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

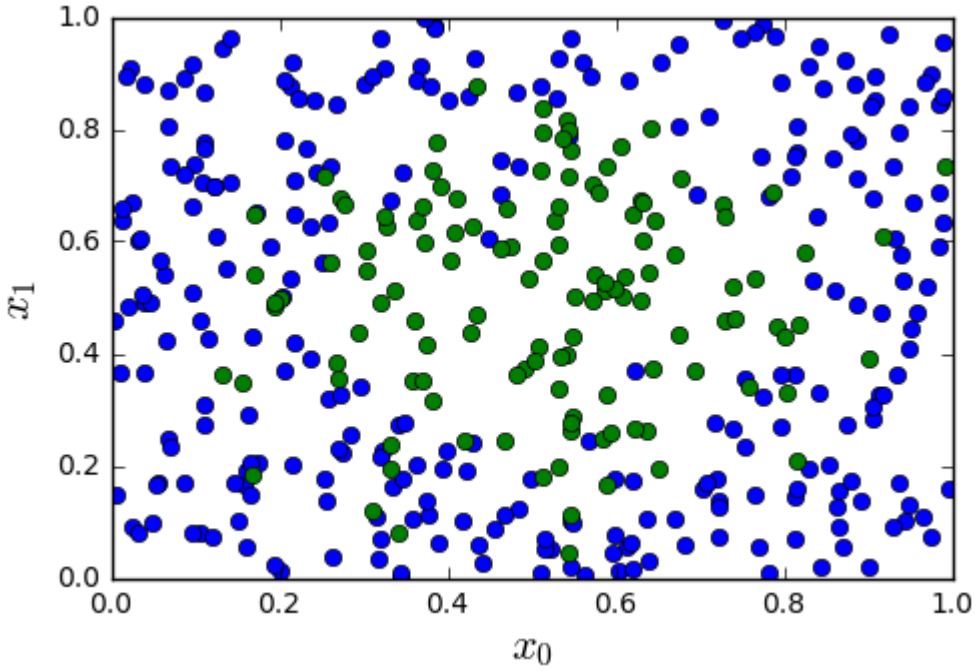
Neural networks ***extend*** logistic regression and linear regression

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

Outline

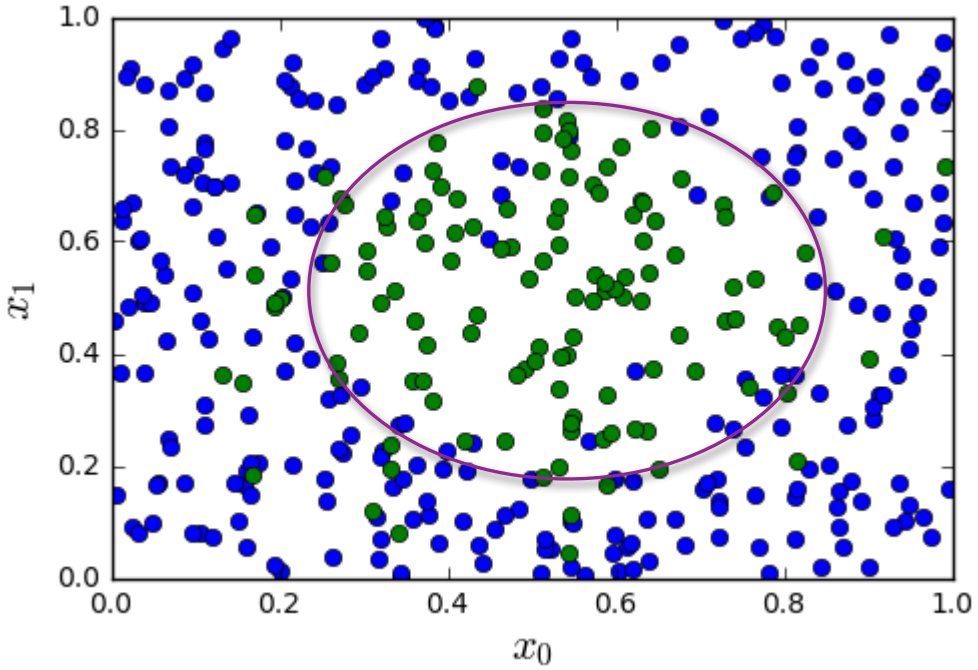
- ❑ Motivation Introduction to neurons
- ❑ Nonlinear classifiers from linear features
- ❑ Neural networks notation
- ❑ Pseudocode for prediction
- ❑ Training a neural network
- ❑ Implementing gradient descent for neural networks
  - Vectorization
  - Pseudocode
- ❑ Preprocessing
- ❑ Initialization
-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

Most Datasets are not Linearly Separable



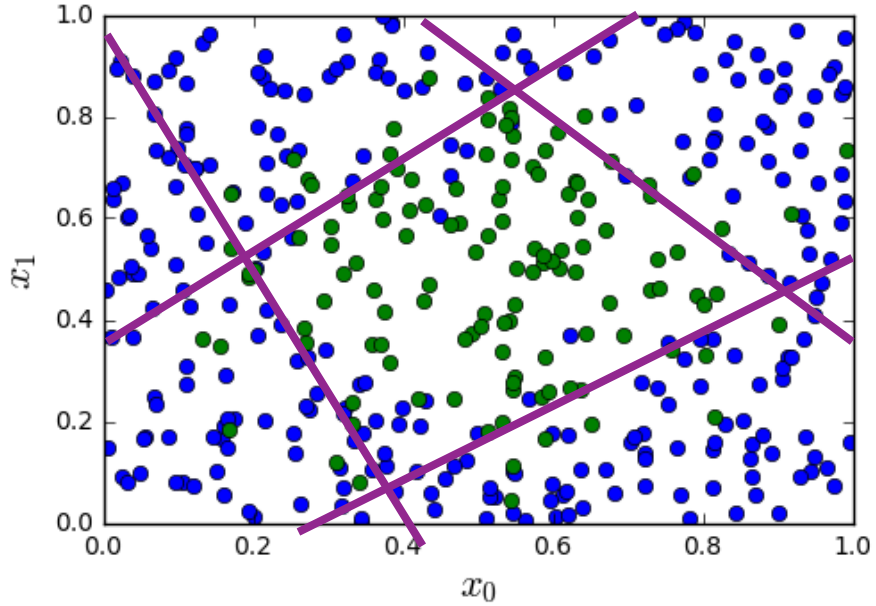
- Consider simple synthetic data
  - See figure to the left
  - 2D features
  - Binary class label
- Not separated linearly
- We can use Logistic Regression with non-linear features

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)?

THAT IS THE MAIN TOPIC OF THIS LECTURE

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

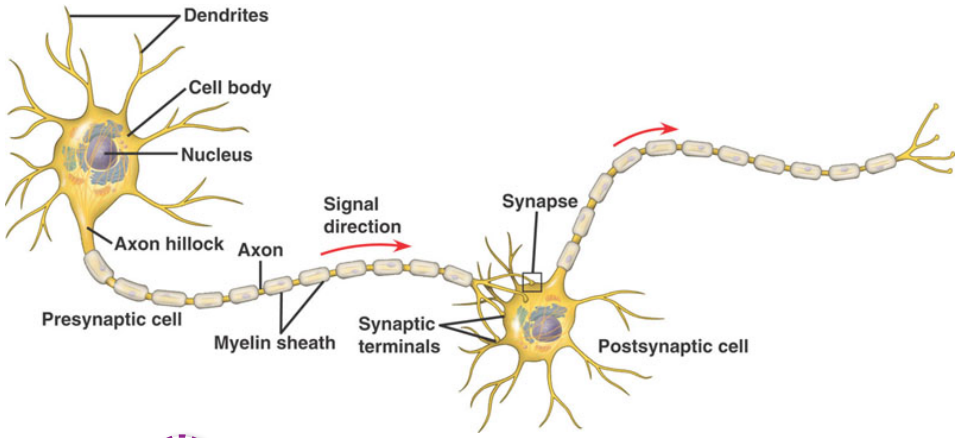
THAT IS THE MAIN TOPIC OF THIS LECTURE

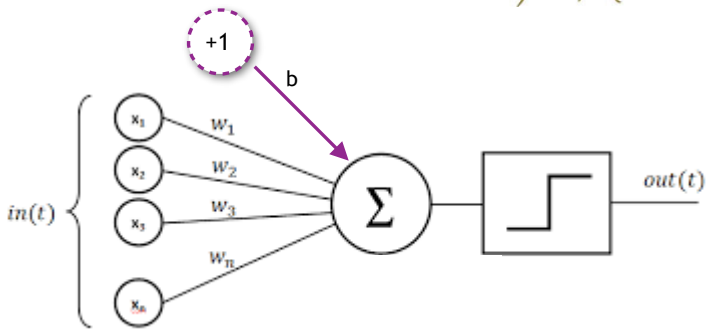
Recent Resurgence and Developments

- 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)

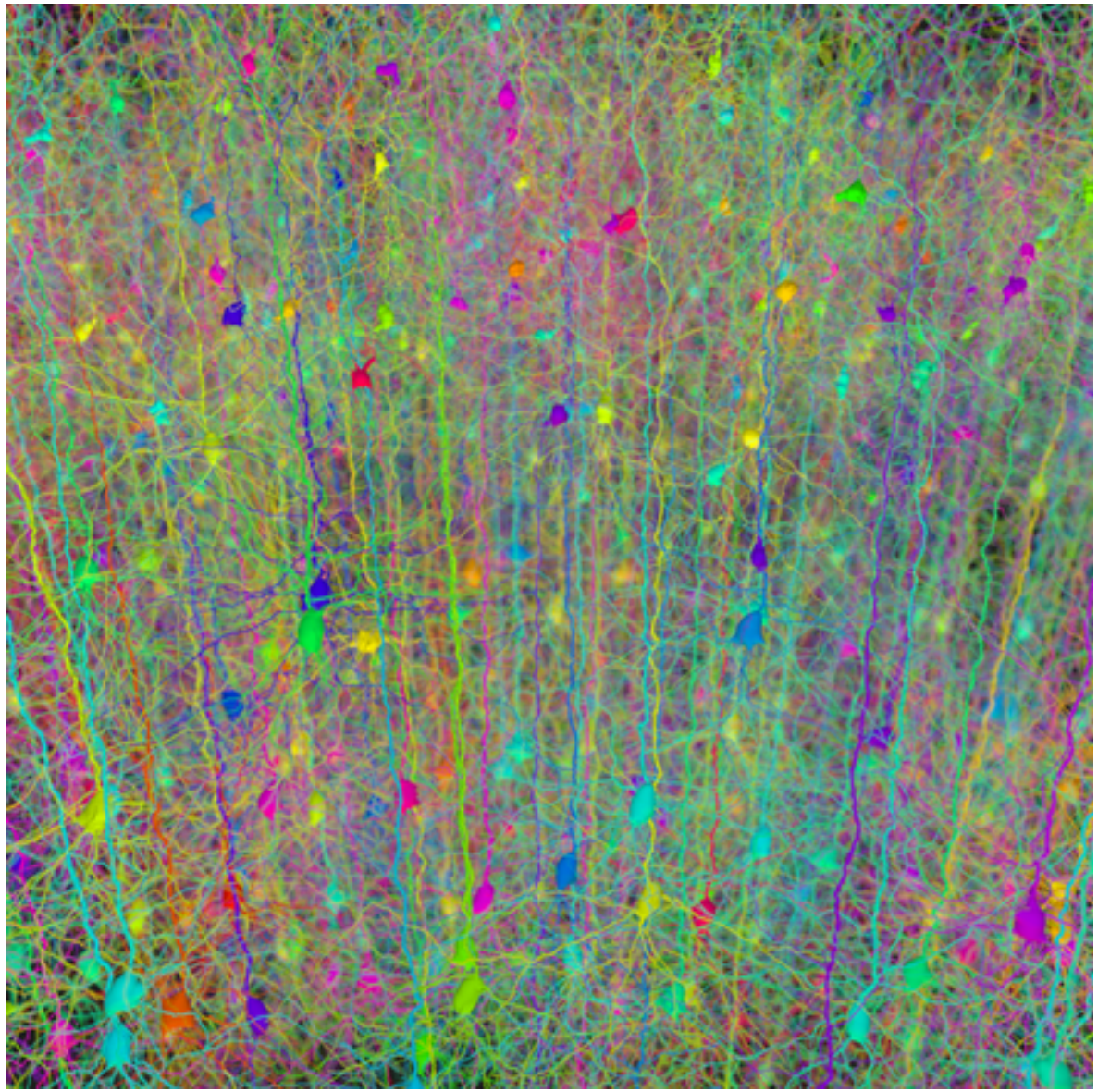
Inspiration from Biology

Abstract away biological construct into a mathematical construct





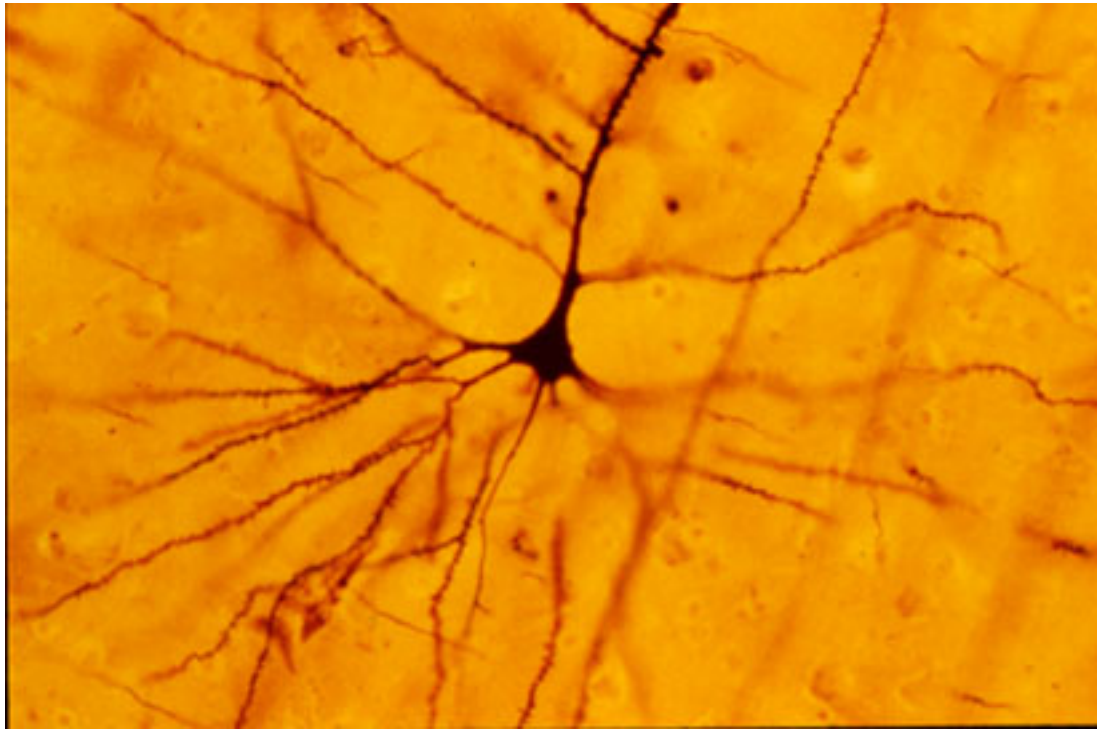
- *Simple* model of neurons
  - Dendrites: Input currents from other neurons
  - Soma: Cell body, accumulation of charge
  - Axon: Outputs to other neurons
  - Synapse: Junction between neurons
- Operation:
  - Take weighted sum of input current
  - Outputs when sum reaches a threshold
- Each neuron 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]

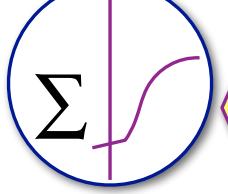
https://en.wikipedia.org/wiki/Neural_network

"**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

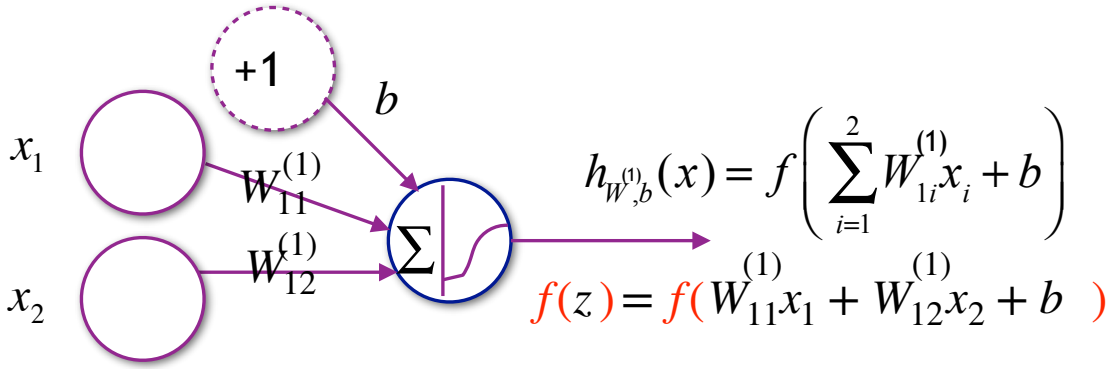
Perceptron / Neuron



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

"A **perceptron** is a simple model of a biological neuron in an artificial neural network. ... The **perceptron algorithm** was designed to classify visual inputs, categorizing subjects into one of two types and separating groups with a line."

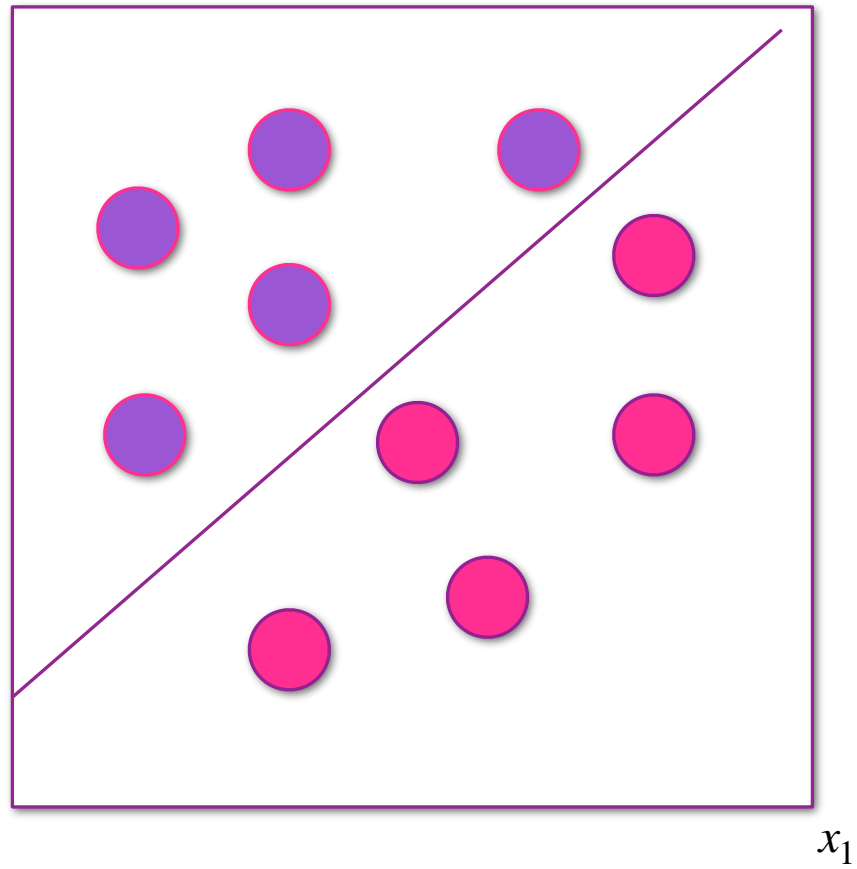
What kind of decision boundary can a single "neuron" create?



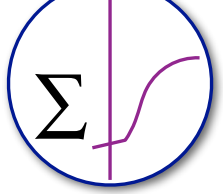
$$h_{W^{(1)},b}(x) = f\left(\sum_{i=1}^{2} W_{1i}^{(1)} x_i + b\right)$$

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

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



Perceptron / Neuron



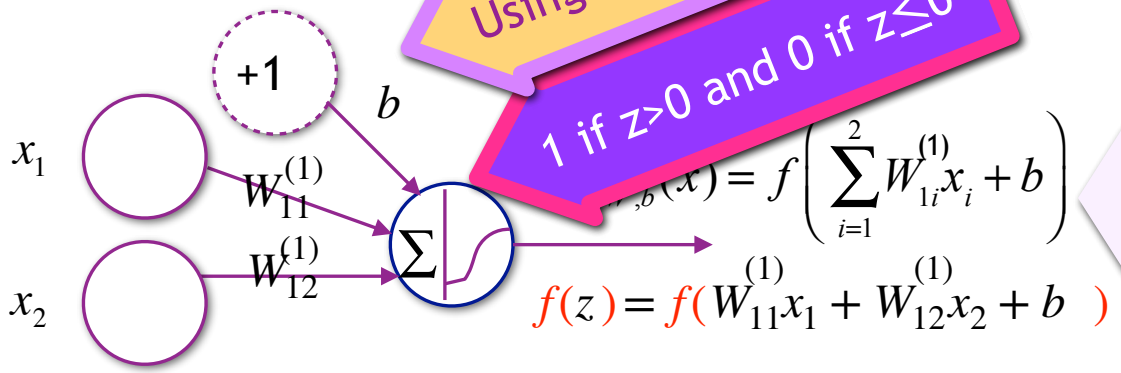
For this lecture will use the sigmoid function

"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 in… 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



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

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

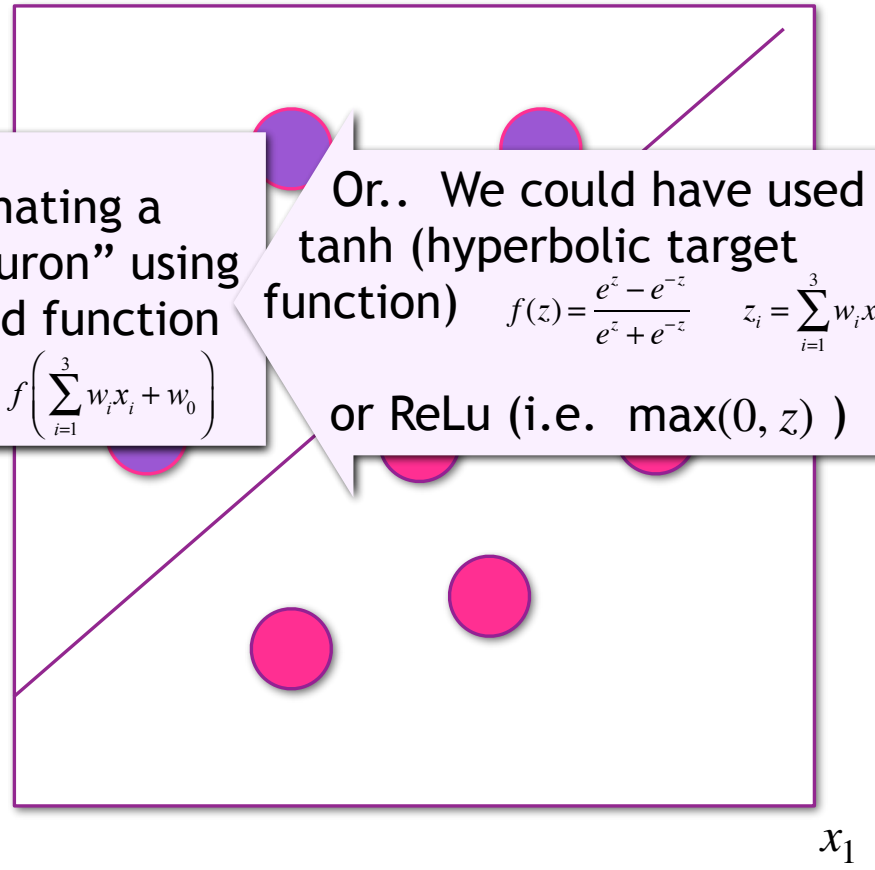
$$z = [W_{11}^{(1)}, W_{12}^{(1)}] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + b$$

