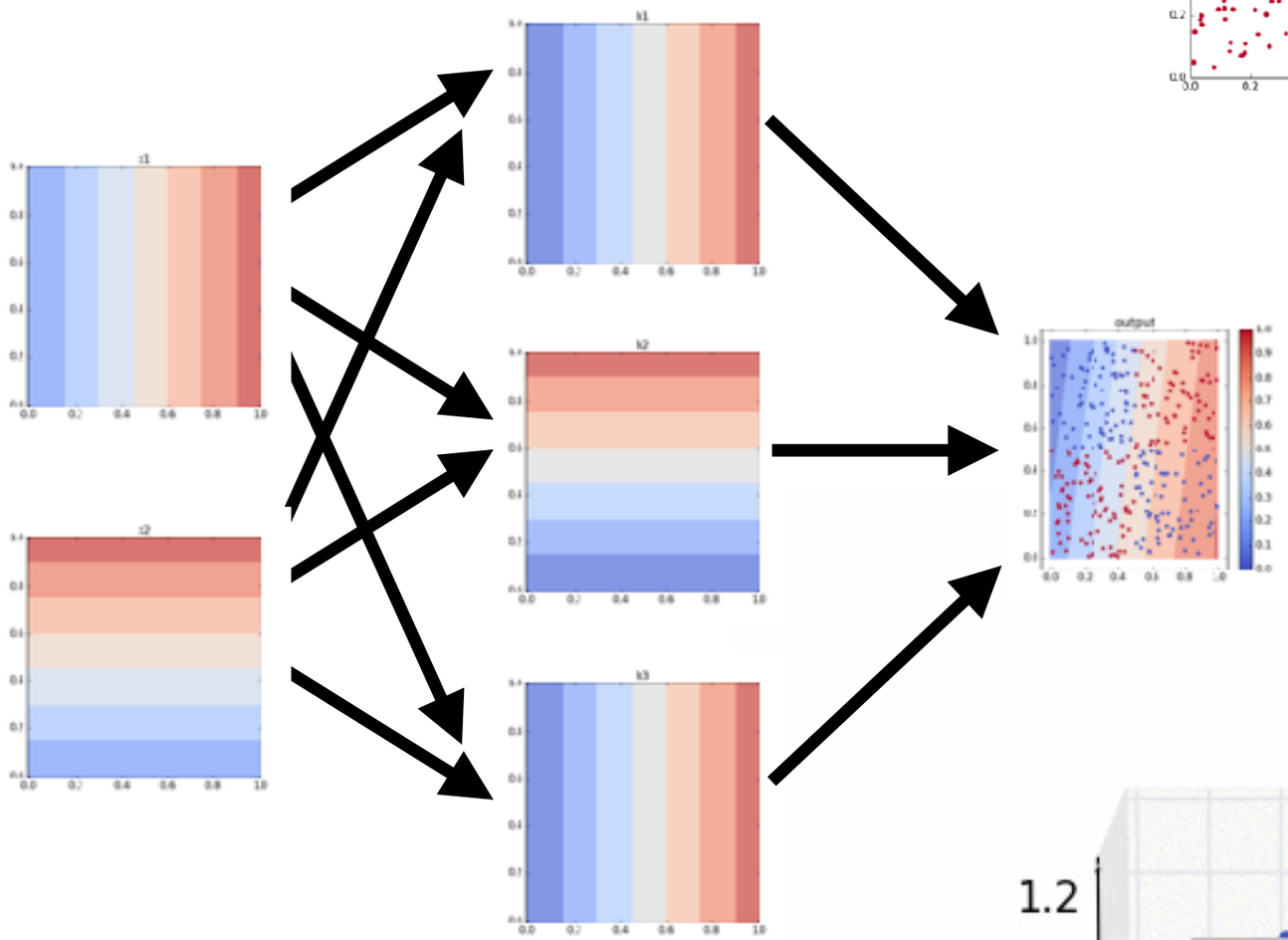


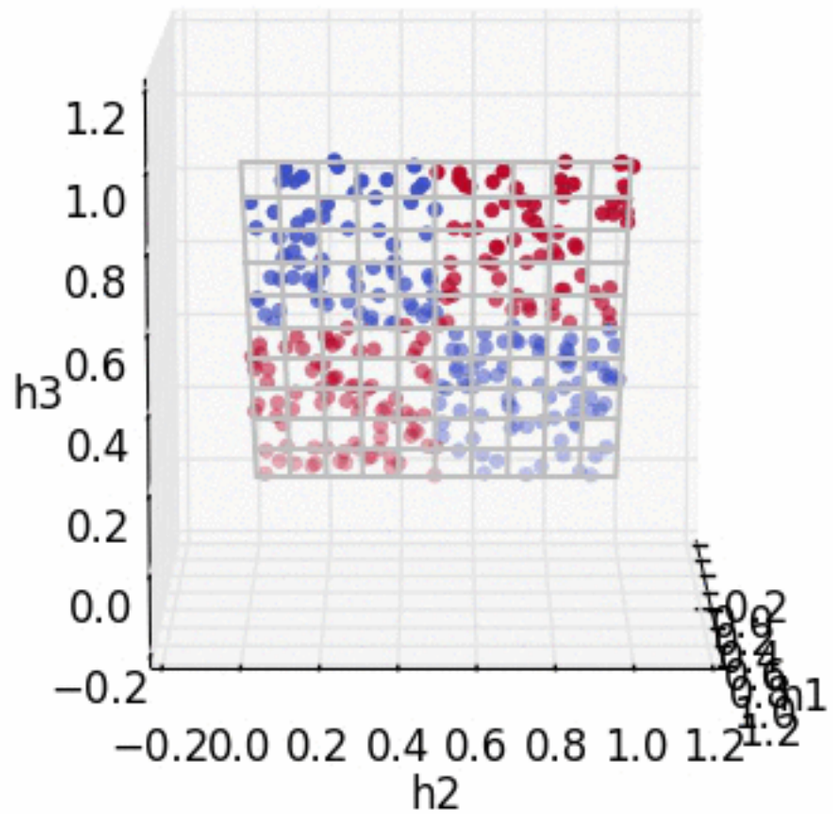


The XOR Data - ReLU

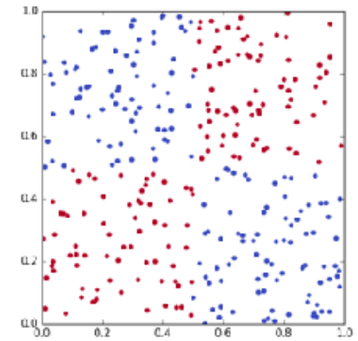
The XOR Data - ReLU

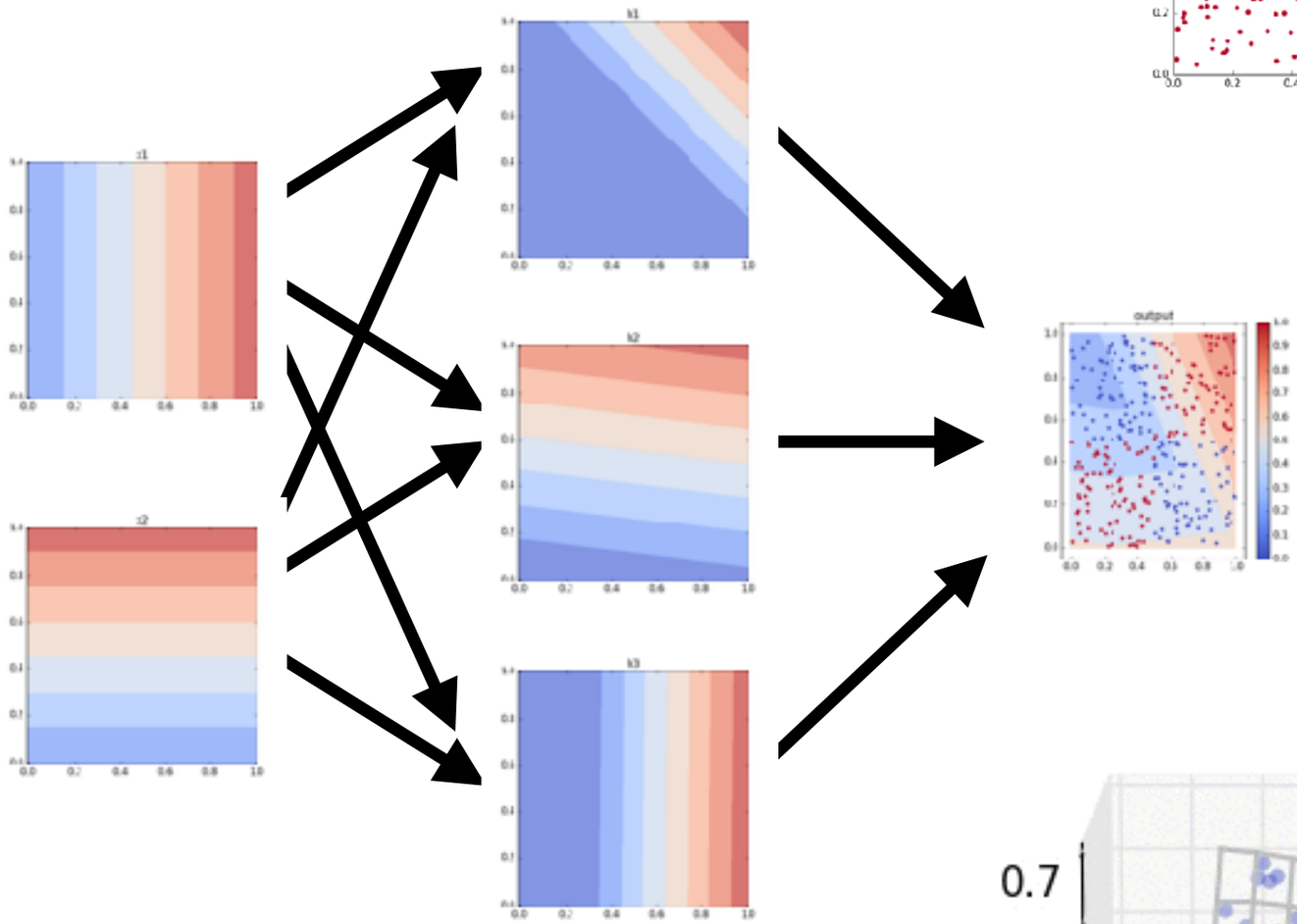


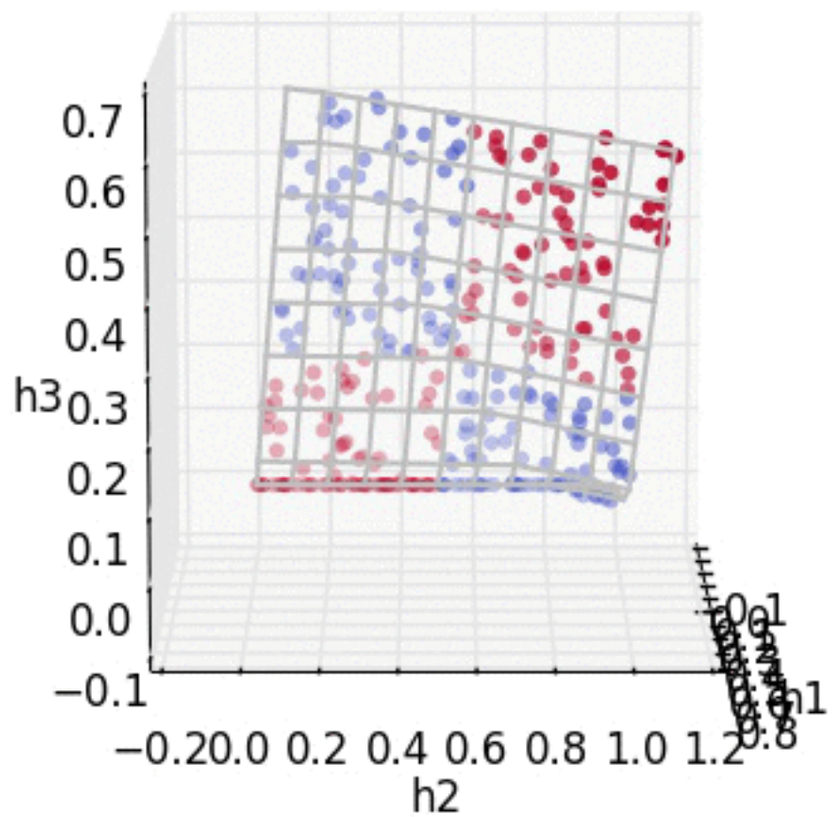




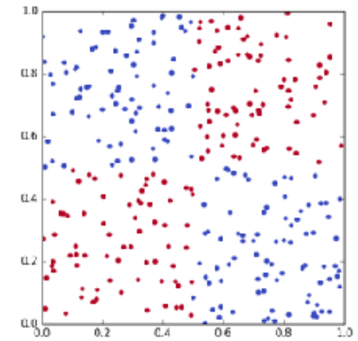
The XOR Data - ReLU

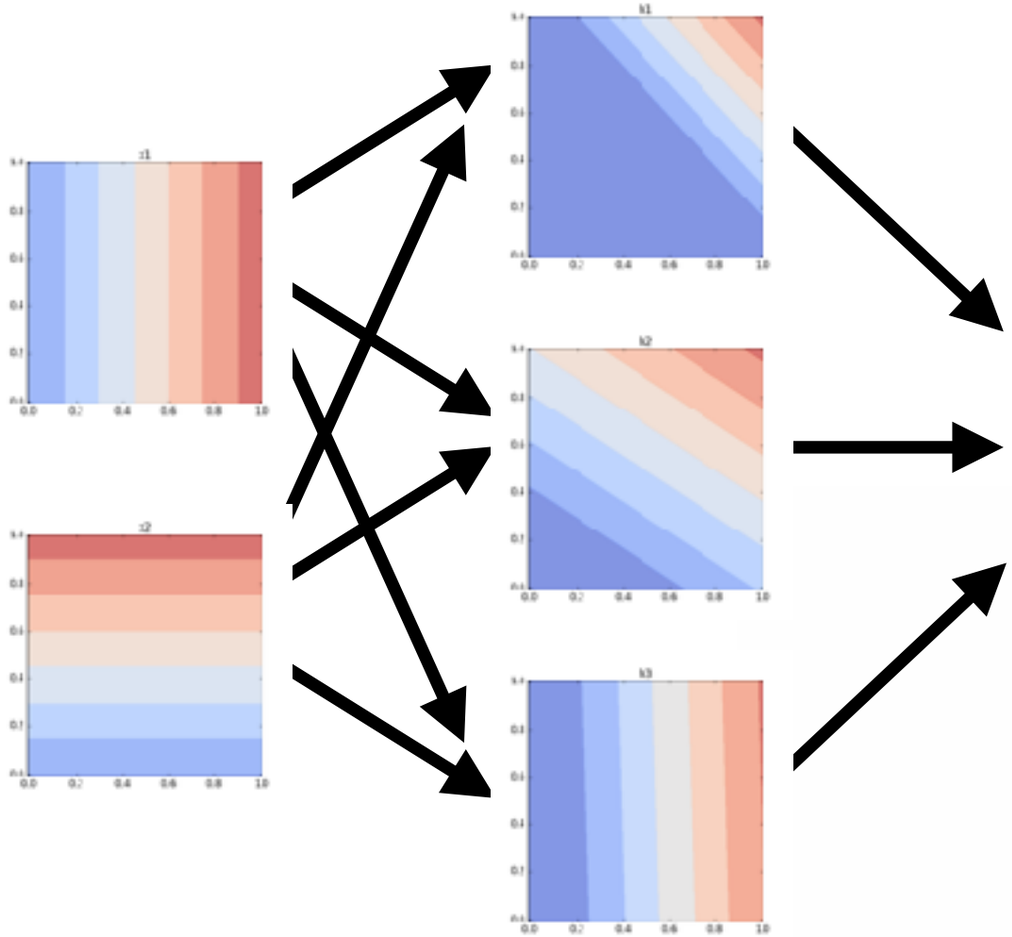


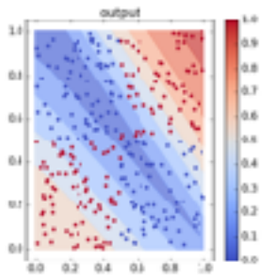


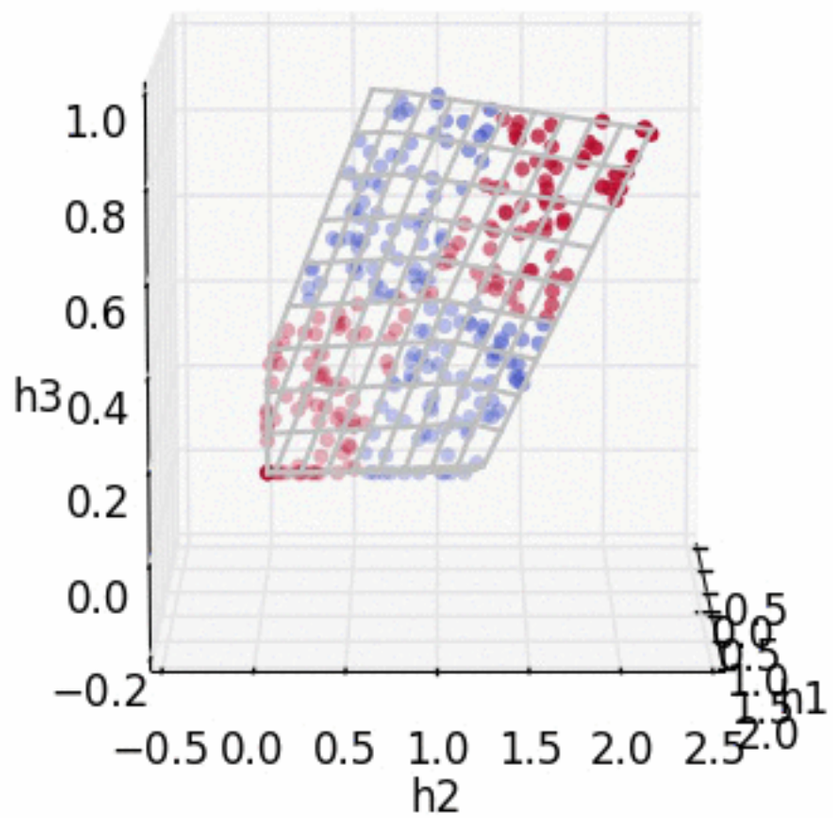


The XOR Data - ReLU

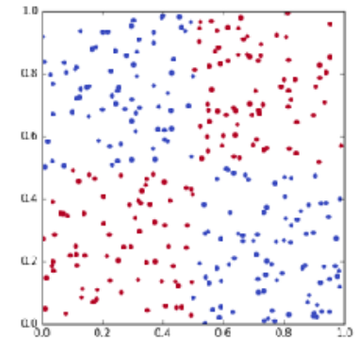


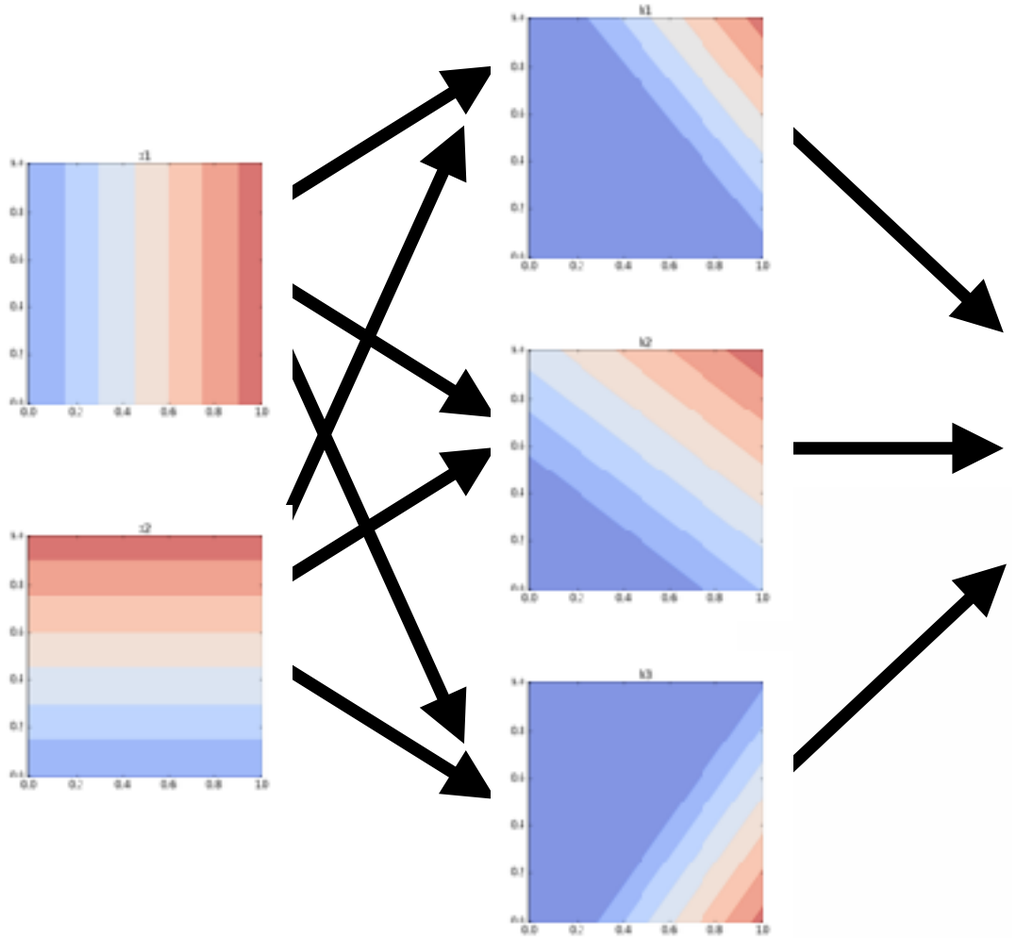


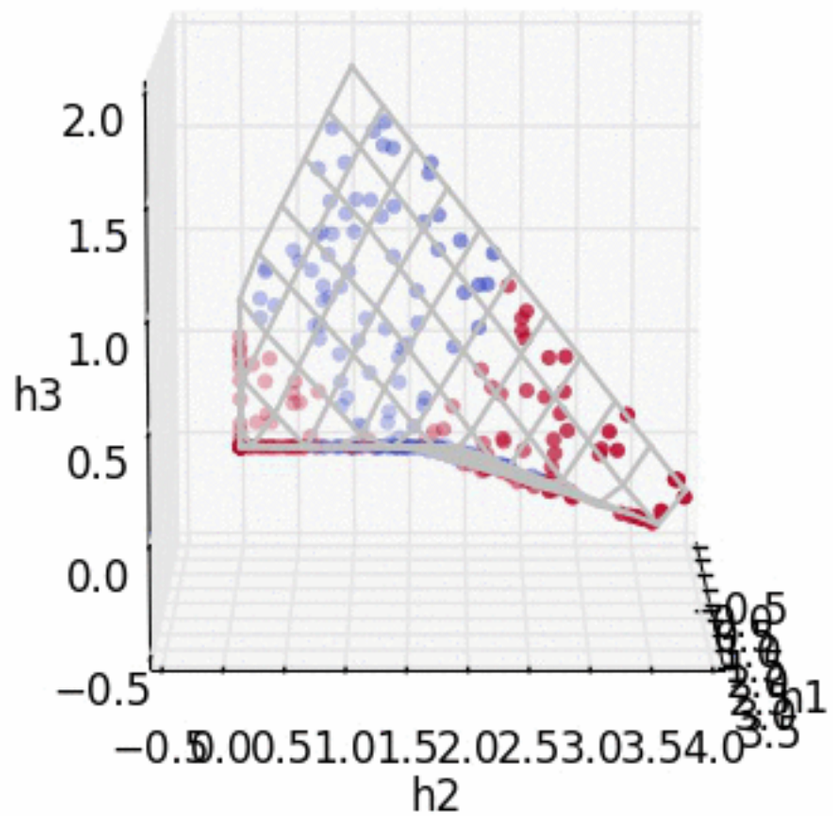




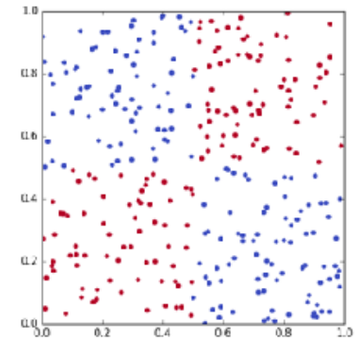
The XOR Data - ReLU

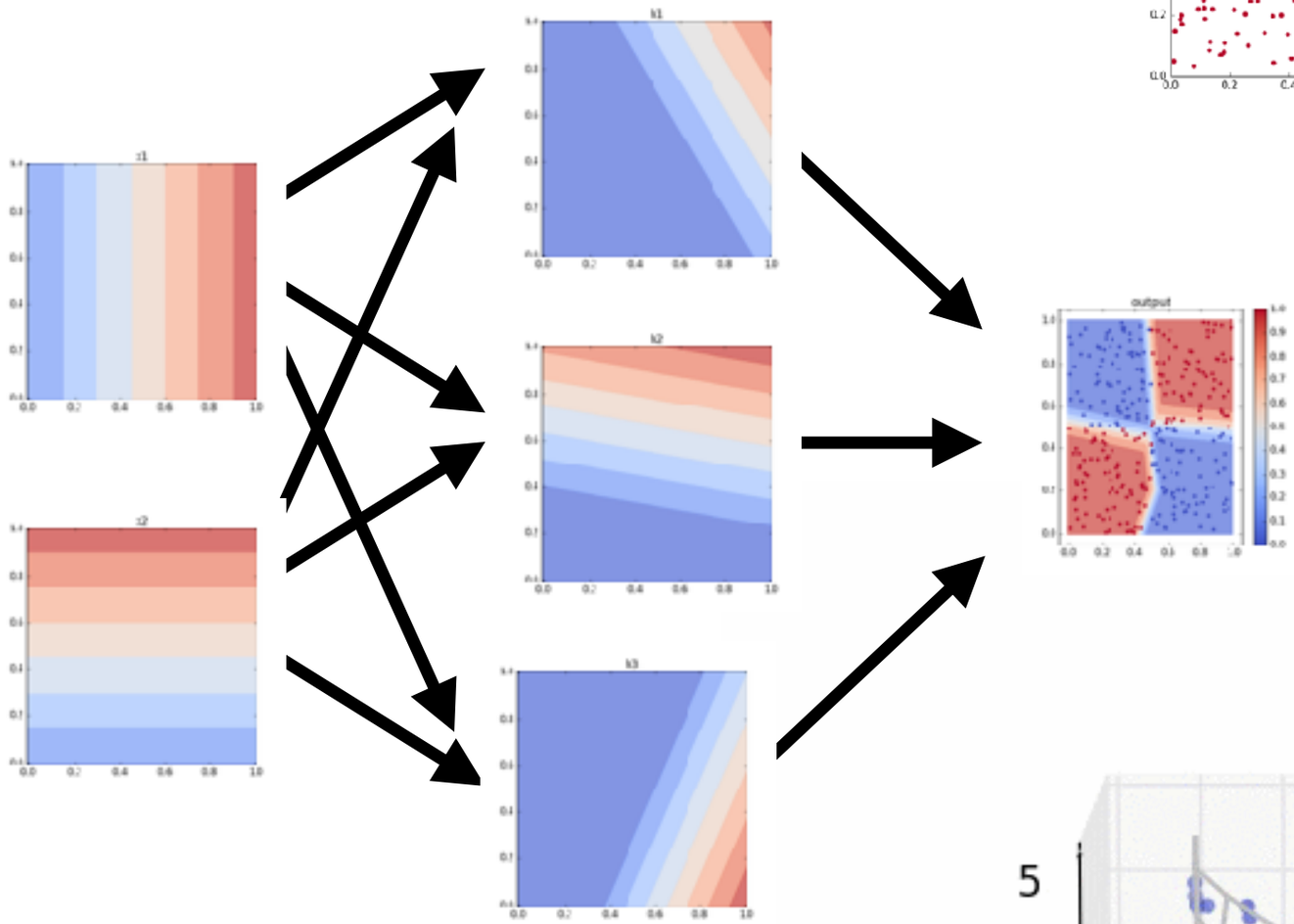


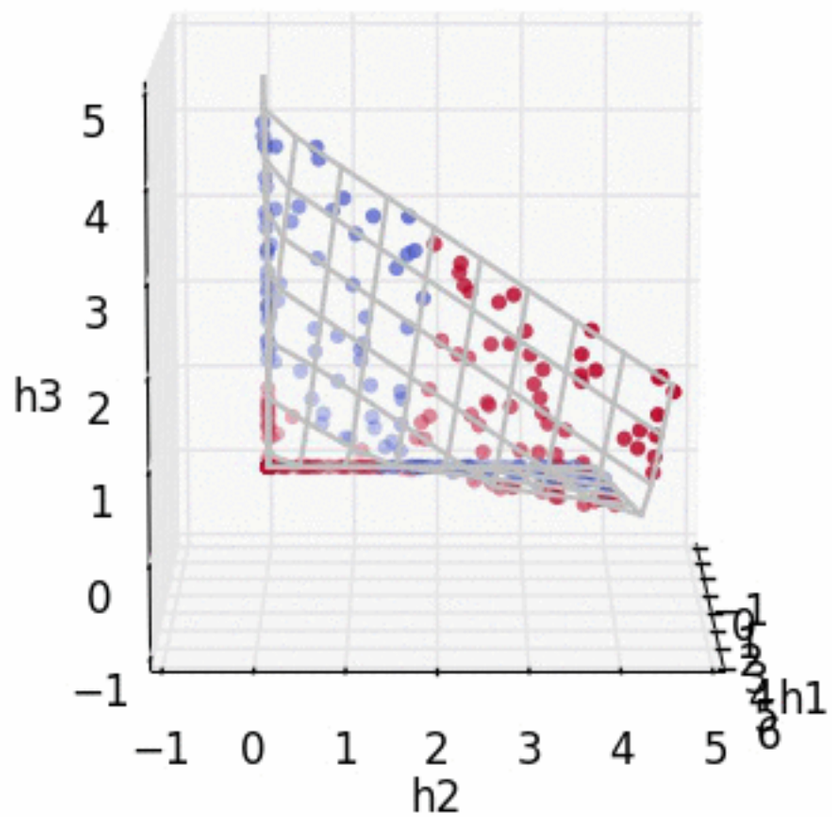




