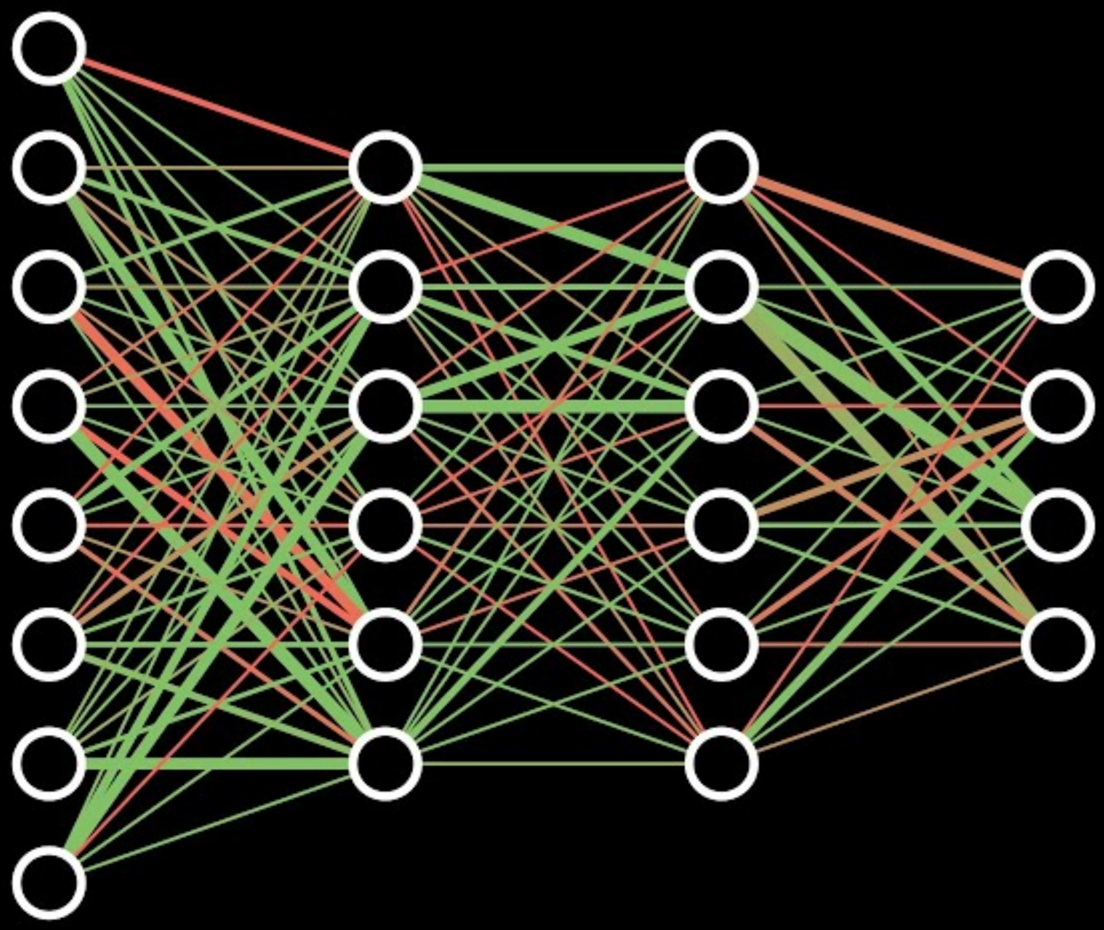
Do not distribute course material

Neural Networks



From the ground up

http://deeplearning.stanford.edu/tutorial/ and then go to MultiLayerNeuralNetworks
http://neuralnetworksanddeeplearning.com/

Topic 6
Neural Networks

INTRODUCTION TO MACHINE LEARNING

PROF. LINDA SELLIE

Some of these slides are from Prof. Rangan

Overview

Used for both regression and classification

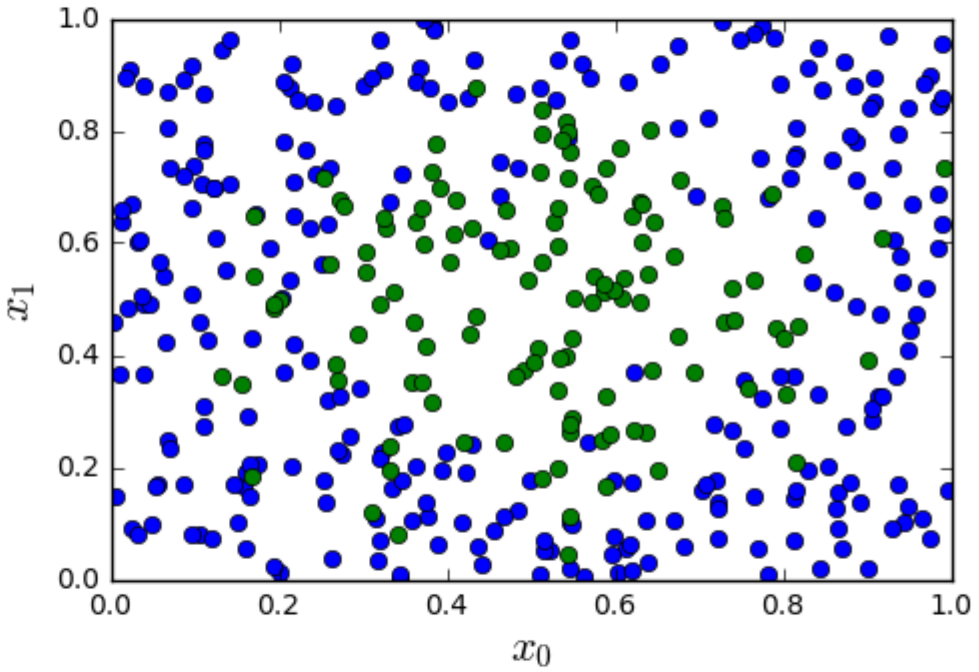
Neural networks ***extend*** logistic regression and linear regression

Neural networks are universal approximators (it is possible to approximate any bounded function)

Outline

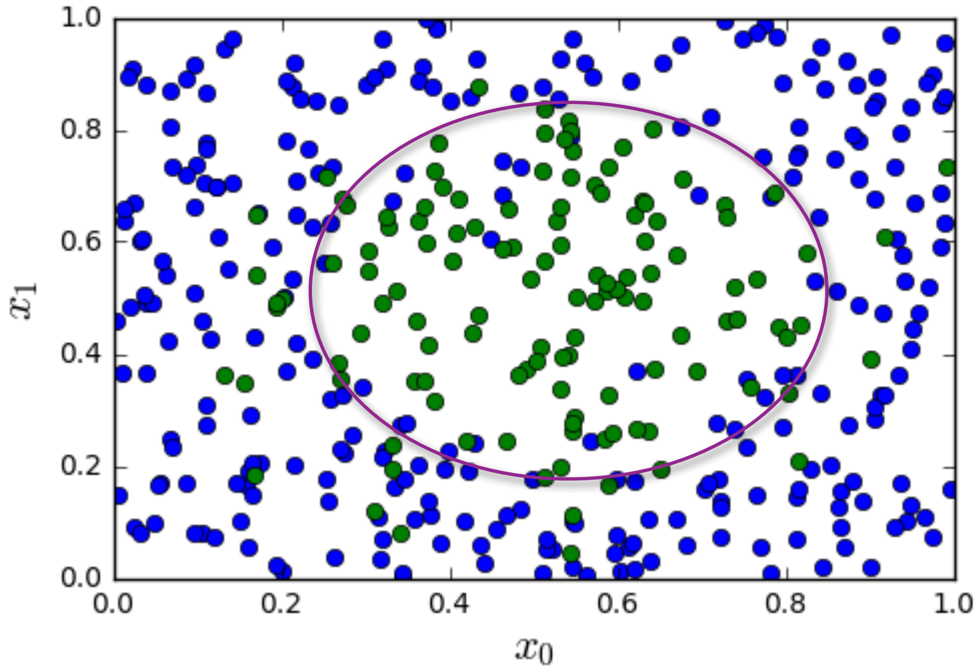
- ❑ Motivation Introduction to neurons
- ❑ Nonlinear classifiers from linear features
- ❑ Neural networks notation
- ❑ Pseudocode for prediction
- ❑ Training a neural network
- ❑ Implementing gradient descent for neural networks
  - Vectorization
  - Pseudocode
- ❑ Preprocessing
- ❑ Initialization
- ❑ Activations

Most Datasets are not Linearly Separable



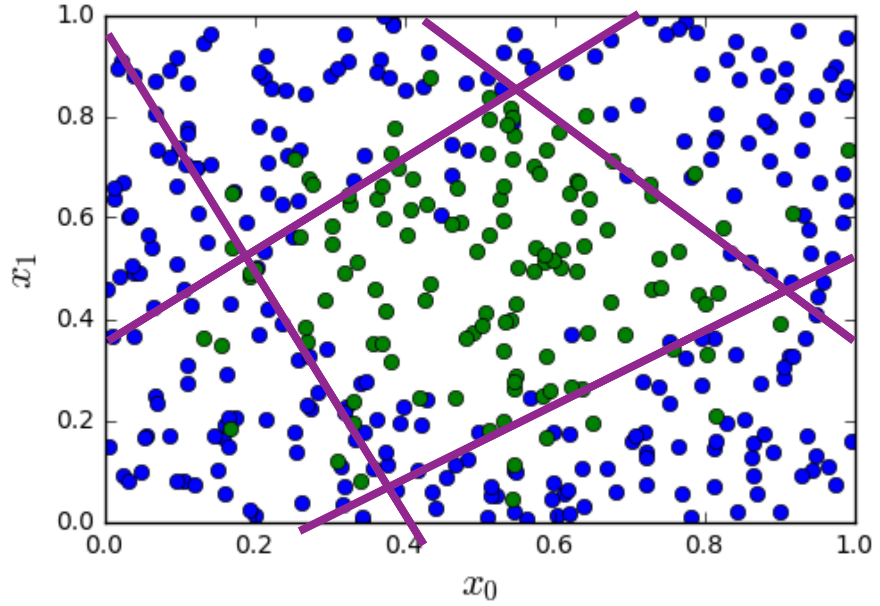
- Consider simple synthetic data
  - See figure to the left
  - 2D features
  - Binary class label
- Not separated linearly
- We can use Logistic Regression with non-linear features

From Linear to Nonlinear



- Idea: Build nonlinear region from linear decisions

From Linear to Nonlinear



- Idea: Build nonlinear region from linear decisions

How can we learn the right feature transformations (aka ***functions*** to transform our features)?

This is what neural networks do! We input the original features and the network learns different function of the features

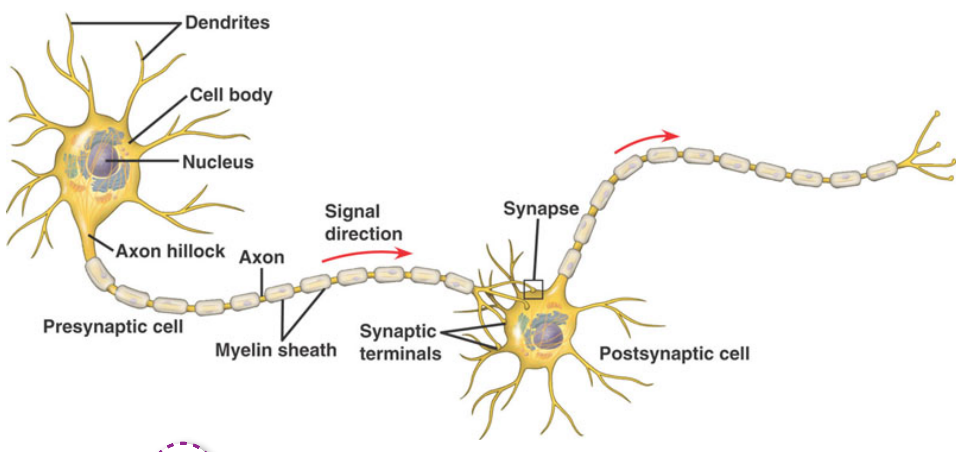
THAT IS THE MAIN TOPIC OF THIS LECTURE

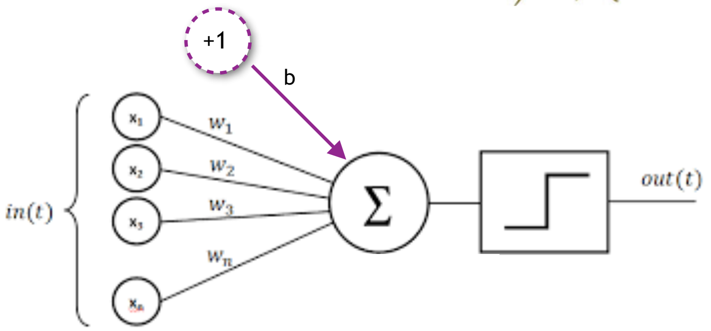
Recent Resurgence and Developments

- In 1980's neural networks were used.
- However interest in them declined due to computation power and not enough data
- Advances in hardware (GPU and high performance computation, etc) and the massive amounts of data being collected have removed has significantly increased the use of neural networks since 2012
- The past 5-7 years have seen algorithmic innovations. Many of these are used in self-driving cars, facial recognition, speech recognition, etc.
- Recently, deep learning has had the most impact to expand what is possible to learn. (Deep learning is just neural networks with many "layers". Later in today's class we will discuss what a "layer" is)

