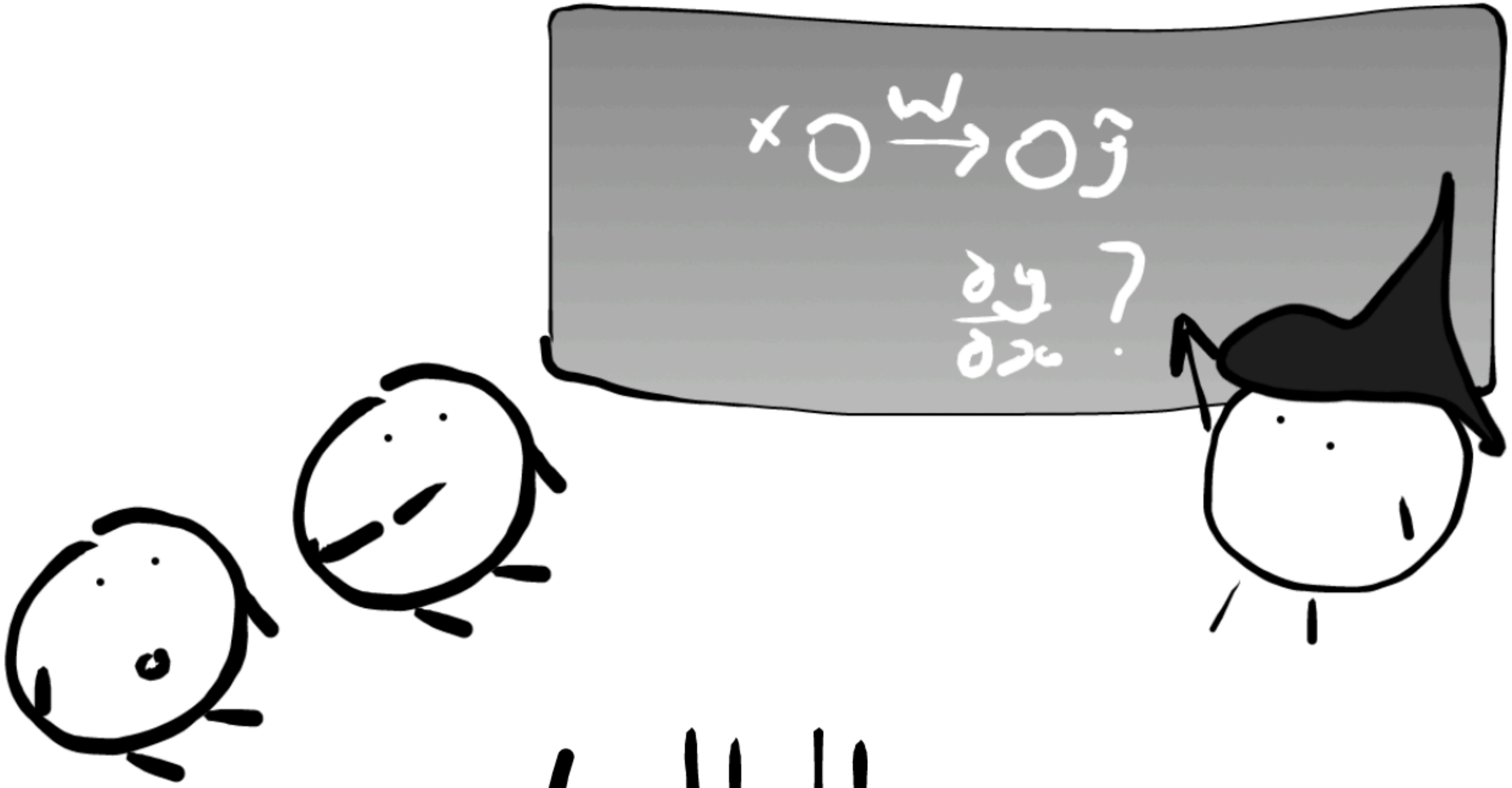
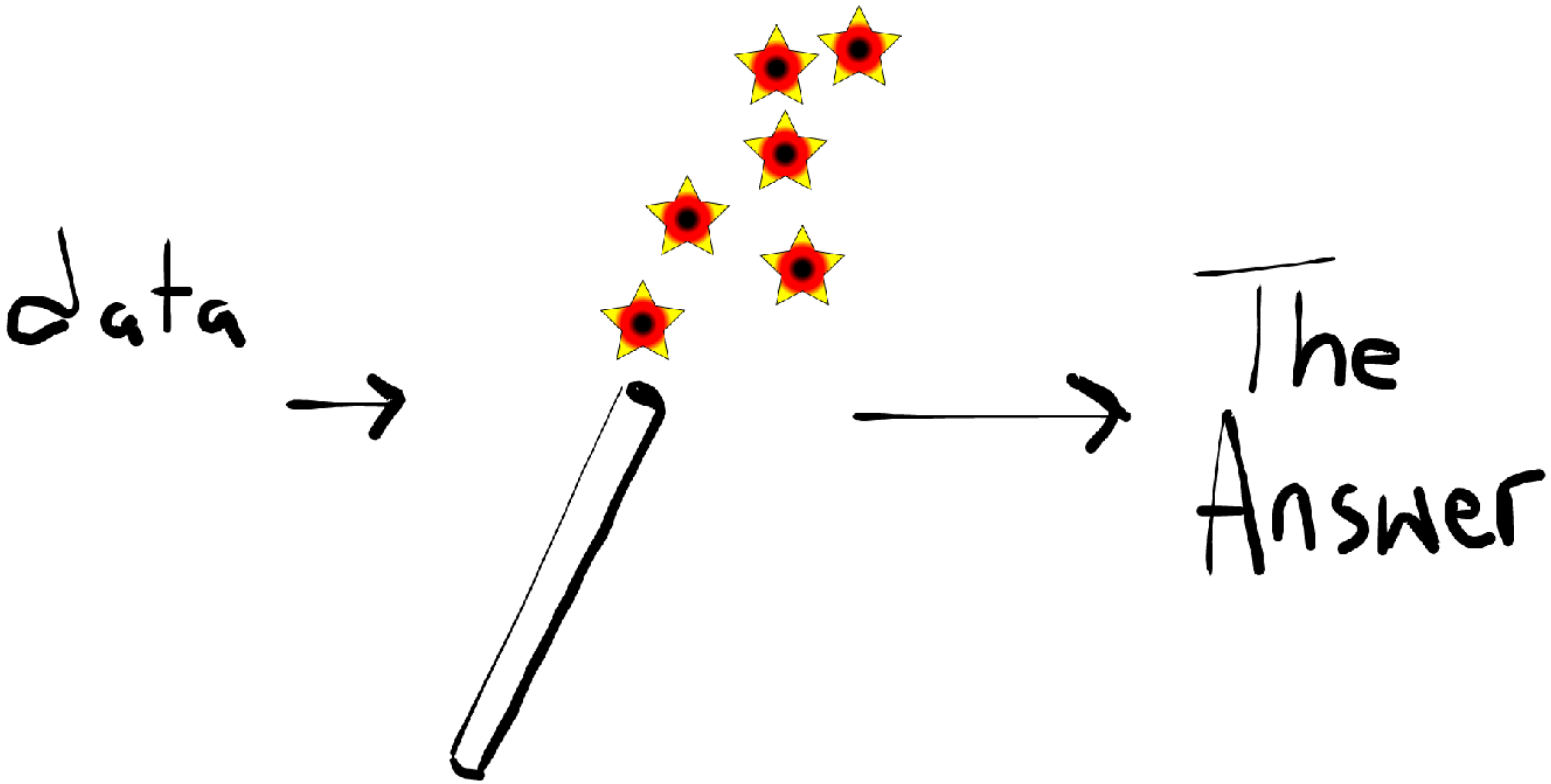
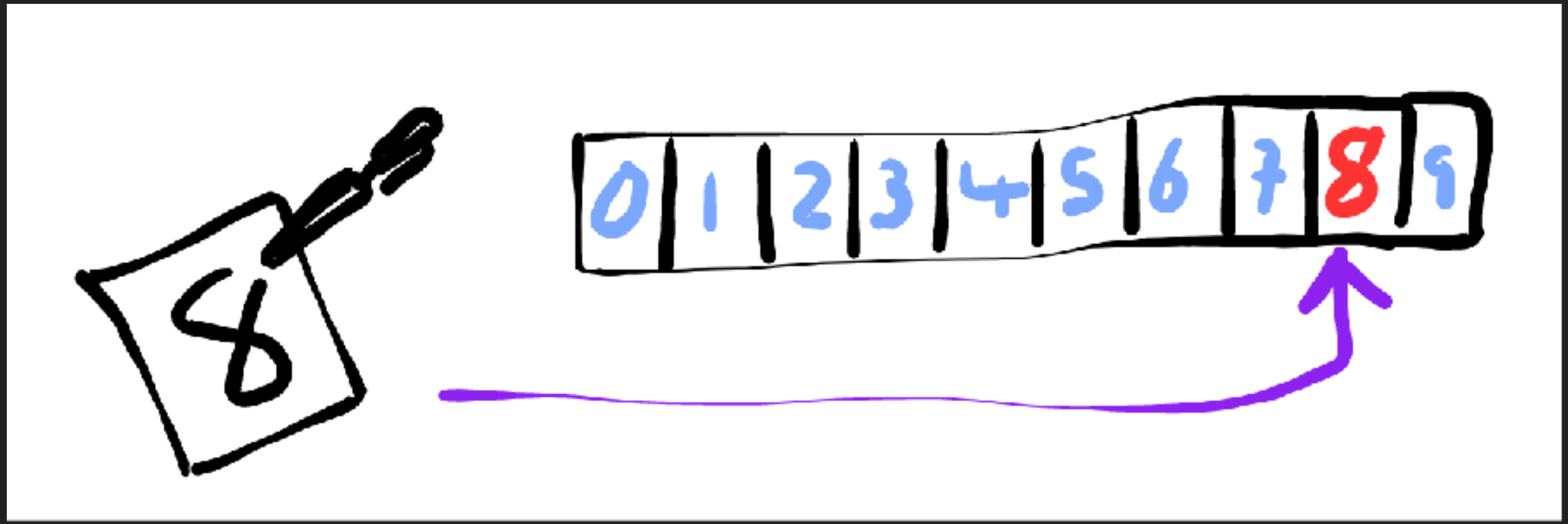
Neural Networks

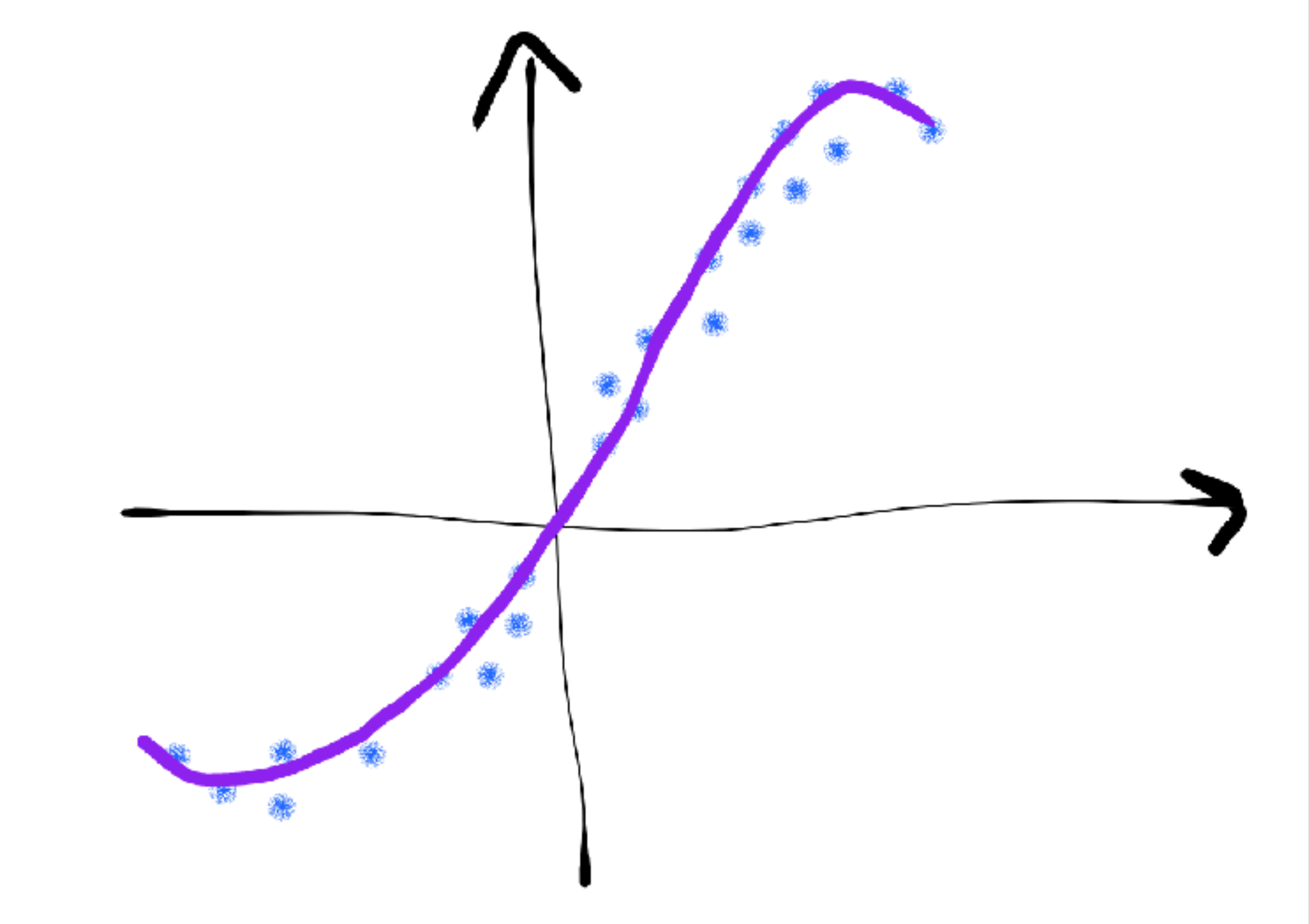


Walkthru


data
The Answer

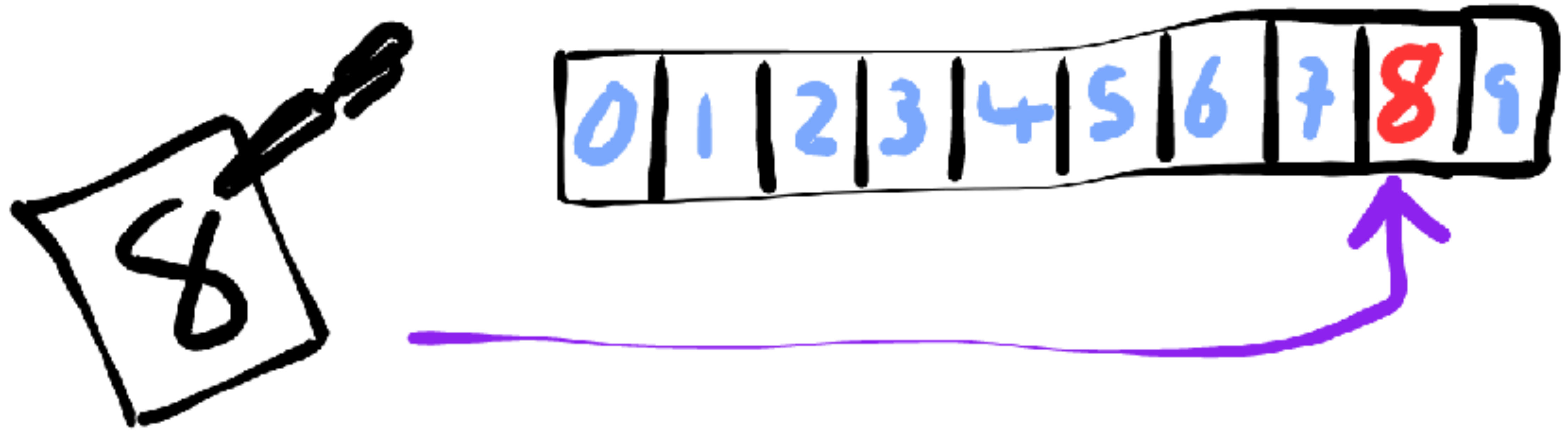
WHAT KINDS OF THINGS ARE WE TRYING TO WORK OUT?

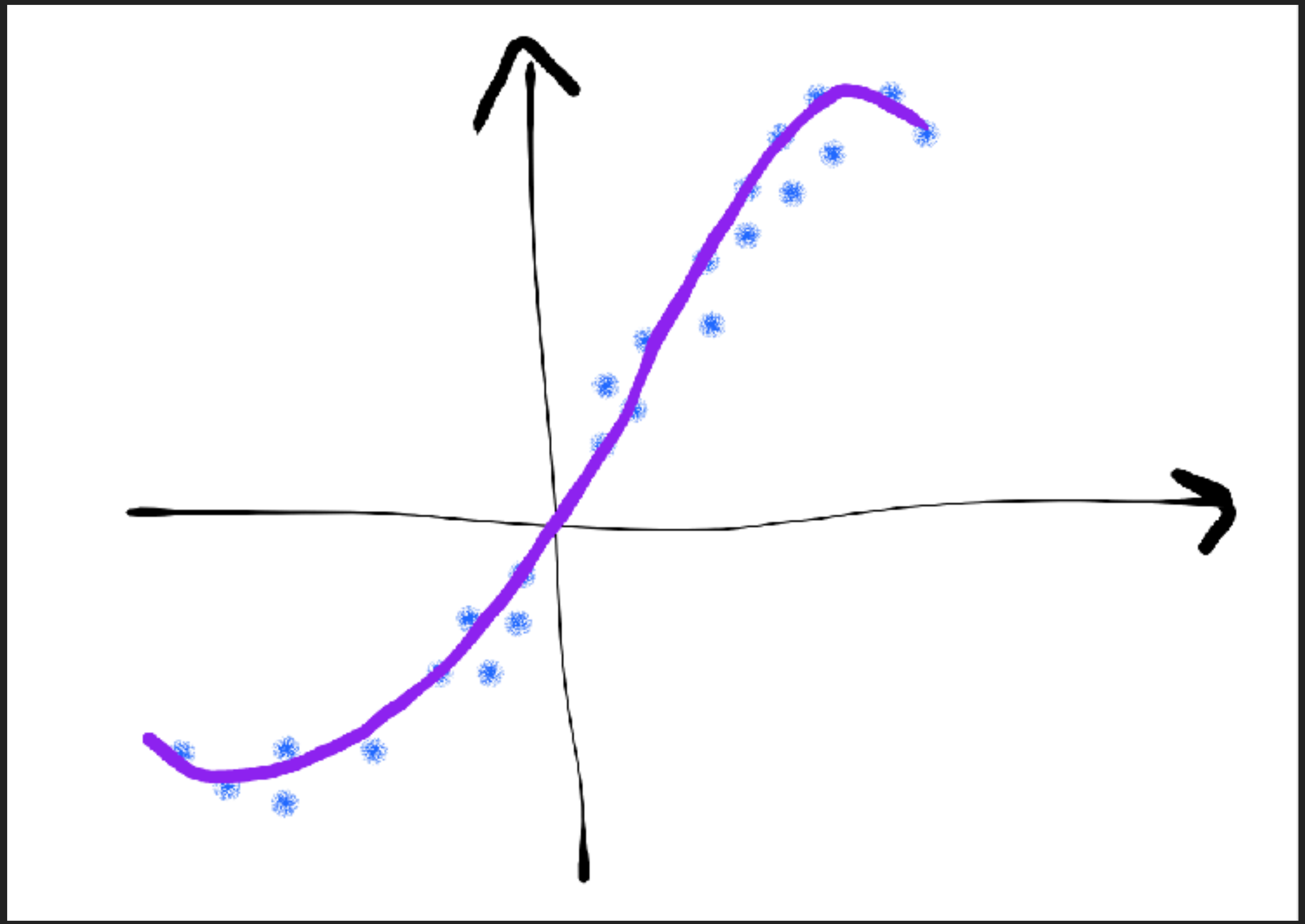


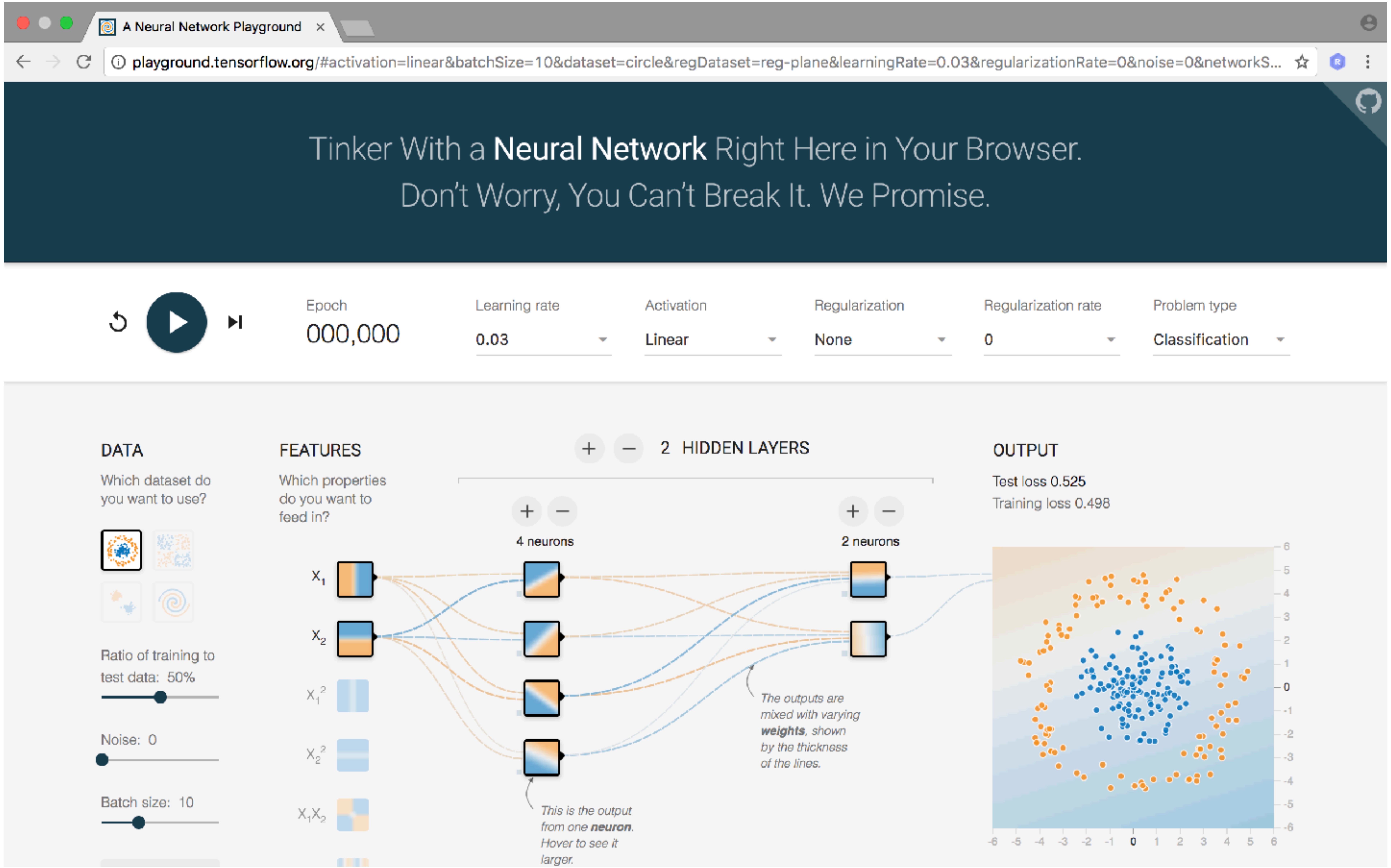


WHAT ARE WE WORKING OUT?

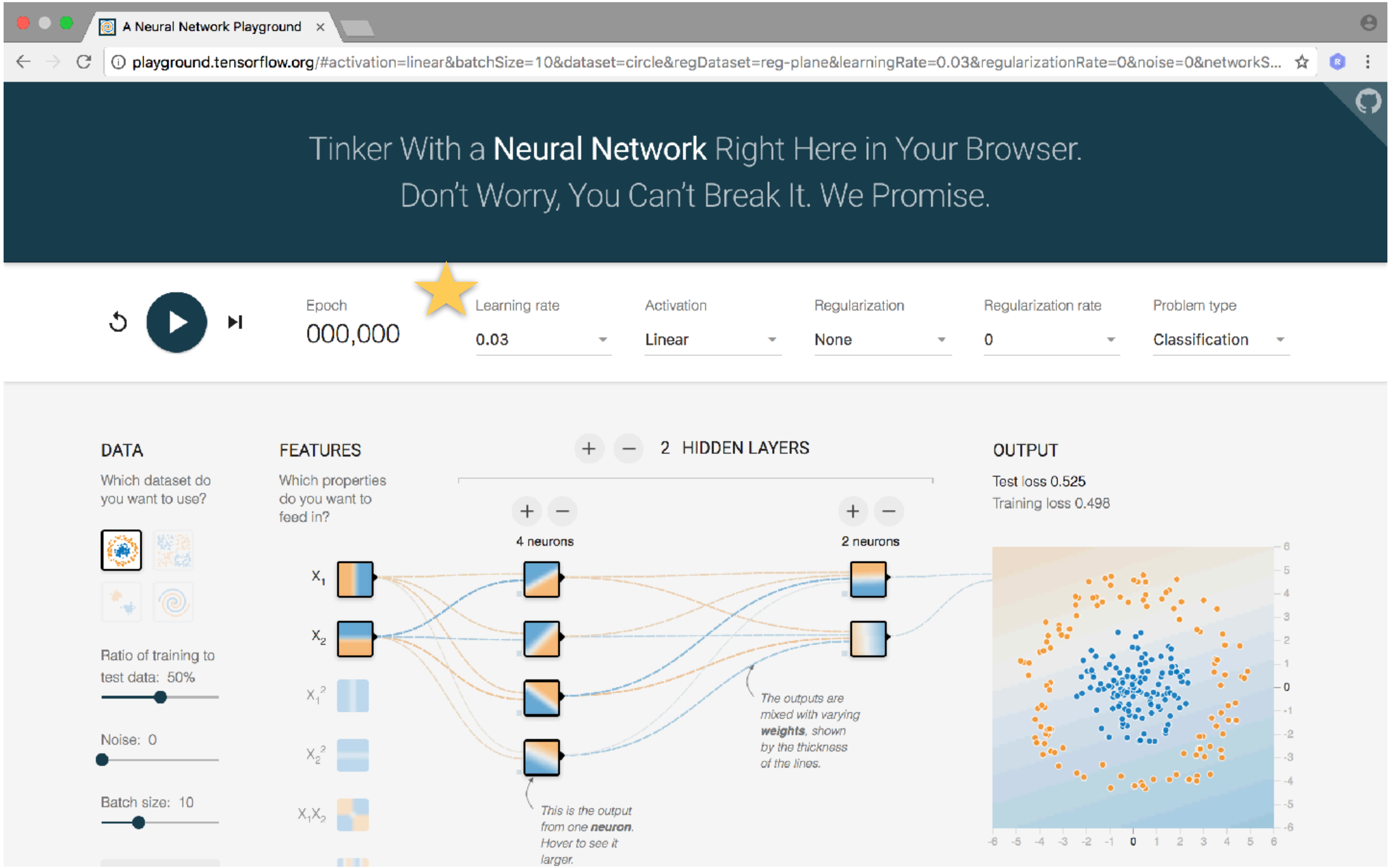
CLASSIFICATION OR REGRESSION


8
0
1
2
3
4
5
6
7
8
9

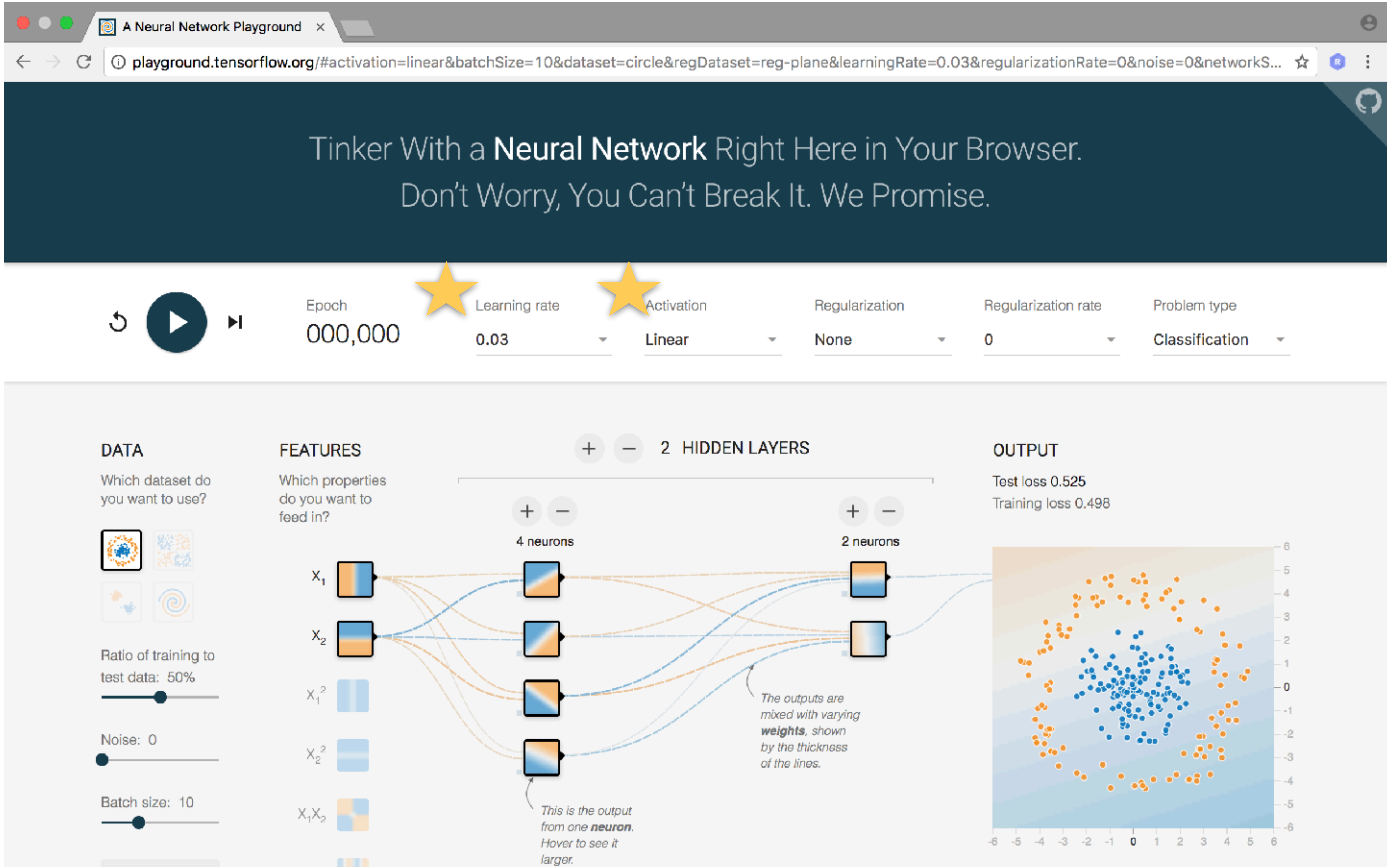




A Neural Network Playground
playground.tensorflow.org/#activation=linear&batchSize=10&dataset=circle&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networkS...
Tinker With a Neural Network Right Here in Your Browser.
Don't Worry, You Can't Break It. We Promise.
Epoch
000,000
Learning rate
0.03
Activation
Linear
Regularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
FEATURES
Which properties do you want to feed in?
2 HIDDEN LAYERS
4 neurons
2 neurons
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 0.525
Training loss 0.498

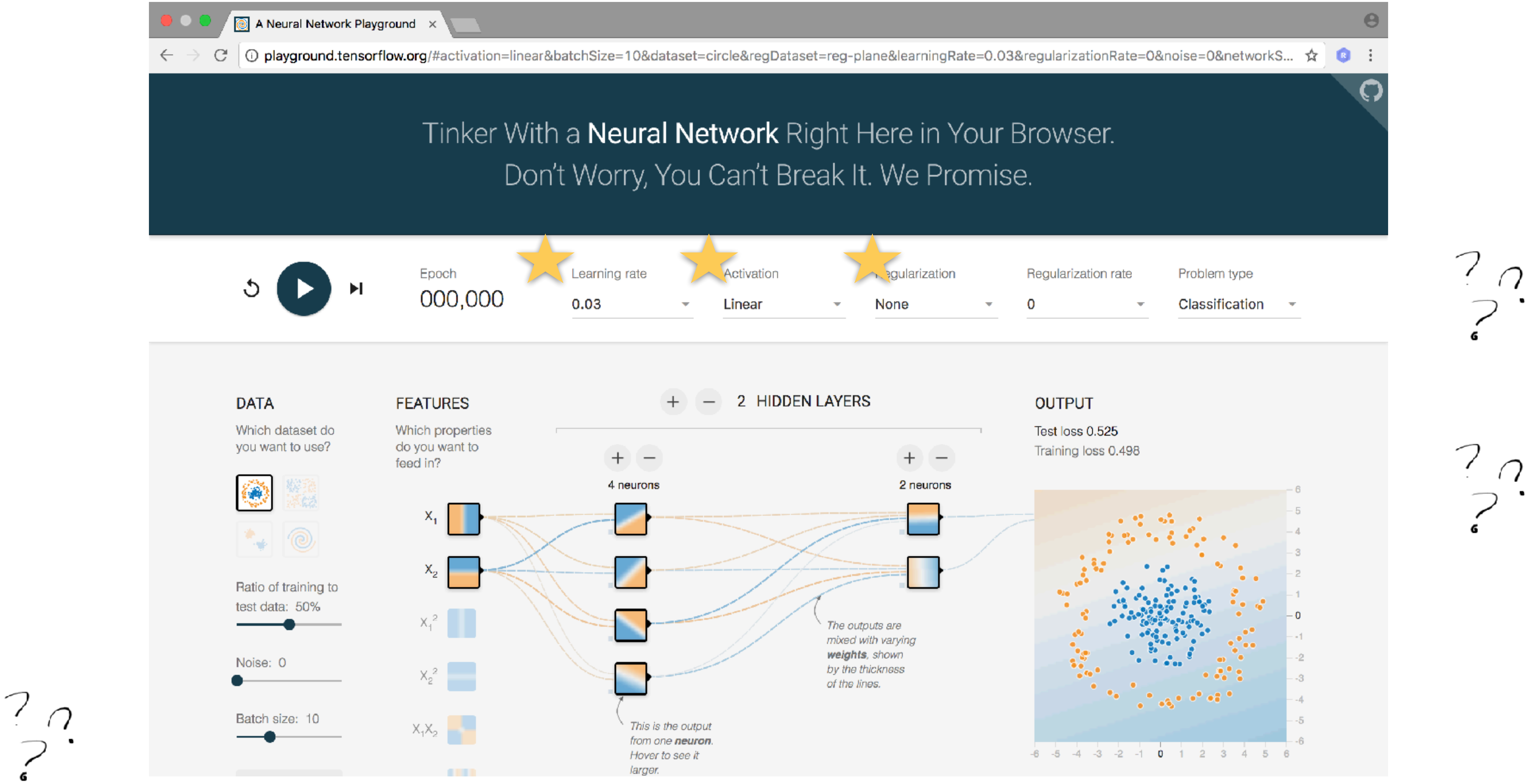

A Neural Network Playground
playground.tensorflow.org/#activation=linear&batchSize=10&dataset=circle&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networkS...
Tinker With a Neural Network Right Here in Your Browser.
Don't Worry, You Can't Break It. We Promise.
Epoch
000,000
Learning rate
0.03
Activation
Linear
Regularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
FEATURES
Which properties do you want to feed in?
2 HIDDEN LAYERS
4 neurons
2 neurons
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 0.525
Training loss 0.498

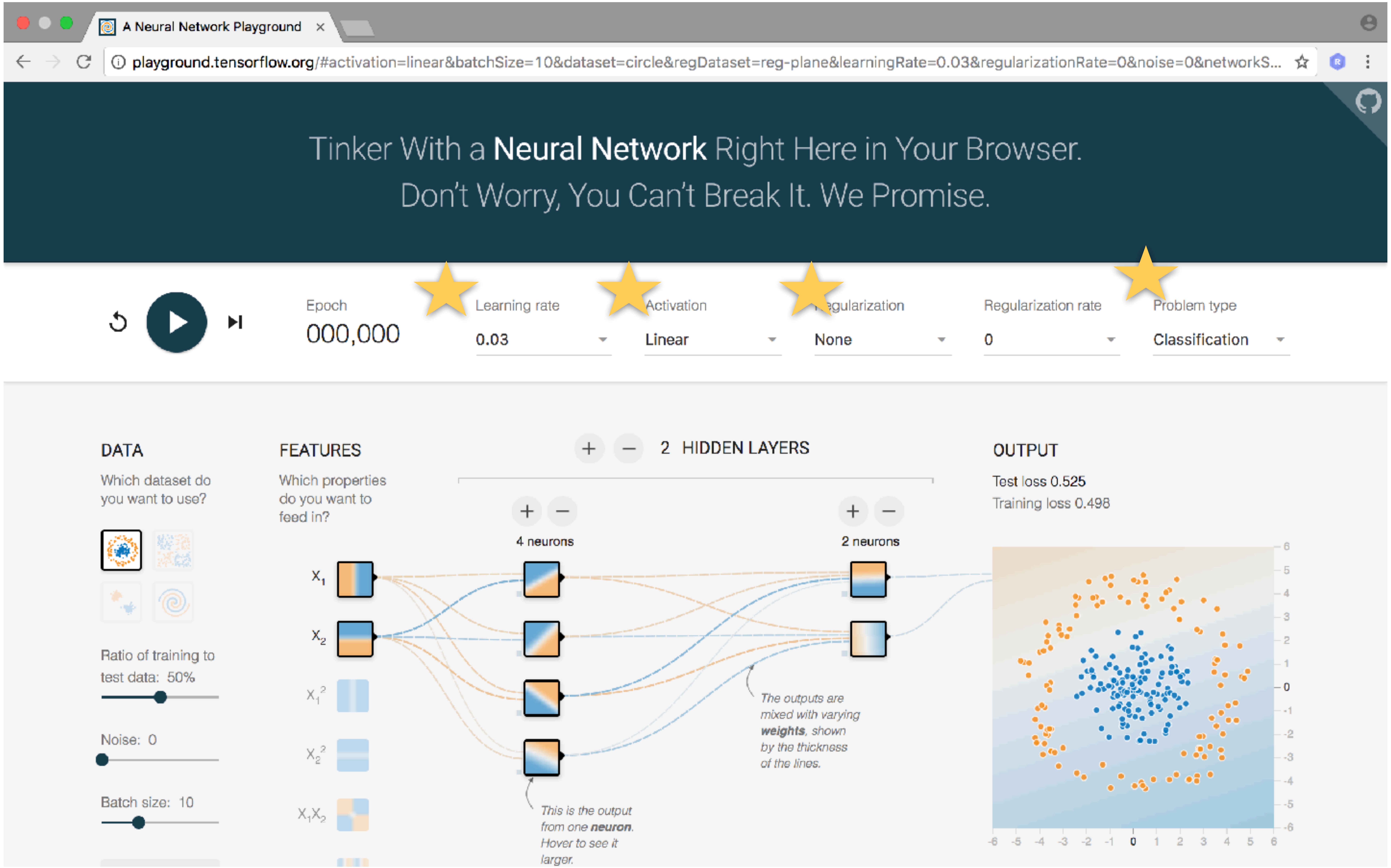
? ?
?

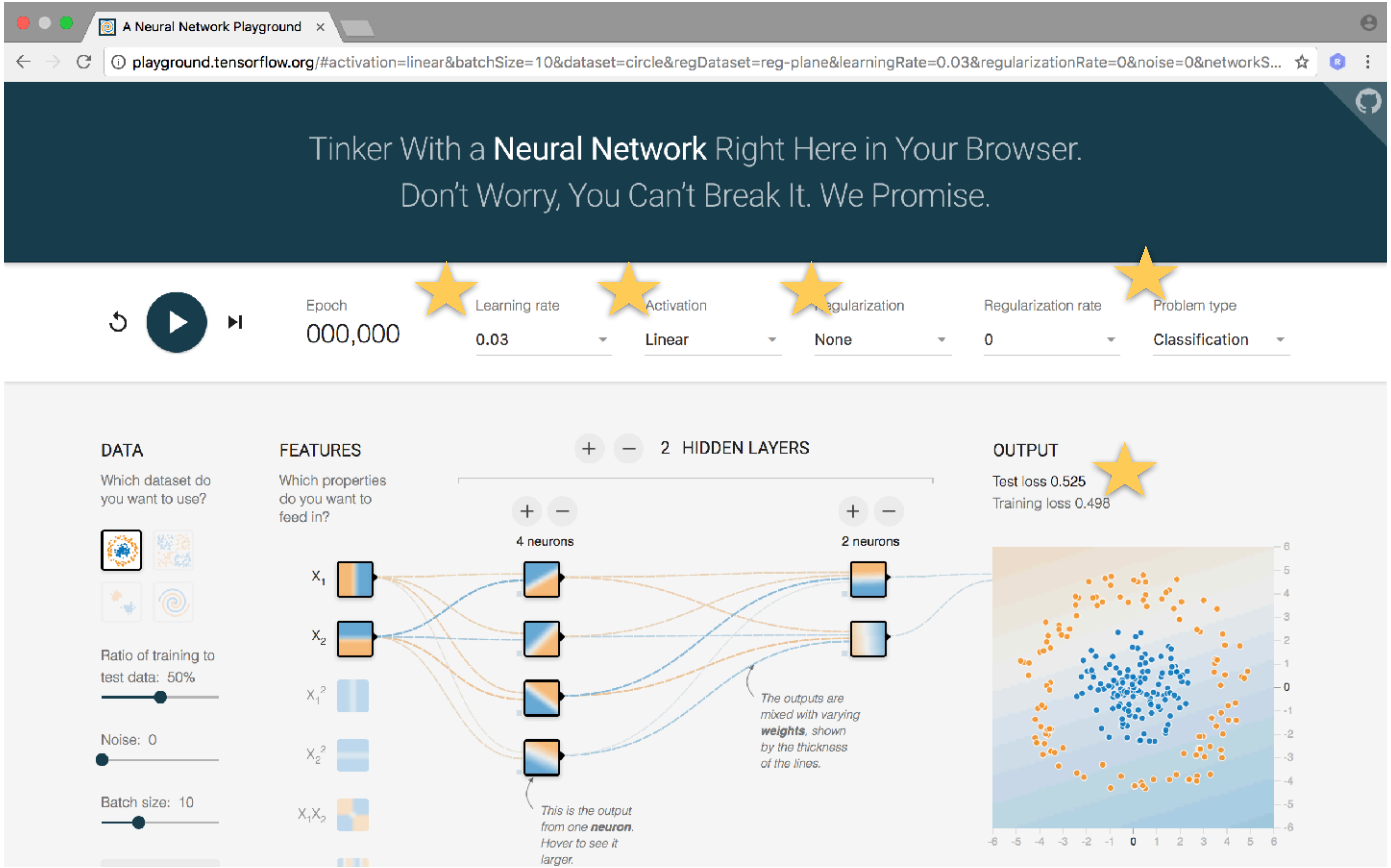

A Neural Network Playground
playground.tensorflow.org/#activation=linear&batchSize=10&dataset=circle&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networkS...
Tinker With a Neural Network Right Here in Your Browser.
Don't Worry, You Can't Break It. We Promise.
Epoch
000,000
Learning rate
0.03
Activation
Linear
Regularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
FEATURES
Which properties do you want to feed in?
2 HIDDEN LAYERS
4 neurons
2 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 lines.
OUTPUT
Test loss 0.525
Training loss 0.498

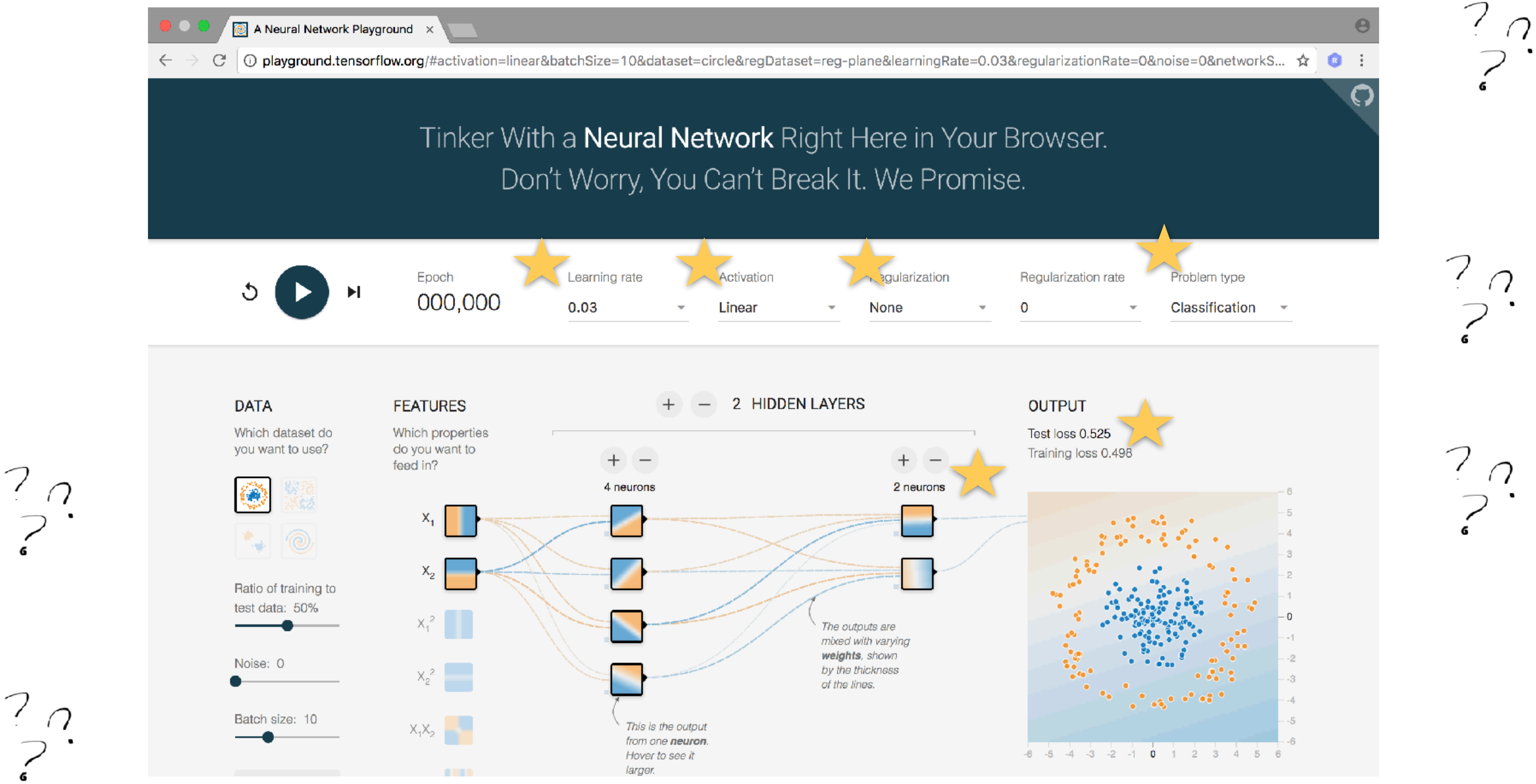
? ? ?

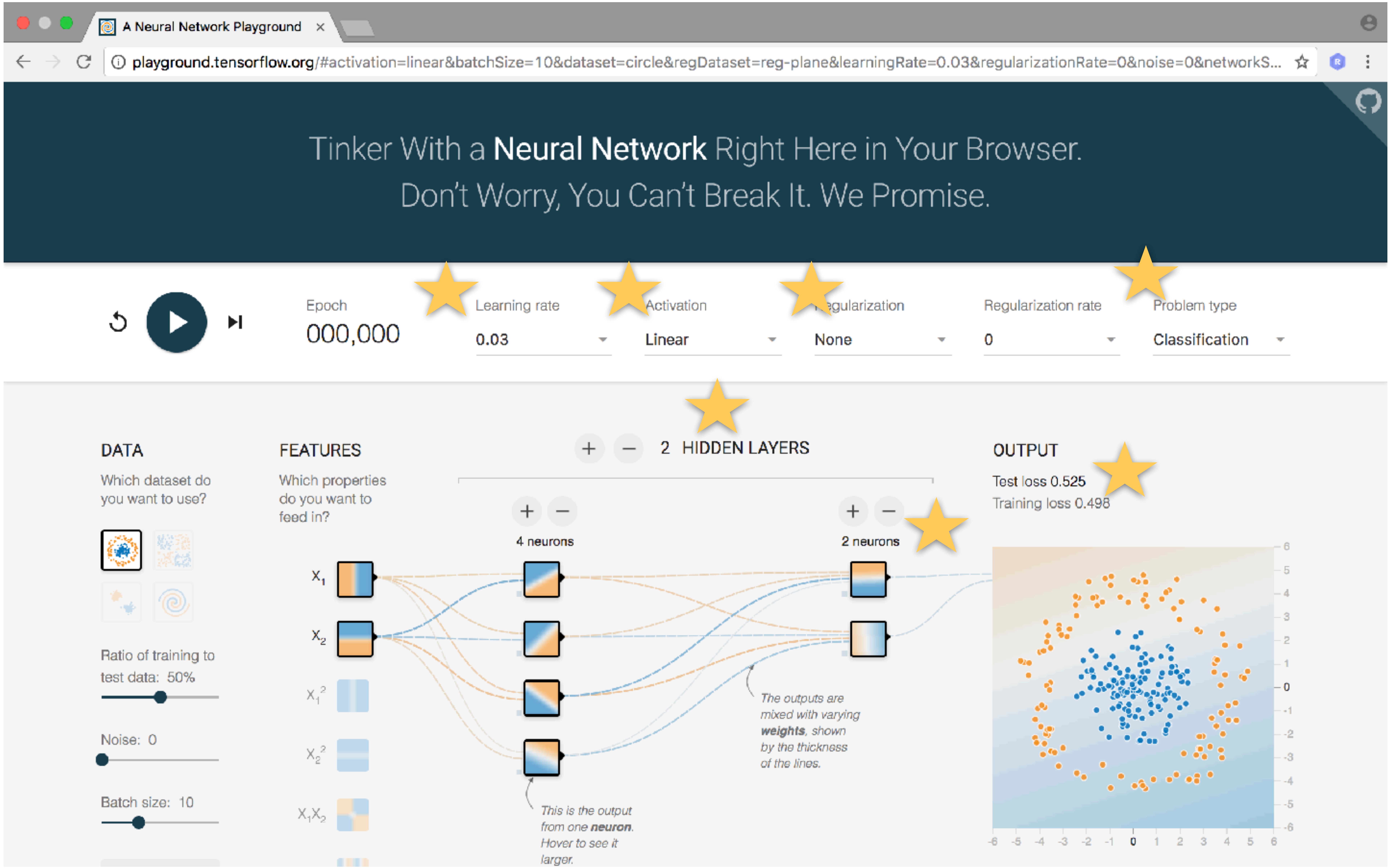
? ? ?

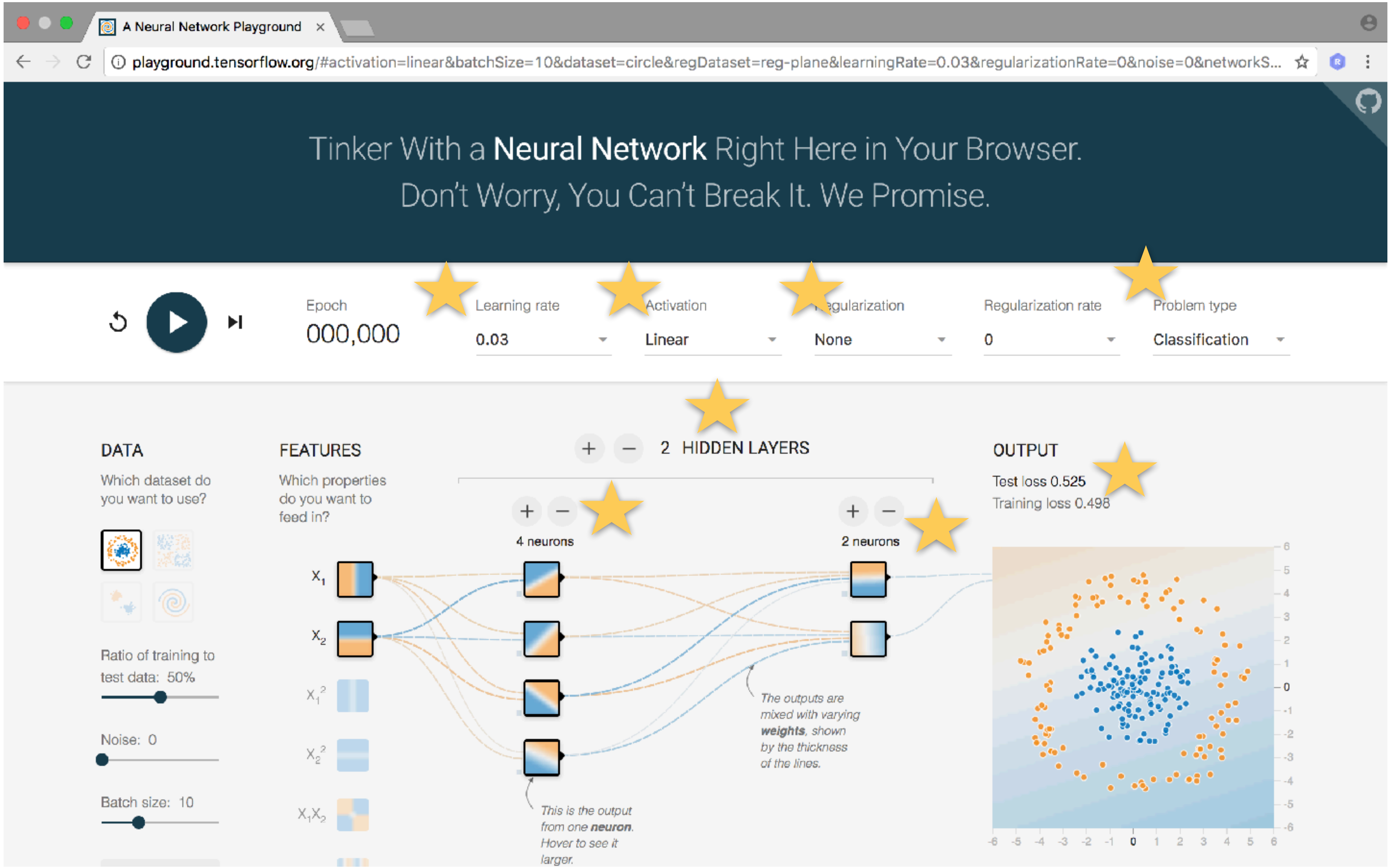

A Neural Network Playground
playground.tensorflow.org/#activation=linear&batchSize=10&dataset=circle&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networkS...
Tinker With a Neural Network Right Here in Your Browser.
Don't Worry, You Can't Break It. We Promise.
Epoch
000,000
Learning rate
0.03
Activation
Linear
Regularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
FEATURES
Which properties do you want to feed in?
2 HIDDEN LAYERS
4 neurons
2 neurons
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 0.525
Training loss 0.498
???
???
???

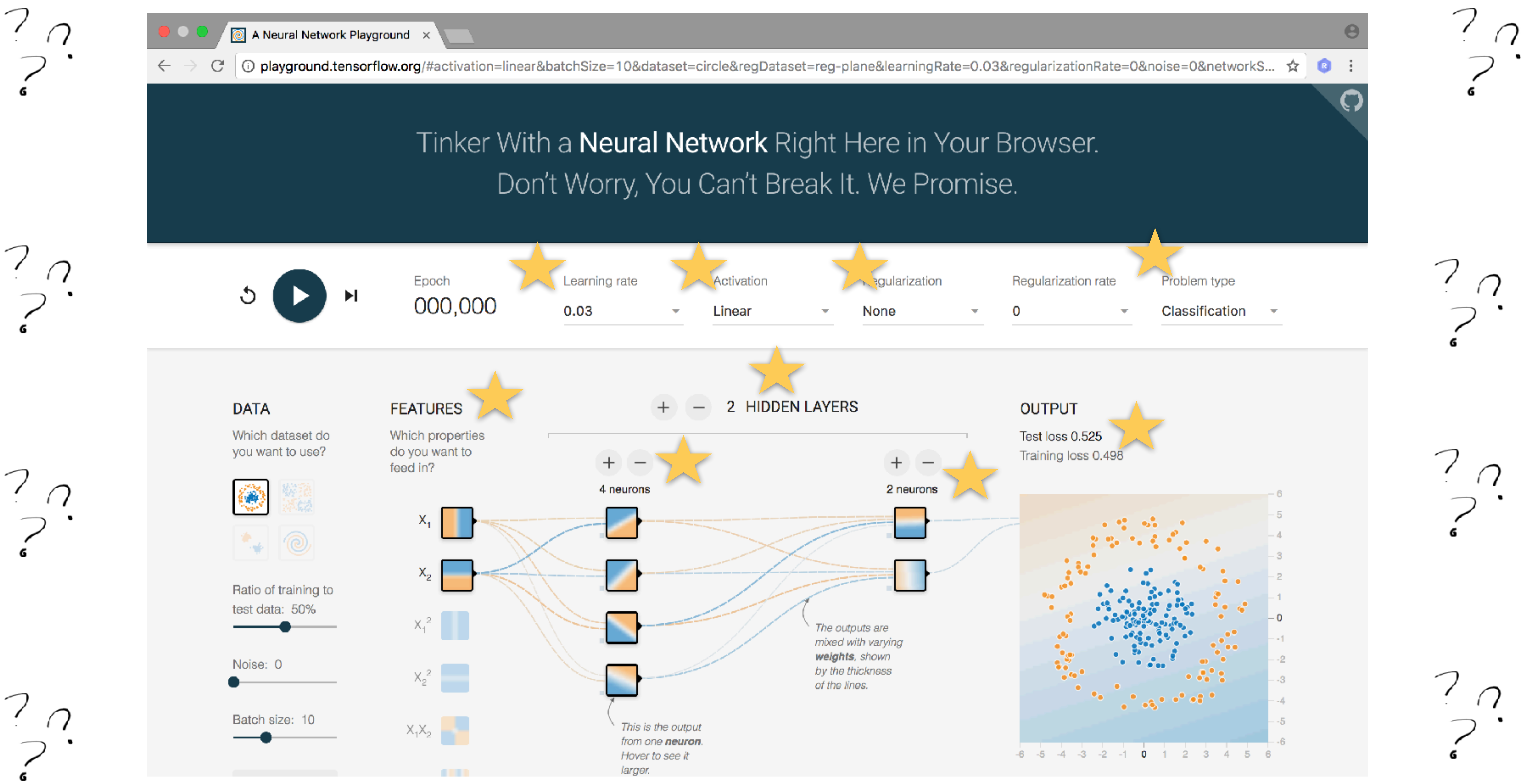

A Neural Network Playground
playground.tensorflow.org/#activation=linear&batchSize=10&dataset=circle&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networkS...
Tinker With a Neural Network Right Here in Your Browser.
Don't Worry, You Can't Break It. We Promise.
Epoch
000,000
Learning rate
0.03
Activation
Linear
Regularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
FEATURES
Which properties do you want to feed in?
2 HIDDEN LAYERS
4 neurons
2 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 lines.
OUTPUT
Test loss 0.525
Training loss 0.498


A Neural Network Playground
playground.tensorflow.org/#activation=linear&batchSize=10&dataset=circle&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networkS...
Tinker With a Neural Network Right Here in Your Browser.
Don't Worry, You Can't Break It. We Promise.
Epoch
000,000
Learning rate
0.03
Activation
Linear
Regularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
FEATURES
Which properties do you want to feed in?
2 HIDDEN LAYERS
4 neurons
2 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 lines.
OUTPUT
Test loss 0.525
Training loss 0.498


A Neural Network Playground
playground.tensorflow.org/#activation=linear&batchSize=10&dataset=circle&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networkS...
Tinker With a Neural Network Right Here in Your Browser.
Don't Worry, You Can't Break It. We Promise.
Epoch
000,000
Learning rate
0.03
Activation
Linear
Regularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
FEATURES
Which properties do you want to feed in?
2 HIDDEN LAYERS
4 neurons
2 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 lines.
OUTPUT
Test loss 0.525
Training loss 0.498


A Neural Network Playground
playground.tensorflow.org/#activation=linear&batchSize=10&dataset=circle&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networkS...
Tinker With a Neural Network Right Here in Your Browser.
Don't Worry, You Can't Break It. We Promise.
Epoch
000,000
Learning rate
0.03
Activation
Linear
egularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
FEATURES
Which properties do you want to feed in?
2 HIDDEN LAYERS
4 neurons
2 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 lines.
OUTPUT
Test loss 0.525
Training loss 0.498


A Neural Network Playground
playground.tensorflow.org/#activation=linear&batchSize=10&dataset=circle&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networkS...
Tinker With a Neural Network Right Here in Your Browser.
Don't Worry, You Can't Break It. We Promise.
Epoch
000,000
Learning rate
0.03
Activation
Linear
Regularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
FEATURES
Which properties do you want to feed in?
2 HIDDEN LAYERS
4 neurons
2 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 lines.
OUTPUT
Test loss 0.525
Training loss 0.498



A Neural Network Playground
playground.tensorflow.org/#activation=linear&batchSize=10&dataset=circle&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networkS...
Tinker With a Neural Network Right Here in Your Browser.
Don't Worry, You Can't Break It. We Promise.
Epoch
000,000
Learning rate
0.03
Activation
Linear
Regularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
FEATURES
Which properties do you want to feed in?
2 HIDDEN LAYERS
4 neurons
2 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 lines.
OUTPUT
Test loss 0.525
Training loss 0.498

PART I: NEURONS

- Neurons
  - Biological
  - Artificial
    - Linear Neuron
    - Binary Threshold Neuron
    - Logistic/sigmoid Neuron
    - reLU
    - Tanh

PART II: LEARNING

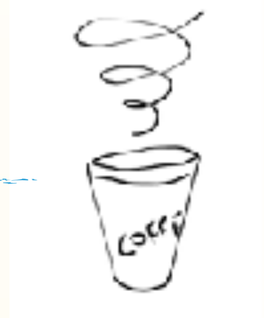
- Network
  - Feed forward
    - Cost functions
  - Back propagation
    - Training
      - Gradient descent
      - Learning rate
      - Regularisation

PART III: IN A NETWORK

- Learning to map input to output
  - Architectures
    - CNN
    - RNN
    - And beyond…

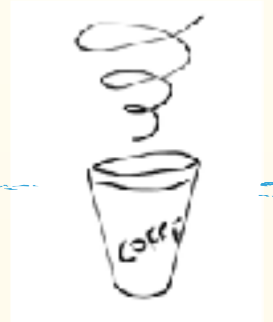
PART I: NEURONS

- Neurons
  - Biological
  - Artificial
    - Linear Neuron
    - Binary Threshold Neuron
    - Logistic/sigmoid Neuron
    - reLU
    - Tanh



PART II: LEARNING

- Network
  - Feed forward
    - Cost functions
  - Back propagation
    - Training
      - Gradient descent
      - Learning rate
      - Regularisation



PART III: IN A NETWORK

- Learning to map input to output
  - Architectures
    - CNN
    - RNN
    - And beyond…

@KATHARINECODES

PART I: NEURONS

WHAT IS A NEURON

PART I: NEURONS

- **Neurons**
  - **Biological**
  - Artificial
    - Linear Neuron
    - Binary Threshold Neuron
    - Logistic/sigmoid Neuron
    - reLU
    - Tanh

PART II: LEARNING

- Network
  - Feed forward — Cost functions
  - Back propagation — Training
    - Gradient descent
    - Learning rate
    - Regularisation

PART III: IN A NETWORK

- Learning to map input to output — Architectures
  - CNN
  - RNN
  - And beyond…

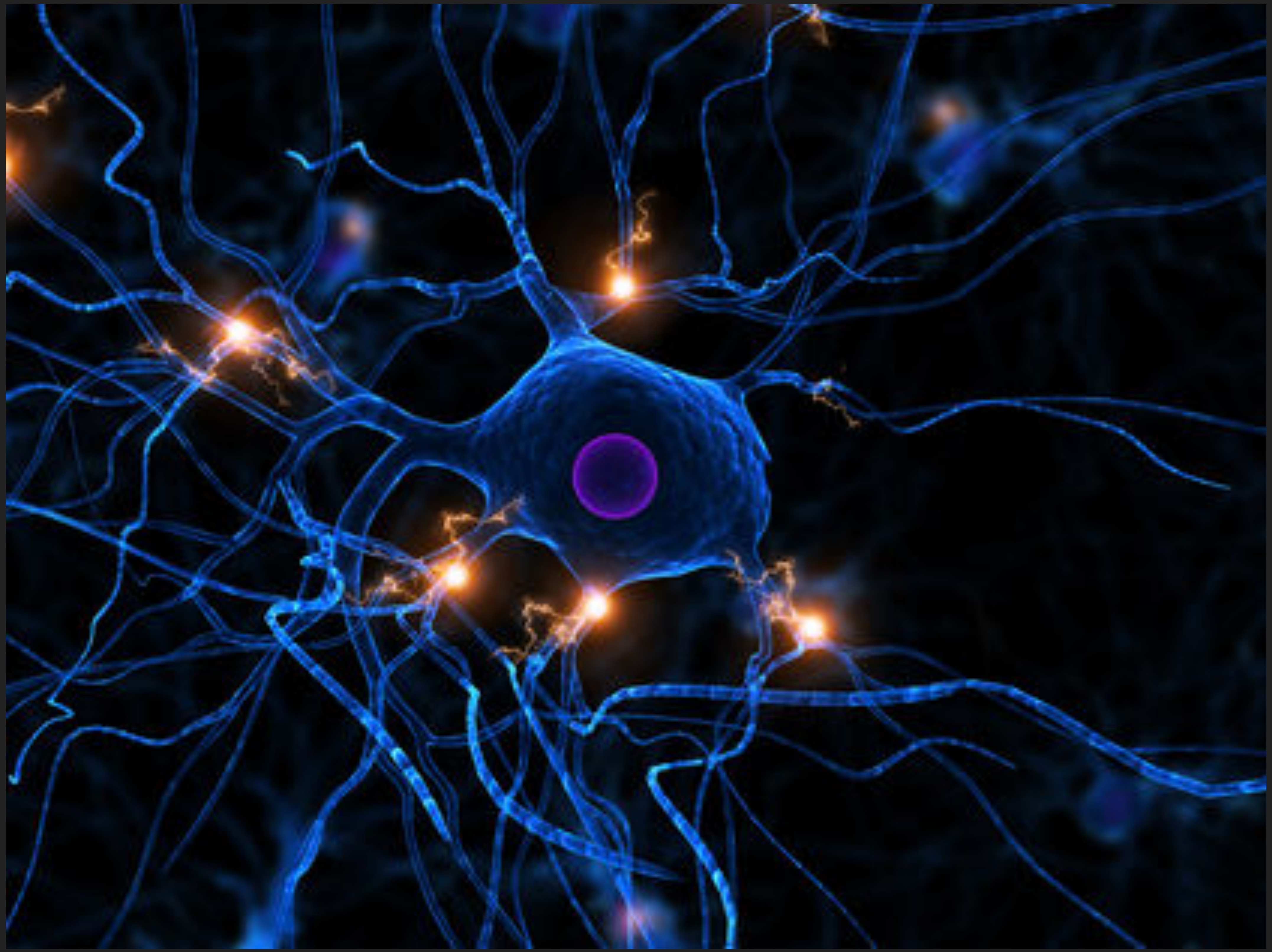


IMAGE CREDIT: SEBASTIAN KAULITZKI

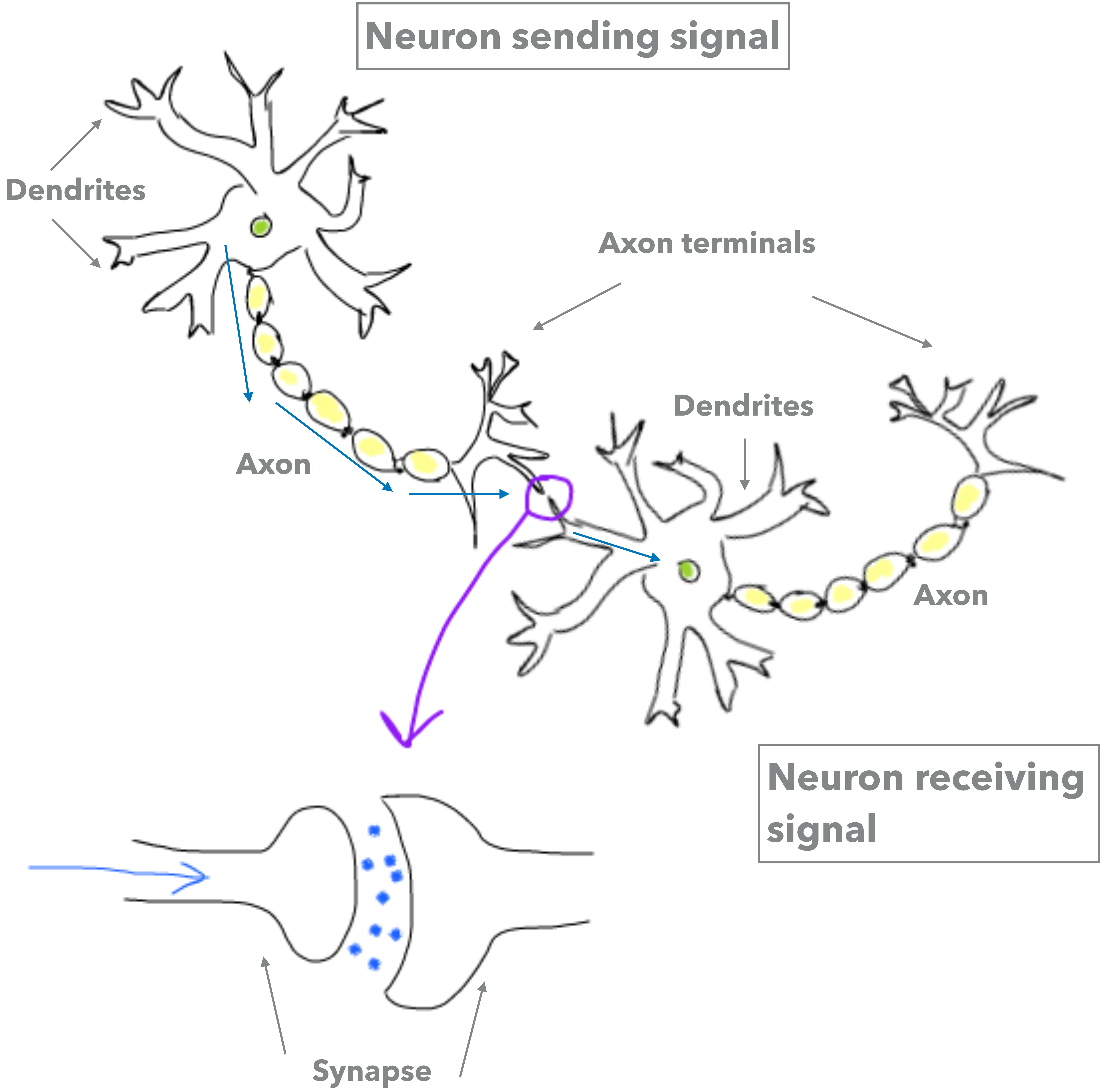
HOW MANY NEURONS ARE THERE IN THE HUMAN BRAIN?

