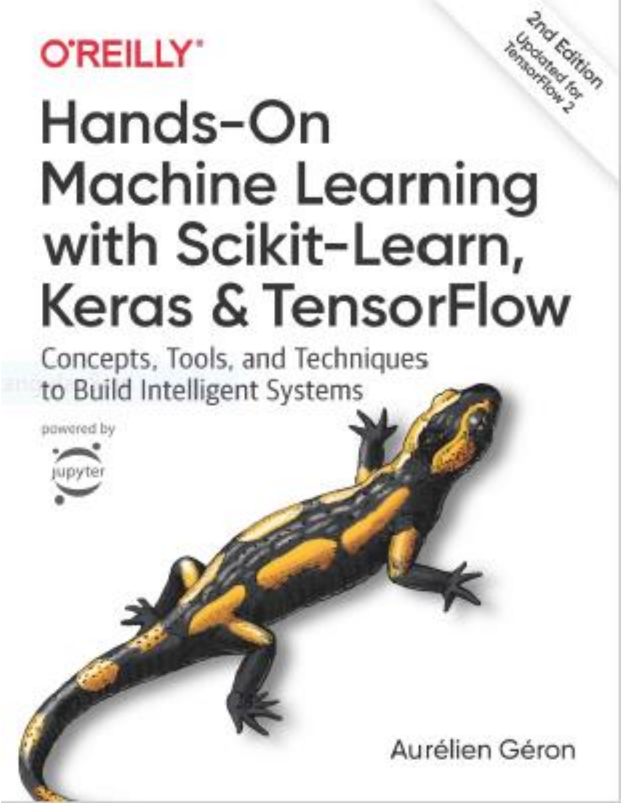
Neural Networks

Prof. Gheith Abandah

Reference



- Chapter 10: **Introduction to Artificial Neural Networks with Keras**

- Aurélien Géron, **Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow**, O'Reilly, 2nd Edition, 2019
  - Material: https://github.com/ageron/handson-ml2