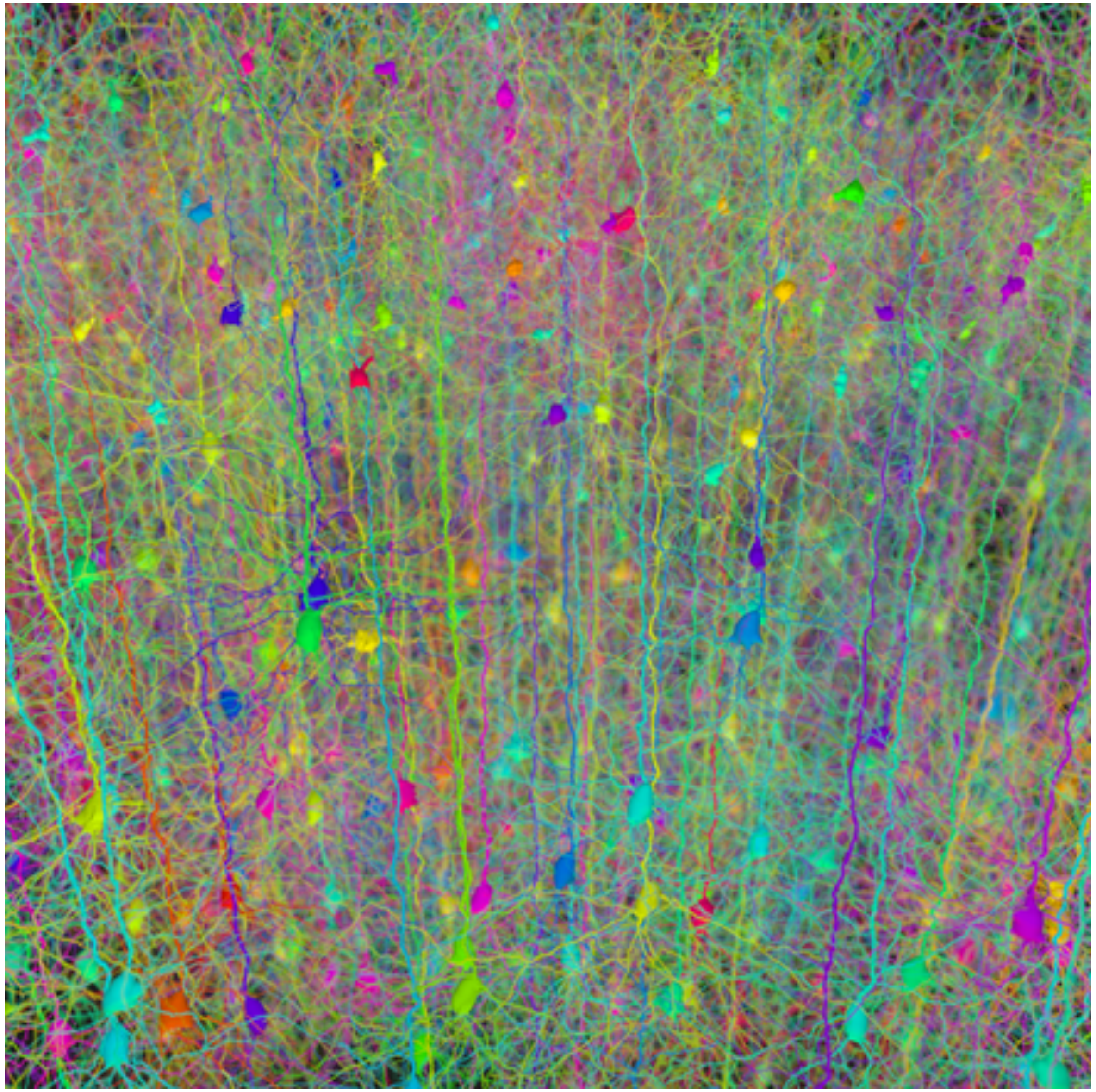
Inspiration from Biology

Abstract away biological construct into a mathematical construct



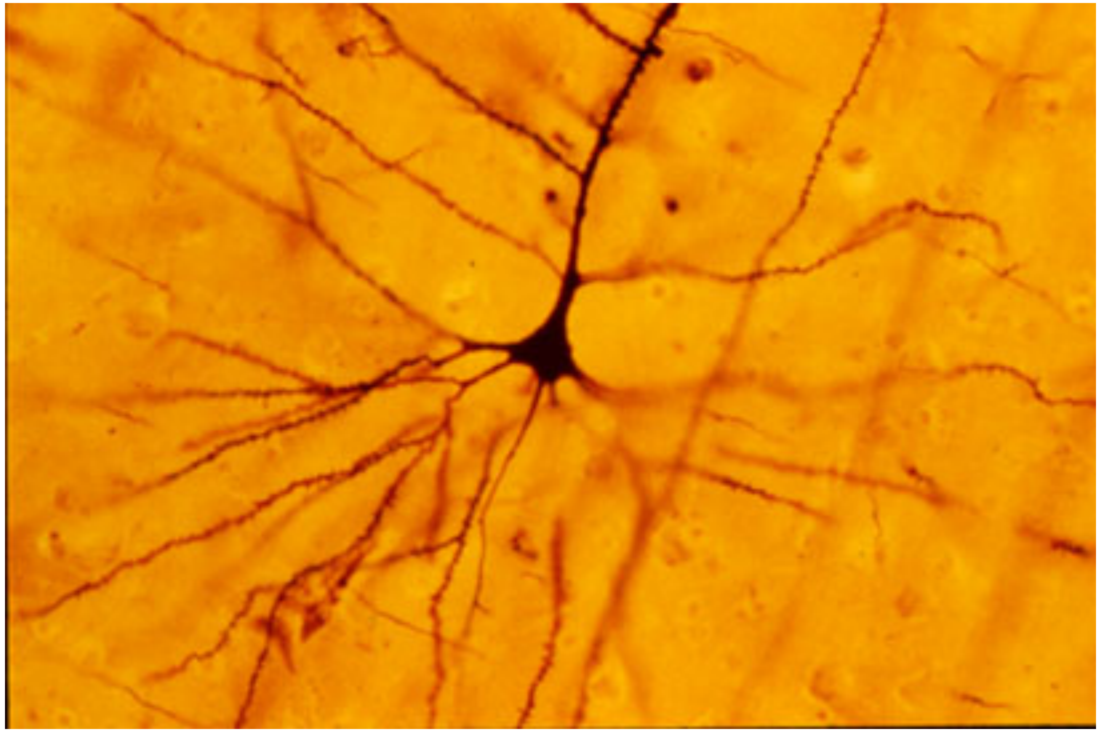


- *Simple* model of neurons
  - Dendrites: Input currents from other neurons
  - Soma: Cell body, accumulation of charge
  - Axon: Outputs to other neurons
  - Synapse: Junction between neurons
- Operation:
  - Take weighted sum of input current
  - Outputs when sum reaches a threshold
- Each neuron is like one unit in neural network
- No one knows how the brain really works - but just like people were inspired by birds to build airplanes, our neurons do not work the same but are inspired by the neurons in our brains



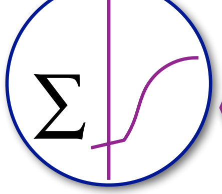
Computer simulation of the branching architecture of the dendritesof pyramidal neurons.[6]
https://en.wikipedia.org/wiki/Neural_network

"**Pyramidal cells**, or **pyramidal neurons**, are a type of multipolar neuronfound in areas of the brain including the cerebral cortex, the hippocampus, and the amygdala"



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

Perceptron / Neuron



Each node has a left and right side. The left side is a weighted linear sum, the right size is a non-linear function

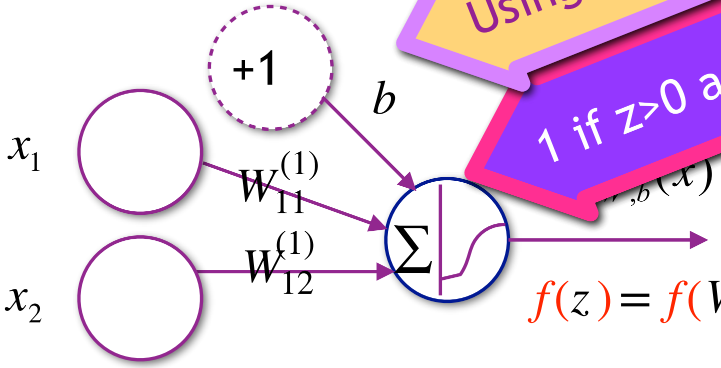
r this lecture will use
e sigmoid function

“A **perceptron** is a simple model of a biological ne an artificial neural network. ... The **perceptron algorithm** was designed to classify visual inputs, categorizing subjects i types and separating groups with a line.“

What kind of decision bound ” create?

Using b instead of w_0

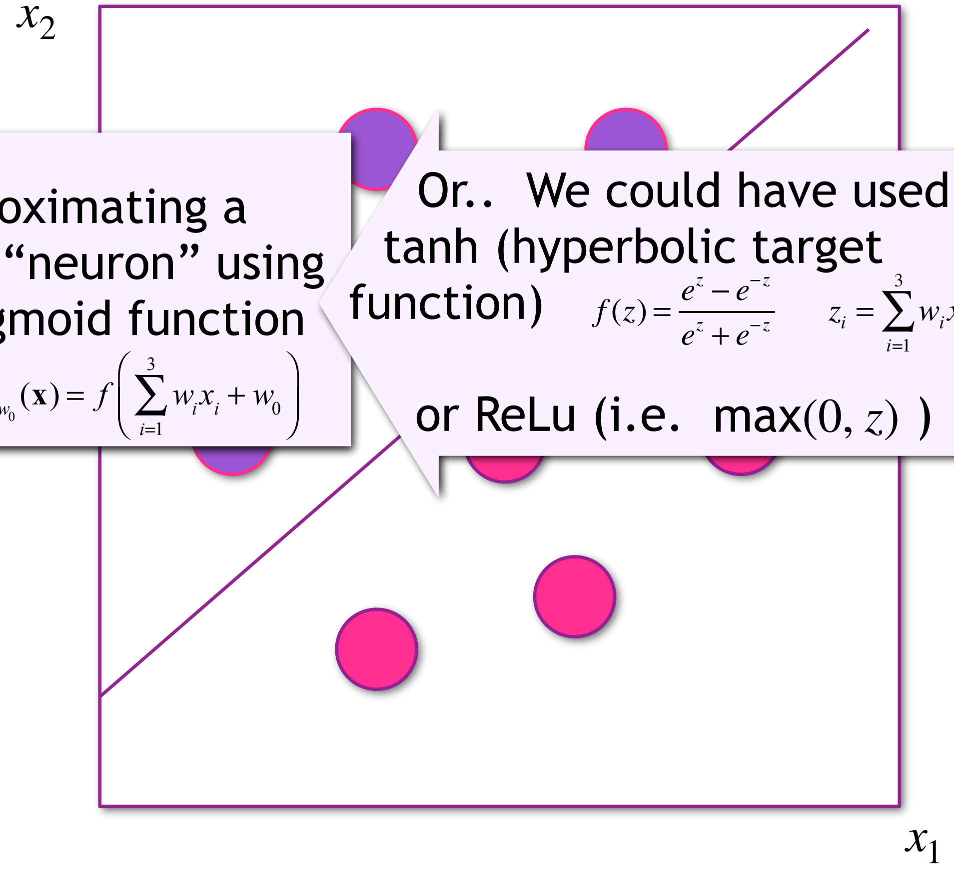
1 if z>0 and 0 if z≤0



$$h_{W,b}(x) = f\left(\sum_{i=1}^{2} W_{1i}^{(1)} x_i + b\right)$$

$$f(z) = f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + b\ )$$

$$z = [W_{11}^{(1)}, W_{12}^{(1)}]\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + b$$



approximating a single “neuron” using the sigmoid function

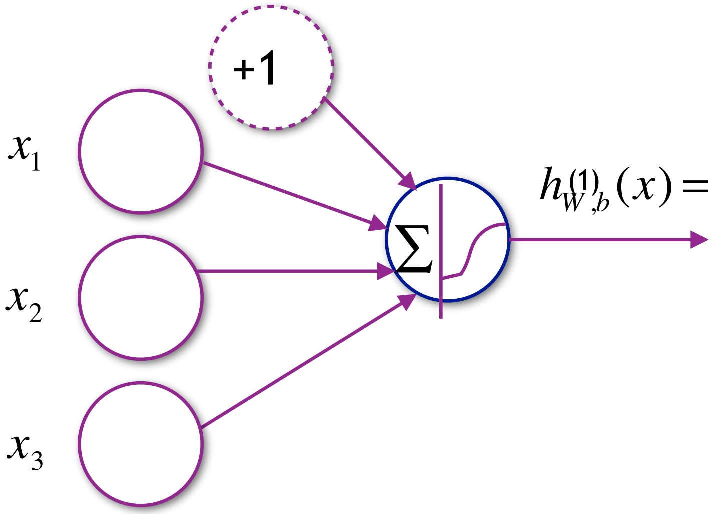
$$h_{\mathbf{w},w_0}(\mathbf{x}) = f\left(\sum_{i=1}^{3} w_i x_i + w_0\right)$$

Or.. We could have used the tanh (hyperbolic target function) $f(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$ $z_i = \sum_{i=1}^{3} w_i x_i + b$

or ReLu (i.e. max$(0, z)$) or ...

New Notation

Unfortunately, there will be quite a bit of notation.....
We will follow the notation from http://deeplearning.stanford.edu/wiki/index.php/Neural_Networks



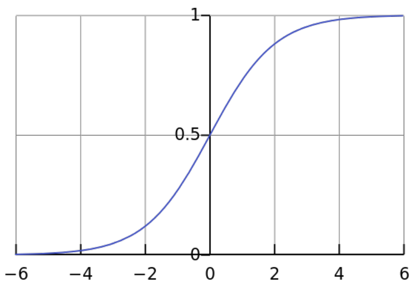
$$h_{W,b}^{(1)}(x) = f\left(\sum_{i=1}^{3} W_{1i}^{(1)} x_i + b\right)$$

$$W^T = \begin{bmatrix} W_{11}^{(1)} \\ W_{12}^{(1)} \\ W_{13}^{(1)} \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$z = [W_{11}^{(1)}, W_{12}^{(1)}, W_{13}^{(1)}] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + b$$

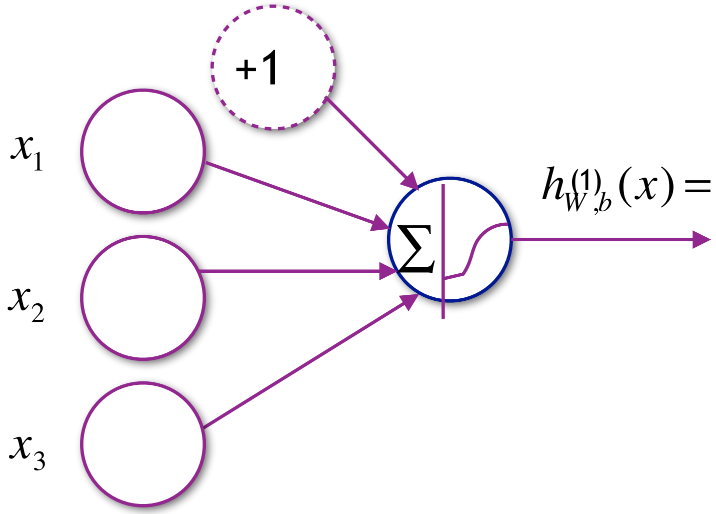
approximating single "neuron" using the sigmoid function

$$f(z) = \frac{1}{1+\exp(-z)}$$



New Notation

Unfortunately, there will be quite a bit of notation…..
We will follow the notation from http://deeplearning.stanford.edu/wiki/index.php/Neural_Networks



$$h^{(1)}_{W,b}(x) = f\left(\sum_{i=1}^{3} W^{(1)}_{1i} x_i + b\right)$$

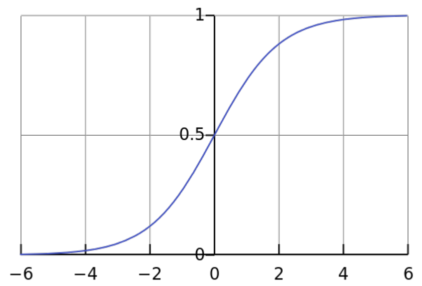
$$W^T = \begin{bmatrix} W^{(1)}_{11} \\ W^{(1)}_{12} \\ W^{(1)}_{13} \end{bmatrix} \mathbf{x} \begin{bmatrix} \\ x_2 \\ x_3 \end{bmatrix}$$

I added another subscript and superscript - we will need these later

$$z = [W^{(1)}_{11}, W^{(1)}_{12}, W^{(1)}_{13}] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + b$$

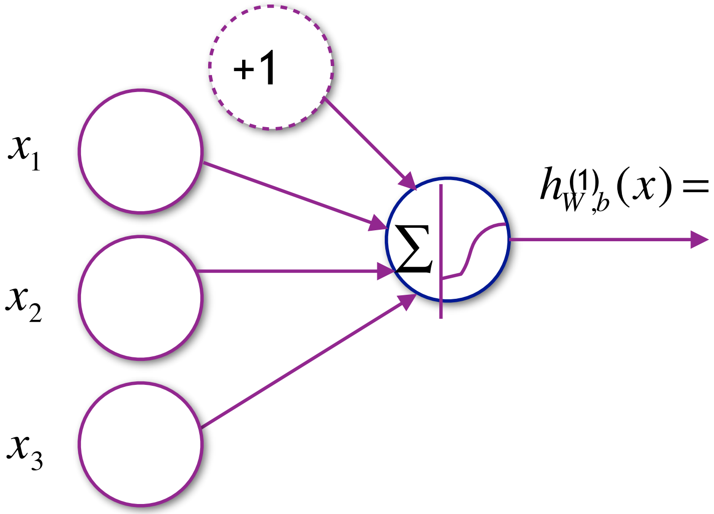
approximating single “neuron” using the sigmoid function

$$f(z) = \frac{1}{1+\exp(-z)}$$



New Notation

Unfortunately, there will be quite a bit of notation…..
We will follow the notation from http://deeplearning.stanford.edu/wiki/index.php/Neural_Networks



$$h^{(1)}_{W,b}(x) = f\left(\sum_{i=1}^{3} W^{(1)}_{1i} x_i + b\right)$$

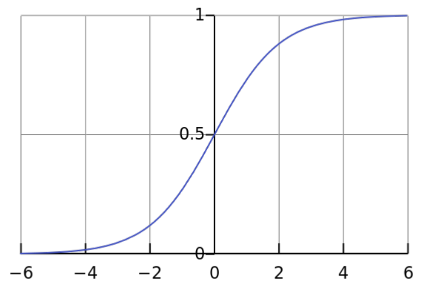
$$W^T = \begin{bmatrix} W^{(1)}_{11} \\ W^{(1)}_{12} \\ W^{(1)}_{13} \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Note we are not adding b the intercept term to the feature vector

$$z = [W^{(1)}_{11}, W^{(1)}_{12}, W^{(1)}_{13}] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + b$$

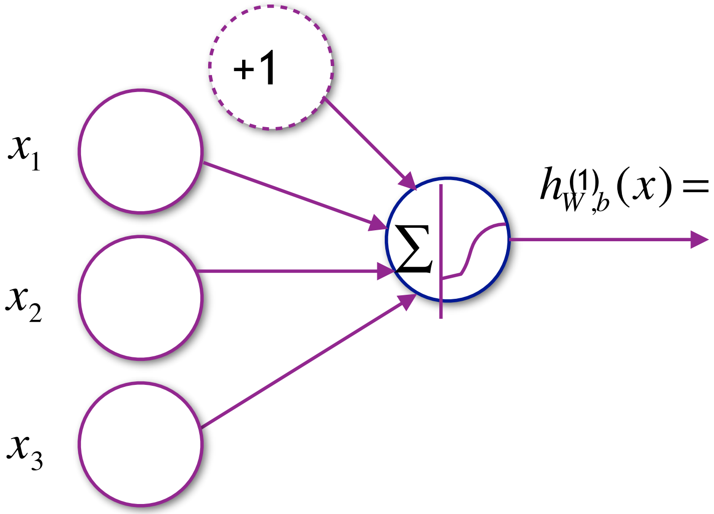
approximating single "neuron" using the sigmoid function

$$f(z) = \frac{1}{1 + \exp(-z)}$$



New Notation

Unfortunately, there will be quite a bit of notation…..
We will follow the notation from http://deeplearning.stanford.edu/wiki/index.php/Neural_Networks



$$h_{W,b}^{(1)}(x) = f\left(\sum_{i=1}^{3} W_{1i}^{(1)} x_i + b\right)$$

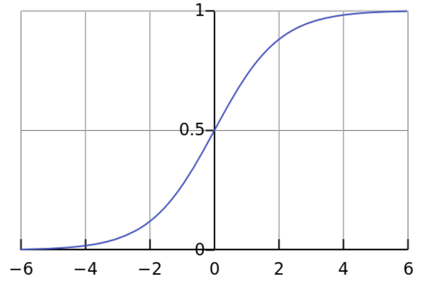
Instead of w_0 use b for the intercept term

