ISLab

Universidade do Minho
Escola de Engenharia
Departamento de Informática

Mestrado Integrado em Engenharia Informática
Mestrado em Engenharia Informática
Computação Natural
2020/2021

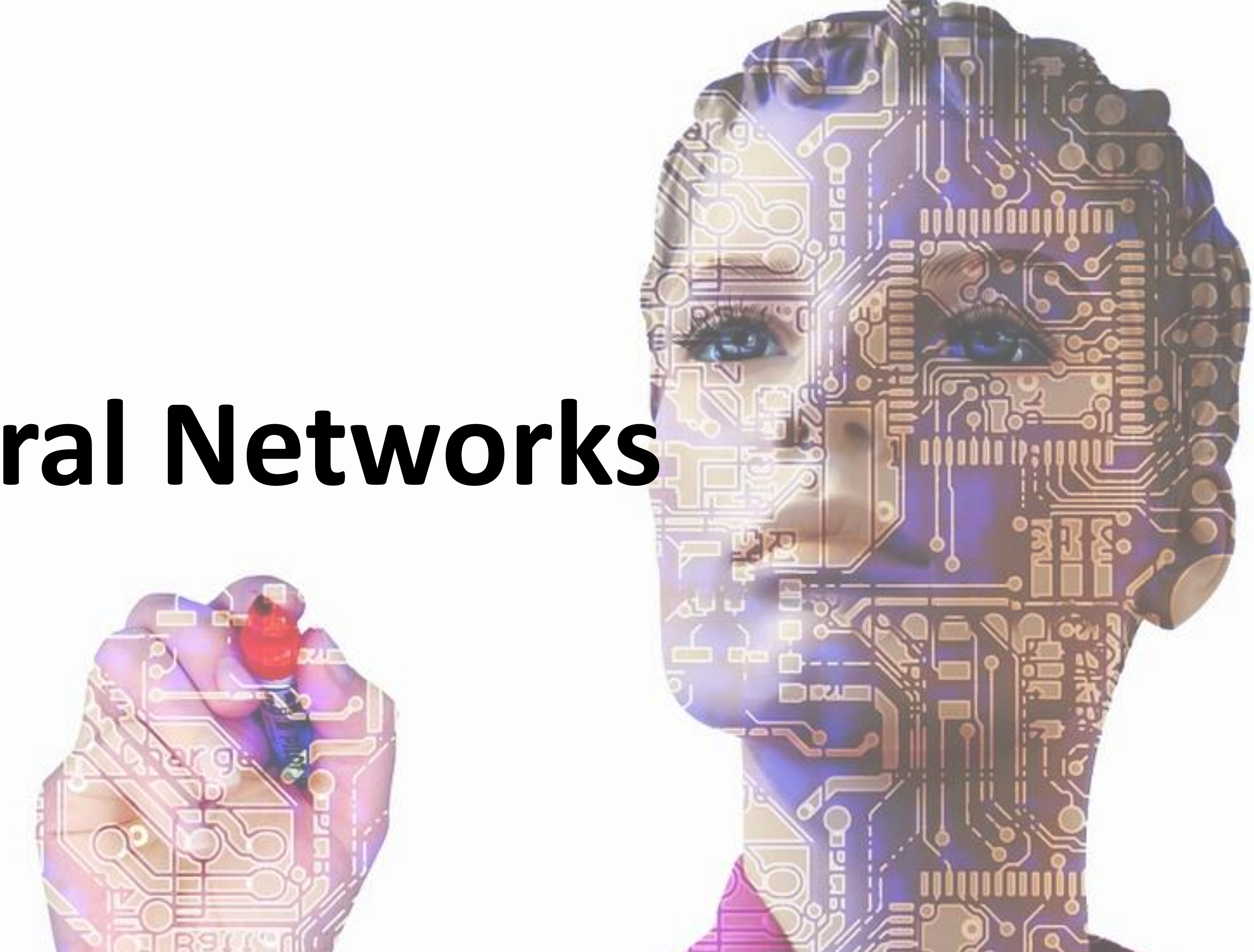
Filipe Gonçalves, Paulo Novais, Cesar Analide

Computação Natural@ 2020/2021

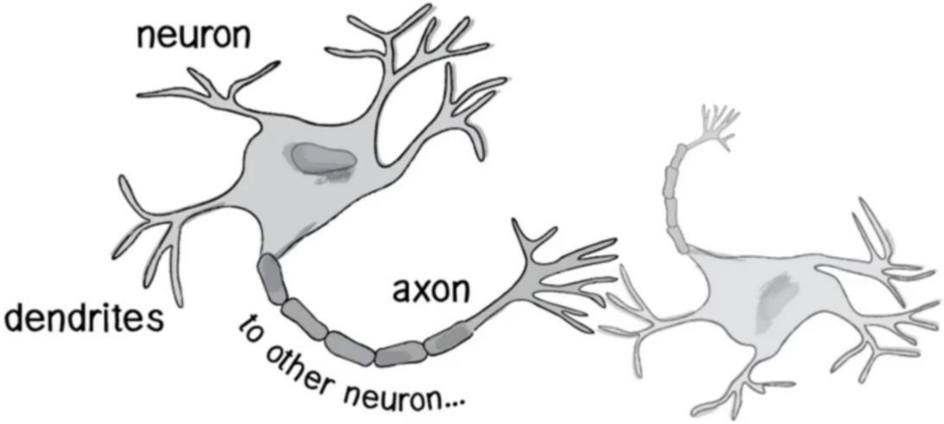
- Paulo Novais – pjon@di.uminho.pt
- Cesar Analide – cesar.analide@di.uminho.pt
- Filipe Gonçalves – fgoncalves@algoritmi.uminho.pt

- Departamento de Informática
  Escola de Engenharia
  Universidade do Minho
- Grupo ISLab – (Synthetic Intelligence Lab)
- Centro ALGORITMI
  Universidade do Minho