86–100 BILLION

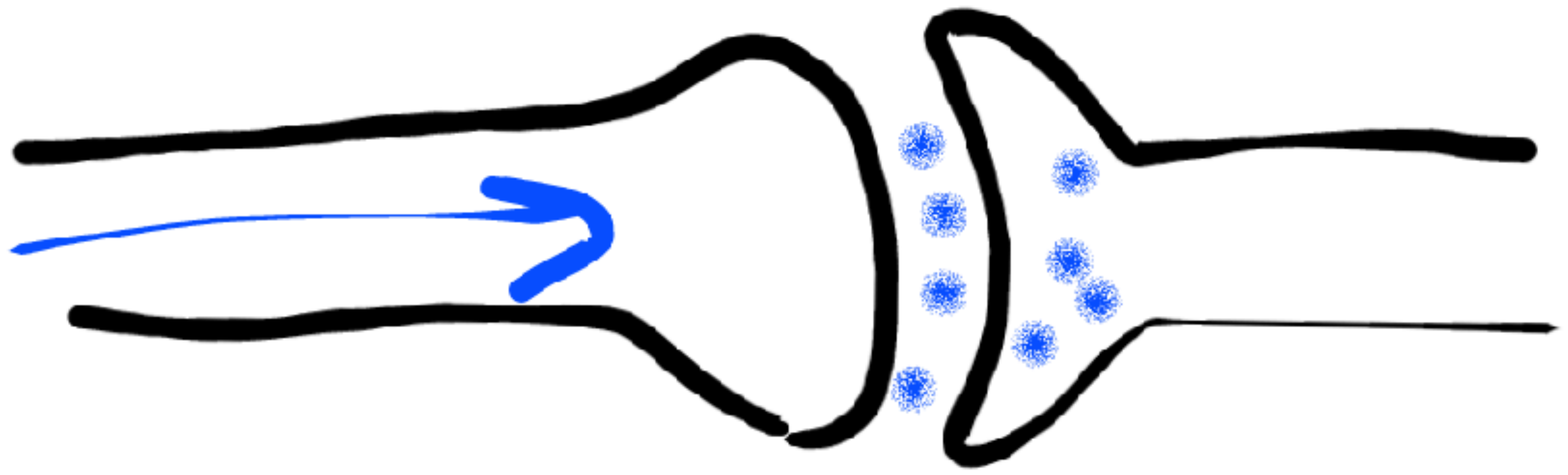
DR SUZANA HERCULANO-HOUZEL

(https://www.theguardian.com/science/blog/2012/feb/28/how-many-neurons-human-brain)

DISCLAIMER: NOT A BABOON


Neuron sending signal
Dendrites
Axon terminals
Dendrites
Axon
Axon
Neuron receiving signal
Synapse

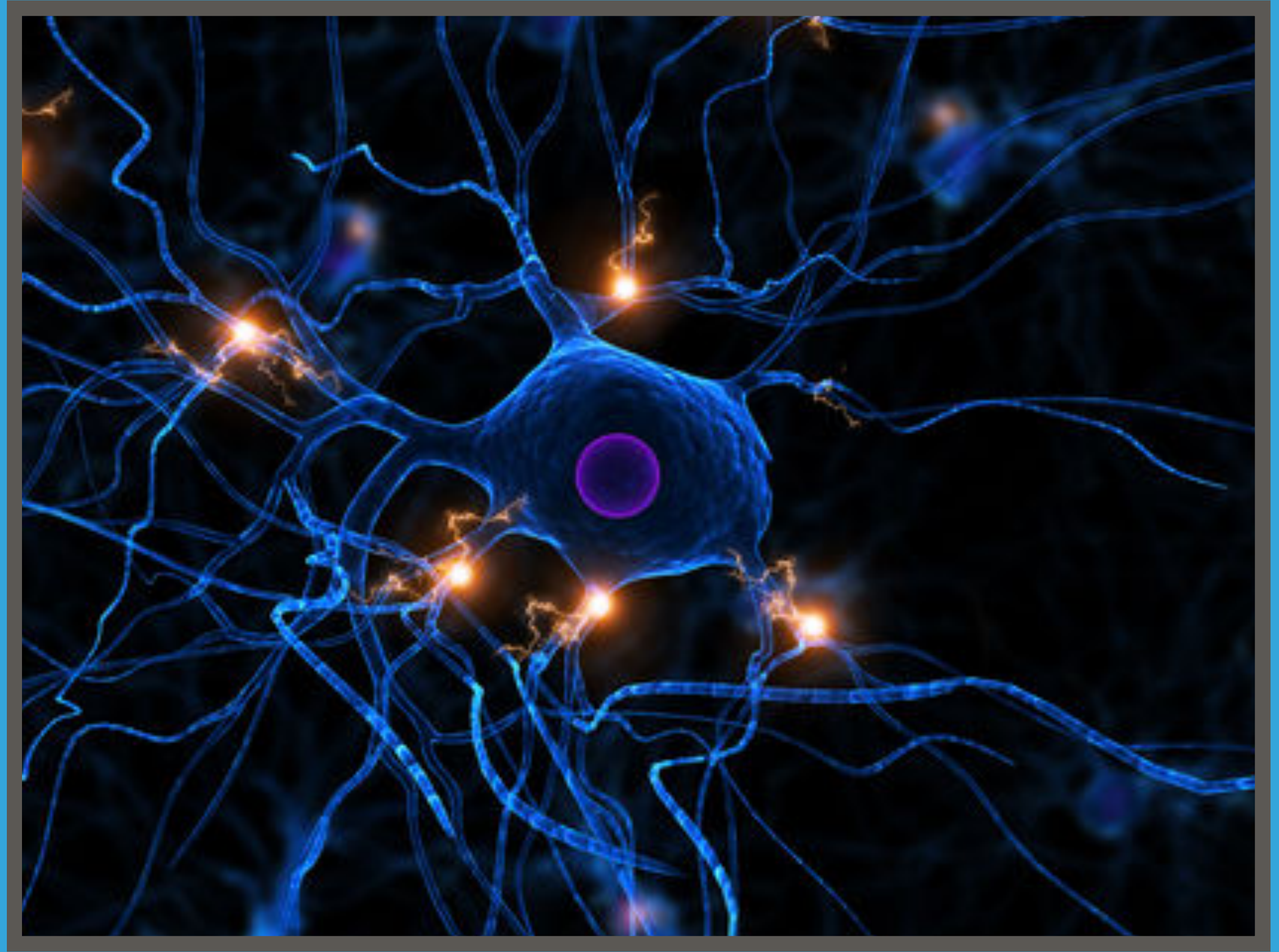
SYNAPSES ADAPT



- Send more
- Receive more

NEURONS ARE…

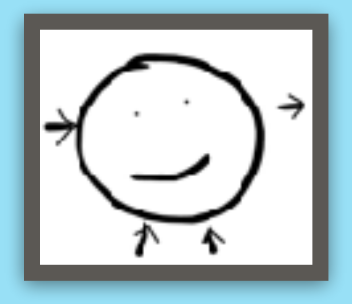
- **Fundamental units** of the brain (….or are they?)
- Work together in a **modular** brain
- Connections between them (synapses) can **adapt**



Biological Neurons

MATHS REQUIRED

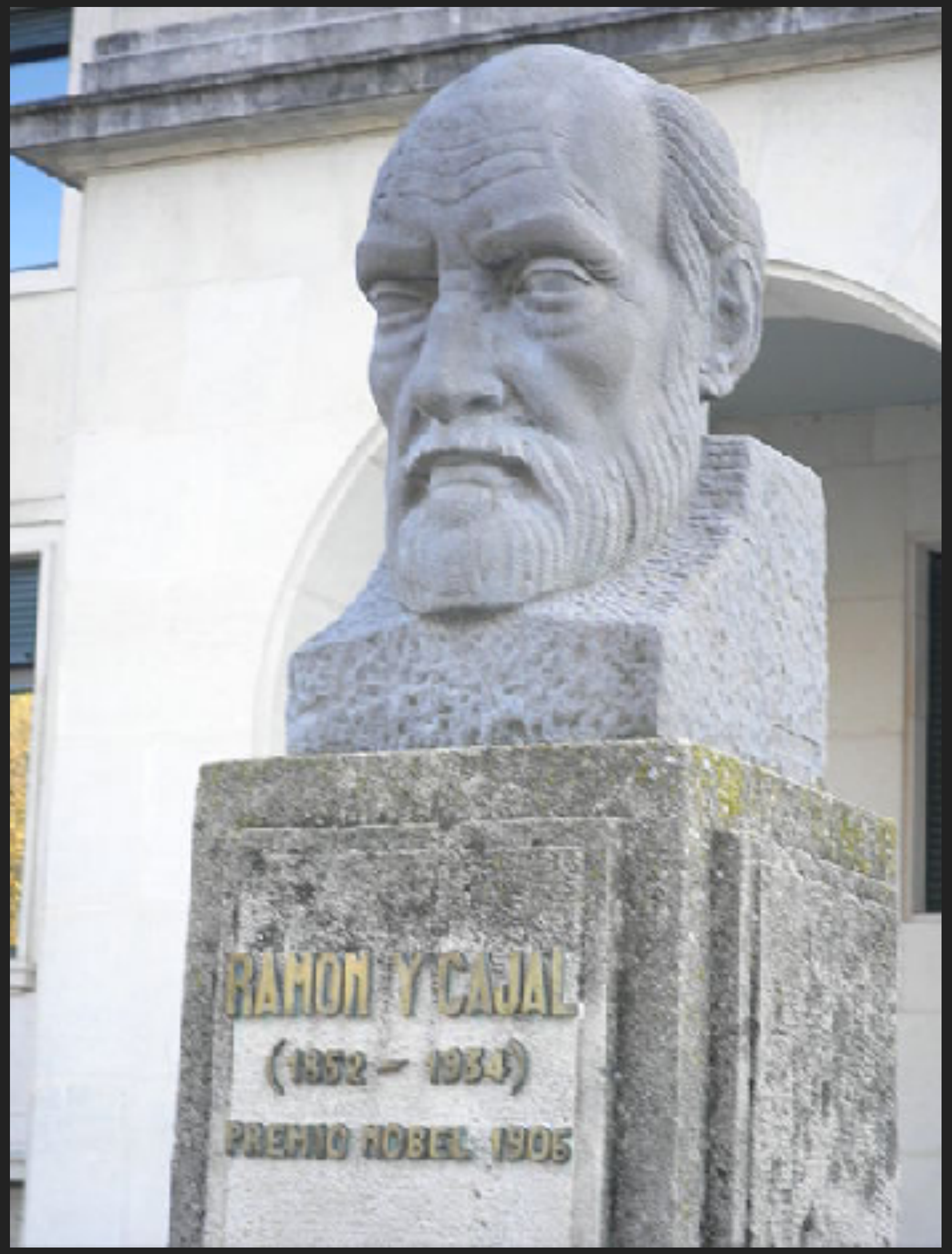
DIFFICULTY



Fundamental units

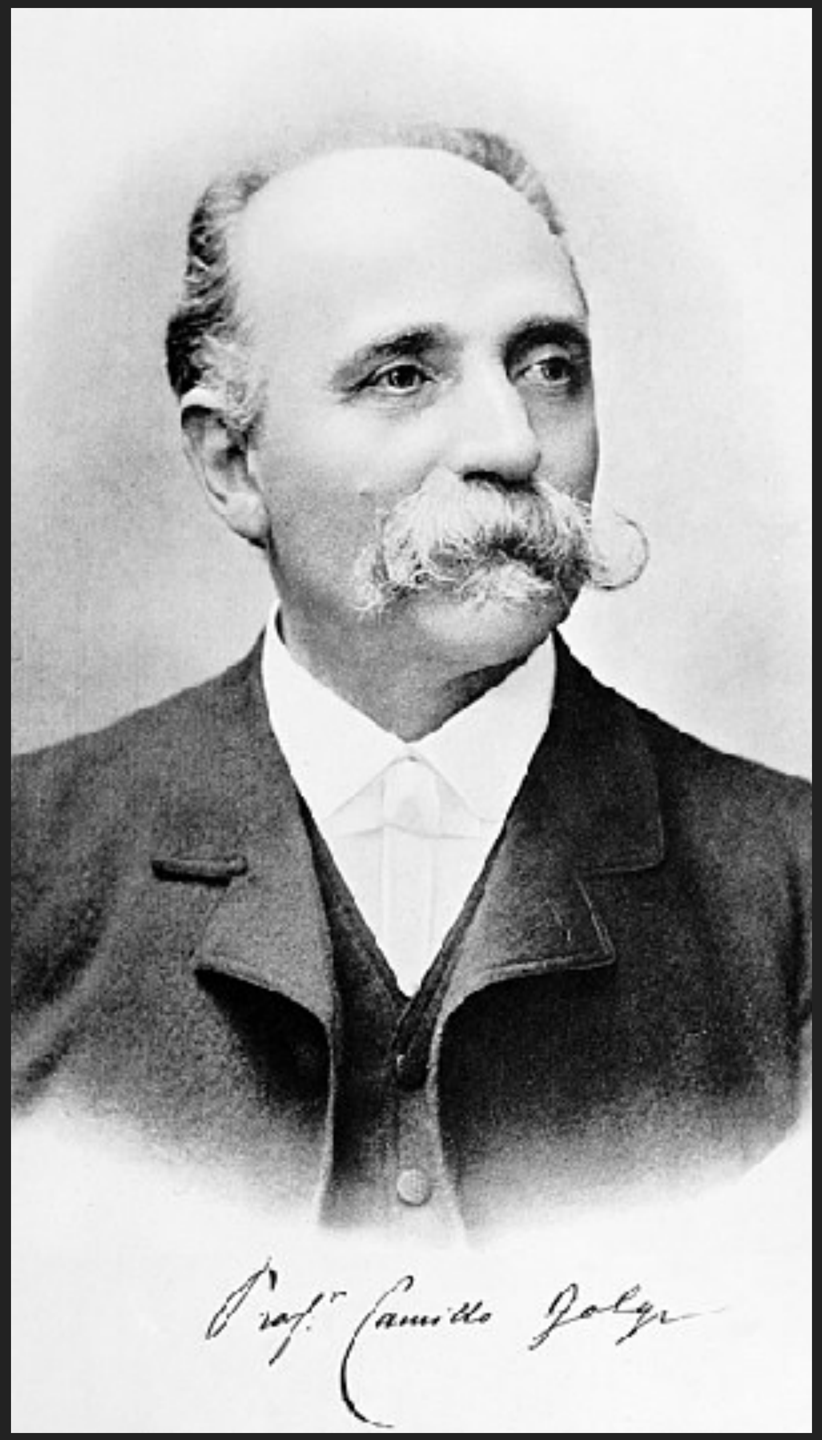
Work together in modular brain

SHOULD IT BE THE SMALLEST UNIT

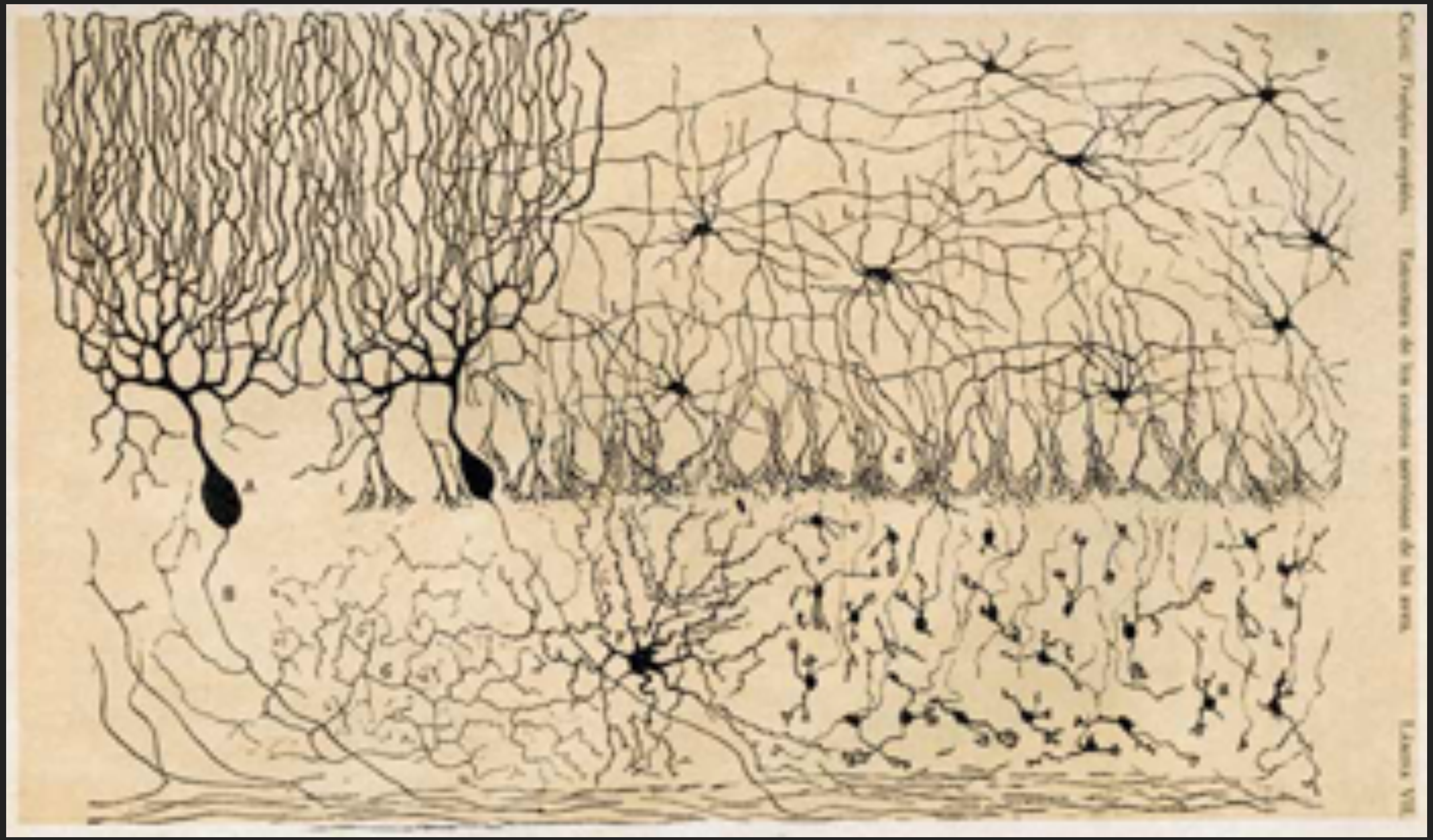


SANTIAGO RAMÓN Y CAJAL

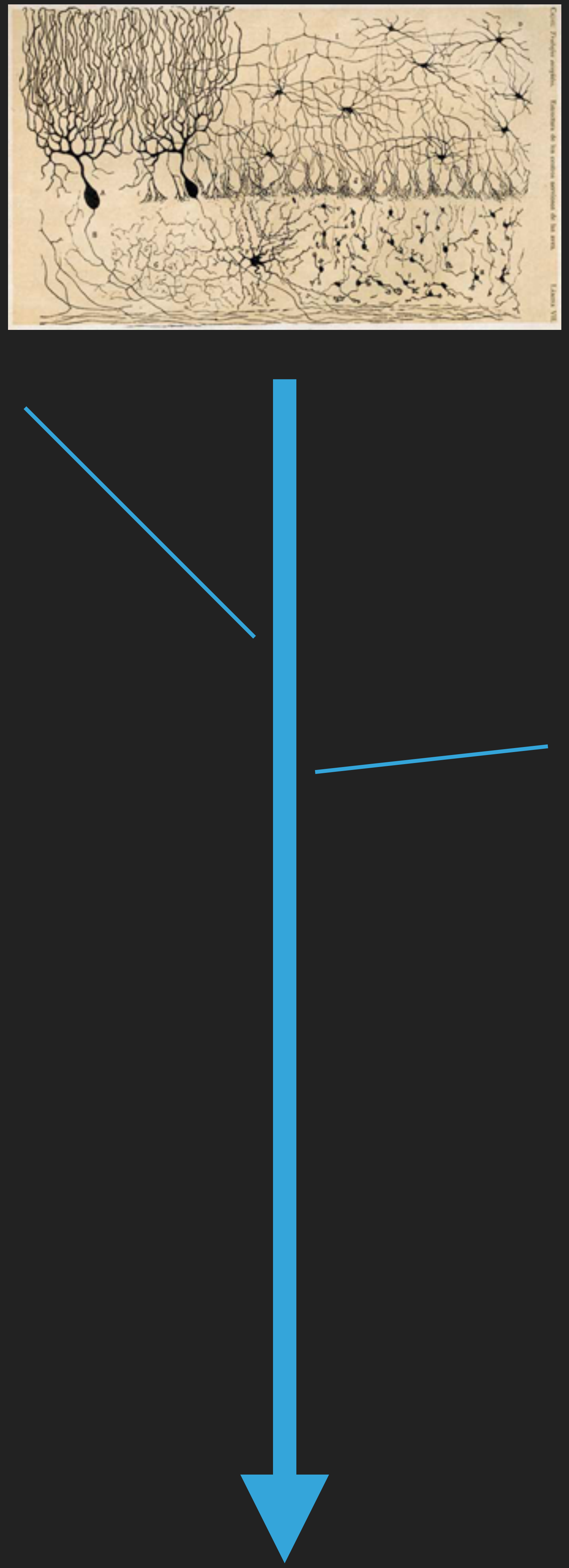
CAMILLO GOLGI



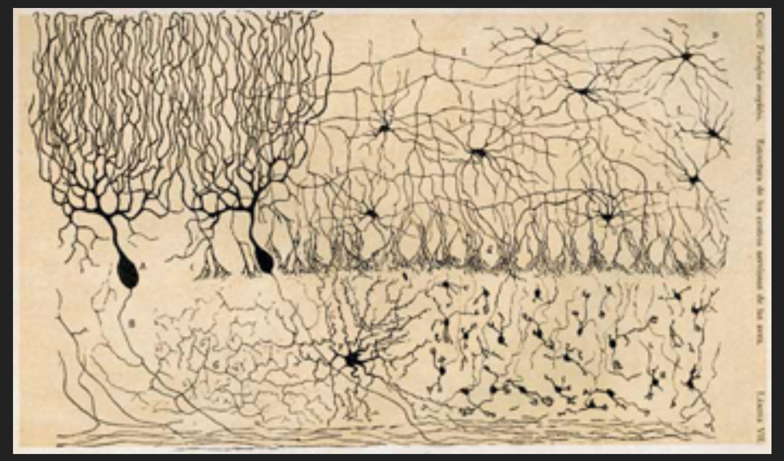
1906 NOBEL PRIZE FOR MEDICINE AND PHYSIOLOGY



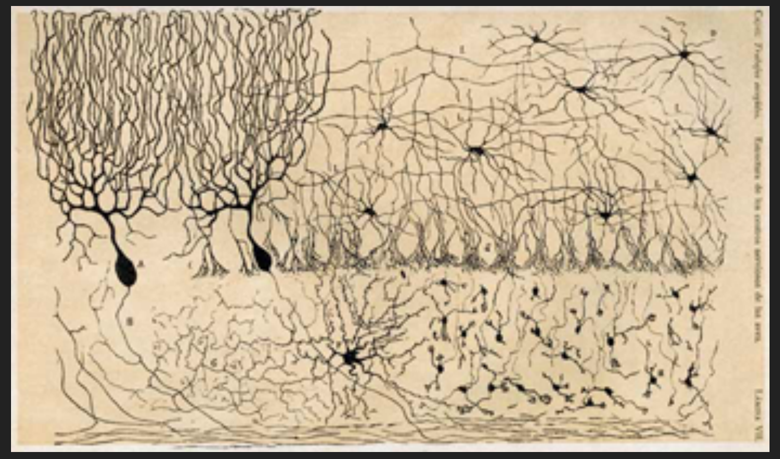
Santiago Ramón y Cajal



1943 Warren McCulloch and Walter Pitts



'A logical calculus of the ideas immanent in nervous activity'



1943 Warren McCulloch and Walter Pitts

'A logical calculus of the ideas immanent in nervous activity'

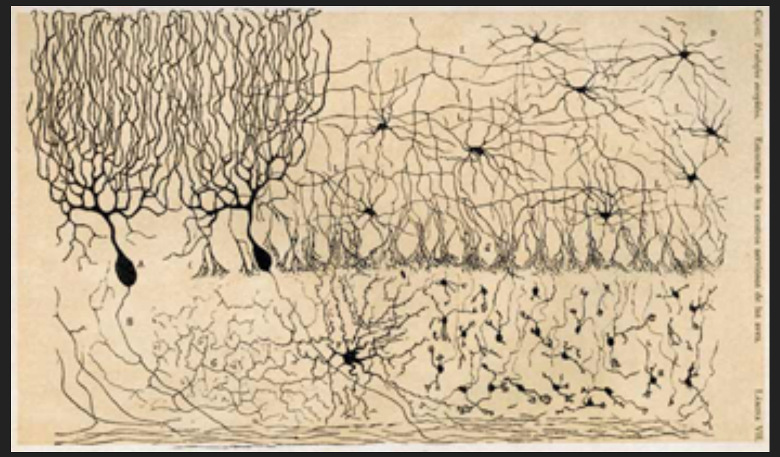
1960 Frank Rosenblatt

Mark I Perceptron

“IT IS NOT MY AIM TO SURPRISE OR SHOCK YOU – BUT THE SIMPLEST WAY I CAN SUMMARISE IS TO SAY THAT THERE ARE NOW IN THE WORLD MACHINES THAT THINK, THAT LEARN AND THAT CREATE. MOREOVER, THEIR ABILITY TO DO THESE THINGS IS GOING TO INCREASE RAPIDLY UNTIL – IN A VISIBLE FUTURE – THE RANGE OF PROBLEMS THEY CAN HANDLE WILL BE COEXTENSIVE WITH THE RANGE TO WHICH THE HUMAN MIND HAS BEEN APPLIED.”

Herbert Simon in 1957

WHO IS THIS MAN?

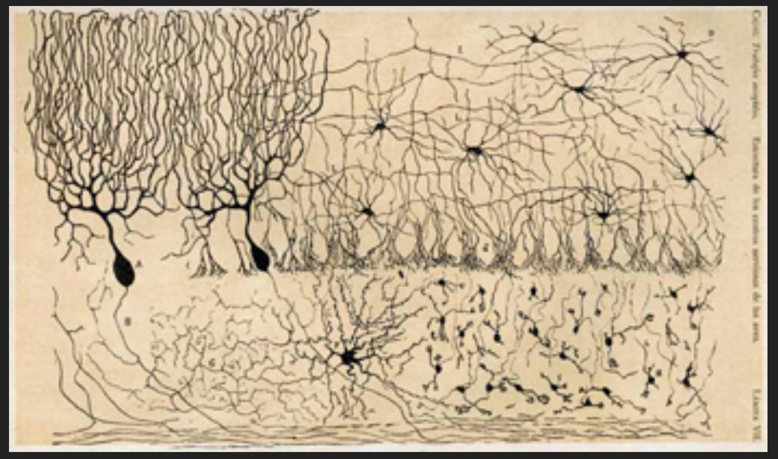


1943 Warren McCulloch and Walter Pitts

'A logical calculus of the ideas immanent in nervous activity'

1960 Frank Rosenblatt

Mark I Perceptron



1943 Warren McCulloch and Walter Pitts

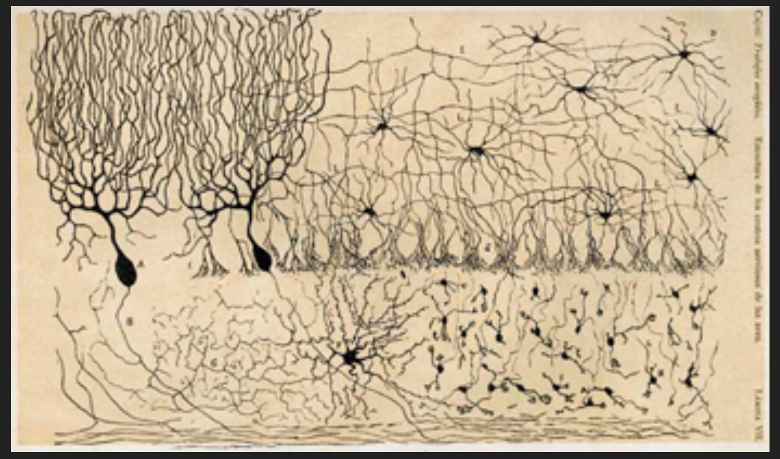
'A logical calculus of the ideas immanent in nervous activity'

1960 Frank Rosenblatt

Mark I Perceptron

1969 Marvin Minksy and Seymour Papert

'Perceptrons'



1943 Warren McCulloch and Walter Pitts

'A logical calculus of the ideas immanent in nervous activity'

1960 Frank Rosenblatt

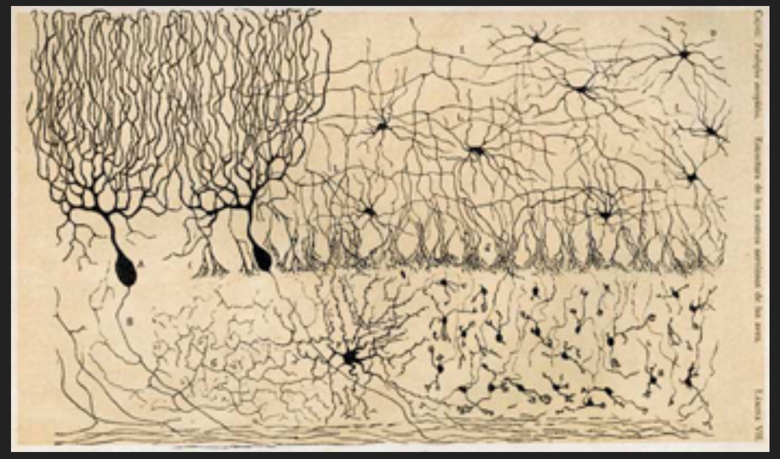
Mark I Perceptron

1969 Marvin Minksy and Seymour Papert

'Perceptrons'

1986 David E. Rumelhart, Geoffrey Hinton & Ronald Williams

'Learning representations by back-propagating errors'



1943 Warren McCulloch and Walter Pitts

'A logical calculus of the ideas immanent in nervous activity'

1960 Frank Rosenblatt

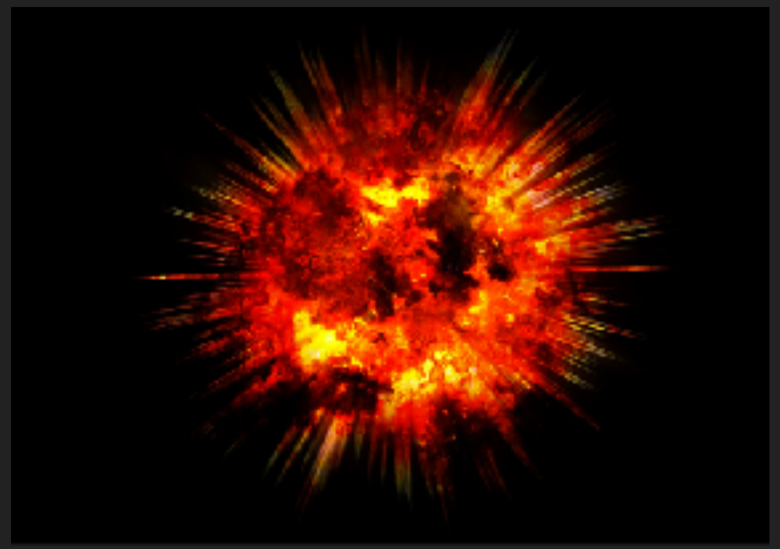
Mark I Perceptron

1969 Marvin Minksy and Seymour Papert

'Perceptrons'

1986 David E. Rumelhart, Geoffrey Hinton & Ronald Williams

'Learning representations by back-propagating errors'

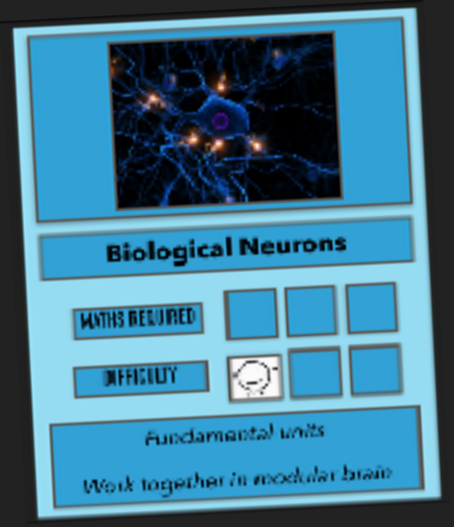


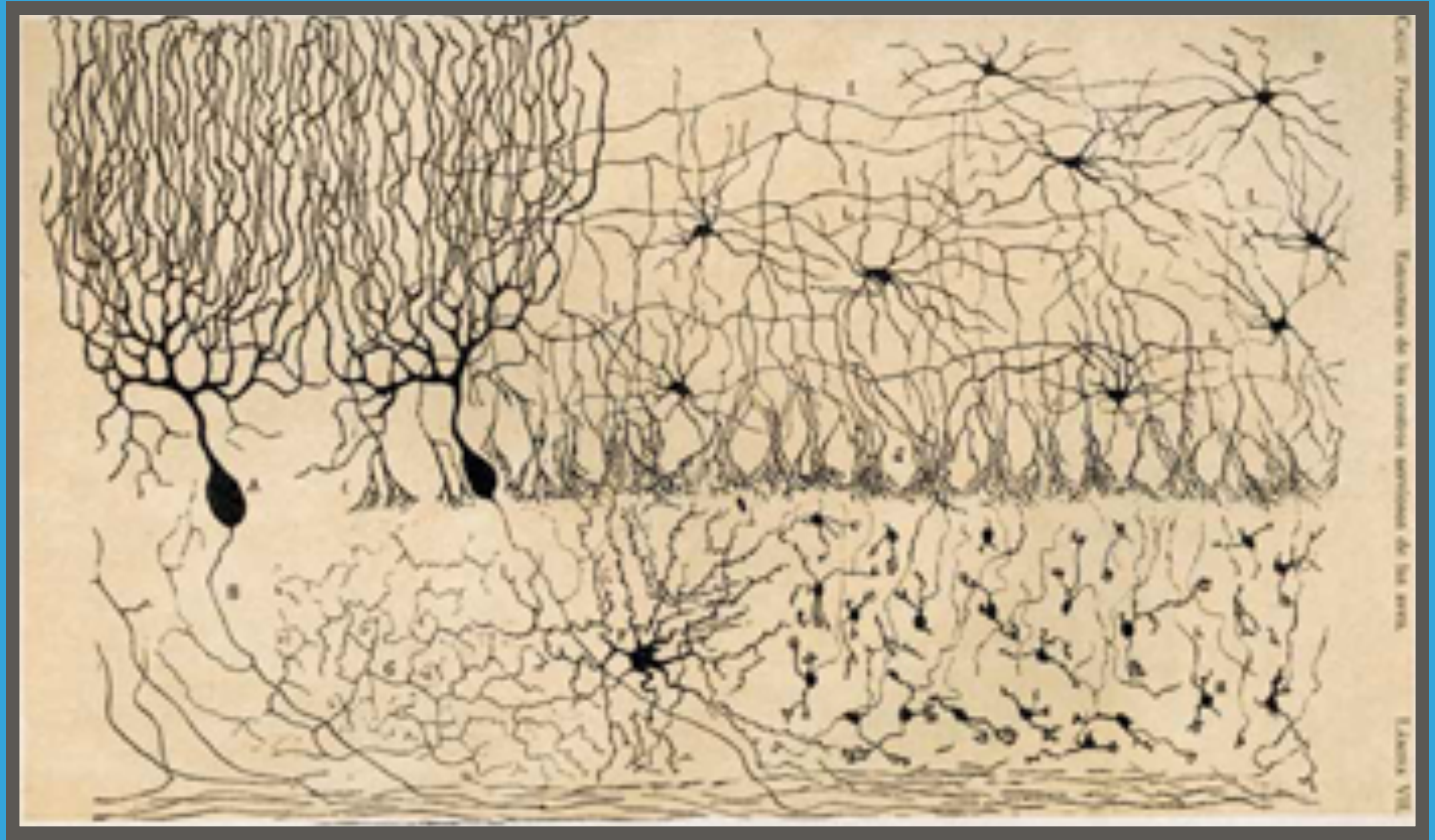
THE NEURON DOCTRINE PROVIDED A STRONG ANALYTICAL APPROACH IN THE PAST, BUT CAN NO LONGER BE SEEN AS CENTRAL TO CONTEMPORARY ADVANCES IN NEUROSCIENCE.

R. W. GUILLERY

Observations of synaptic structures: origins of the neuron doctrine and its current status

Philos Trans R Soc Lond B Biol Sci. 2005 Jun 29; 360(1458): 1281–1307

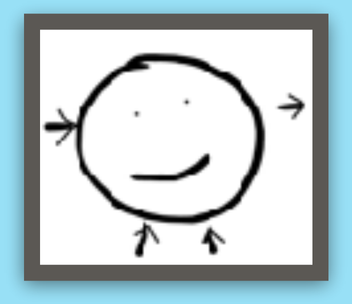

Biological Neurons
MATHS REQUIRED
DIFFICULTY
Fundamental units
Work together in modular brain



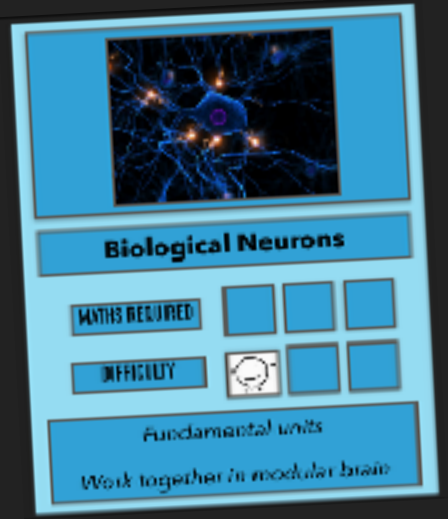
The Neuron Doctrine

MATHS REQUIRED

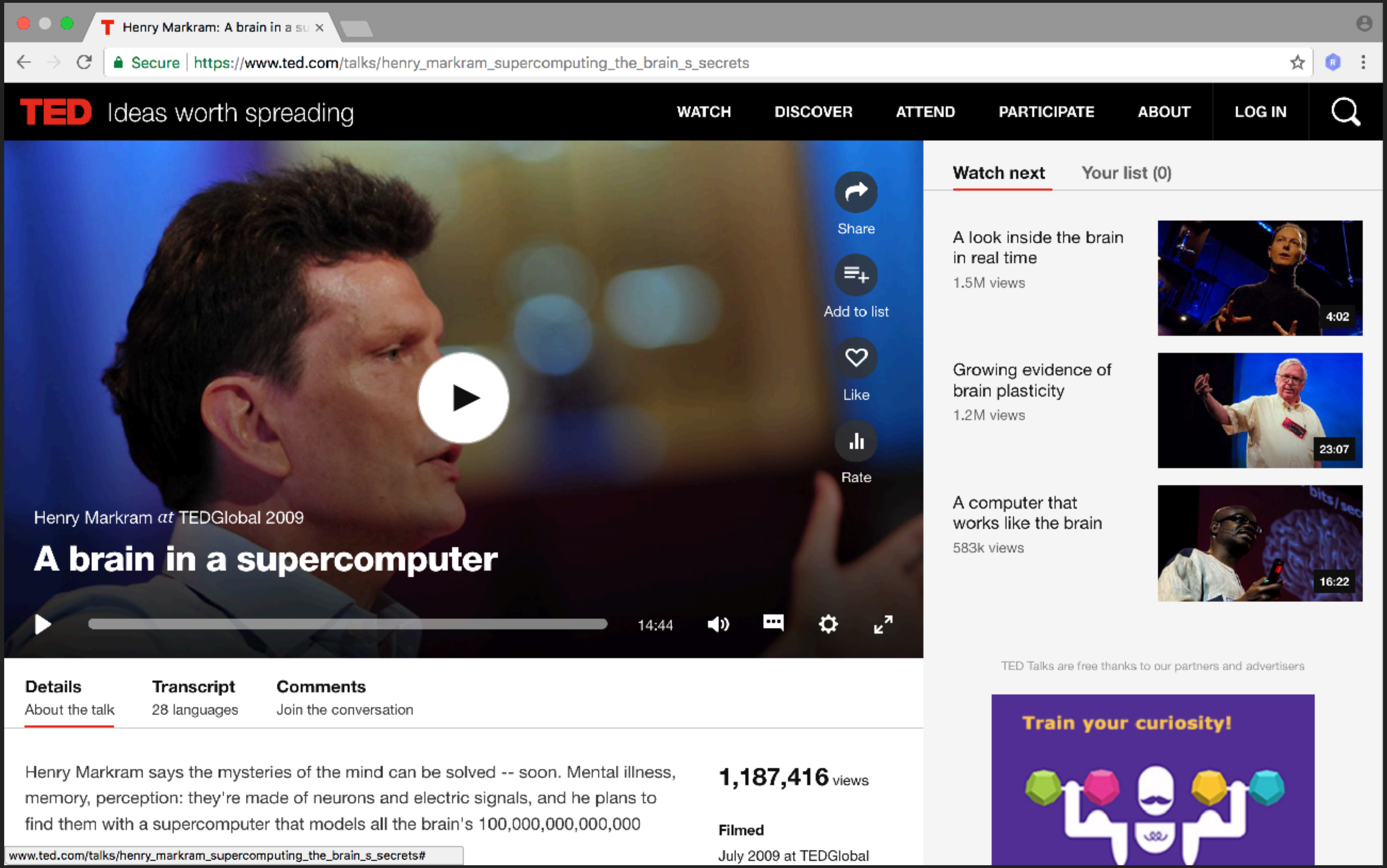
DIFFICULTY



To understand brain, smallest unit is the neuron



WHY IS EVERYONE EXCITED ABOUT NEURONS


Henry Markram: A brain in a su
Secure
https://www.ted.com/talks/henry_markram_supercomputing_the_brain_s_secrets
TED Ideas worth spreading
WATCH
DISCOVER
ATTEND
PARTICIPATE
ABOUT
LOG IN
Share
Add to list
Like
Rate
Henry Markram at TEDGlobal 2009
A brain in a supercomputer
14:44
Watch next
Your list (0)
A look inside the brain in real time
1.5M views
4:02
Growing evidence of brain plasticity
1.2M views
23:07
A computer that works like the brain
583k views
16:22
TED Talks are free thanks to our partners and advertisers
Train your curiosity!
Details
About the talk
Transcript
28 languages
Comments
Join the conversation
Henry Markram says the mysteries of the mind can be solved -- soon. Mental illness, memory, perception: they're made of neurons and electric signals, and he plans to find them with a supercomputer that models all the brain's 100,000,000,000,000
1,187,416 views
Filmed
July 2009 at TEDGlobal
www.ted.com/talks/henry_markram_supercomputing_the_brain_s_secrets#

WHAT IS THE ANSWER TO THIS QUESTION?

WHAT KIND OF ANIMAL IS THIS?

BIOLOGICAL TO ARTIFICIAL

PART I: NEURONS

- **Neurons**
  - **Biological**
  - **Artificial**
    - Linear Neuron
    - Binary Threshold Neuron
    - Logistic/sigmoid Neuron
    - reLU
    - Tanh

PART II: LEARNING

- Network
  - Feed forward — Cost functions
  - Back propagation — Training
    - Gradient descent
    - Learning rate
    - Regularisation

PART III: IN A NETWORK

- Learning to map input to output — Architectures
  - CNN
  - RNN
  - And beyond…

LINEAR NEURONS

Neurons

Biological

Artificial

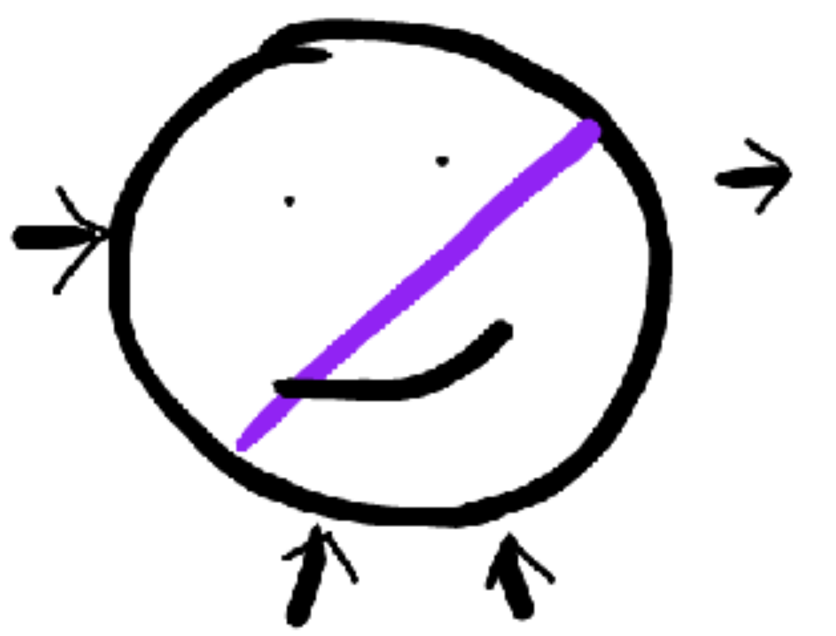
Linear Neuron

Binary Threshold Neuron

Logistic/sigmoid Neuron

reLU

Tanh



Network

Feed forward — Cost functions

Back propagation — Training — Gradient descent

Learning rate

Regularisation

Learning to map input to output — Architectures — CNN

RNN

And beyond…

$$x \xrightarrow{w} (x \times w) + b$$

$$b \rightarrow (x \times w) + b$$

WHAT KIND OF FUNCTION IS THIS? (EXPONENTIAL, QUADRATIC..?)

$x \quad 2$

$$(x \times 2) + 2$$

2

$x \quad w$

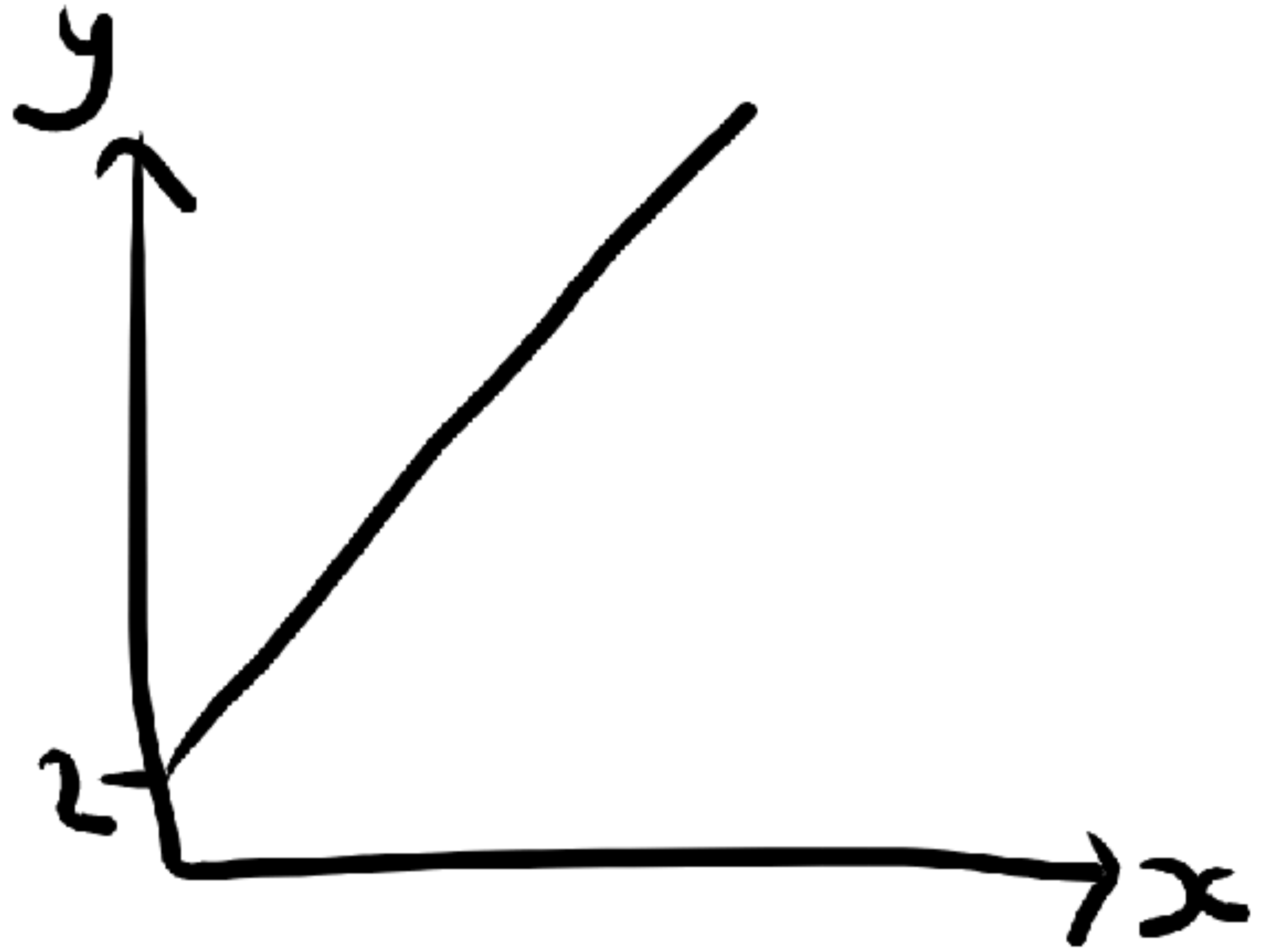
$$(x \times w) + b$$

b

$x \quad 2$

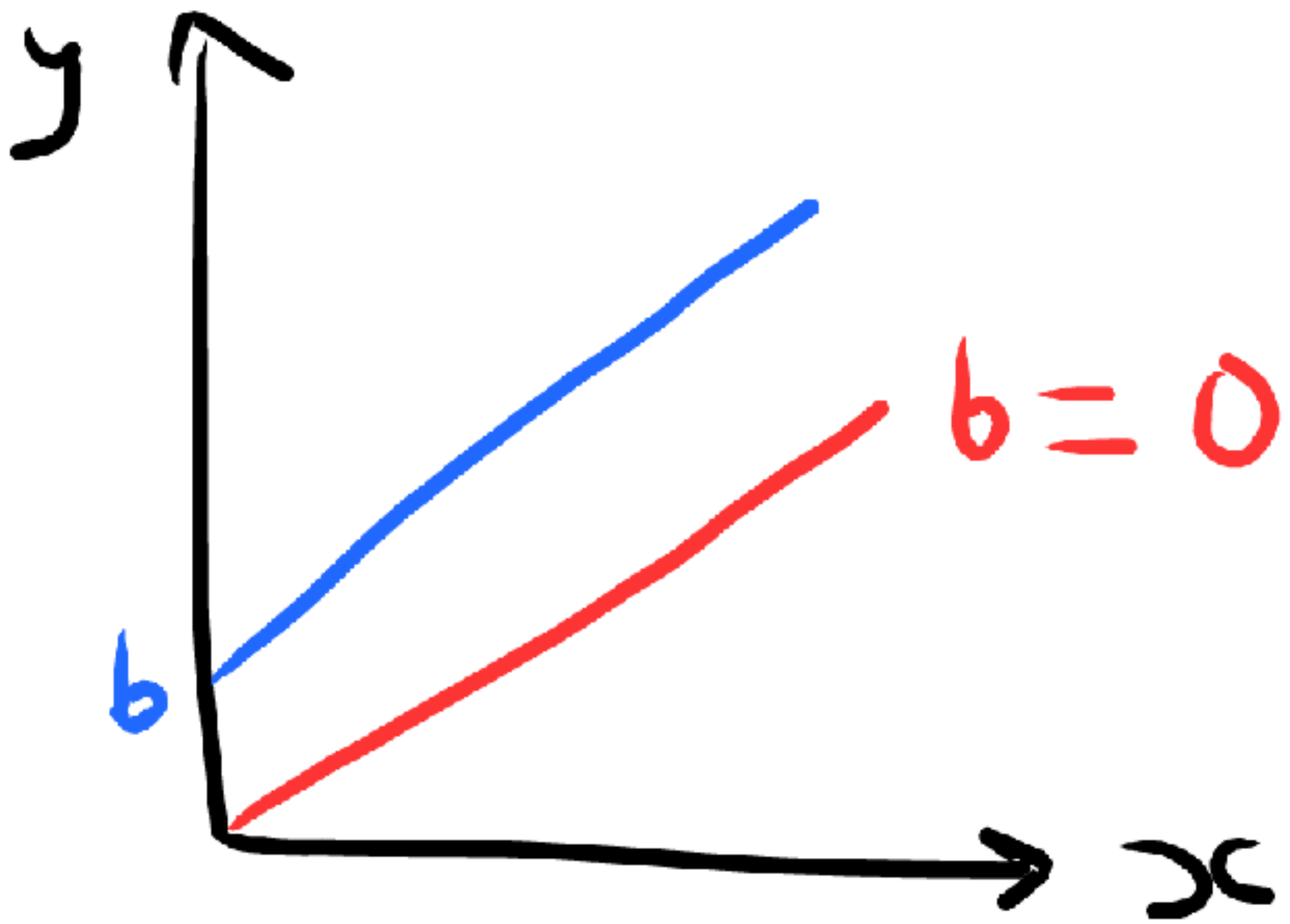
$$(x \times 2) + 2$$

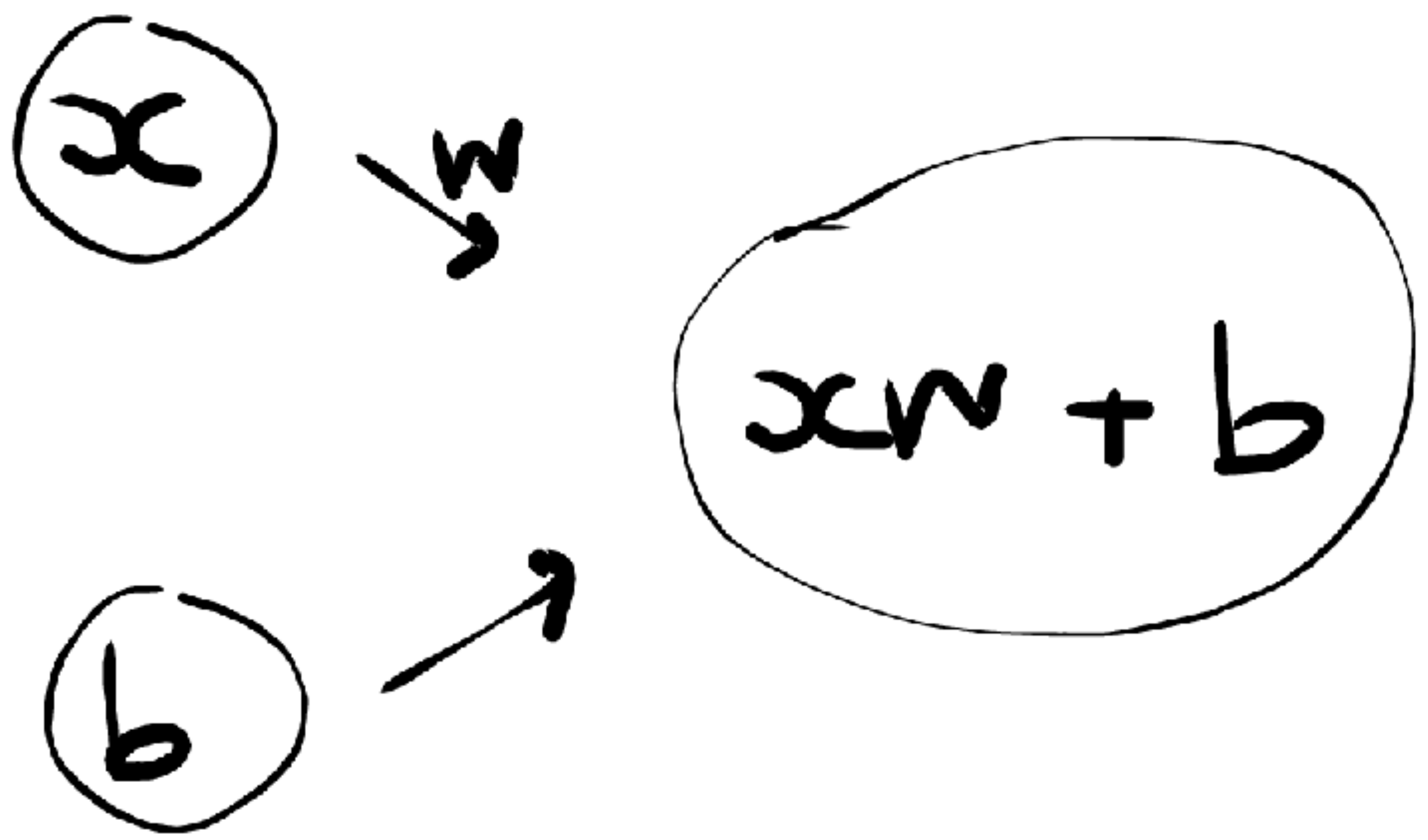
2

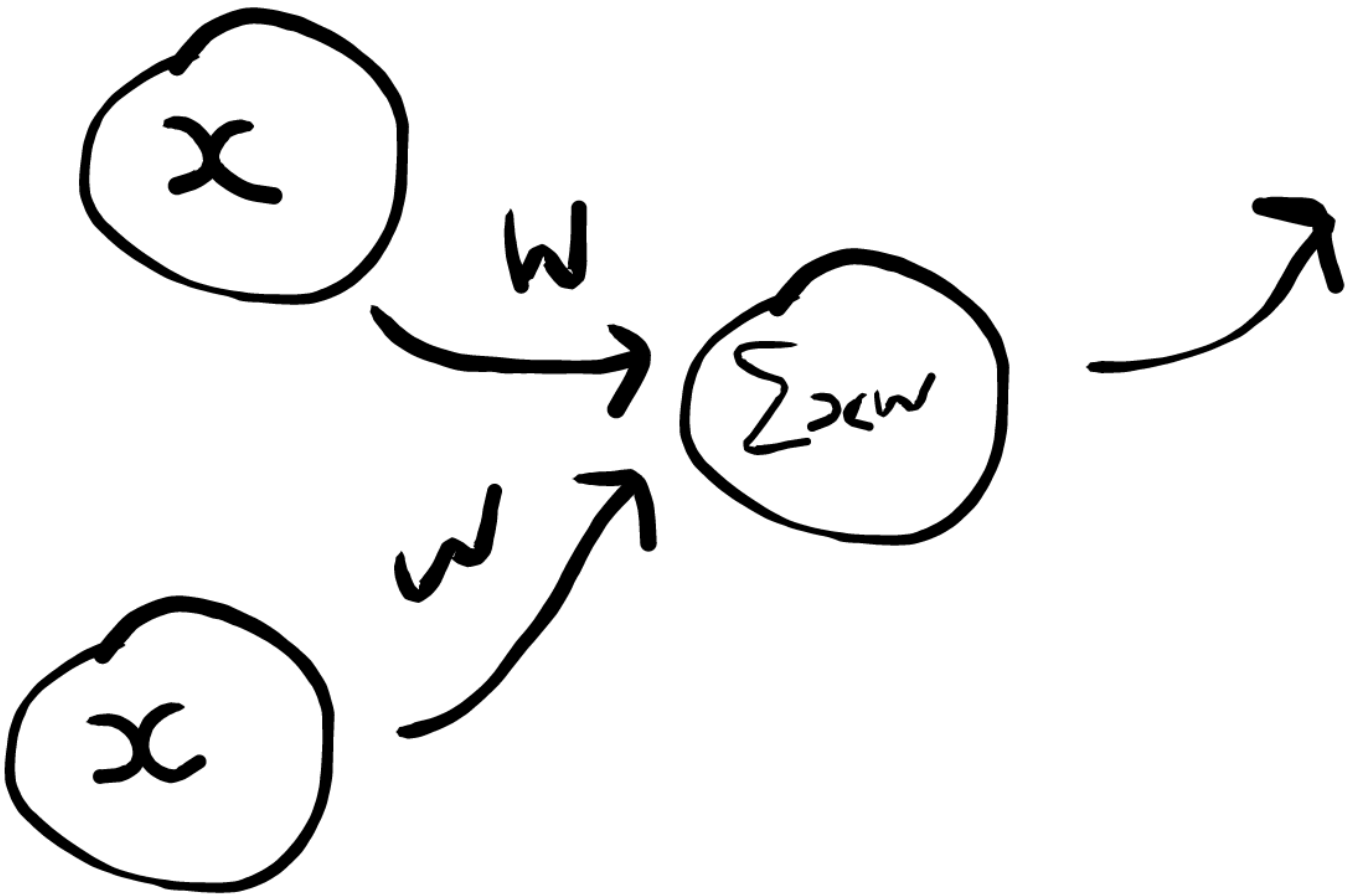


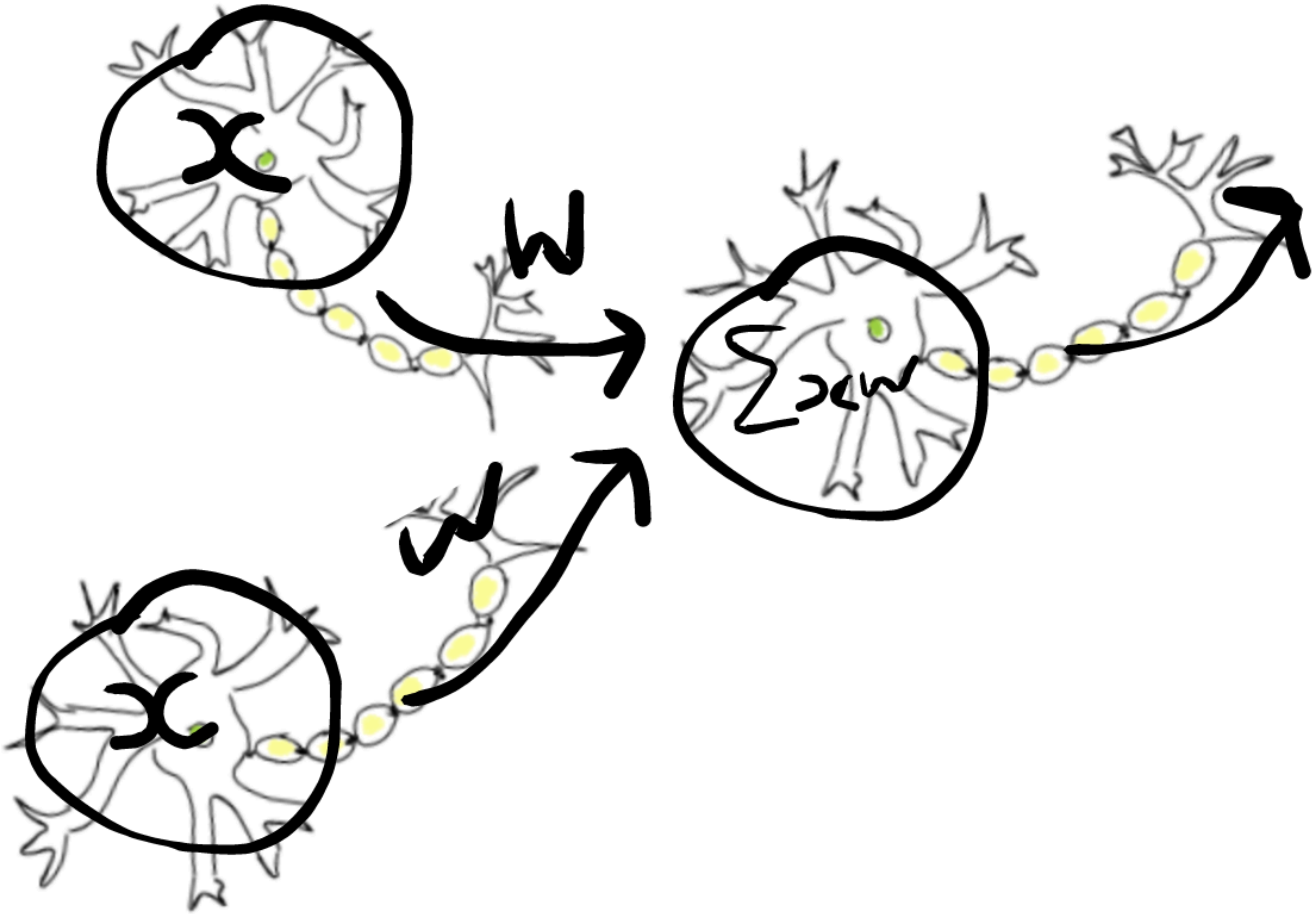
y
2
x

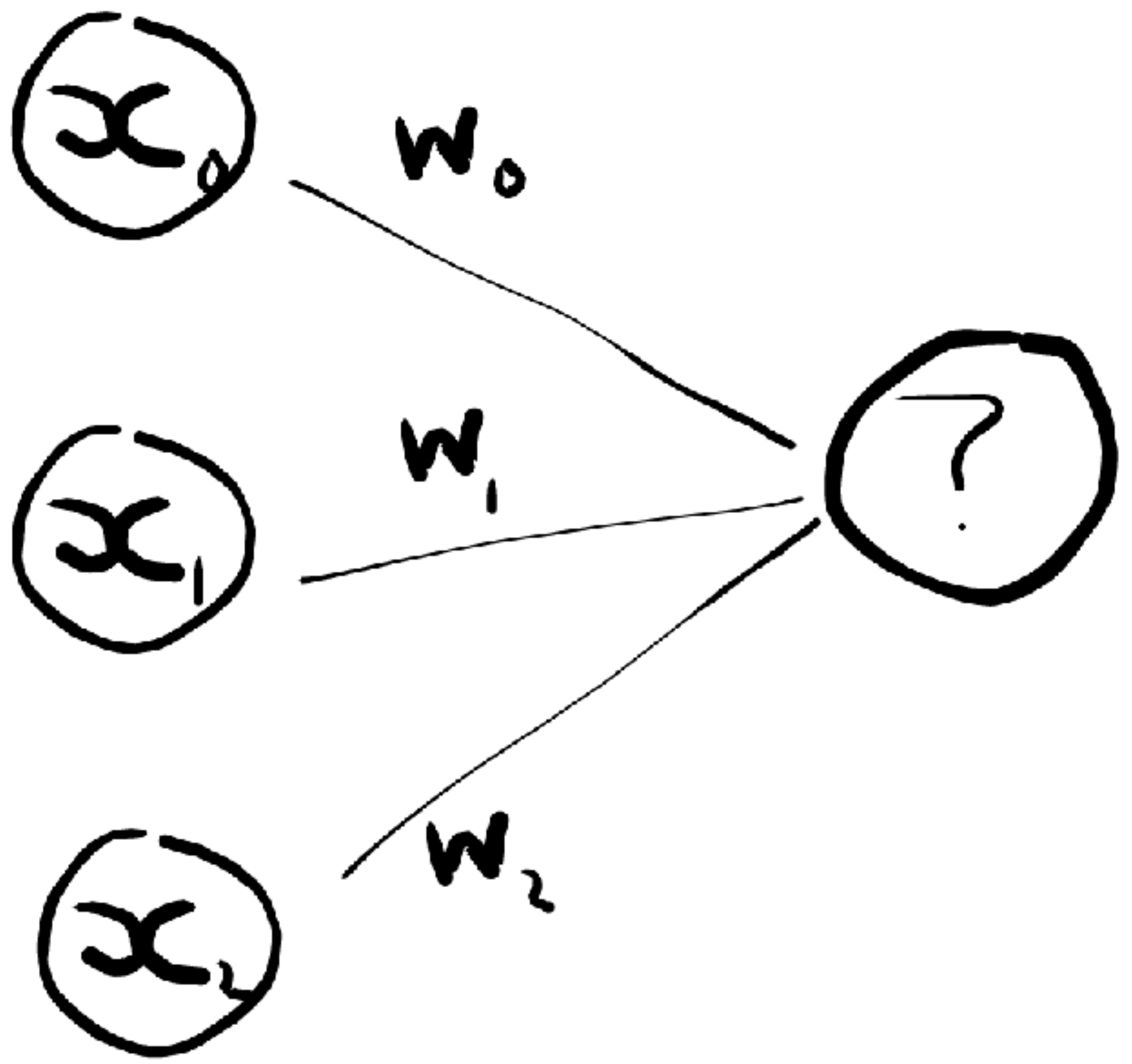
BIAS?



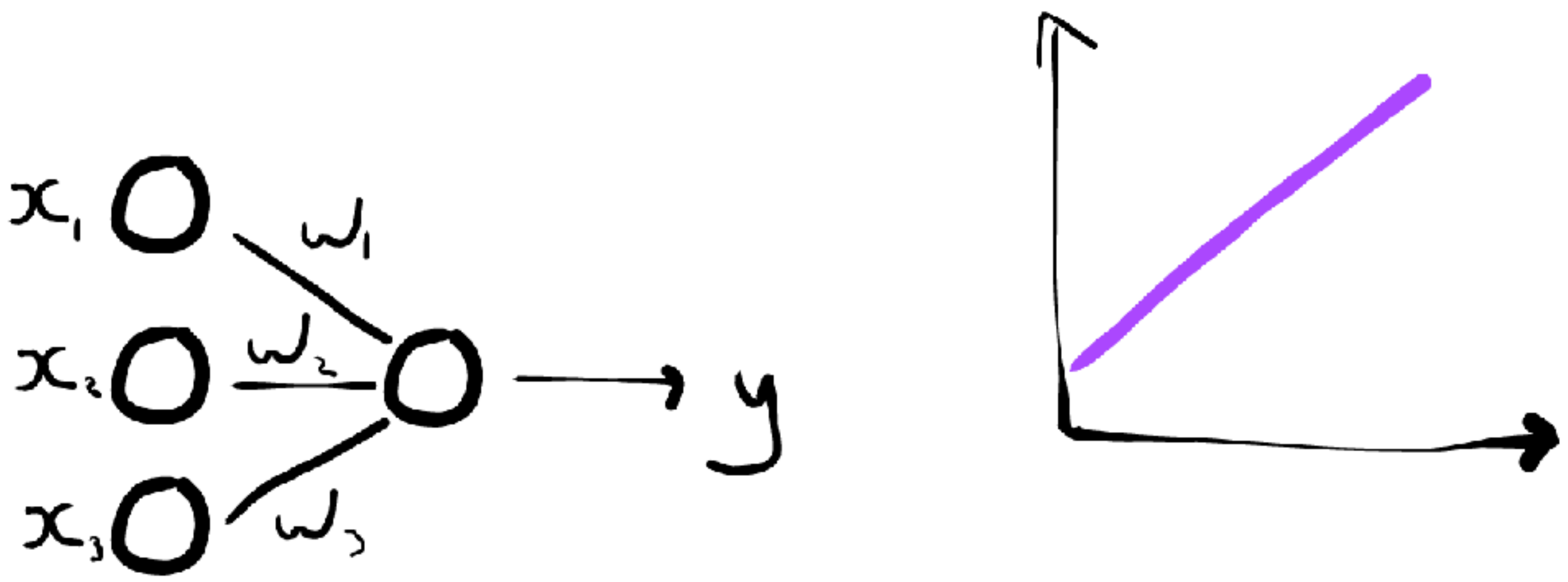

x
w
b
xw + b


x
w
x
w
∑xw


x
w
w
x
Σxw



x0
x1
x2
w0
w1
w2
?



$$y = x_1 w_1 + x_2 w_2 + x_3 w_3$$

$$\sum_{i=0}^{n} x_i \theta_i$$

$$\sum_{i=0}^{n} x_i \theta_i$$

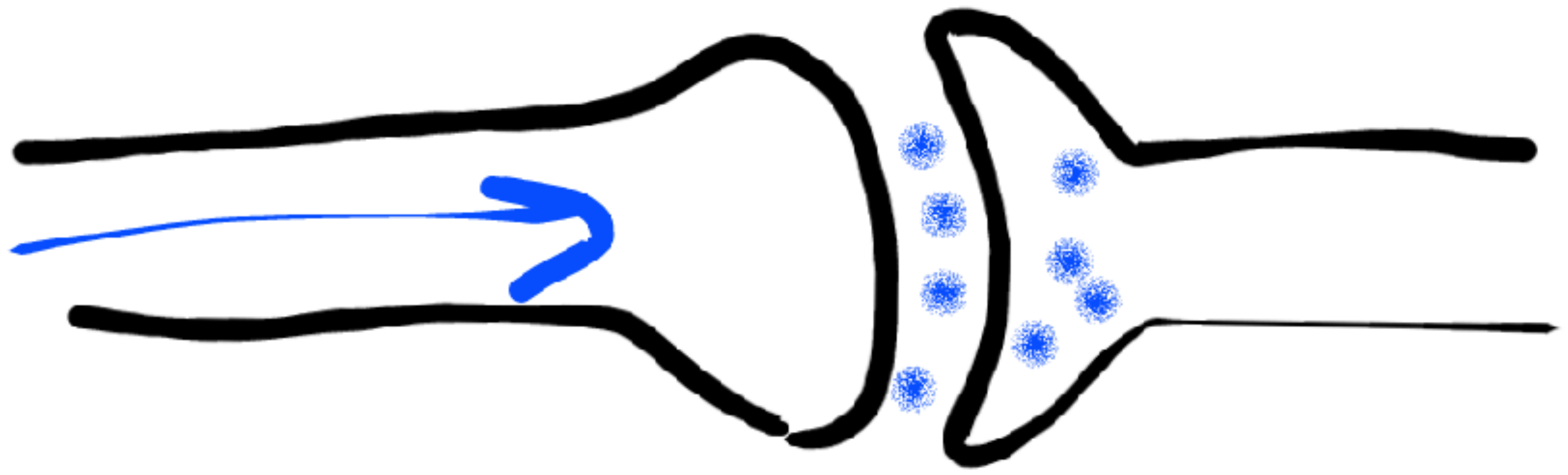
$$\sum_{i=0}^{n} x_i w_i$$

$$\sum_{i=0}^{n} x_i \theta_i$$

$$\sum_{i=0}^{n} x_i w_i$$

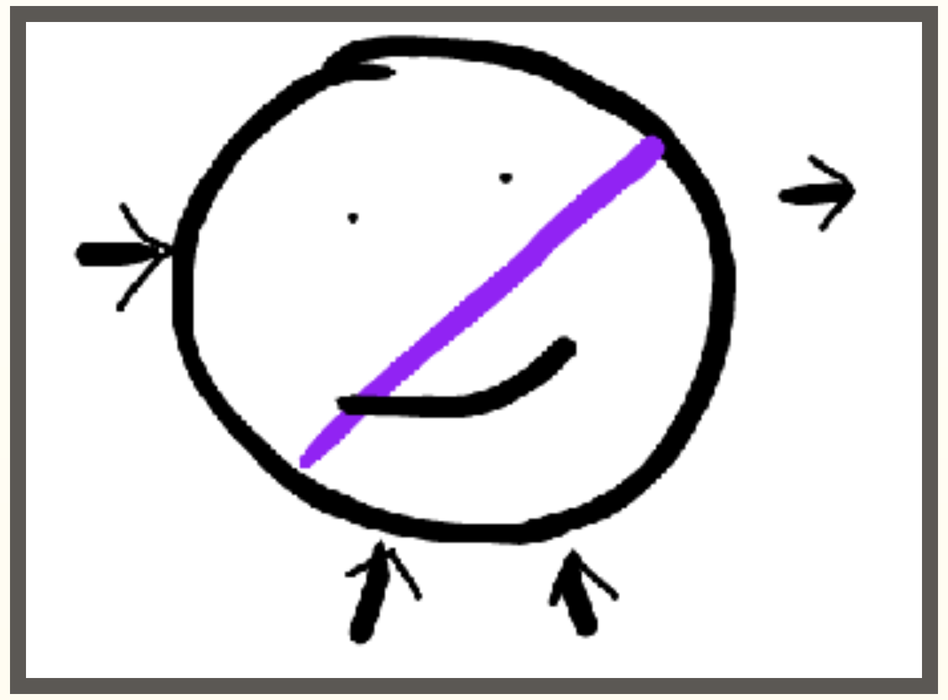
$$\sum_{i=1}^{n} x_i w_i + b$$

WEIGHTS ADAPT



- Weight is large (3x): bigger input
- Weight is small (0.0003x): smaller input

What can I do?

