

Neural Networks: Introduction to Neural Networks in Python

Andrzej Kordecki

Neural Networks for Classification and Identification (ML.EM05): Exercise 01

Division of Theory of Machines and Robots
Institute of Aeronautics and Applied Mechanics
Faculty of Power and Aeronautical Engineering
Warsaw University of Technology

Table of Contents

- 1 Organizational Matters
 - Organizational Matters
 - Bibliography
- 2 Python
 - Introduction
 - Libraries
- 3 Python basics
 - Data types

Organizational Matters

Coarse Course Schedule

Characteristic of the undertaken topics:

- Introduction to Python. Basic Python libraries,
- Neural Networks in Python. Implementation of basic neural network functions,
- Neural networks libraries. Advance of Python with Tensorflow and Keras libraries,

Prerequisites and Evaluation

Prerequisites:

- Knowledge of any programming language.

Evaluation:

- Two activity tasks,
- Project and presentation.

Coarse Course Schedule

Activity tasks:

- After the each of first two parts, a list of tasks to do in Python will be available on the website.
- Tasks solutions should be sent via the MS Forms. The link will be available on website.
- You have a week to submit the assignment.

Project

Project parts:

- Theory:
 - 1 title page + 1 table of content page,
 - 10 - 20 pages - about: purpose of the work, description of the method and description of its implementation, obtained results and short summary,
 - 1 bibliography page - at least 5 positions.
- Program - code with comments. It has to contain all the necessary elements needed to run it, e.g. list of used libraries and images.

Project

Project parts:

- 15 minute Presentation in January. The order of the presentation in accordance with the date of handing over the work. The maximum number of presentations during project classes is 3.

Project organization:

- The project is written in groups of 3 people.
- The project topics will be available after "Introduction to Python" part.

Grades

For task you can achieve 10 points:

- Theory: 0 - 4 points,
- Program: 0 - 2 points,
- Presentation and questions: -2 - 2 points (negative points),
- The project deadline is the end of year (theory + program). Each week of delay: -1 point,
- Activity task: 0 - 2 point,
- 10 points corresponds to 100%.

Bibliography

- Paul Deitel, Harvey Deitel, Python for Programmers: with Big Data and Artificial Intelligence Case Studies, Pearson, 2019
- Aurélien Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and Tensorflow: Concepts, Tools, and Techniques to Build Intelligent Systems, O'Reilly Media, 2019
- Beginner's Guide to Python,
<https://wiki.python.org/moin/BeginnersGuide>
- TensorFlow tutorials,
<https://www.tensorflow.org/tutorials>

Python

Introduction

This simple idea of giving humans priority over machines is at the core of the philosophy behind Python.

Anthony Wing Kosner, Guido van Rossum on how Python makes thinking in code easier, 2019

Introduction

The programming language Python implementation was started by Guido van Rossum in 1989. Van Rossum is Python's principal author, and his continuing central role in deciding the direction of Python.

- Python is an easy to learn, powerful programming language.
- Python is an interpreted language,
- Python enables programs to be written compactly and readably.

Introduction

Large Python community:

- Interpreter and the extensive standard library are freely available from the Python Web site.
- Easily available documentation, GitHub projects and large community support.

Python versions:

Python A.B.C e.g. 3.9.7

- A is the major version number - major changes in the language,
- B is the minor version number - less earth-shattering changes,
- C is the micro version number – mostly bugfix release.

Python versions

Python webpage:

`http://www.python.org`

Remarks about Python versions:

- You should use the latest major version of Python, but not necessarily the latest minor version.
- Python language is growing rapidly and often its latest version is not supported by all libraries.
- You can check your installed Python version at the command line by running: "python -version".

Python versions

The anaconda is the distribution of the Python and R programming languages for data science and machine learning related applications that aims to simplify package management and deployment.

The following applications are available by default:

- Python and R language,
- IDE: Spyder, JupyterLab and Jupyter Notebook,
- Installed libraries: scikit-image, scikit-learn, scipy and many more,
- Glueviz, QtConsole, Rstudio and Visual Studio Code

GPU support

Calculation time on GPU is usually drastically shorter than on CPU. Full GPU support needs:

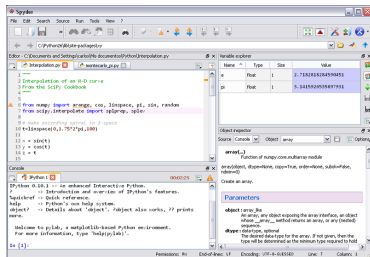
- NVIDIA GPU and CUDA drivers,
- NVIDIA CUDA Toolkit,
- cuDNN SDK.

Additionally:

- If an error unidentified occurs after running a program, you have to check all above programs version.
Installation need all drivers and SDK in proper version.
- Alternative is a free cloud server.

IDE

Python IDEs (Integrated Development Environment):



- PyCharm (www.jetbrains.com/pycharm/),
- Jupyter Notebook (jupyter.org),
- Visual Studio Code (<https://visualstudio.microsoft.com/pl/>),
- Visual Studio Community (<https://visualstudio.microsoft.com/pl/>).