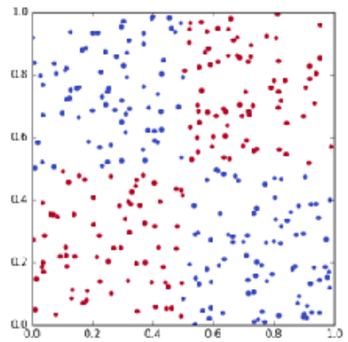
The XOR Data - ReLU

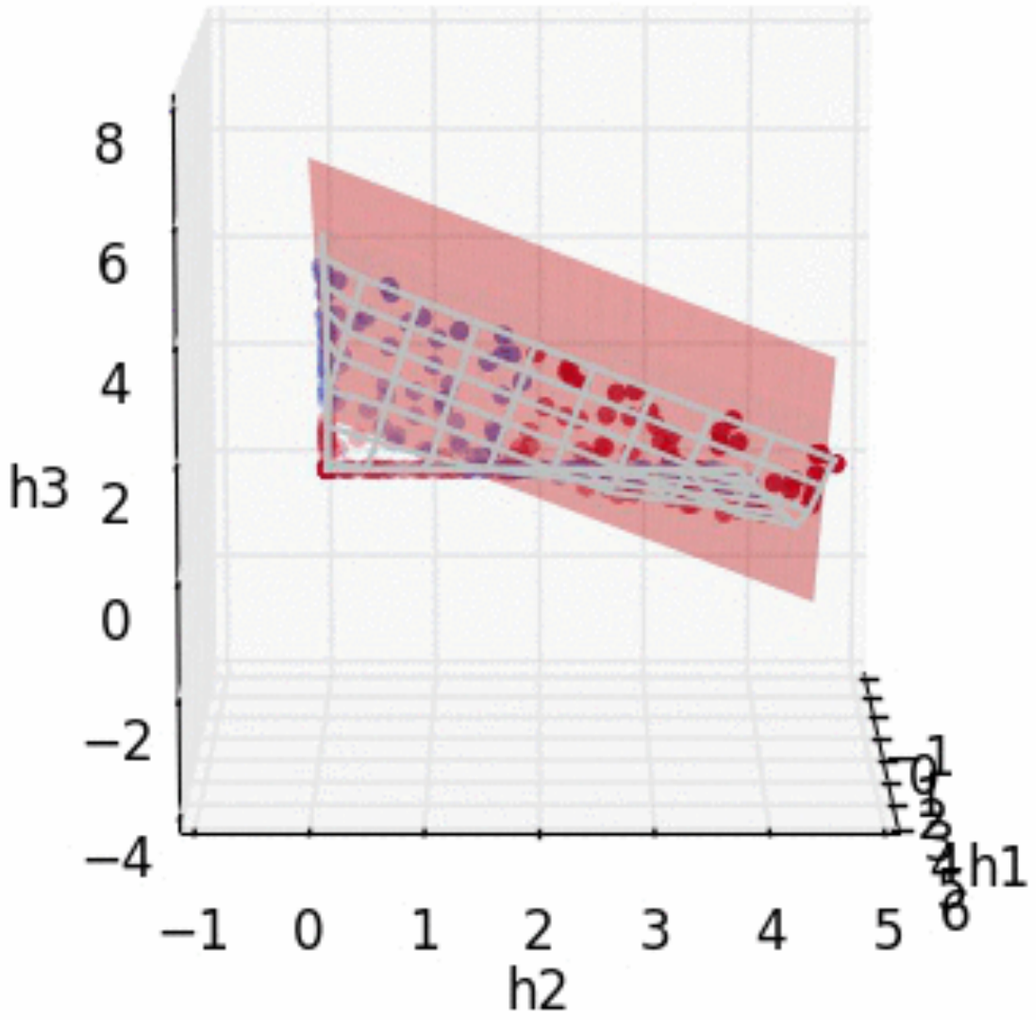






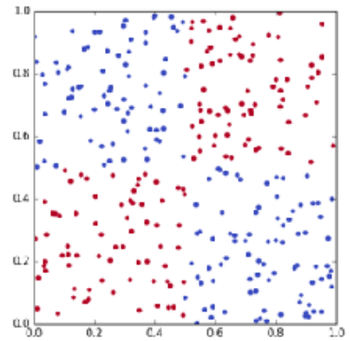
The XOR Data - ReLU

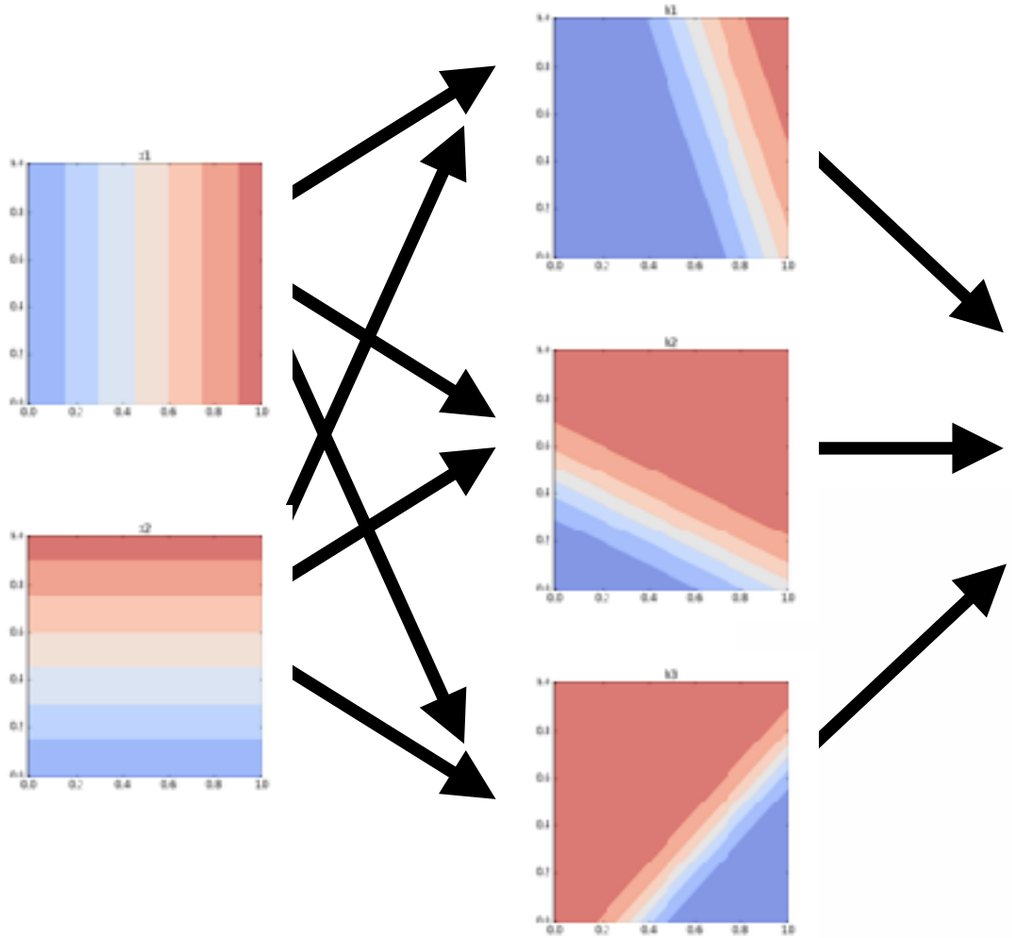


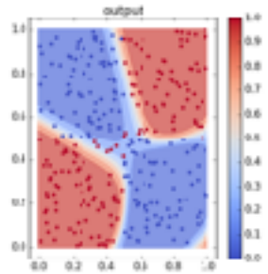


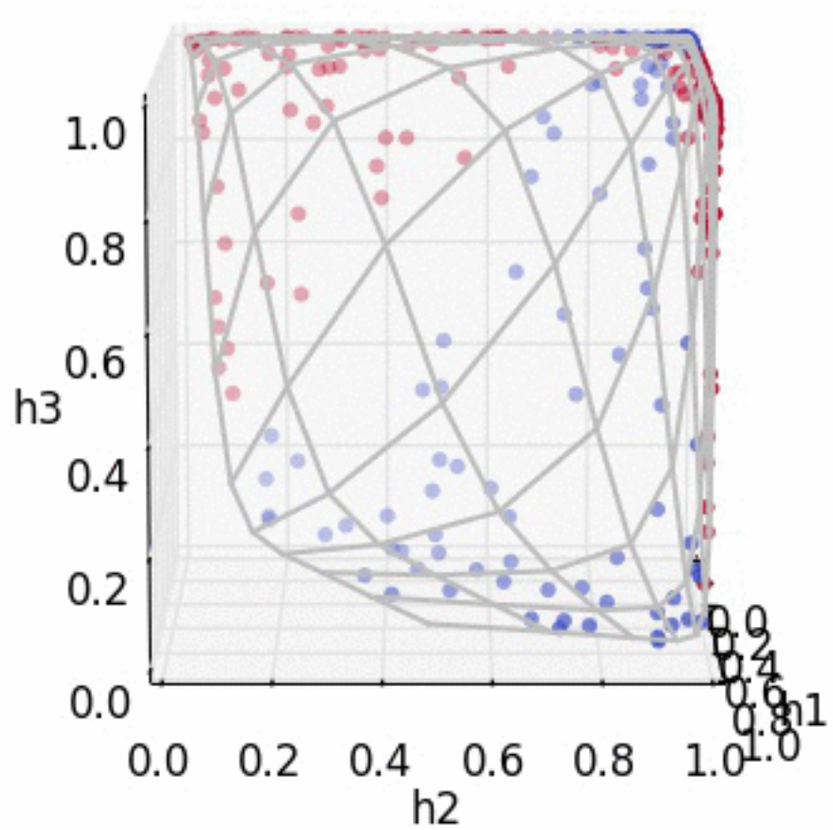
The XOR Data - Sigmoid

The XOR Data - Sigmoid

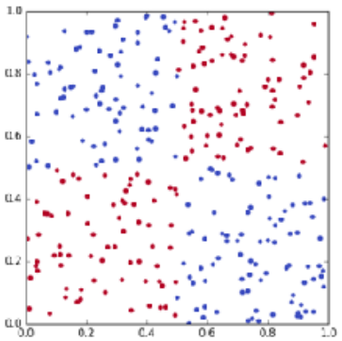


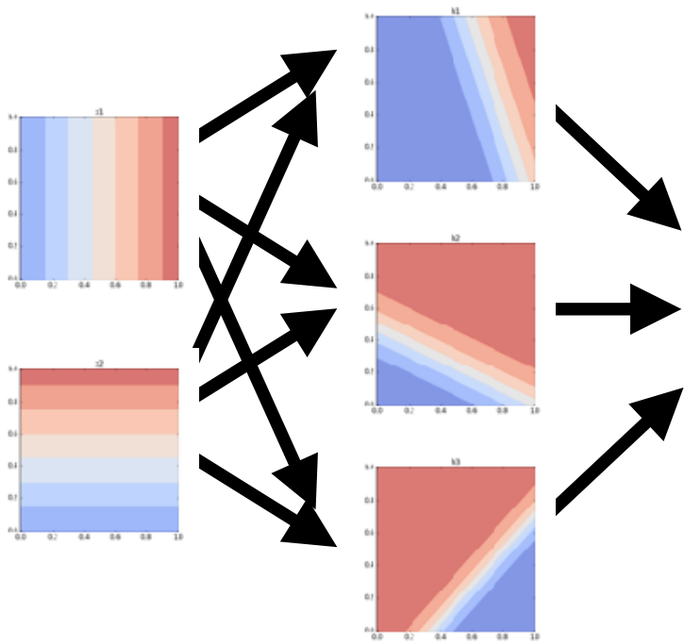


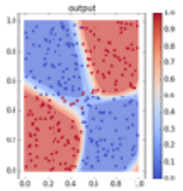


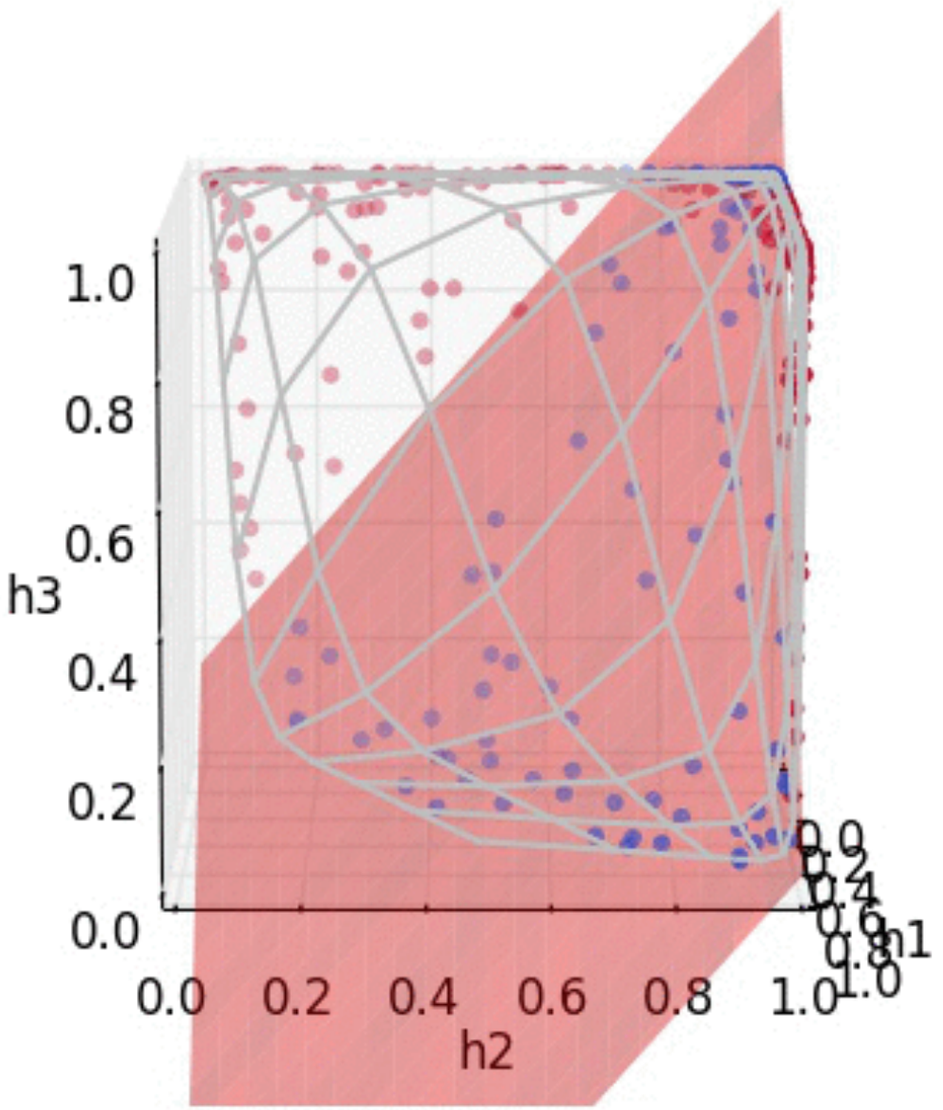


The XOR Data - Sigmoid





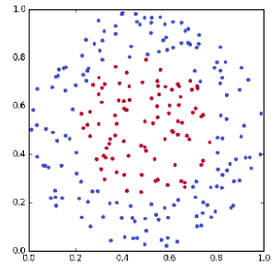


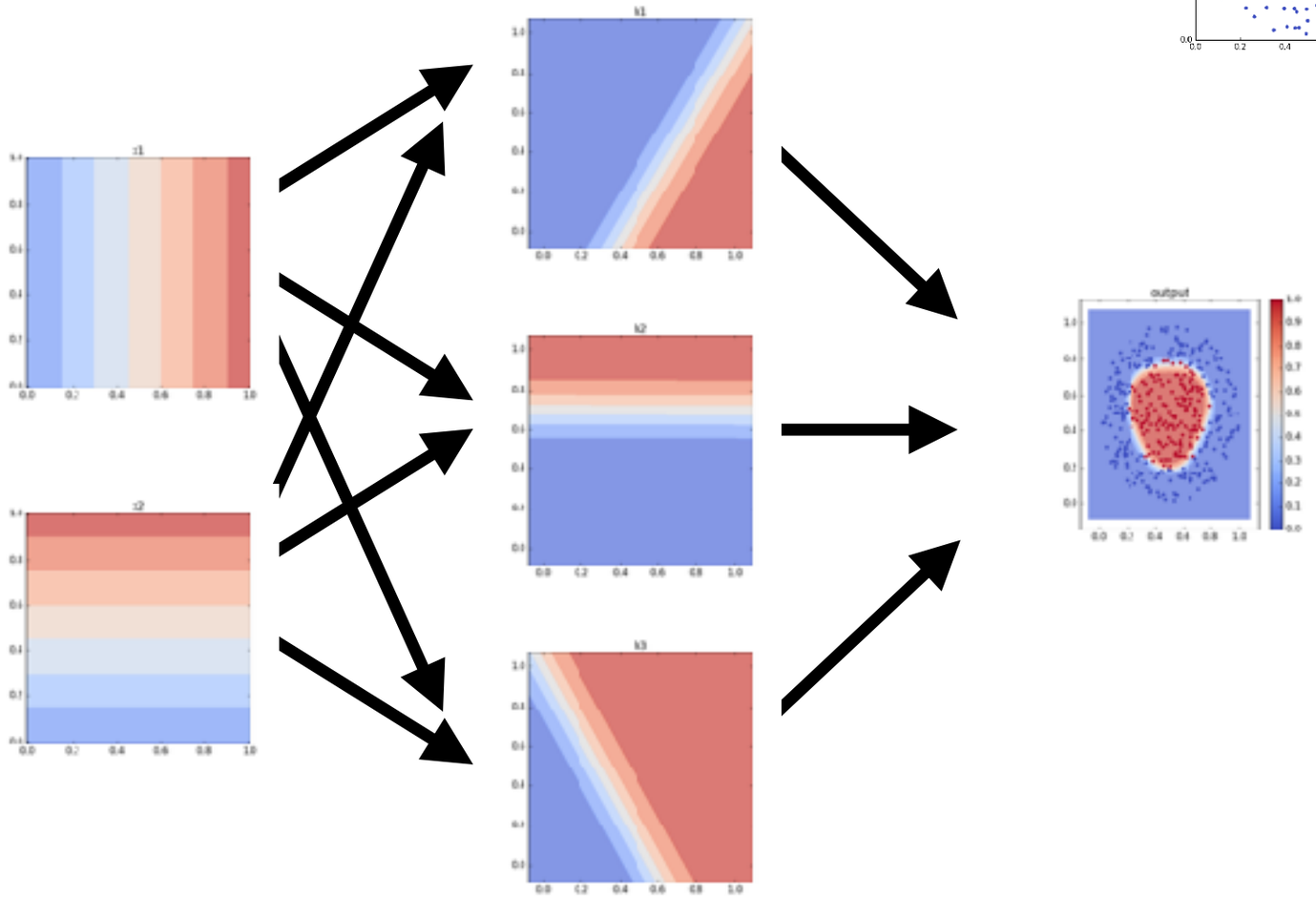


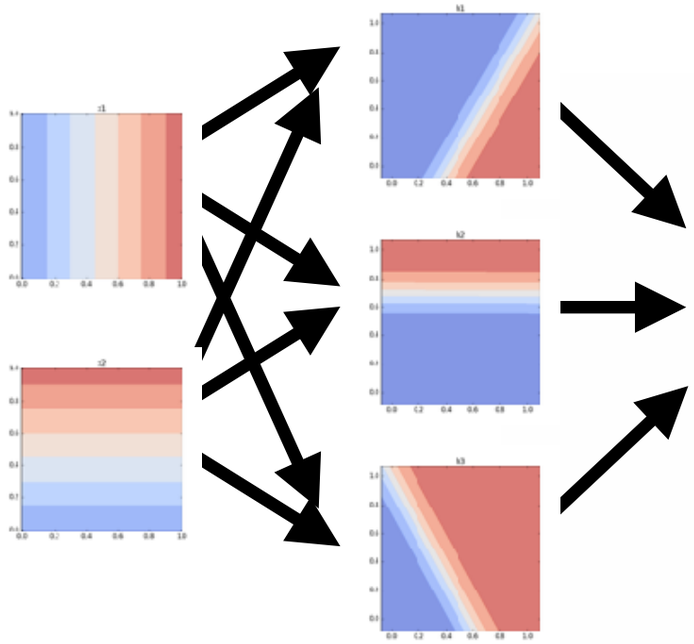
The Ring Data - Sigmoid

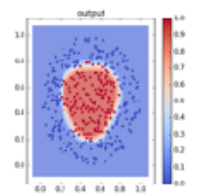
how the manifold is being fold?