Neural Networks

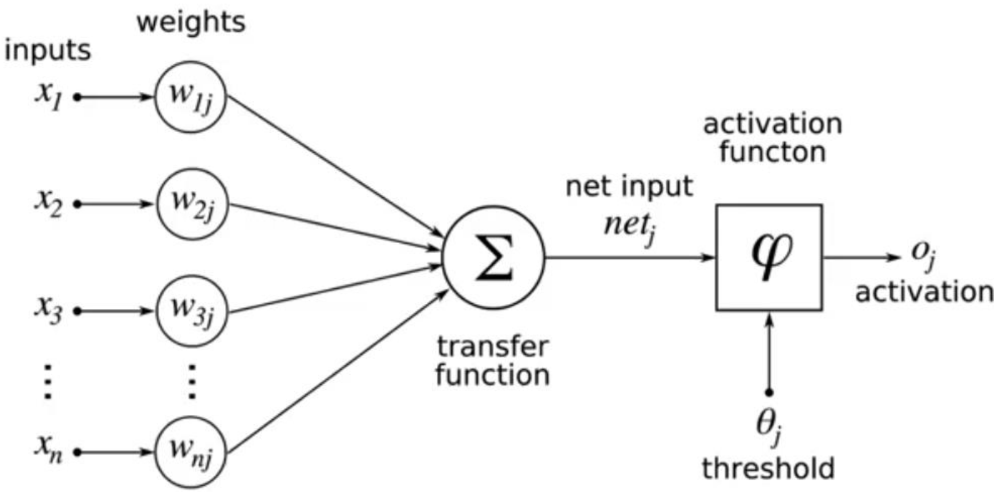


- Artificial neural networks are modeled after biological neural networks and attempt to allow computers to learn in a similar manner to humans
- The human brain has interconnected neurons that receive inputs, and then based on those inputs, produce an electrical signal output through the axon
- Use cases:
  - Pattern Recognition
  - Time Series Predictions
  - Signal Processing
  - Anomaly Detection
  - Control



- There are problems that are complex for humans but easy for computers
  - E.g. calculating large arithmetic problems
- Then there are problems easy for humans, but difficult for computers
  - E.g. recognizing a picture of a person from the side
- Neural networks attempt to solve problems that would normally be easy for humans but hard for computers!
- Let’s start by looking at the simples neural network possible – the perceptron

- Neural Networks are composed by:
  - Input Layer
  - Hidden Layer
  - Output Layer
  - Weights and Biases between Layers
  - Activation Function

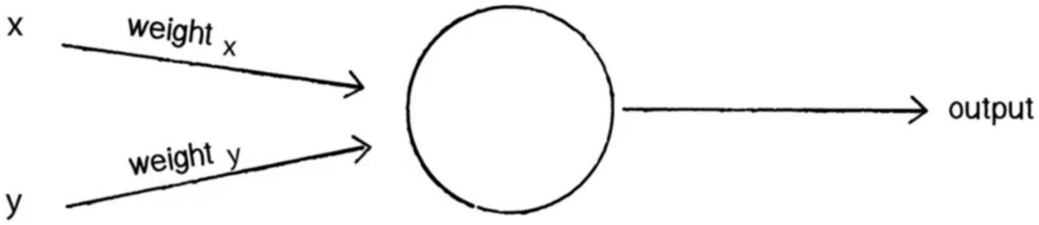


- A perceptron consists of one or more inputs, a processor, and a single output

- A perceptron follows the “feed-forward” model, meaning inputs are sent into the neuron, are processed, and result in an output

- A perceptron process follows 4 main tasks:

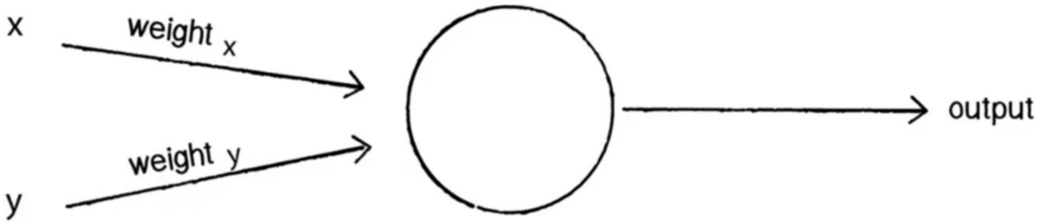
1. Receive Inputs
2. Weight Inputs
3. Sum Inputs
4. Generate Output



- Say we have a perceptron with two inputs:
  - Input 0: x1 = 12
  - Input 1: x2 = 4

- Each input that is sent into the neuron must first be weighted, i.e. multiplied by some value (often a number between [-1,1])

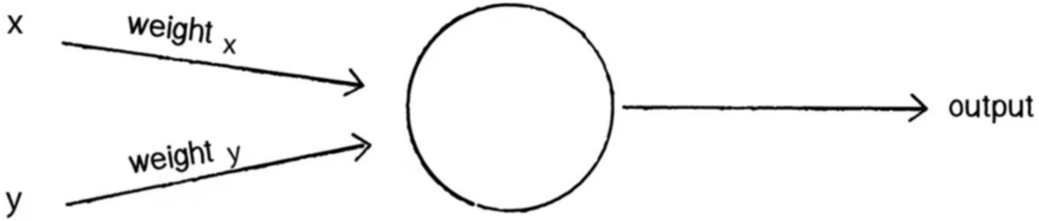
- When creating a perceptron, we'll typically begin by assigning random weights
  - Weight 0: 0,5
  - Weight 1: -1



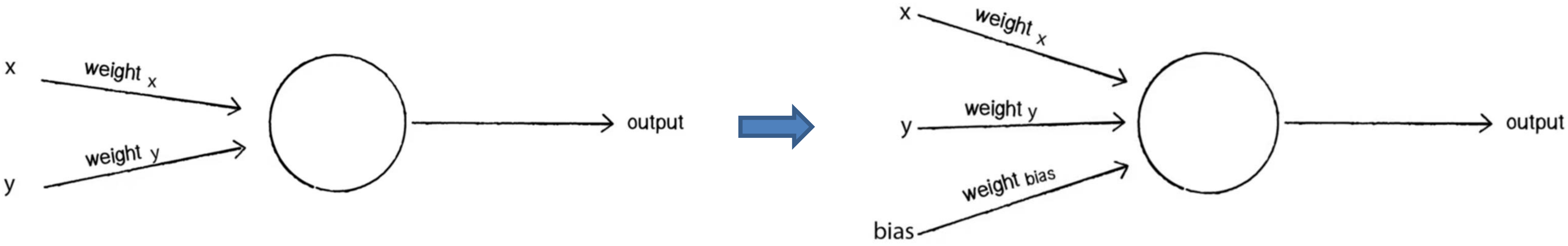
- We take each input and multiply it by its weight
  - Input 0 x Weight 0 = 12 x 0,5 = 6
  - Input 1 x Weight 1 = 4 x (-1) = -4

→ 6+(-4) = 2 [Since 2 >= 0 (binary activation function), perceptron outputs the value 1]

- The output of a perceptron is generated by passing that sum through an **activation function**. In the case of a simple binary output, the activation function is what tells the perceptron whether to "fire" or not
- Many activation functions to choose from (Logistic, Trigonometric, Step, etc..). Let's make the activation function the sign of the sum. In other words: if the sum is a positive number, the output is 1; if it is negative, the output is -1



- One more thing to consider is the Bias. Imagine that both inputs were equal to zero, then any sum no matter what multiplative weight would also be zero!
- To avoid this problem, we add a third input known as bias input with a value of 1. This avoids the zero issue!



