Co-funded by the
Erasmus+ Programme
of the European Union

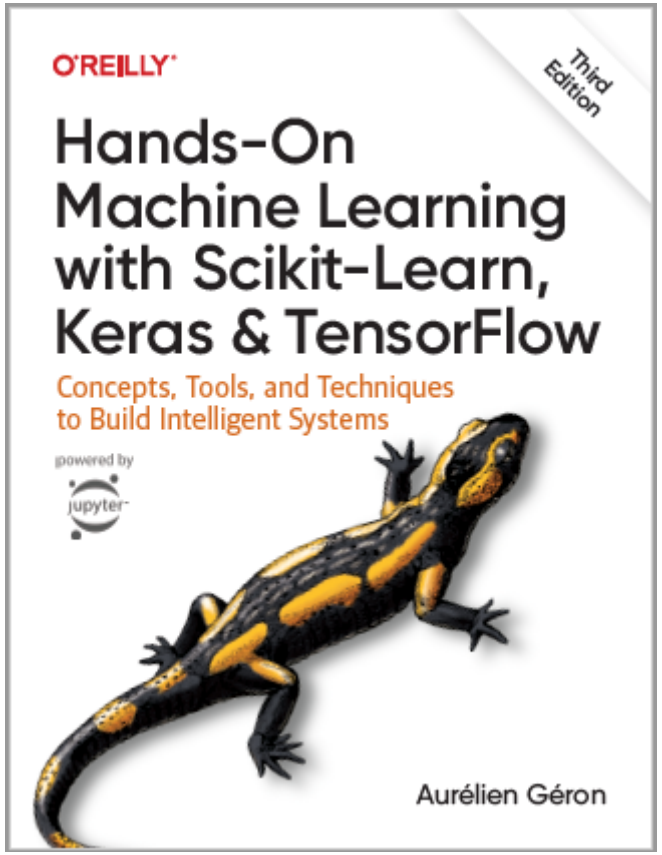
DeCAIR

Recurrent Neural Networks

Prof. Gheith Abandah

Reference

- Chapter 15: **Processing Sequences Using RNNs and CNNs**



- Aurélien Géron, **Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow**, O'Reilly, 3rd Edition, 2022
  - Material: https://github.com/ageron/handson-ml3

Outline

1. Introduction
2. Recurrent neurons and layers
3. Forecasting a time series
    1. Implementing a simple RNN
    2. Deep RNNs
4. Exercises

Introduction

- YouTube Video: **Deep Learning with Tensorflow - The Recurrent Neural Network Model** from Cognitive Class

https://youtu.be/C0xoB8L8ms0

1. Introduction

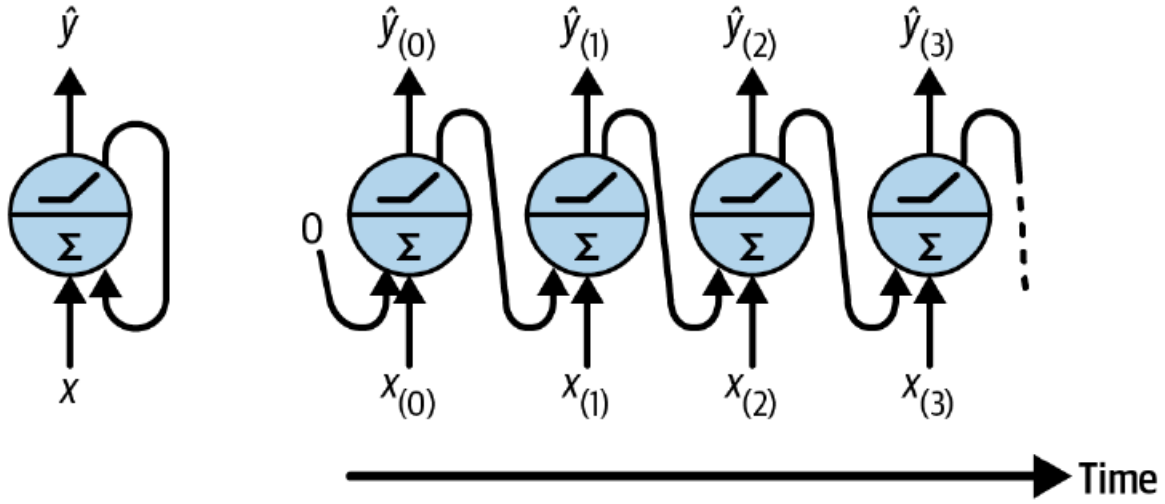
- **Recurrent neural networks (RNNs)** are used to handle time series data or sequences.
- **Applications**:
  - Predicting the future (stock prices)
  - Autonomous driving systems (predicting trajectories)
  - Natural language processing (automatic translation, speech-to-text, or sentiment analysis)
  - Creativity (music composition, handwriting, drawing)
  - Image analysis (image captions)