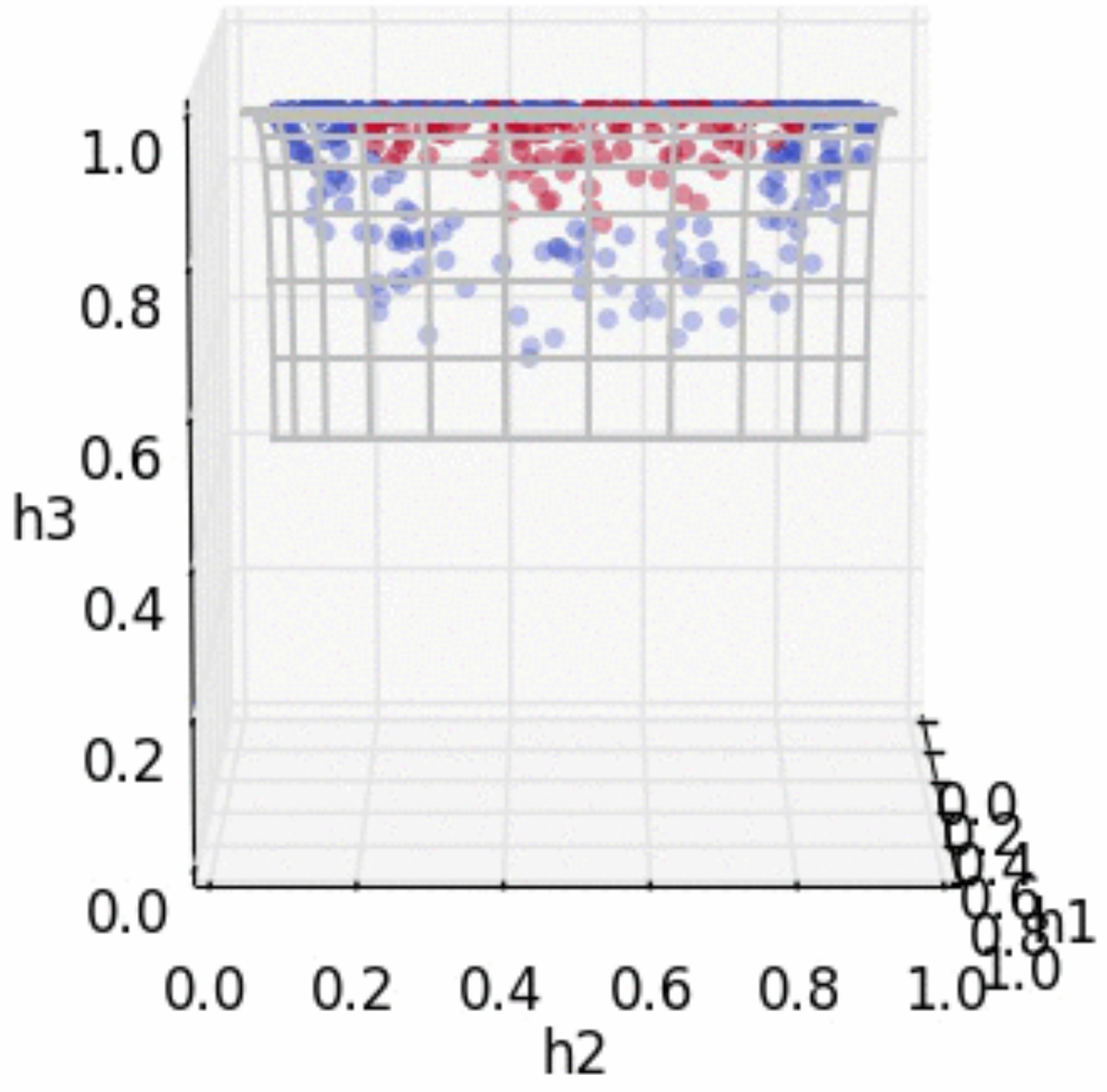
The Ring Data - Sigmoid

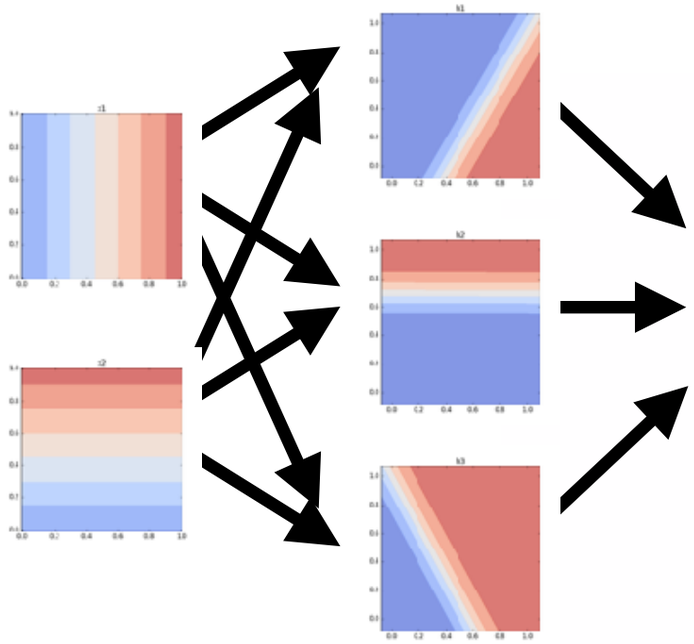


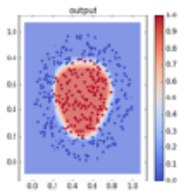


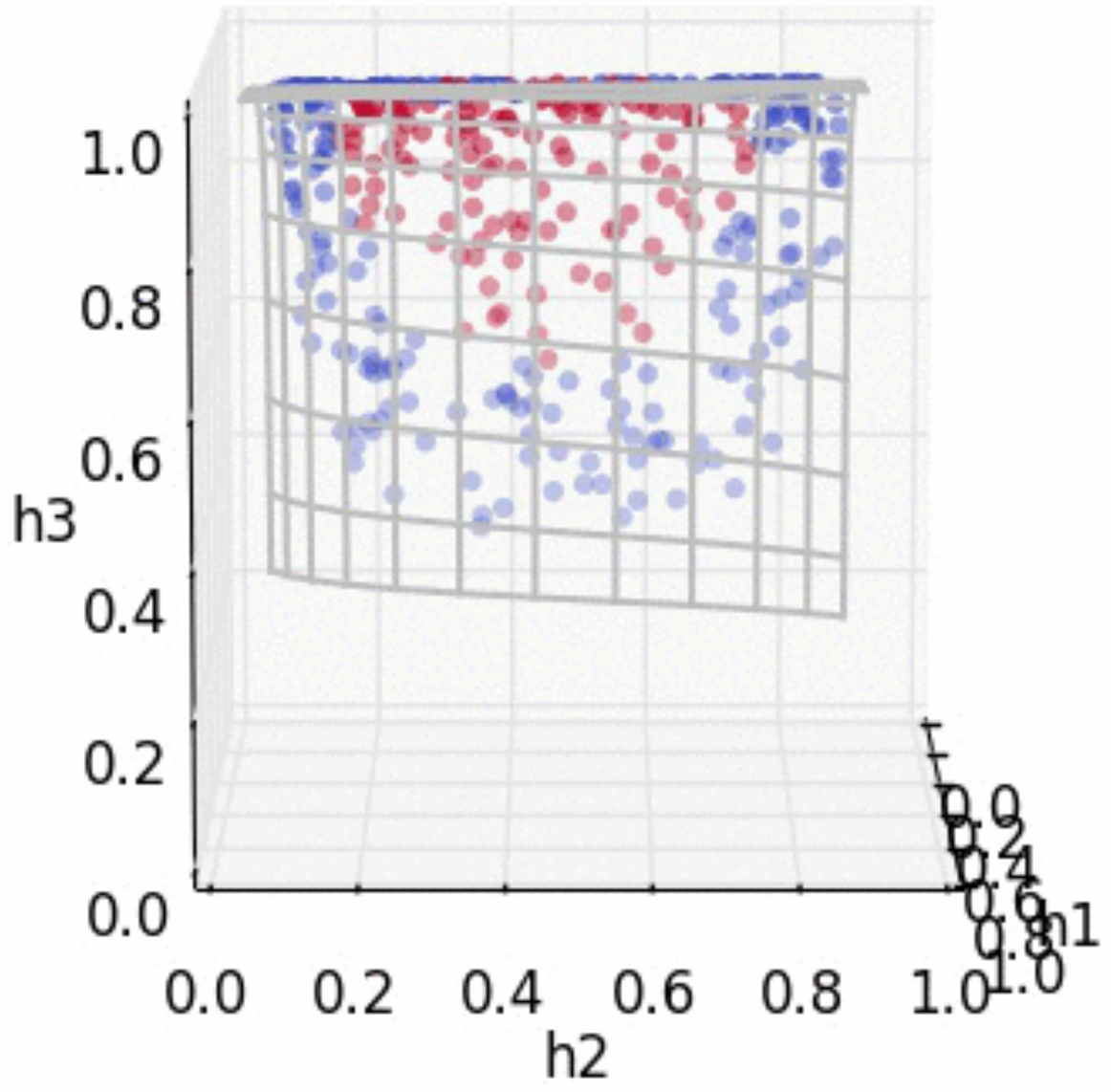

z1
z2
k1
k2
k3

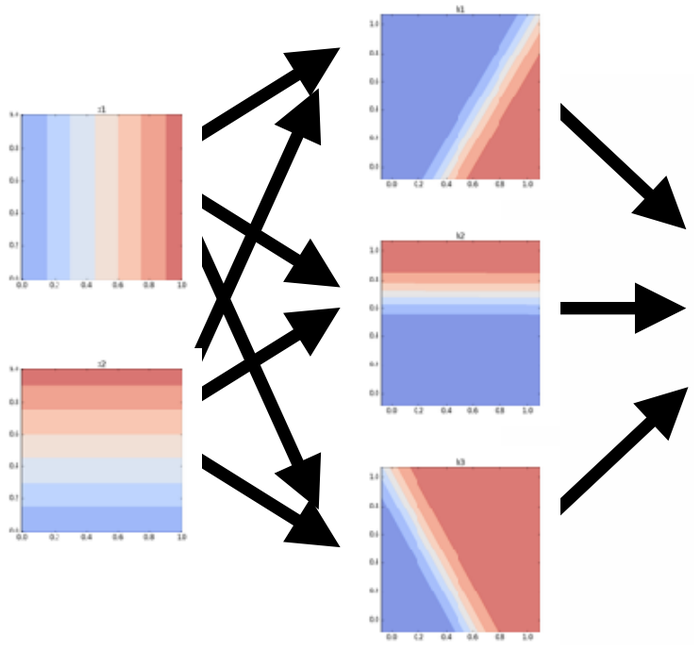

output

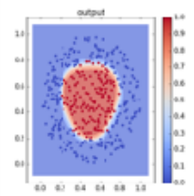

1.0
0.8
0.6
0.4
0.2
0.0
h3
0.0
0.2
0.4
0.6
0.8
1.0
h2
h1

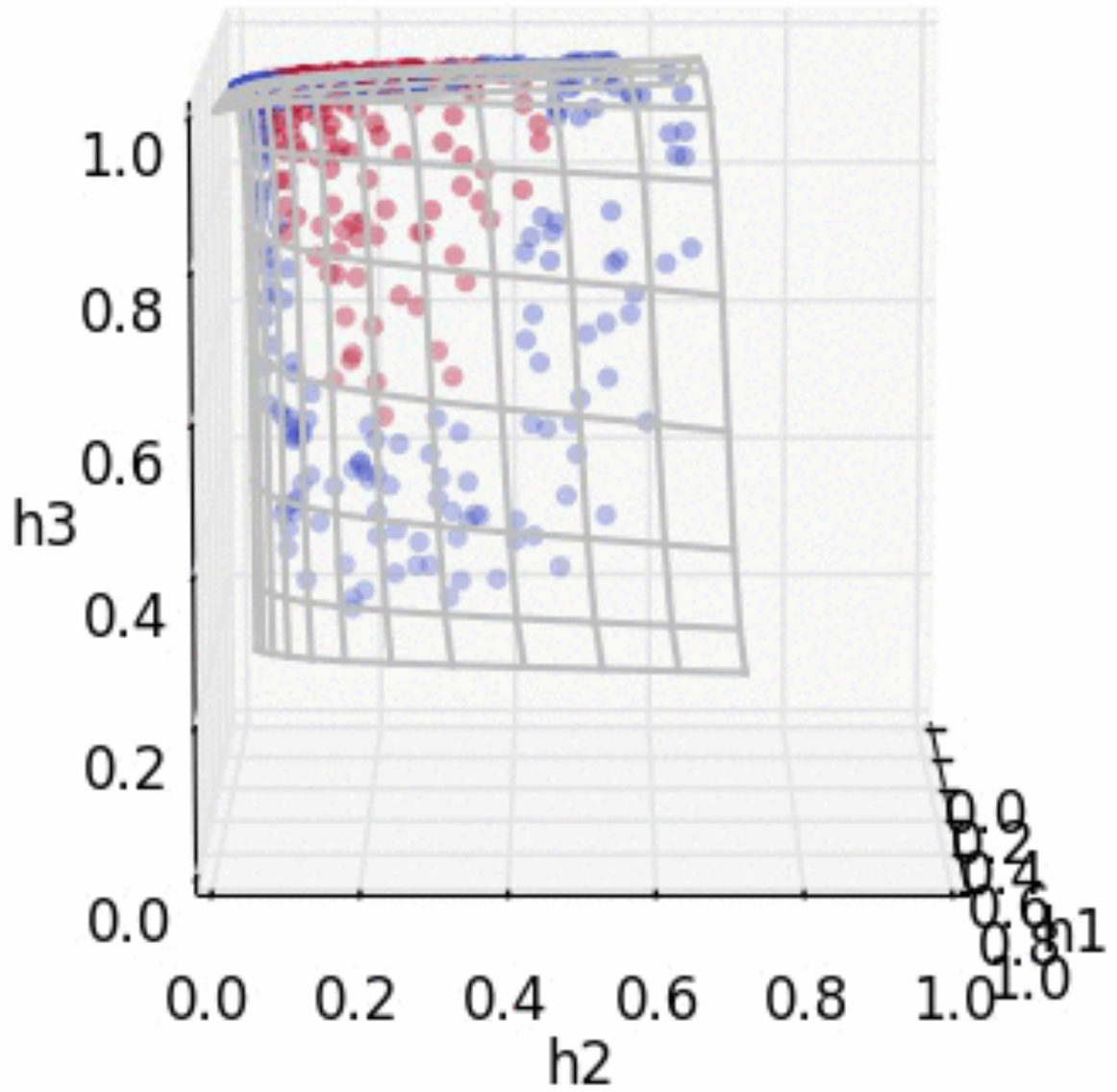

z1
z2
k1
k2
k3

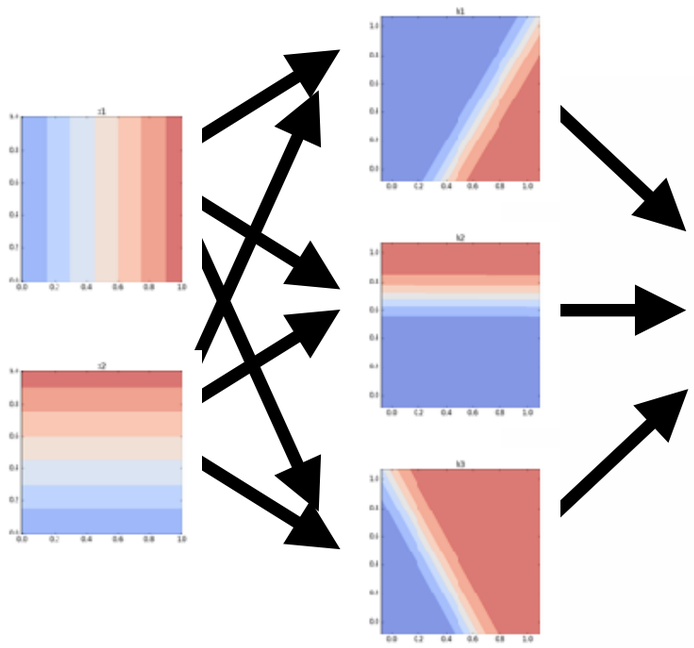

output

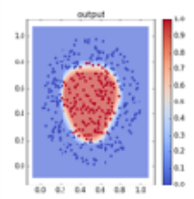

1.0
0.8
0.6
0.4
0.2
0.0
h3
0.0
0.2
0.4
0.6
0.8
1.0
h2
h1

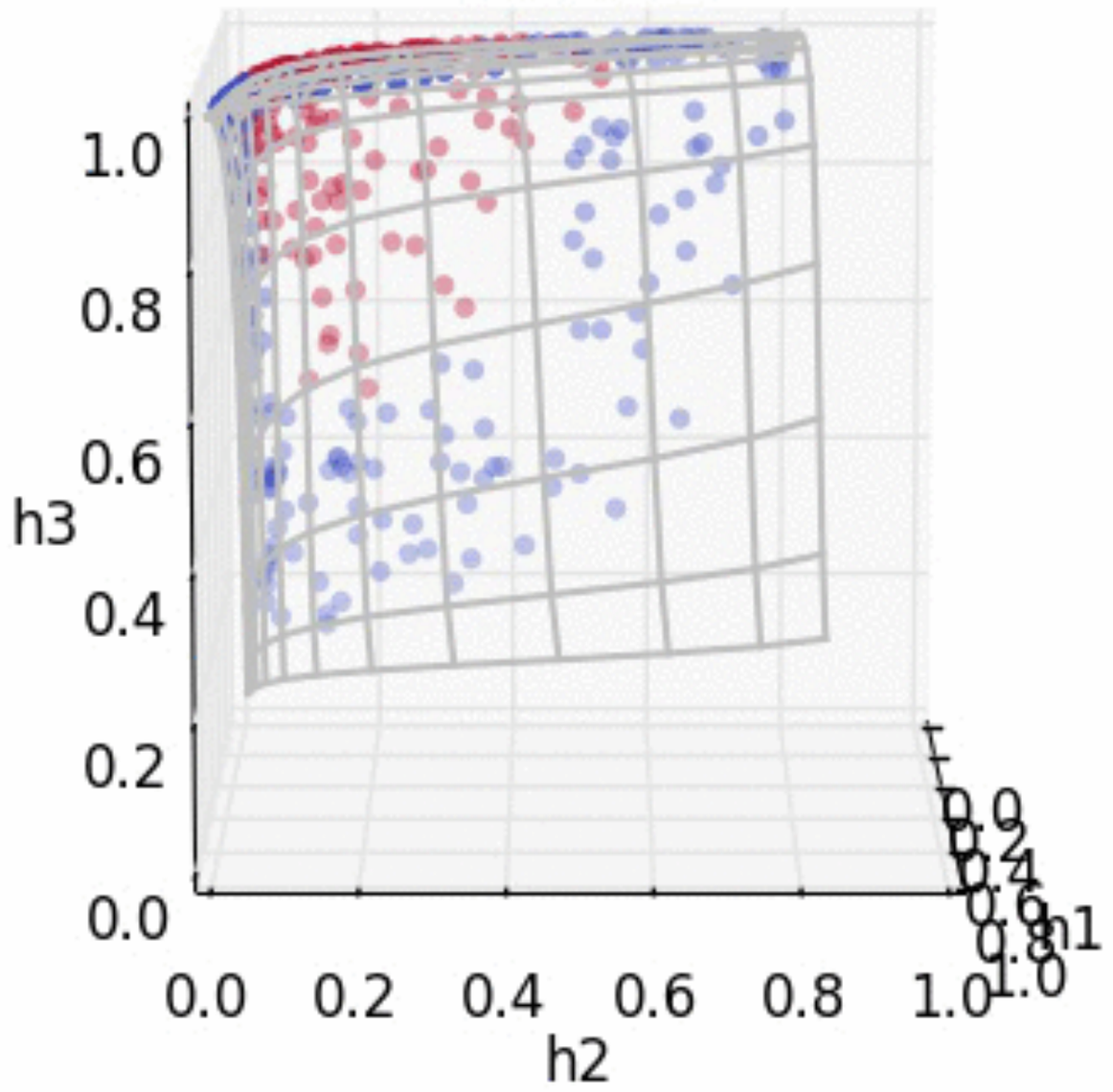

z1
z2
k1
k2
k3

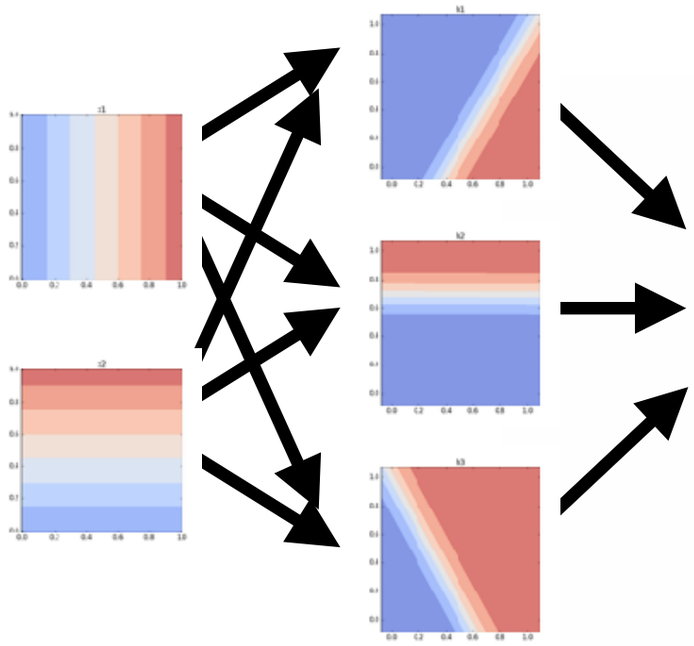

