lrz

Leibniz Supercomputing Centre

of the Bavarian Academy of Sciences and Humanities

Hands-on on classical Machine Learning

AI in Astronomy – ESO Garching | 22.7.2019 | Luigi Iapichino

Structure of this hands-on

- Required **software** and how to install it
- Login on the **training system**
- ML algorithm I: **K-means**
- ML algorithm II: **Principal Component Analysis**
- Q&A

Why this hands-on

- We won’t provide a thorough introduction on ML or its algorithms (although we present some basic theory)
- We want to provide information and working examples on how to accelerate them by using modern libraries
- Surely you can devise good test cases on astronomical problems and data sets
- If you think you got a nice project idea and you need support to develop it and a large HPC system to work on, talk with us!

https://doku.lrz.de/display/PUBLIC/Astro-Lab

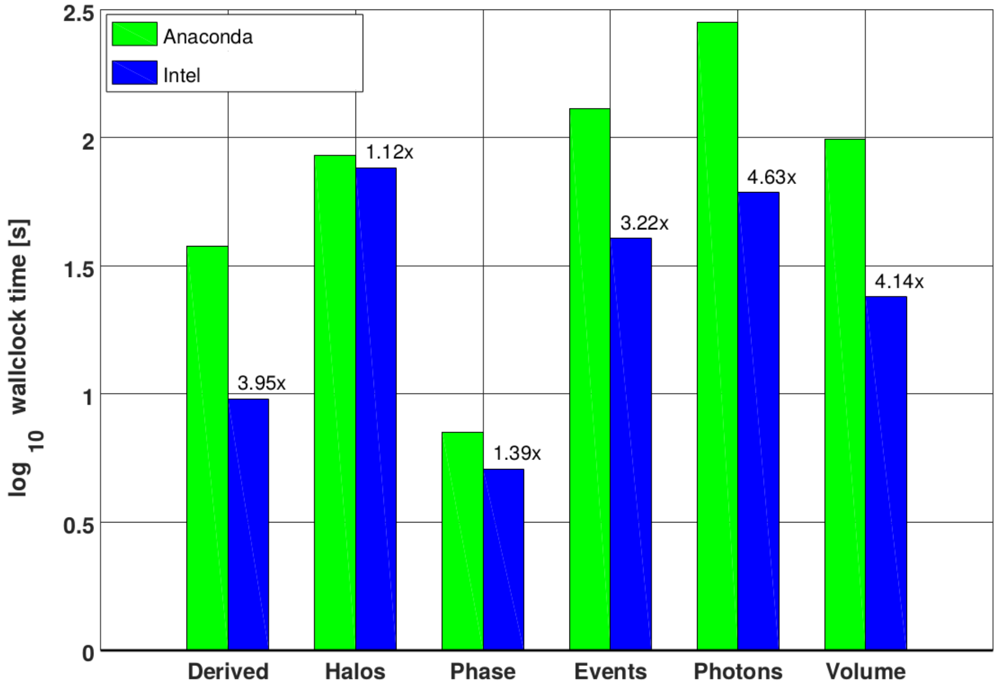
Setting the stage: **Intel® Distribution for Python**

- Out-of-the-box Python distribution, highly optimized for high performance
- Seamless substitution of the system distribution
- It comes with optimized libraries (NumPy, SciPy, Scikit-learn, mpi4py…)
- On LRZ systems: available as *module*

```
<user>@login08:~> module av python
--/lrz/sys/share/modules/files/tools ---
python/2.7_anaconda_nompi  python/2.7_intel(default)  python/3.5_intel
python/3.6_intel
```

Current on-going research project: <u>Python performance</u>

- Application: **yt** (Python toolkit for data analysis and visualization of simulation datasets)
- Performance comparison of system (Anaconda) Python vs. Intel Python on different analysis types
- Intel Python's strength: exploiting parallelism



Cielo, Baruffa & Iapichino 2019, in preparation

Accessing the training system

Workshop Material

Go to the following url to download slides and samples:

https://tinyurl.com/eso-ai-workshop

For installing on your own system:

- You need to have Conda on your system:
  https://docs.conda.io/projects/conda/en/latest/user-guide/getting-started.html