Note we are not adding b the intercept term to the feature vector

$$z = [W_{11}^{(1)}, W_{12}^{(1)}, W_{13}^{(1)}] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + b$$

approximating single "neuron" using the sigmoid function

$$f(z) = \frac{1}{1+\exp(-z)}$$

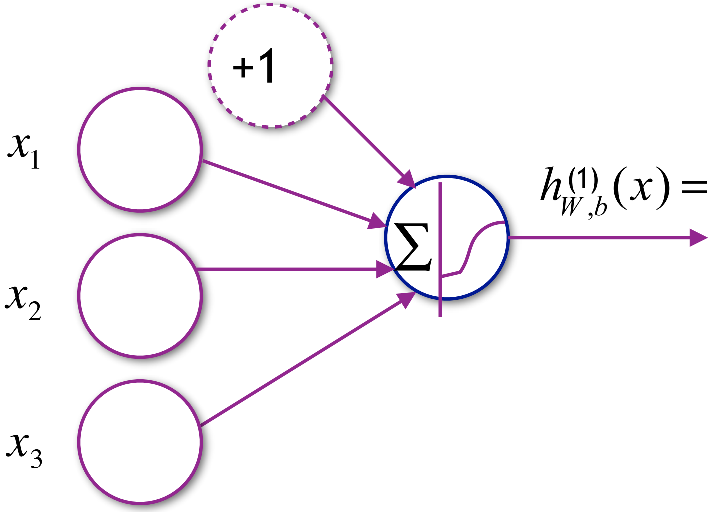


Outline

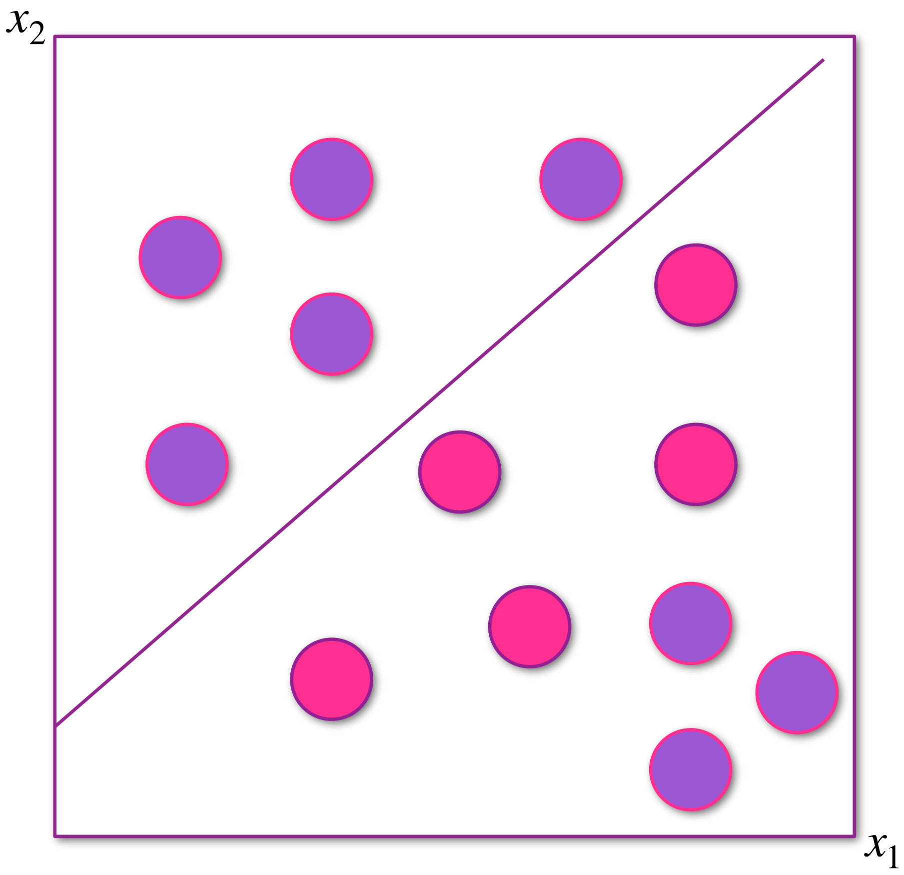
- ❑ Introduction to neurons
- ❑ Nonlinear classifiers from linear features
- ❑ Neural networks notation
- ❑ Pseudocode for prediction
- ❑ Training a neural network
- ❑ Implementing gradient descent for neural networks
  - Vectorization
  - Pseudocode
- ❑ Preprocessing
- ❑ Initialization
- ❑ Activations

A more complicated decision boundary?

How can we get a more complicated decision boundary?



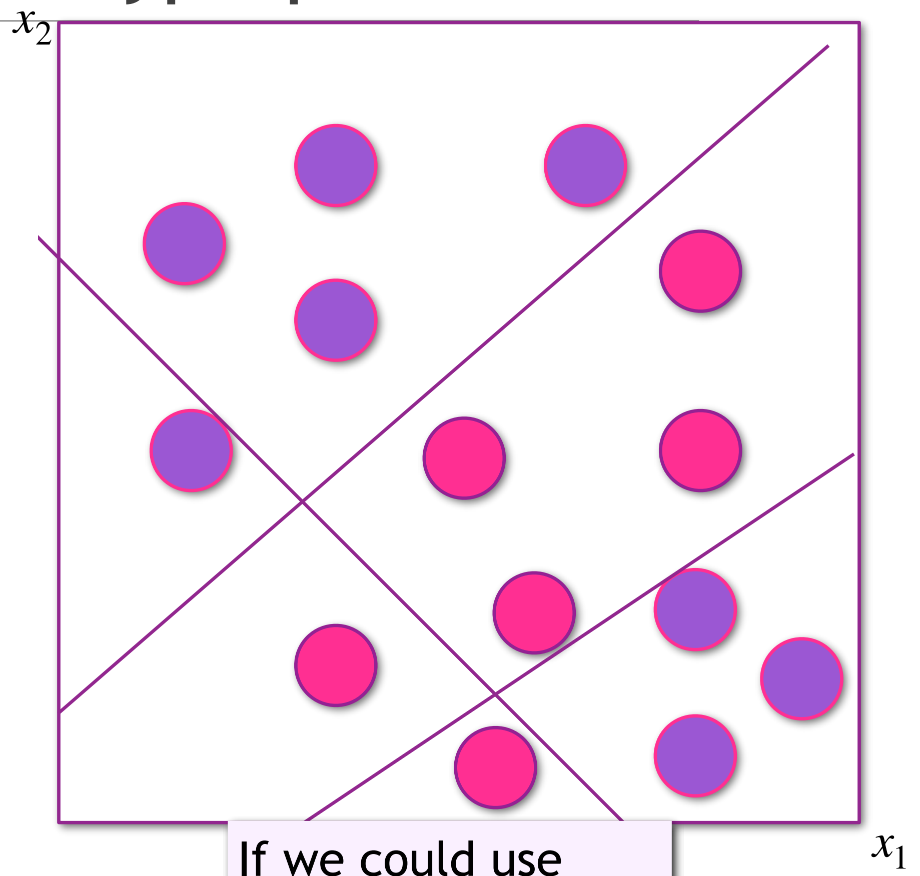
$$h^{(1)}_{W,b}(x) = f\left(\sum_{i=1}^{3} W^{(1)}_{1i} x_i + b\right)$$



How could we learn a more complicated decision boundary?

Pair share: How can we do this?

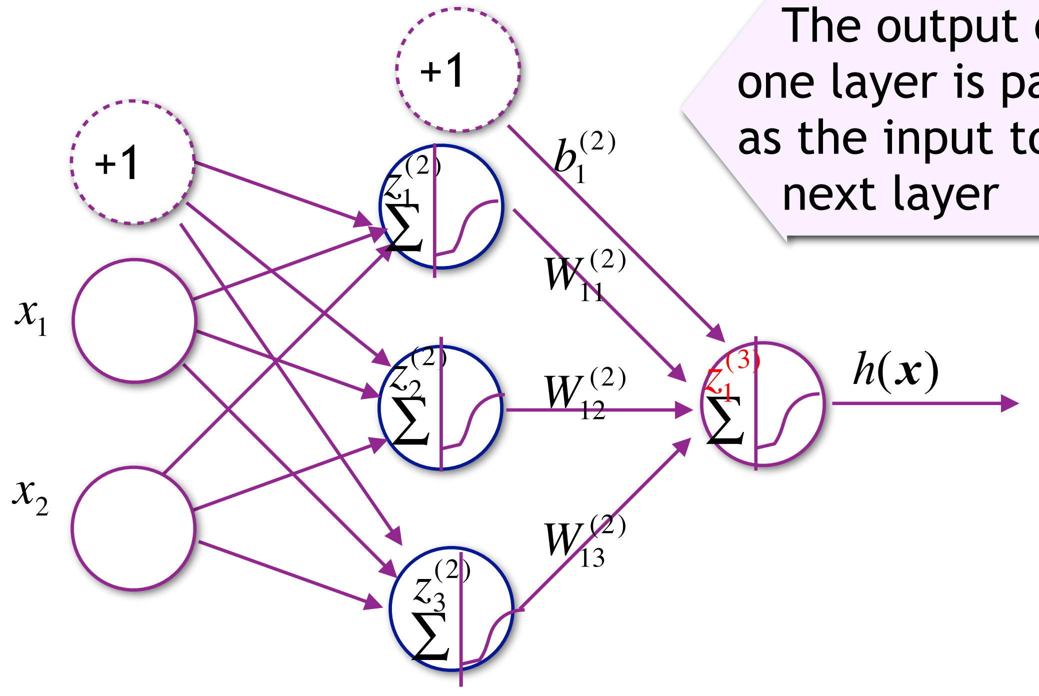
It would be easy if our features were the 3 hyperplanes…



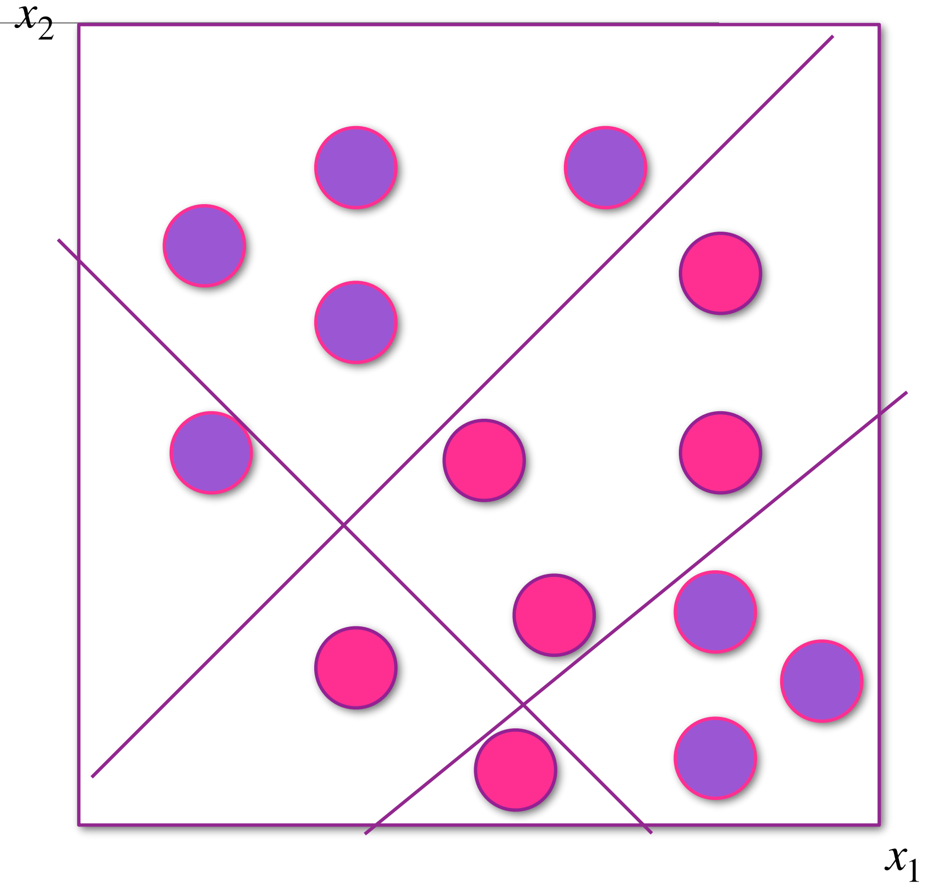
If we could use these as features we can create a more complex decision boundary.

Feature Construction!

Creating a more useful set of features that allow for a linear decision boundary



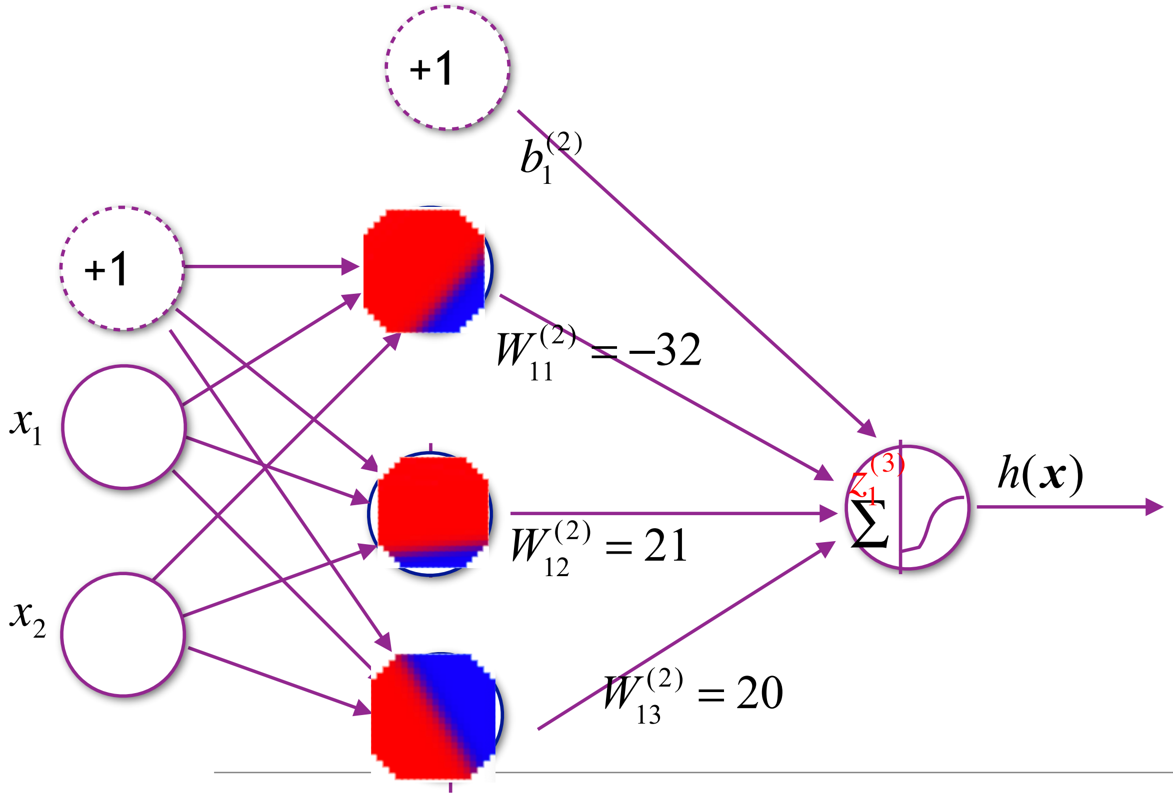
The output of one layer is passed as the input to the next layer

