An Introduction to Deep Learning

(part 1)

Farnoosh Heidarivincheh

EMAT31530 - February 2021

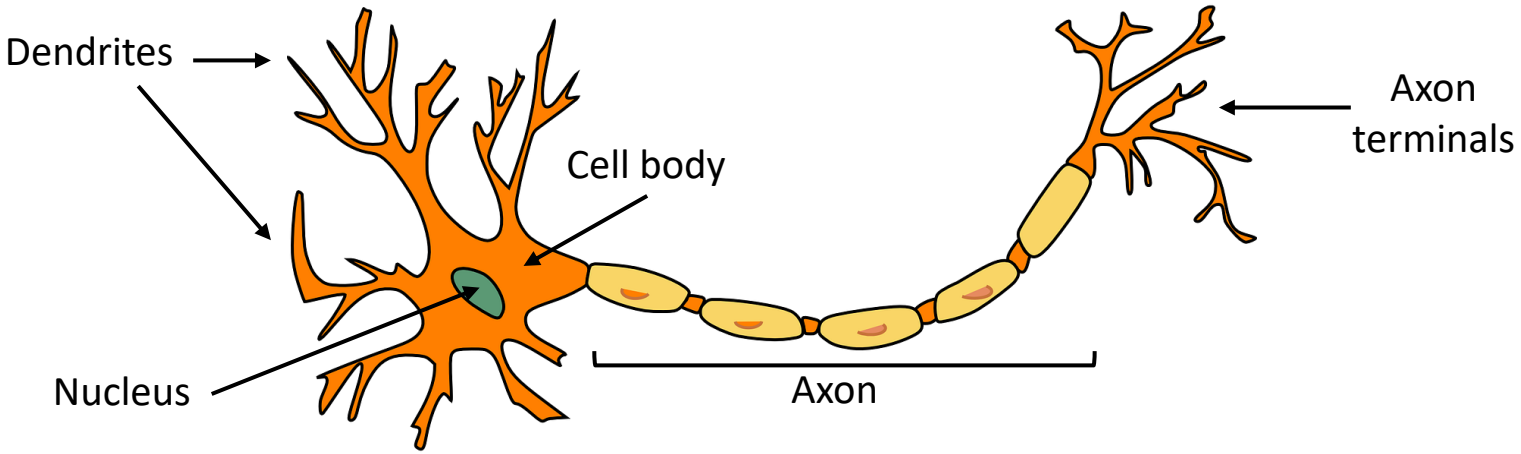
Outline

- 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

Input

Biological Neuron

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

Input

Output

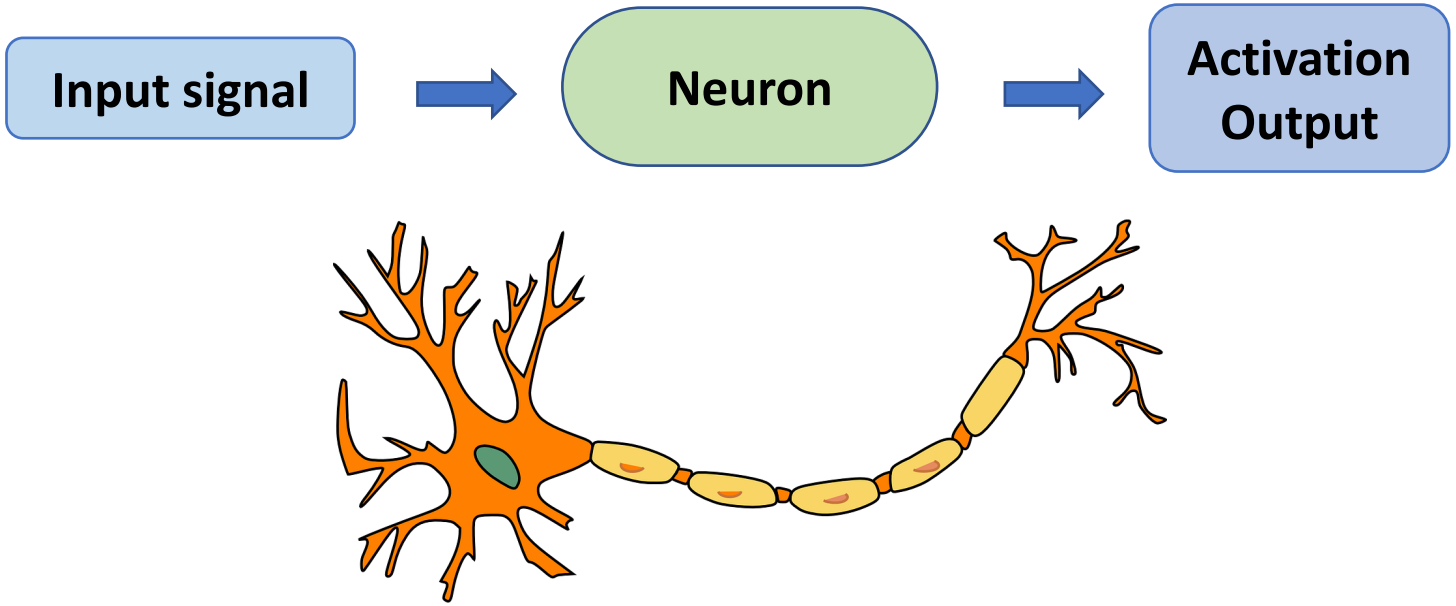
This is a dog

👍

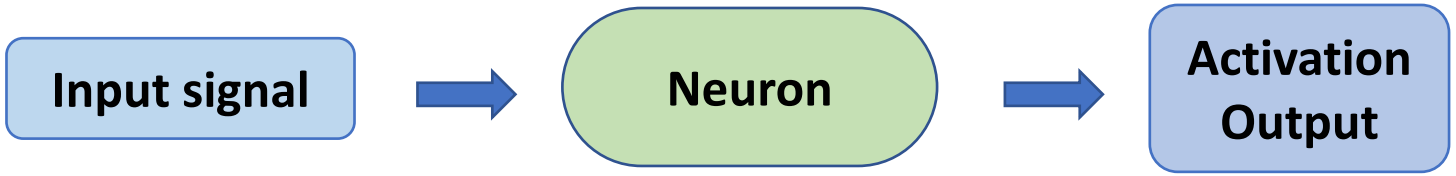
Artificial Neuron

Artificial Neuron

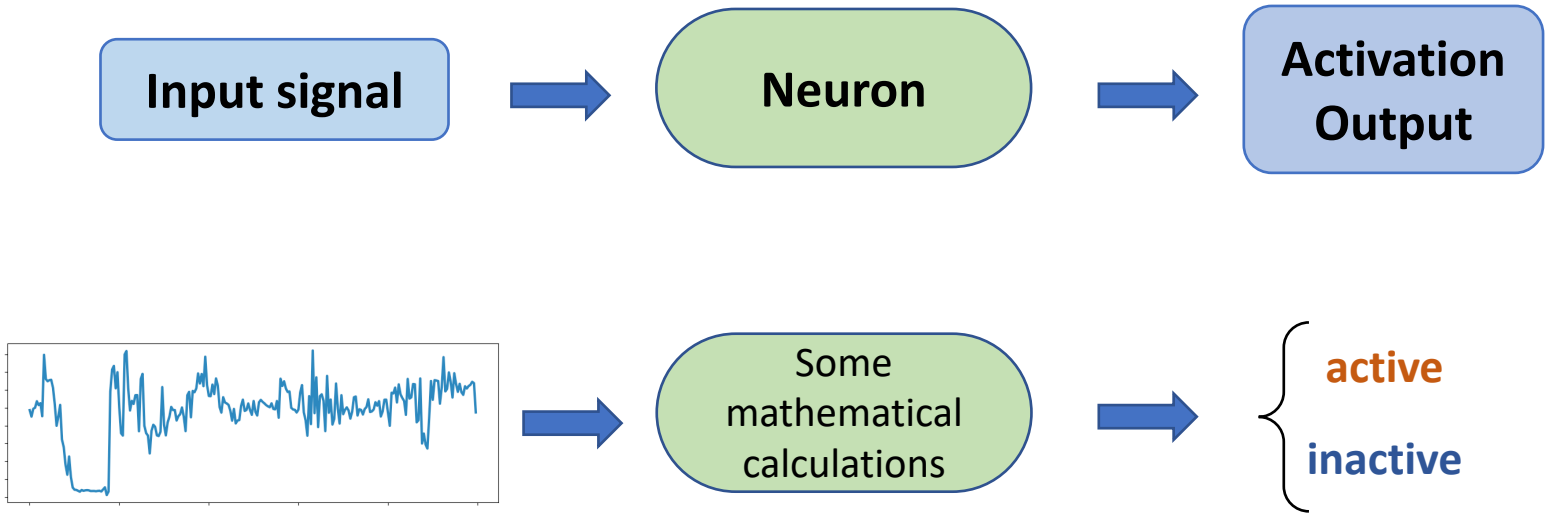
- An artificial neuron is the most basic unit of an Artificial Neural network
- It is inspired by the neurons in our brains