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 …

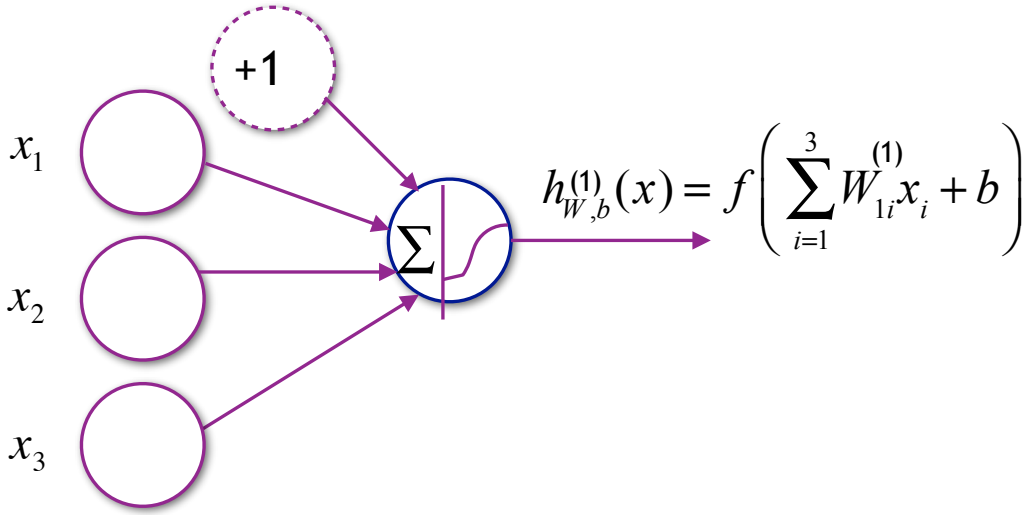


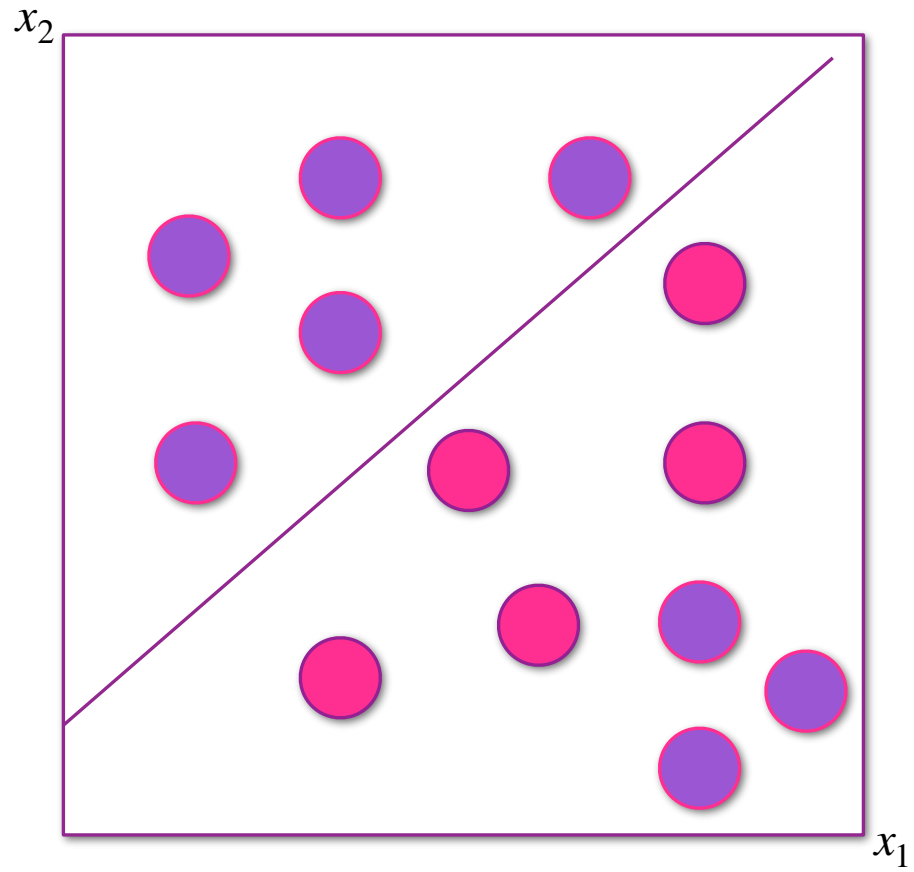
Outline

- ❑ 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?

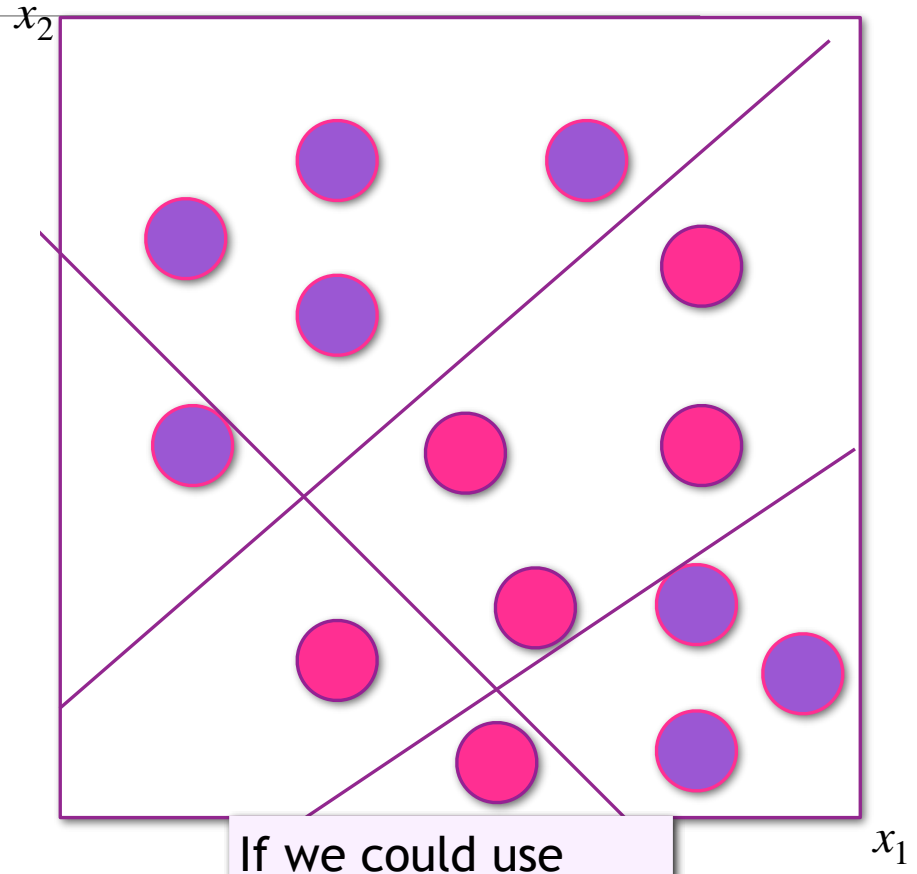




How could we learn a more complicated decision boundary?

Pair share: How can we do this?

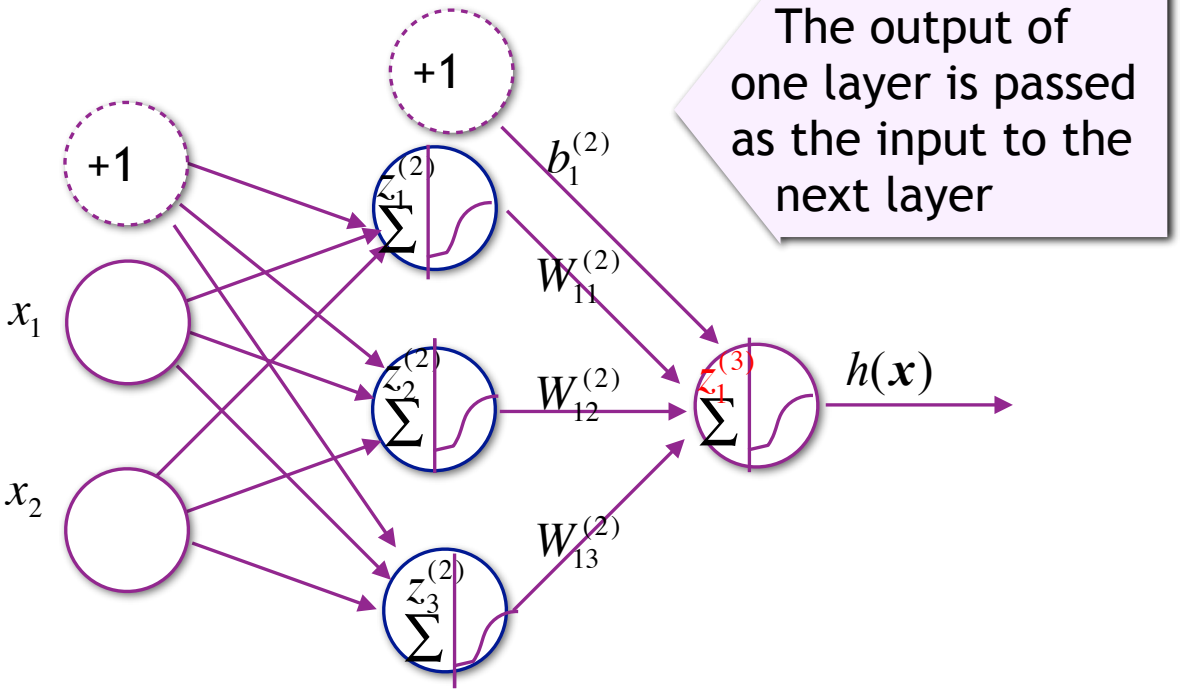
It would be easy if our features were the 3 hyperplanes…



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

Feature Construction!

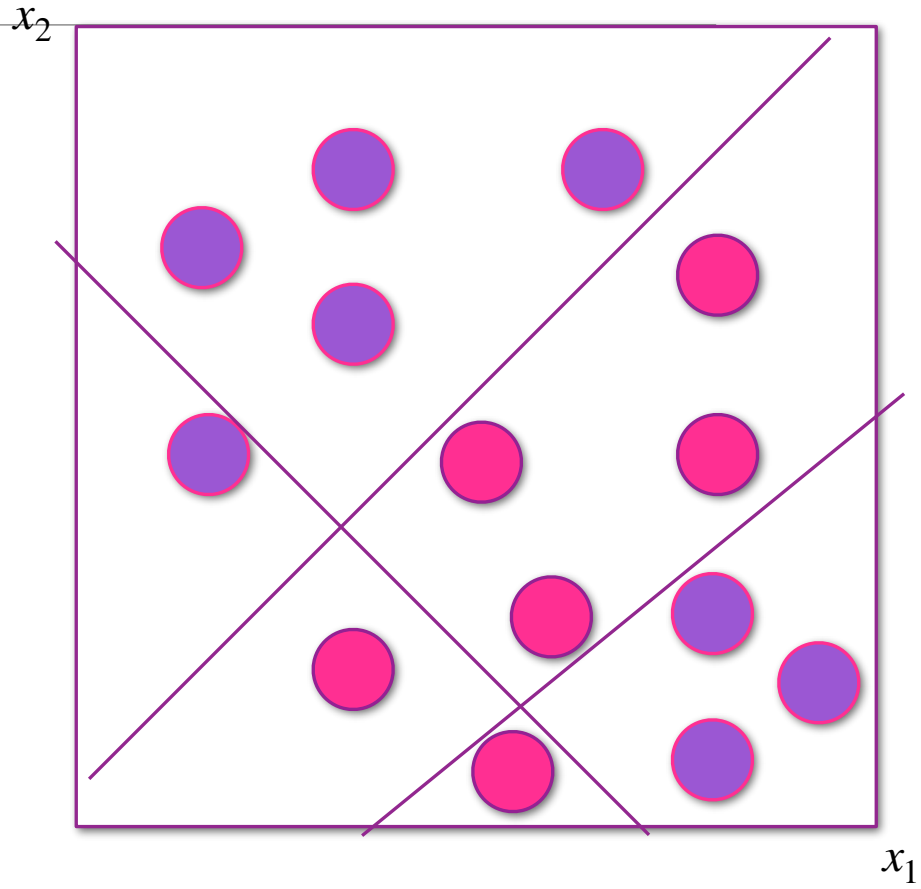
Creating a more useful set of features that allow for a linear decision boundary



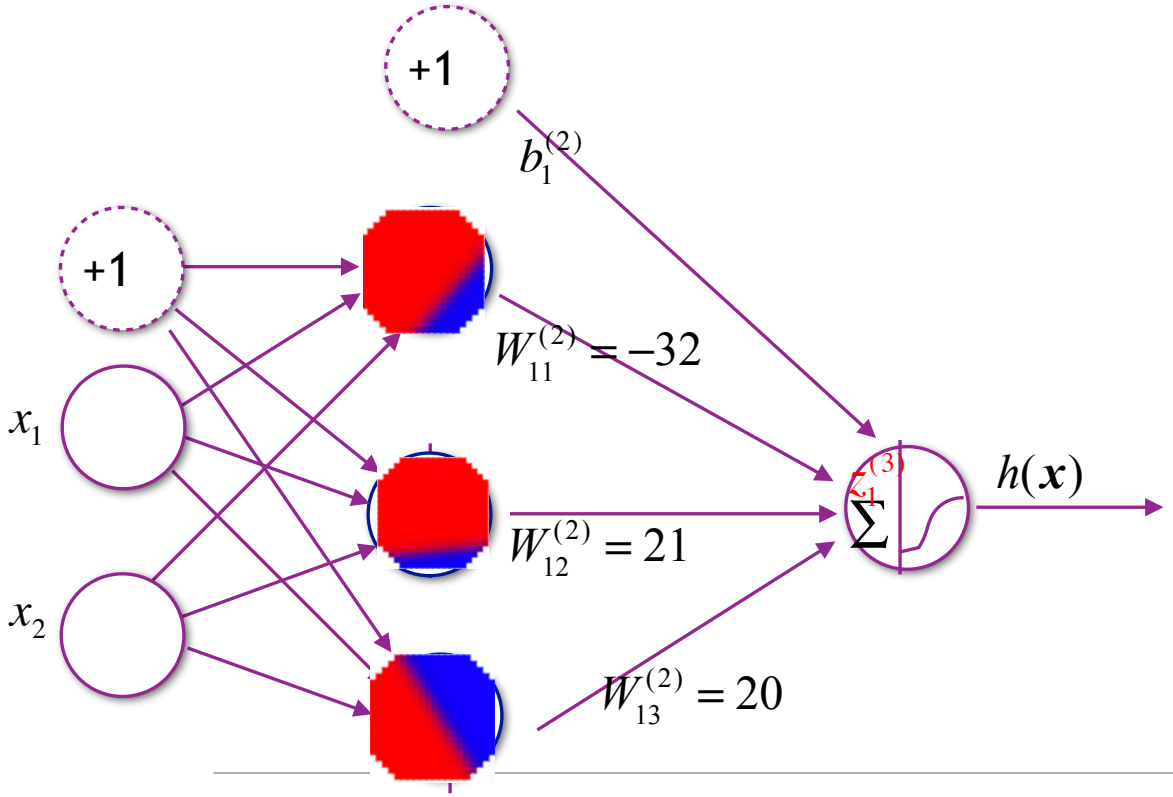
The output of one layer is passed as the input to the next layer

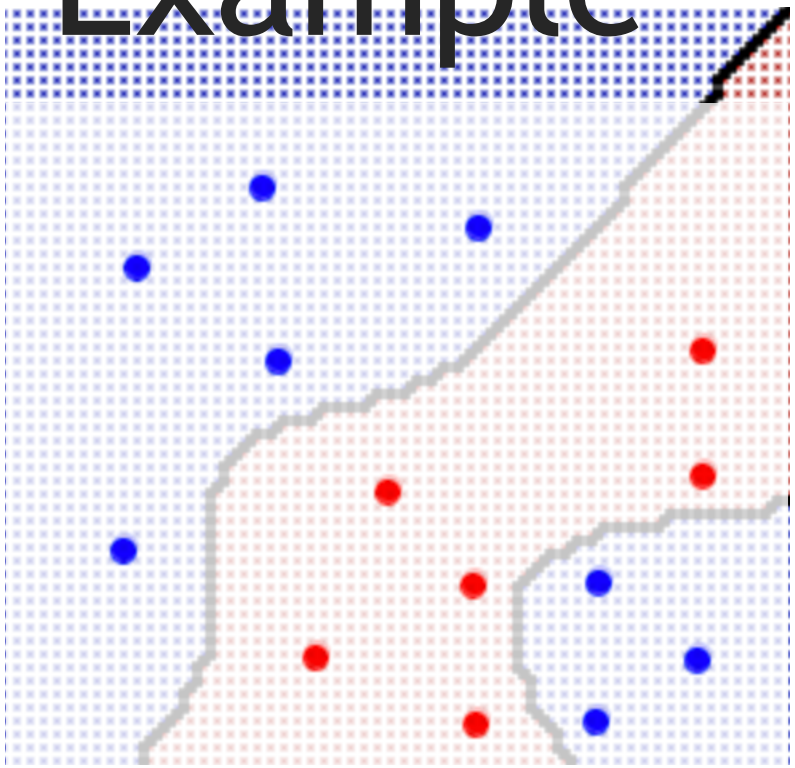
$$z_1^{(3)} = \sum_{j=1}^{3} W_{1j}^{(2)} f(z_j^{(2)}) + b_1^{(2)}$$