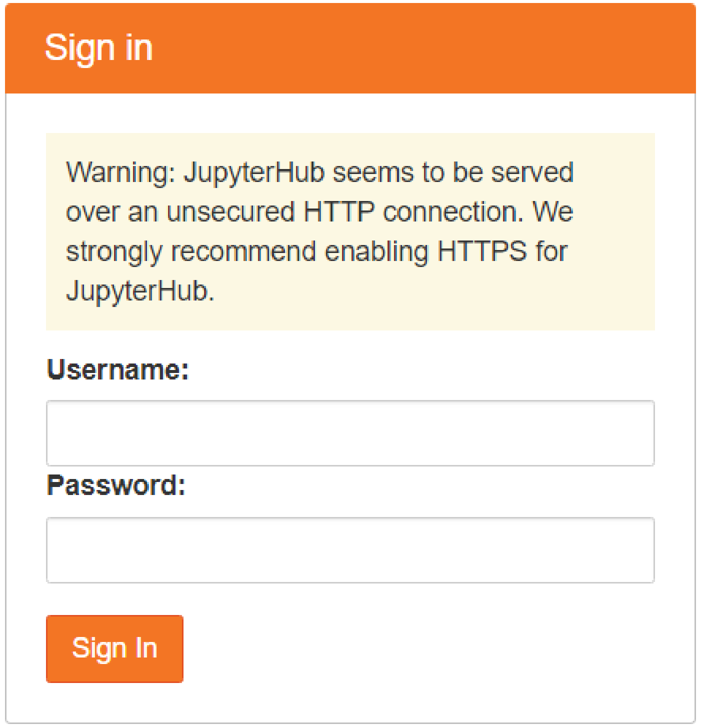
- Install Intel Optimized Software:

```
> conda create -n intel-py -c intel install intelpython3_full==2019.4
```

Access the Jupyter Server provided by LRZ Cloud

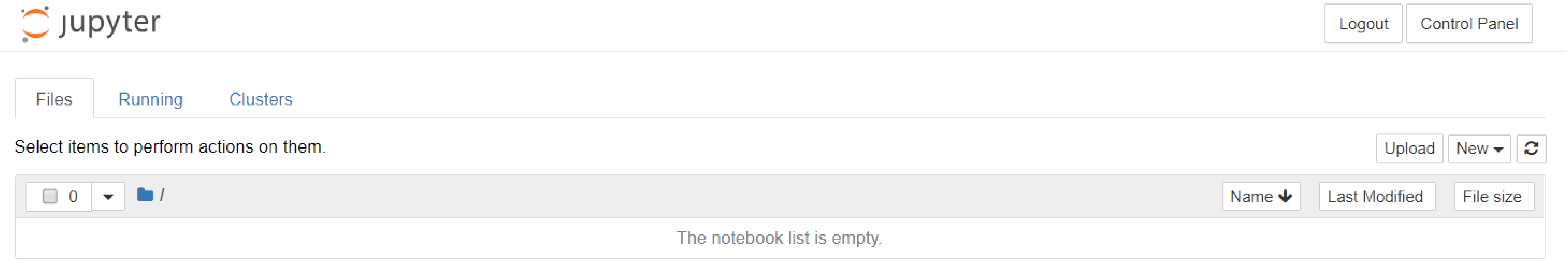
- Open your browser and go to the following url:

  http://138.246.233.51

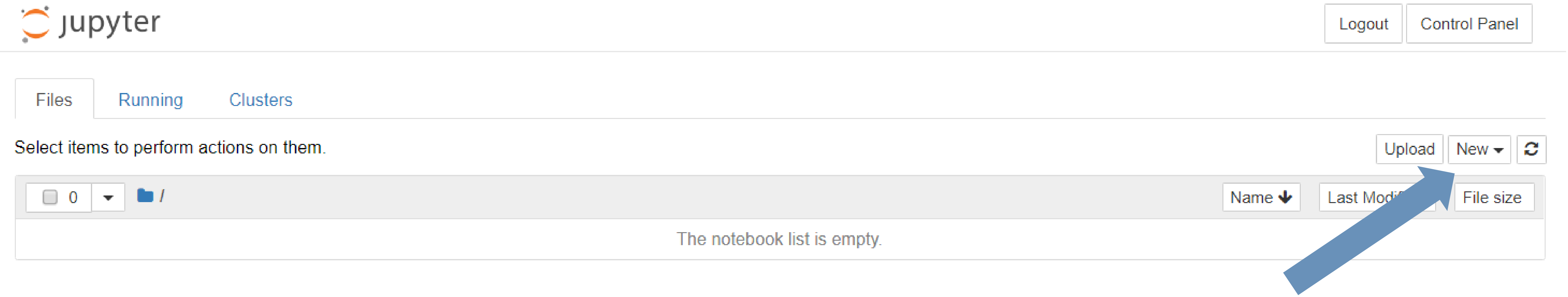


- Choose a Username and Password for your access
- Then Sign in

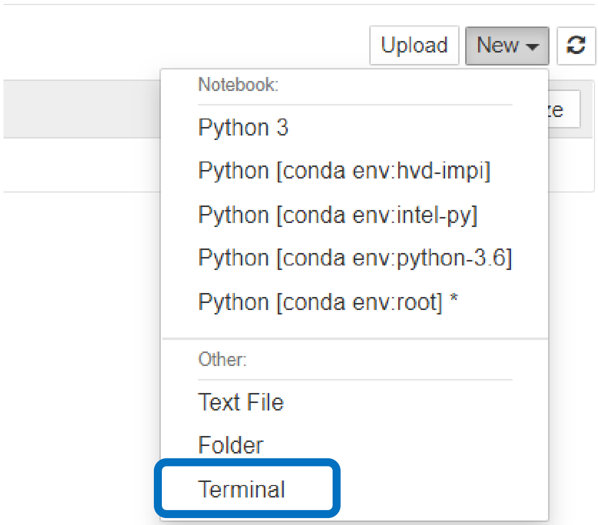
Access the Jupyter Server provided by LRZ Cloud



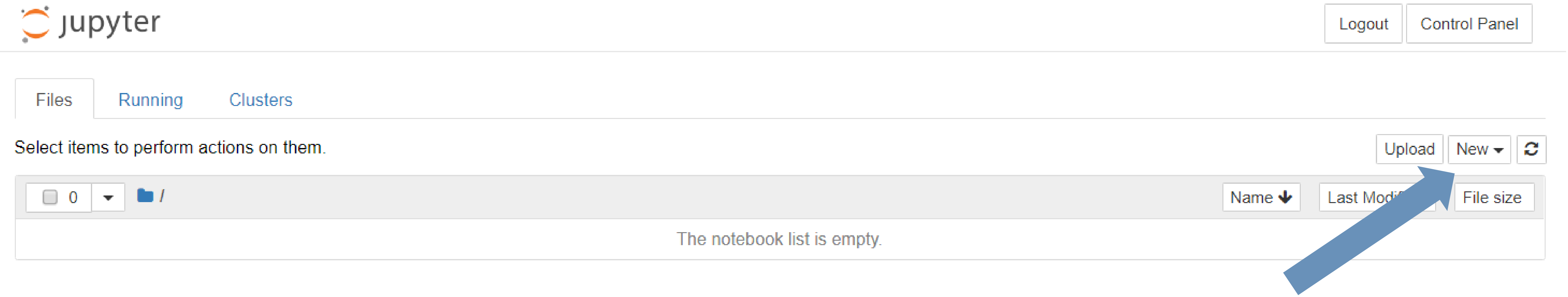
Access the Jupyter Server provided by LRZ Cloud



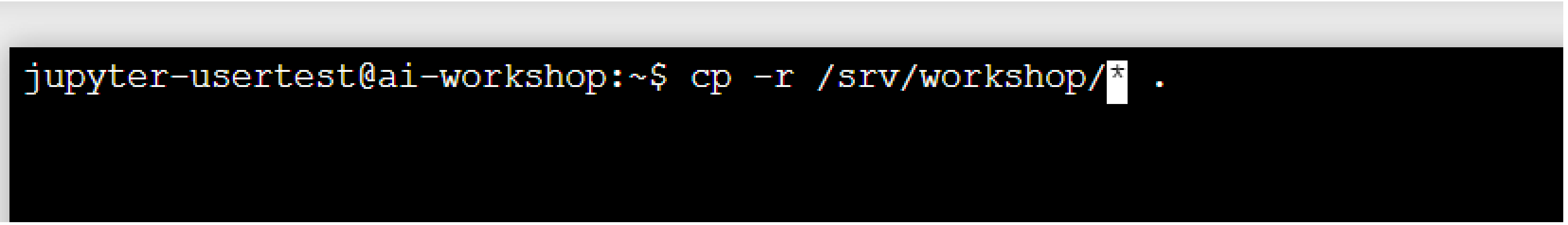
- Go to the tab New for creating a Terminal window

