

An Introduction to Deep Learning

(part 1)



Farnoosh Heidarivincheh
EMAT31530 - February 2021

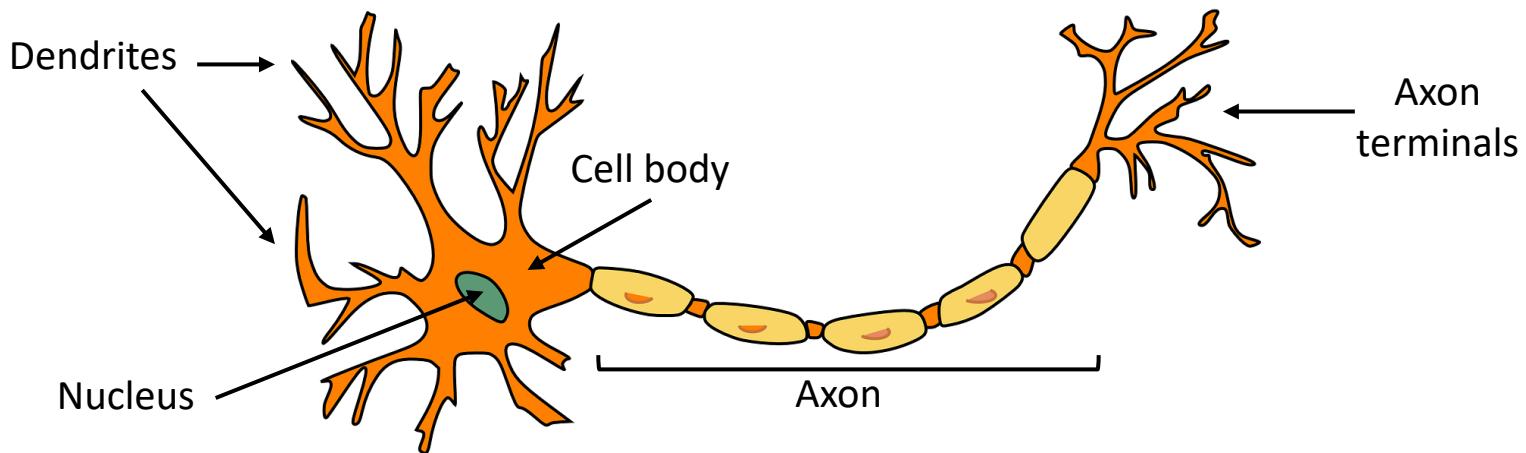
Outline

- Biological Neuron
- Artificial Neuron
- Artificial Neural Network
- Network Training

Biological Neuron

Biological Neuron

A neuron is the most basic unit of our nervous system



- It receives sensory input/electrical signal from neighbor neuron
- It sends electrical message to other neurons
- Biological neurons can get “*active*”, depending on their input signal

Biological Neuron

- Our perception of the world is based on the interconnections among the neurons in our brain

Input



Biological Neuron

- Our perception of the world is based on the interconnections among the neurons in our brain

Input



Biological Neuron

- Our perception of the world is based on the interconnections among the neurons in our brain



Biological Neuron

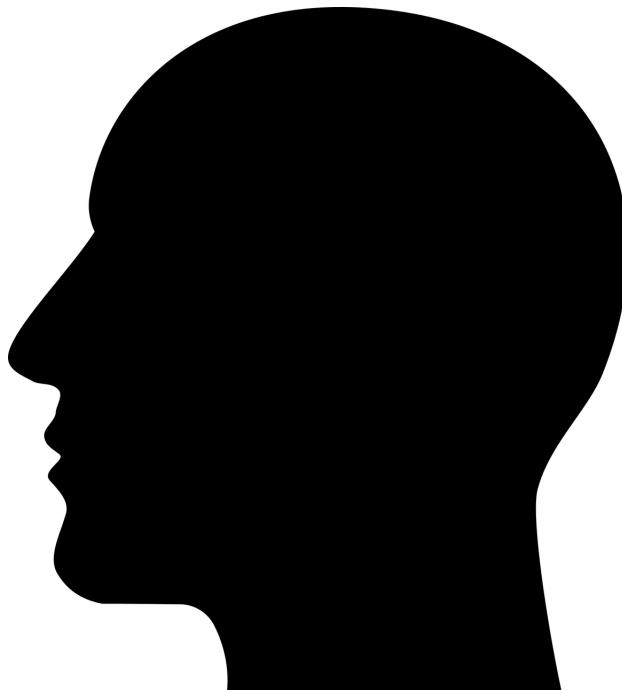
- Our perception of the world is based on the interconnections among the neurons in our brain

Input



Output

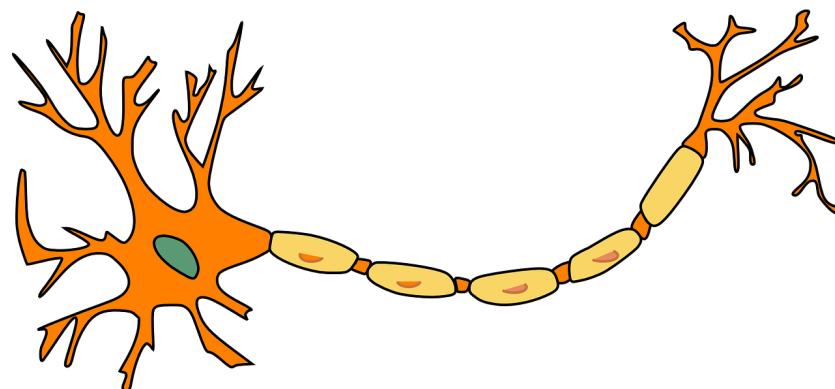
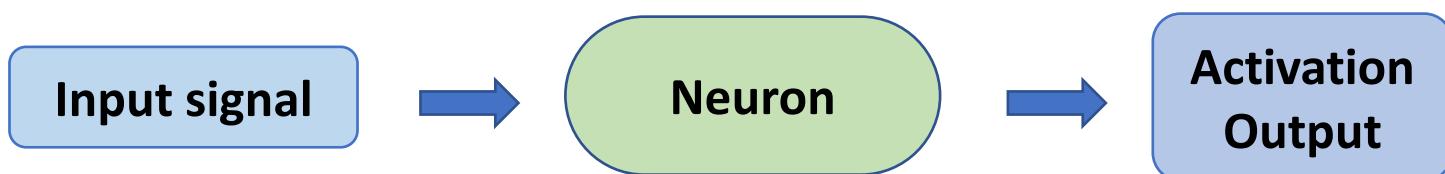
This is a dog



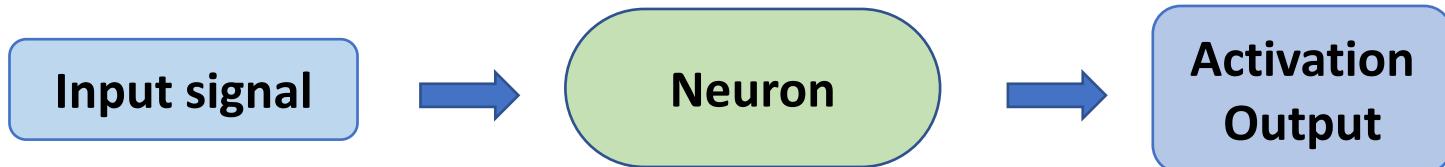
Artificial Neuron

Artificial Neuron

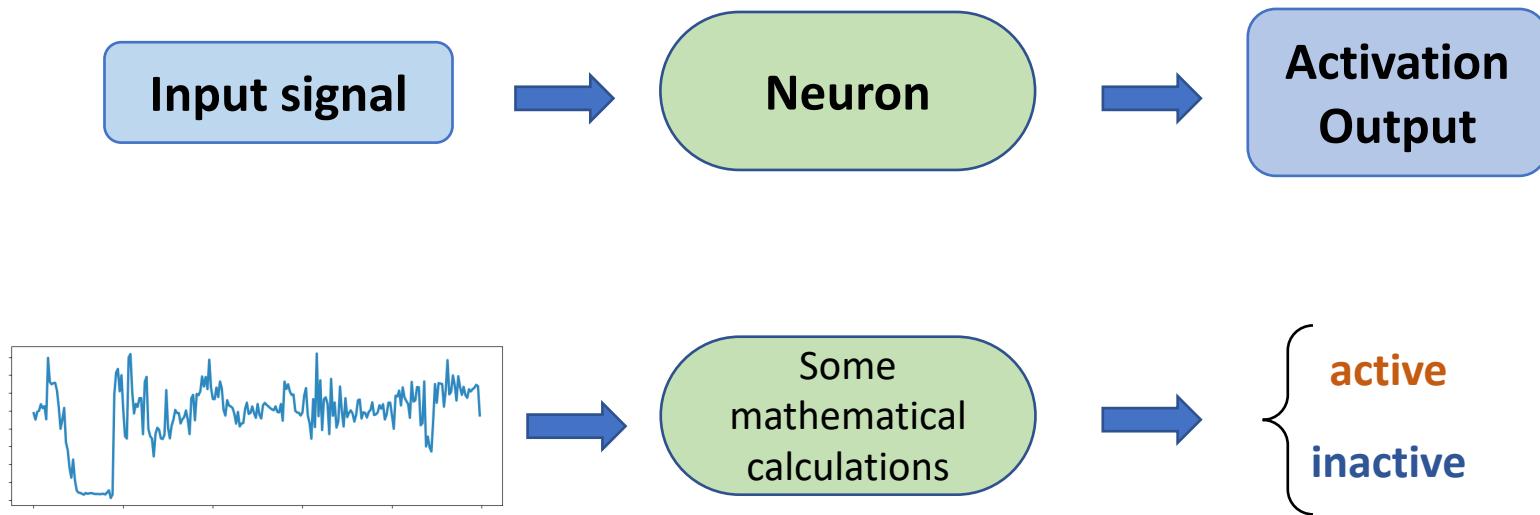
- An artificial neuron is the most basic unit of an Artificial Neural network
- It is inspired by the neurons in our brains



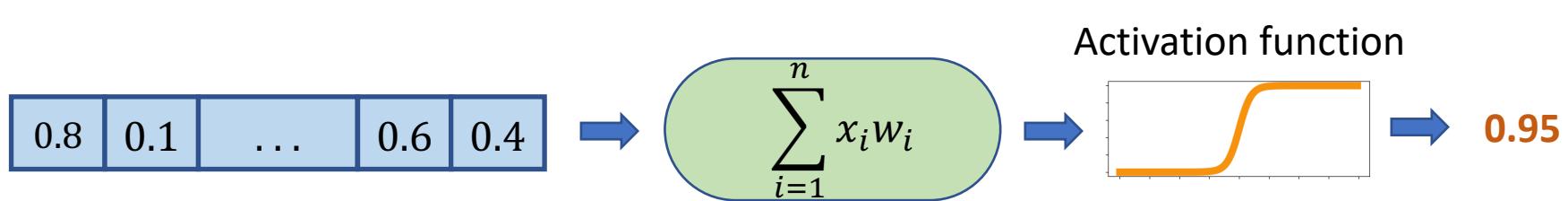
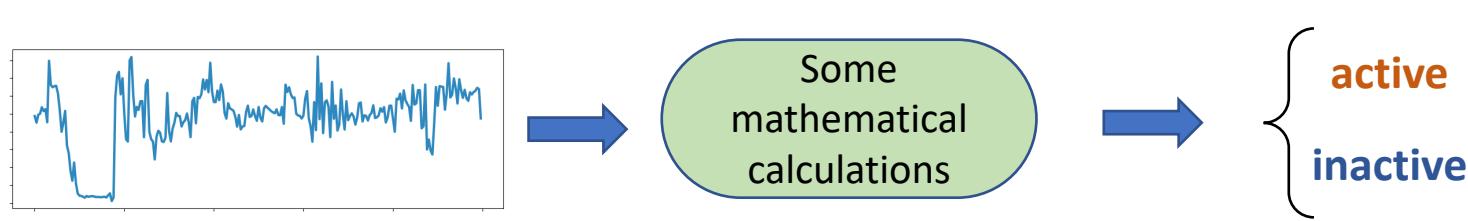
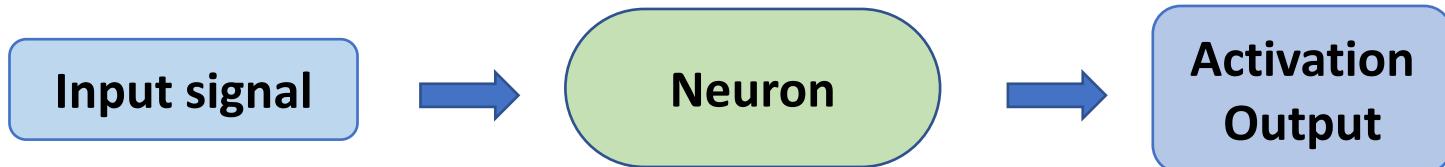
Artificial Neuron