Invoking the Interpreter

Running script (MS Windows):

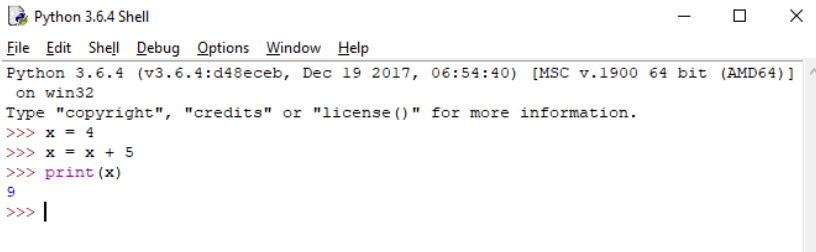
- 1 Open Command line (Start menu → Run and type cmd),
- 2 Type: "python script.py" (or full path e.g.
C:\python\python.exe D:\code\script.py)

Install libraries (MS Windows):

- 1 Open Command line (Start menu → Run and type cmd),
- 2 Python, type: pip3 install LibraryName,
- 3 Anaconda, type: conda install LibraryName.

Interactive Mode

The code can be executed with use of Python command window with the primary prompt, usually three greater-than signs (`>>>`):



```
Python 3.6.4 Shell
File Edit Shell Debug Options Window Help
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:54:40) [MSC v.1900 64 bit (AMD64)]
on win32
Type "copyright", "credits" or "license()" for more information.
>>> x = 4
>>> x = x + 5
>>> print(x)
9
>>> |
```

The line continuation - default `'\'`.

Libraries

Python libraries

List some of the most commonly used Python modules in Neural Network processing:

- Tensorflow - library for machine learning applications such as neural networks,
- PyTorch - library for computer vision and natural language processing,

Python libraries

List some of the most commonly used Python modules:

- Numpy - the fundamental package for scientific computing.
- Matplotlib - 2D plotting library which produces publication quality figures in a variety of formats.
- Sklearn - tools for image processing, machine learning, data mining and data analysis.
- SciPy - tools for optimization, linear algebra, integration, interpolation, FFT, signal and image processing, ODE solvers and others.

Python libraries

List some of the most commonly used Python modules:

- OpenCV - library for real-time computer vision.
- Pandas - library providing high-performance data structures and data analysis tools.
- logging - logging framework included in standard python libraries.
- tqdm - allows to show a smart progress meter in loops.

Python basics

Basic data types

Python has a number of basic data types:

- logic: bool,
- numerics: int, float, complex,
- text: str,
- sequences: list, tuple,
- mappings: dict
- classes,
- instances
- exceptions.

Basic data types

Integer operation:

```
x = 5
print(type(x)) # Prints "<class 'int'>"
print(x)       # Prints "5"
print(x + 1)    # Prints "6"
print(x * 2)    # Prints "10"
print(x ** 2)   # Prints "25"
x += 1
print(x)       # Prints "6"
```

Python does not have unary increment ($x++$) or decrement ($x--$) operators.

Basic data types

Python does not require any type declaration or dimension statements:

```
x = 5  
print(type(x)) # Prints "<class 'int'>"
```

```
x = 5.43  
print(type(x)) # Prints "<class 'float'>"
```

```
x = "dog"  
print(type(x)) # Prints "<class 'str'>"
```

Python variables are used more as objects than data types.

Booleans

Python implements all of the usual operators for Boolean logic, but uses English words rather than symbols (e.g. `||` symbol in C#):

```
t = True # = 1  
f = False # = 0
```

```
print(t and f) # prints "False"  
print(t or f)  # prints "True"  
print(not t)   # prints "False"  
print(t != f)  # prints "True"
```

Strings

String type:

```
hello = 'hello'
world = "world"
print(hello)          # Prints "hello"
print(len(hello))     # String length, Prints "5"

txt = hello + ' ' + world
print(txt)            # prints "hello world"

print( hello, world)  # prints "hello world"
```

Strings

Strings support styles:

```
# Style formatting
```

```
hw2 = '%f %s %s %d' % (3.14, hello, world, 0)
```

```
print(hw2)
```

```
# prints "3.140000 hello world 0"
```

```
print('{0} {1} {2}'.format(3.14, hello, world, 0))
```

```
# formatted string literals starts with f or F
```

```
print(f'{0} {hello} {world}')
```

```
# both prints "3.14 hello world"
```

String formatting

String formatting provides a large degree of flexibility and customization (str.format).

```
print('{} {}'.format('one', 'two'))
```

```
# Prints "one two"
```

```
print('{} {}'.format(1, 2))
```

```
# Prints "1 2"
```

```
print('{x} {y}'.format(x=1, y='two'))
```

```
# Prints "1 two"
```

```
# Order
```

```
print('{0} {1}'.format(1, 2)) # Prints "1 2"
```

```
print('{1} {0}'.format(1, 2)) # Prints "2 1"
```


String formatting

Padding

```
print('{:>10}'.format('text')) # Prints "          text"  
print('{:10}'.format('text'))  # Prints "text          "  
print('{:^10}'.format('text')) # Prints "    text    "
```