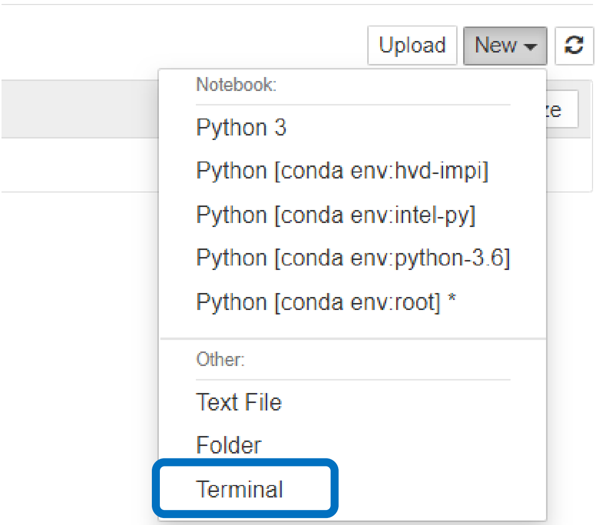


Access the Jupyter Server provided by LRZ Cloud



- Go to the tab New for creating a Terminal window
- Type the following in the Terminal Window to copy the workshop samples: *cp -r /srv/workshop/* .*



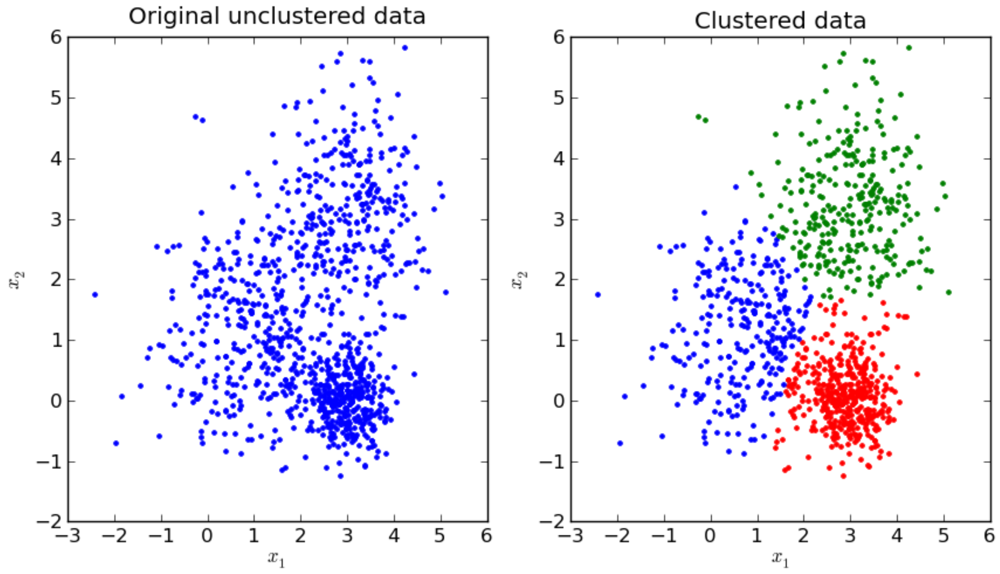


Access the Jupyter Server provided by LRZ Cloud

ML algorithm I: K-Means

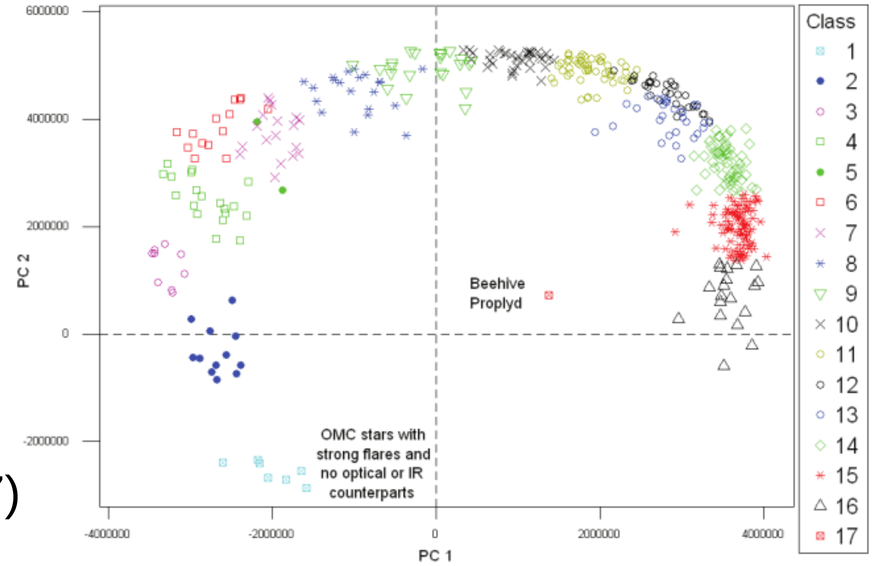
K-means

- K-means is a clustering method
- It is used to group objects in a (high-dimensional) sample
- K-means is based on centroids