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



Example

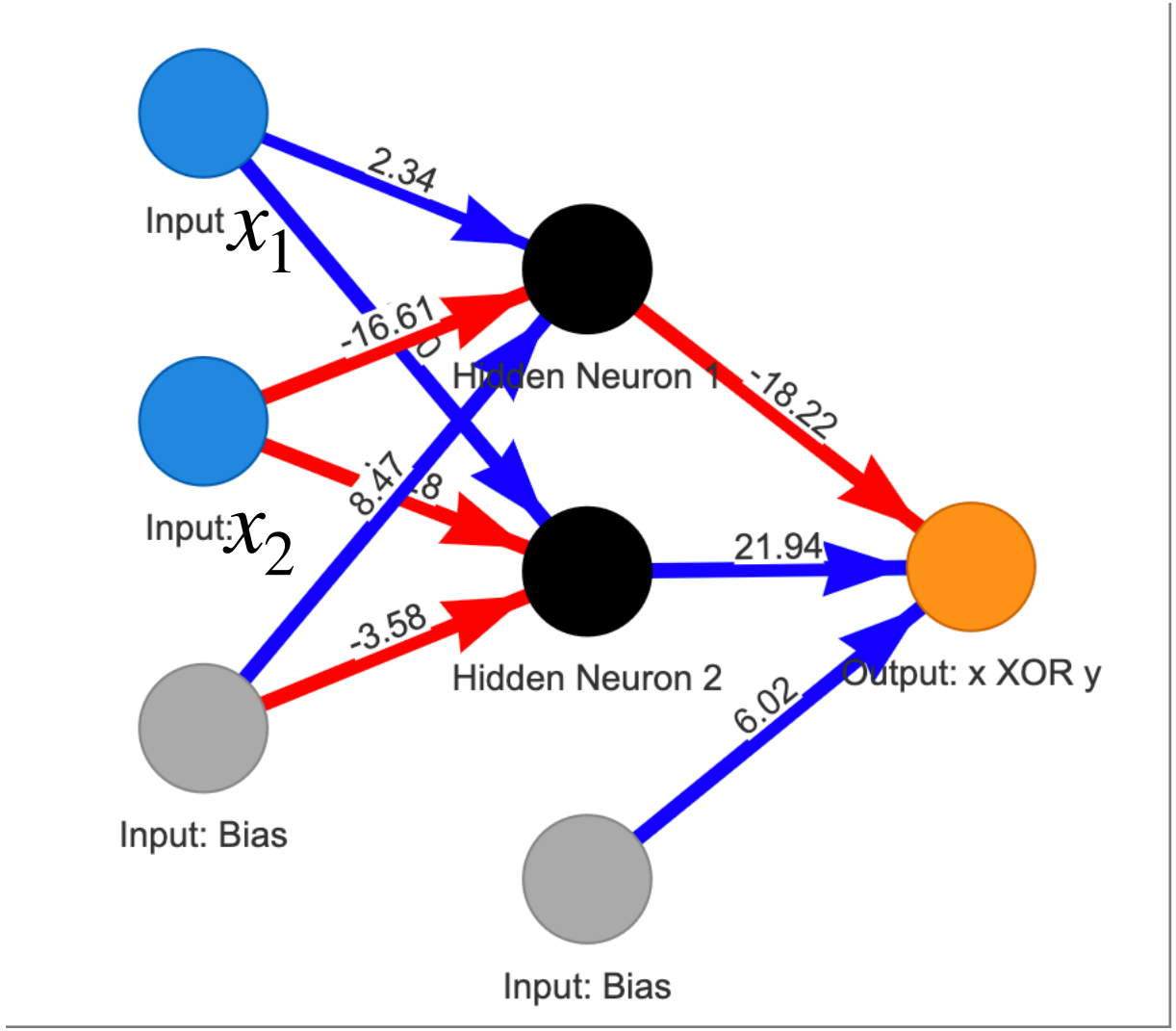




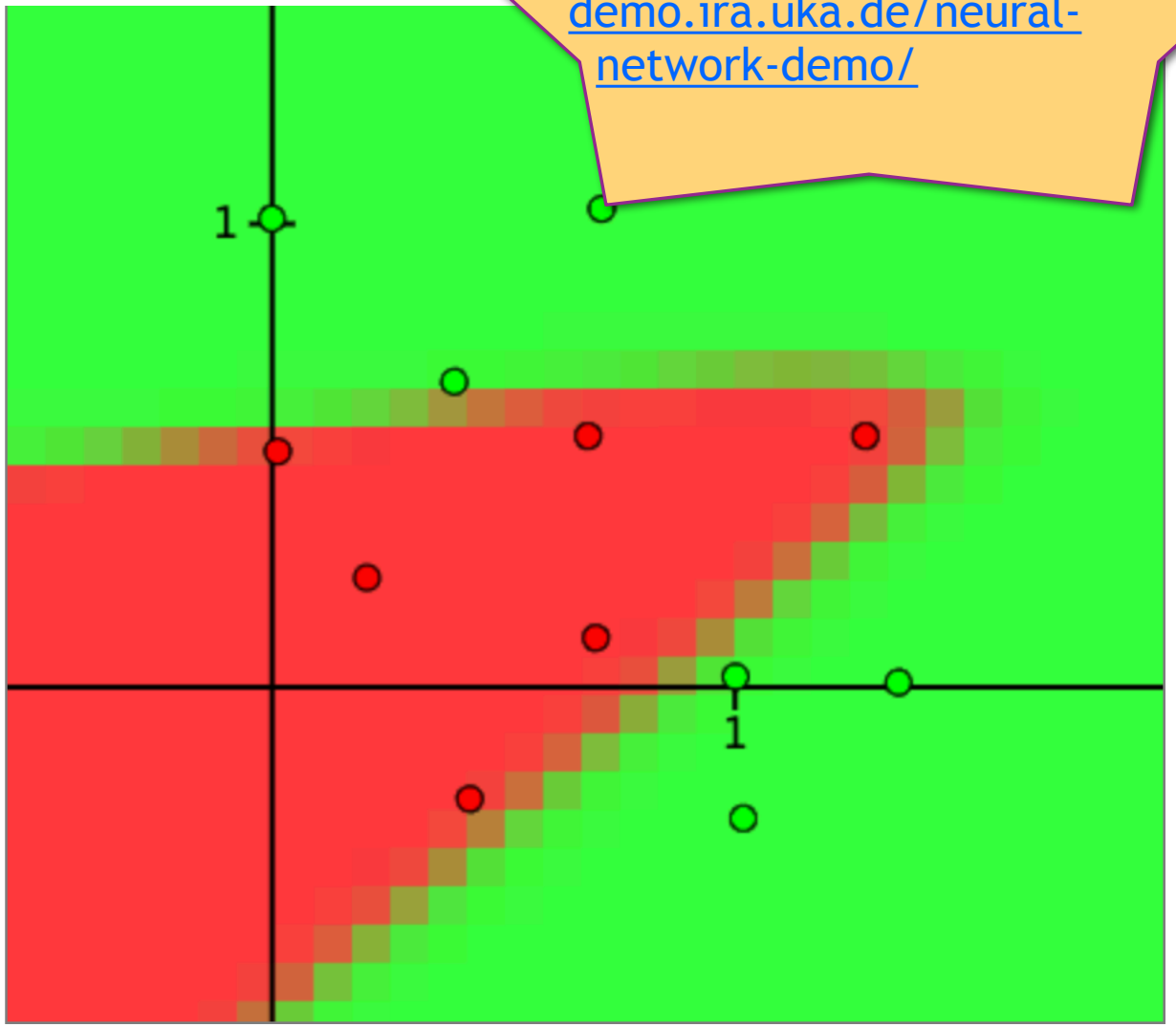
Example generated from

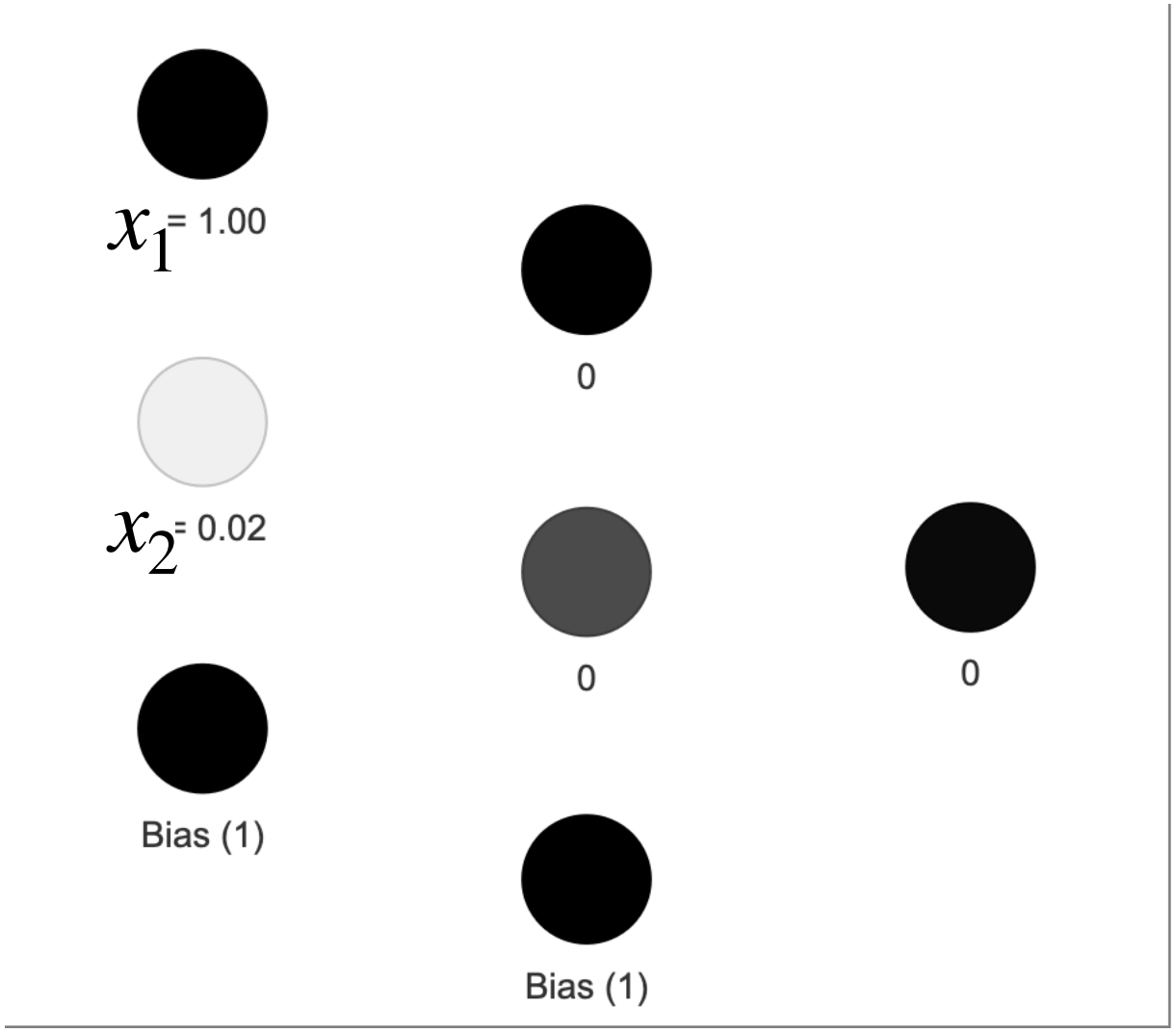
http://www.ccom.ucsd.edu/~cdeotte/programs/neuralnetwork.html

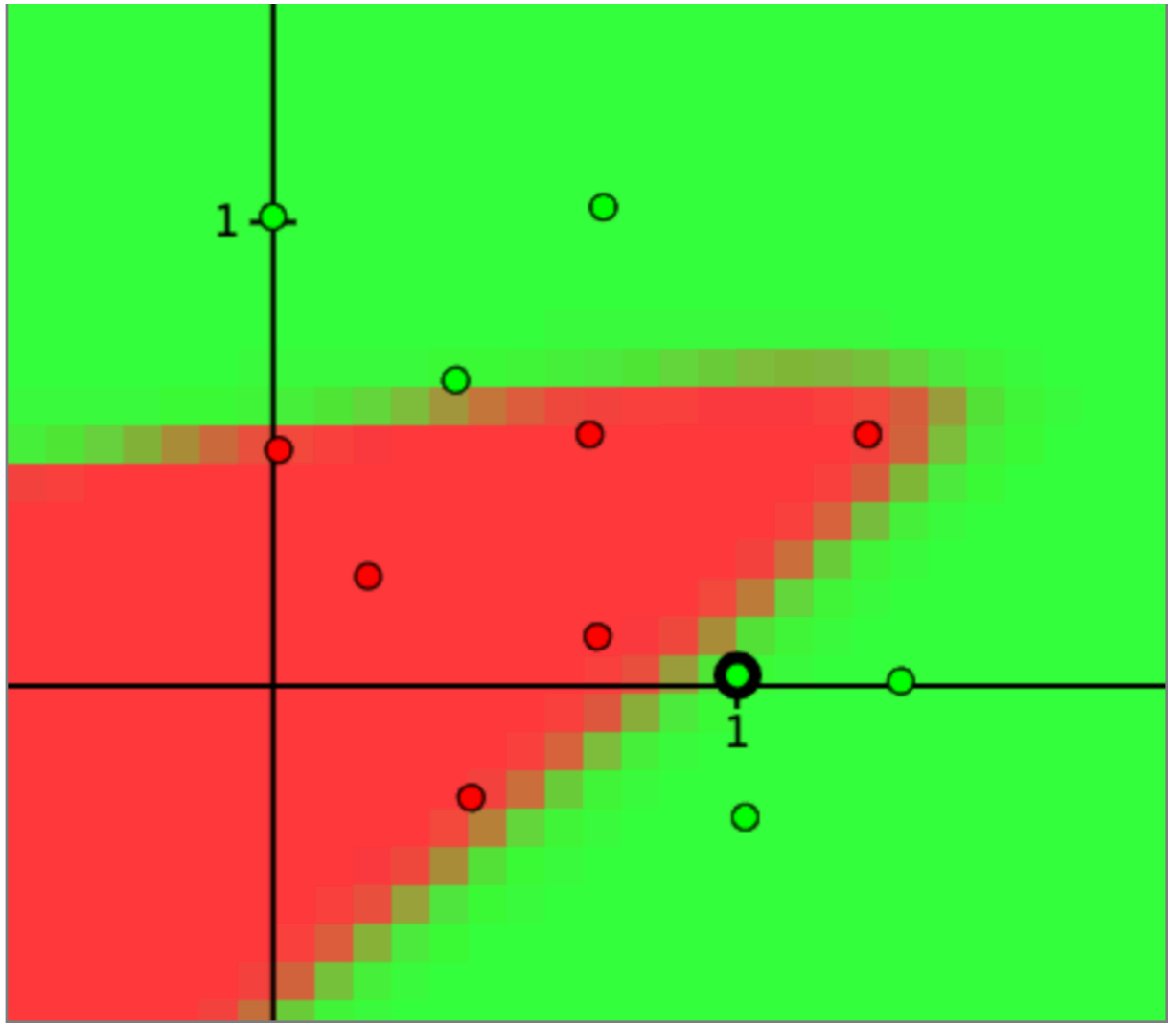
Prediction using a neural network

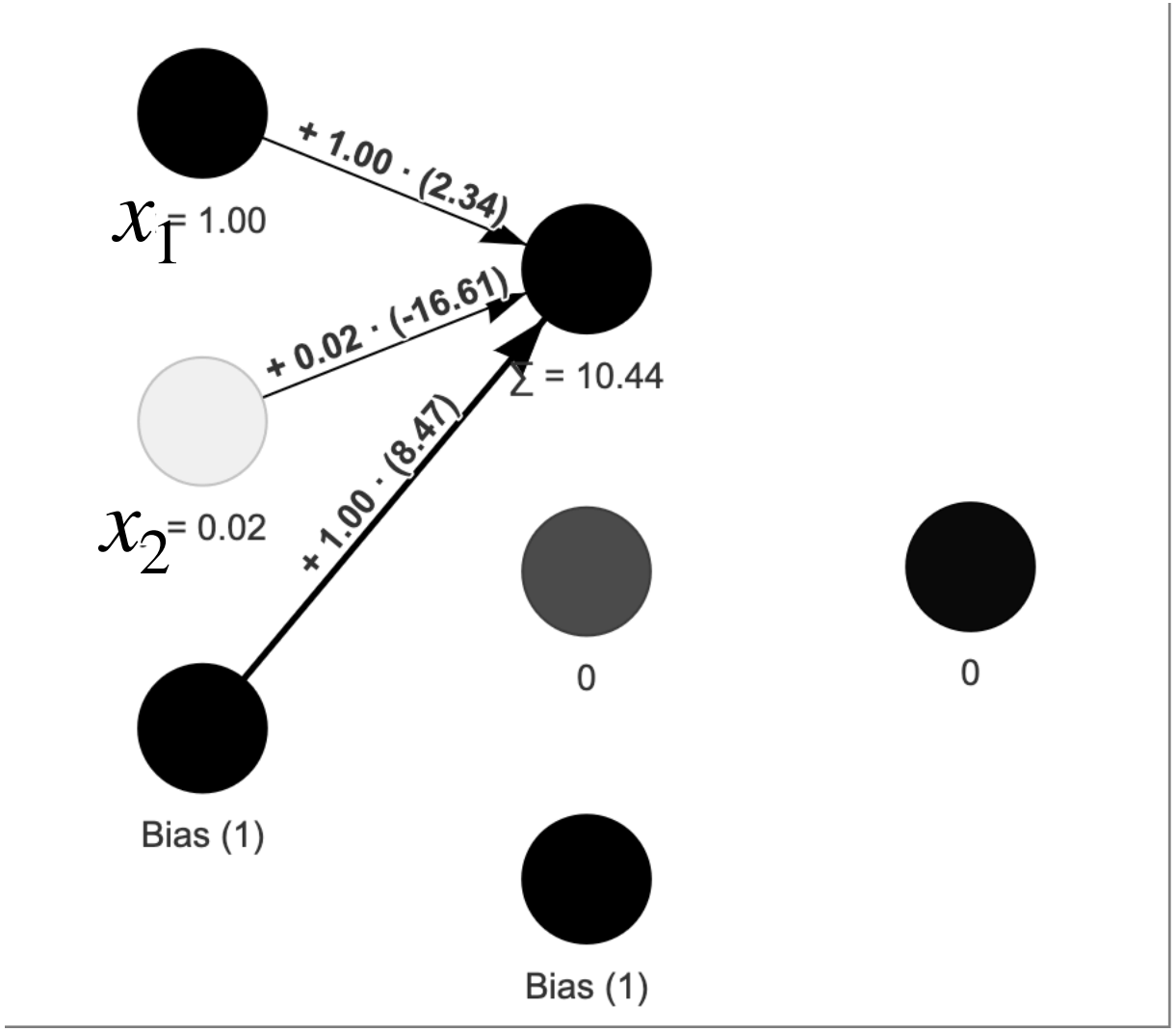


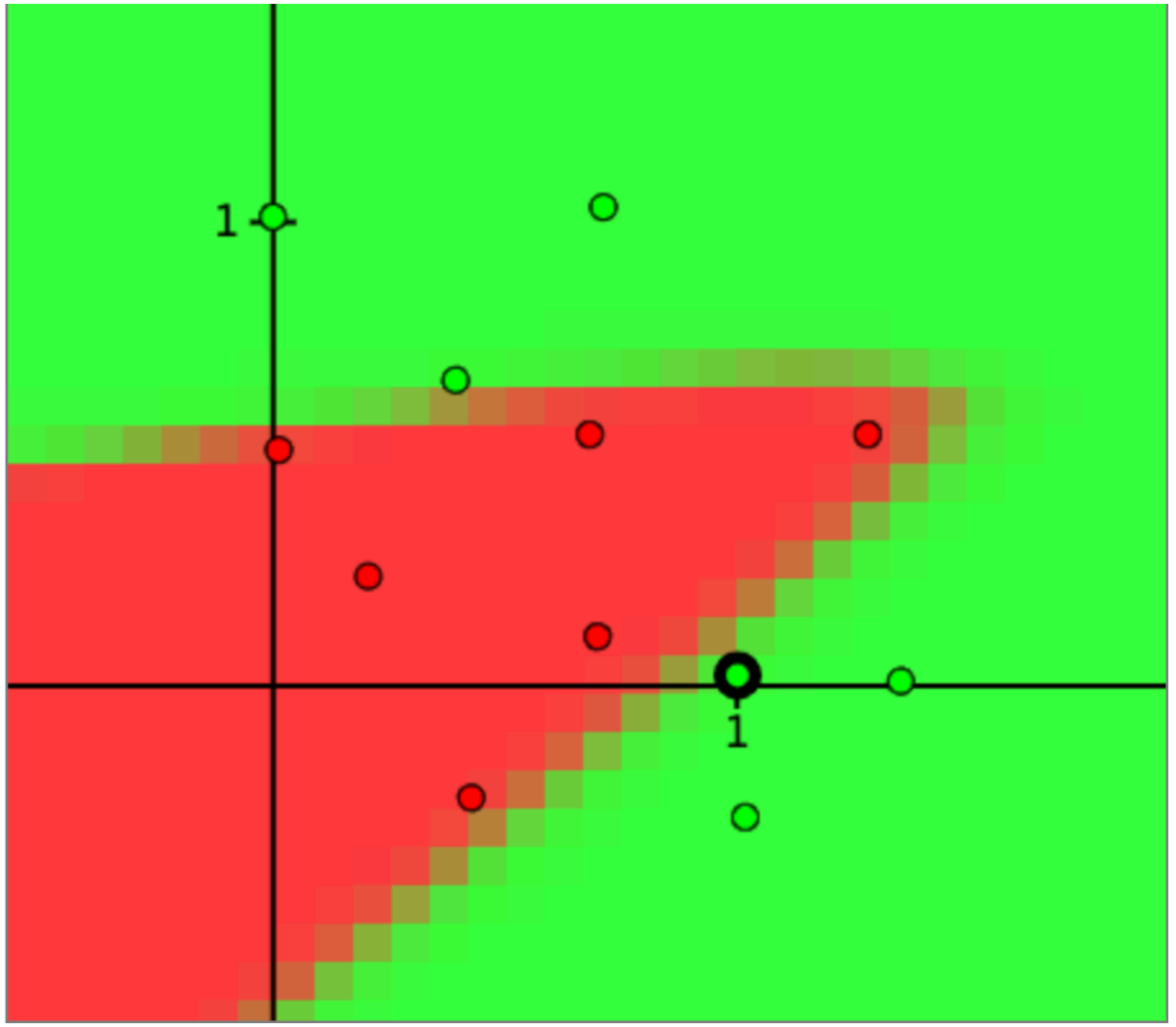
Demo! https://lecture-demo.ira.uka.de/neural-network-demo/

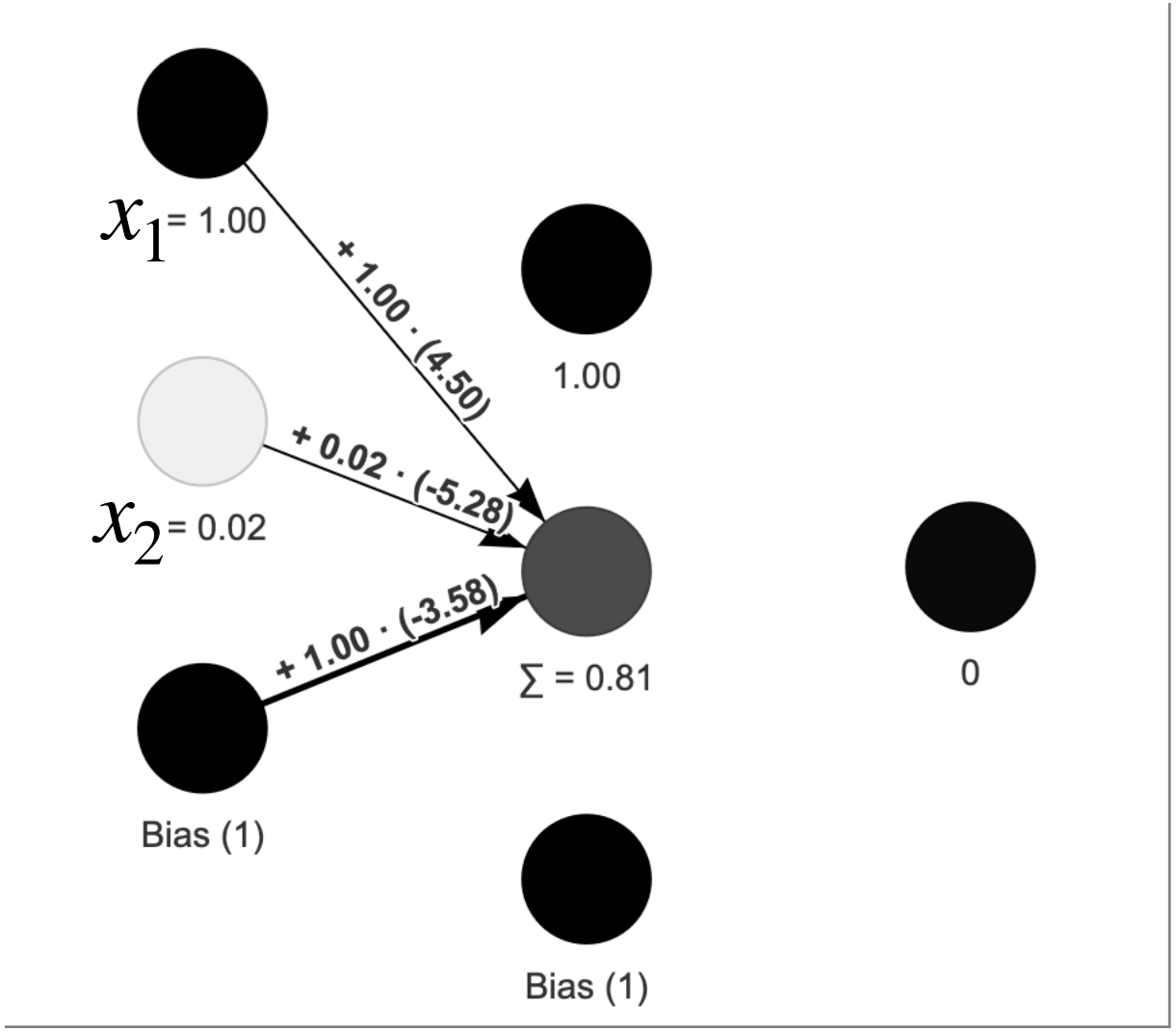


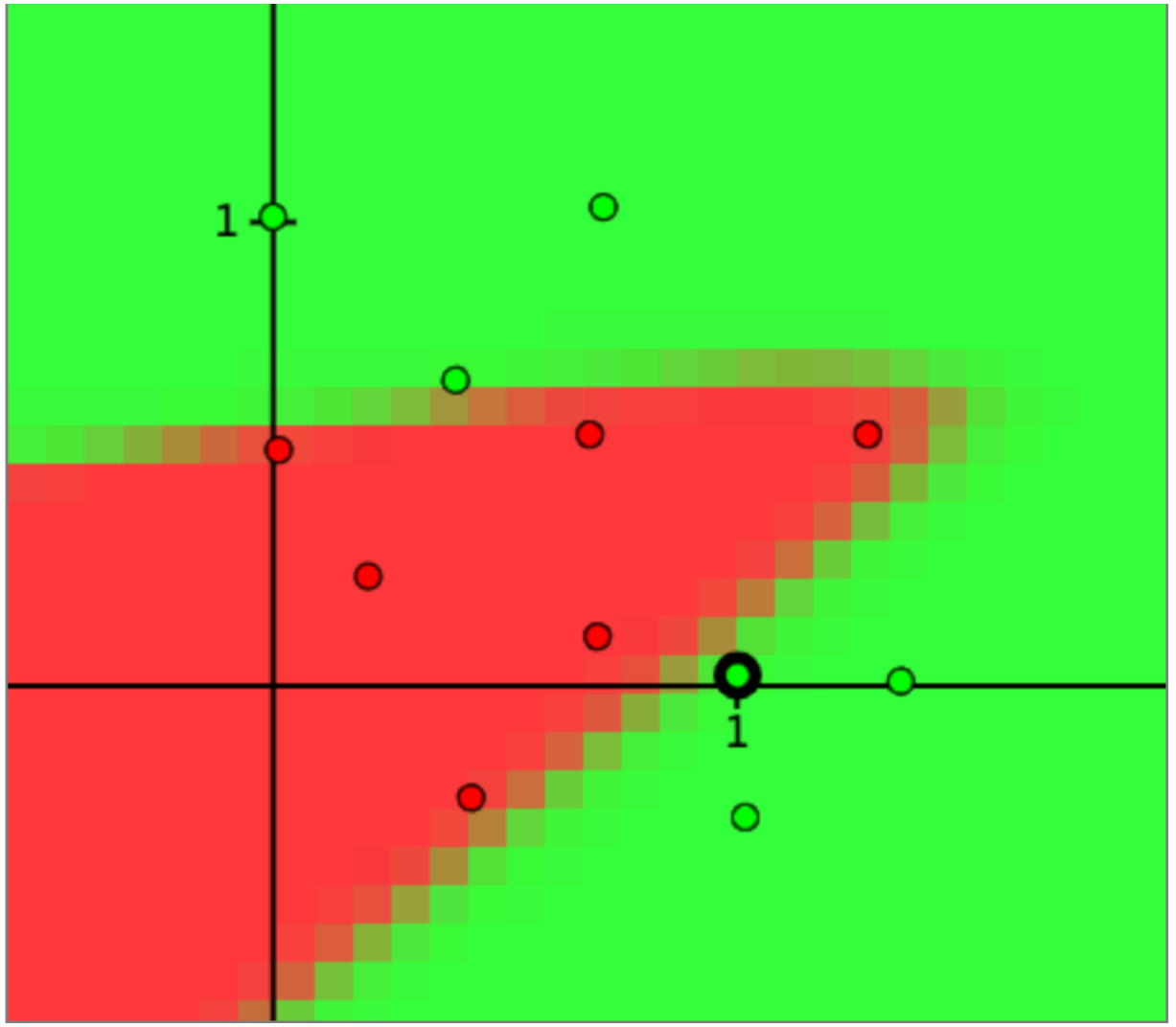

x_1 = 1.00
x_2 = 0.02
Bias (1)
0
0
0
Bias (1)