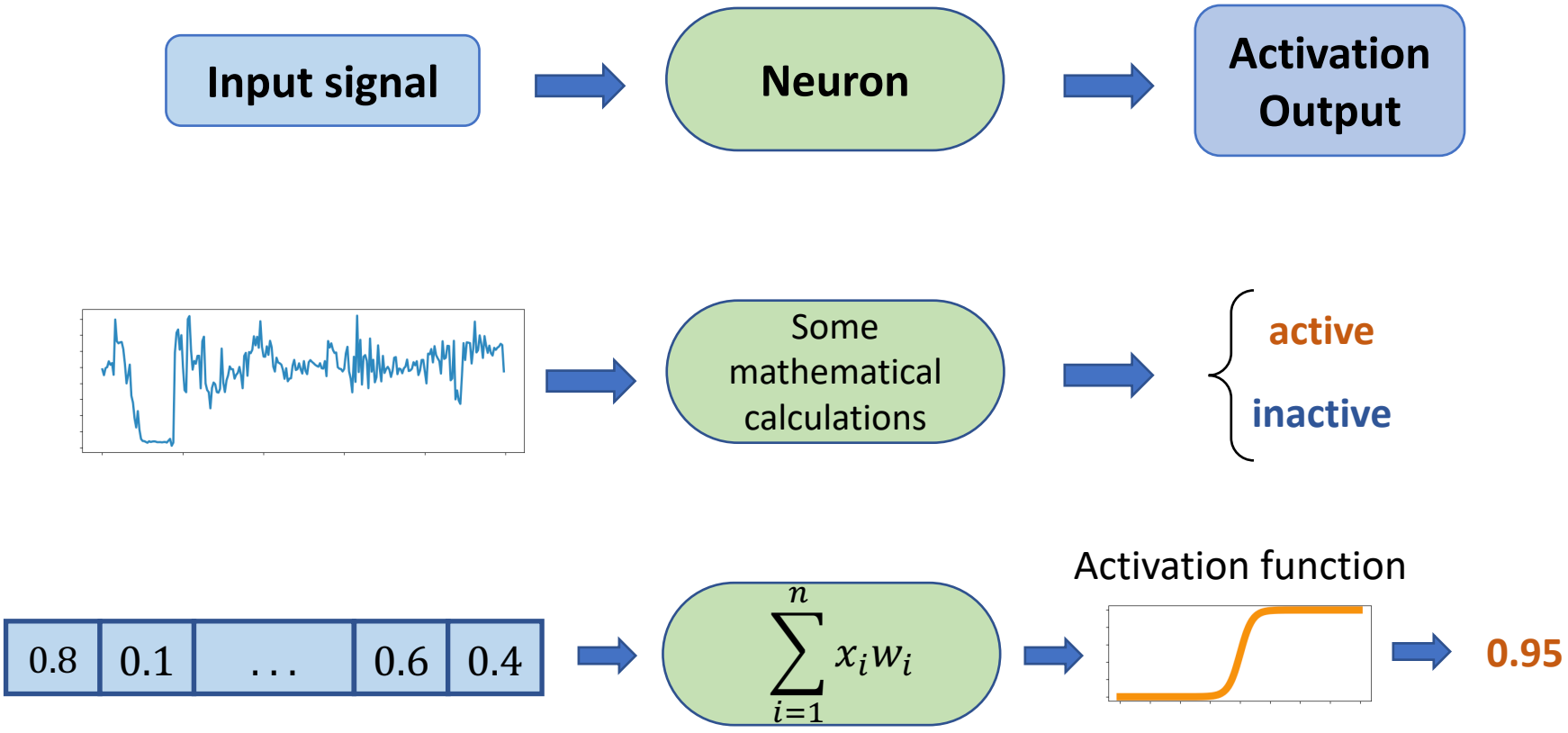
Artificial Neuron



Artificial Neuron

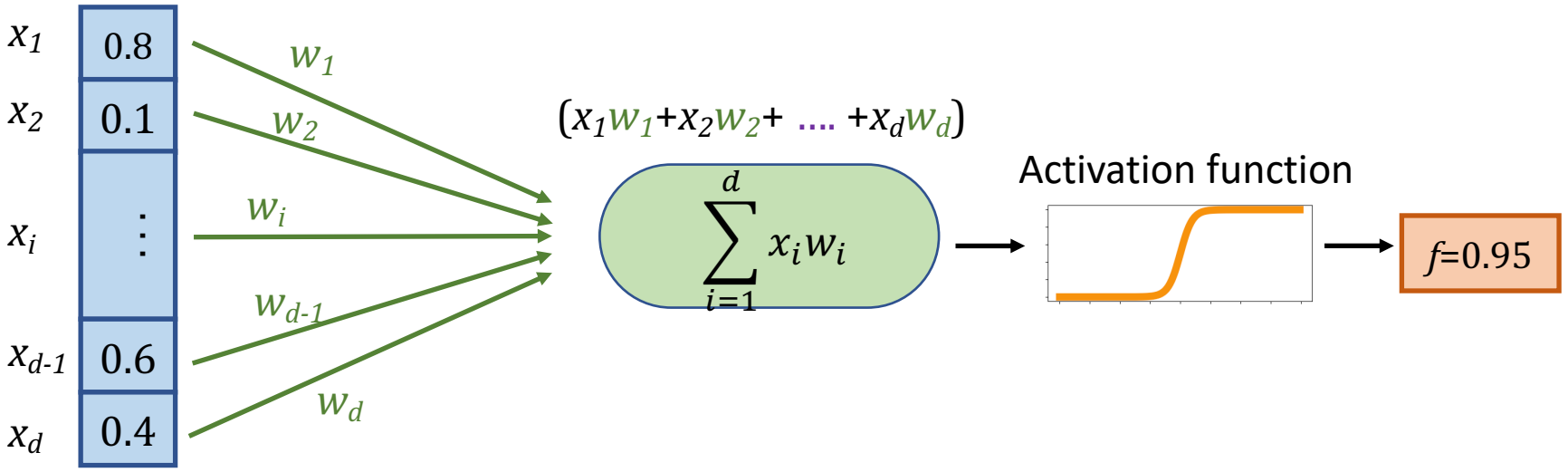


Artificial Neuron



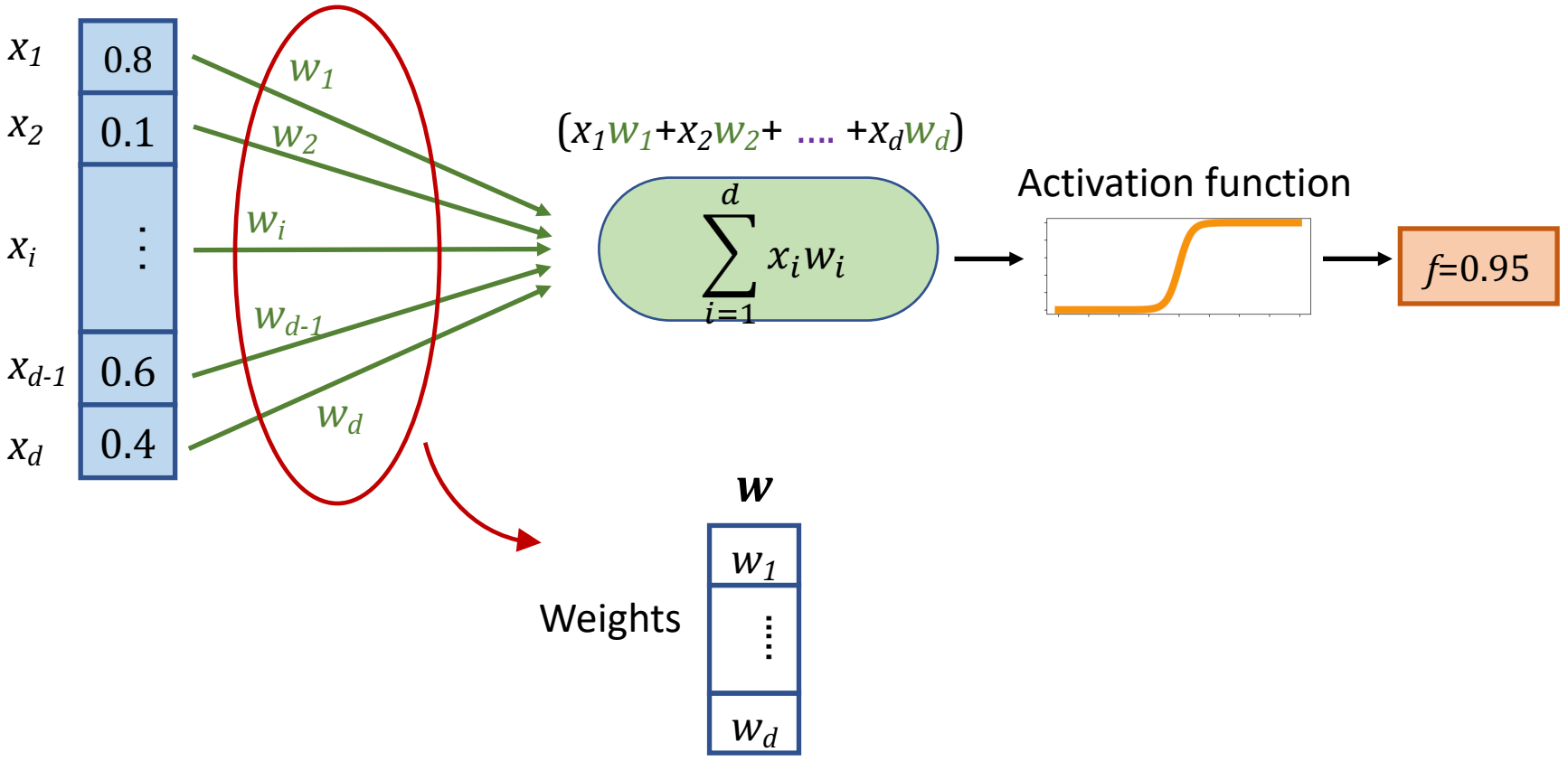
Artificial Neuron

Input vector $\boldsymbol{x}$



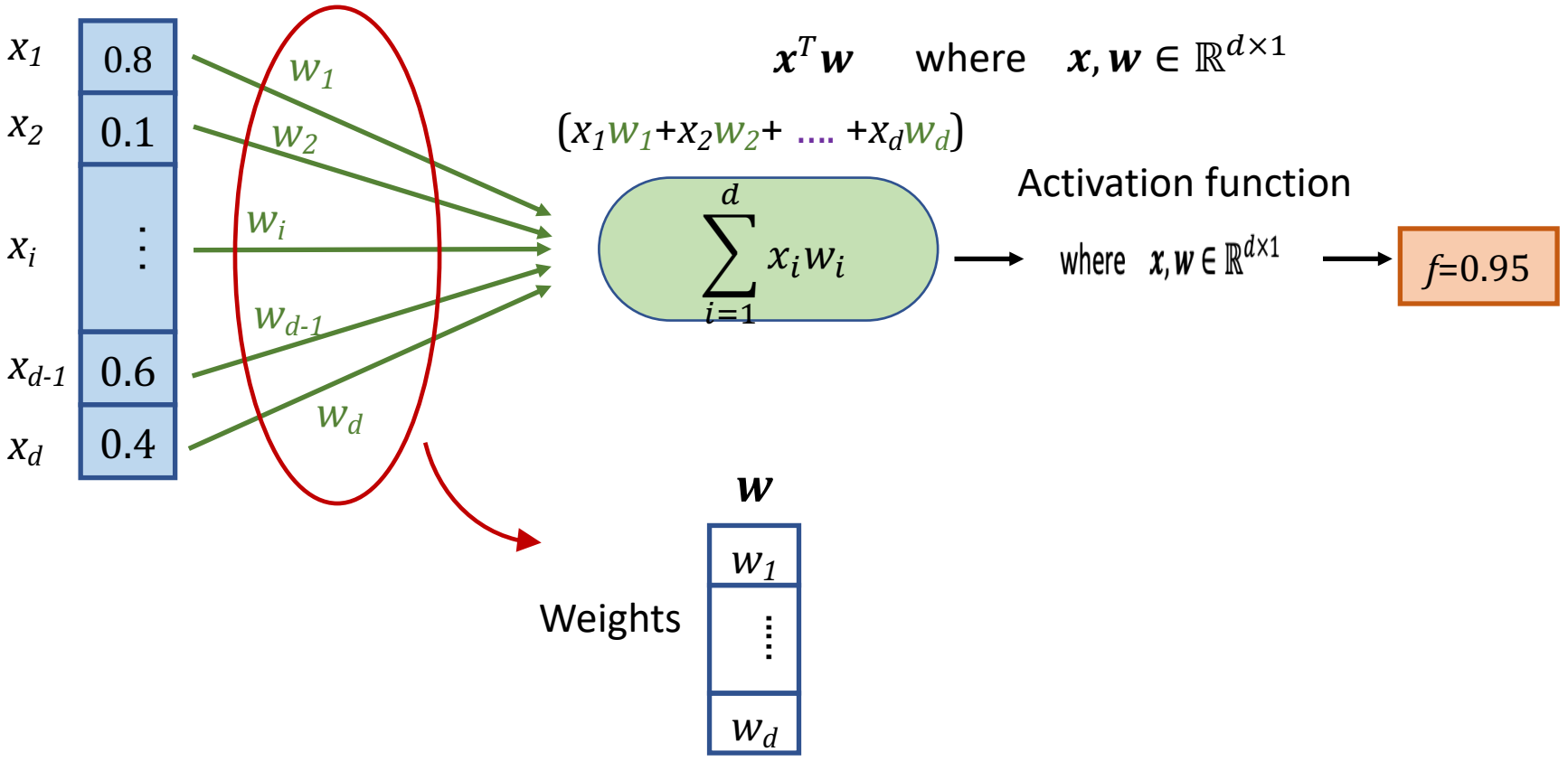
Artificial Neuron

Input vector $\boldsymbol{x}$



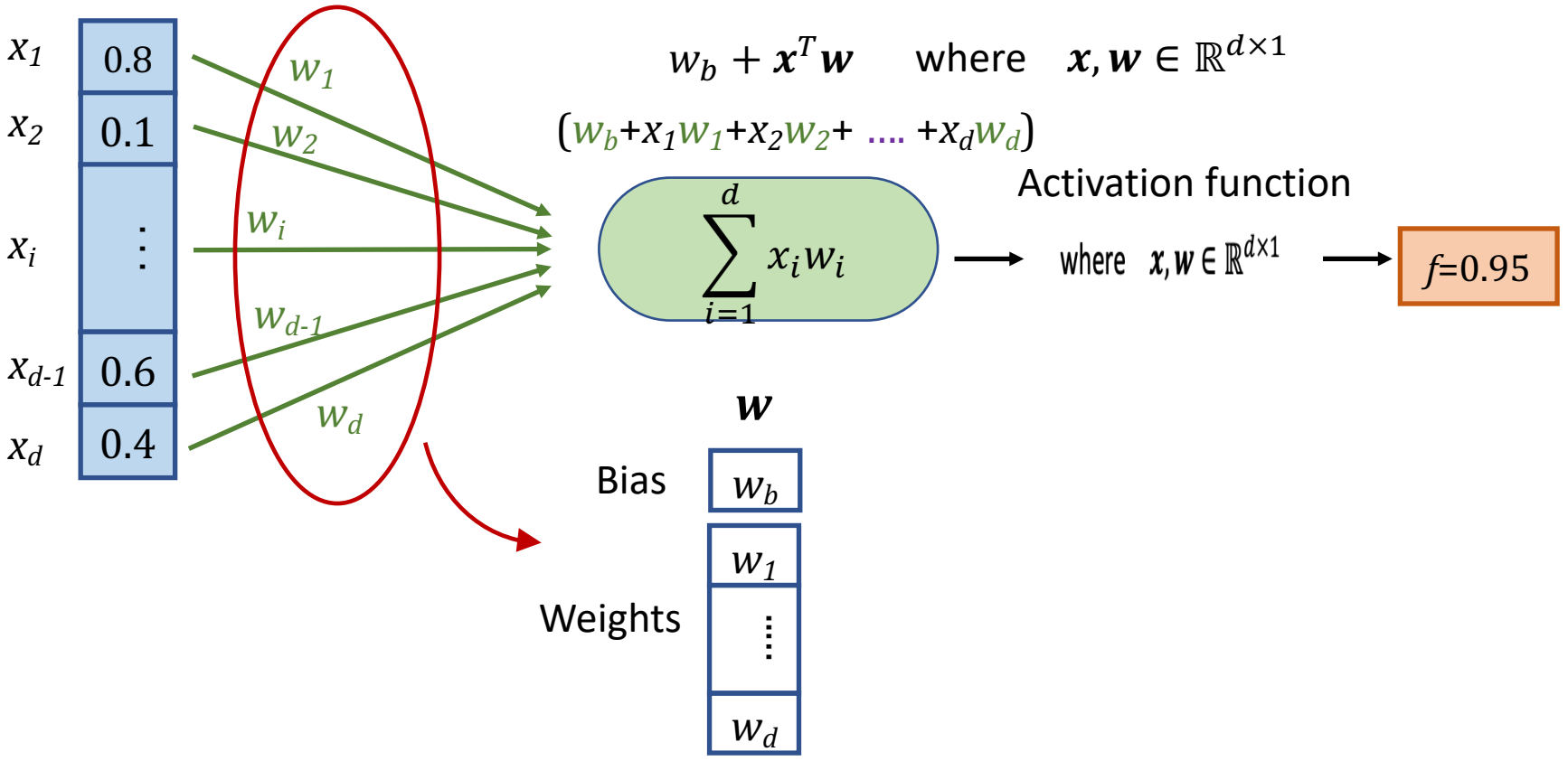
Artificial Neuron

Input vector $\boldsymbol{x}$

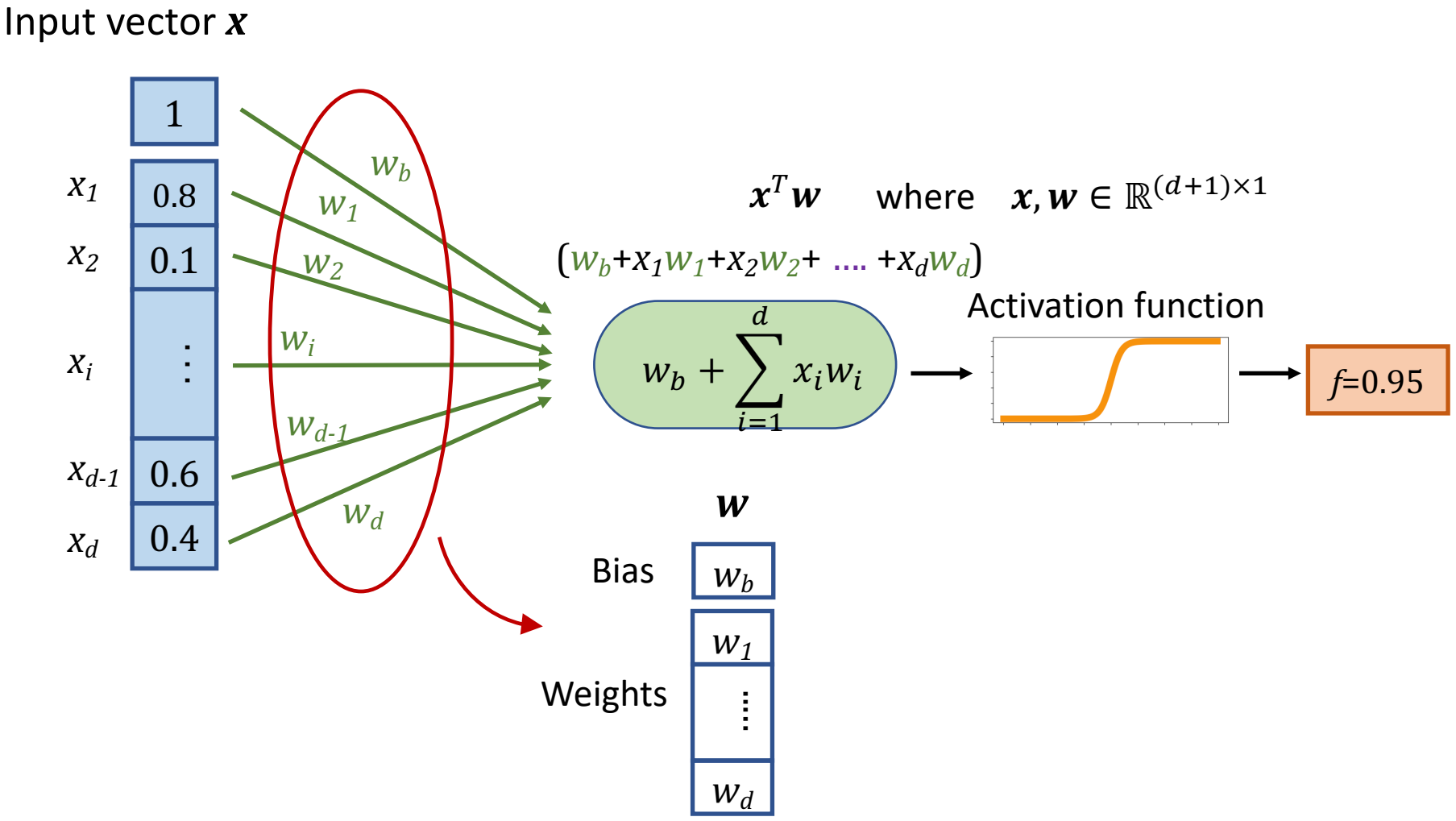


Artificial Neuron

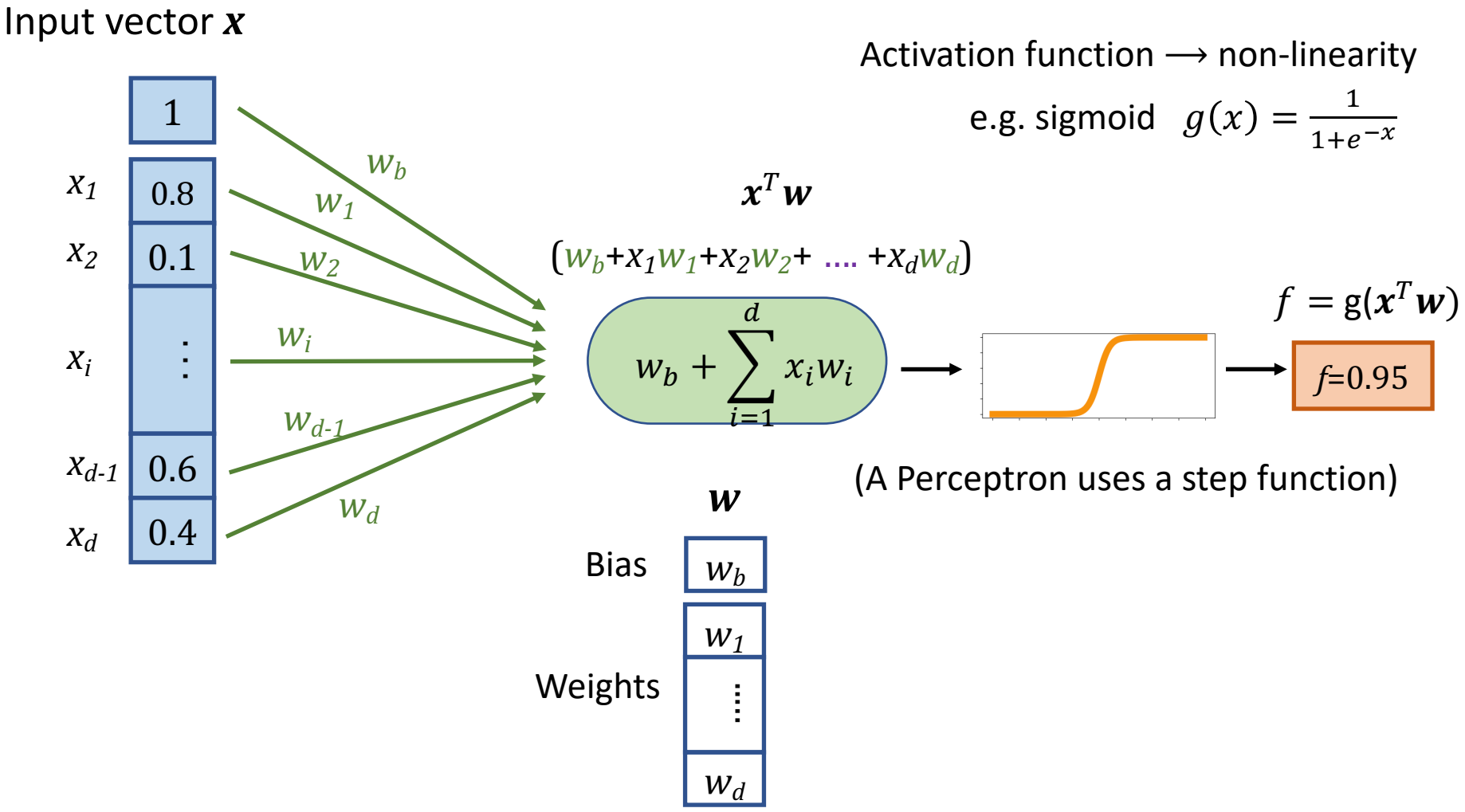
Input vector $\boldsymbol{x}$