- To actually train the perceptron we use the following steps:

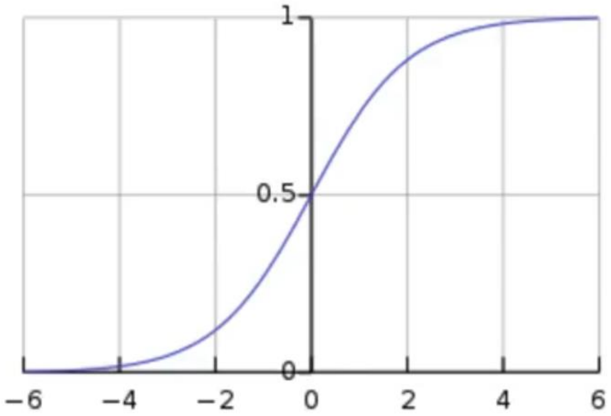
1. Provide the perceptron with inputs for which there is a known answer
2. Ask the perceptron to guess an answer
3. Compute the error (how far off from the correct answer?)
4. Adjust all the weights according to the error
5. Return to Step 1 and repeat!

- We repeat this until we reach an error we are satisfied with

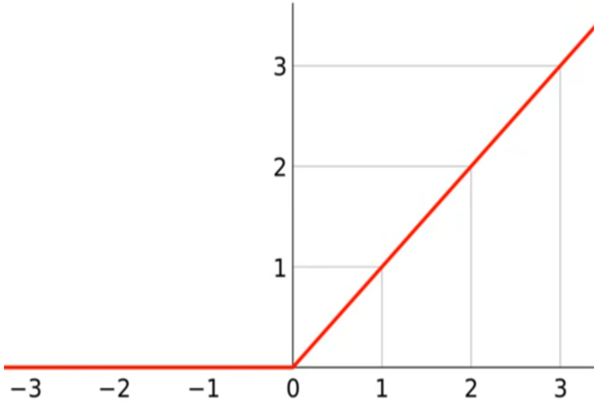
- That is how a single perceptron would work, now to create a neural network all you have to do is link many perceptrons together in layers!

Activation Functions

- Sigmoid Activation Function



- ReLU Activation Function

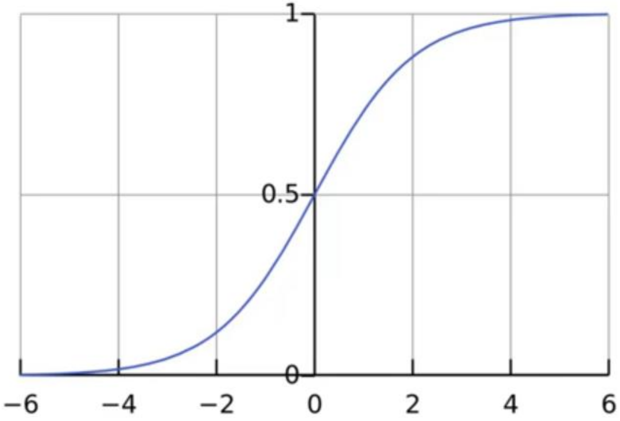


softmax

- Used for classification
  - Chooses the most probable classification given several input values
  - It produces a probability for each class
  - The class with the highest probability is the “answer” you get

$$h_\theta(x) = \frac{1}{1 + \exp(-\theta^T x)},$$

x is a vector of input values
theta is a vector of weights



Optional: Anaconda Virtual Environments

- Virtual Environments allow you to set up virtual installations of Python and libraries on your computer
- You can have multiple versions of Python or libraries and easily activate or deactivate these environments
- Let’s see some examples of why you may want to do this

- Sometimes you'll want to program in different versions of a library
- For example:
  - You develop a program with SciKit-Learn 0.17
  - SciKit-Learn 0.18 is released
  - You want to explore 0.18 but don't want your old code to break

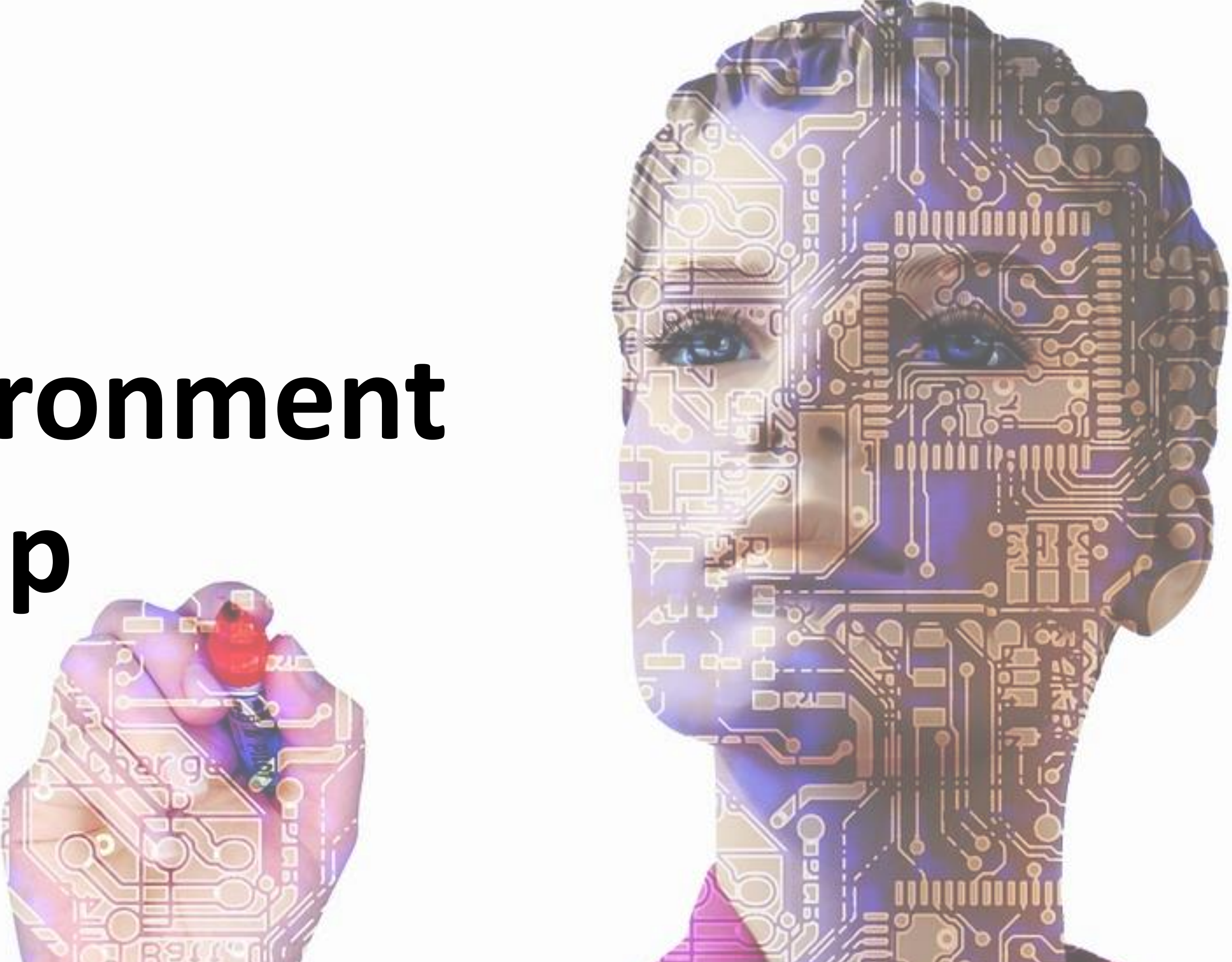
- Sometimes you'll want to make sure your library installations are in the correct location
- For example:
  - You want multiple versions of Python on your computer
  - You want one environment with Python 2.7 and another with Python 3.6