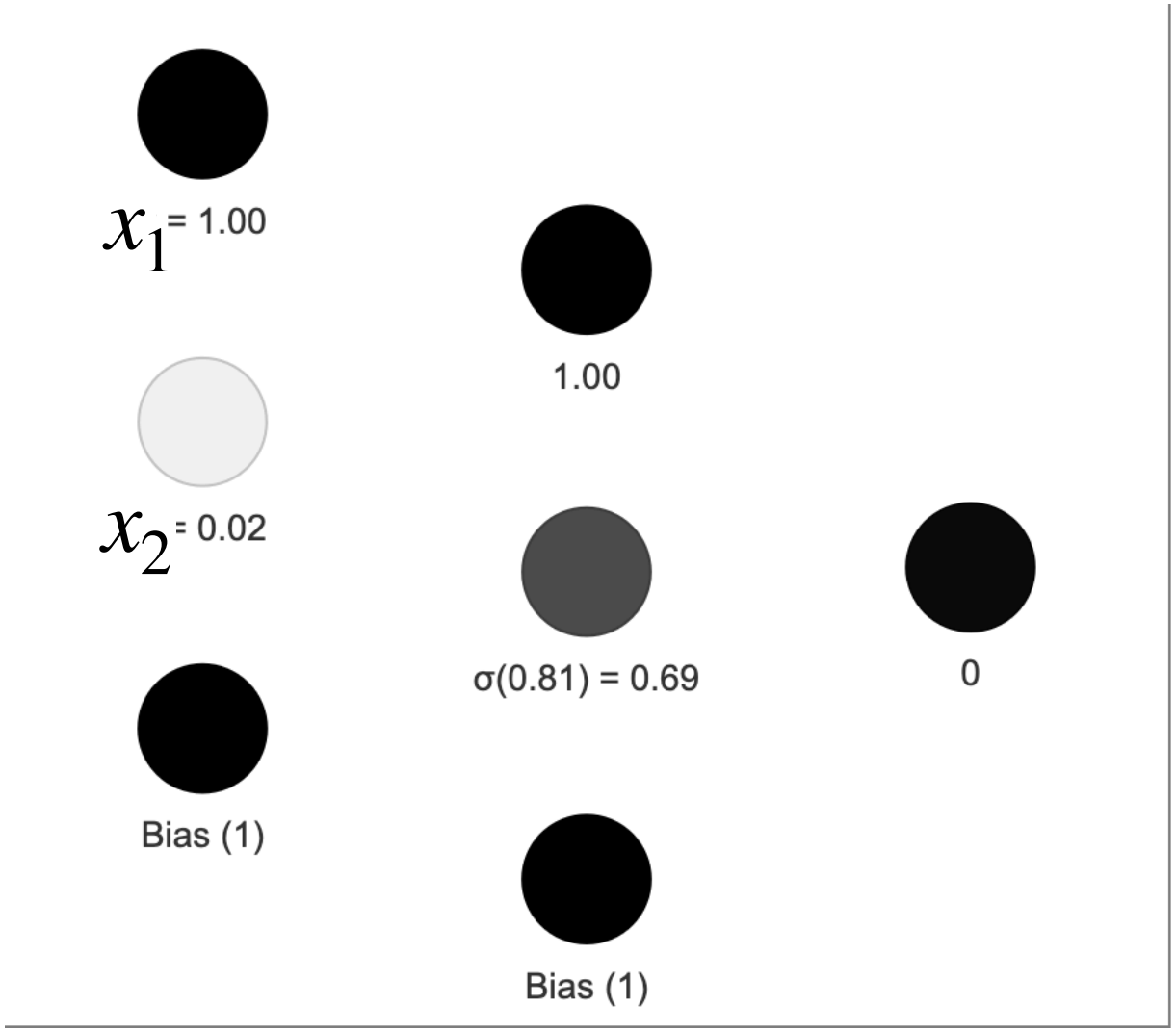

1
1

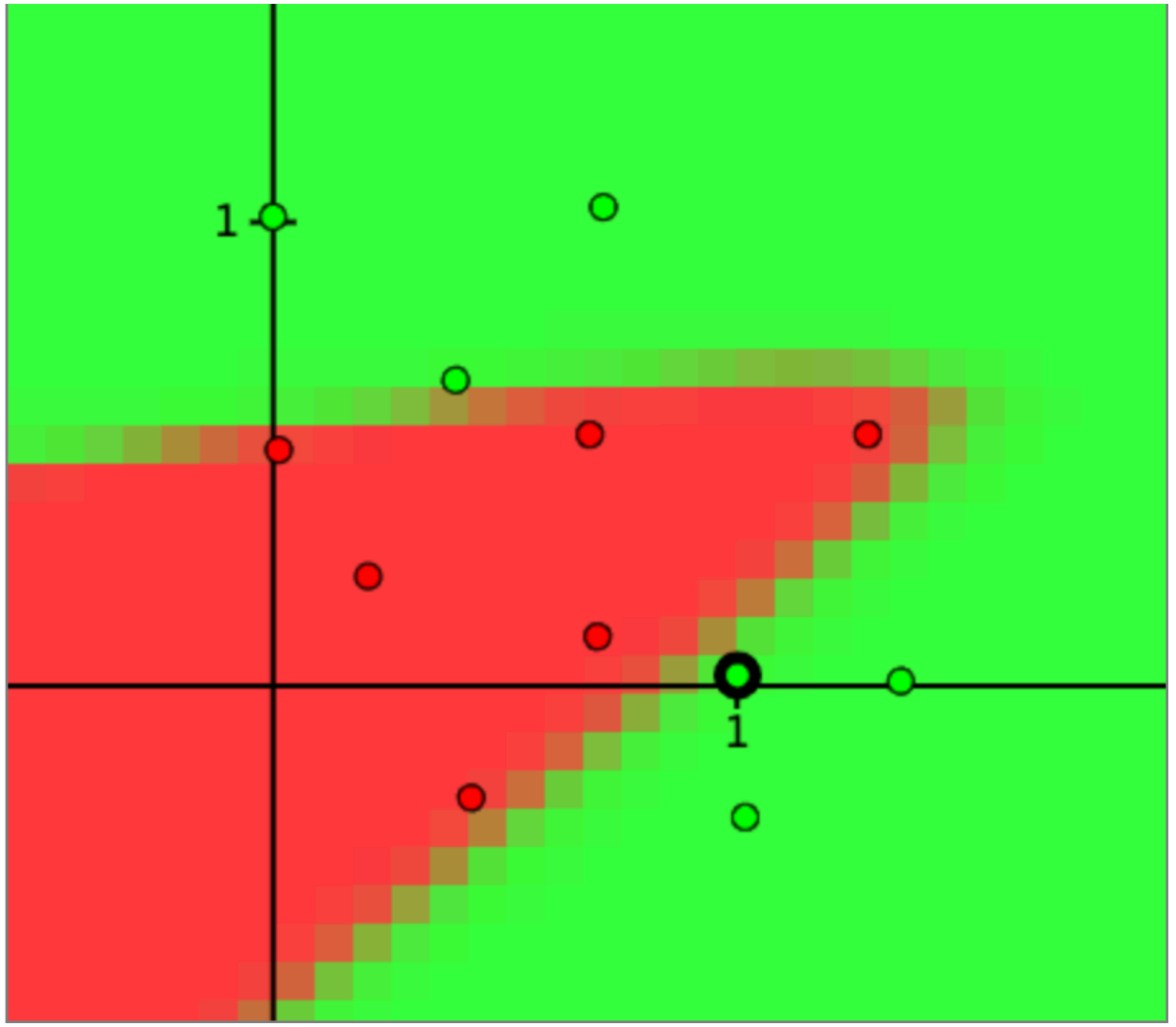

+ 1.00 · (2.34)
x_1 = 1.00
+ 0.02 · (-16.61)
Σ = 10.44
x_2 = 0.02
+ 1.00 · (8.47)
0
0
Bias (1)
Bias (1)

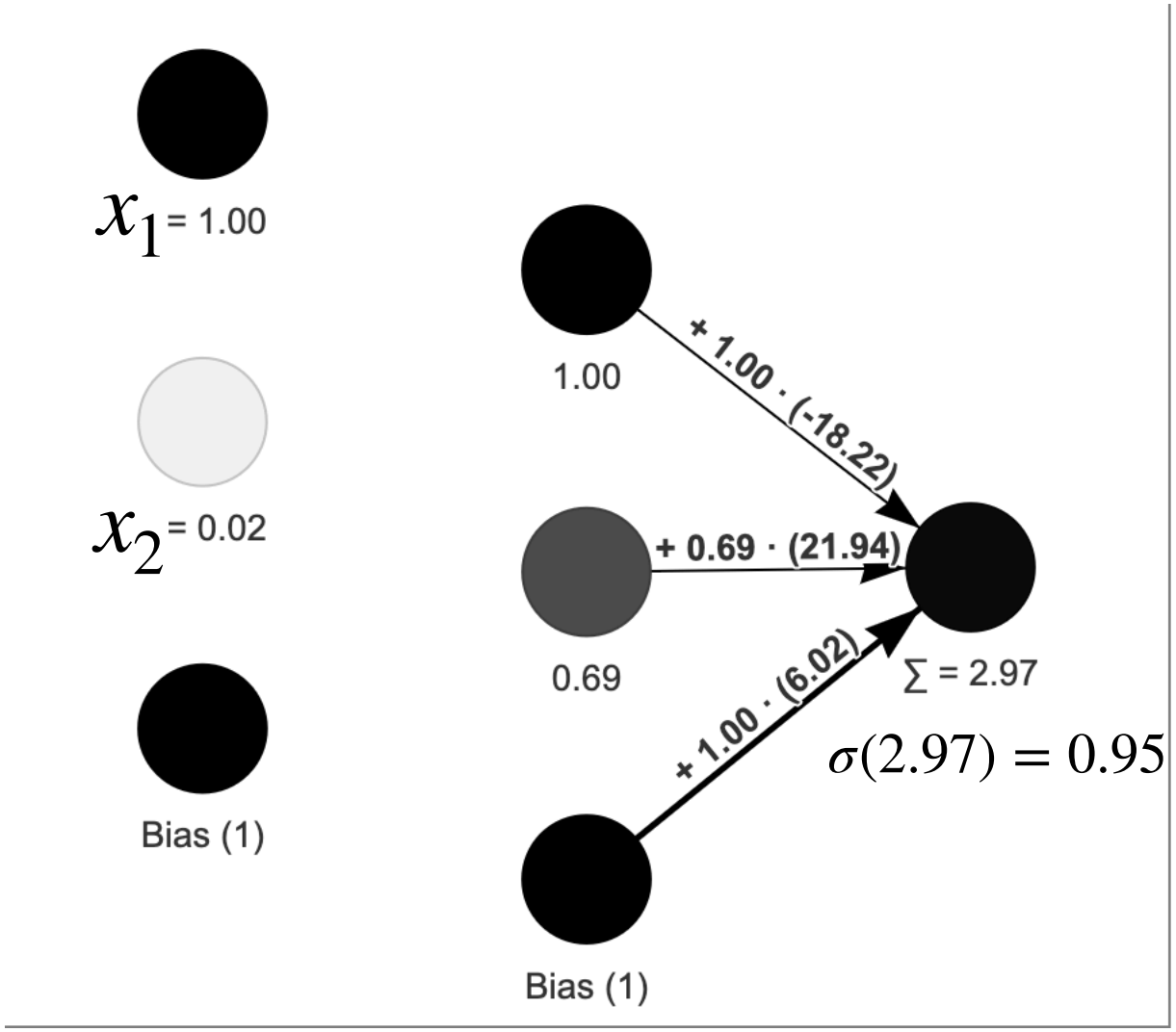

1
1

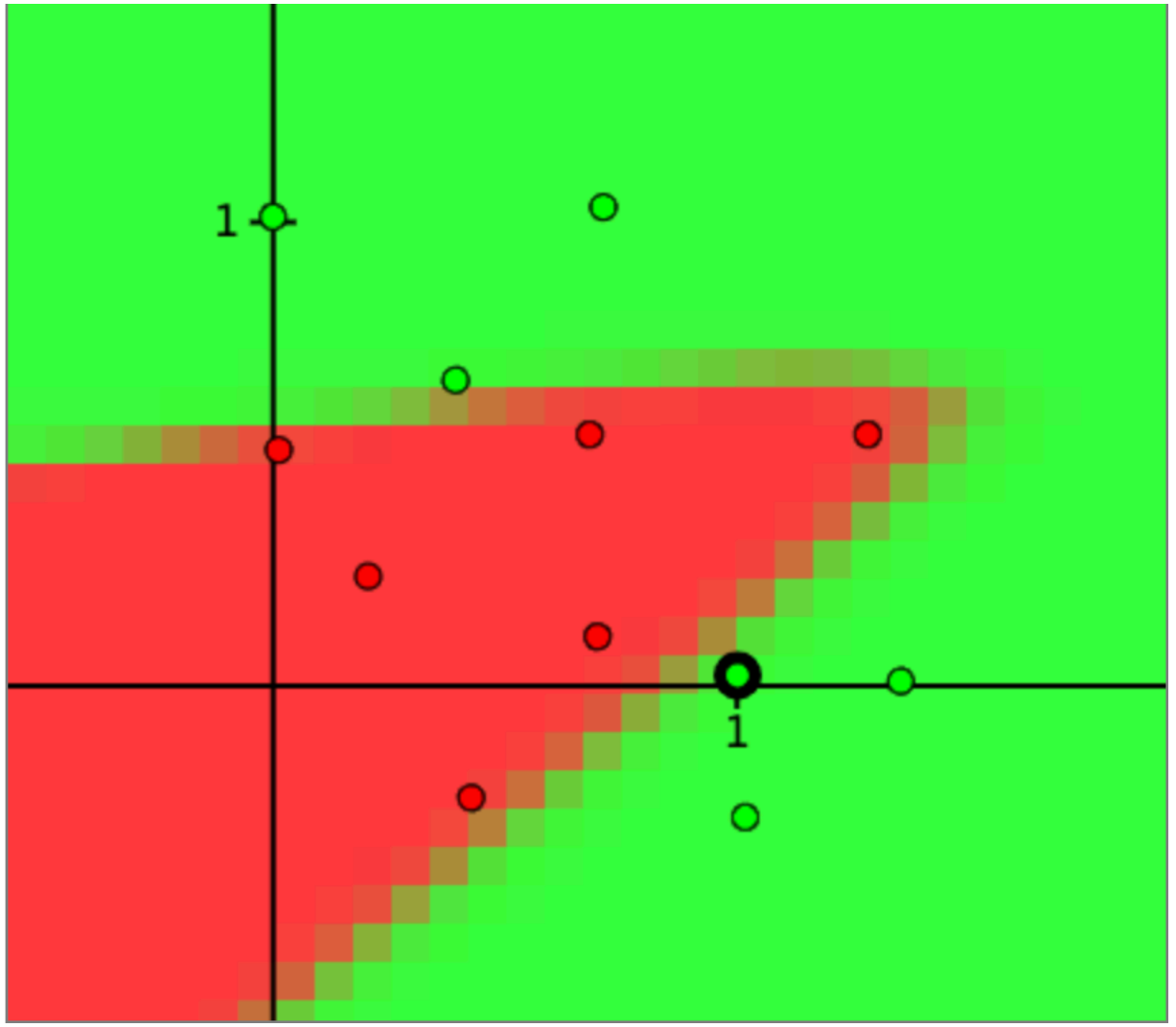

x_1 = 1.00
x_2 = 0.02
+ 1.00 · (4.50)
+ 0.02 · (-5.28)
+ 1.00 · (-3.58)
1.00
∑ = 0.81
0
Bias (1)
Bias (1)