Introduction

- YouTube Video: *But what *is* a Neural Network?* from 3Blue1Brown

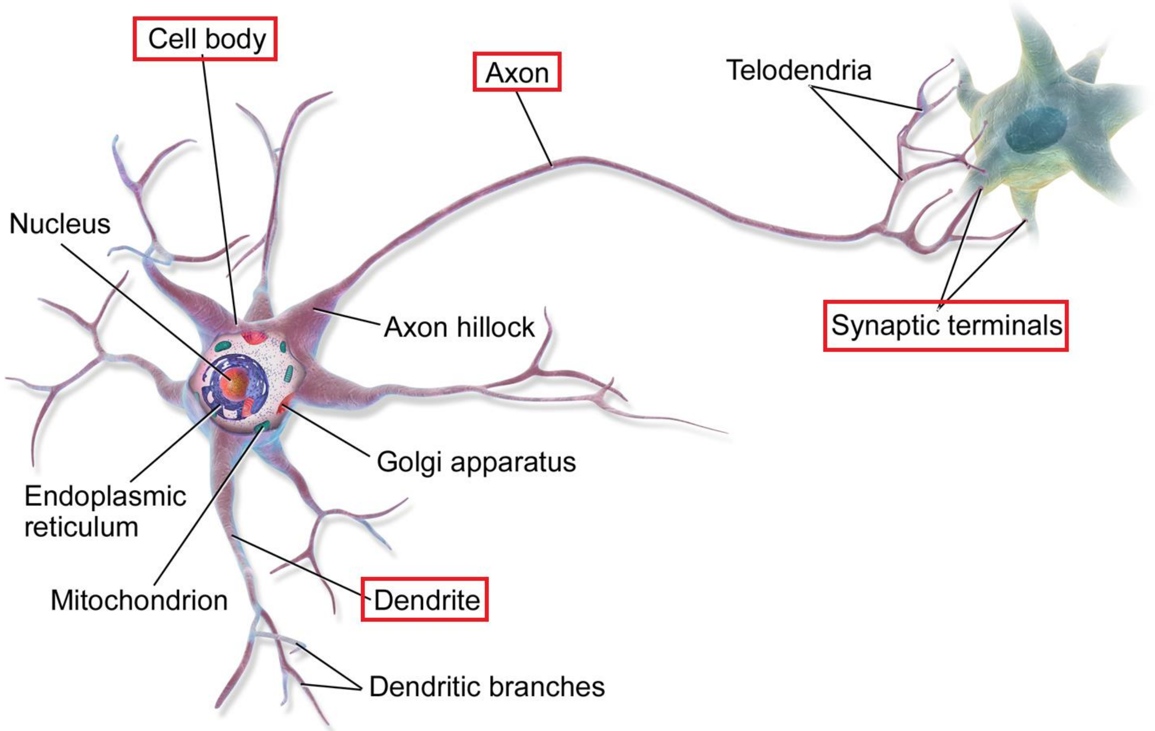
https://youtu.be/aircAruvnKk

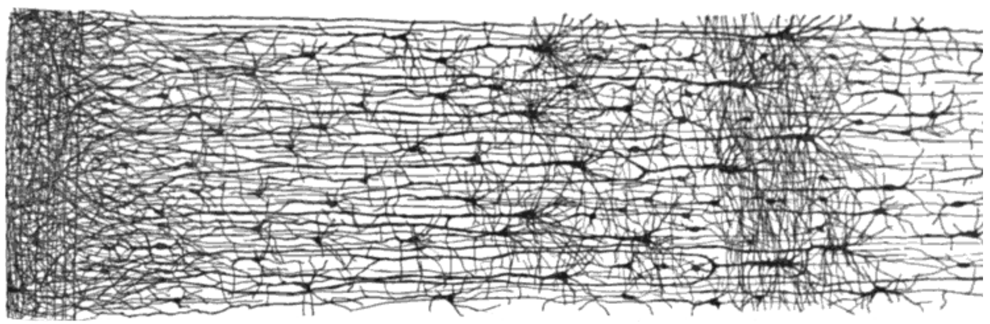
Outline

1. Introduction
2. The perceptron
3. Multi-layer perceptron (MLP)
4. Regression MLPs
5. Classification MLPs

1. Introduction

- **Artificial neural networks** (ANNs) are inspired by the brain's architecture.
- First suggested in 1943. Is now **flourishing** due to the availability of:
  - Data
  - Computing power
  - Better algorithms

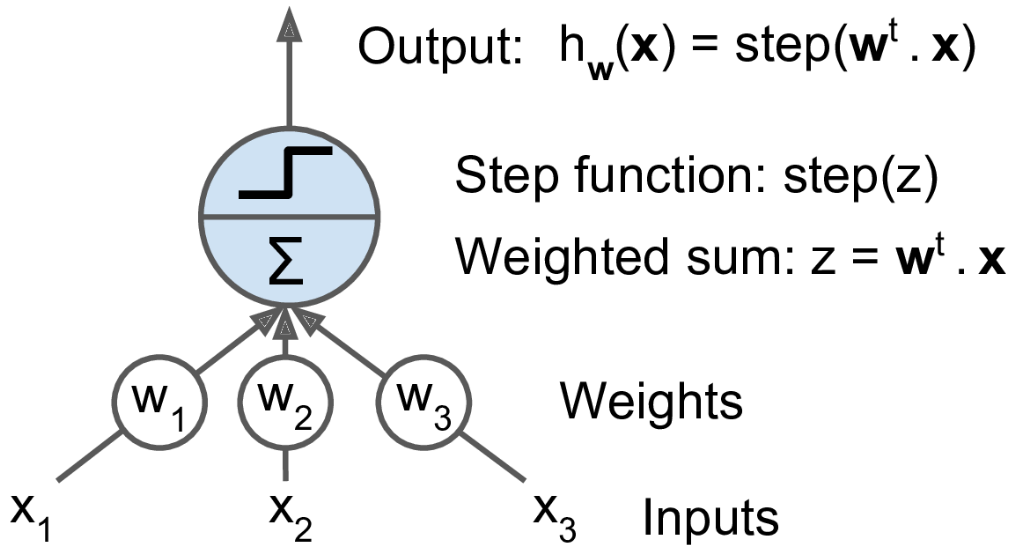




2. The Perceptron

- The **Perceptron** is a simple ANN, invented in 1957 and can perform linear binary classification or regression.