- Anaconda has a built-in virtual environment manager that makes the whole process really easy
- Check out the resource link for the oficial documentation:
  - http://conda.pydata.org/docs/using/envs.html

- Command Prompt Example (create env. and activate it):

```
conda create –-name mypython3version python=3.6 numpy
conda info --evns
activate mypython3version
python
import numpy as np
import pandas as pd     -> Error
quit()
conda install pandas
deactivate
```

Environment Setup



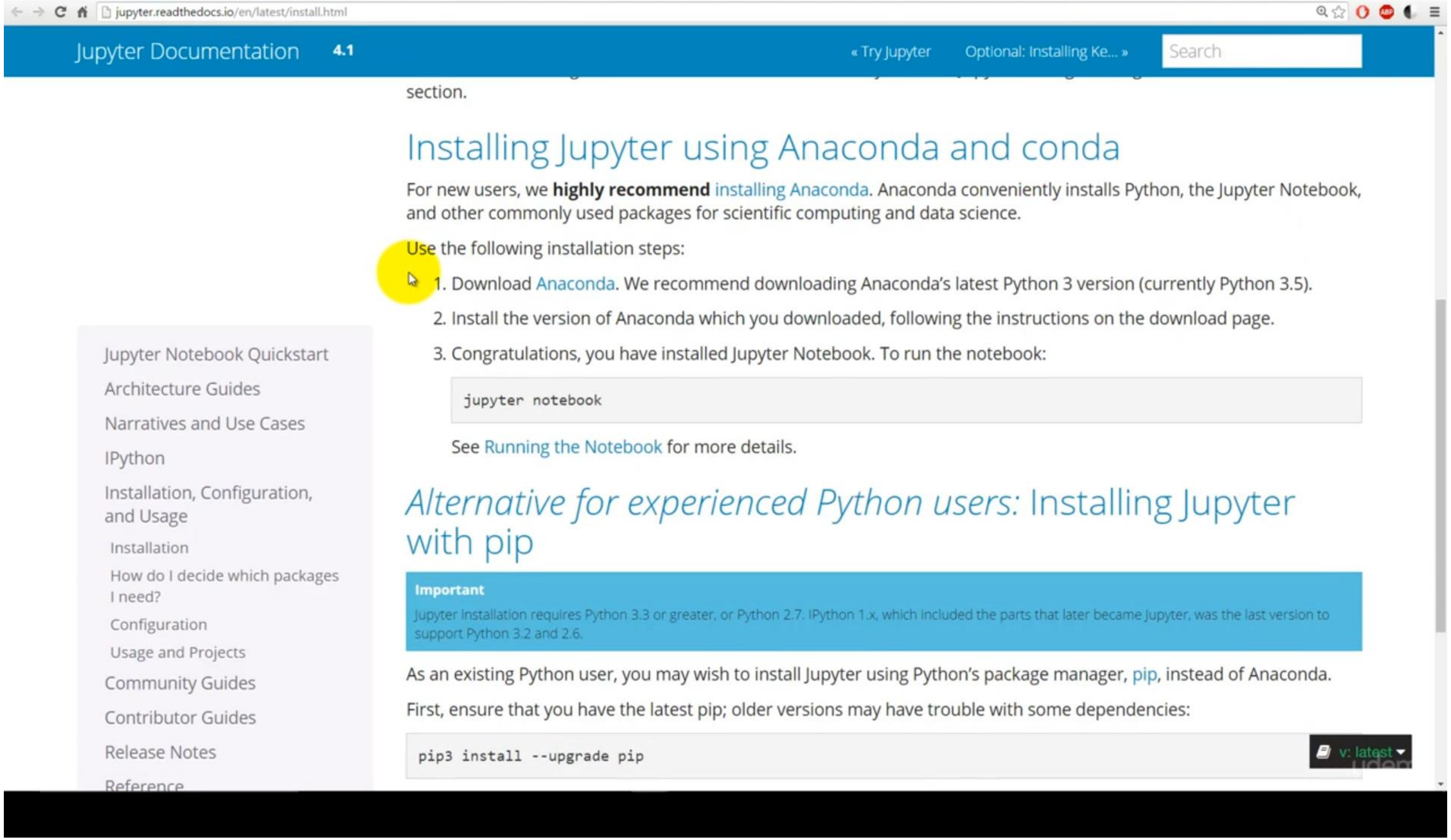
- This course will use Jupyter Notebooks for teaching and to provide notes
- **Note**: you are free to use **whatever development environment you prefer**

- We will be using the Python 3.6 for this course through the Anaconda Distribution
- Now let’s go over your installation options for Jupyter Notebook!