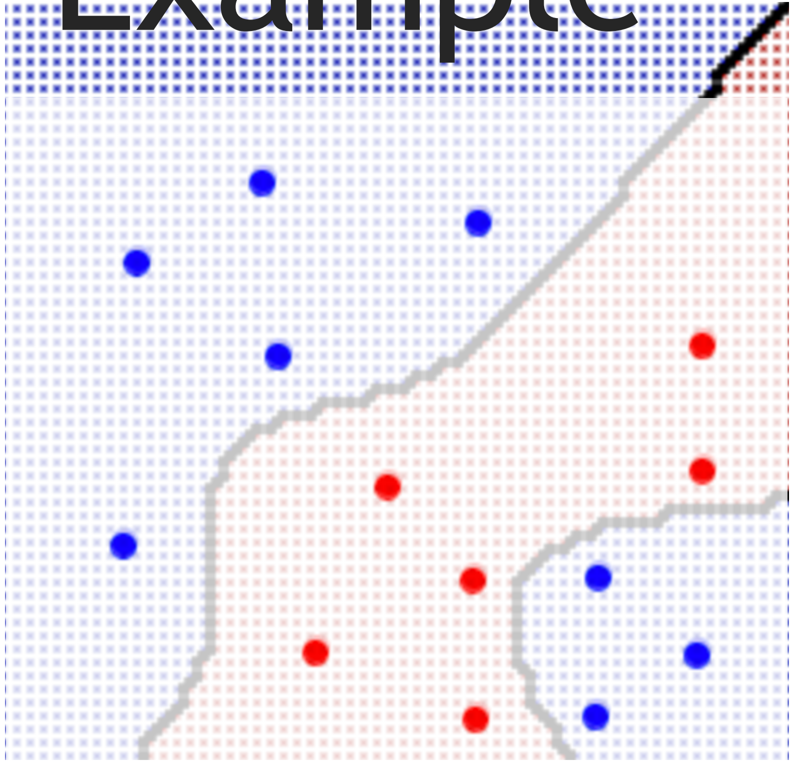


$$z_1^{(3)} = \sum_{j=1}^{3} W_{1j}^{(2)} f(z_j^{(2)}) + b_1^{(2)}$$

$$h(\boldsymbol{x}) = f(z_1^{(3)})$$

Example



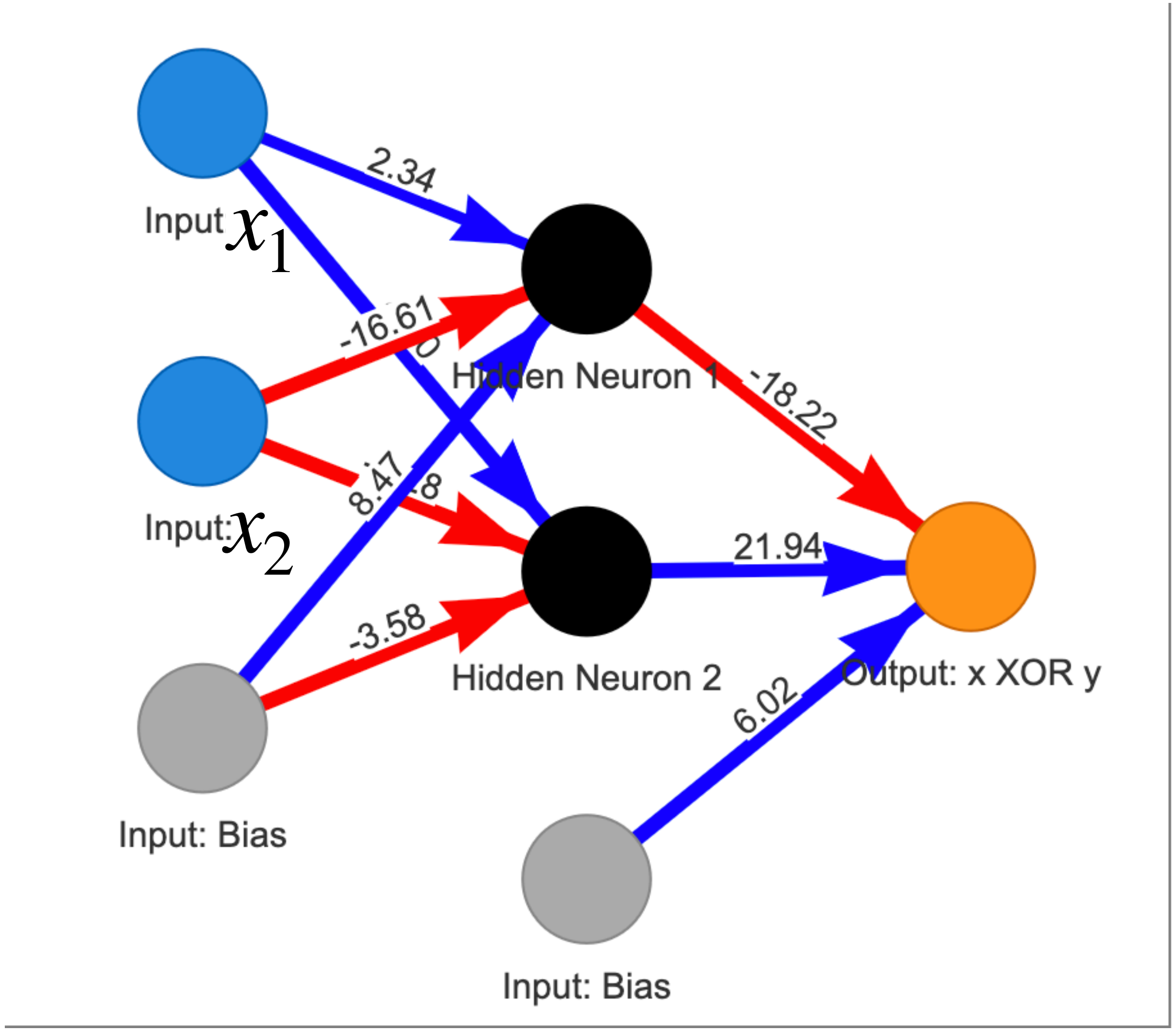


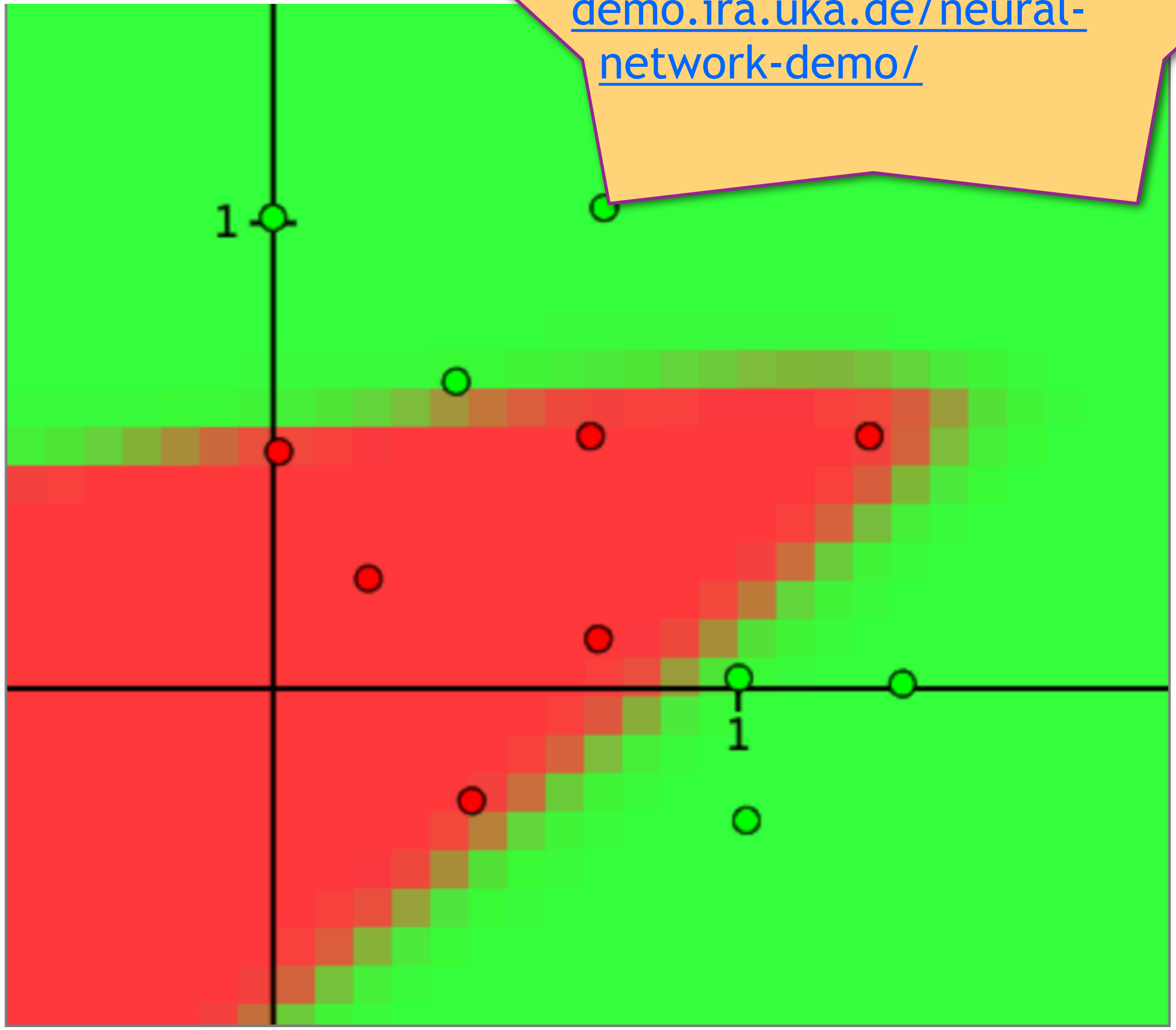
Example generated from

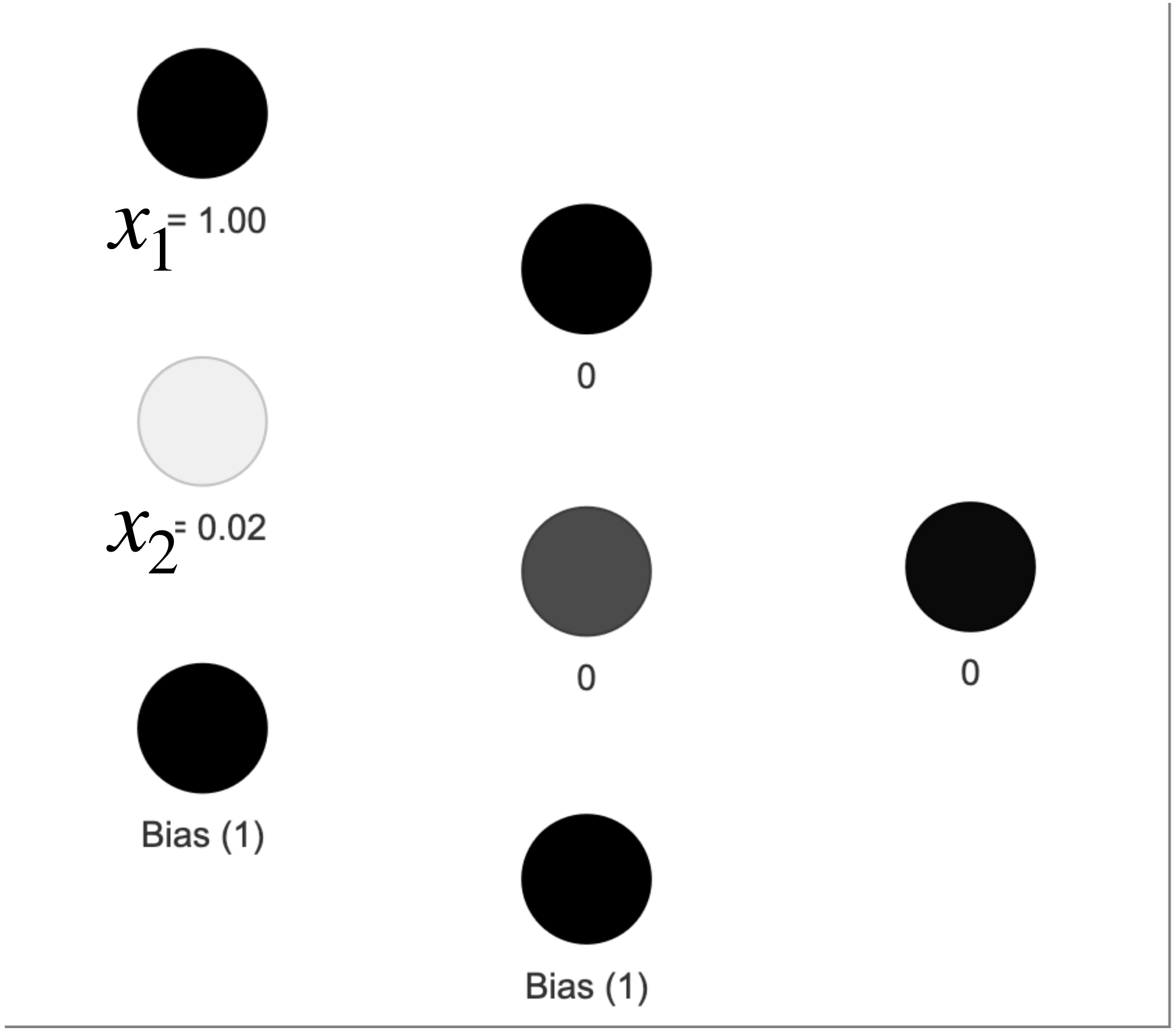
http://www.ccom.ucsd.edu/~cdeotte/programs/neuralnetwork.html

Prediction using a neural network

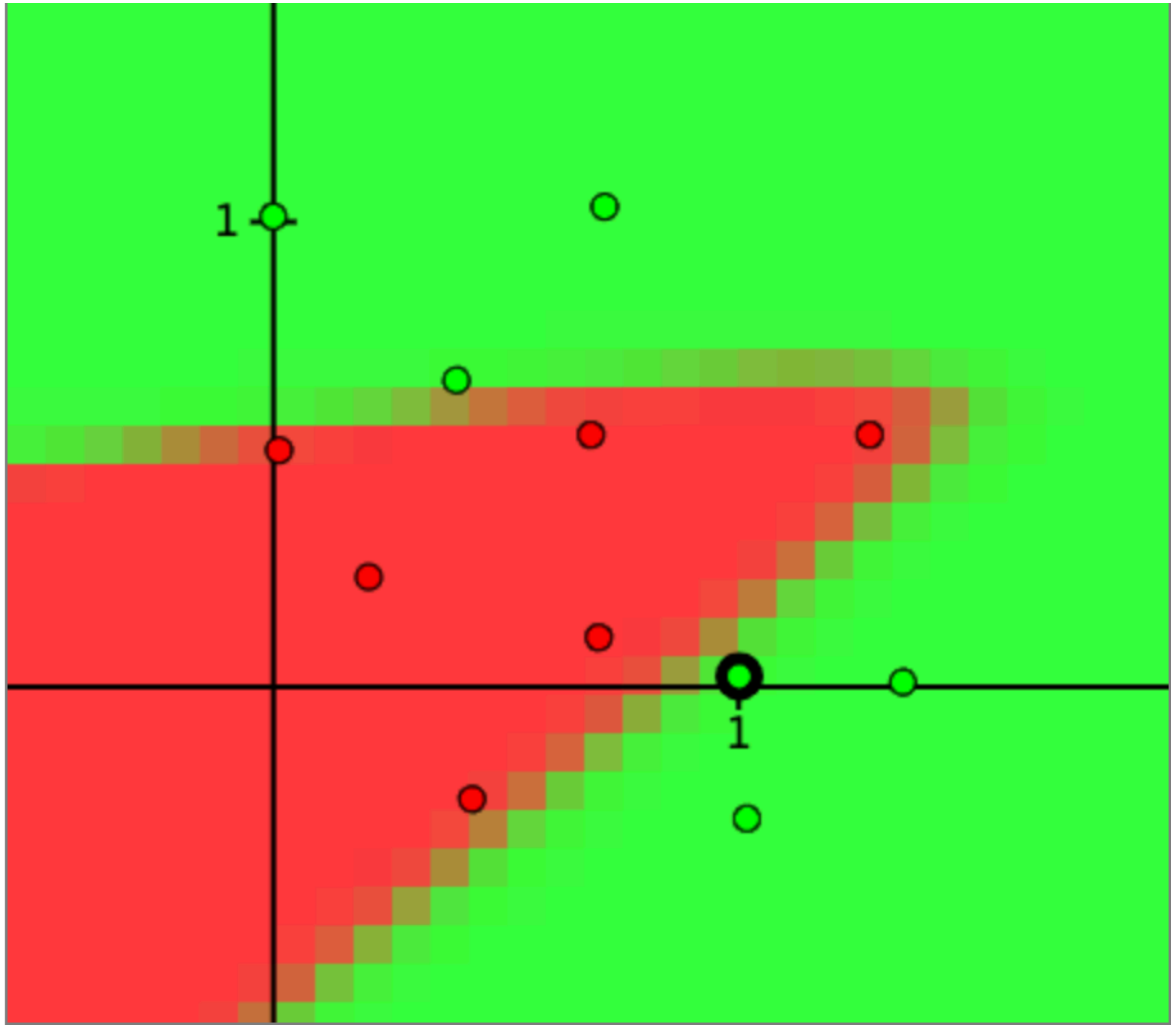
Demo! https://lecture-demo.ira.uka.de/neural-network-demo/