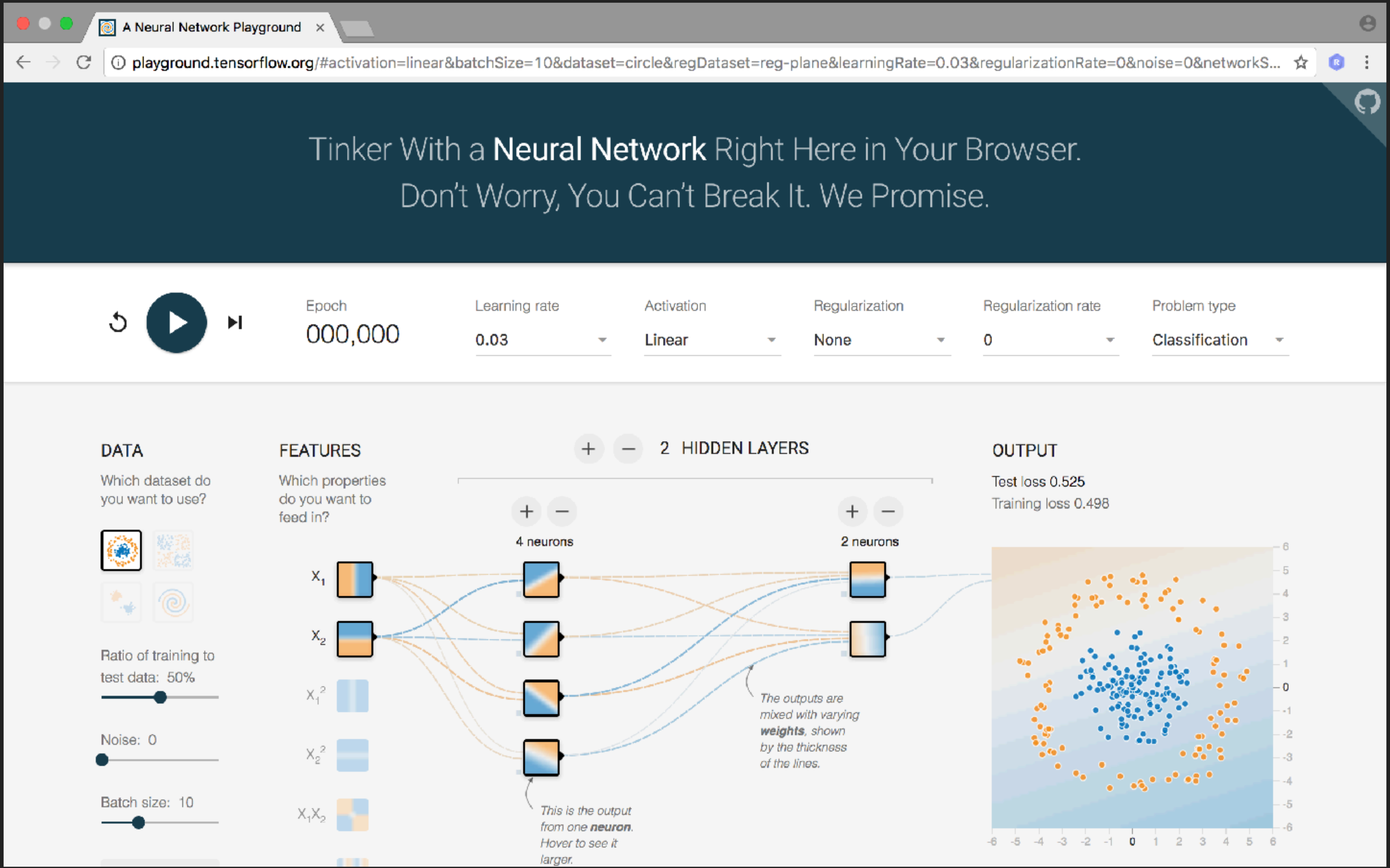


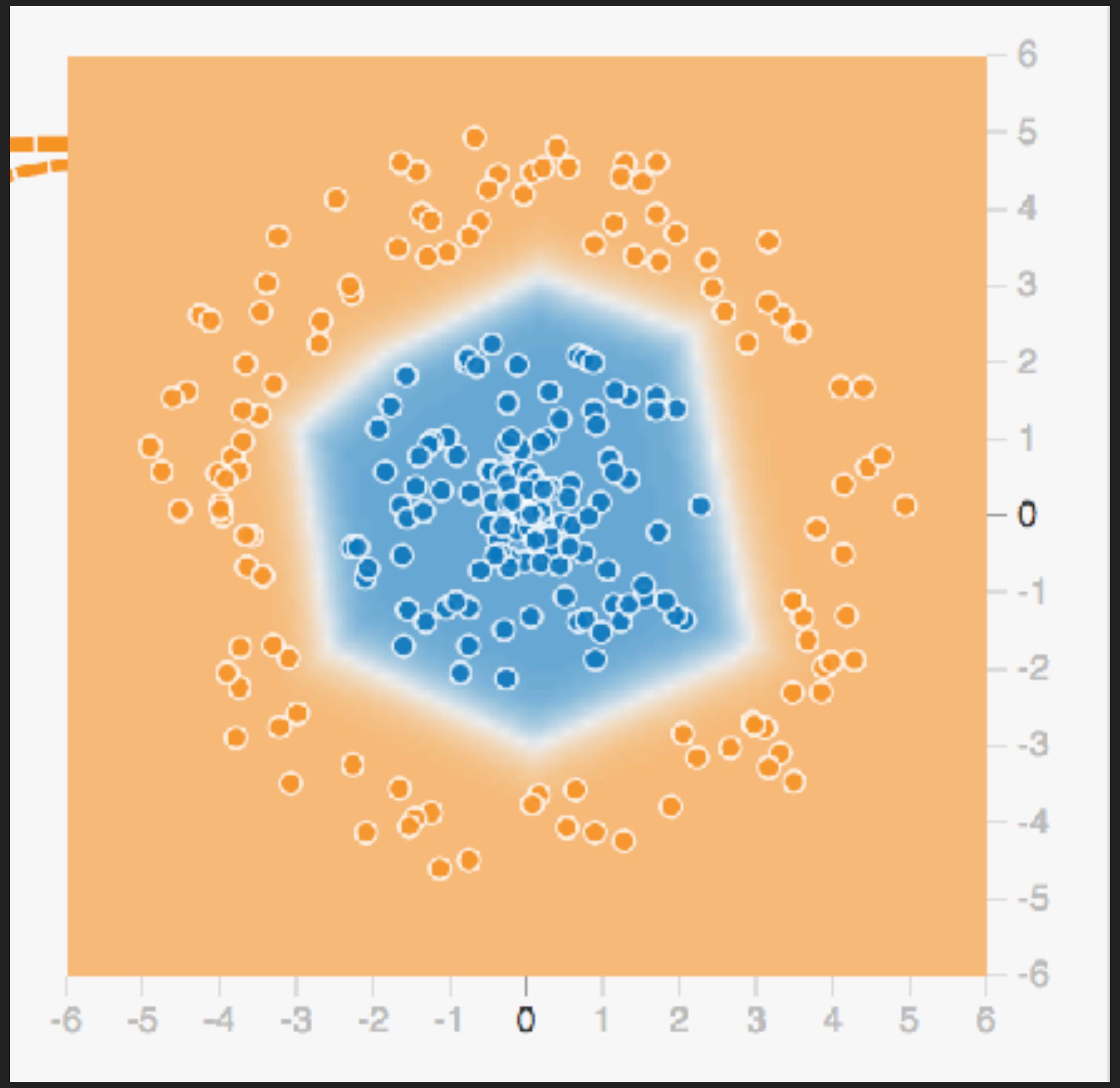
WHAT SHOULD THE Y-INTERCEPT BE?

WHAT SHOULD THE GRADIENT BE?

LINEAR NEURONS

- Networks of linear neurons give a linear output
- Sometimes called the **identity activation function**
- Go to http://playground.tensorflow.org/
- Select the first data set
- Make the 'activation' linear
- Can you separate the data?


A Neural Network Playground
playground.tensorflow.org/#activation=linear&batchSize=10&dataset=circle&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networkS...
Tinker With a Neural Network Right Here in Your Browser.
Don't Worry, You Can't Break It. We Promise.
Epoch
000,000
Learning rate
0.03
Activation
Linear
Regularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
FEATURES
Which properties do you want to feed in?
2 HIDDEN LAYERS
4 neurons
2 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 lines.
OUTPUT
Test loss 0.525
Training loss 0.498



6
5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6

Biological Neurons

MATHS REQUIRED

DIFFICULTY

Fundamental units

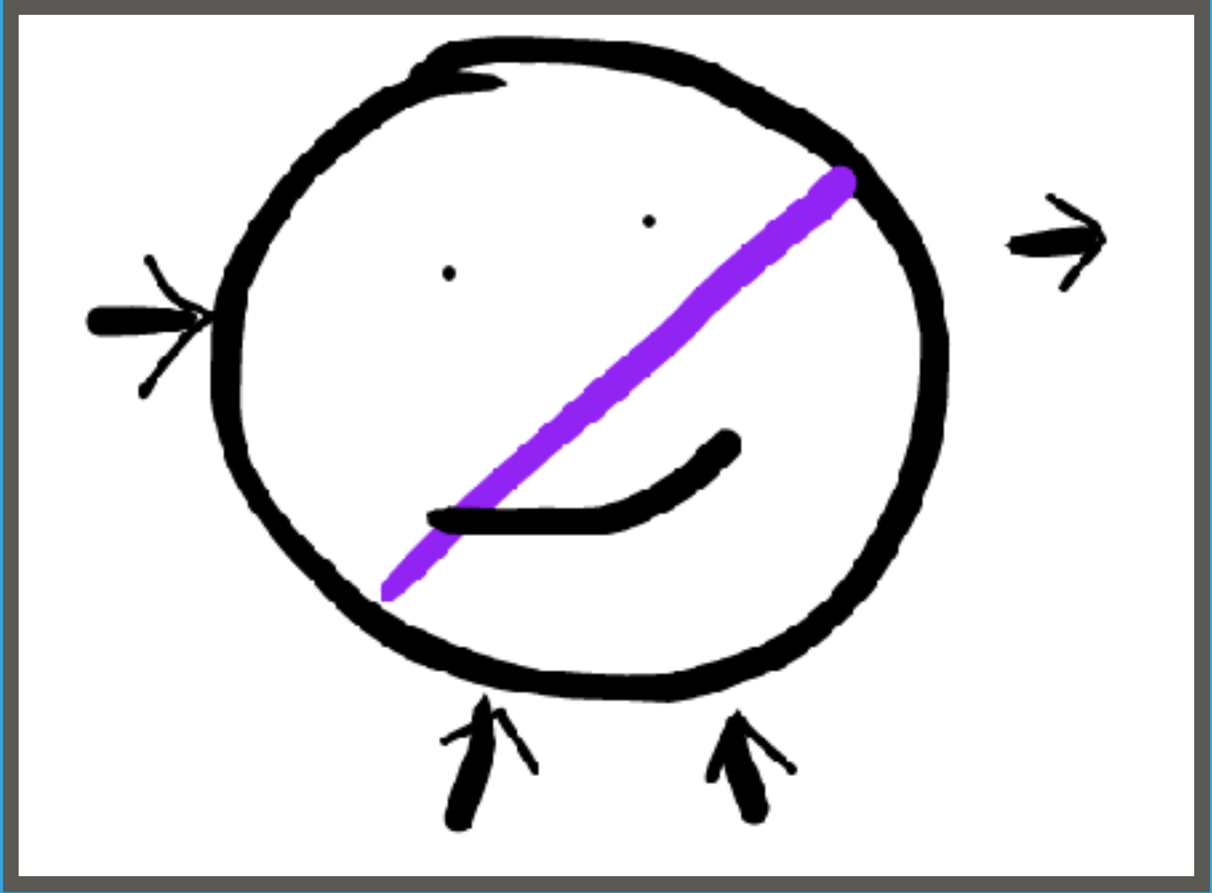
Work together in modular brain

The Neuron Doctrine

MATHS REQUIRED

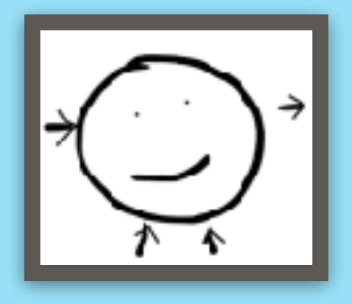
DIFFICULTY

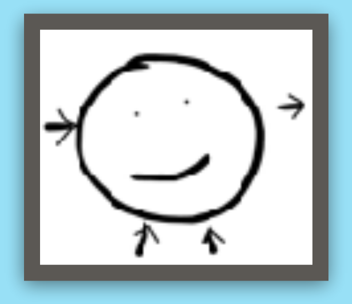
To understand brain, smallest unit is the neuron



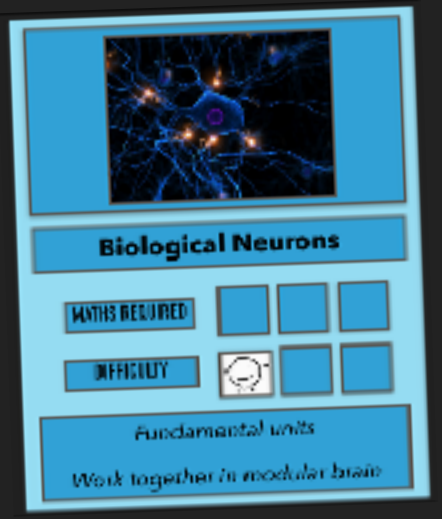
$$\sum_{i=0}^{n} x_i w_i$$

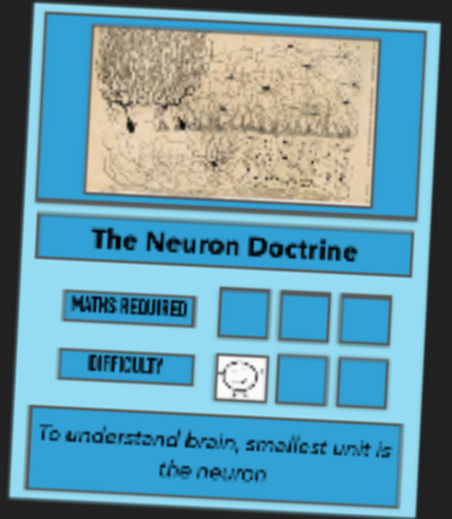
Linear Neurons

MATHS REQUIRED 

DIFFICULTY 

*Output is the sum of the inputs * weights, which gives a linear output*





BINARY THRESHOLD NEURONS

PART I:
NEURONS

Neurons

Biological

Artificial

Linear Neuron

Binary Threshold Neuron

Logistic/sigmoid Neuron

reLU

Tanh

PART II:
LEARNING

Network

Feed forward — Cost functions

Back propagation — Training — Gradient descent, Learning rate, Regularisation

PART III:
IN A NETWORK

Learning to map input to output — Architectures — CNN, RNN, And beyond…

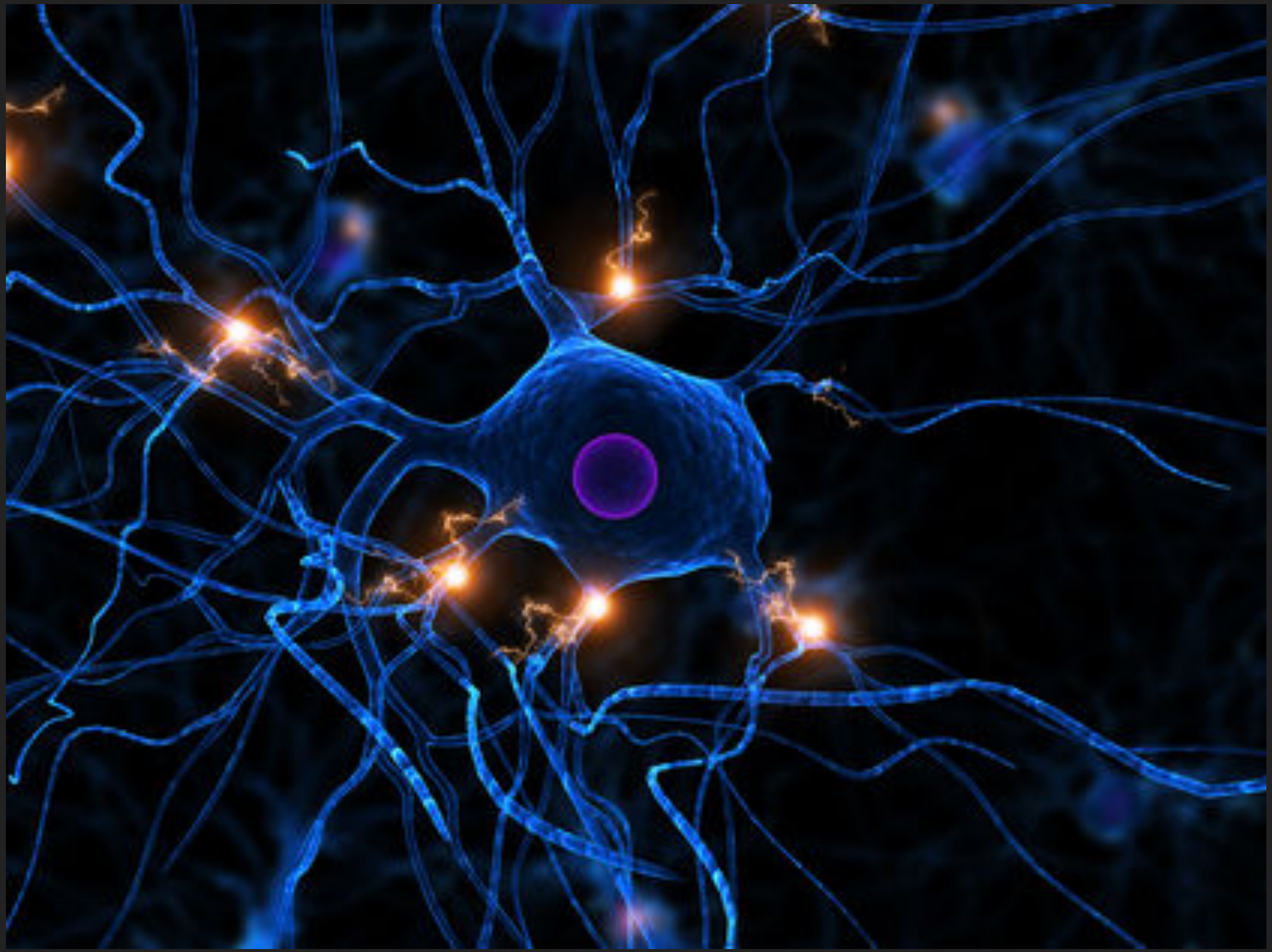
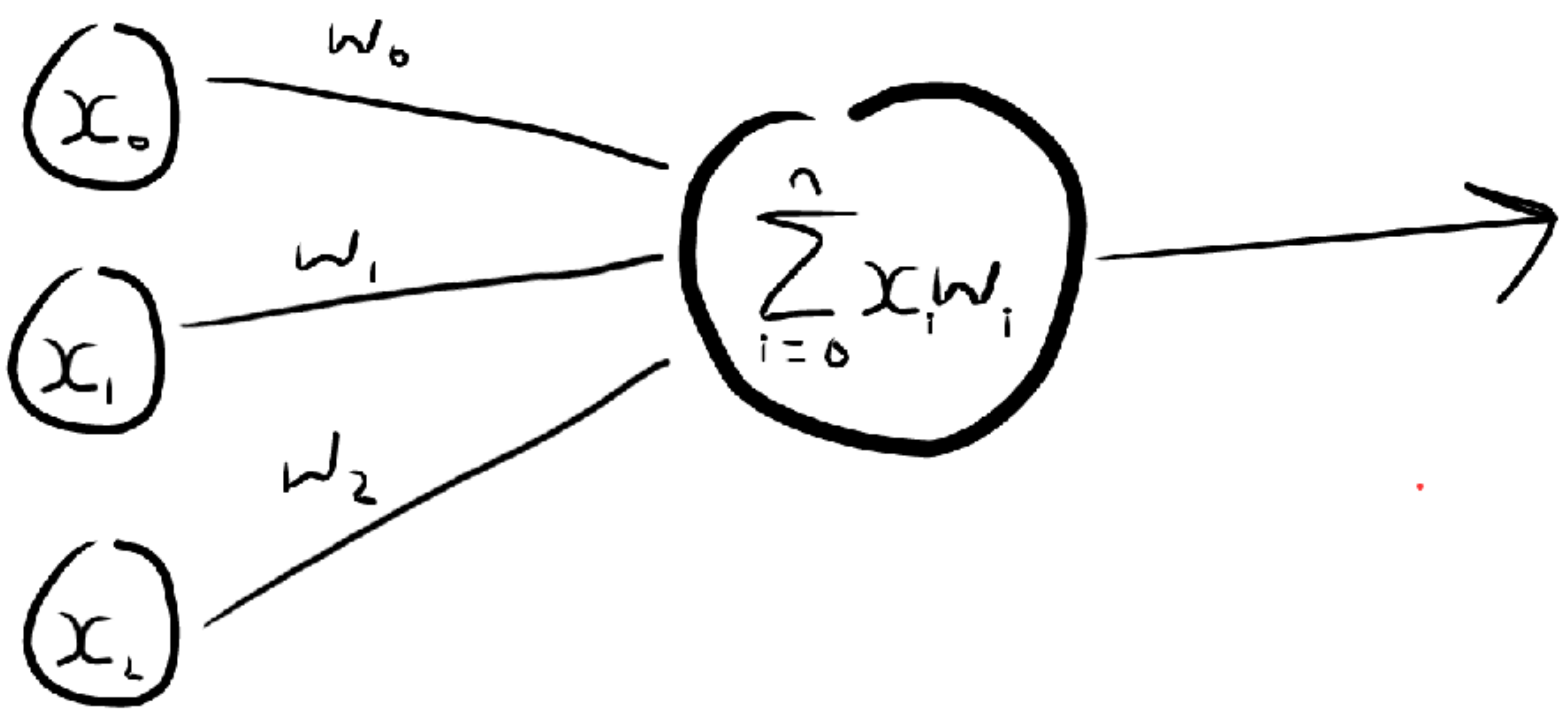
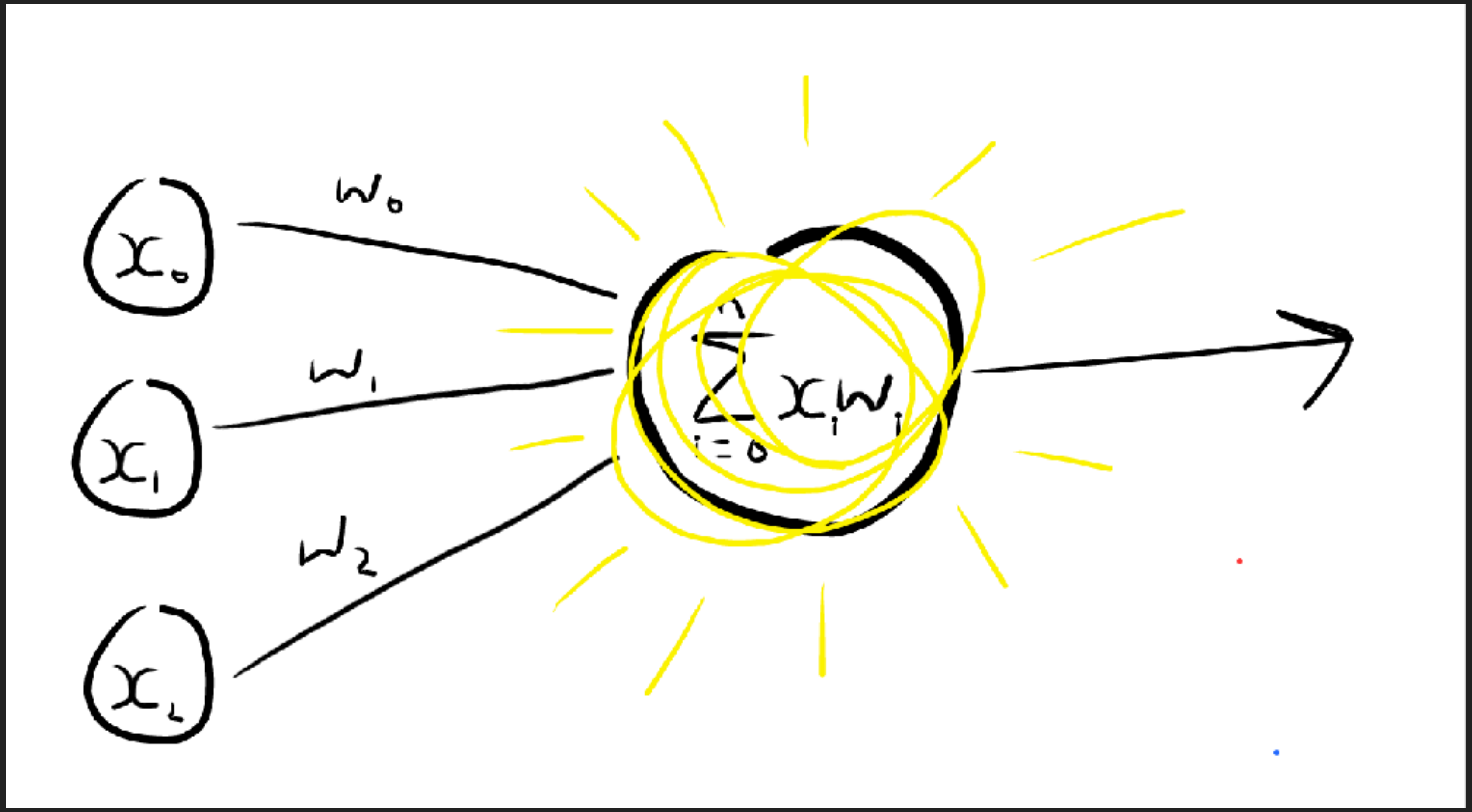
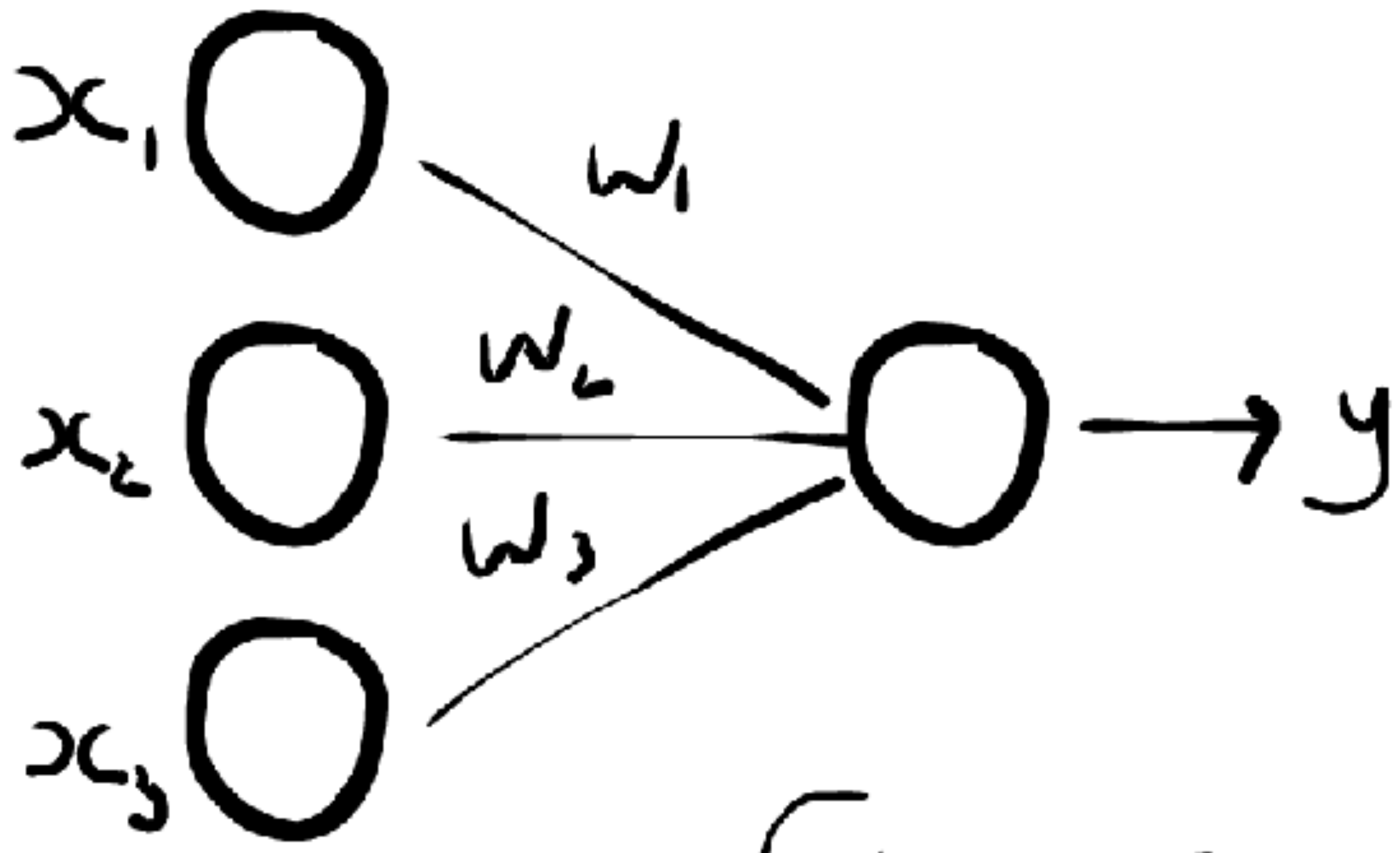
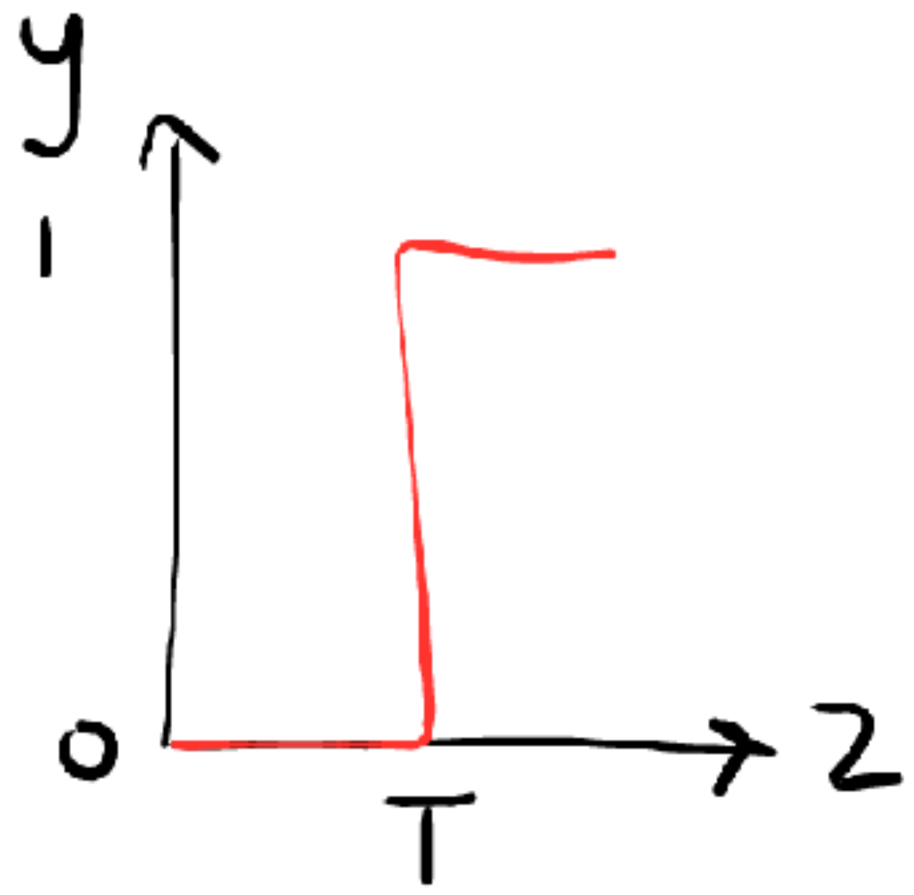


IMAGE CREDIT: SEBASTIAN KAULITZKI


x_0
x_1
x_2
w_0
w_1
w_2
$\sum_{i=0}^{n} x_i w_i$


x_0
x_1
x_2
w_0
w_1
w_2
$\sum_{i=0}^{n} x_i w_i$

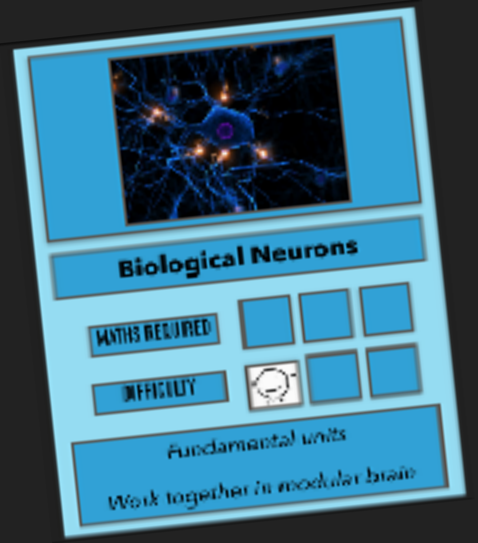


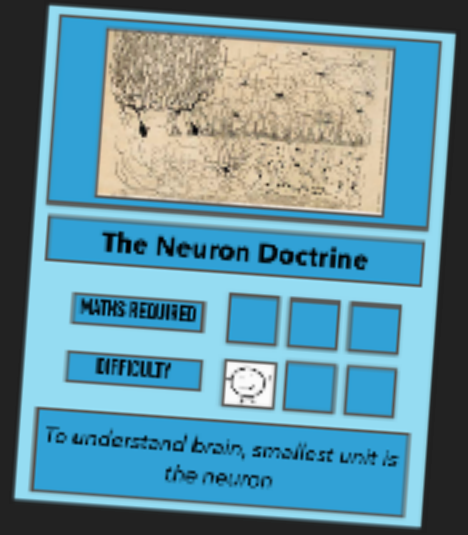


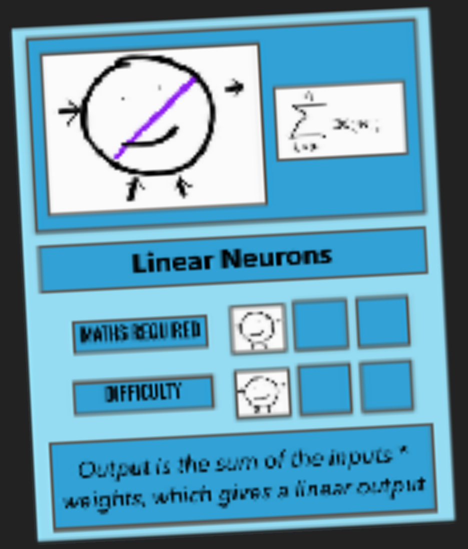
$$y = \begin{cases} 1 & \text{if } x_1w_1 + x_2w_2 + x_3w_3 > T \\ 0 & \text{if } x_1w_1 + x_2w_2 + x_3w_3 \leq T \end{cases}$$

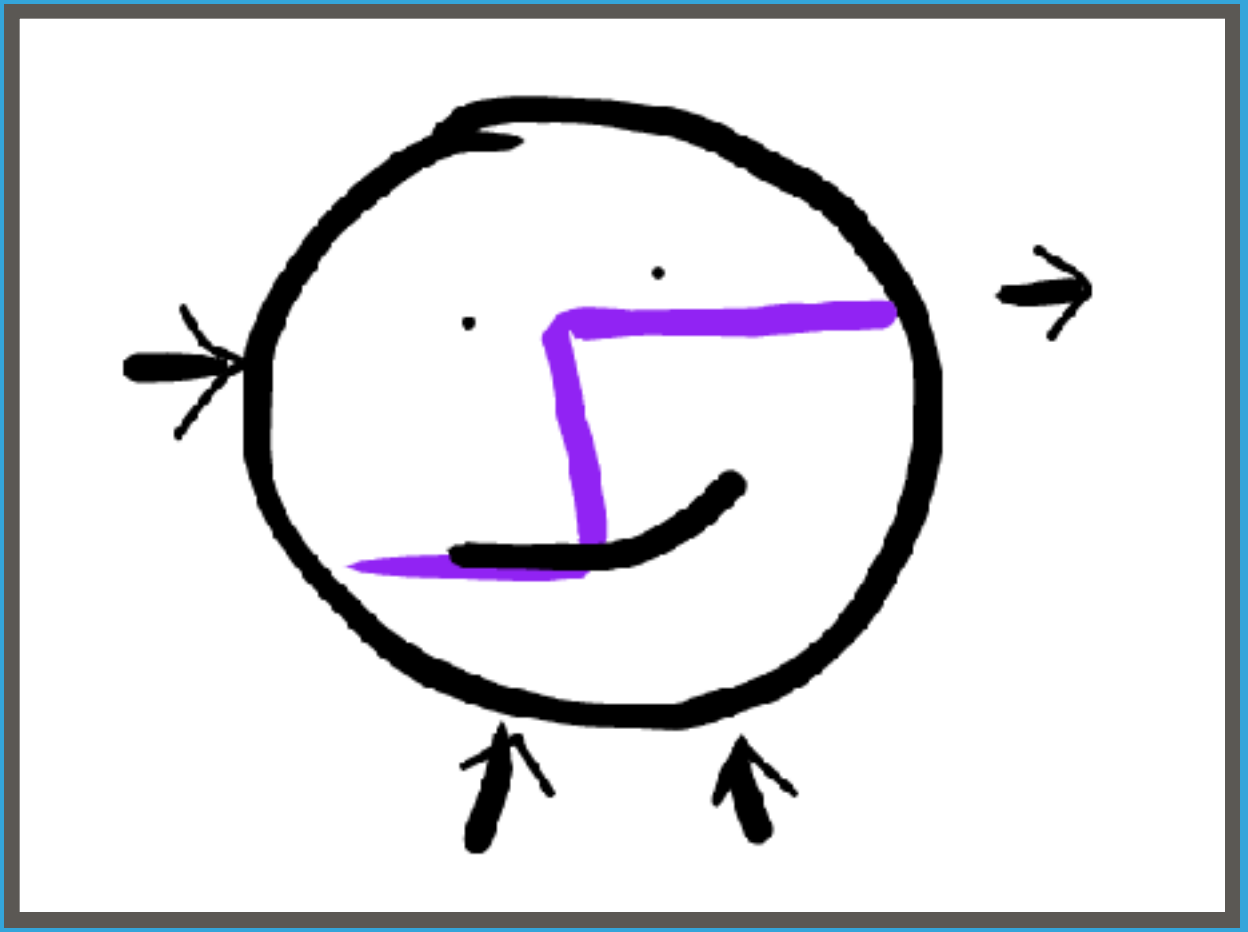
BINARY THRESHOLD NEURONS

- Output is either 0 and 1
- Similar limitation to linear.
- Can you solve XOR with binary threshold neurons?
- Why not?

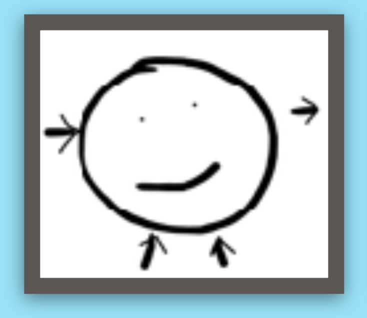

Biological Neurons
MATHS REQUIRED
DIFFICULTY
Fundamental units
Work together in modular brain

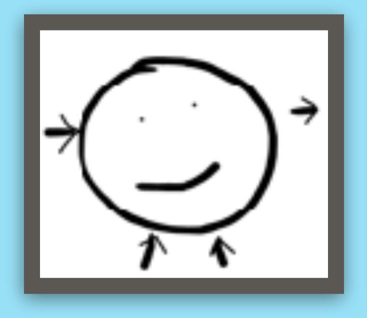

The Neuron Doctrine
MATHS REQUIRED
DIFFICULTY
To understand brain, smallest unit is the neuron


Linear Neurons
MATHS REQUIRED
DIFFICULTY
Output is the sum of the inputs * weights, which gives a linear output

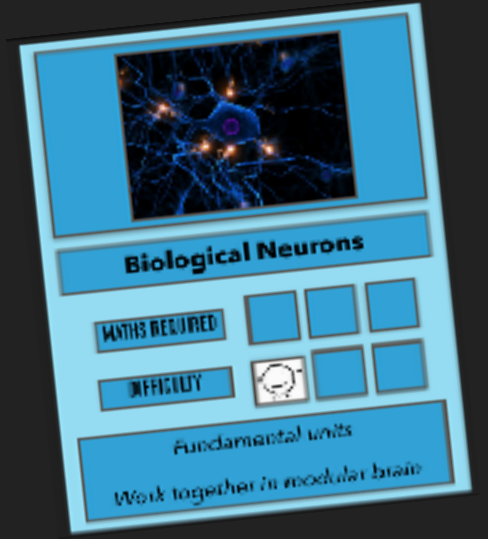


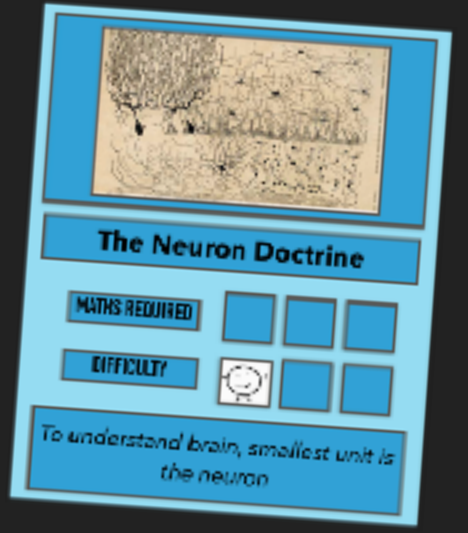
Binary Threshold Neurons

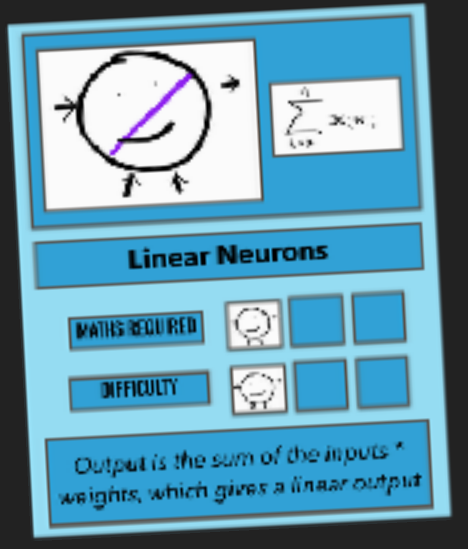
MATHS REQUIRED 

DIFFICULTY 

Output 1 if input is over a threshold, or 0 otherwise







LOGISTIC NEURONS

PART I:
NEURONS

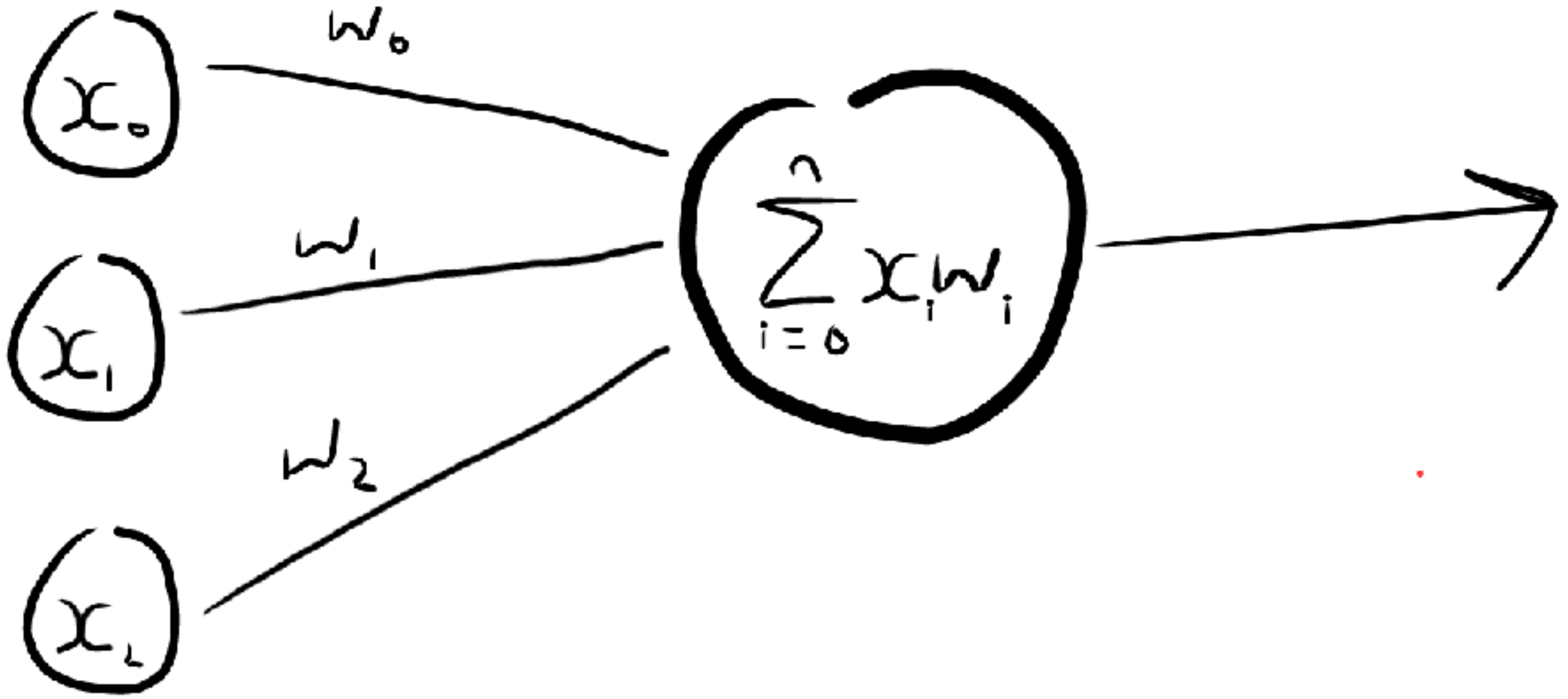
Neurons
- Biological
- Artificial — Linear Neuron; Binary Threshold Neuron; Logistic/sigmoid Neuron; reLU; Tanh

PART II:
LEARNING

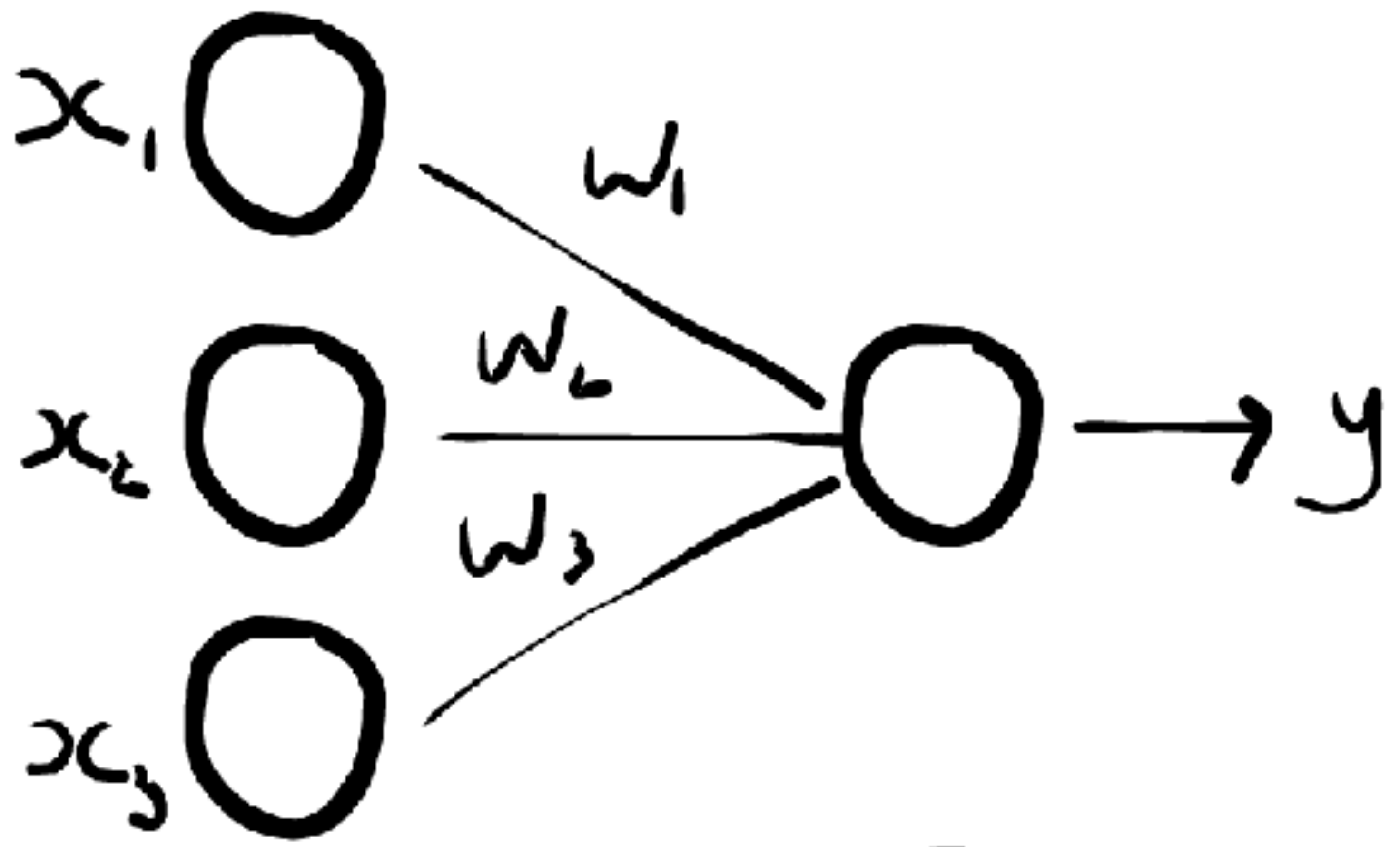
Network
- Feed forward — Cost functions
- Back propagation — Training — Gradient descent; Learning rate; Regularisation

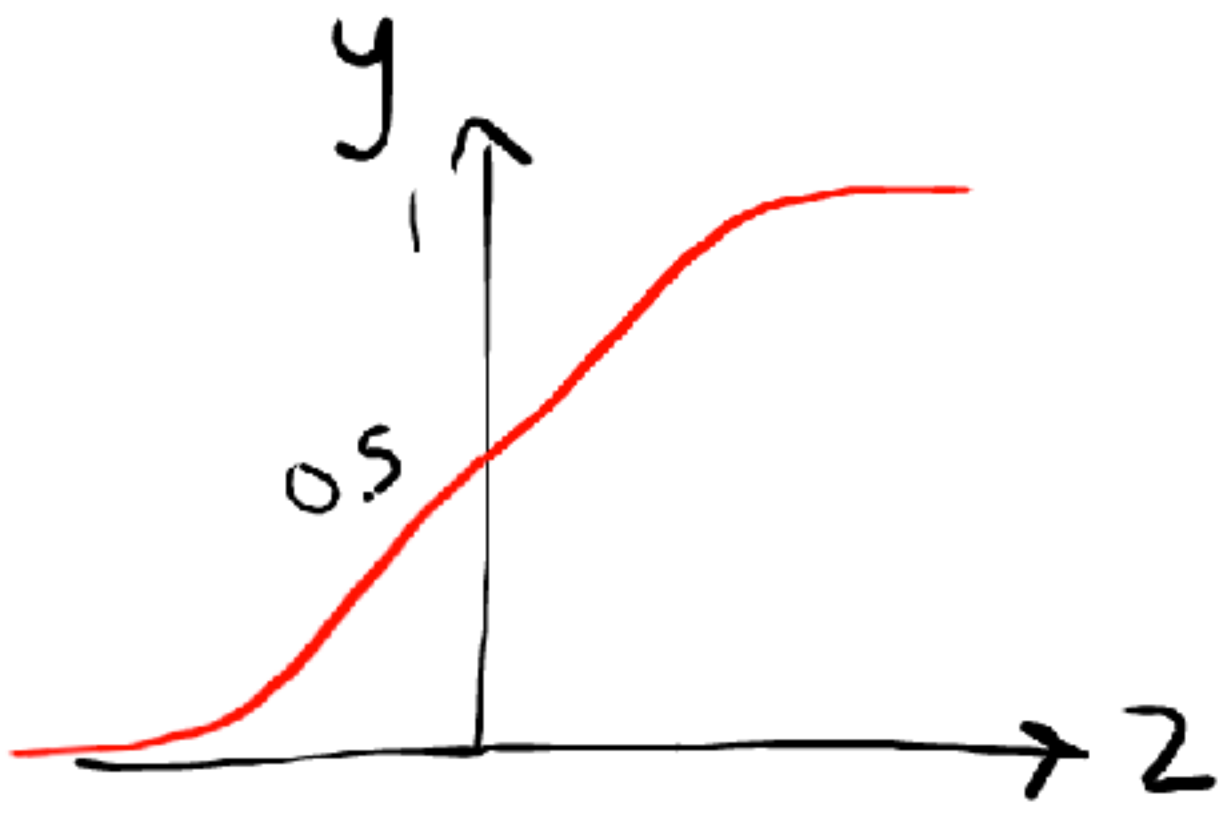
PART III:
IN A NETWORK

Learning to map input to output — Architectures — CNN; RNN; And beyond…



x_0
x_1
x_2
w_0
w_1
w_2
$\sum_{i=0}^{n} x_i w_i$

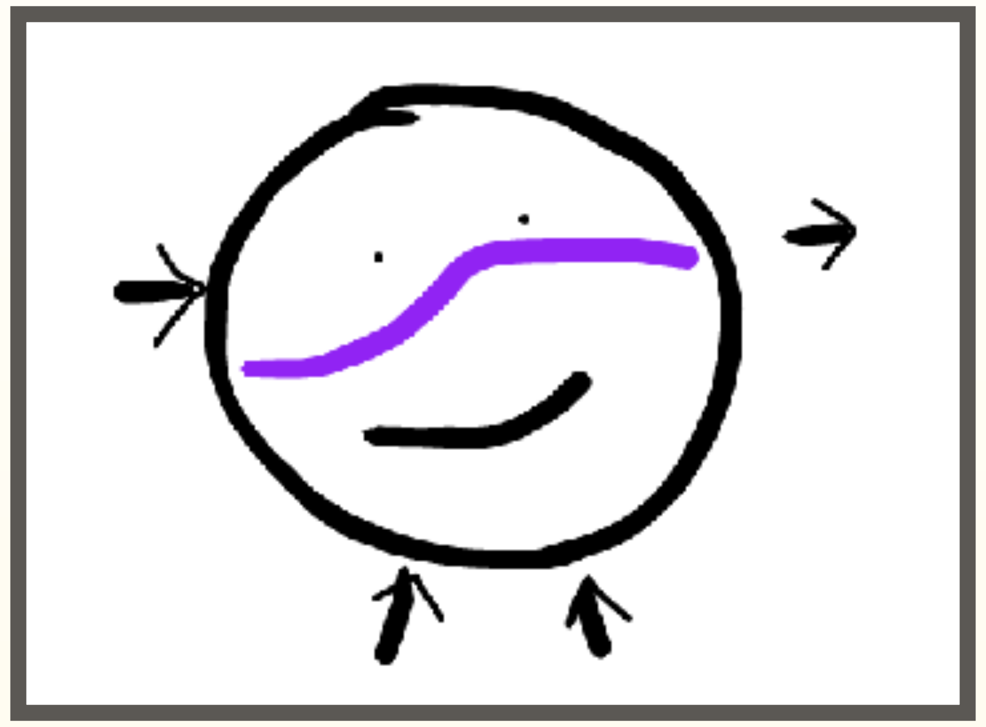




$$z = x_1 w_1 + x_2 w_2 + x_3 w_3$$

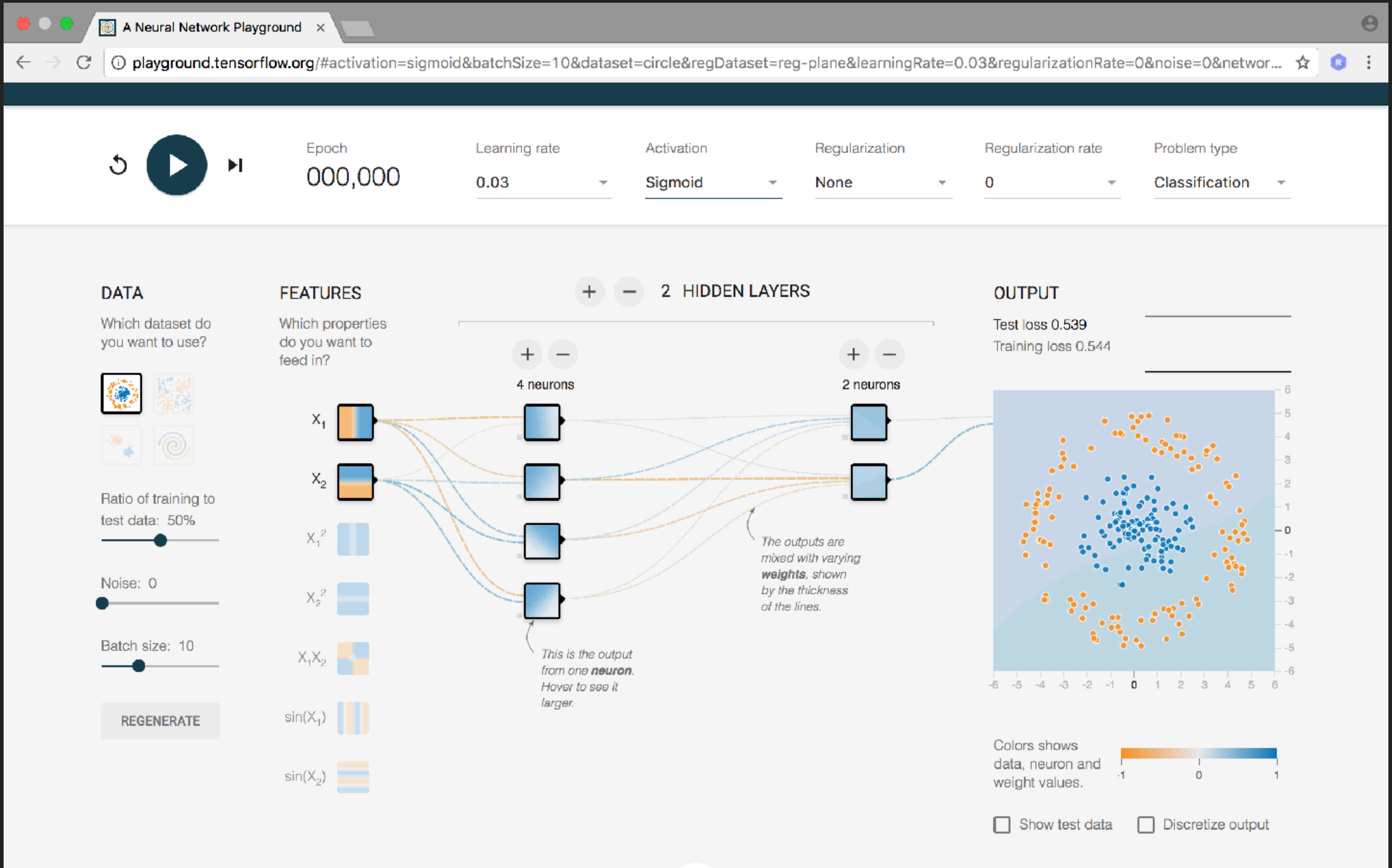
$$y = \frac{1}{1 + e^{-z}}$$

What can I do?



LOGISTIC NEURONS

- Real valued output between 0 and 1
- Can map non-linear functions
- Go to http://playground.tensorflow.org/
- Select the first data set
- Make the 'activation' sigmoid
- Can you separate the data?



A Neural Network Playground
playground.tensorflow.org/#activation=sigmoid&batchSize=10&dataset=circle&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networ...
Epoch
000,000
Learning rate
0.03
Activation
Sigmoid
Regularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
REGENERATE
FEATURES
Which properties do you want to feed in?
2 HIDDEN LAYERS
4 neurons
2 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 lines.
OUTPUT
Test loss 0.539
Training loss 0.544
Colors shows data, neuron and weight values.
Show test data
Discretize output

WHAT KIND PROBLEMS CAN THE LOGISTIC NEURON SOLVE?

Biological Neurons

MATHS REQUIRED

DIFFICULTY

Fundamental units

Work together in modular brain

The Neuron Doctrine

MATHS REQUIRED

DIFFICULTY

To understand brain, smallest unit is the neuron

Linear Neurons

MATHS REQUIRED

DIFFICULTY

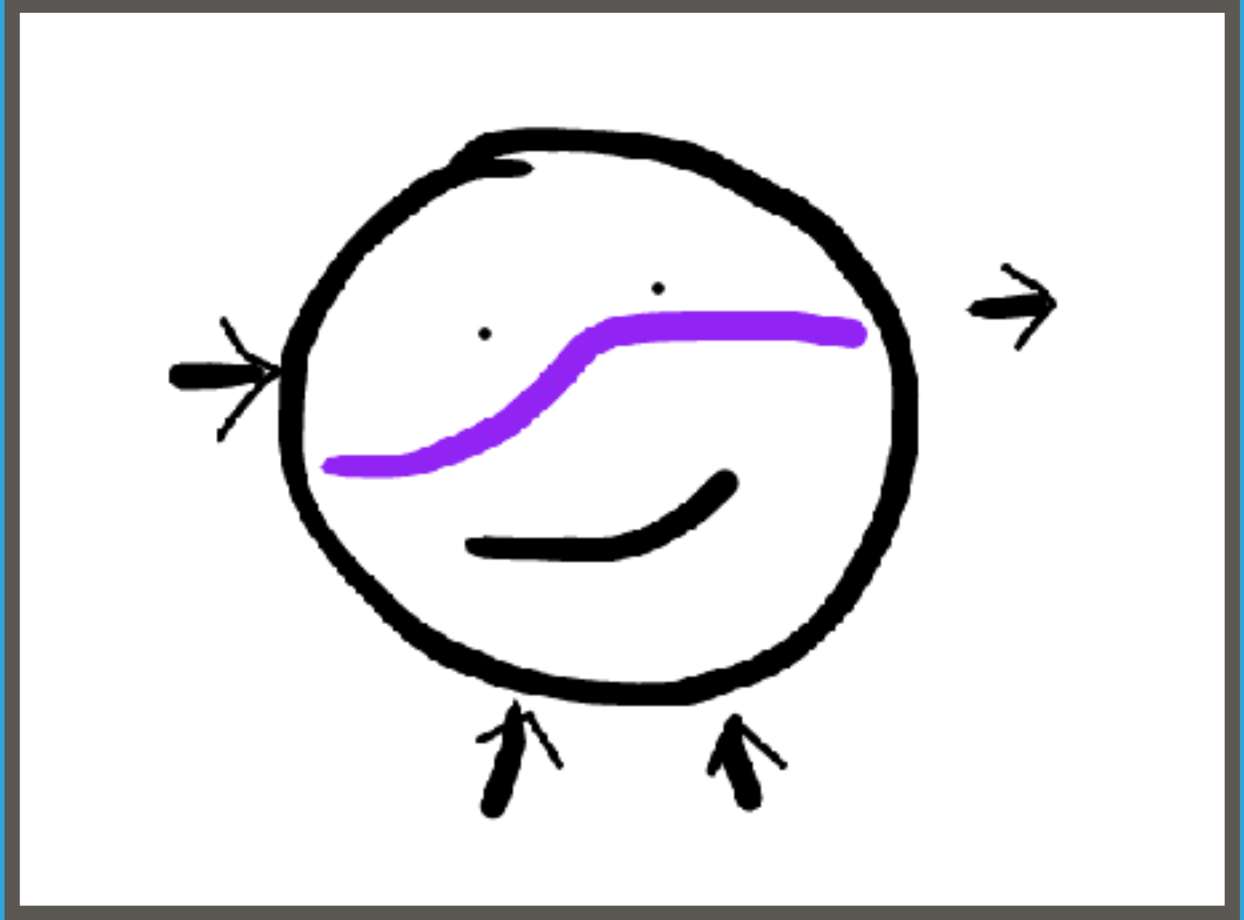
*Output is the sum of the inputs * weights, which gives a linear output*

Binary Threshold Neurons

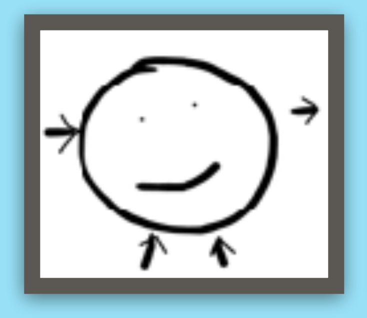
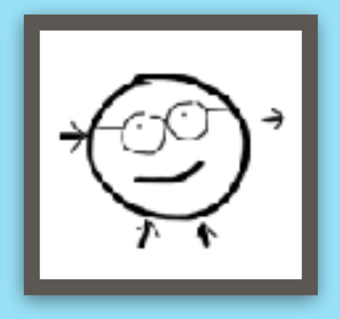
MATHS REQUIRED

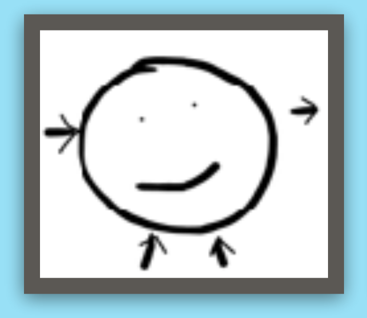
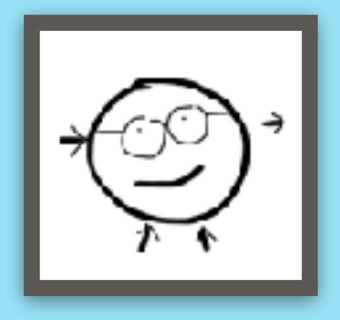
DIFFICULTY

Output 1 if input is over a threshold, or 0 otherwise

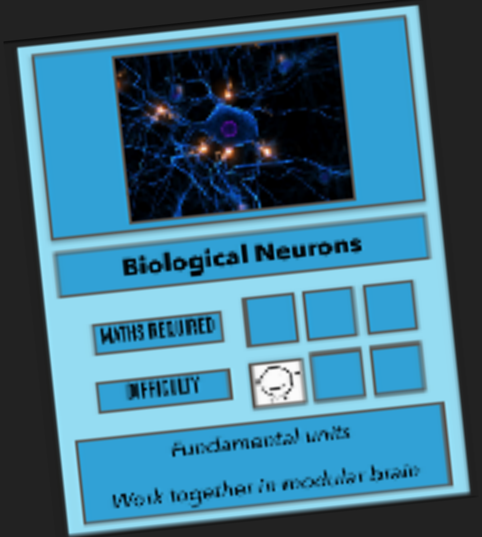


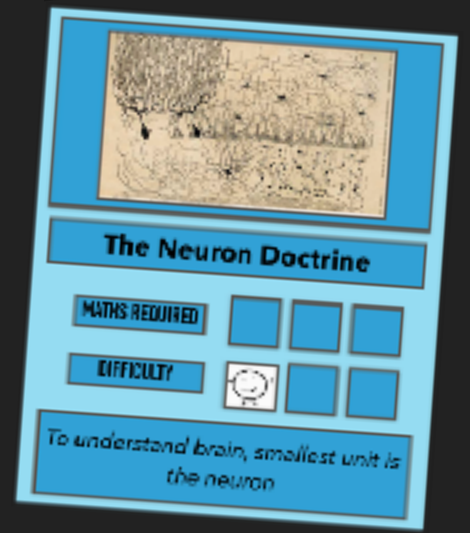
Logistic Neurons

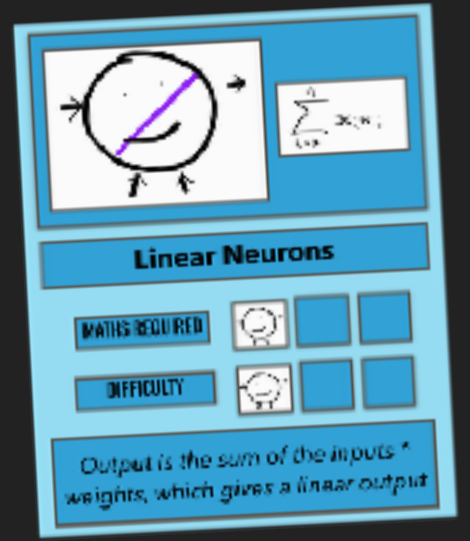
MATHS REQUIRED  

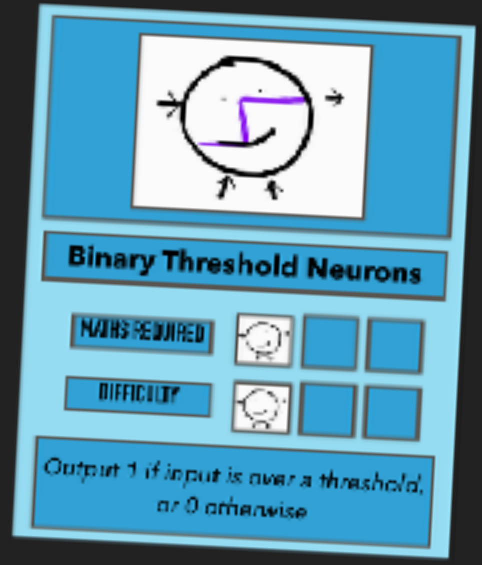
DIFFICULTY  

Output is between 1 and 0





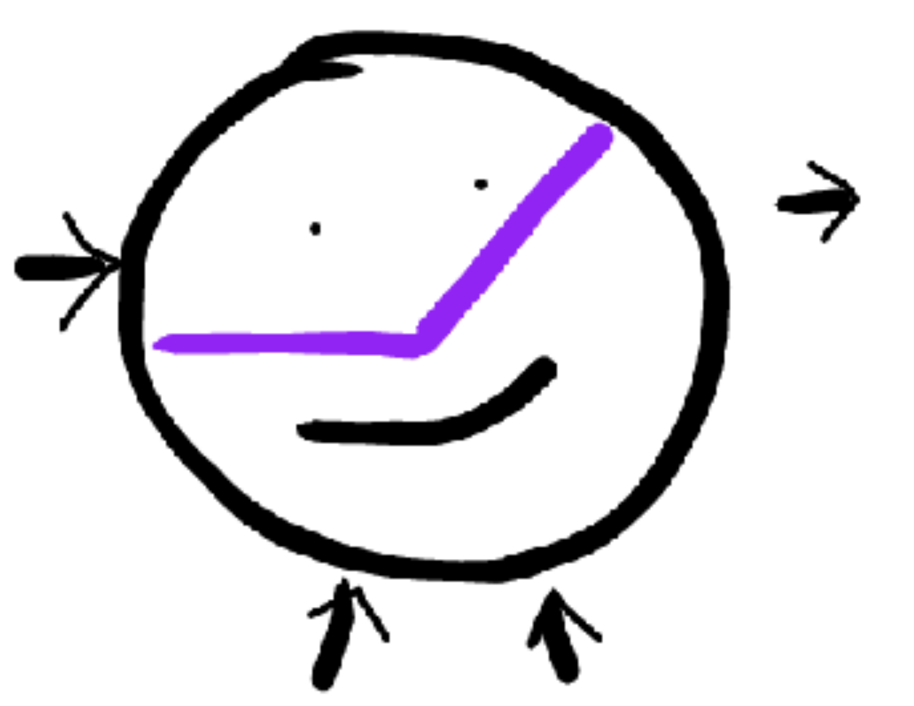




RECTIFIED LINEAR NEURONS

PART I: NEURONS

Neurons
- **Biological**
- **Artificial**
  - **Linear Neuron**
  - **Binary Threshold Neuron**
  - **Logistic/sigmoid Neuron**
  - **reLU**
  - Tanh

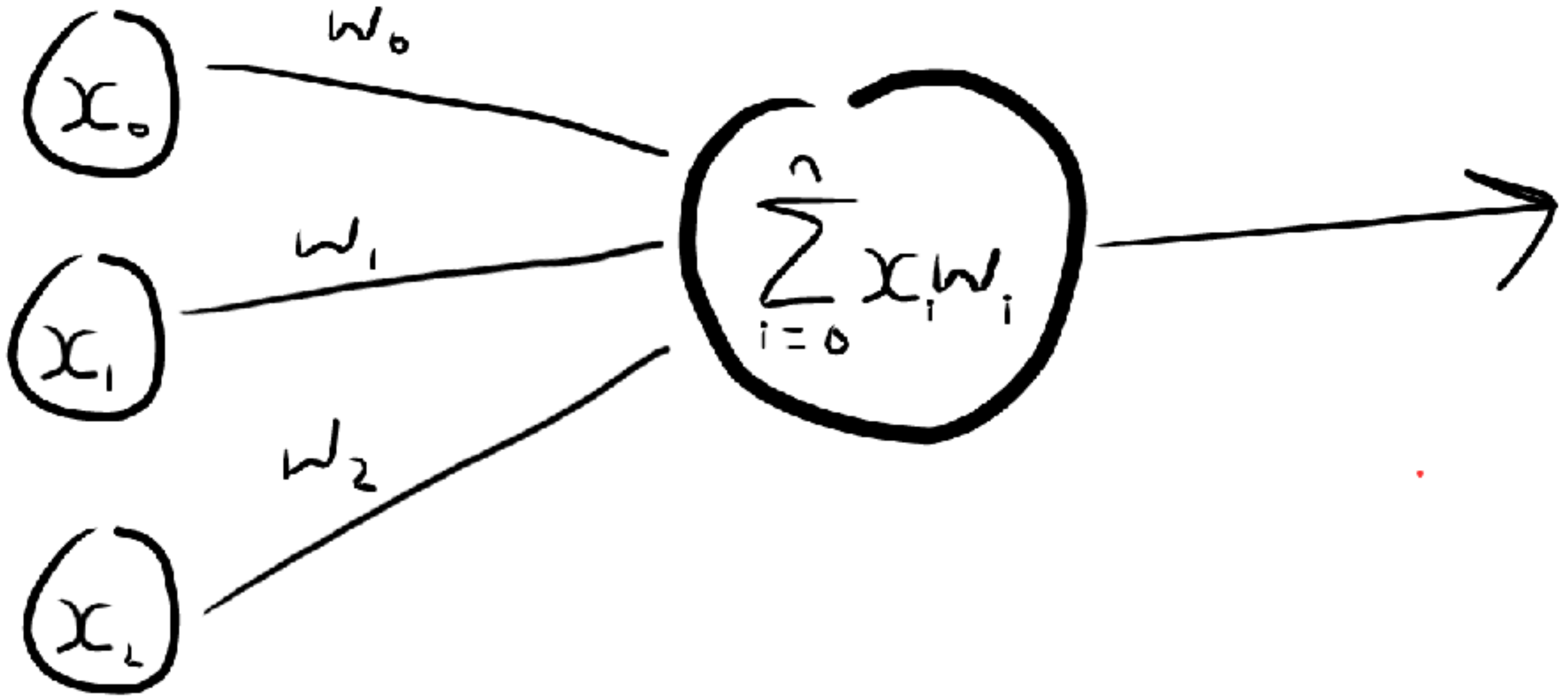


PART II: LEARNING

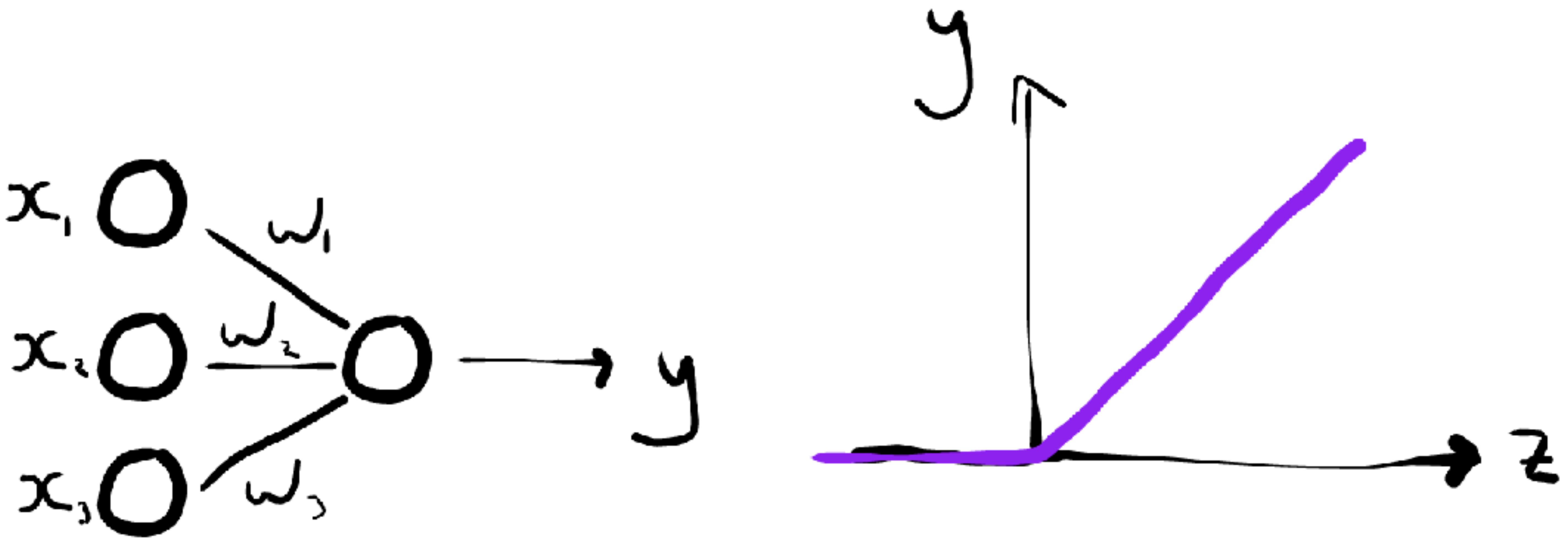
Network
- Feed forward — Cost functions
- Back propagation — Training
  - Gradient descent
  - Learning rate
  - Regularisation