- For experienced users who already have Python
  - As an existing Python user, you may wish to install Jupyter and required APIs using Python’s package manager pip, instead of Anaconda
  - Just go to your command prompt or terminal and use:

```
pip install jupyter
```

- For new users, we highly recommend installing Anaconda
  - Anaconda conveniently installs Python, the Jupyter Notebook, and other commonly used packages for scientific computing and data science
  - Let’s go to www.jupyter.org to walkthrough the installation steps!



jupyter.readthedocs.io/en/latest/install.html
Jupyter Documentation 4.1
« Try Jupyter
Optional: Installing Ke... »
Search
v: latest

section.

Installing Jupyter using Anaconda and conda

For new users, we **highly recommend** installing Anaconda. Anaconda conveniently installs Python, the Jupyter Notebook, and other commonly used packages for scientific computing and data science.

Use the following installation steps:

1. Download Anaconda. We recommend downloading Anaconda's latest Python 3 version (currently Python 3.5).
2. Install the version of Anaconda which you downloaded, following the instructions on the download page.
3. Congratulations, you have installed Jupyter Notebook. To run the notebook:

```
jupyter notebook
```

See Running the Notebook for more details.

Alternative for experienced Python users: Installing Jupyter with pip

Important

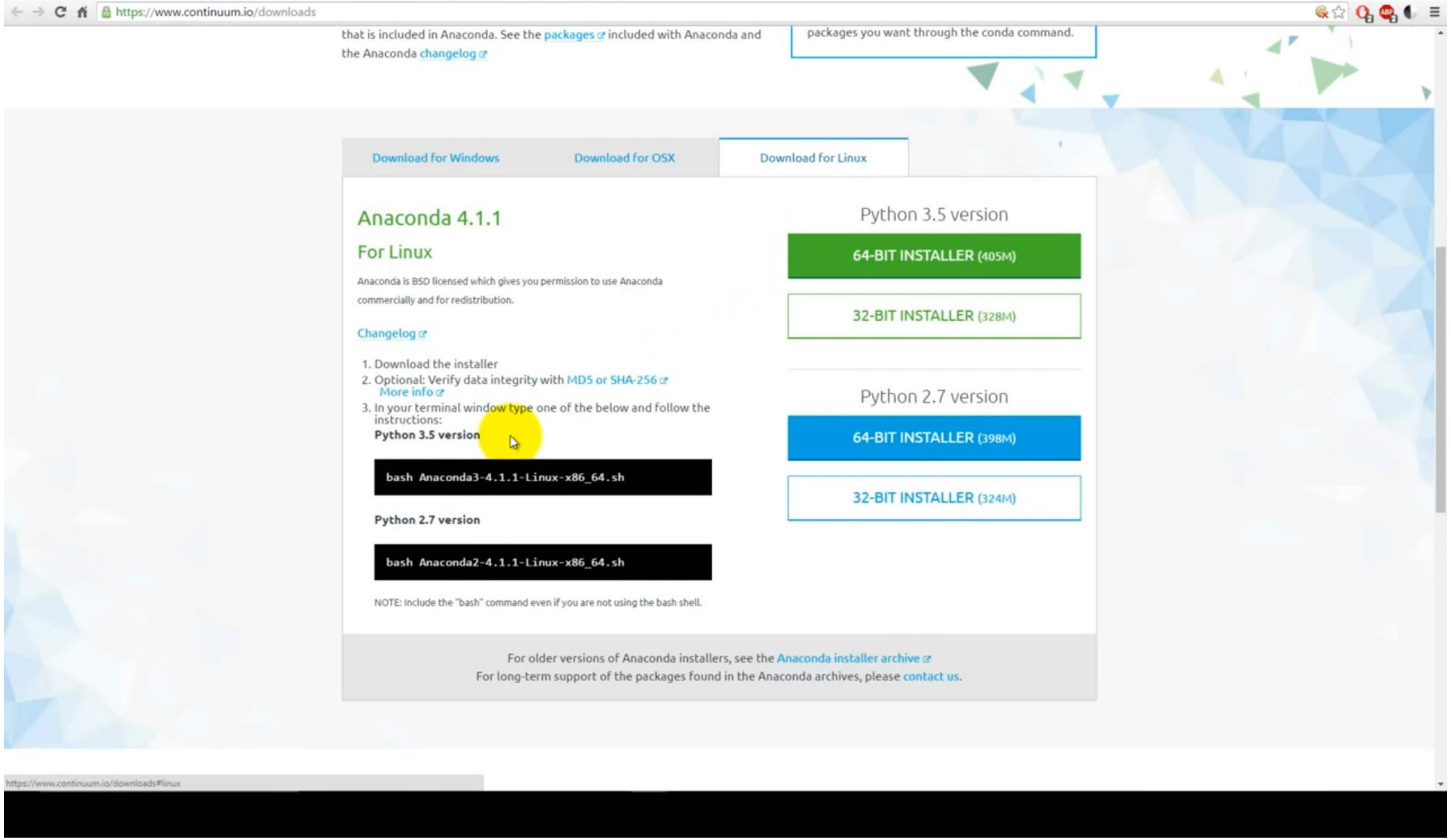
Jupyter installation requires Python 3.3 or greater, or Python 2.7. IPython 1.x, which included the parts that later became Jupyter, was the last version to support Python 3.2 and 2.6.

As an existing Python user, you may wish to install Jupyter using Python's package manager, pip, instead of Anaconda.

First, ensure that you have the latest pip; older versions may have trouble with some dependencies:

```
pip3 install --upgrade pip
```

Computação Natural@ 2020/2021



that is included in Anaconda. See the packages included with Anaconda and the Anaconda changelog

packages you want through the conda command.

Anaconda 4.1.1

For Linux

Anaconda is BSD licensed which gives you permission to use Anaconda commercially and for redistribution.