output

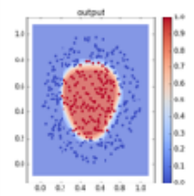

1.0
0.8
0.6
0.4
0.2
0.0
h3
0.0
0.2
0.4
0.6
0.8
1.0
h2
h1

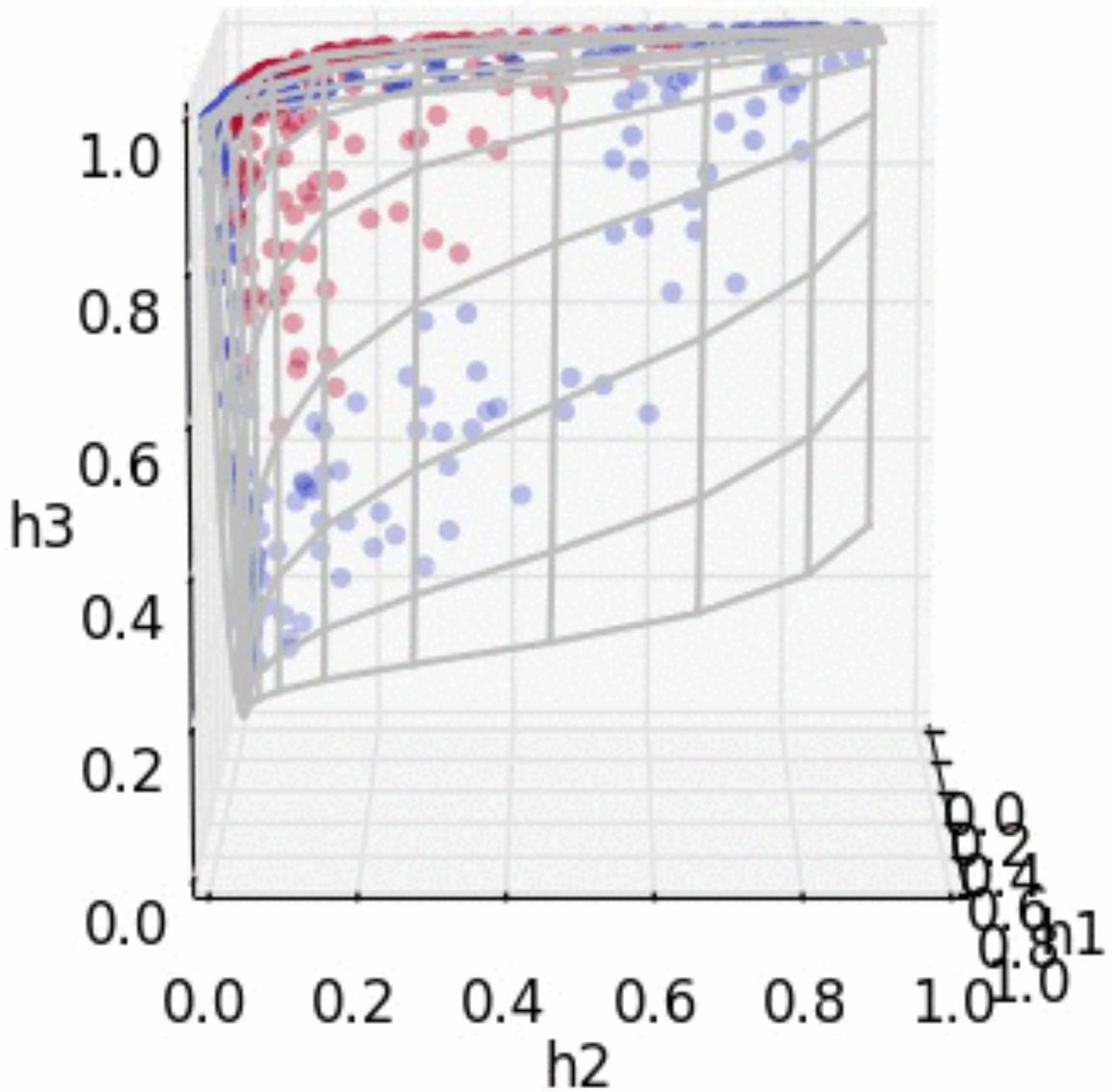

z1
z2
k1
k2
k3

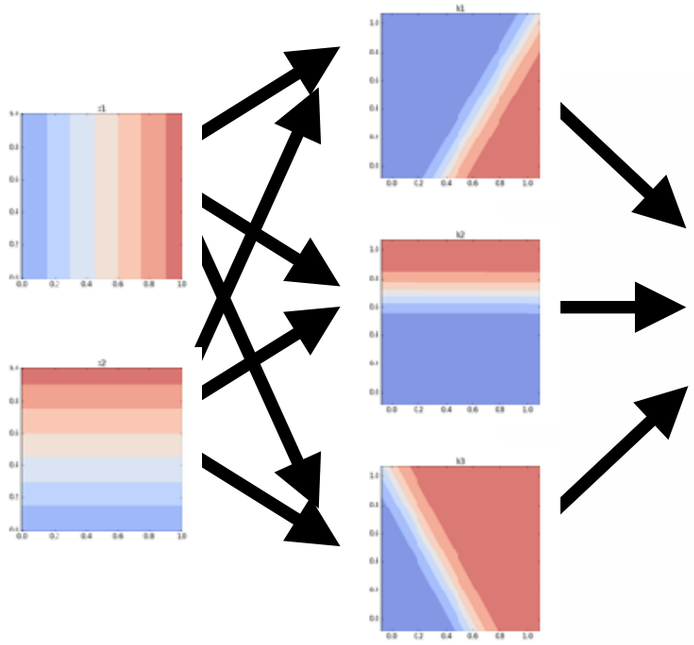

output

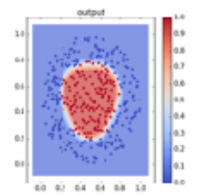

1.0
0.8
0.6
0.4
0.2
0.0
h3
0.0
0.2
0.4
0.6
0.8
1.0
h2
h1

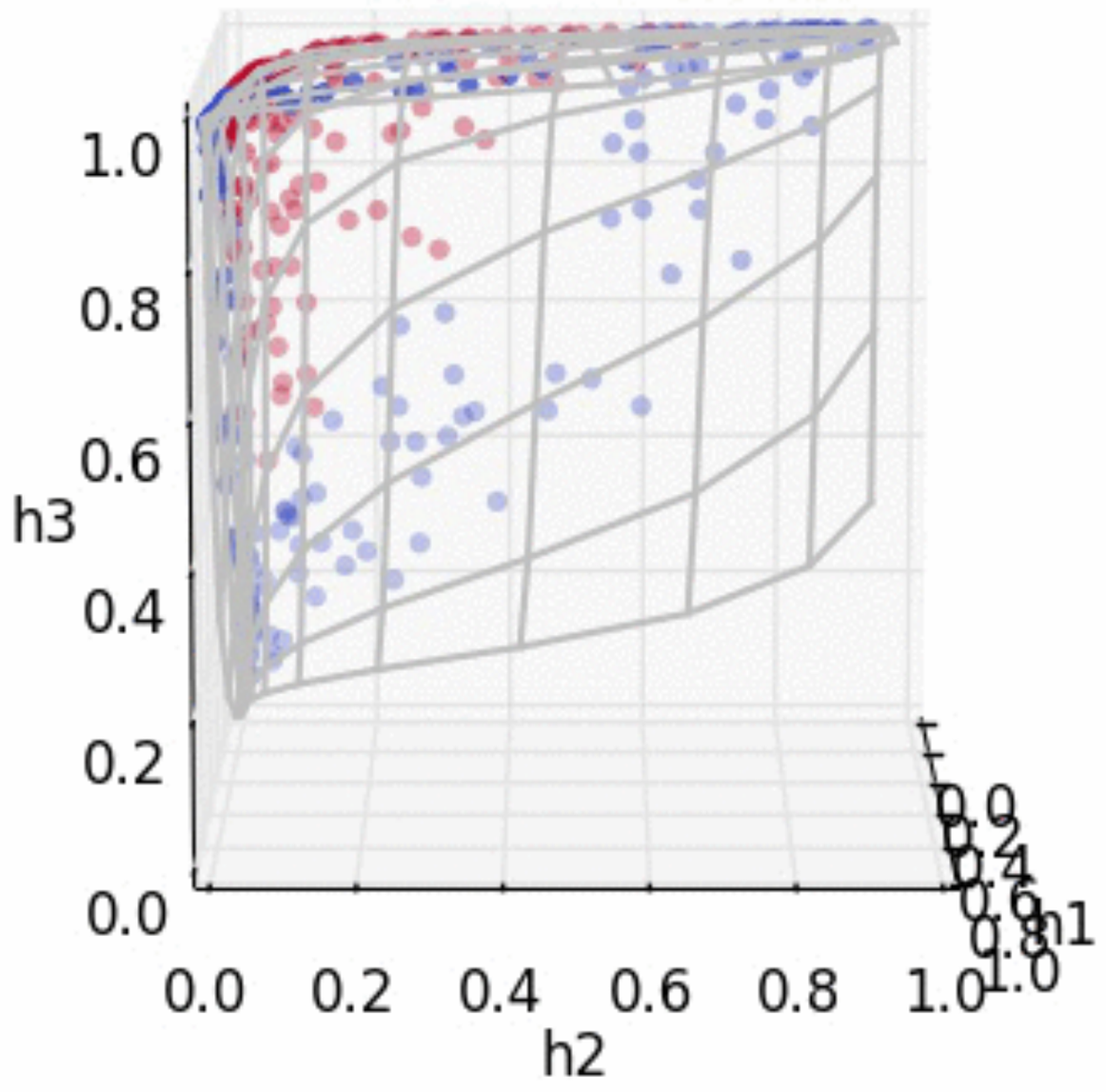

z1
z2
k1
k2
k3

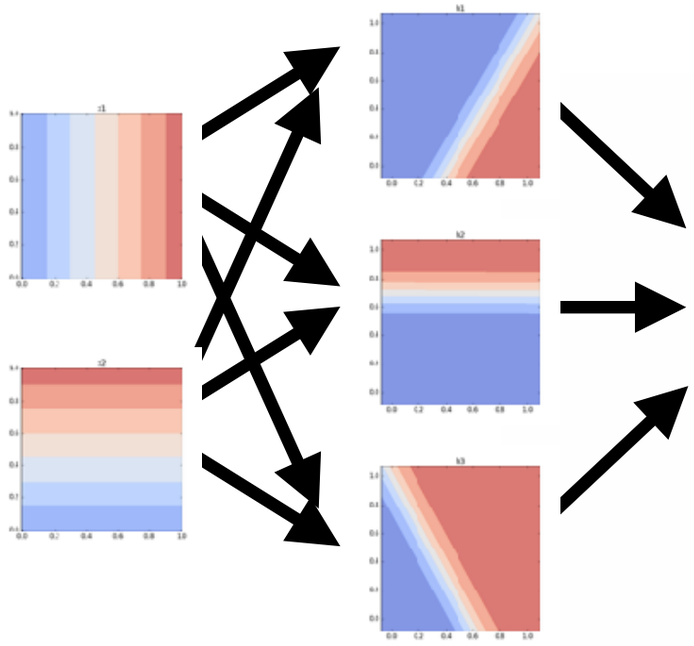

output

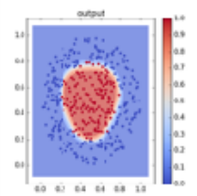

1.0
0.8
0.6
0.4
0.2
0.0
h3
0.0
0.2
0.4
0.6
0.8
1.0
h2
h1

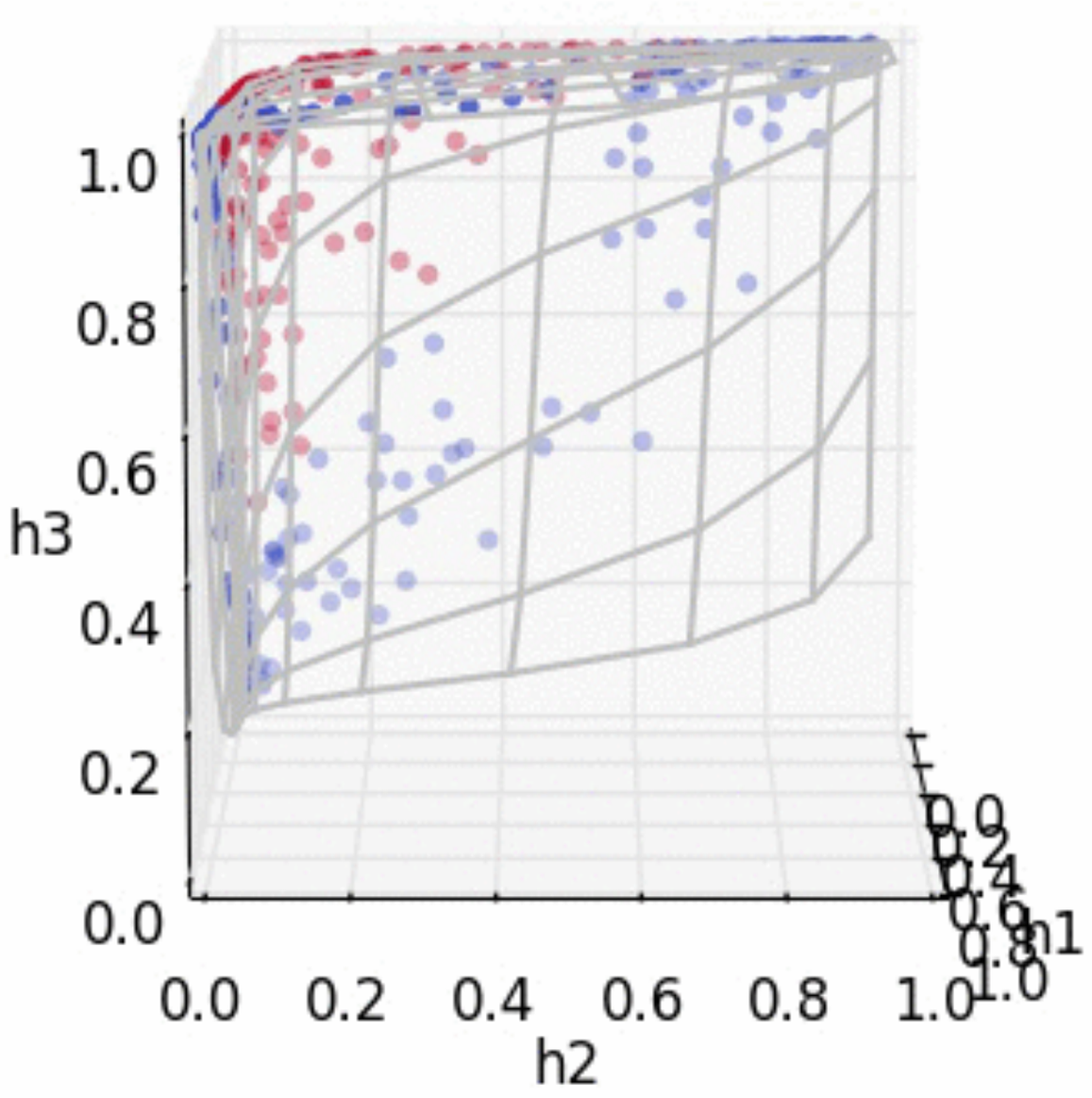

z1
z2
k1
k2
k3

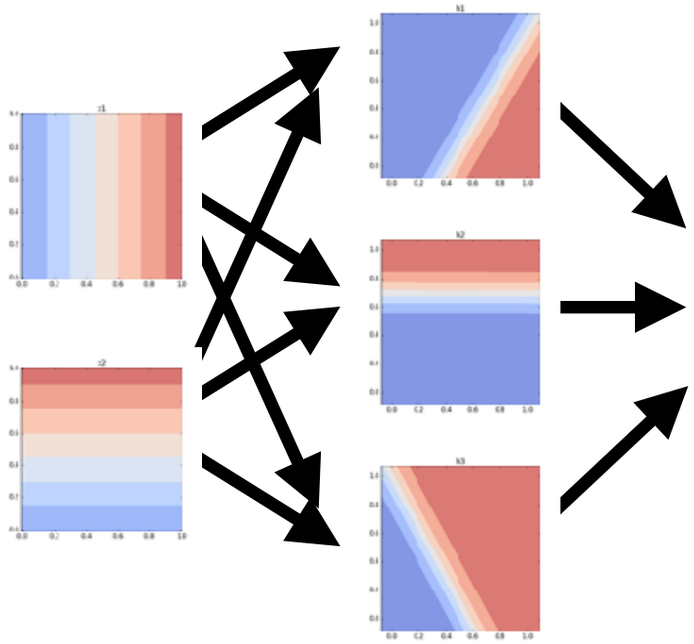

output

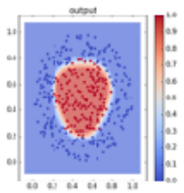