PART III: IN A NETWORK

Learning to map input to output — Architectures
- CNN
- RNN
- And beyond…

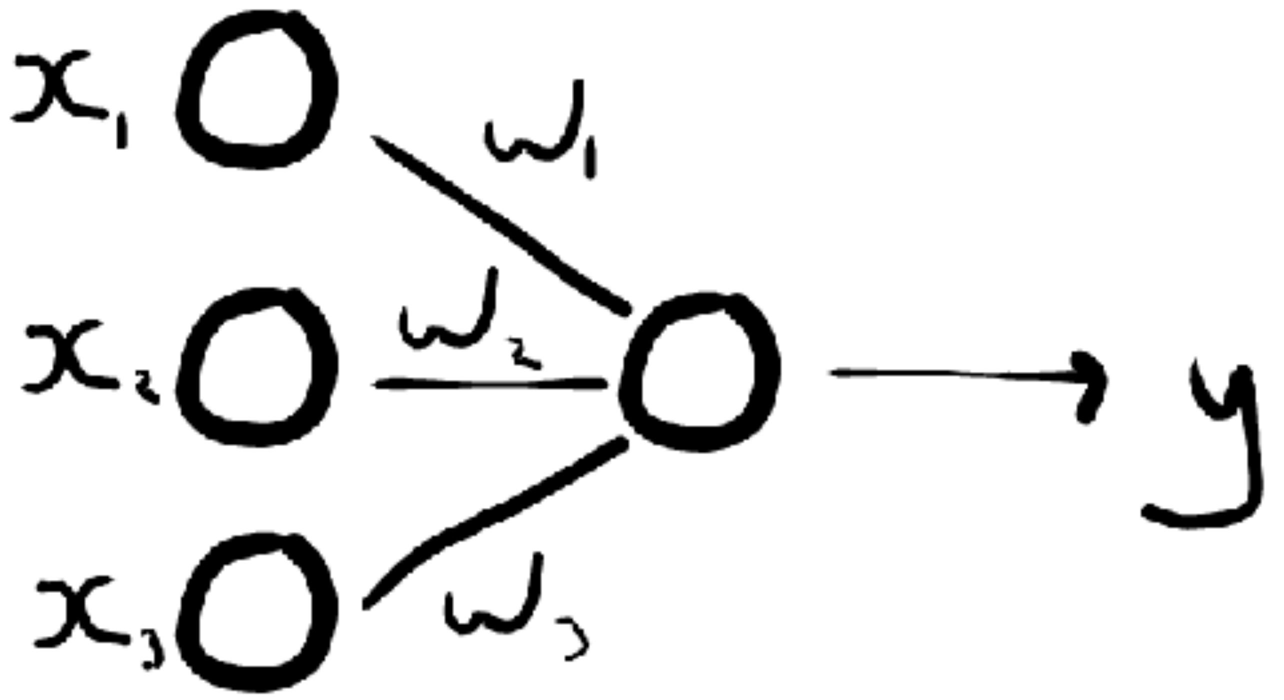


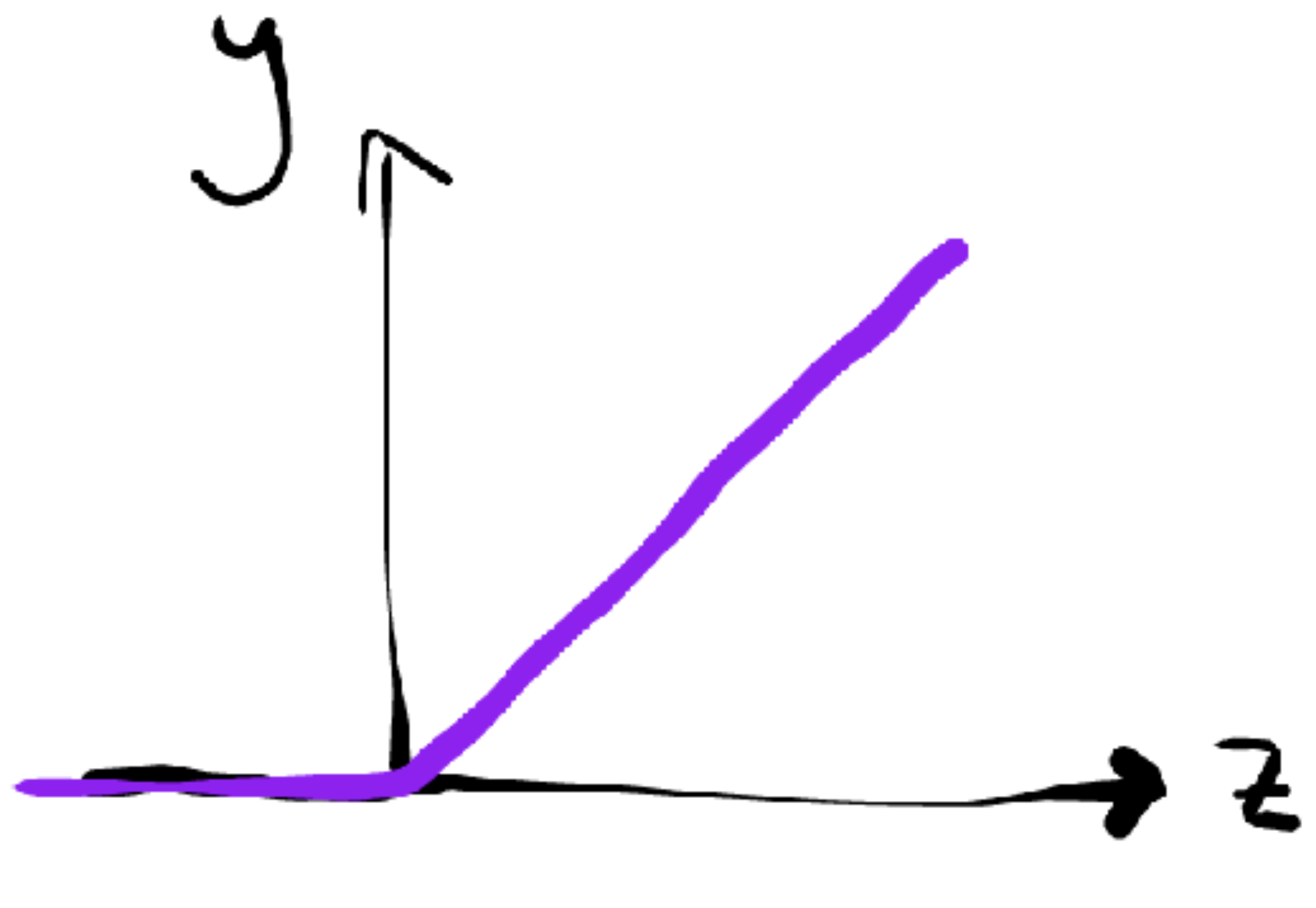
x_0
x_1
x_2
w_0
w_1
w_2
$\sum_{i=0}^{n} x_i w_i$



$$z = x_1 w_1 + x_2 w_2 + x_3 w_3$$

$$y = \max(0, z)$$

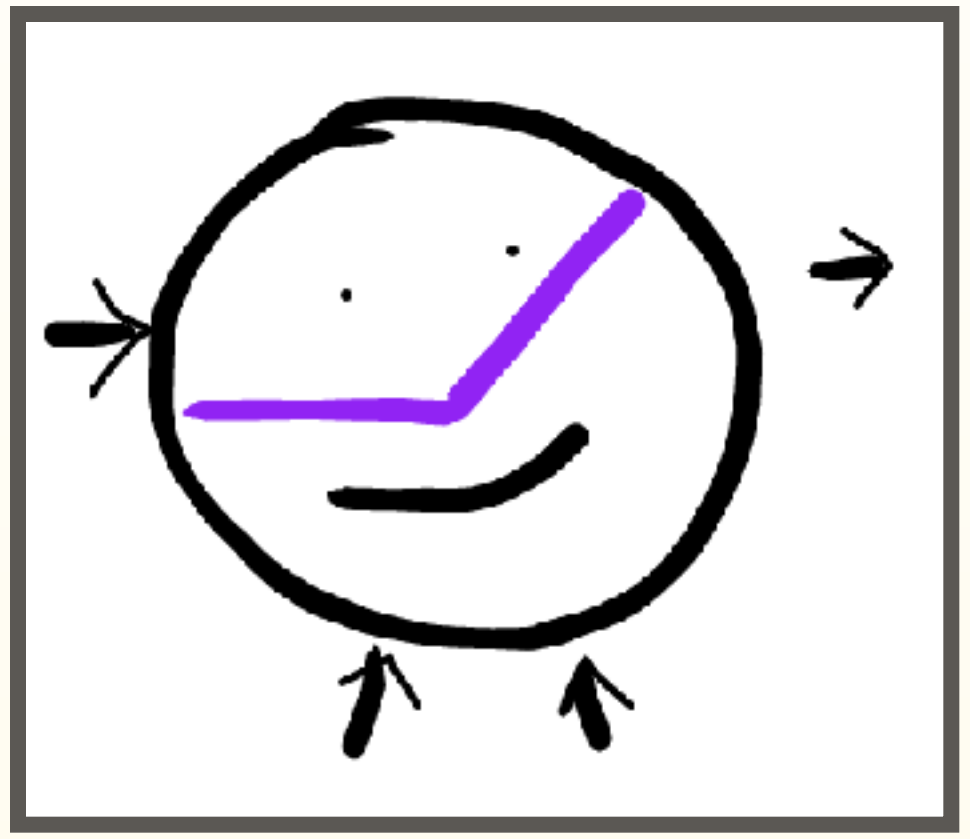




$$z = x_1 w_1 + x_2 w_2 + x_3 w_3$$

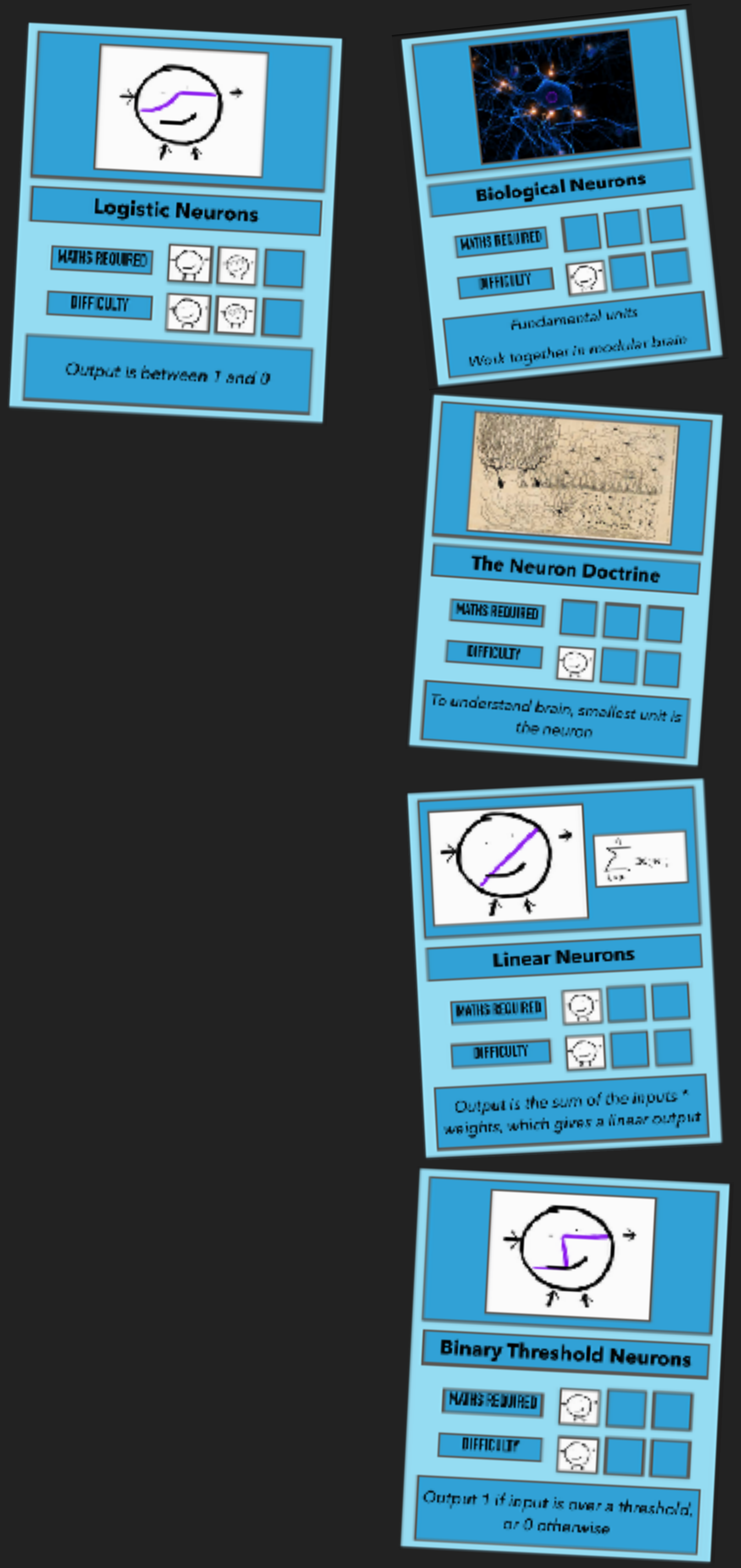
$$y = \max(0, z)$$

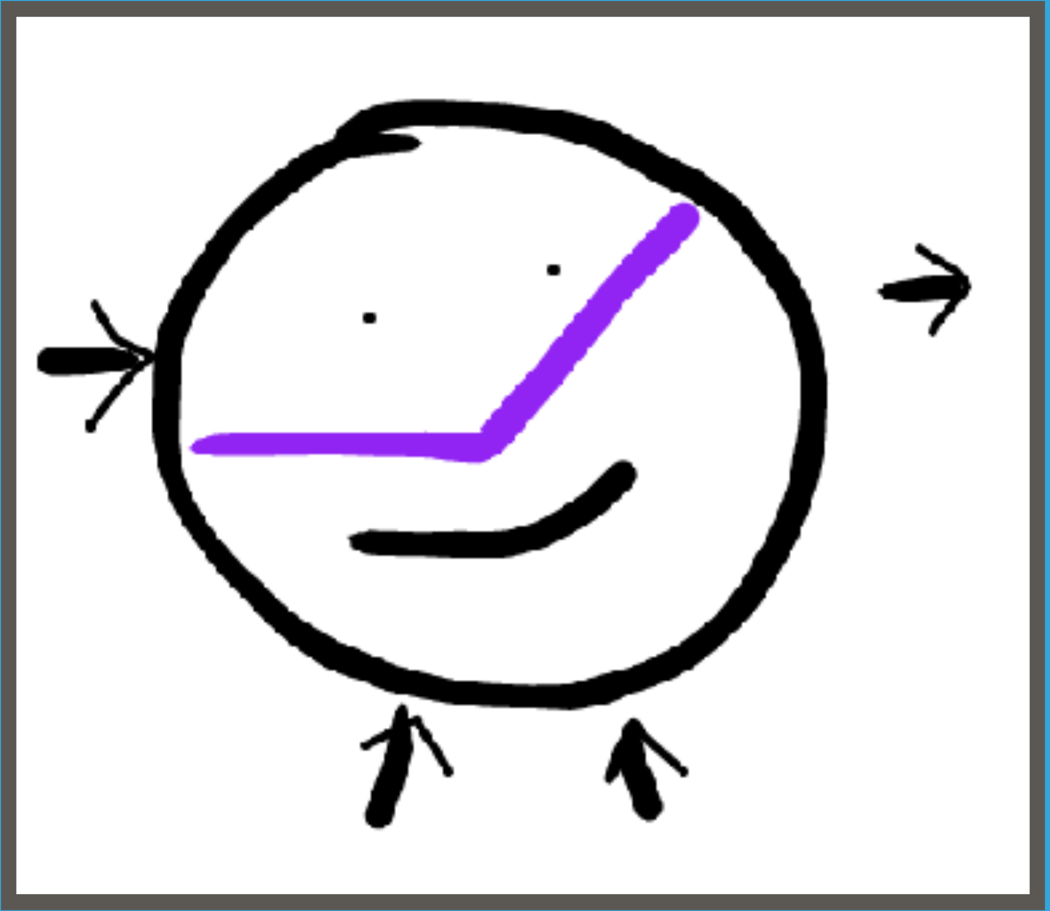
What can I do?



RECTIFIED NEURONS

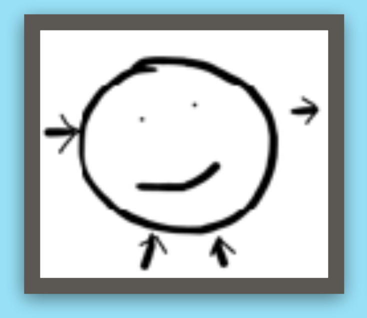
- Real valued output between 0 and infinity
- Can map non-linear functions - and fast
- Go to http://playground.tensorflow.org/
- Select the first data set
- Make the ‘activation’ ReLU
- Compare the speed to the speed using sigmoid


Logistic Neurons
MATHS REQUIRED
DIFFICULTY
Output is between 1 and 0
Biological Neurons
MATHS REQUIRED
DIFFICULTY
Fundamental units
Work together in modular brain
The Neuron Doctrine
MATHS REQUIRED
DIFFICULTY
To understand brain, smallest unit is the neuron
Linear Neurons
MATHS REQUIRED
DIFFICULTY
Output is the sum of the inputs * weights, which gives a linear output
Binary Threshold Neurons
MATHS REQUIRED
DIFFICULTY
Output 1 if input is over a threshold, or 0 otherwise

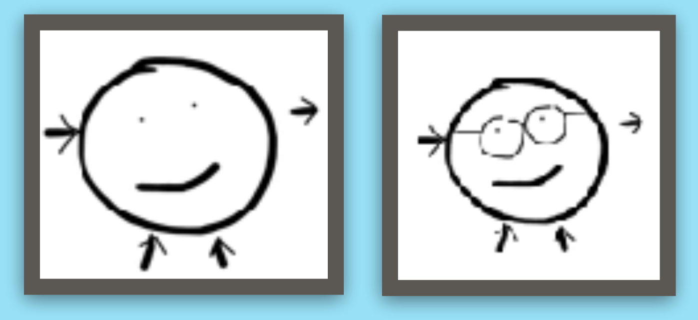


Rectified Linear Neurons

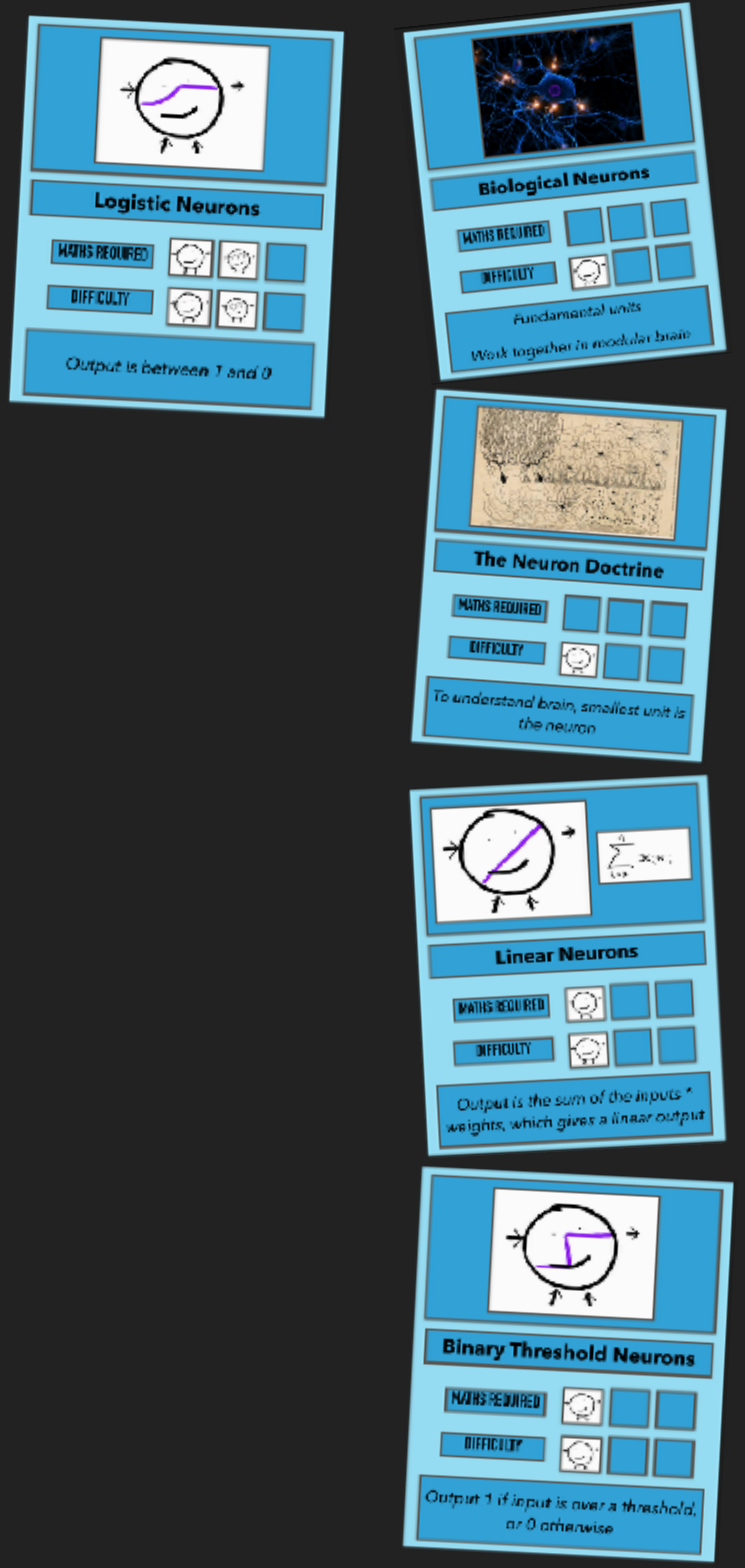
MATHS REQUIRED



DIFFICULTY



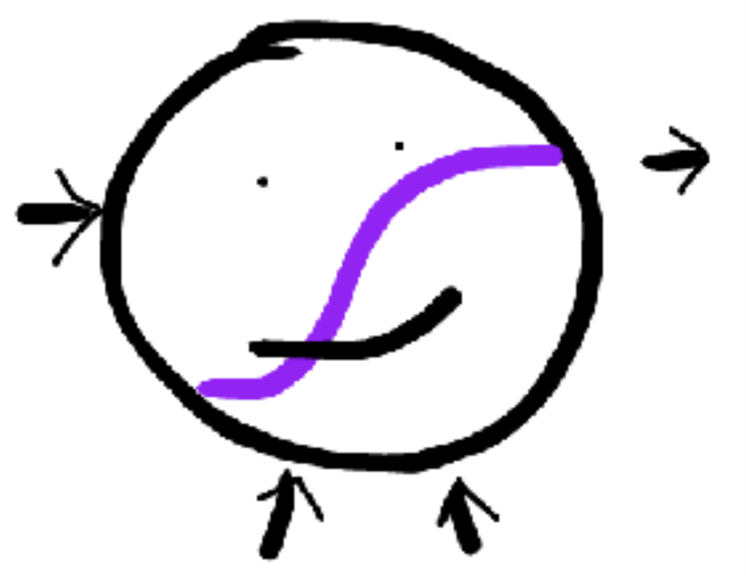
Output is linear if input is greater than 0, or 0 otherwise



TANH NEURONS

PART I: NEURONS

Neurons
- **Biological**
- **Artificial** — **Linear Neuron**, **Binary Threshold Neuron**, **Logistic/sigmoid Neuron**, **reLU**, **Tanh**

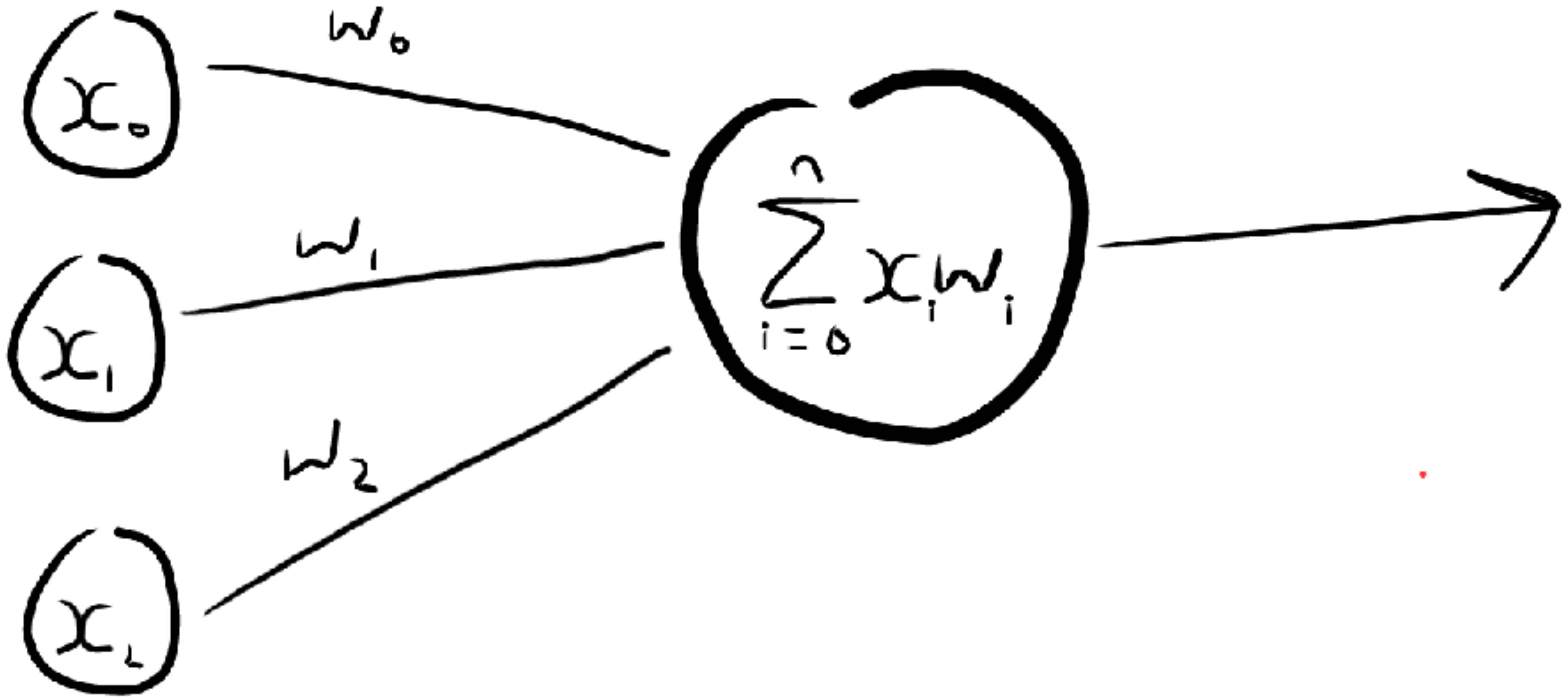


PART II: LEARNING

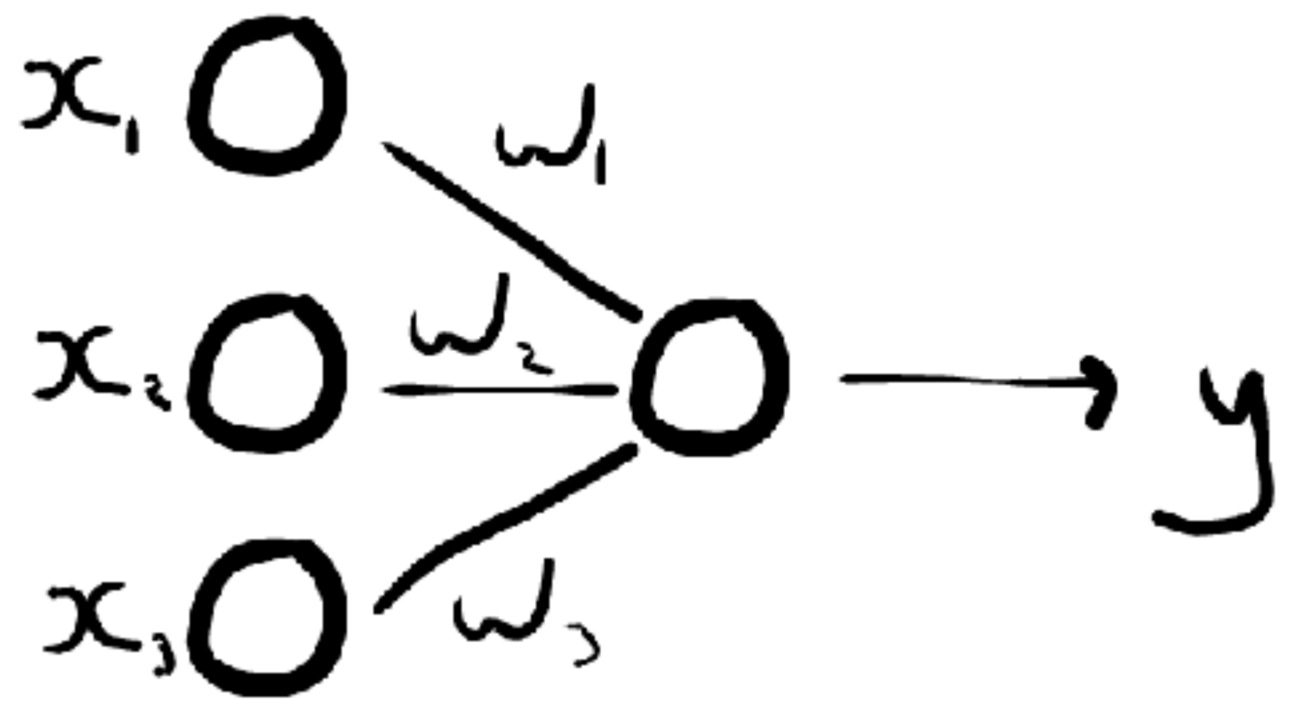
Network
- Feed forward — Cost functions
- Back propagation — Training — Gradient descent, Learning rate, Regularisation

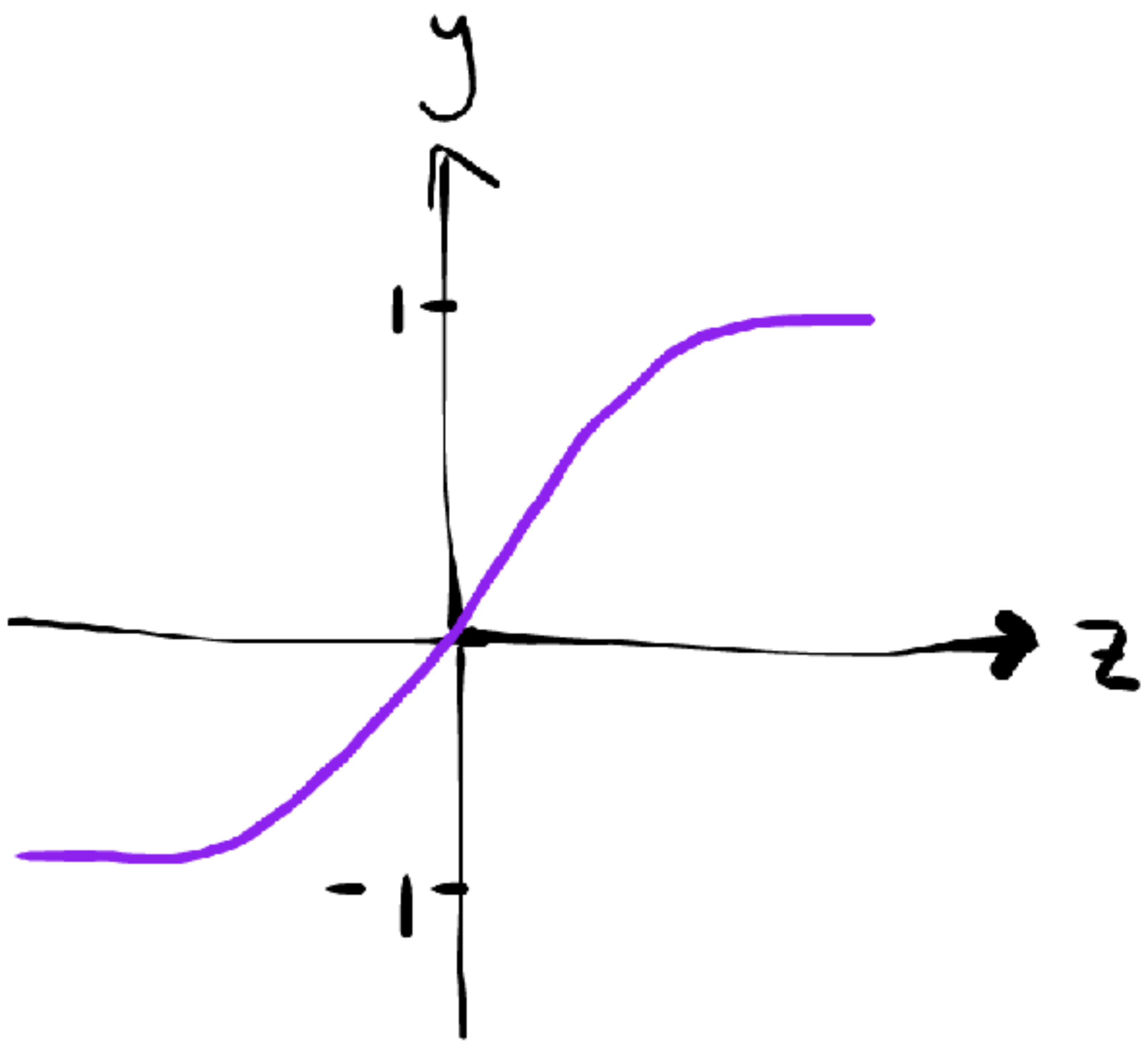
PART III: IN A NETWORK

Learning to map input to output — Architectures — CNN, RNN, And beyond…



x_0
x_1
x_2
w_0
w_1
w_2
$\sum_{i=0}^{n} x_i w_i$

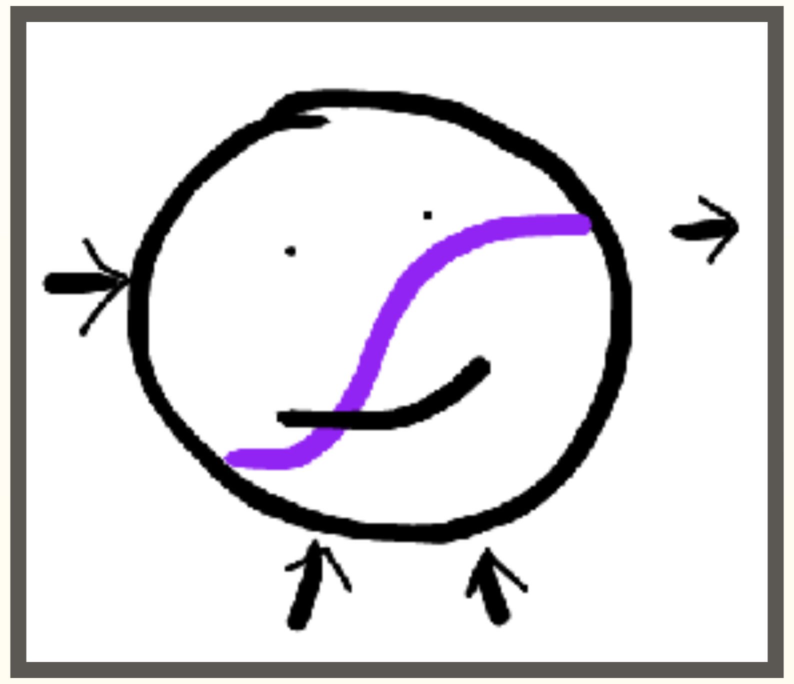




$$z = x_1 w_1 + x_2 w_2 + x_3 w_3$$

$$y = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

What can I do?

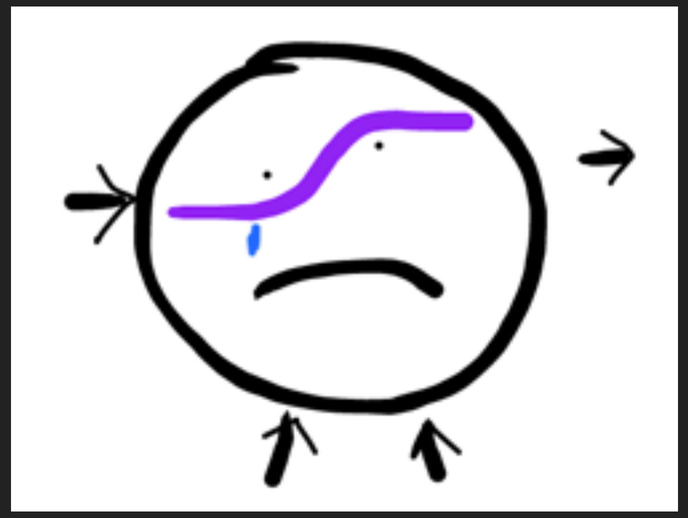


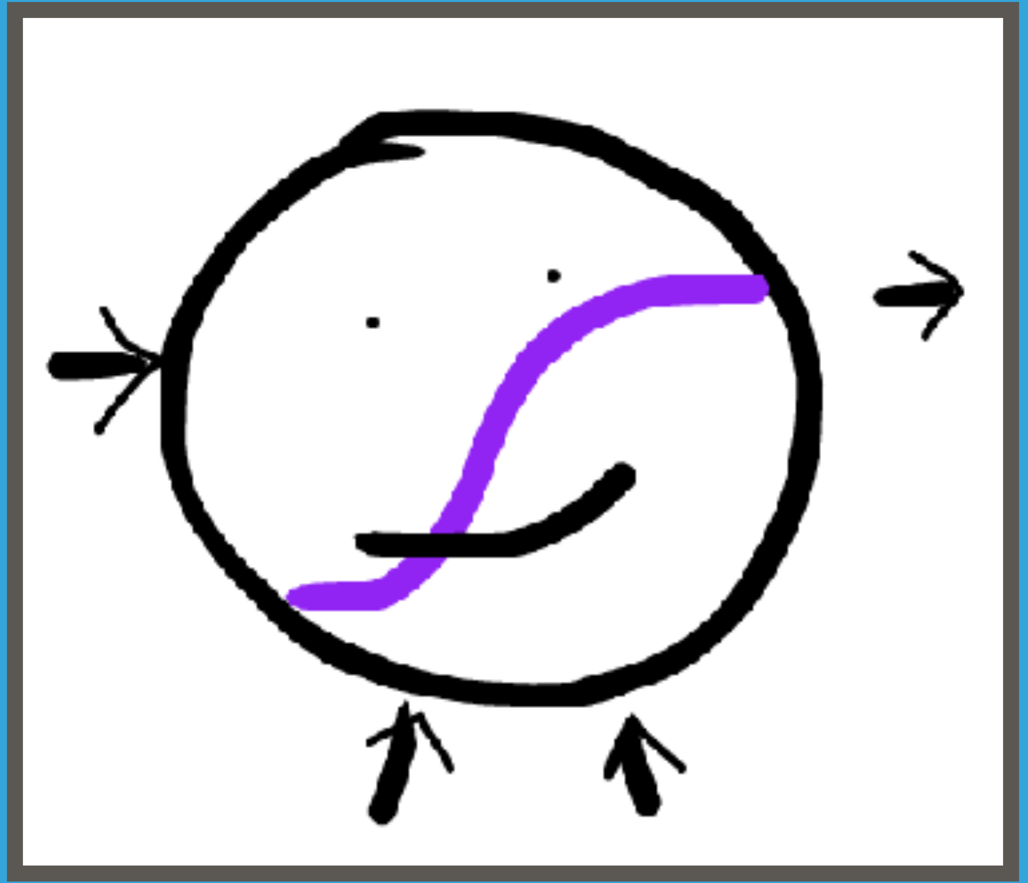
TANH NEURONS

- Real valued output between -1 and 1
- Like sigmoid…. But faster
- Go to http://playground.tensorflow.org/
- Select the first data set
- Make the ‘activation’ Tanh
- Compare the speed to the speed using sigmoid

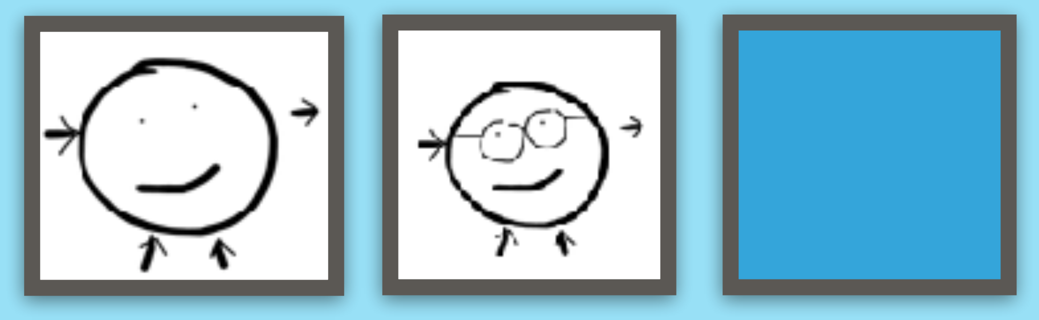
TANH NEURONS

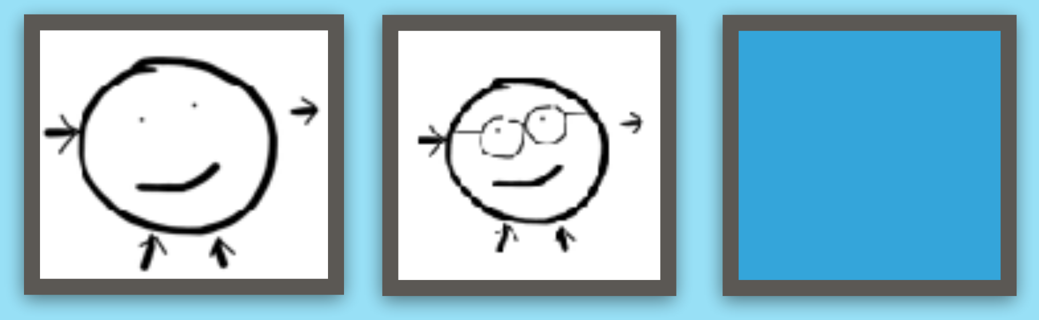
- Real valued output between -1 and 1
- Like sigmoid…. But faster
- Go to http://playground.tensorflow.org/
- Select the first data set
- Make the ‘activation’ Tanh
- Compare the speed to the speed using sigmoid



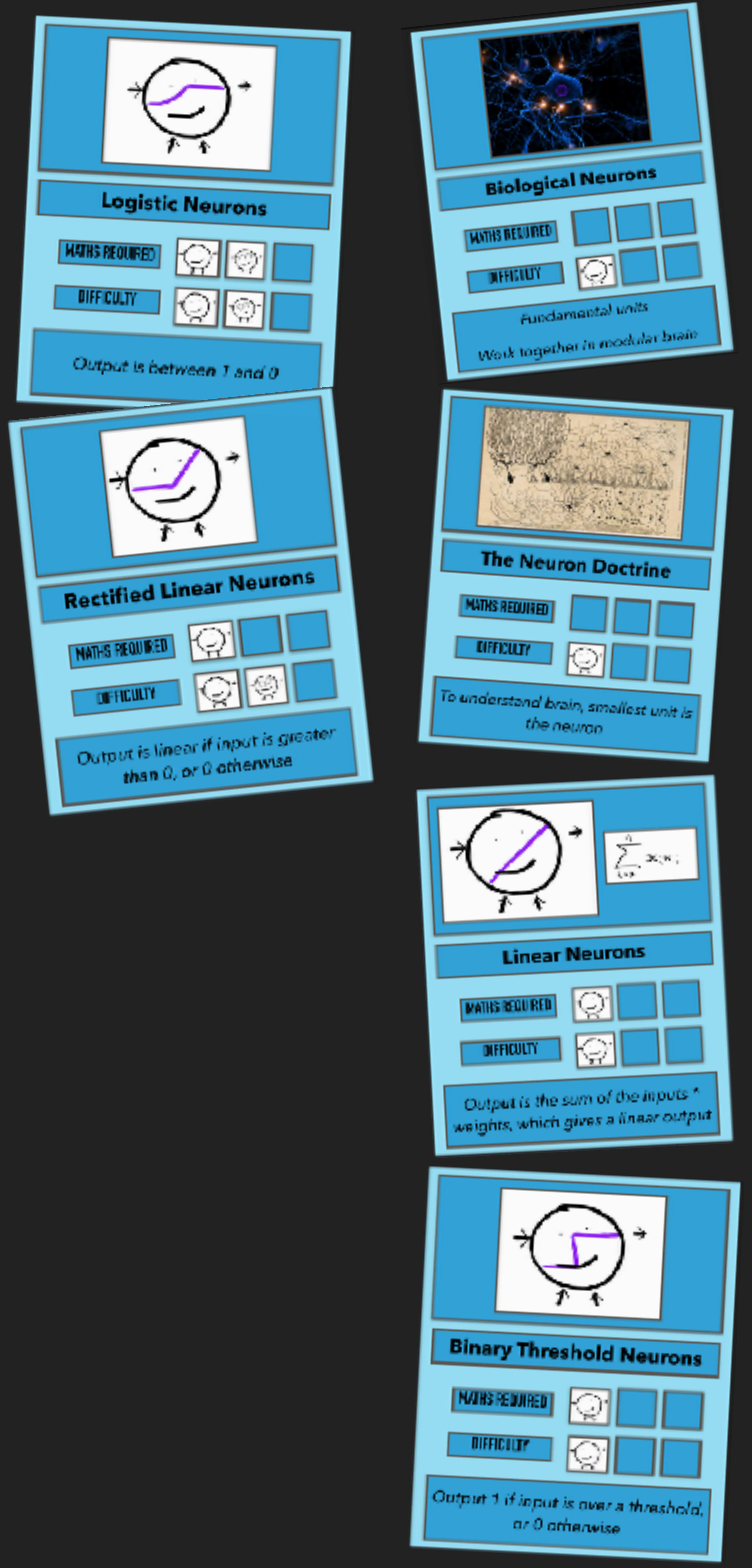


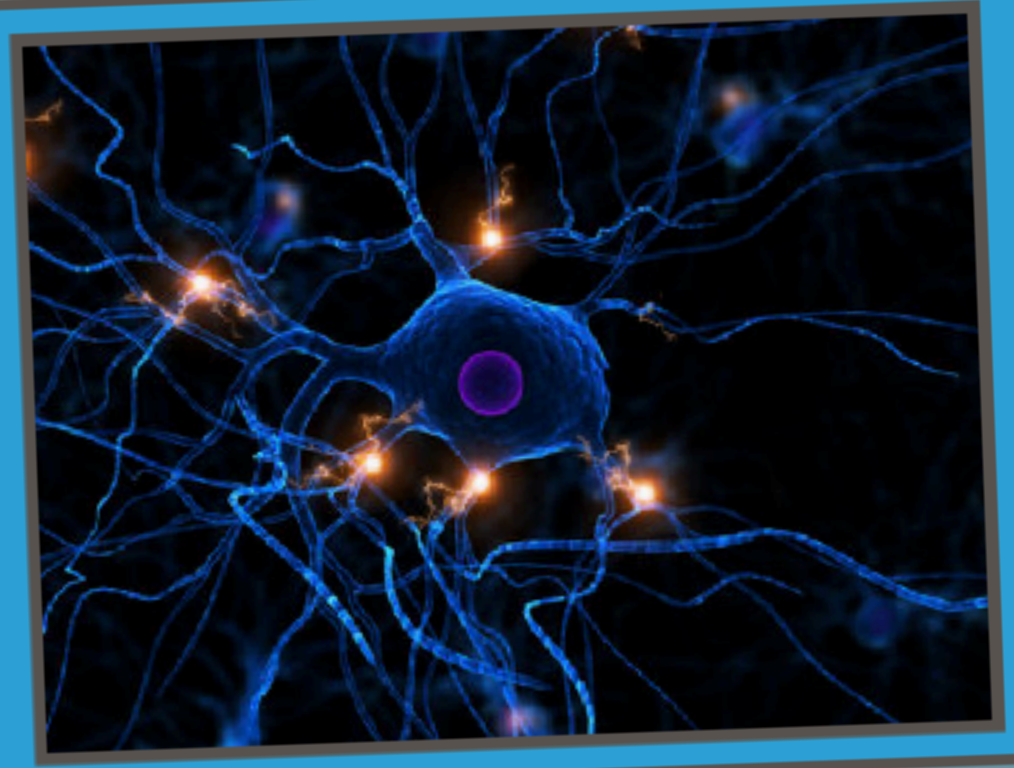
Tanh Neurons

MATHS REQUIRED 

DIFFICULTY 

Output is between -1 and 1





Biological Neurons

MATHS REQUIRED

DIFFICULTY

Fundamental units

Work together in modular brain

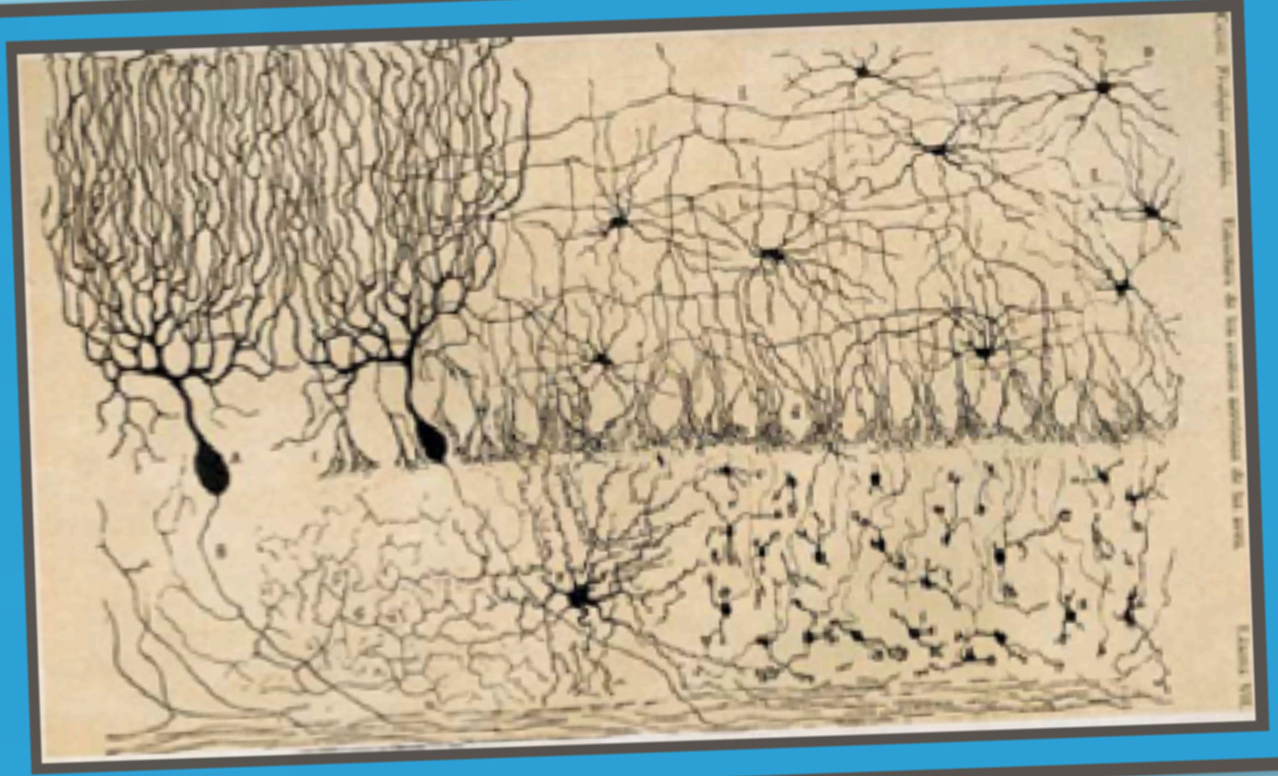
Biological

MATHS REQUIRED

DIFFICULTY

Fundamen

Work together in

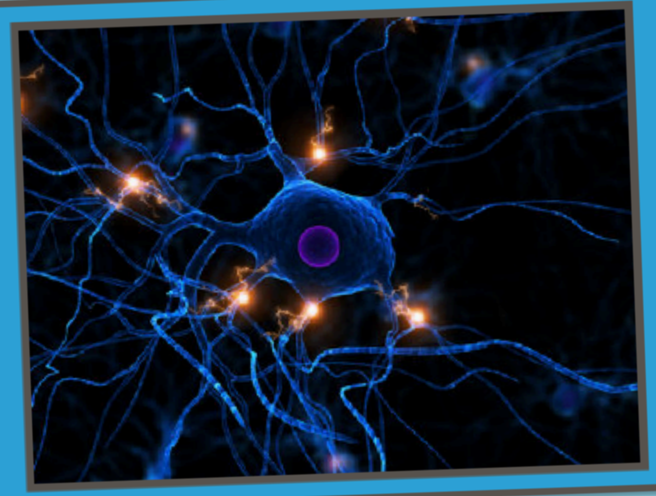


The Neuron Doctrine

MATHS REQUIRED

DIFFICULTY

To understand brain, smallest unit is the neuron



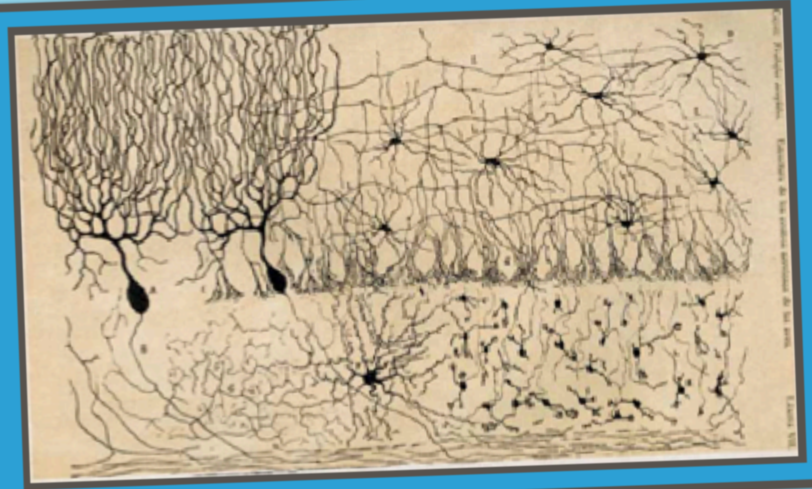
Biological Neurons

MATHS REQUIRED

DIFFICULTY

Fundamental units

Work together in modular brain

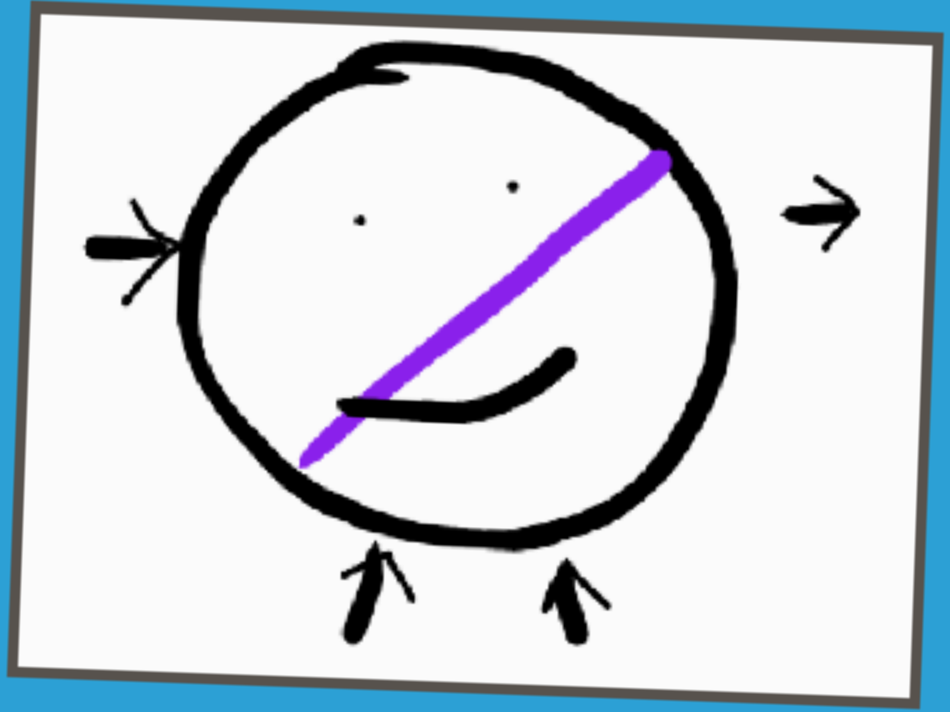


The Neuron Doctrine

MATHS REQUIRED

DIFFICULTY

To understand brain, smallest unit is the neuron



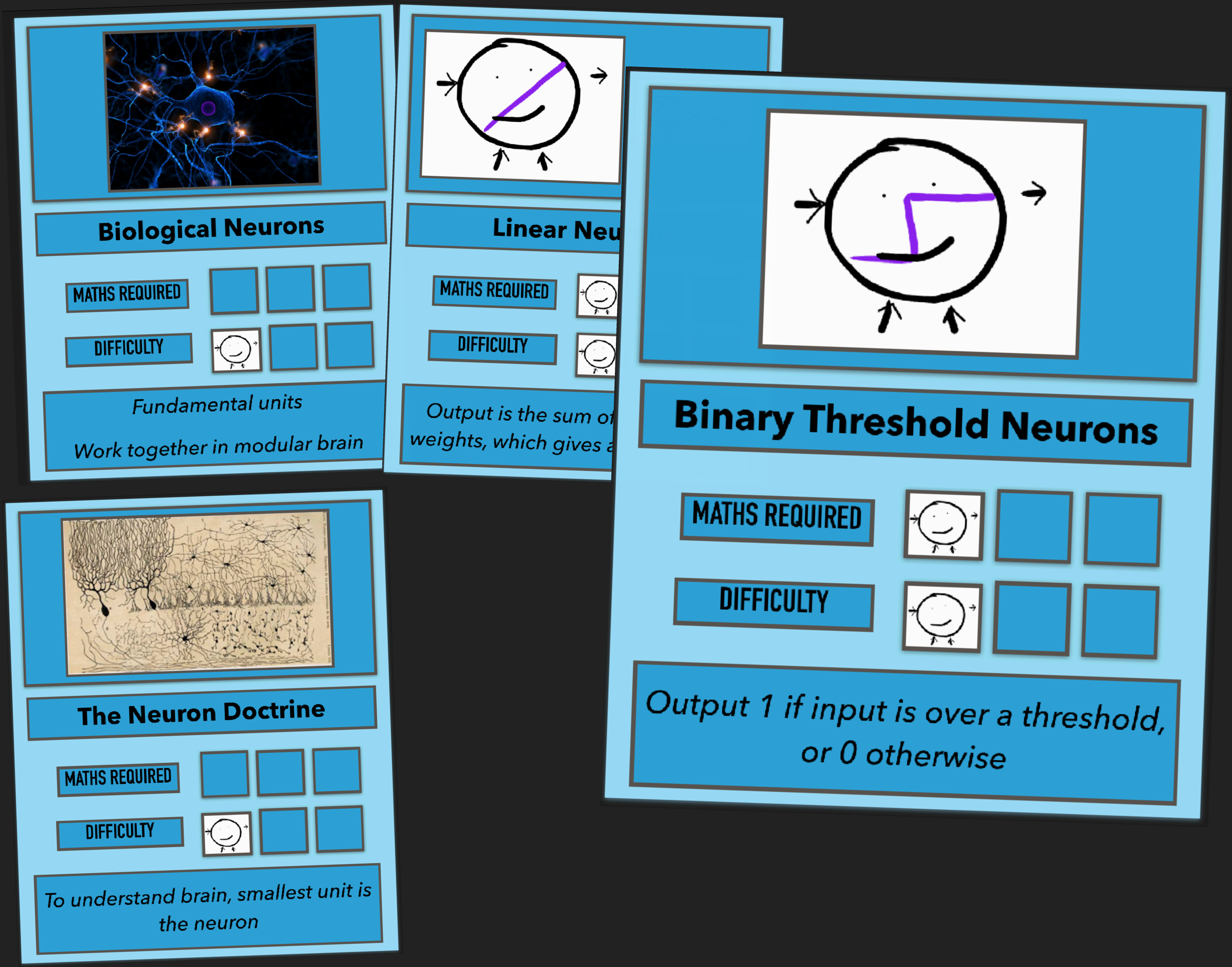
$$\sum_{i=0}^{n} x_i w_i$$

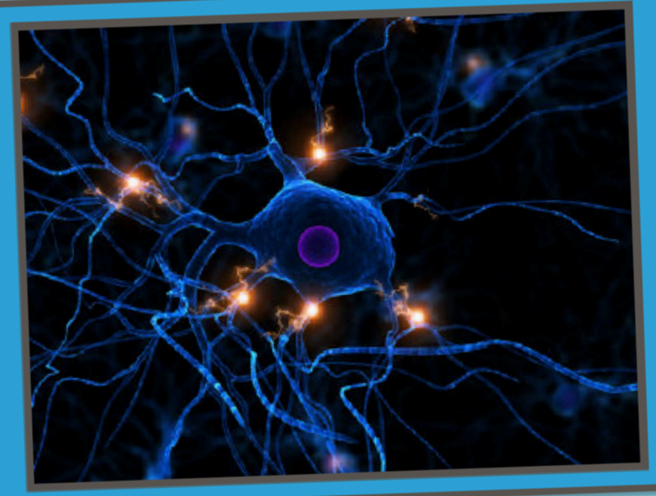
Linear Neurons

MATHS REQUIRED

DIFFICULTY

*Output is the sum of the inputs * weights, which gives a linear output*


Biological Neurons
MATHS REQUIRED
DIFFICULTY
Fundamental units
Work together in modular brain
Linear Neu
MATHS REQUIRED
DIFFICULTY
Output is the sum of
weights, which gives a
Binary Threshold Neurons
MATHS REQUIRED
DIFFICULTY
Output 1 if input is over a threshold,
or 0 otherwise
The Neuron Doctrine
MATHS REQUIRED
DIFFICULTY
To understand brain, smallest unit is
the neuron



Biological Neurons

MATHS REQUIRED

DIFFICULTY

Fundamental units

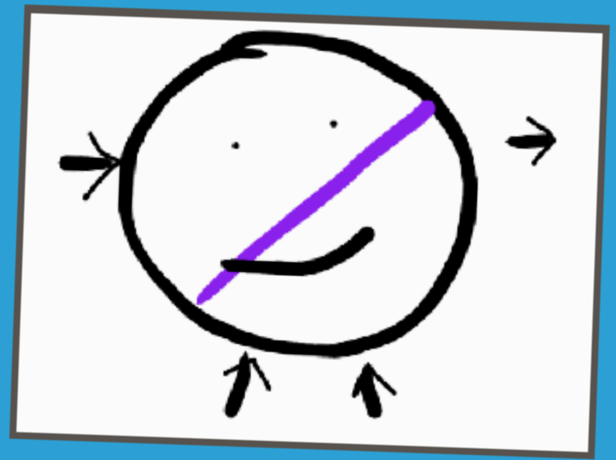
Work together in modular brain

The Neuron Doctrine

MATHS REQUIRED

DIFFICULTY

To understand brain, smallest unit is the neuron



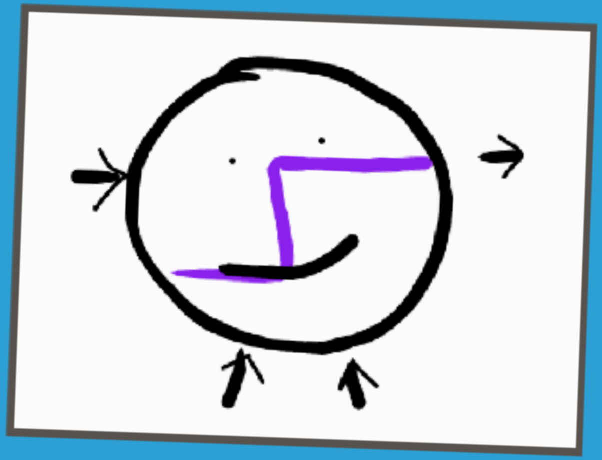
$$\sum_{i=0}^{n} x_i w_i$$

Linear Neurons

MATHS REQUIRED

DIFFICULTY

*Output is the sum of the inputs * weights, which gives a linear output*



Binary Threshold Neurons

MATHS REQUIRED

DIFFICULTY

Output 1 if input is over a threshold, or 0 otherwise

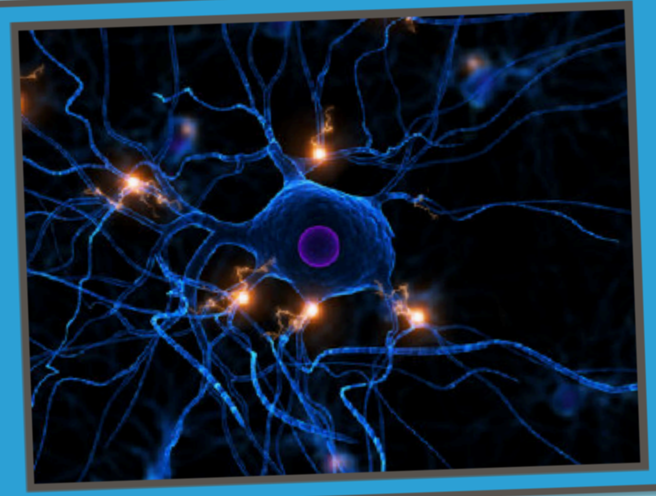
Logistic Neurons

MATHS REQUIRED

DIFFICULTY

Output is between 1 and 0


Biological Neurons
MATHS REQUIRED
DIFFICULTY
Fundamental units
Work together in modular brain
$\sum_{i=0}^{n} x_i w_i$
Linear Neurons
MATHS REQUIRED
DIFFICULTY
Output is the sum of the inputs * weights, which gives a linear output
Logistic
MATHS REQUIRED
DIFFICULTY
Output is be
Tanh Neurons
MATHS REQUIRED
DIFFICULTY
Output is between -1 and 1
The Neuron Doctrine
MATHS REQUIRED
DIFFICULTY
To understand brain, smallest unit is the neuron
Binary Threshold Neurons
MATHS REQUIRED
DIFFICULTY
Output 1 if input is over a threshold, or 0 otherwise



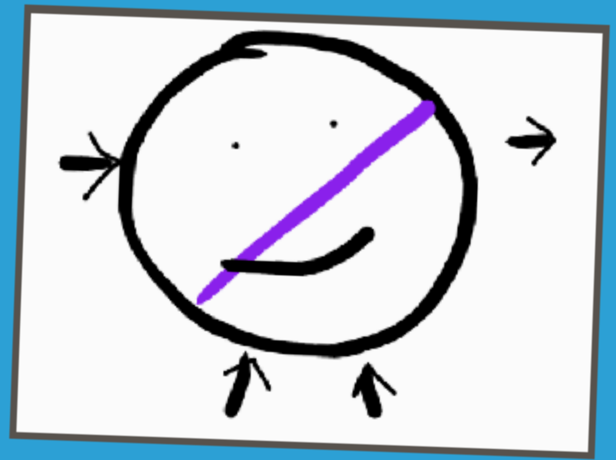
Biological Neurons

MATHS REQUIRED

DIFFICULTY

Fundamental units

Work together in modular brain



$$\sum_{i=0}^{n} x_i w_i$$

Linear Neurons

MATHS REQUIRED

DIFFICULTY

*Output is the sum of the inputs * weights, which gives a linear output*

Logistic Neurons

MATHS REQUIRED

DIFFICULTY

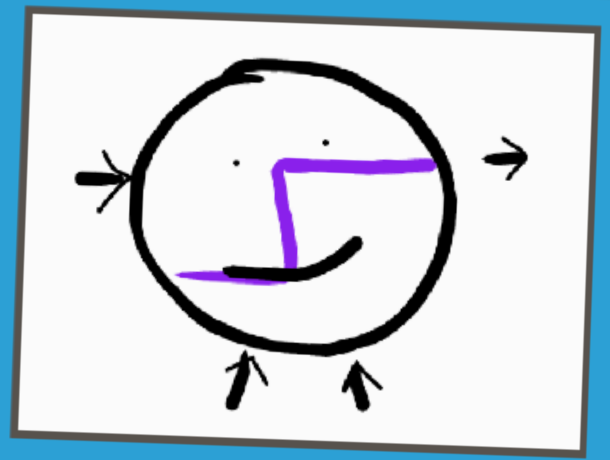
Output is between 1 and 0

The Neuron Doctrine

MATHS REQUIRED

DIFFICULTY

To understand brain, smallest unit is the neuron

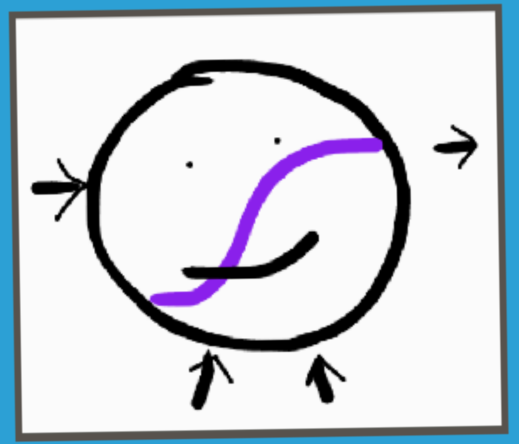


Binary Threshold Neurons

MATHS REQUIRED

DIFFICULTY

Output 1 if input is over a threshold, or 0 otherwise

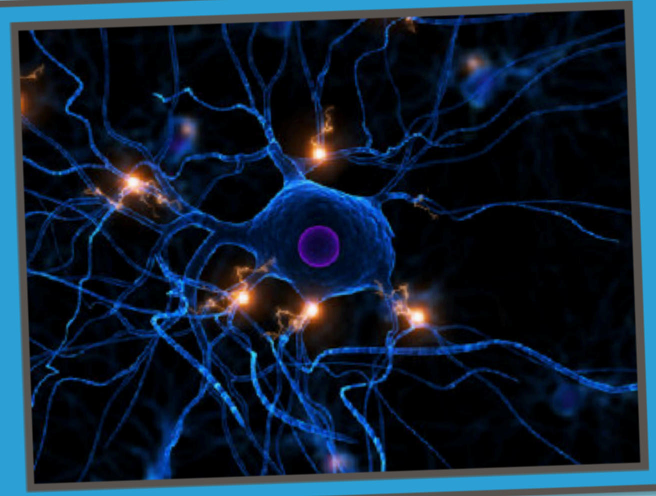


Tanh Neurons

MATHS REQUIRED

DIFFICULTY

Output is between -1 and 1



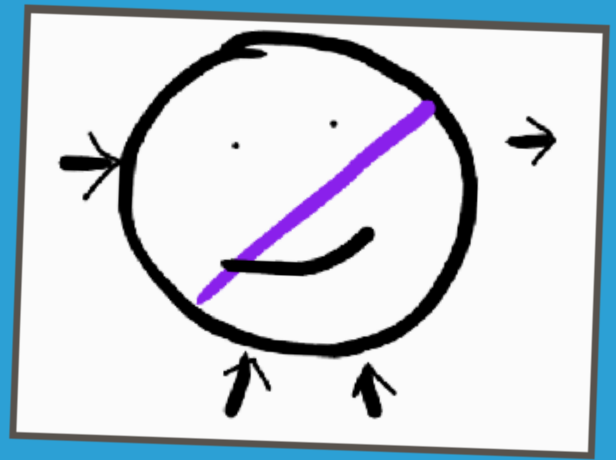
Biological Neurons

MATHS REQUIRED

DIFFICULTY

Fundamental units

Work together in modular brain



$$\sum_{i=0}^{n} x_i w_i$$

Linear Neurons

MATHS REQUIRED

DIFFICULTY

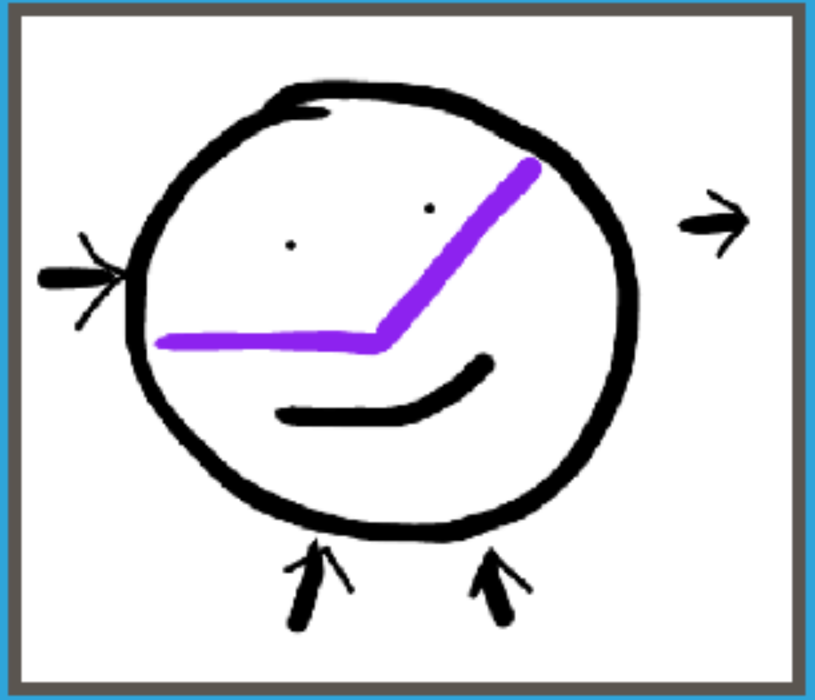
*Output is the sum of the inputs * weights, which gives a linear output*