Artificial Neuron

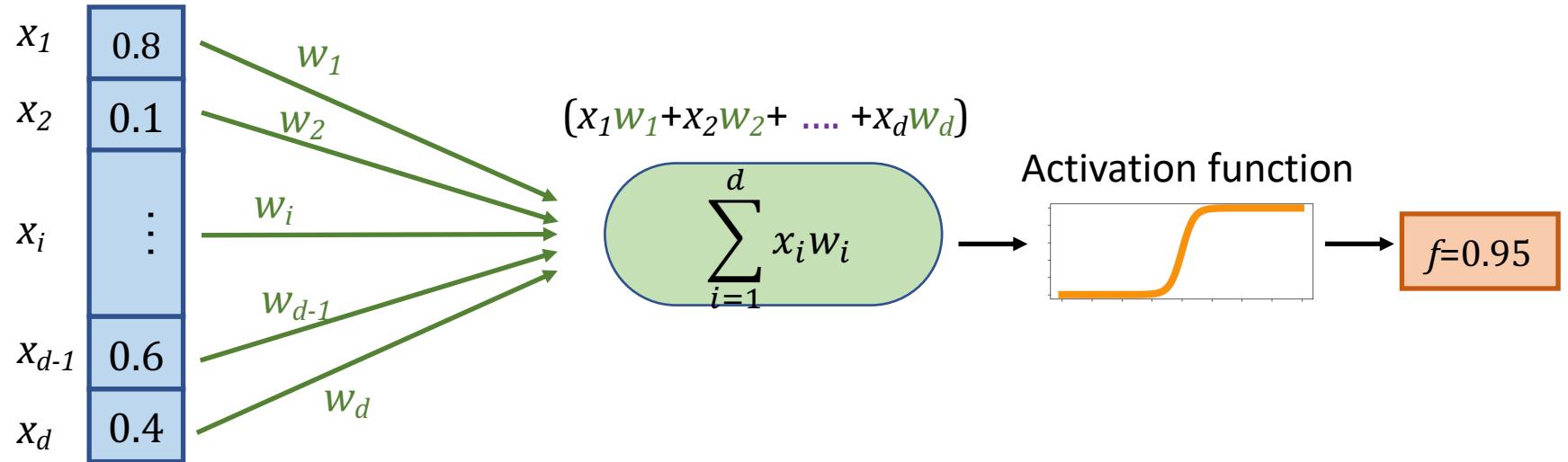


Artificial Neuron



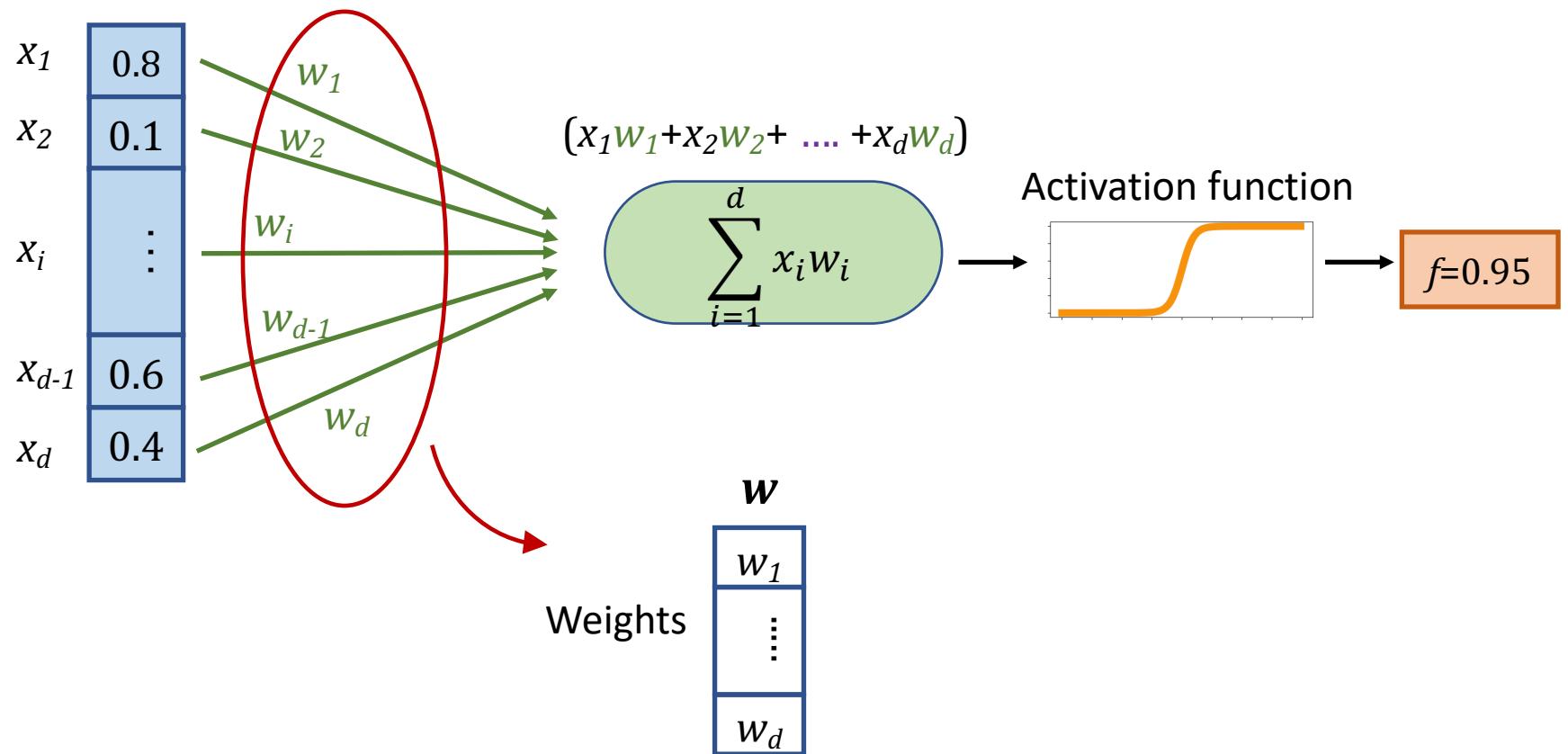
Artificial Neuron

Input vector \mathbf{x}



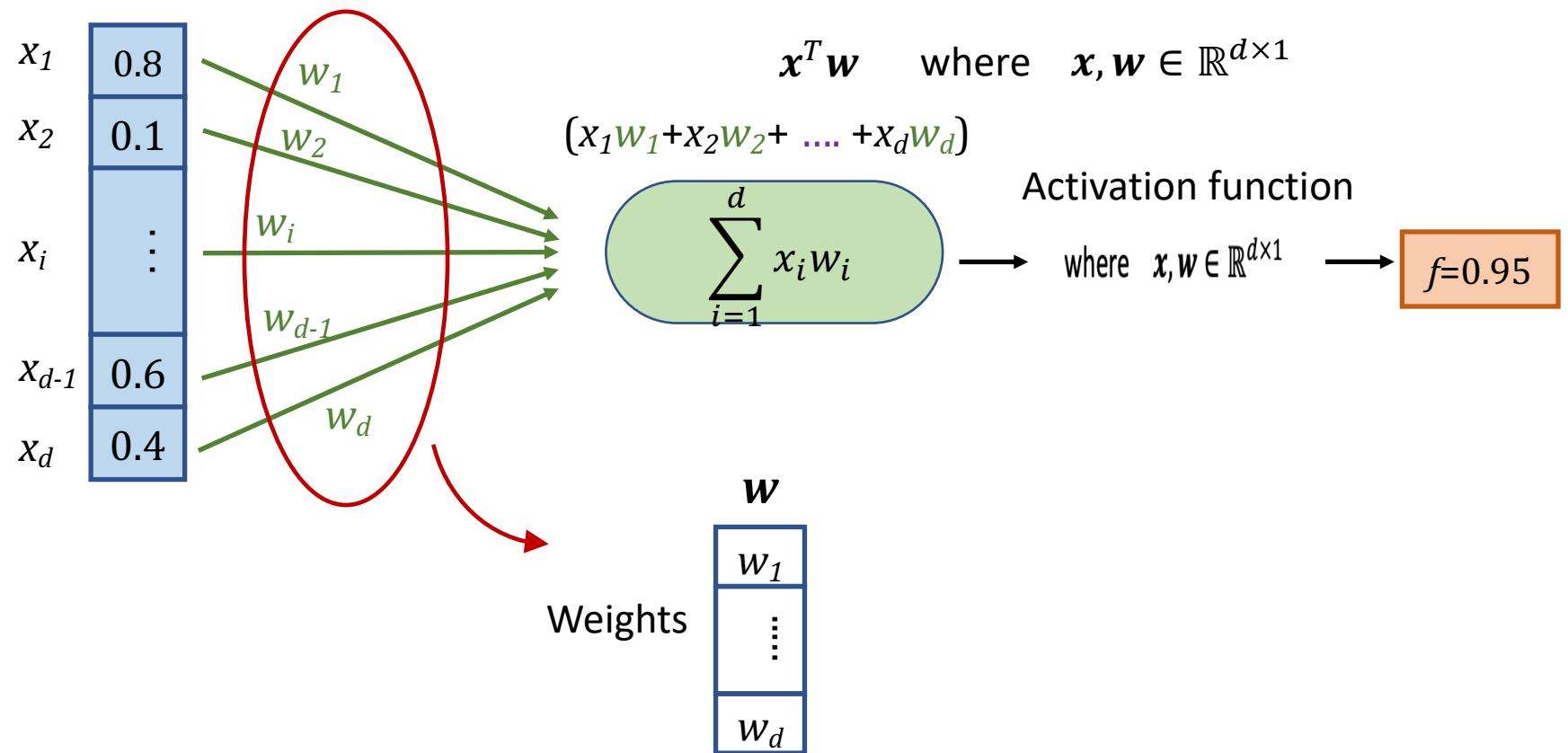
Artificial Neuron

Input vector x



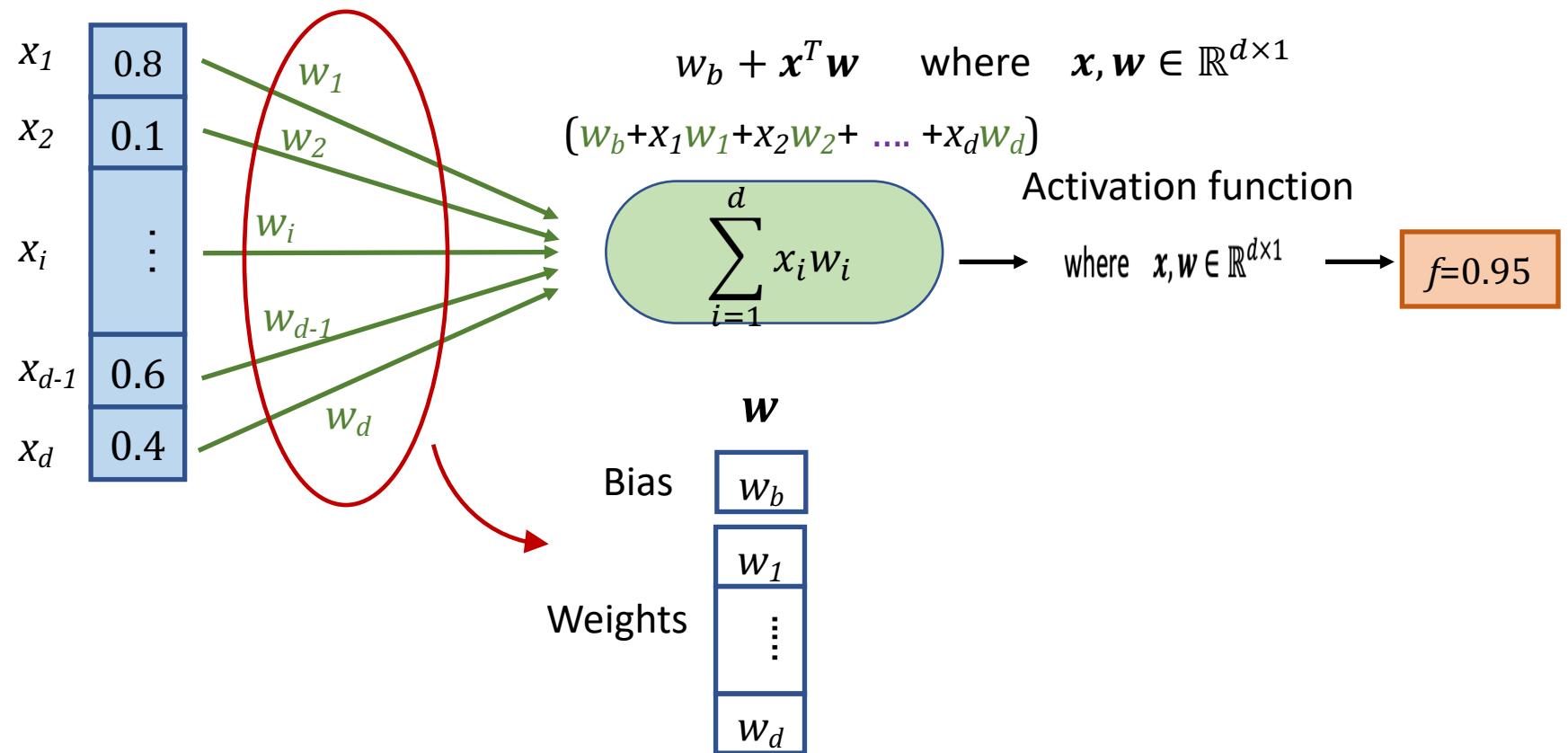
Artificial Neuron

Input vector \mathbf{x}



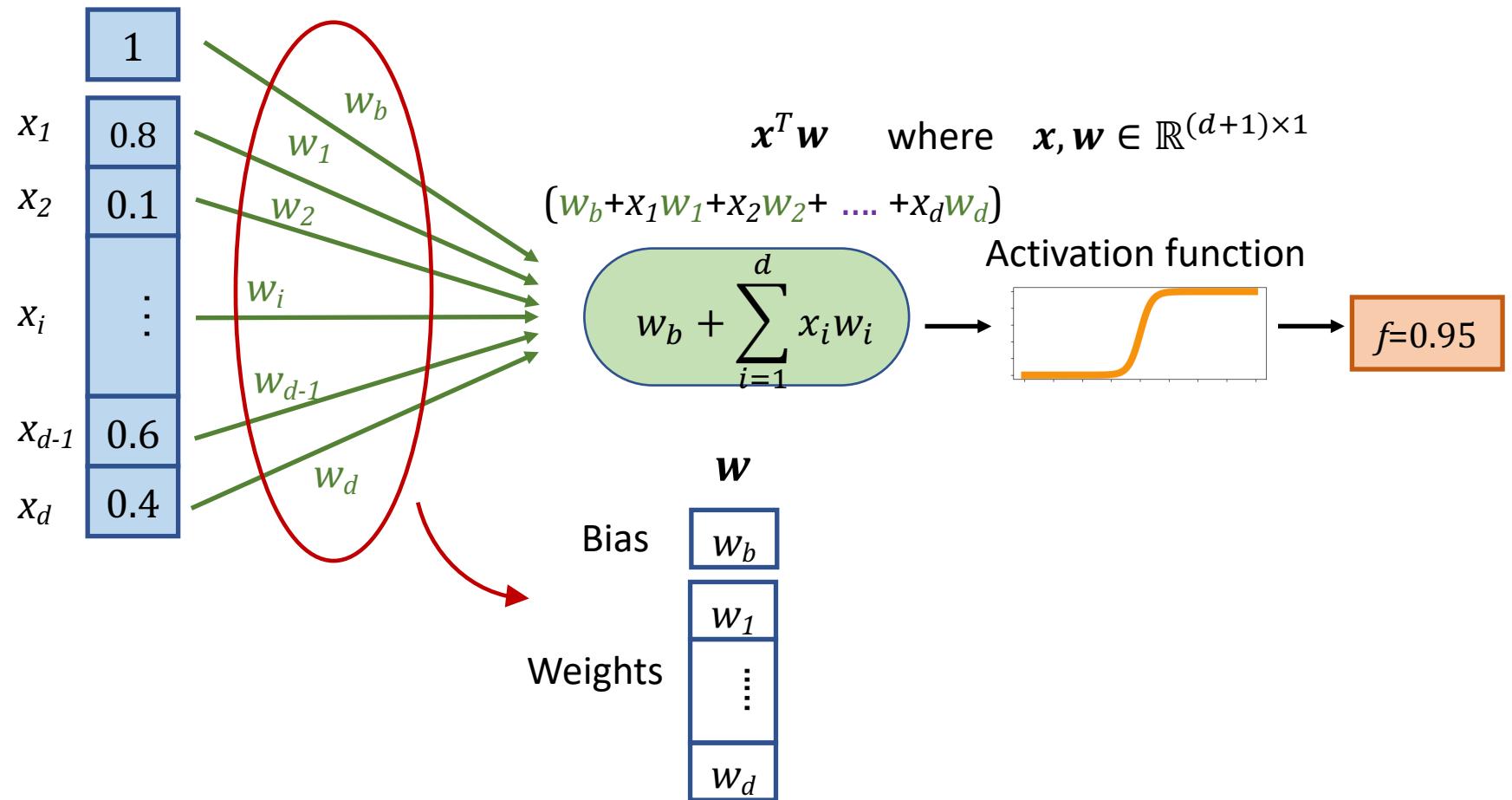
Artificial Neuron

Input vector \mathbf{x}



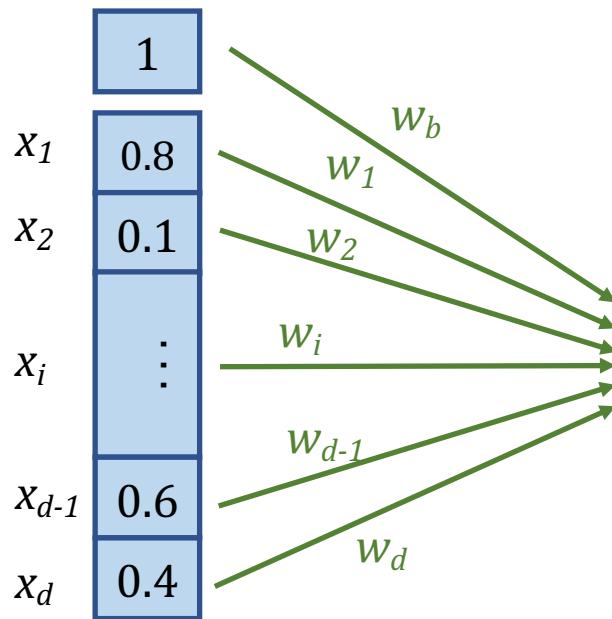
Artificial Neuron

Input vector \mathbf{x}



Artificial Neuron

Input vector \mathbf{x}



Activation function → non-linearity

e.g. sigmoid $g(x) = \frac{1}{1+e^{-x}}$

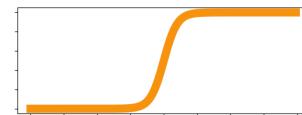
$$\mathbf{x}^T \mathbf{w}$$

$$(w_b + x_1 w_1 + x_2 w_2 + \dots + x_d w_d)$$

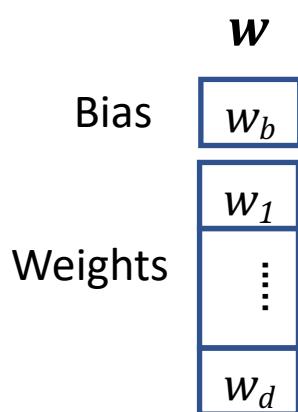
$$w_b + \sum_{i=1}^d x_i w_i$$

$$f = g(\mathbf{x}^T \mathbf{w})$$

$$f=0.95$$

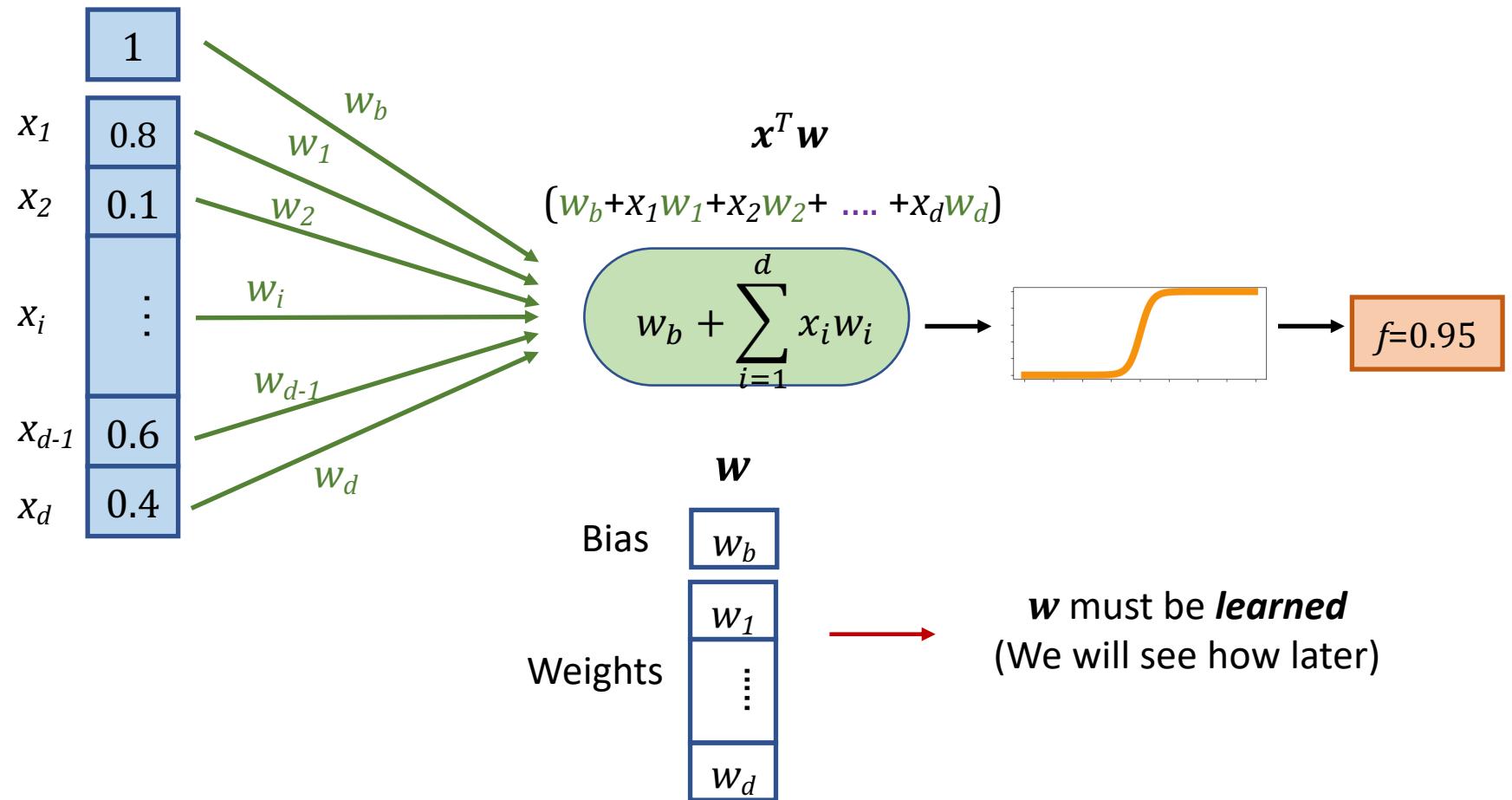


(A Perceptron uses a step function)