1.0
0.8
0.6
0.4
0.2
0.0
h3
0.0
0.2
0.4
0.6
0.8
1.0
h2
h1


z1
z2
k1
k2
k3

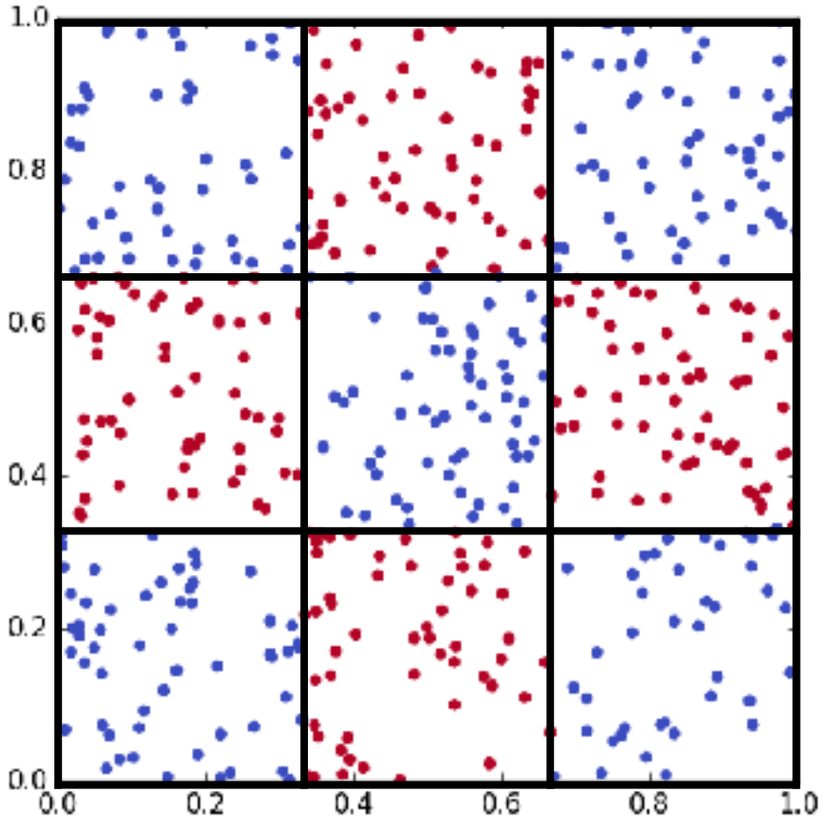

output

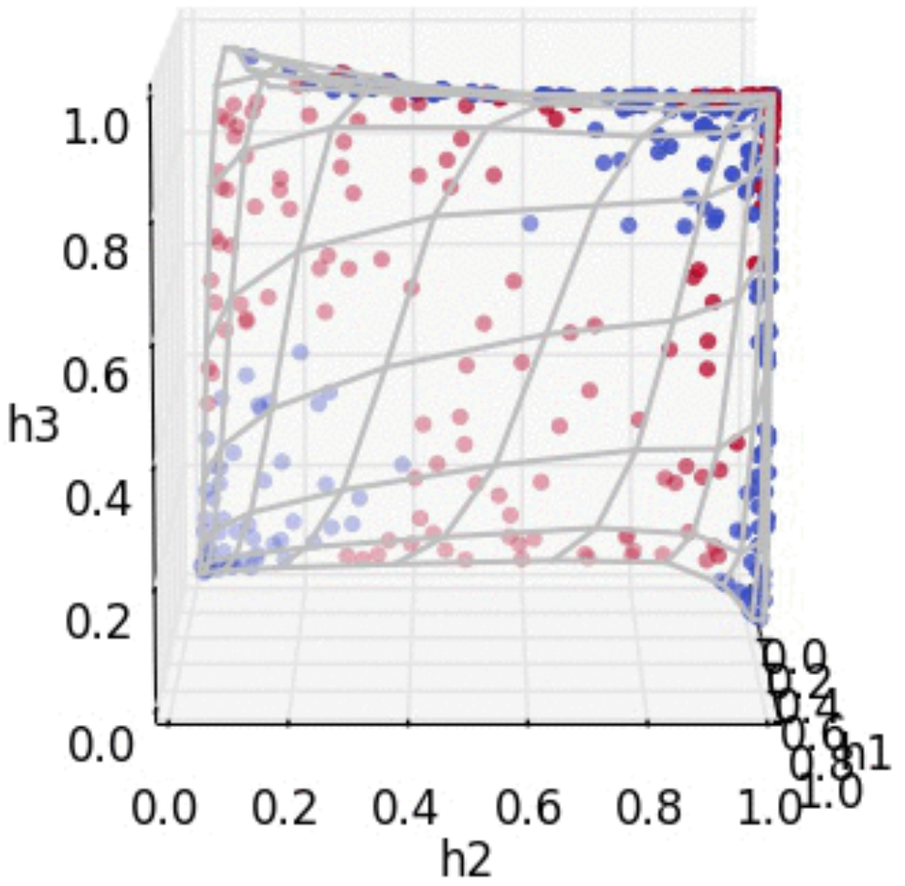

1.0
0.8
0.6
0.4
0.2
0.0
h3
0.0
0.2
0.4
0.6
0.8
1.0
h2
h1

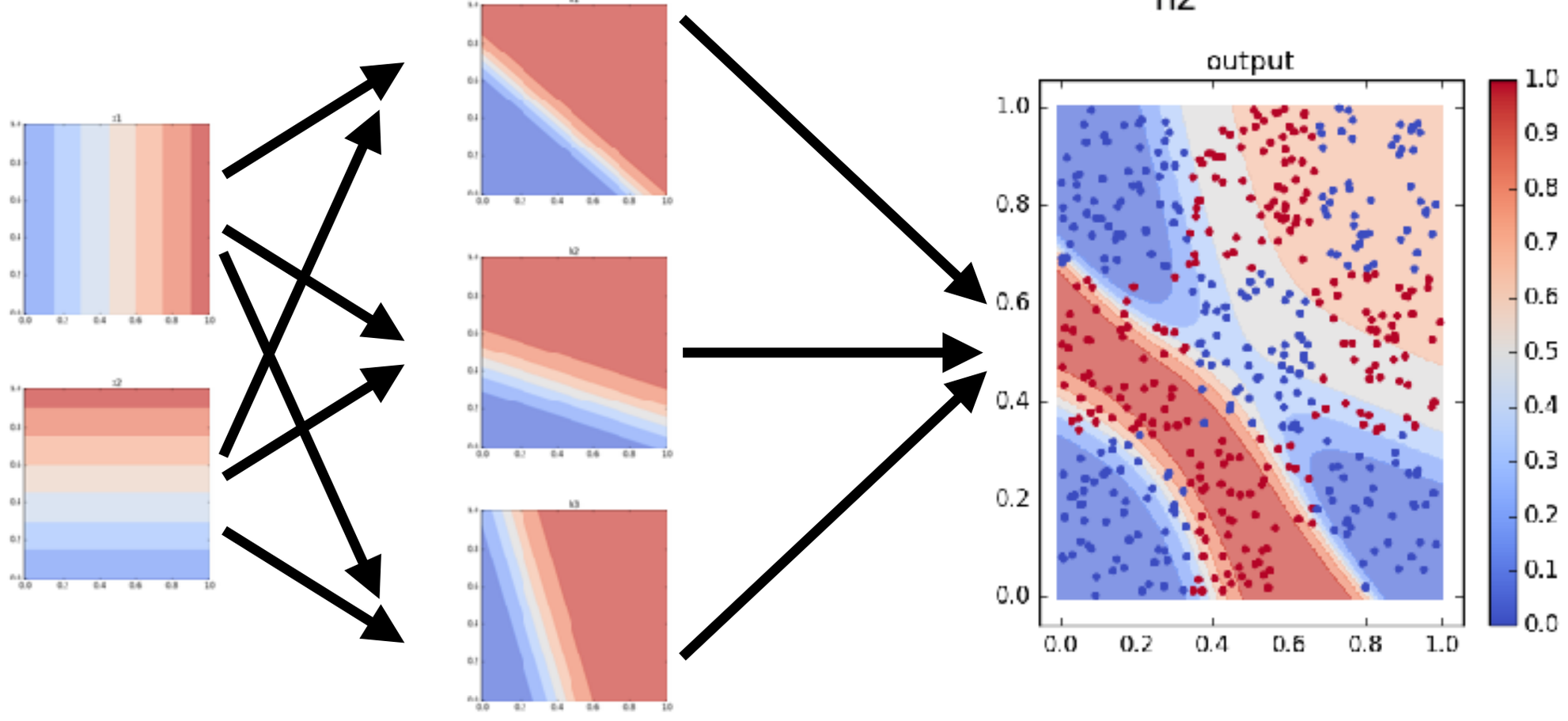

z1
z2
k1
k2
k3


output

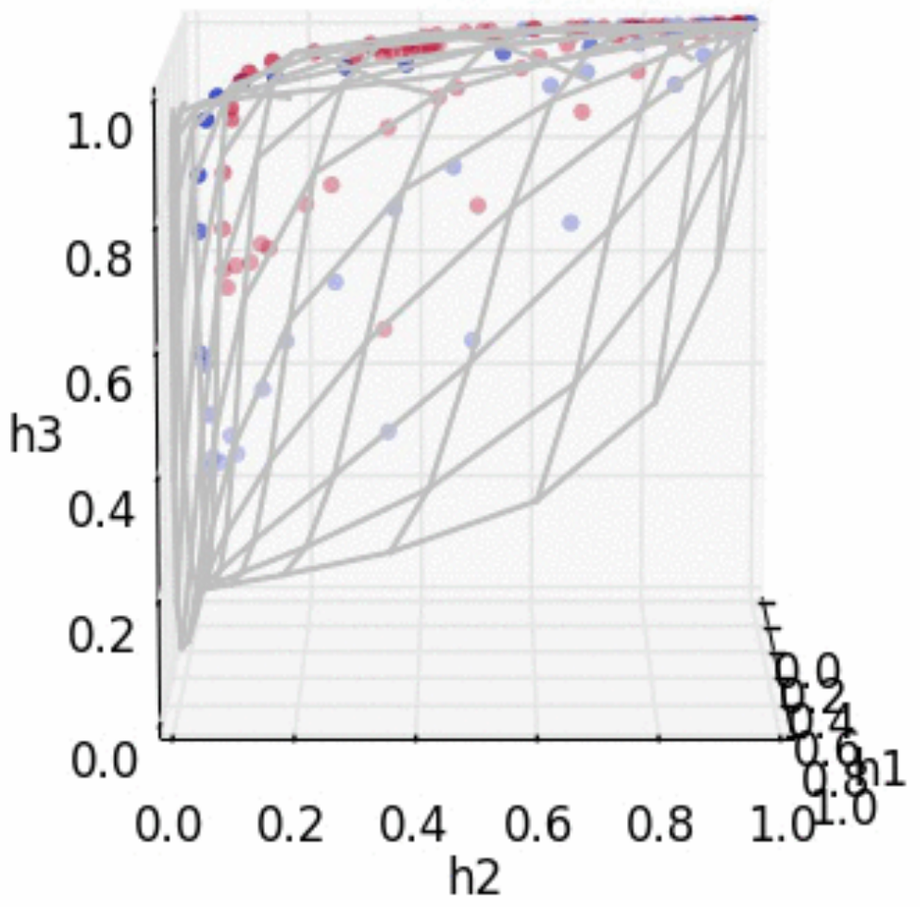
1.0
0.8
0.6
0.4
0.2
0.0
h3
0.0
0.2
0.4
0.6
0.8
1.0
h2
h1

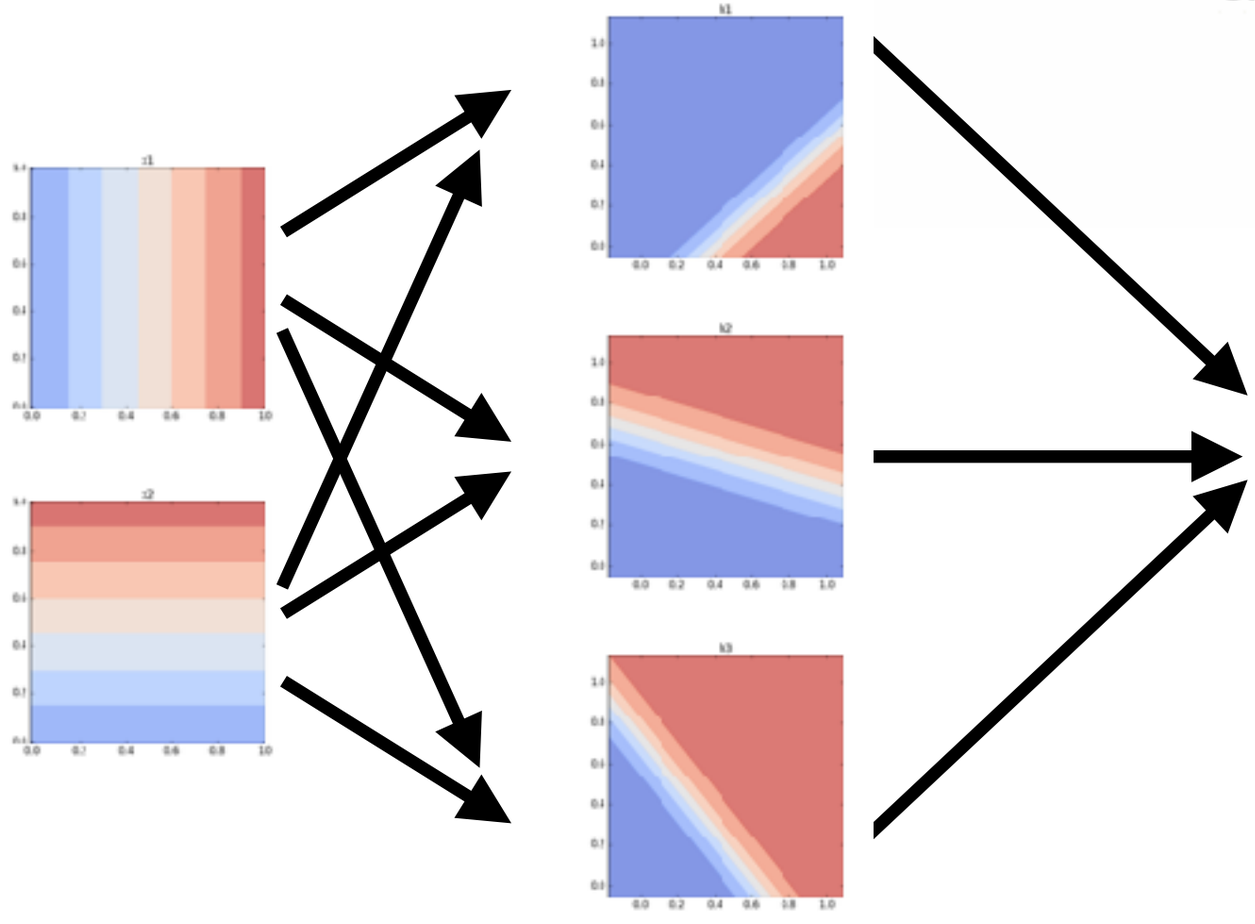

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

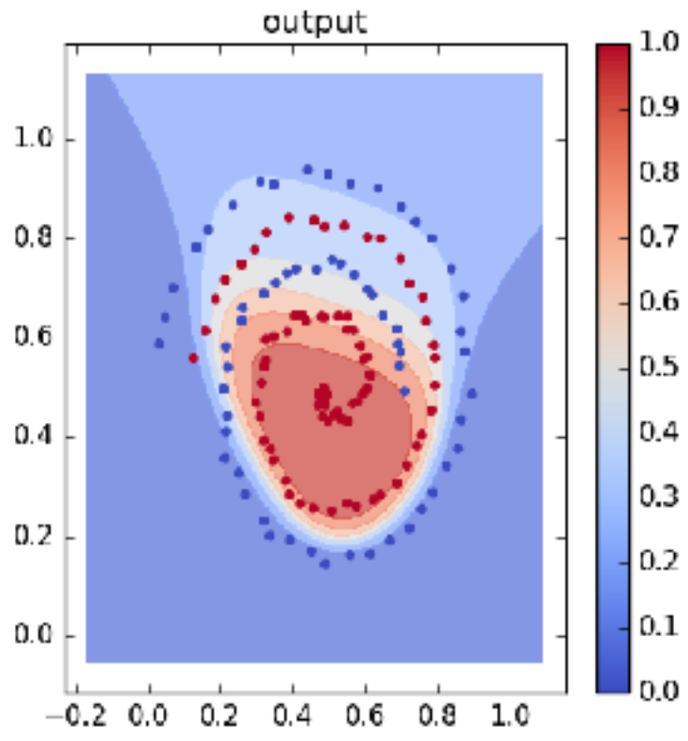

h3
h2
h1


z1
z2
h1
h2
h3
output

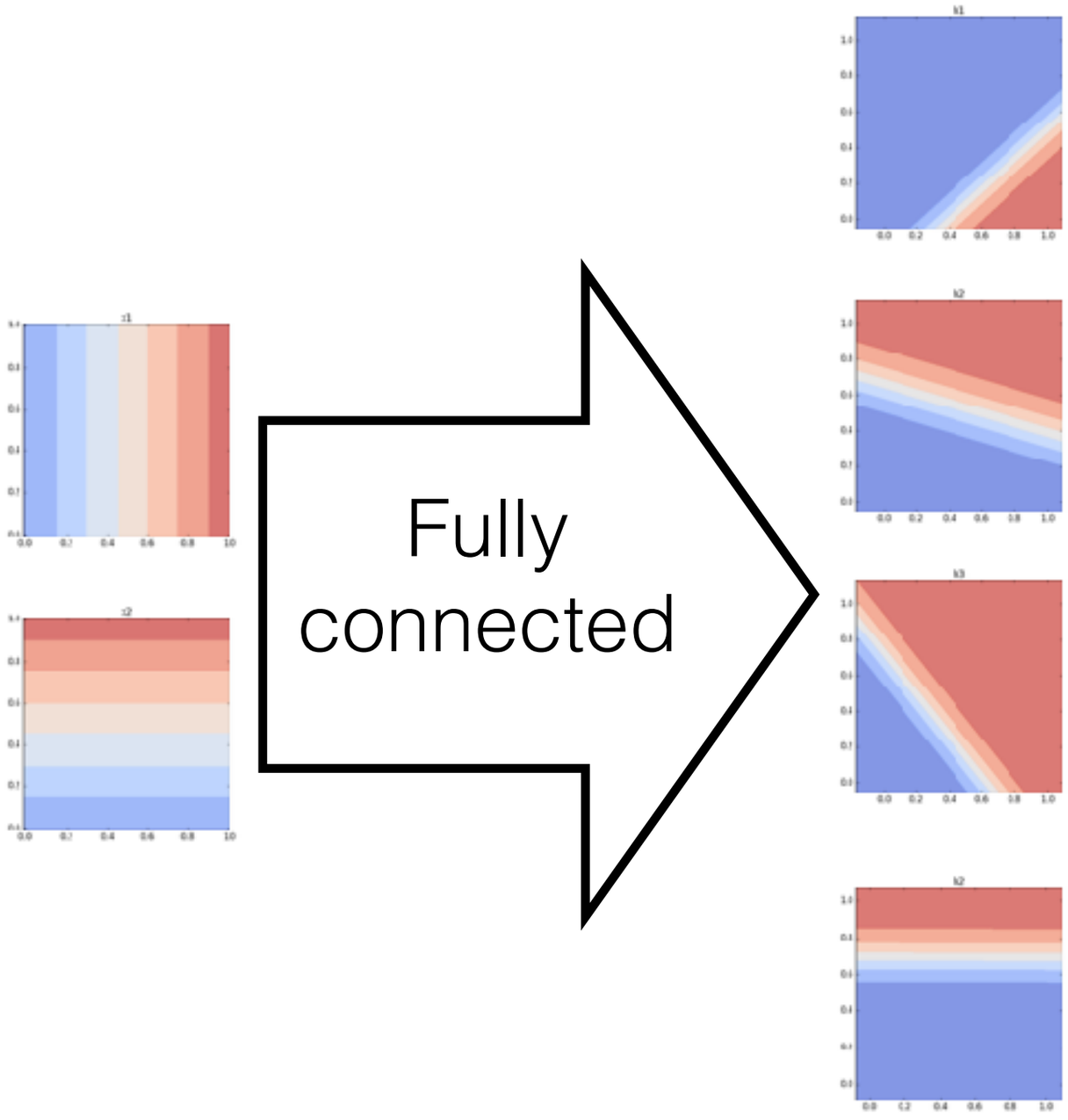
Spiral


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?