Logistic Neurons

MATHS REQUIRED

DIFFICULTY

Output is between 1 and 0



Rectified Linear Neurons

MATHS REQUIRED

DIFFICULTY

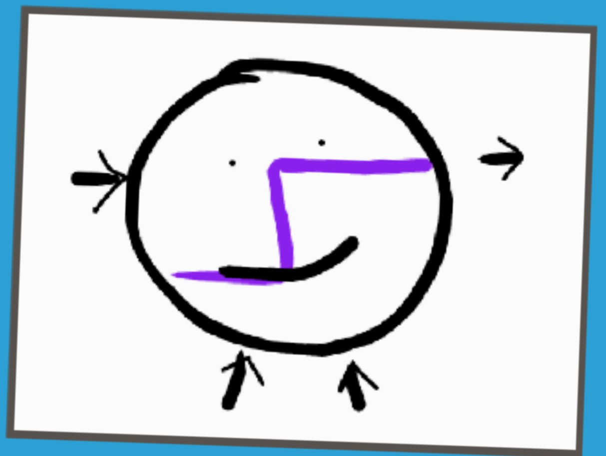
Output is linear if input is greater than 0, or 0 otherwise

The Neuron Doctrine

MATHS REQUIRED

DIFFICULTY

To understand brain, smallest unit is the neuron

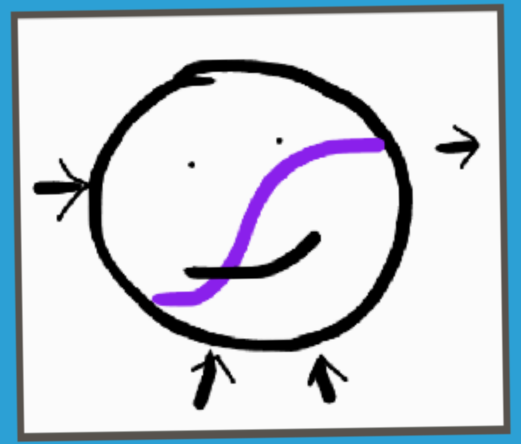


Binary Threshold Neurons

MATHS REQUIRED

DIFFICULTY

Output 1 if input is over a threshold, or 0 otherwise



Tanh Neurons

MATHS REQUIRED

DIFFICULTY

Output is between -1 and 1

PART I: NEURONS

Neurons

- **Biological**
- **Artificial**
  - **Binary Threshold Neuron**
  - **Linear Neuron**
  - **Logistic/sigmoid Neuron**
  - **reLU**
  - **Tanh**

PART II: LEARNING

Network

- Feed forward
  - Cost functions
- Back propagation
  - Training
    - Gradient descent
    - Learning rate
    - Regularisation

PART III: IN A NETWORK

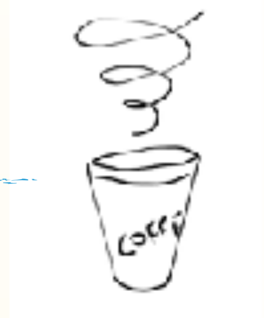
Learning to map input to output

- Architectures
  - CNN
  - RNN
  - And beyond…

PART I: NEURONS

Neurons

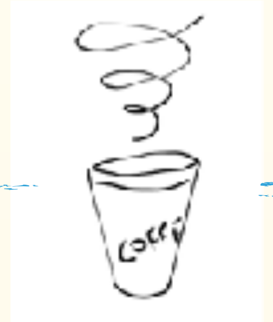
- **Biological**
- **Artificial** — **Binary Threshold Neuron**, **Linear Neuron**, **Logistic/sigmoid Neuron**, **reLU**, **Tanh**



PART II: LEARNING

Network

- Feed forward — Cost functions
- Back propagation — Training — Gradient descent, Learning rate, Regularisation



PART III: IN A NETWORK

Learning to map input to output — Architectures — CNN, RNN, And beyond…

@KATHARINECODES

PART II: LEARNING

LEARNING IN A SIMPLE NETWORK

PART I: NEURONS

- Neurons
  - Biological
  - Artificial
    - Linear Neuron
    - Binary Threshold Neuron
    - Logistic/sigmoid Neuron
    - reLU
    - Tanh

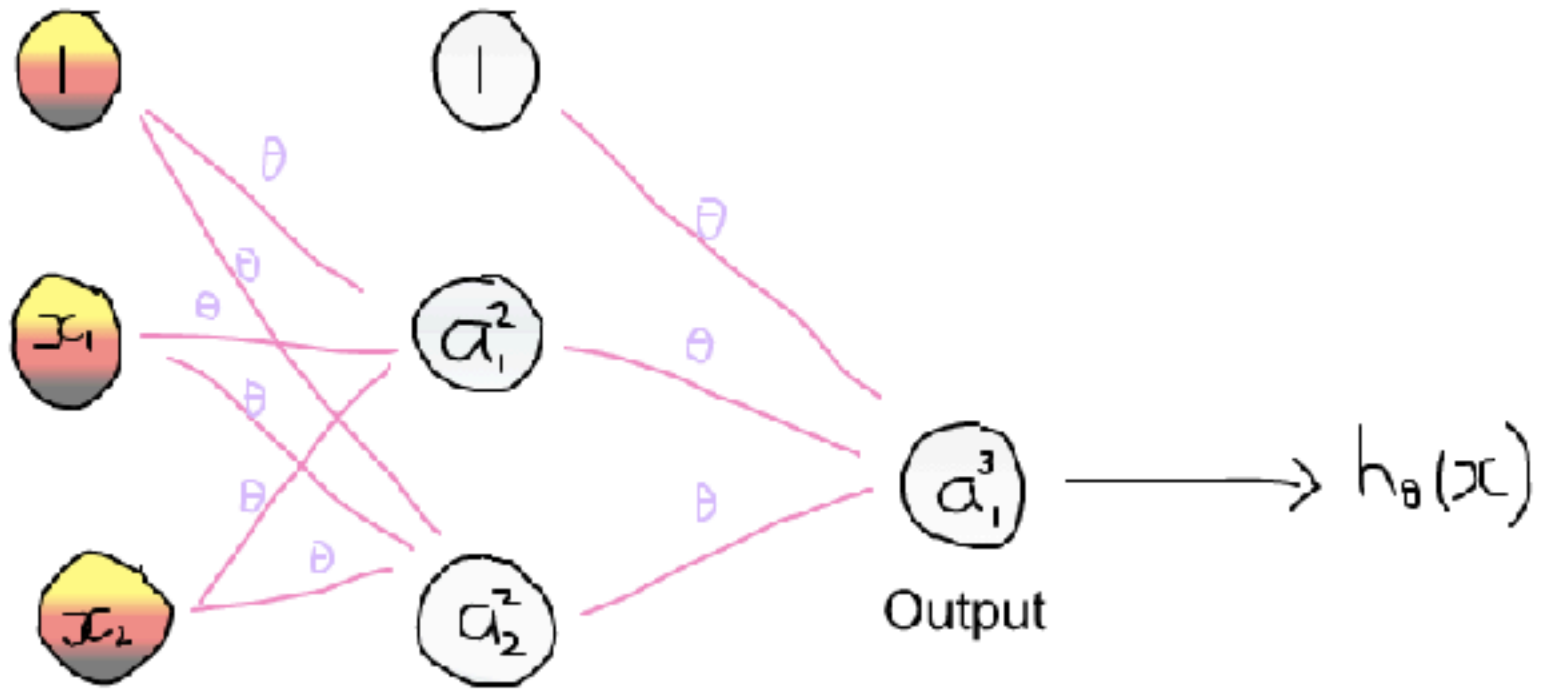
PART II: LEARNING

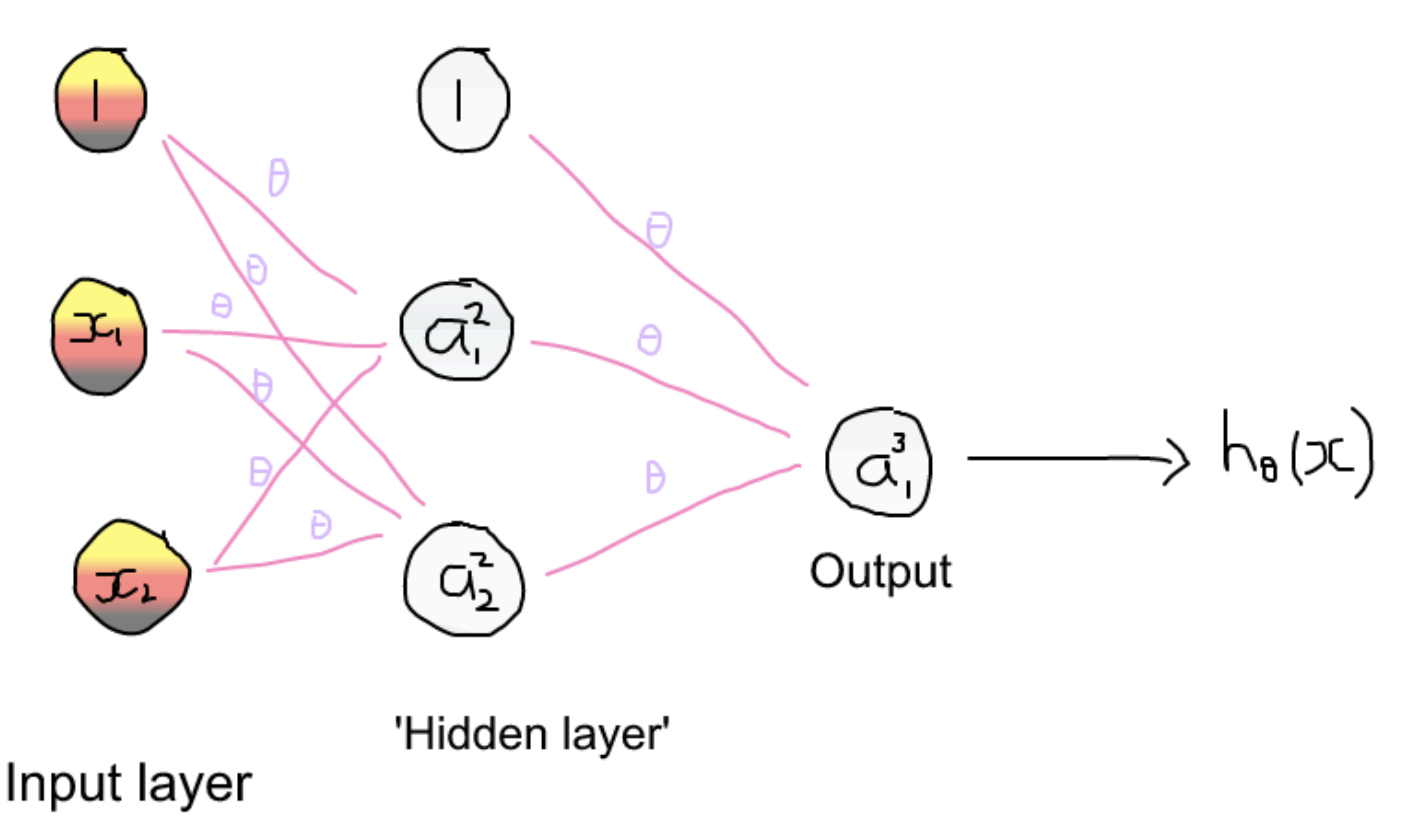
- Network
  - Feed forward — Cost functions
  - Back propagation — Training
    - Gradient descent
    - Learning rate
    - Regularisation

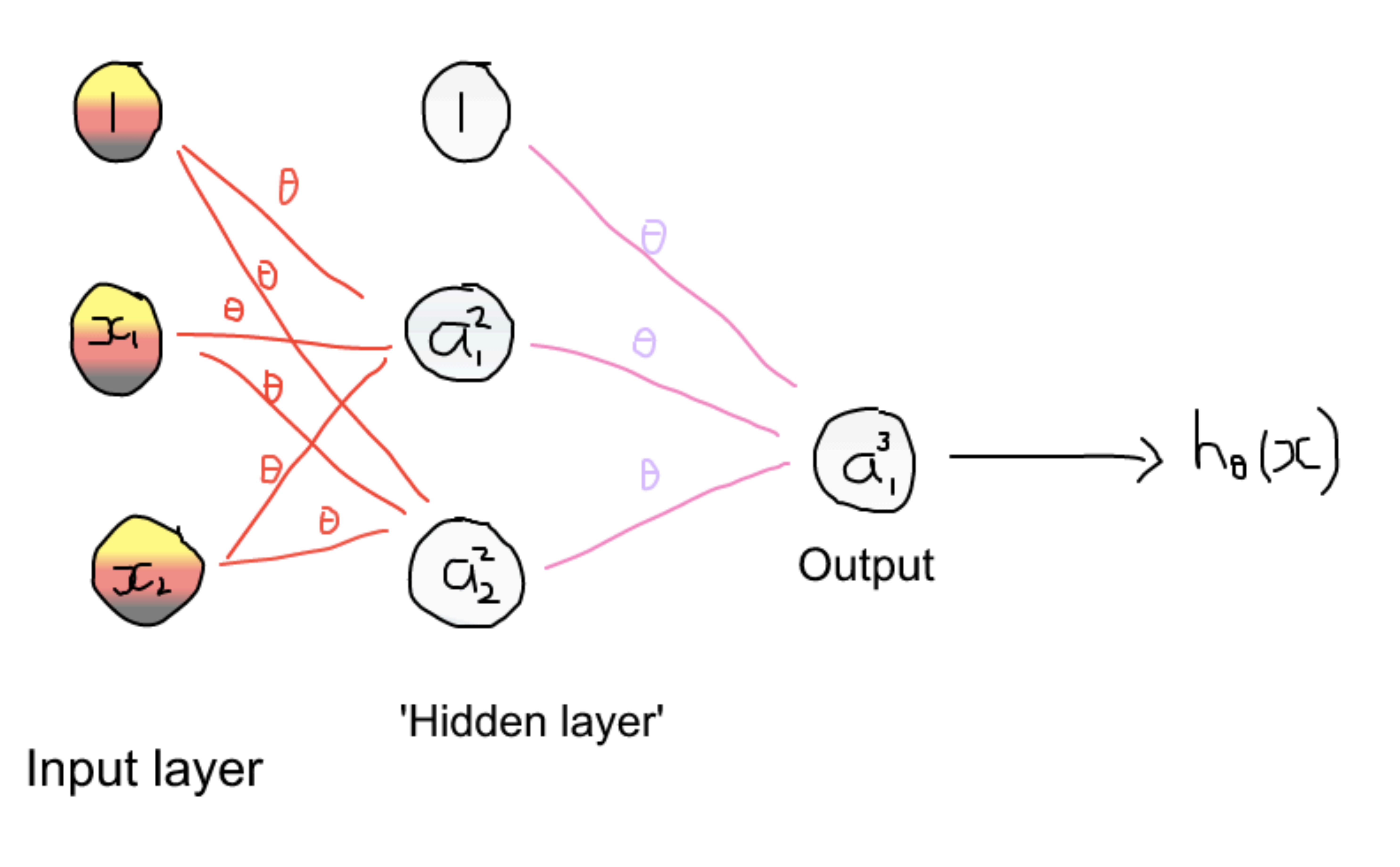
PART III: IN A NETWORK

- Learning to map input to output — Architectures
  - CNN
  - RNN
  - And beyond…

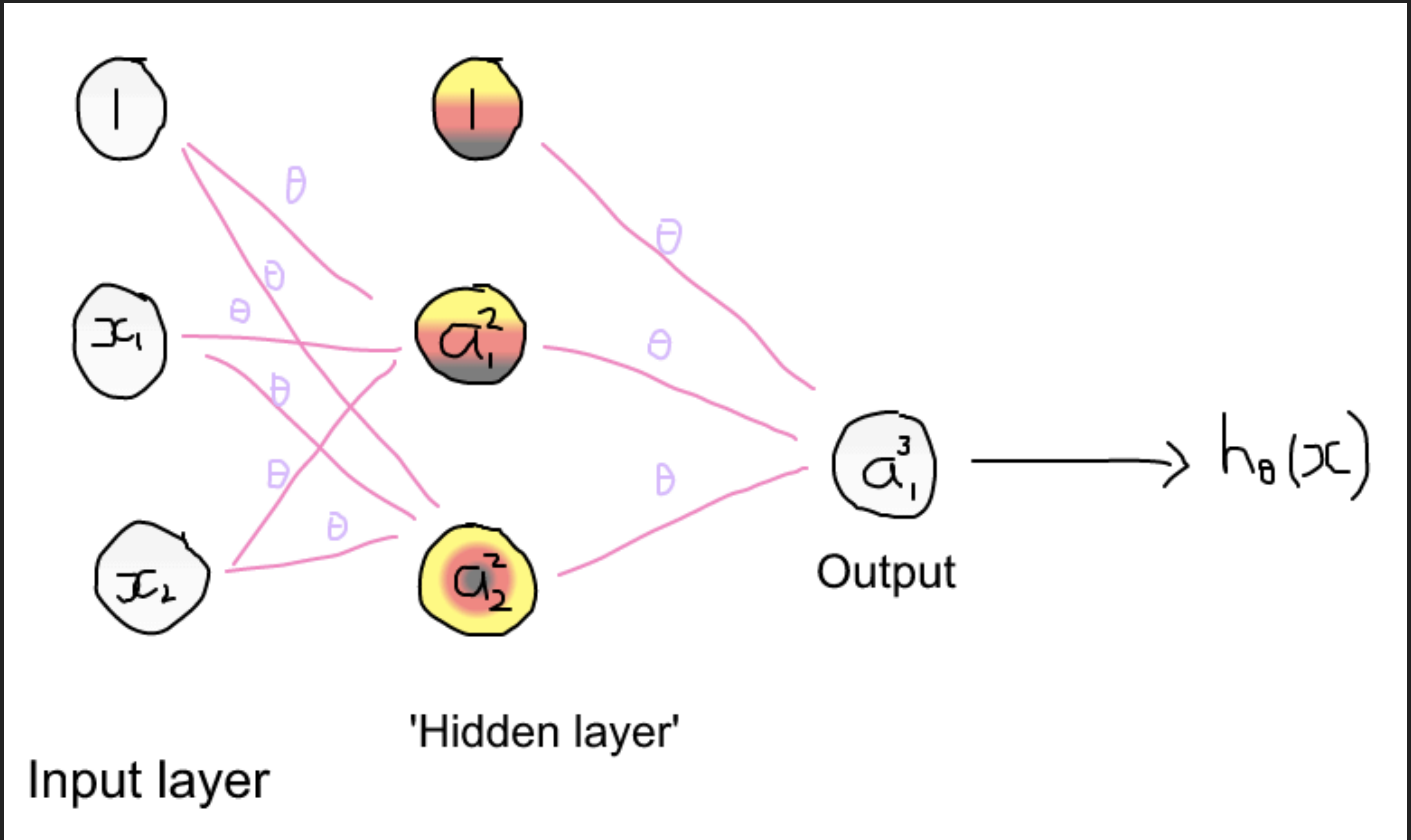
HOW MANY LAYERS DOES THIS NETWORK HAVE?

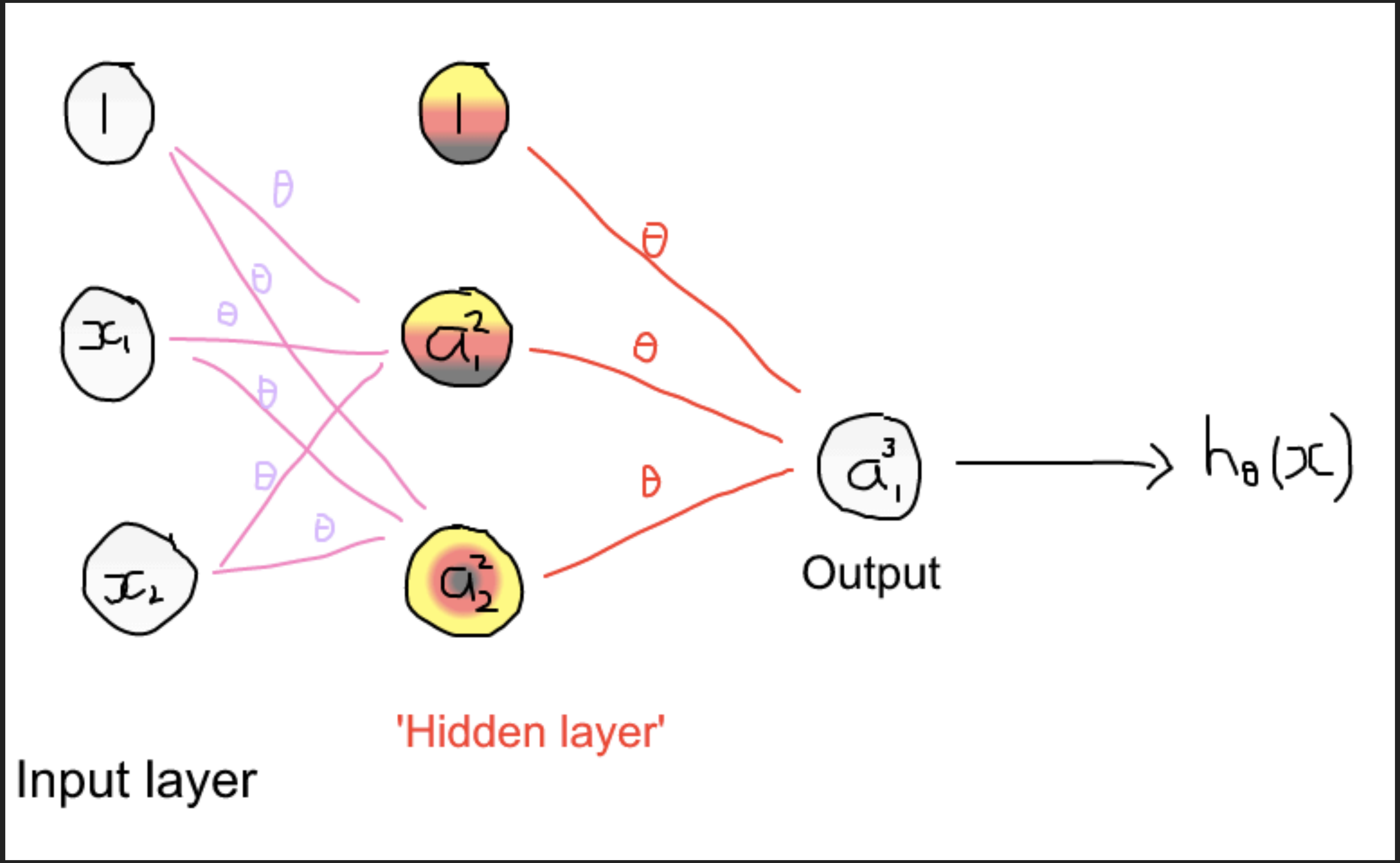


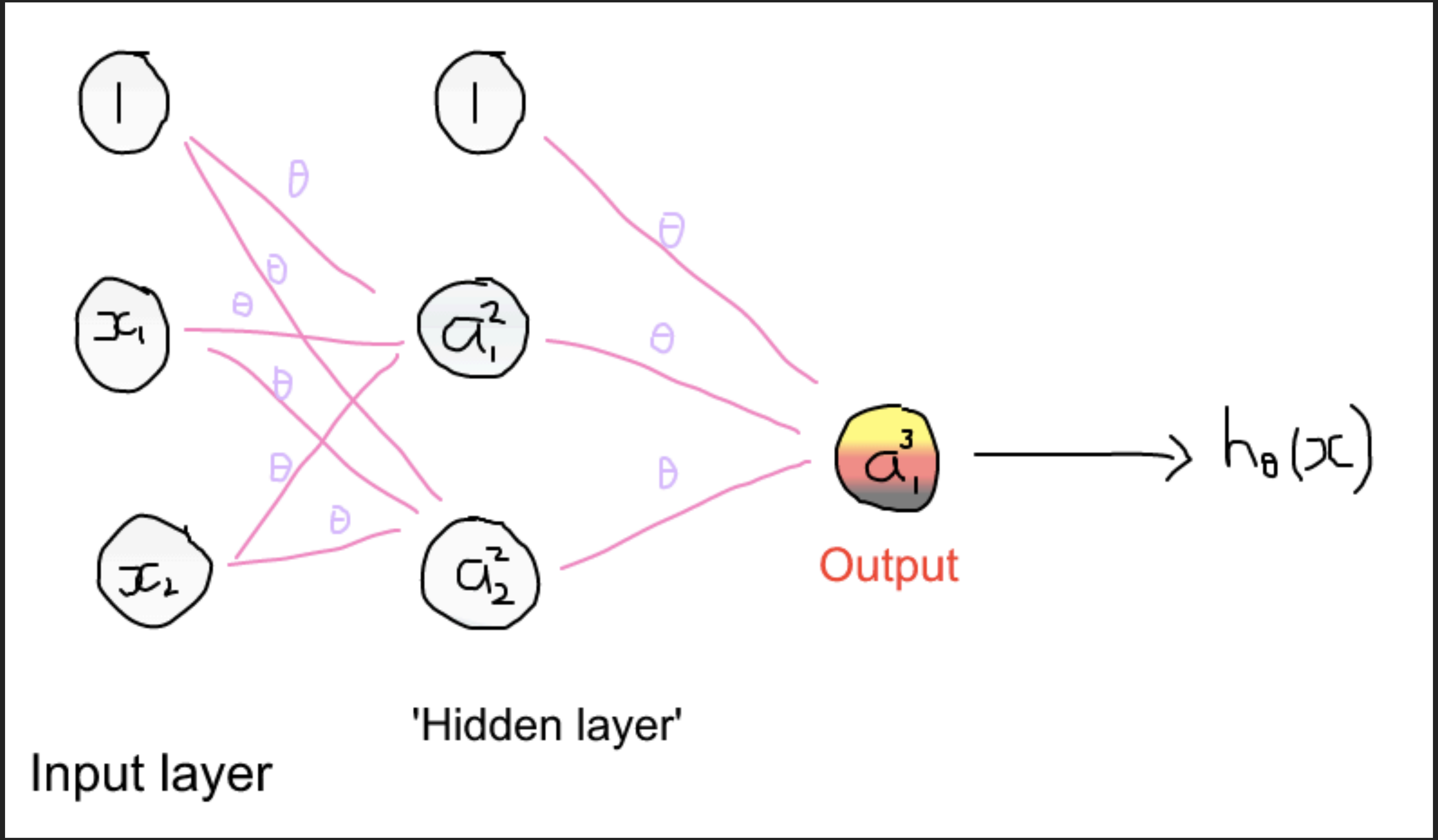

1
x_1
x_2
1
a_1^2
a_2^2
a_1^3
θ
$h_\theta(x)$
Output
'Hidden layer'
Input layer



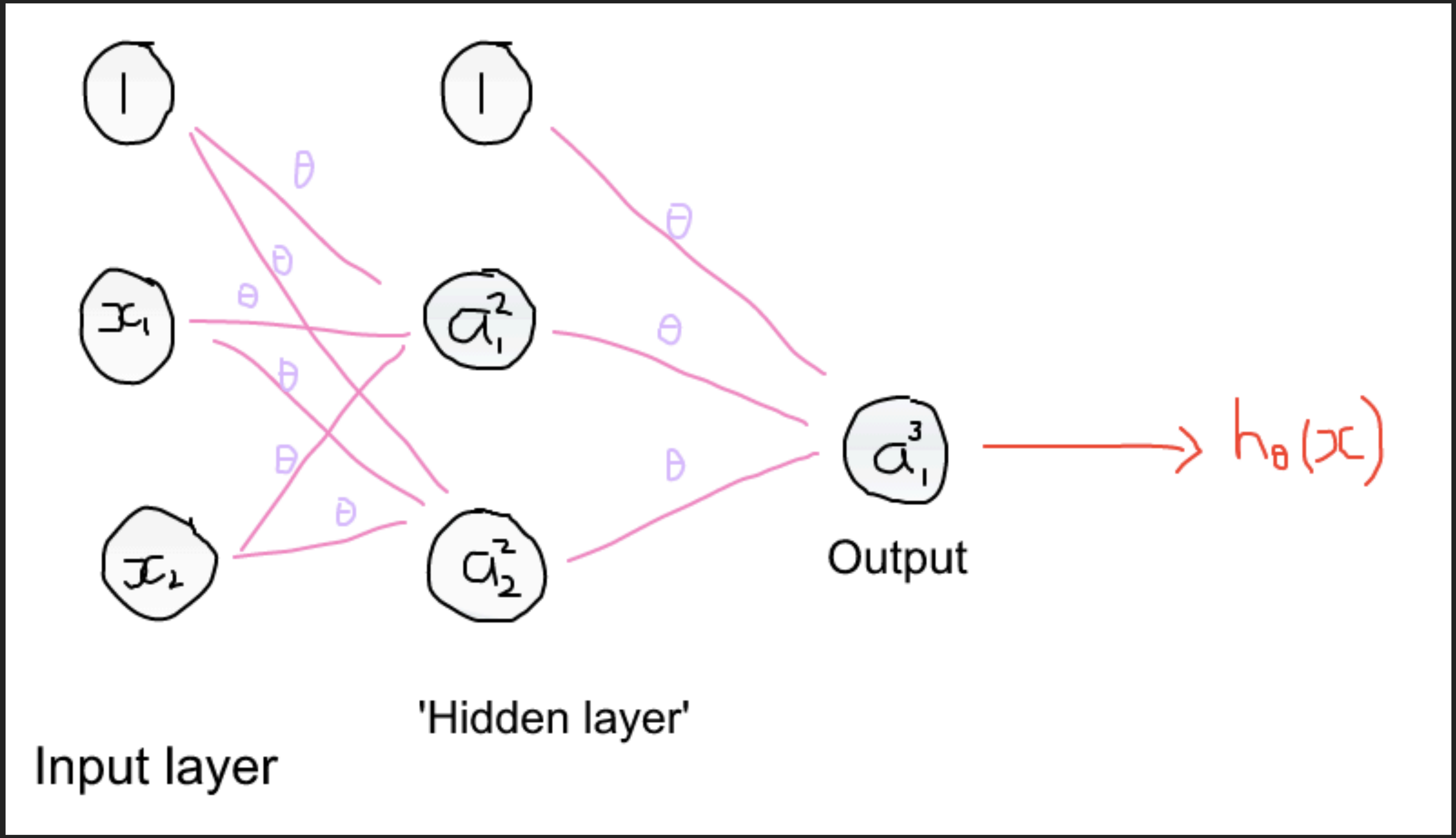
1
x_1
x_2
1
a_1^2
a_2^2
a_1^3
$h_\theta(x)$
Output
'Hidden layer'
Input layer


1
x_1
x_2
1
a_1^2
a_2^2
a_1^3
$h_\theta(x)$
θ
Output
'Hidden layer'
Input layer


1
x_1
x_2
1
a_1^2
a_2^2
θ
a_1^3
$h_\theta(x)$
Output
'Hidden layer'
Input layer



1
1
x_1
a_1^2
x_2
a_2^2
θ
a_1^3
Output
$h_\theta(x)$
'Hidden layer'
Input layer


1
1
x_1
x_2
a_1^2
a_2^2
a_1^3
θ
$h_\theta(x)$
Output
'Hidden layer'
Input layer

PART I: NEURONS

- **Neurons**
  - **Biological**
  - **Artificial**
    - **Linear Neuron**
    - **Binary Threshold Neuron**
    - **Logistic/sigmoid Neuron**
    - **reLU**
    - **Tanh**

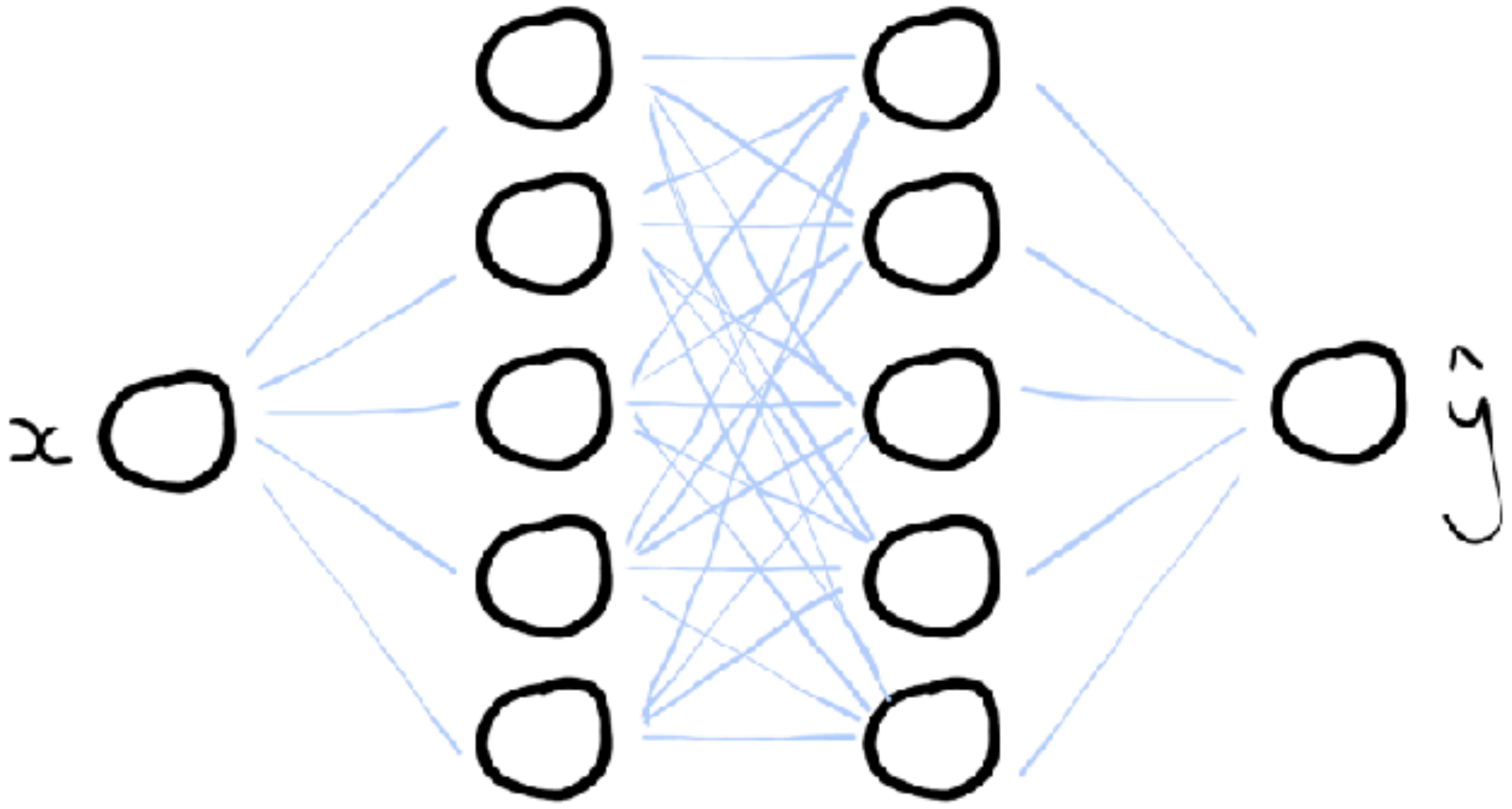
PART II: LEARNING

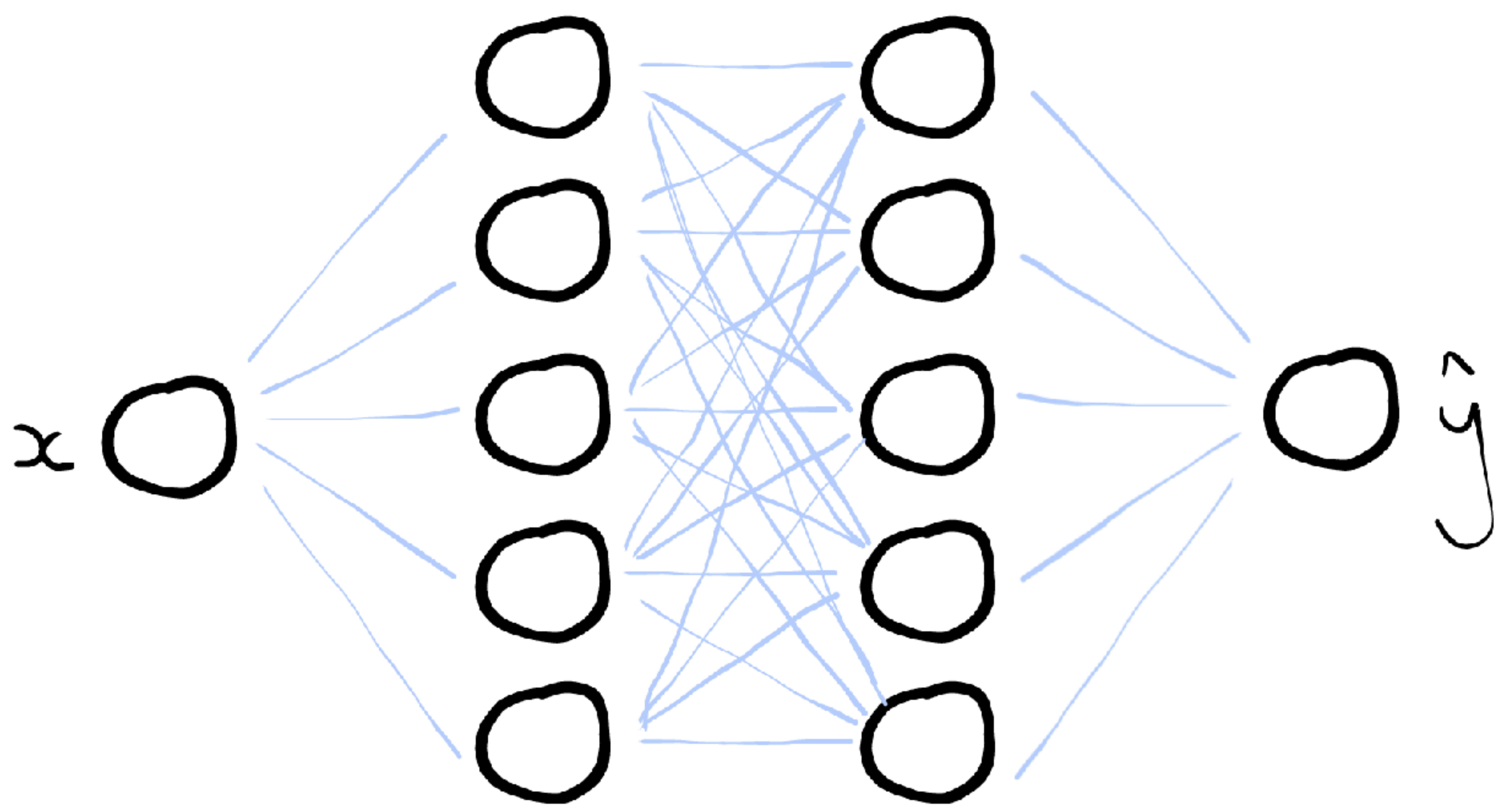
- **Network**
  - **Feed forward** — Cost functions
  - Back propagation — Training
    - Gradient descent
    - Learning rate
    - Regularisation

PART III: IN A NETWORK

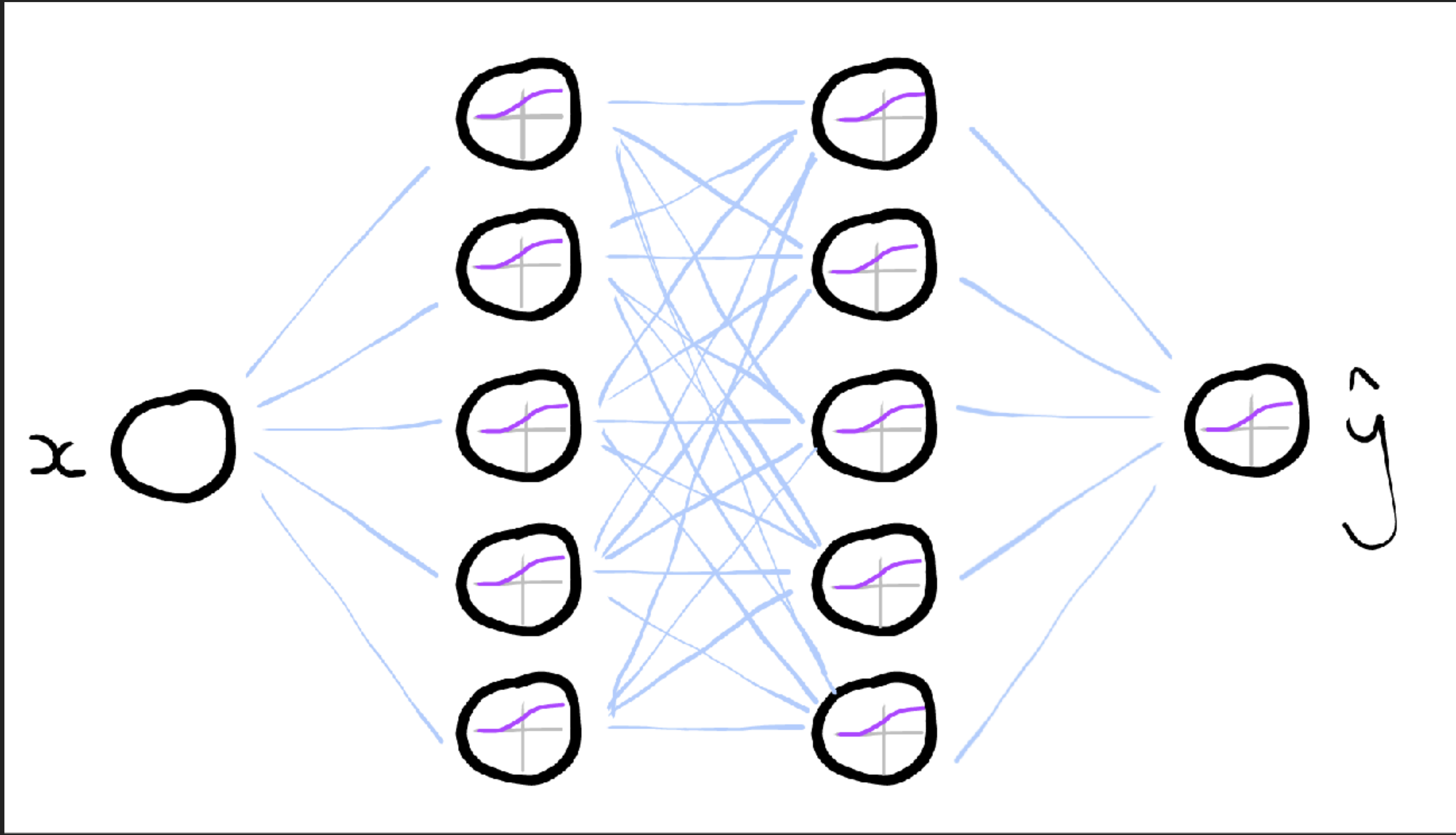
- Learning to map input to output — Architectures
  - CNN
  - RNN
  - And beyond…

HOW MANY LAYERS DOES THIS NETWORK HAVE?

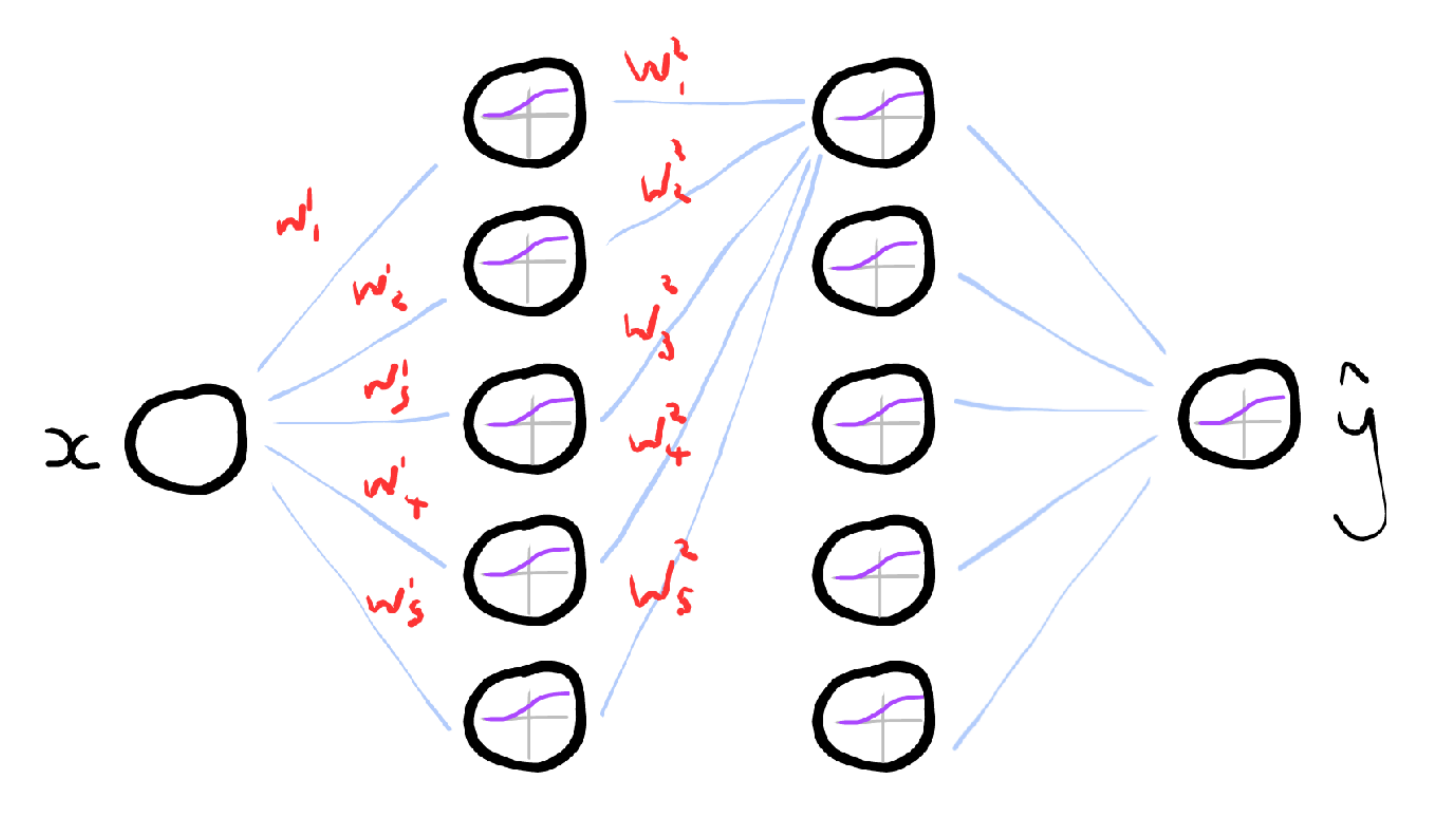




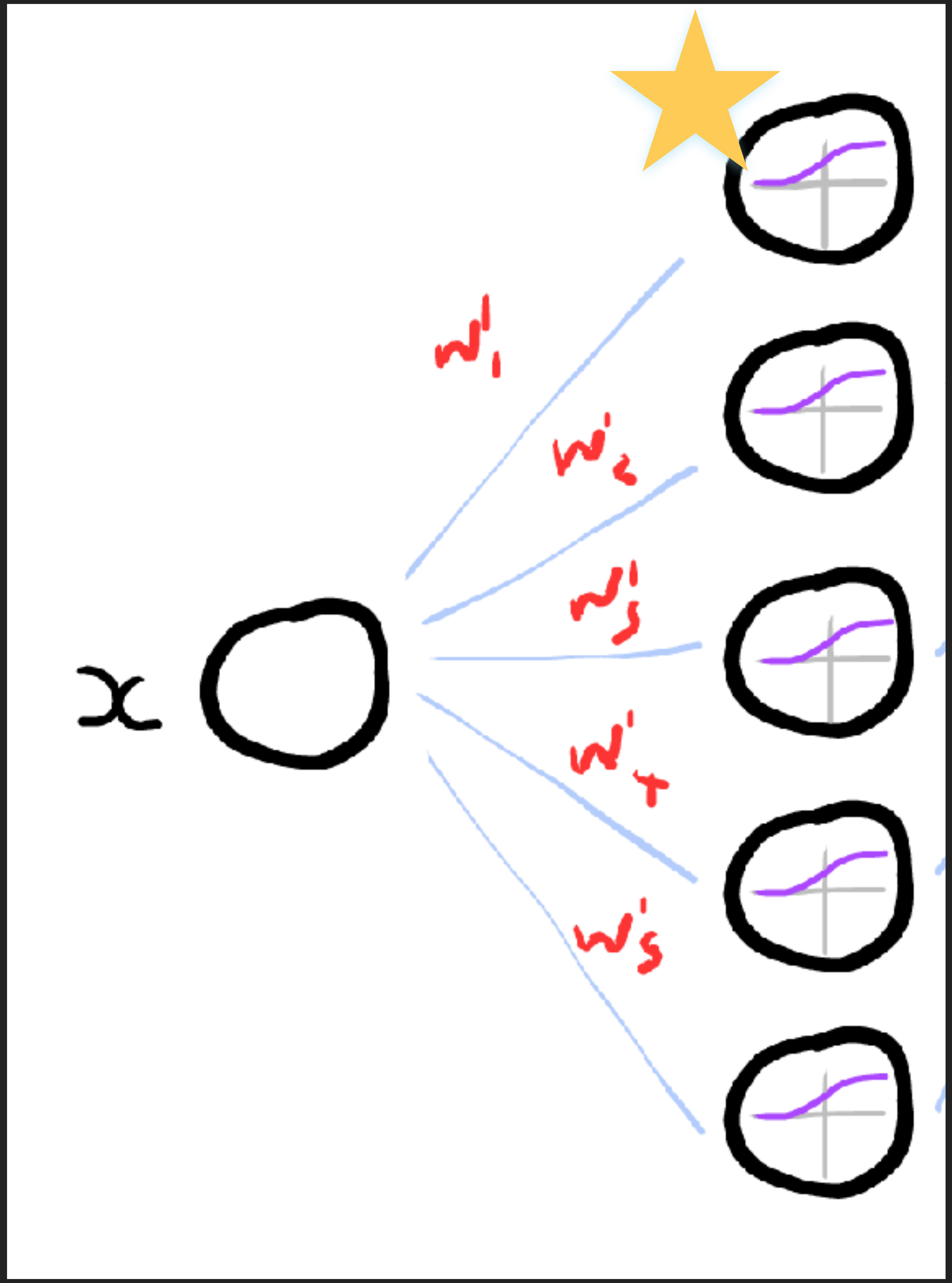
x
ŷ

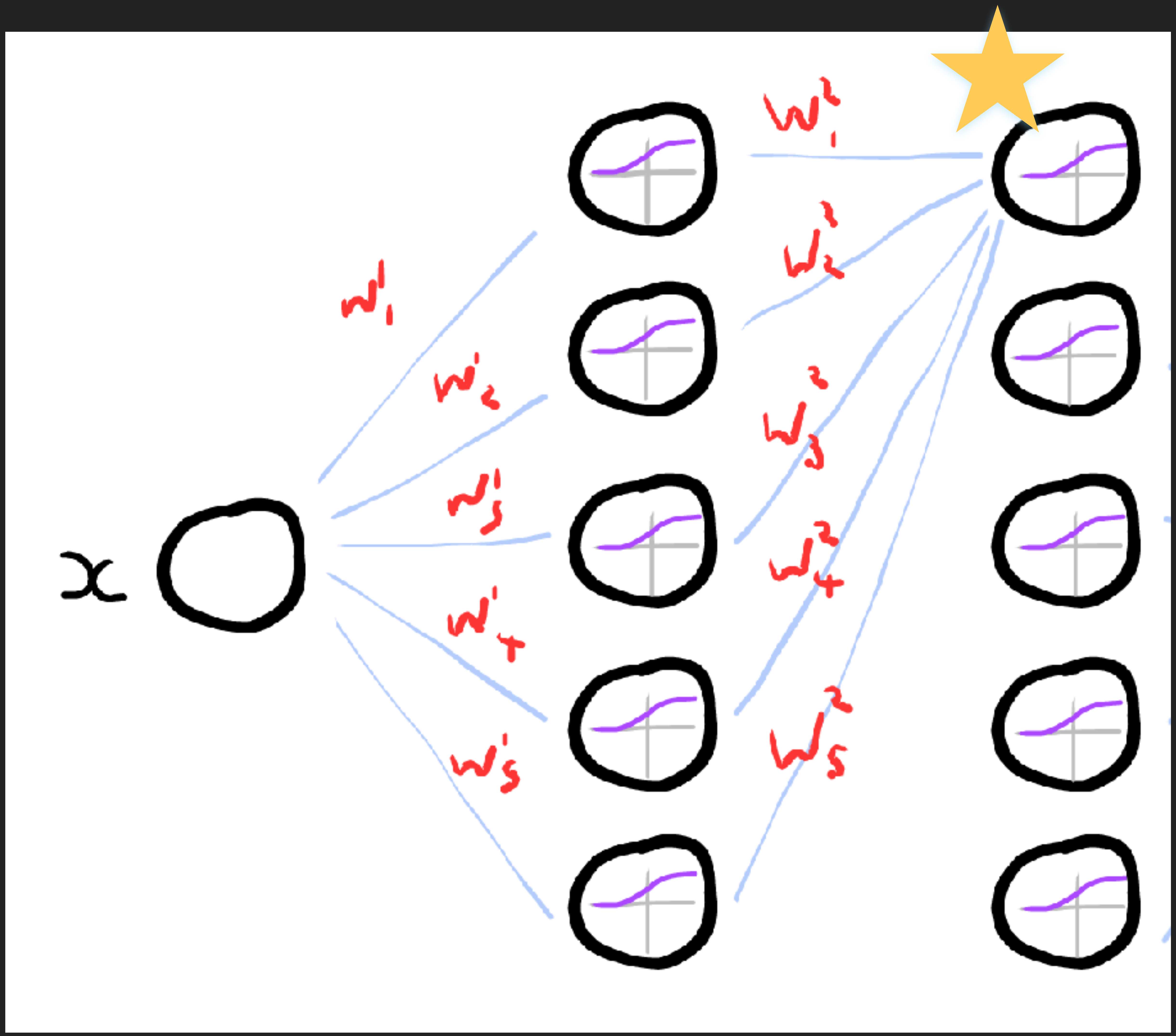


x
$\hat{y}$



x
w_1^1
w_2^1
w_3^1
w_4^1
w_5^1
w_1^2
w_2^2
w_3^2
w_4^2
w_5^2
$\hat{y}$

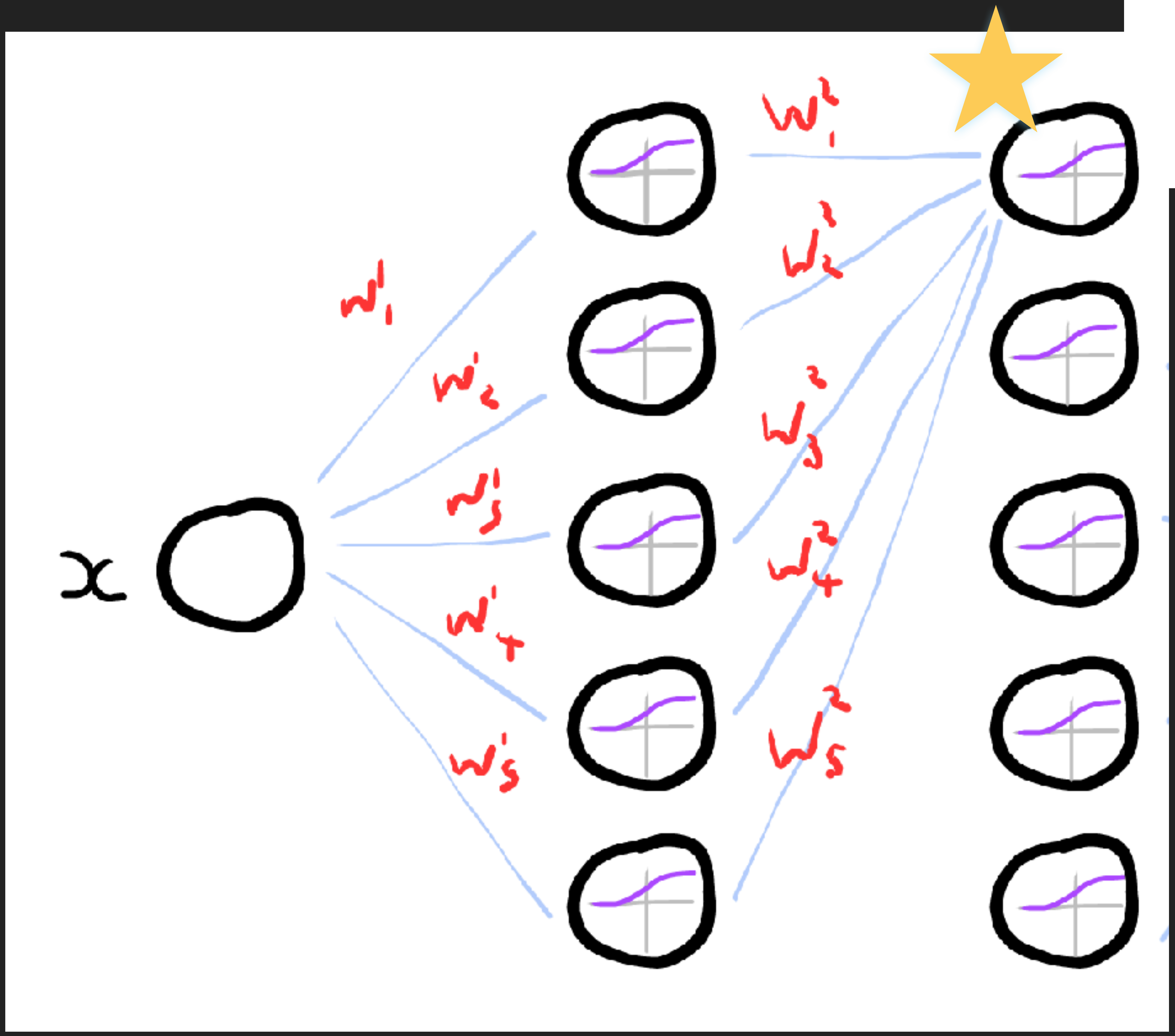

x
w'_1
w'_2
w'_3
w'_4
w'_5

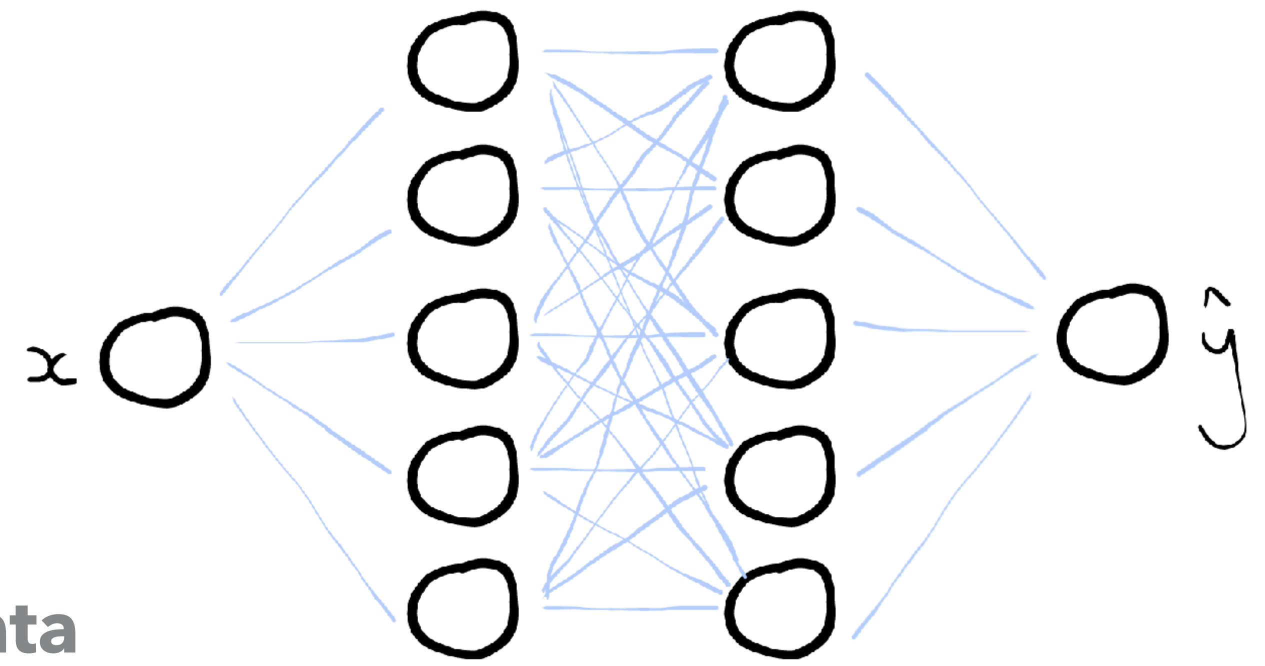


x

$$Z = a_1 W_1^2 + a_2 W_2^2 + a_3 W_3^2 + a_4 W_4^2 + a_5 W_5^2$$

$$a_1^2 = \frac{1}{1 + e^{-Z}}$$





Data

100 data points

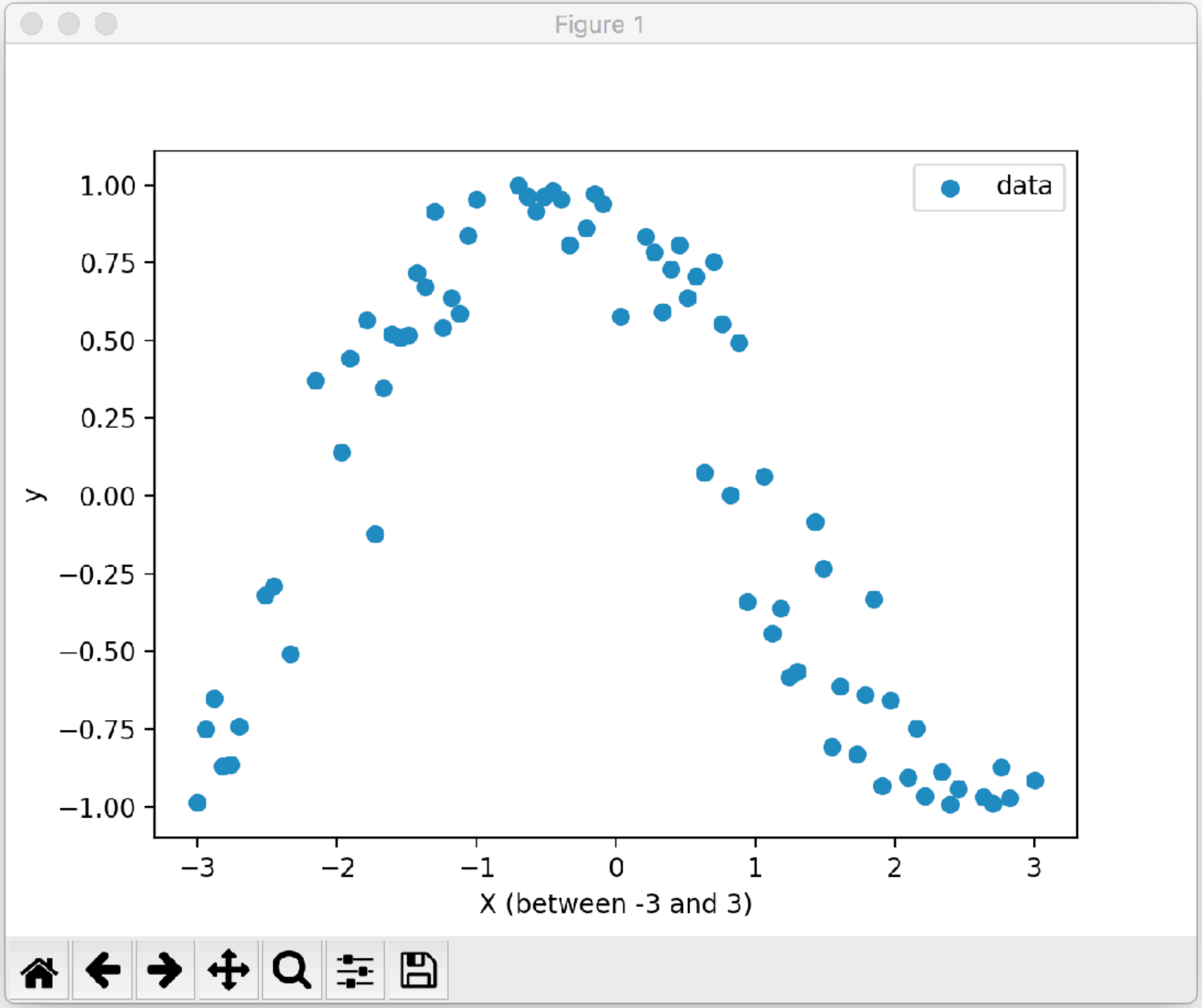
Inputs are between -3 and 3

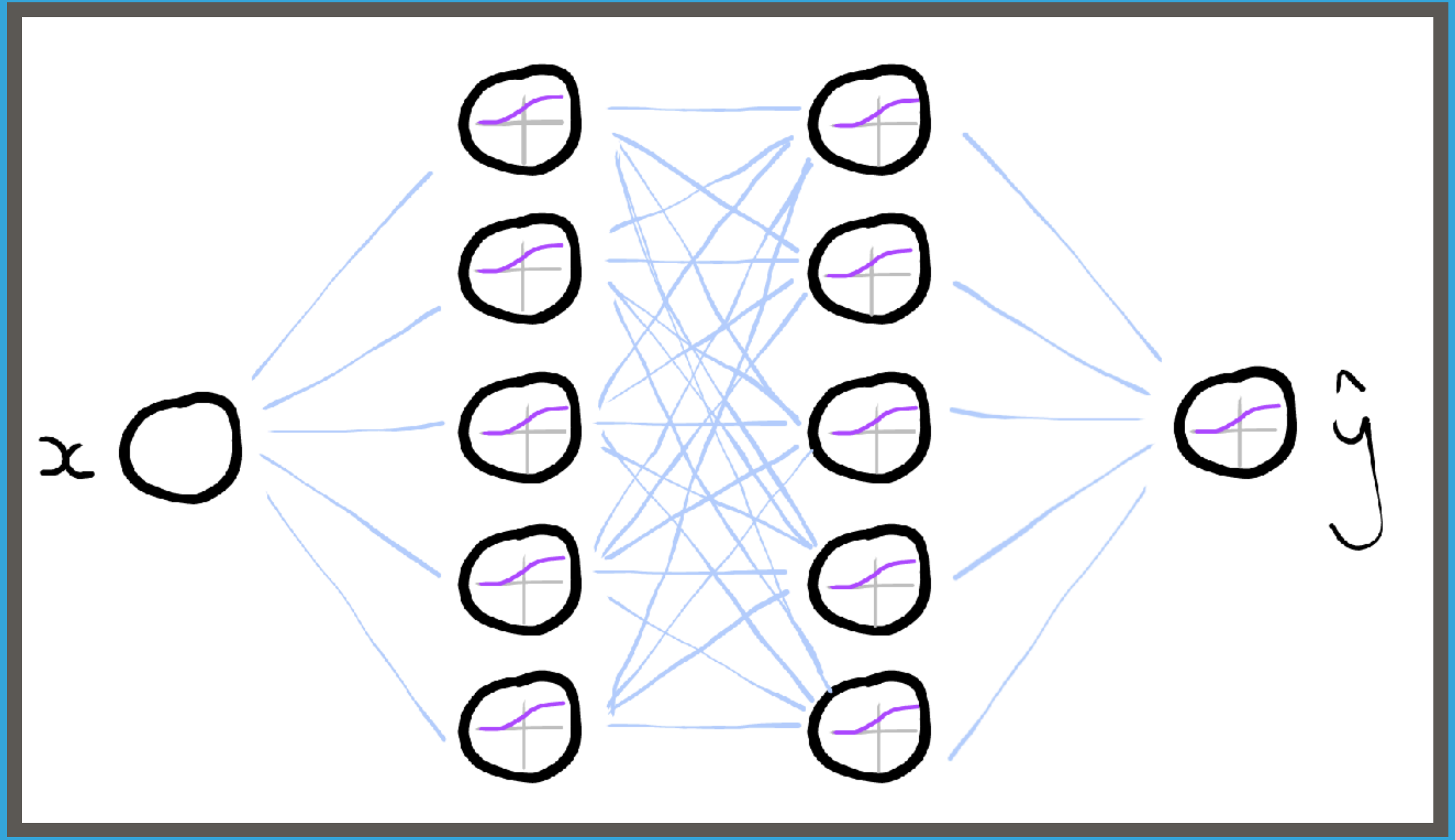
Output is a function y = cos(X + random noise)

Network

2 hidden layers, each with 5 hidden units

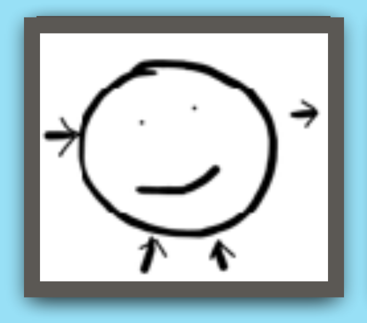
Logistic activation function



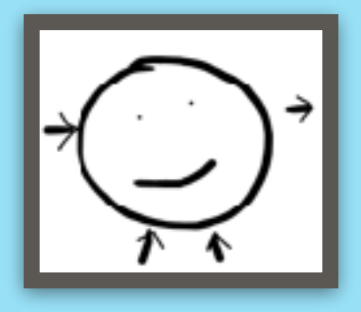


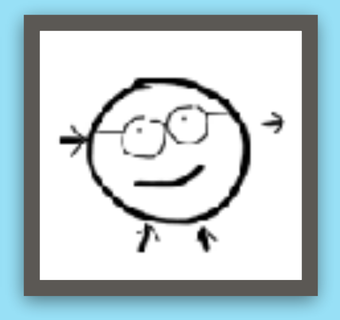
Feed Forward

MATHS REQUIRED



DIFFICULTY





Move forwards through network to get the output

COST FUNCTIONS

PART I: NEURONS

Neurons

- Biological
- Artificial
  - Linear Neuron
  - Binary Threshold Neuron
  - Logistic/sigmoid Neuron
  - reLU
  - Tanh

PART II: LEARNING

Network

- Feed forward — Cost functions
- Back propagation — Training
  - Gradient descent
  - Learning rate
  - Regularisation