Linear threshold unit (LTU)



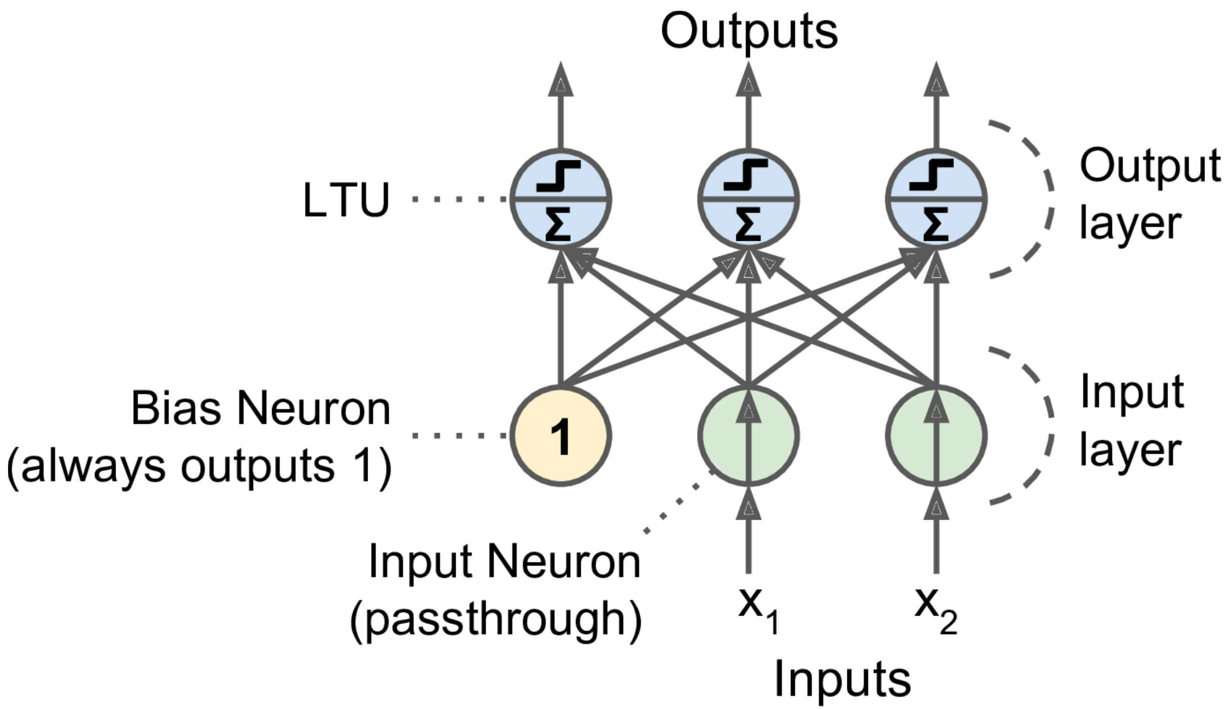
- Common **step function**:

$$\boxed{\text{heaviside}(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases} \qquad \text{sgn}(z) = \begin{cases} -1 & \text{if } z < 0 \\ 0 & \text{if } z = 0 \\ +1 & \text{if } z > 0 \end{cases}}$$

2. The Perceptron

- The Perceptron has an **input layer** with **bias** and **output layer**.
- With **multiple output nodes**, it can perform multiclass classification.
- Hebbian learning "**Cells that fire together, wire together**."

$$\boxed{w_{i,j}^{(\text{next step})} = w_{i,j} + \eta\left(y_j - \hat{y}_j\right)x_i}$$