Feed backward

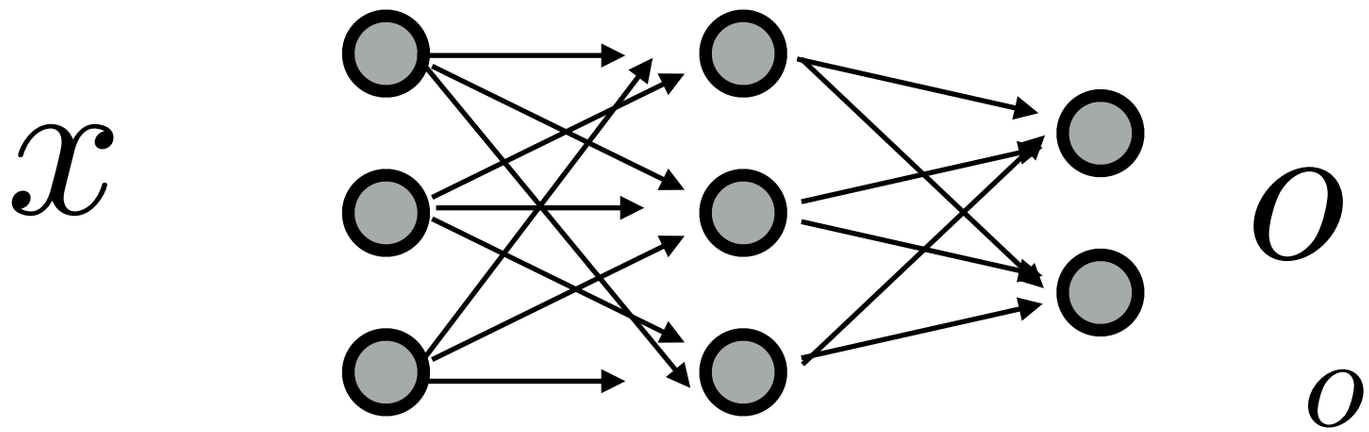
A general objective

Tweak the output of neural network to be as similar to the label as possible.

$$o \approx y$$

How to tweak? Adjust the weights

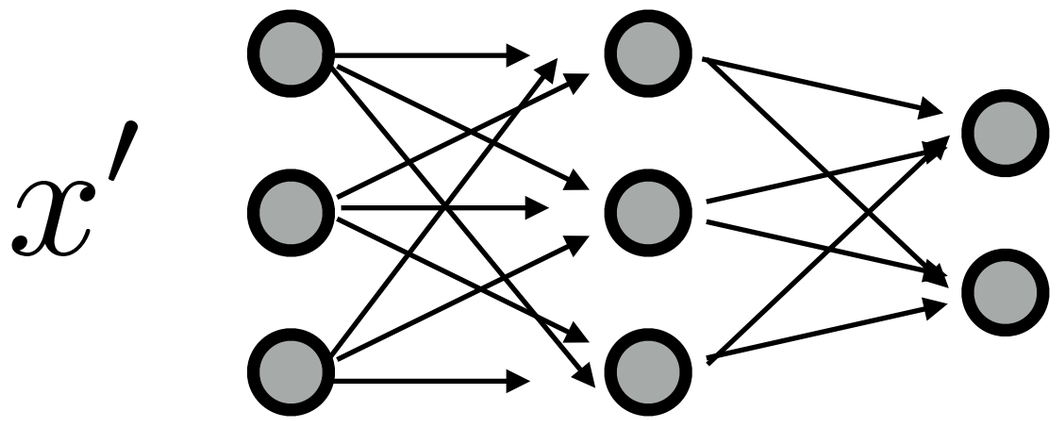
Given an example (x, y)



For this example, the neural network ‘behaves well’

After the first example, give another example

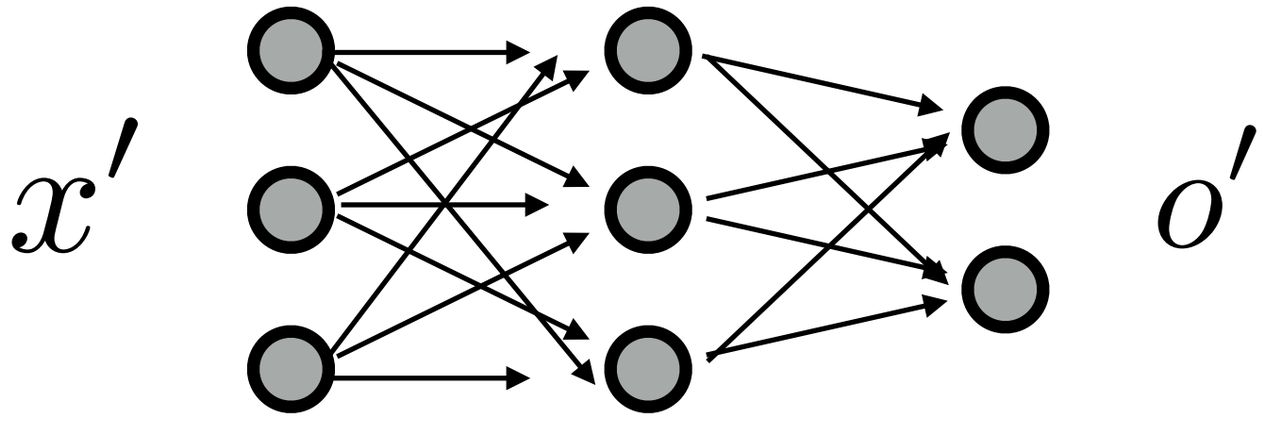
$$(x', ?)$$



Now we say for a well behaved neural network

$$o' = y'$$

How good is the trained network?

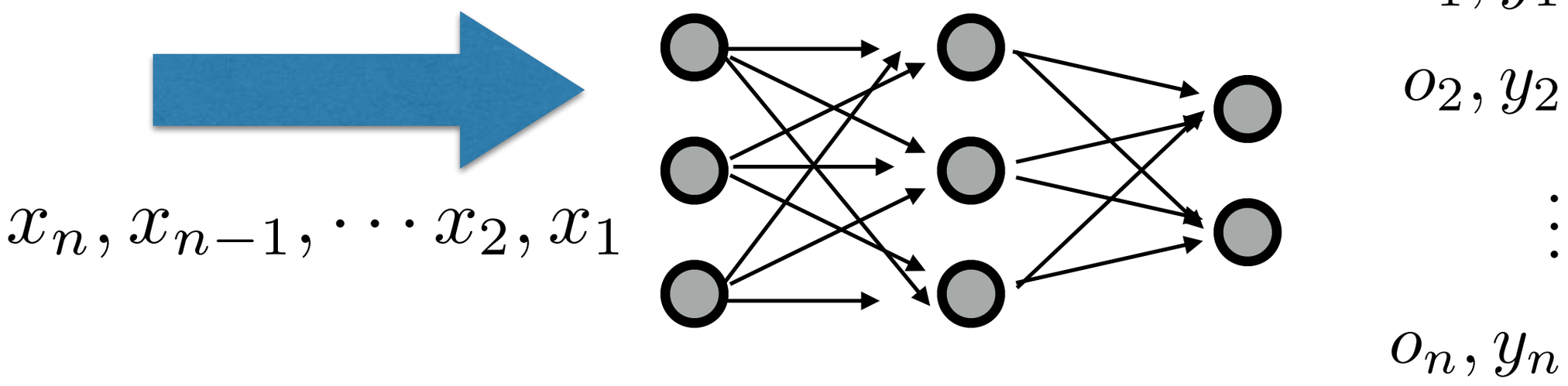


Square error

Given data points $(x_i, y_i), i = 1, \cdots n$

$$l(y_i, o_i) = \|y_i - o_i\|^2$$

$$C = \frac{1}{n} \sum_i l(y_i, o_i)$$



Adjust the weights until C = 0 (or close to zero)

Cross entropy cost function

Two class example $\quad y_i \in \{0, 1\}$

$$C = -\frac{1}{n} \sum_i y_i \log[o(x_i)] + (1 - y_i) \log[1 - o(x_i)]$$

Three class example

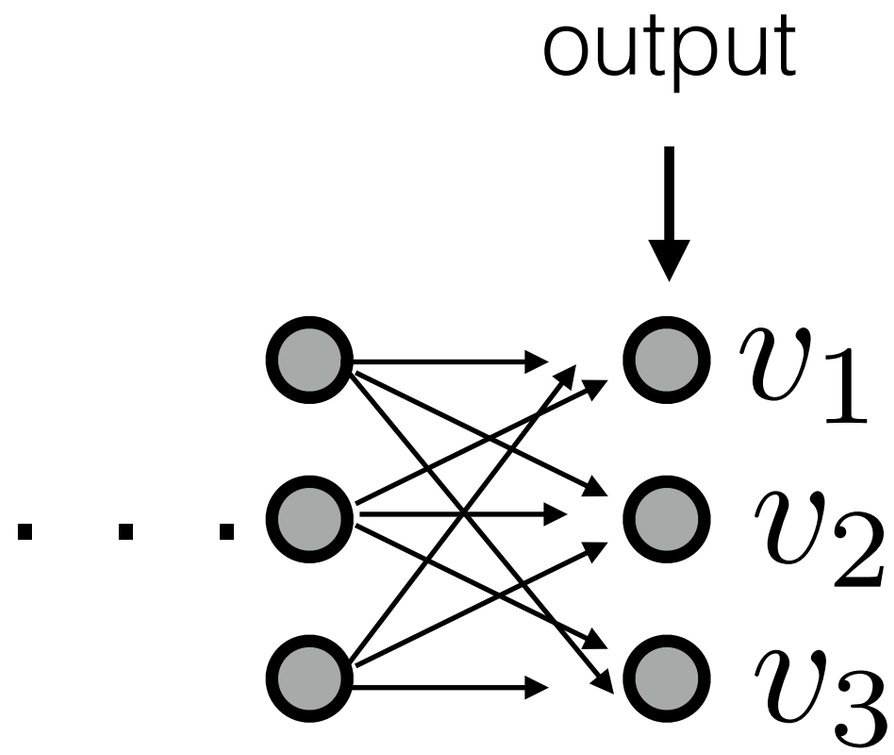
$$y_i \in \{(1,0,0), (0,1,0), (0,0,1)\}$$

$$m = \exp(v_1) + \exp(v_2) + \exp(v_3)$$

$$o_i = (\frac{\exp(v_1)}{m}, \frac{\exp(v_2)}{m}, \frac{\exp(v_3)}{m})$$

$$l(y_i, o_i) = -y_i \cdot \log(o_i)$$

$$C = \frac{1}{n} \sum_i l(y_i, o_i)$$



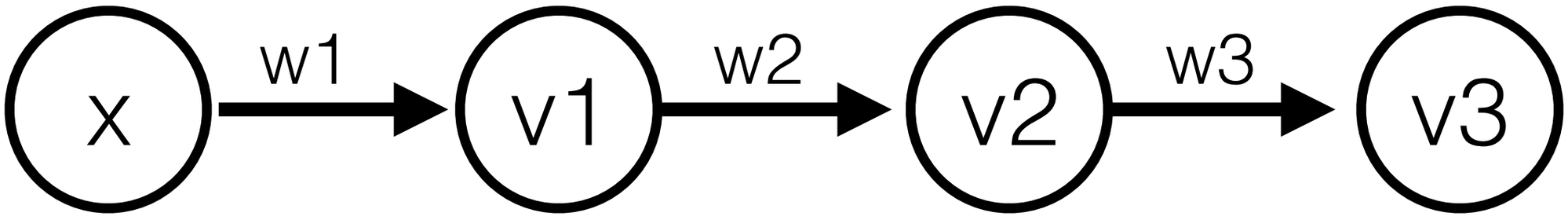
How to adjust the weights?

Conceptually, this will work:

> Try all possible weights combinations, for all weights combinations, calculate the cost. Then pick the weight combination that has the lowest cost.

Practically, there are too many weights combinations to try.

Gradient descend



$$v_1 = \sigma(z_1) = \sigma(w_1 x + b_1)$$

$$v_2 = \sigma(z_2) = \sigma(w_2 v_1 + b_2)$$

$$v_3 = \sigma(z_3) = \sigma(w_3 v_2 + b_3)$$

$$\frac{\partial C}{\partial w_j} = 0 \qquad \frac{\partial C}{\partial b_j} = 0$$

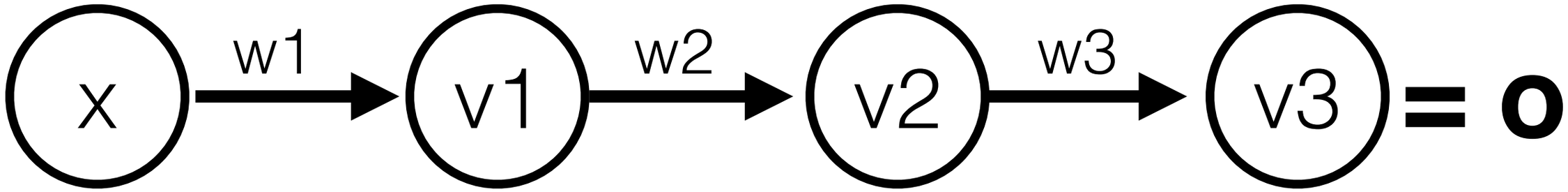
$$\frac{\partial C}{\partial w_j} = 0 \qquad\qquad \frac{\partial C}{\partial b_j} = 0$$

$$\frac{\partial C}{\partial w_j} = \frac{1}{n}\sum_i \frac{\partial (y_i - o_i)^2}{\partial w_j}$$

$$= -\frac{2}{n}\sum_i (y_i - o_i)\frac{\partial o_i}{\partial w_j}$$

We just need $\dfrac{\partial o_i}{\partial w_j}$

$$w_j(t+1) = w_j(t) - \eta \frac{\partial C}{\partial w_j}$$



$$v_1 = \sigma(z_1) = \sigma(w_1 x + b_1)$$

$$v_2 = \sigma(z_2) = \sigma(w_2 v_1 + b_2)$$

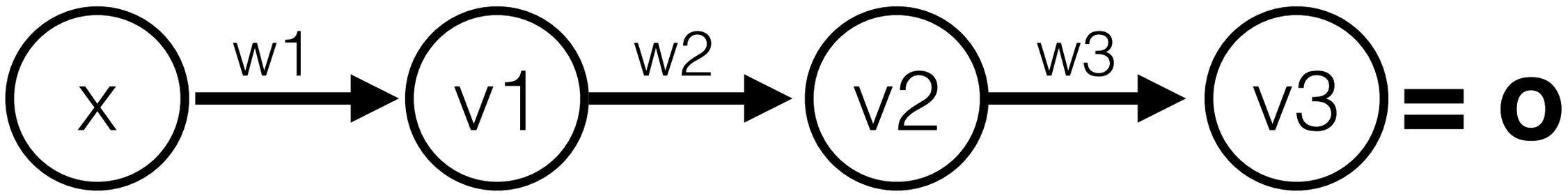
$$v_3 = \sigma(z_3) = \sigma(w_3 v_2 + b_3)$$

$$\frac{\partial v_3}{\partial w_3} = \frac{\partial v_3}{\partial z_3}\frac{\partial z_3}{\partial w_3} = \sigma'(z_3)v_2$$

$$\frac{\partial v_3}{\partial w_2} = \frac{\partial v_3}{\partial z_3}\frac{\partial z_3}{\partial w_2} = \sigma'(z_3)w_3\frac{\partial v_2}{\partial w_2} = \sigma'(z_3)w_3\sigma'(z_2)v_1$$

$$\frac{\partial v_3}{\partial w_1} = \frac{\partial v_3}{\partial z_3}\frac{\partial z_3}{\partial w_1} = \sigma'(z_3)w_3\frac{\partial v_2}{\partial w_2} = \sigma'(z_3)w_3\sigma'(z_2)w_2\frac{\partial v_1}{\partial w_1}$$

$$= \sigma'(z_3)w_3\sigma'(z_2)w_2\sigma'(z_1)x$$



$$\frac{\partial v_3}{\partial w_3} = \frac{\partial v_3}{\partial z_3}\frac{\partial z_3}{\partial w_3} = \sigma'(z_3)v_2$$

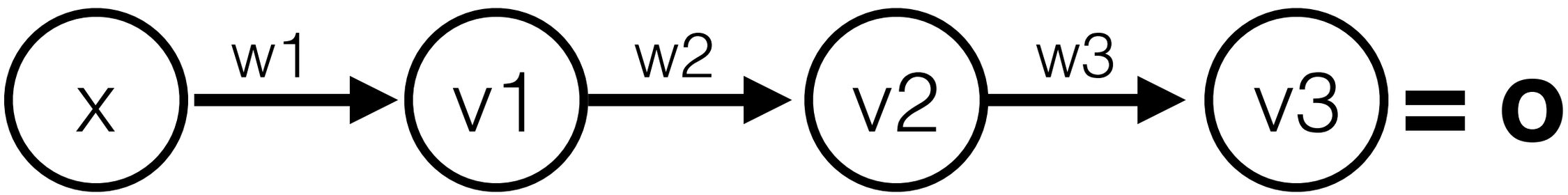
$$\frac{\partial v_3}{\partial w_2} = \frac{\partial v_3}{\partial z_3}\frac{\partial z_3}{\partial w_2} = \sigma'(z_3)w_3\frac{\partial v_2}{\partial w_2} = \sigma'(z_3)w_3\sigma'(z_2)v_1$$

$$\frac{\partial v_3}{\partial w_1} = \frac{\partial v_3}{\partial z_3}\frac{\partial z_3}{\partial w_1} = \sigma'(z_3)w_3\frac{\partial v_2}{\partial w_2} = \sigma'(z_3)w_3\sigma'(z_2)w_2\frac{\partial v_1}{\partial w_1}$$

$$= \sigma'(z_3)w_3\sigma'(z_2)w_2\sigma'(z_1)x$$

problem!! : long mathematical expression leads to large computational time for deep network

Compute and store strategy



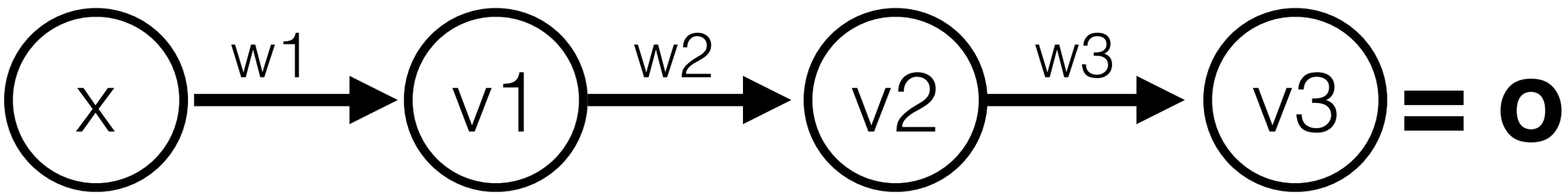
$$\frac{\partial v_3}{\partial z_1} \qquad \frac{\partial v_3}{\partial z_2} \qquad \frac{\partial v_3}{\partial z_3}$$

$$\frac{\partial v_3}{\partial z_3} = \sigma'(z_3)$$

$$\frac{\partial v_3}{\partial z_2} = \frac{\partial v_3}{\partial z_3}\frac{\partial z_3}{\partial z_2} = \frac{\partial v_3}{\partial z_3} w_3 \sigma'(z_2)$$

$$\frac{\partial v_3}{\partial z_1} = \frac{\partial v_3}{\partial z_2}\frac{\partial z_2}{\partial z_1} = \frac{\partial v_3}{\partial z_2} w_2 \sigma'(z_1)$$

Compute and store strategy



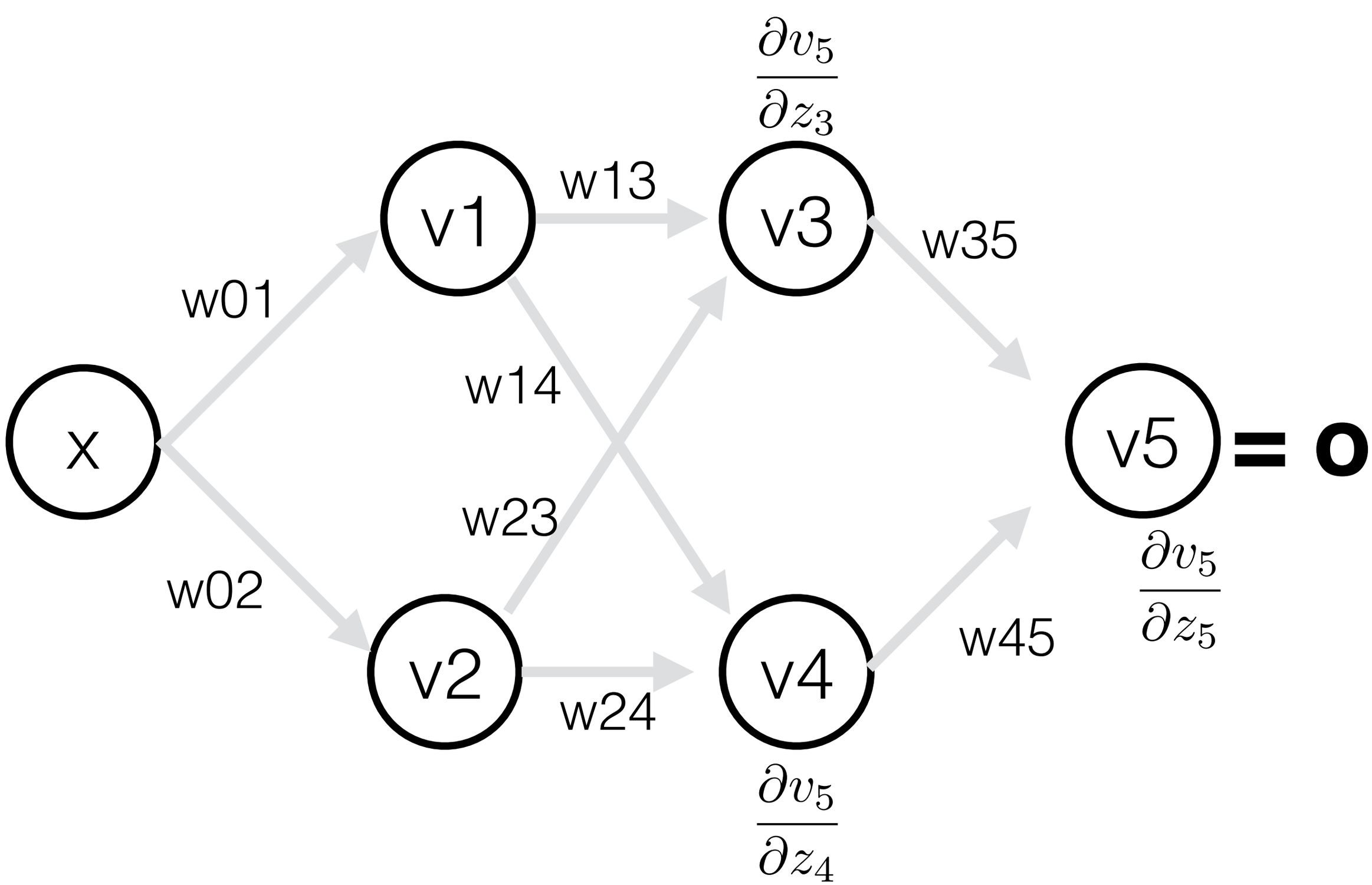
$$\frac{\partial v_3}{\partial z_1} \qquad \frac{\partial v_3}{\partial z_2} \qquad \frac{\partial v_3}{\partial z_3}$$

$$\frac{\partial v_3}{\partial w_3} = \frac{\partial v_3}{\partial z_3}\frac{\partial z_3}{\partial w_3} = \frac{\partial v_3}{\partial z_3} v_2$$