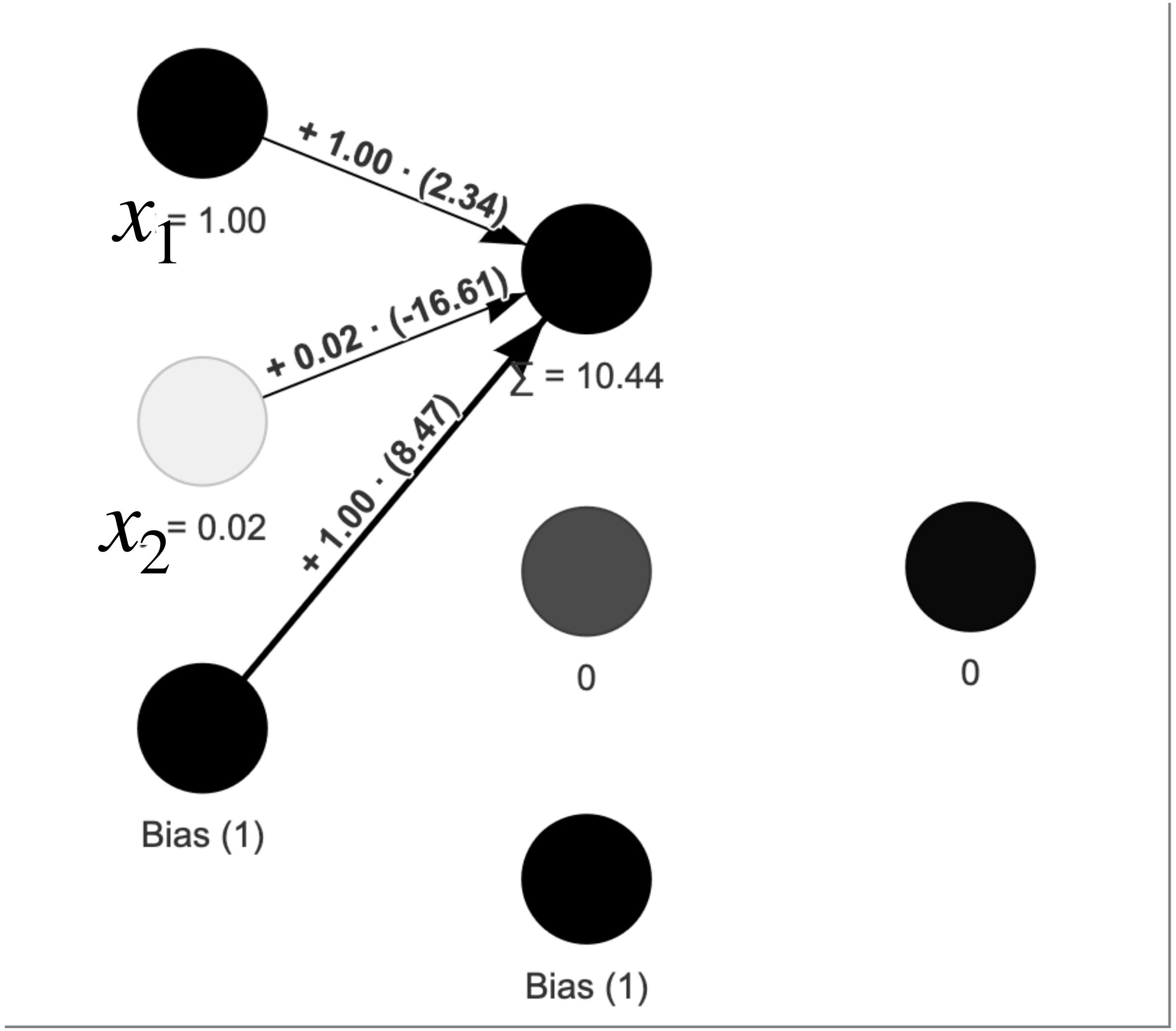


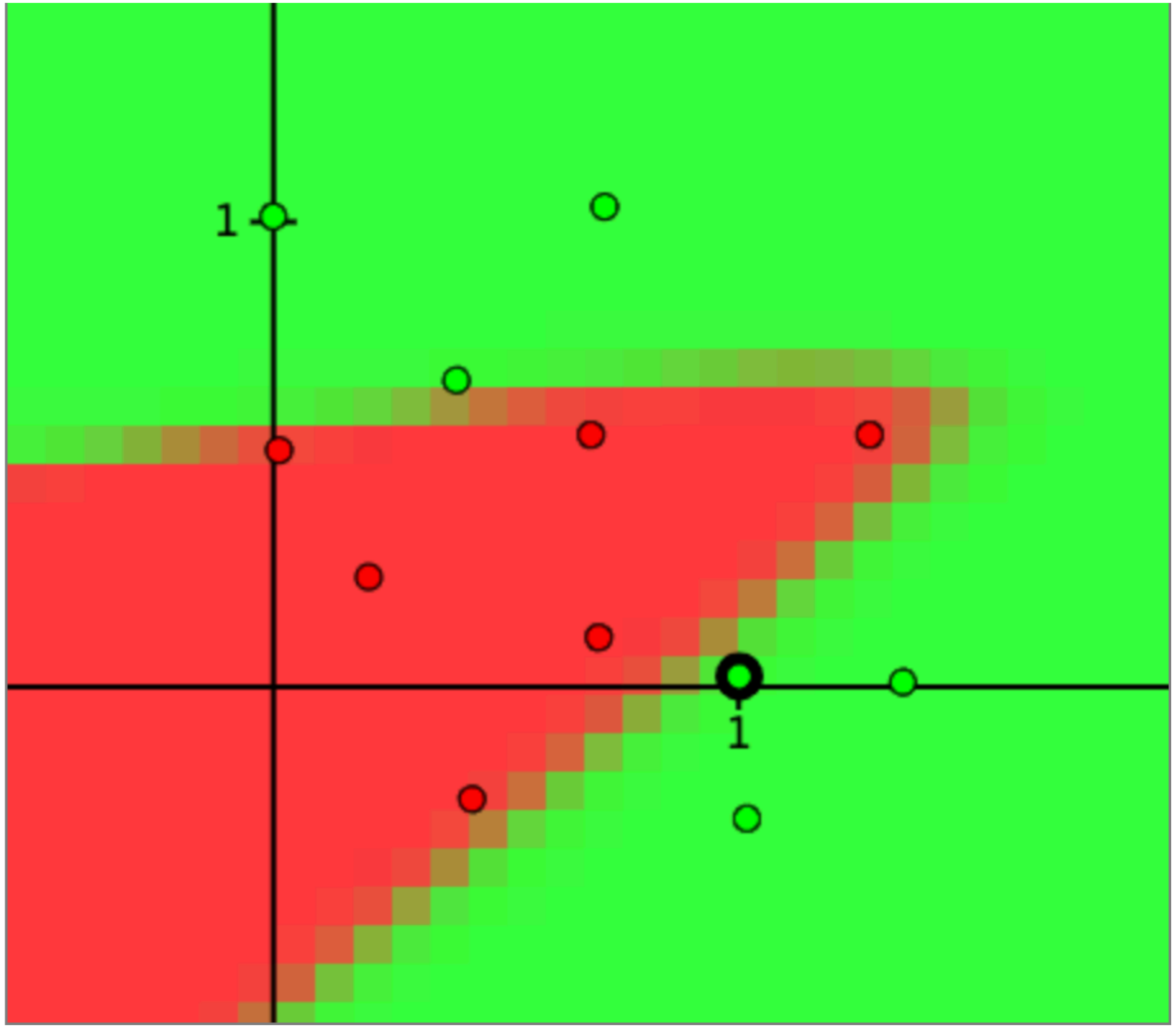


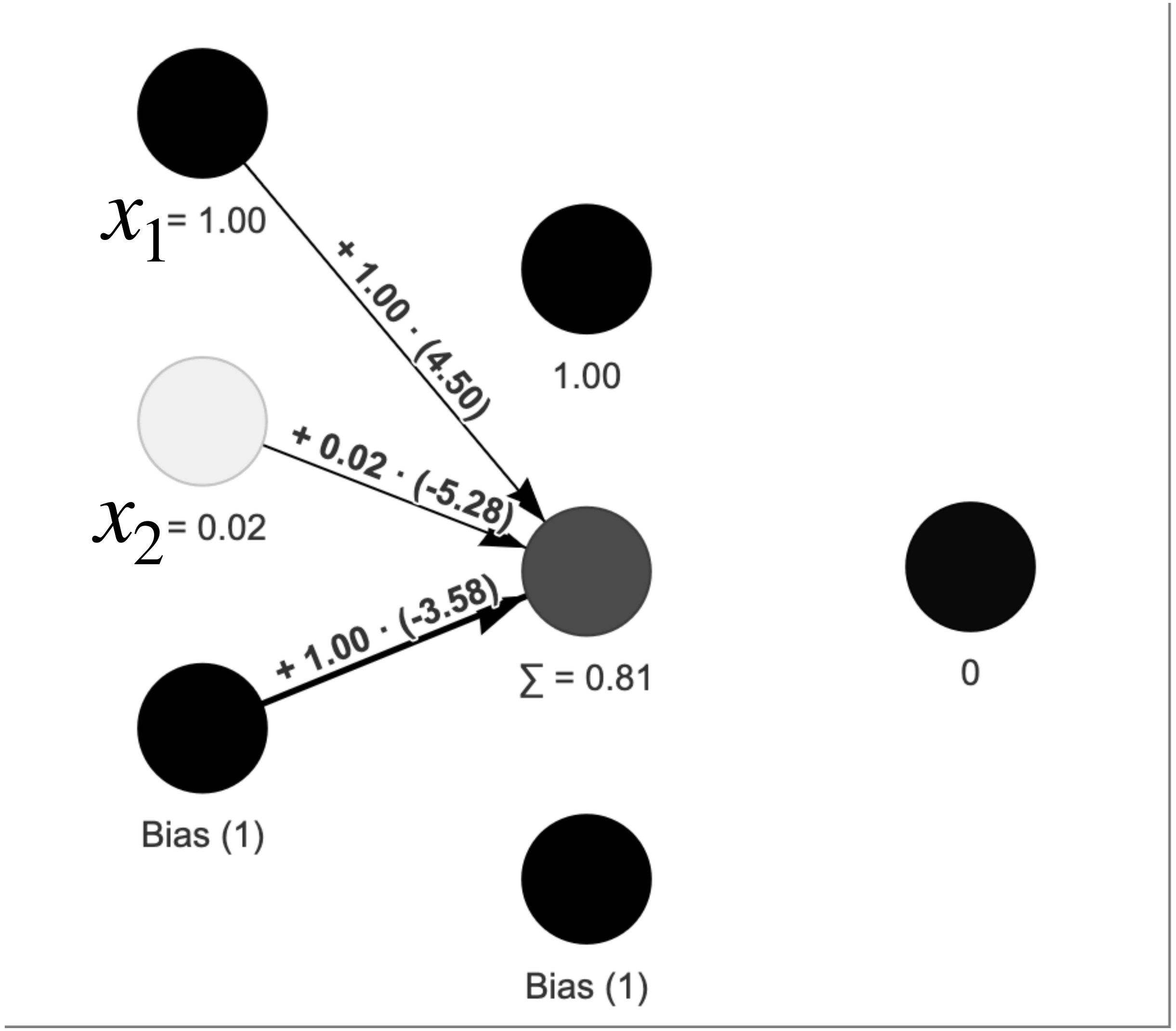
x_1 = 1.00
x_2 = 0.02
Bias (1)
0
0
Bias (1)
0