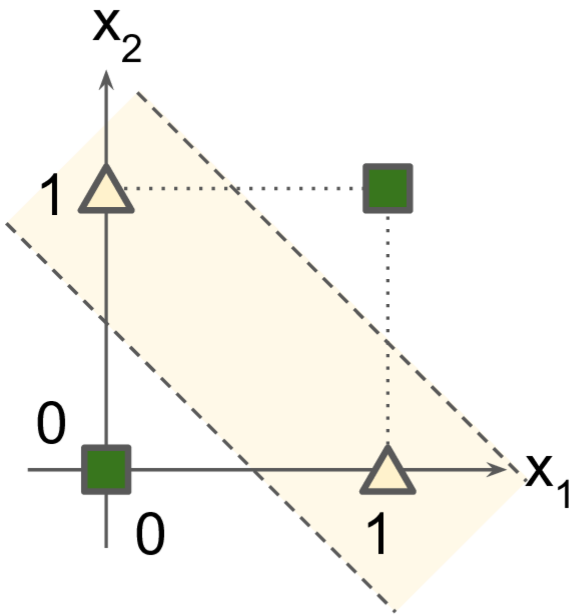
2. The Perceptron

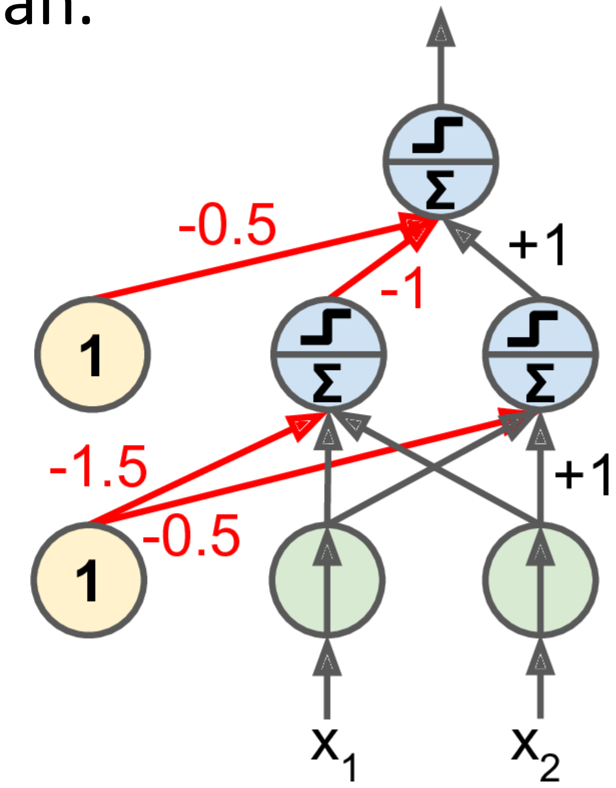
- Scikit-Learn provides a **Perceptron** class.

```python
import numpy as np
from sklearn.datasets import load_iris
from sklearn.linear_model import Perceptron

iris = load_iris()
X = iris.data[:, (2, 3)]  # petal length, petal width
y = (iris.target == 0).astype(np.int)  # Iris Setosa?
per_clf = Perceptron(random_state=42)
per_clf.fit(X, y)

y_pred = per_clf.predict([[2, 0.5]])
```

2. The Perceptron

- The perceptron **cannot solve non-linear problems** such as the XOR problem.
- The **Multi-Layer Perceptron** (MLP) can.