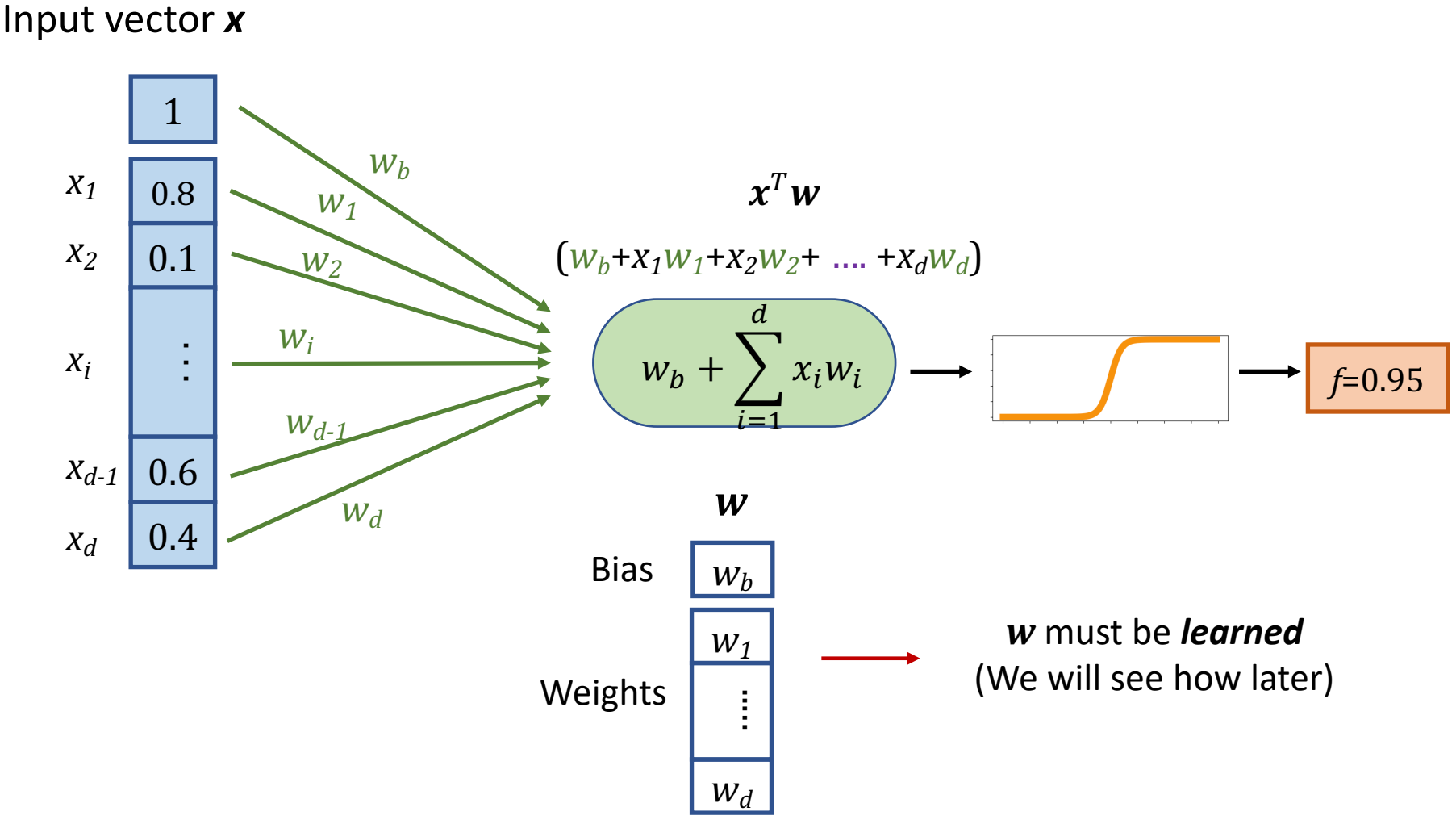
Artificial Neuron



Artificial Neuron



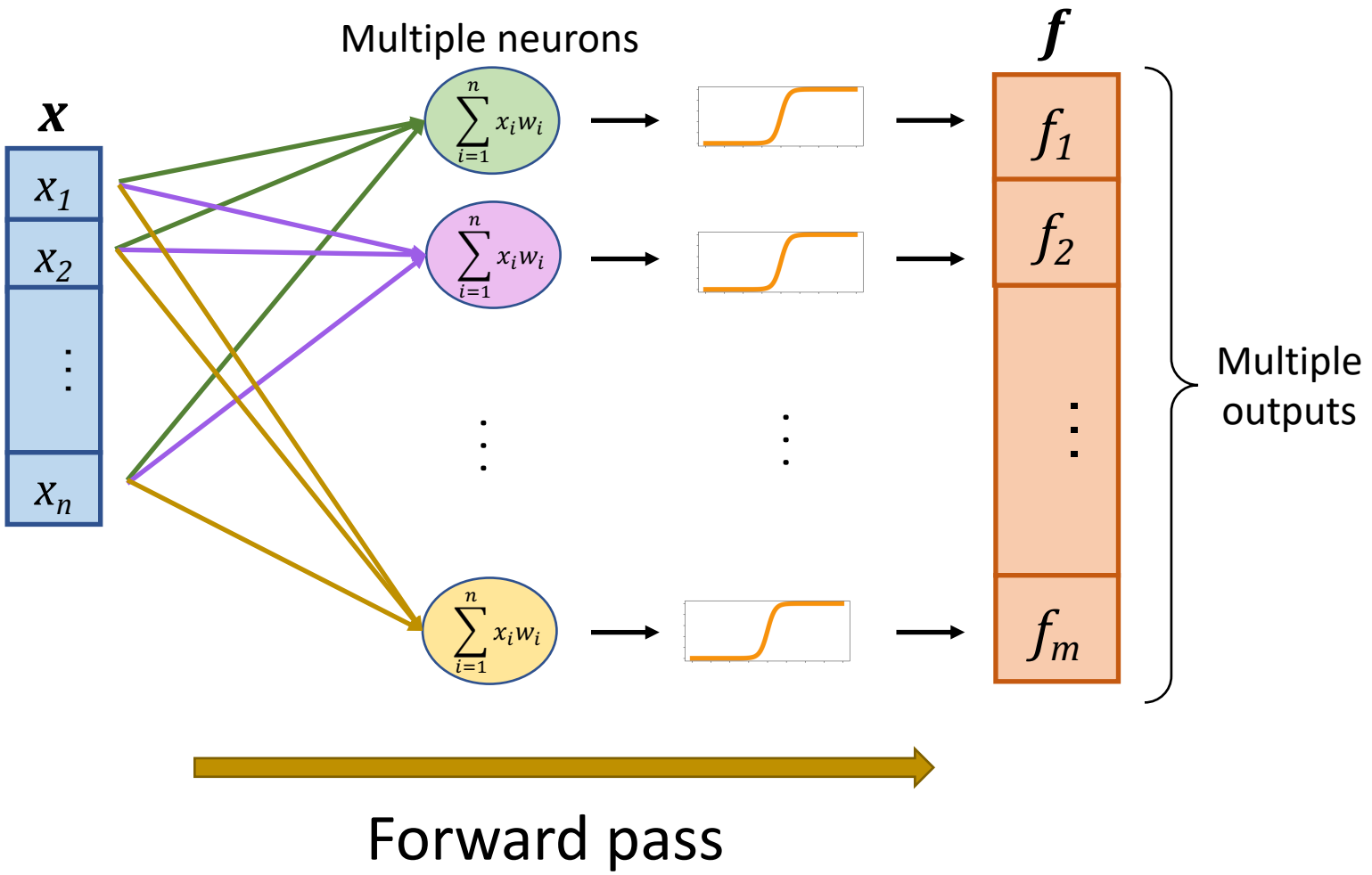
Artificial Neuron



Artificial Neural Network

Artificial Neural Network

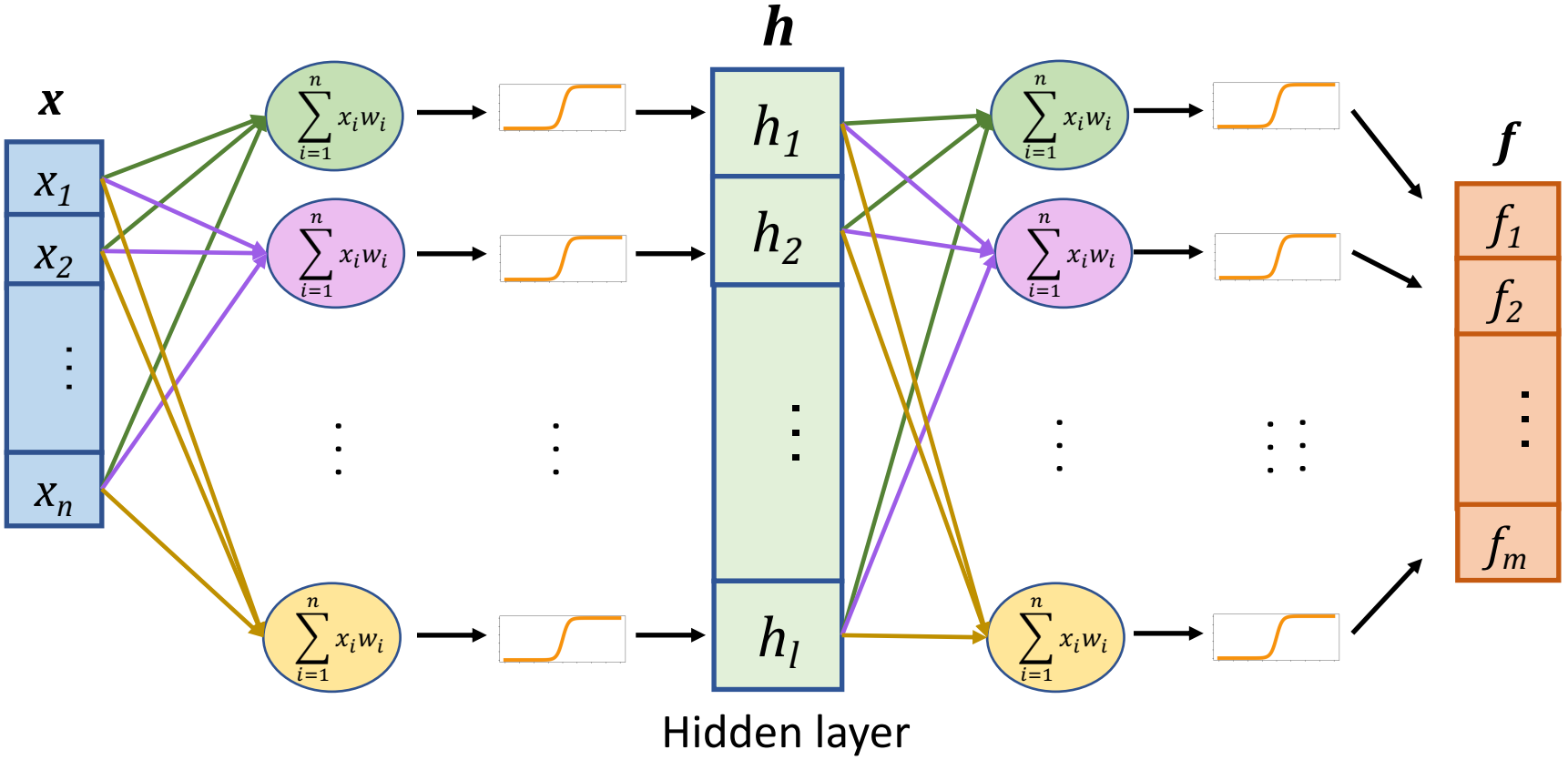
A Fully-Connected (FC/Dense) layer



Artificial Neural Network

A single-layer perceptron

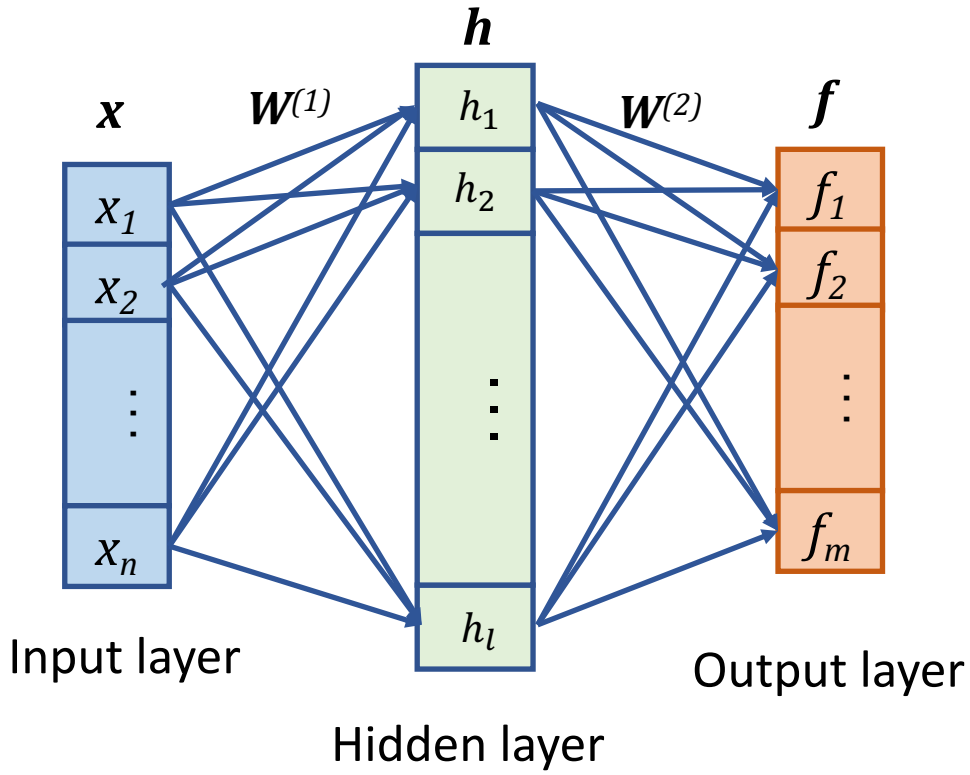
- We add a hidden layer between input and output layers



Artificial Neural Network

A single-layer perceptron

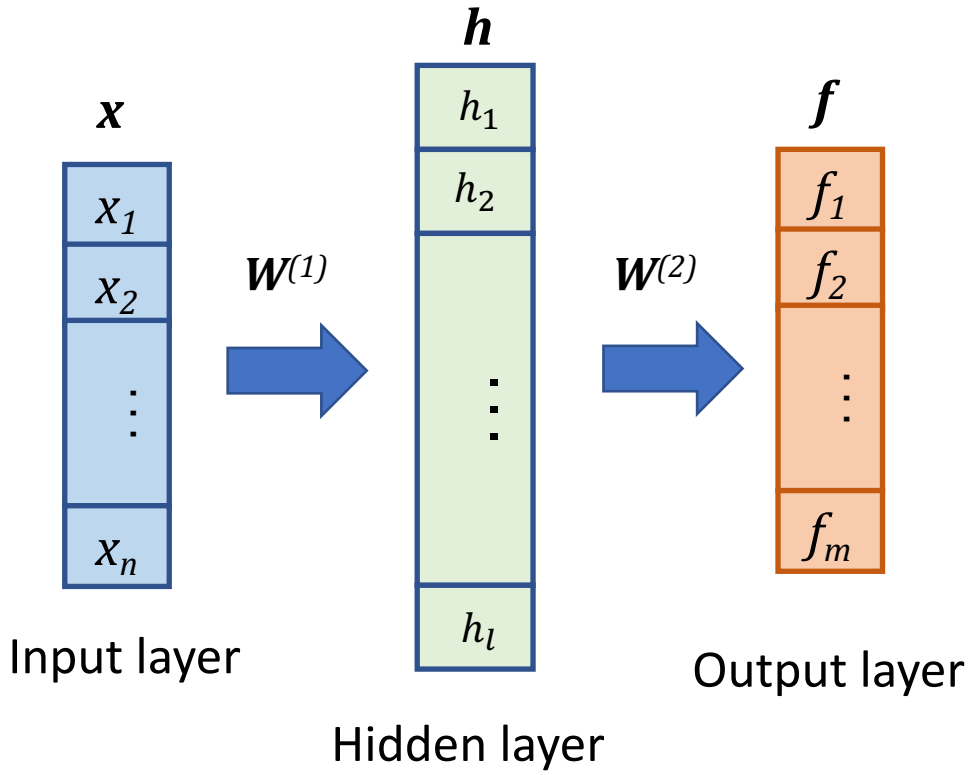
- We add a hidden layer between input and output layers