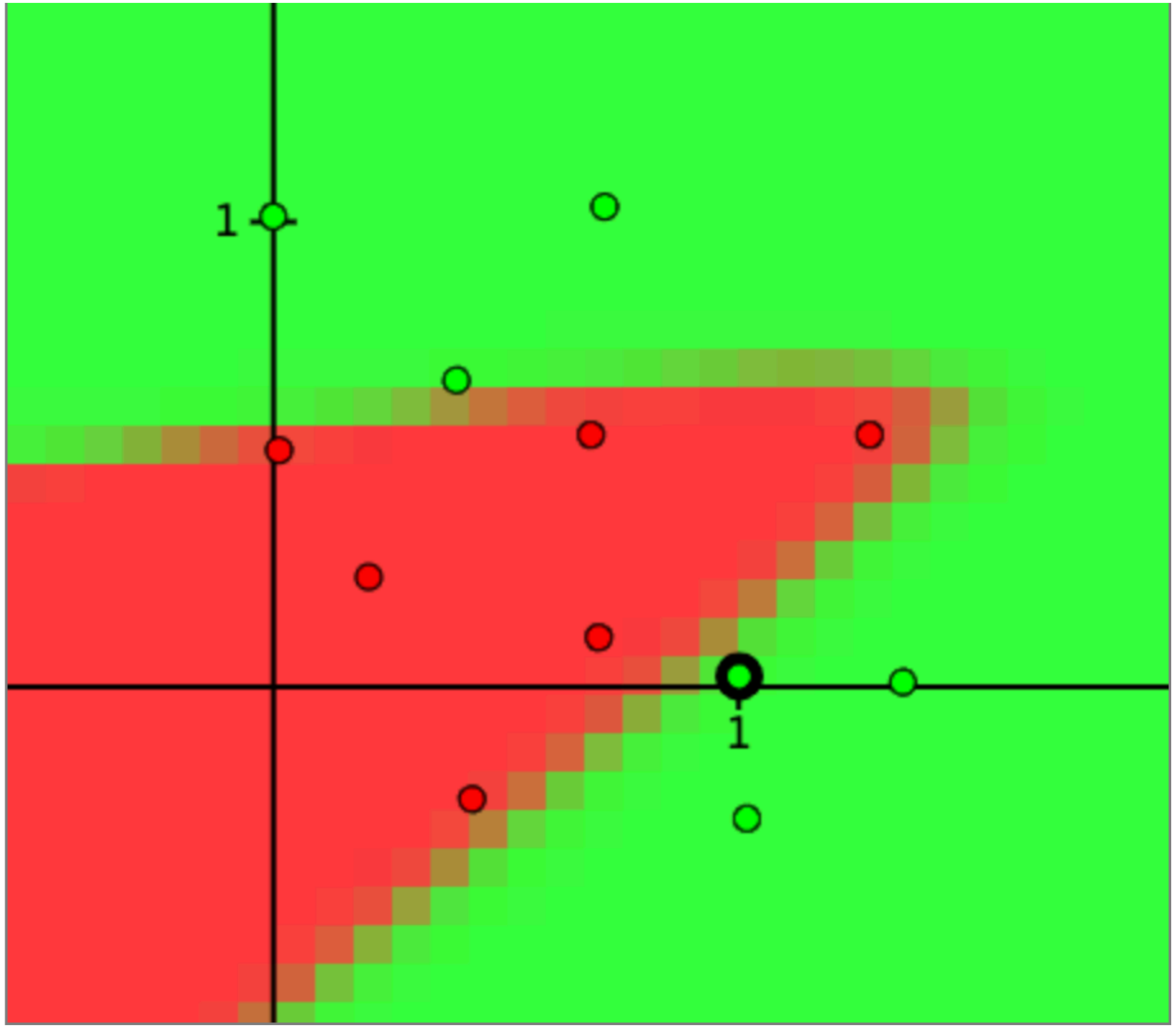


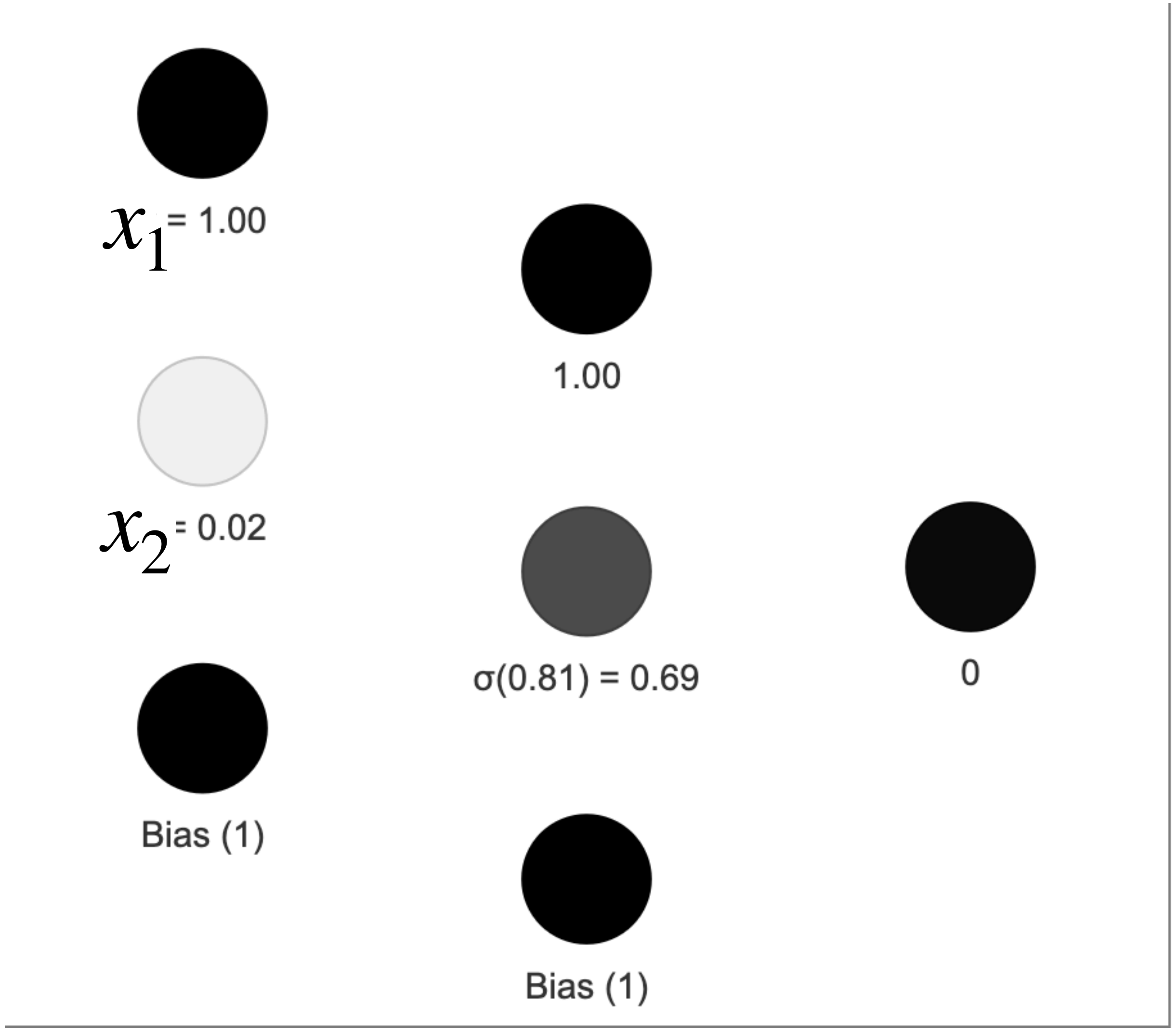
1
1

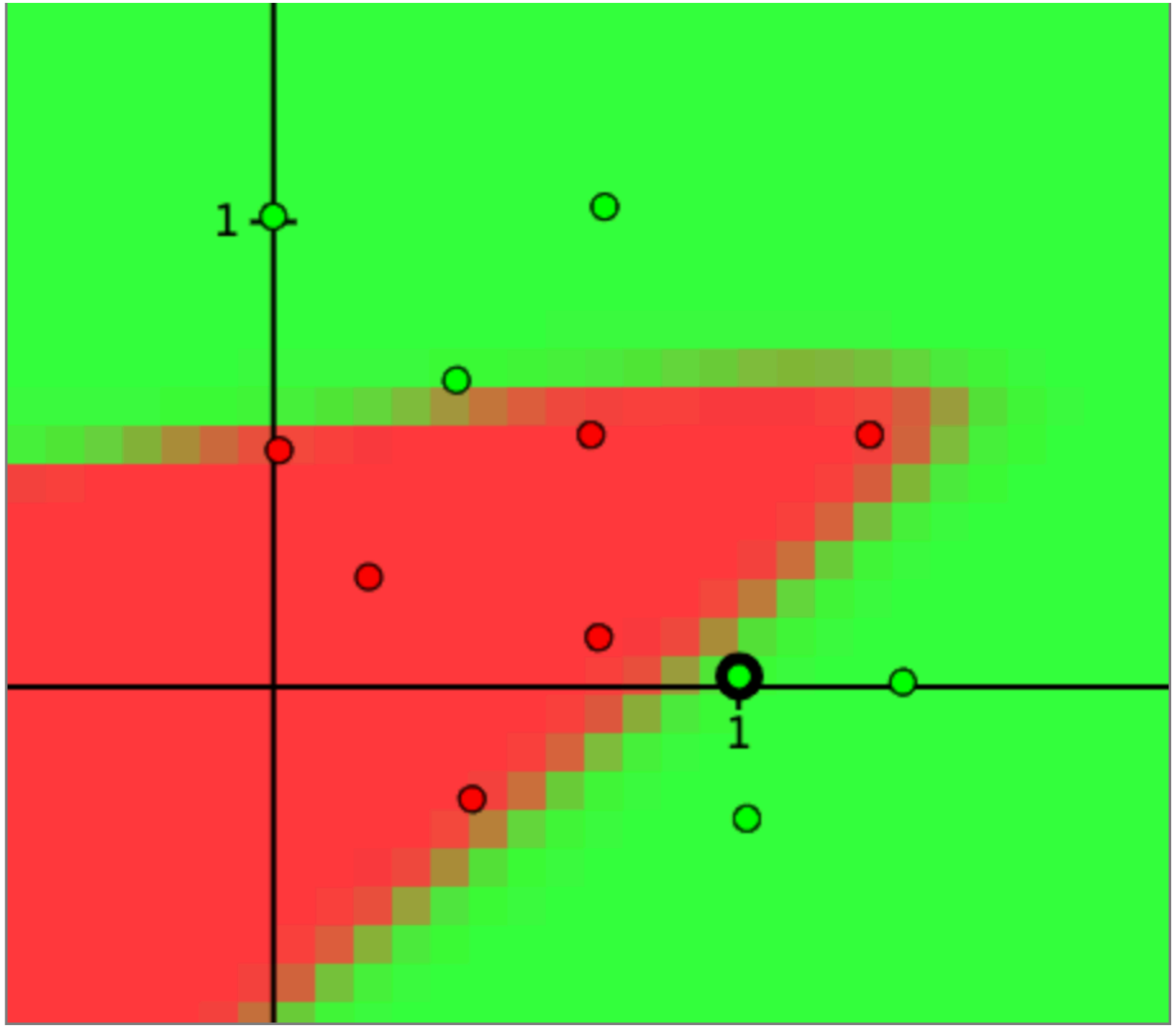

+ 1.00 · (2.34)
x_1 = 1.00
+ 0.02 · (-16.61)
Σ = 10.44
x_2 = 0.02
+ 1.00 · (8.47)
0
0
Bias (1)
Bias (1)

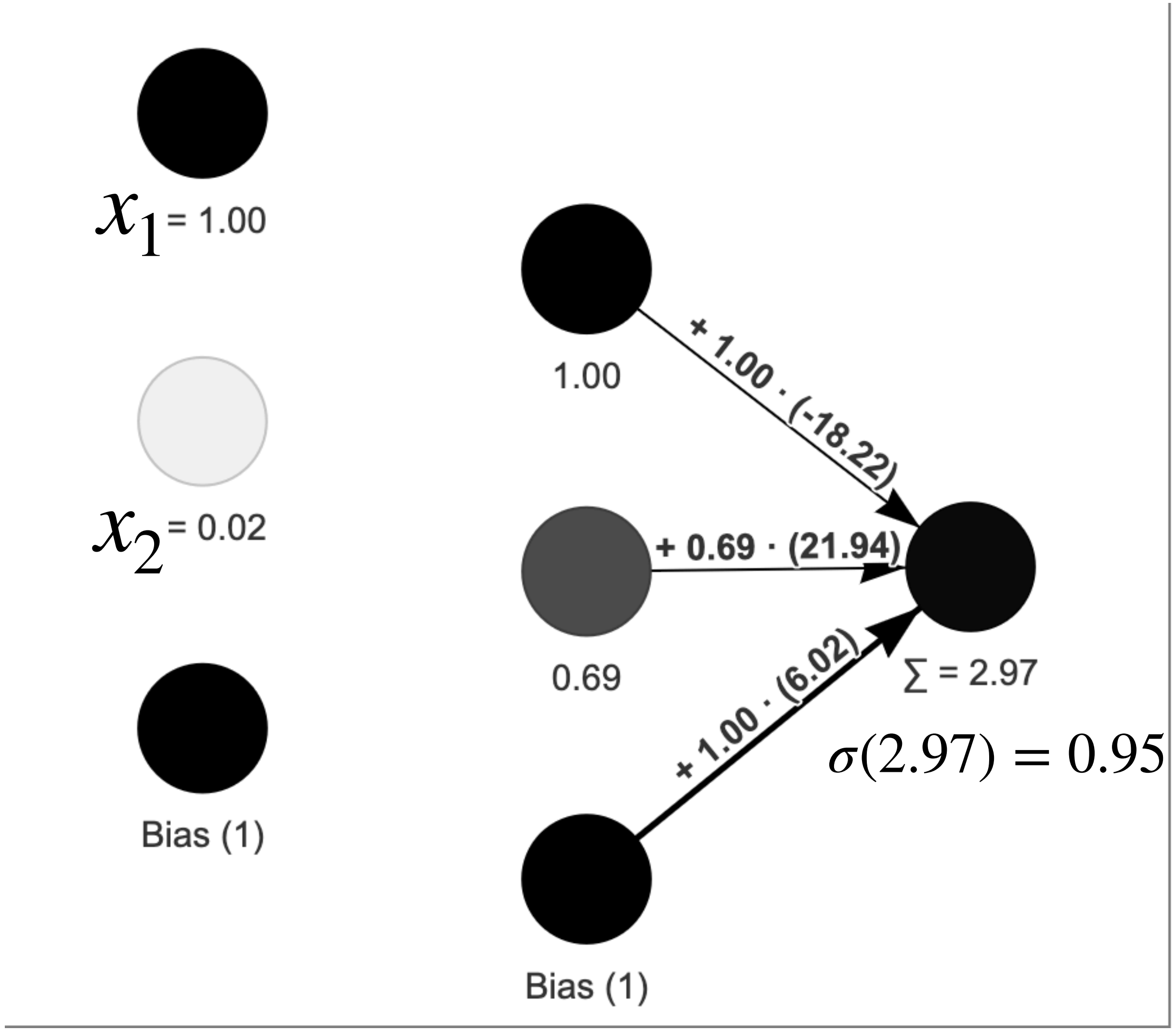

1
1


x_1 = 1.00
+ 1.00 · (4.50)
1.00
+ 0.02 · (-5.28)
x_2 = 0.02
+ 1.00 · (-3.58)
∑ = 0.81
0
Bias (1)
Bias (1)

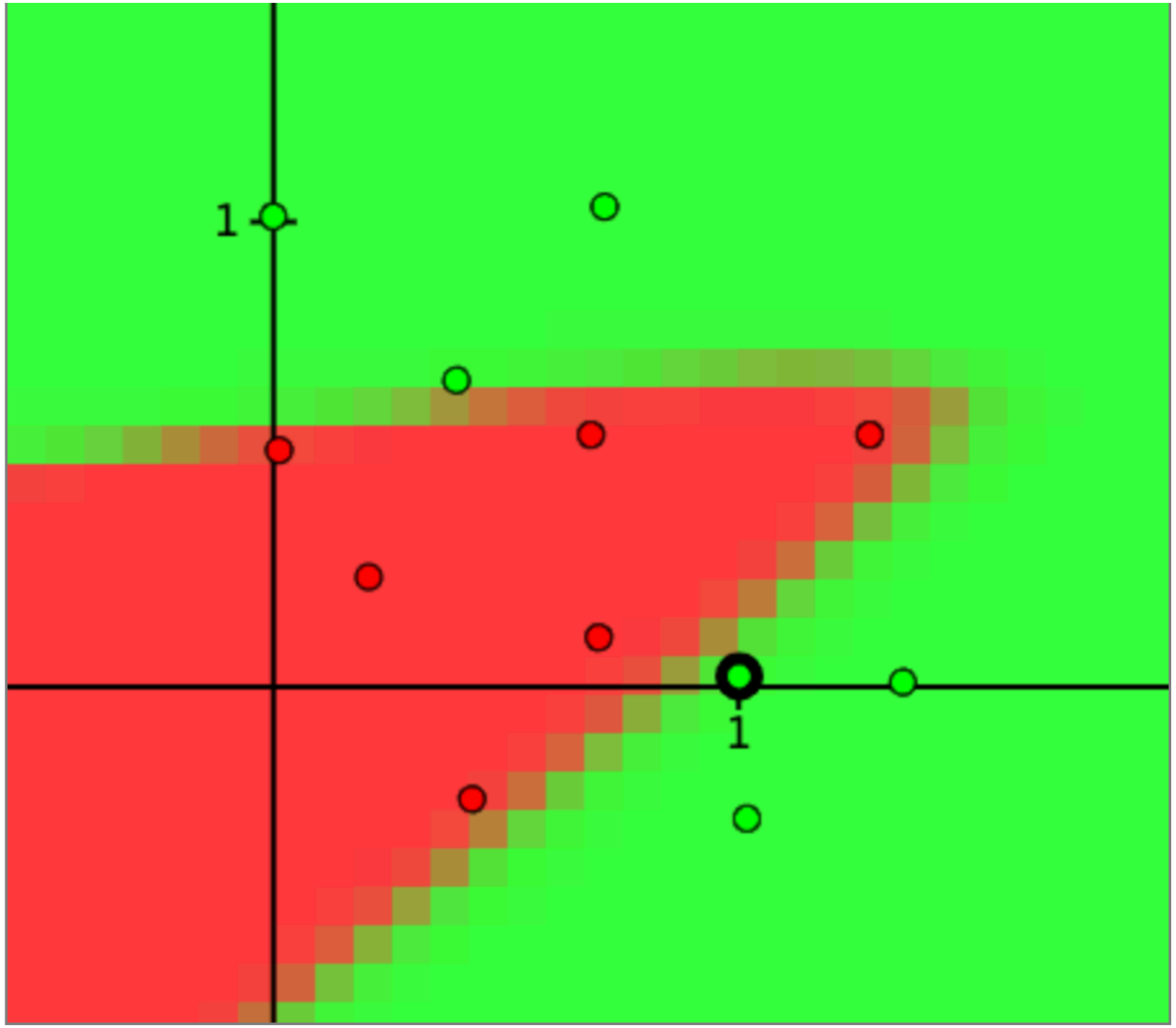

1
1


x_1 = 1.00
1.00
x_2 = 0.02
σ(0.81) = 0.69
0
Bias (1)
Bias (1)