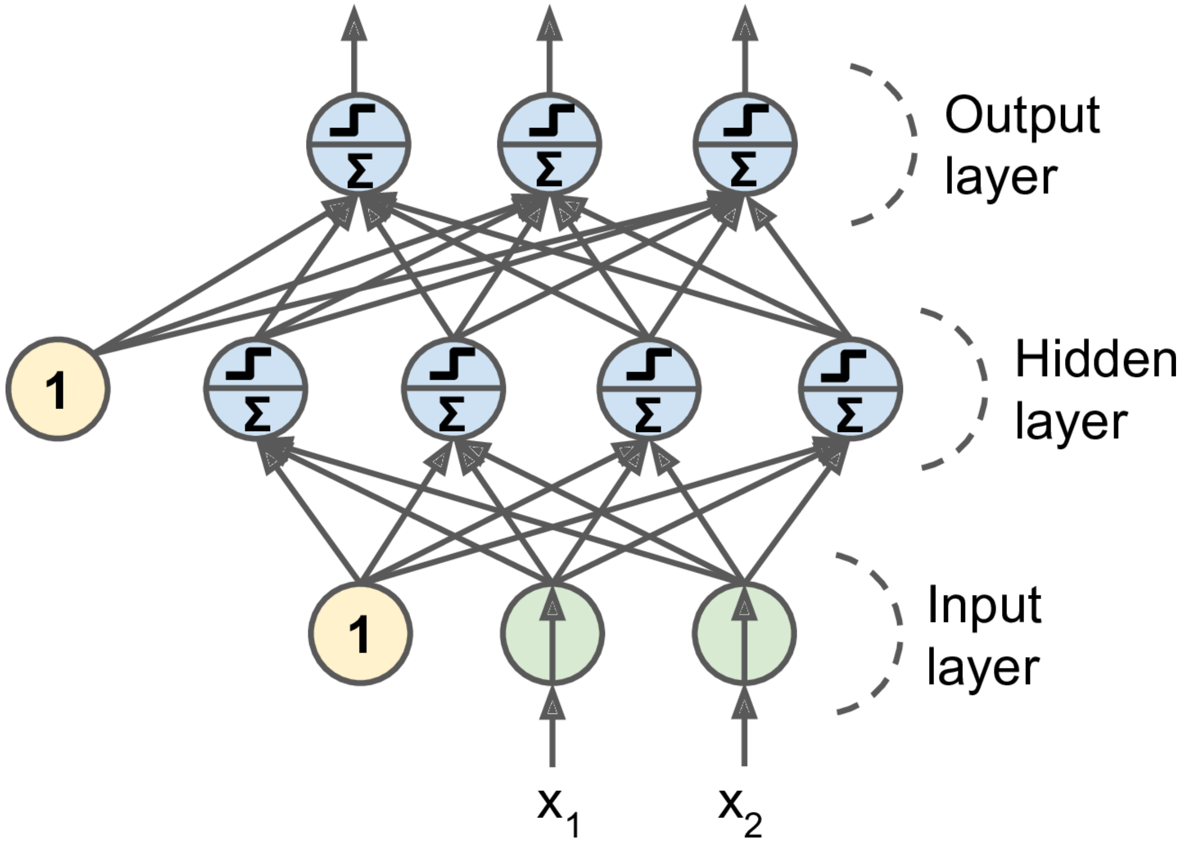


Outline

1. Introduction
2. The perceptron
3. Multi-layer perceptron (MLP)
4. Regression MLPs
5. Classification MLPs

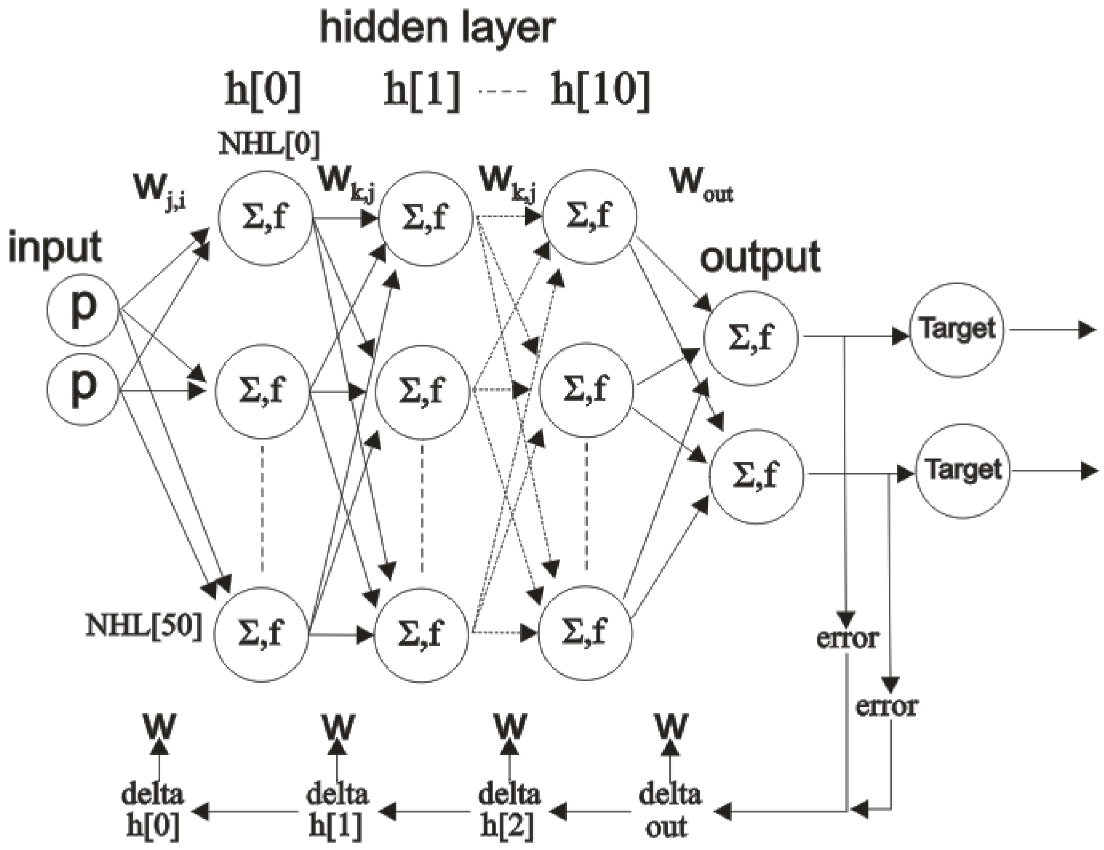
3. Multi-Layer Perceptron (MLP)

- An MLP is composed of a (pass-through) **input layer**, one or more layers of LTUs, called **hidden layers**, and a final layer of LTUs called the **output layer**.
- When an ANN has **two or more** hidden layers, it is called a **deep neural network** (DNN).