Artificial Neuron

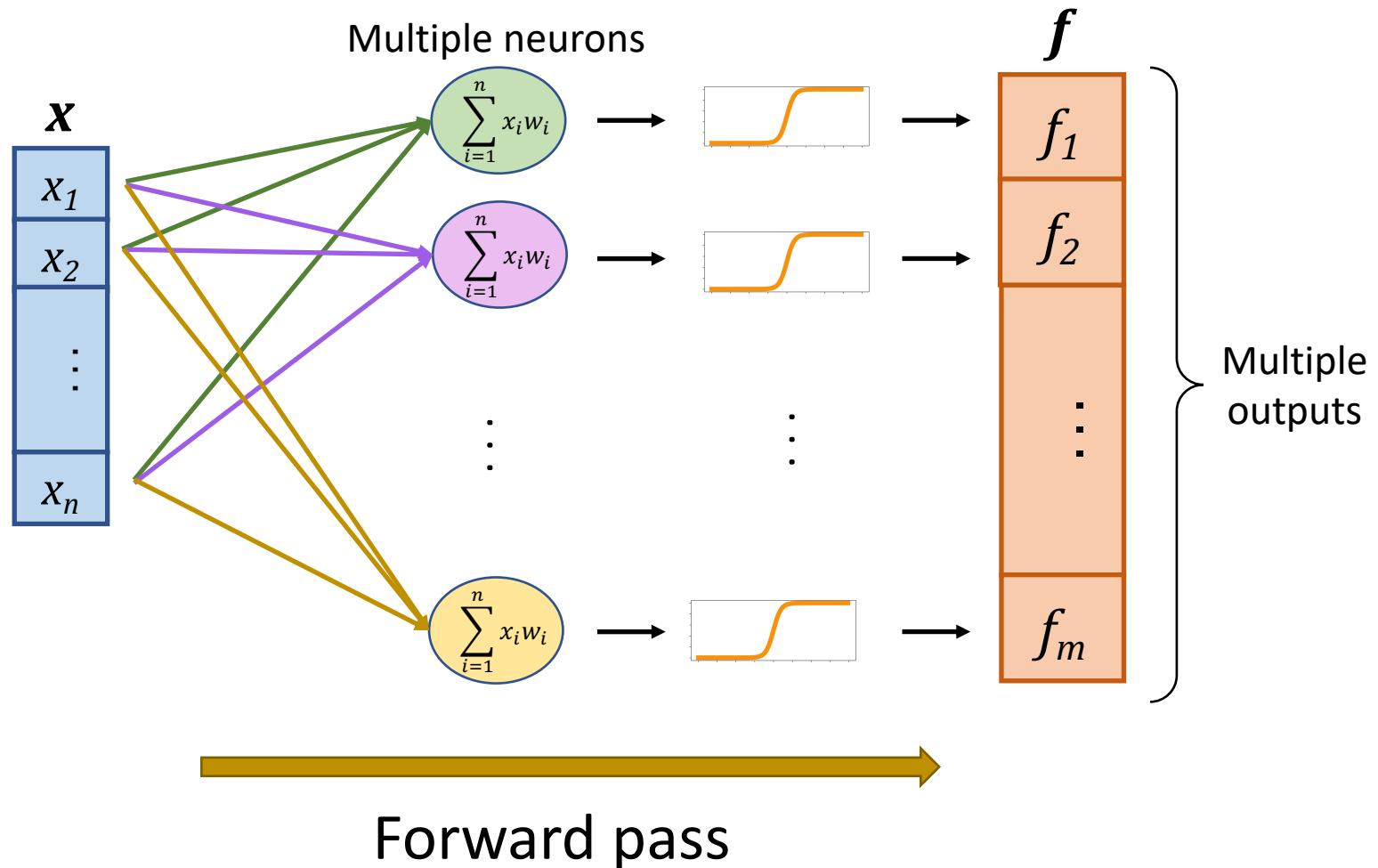
Input vector \mathbf{x}



Artificial Neural Network

Artificial Neural Network

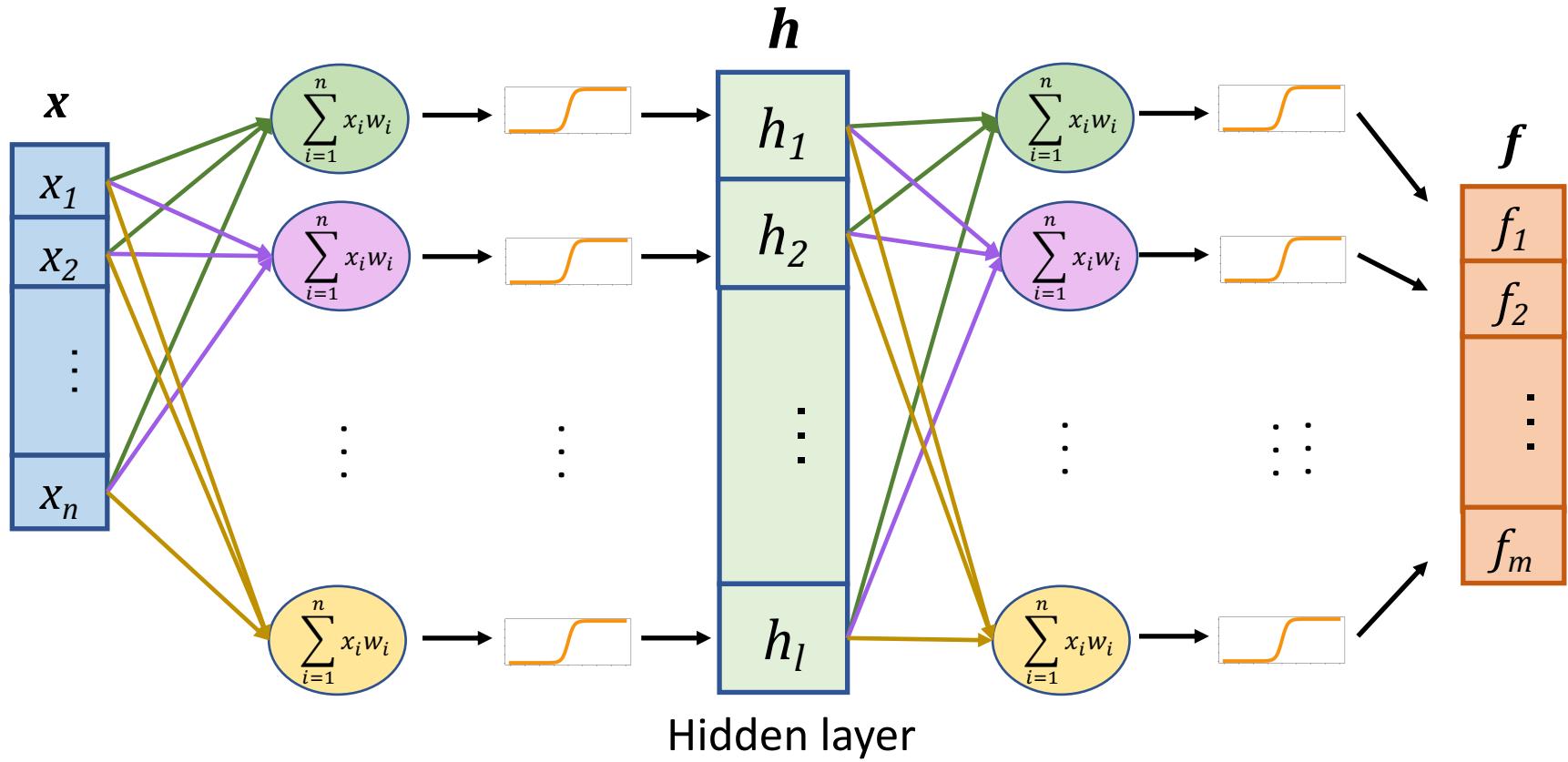
A Fully-Connected (FC/Dense) layer



Artificial Neural Network

A single-layer perceptron

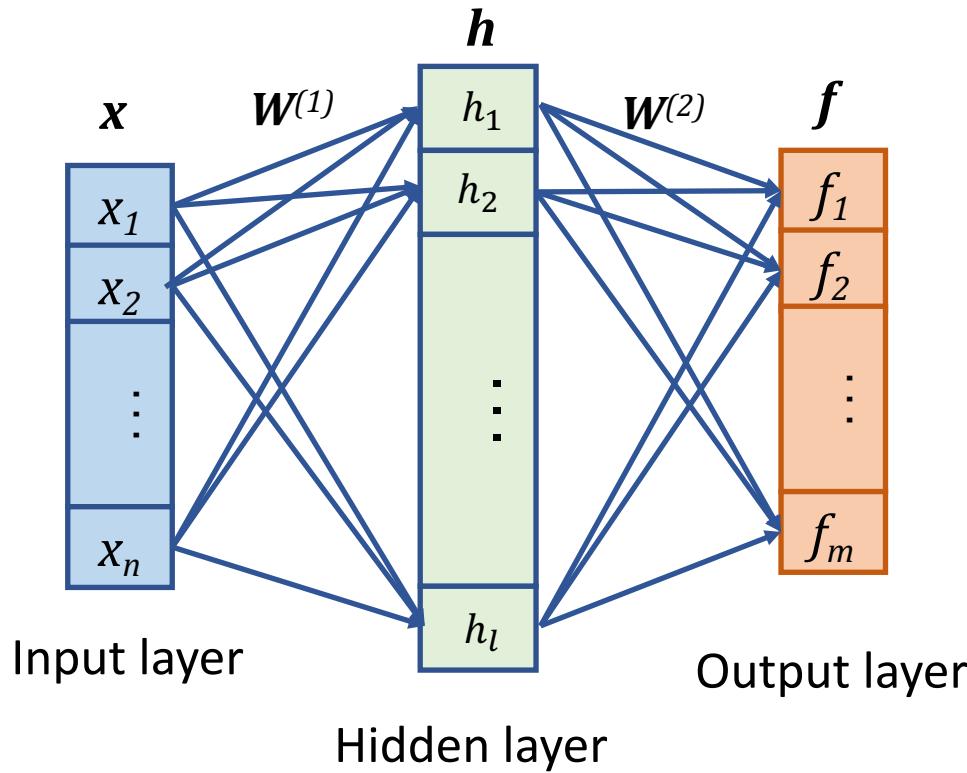
- We add a hidden layer between input and output layers



Artificial Neural Network

A single-layer perceptron

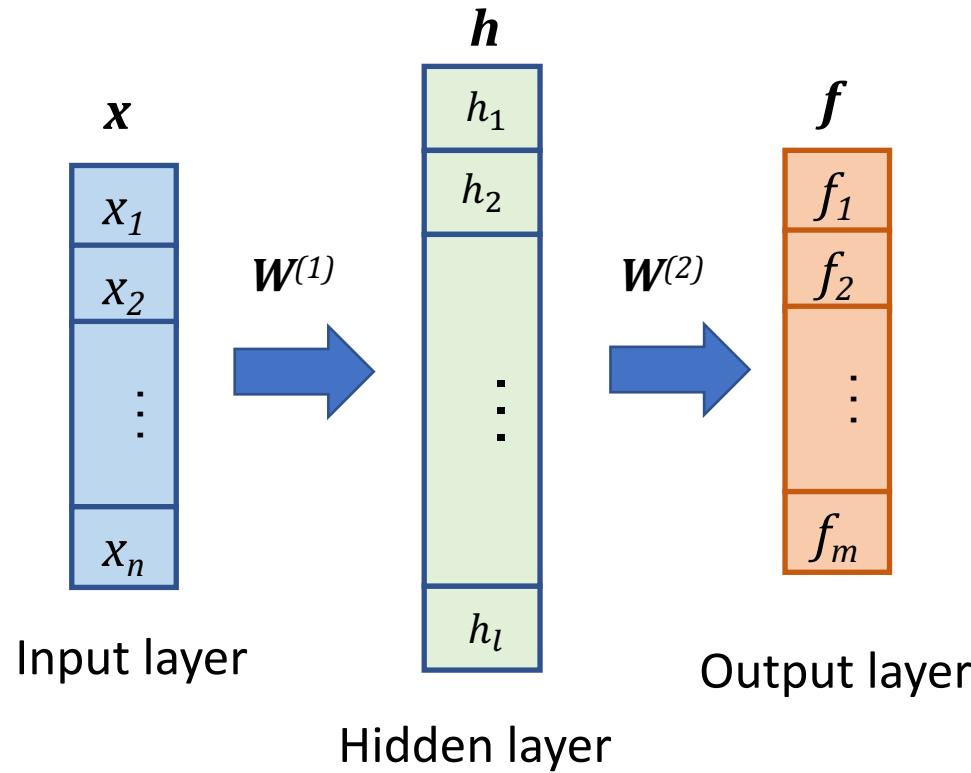
- We add a hidden layer between input and output layers