Artificial Neural Network

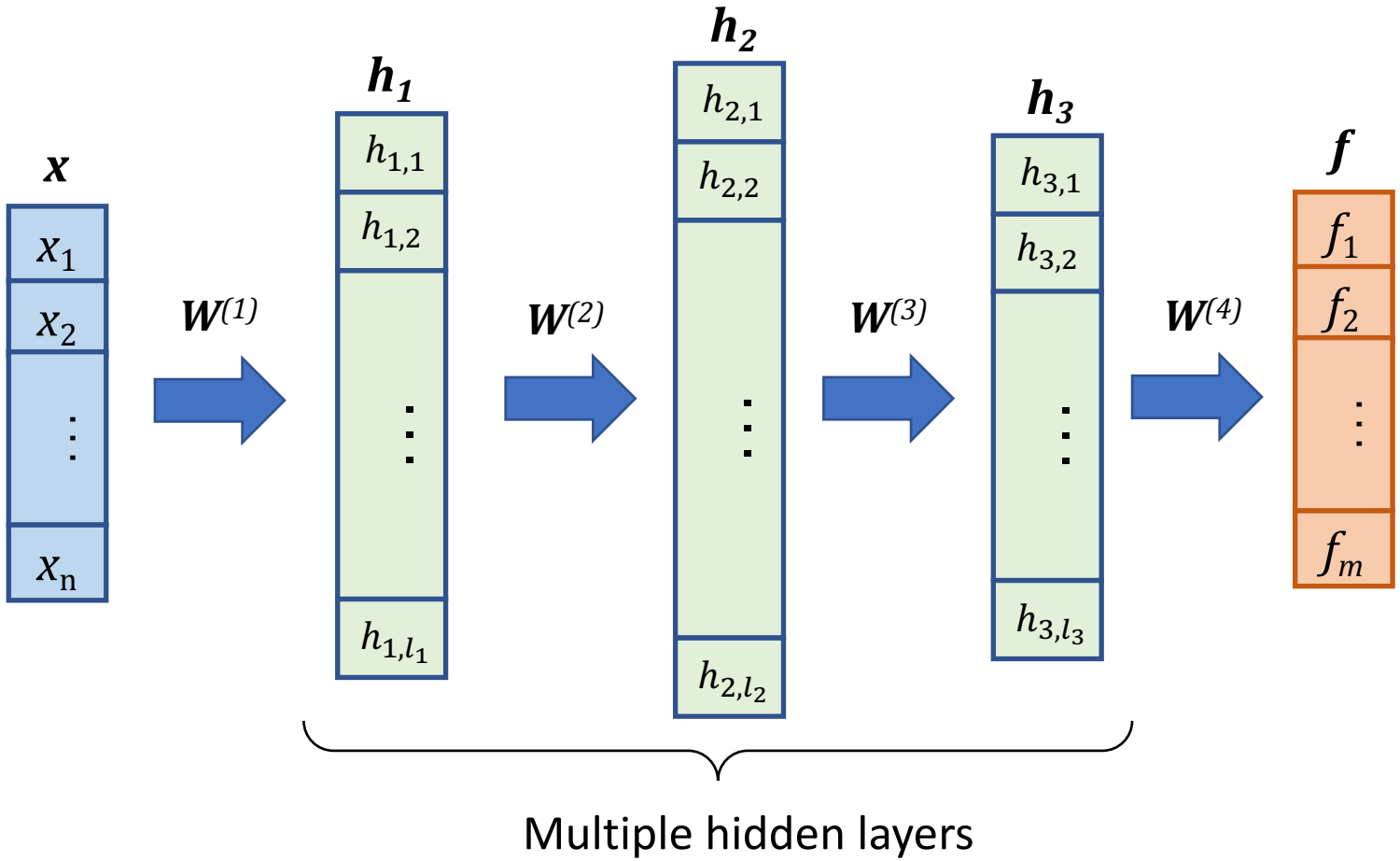
A single-layer perceptron

- We add a hidden layer between input and output layers



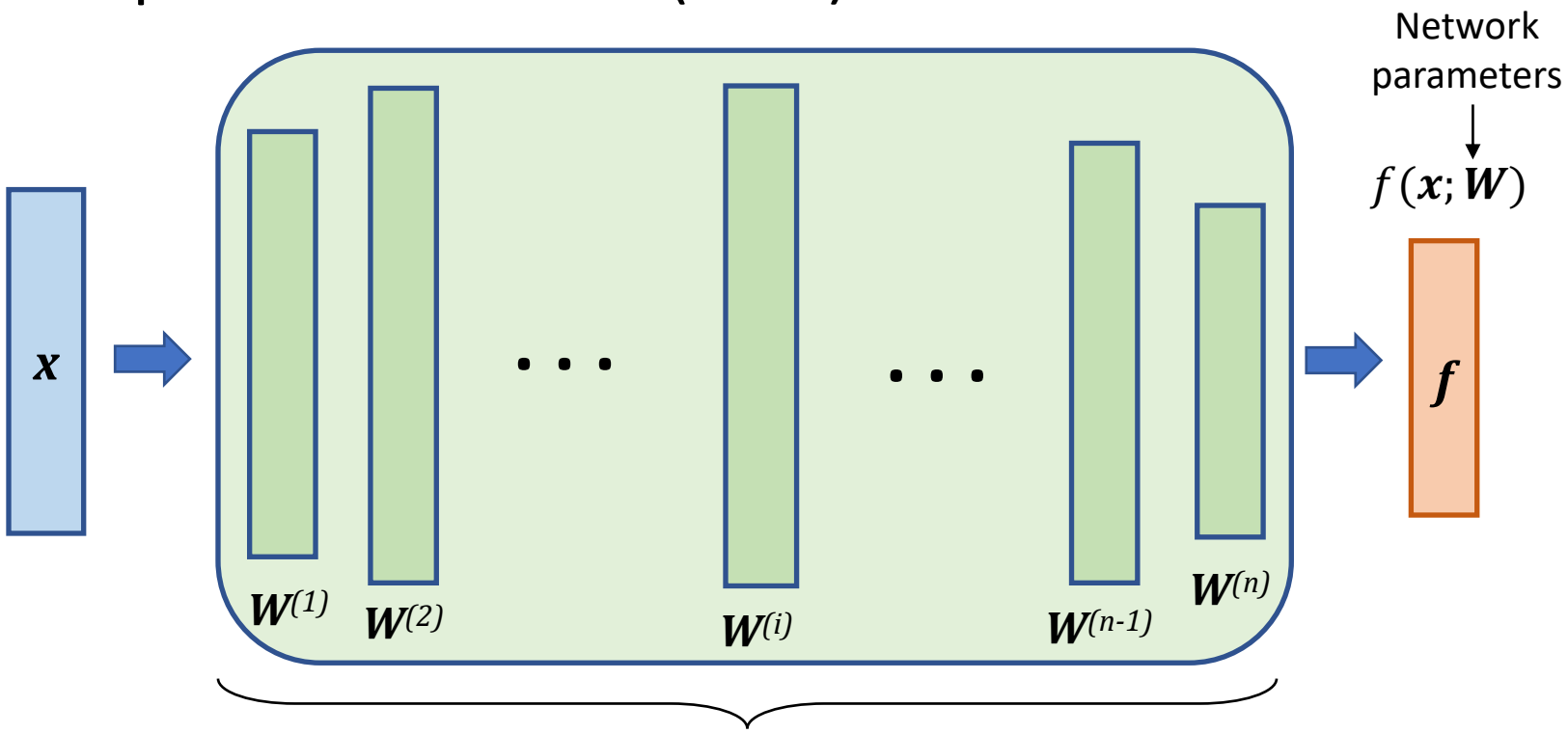
Artificial Neural Network

A Multi-Layer Perceptron (MLP)



Artificial Neural Network

A Deep Neural Network (DNN)

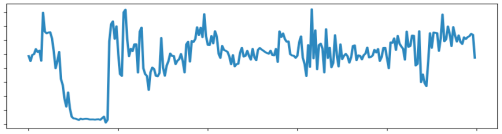


Not all of them necessarily dense layers – we will see other types of layers later

Artificial Neural Network

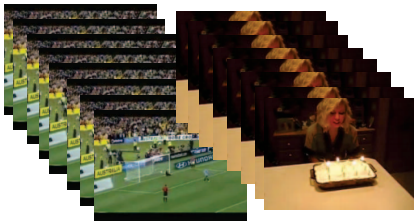
Different types of data used by a DNN

1D (text, audio, …)



2D (image, …)

3D (video, …)



Deep Neural Network

f

Artificial Neural Network

DNNs have been applied in many fields

- Natural Language Processing
- Computer Vision
- Health-care

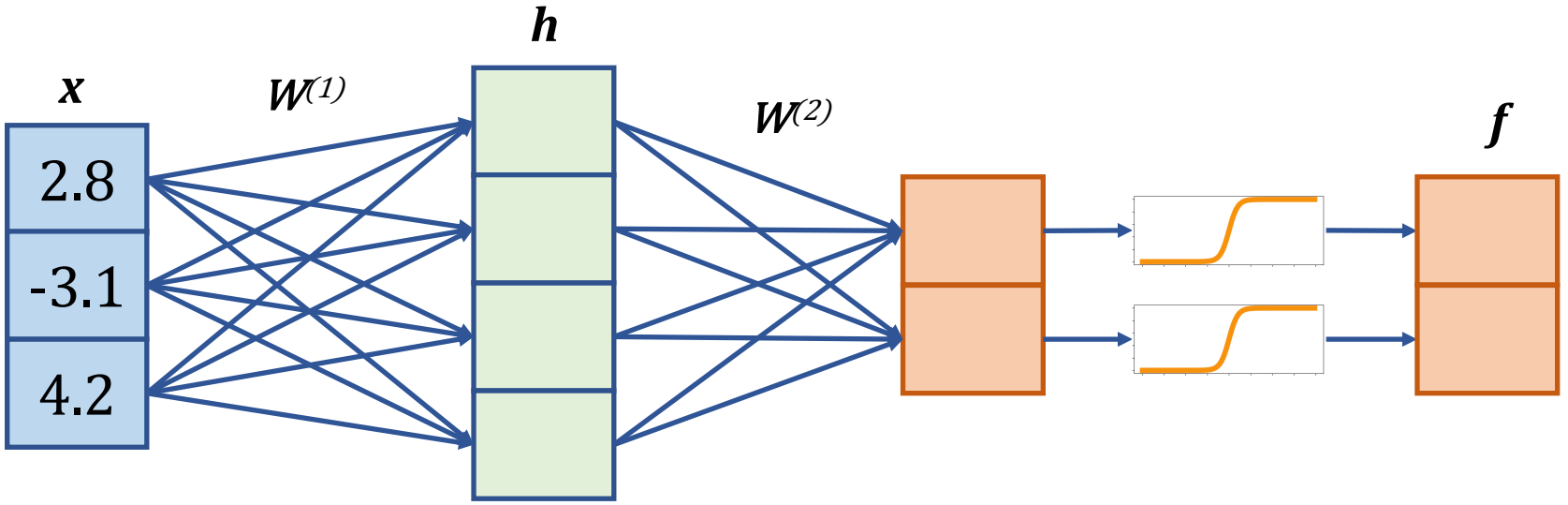
⋮

- Audio recognition
- Speech Recognition
- Machine Translation
- Question Answering
- Image Captioning
- Image Recognition
- Action Recognition
- Medical Image Analysis
- Personalised Medicine

⋮

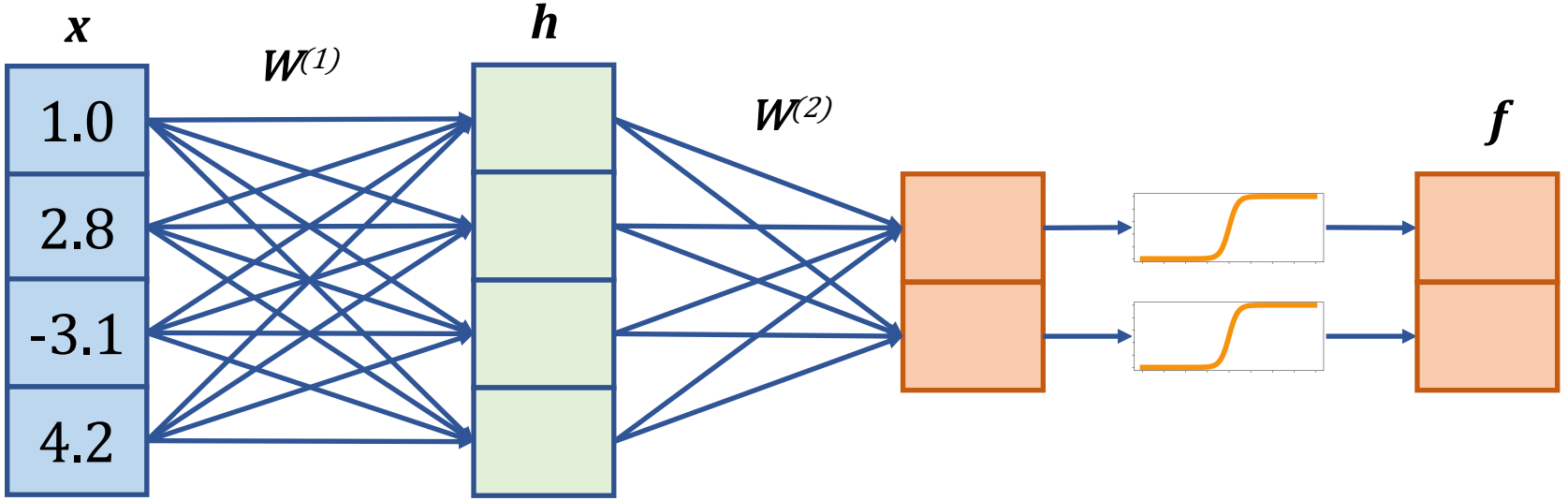
Artificial Neural Network

Example



Artificial Neural Network

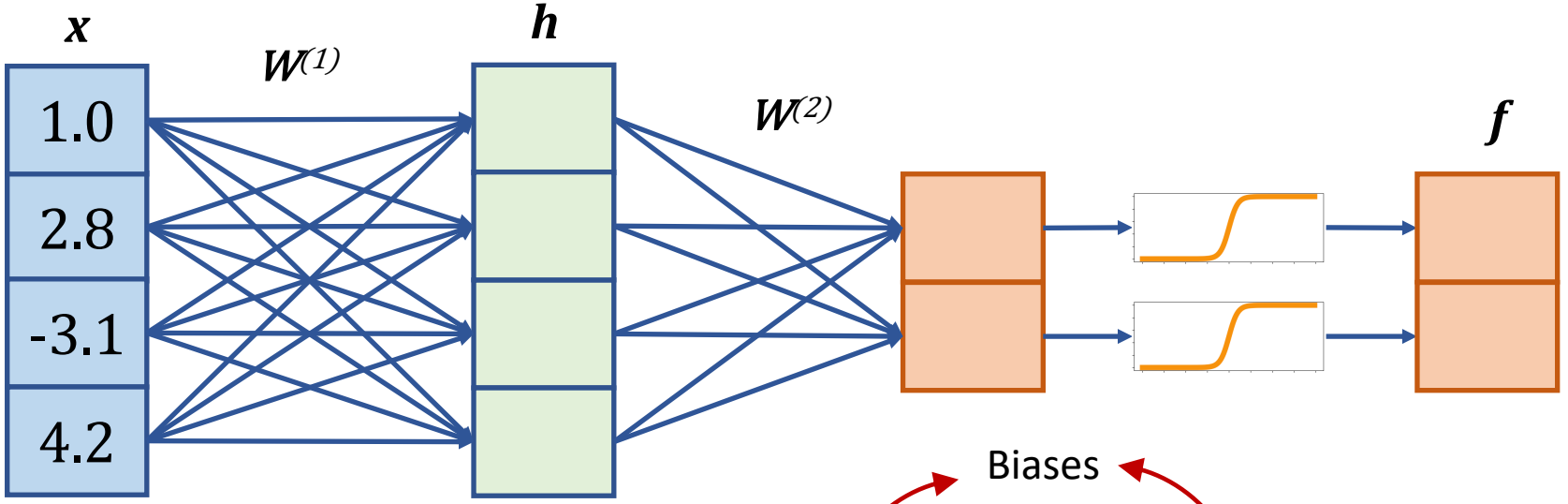
Example



$$\boldsymbol{x} = \begin{bmatrix} 1.0 \\ 2.8 \\ -3.1 \\ 4.2 \end{bmatrix}_{4\times1} \quad \boldsymbol{W^{(1)}} = \begin{bmatrix} 2.5 & -1.3 & -0.2 & 0.9 \\ 5.0 & 3.0 & -1.5 & 1.1 \\ 2.3 & -1.7 & 3.3 & -0.4 \\ -1.0 & 0.5 & 2.0 & -0.2 \end{bmatrix}_{4\times4} \quad \boldsymbol{W^{(2)}} = \begin{bmatrix} 0.6 & -1.7 \\ 1.9 & 3.0 \\ 0.6 & -1.2 \\ -1.3 & 2.6 \\ 2.4 & -0.8 \end{bmatrix}_{5\times2}$$