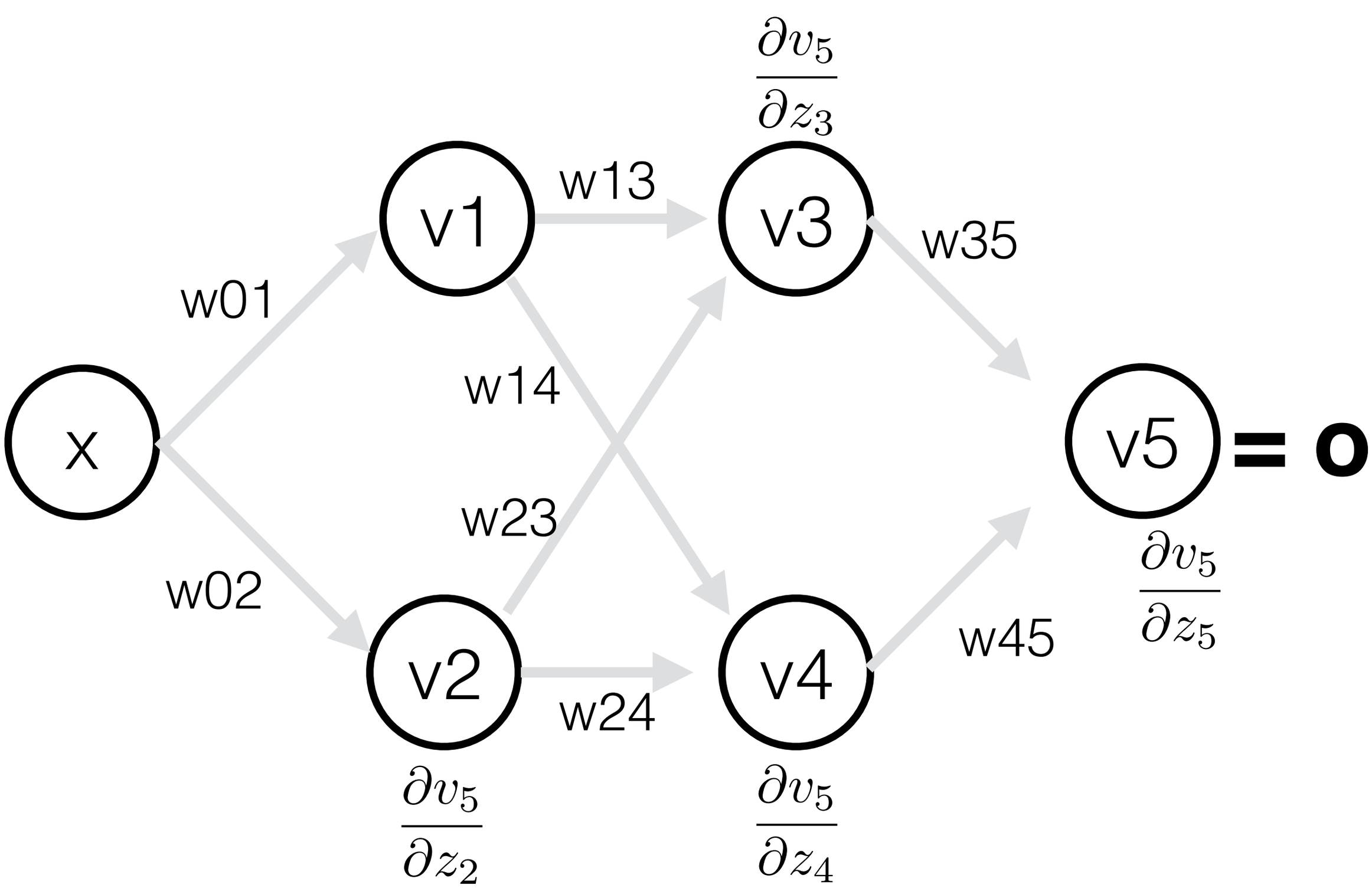
$$\frac{\partial v_3}{\partial w_2} = \frac{\partial v_3}{\partial z_2}\frac{\partial z_2}{\partial w_2} = \frac{\partial v_3}{\partial z_2} v_1$$

$$\frac{\partial v_3}{\partial w_1} = \frac{\partial v_3}{\partial z_1}\frac{\partial z_1}{\partial w_1} = \frac{\partial v_1}{\partial z_1} x$$

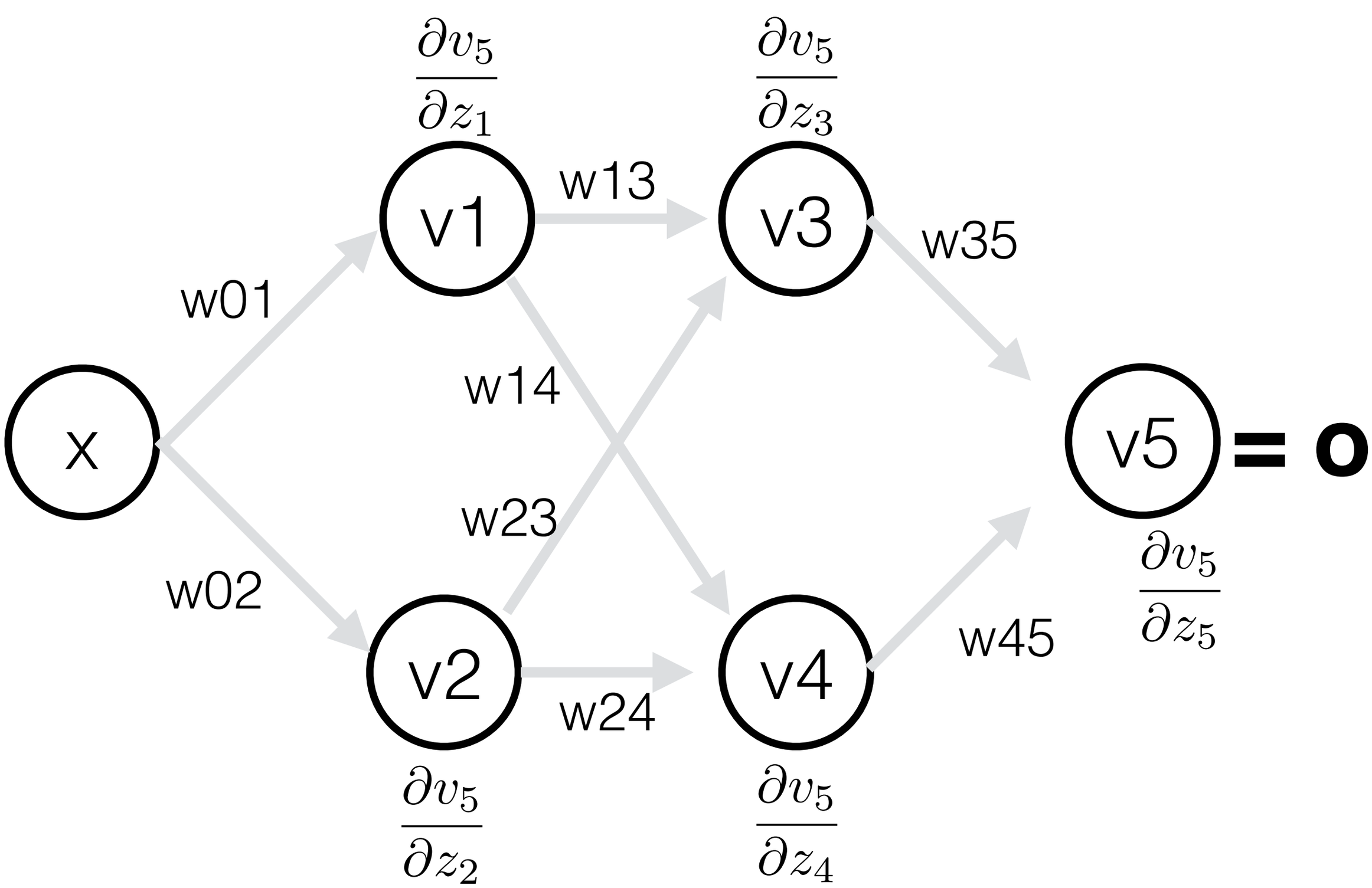
$\frac{\partial v_3}{\partial b_j}$? homework question



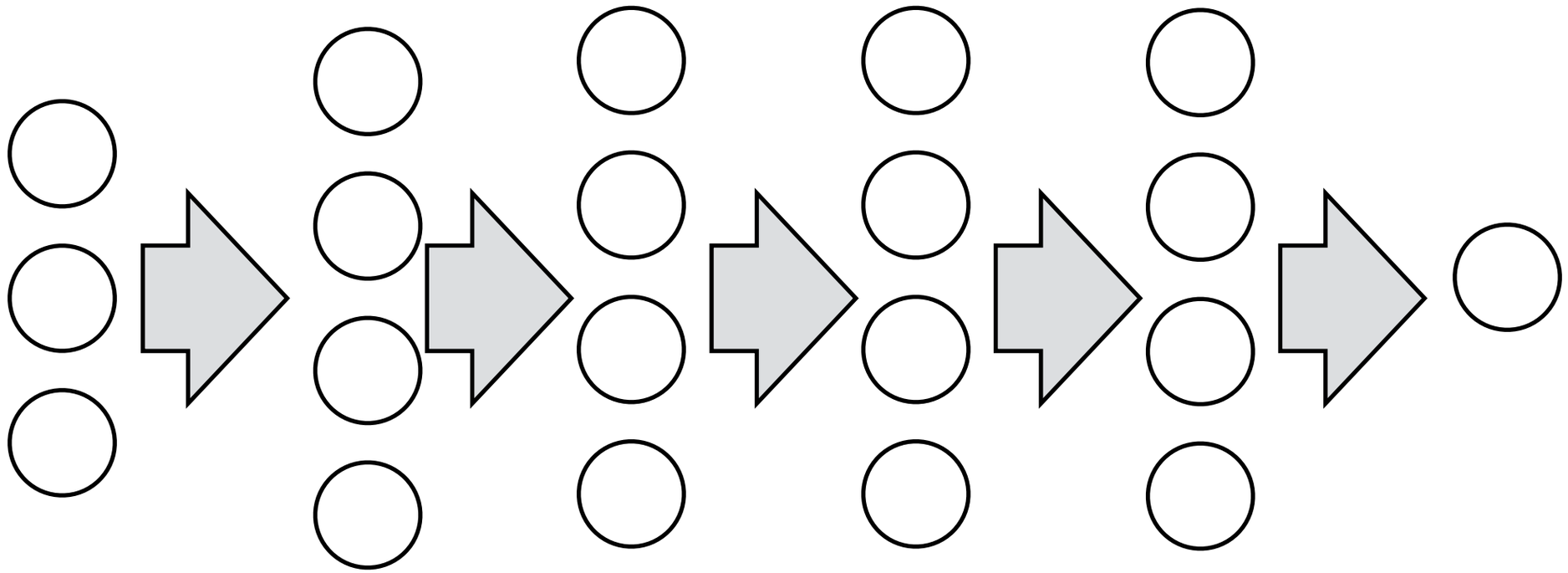
$$\frac{\partial v_5}{\partial z_4} = \frac{\partial v_5}{\partial z_5}\sigma'(z_4)w_{45}$$

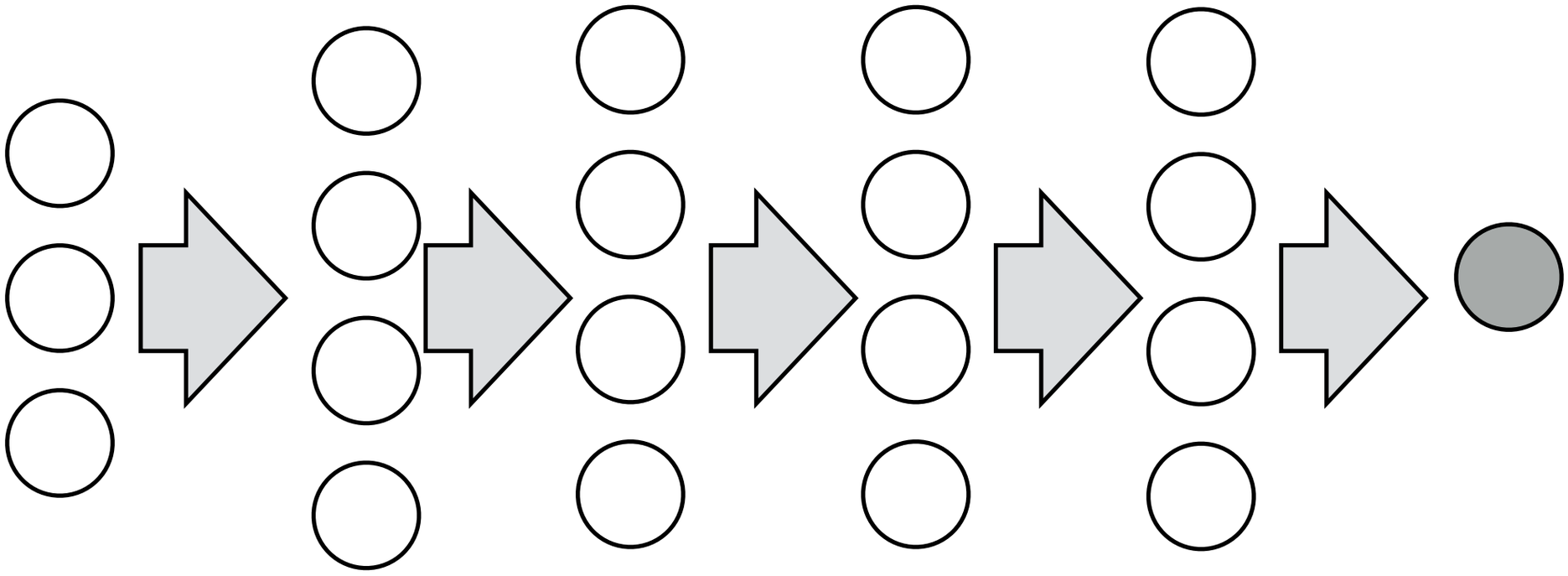


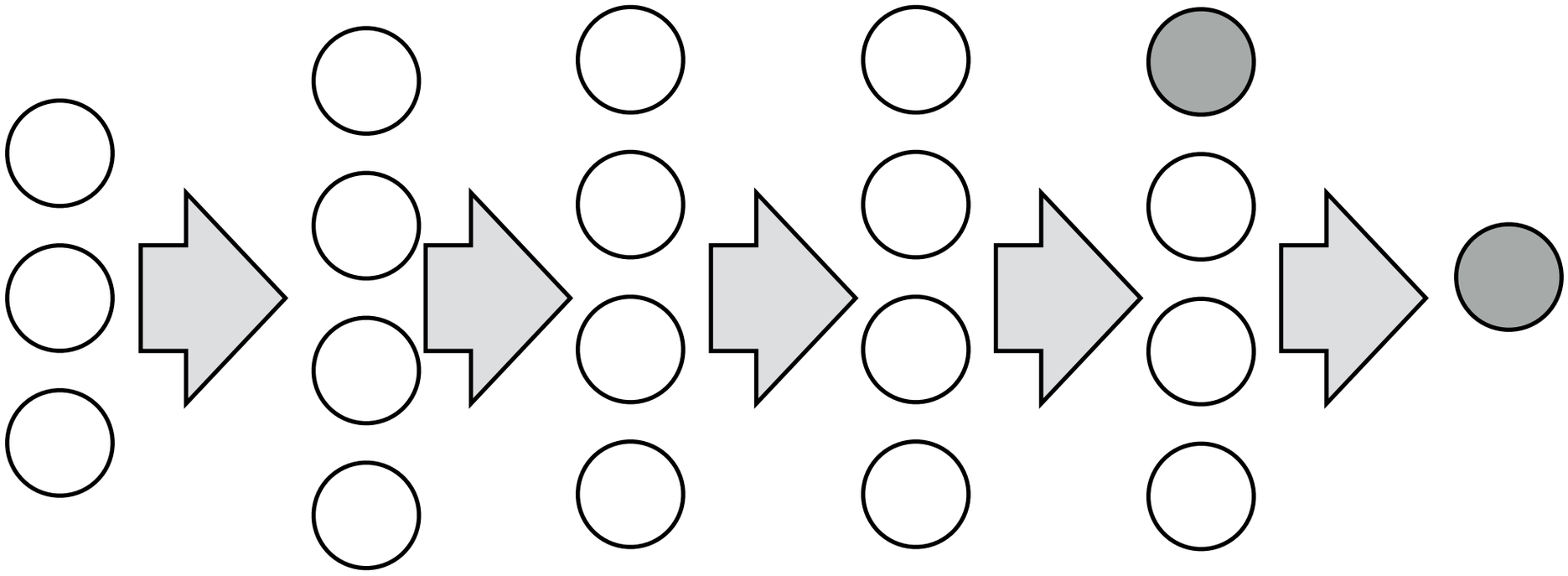
$$\frac{\partial v_5}{\partial z_2} = \frac{\partial v_5}{\partial z_4}\sigma'(z_2)w_{24} + \frac{\partial v_5}{\partial z_3}\sigma'(z_2)w_{23}$$

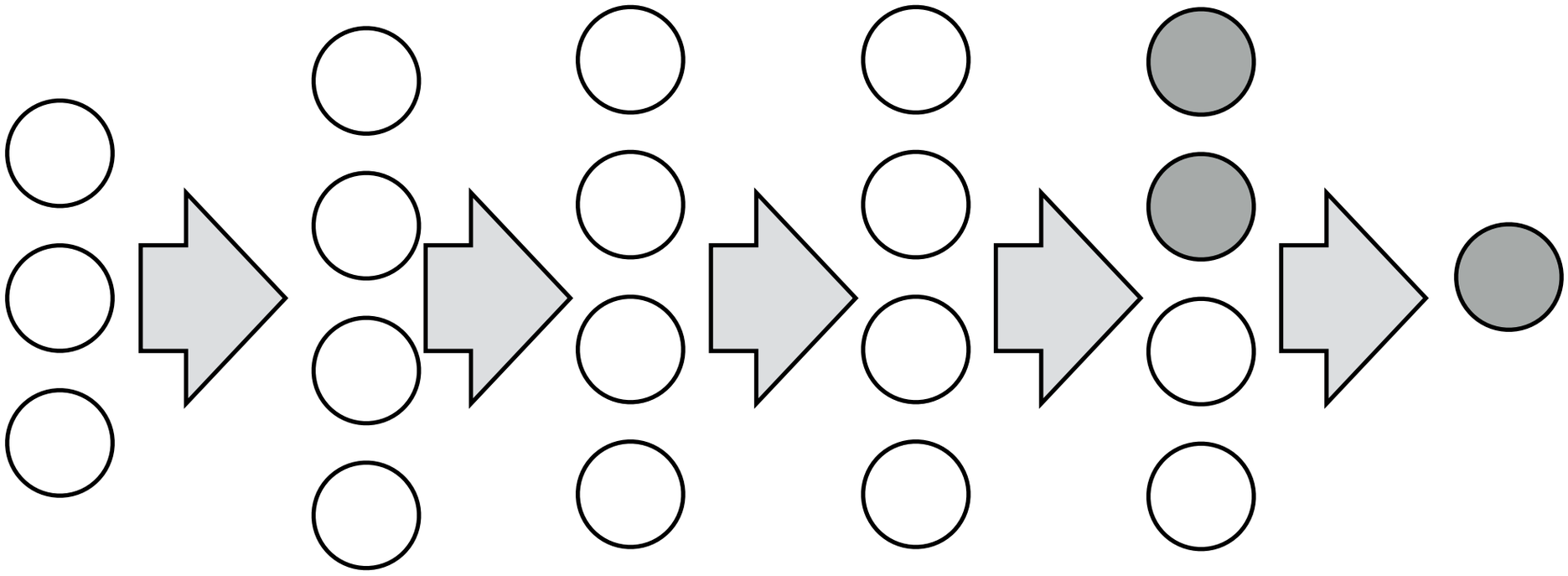


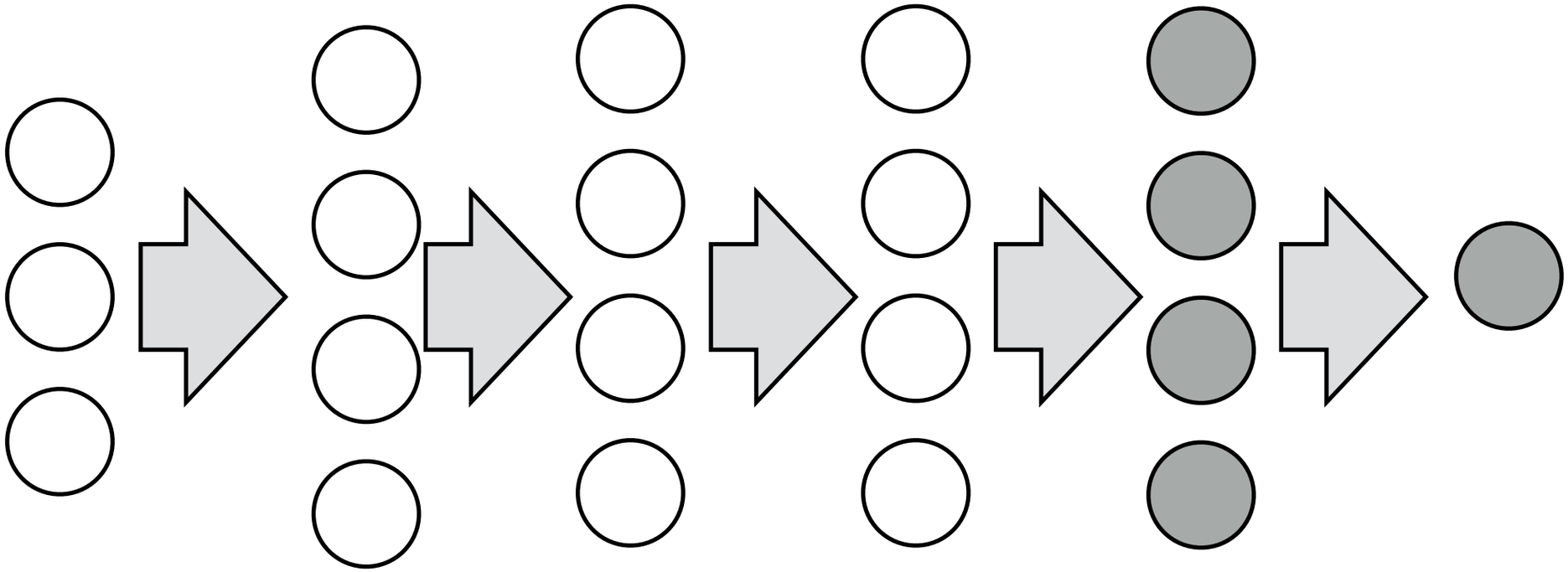
This is call the back propagation algorithm

