Machine learning with Python: Create, populate & use a Python Virtual Environment (PVE)

DuMAS Department Day

September 22, 2023

Jean-Luc.Charles@mailo.com

Why use PVE to develop ML programs with Python

PVE Advantages

- To create a dedicated environment (disk tree) with fixed version of the Python interpreter and sensitive modules (like tensorflow)
- Easy to create and destroy as many times as you want
- To Protect your projects against operating system updates or hazardous manipulations...
  ⇝ you can load/update modules within a PVE without breaking modules compatibility for the other projects
- Each Python project should have its own PVE...

Disadvantages

- ? (just do it)

Different ways to Create/Populate a PVE

The naive way: install the modules *by hand* in a **conda** PVE

- Create a PVE with conda and load the desired modules *by hand* with `conda install module ...` or `pip install module ...`
- Mixing `conda install` and `pip install` *by hand* within a `conda` PVE can lead to operating difficulties
- Difficult to maintain, dupplicate, re-create, share...

The naive way: install the modules *by hand* in a **venv** PVE

- Create a PVE with venv and load the desired modules *by hand* with `pip install module ...`
- Most practiced by beginners, but can quickly become a mess
- Difficult to maintain, dupplicate, re-create, share...

Different ways to Create/Populate a PVE

The standard way: the **venv** module & **pip** command

- Create the PVE with the venv Python module
- Install the Python modules with a TXT file (list of the desired modules) and the command `pip install -r myfile.txt`
-  The most used on internet tutos
-  Difficult to create a PVE with a version of Python different from the version of Python installed with your OS

Different ways to Create/Populate a PVE

The preferred way: the **conda** command

- Create the PVE with the conda command (available thanks to `Miniconda` distribution)
- Install the modules with a YAML file (list of the desired modules) and the command `conda update ... --file myfile.yml`
- The installation of `Miniconda` can be tricky on some computers
- You can create a PVE with the version of Python you need [3.6, 3.7, 3.8 ...]
- `conda` installs transparently some modules (numpy, tensorflow...) linked with the MKL library (gives high computing performances for Intel CPUs)

Programming *Machine Learning* (ML) in Python3

- `conda` or `venv` lets you install **dedicated** PVEs on your laptop GNU/Linux, macOS or Windows for every ML project.
- Very often used IDE[a] for ML in Python:
  - **jupyter notebook**: for creating Python *notebooks* $\rightsquigarrow$ files `*.ipynb` for ML, data processing, reports... Used in most tutorials on the internet.
  - **VSCode**, *a.k.a Visual Studio code* from Microsoft: multi-language, very powerful, requires some work (time) to get started, especially to make it work with PVE...
  - **idlex**: the smallest & simplest IDE for creating/running `*.py` files (a "Python interpreter" window and a "program editor" window)
  - **pycharme**, **pyzo, spyder** and many others here …

[a] Integrated Development Environment

Examples of YAML/TXT files to install Python modules for ML:

YAML format for `conda`

```yaml
name: dumas1
channels:
  - defaults
dependencies:
  - python=3.8
  - tensorflow==2.9.*
  - pandas
  - matplotlib
  - opencv
  - jupyter
  - notebook
  - scikit-learn
  - seaborn
  - pip
```

TXT format for `pip`

```
tensorflow==2.9.*
pandas
matplotlib
opencv-python
jupyter
notebook
scikit-learn
seaborn
```

Installation of **Miniconda3** (if not already installed)

- Download the latest version of `Miniconda3` for your OS at docs.conda.io/en/latest/miniconda.html.
- Start the installation of `Miniconda3`...
  [Linux] In a terminal type the command:
  `bash ...some_where.../miniconda3-latest-Linux-x86_64.sh`]

Warning : the path of the installation folder `Miniconda3` must contain no **space** nor **accented character**!

Windows :
`C:\Miconda3` ou `C:\Users\Marie\miniconda3` ⇝ OK
`C:\Yoann\Mes install\miniconda3` ⇝ not OK (space)
`C:\Users\Léon\miniconda3` ⇝ not OK (accentuated e)
MacOSX & GNU/Linux :
`/home/moi/miconda3` ou `/Users/moi/opt/miniconda3` ⇝ OK
`/Users/Léon/miniconda3` ⇝ not OK (accentuated e)

Already have **Anaconda3** or **Miniconda3** installed on your laptop...