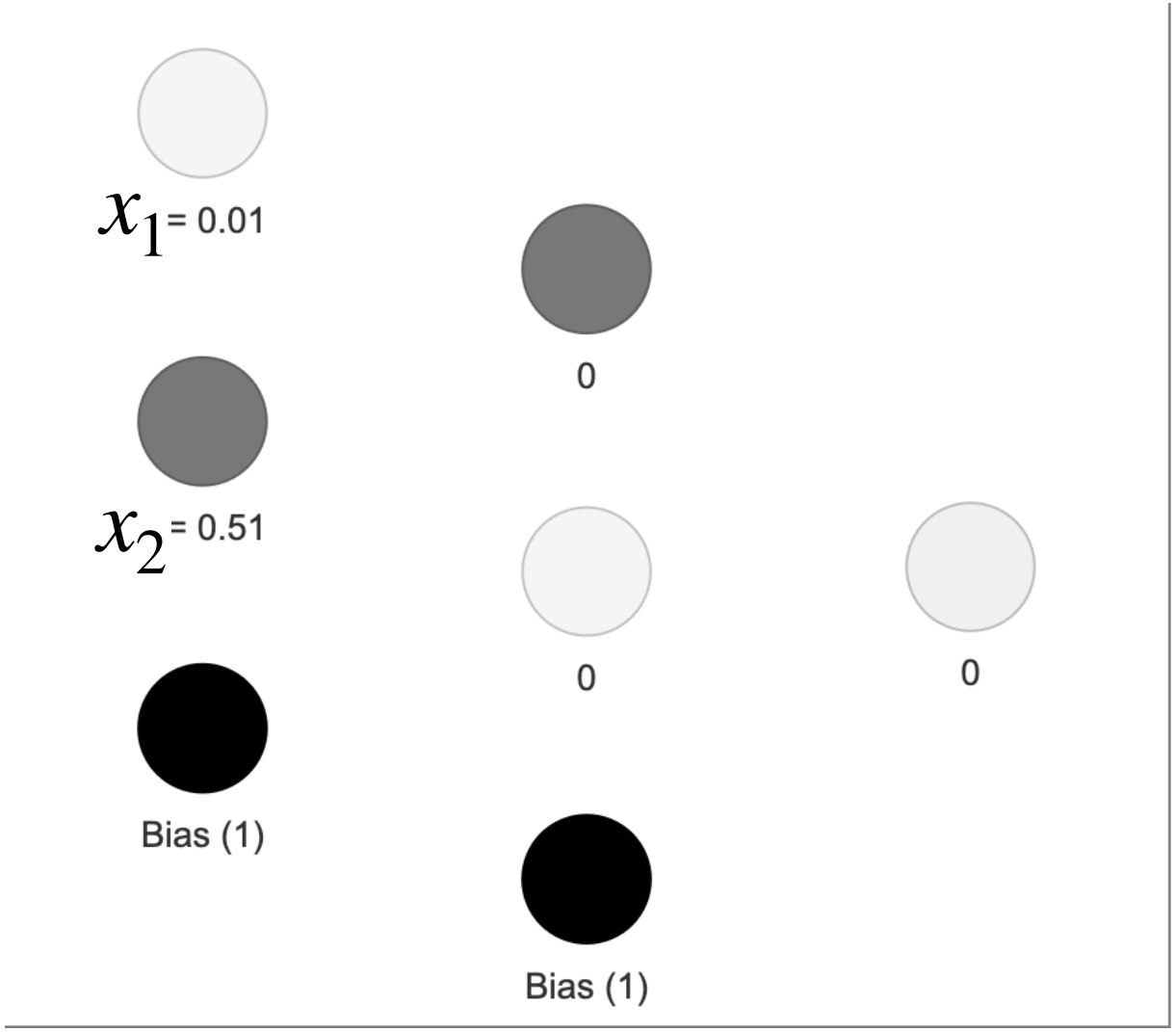

1
1

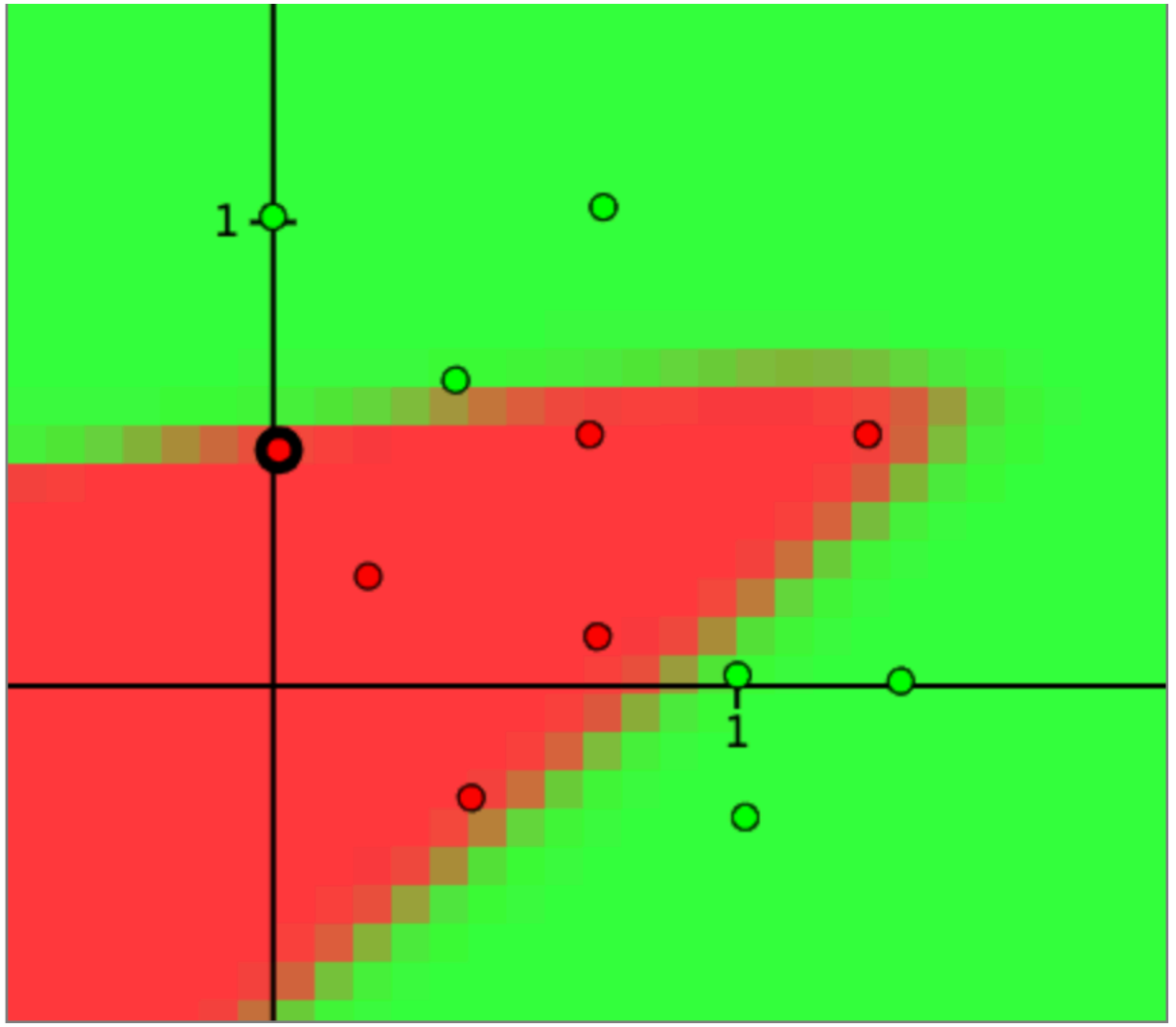

x_1 = 1.00
x_2 = 0.02
Bias (1)
1.00
σ(0.81) = 0.69
Bias (1)
0

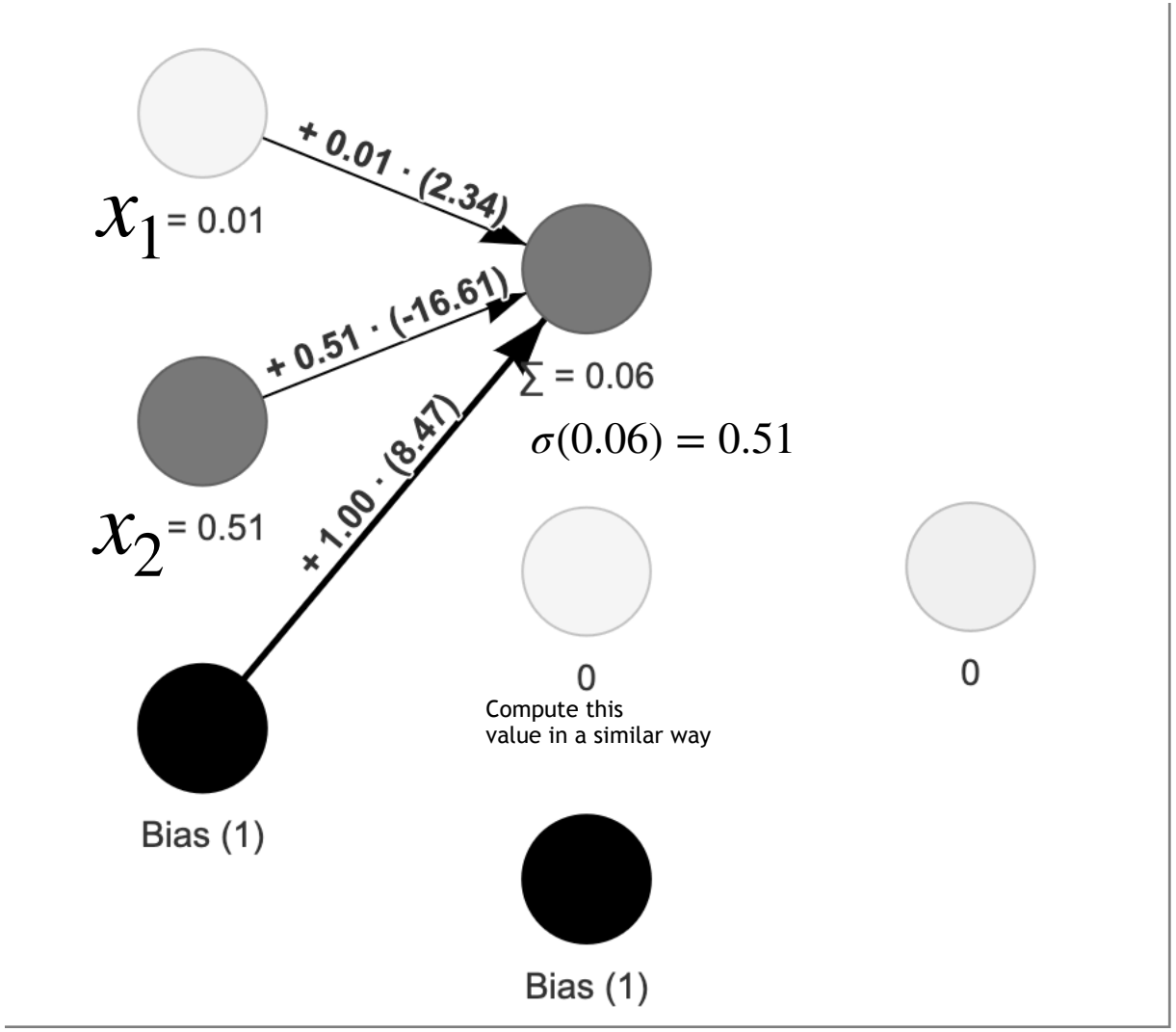

1
1

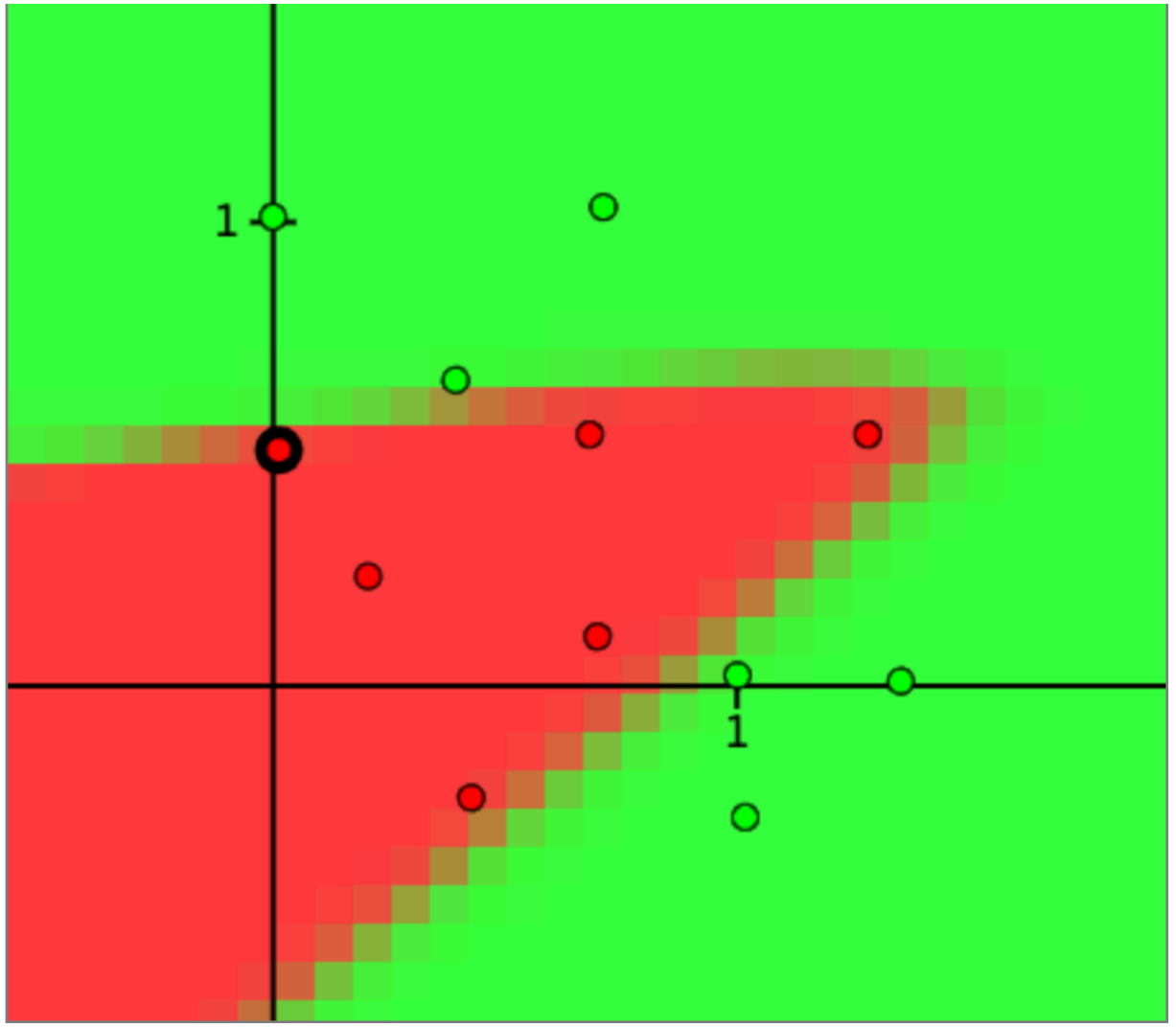

x_1 = 1.00
x_2 = 0.02
Bias (1)
1.00
0.69
Bias (1)
+ 1.00 · (-18.22)
+ 0.69 · (21.94)
+ 1.00 · (6.02)
∑ = 2.97
$\sigma(2.97) = 0.95$

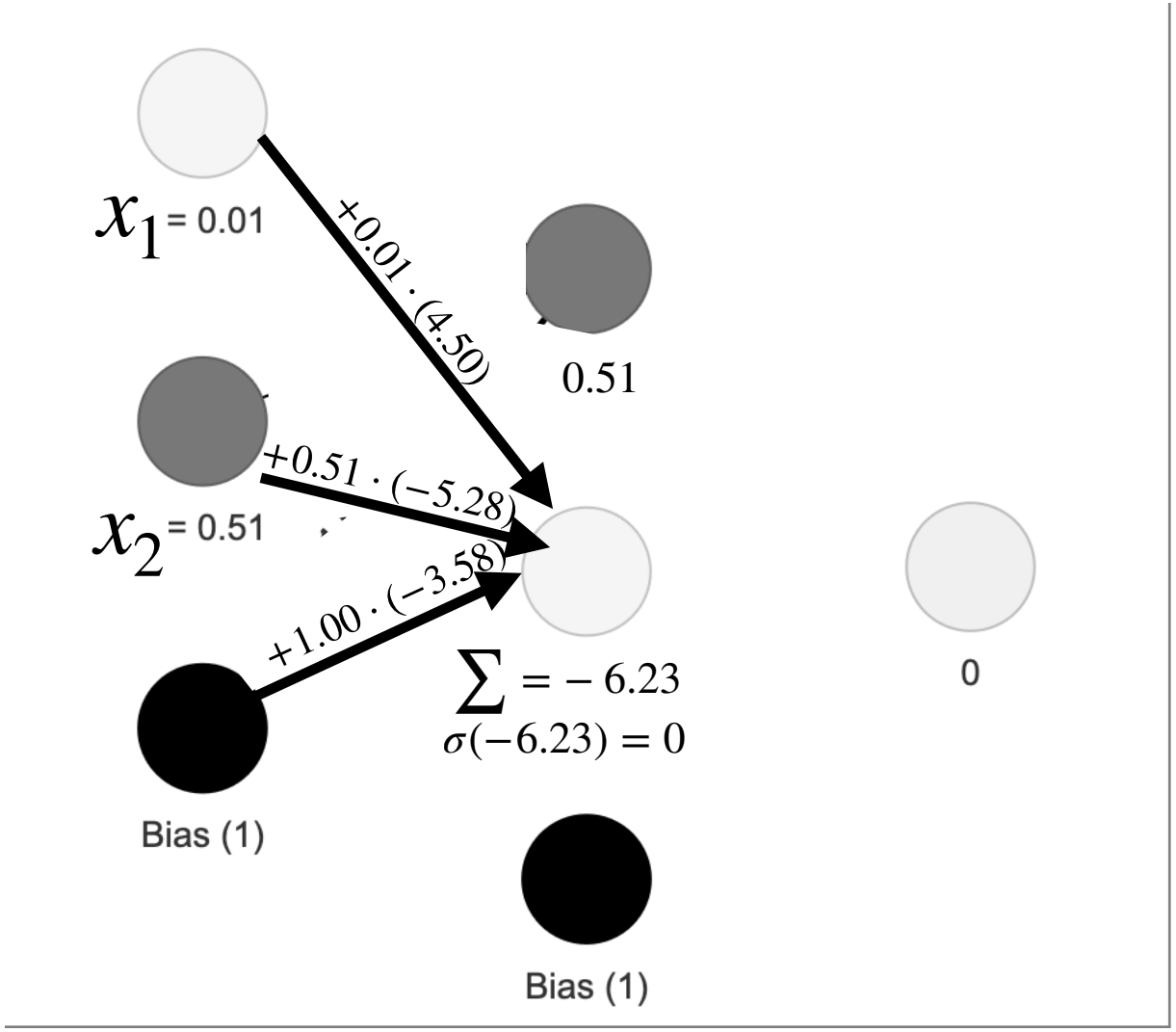

1
1

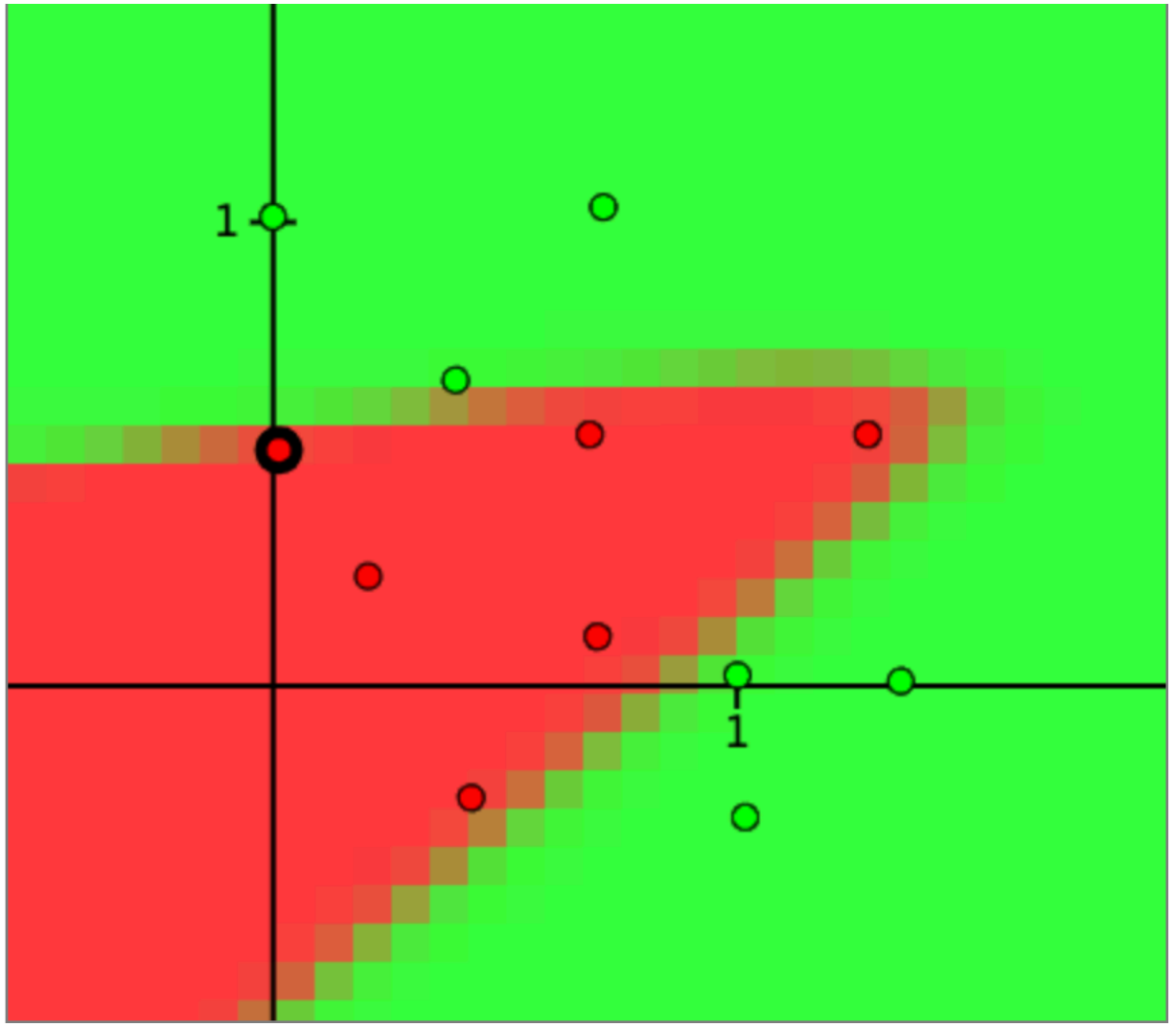

x_1 = 0.01
x_2 = 0.51
Bias (1)
0
0
0
Bias (1)

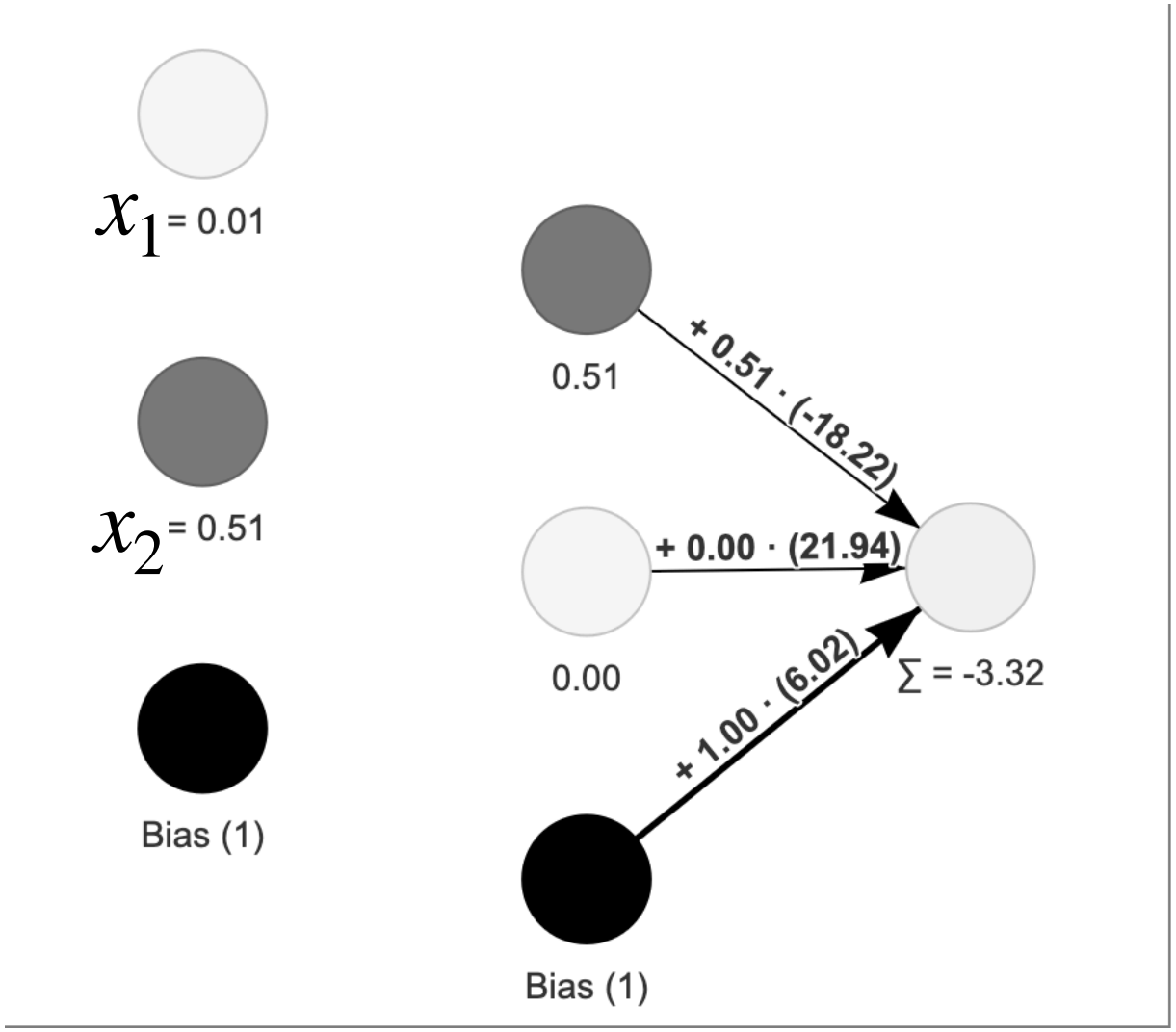

1
1

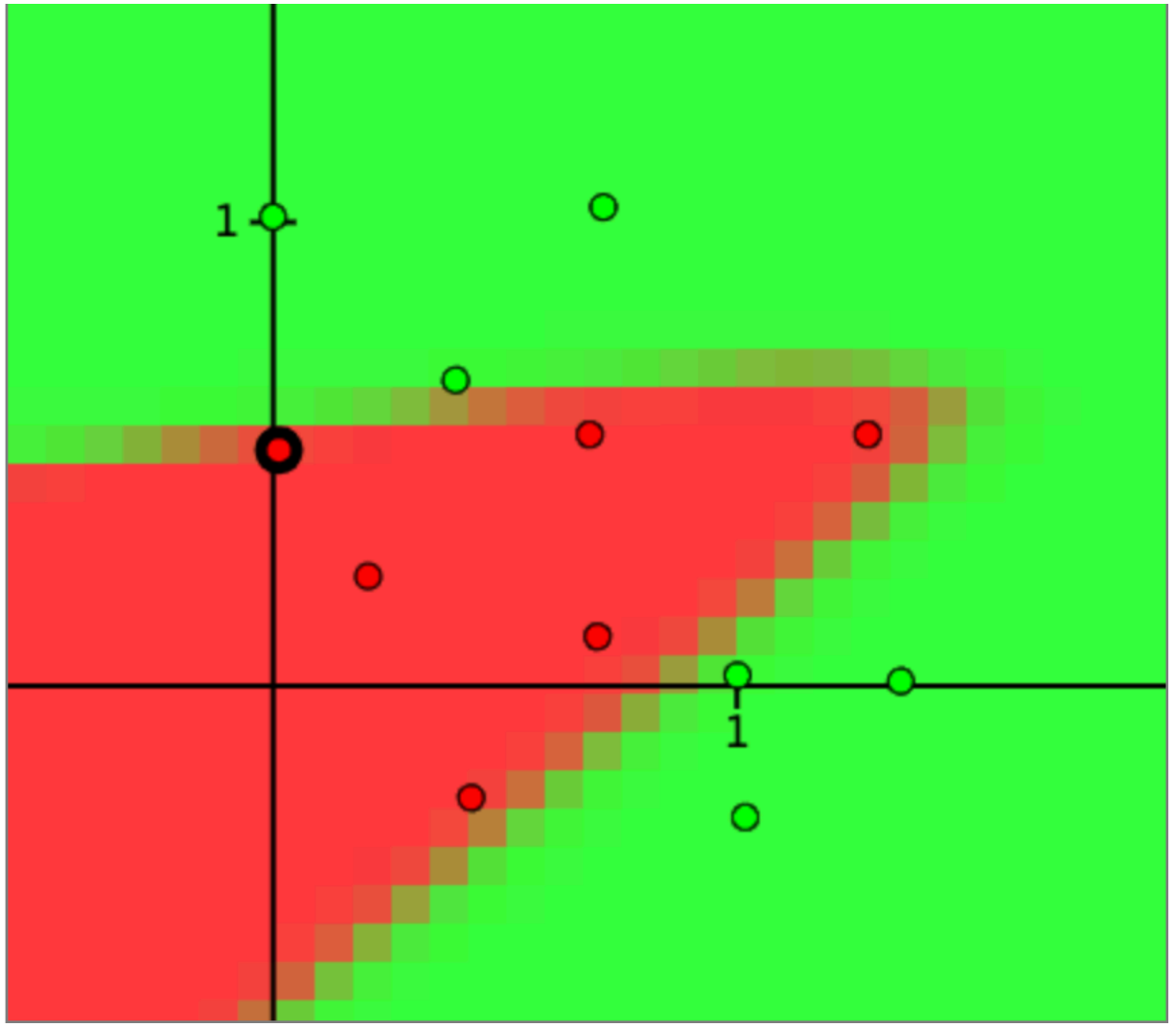

x_1 = 0.01
+ 0.01 · (2.34)
x_2 = 0.51
+ 0.51 · (-16.61)
+ 1.00 · (8.47)
Σ = 0.06
$\sigma(0.06) = 0.51$
0
Compute this value in a similar way
0
Bias (1)
Bias (1)