Changelog

1. Download the installer
2. Optional: Verify data integrity with MD5 or SHA-256 More info
3. In your terminal window type one of the below and follow the instructions:

Python 3.5 version

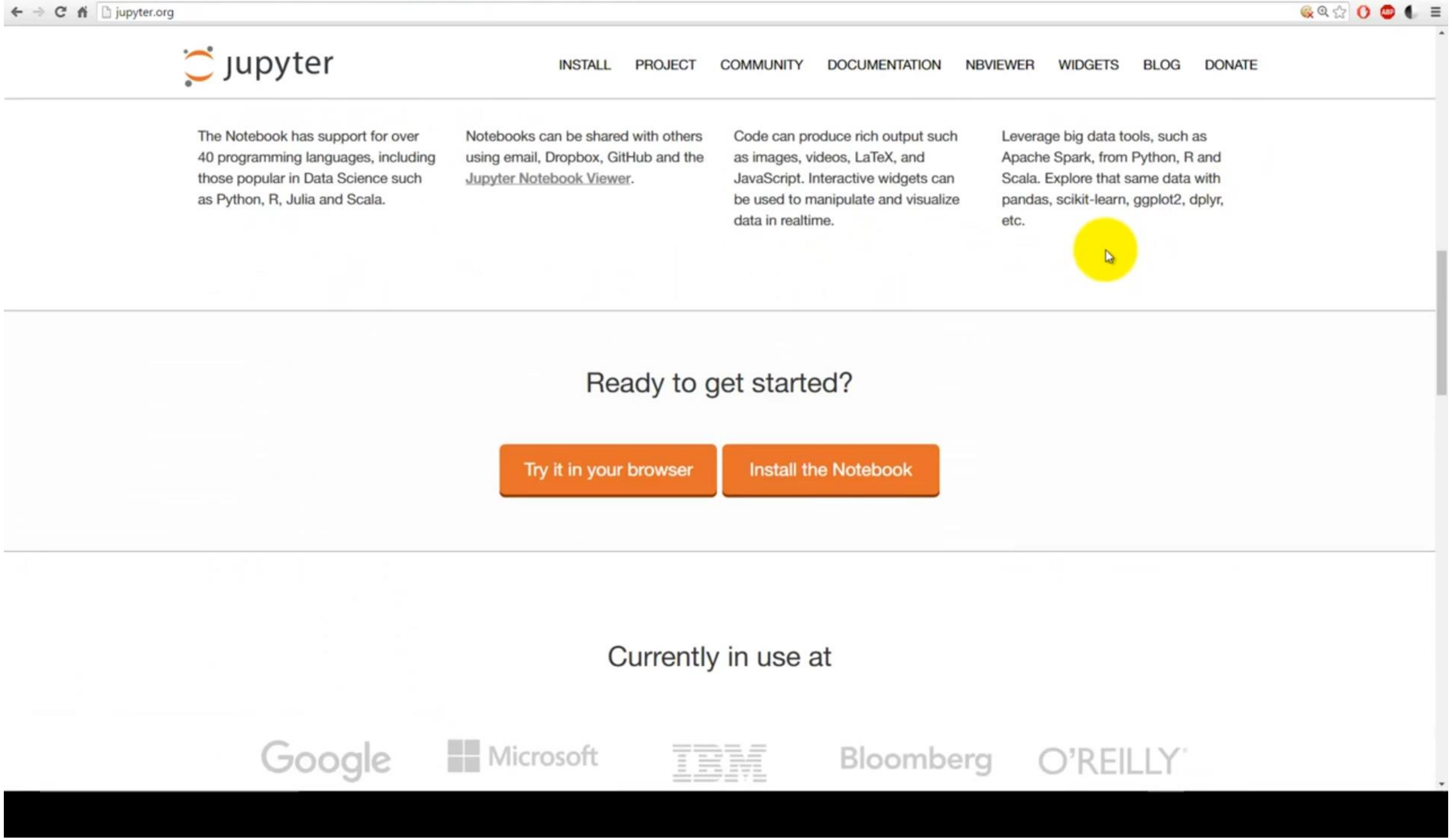
```
bash Anaconda3-4.1.1-Linux-x86_64.sh
```

Python 2.7 version

```
bash Anaconda2-4.1.1-Linux-x86_64.sh
```

NOTE: Include the "bash" command even if you are not using the bash shell.

For older versions of Anaconda installers, see the Anaconda installer archive
For long-term support of the packages found in the Anaconda archives, please contact us.



The Notebook has support for over 40 programming languages, including those popular in Data Science such as Python, R, Julia and Scala.

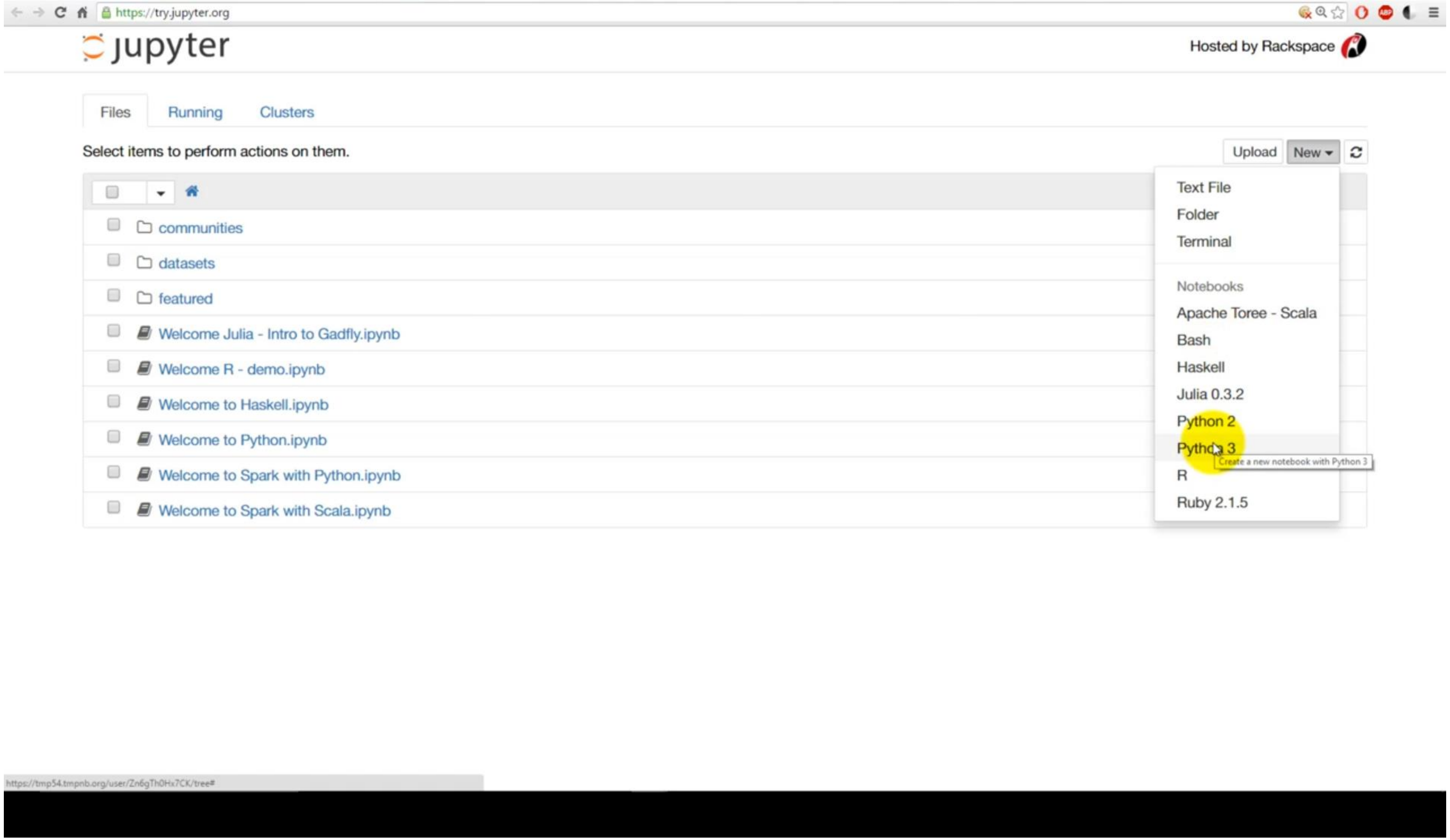
Notebooks can be shared with others using email, Dropbox, GitHub and the Jupyter Notebook Viewer.