In the terminal [macOS, Linux], or the "Anaconda prompt" window [Windows], update `conda` with the command:

```
conda update -n base -c defaults conda
```

Miniconda3 post-Installation

In the terminal [macOS, Linux], or the "Anaconda prompt" window [Windows]

- if you want to disable the automatic activation of the **base** default PVE, type:

```
conda config --set auto_activate_base false
```

Info on the **conda** installation

In the terminal [macOS, Linux], or the "Anaconda prompt" window [Windows]

- to get information on the `Miniconda3` installation:

```
conda info
```

Create/Populate a PVE with **conda**

Create a conda PVE

In a **NEW terminal** [macOS, Linux] or an "Anaconda prompt" window [Windows], create the `dumas1` PVE:

```
conda create -n dumas1 python=3.8 -y
```

Activate a conda PVE

Once the `dumas1` PVE is created, you must **activate** it to use it:

- In the terminal [macOS, Linux], or the "Anaconda prompt" window [Windows], type:

  ```
  conda activate dumas1
  ```

- the *prompt* is now prefixed with `(dumas1)`, for example:

  ```
  Windows:      (dumas1) C:\Users\me>
  macOs:        (dumas1) /Users/me $
  GNU Linux:    (dumas1) user@home $
  ```

Create/Populate a PVE with **conda**

Some useful `conda` commands you can type in the terminal [macOS, Linux], or the "Anaconda prompt" window [Windows]:

List all the conda PVEs

```
conda env list
```

Deactivate a conda PVE

If you need to deactivate any activated PVE:

```
conda deactivate
```

$\rightsquigarrow$ and the prompt is no more prefixed by any PVE name.

Remove a conda PVE

If you want to remove the `dumas1` PVE, type:

```
conda deactivate # in case the dumas1 PVE is activated
conda env remove -n dumas1
```

Create/Populate a PVE with **conda**

Populate a conda PVE

- The `--file` option of the `conda env update` command takes the name of an ASCII file in YAML format containing the list of Python modules to install.
- It is imperative to name the selected PVE with the option:
  `-n <PVE_name>`

Install the modules for the `dumas1` PVE with `dumas1.yml`

In the terminal [macOS, Linux], or the "Anaconda prompt" window [Windows]

- with the `cd` command go into the folder holding the YAML file:
  `cd <path_of_the_folder_containing_the_YAML_file>`
- then install the Python modules in the `dumas1` PVE:
  `conda env update -n dumas1 --file dumas1.yml`

Create/Populate a PVE with **venv**

If the creation of the PVE with `conda` fails you can switch to `venv`:

Create a venv PVE

In a terminal [macOS, Linux] or a "CMD" window [Windows]

To create a PVE, type: `python -m venv <PVE_dir>`

- `<PVE_dir>` is the name of the directory (or directory path) that will be created for the PVE (the shortest, the easiest to use...)
- `<PVE_dir>` can be as simple as `dumas1` or a full path like:
  [Windows] `C:\Users\me\dir1\...\dumas1`
  [macOS, Linux] `/home/users/.../dumas1`

Create the `dumas1` PVE

In a terminal [macOS, Linux] or a "CMD" window [Windows]:

```
python -m venv dumas1
```

Create/Populate a PVE with **venv**

Activate a venv PVE

Once the `dumas1` PVE is created, you must **activate** it to use it:

- [Windows] `<PVE_dir>\Scripts\activate.bat`
- [macOS, Linux] `source <PVE_dir>/Scripts/activate`
- the *prompt* is prefixed with the activated PVE:

```
Windows:      (dumas1) C:\Users\me>
macOs:        (dumas1) /Users/me $
GNU Linux:    (dumas1) user@home $
```

Activate the `dumas1` PVE

In the "CMD" window [Windows]:

```
dumas1\Scripts\activate.bat
```

In the terminal [macOS, Linux]:

```
source dumas1/Scripts/activate
```

Check the *prompt*: it should be be prefixed by `dumas1`...

Create/Populate a PVE with **conda**

Populate a venv PVE

- The `-r` option of the `pip install` command takes the name of a TXT file containing the list of the Python modules to install.
- beforehand, the **target PVE must have been activated**

Install the modules in the `dumas1` PVE with `dumas1.txt`

In a terminal [macOS, Linux] or a "CMD" window [Windows] with the `dumas1` PVE activated:

```
pip install -r path_of_dumas1.txt -y
```

Launch **jupyter notebook**

In a terminal [macOS, Linux] or a "CMD/Anaconda prompt" window [Windows], with the `dumas1` PVE **activated**, type:

```
jupyter notebook
```

Access notebooks anywhere with **jupyter**

- In a "CMD/Anaconda prompt" window [Windows], access notebooks on a logical unit other than `C:\` (partition disk `D:\`, USB key `E:\`):

```
jupyter notebook D:\
jupyter notebook D:\folder1\folder2
```

- In a terminal [macOS, Linux], access the notebooks in a given folder:

```
jupyter notebook /home/users/me/folder1/folder2
```

Quit **jupyter notebook**

In a terminal [macOS, Linux] or a "CMD" window [Windows], after quitting the jupyter window, type CTRL-C twice to interrupt the jupyter server.