Outline

1. Introduction
2. Recurrent neurons and layers
3. Forecasting a time series
    1. Implementing a simple RNN
    2. Deep RNNs
4. Exercises

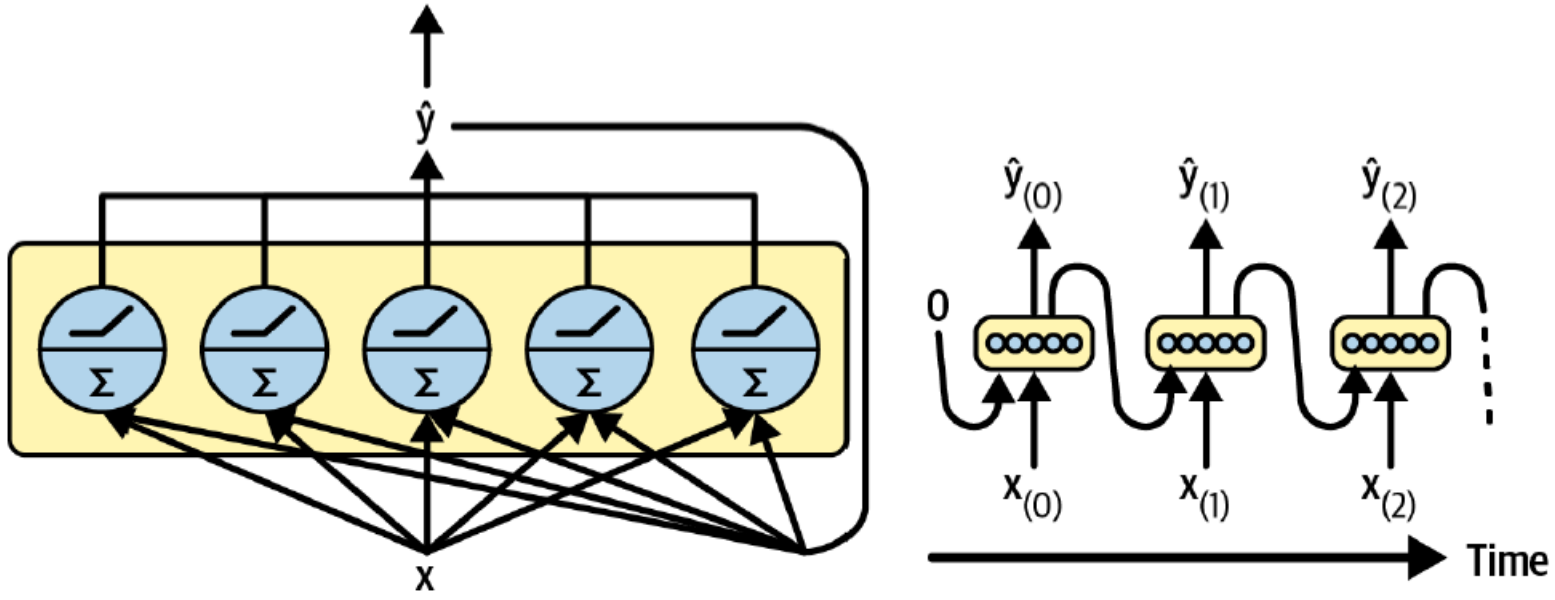
2. Recurrent Neurons and Layers

- The figure below shows a **recurrent neuron** (left), unrolled through time (right).