Code can produce rich output such as images, videos, LaTeX, and JavaScript. Interactive widgets can be used to manipulate and visualize data in realtime.

Leverage big data tools, such as Apache Spark, from Python, R and Scala. Explore that same data with pandas, scikit-learn, ggplot2, dplyr, etc.

Ready to get started?

Currently in use at



https://try.jupyter.org

jupyter

Hosted by Rackspace

Files | Running | Clusters

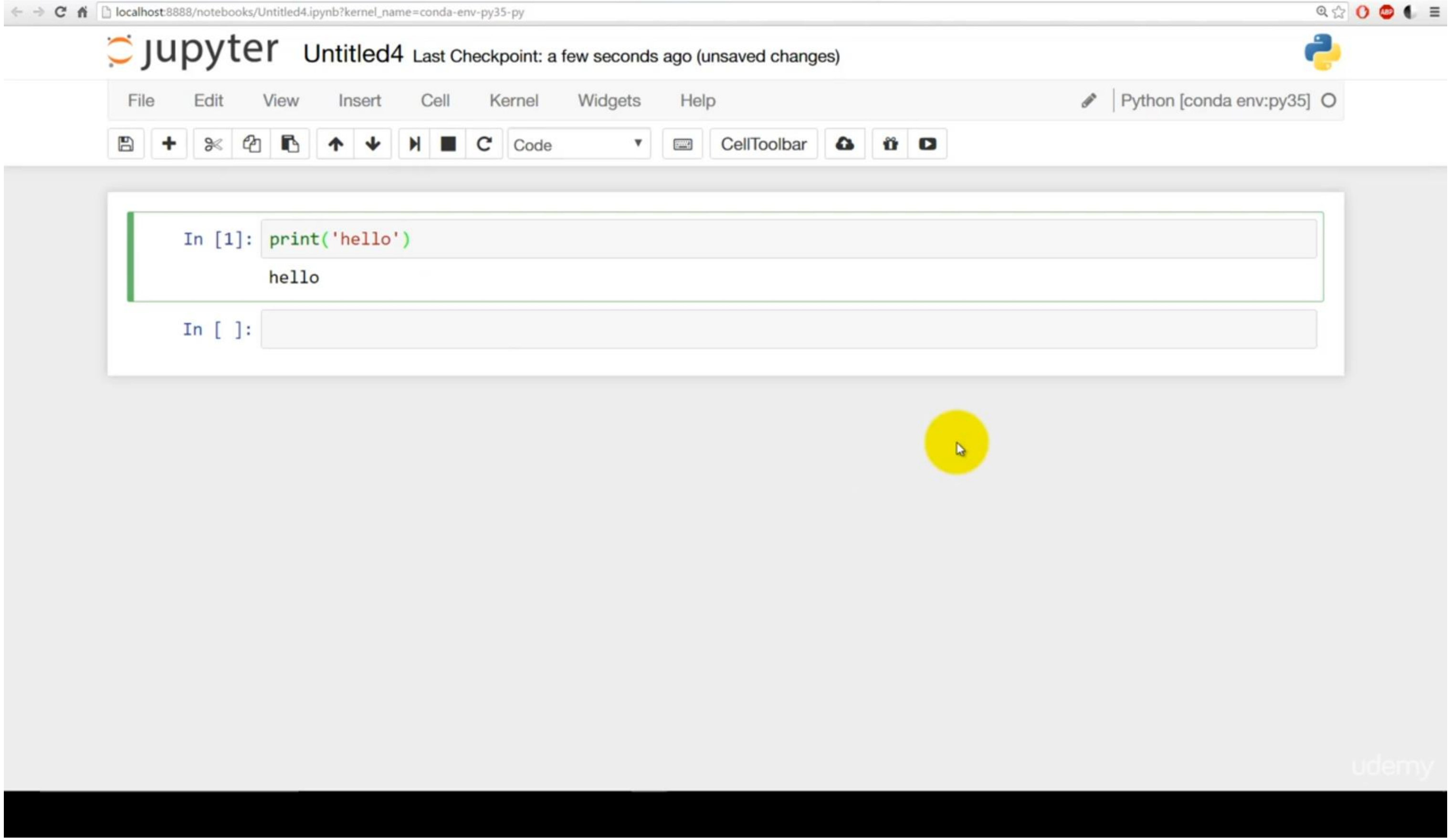
Select items to perform actions on them.

Upload | New ▾

- ☐ communities
- ☐ datasets
- ☐ featured
- ☐ Welcome Julia - Intro to Gadfly.ipynb
- ☐ Welcome R - demo.ipynb
- ☐ Welcome to Haskell.ipynb
- ☐ Welcome to Python.ipynb
- ☐ Welcome to Spark with Python.ipynb
- ☐ Welcome to Spark with Scala.ipynb