Precision

```
print('{:d}'.format(3))  
# Prints "3"  
print('{:f}'.format(3.141592653))  
# Prints "3.141593"  
print('{:.2f}'.format(3.141592653))  
# Prints "3.14"
```

List

A list is the Python equivalent of an array, but is resizable and can contain elements of different types:

```
xs = [3, 1, 2]
print(xs, xs[2]) # Prints "[3, 1, 2] 2"

# Negative indices count from the end of the list,
print(xs[-1])    # Prints "2"

# Lists can contain elements of different types
xs[2] = 'world'
print(xs)         # Prints "[3, 1, 'world']"
```

List

We can add, subtract or change elements of List:

```
xs = [3, 1, 2]
```

```
# Add a new element to the end of the list
```

```
xs.append('tree')
```

```
print(xs)           # Prints "[3, 1, 2, 'tree']"
```

```
# Remove and return the last element of the list
```

```
x = xs.pop() # Prints "[3, 1, 2]"
```

```
# Remove value 3, but not element with index 3
```

```
xs.remove(3) # Prints "[1, 2]"
```

List

We can add, subtract or change elements of List:

```
x1 = [3, 1, 2, 12 ,1]
```

```
x2 = [34, 21, 42]
```

```
# We concatenate two lists
```

```
x1 = x1 + x2
```

```
# Print "[3, 20, 21, 22, 23, 12, 1, 34, 21, 42]"
```

```
# Number of elements can be different
```

```
x1[1:2] = [20, 21, 22]
```

```
# Print "[3, 20, 21, 22, 23, 12, 1]"
```

```
# But results can be sometimes not so obvious
```

```
x2 = x2*2 # Print "[34, 21, 42, 34, 21, 42]"
```

List

Data operation typical to Python:

```
x = [99, 101, 1003]
```

```
x_1 = x          # Prints "[99, 101, 1003]"
```

```
x_2 = list(x)    # Prints "[99, 101, 1003]"
```

```
x_3 = x.copy()   # Prints "[99, 101, 1003]"
```

```
x[:] = [102, 111, 143, 199]
```

```
x_1 # Prints "[102, 111, 143, 199]"
```

```
x_2 # Prints "[99, 101, 1003]"
```

```
x_3 # Prints "[99, 101, 1003]"
```

List

Python provides concise syntax to access sublists:

```
nums = list(range(5))  
print(nums)           # Prints "[0, 1, 2, 3, 4]"  
print(nums[2:4])      # Prints "[2, 3]"  
print(nums[2:])        # Prints "[2, 3, 4]"  
print(nums[:2])        # Prints "[0, 1]"  
print(nums[:])         # Prints "[0, 1, 2, 3, 4]"  
print(nums[::2])       # Prints "[0, 2, 4]"
```

Tuples

A tuple is a collection which is ordered and unchangeable (immutable):

- The most important difference between list and tuple is that tuples can be used as keys in dictionaries and as elements of sets, while lists cannot.
- Tuple is rarely used when is not obligatory (e.g. in classes, dict).

```
t = (5, 6)           # Create a tuple (brackets)
print(type(t))      # Prints "<class 'tuple'>"
print(d)            # Prints "{(0, 1): 0, (1, 2): 1,...}"
print(d[t])         # Prints "5"
print(d[(1, 2)])    # Prints "1"
```

Set

A set is an unordered collection with no duplicate elements:

```
# Create a new set
```

```
basket = {'apple', 'orange', 'apple',  
          'pear', 'orange'}
```

```
# Duplicates have been removed
```

```
print(basket) # Prints {"'orange', 'apple', 'pear'}"
```

```
print(basket) # Prints {"'apple', 'pear', 'orange'}"
```

```
print(basket) # Prints {"'pear', 'apple', 'orange'}"
```


Dictionaries

A dictionary is a collection which is unordered, changeable and indexed. But, dictionary stores (key, value) pairs:

```
# Create a new dictionary with some data  
d = {'car': 'small', 'truck': 'big'}
```

```
# Set an entry in a dictionary  
d['bike'] = 'tiny'  
print(d['car'])          # Prints "small"  
print('car' in d)        # Prints "True"
```

```
# Remove an element from a dictionary  
del d['bike']
```

Dictionaries

We usually use nested dictionaries:

```
car = {"brand": "Ford", "model": "Mustang", "year": 1956}
```

```
cars = {  
    "car1": {"brand": "Ford",  
             "model": "Mustang",  
             "year": 1956 },  
    "car2": {"brand": "Opel",  
             "model": "Corsa",  
             "year": 2018 },  
    "car3": {"brand": "KIA",  
             "model": "Ceed",  
             "year": 2020 }  
}
```

Dictionaries vs List

Dictionaries and lists similarities:

- Both are mutable,
- Both are dynamic (allows to add and remove elements),
- Both can be nested (a list can contain another list).

Dictionaries and lists differences:

- List elements are accessed by their position in the list (index).
- Dictionary elements are accessed via keys.