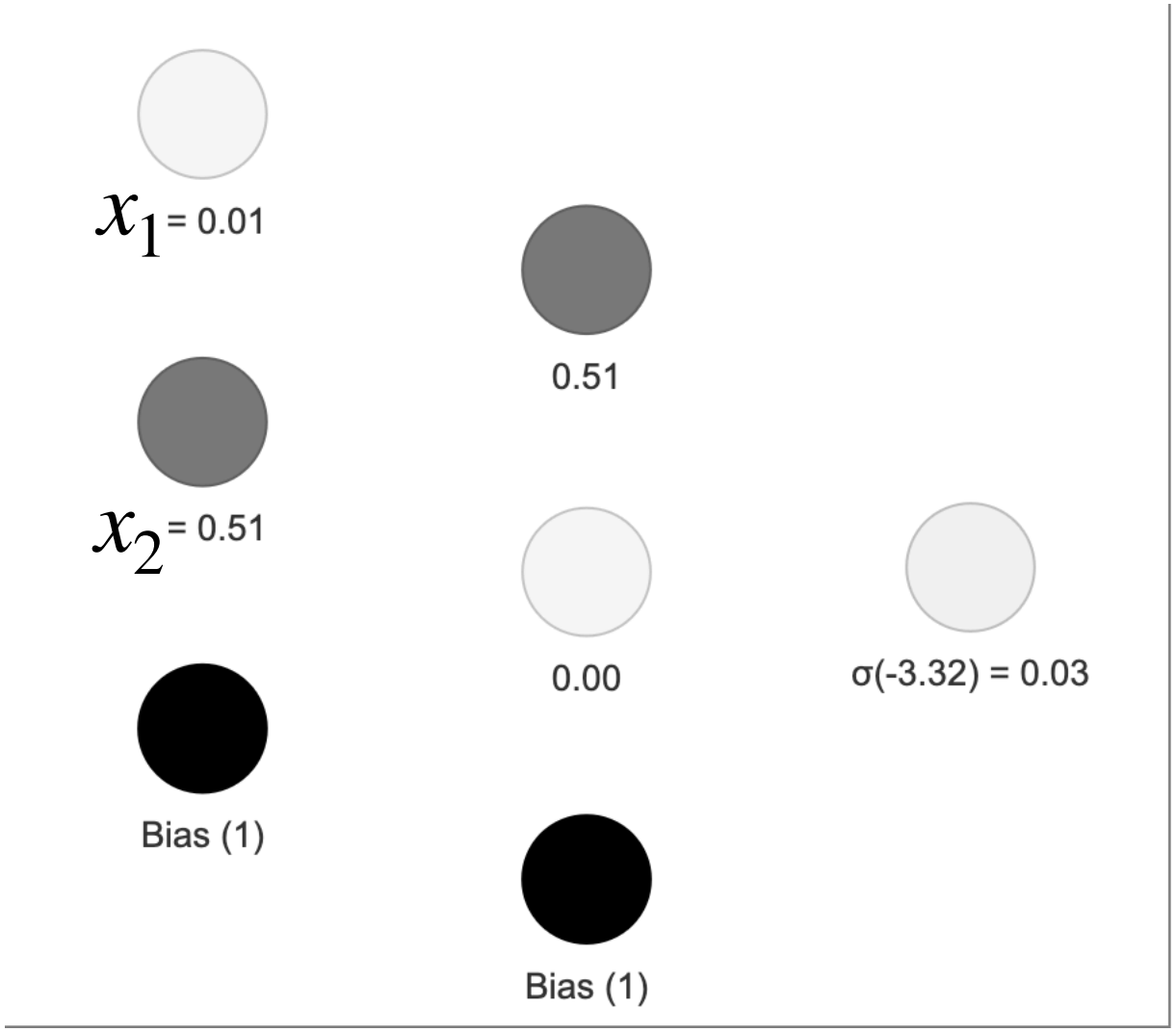

1
1

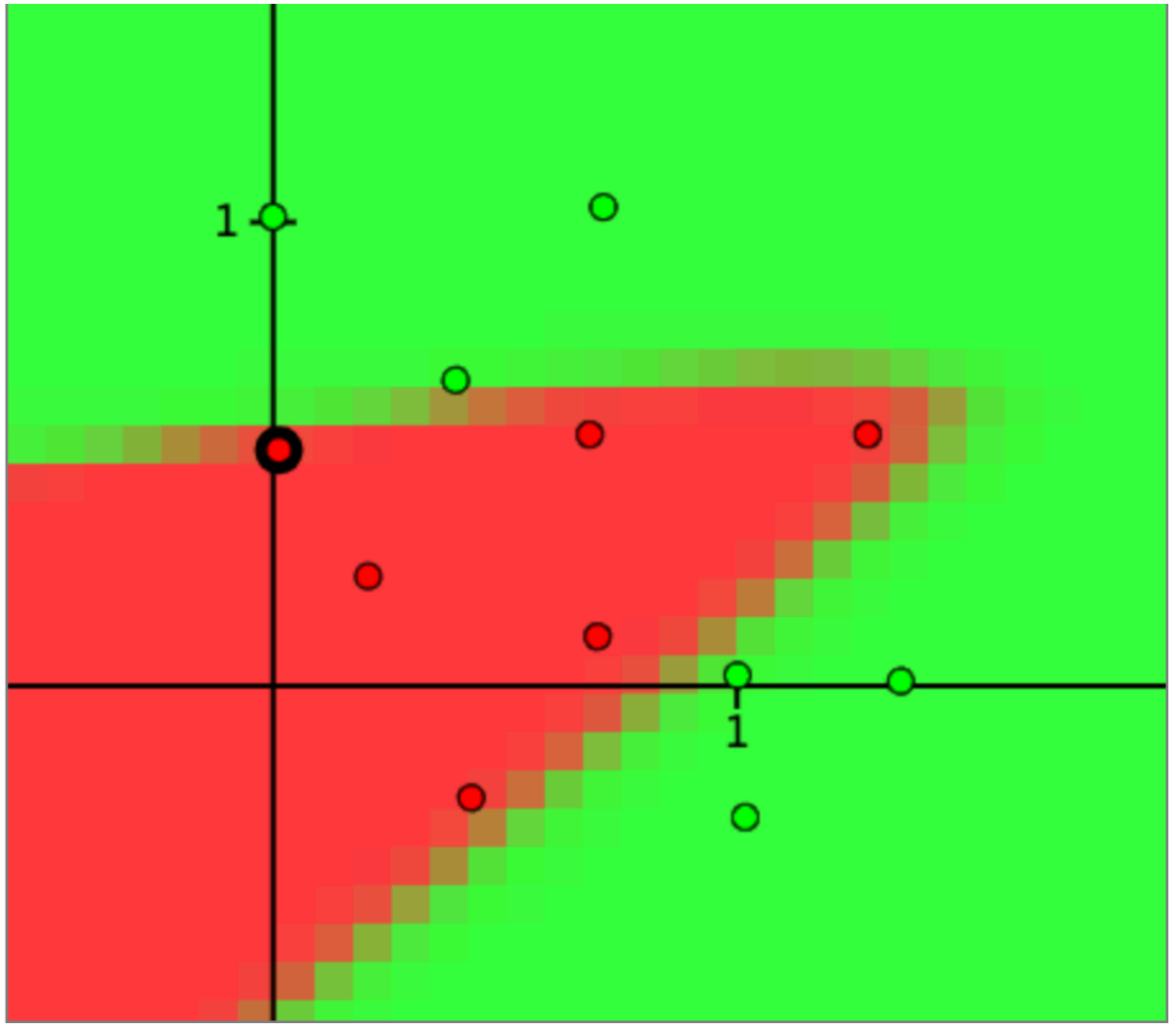

x_1 = 0.01
x_2 = 0.51
+0.01 · (4.50)
+0.51 · (−5.28)
+1.00 · (−3.58)
0.51
$\sum = -6.23$
$\sigma(-6.23) = 0$
0
Bias (1)
Bias (1)


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

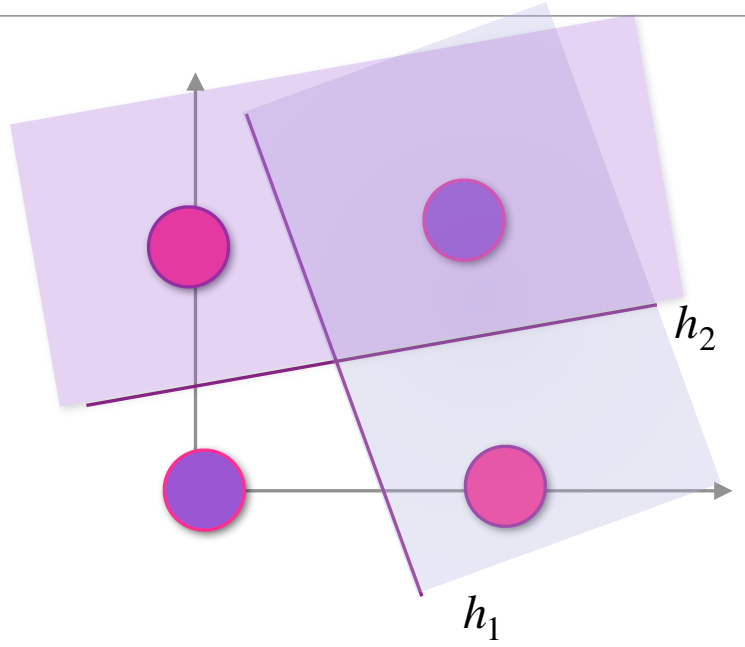

x_1 = 0.01
x_2 = 0.51
Bias (1)
0.51
0.00
Bias (1)
σ(-3.32) = 0.03


1
1

Pair share: can a neural network compute xor?

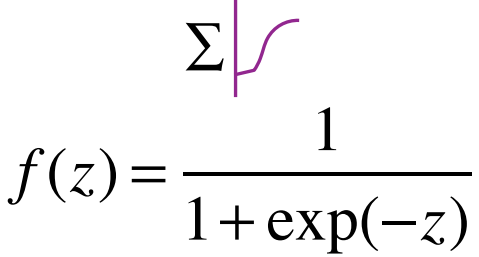
Boolean functions

x_1	x_2	Xor
0	0	0
0	1	1
1	0	1
1	1	0

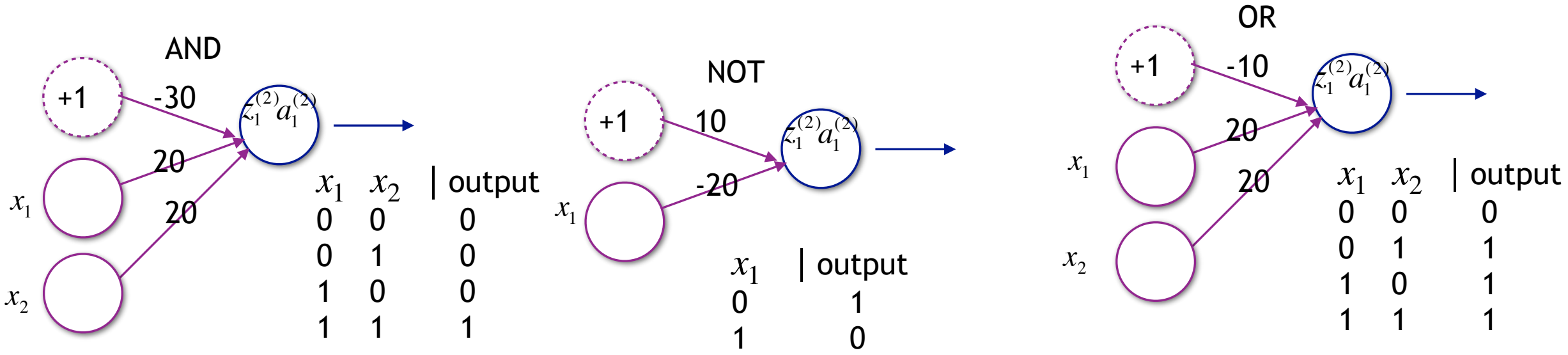


$$h(\mathbf{x}) = \begin{cases} 1 & (h_1(\mathbf{x}) = 1 \text{ and } h_2(x) = 0) \text{ or } (h_1(\mathbf{x}) = 0 \text{ and } h_2(x) = 1) \\ 0 & \text{otherwise} \end{cases}$$

Computing Boolean Functions



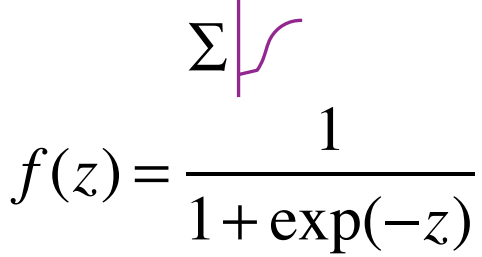
It is possible to build a neural network that computes any logical proposition:



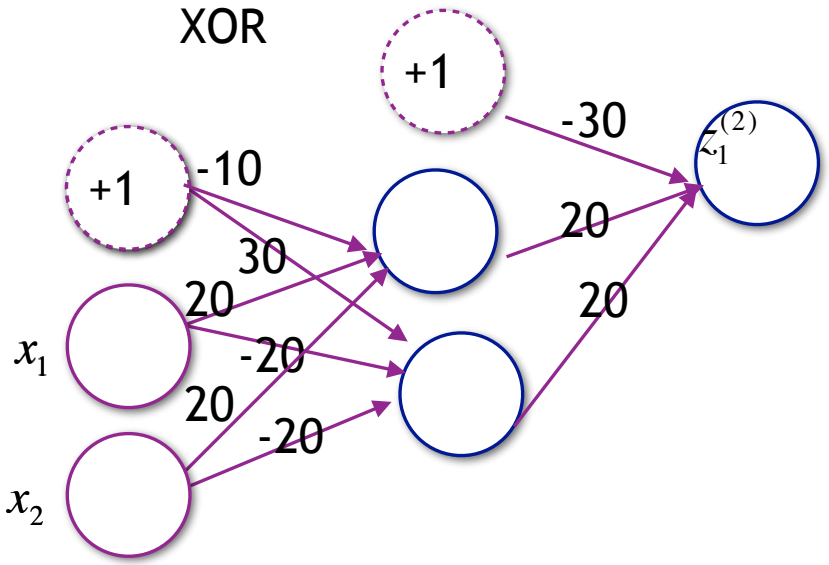
We can create a neural network to approximate XOR using the functions above and the definition:

$$h(\mathbf{x}) = \begin{cases} 1 & (h_1(\mathbf{x}) = 1 \text{ and not } h_2(x) = 1) \text{ or } (\text{not } h_1(\mathbf{x}) = 1 \text{ and } h_2(x) = 1) \\ 0 & \text{otherwise} \end{cases}$$

Computing Boolean Functions



We can approximate XOR using the following:



x_1	x_2	Xor
0	0	0
0	1	1
1	0	1
1	1	0

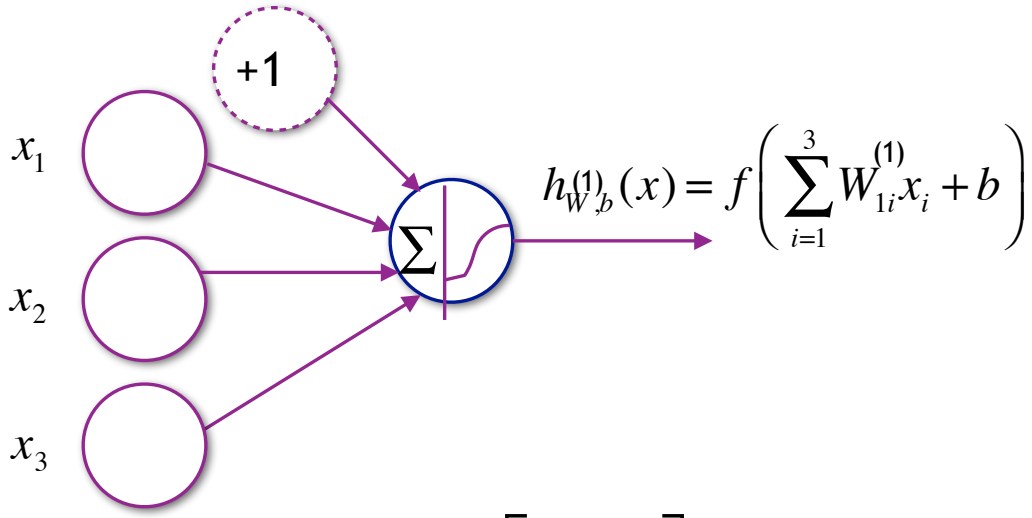
Every neural network represents an equation

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

New Notation

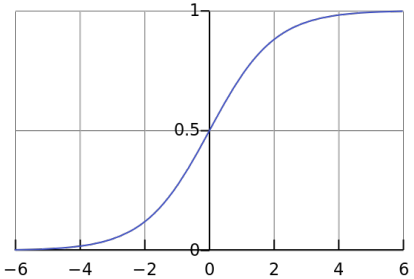
Unfortunately, there will be quite a bit of notation…..
We will follow the notation from http://deeplearning.stanford.edu/wiki/index.php/Neural_Networks



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

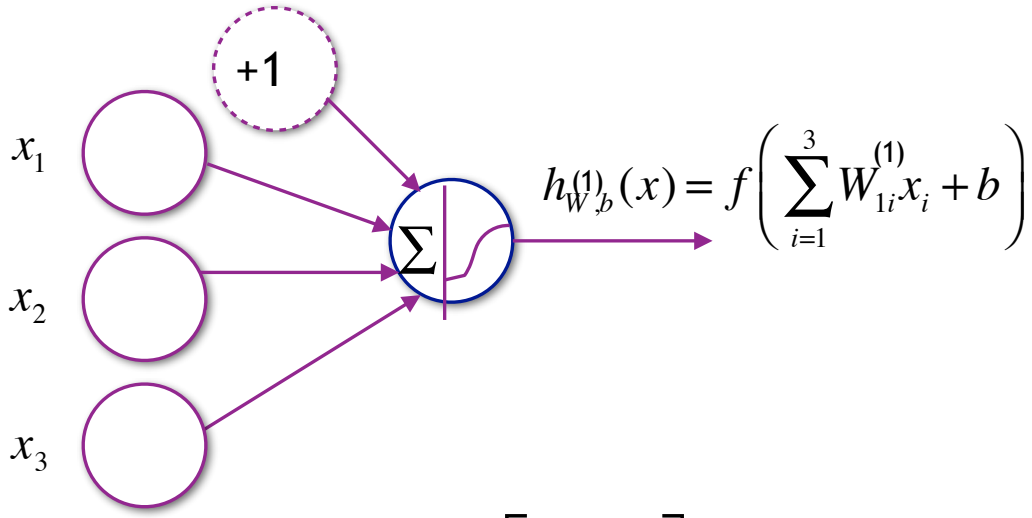
$$z = [W^{(1)}_{11}, W^{(1)}_{12}, W^{(1)}_{13}] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + b$$

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



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



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

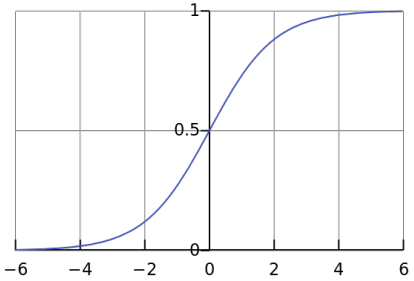
$$W^{T} = \begin{bmatrix} W_{11}^{(1)} \\ W_{12}^{(1)} \\ W_{13}^{(1)} \end{bmatrix} \mathbf{x} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

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

$$z = [W_{11}^{(1)}, W_{12}^{(1)}, W_{13}^{(1)}] \begin{bmatrix} x \\ \\ \end{bmatrix}$$

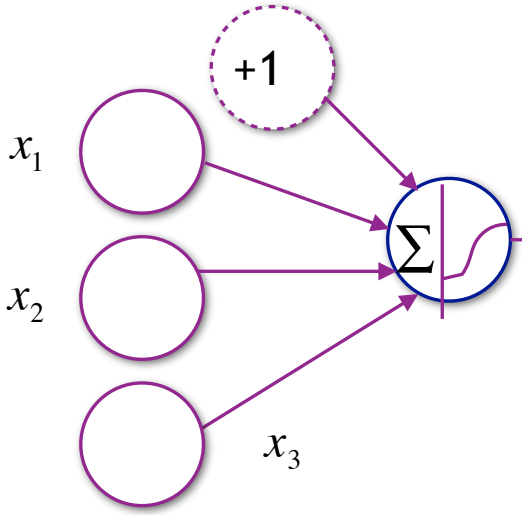
approximating single "neuron" using the sigmoid function

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



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

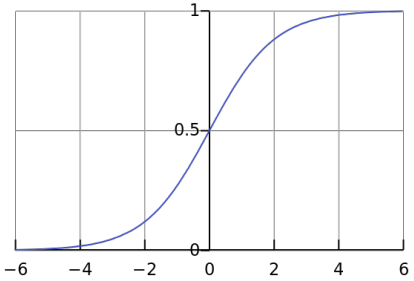
$$W = \begin{bmatrix} W_{11}^{(1)} \\ W_{12} \\ W_{13}^{(1)} \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

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

$$z = [W_{11}^{(1)}, W_{12}^{(1)}, W_{13}^{(1)}] \begin{bmatrix} x \\ \\ \end{bmatrix}$$

approximating single "neuron" using the sigmoid function

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

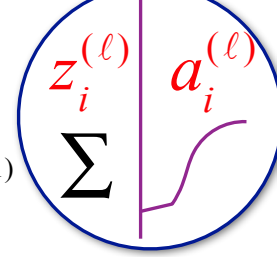


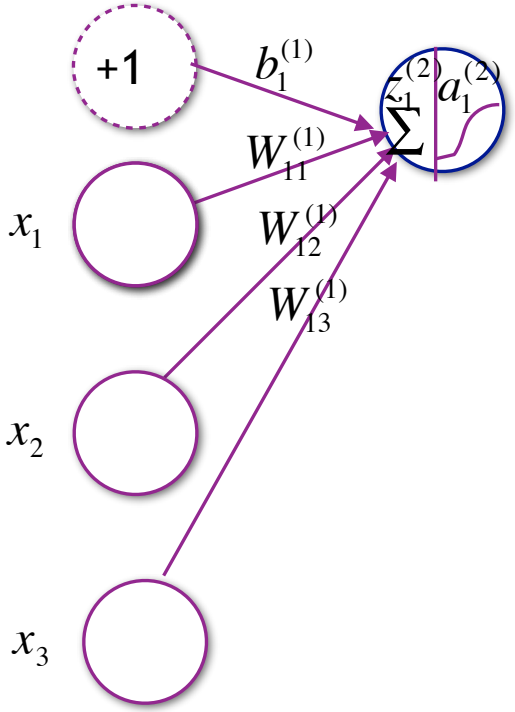
Neural Network Model & Notation

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

$$z_i^{(2)} = \sum_{j=1}^{3} W_{ij}^{(1)} x_j + b_i^{(1)}$$

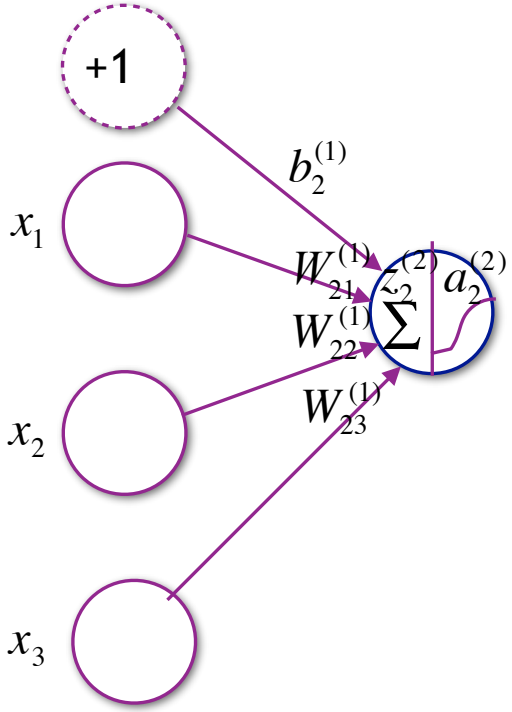
$$a_i^{(2)} = f(z_i^{(2)})$$





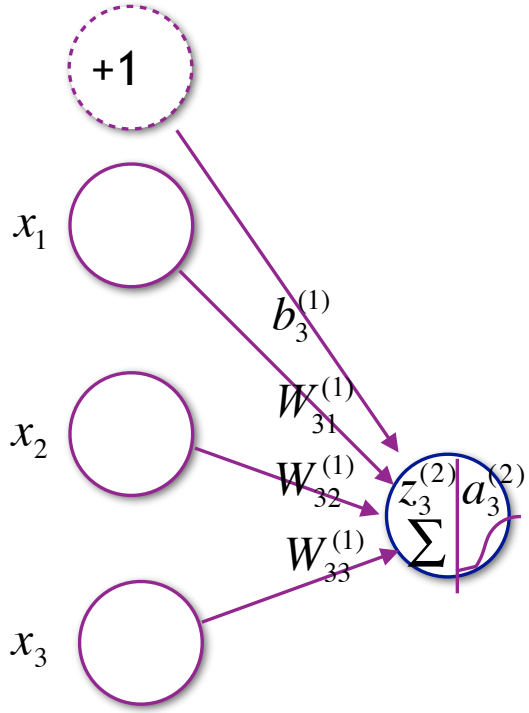
$$z_1^{(2)} = \sum_{j=1}^{3} W_{1j}^{(1)} x_j + b_1^{(1)}$$

$$a_1^{(2)} = f(z_1^{(2)})$$



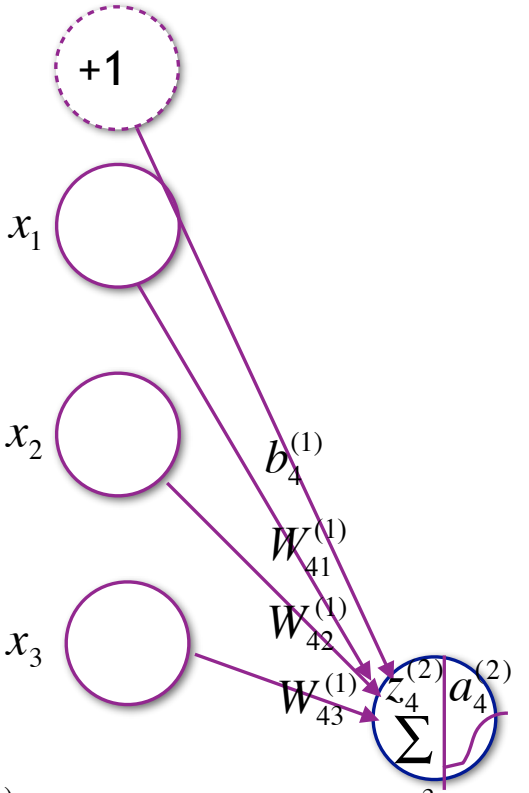
$$z_2^{(2)} = \sum_{j=1}^{3} W_{2j}^{(1)} x_j + b_2^{(1)}$$

$$a_2^{(2)} = f(z_2^{(2)})$$



$$z_3^{(2)} = \sum_{j=1}^{3} W_{3j}^{(1)} x_j + b_3^{(1)}$$

$$a_3^{(2)} = f(z_3^{(2)})$$



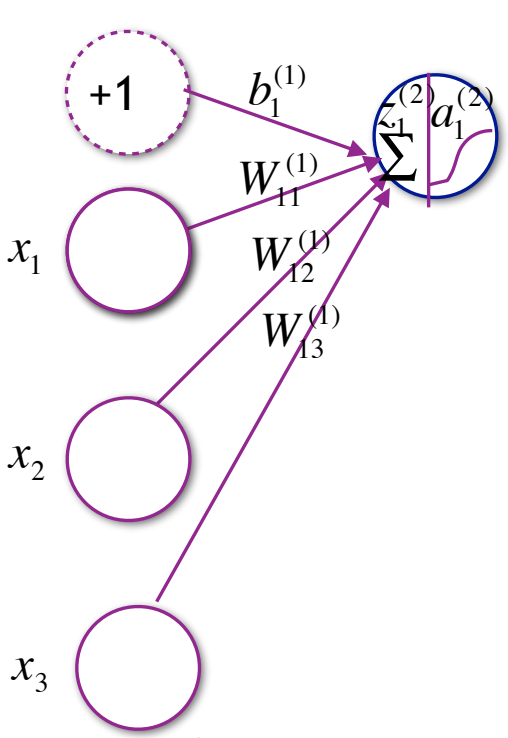
$$z_4^{(2)} = \sum_{j=1}^{3} W_{4j}^{(1)} x_j + b_4^{(1)}$$

$$a_4^{(2)} = f(z_4^{(2)})$$

Neural Network Model & Notation

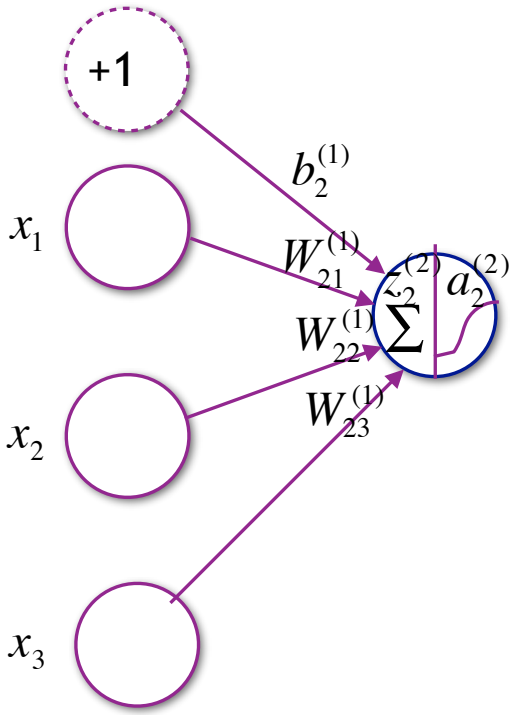
$$z_i^{(2)} = \sum_{j=1}^{3} W_j^{(1)} x_j + b_i^{(1)} \qquad a_i^{(2)} = f(z_i^{(2)})$$