Artificial Neural Network

Example



$$\boldsymbol{x} = \begin{bmatrix} 1.0 \\ 2.8 \\ -3.1 \\ 4.2 \end{bmatrix}_{4\times1}$$

$$\boldsymbol{W^{(1)}} = \begin{bmatrix} 2.5 & -1.3 & -0.2 & 0.9 \\ 5.0 & 3.0 & -1.5 & 1.1 \\ 2.3 & -1.7 & 3.3 & -0.4 \\ -1.0 & 0.5 & 2.0 & -0.2 \end{bmatrix}_{4\times4}$$

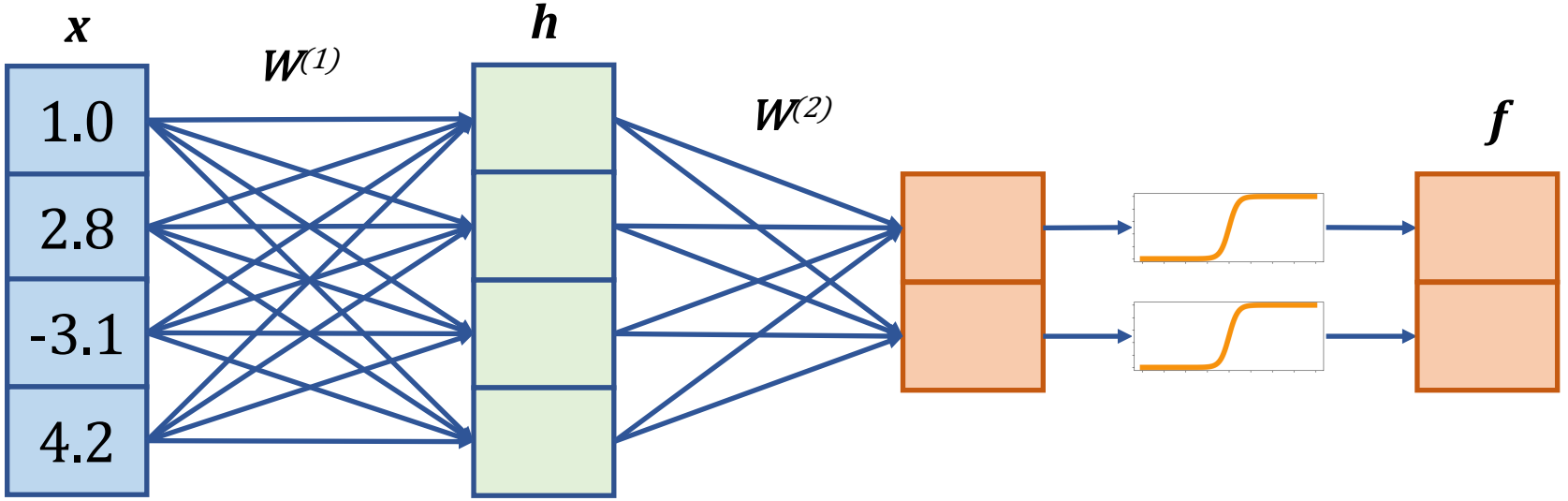
$$\boldsymbol{W^{(2)}} = \begin{bmatrix} 0.6 & -1.7 \\ 1.9 & 3.0 \\ 0.6 & -1.2 \\ -1.3 & 2.6 \\ 2.4 & -0.8 \end{bmatrix}_{5\times2}$$

Biases

Weights

Artificial Neural Network

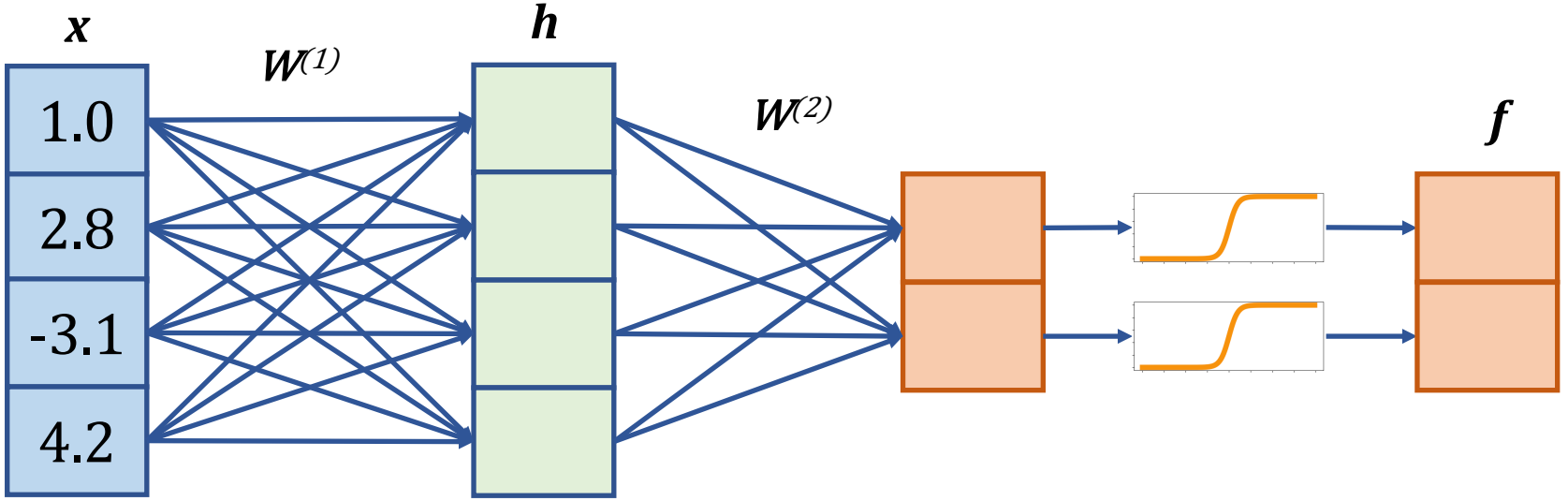
Example



$$\boldsymbol{x} = \begin{bmatrix} \boxed{1.0} \\ 2.8 \\ -3.1 \\ 4.2 \end{bmatrix}_{4\times1} \qquad \boldsymbol{W^{(1)}} = \begin{bmatrix} 2.5 & -1.3 & -0.2 & 0.9 \\ 5.0 & 3.0 & -1.5 & 1.1 \\ 2.3 & -1.7 & 3.3 & -0.4 \\ -1.0 & 0.5 & 2.0 & -0.2 \end{bmatrix}_{4\times4} \qquad \boldsymbol{W^{(2)}} = \begin{bmatrix} 0.6 & -1.7 \\ 1.9 & 3.0 \\ 0.6 & -1.2 \\ -1.3 & 2.6 \\ 2.4 & -0.8 \end{bmatrix}_{5\times2}$$

Artificial Neural Network

Example

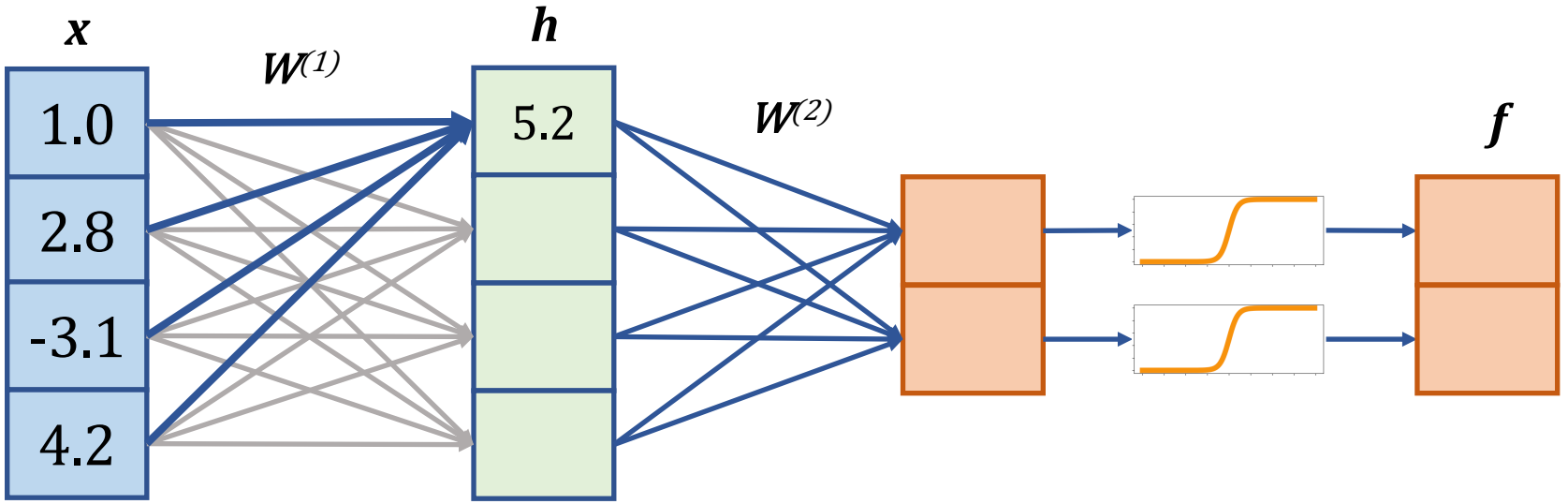


$$\boldsymbol{x}^{\boldsymbol{T}} \times \boldsymbol{W}^{(1)} = [1.0 \quad 2.8 \quad -3.1 \quad 4.2]_{1\times4} \times \begin{bmatrix} 2.5 & -1.3 & -0.2 & 0.9 \\ 5.0 & 3.0 & -1.5 & 1.1 \\ 2.3 & -1.7 & 3.3 & -0.4 \\ -1.0 & 0.5 & 2.0 & -0.2 \end{bmatrix}_{4\times4}$$

$$= [\qquad\qquad\qquad\qquad]_{1\times4}$$

Artificial Neural Network

Example

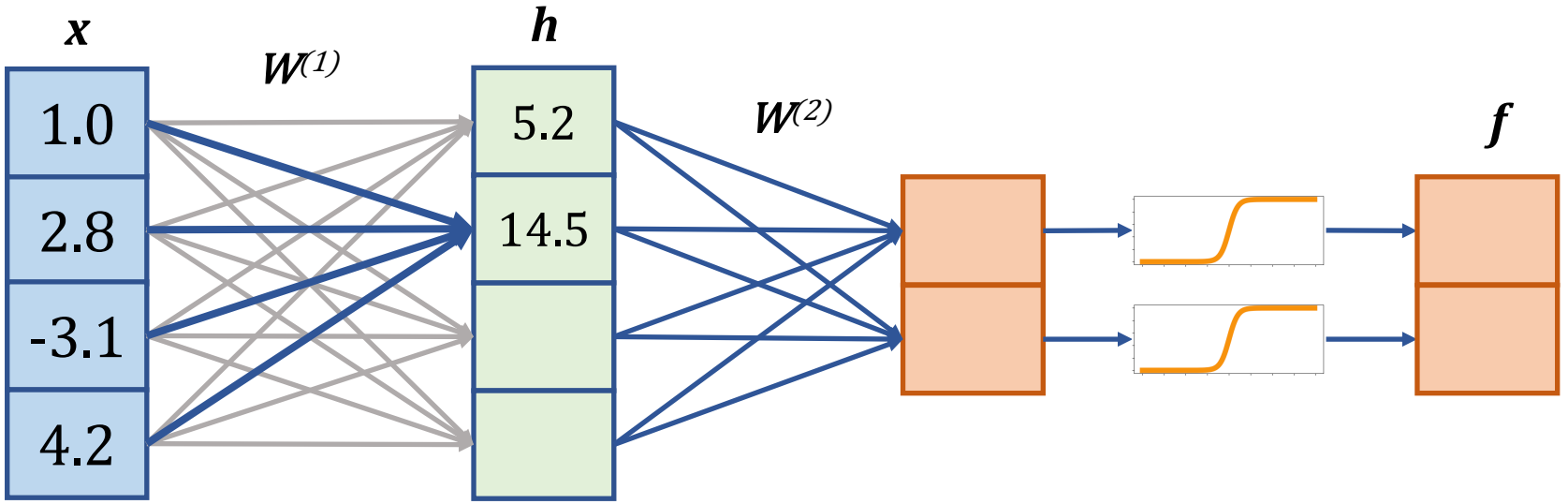


$$\boldsymbol{x}^{\boldsymbol{T}} \times \boldsymbol{W}^{(1)} = [1.0 \quad 2.8 \quad -3.1 \quad 4.2]_{1\times4} \times \begin{bmatrix} 2.5 & -1.3 & -0.2 & 0.9 \\ 5.0 & 3.0 & -1.5 & 1.1 \\ 2.3 & -1.7 & 3.3 & -0.4 \\ -1.0 & 0.5 & 2.0 & -0.2 \end{bmatrix}_{4\times4}$$

$$= [5.2 \qquad\qquad\qquad]_{1\times4}$$

Artificial Neural Network

Example

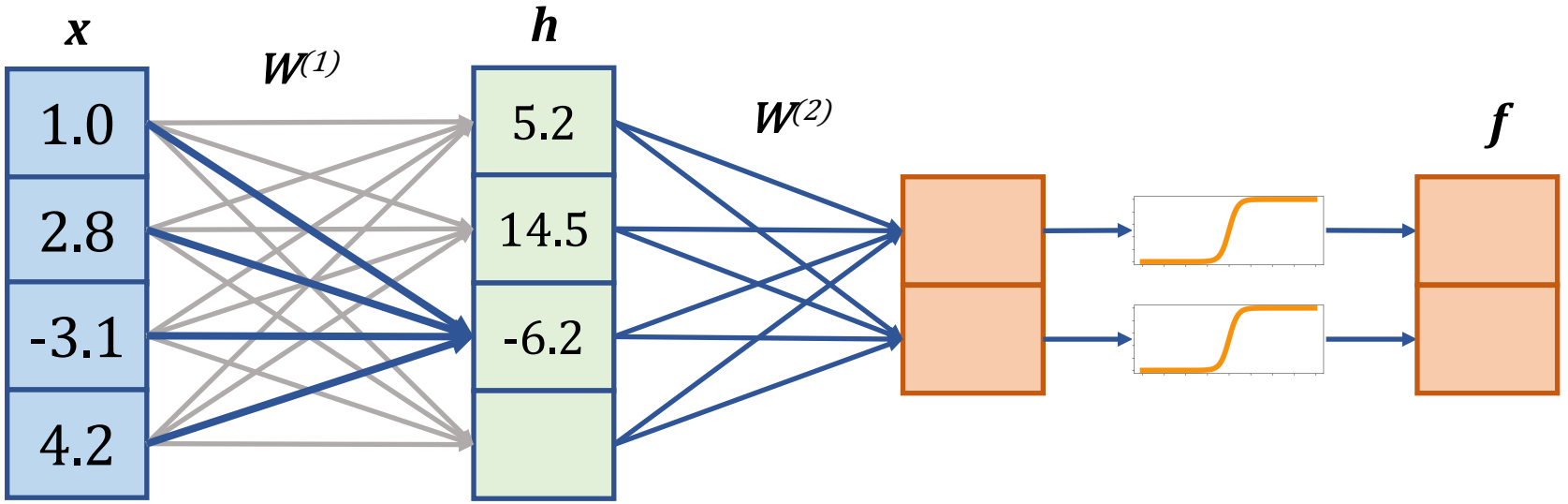


$$x^T \times W^{(1)} = [1.0 \quad 2.8 \quad -3.1 \quad 4.2]_{1\times4} \times \begin{bmatrix} 2.5 & -1.3 & -0.2 & 0.9 \\ 5.0 & 3.0 & -1.5 & 1.1 \\ 2.3 & -1.7 & 3.3 & -0.4 \\ -.01 & 0.5 & 2.0 & -0.2 \end{bmatrix}_{4\times4}$$