2. Recurrent Neurons and Layers

- Multiple recurrent neurons can be used in a **layer**.

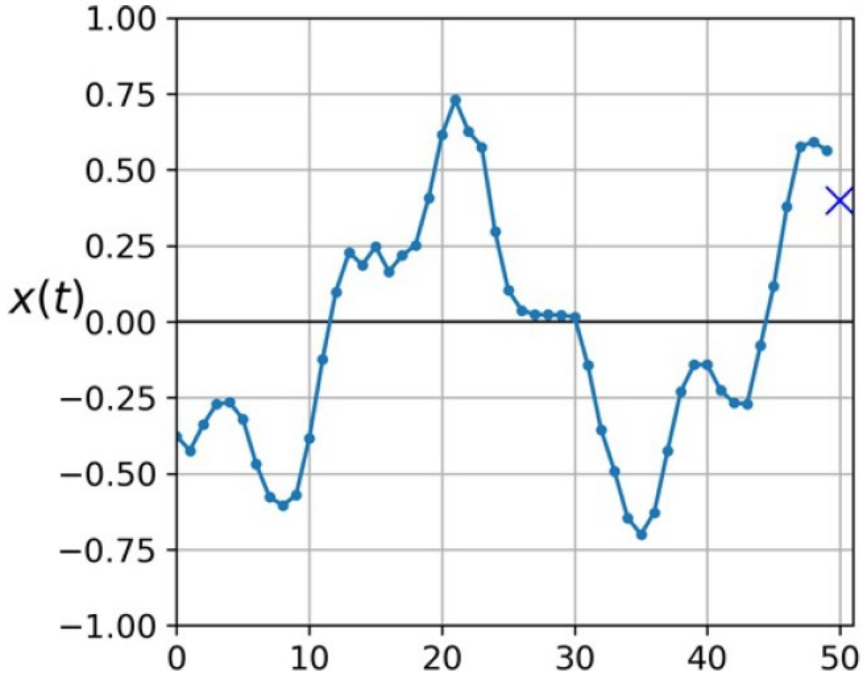


Outline

1. Introduction
2. Recurrent neurons and layers
3. Forecasting a time series
    1. Implementing a simple RNN
    2. Deep RNNs
4. Exercises

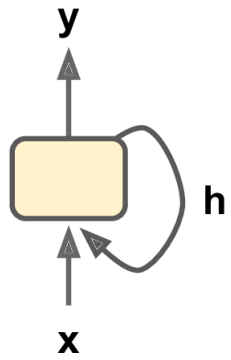
3. Forecasting a Time Series

- The data is a sequence of one or more values per **time step**.
  - **Univariate** time series
  - **Multivariate** time series
- **Forecasting**: predicting future values