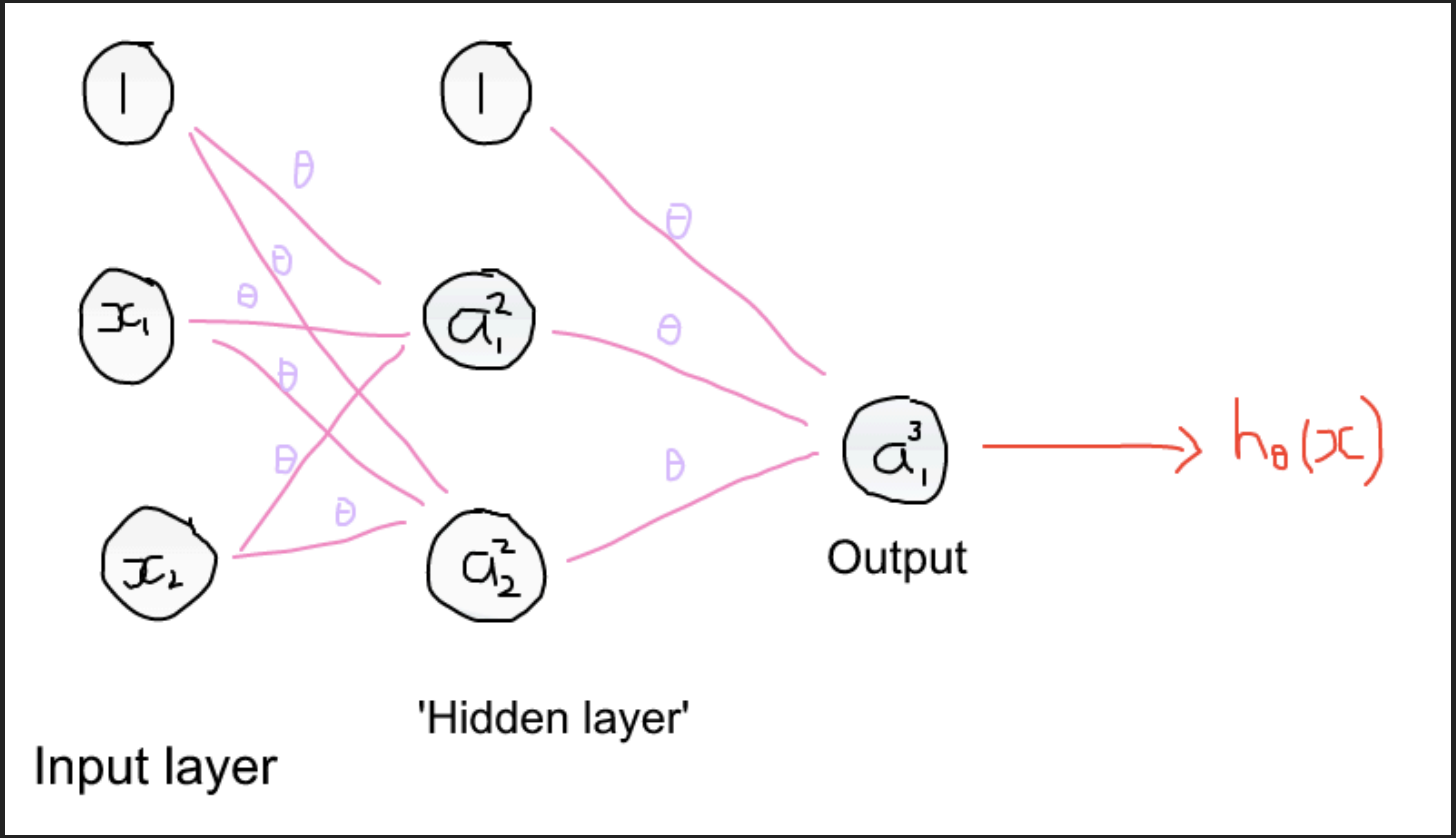
PART III: IN A NETWORK

Learning to map input to output — Architectures

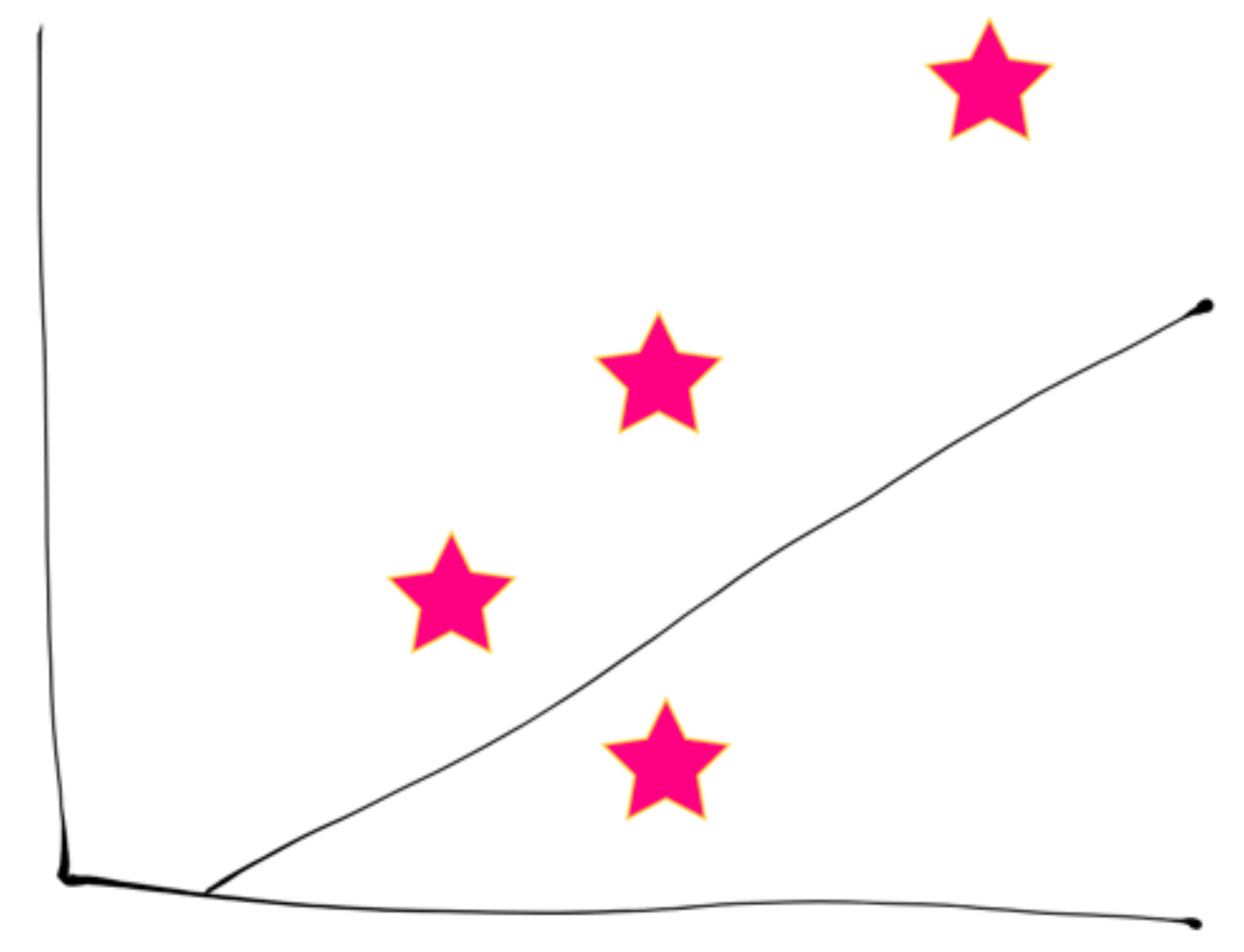
- CNN
- RNN
- And beyond…

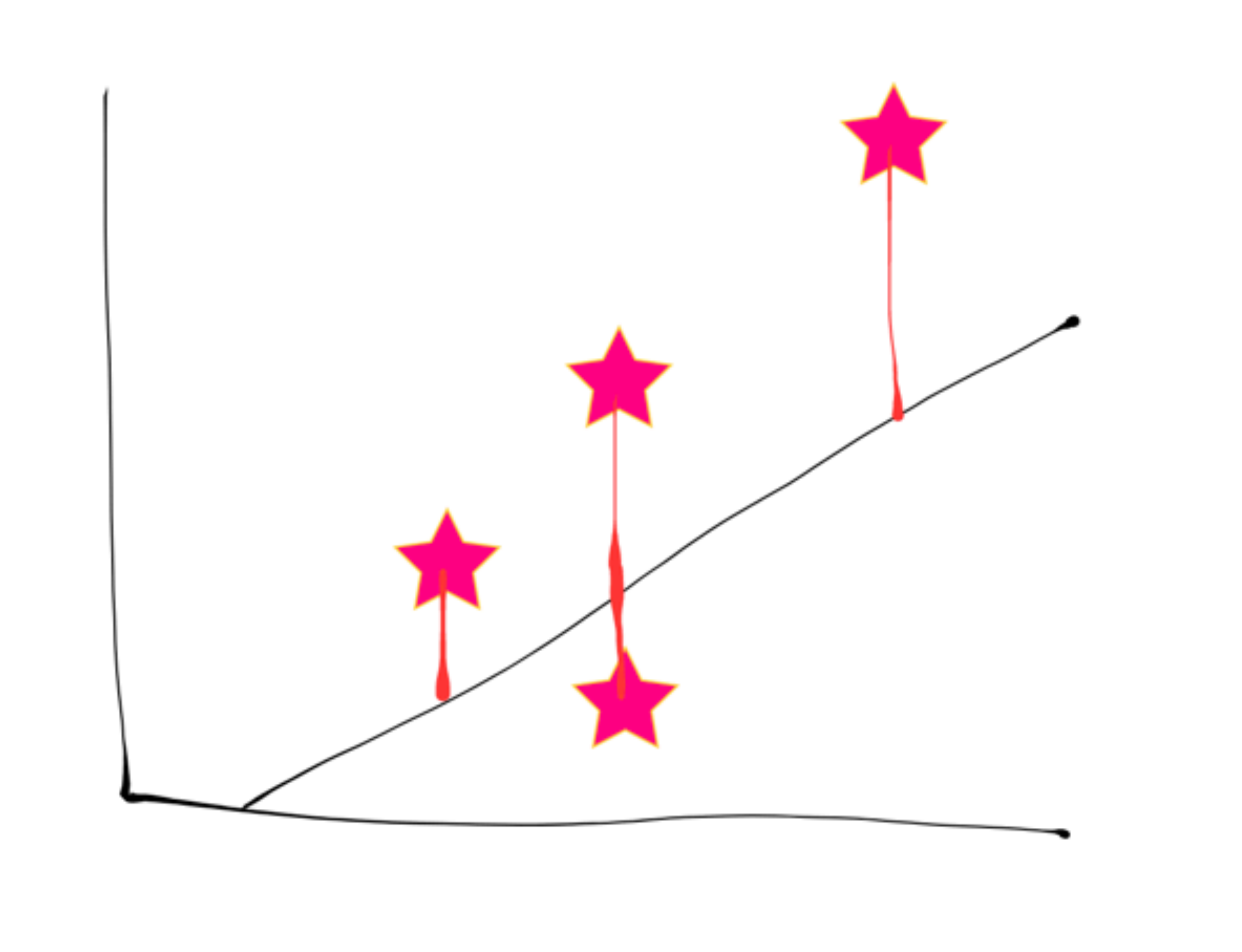
THE COST OF A NETWORK: COST FUNCTION

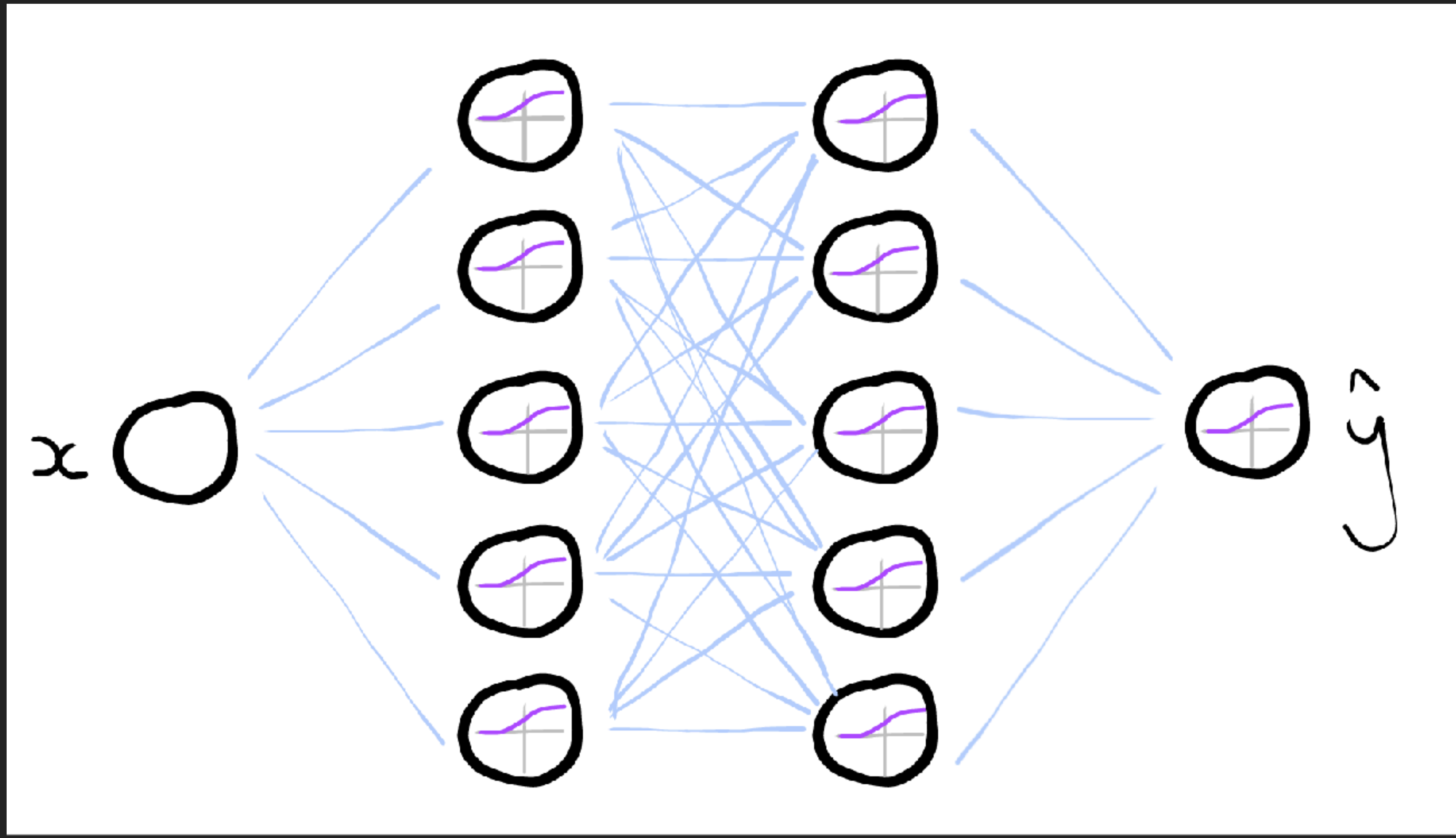
- What is the discrepancy between what the network output, and what the ‘real’ answer is?
- Different ways to calculate this depending on the data:
  - For a real valued output, might use the mean squared error
  - For a probability distribution, might use softmax



$$\frac{1}{2m}\sum_{i=0}^{n}\left(h(x)-y\right)^2$$







$$y=1 \quad \text{Cost} = -\log(h(x))$$

$$y=0 \quad \text{Cost} = -\log(1 - h(x))$$

$$\frac{1}{m}\sum \text{Cost}(h(x), y)$$

BACK PROPAGATION

PART I: NEURONS

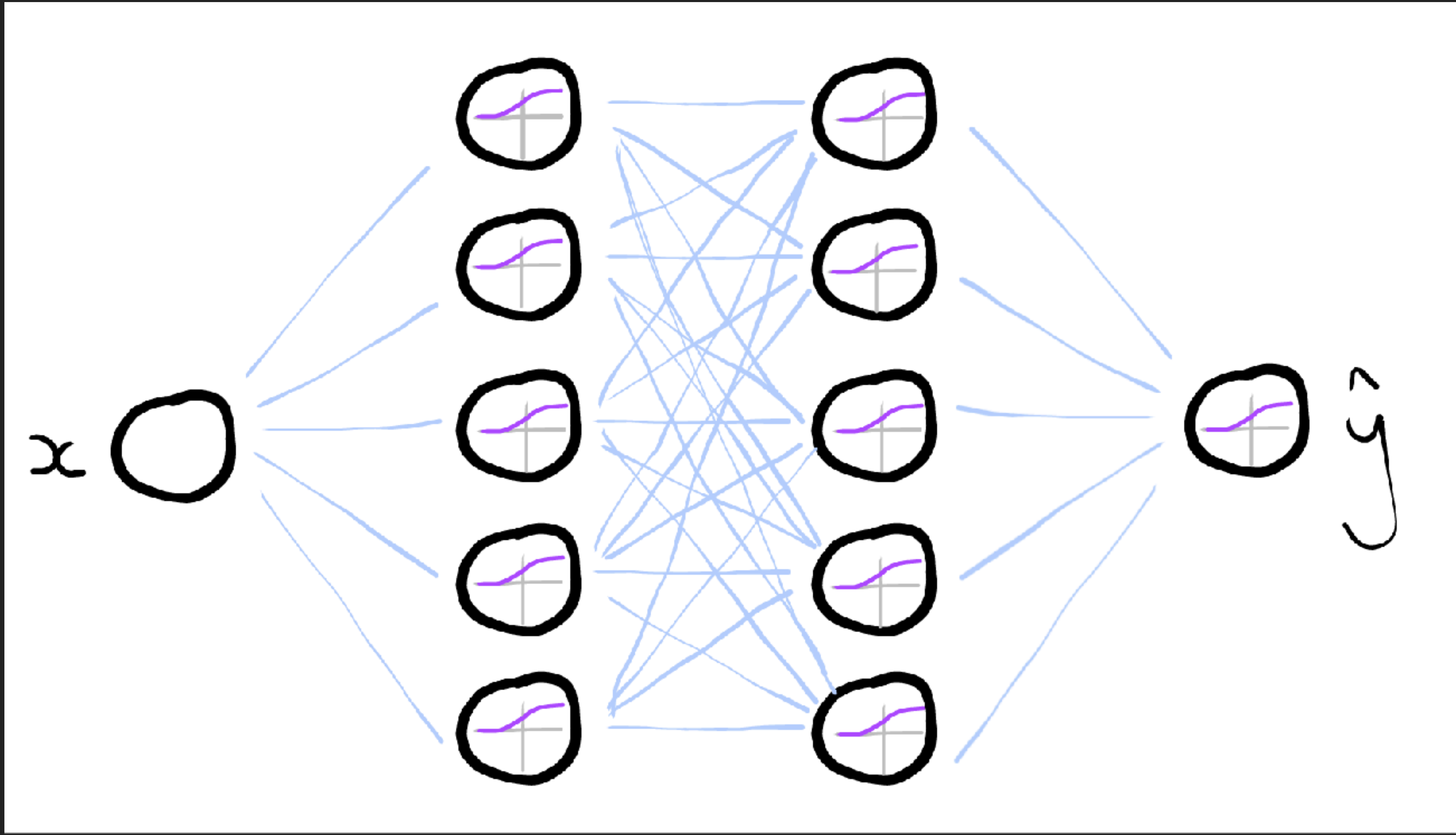
Neurons
- Biological
- Artificial
  - Linear Neuron
  - Binary Threshold Neuron
  - Logistic/sigmoid Neuron
  - reLU
  - Tanh

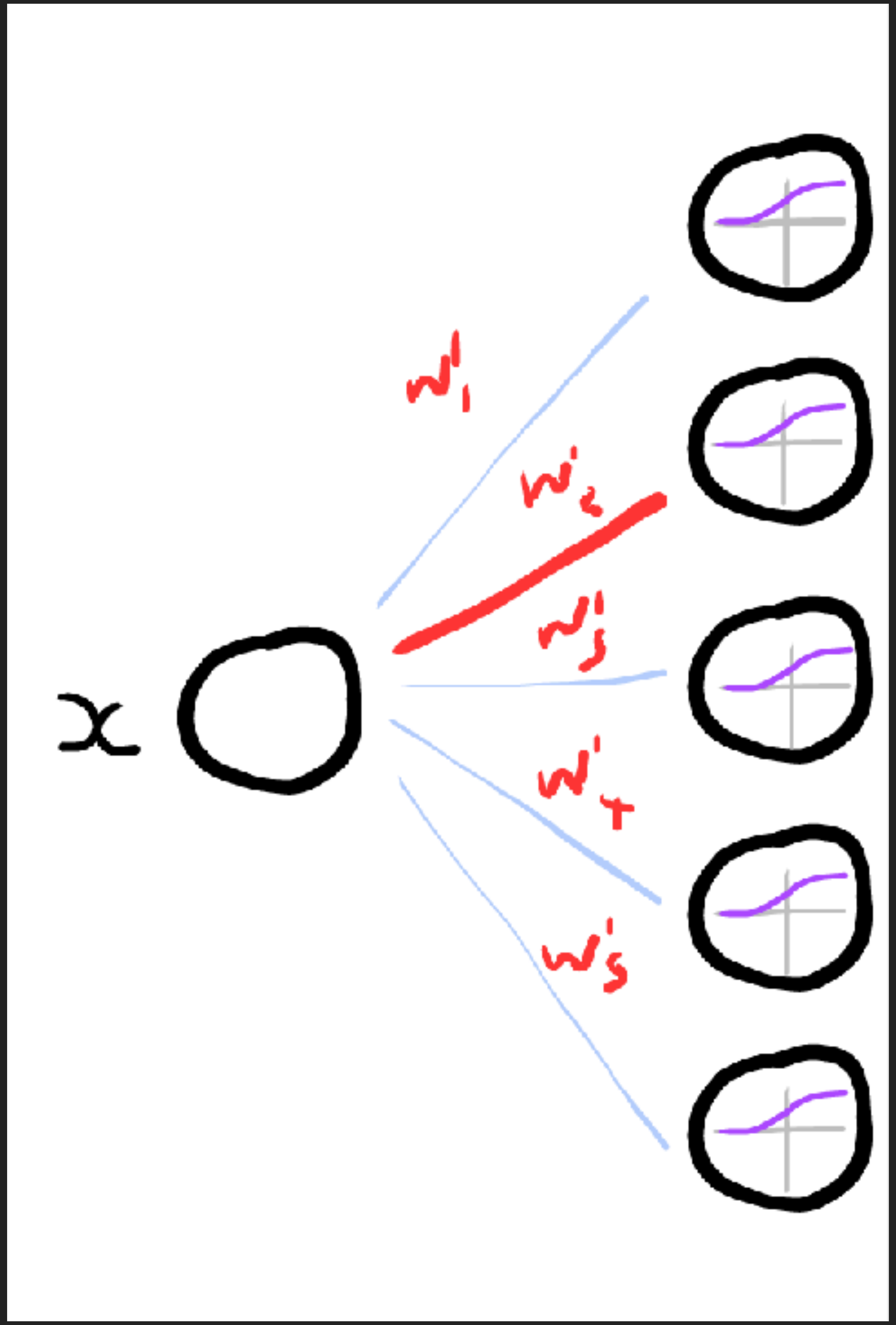
PART II: LEARNING

Network
- Feed forward — Cost functions
- Back propagation — Training
  - Gradient descent
  - Learning rate
  - Regularisation

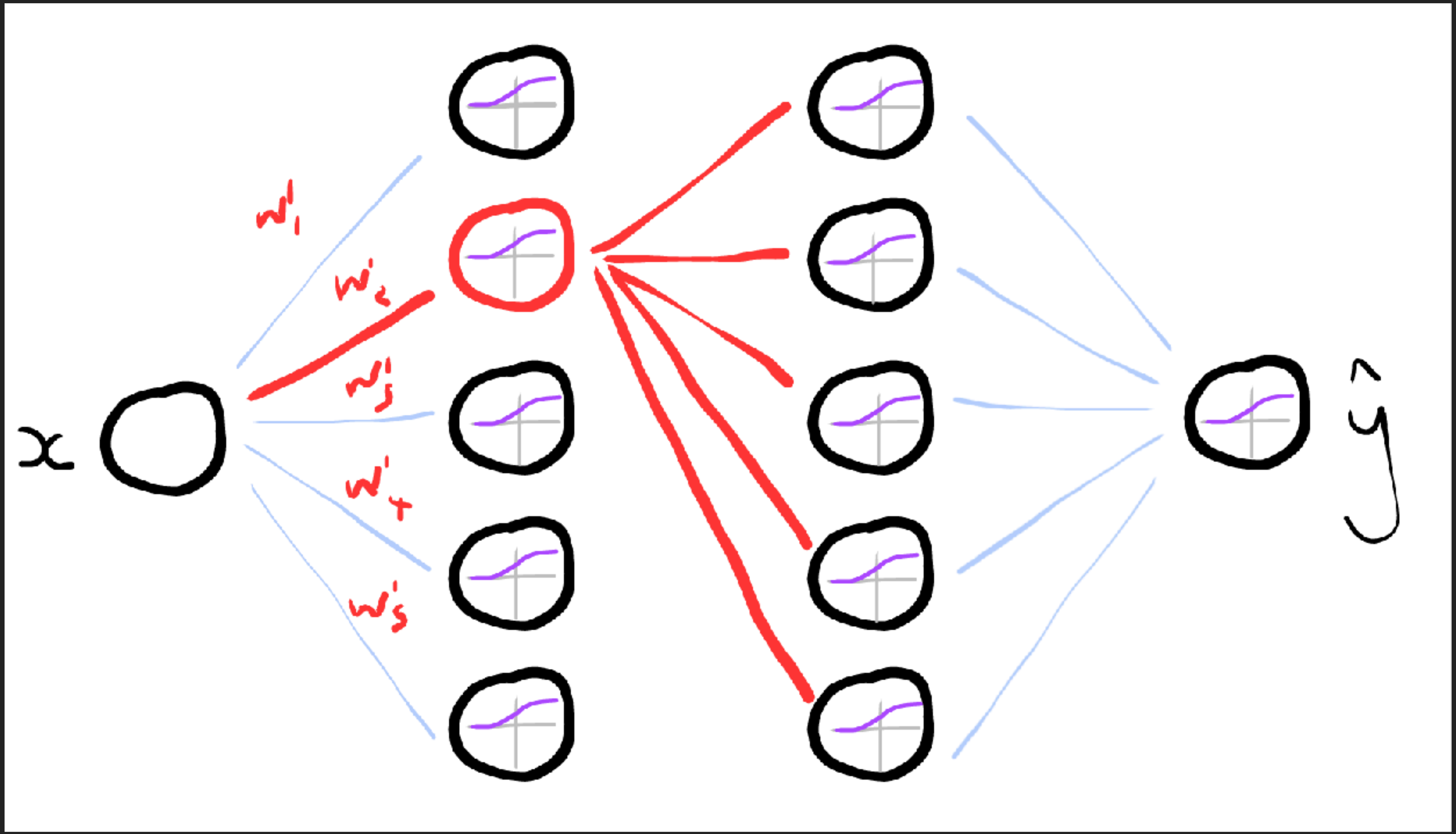
PART III: IN A NETWORK

Learning to map input to output — Architectures
- CNN
- RNN
- And beyond…

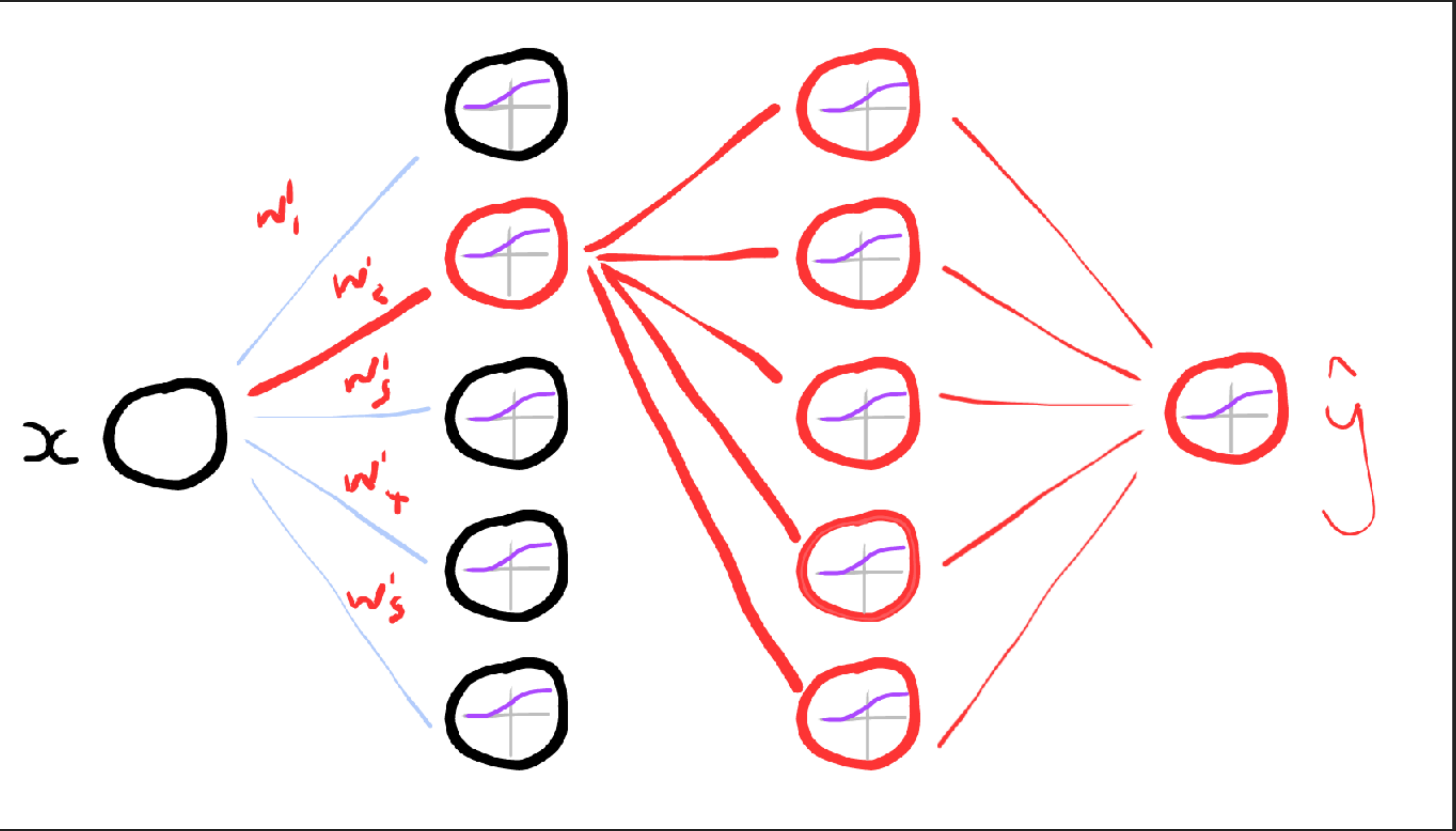

x
ŷ



x
w'1
w'2
w'3
w'4
w'5

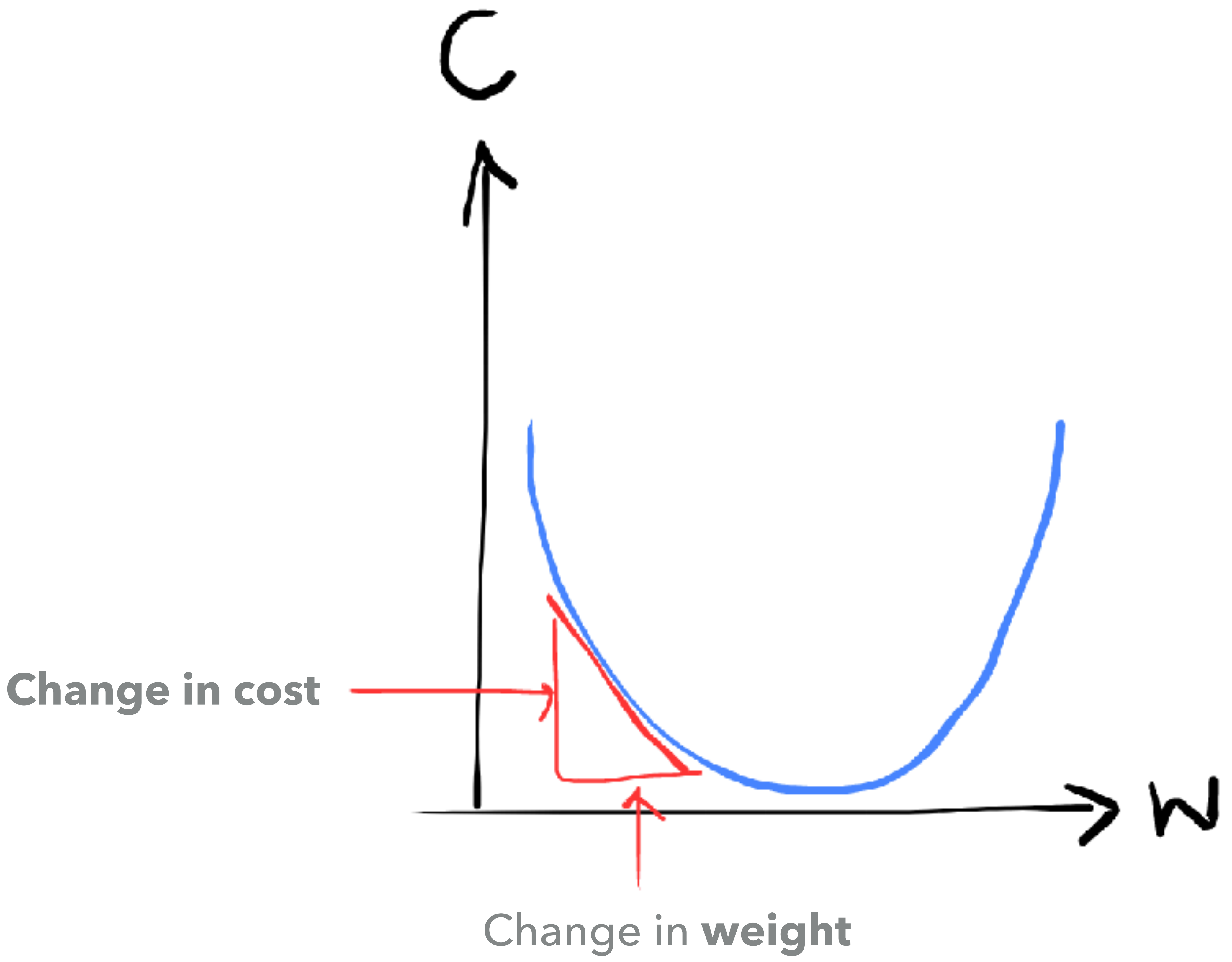


x
w_1^1
w_2^1
w_3^1
w_4^1
w_5^1
$\hat{y}$



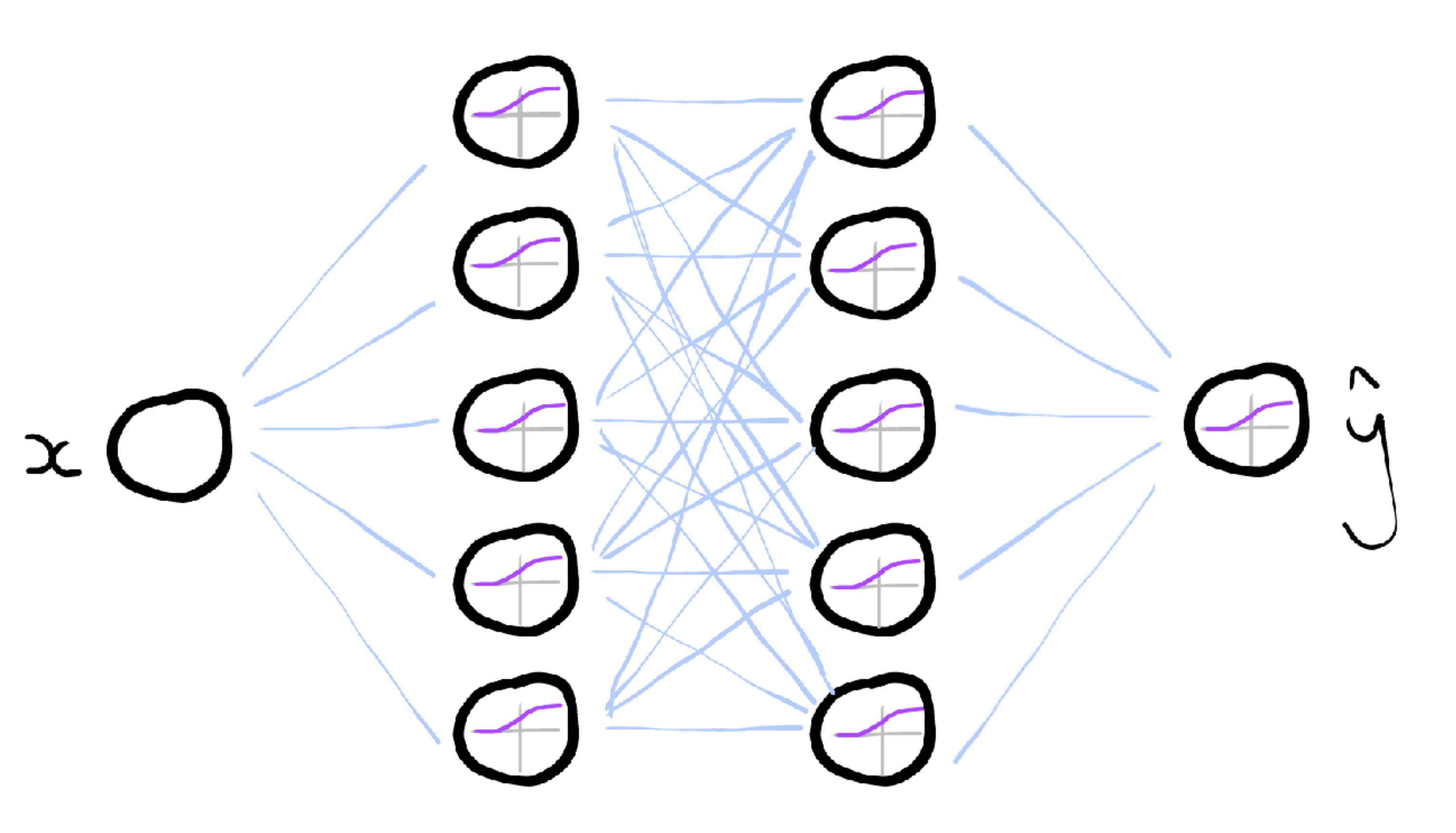
x
w'1
w'2
w'3
w'4
w'5
ŷ

GRADIENT DESCENT

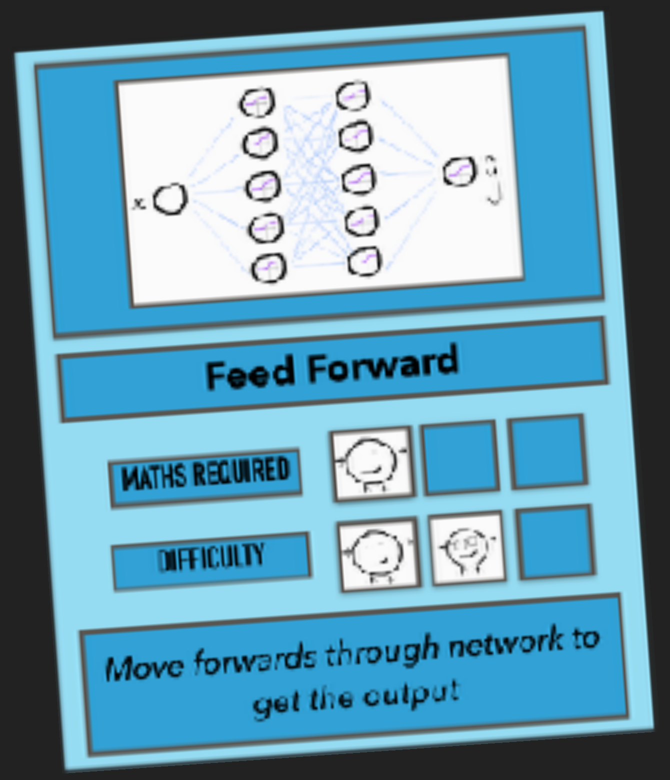


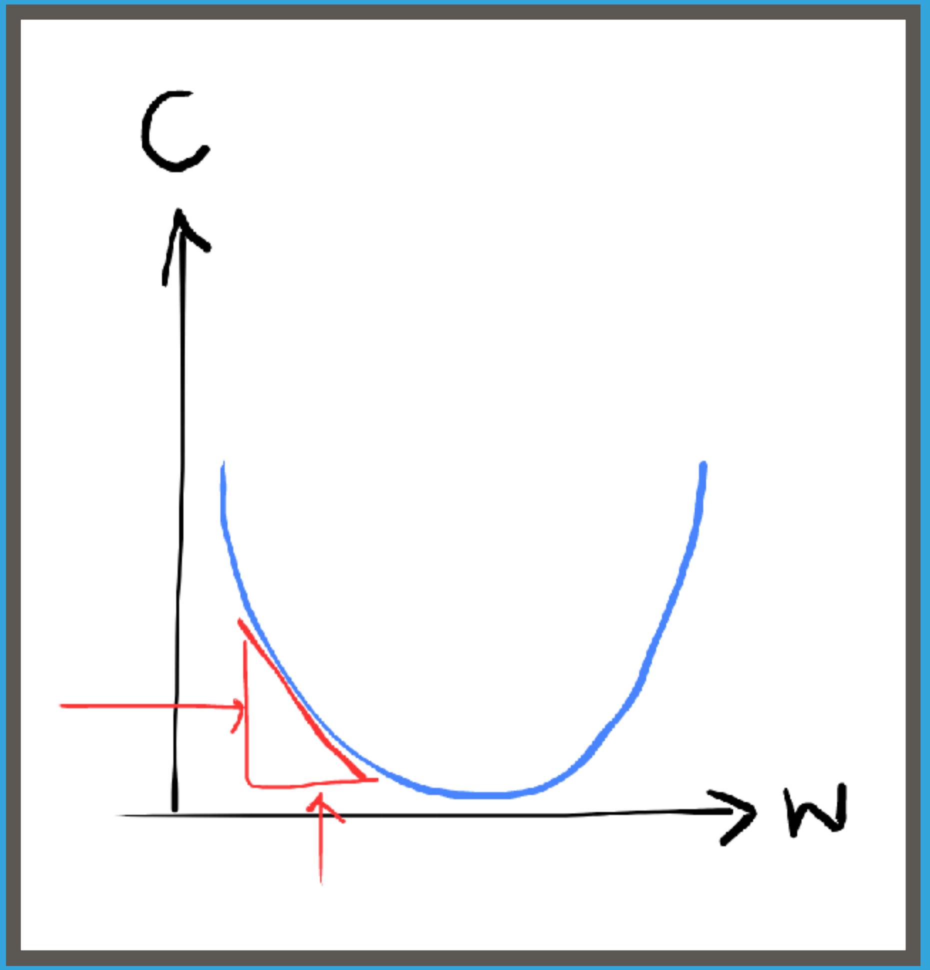
We want the **lowest cost function** for the **weights**.

BACK PROPAGATION



- Having made a forward pass through the network,
- And for each output unit, worked out how changing the output changes the cost…
- For the previous layer, work out how changing the input to the layer changes the output
- For the weight on the incoming connections to that layer, work out how changing the weight changes the cost
- Work backwards!

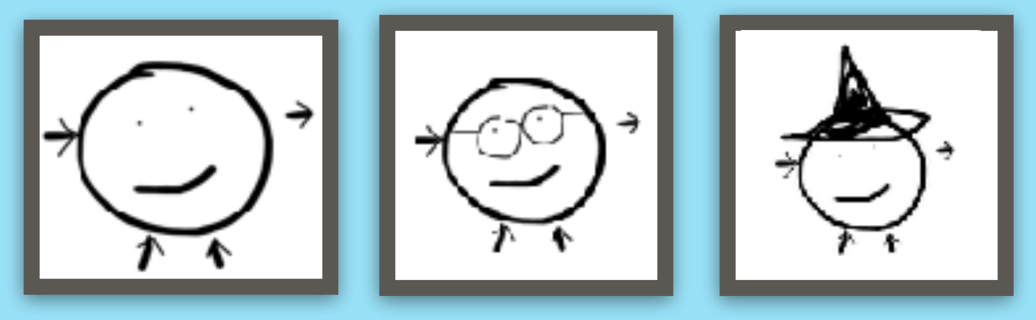

Feed Forward
MATHS REQUIRED
DIFFICULTY
Move forwards through network to get the output



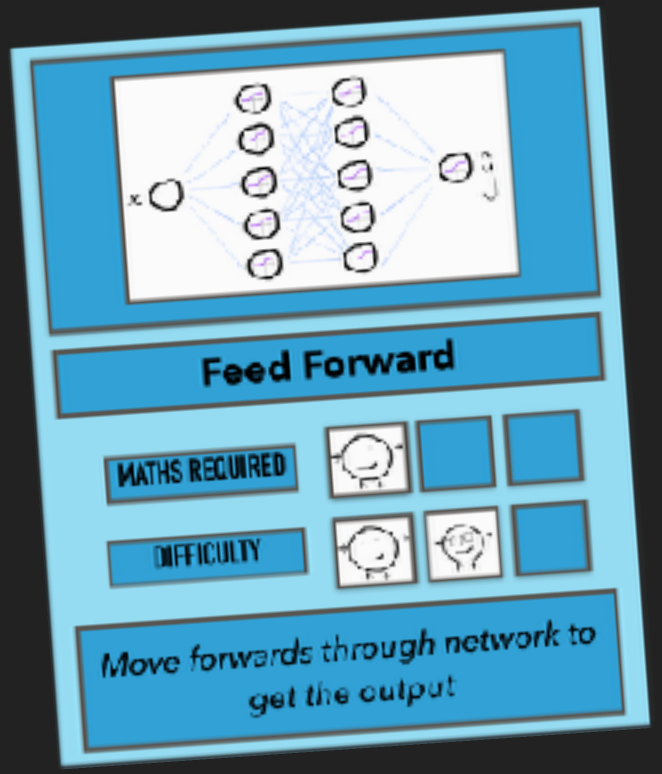
Back Propagation

MATHS REQUIRED

DIFFICULTY



Determine how the weights in the network affect the error



TRAINING

PART I: NEURONS

- **Neurons**
  - **Biological**
  - **Artificial**
    - **Linear Neuron**
    - **Binary Threshold Neuron**
    - **Logistic/sigmoid Neuron**
    - **reLU**
    - **Tanh**

PART II: LEARNING

- **Network**
  - **Feed forward** — **Cost functions**
  - **Back propagation** — **Training**
    - Gradient descent
    - Learning rate
    - Regularisation

PART III: IN A NETWORK

- Learning to map input to output — Architectures
  - CNN
  - RNN
  - And beyond…

- Randomly initialise weights
- Per example:
  - Forward pass through the network
  - Calculate the cost/error
  - Backward pass to get the error derivatives
- Run gradient descent

GRADIENT DESCENT

PART I: NEURONS

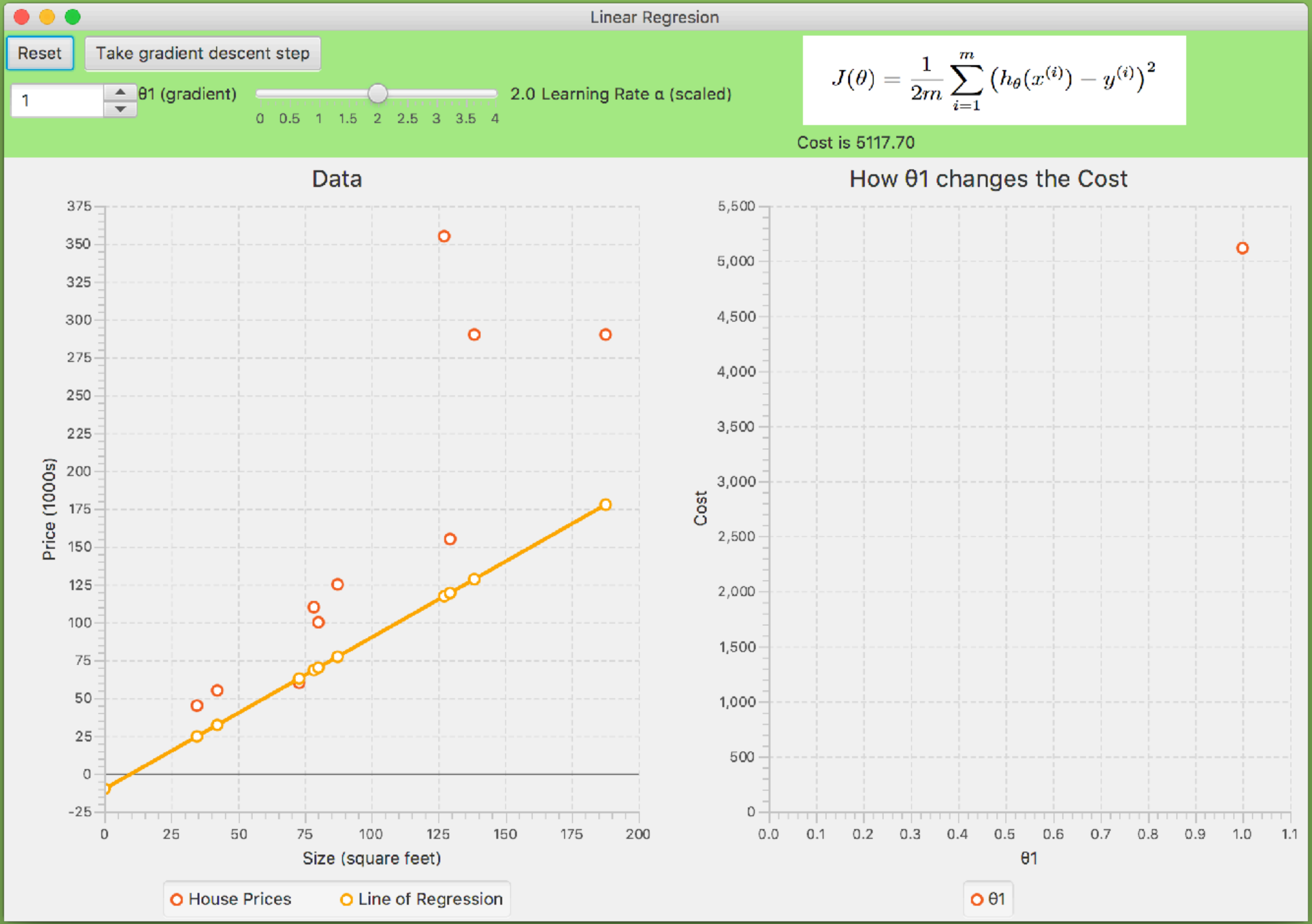
- **Neurons**
  - **Biological**
  - **Artificial**
    - **Linear Neuron**
    - **Binary Threshold Neuron**
    - **Logistic/sigmoid Neuron**
    - **reLU**
    - **Tanh**

PART II: LEARNING

- **Network**
  - **Feed forward** — **Cost functions**
  - **Back propagation** — **Training**
    - **Gradient descent**
    - Learning rate
    - Regularisation

PART III: IN A NETWORK

- Learning to map input to output — Architectures
  - CNN
  - RNN
  - And beyond…


Linear Regresion
Reset
Take gradient descent step
θ1 (gradient)
2.0 Learning Rate α (scaled)
$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$
Cost is 5117.70
Data
Price (1000s)
Size (square feet)
House Prices
Line of Regression
How θ1 changes the Cost
Cost
θ1

- Randomly initialise weights
- Per example:
  - Forward pass through the network
  - Calculate the cost/error
  - Backward pass to get the error derivatives
- Run gradient descent

WHEN SHOULD GRADIENT DESCENT BE RUN?

DIFFERENT WAYS TO DO THIS...

- Online
- Mini-batch gradient descent

I DON'T WANT TO IMPLEMENT GRADIENT DESCENT

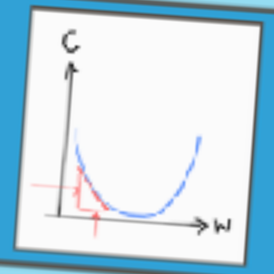
- Tensorflow
- Deep Learning 4 J

Feed Forward

MATHS REQUIRED

DIFFICULTY

Move forwards through network to get the output

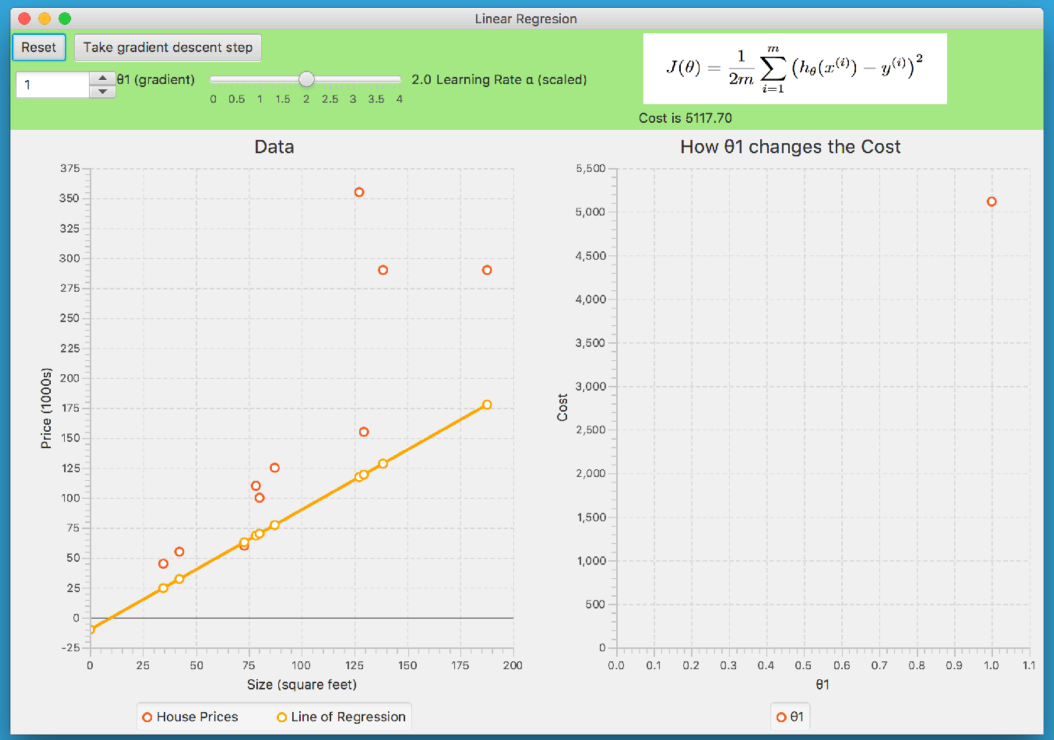


Back Propagation

MATHS REQUIRED

DIFFICULTY

Determine how the weights in the network affect the error

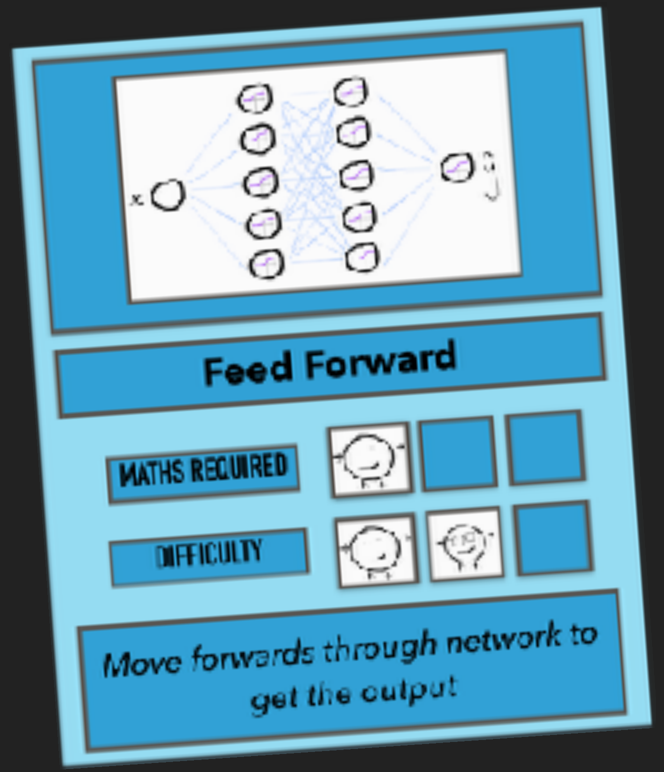


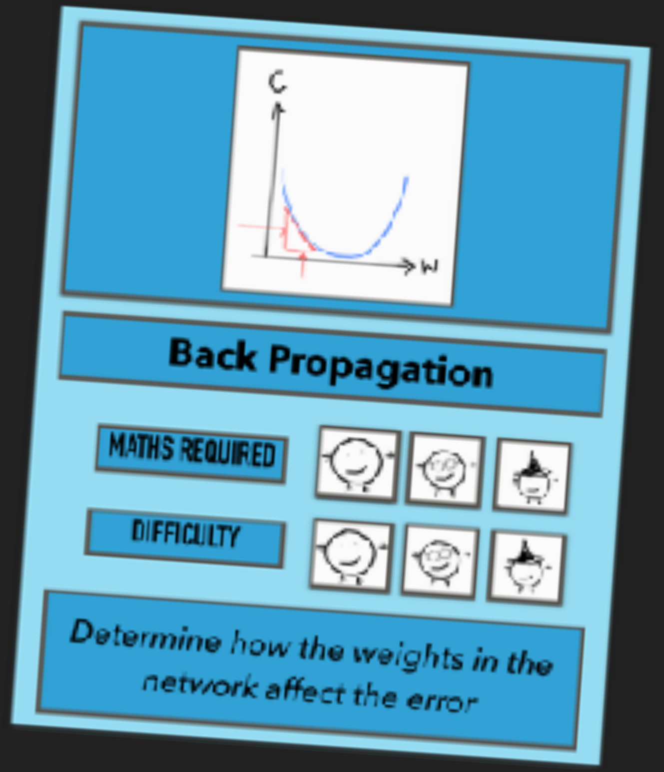
Gradient Descent

MATHS REQUIRED

DIFFICULTY

Reduce the weights in proportion to the gradients





THE LEARNING RATE

PART I: NEURONS

- Neurons
  - Biological
  - Artificial
    - Linear Neuron
    - Binary Threshold Neuron
    - Logistic/sigmoid Neuron
    - reLU
    - Tanh

PART II: LEARNING

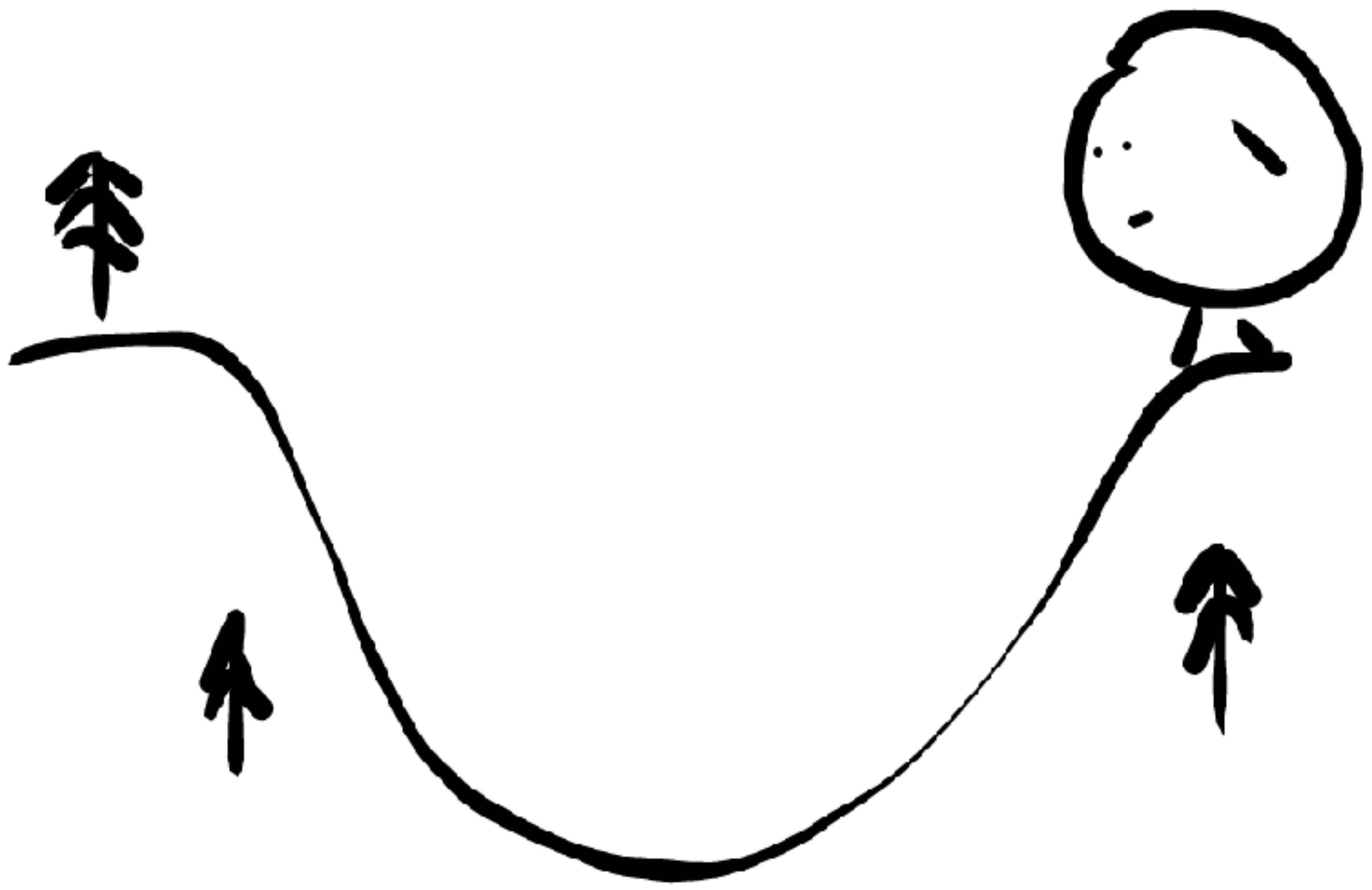
- Network
  - Feed forward — Cost functions
  - Back propagation — Training
    - Gradient descent
    - Learning rate
    - Regularisation

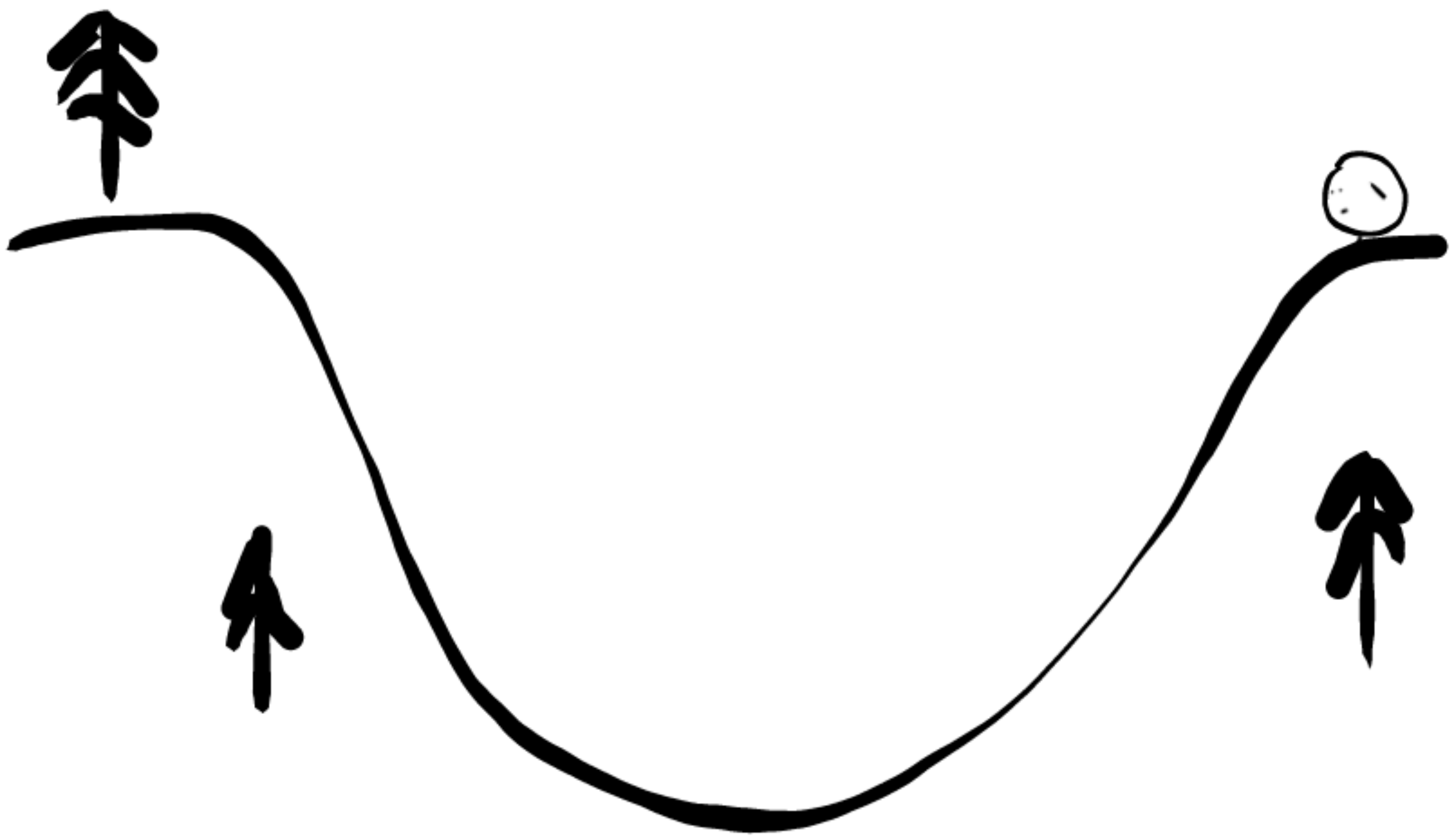
PART III: IN A NETWORK

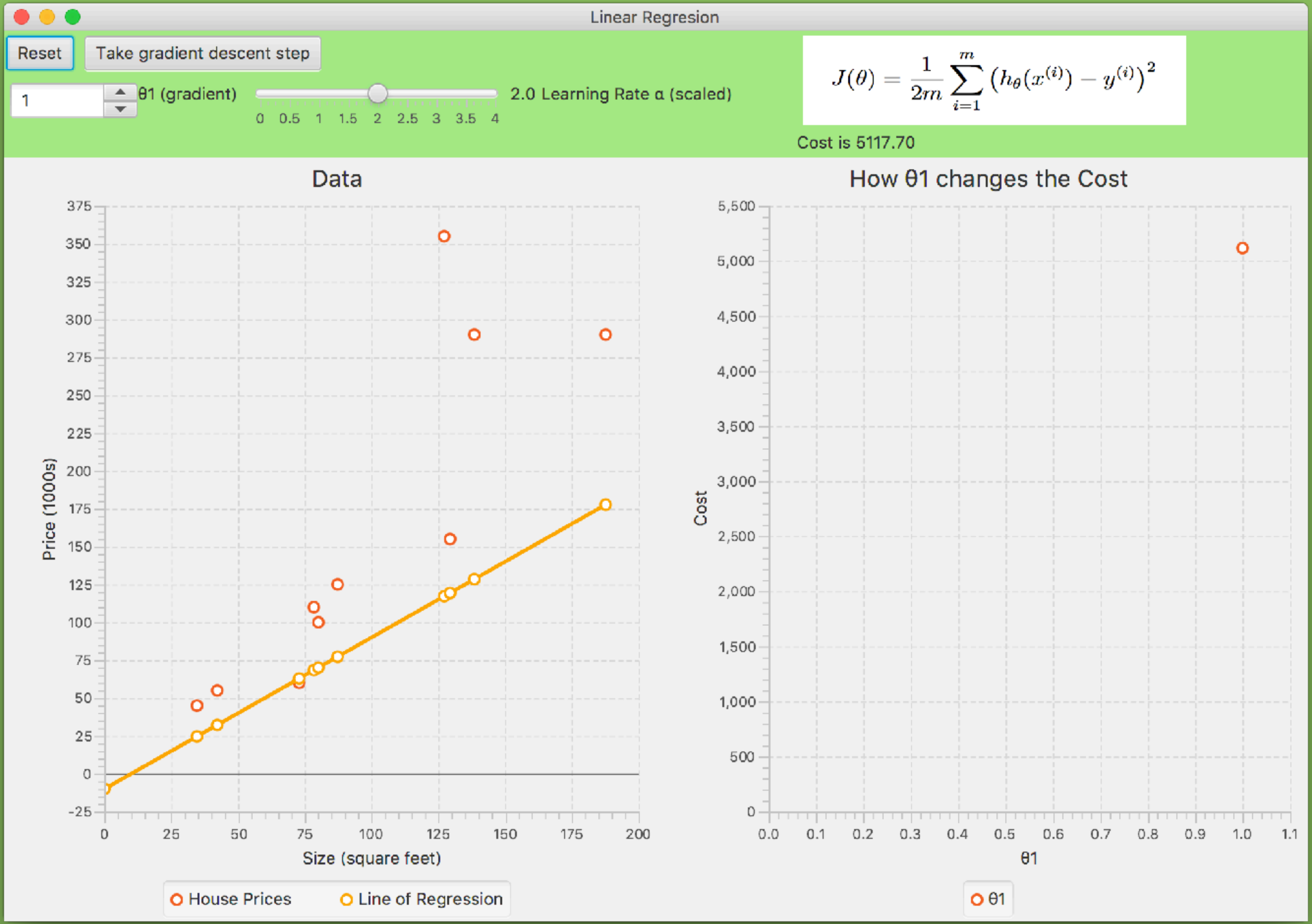
- Learning to map input to output — Architectures
  - CNN
  - RNN
  - And beyond...

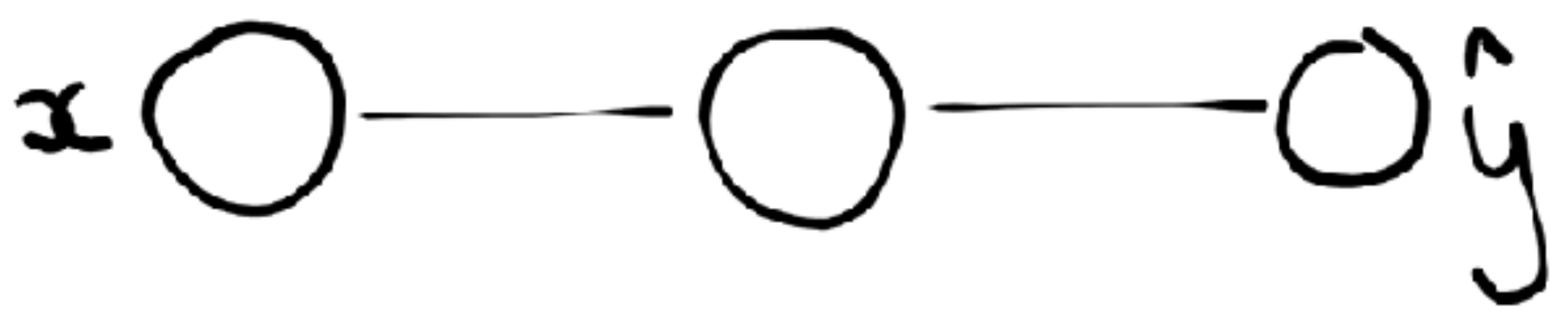
WHAT IF THE LEARNING RATE IS TOO HIGH?

WHAT IF THE LEARNING RATE IS TOO LOW?