$$= [5.2 \quad 14.5 \quad \quad \quad]_{1\times4}$$

Artificial Neural Network

Example

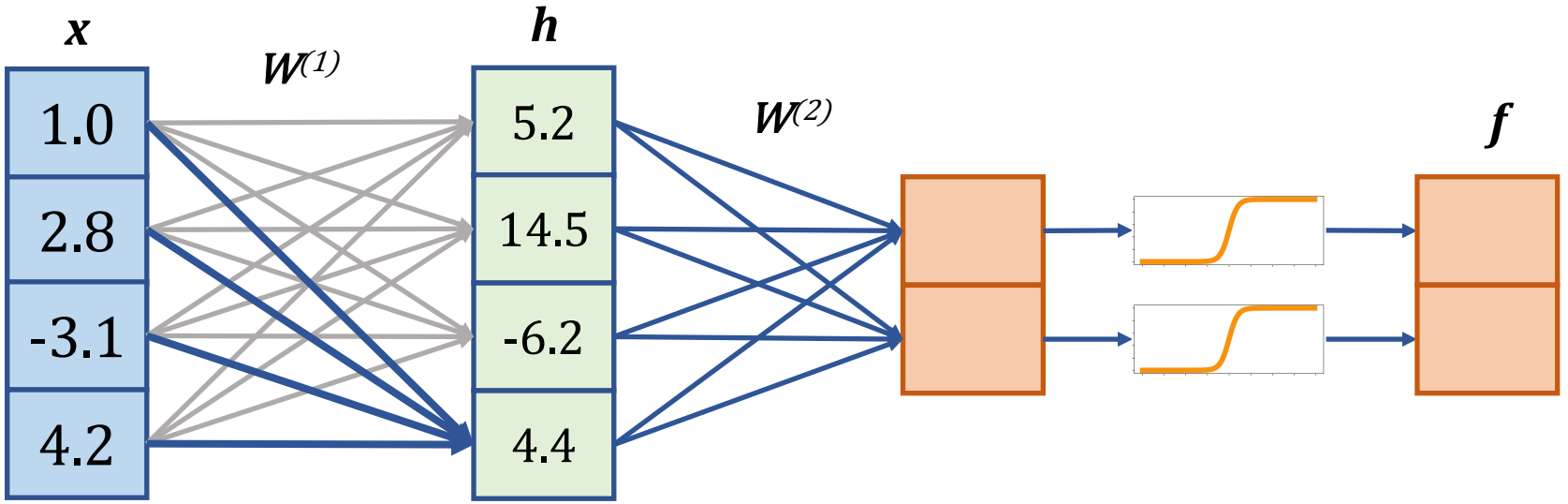


$$\boldsymbol{x}^{\boldsymbol{T}} \times \boldsymbol{W}^{(1)} = [1.0 \quad 2.8 \quad -3.1 \quad 4.2]_{1\times4} \times \begin{bmatrix} 2.5 & -1.3 & -0.2 & 0.9 \\ 5.0 & 3.0 & -1.5 & 1.1 \\ 2.3 & -1.7 & 3.3 & -0.4 \\ -.01 & 0.5 & 2.0 & -0.2 \end{bmatrix}_{4\times4}$$

$$= [5.2 \quad 14.5 \quad -6.2 \quad \quad]_{1\times4}$$

Artificial Neural Network

Example

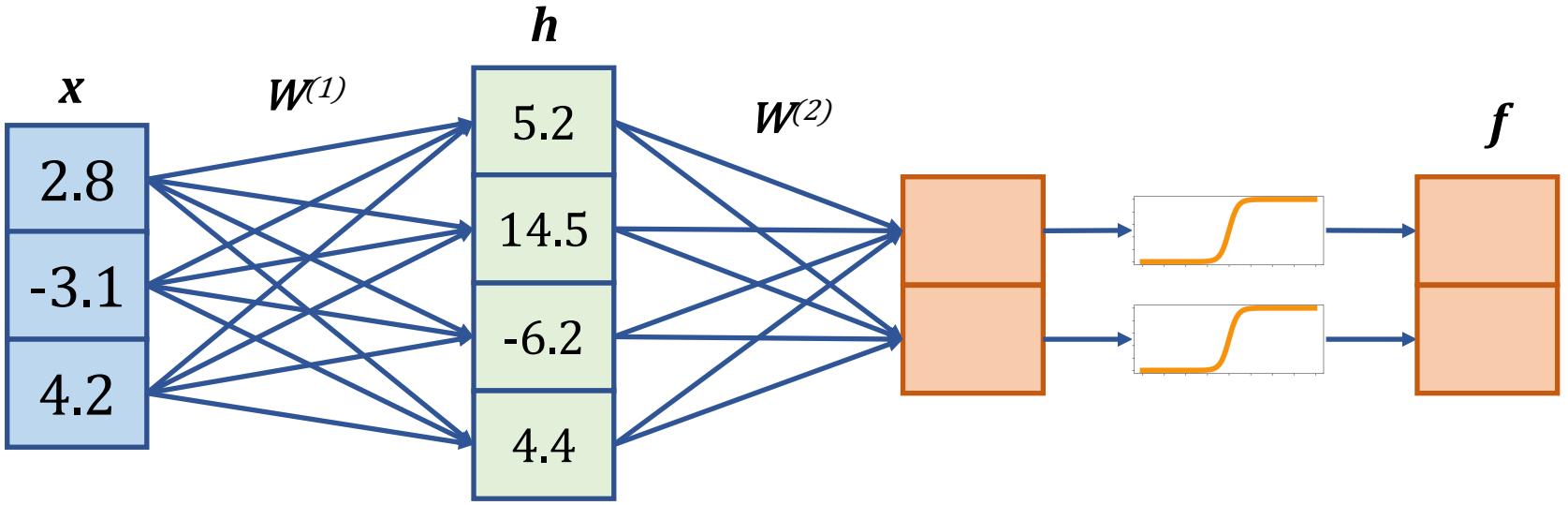


$$\boldsymbol{x}^{\boldsymbol{T}} \times \boldsymbol{W}^{(1)} = [1.0 \quad 2.8 \quad -3.1 \quad 4.2]_{1\times4} \times \begin{bmatrix} 2.5 & -1.3 & -0.2 & 0.9 \\ 5.0 & 3.0 & -1.5 & 1.1 \\ 2.3 & -1.7 & 3.3 & -0.4 \\ -.01 & 0.5 & 2.0 & -0.2 \end{bmatrix}_{4\times4}$$

$$= [5.2 \quad 14.5 \quad -6.2 \quad 4.4]_{1\times4}$$

Artificial Neural Network

Example

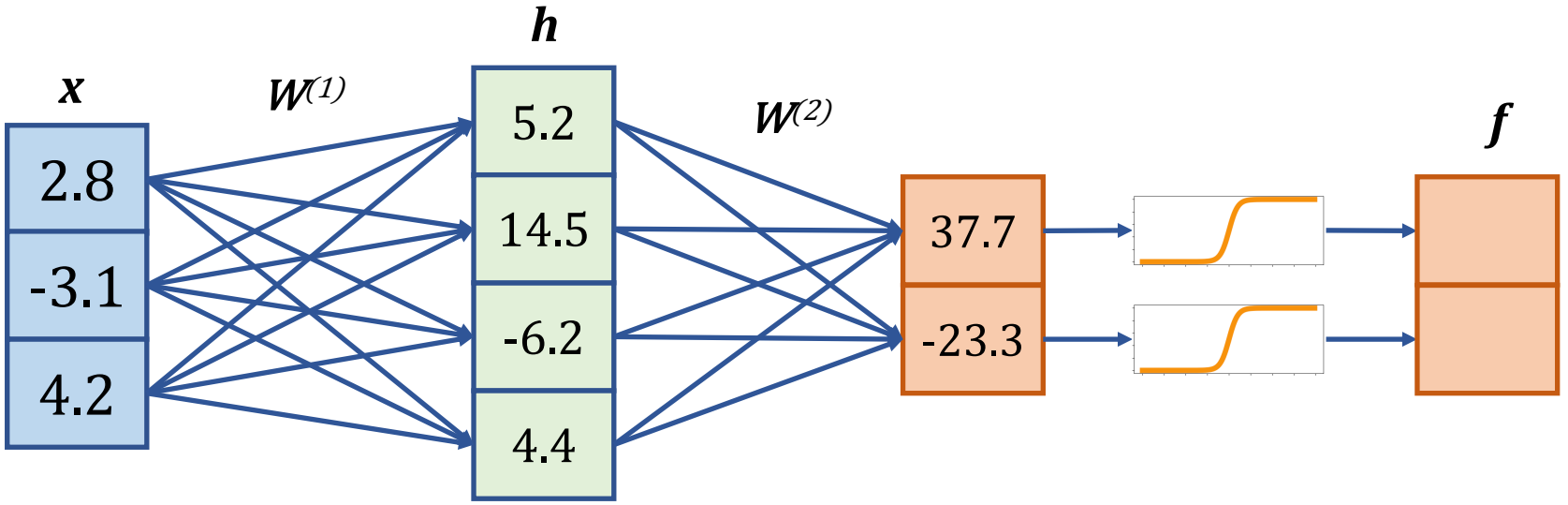


$$h = [1.0 \quad 5.2 \quad 14.5 \quad -6.2 \quad 4.4]_{1\times5}$$

$$\boldsymbol{W^{(2)}} = \begin{bmatrix} 0.6 & -1.7 \\ 1.9 & 3.0 \\ 0.6 & -1.2 \\ -1.3 & 2.6 \\ 2.4 & -0.8 \end{bmatrix}_{5\times2}$$

Artificial Neural Network

Example

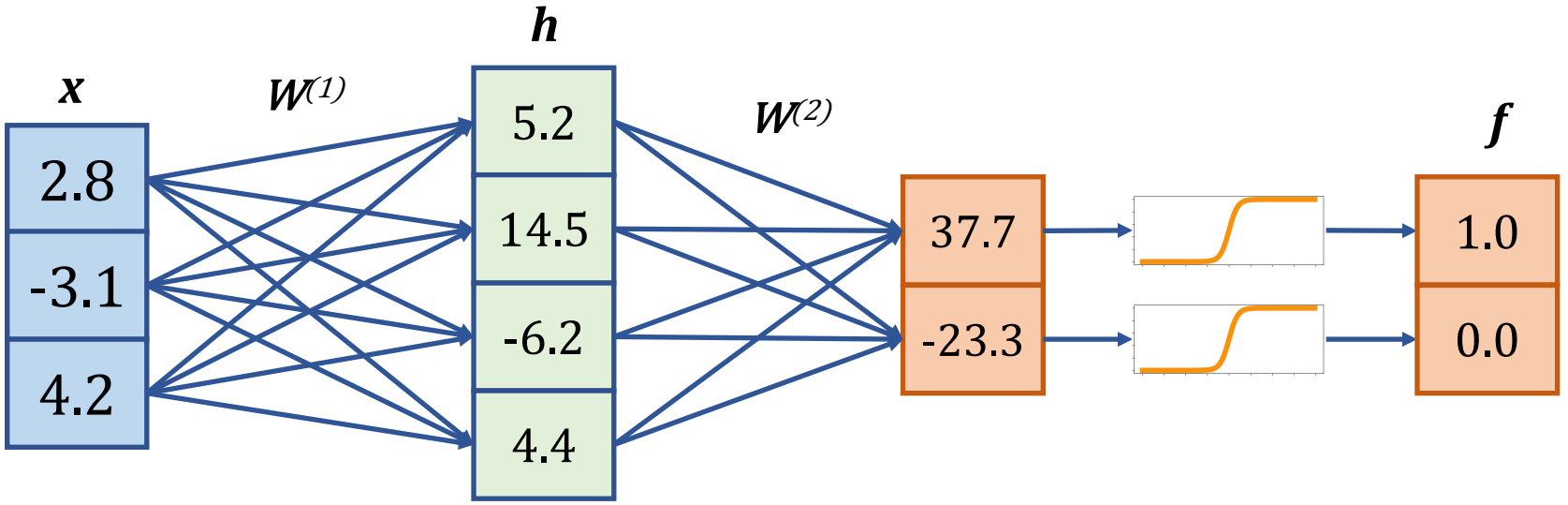


$$\boldsymbol{h} \times \boldsymbol{W^{(2)}} = [1.0 \quad 5.2 \quad 14.5 \quad -6.2 \quad 4.4]_{1\times5} \times \begin{bmatrix} 0.6 & -1.7 \\ 1.9 & 3.0 \\ 0.6 & -1.2 \\ -1.3 & 2.6 \\ 2.4 & -0.8 \end{bmatrix}_{5\times2}$$