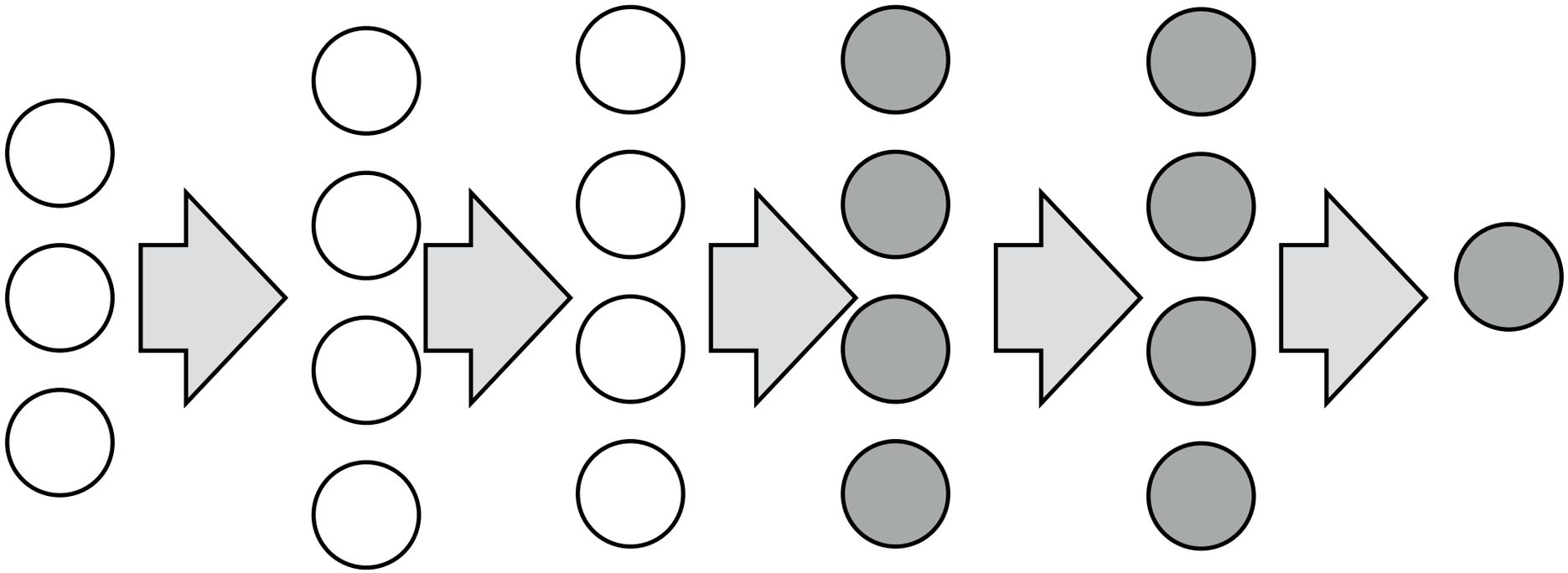


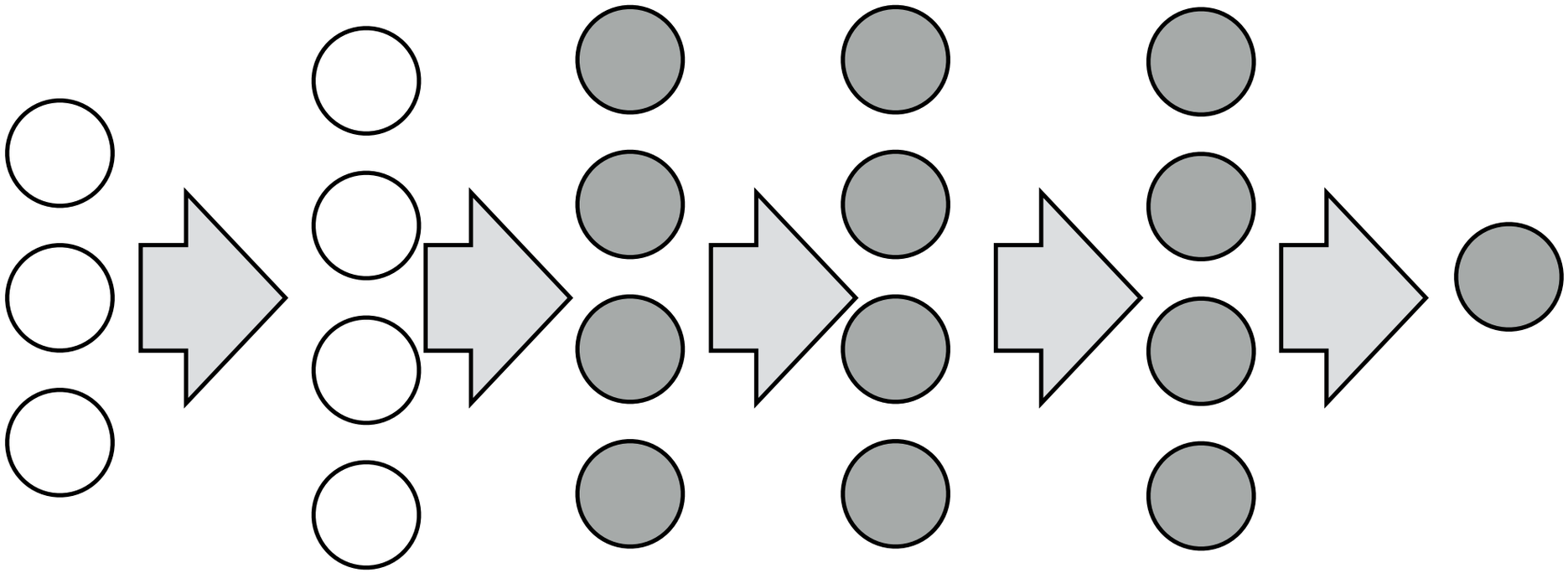


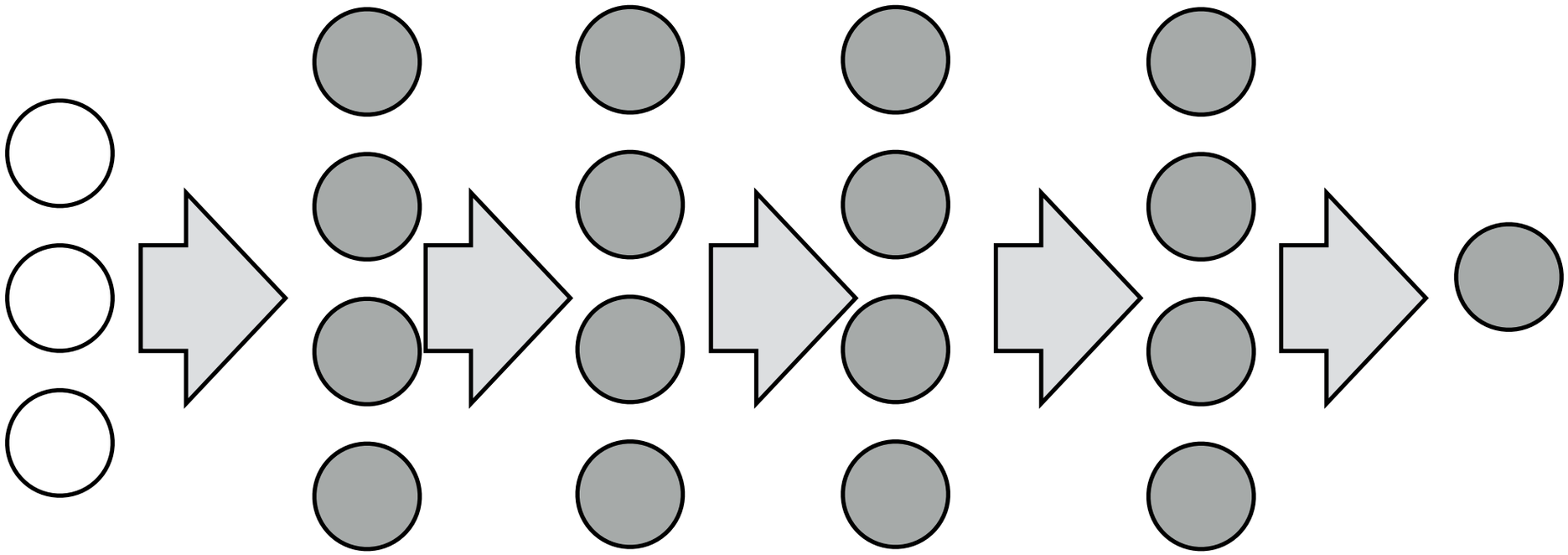




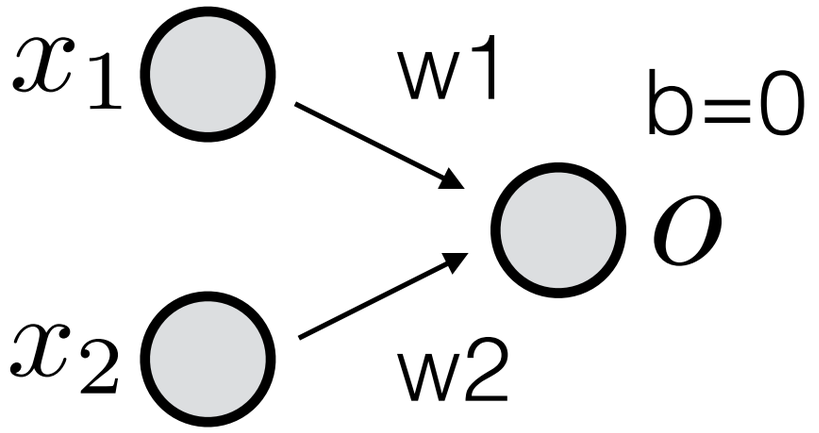






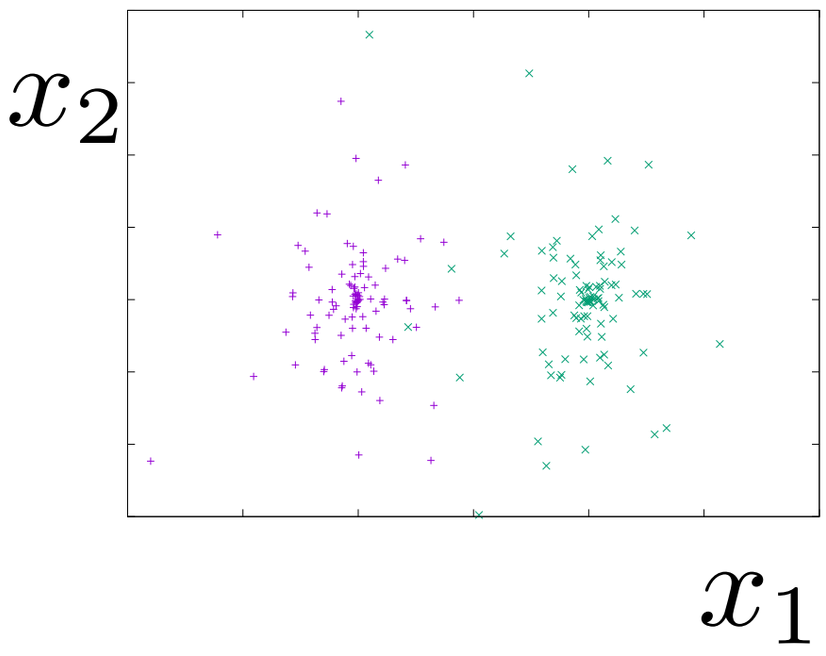


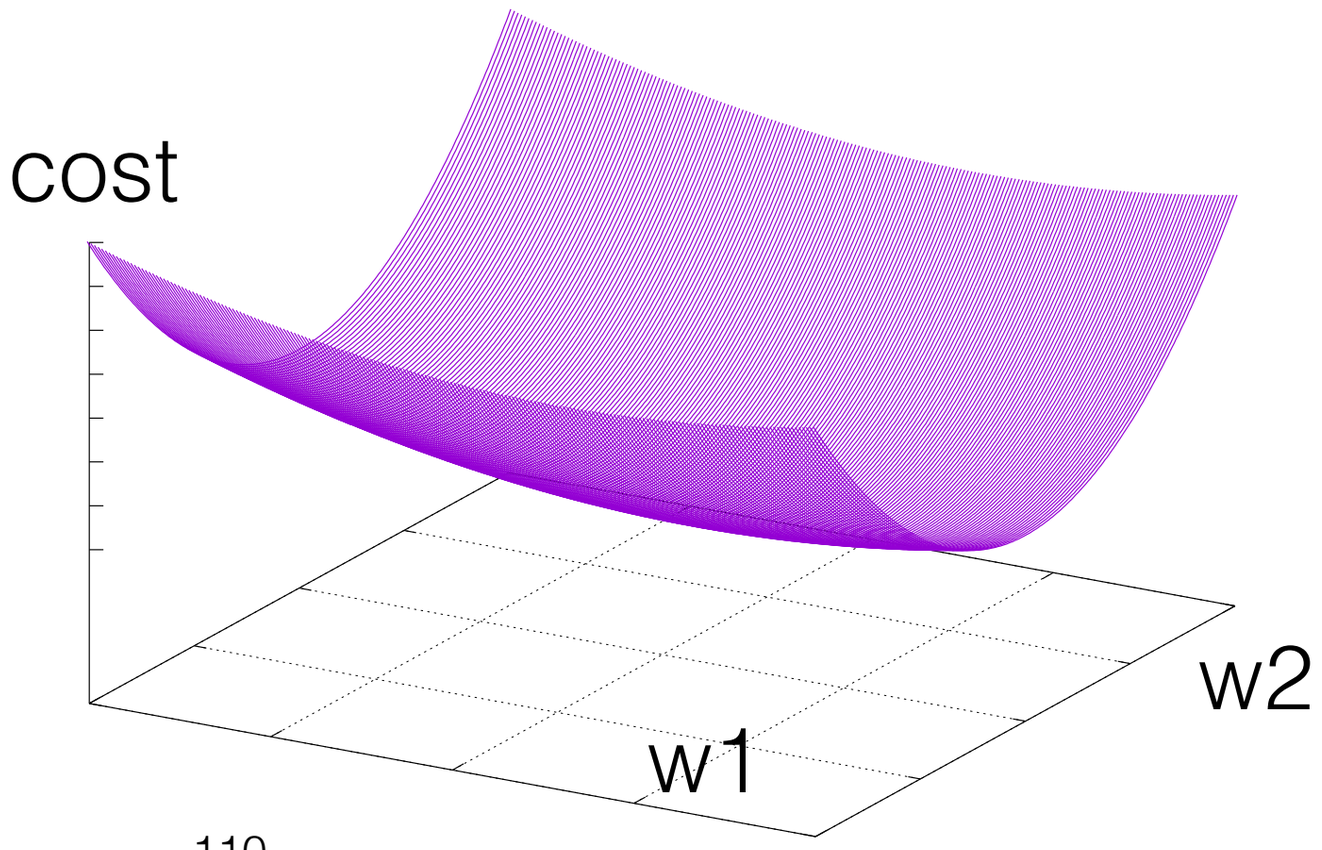
Cost function is a surface

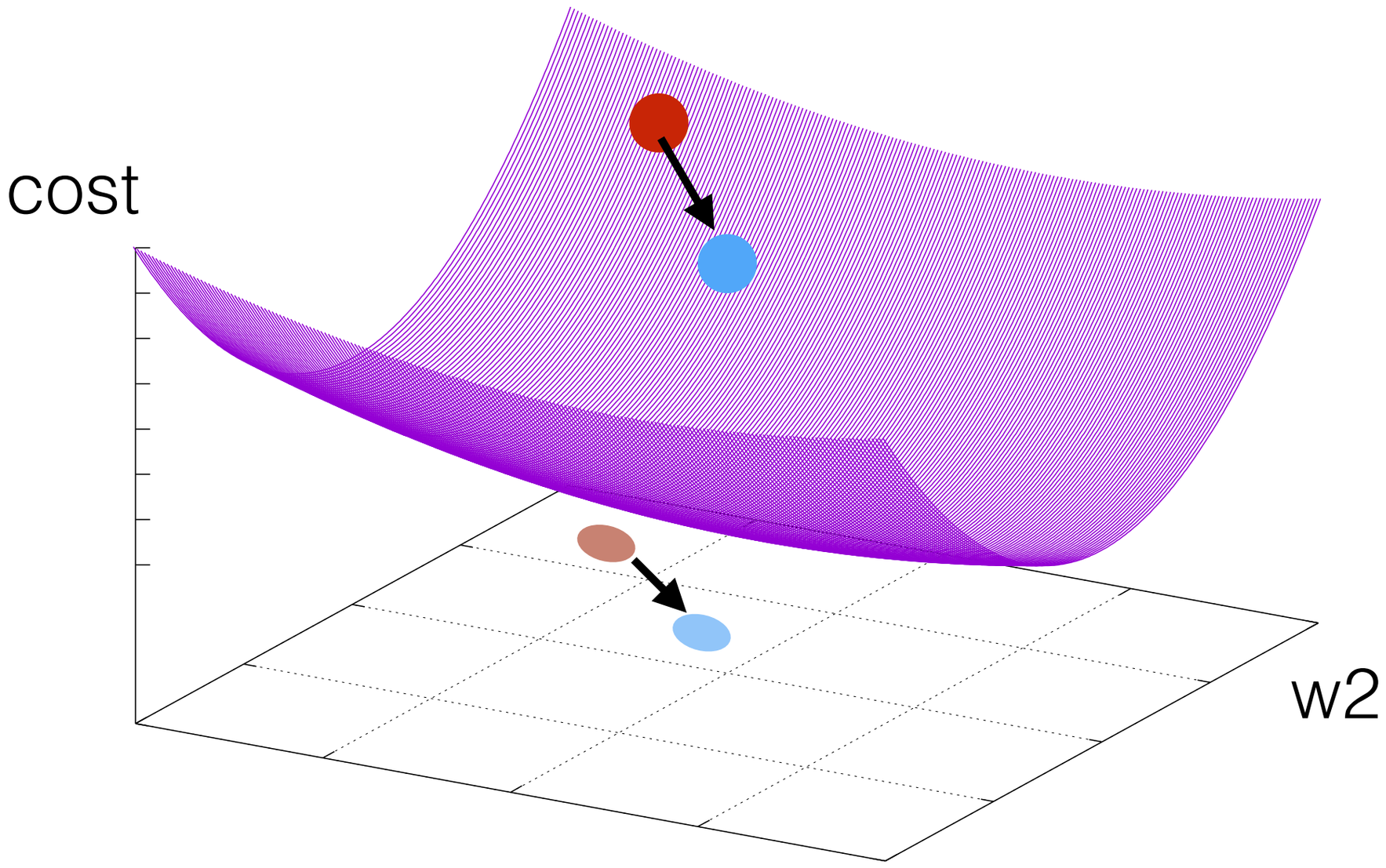


$$l(y_i, o_i) = \|y_i - o_i\|^2$$

$$C(w_1, w_2) = \frac{1}{n} \sum_i l(y_i, o_i) + 0.2(w_1^2 + w_2^2)$$







$$w_j(t+1) = w_j(t) - \eta \frac{\partial C}{\partial w_j}$$

How to use data set?

1. Train a network
2. Keep it for future use
3. When previously unseen data comes in without labels, use the network to predict

There is a problem here!

How can we know when we “keep” the network at step (2), it is good enough to fulfil the task of step (3)?

We got to find some ways to test it before we use the network

Training and Validation set

Given an annotated data set, $(x_i, y_i), i = 1, \cdots n$
randomly sample x% of it and keep aside.

Train the network on the remaining (100-x)%

Validate the network with the x% of the data that you have not yet show to the network

Network is ready to use if validation results is good.
Else, start troubleshoot.