3. Multi-Layer Perceptron (MLP)

- Trained using the **backpropagation training algorithm**.
  - For each training instance the algorithm first makes a prediction (**forward pass**), measures the error,
  - then goes through each layer in reverse to measure the error contribution from each connection (**reverse pass**),
  - and finally slightly tweaks the connection weights to reduce the error (**Gradient Descent step**).



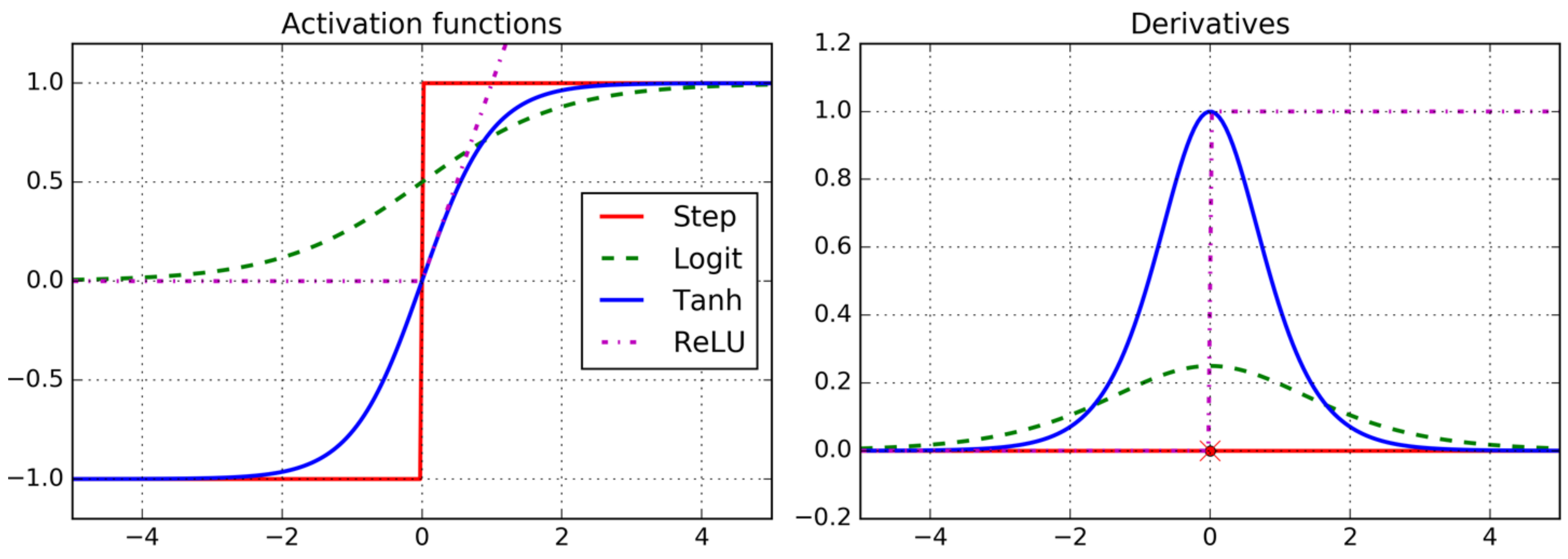
3. Multi-Layer Perceptron (MLP)

- **Common activation functions**: **logistic**, **hyperbolic tangent**, and **rectified linear unit**.