Create a new notebook with Python 3

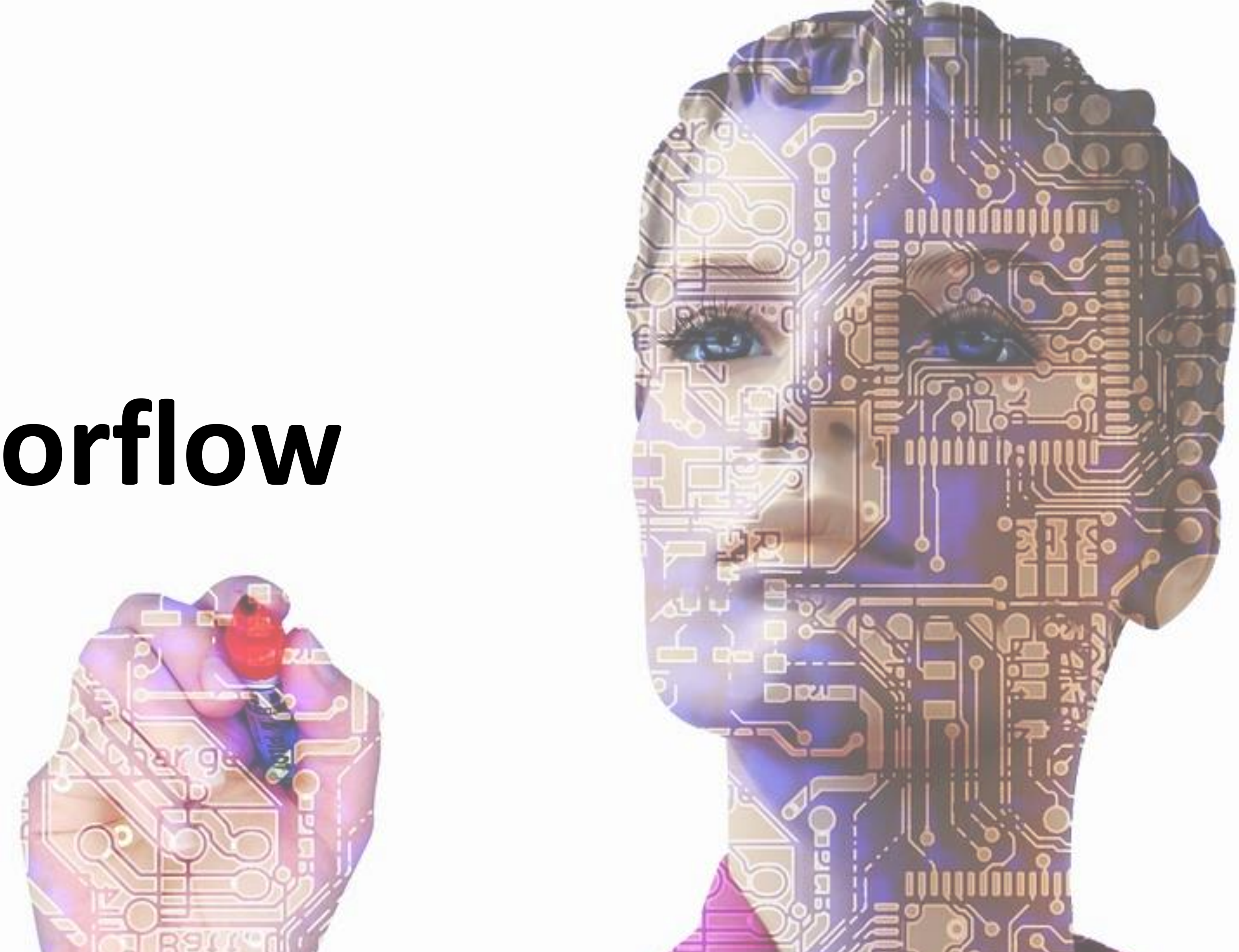
https://tmp54.tmpnb.org/user/Zn6gTh0Hx7CK/tree#


localhost:8888/notebooks/Untitled4.ipynb?kernel_name=conda-env-py35-py
jupyter Untitled4 Last Checkpoint: a few seconds ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Python [conda env:py35]
Code
CellToolbar
udemy

```
In [1]: print('hello')
        hello

In [ ]:
```

Tensorflow



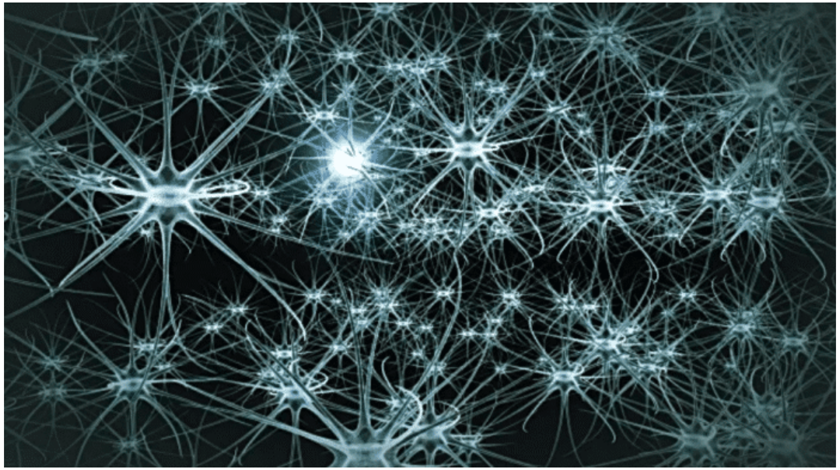
- TensorFlow is an open source software library developed by Google
- It’s not specifically for neural networks – it’s more generally an architecture for executing a graph of numerical operations / data flow graphs
- These graphs have nodes and edges, just as we saw in any typical neural network
- The arrays (data) passed from layer to layer is known as tensor

- Tensorflow can optimize the processing of that graph, and distribute its processing across a network
- It can also distribute work across GPU’s - can handle massive scale
- Runs on about anything - Windows, Linux/macOS, Raspberry Pi, Android

- Documentation: https://www.tensorflow.org/learn

Creating a neural network

1. Load up the training and testing data
2. Construct a graph describing the neural network
3. Associate the optimizer (i.e. gradient descent) to the network
4. Fit your model to your training data
5. Evaluate your trained network with your testing data



Notes:

- Make sure your dataset is well distributed between training/valididation/testing sub-samples
  - Analyze your features and targets before preparing a model
  - Identify outliers or missing data in order and mechanisms to mitigate these problems
- Make sure your features are normalized
  - Neural networks usually work best if your input data is normalized
    - That is, values within “[0-1]” OR “0 mean and unit variance”
    - Every input feature is comparable in terms of magnitude
  - E.g., scikit_learn’s StandardScaler can do this for you
- For classification purposes
  - Make sure targets are on one-hot encoding format
  - E.g., onehot_encoding([cat, dog, cat], n_classes=2) = [[1,0], [0,1], [1,0]]

Universidade do Minho
Escola de Engenharia
Departamento de Informática

Mestrado Integrado em Engenharia Informática
Mestrado em Engenharia Informática
Computação Natural
2020/2021

Filipe Gonçalves, Paulo Novais, Cesar Analide