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



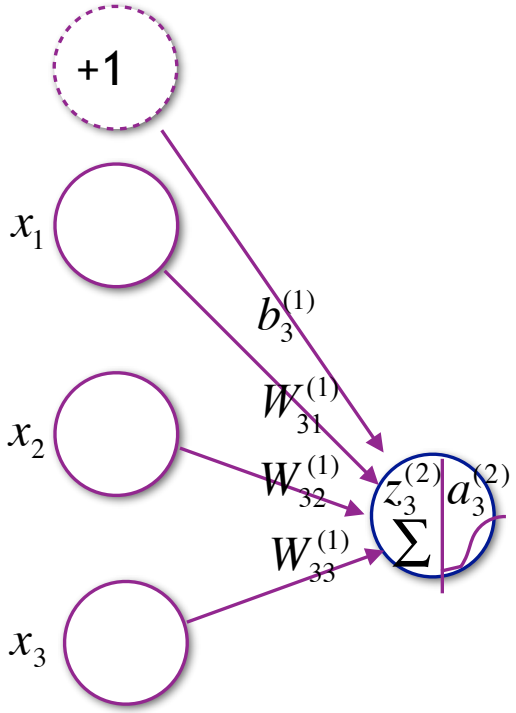
$$z_1^{(2)} = \sum_{j=1}^{3} W_{1j}^{(1)} x_j + b_1^{(1)}$$

$$a_1^{(2)} = f(z_1^{(2)})$$



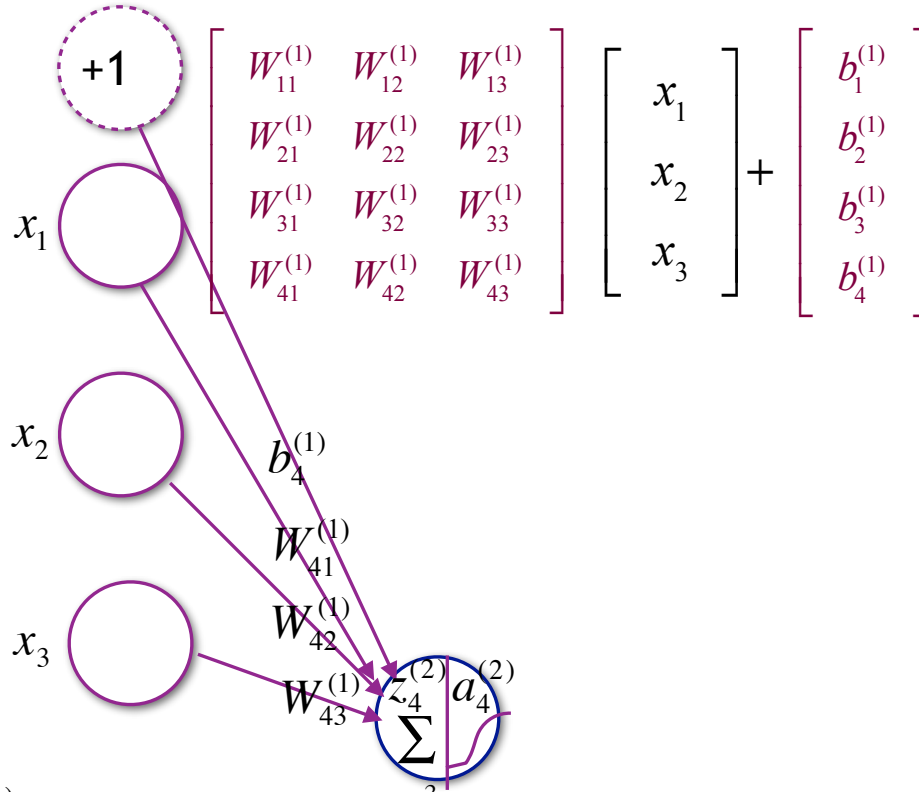
$$z_2^{(2)} = \sum_{j=1}^{3} W_{2j}^{(1)} x_j + b_2^{(1)}$$

$$a_2^{(2)} = f(z_2^{(2)})$$



$$z_3^{(2)} = \sum_{j=1}^{3} W_{3j}^{(1)} x_j + b_3^{(1)}$$

$$a_3^{(2)} = f(z_3^{(2)})$$



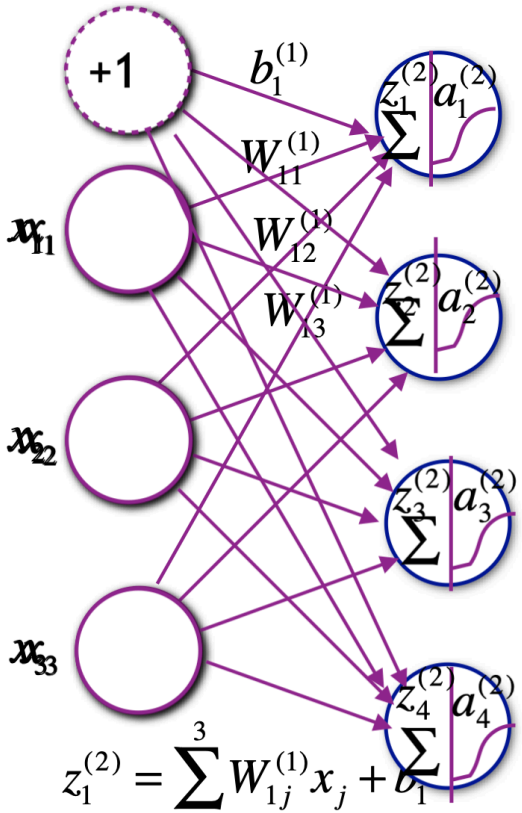
$$z_4^{(2)} = \sum_{j=1}^{3} W_{4j}^{(1)} x_j + b_4^{(1)}$$

$$a_4^{(2)} = f(z_4^{(2)})$$

Neural Network Model & Notation

$$z_i^{(2)} = \sum_{j=1}^{3} W_j^{(1)} x_j + b_i^{(1)} \qquad a_i^{(2)} = f(z_i^{(2)})$$

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



$$\begin{bmatrix} W_{11}^{(1)} & W_{12}^{(1)} & W_{13}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} & W_{23}^{(1)} \\ W_{31}^{(1)} & W_{32}^{(1)} & W_{33}^{(1)} \\ W_{41}^{(1)} & W_{42}^{(1)} & W_{43}^{(1)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \\ b_3^{(1)} \\ b_4^{(1)} \end{bmatrix}$$

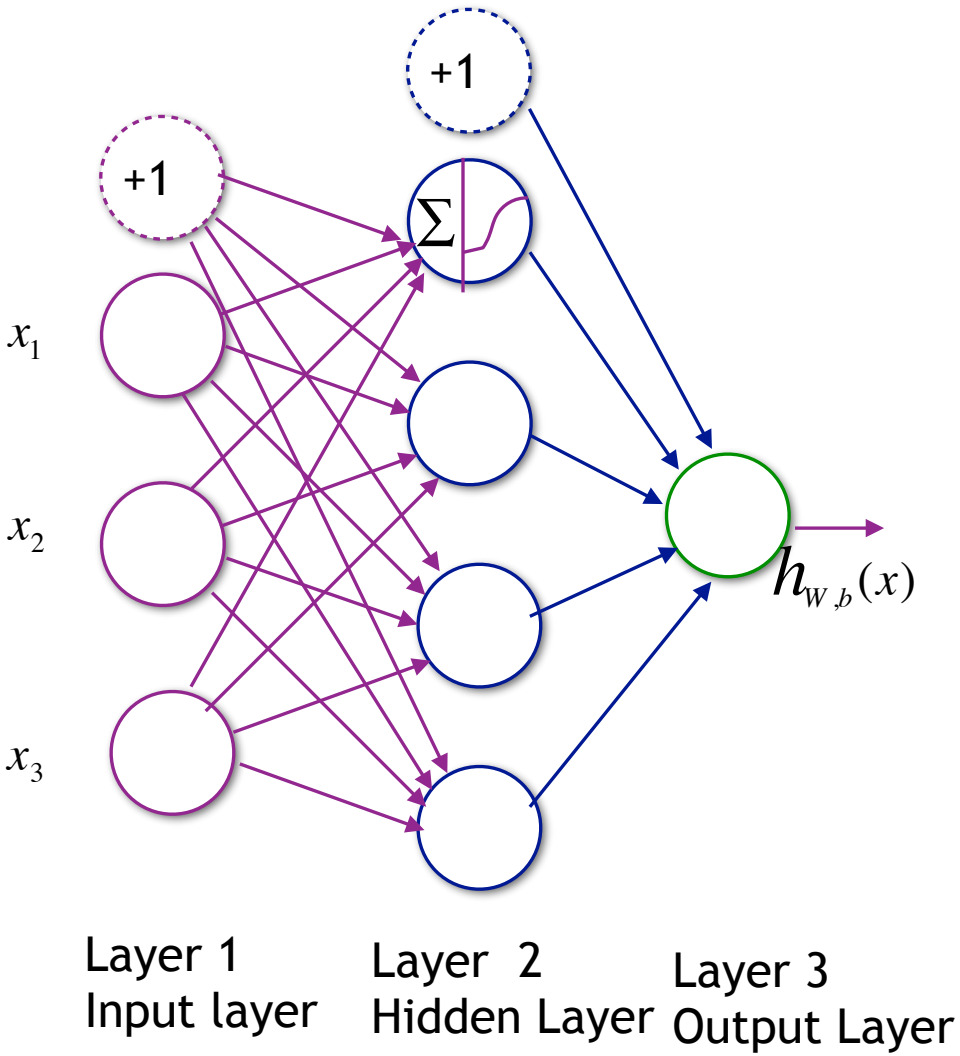
$$z_1^{(2)} = \sum_{j=1}^{3} W_{1j}^{(1)} x_j + b_1^{(1)} \qquad a_1^{(2)} = f(z_1^{(2)})$$

$$z_2^{(2)} = \sum_{j=1}^{3} W_{2j}^{(1)} x_j + b_2^{(1)} \qquad a_2^{(2)} = f(z_2^{(2)})$$

$$z_3^{(2)} = \sum_{j=1}^{3} W_{3j}^{(1)} x_j + b_3^{(1)} \qquad a_3^{(2)} = f(z_3^{(2)})$$

$$z_4^{(2)} = \sum_{j=1}^{3} W_{4j}^{(1)} x_j + b_4^{(1)} \qquad a_4^{(2)} = f(z_4^{(2)})$$

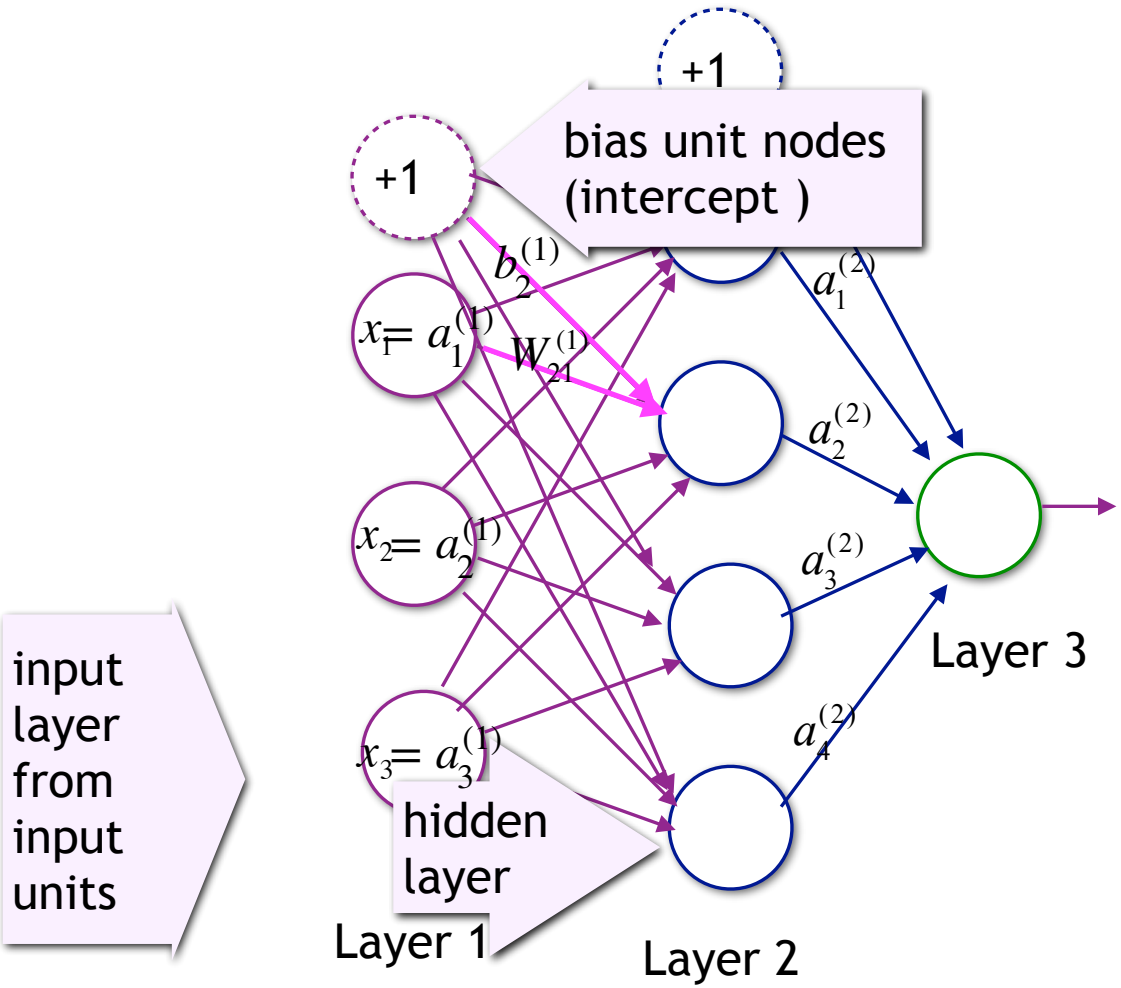
Neural Network Model & Notation



- We can combine many of the simple "neurons" so the output of one is the input of another neuron
- Input layer units are set by **x**
- The input for all other nodes are typically the *weighted* sum of the **activation** on the links connected to this node
- The **activation** function is applied to the input to provide the value (we will use the sigmoid function)

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

Neural Network Model & Notation



$$W_{ij}^{(\ell)} \begin{cases} 1 \le \ell < n_l & \text{layers} \\ 1 \le j \le s_\ell & \text{inputs} \\ 1 \le i \le s_{\ell+1} & \text{outputs} \end{cases}$$

- The network has 3 input units (not 4), 4 hidden units (not 5) and 1 output unit
- n_ℓ is the number of layers in the network
- Each "neuron" has it own weights. We will store all the weights for one level together in a matrix $W^{(\ell)}$ such that $W_{ij}^{(1)}$ is the weight for leaving node j at level 1 and entering node i at level 2
- $b_i^{(1)}$ is the bias of unit i in level 2

$$W^{(1)} \in \mathbb{R}^{4\times 3} \qquad W^{(2)} \in \mathbb{R}^{1\times 4}$$

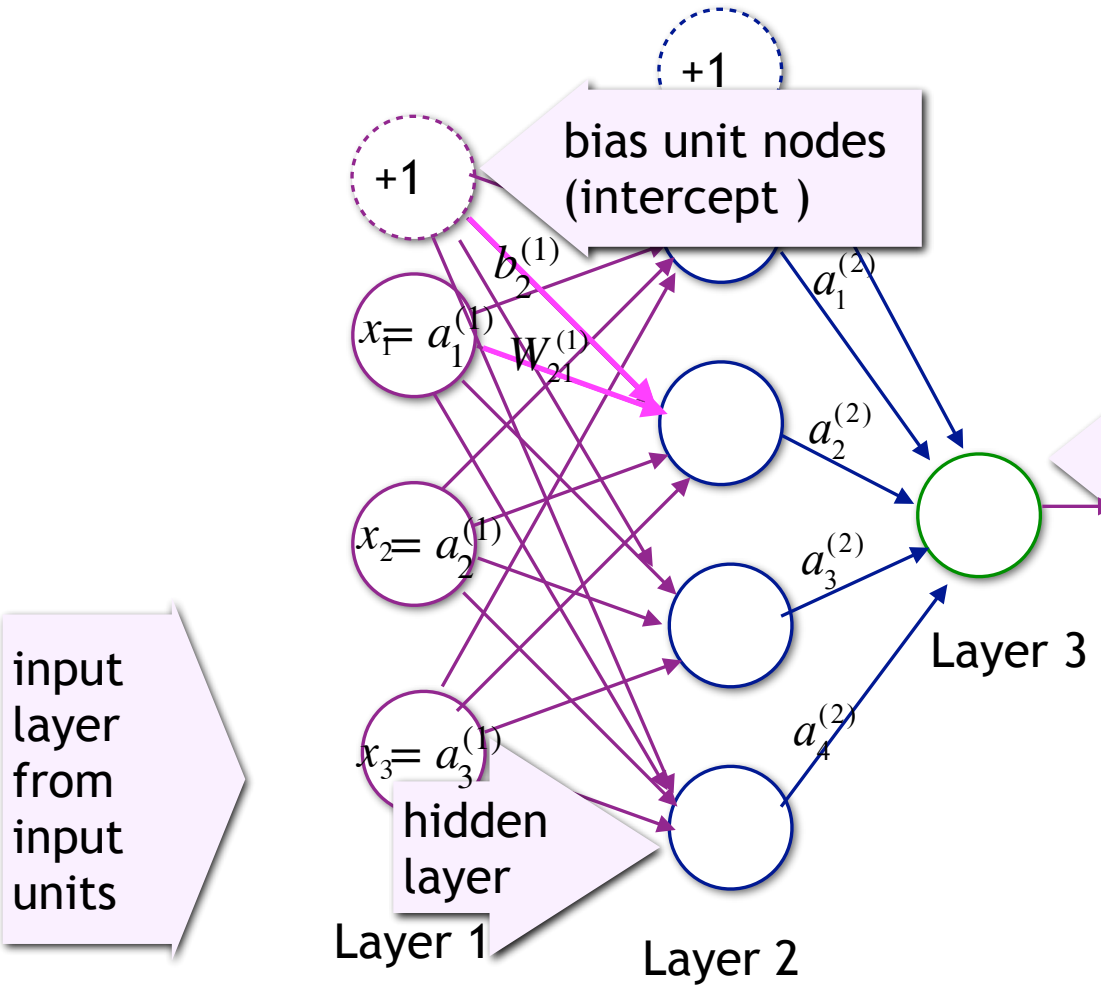
Neural Network Model & Notation

$W_{ij}^{(\ell)}$

$1 \le \ell < n_l$ layers

$1 \le j \le s_\ell$ inputs

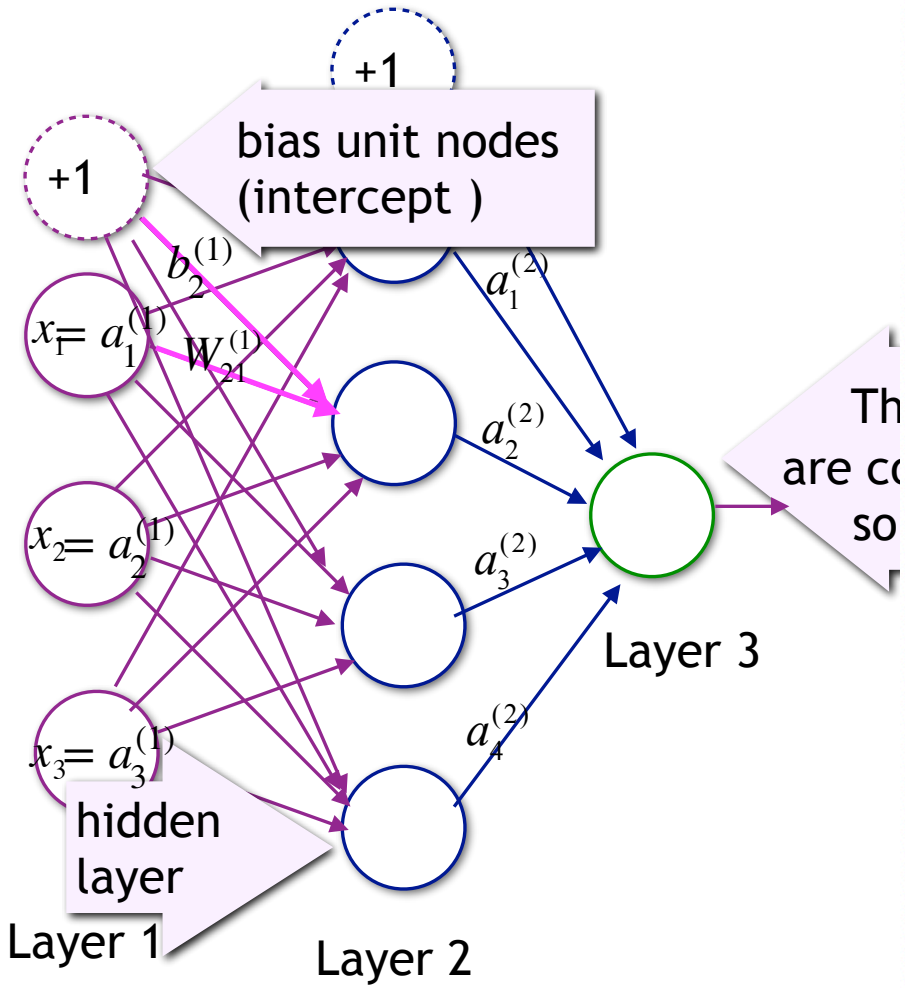
$1 \le i \le s_{\ell+1}$ outputs



- The network has 3 input units (not 4), 4 hidden units (not 5) and 1 output unit
- …n the network
- …eights. We will store all the weights for one level together in a matrix $W^{(\ell)}$ such that $W_{ij}^{(1)}$ is the weight for leaving node j at level 1 and entering node i at level 2
- $b_i^{(1)}$ is the bias of unit i in level 2