$$= [37.7 \quad -23.3]_{1\times2}$$

Artificial Neural Network

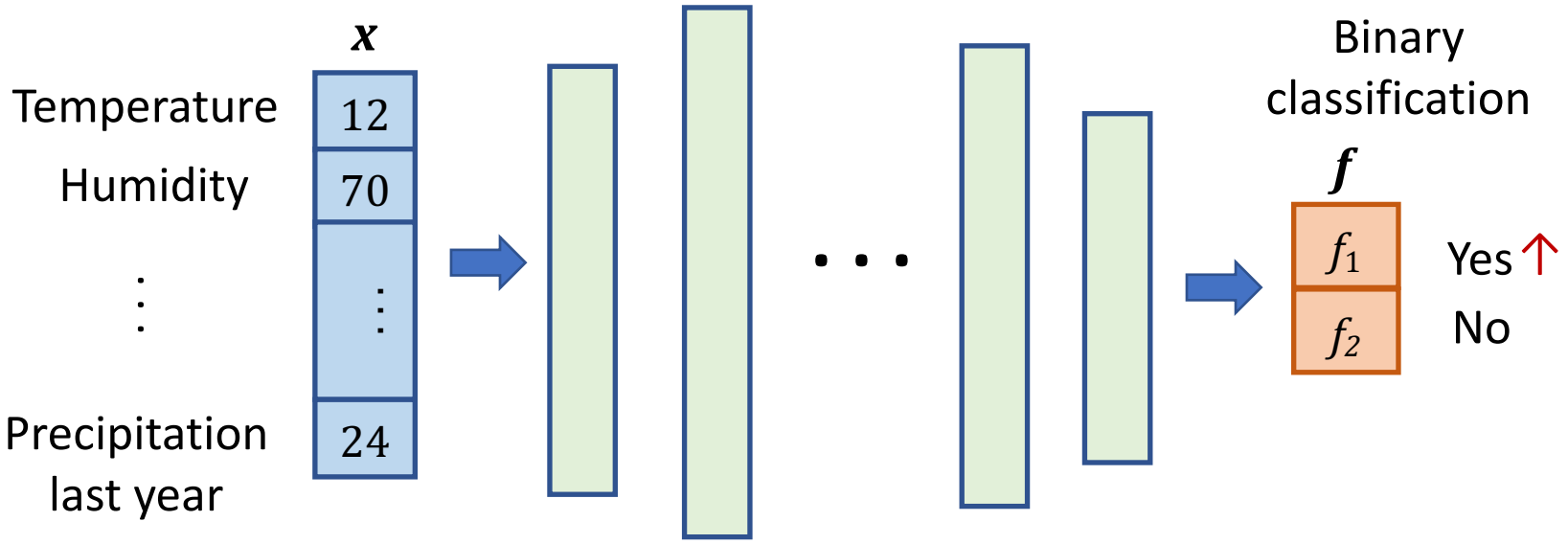
Example



Artificial Neural Network

Example

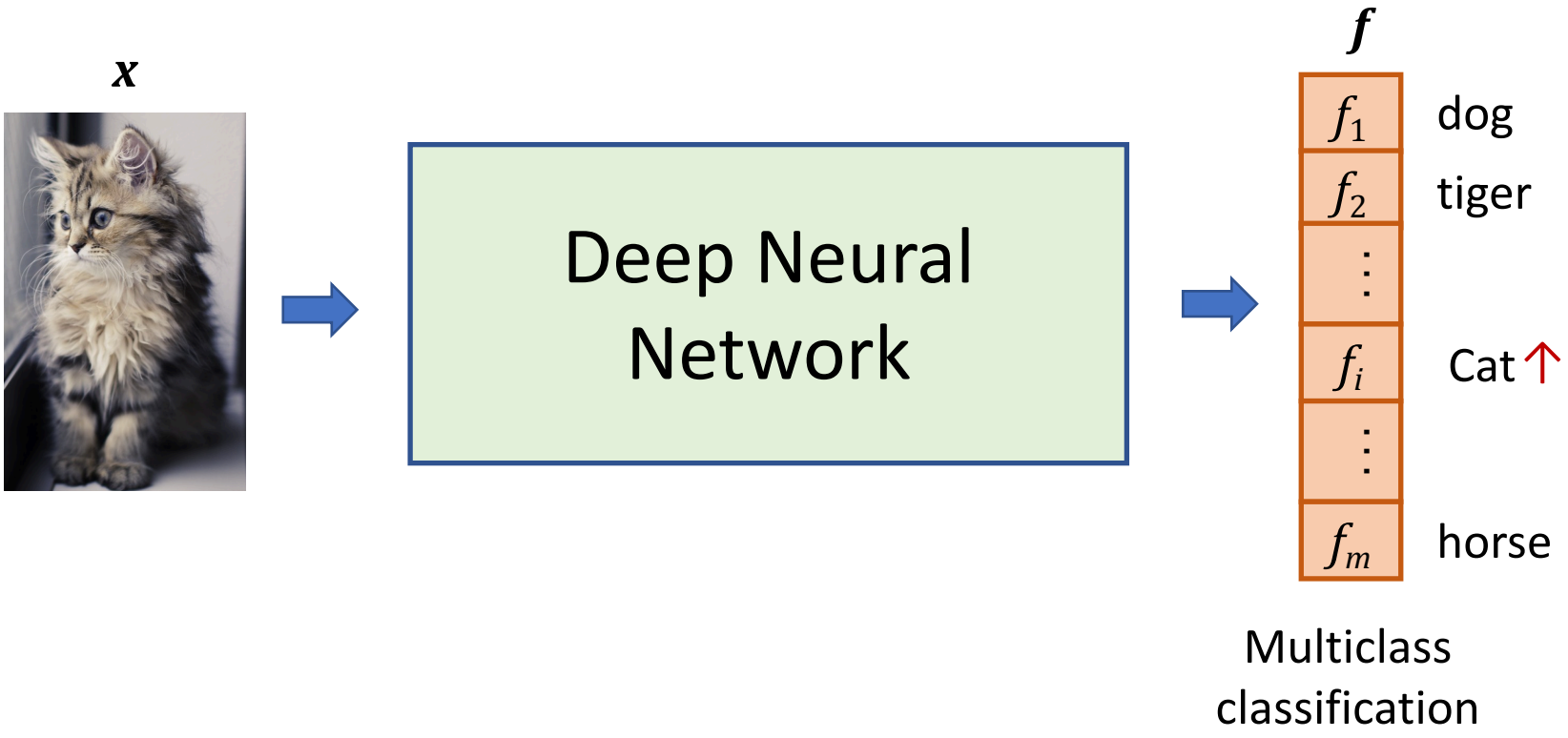
- A **classification** task – Is it going to rain today?



Artificial Neural Network

Example

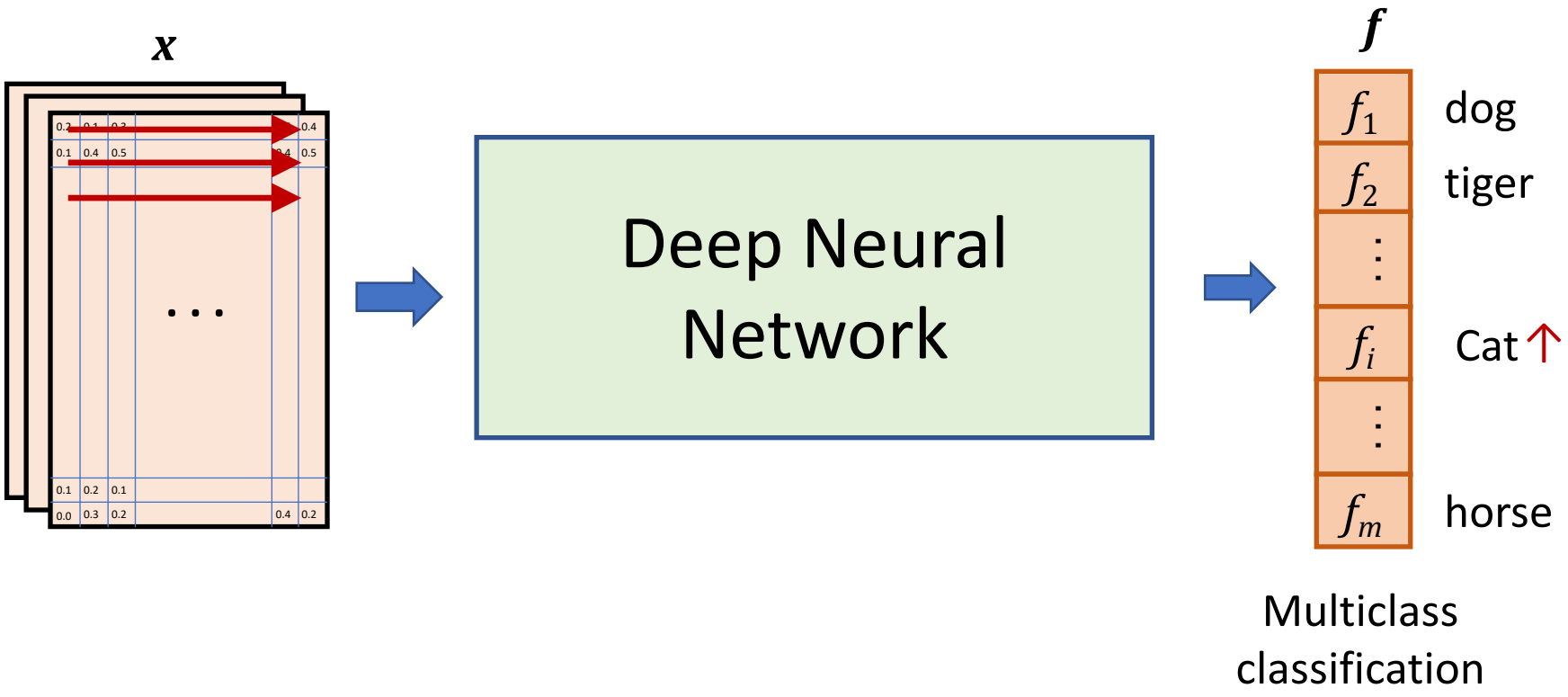
- A **classification** task – What animal is in this picture?



Artificial Neural Network

Example

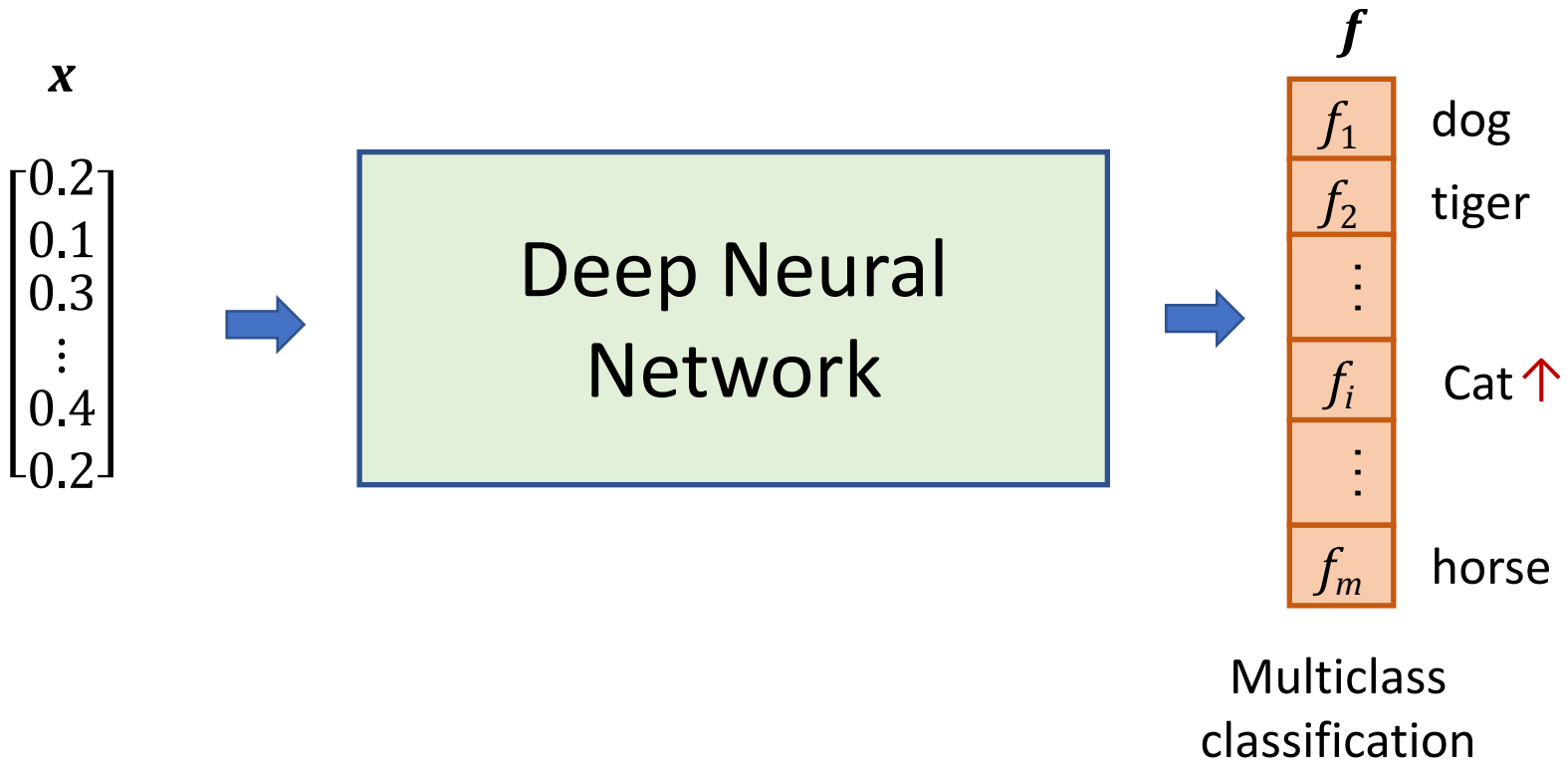
- A **classification** task – What animal is in this picture?



Artificial Neural Network

Example

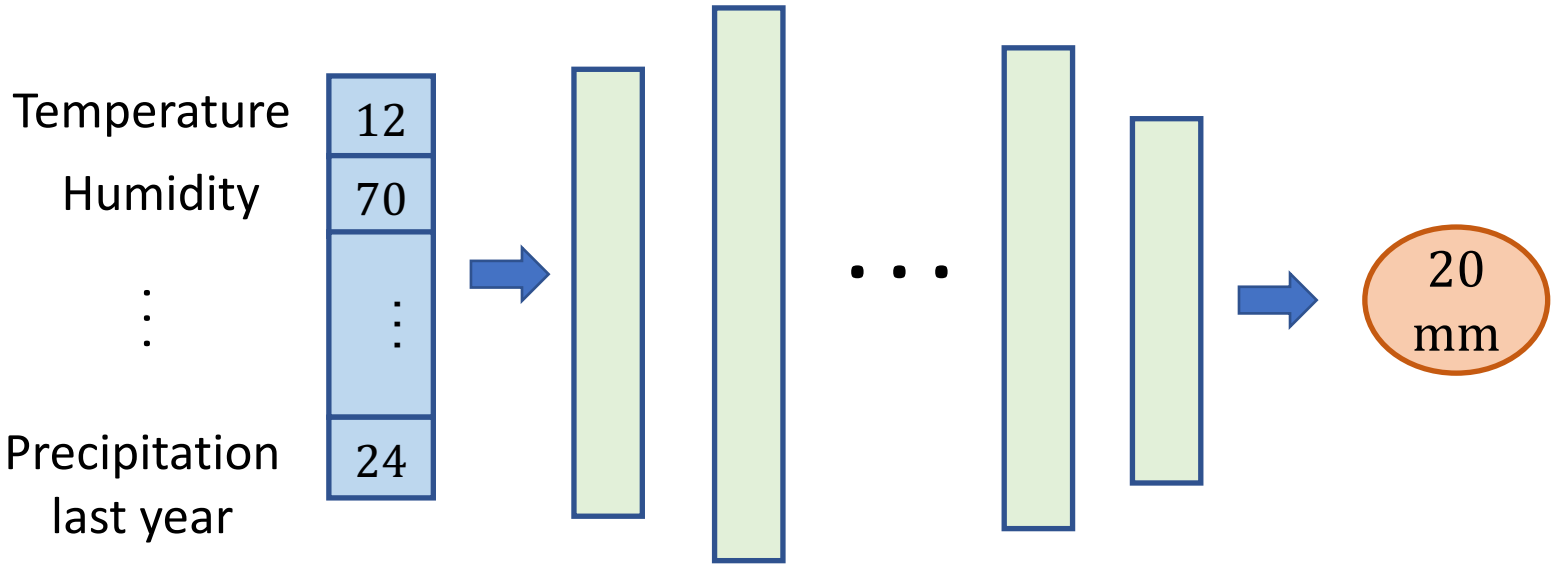
- A **classification** task – What animal is in this picture?



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?

Deep Neural Network

£ 800,000

Network Training

(Loss Function)