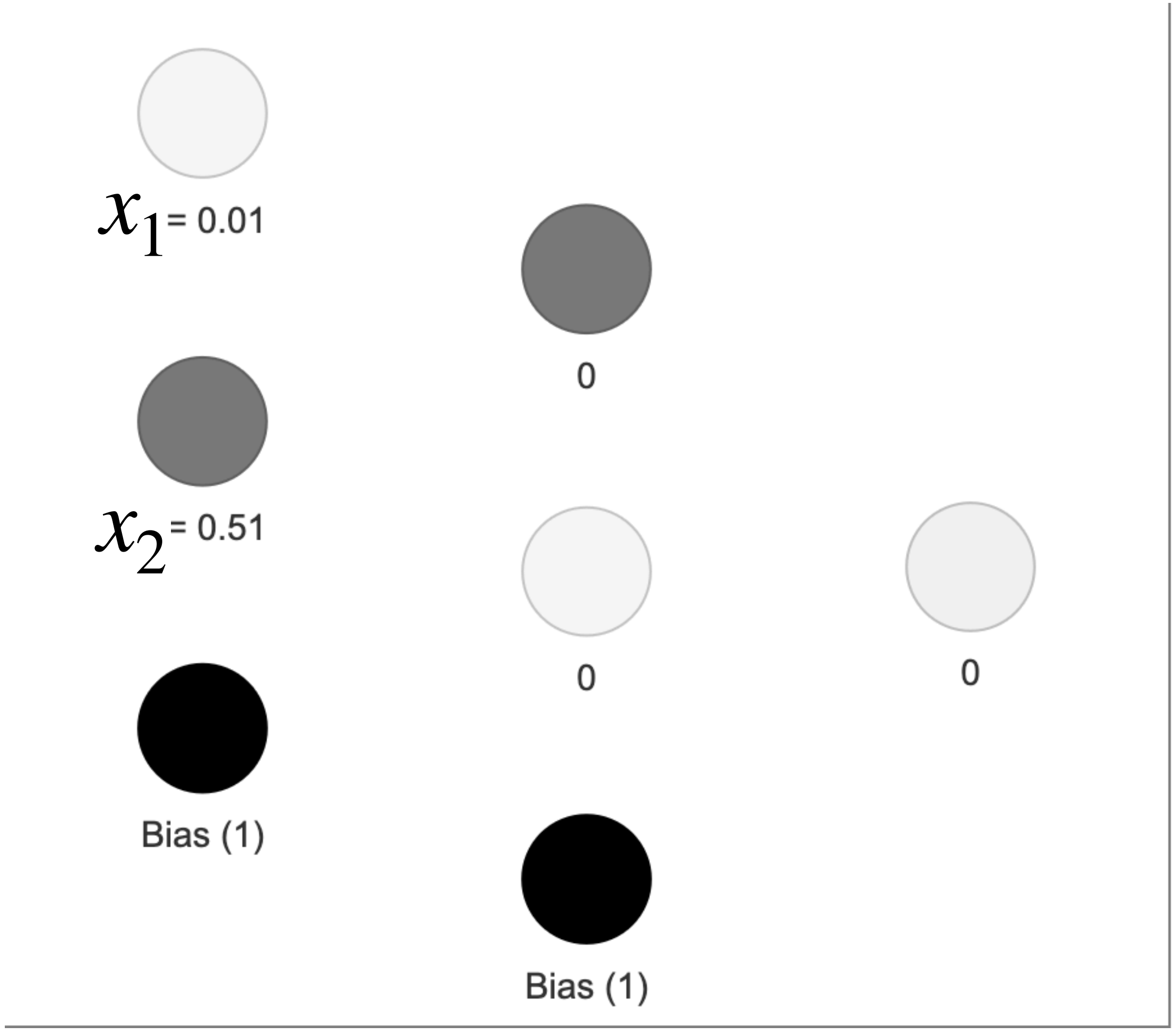

1
1

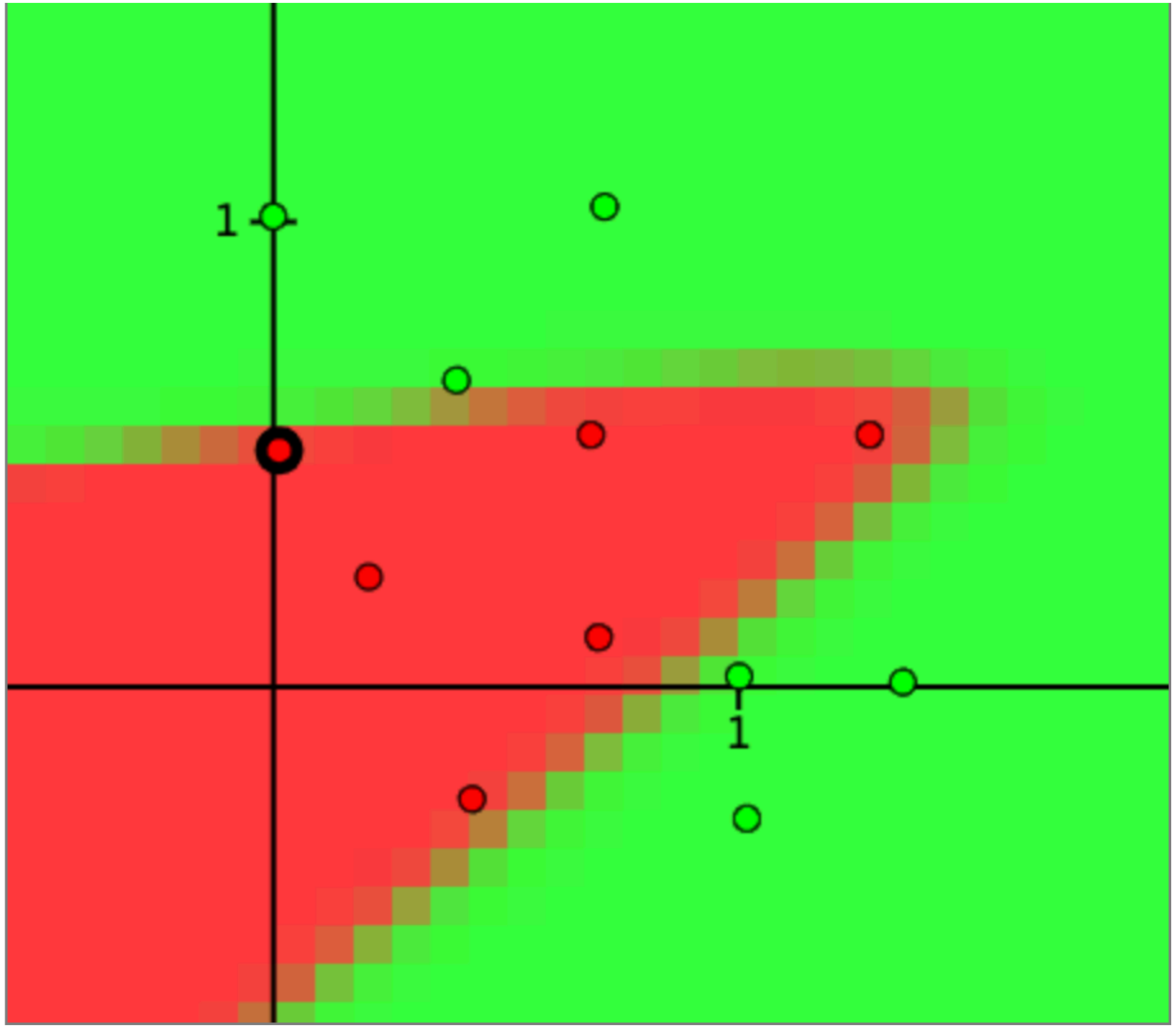


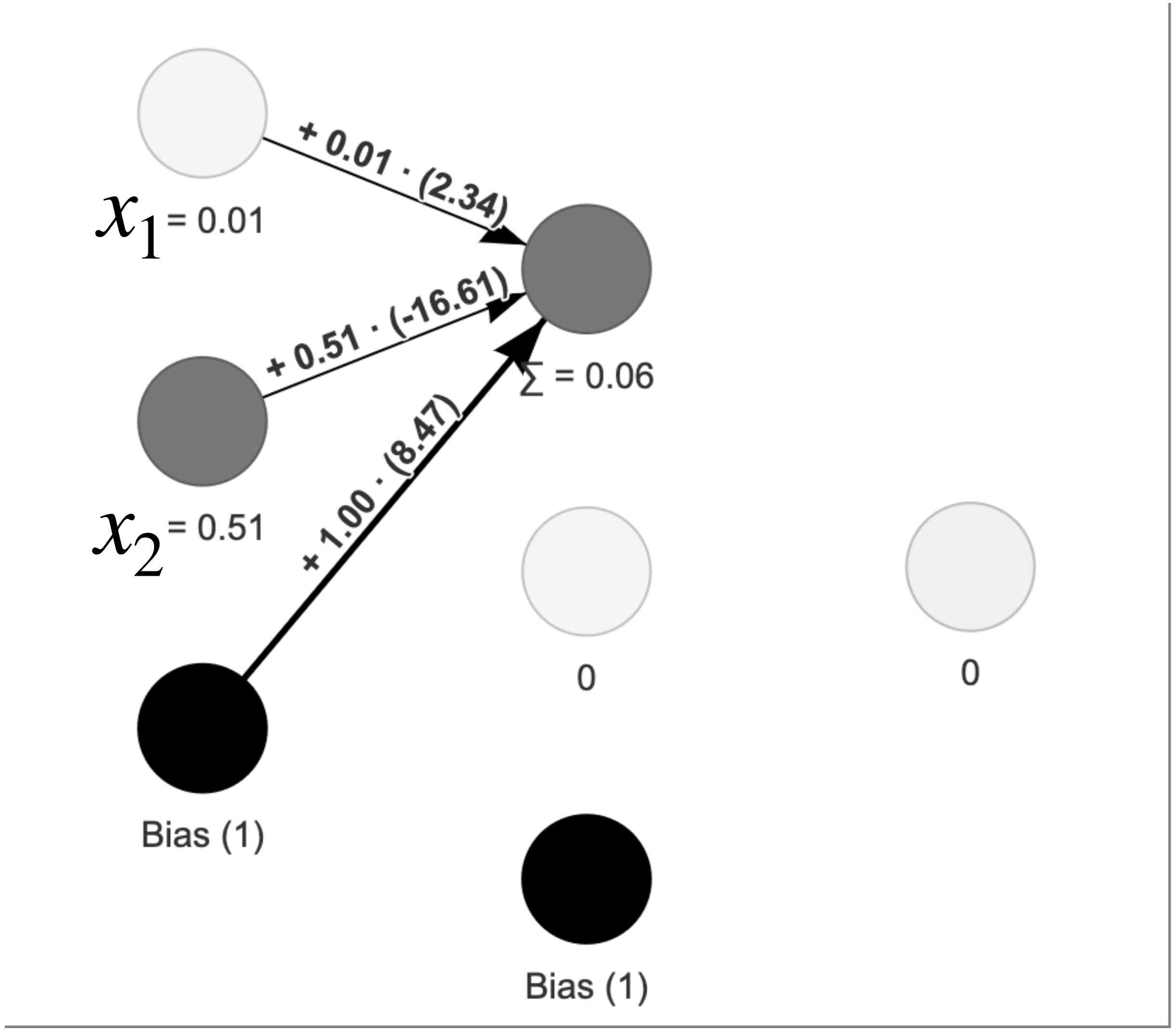
x_1 = 1.00
x_2 = 0.02
Bias (1)
1.00
0.69
Bias (1)
+ 1.00 · (-18.22)
+ 0.69 · (21.94)
+ 1.00 · (6.02)
∑ = 2.97
$\sigma(2.97) = 0.95$