Artificial Neural Network

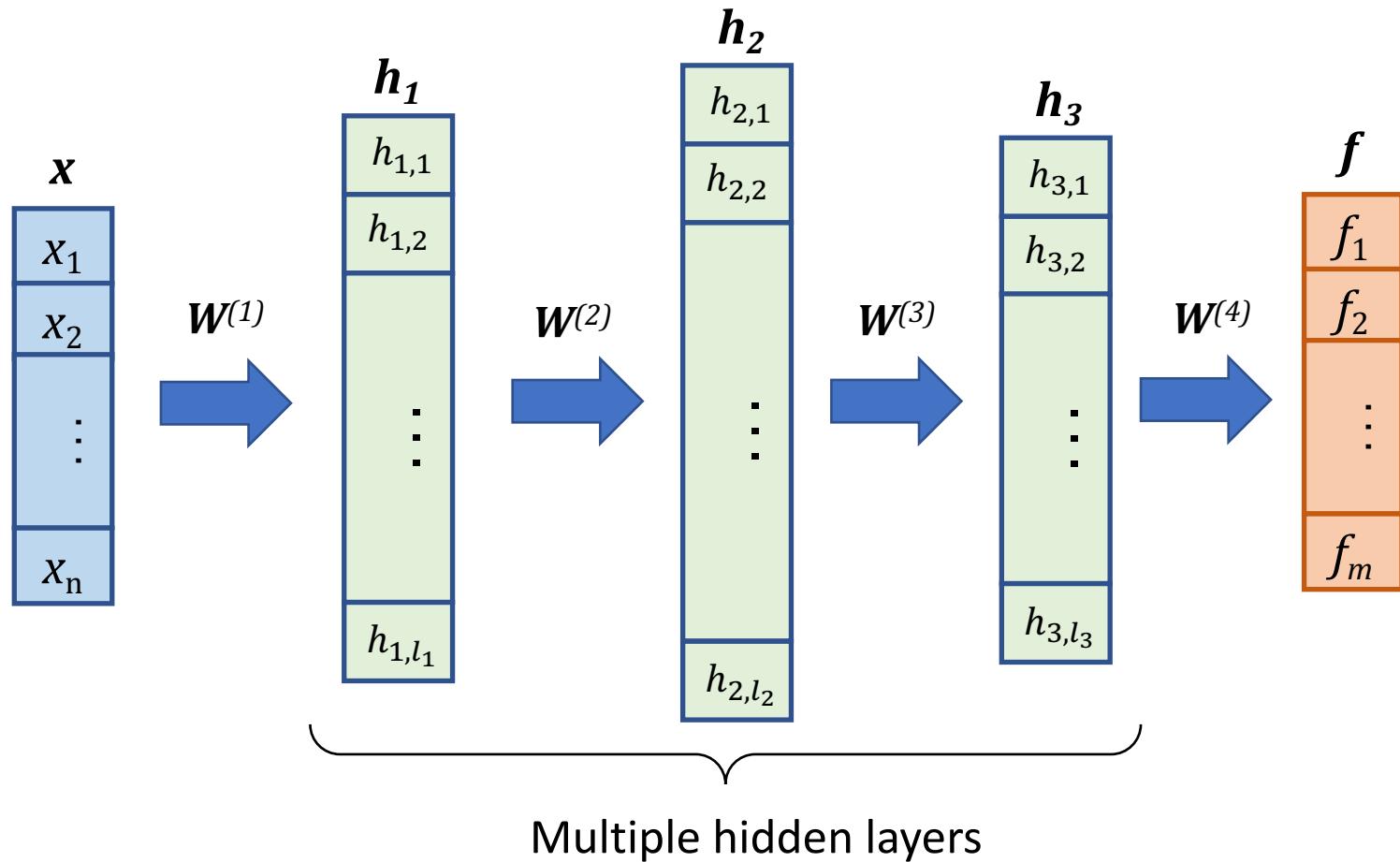
A single-layer perceptron

- We add a hidden layer between input and output layers



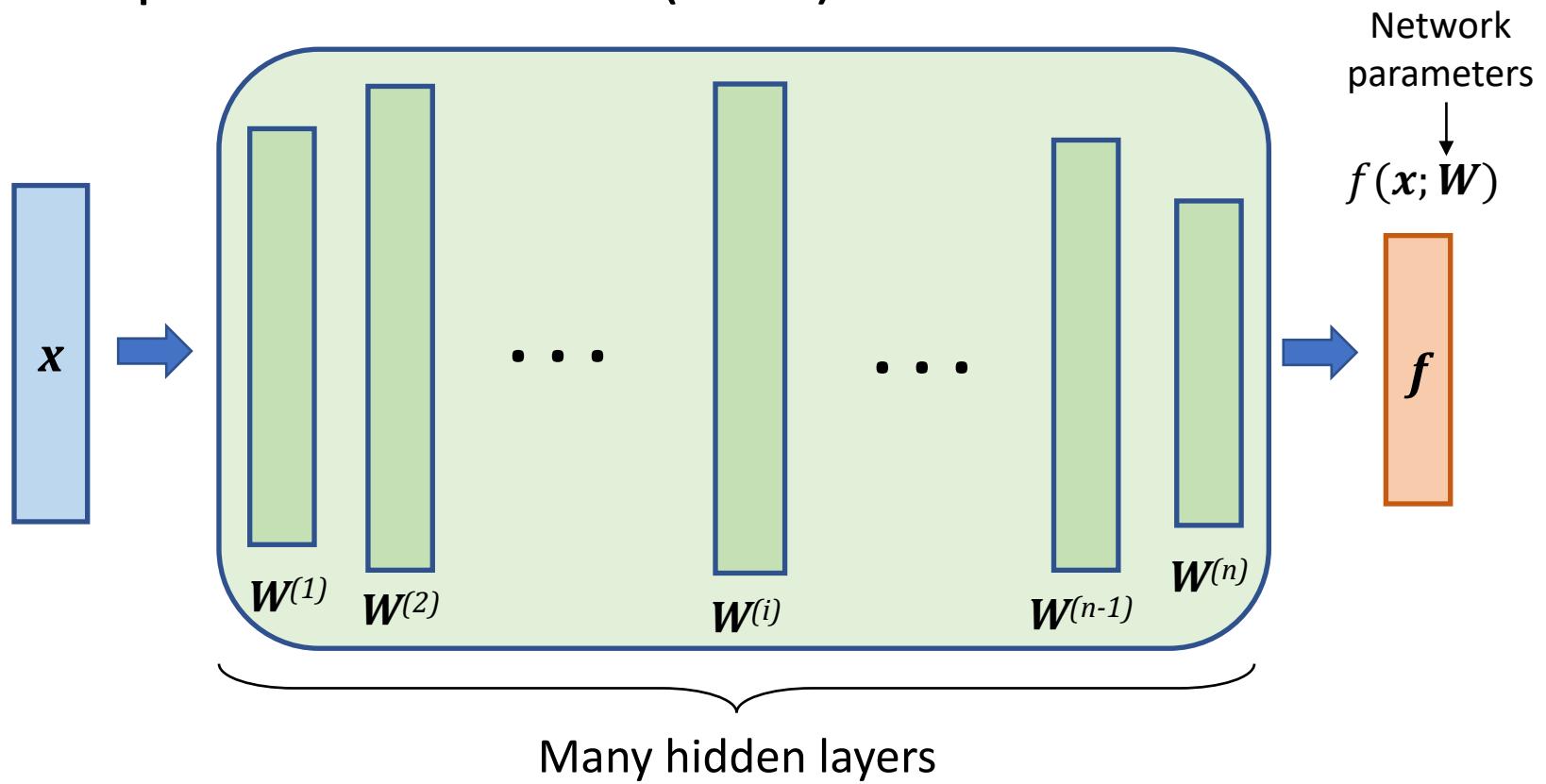
Artificial Neural Network

A Multi-Layer Perceptron (MLP)



Artificial Neural Network

A Deep Neural Network (DNN)



Not all of them necessarily dense layers – we will see other types of layers later

Artificial Neural Network

Different types of data used by a DNN

1D (text, audio, ...)



2D (image, ...)



Deep Neural
Network



3D (video, ...)



f

Artificial Neural Network

DNNs have been applied in many fields

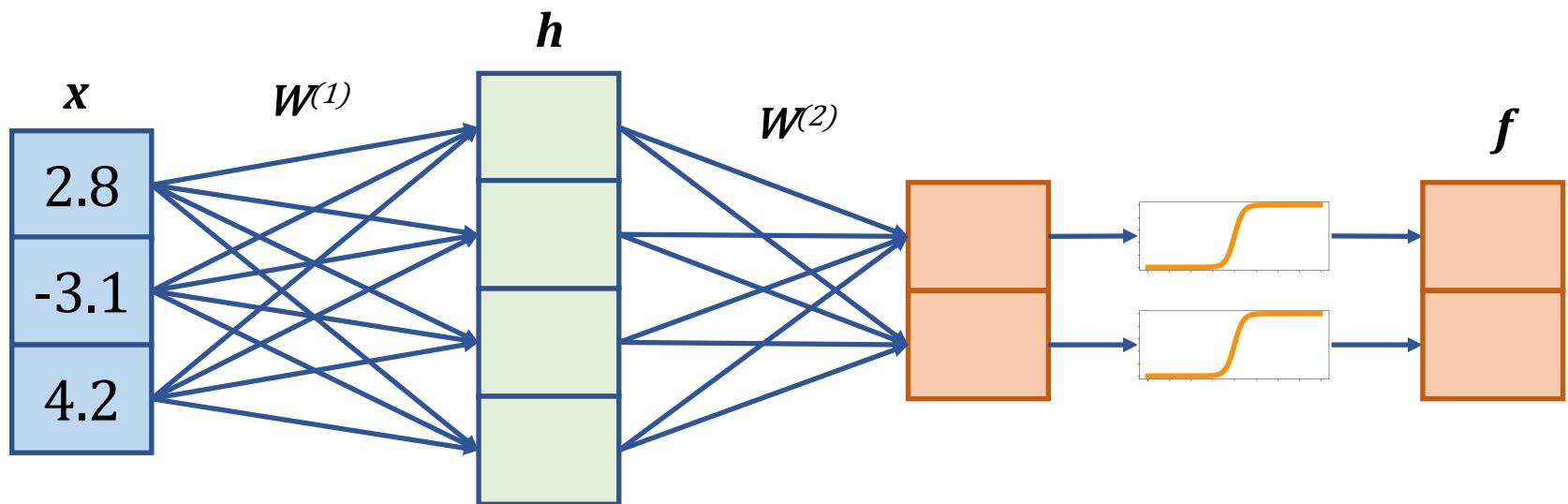
- Natural Language Processing
 - Audio recognition
 - Speech Recognition
 - Machine Translation
 - Question Answering
 - Image Captioning
 - Image Recognition
 - Action Recognition
 - Medical Image Analysis
 - Personalised Medicine
- Computer Vision
- Health-care

:

:

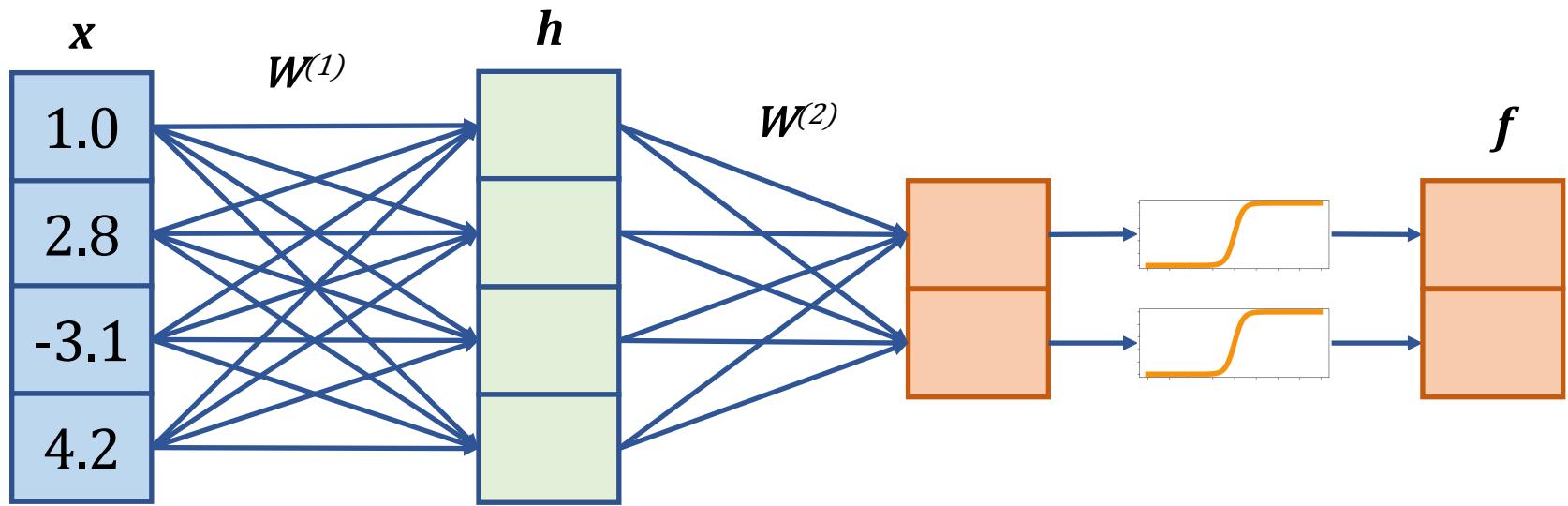
Artificial Neural Network

Example



Artificial Neural Network

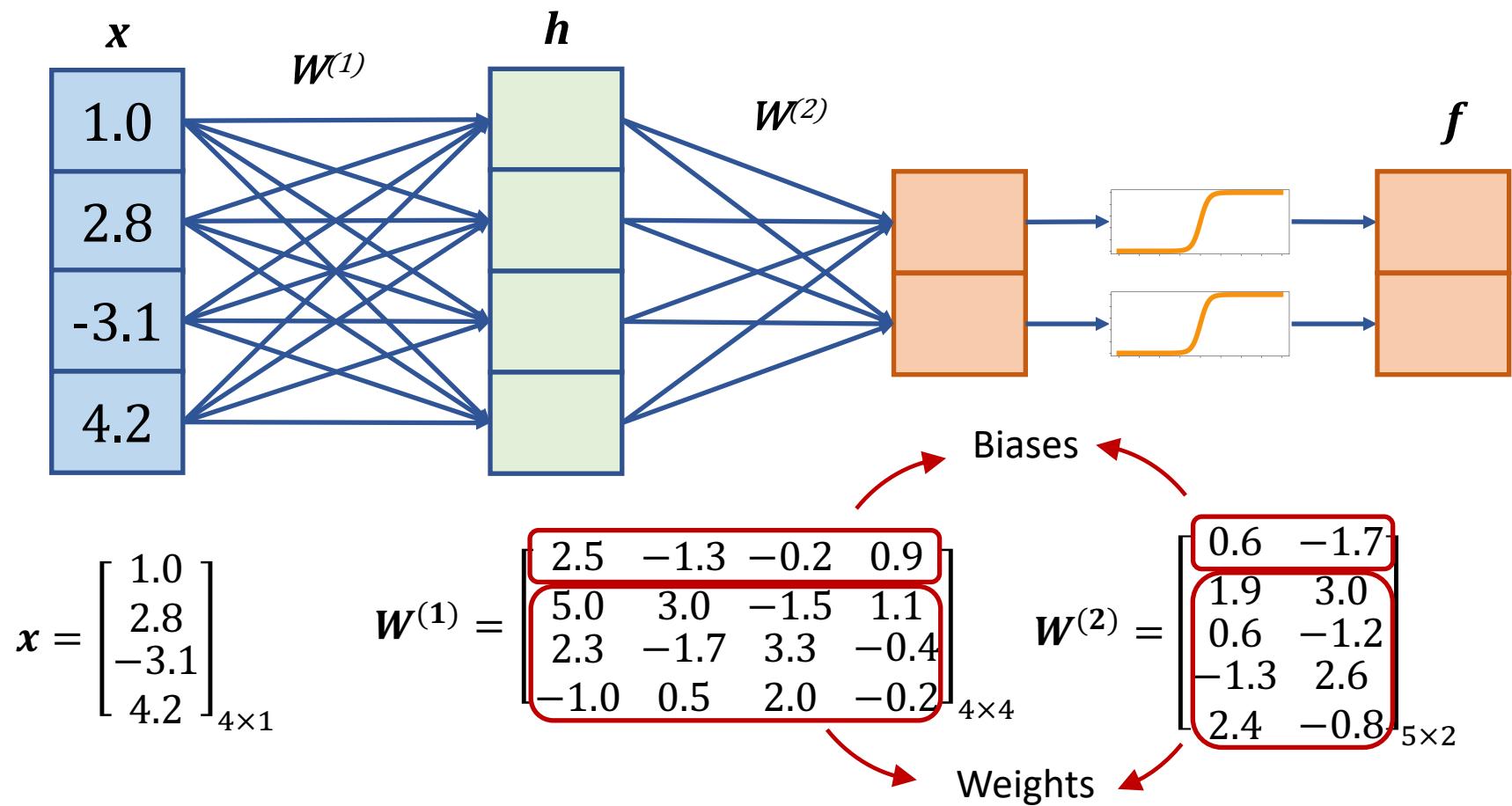
Example



$$x = \begin{bmatrix} 1.0 \\ 2.8 \\ -3.1 \\ 4.2 \end{bmatrix}_{4 \times 1} \quad W^{(1)} = \begin{bmatrix} 2.5 & -1.3 & -0.2 & 0.9 \\ 5.0 & 3.0 & -1.5 & 1.1 \\ 2.3 & -1.7 & 3.3 & -0.4 \\ -1.0 & 0.5 & 2.0 & -0.2 \end{bmatrix}_{4 \times 4} \quad W^{(2)} = \begin{bmatrix} 0.6 & -1.7 \\ 1.9 & 3.0 \\ 0.6 & -1.2 \\ -1.3 & 2.6 \\ 2.4 & -0.8 \end{bmatrix}_{5 \times 2}$$