Linear Regresion
Reset
Take gradient descent step
1
θ1 (gradient)
0 0.5 1 1.5 2 2.5 3 3.5 4
2.0 Learning Rate α (scaled)
$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$
Cost is 5117.70
Data
Price (1000s)
375 350 325 300 275 250 225 200 175 150 125 100 75 50 25 0 -25
0 25 50 75 100 125 150 175 200
Size (square feet)
House Prices
Line of Regression
How θ1 changes the Cost
Cost
5,500 5,000 4,500 4,000 3,500 3,000 2,500 2,000 1,500 1,000 500 0
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 1.1
θ1
θ1



Data

100 data points

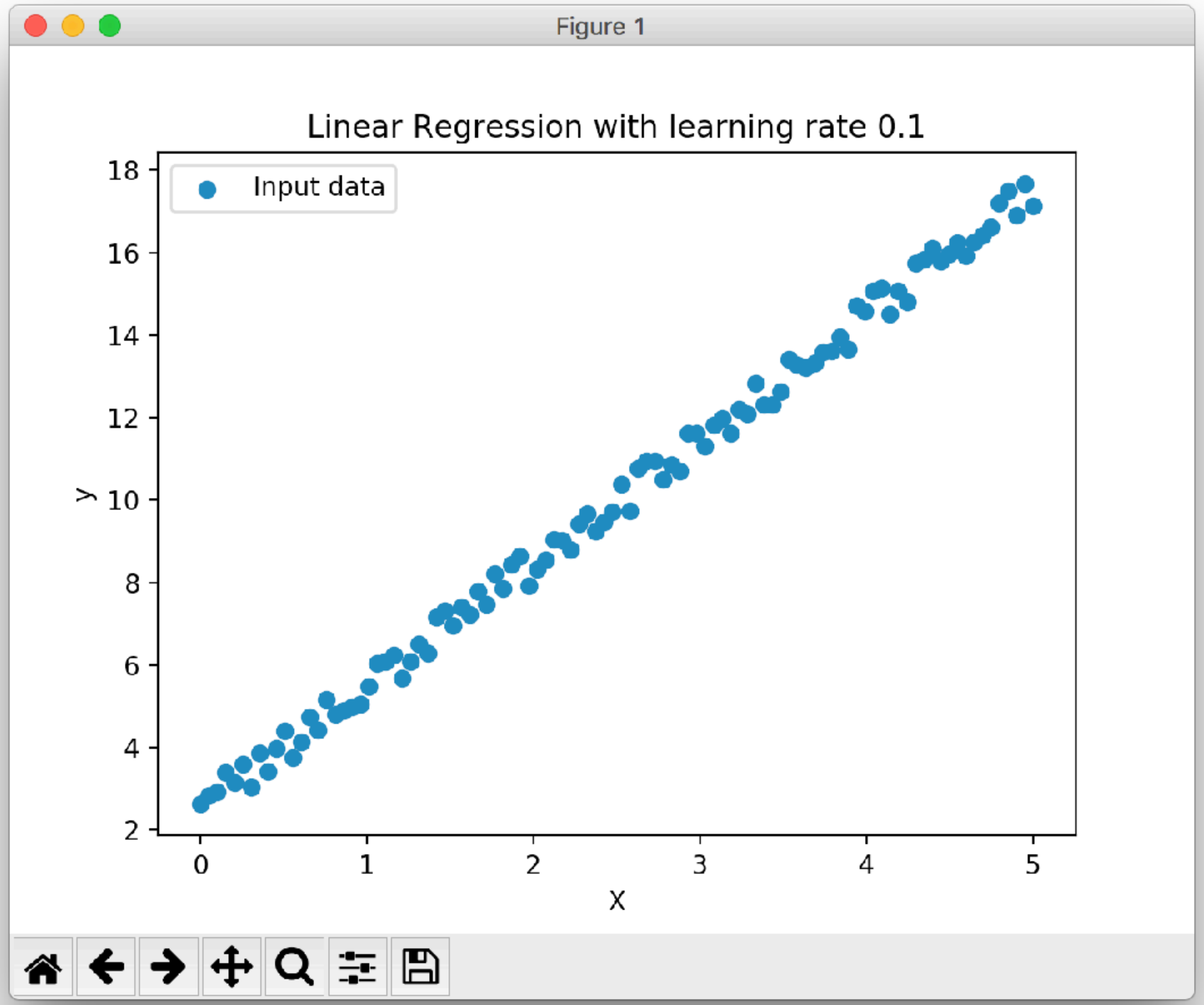
Inputs are between 0 and 5

Output is a function y = 3X + 2 + random noise

Network

1 hidden layers, with 1 hidden unit

Linear/identity activation function

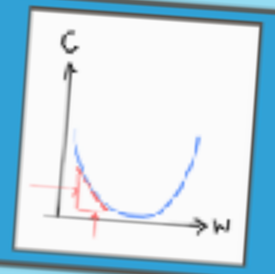


Feed Forward

MATHS REQUIRED

DIFFICULTY

Move forwards through network to get the output



Back Propagation

MATHS REQUIRED

DIFFICULTY

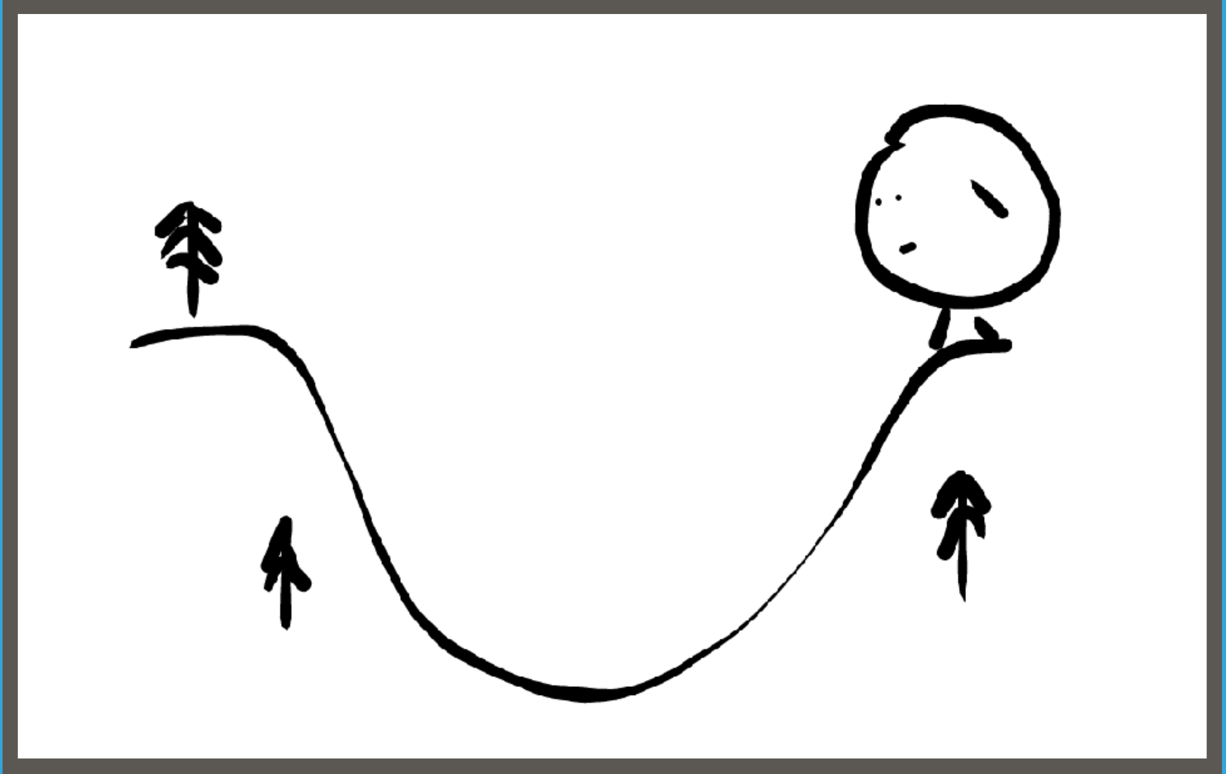
Determine how the weights in the network affect the error

Gradient Descent

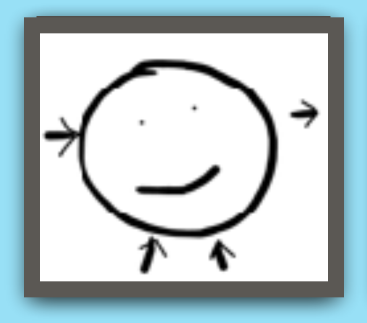
MATHS REQUIRED

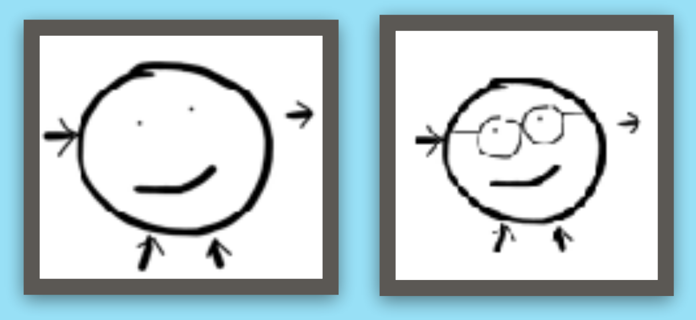
DIFFICULTY

Reduce the weights in proportion to the gradients

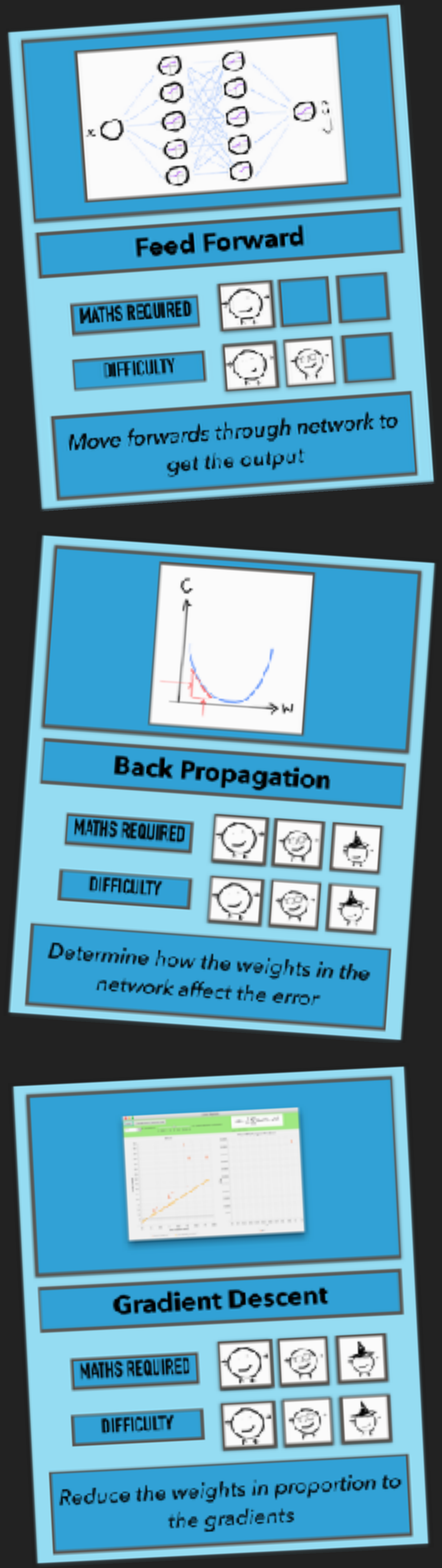


The Learning Rate

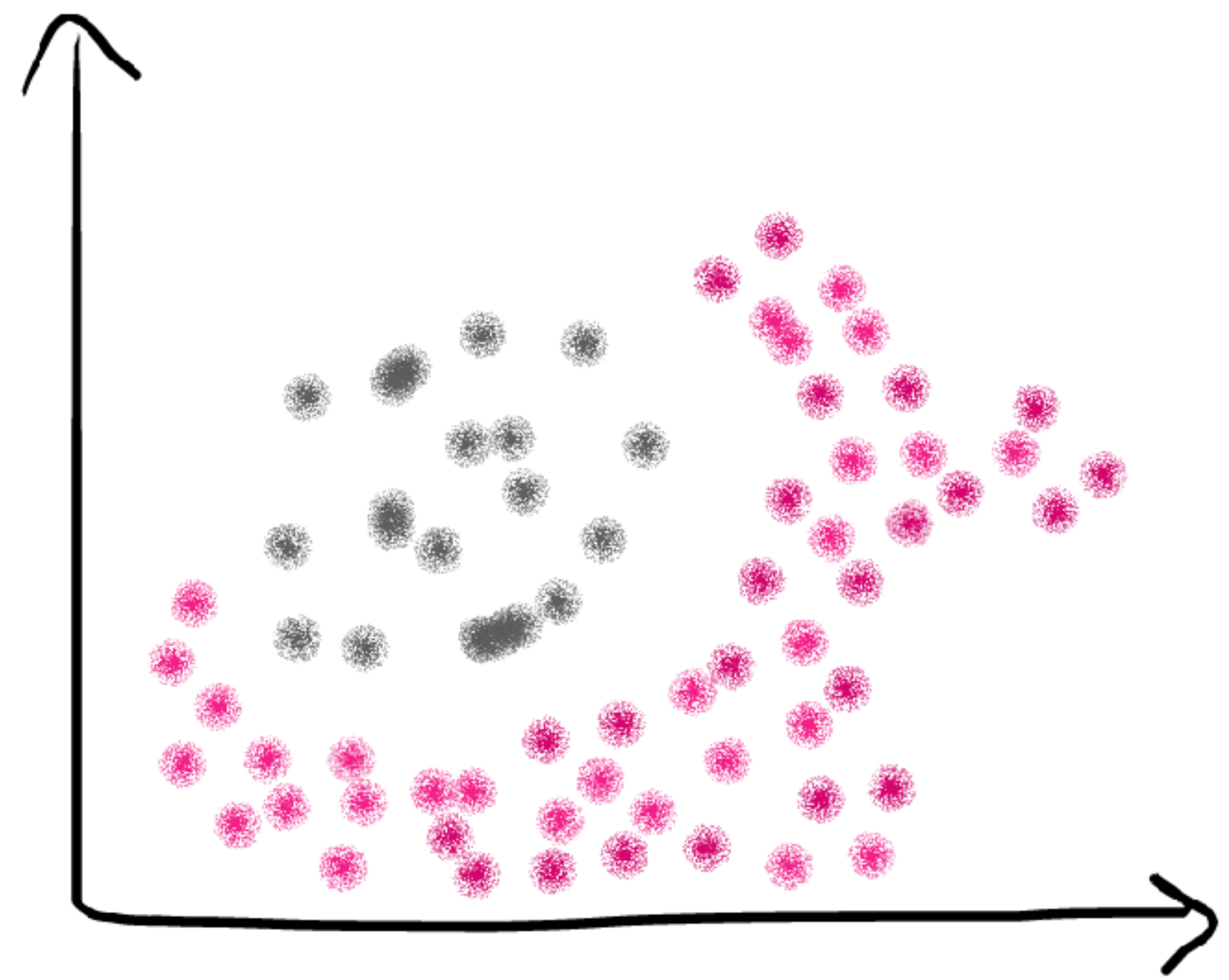
MATHS REQUIRED 

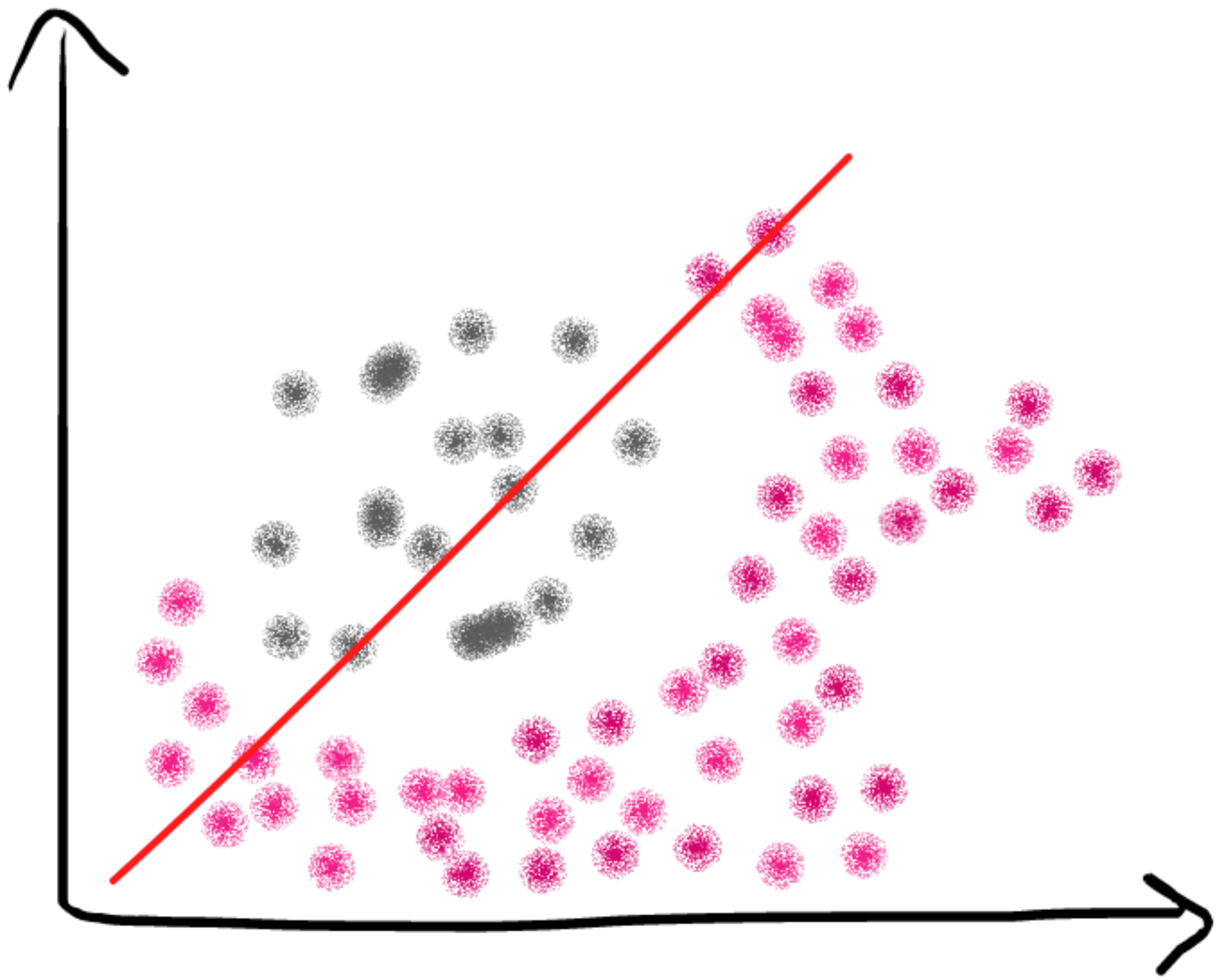
DIFFICULTY 

How big a step to take during gradient descent



UNDERFITTING





REASONS FOR UNDERFITTING

- Network is too simple!
  - Try more hidden units
  - Try more hidden layers
- Activation function is not suitable!
  - Are you using linear activation function? Or binary threshold? Or… perceptrons?

The Learning Rate

MATHS REQUIRED

DIFFICULTY

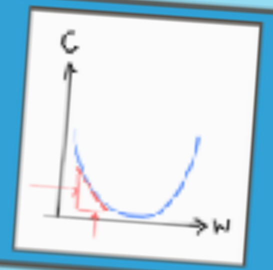
How big a step to take during gradient descent

Feed Forward

MATHS REQUIRED

DIFFICULTY

Move forwards through network to get the output



Back Propagation

MATHS REQUIRED

DIFFICULTY

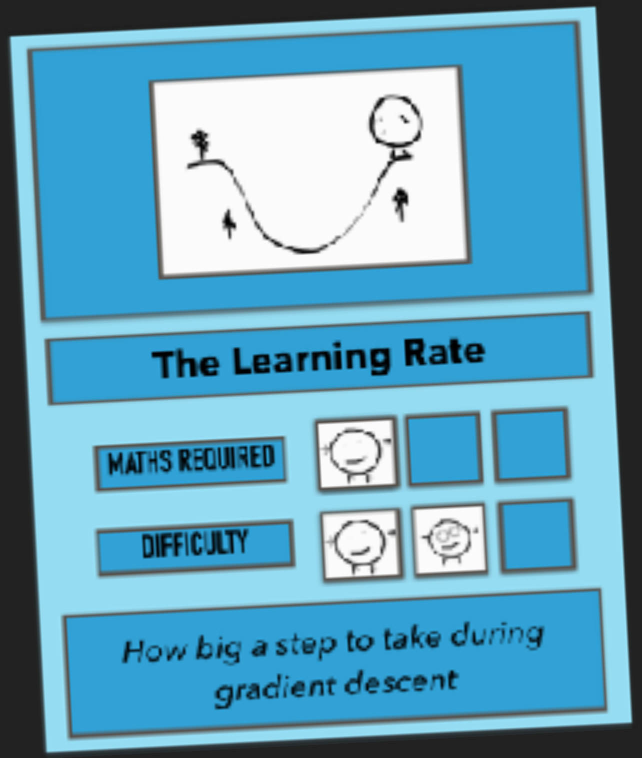
Determine how the weights in the network affect the error

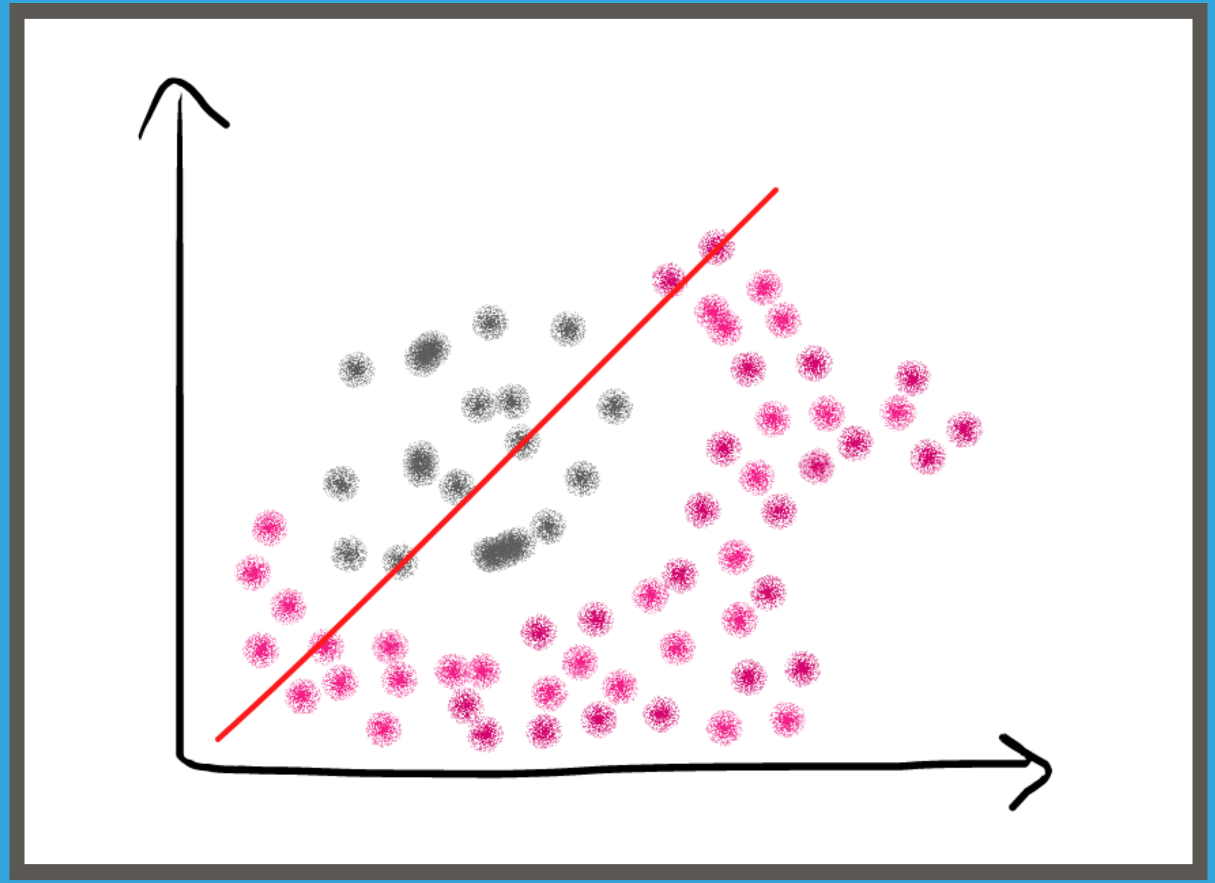
Gradient Descent

MATHS REQUIRED

DIFFICULTY

Reduce the weights in proportion to the gradients

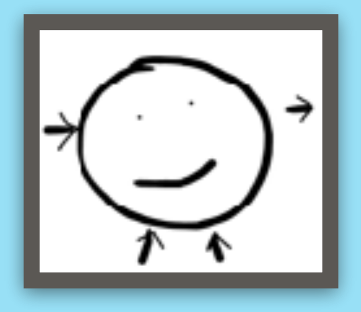




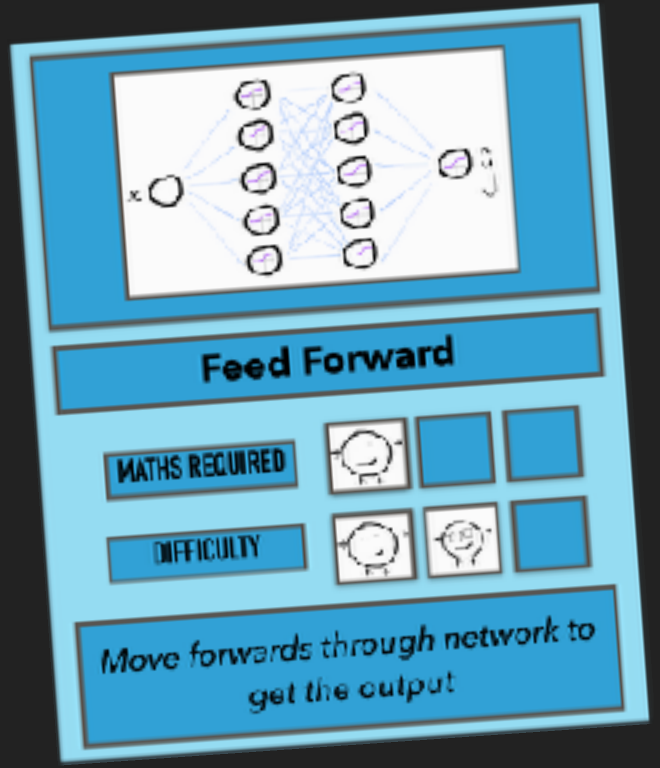
Underfitting

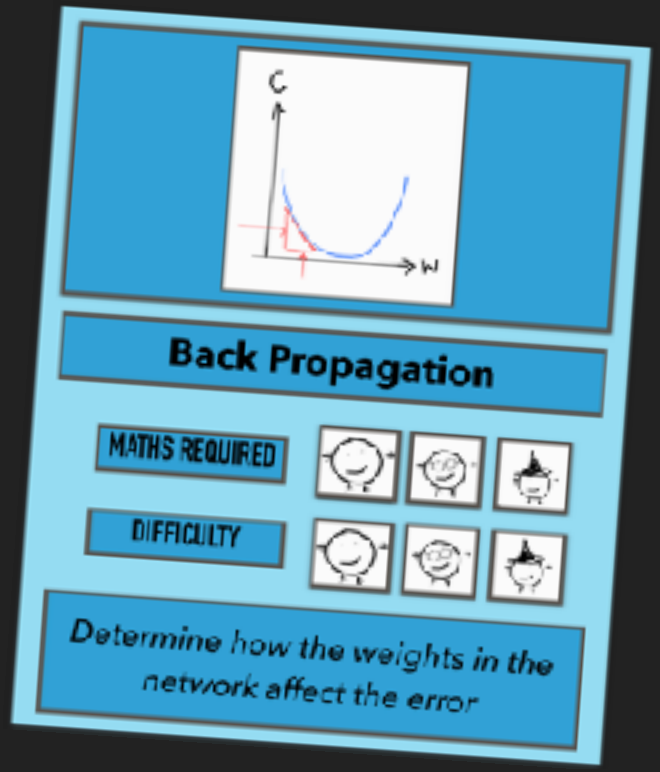
MATHS REQUIRED

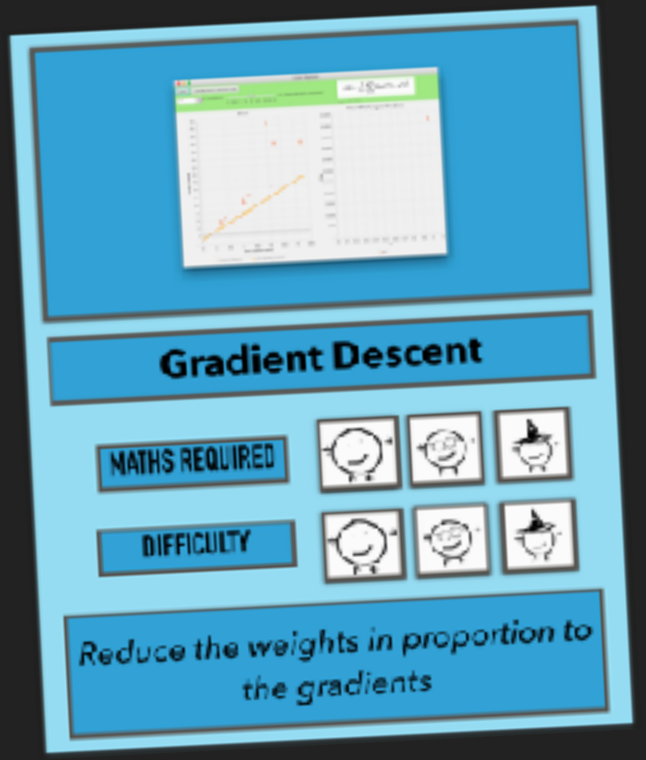
DIFFICULTY



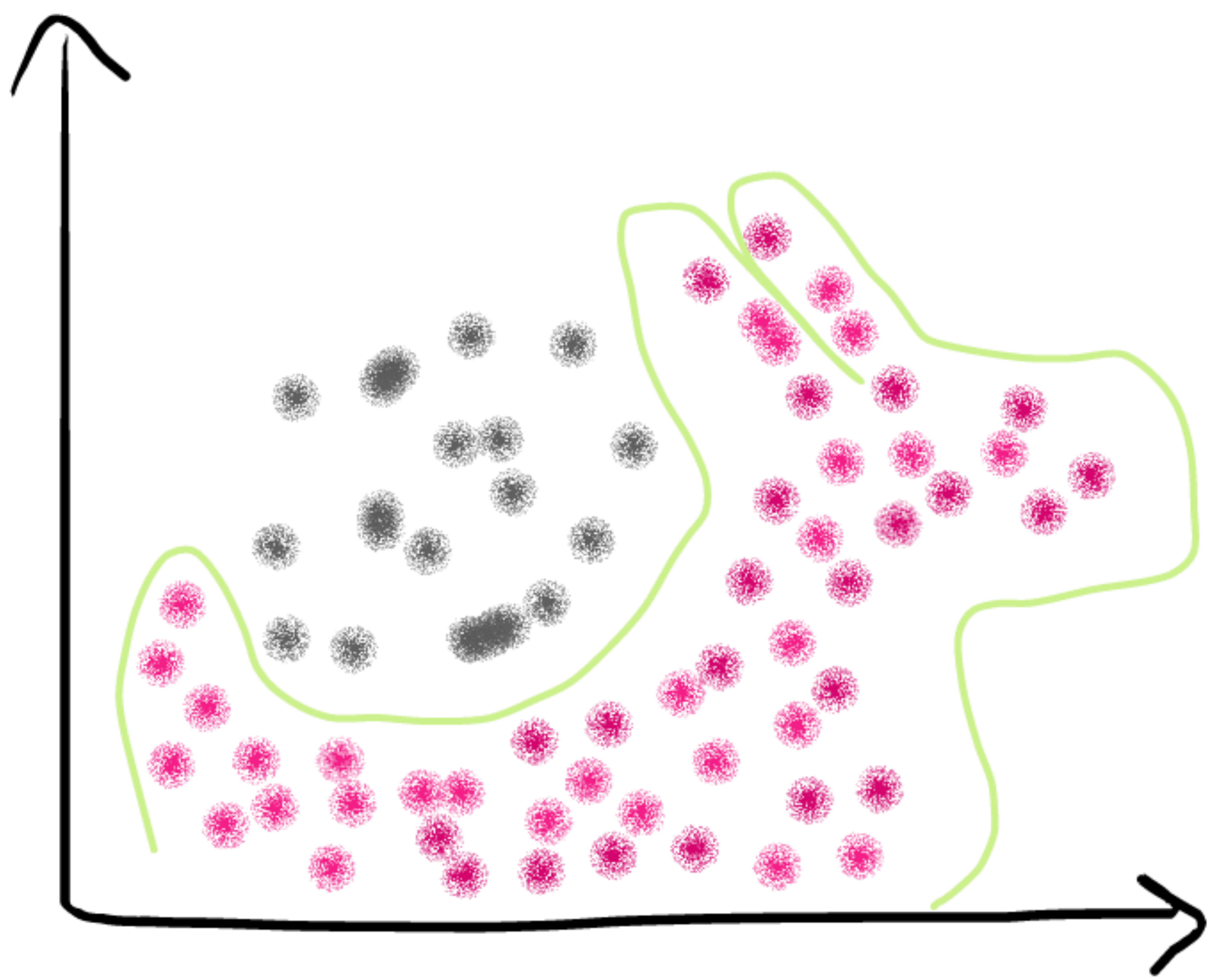
Predictions don't fit the data well enough

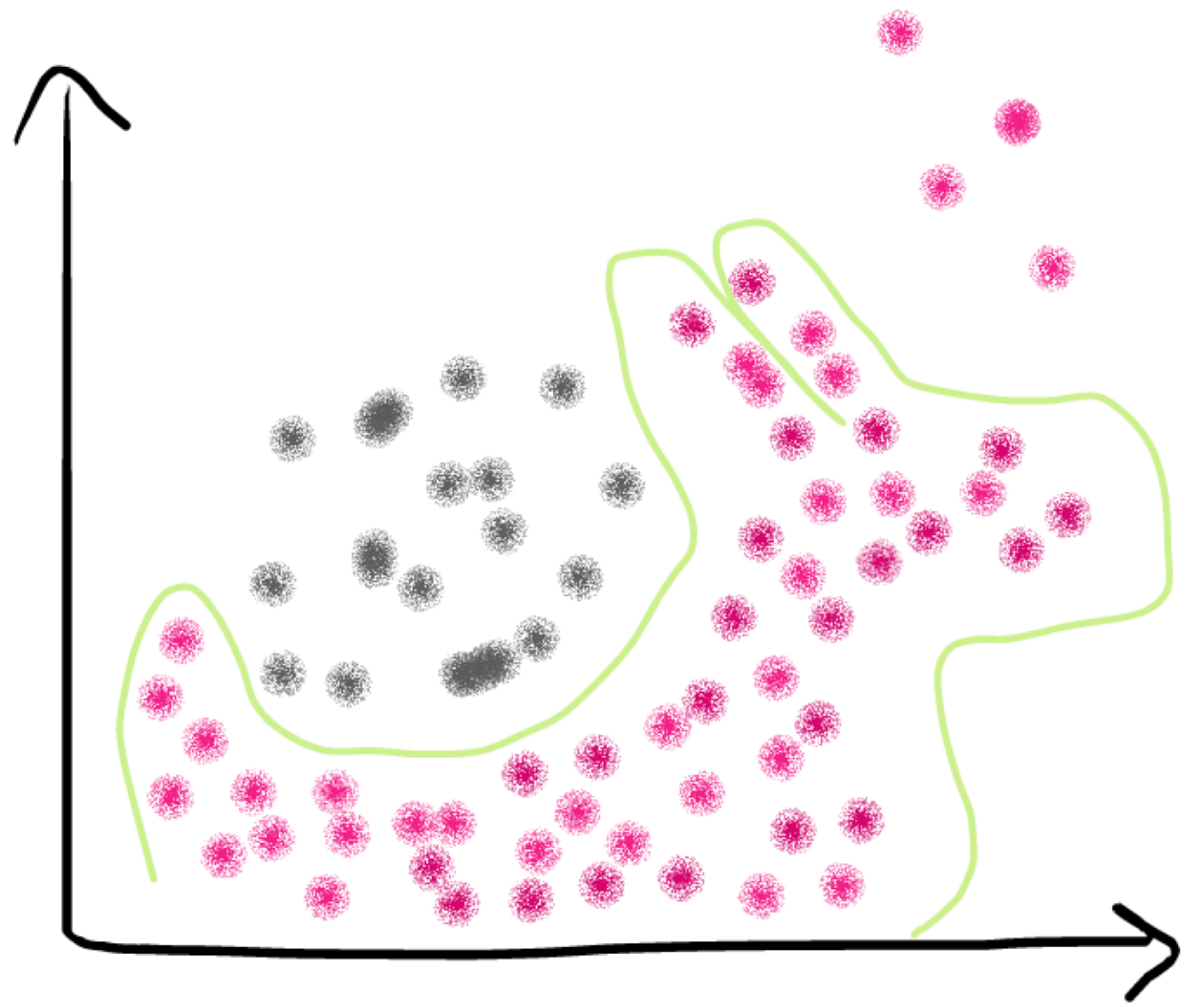






OVERFITTING





REASONS FOR OVERFITTING

- Haven’t trained the network properly
  - Don’t train on all the data, hold back a chunk to check your hyperparameters (a cross-validation set) like learning rate, number of hidden units, layers, etc.
  - Hold back another chunk to test the final configuration (a test-set)
- Get more data
- Network is ‘too complicated’: get perfect architecture
- Average different models (bagging), early stopping,…
- Too complicated? Try…

The Learning Rate

MATHS REQUIRED

DIFFICULTY

How big a step to take during gradient descent

Underfitting

MATHS REQUIRED

DIFFICULTY

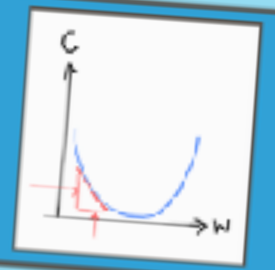
Predictions don't fit the data well enough

Feed Forward

MATHS REQUIRED

DIFFICULTY

Move forwards through network to get the output



Back Propagation

MATHS REQUIRED

DIFFICULTY

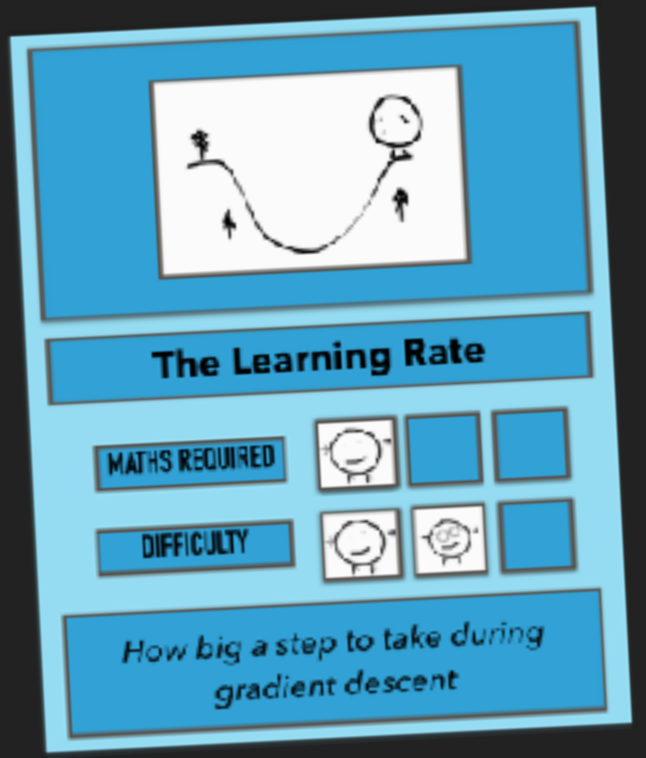
Determine how the weights in the network affect the error

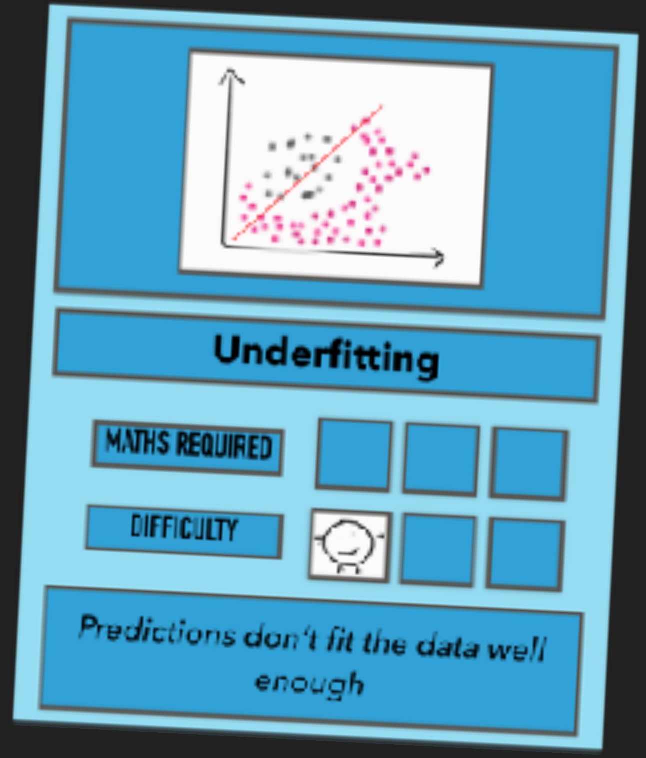
Gradient Descent

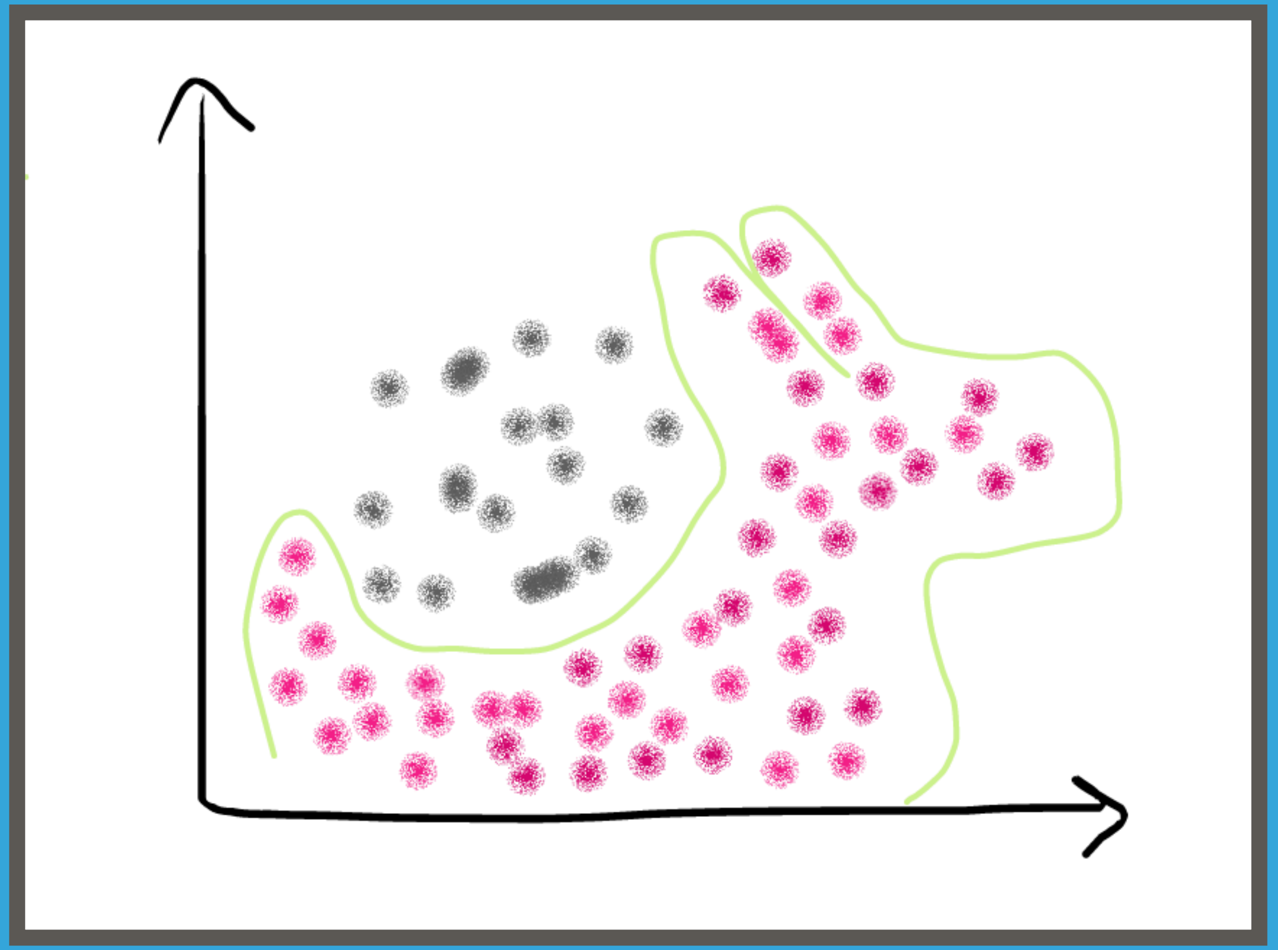
MATHS REQUIRED

DIFFICULTY

Reduce the weights in proportion to the gradients



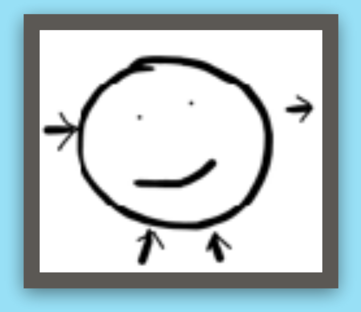




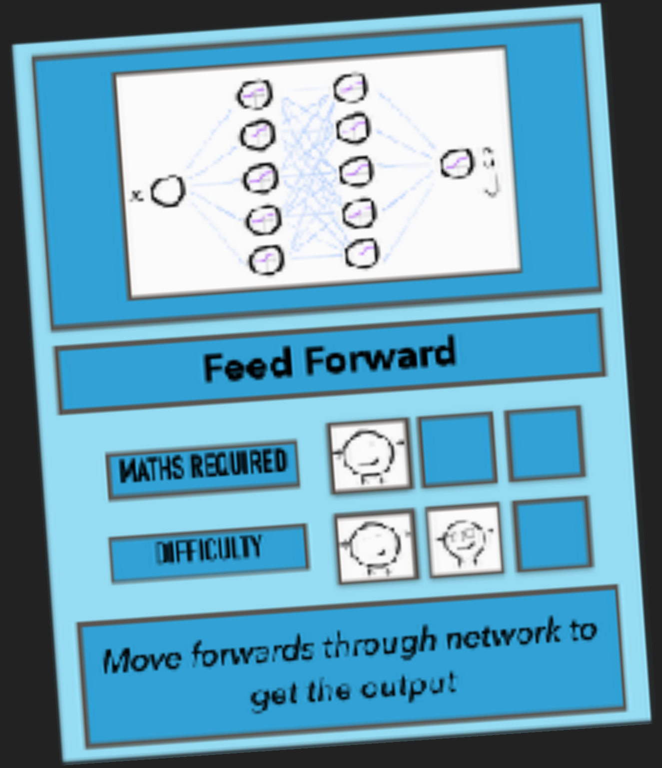
Overfitting

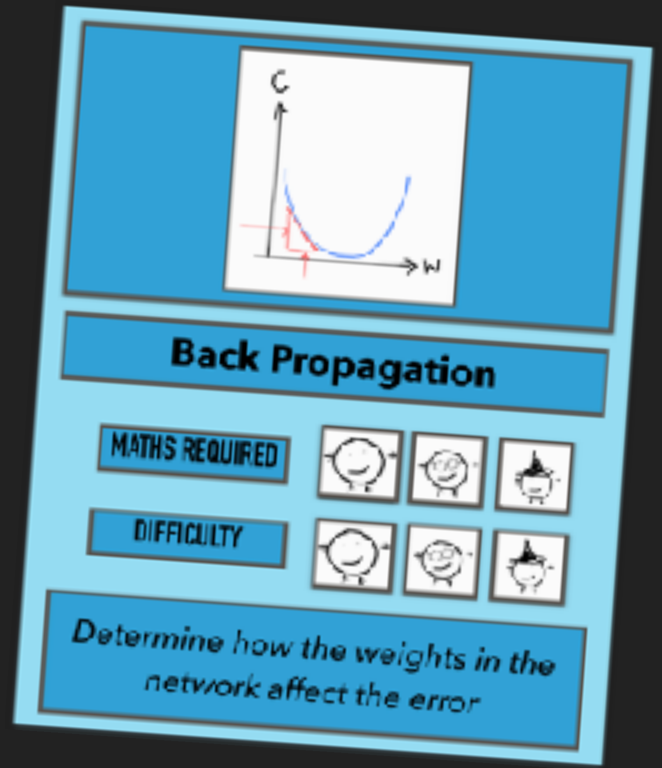
MATHS REQUIRED

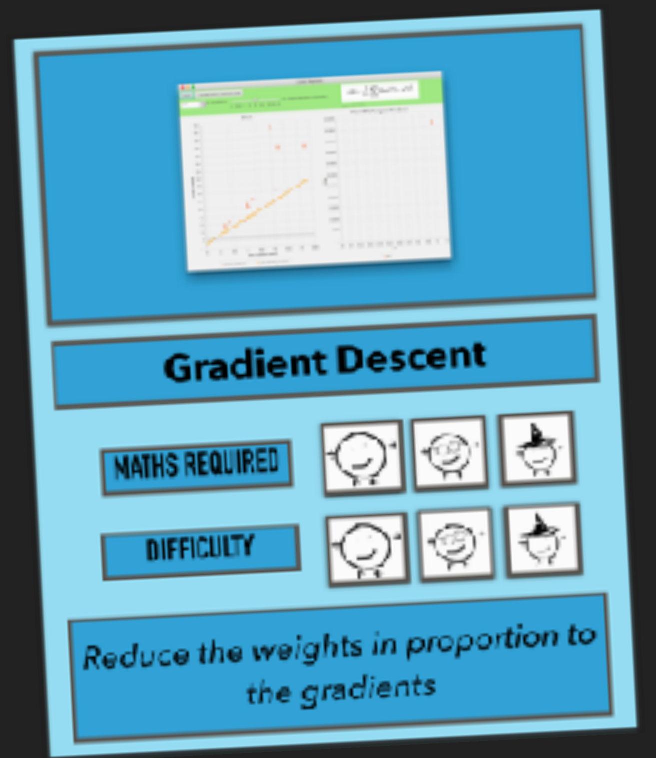
DIFFICULTY



Prediction fits the data too well







REGULARISATION

PART I: NEURONS

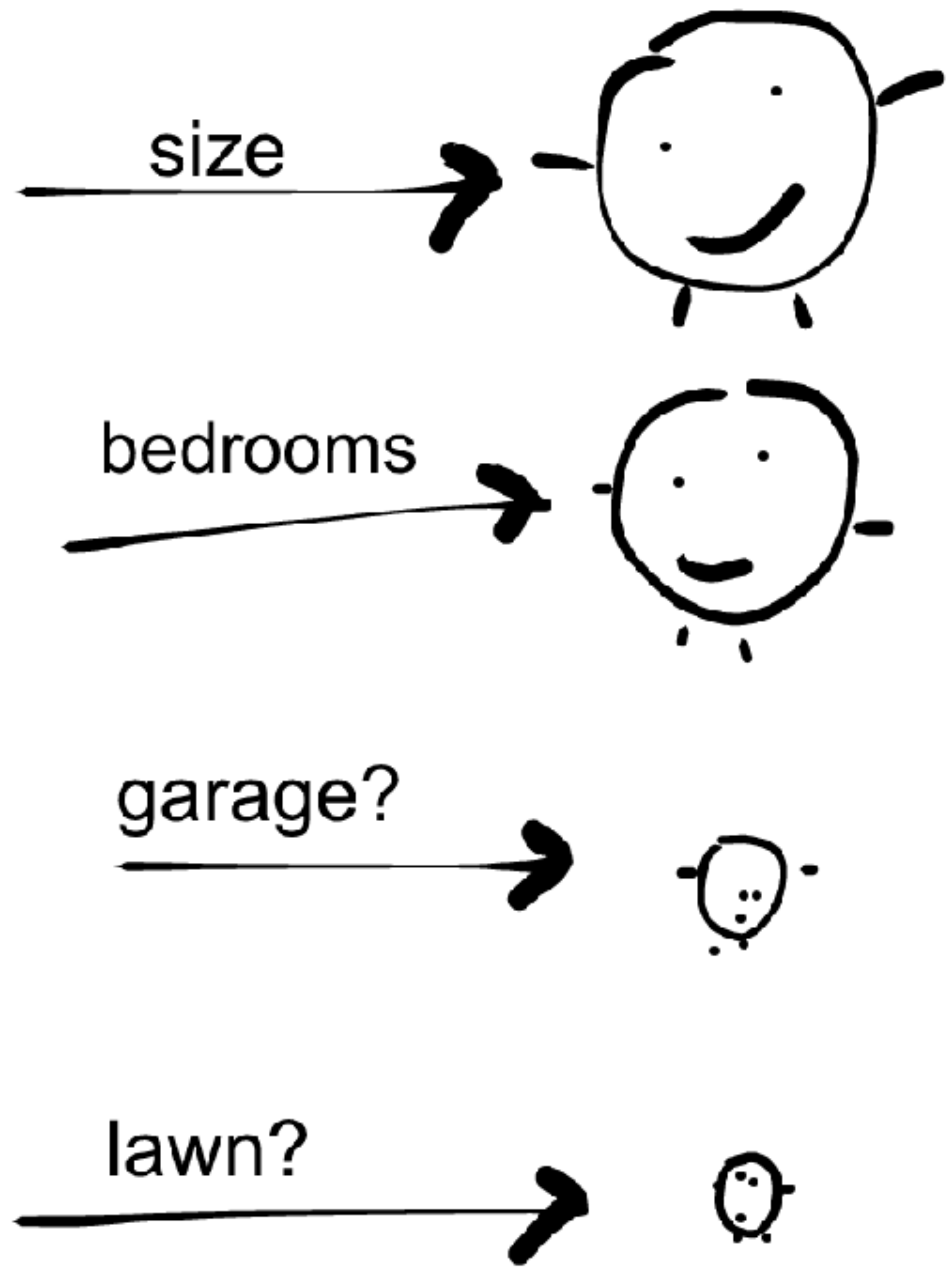
- Neurons
  - Biological
  - Artificial
    - Linear Neuron
    - Binary Threshold Neuron
    - Logistic/sigmoid Neuron
    - reLU
    - Tanh

PART II: LEARNING

- Network
  - Feed forward — Cost functions
  - Back propagation — Training
    - Gradient descent
    - Learning rate
    - Regularisation

PART III: IN A NETWORK

- Learning to map input to output — Architectures
  - CNN
  - RNN
  - And beyond…


size
bedrooms
garage?
lawn?

OR – REDUCING NOISE

- Ways, during training, to reduce some of the weights.
- L1 regularisation
- L2 regularisation

The Learning Rate

MATHS REQUIRED

DIFFICULTY

How big a step to take during gradient descent

Underfitting

MATHS REQUIRED

DIFFICULTY

Predictions don't fit the data well enough

Overfitting

MATHS REQUIRED

DIFFICULTY

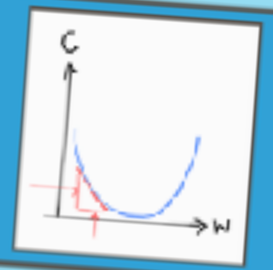
Prediction fits the data too well

Feed Forward

MATHS REQUIRED

DIFFICULTY

Move forwards through network to get the output



Back Propagation

MATHS REQUIRED

DIFFICULTY

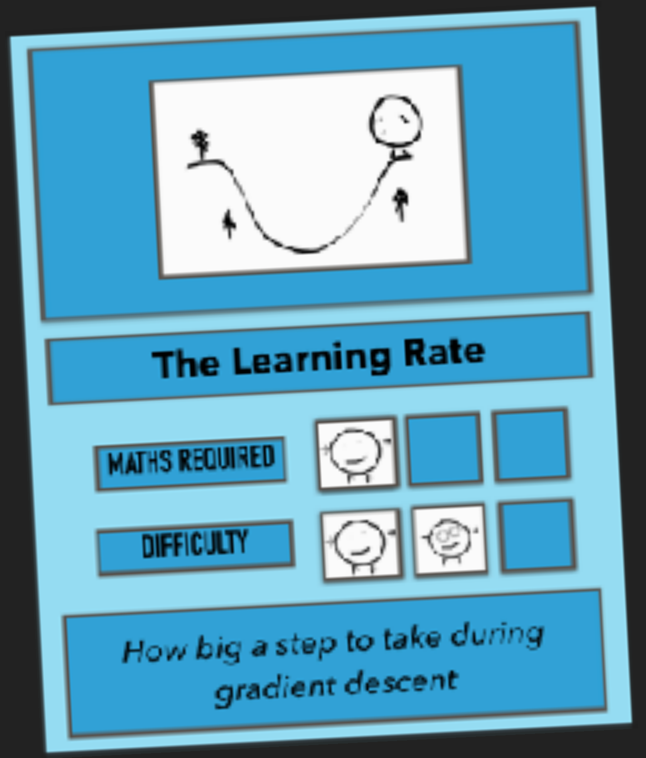
Determine how the weights in the network affect the error

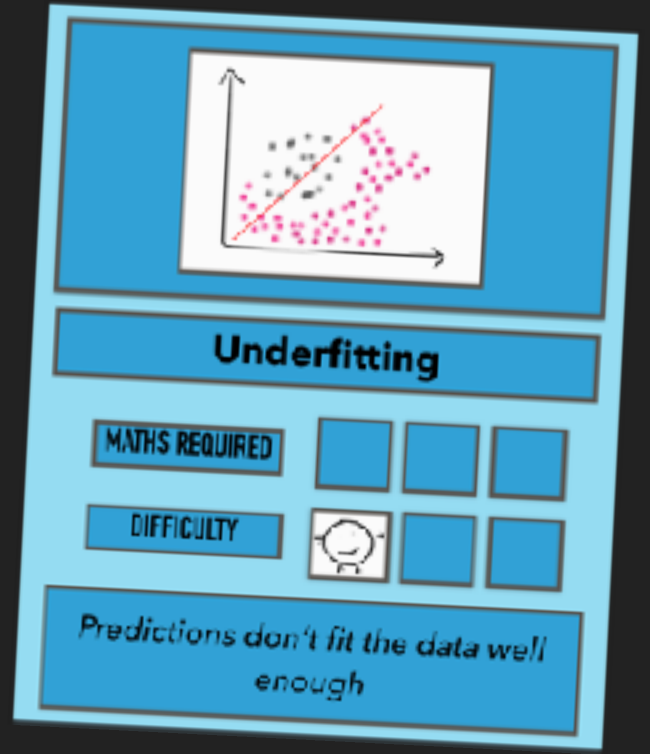
Gradient Descent

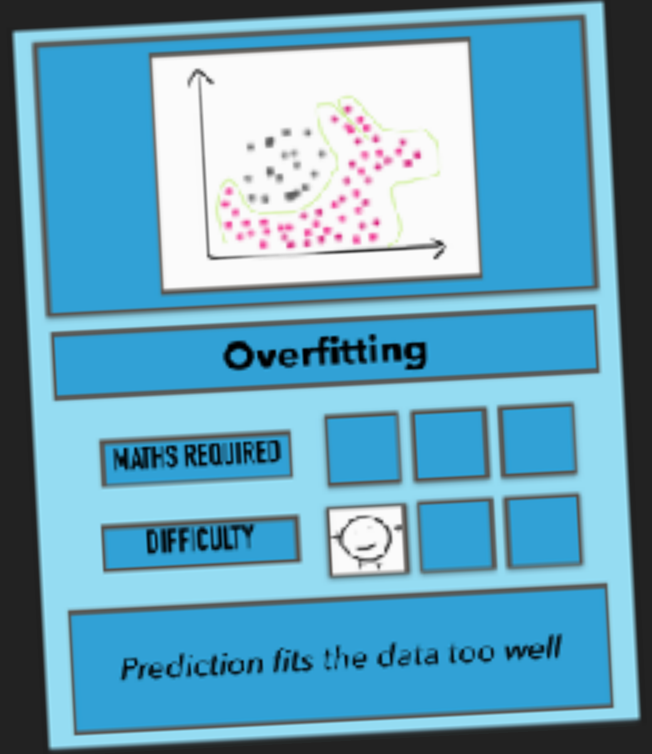
MATHS REQUIRED

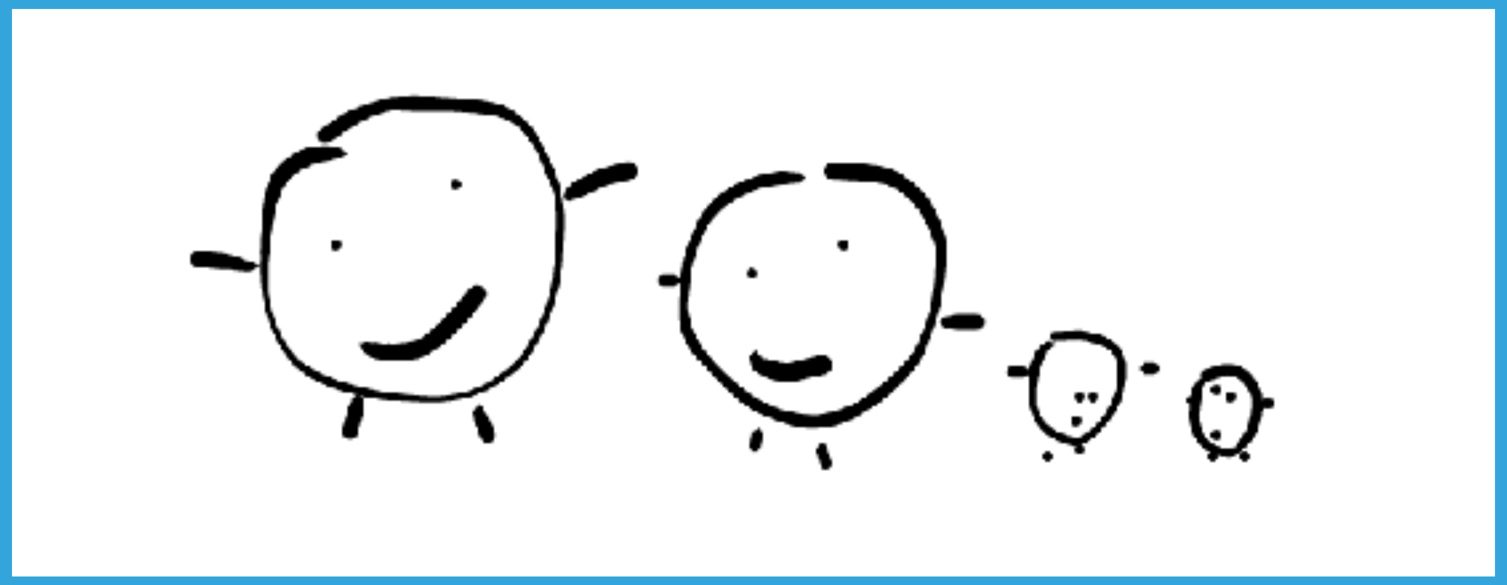
DIFFICULTY

Reduce the weights in proportion to the gradients







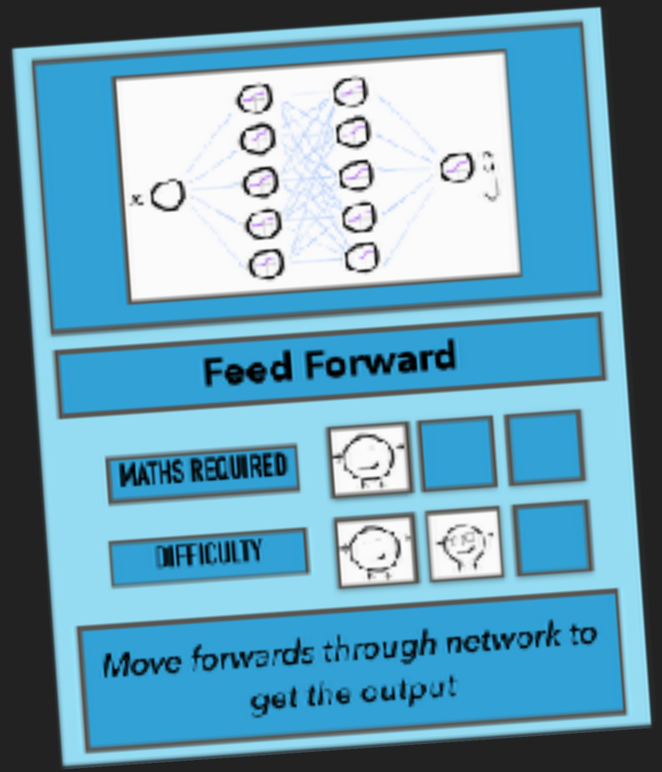


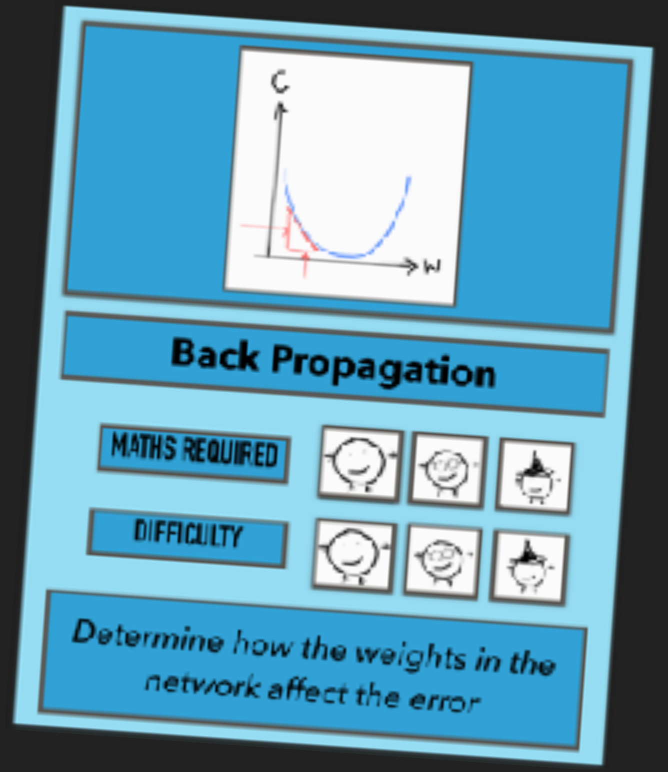
Regularisation

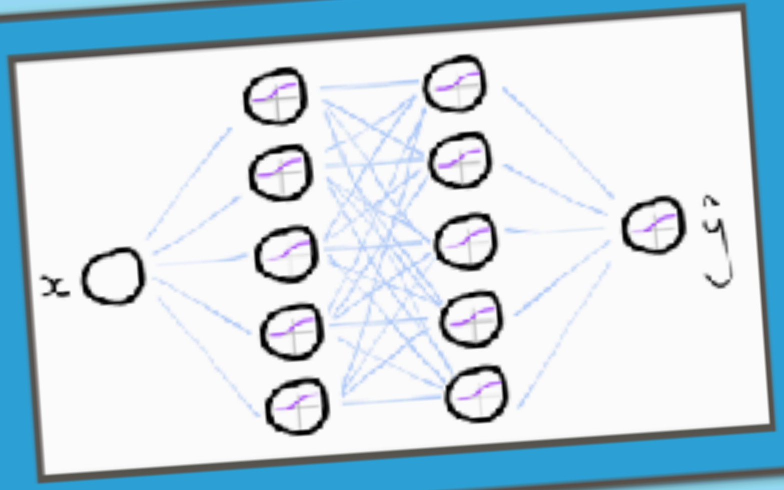
MATHS REQUIRED 

DIFFICULTY 

Reduce overfitting with weight penalties





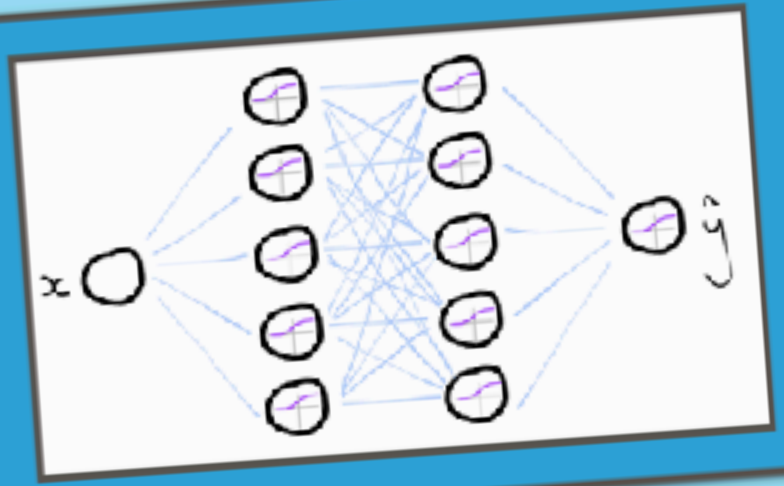


Feed Forward

MATHS REQUIRED

DIFFICULTY

Move forwards through network to get the output

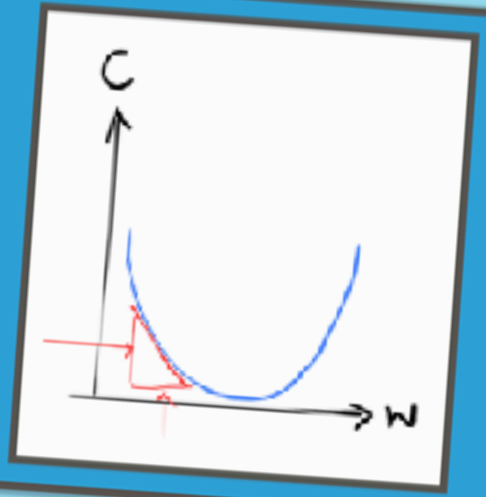


Feed Forward

MATHS REQUIRED

DIFFICULTY

Move forwards through network to get the output

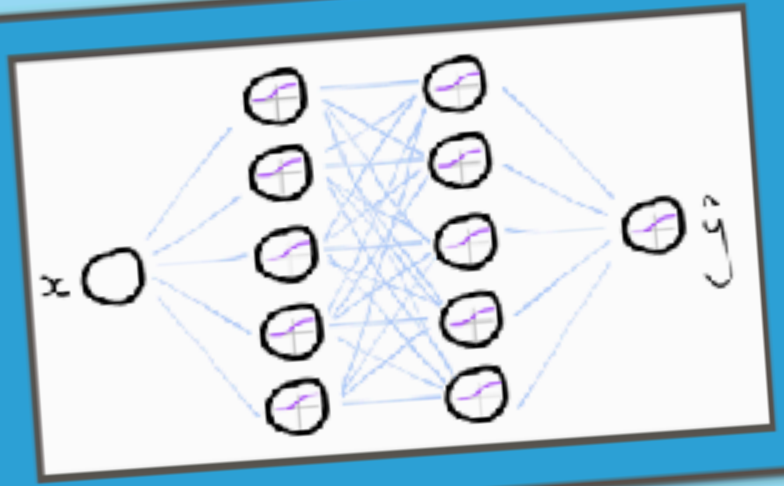


Back Propagation

MATHS REQUIRED

DIFFICULTY

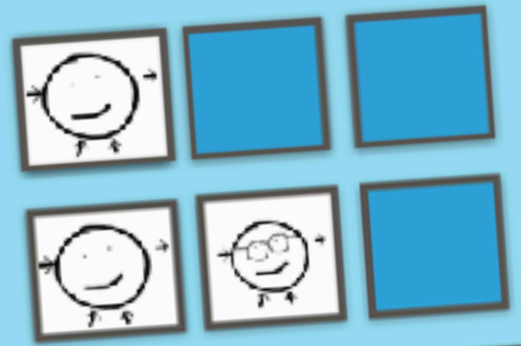
Determine how the weights in the network affect the error



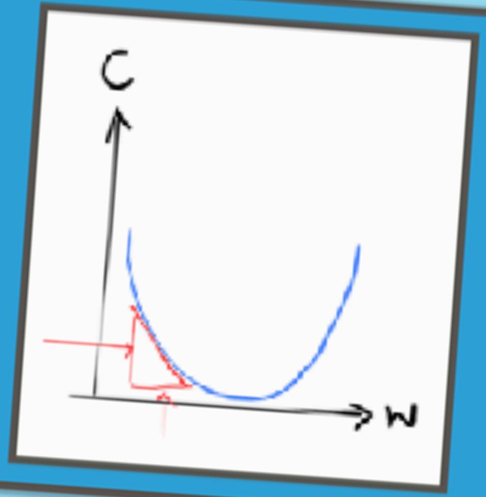
Feed Forward

MATHS REQUIRED

DIFFICULTY



Move forwards through network to get the output

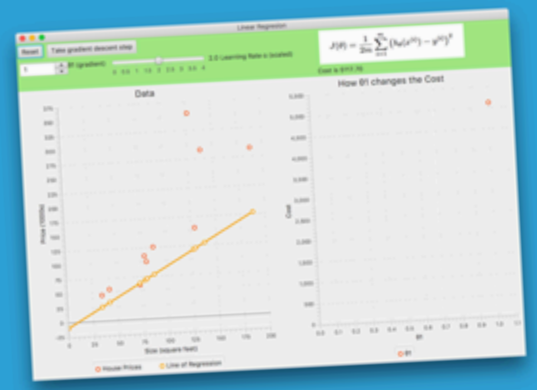


Back Propagation

MATHS REQUIRED

DIFFICULTY

Determine how the weights in the network affect the error

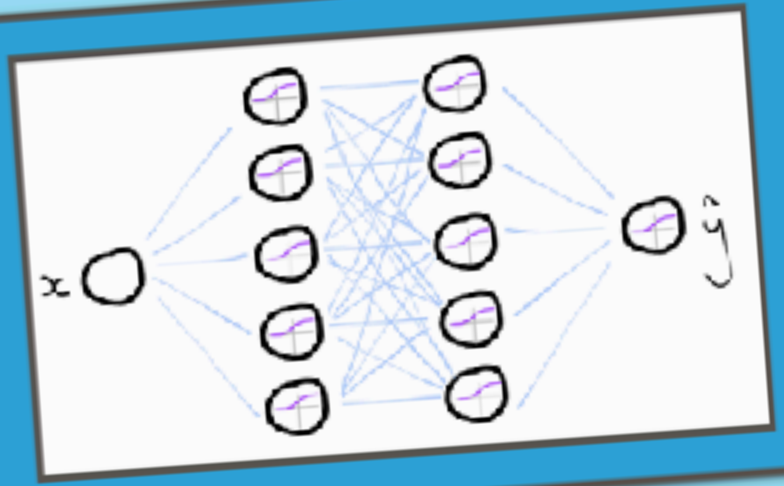


Gradient Descent

MATHS REQUIRED

DIFFICULTY

Reduce the weights in proportion to the gradients

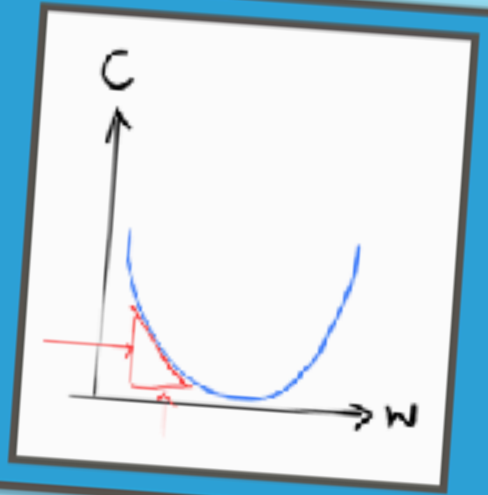


Feed Forward

MATHS REQUIRED

DIFFICULTY

Move forwards through network to get the output

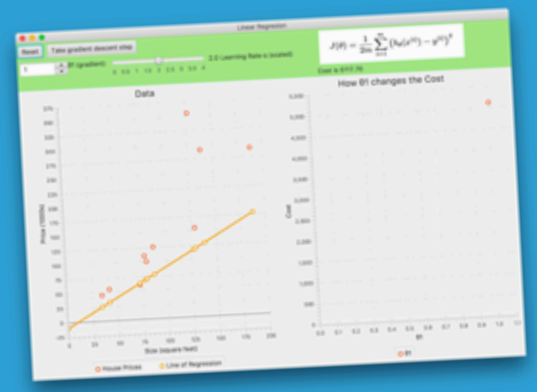


Back Propagation

MATHS REQUIRED

DIFFICULTY

Determine how the weights in the network affect the error



Gradient Descent

MATHS REQUIRED

DIFFICULTY

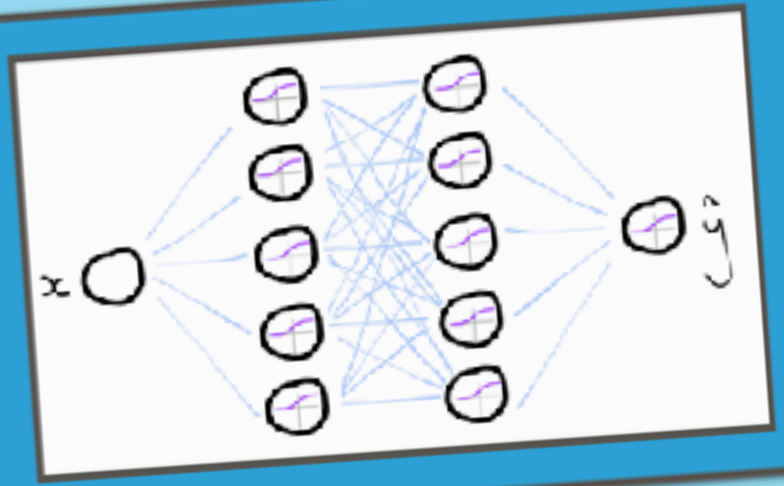
Reduce the weights in proportion to the gradients