$$\sigma(z) = 1 / (1 + \exp(-z))$$

$$tanh(z) = 2\sigma(2z) - 1$$

$$\mathrm{ReLU}(z) = \max(0, z)$$



Outline

1. Introduction
2. The perceptron
3. Multi-layer perceptron (MLP)
4. Regression MLPs
5. Classification MLPs

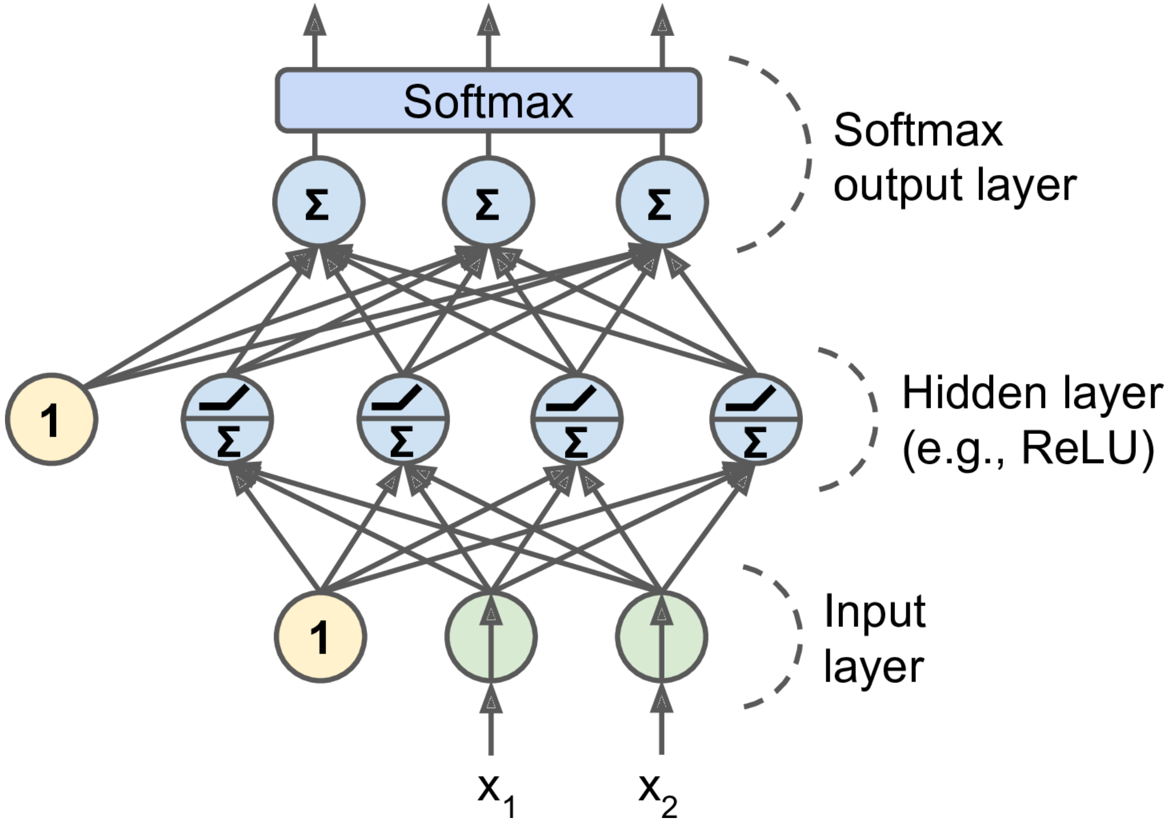
4. Regression MLPs

- Typical MLP architecture for **regression**:

Hyperparameter	Typical Value
# input neurons	One per input feature (e.g., 28 x 28 = 784 for MNIST)
# hidden layers	Depends on the problem. Typically 1 to 5.
# neurons per hidden layer	Depends on the problem. Typically 10 to 100.
# output neurons	1 per prediction dimension
Hidden activation	ReLU (or SELU, see Chapter 11)
Output activation	None or ReLU/Softplus (if positive outputs) or Logistic/Tanh (if bounded outputs)
Loss function	MSE or MAE/Huber (if outliers)

5. Classification MLPs

- For **classification**, the output layer uses the **softmax function**.
- The output of each neuron corresponds to the **estimated probability** of the corresponding class.



$$\hat{p}_k = \sigma(\mathbf{s}(\mathbf{x}))_k = \frac{\exp\left(s_k(\mathbf{x})\right)}{\sum_{j=1}^{K} \exp\left(s_j(\mathbf{x})\right)}$$

$$\hat{y} = \underset{k}{\operatorname{argmax}}\ \sigma(\mathbf{s}(\mathbf{x}))_k$$

5. Classification MLPs

- Typical MLP architecture for **classification**:

Hyperparameter	Binary classification	Multilabel binary classification	Multiclass classification
Input and hidden layers	Same as regression	Same as regression	Same as regression
# output neurons	1	1 per label	1 per class
Output layer activation	Logistic	Logistic	Softmax
Loss function	Cross-Entropy	Cross-Entropy	Cross-Entropy

Summary

1. Introduction
2. The perceptron
3. Multi-layer perceptron (MLP)
4. Regression MLPs
5. Classification MLPs