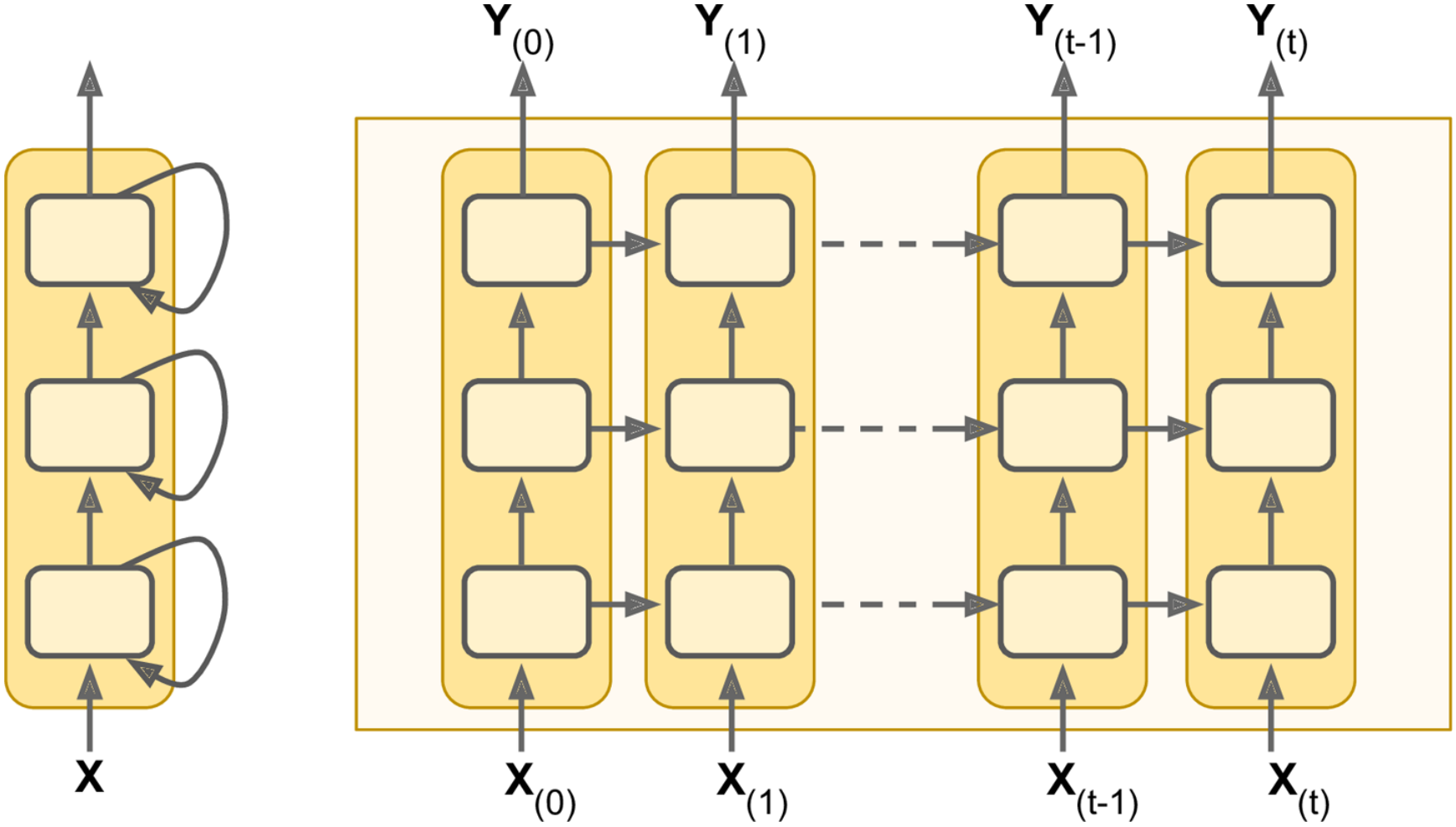
3.1 Implementing a Simple RNN

```
# Sequential model of one neuron
model = keras.Sequential([
        layers.SimpleRNN(1, input_shape=[None, 1])
])
```




```
# MSE = 0.011, Dense achieves 0.004
```

3.2 Deep RNNs



3.2 Deep RNNs

```
# Sequential model of two hidden RNN layers
model = keras.Sequential([
        layers.SimpleRNN(20,
                return_sequences=True, # output all steps
                input_shape=[None, 1]),
        layers.SimpleRNN(20),
        layers.Dense(1)
])

# MSE = 0.0026
```

4. Exercises

15.1. Can you think of a few applications for a sequence-to-sequence RNN? What about a sequence-to-vector RNN, and a vector-to-sequence RNN?

15.2. How many dimensions must the inputs of an RNN layer have? What does each dimension represent? What about its outputs?

15.3. If you want to build a deep sequence-to-sequence RNN, which RNN layers should have return_sequences=True? What about a sequence-to-vector RNN?

Summary

1. Introduction
2. Recurrent neurons and layers
3. Forecasting a time series
    1. Implementing a simple RNN
    2. Deep RNNs
4. Exercises