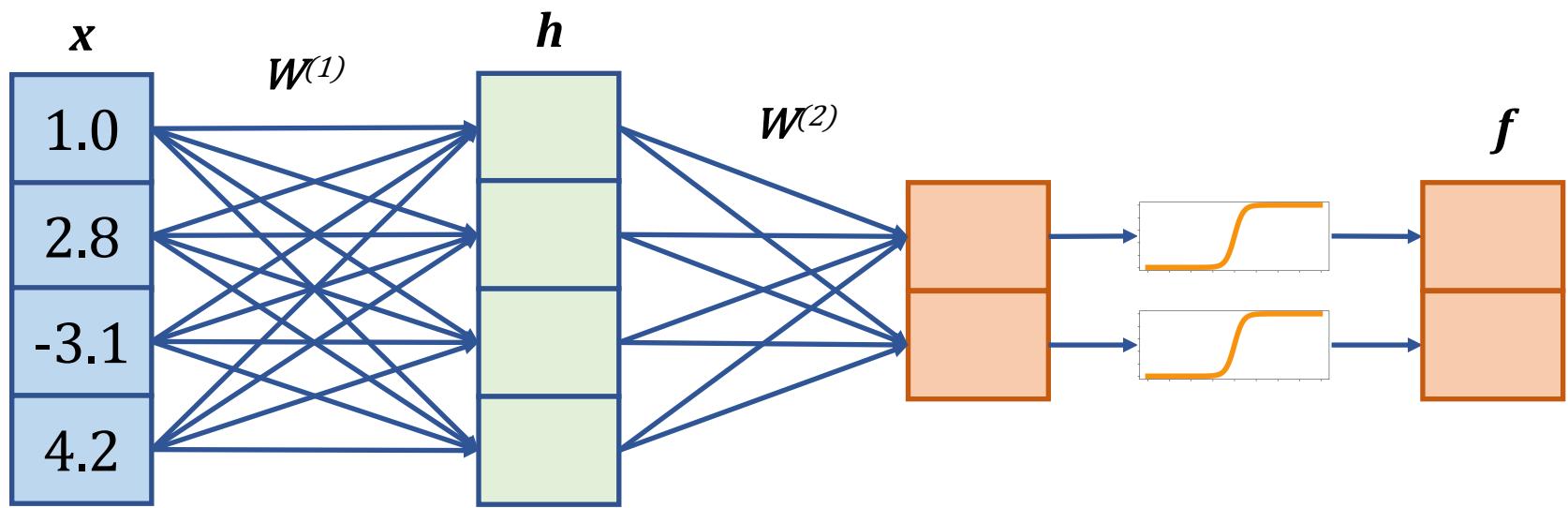
Artificial Neural Network

Example



Artificial Neural Network

Example



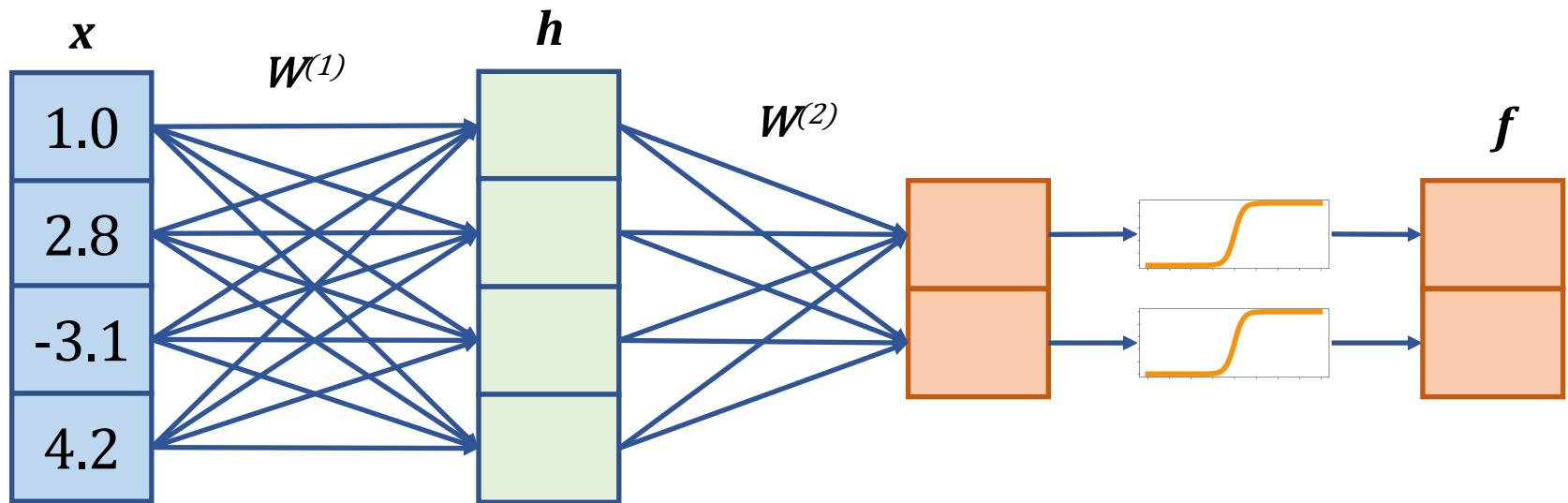
$$x = \begin{bmatrix} 1.0 \\ 2.8 \\ -3.1 \\ 4.2 \end{bmatrix}_{4 \times 1}$$

$$W^{(1)} = \begin{bmatrix} 2.5 & -1.3 & -0.2 & 0.9 \\ 5.0 & 3.0 & -1.5 & 1.1 \\ 2.3 & -1.7 & 3.3 & -0.4 \\ -1.0 & 0.5 & 2.0 & -0.2 \end{bmatrix}_{4 \times 4}$$

$$W^{(2)} = \begin{bmatrix} 0.6 & -1.7 \\ 1.9 & 3.0 \\ 0.6 & -1.2 \\ -1.3 & 2.6 \\ 2.4 & -0.8 \end{bmatrix}_{5 \times 2}$$

Artificial Neural Network

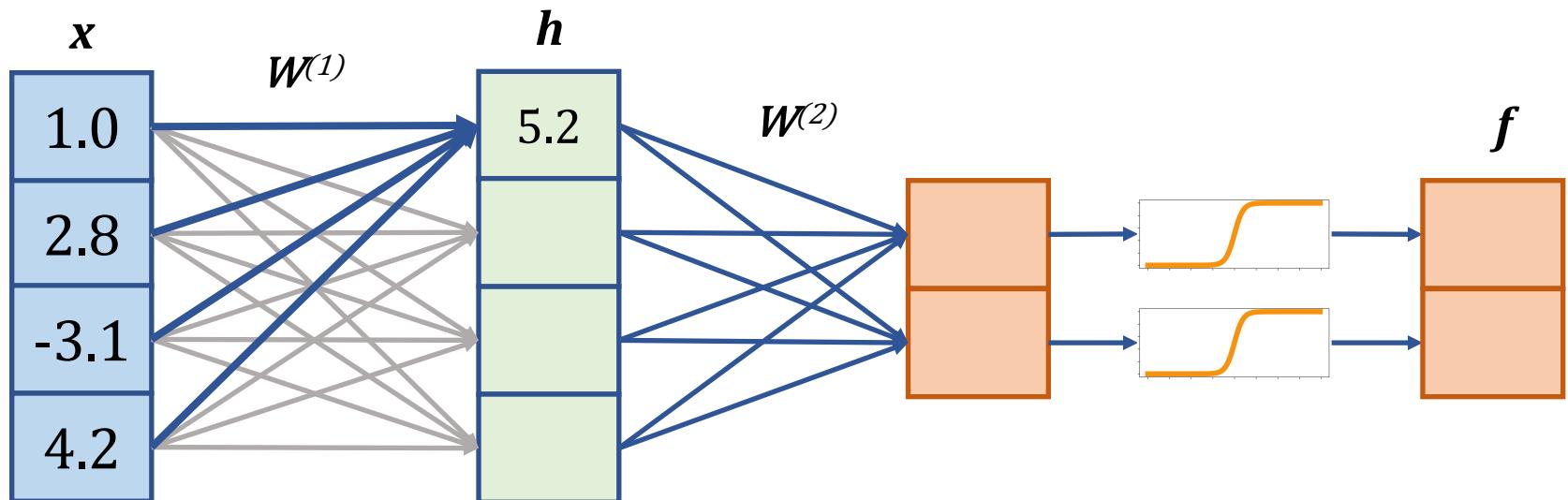
Example



$$x^T \times W^{(1)} = [1.0 \quad 2.8 \quad -3.1 \quad 4.2]_{1 \times 4} \times \begin{bmatrix} 2.5 & -1.3 & -0.2 & 0.9 \\ 5.0 & 3.0 & -1.5 & 1.1 \\ 2.3 & -1.7 & 3.3 & -0.4 \\ -1.0 & 0.5 & 2.0 & -0.2 \end{bmatrix}_{4 \times 4}$$
$$= [\quad]_{1 \times 4}$$

Artificial Neural Network

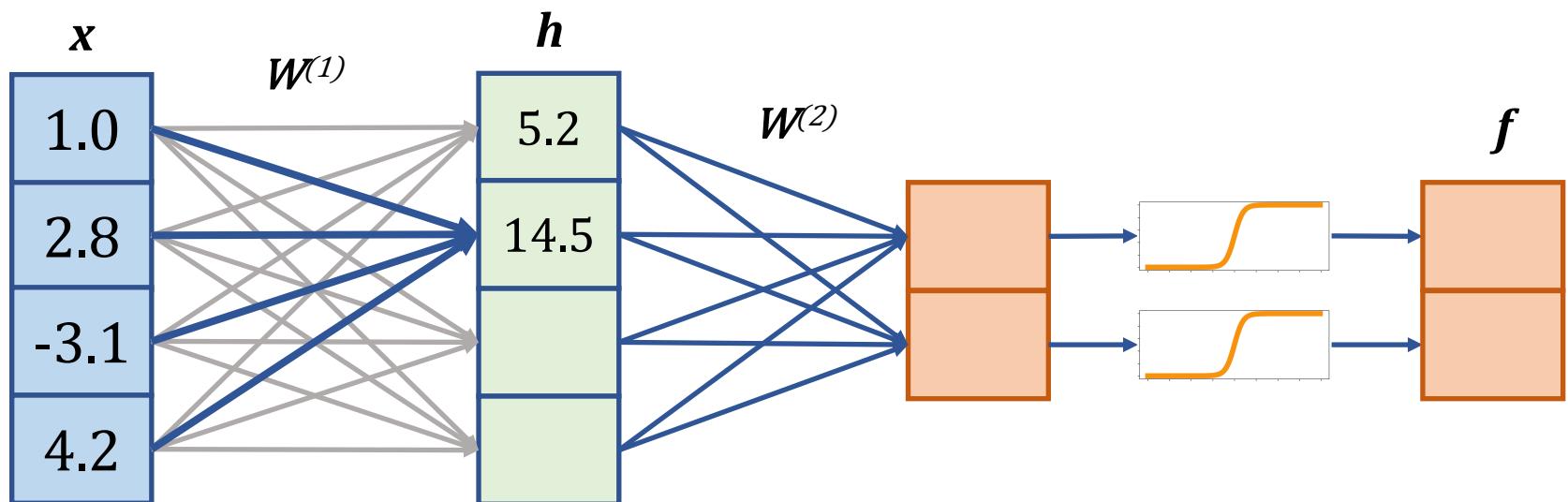
Example



$$\begin{aligned}x^T \times W^{(1)} &= [1.0 \quad 2.8 \quad -3.1 \quad 4.2]_{1 \times 4} \times \begin{bmatrix} 2.5 & -1.3 & -0.2 & 0.9 \\ 5.0 & 3.0 & -1.5 & 1.1 \\ 2.3 & -1.7 & 3.3 & -0.4 \\ -1.0 & 0.5 & 2.0 & -0.2 \end{bmatrix}_{4 \times 4} \\&= [5.2]_{1 \times 4}\end{aligned}$$

Artificial Neural Network

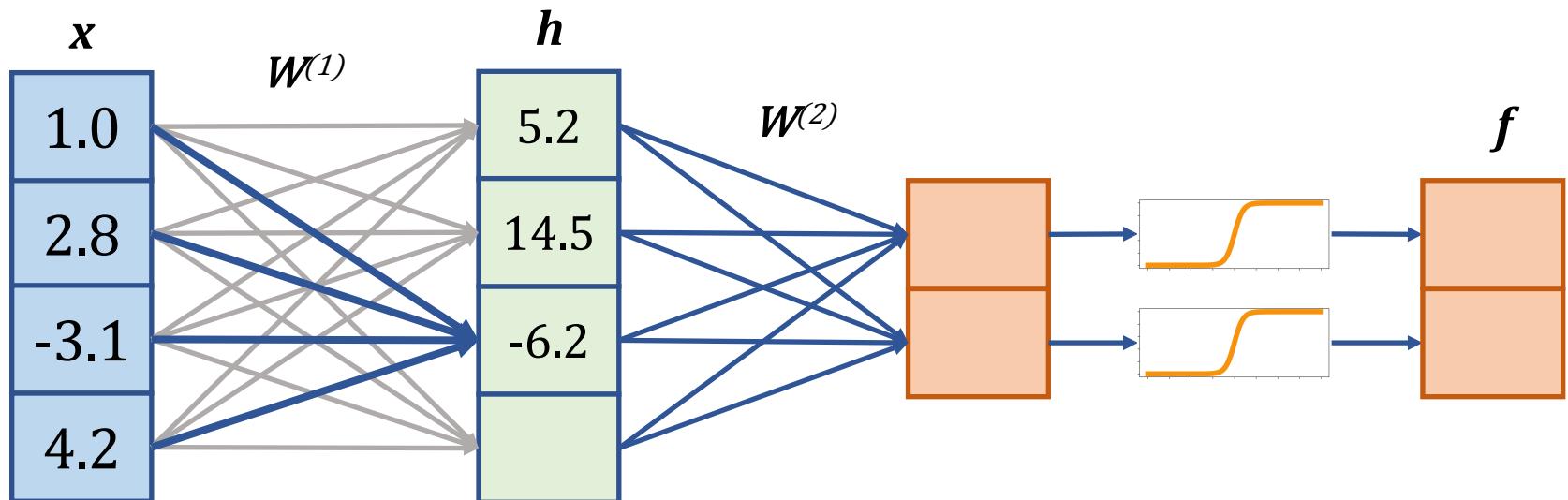
Example



$$\begin{aligned}x^T \times W^{(1)} &= [1.0 \quad 2.8 \quad -3.1 \quad 4.2]_{1 \times 4} \times \begin{bmatrix} 2.5 & -1.3 & -0.2 & 0.9 \\ 5.0 & 3.0 & -1.5 & 1.1 \\ 2.3 & -1.7 & 3.3 & -0.4 \\ -.01 & 0.5 & 2.0 & -0.2 \end{bmatrix}_{4 \times 4} \\&= [5.2 \quad 14.5]_{1 \times 4}\end{aligned}$$

Artificial Neural Network

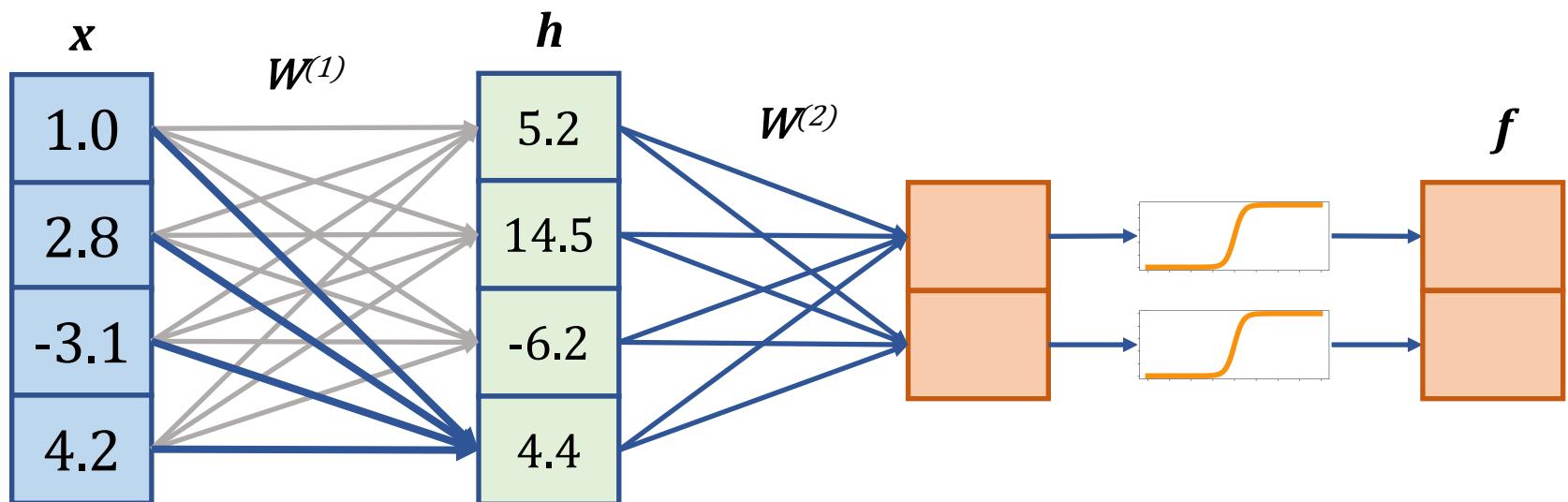
Example



$$\begin{aligned}x^T \times W^{(1)} &= [1.0 \quad 2.8 \quad -3.1 \quad 4.2]_{1 \times 4} \times \begin{bmatrix} 2.5 & -1.3 & \boxed{-0.2} & 0.9 \\ 5.0 & 3.0 & \boxed{-1.5} & 1.1 \\ 2.3 & -1.7 & 3.3 & -0.4 \\ -.01 & 0.5 & 2.0 & -0.2 \end{bmatrix}_{4 \times 4} \\&= [5.2 \quad 14.5 \quad -6.2]_{1 \times 4}\end{aligned}$$