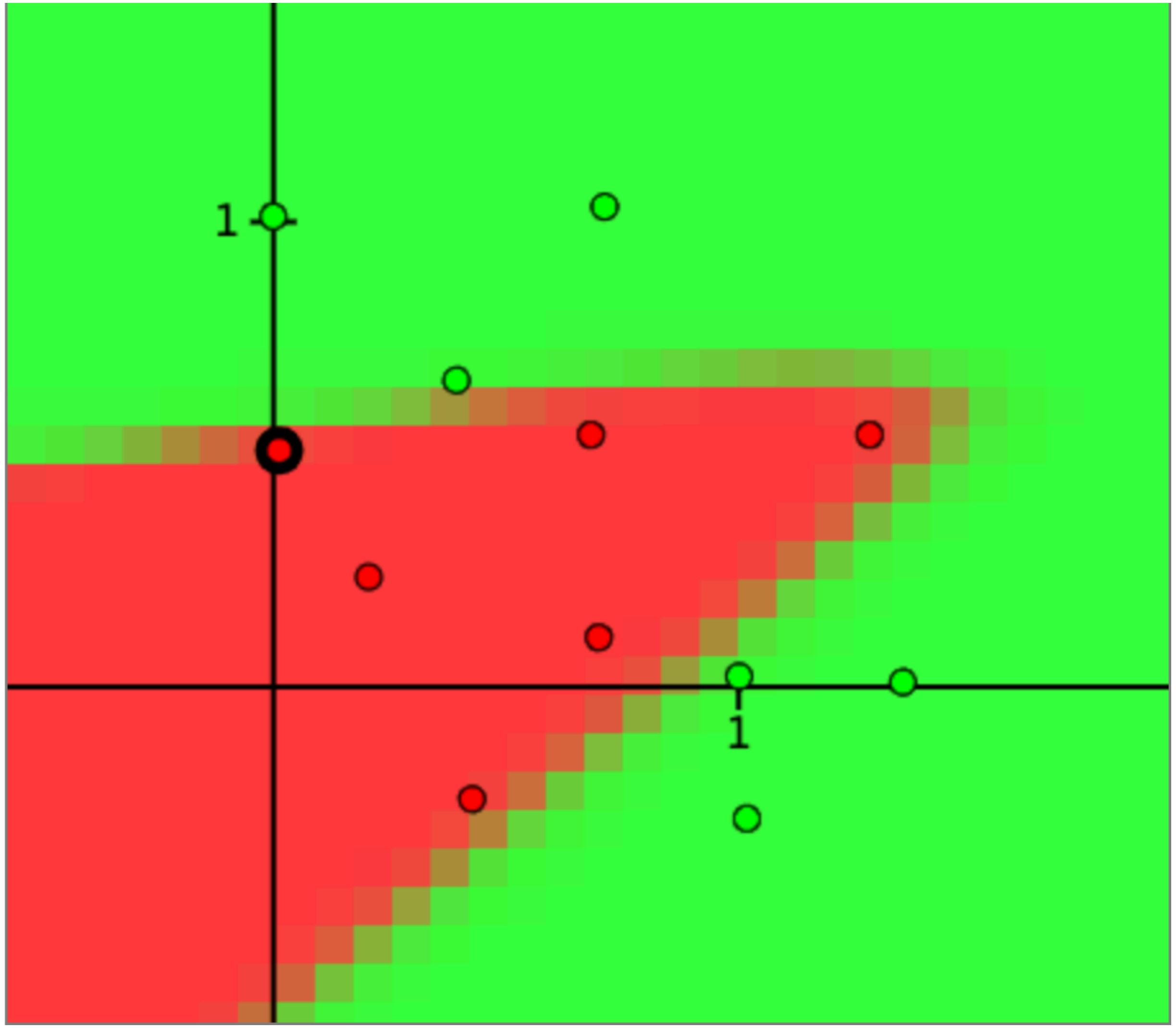


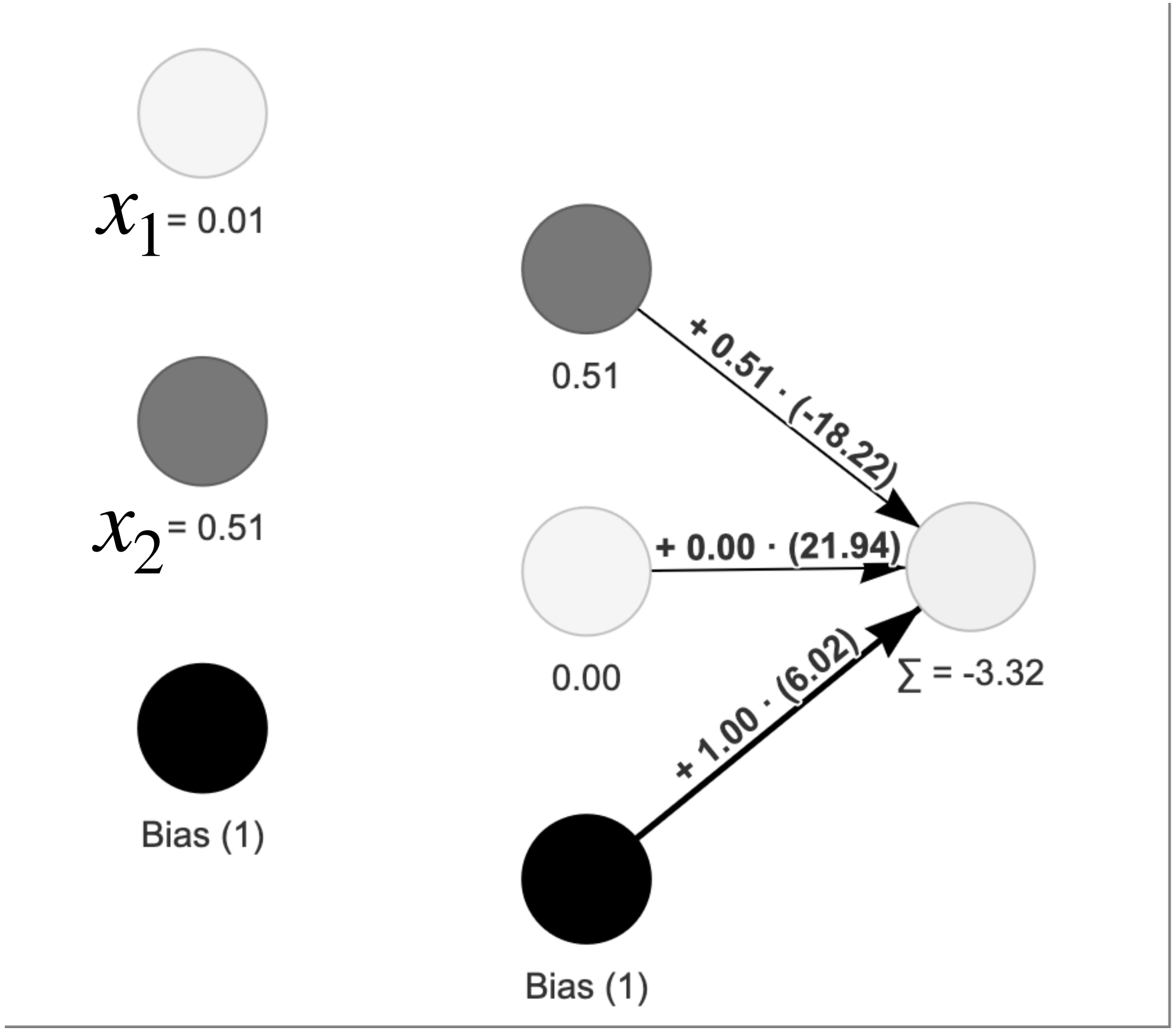
1
1


x_1 = 0.01
x_2 = 0.51
Bias (1)
0
0
Bias (1)
0

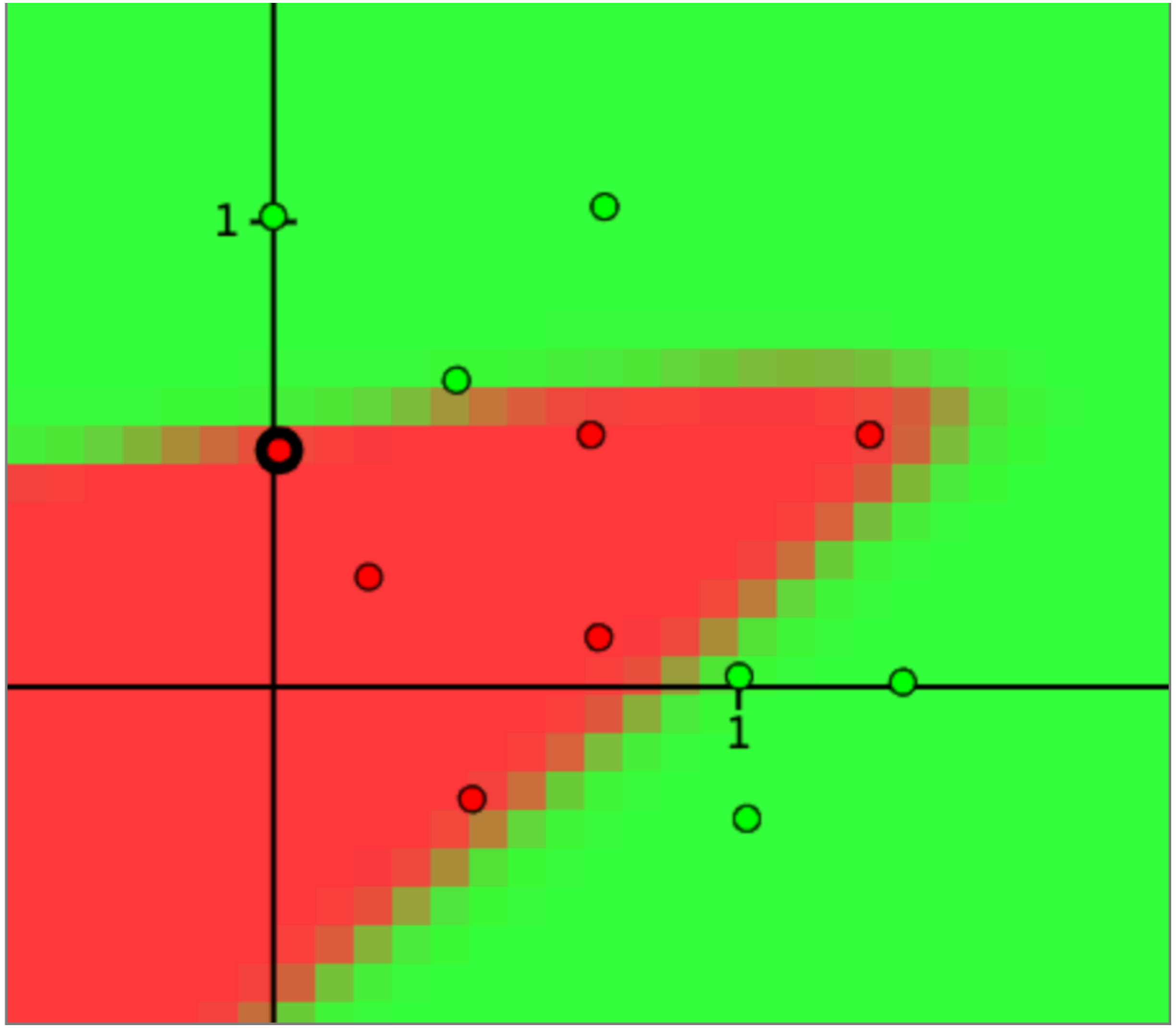

1
1


+ 0.01 · (2.34)
+ 0.51 · (-16.61)
+ 1.00 · (8.47)
x_1 = 0.01
x_2 = 0.51
∑ = 0.06
0
0
Bias (1)
Bias (1)

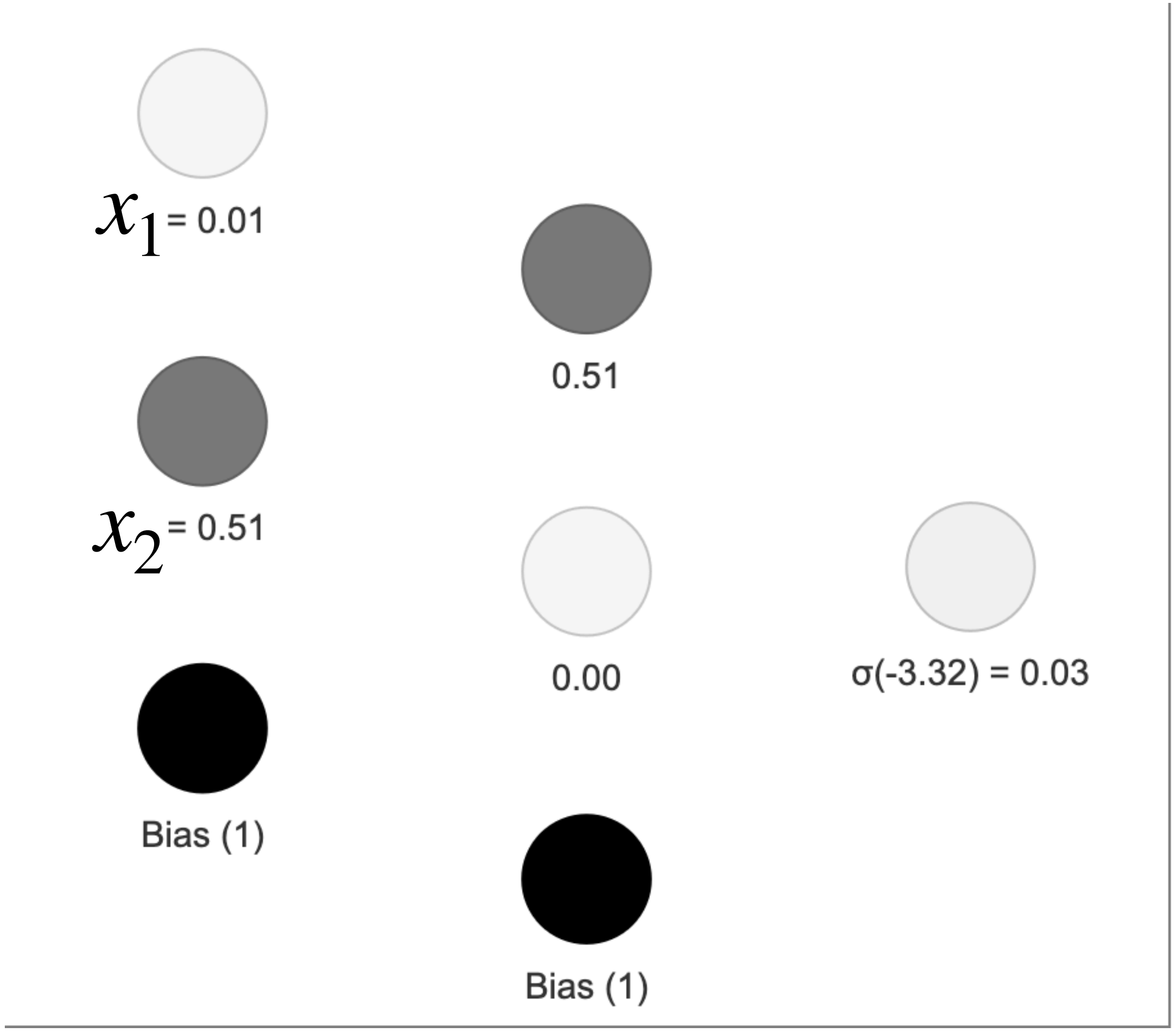

1
1

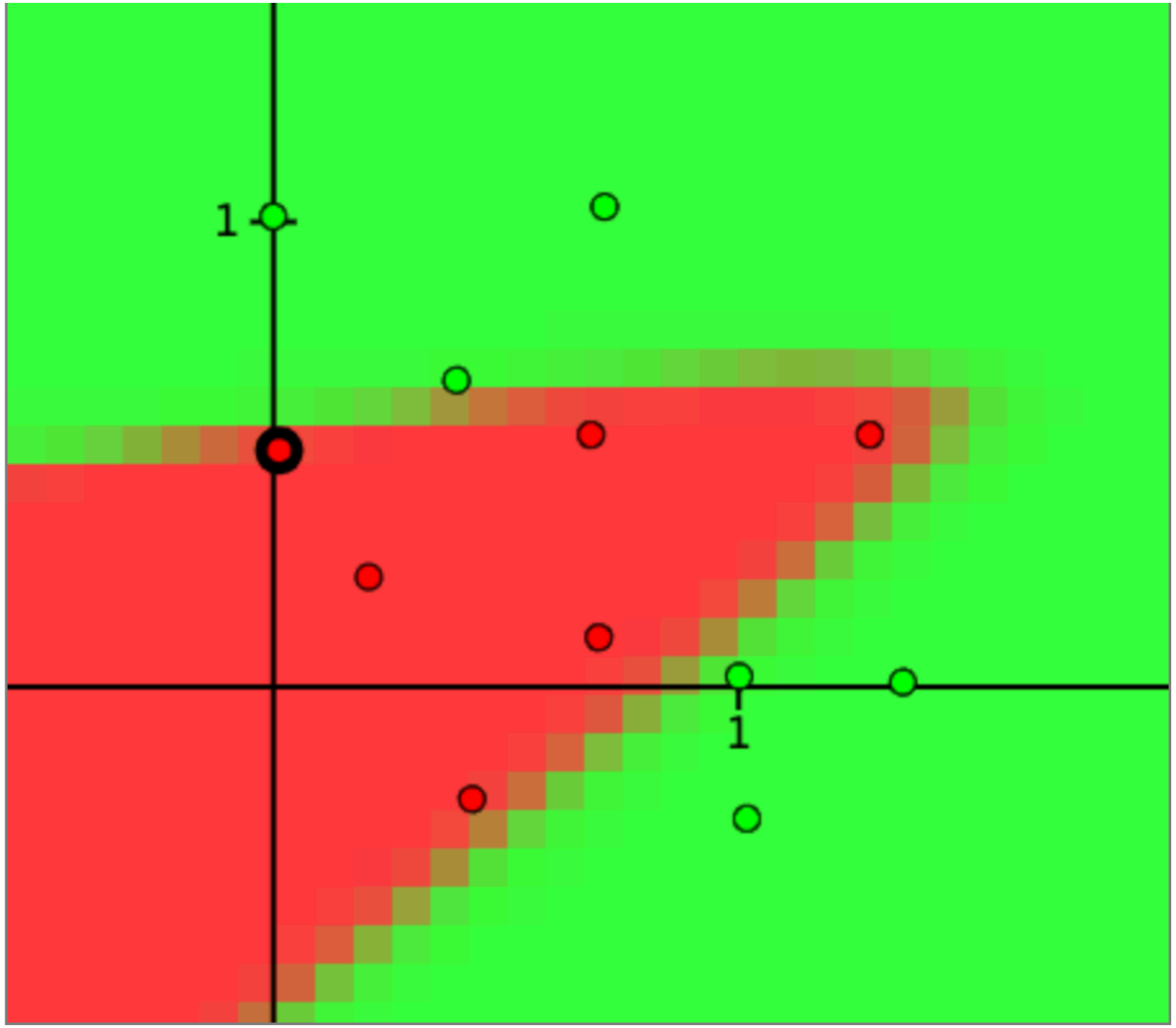


x_1 = 0.01
x_2 = 0.51
Bias (1)
0.51
0.00
Bias (1)
+ 0.51 · (-18.22)
+ 0.00 · (21.94)
+ 1.00 · (6.02)
∑ = -3.32

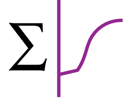


1
1


x_1 = 0.01
0.51
x_2 = 0.51
0.00
σ(-3.32) = 0.03
Bias (1)
Bias (1)

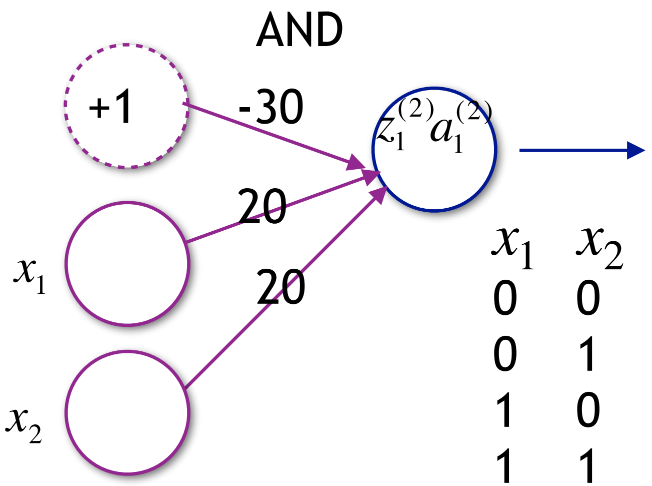

1
1

Computing Boolean Functions



$$f(z) = \frac{1}{1+\exp(-z)}$$

It is possible to build a neural network that computes any logical proposition:



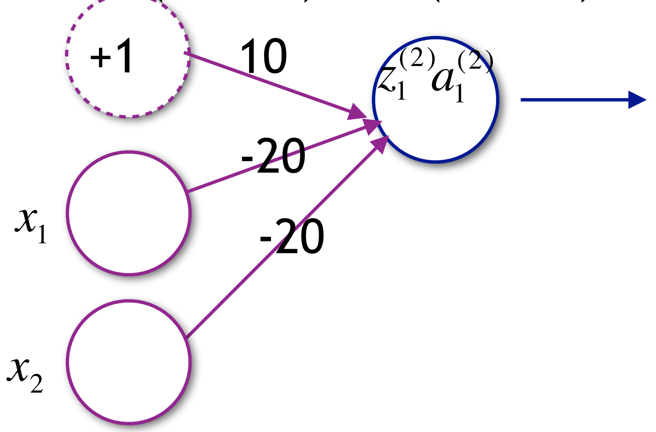
x_1	x_2	output
0	0	0
0	1	0
1	0	0
1	1	1

Computing Boolean Functions

$$f(z) = \frac{1}{1+\exp(-z)}$$

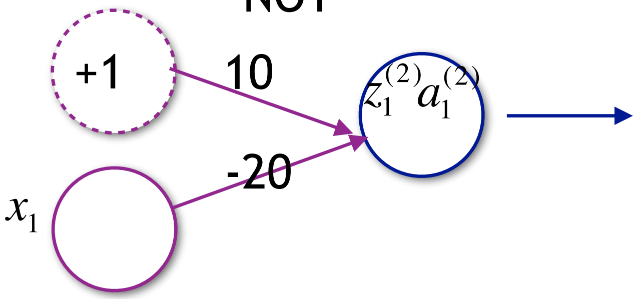
It is possible to build a neural network that computes any logical proposition:

(not x1) and (not x2)



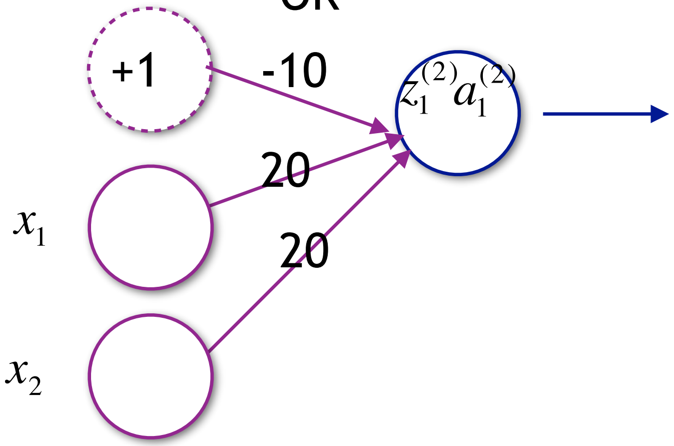
x_1	x_2	output
0	0	1
0	1	1
1	0	1
1	1	0

NOT



x_1	output
0	1
1	0

OR



x_1	x_2	output
0	0	0
0	1	1
1	0	1
1	1	1