The Learning Rate

MATHS REQUIRED

DIFFICULTY

How big a step to take during gradient descent

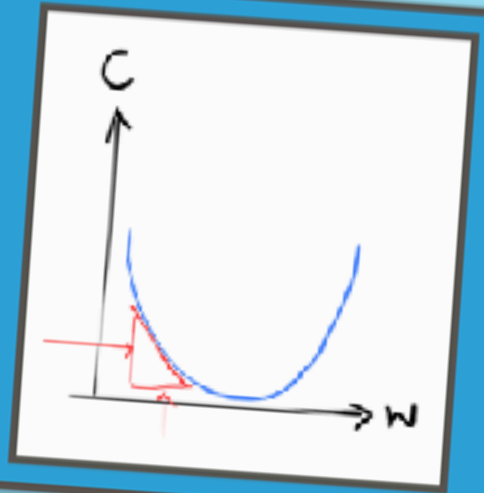


Feed Forward

MATHS REQUIRED

DIFFICULTY

Move forwards through network to get the output



Back Propagation

MATHS REQUIRED

DIFFICULTY

Determine how the weights in the network affect the error

Gradient Descent

MATHS REQUIRED

DIFFICULTY

Reduce the weights in proportion to the gradients

The Learning Rate

MATHS REQUIRED

DIFFICULTY

How big a step to take during gradient descent

Underfitting

MATHS REQUIRED

DIFFICULTY

Predictions don't fit the data well enough


Feed Forward
MATHS REQUIRED
DIFFICULTY
Move forwards through network to get the output
Back Propagation
C
w
MATHS REQUIRED
DIFFICULTY
Determine how the weights in the network affect the error
Gradient Descent
MATHS REQUIRED
DIFFICULTY
Reduce the weights in proportion to the gradients
The Learning Rate
MATHS REQUIRED
DIFFICULTY
How big a step to take during gradient descent
Underfitting
MATHS REQUIRED
DIFFICULTY
Predictions don't fit the data well enough
Overfitting
MATHS REQUIRED
DIFFICULTY
Prediction fits the data too well


x
ŷ
Feed Forward
MATHS REQUIRED
DIFFICULTY
Move forwards through network to get the output
C
w
Back Propagation
MATHS REQUIRED
DIFFICULTY
Determine how the weights in the network affect the error
Gradient Descent
MATHS REQUIRED
DIFFICULTY
Reduce the weights in proportion to the gradients
The Learning Rate
MATHS REQUIRED
DIFFICULTY
How big a step to take during gradient descent
Underfitting
MATHS REQUIRED
DIFFICULTY
Predictions don't fit the data well enough
Overfitting
MATHS REQUIRED
DIFFICULTY
Prediction fits the data too well
Regularisation
MATHS REQUIRED
DIFFICULTY
Reduce overfitting with weight penalties

PART I: NEURONS

- **Neurons**
  - **Biological**
  - **Artificial**
    - **Binary Threshold Neuron**
    - **Linear Neuron**
    - **Logistic/sigmoid Neuron**
    - **reLU**
    - **Tanh**

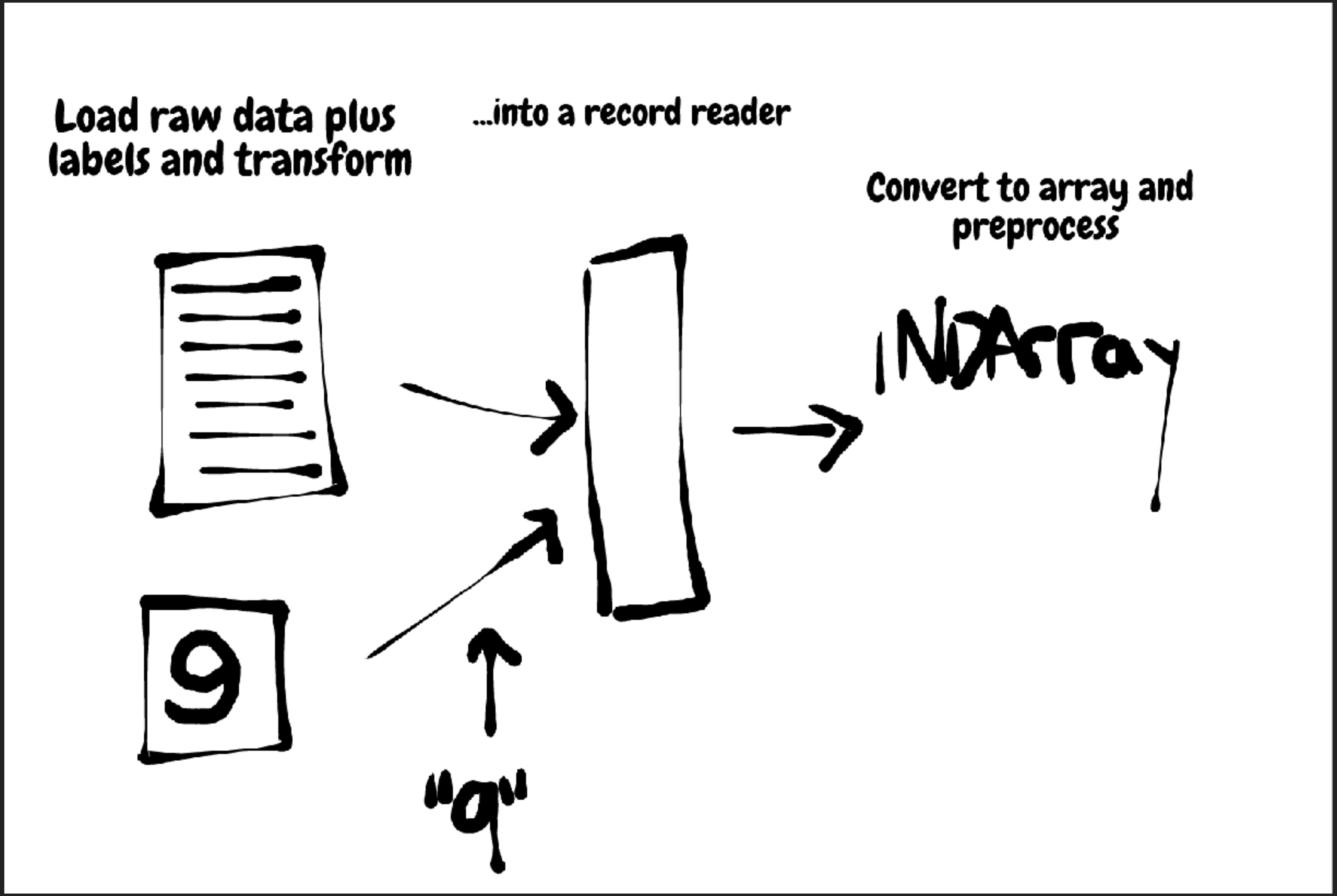
PART II: LEARNING

- **Network**
  - **Feed forward** — **Cost functions**
  - **Back propagation** — **Training**
    - **Gradient descent**
    - **Learning rate**
    - **Regularisation**

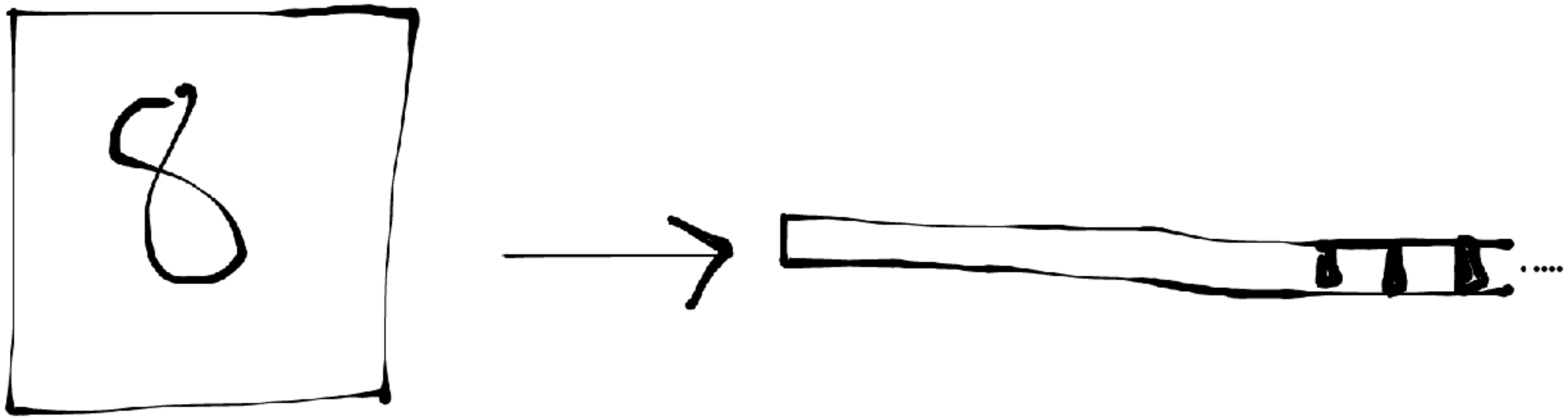
PART III: IN A NETWORK

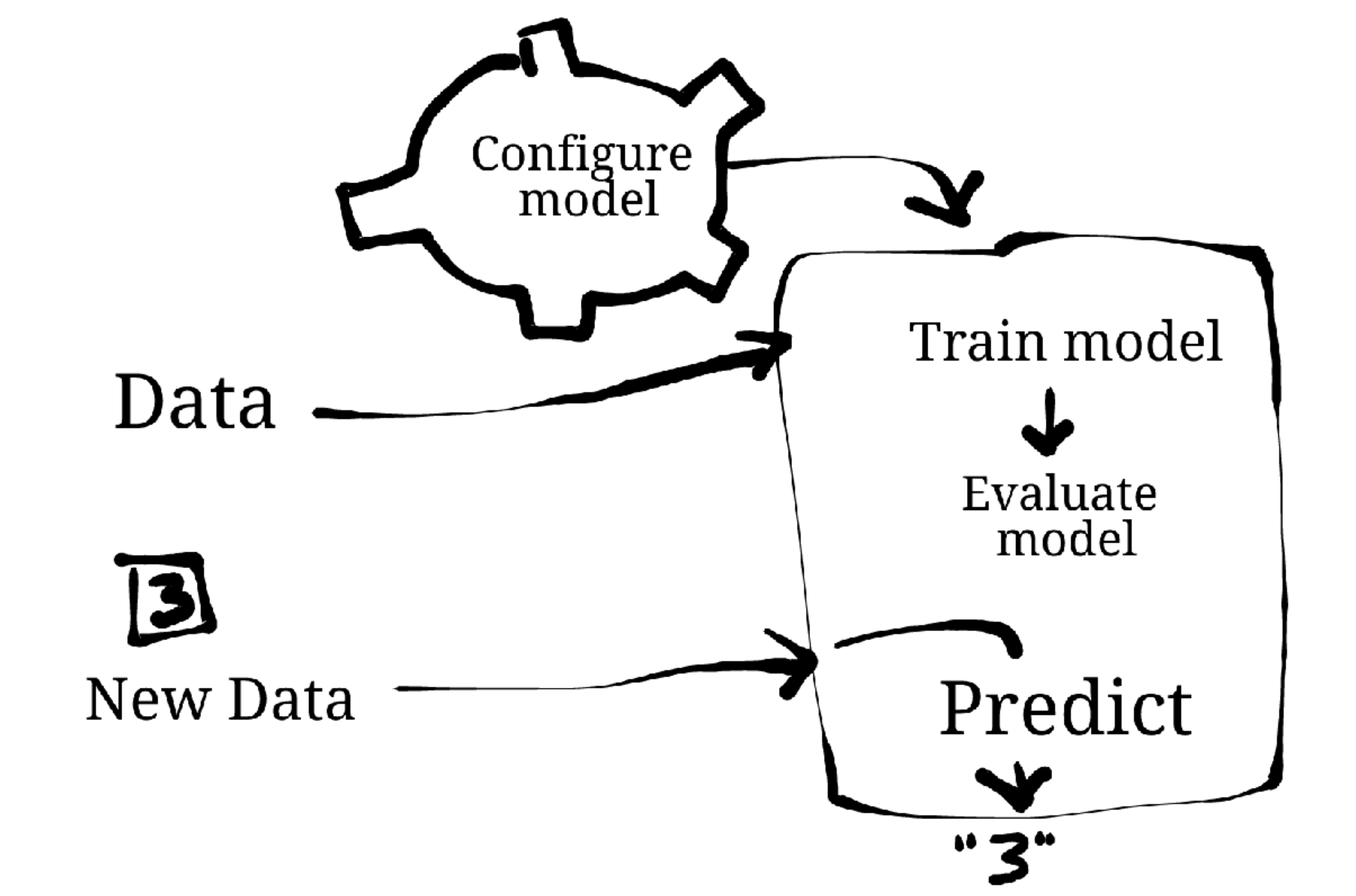
- Learning to map input to output — Architectures
  - CNN
  - RNN
  - And beyond...

DEEP LEARNING 4 J



Load raw data plus labels and transform
...into a record reader
Convert to array and preprocess
9
"9"
INDArray




Configure model
Data
Train model
Evaluate model
3
New Data
Predict
"3"

PART I: NEURONS

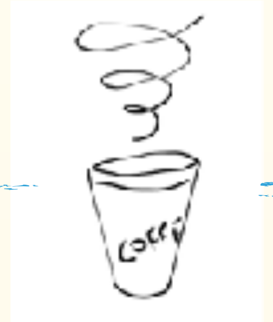
Neurons

- Biological
- Artificial
  - Binary Threshold Neuron
  - Linear Neuron
  - Logistic/sigmoid Neuron
  - reLU
  - Tanh

PART II: LEARNING

Network

- Feed forward — Cost functions
- Back propagation — Training
  - Gradient descent
  - Learning rate
  - Regularisation



PART III: IN A NETWORK

Learning to map input to output — Architectures

- CNN
- RNN
- And beyond…

@KATHARINECODES

PART III: NETWORKS

PART I: NEURONS

- Neurons
  - Biological
  - Artificial
    - Binary Threshold Neuron
    - Linear Neuron
    - Logistic/sigmoid Neuron
    - reLU
    - Tanh

PART II: LEARNING

- Network
  - Feed forward — Cost functions
  - Back propagation — Training
    - Gradient descent
    - Learning rate
    - Regularisation

PART III: IN A NETWORK

- Learning to map input to output — Architectures
  - CNN
  - RNN
  - And beyond…

DATA

PART I:
NEURONS

Neurons

- Biological
- Artificial
  - Binary Threshold Neuron
  - Linear Neuron
  - Logistic/sigmoid Neuron
  - reLU
  - Tanh

PART II:
LEARNING

Network

- Feed forward — Cost functions
- Back propagation — Training
  - Gradient descent
  - Learning rate
  - Regularisation

PART III:
IN A NETWORK

Learning to map input to output — Architectures
- CNN
- RNN
- And beyond…

DATA, DATA, EVERYWHERE….

- **Feature selection**
- Normalising and standardising data
- Dimensionality reduction

FILTERS AND WRAPPERS

- Filters look at each feature in isolation
- Wrappers look at how groups of features perform in a classifier

WHICH IS FASTER? FILTERS OR WRAPPERS?

DATA, DATA, EVERYWHERE….

- Feature selection
- **Normalising and standardising data**
- Dimensionality reduction

DATA, DATA, EVERYWHERE….

- Feature selection
- Normalising and standardising data
- **Dimensionality reduction**

THE DARK SIDE

DEEP LEARNING 4 J

ARCHITECTURES

PART I: NEURONS

- Neurons
  - Biological
  - Artificial
    - Binary Threshold Neuron
    - Linear Neuron
    - Logistic/sigmoid Neuron
    - reLU
    - Tanh

PART II: LEARNING

- Network
  - Feed forward — Cost functions
  - Back propagation — Training
    - Gradient descent
    - Learning rate
    - Regularisation

PART III: IN A NETWORK

- Learning to map input to output — Architectures
  - CNN
  - RNN
  - And beyond…

Feed Forward

Data goes forwards

Recurrent

Data goes forwards… and backwards

Architectures

Feed Forward

Data goes forwards

Autoencoder

Recurrent

Data goes forwards… and backwards

Architectures

Feed Forward

Data goes forwards

Recurrent

Data goes forwards… and backwards

Autoencoder

Convolutional

Architectures

Feed Forward

Data goes forwards

Autoencoder

Convolutional

Recurrent

Data goes forwards… and backwards

Hopfield Nets

Architectures

Feed Forward

Data goes forwards

Autoencoder

Convolutional

Recurrent

Data goes forwards… and backwards

Hopfield Nets

Echo state

Architectures

Feed Forward

Data goes forwards

Autoencoder

Convolutional

Recurrent

Data goes forwards… and backwards

Hopfield Nets

Echo state

Long term short term

Architectures

Feed Forward

Data goes forwards

Autoencoder

Convolutional

Recurrent

Data goes forwards… and backwards

Hopfield Nets

Echo state

Long term short term

Architectures

Memory

Feed Forward

Data goes forwards

Autoencoder

Convolutional

Recurrent

Data goes forwards… and backwards

Hopfield Nets

Echo state

Long term short term

Architectures

Memory

Spiking

CNNS

PART I: NEURONS

- Neurons
  - Biological
  - Artificial
    - Binary Threshold Neuron
    - Linear Neuron
    - Logistic/sigmoid Neuron
    - reLU
    - Tanh

PART II: LEARNING

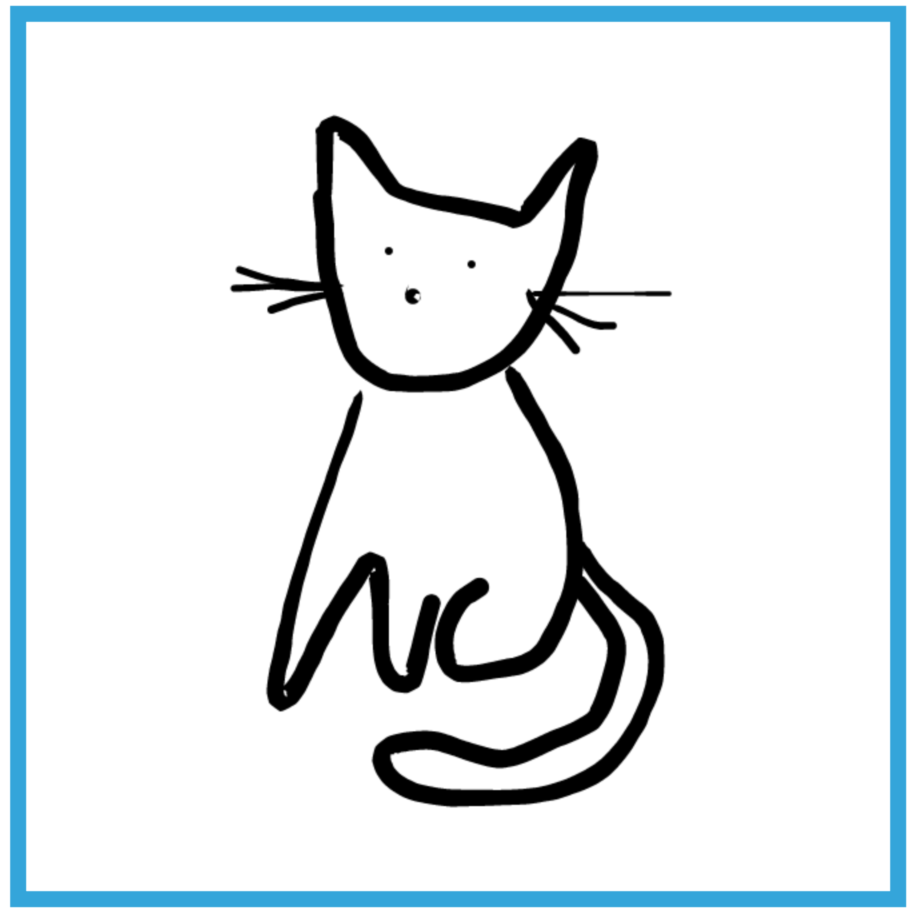
- Network
  - Feed forward
    - Cost functions
  - Back propagation
    - Training
      - Gradient descent
      - Learning rate
      - Regularisation

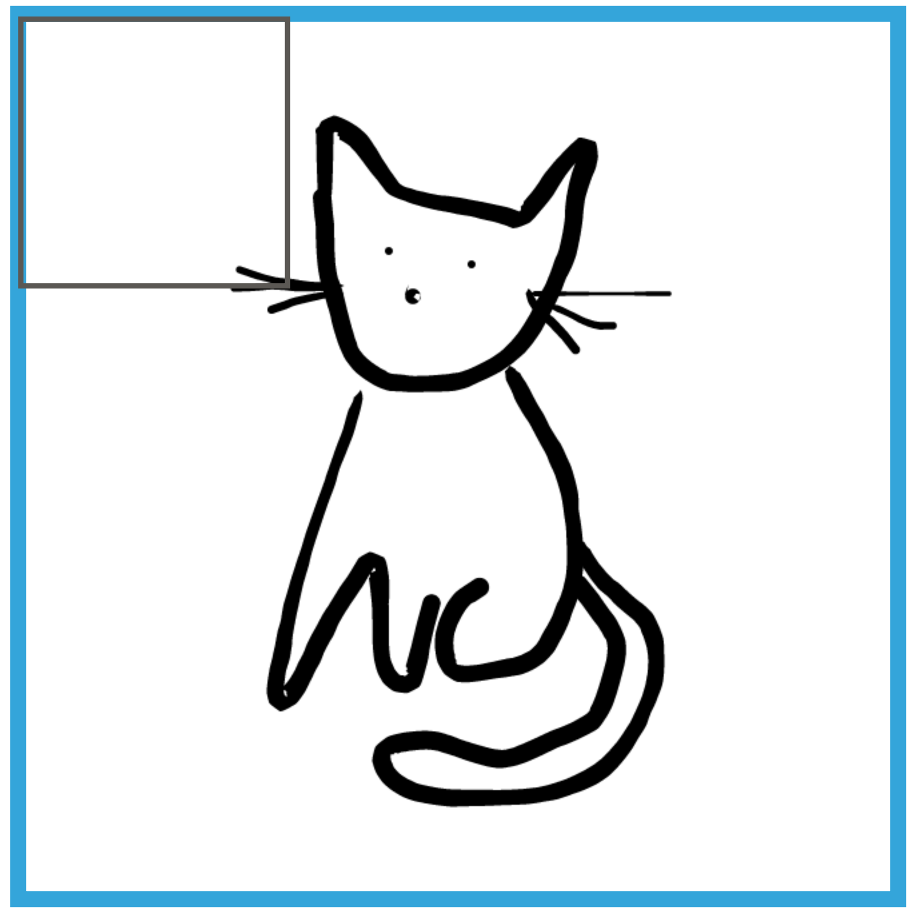
PART III: IN A NETWORK

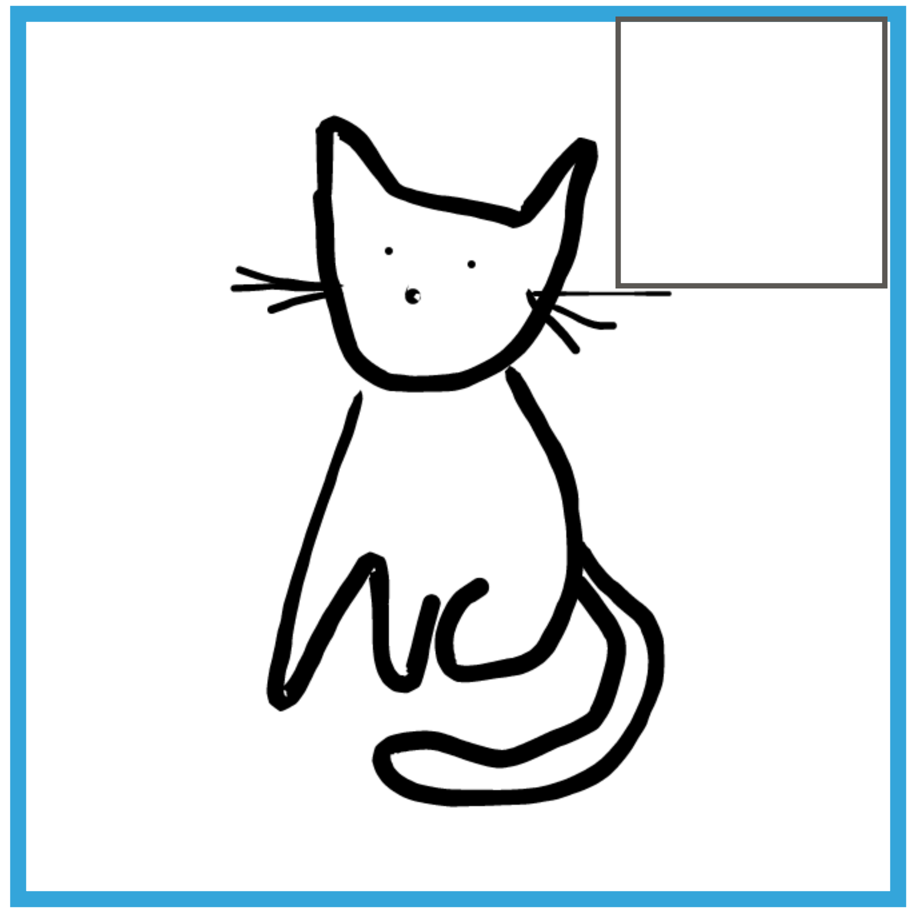
- Learning to map input to output
  - Architectures
    - CNN
    - RNN
    - And beyond…

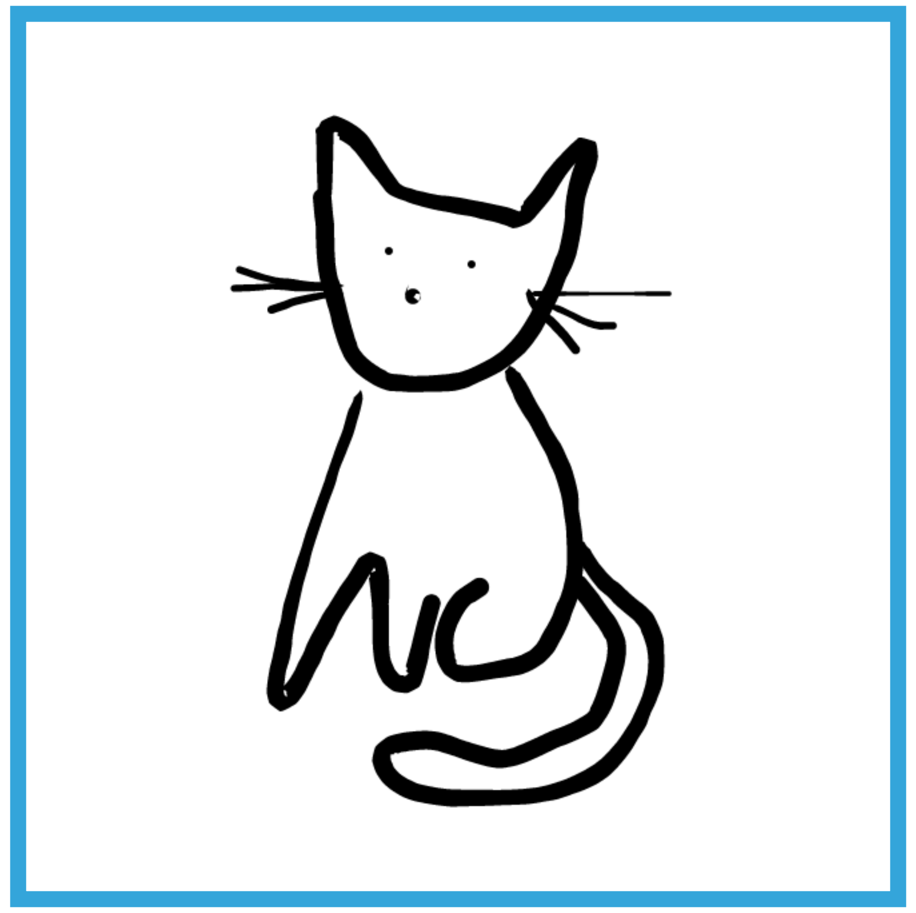
CNNS - EXAMPLE

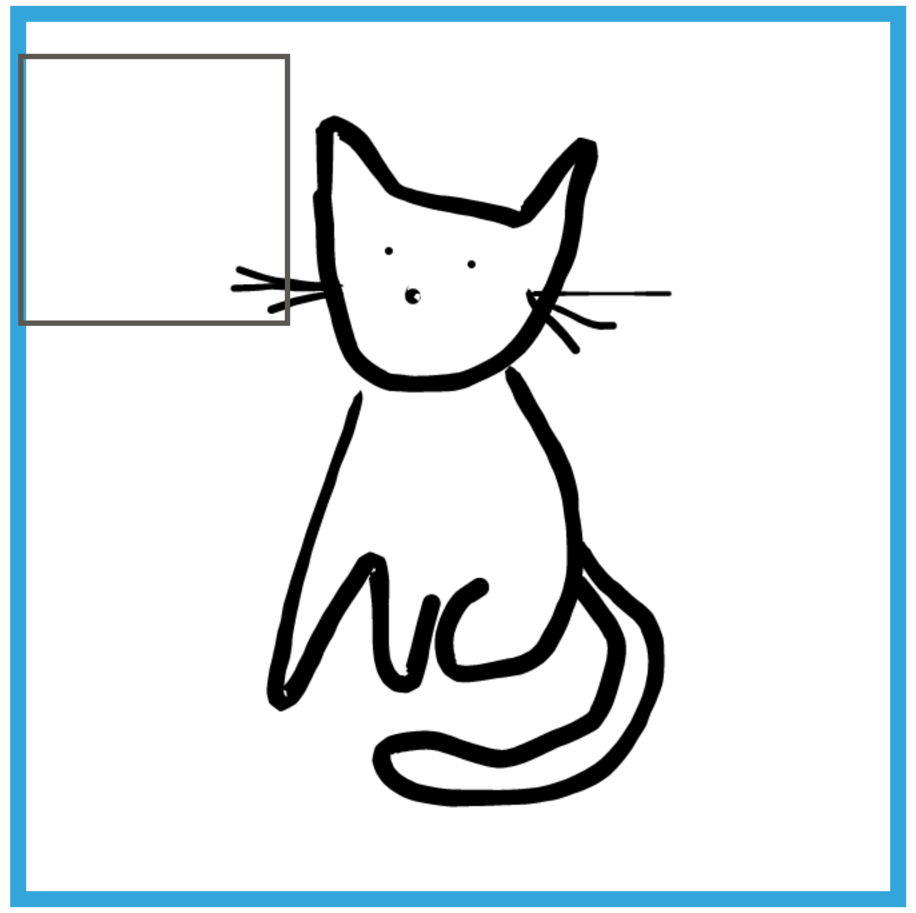
- **Input an image (e.g. 32x32x3)**
- **Convolutional layer**
- ReLU layer
- Pool layer
- Fully connected layer

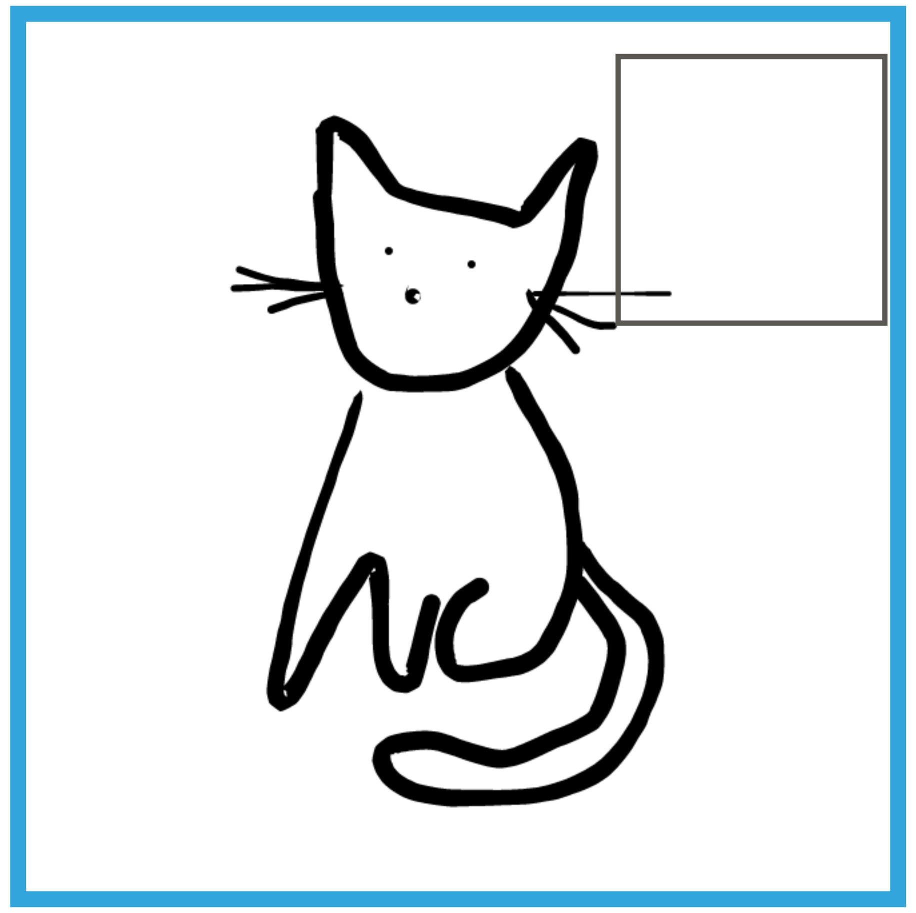


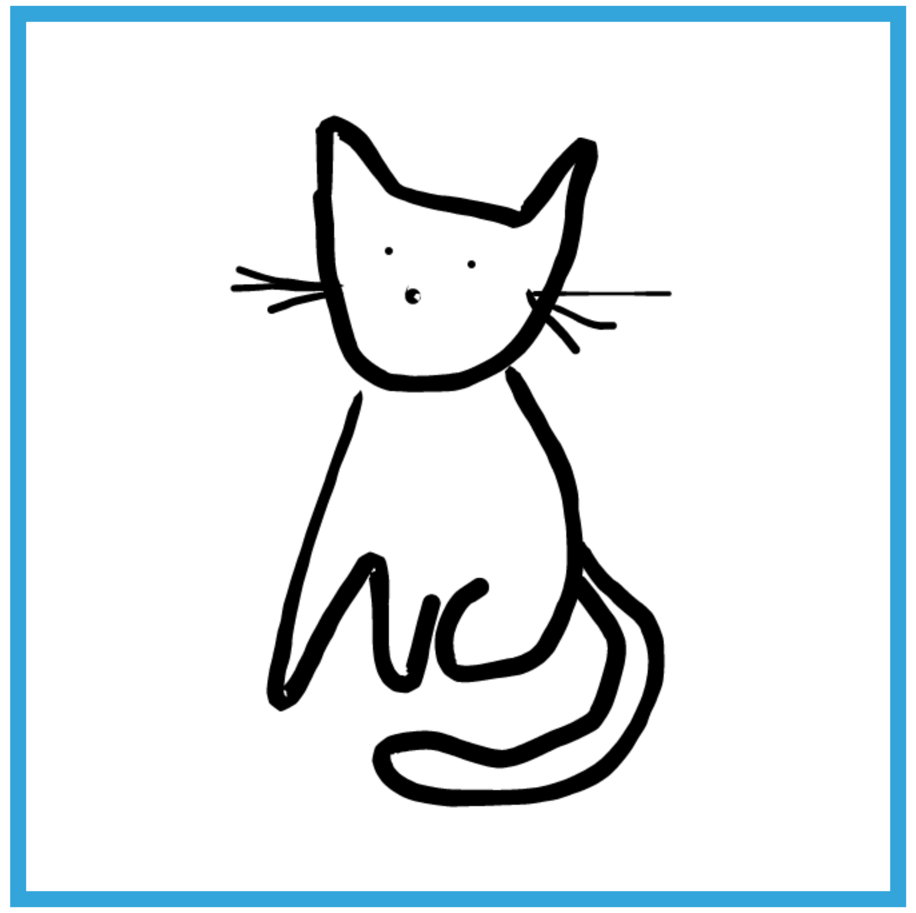


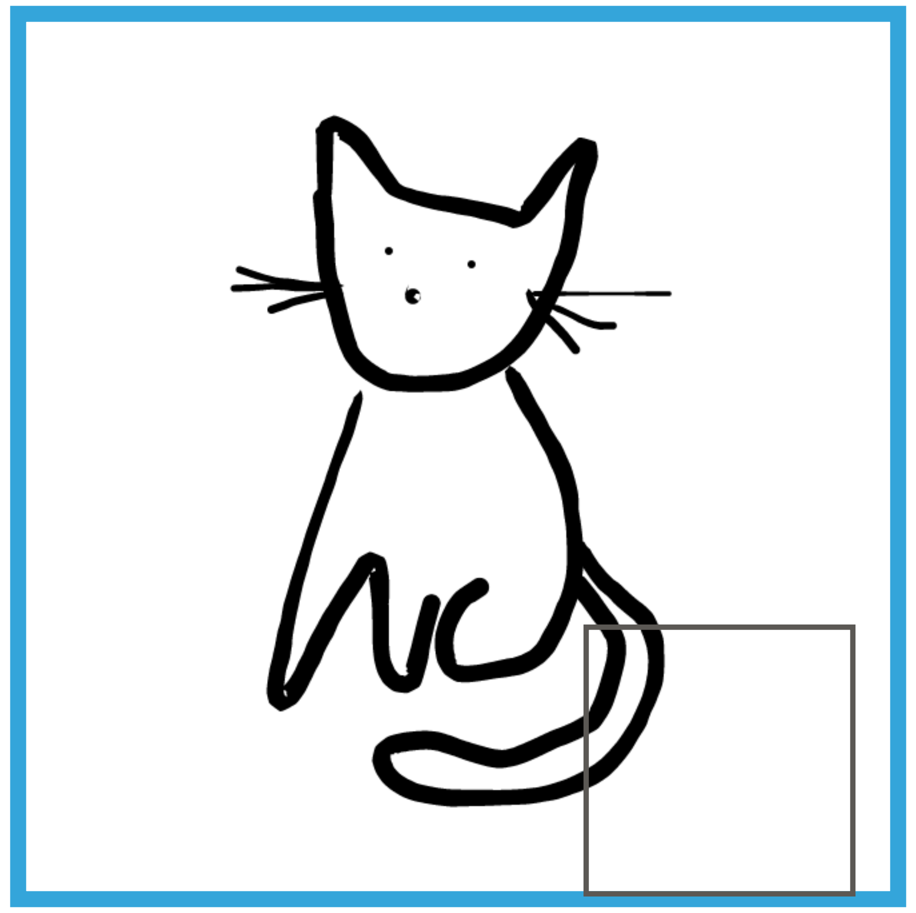


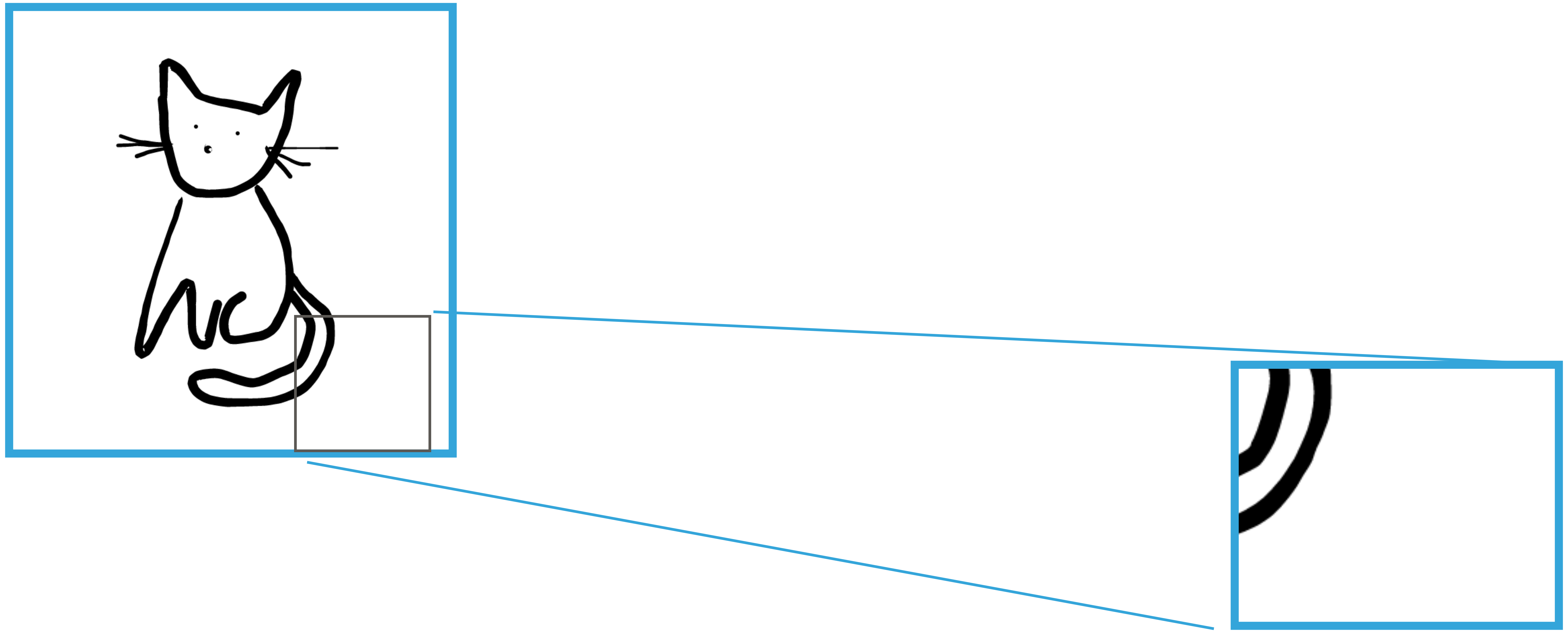




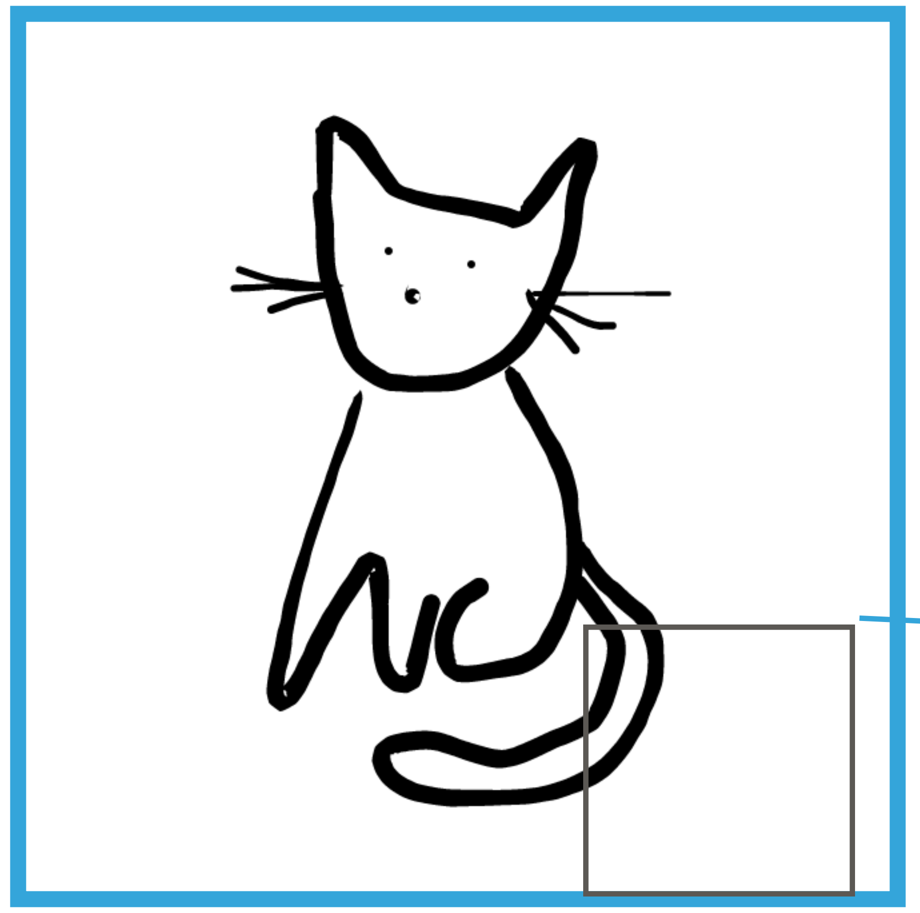






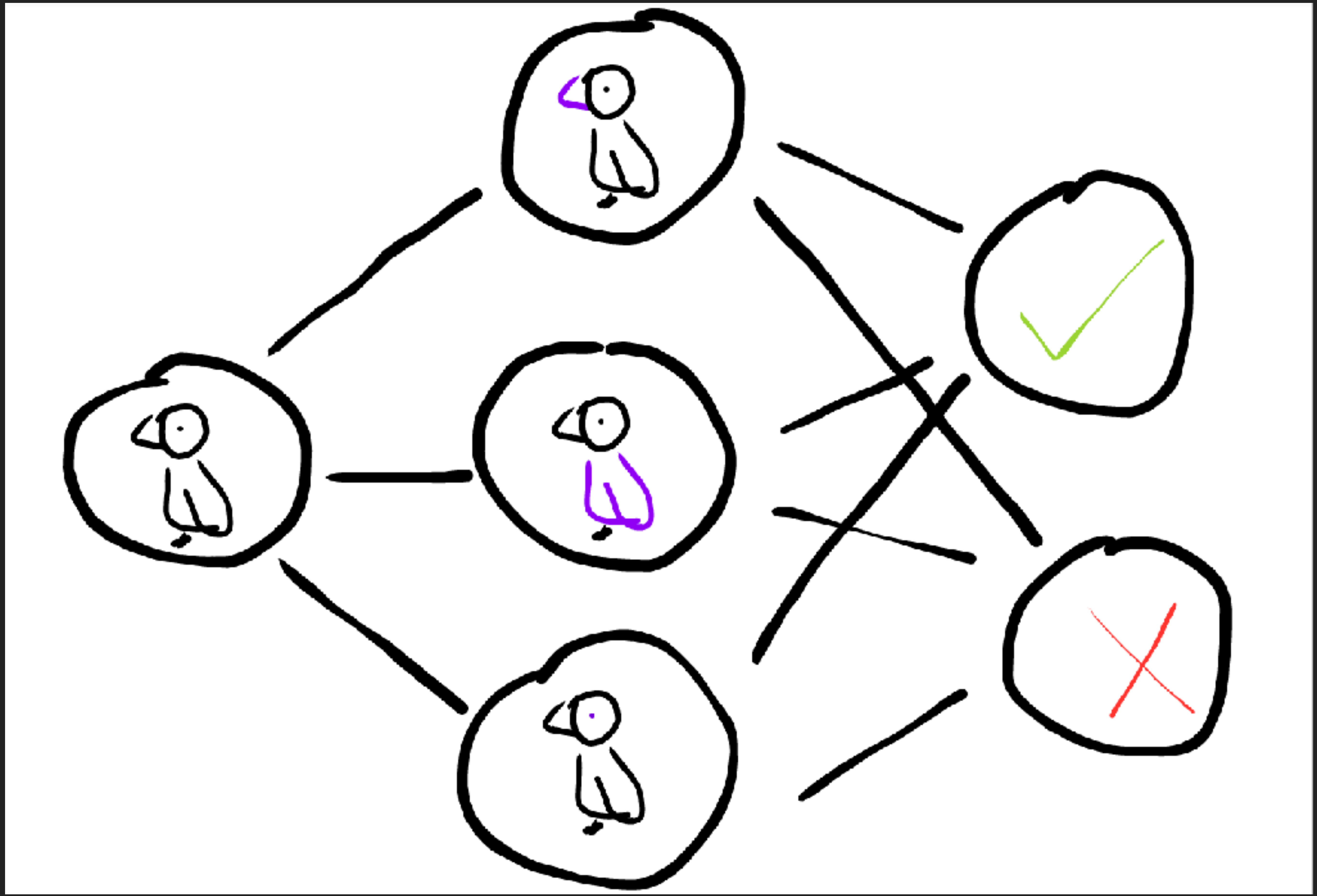


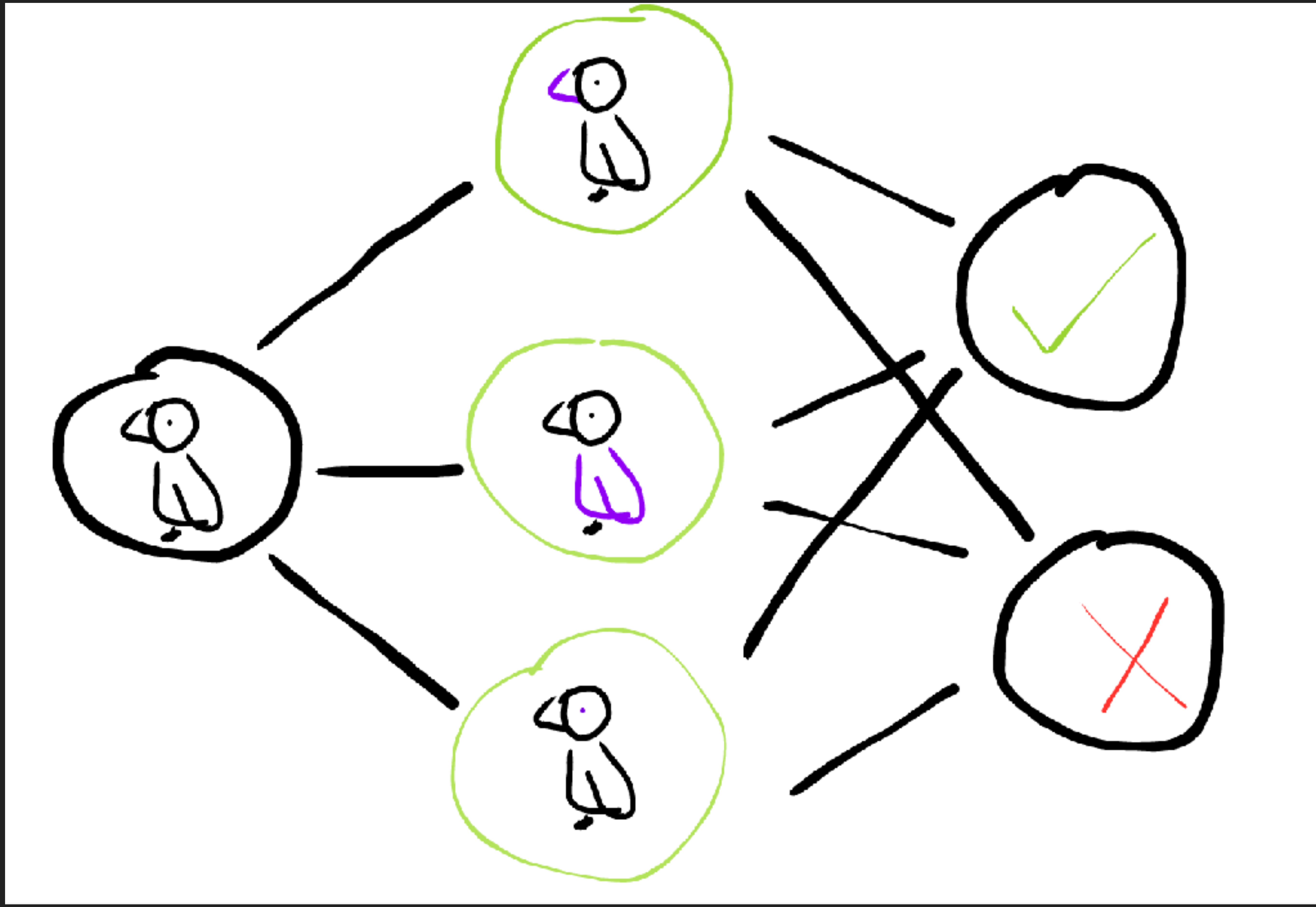
0	70	0	70	0	0	0
0	70	0	70	0	0	0
0	70	0	70	0	0	0
70	0	70	0	0	0	0
0	70	0	0	0	0	0
70	0	0	0	0	0	0

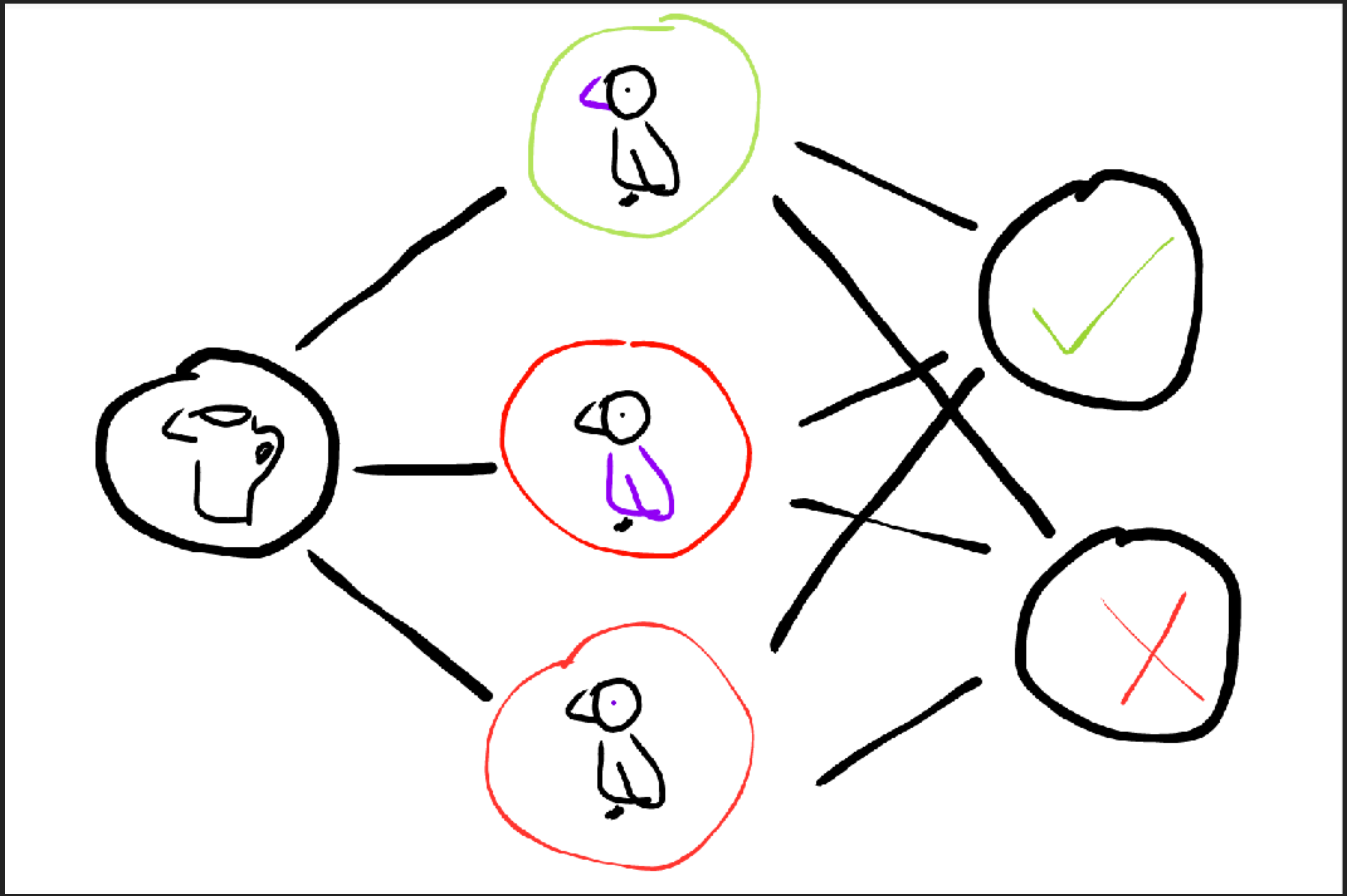


CNNS - EXAMPLE

- Input an image (e.g. 32x32x3)
- Convolutional layer
- **ReLU layer**
- **Pool layer**
- **Fully connected layer**







- What do we want? Some kind of internal structure, for the task domain.
- Trying to learn representations. Deciding what units should represent.
- *Internal representations*

RNNS

PART I: NEURONS

- Neurons
  - Biological
  - Artificial
    - Binary Threshold Neuron
    - Linear Neuron
    - Logistic/sigmoid Neuron
    - reLU
    - Tanh

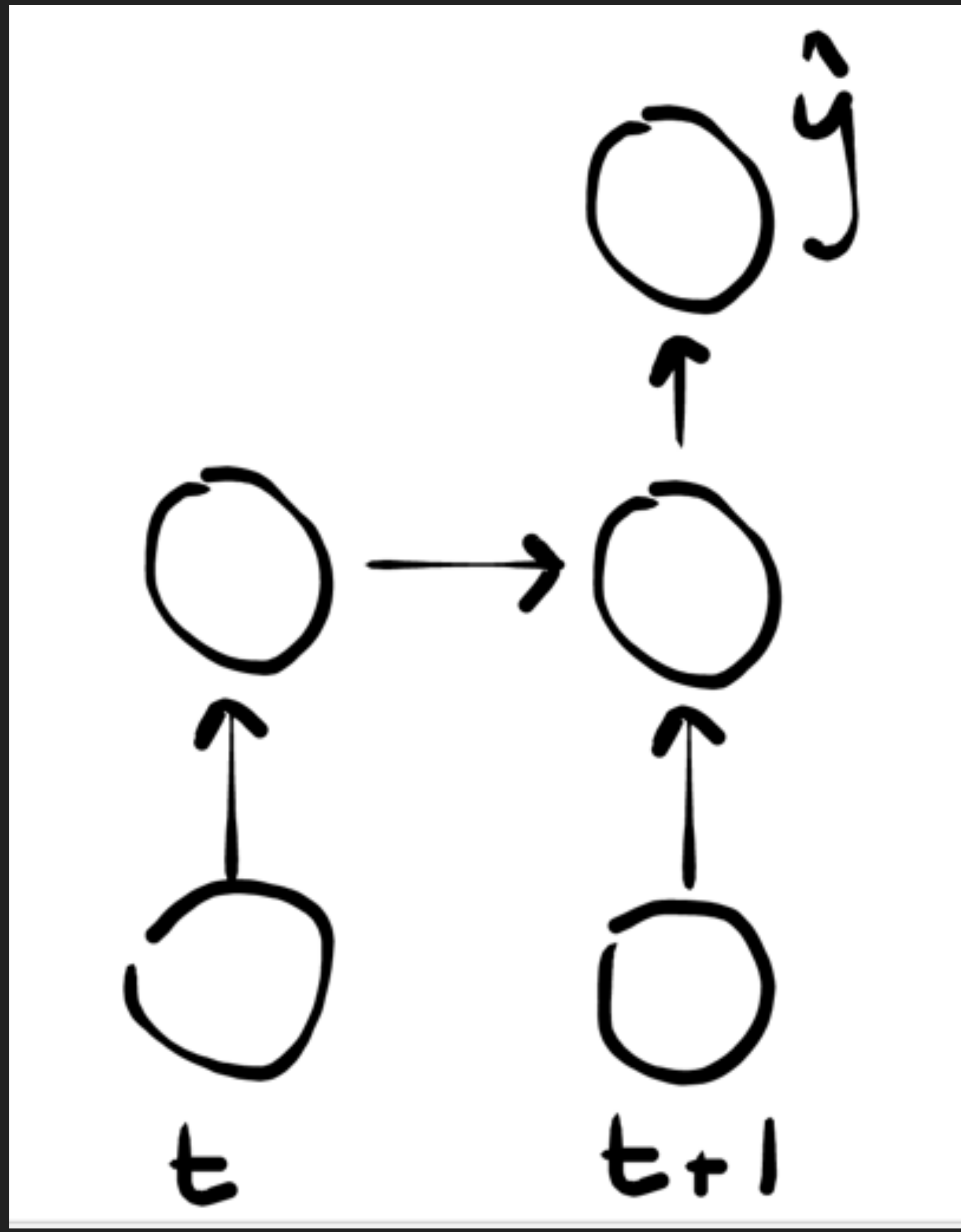
PART II: LEARNING

- Network
  - Feed forward
    - Cost functions
  - Back propagation
    - Training
      - Gradient descent
      - Learning rate
      - Regularisation

PART III: IN A NETWORK

- Learning to map input to output
  - Architectures
    - CNN
    - RNN
    - And beyond…

- Learn from the present, and the past
- Said to have memory



AND BEYOND...

PART I: NEURONS

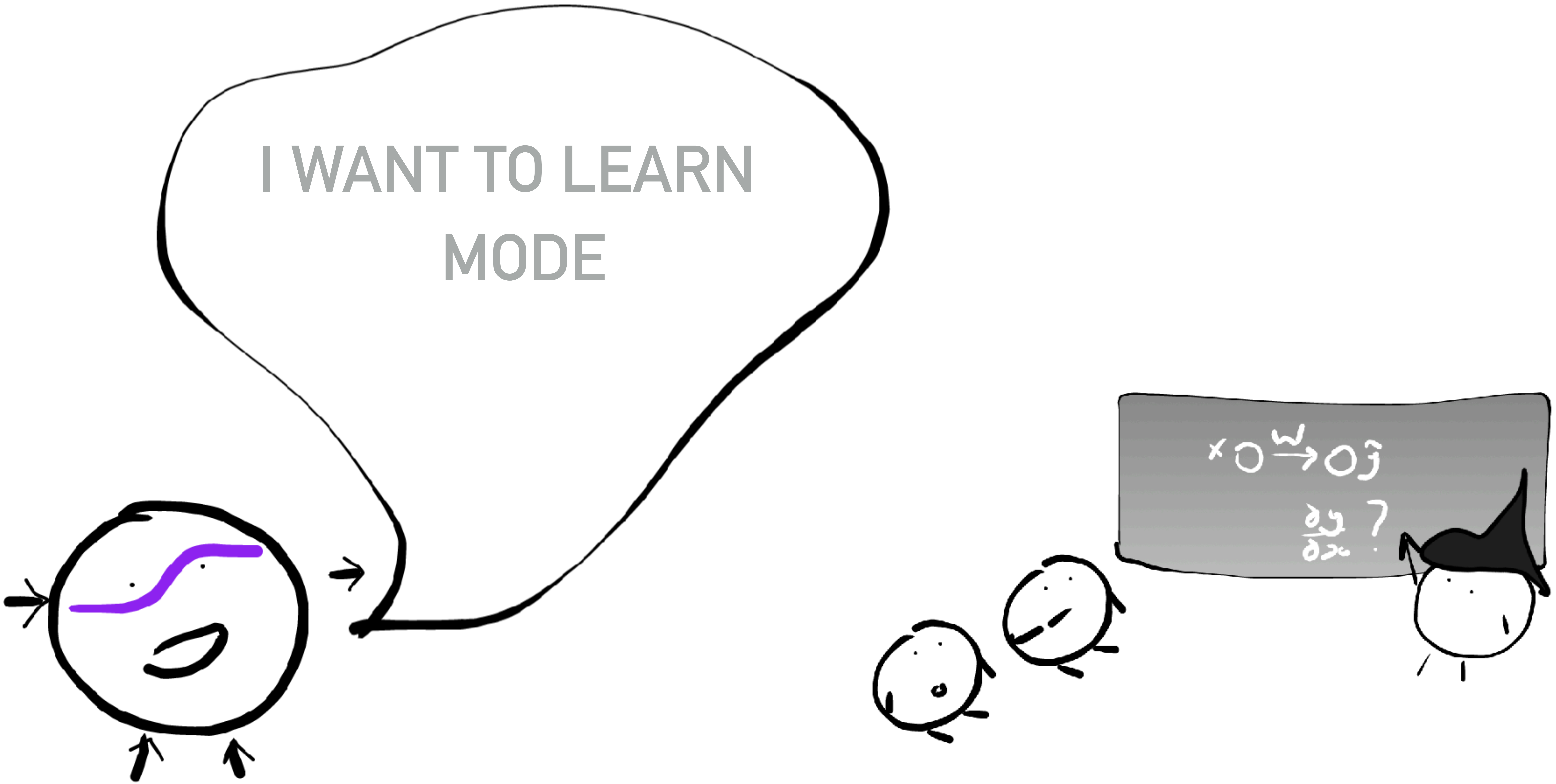
- Neurons
  - Biological
  - Artificial
    - Binary Threshold Neuron
    - Linear Neuron
    - Logistic/sigmoid Neuron
    - reLU
    - Tanh

PART II: LEARNING

- Network
  - Feed forward — Cost functions
  - Back propagation — Training
    - Gradient descent
    - Learning rate
    - Regularisation

PART III: IN A NETWORK

- Learning to map input to output — Architectures
  - CNN
  - RNN
  - And beyond...


I WANT TO LEARN MODE

LEARNING PATHS

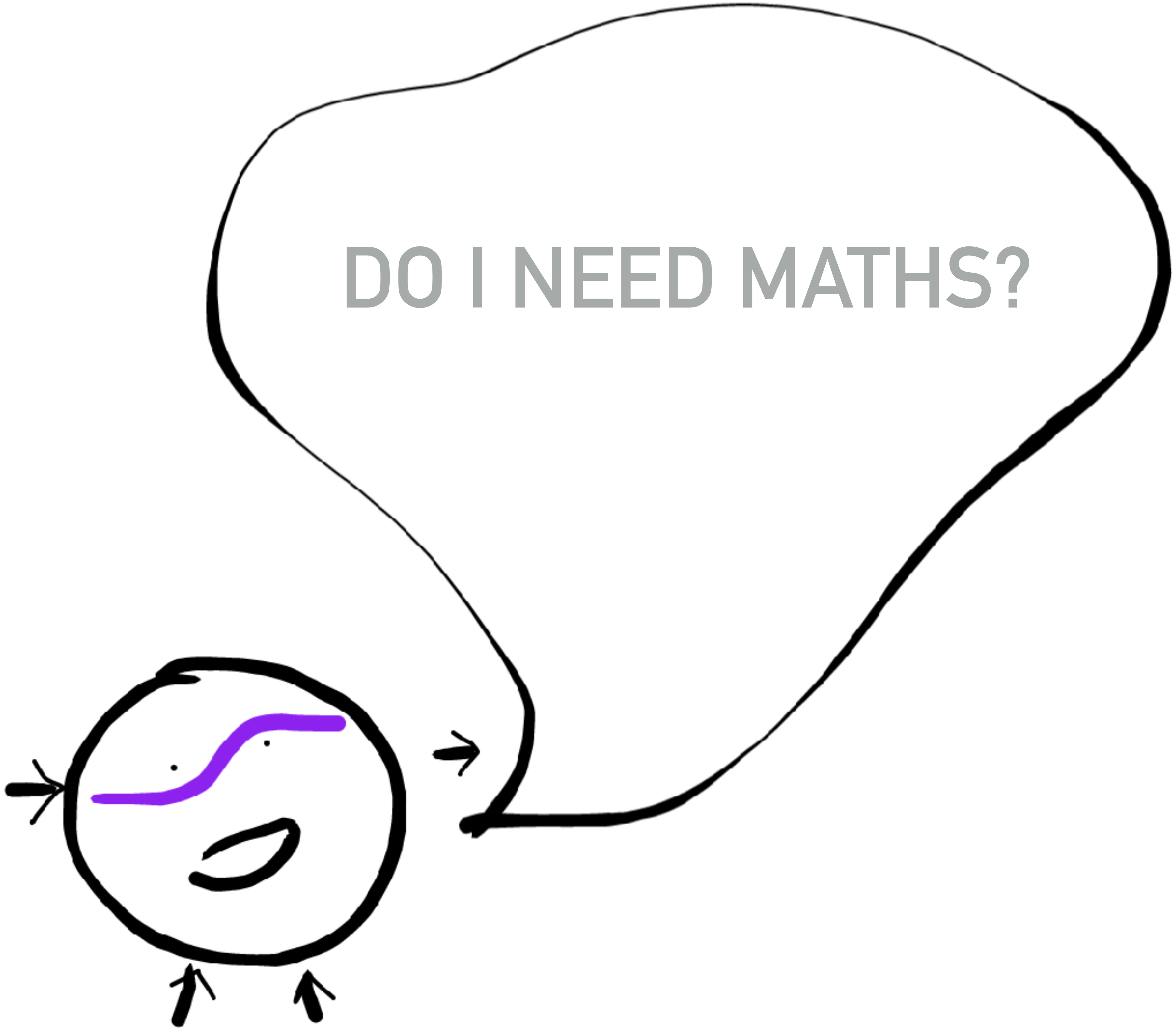
GENERAL MACHINE LEARNING: START WITH ANDREW NG (STANFORD ONLINE) ON COURSERA

NEURAL NETWORKS: START WITH GEOFFREY HINTON (UNIVERSITY OF TORONTO) ON COURSERA
WARNING: CONTAINS MATHS

https://katharinecodes.wordpress.com/2017/08/31/machine-learning-journey-before-the-masters/

$$\frac{1}{1+e^{-z}}$$

$$\sum \beta_0 + \beta_1 x$$



$$\frac{\partial}{\partial x}$$

$$\theta^T X$$

DO I NEED MATHS?

DO I NEED MATHS?

Equation of a straight line

Quadratics

DO I NEED MATHS?

Equation of a straight line

Quadratics

Differentiation

DO I NEED MATHS?

Equation of a straight line

Quadratics

Differentiation

DO I NEED MATHS?

Partial differentiation

Equation of a straight line

Quadratics

Differentiation

DO I NEED MATHS?

Partial differentiation

Logaritms

Equation of a straight line

Quadratics

Differentiation

DO I NEED MATHS?

Partial differentiation

Logaritms

Matrices

Equation of a straight line

Quadratics

Differentiation

DO I NEED MATHS?

Partial differentiation

Exponentials

Logaritms

Matrices

Equation of a straight line

Quadratics

Differentiation

DO I NEED MATHS?

Partial differentiation

Exponentials

Logaritms

Vectors

Matrices

Equation of a straight line

Distance measures

Quadratics

Differentiation

DO I NEED MATHS?

Partial differentiation

Exponentials

Logaritms

Matrices

Vectors

Equation of a straight line

Distance measures

Quadratics

Differentiation

DO I NEED MATHS?

Partial differentiation

Exponentials

Eigen vectors

Logaritms

Vectors

Matrices

Equation of a straight line

Distance measures

Quadratics

Probability

Differentiation

DO I NEED MATHS?

Partial differentiation

Exponentials

Eigen vectors

Logaritms

Vectors

Matrices

Equation of a straight line

Distance measures

Quadratics

Probability

Differentiation

DO I NEED MATHS?

Hessian matrices

Partial differentiation

Exponentials

Eigen vectors

Logaritms

Vectors

Matrices

Equation of a straight line

Bayes Theorem

Distance measures

Quadratics

Probability

Differentiation

DO I NEED MATHS?

Hessian matrices

Partial differentiation

Exponentials

Eigen vectors

Logaritms

Vectors

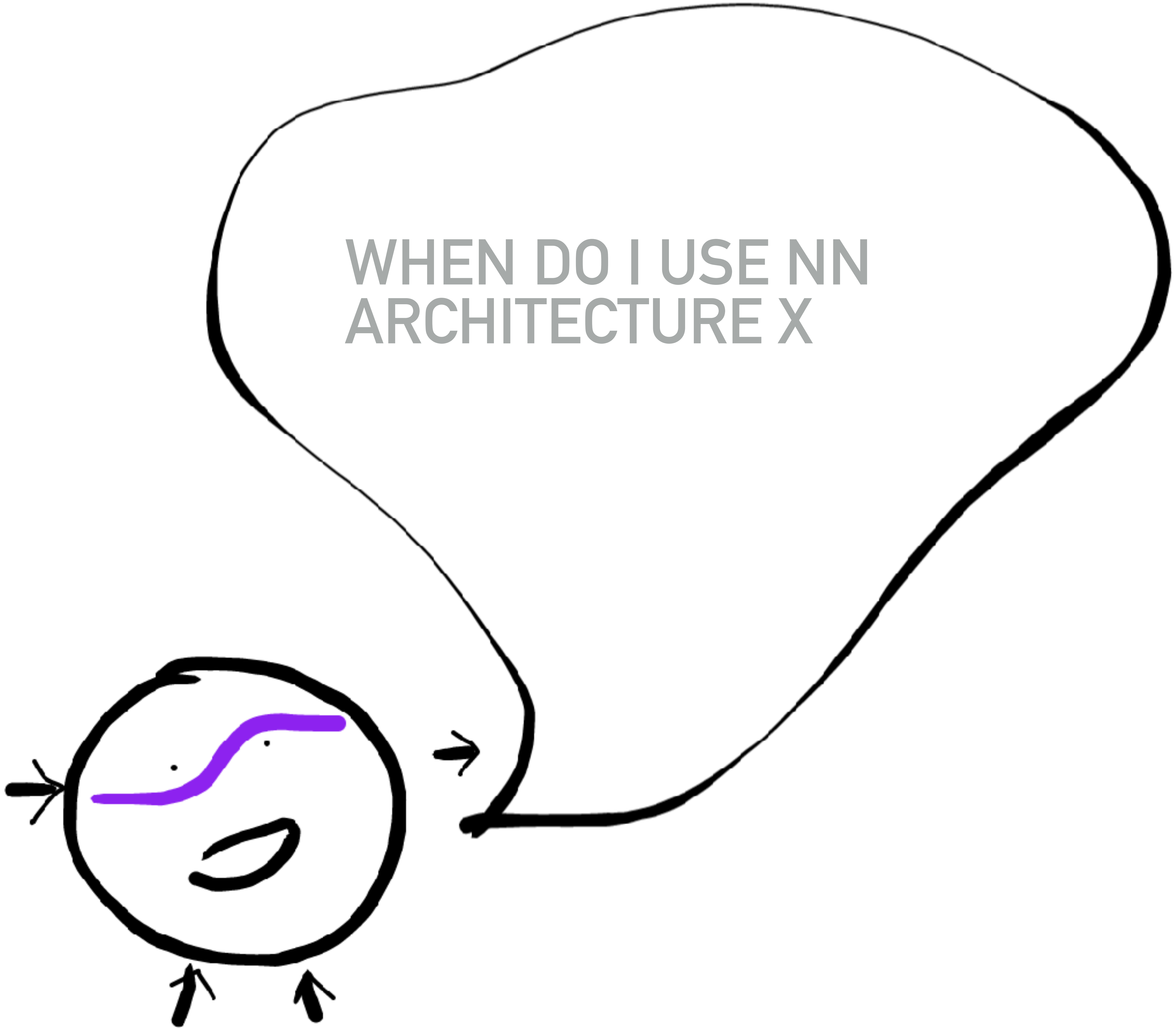
Matrices

```
np.sqrt(np.sum((y-pred-y**2))
```


```
Epoch e = new Epoch();
try {
    e.play();
} ...
```


x?
Привіт → x?
WHAT SHOULD MY DATA LOOK LIKE?
x?


I'M TRYING TO SOLVE THIS PROBLEM,
WHERE DO I START


WHEN DO I USE NN
ARCHITECTURE X