Network Training

How the network *learns* to predict the class labels

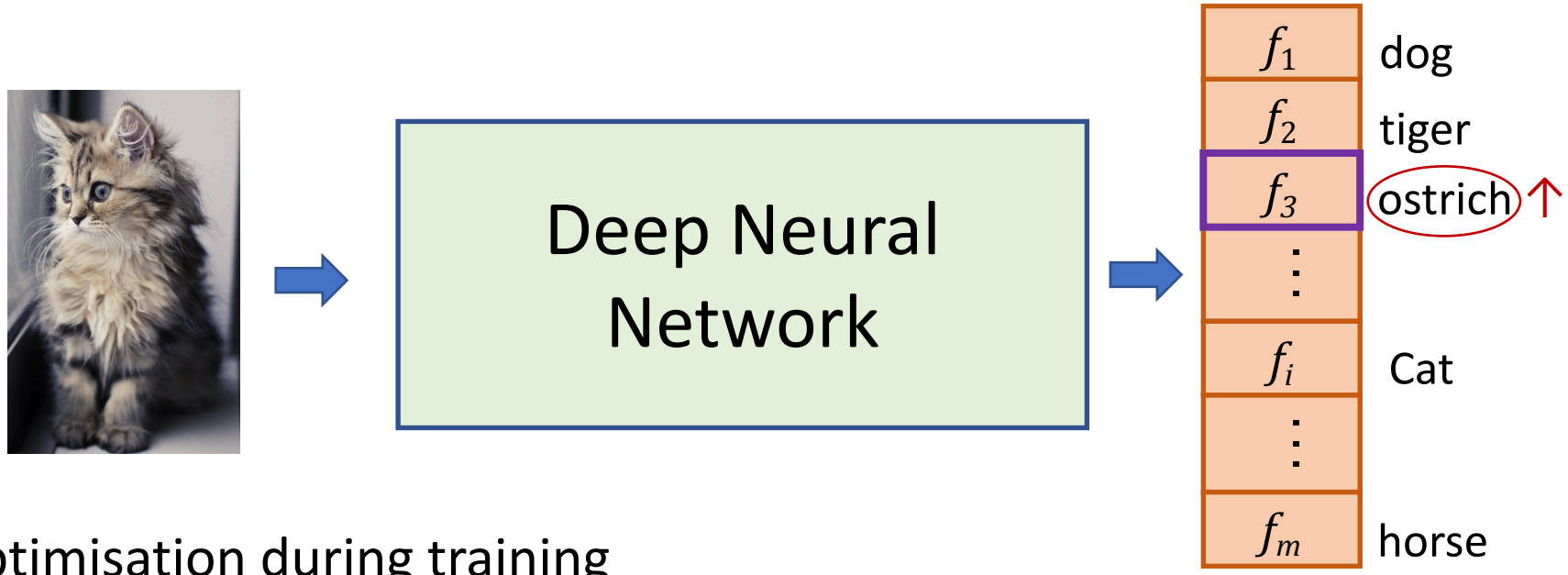
Deep Neural Network

f_1 dog
f_2 tiger
f_3 ostrich
⋮
f_i Cat
⋮
f_m horse

Classification example

Network Training

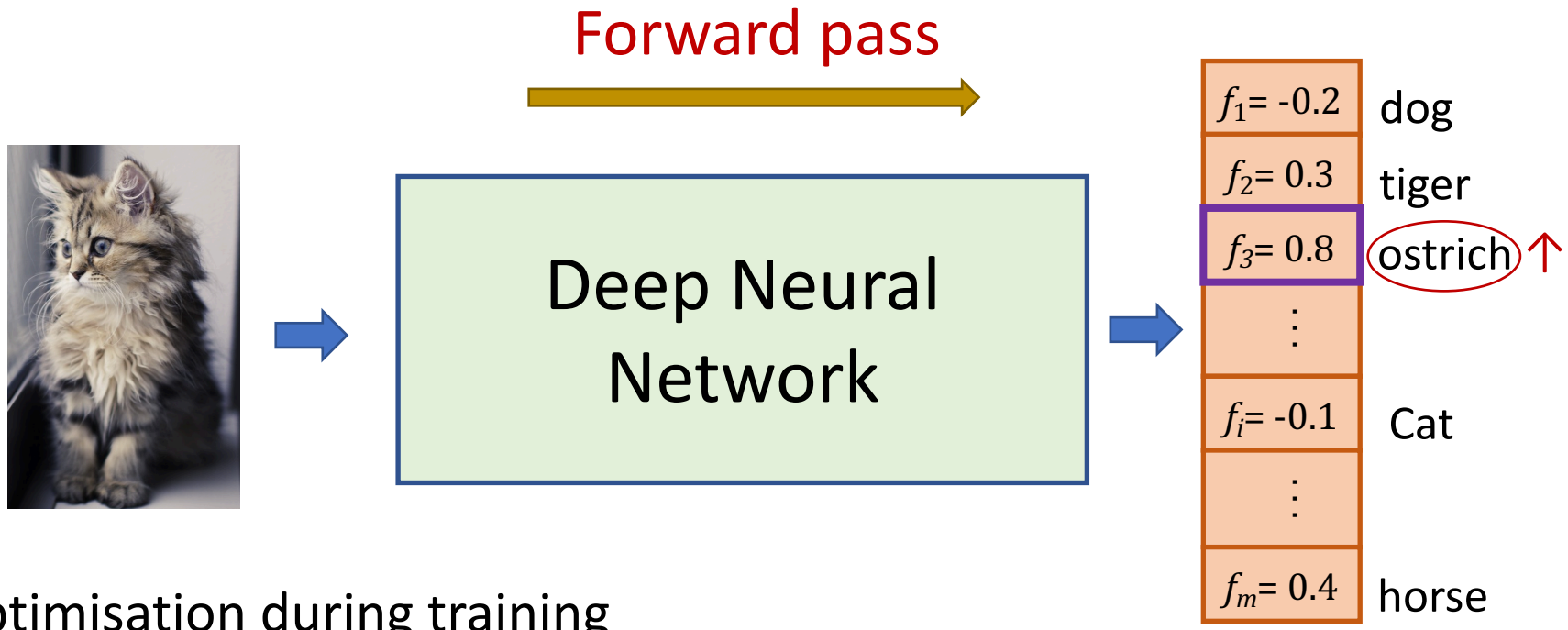
Before being trained, our network has no idea what this image is!



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



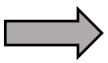
Optimisation during training

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

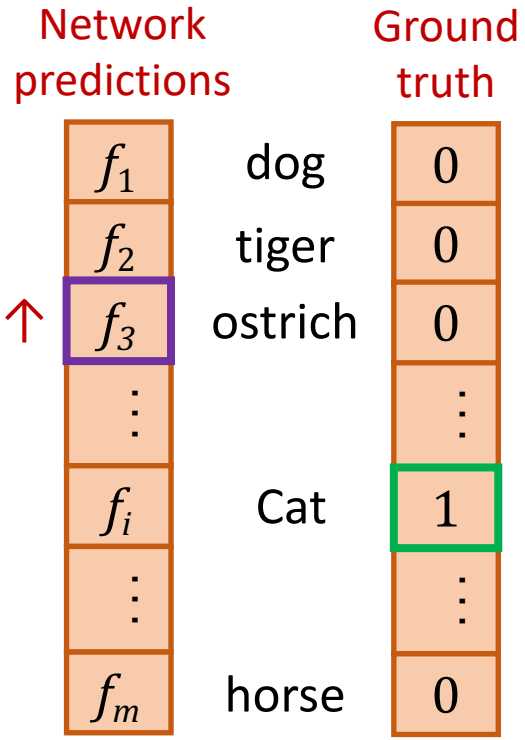
Network Training

Loss function

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



Our objective will be
to minimise the loss



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

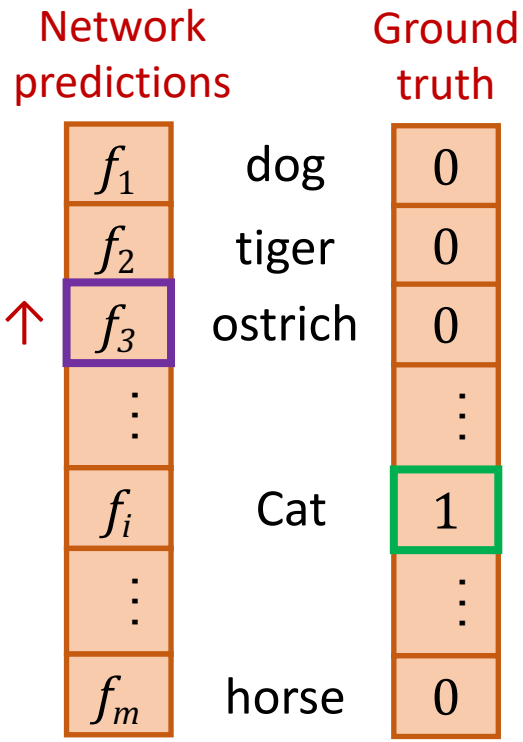
Network Training

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)

$$H(p,q) = -\sum_{x} p(x)\log(q(x))$$



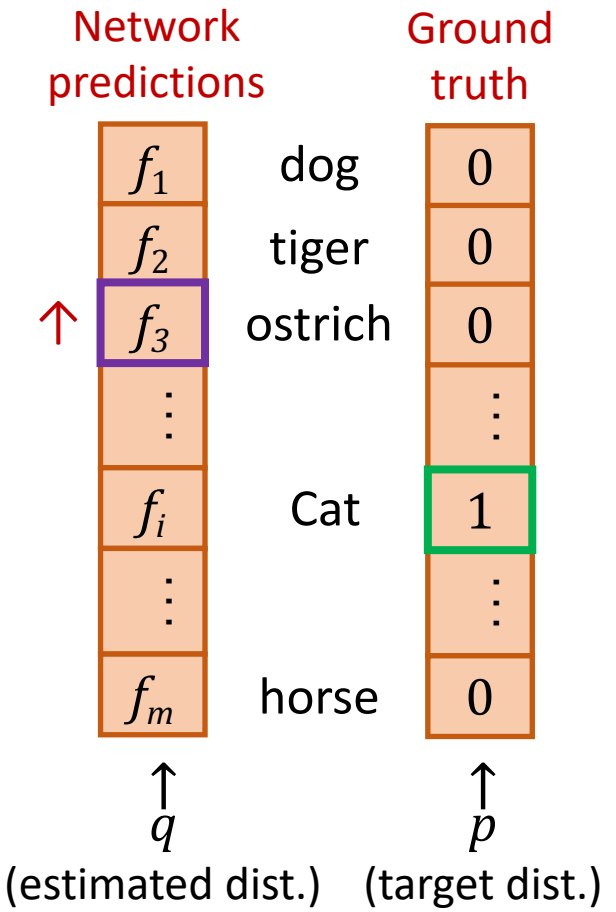
Network Training

Cross-entropy Loss function

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

$$H(p,q) = -\sum_{x} p(x)\log(q(x))$$

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



Network Training

Probabilistic interpretation for classification

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

Deep Neural Network

	0.05	dog	y_1
	0.1	tiger	y_2
	0.01	ostrich	y_3
	⋮		⋮
$\hat{y}$	0.7	Cat ↑	y_i
	⋮		⋮
	0.02	horse	y_m

and $y_i \in Y$

$0 \leq p(Y{=}y_i | \boldsymbol{x}) \leq 1$

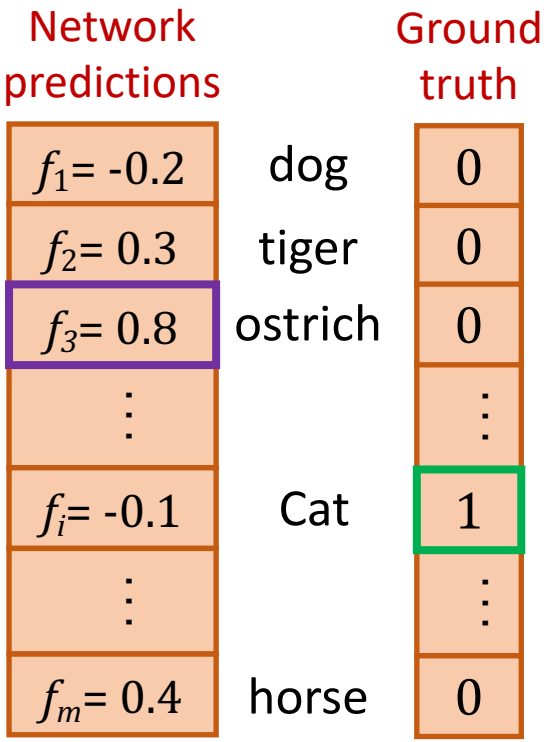
$\sum_i p(Y{=}y_i | \boldsymbol{x}){=}1$

Network Training

***Softmax* function**

Input: An arbitrary real-valued vector

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



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

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

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

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

Network predictions		Ground truth
f_1= -0.2	dog	0
f_2= 0.3	tiger	0
f_3= 0.8	ostrich	0
⋮		⋮
f_i= -0.1	Cat	1
⋮		⋮
f_m= 0.4	horse	0

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

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

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

Network predictions		Ground truth
f_1= -0.2	dog	0
f_2= 0.3	tiger	0
f_3= 0.8	ostrich	0
⋮		⋮
f_i= -0.1	Cat	1
⋮		⋮
f_m= 0.4	horse	0

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

Probability of the image to be a dog

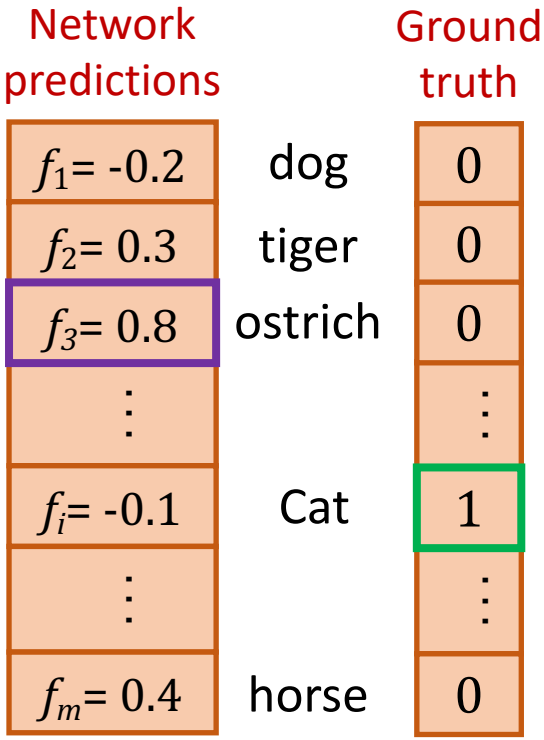
Network Training

***Softmax* function**

Input: An arbitrary real-valued vector

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

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



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

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

Probability of the image to be a dog

Network Training

Softmax function

Deep Neural Network

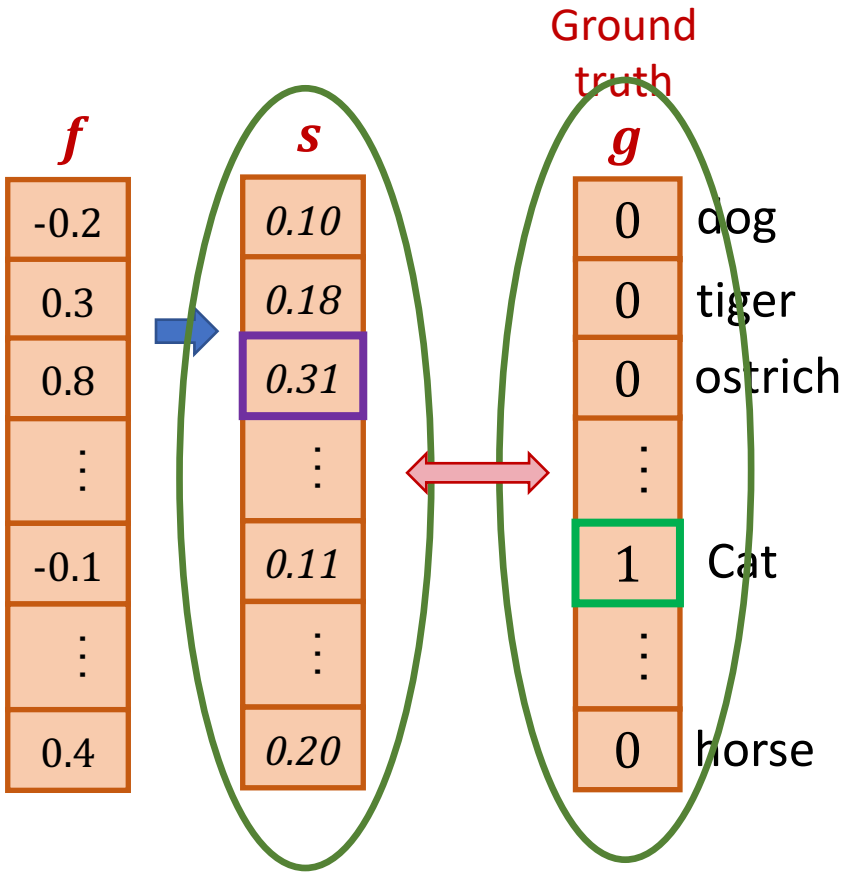
f	s	Ground truth g	
-0.2	*0.10*	0	dog
0.3	*0.18*	0	tiger
0.8	*0.31*	0	ostrich
⋮	⋮	⋮	
-0.1	*0.11*	1	Cat
⋮	⋮	⋮	
0.4	*0.20*	0	horse

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

Network Training

Cross-entropy Loss function

Deep Neural Network



Now our cross-entropy loss is

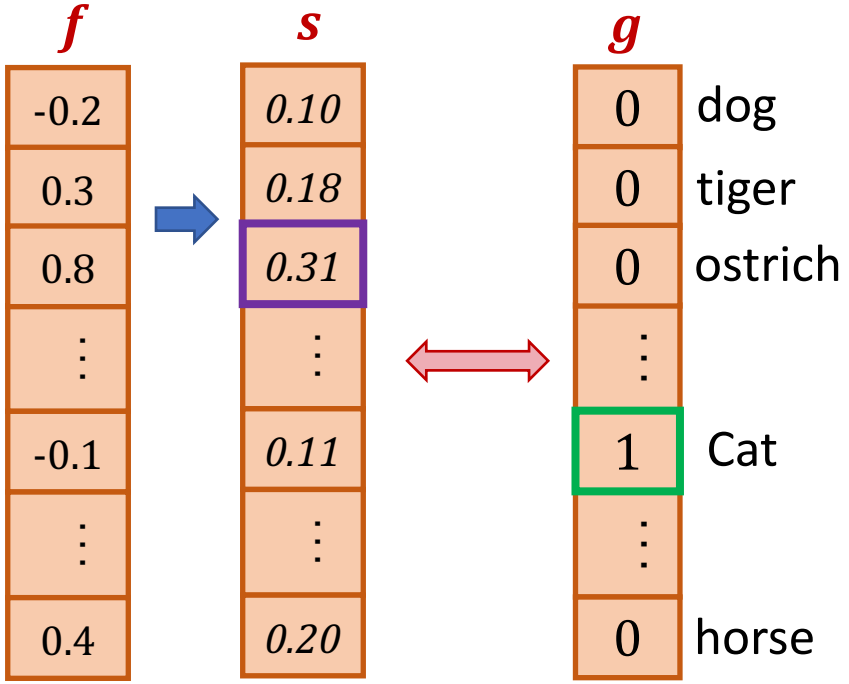
$$L = -\sum_i g_i \log(s_i)$$

$$L = -(0\times log0.10 + 0\times log0.18 + 0\times log0.31 + \cdots + 1\times log0.11 + \cdots + 0\times log0.20)$$

Network Training

Cross-entropy Loss function

Deep Neural Network



Now our cross-entropy loss is

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

$$L = -(0\times log0.10 + 0\times log0.18 + 0\times log0.31 + \cdots + 1\times log0.11 + \cdots + 0\times log0.20)$$

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