Artificial Neural Network

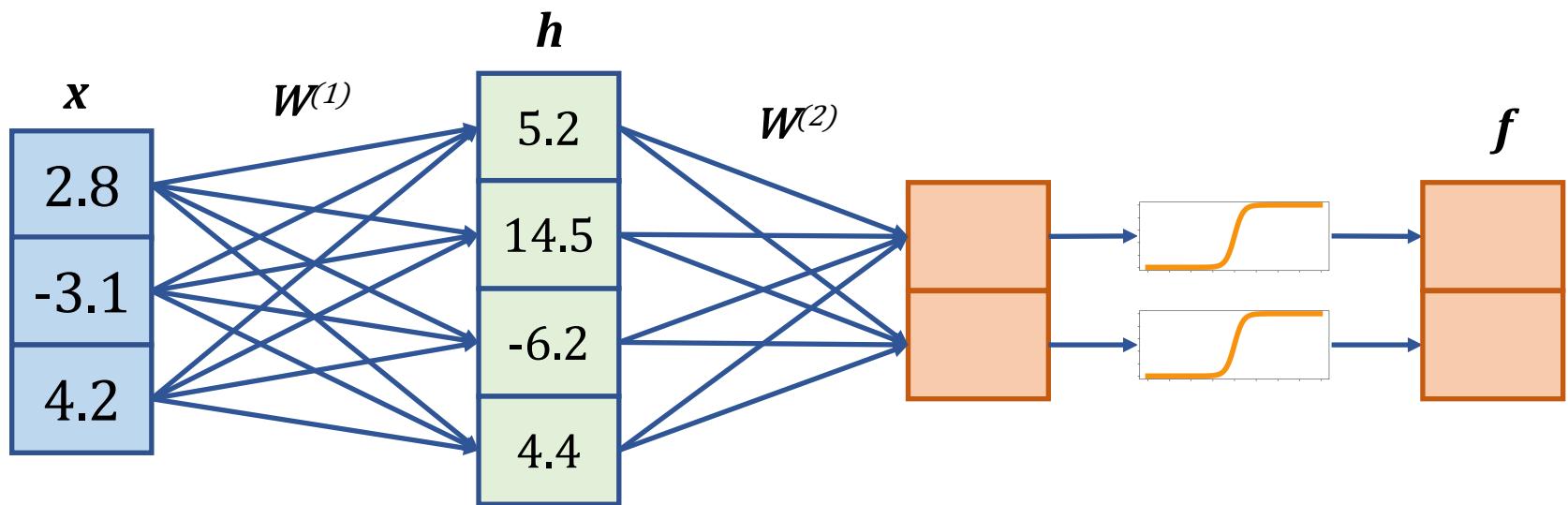
Example



$$\begin{aligned}x^T \times W^{(1)} &= [1.0 \quad 2.8 \quad -3.1 \quad 4.2]_{1 \times 4} \times \begin{bmatrix} 2.5 & -1.3 & -0.2 & 0.9 \\ 5.0 & 3.0 & -1.5 & 1.1 \\ 2.3 & -1.7 & 3.3 & -0.4 \\ -.01 & 0.5 & 2.0 & -0.2 \end{bmatrix}_{4 \times 4} \\&= [5.2 \quad 14.5 \quad -6.2 \quad 4.4]_{1 \times 4}\end{aligned}$$

Artificial Neural Network

Example

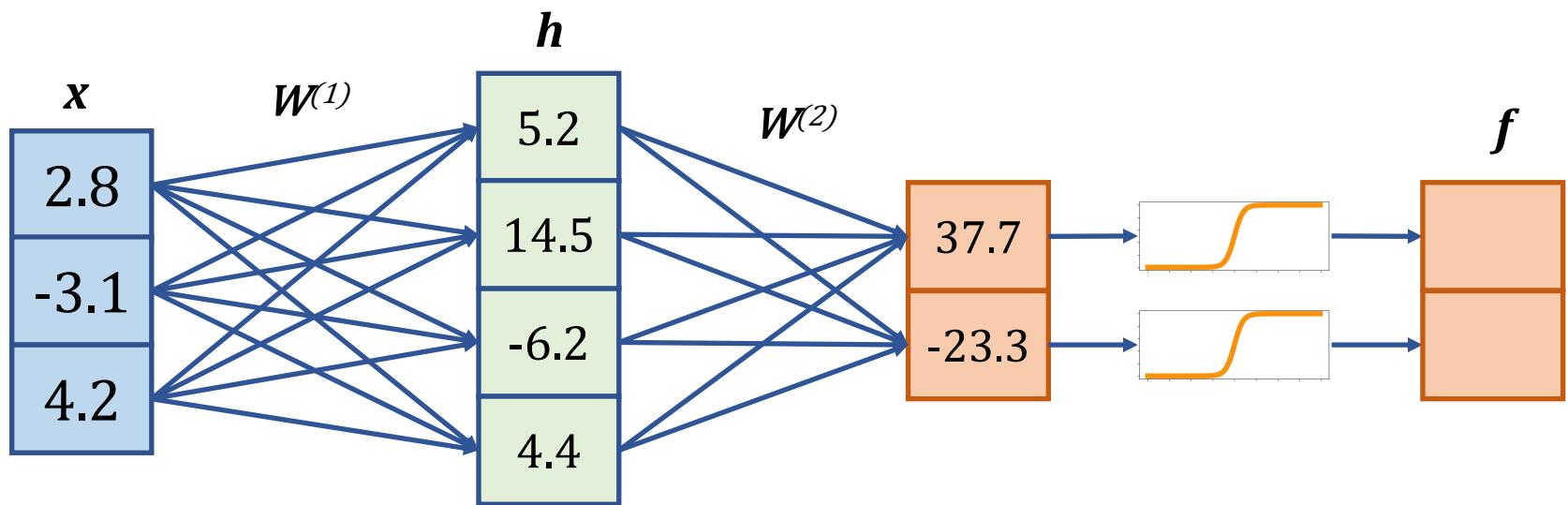


$$h = [1.0 \quad 5.2 \quad 14.5 \quad -6.2 \quad 4.4]_{1 \times 5}$$

$$W^{(2)} = \begin{bmatrix} 0.6 & -1.7 \\ 1.9 & 3.0 \\ 0.6 & -1.2 \\ -1.3 & 2.6 \\ 2.4 & -0.8 \end{bmatrix}_{5 \times 2}$$

Artificial Neural Network

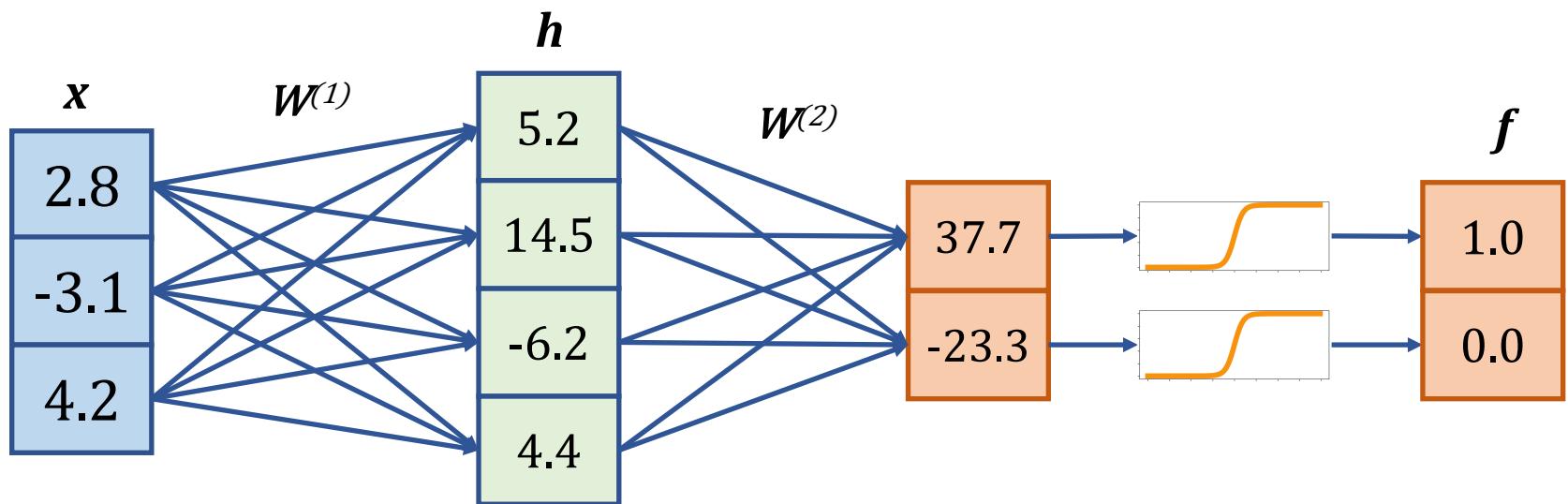
Example



$$\begin{aligned} h \times W^{(2)} &= [1.0 \quad 5.2 \quad 14.5 \quad -6.2 \quad 4.4]_{1 \times 5} \times \begin{bmatrix} 0.6 & -1.7 \\ 1.9 & 3.0 \\ 0.6 & -1.2 \\ -1.3 & 2.6 \\ 2.4 & -0.8 \end{bmatrix}_{5 \times 2} \\ &= [37.7 \quad -23.3]_{1 \times 2} \end{aligned}$$

Artificial Neural Network

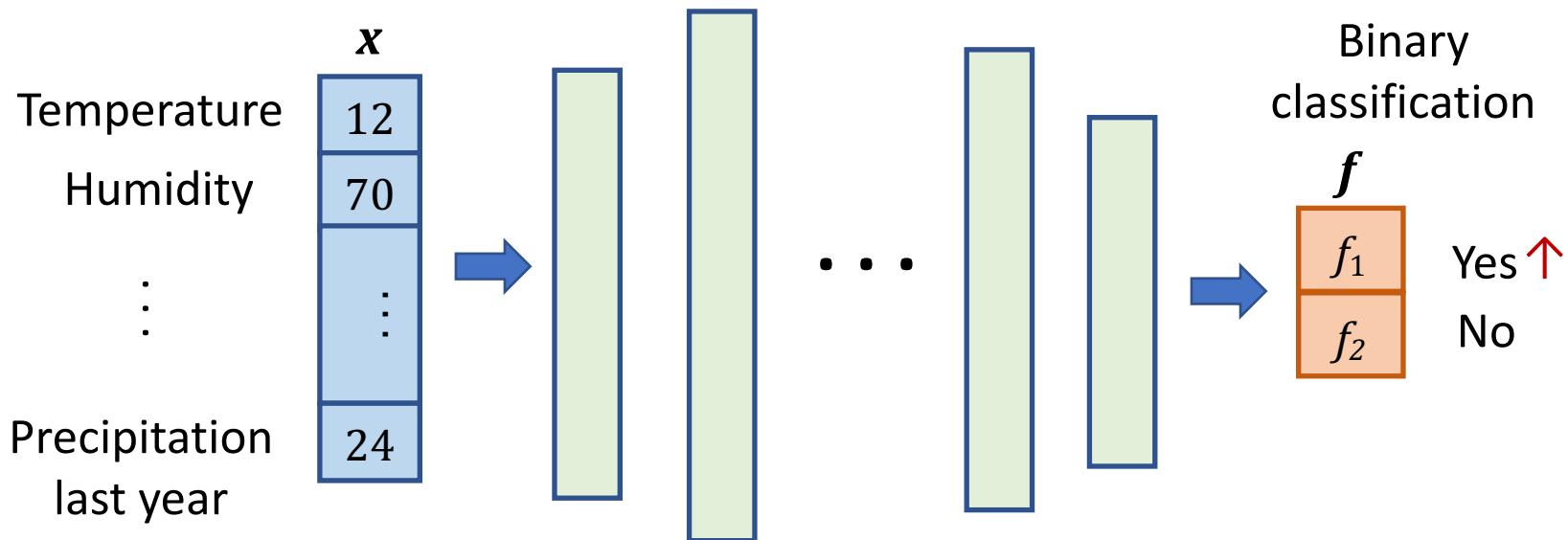
Example



Artificial Neural Network

Example

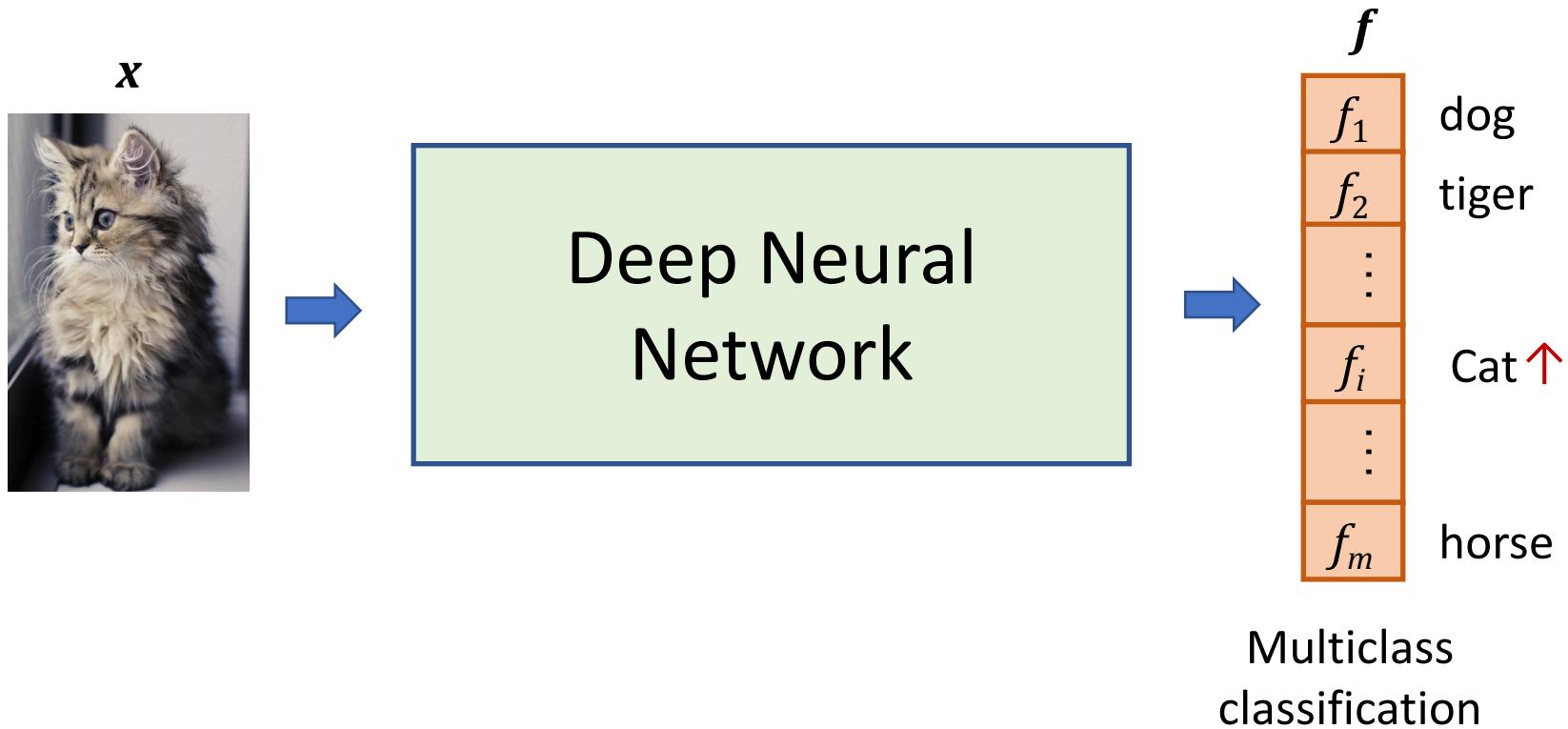
- A **classification** task – Is it going to rain today?