$W^{(1)} \in \mathbb{R}^{4\times 3}$ $W^{(2)} \in \mathbb{R}^{1\times 4}$

Neural Network Model & Notation



1. How many parameters to train in this neural network?

a) 0-3

b) 4-6

c) 7-10

d) 11-14

e) 15-18

f) 19-22

g) 23-26

h) more than 26

layers
inputs
outputs

4 hidden

ork

e will
her in a
t for
ode i at

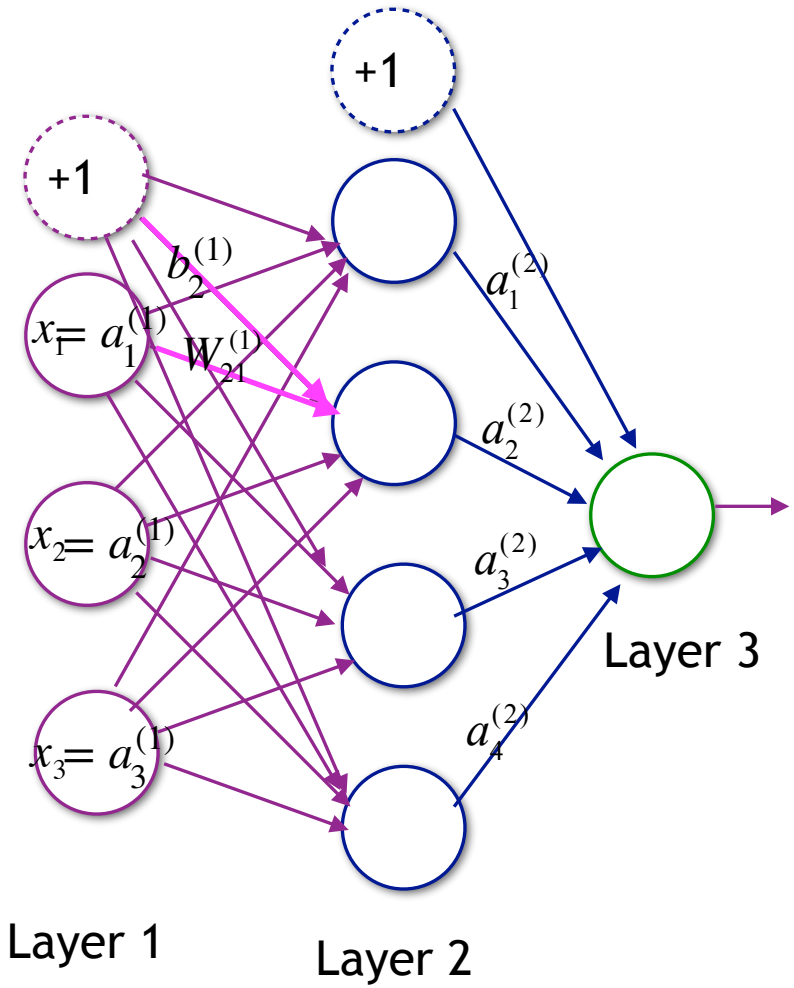
Neural Network Model & Notation

$W_{ij}^{(\ell)}$

$1 \le \ell < n_l$ layers

$1 \le j \le s_\ell$ inputs

$1 \le i \le s_{\ell+1}$ outputs



input layer from input units

- The network has 3 input units (not 4), 4 hidden units (not 5) and 1 output unit
- n_ℓ is the number of layers in the network
- Each "neuron" has it own weights. We will store all the weights for one level together in a matrix $W^{(1)}$ such that $W_{ij}^{(1)}$ is the weight for leaving node j at level 1 and entering node i at level 2
- $b_i^{(1)}$ is the bias of unit i in level 2

$W^{(1)} \in \mathbb{R}^{4\times 3}$ $\quad$ $W^{(2)} \in \mathbb{R}^{1\times 4}$

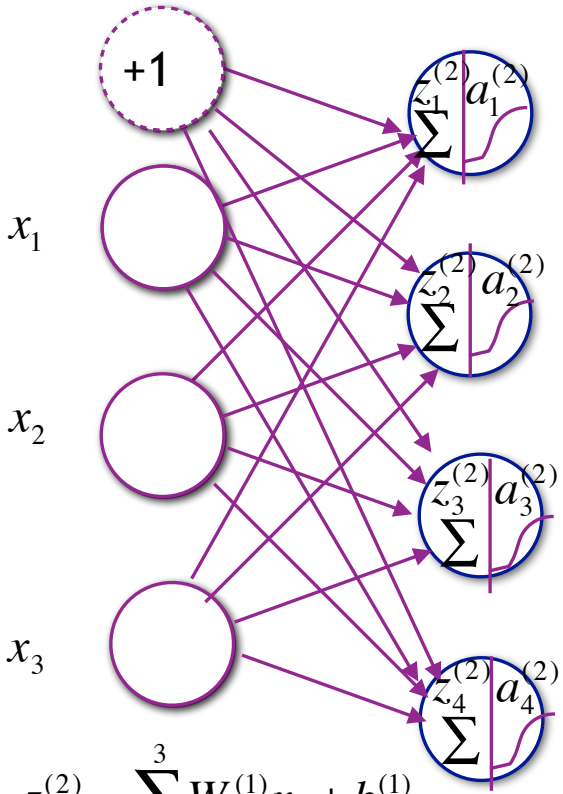
Let s_j be the number of nodes at layer j where we don't include the bias unit

Neural Network Model & Notation

$$z_i^{(2)} = \sum_{j=1}^{3} W_{ij}^{(1)} x_j + b_i^{(1)} \qquad a_i^{(2)} = f(z_i^{(2)})$$

$$z^{(2)} = W^{(1)} x + b^{(1)}$$
$$a^{(2)} = f(z^{(2)}) = f(W^{(1)} x + b^{(1)})$$

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



$$\begin{bmatrix} W_{11}^{(1)} & W_{12}^{(1)} & W_{13}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} & W_{23}^{(1)} \\ W_{31}^{(1)} & W_{32}^{(1)} & W_{33}^{(1)} \\ W_{41}^{(1)} & W_{42}^{(1)} & W_{43}^{(1)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \\ b_3^{(1)} \\ b_4^{(1)} \end{bmatrix} = \begin{bmatrix} W_{11}^{(1)} x_1 + W_{12}^{(1)} x_2 + W_{13}^{(1)} x_3 \\ W_{21}^{(1)} x_1 + W_{22}^{(1)} x_2 + W_{23}^{(1)} x_3 \\ W_{31}^{(1)} x_1 + W_{32}^{(1)} x_2 + W_{33}^{(1)} x_3 \\ W_{41}^{(1)} x_1 + W_{42}^{(1)} x_2 + W_{43}^{(1)} x_3 \end{bmatrix} + \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \\ b_3^{(1)} \\ b_4^{(1)} \end{bmatrix}$$

$$= \begin{bmatrix} W_{11}^{(1)} x_1 + W_{12}^{(1)} x_2 + W_{13}^{(1)} x_3 + b_1^{(1)} \\ W_{21}^{(1)} x_1 + W_{22}^{(1)} x_2 + W_{23}^{(1)} x_3 + b_2^{(1)} \\ W_{31}^{(1)} x_1 + W_{32}^{(1)} x_2 + W_{33}^{(1)} x_3 + b_3^{(1)} \\ W_{41}^{(1)} x_1 + W_{42}^{(1)} x_2 + W_{43}^{(1)} x_3 + b_4^{(1)} \end{bmatrix} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \\ z_4^{(2)} \end{bmatrix}$$

$$a^{(2)} = \begin{bmatrix} f(z_1^{(2)}) \\ f(z_2^{(2)}) \\ f(z_3^{(2)}) \\ f(z_4^{(2)}) \end{bmatrix} = \begin{bmatrix} a_1^{(2)} \\ a_2^{(2)} \\ a_3^{(2)} \\ a_4^{(2)} \end{bmatrix}$$

$$z_1^{(2)} = \sum_{j=1}^{3} W_{1j}^{(1)} x_j + b_1^{(1)} \qquad a_1^{(2)} = f(z_1^{(2)})$$

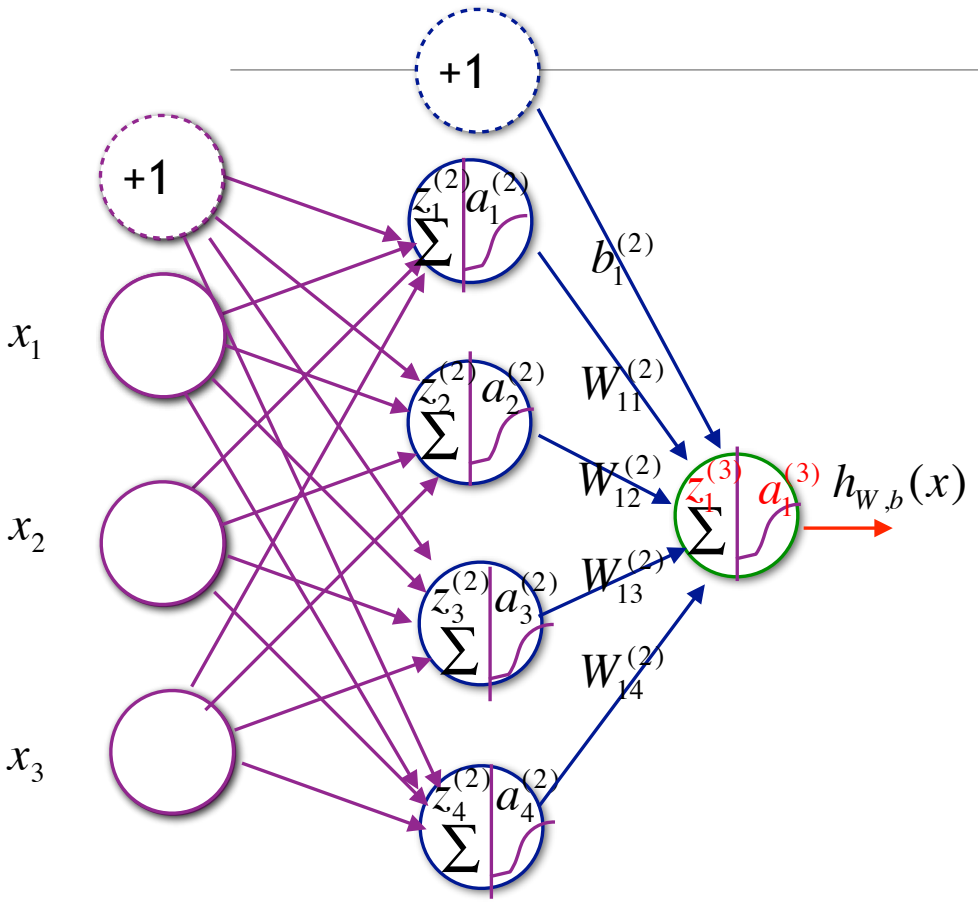
$$z_2^{(2)} = \sum_{j=1}^{3} W_{2j}^{(1)} x_j + b_2^{(1)} \qquad a_2^{(2)} = f(z_2^{(2)})$$

$$z_3^{(2)} = \sum_{j=1}^{3} W_{3j}^{(1)} x_j + b_3^{(1)} \qquad a_3^{(2)} = f(z_3^{(2)})$$

$$z_4^{(2)} = \sum_{j=1}^{3} W_{4j}^{(1)} x_j + b_4^{(1)} \qquad a_4^{(2)} = f(z_4^{(2)})$$

Neural Network Model & Notation

$$z^{(l+1)} = W^{(l)} a^{(l)} + b^{(l)} \qquad a^{(l+1)} = f(z^{(l+1)}) = f(W^{(l)} a^{(l)} + b^{(l)}) \qquad f(z) = \frac{1}{1+\exp(-z)}$$



$$\begin{bmatrix} W_{11}^{(2)} & W_{12}^{(2)} & W_{13}^{(2)} & W_{14}^{(2)} \end{bmatrix} \begin{bmatrix} a_1^{(2)} \\ a_2^{(2)} \\ a_3^{(2)} \\ a_4^{(2)} \end{bmatrix} + \begin{bmatrix} b_1^{(2)} \end{bmatrix} = \begin{bmatrix} z_1^{(3)} \end{bmatrix}$$

$$\mathbf{a}^{(3)} = \begin{bmatrix} f(z_1^{(2)}) \end{bmatrix} = \begin{bmatrix} a_1^{(3)} \end{bmatrix}$$

$$\mathbf{z}^{(3)} = W^{(2)} a^{(2)} + b^{(2)}$$

$$\mathbf{a}^{(3)} = f(z^{(3)}) = f(W^{(2)} x + b^{(2)})$$

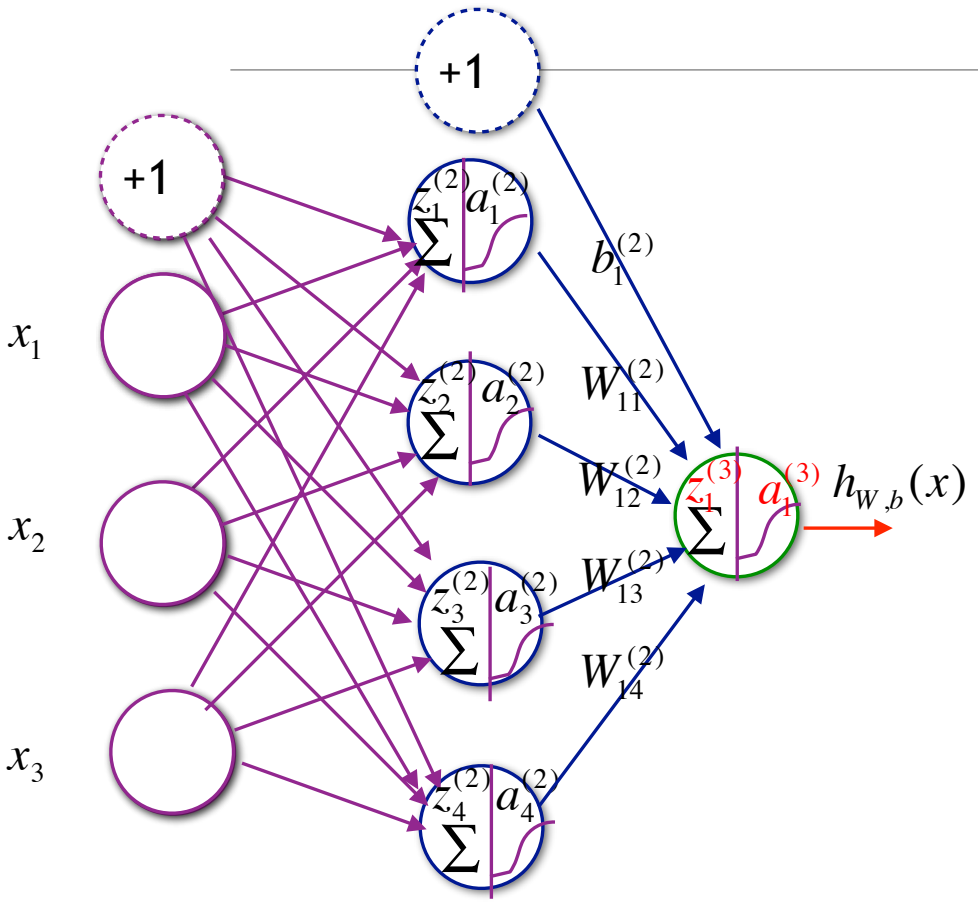
$$z_1^{(3)} = \sum_{j=1}^{4} W_{1j}^{(2)} a_j^{(2)} + b_1^{(2)}$$

$$a_1^{(3)} = f(z_1^{(3)})$$

$$h_{W,\mathbf{b}} = \mathbf{a}^{(3)} = f(\mathbf{z}^{(3)})$$

Neural Network Model & Notation

$$z^{(l+1)} = W^{(l)} a^{(l)} + b^{(l)} \qquad a^{(l+1)} = f(z^{(l+1)}) = f(W^{(l)} a^{(l)} + b^{(l)}) \qquad f(z) = \frac{1}{1+\exp(-z)}$$



What is $a_3^{(2)}$?

$$+\left[b_1^{(2)}\right] = \left[z_1^{(3)}\right]$$

a) One of the inputs to neurons on layer 2

b) One of the inputs to neurons on layer 3

c) activation value of 3rd neuron on layer 2

d) activation value of 2nd neuron on layer 3

e) activation value of 3rd neuron on layer 3

f) activation value of 2nd neuron on layer 3

$$h_{W,\mathbf{b}} = \mathbf{a}^{(3)} = f(\mathbf{z}^{(3)})$$

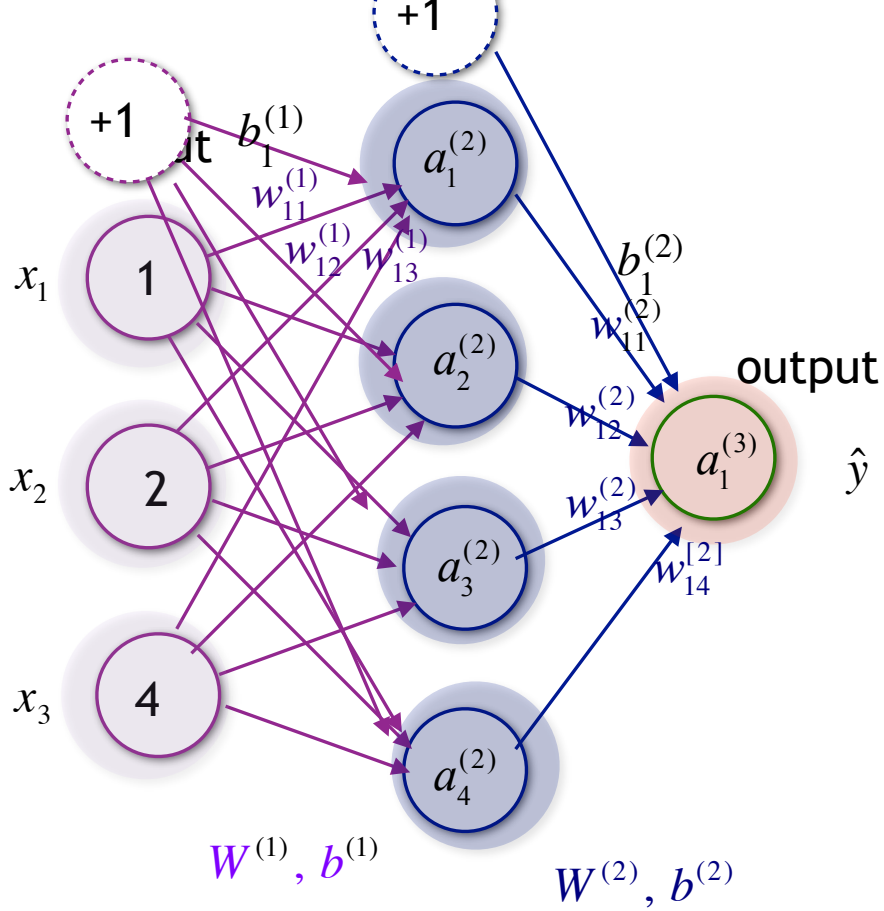
Artificial Neural Networks

- Each node (we call the nodes ***neurons***) represents a linear function of its input, similar to logistic regression

$$z_1^{(2)} = W_{11}^{(1)} x_1 + W_{12}^{(1)} x_2 + W_{13}^{(1)} x_3 + b_1^{(1)}$$
$$a_1^{(2)} = f(z_1^{(2)})$$

  - $W_{11}^{(1)}\ W_{12}^{(1)}\ W_{13}^{(1)}\ b_1^{(1)}$ are the ***weights*** (aka parameters)
  - $W^{(1)}$ is the matrix of weights that map values from input layer 1 to the first hidden layer (layer 2) ($W^{(\ell)}$ is the matrix that maps layer ℓ to layer ℓ+1)
  - $b^{(1)}$ is the *vector* of bias values for the neurons on layer 1

- $f^{(1)}$ is the "activation function" for layer 1. ($f^{(\ell)}$ is the activation function for layer ℓ)
- $a_j^{(2)}$ is the activation value for the j^{th} neuron of layer 2
  $a_j^{(\ell)}$ is the activation value for the j^{th} neuron of layer the ℓ



$$W^{(1)} = \begin{bmatrix} W_{11}^{(1)} & W_{12}^{(1)} & W_{13}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} & W_{23}^{(1)} \\ W_{31}^{(1)} & W_{32}^{(1)} & W_{33}^{(1)} \\ W_{41}^{(1)} & W_{42}^{(1)} & W_{43}^{(1)} \end{bmatrix} \quad b^{(1)} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \\ b_3^{(1)} \\ b_4^{(1)} \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} W_{11}^{(2)} & W_{12}^{(2)} & W_{13}^{(2)} & W_{14}^{(2)} \end{bmatrix} \quad \mathbf{b}^{(2)} = \begin{bmatrix} b_1^{(2)} \end{bmatrix}$$

Let s_j is the number of nodes at layer j where we don't include the bias unit

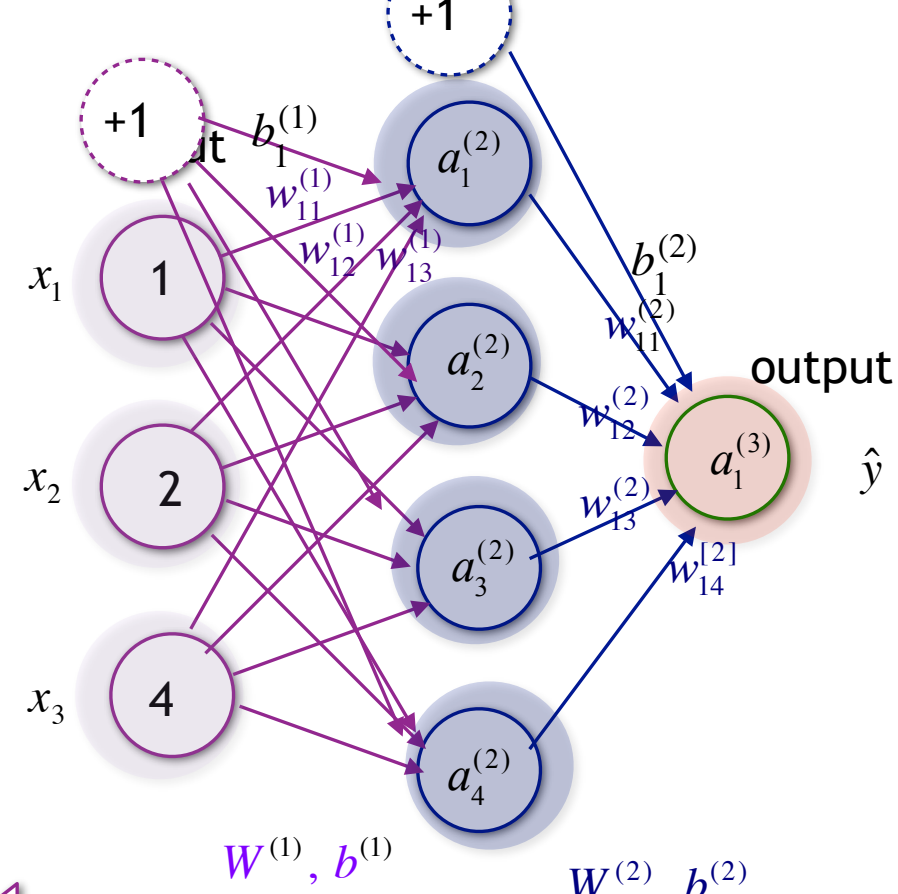
Artificial Neural Networks

- Each node (we call the nodes ***neurons***) represents a linear function of its input, similar to logistic regression

$$z_1^{(2)} = W_{11}^{(1)} x_1 + W_{12}^{(1)} x_2 + W_{13}^{(1)} x_3 + b_1^{(1)}$$
$$a_1^{(2)} = f(z_1^{(2)})$$

  - $W_{11}^{(1)}\ W_{12}^{(1)}\ W_{13}^{(1)}\ b_1^{(1)}$ are the ***weights*** (aka parameters)
  - $W^{(1)}$ is the matrix of weights that map values from input layer 1 to the first hidden layer (layer 2) ($W^{(\ell)}$ is the matrix that maps layer ℓ to layer ℓ+1)
  - $b^{(1)}$ is the *vector* of bias values for the neurons on layer 1

- $f^{(1)}$ is the "activation function" for layer 1. ($f^{(\ell)}$ is the activation function for layer ℓ)
- $a_j^{(2)}$ is the activation value for the j^{th} neuron of layer 2
  $a_j^{(\ell)}$ is the activation value for the j^{th} neuron of layer the ℓ



One example of an activation function is the logistic sigmoid

$$W = \begin{bmatrix} W_{31}^{(1)} & W_{32}^{(1)} & W_{33}^{(1)} \\ W_{41}^{(1)} & W_{42}^{(1)} & W_{43}^{(1)} \end{bmatrix} \qquad b = \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \\ b_3^{(1)} \\ b_4^{(1)} \end{bmatrix}$$

Let s_j is the number of nodes at layer j where we don't include the bias unit

$$W^{(2)} = \begin{bmatrix} W_{11}^{(2)} & W_{12}^{(2)} & W_{13}^{(2)} & W_{14}^{(2)} \end{bmatrix} \qquad \mathbf{b}^{(2)} = \begin{bmatrix} b_1^{(2)} \end{bmatrix}$$