Artificial Neural Network

Example

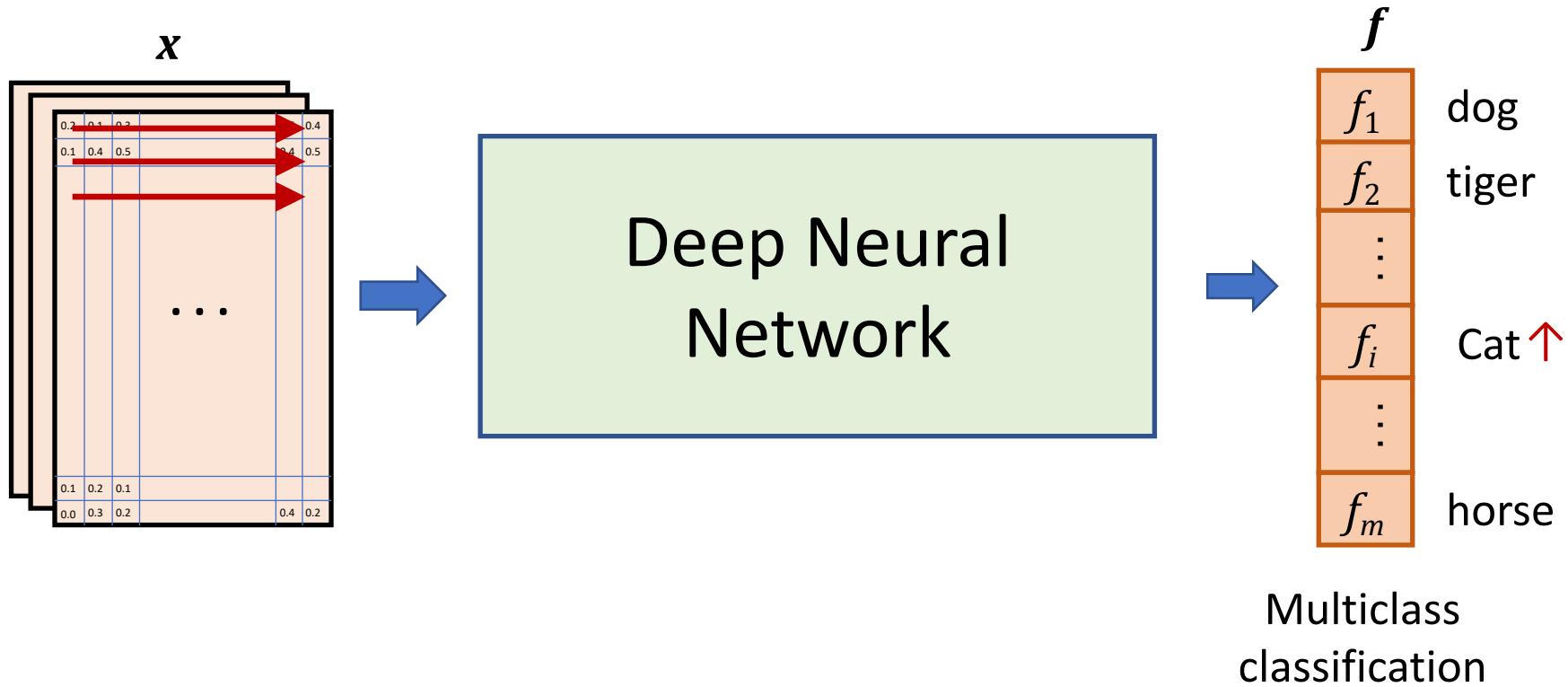
- A **classification** task – What animal is in this picture?



Artificial Neural Network

Example

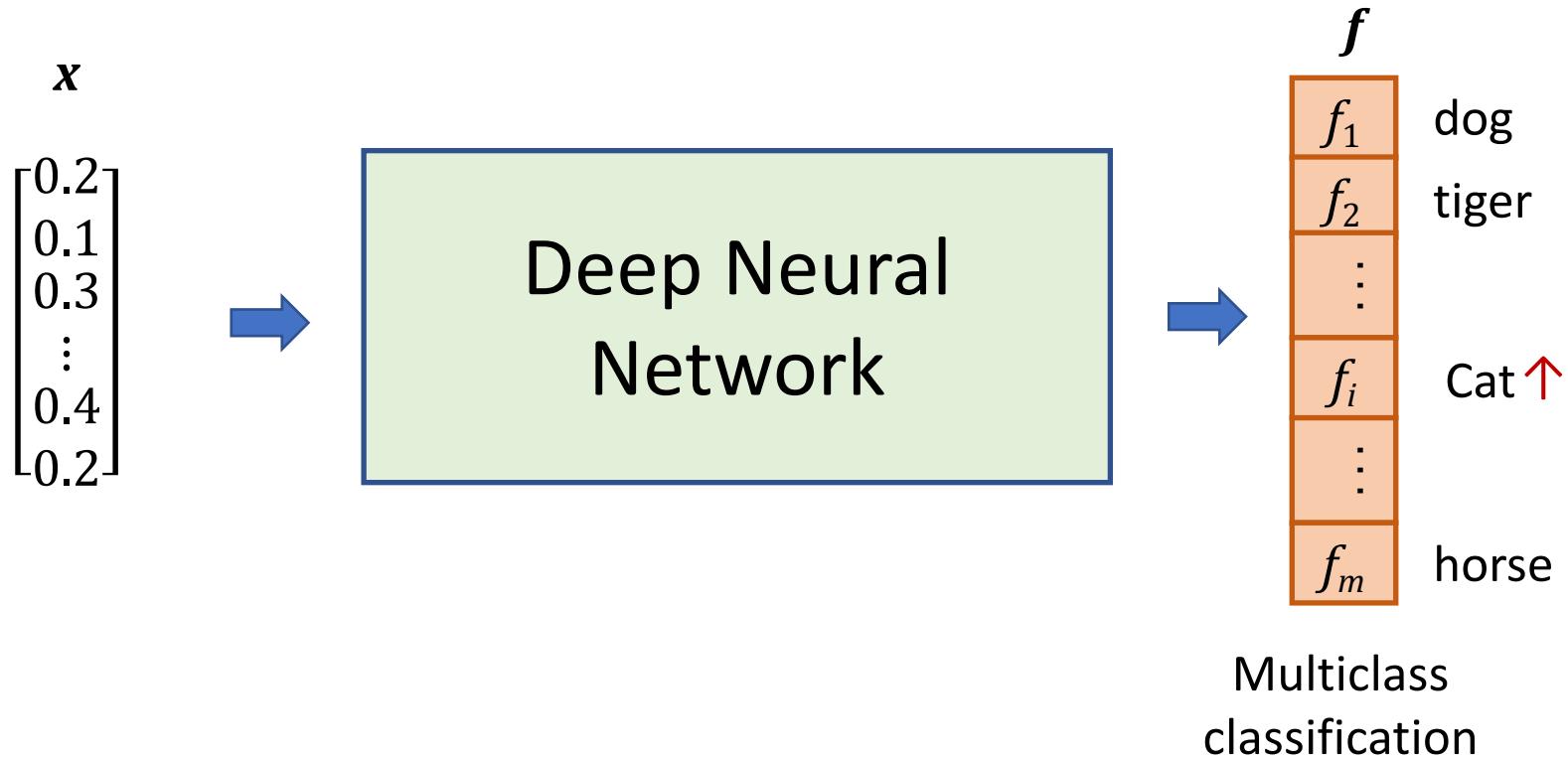
- A **classification** task – What animal is in this picture?



Artificial Neural Network

Example

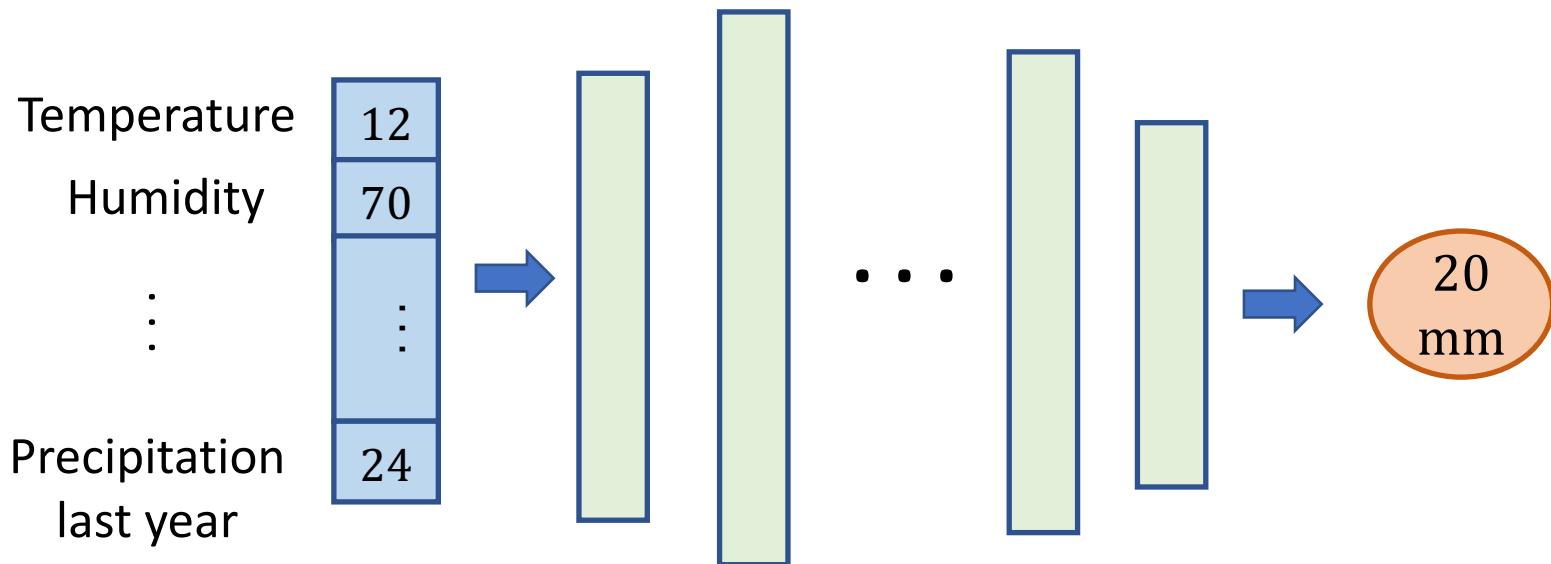
- A **classification** task – What animal is in this picture?



Artificial Neural Network

Example

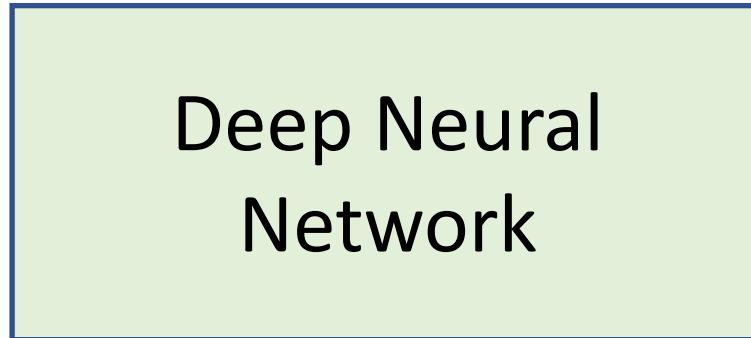
- A **regression task** – How much is it going to rain today?



Artificial Neural Network

Example

- A **regression** task – How much is the price of this house?



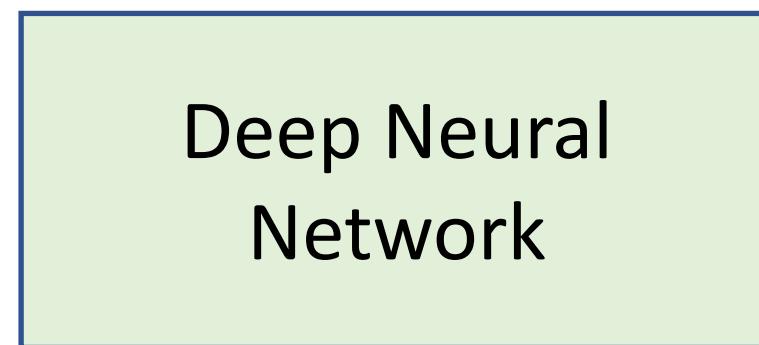
An orange oval with a thin orange border. Inside the oval, the text "£ 800,000" is written in a black, sans-serif font, representing the predicted price.

Network Training

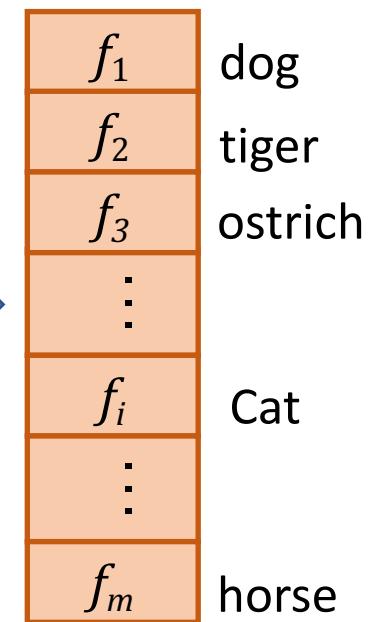
(Loss Function)

Network Training

How the network *learns* to predict the class labels

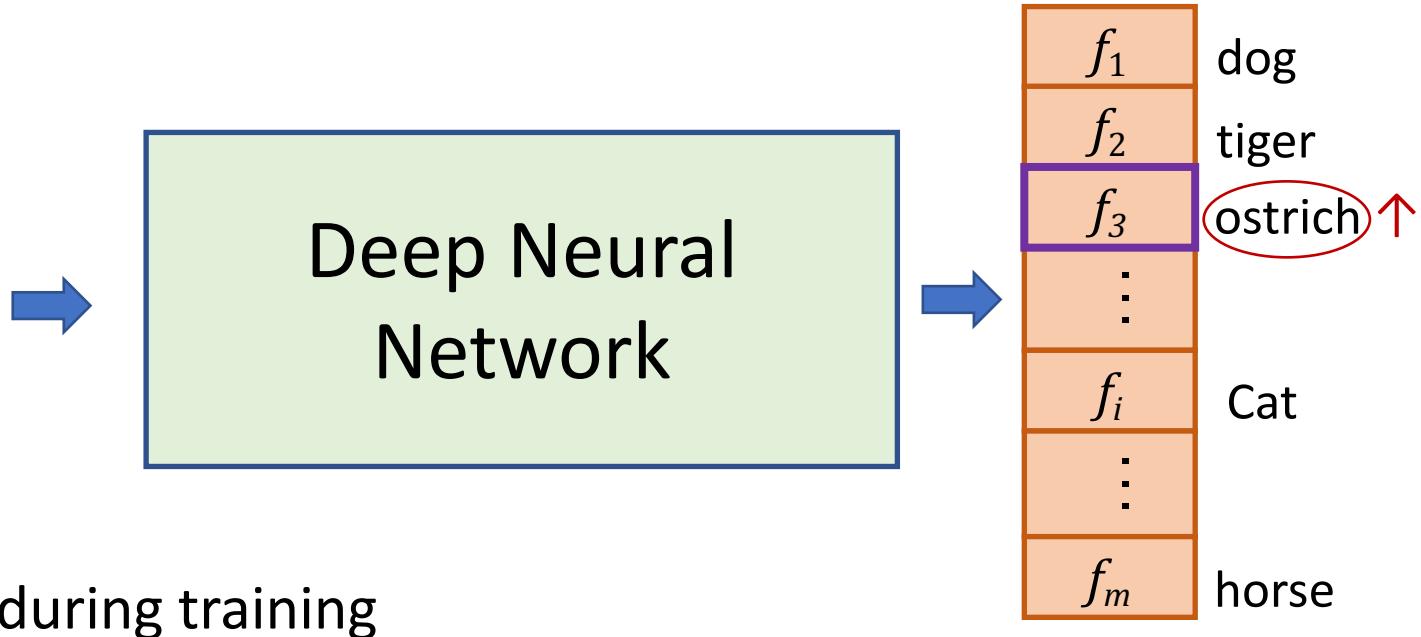


Classification example



Network Training

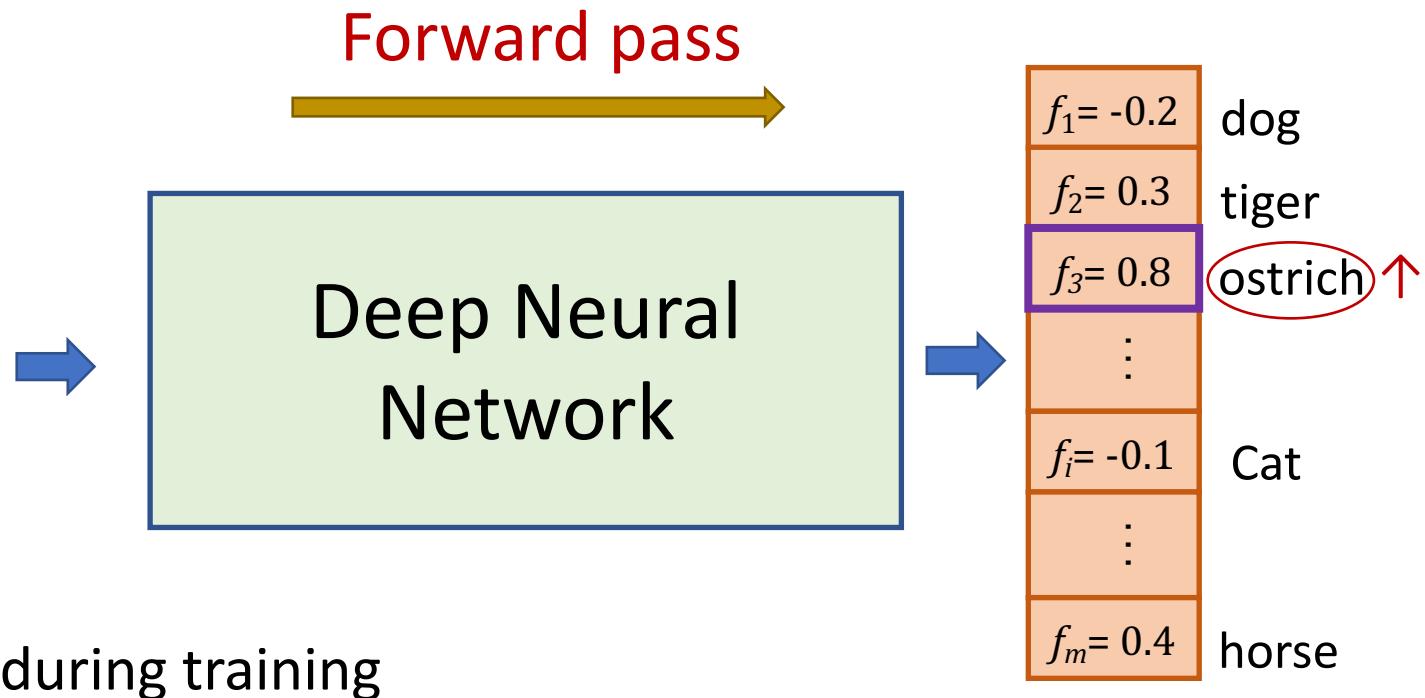
Before being trained, our network has no idea what this image is!



Optimisation during training

1. It makes a prediction using its current parameter set (weights).
2. We give it a feedback of how good its prediction was.
3. It updates (refines) its parameter set according to our feedback

Network Training



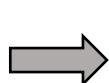
Optimisation during training

1. It makes a prediction using its current parameter set (weights).
2. We give it a feedback of how good its prediction was.
3. It updates (refines) its parameter set according to our feedback

Network Training

Loss function

- Compares network predictions against ground-truth
- Quantifies how good the network predictions are



Our objective will be
to minimise the loss

Network predictions	Ground truth
f_1	dog
f_2	tiger
f_3	ostrich
:	⋮
f_i	Cat
:	⋮
f_m	horse

Optimisation during training

1. It makes a prediction using its current parameter set (weights).
2. We give it a feedback of how good its prediction was.
3. It updates (refines) its parameter set according to our feedback

Network Training

Cross-entropy Loss function

One of the most common loss functions for classification

In general, cross-entropy measures the difference between a target distribution (p) and an estimated distribution (q)

$$H(p, q) = - \sum_x p(x) \log(q(x))$$

Network predictions	Ground truth
f_1	dog
f_2	tiger
f_3	ostrich
:	:
f_i	Cat
:	
f_m	horse

Network Training

Cross-entropy Loss function

- $q \rightarrow$ the estimated class **probabilities**
- $p \rightarrow$ the **probability** of the correct class is 1, others are 0

$$H(p, q) = - \sum_x p(x) \log(q(x))$$

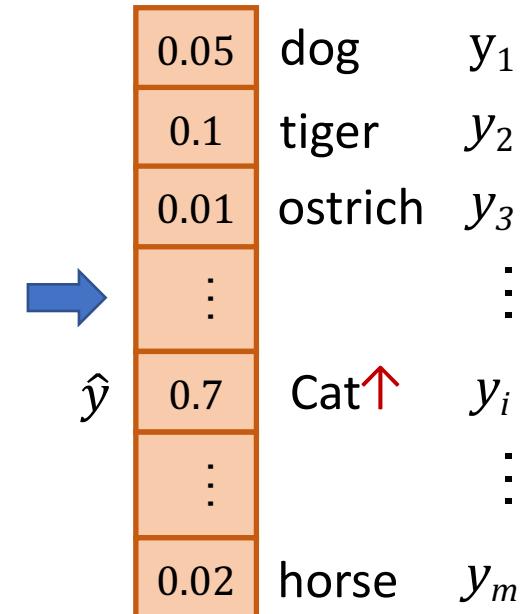
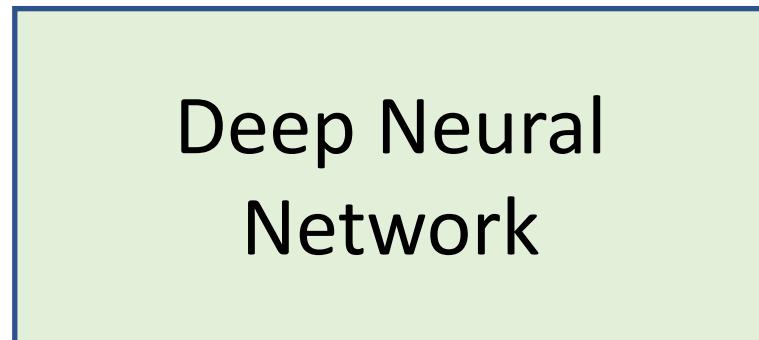
Network predictions	Ground truth
f_1	dog
f_2	tiger
f_3	ostrich
:	:
f_i	Cat
:	
f_m	horse
$\uparrow q$	$\uparrow p$
(estimated dist.)	(target dist.)

But how do we change our network output (f) to a probability distribution?

Network Training

Probabilistic interpretation for classification

$$\hat{y} = \operatorname{argmax}_{y_i} p(Y=y_i | \mathbf{x}), \text{ where } Y \text{ is the label set}$$



and $y_i \in Y$
 $0 \leq p(Y=y_i | \mathbf{x}) \leq 1$
 $\sum_i p(Y=y_i | \mathbf{x})=1$

Network Training

Softmax function

Input: An arbitrary real-valued vector

Output: A vector with values between 0 and 1, that all sum to 1

$$s_i = p(Y=y_i \mid \mathbf{x}; \mathbf{W}) = \frac{e^{f_{y_i}}}{\sum_j e^{f_{y_j}}}$$

$$p(Y=y_1 \mid x; W) = \frac{e^{-0.2}}{e^{-0.2} + e^{0.3} + e^{0.8} + \dots + e^{-0.1} + \dots + e^{0.4}}$$

Probability of the image to be a dog

Network predictions	Ground truth
$f_1 = -0.2$	dog
$f_2 = 0.3$	tiger
$f_3 = 0.8$	ostrich
:	:
$f_i = -0.1$	Cat
:	:
$f_m = 0.4$	horse

Network Training

Softmax function

Input: An arbitrary real-valued vector

Output: A vector with values between 0 and 1, that all sum to 1

$$s_i = p(Y=y_i | \mathbf{x}; \mathbf{W}) = \frac{e^{f_{y_i}}}{\sum_j e^{f_{y_j}}}$$

$$p(Y=y_1 | \mathbf{x}; \mathbf{W}) = \frac{e^{-0.2}}{e^{-0.2} + e^{0.3} + e^{0.8} + \dots + e^{-0.1} + \dots + e^{0.4}}$$



Probability of the image to be a dog

Network predictions	Ground truth
$f_1 = -0.2$	dog
$f_2 = 0.3$	tiger
$f_3 = 0.8$	ostrich
:	:
$f_i = -0.1$	Cat
:	
$f_m = 0.4$	horse

Network Training

Softmax function

Input: An arbitrary real-valued vector

Output: A vector with values between 0 and 1, that all sum to 1

$$s_i = p(Y=y_i \mid x; W) = \frac{e^{f_{y_i}}}{\sum_j e^{f_{y_j}}}$$

Network predictions	Ground truth
$f_1 = -0.2$	dog
$f_2 = 0.3$	tiger
$f_3 = 0.8$	ostrich
:	:
$f_i = -0.1$	Cat
:	:
$f_m = 0.4$	horse

$$p(Y=y_1 | x; W) = \frac{e^{-0.2}}{e^{-0.2} + e^{0.3} + e^{0.8} + \dots + e^{-0.1} + \dots + e^{0.4}}$$

Probability of the image to be a dog

Network Training

Softmax function

Input: An arbitrary real-valued vector

Output: A vector with values between 0 and 1, that all sum to 1

$$s_i = p(Y=y_i | \mathbf{x}; \mathbf{W}) = \frac{e^{f_{y_i}}}{\sum_j e^{f_{y_j}}}$$

$$f_{y_1} \leftarrow e^{-0.2}$$

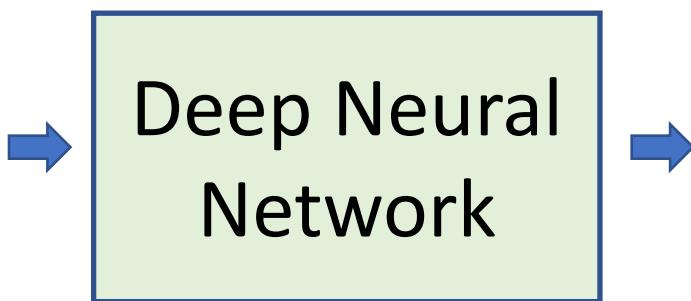
$$p(Y=y_1 | \mathbf{x}; \mathbf{W}) = \frac{e^{-0.2}}{e^{-0.2} + e^{0.3} + e^{0.8} + \dots + e^{-0.1} + \dots + e^{0.4}}$$

Probability of the image to be a dog

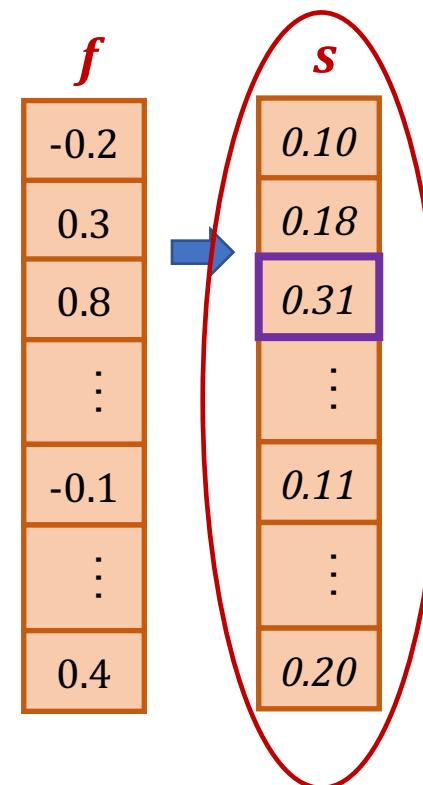
Network predictions	Ground truth
$f_1 = -0.2$	dog
$f_2 = 0.3$	tiger
$f_3 = 0.8$	ostrich
:	:
$f_i = -0.1$	Cat
:	:
$f_m = 0.4$	horse

Network Training

Softmax function



$$s_i = p(Y=y_i | \mathbf{x}; \mathbf{W}) = \frac{e^{f_{y_i}}}{\sum_j e^{f_{y_j}}}$$



Ground truth

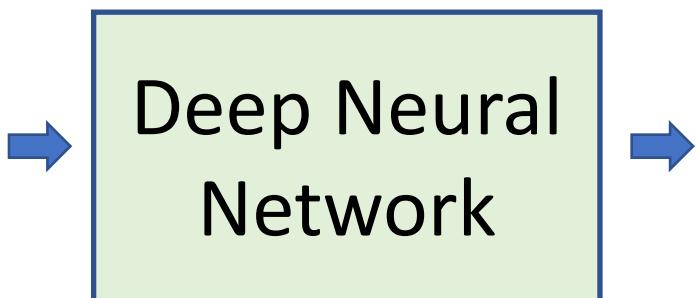
g

0	dog
0	tiger
0	ostrich
⋮	⋮
1	Cat
⋮	⋮
0	horse

The table shows the ground truth labels *g* for various animals. The row for "Cat" is highlighted with a green rectangle around the value 1.

Network Training

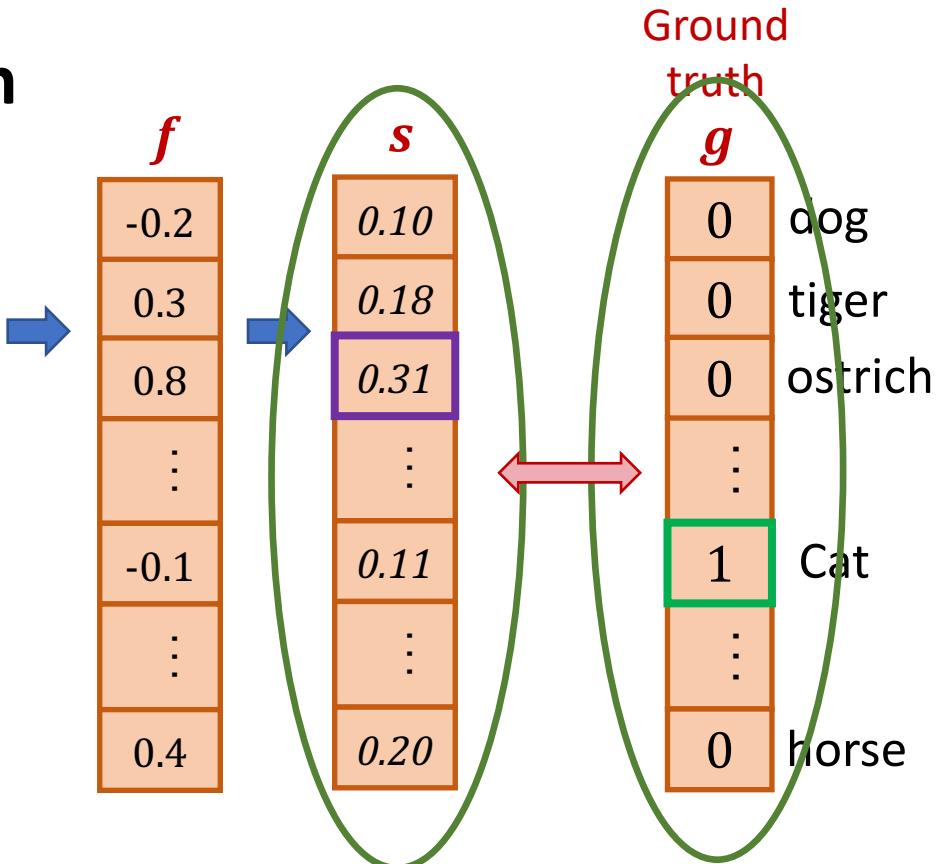
Cross-entropy Loss function



Now our cross-entropy loss is

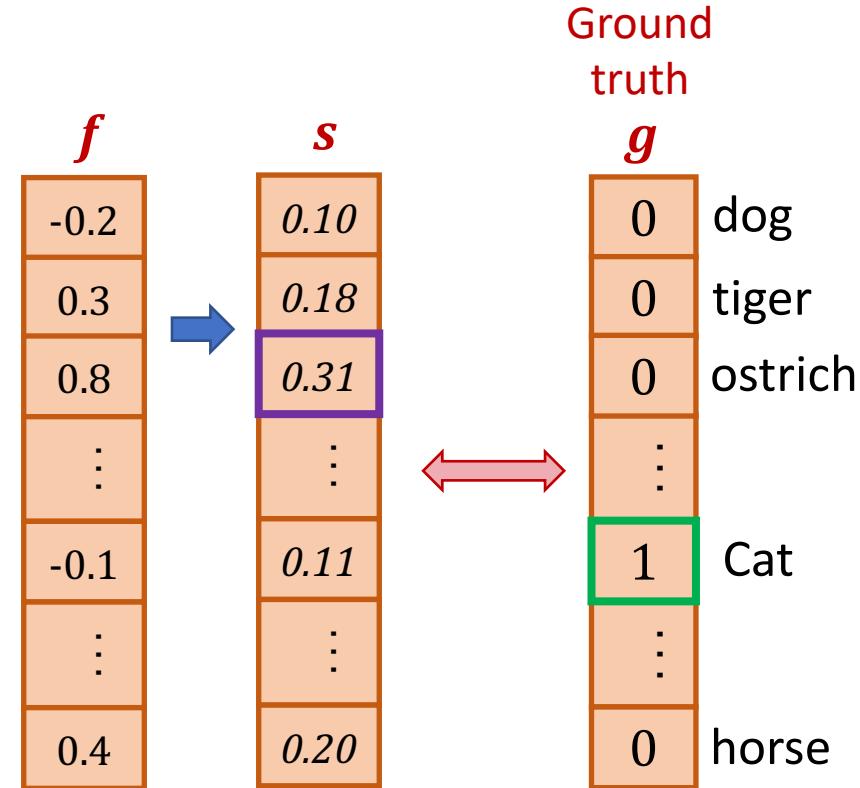
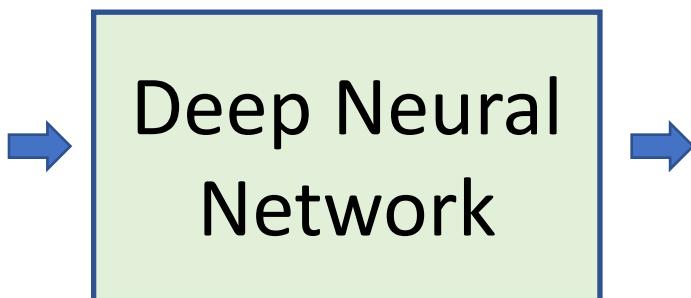
$$L = - \sum_i g_i \log(s_i)$$

$$L = -(0 \times \log 0.10 + 0 \times \log 0.18 + 0 \times \log 0.31 + \dots + 1 \times \log 0.11 + \dots + 0 \times \log 0.20)$$



Network Training

Cross-entropy Loss function



Now our cross-entropy loss is

$$L = - \sum_i g_i \log(s_i)$$

$$L = -(0 \times \log 0.10 + 0 \times \log 0.18 + 0 \times \log 0.31 + \dots + \textcircled{1} \times \log 0.11 + \dots + 0 \times \log 0.20)$$

Summary and Conclusion

- An artificial neuron is inspired by the biological neuron
- FC layer, where all the inputs are connected to the output neurons
- MLP is a neural network with stacked FC layers
- DNNs have many hidden layers
- ANNs have various applications such as classification and regressions
- A network loss measures the quality of its predictions

Next part

We will see how
the network parameters are
updated to minimise the loss

Thanks for your attention