Examples of astronomical applications:

- X-ray spectral classification of joung stellar clusters (Hojnacki et al. 2007)
- Asteroids spectra (Galluccio et al. 2008)
- SDSS galaxy spectra classification (Sanchez-Almeida et al. 2010)
- Stellar spectra (Simpson et al. 2012)
- …



(from Hojnacki et al. 2007)

Algorithm description

- *k* (number of clusters in the dataset) is an external free parameter
- *k* random objects are associated with initial centroids
- each object of the dataset is assigned to form a cluster with its closest centroid
- new centroids are computed by taking the average position of the objects in a given cluster
- Evaluate convergence condition
- Output: centroids location and association of the objects

Pros and cons:

\+ Simple and relatively robust

\- Finding the best *k* is not trivial

\- Results might depend on initial centroids

\- Sensitive to outliers and to features with different dynamical range → data preparation

Hands-on: Color quantization using K-means

- An interesting application of K-means is to reduce the number of colors in a figure, while preserving the overall quality
- Initial data: every pixel has three color channels (RGB) expressed with 8 bit (0…255)
- For comparison, a clustering based on randomly picked colors will be shown.

Original image (96,615 colors)

The image of the Summer Palace (China) is among the sample data contained on scikit-learn

Description of the hands-on

The hands-on is based on six main steps:

- Data preparation
- Computing the cluster centers
- Labelling the data
- From scikit-learn to daal4py: command line
- From scikit-learn to daal4py: monkey-patch in the script
- K-means with daal4py

- The files for the hands-on are in the directory `/day1-ML/k-means`
- There are `ver…` folders with a Python script, and its solution in a subfolder
- You can also use the Notebook: `kmeans-hanson.ipynb`
- Most of the exercise scripts are not working, because lines are missing!
- The solutions however do work, but I added a lot of blank spaces to avoid spoilers for the next steps ☺

Useful resources

The hands-on Python script comes mostly from the scikit-learn example on color quantization using K-means:
`https://scikit-learn.org/stable/auto_examples/cluster/plot_color_quantization.html#sphx-glr-auto-examples-cluster-plot-color-quantization-py`

K-means documentation on scikit-learn:
`https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html`

K-means documentation on daal4py:
`https://intelpython.github.io/daal4py/algorithms.html#k-means-clustering`

Data preparation

Go to `day1-ML/k-means/ver0-data-preparation` and open `plot_color_quantization.py`

First hands-on: exploring the initial data and their shape

Data preparation: changing to a representation based on floats in [0; 1]

Moreover: need to reshape the data to a 2D array (**second hands-on**)

Reference here: `https://docs.scipy.org/doc/numpy/reference/generated/numpy.reshape.html`

Please go to `ver0-data-preparation/solution` to check your work and run the script

Computing the cluster centers

Go to `day1-ML/k-means/ver1-cluster-centers` and open `plot_color_quantization.py`

Three main tasks here:

- Defining a sub-sample of the data: this is unsupervised ML without a real training stage, however it can be useful to perform some operations on a data subsample

- Preparing timings: maybe a bit off-topic, but it always comes handy

- Computing the cluster centers: please use the documentation of `KMeans` in scikit-learn for it. It is also a good idea to print the computed centers.

Please go to `ver1-cluster-centers/solution` to check your work and run the script

Labelling the data to create the color clusters

Go to `day1-ML/k-means/ver2-labeling` and open `plot_color_quantization.py`

First hands-on:

- Using the appropriate method of `KMeans` to assign the data to the centroids computed in the previous version

Second hands-on:

- For a quality comparison, let's identify random centroids and assign the data to them
- Reference: `https://scikit-learn.org/stable/modules/generated/sklearn.utils.shuffle.html`

Please go to `ver2-labeling/solution` to check your work and run the script

Comparison of picture quality

Original image (96,615 colors)

Quantized image (64 colors, K-Means)

Quantized image (64 colors, Random)

You can make experiments modifying the number of clusters and sampling…

From scikit-learn to daal4py

First method (`day1-ML/k-means/ver3-run-daal4py`): without changing at all the script

- A flag to monkey-patch is needed when running to script via `python`
- Can you find it in the documentation (or remember from the previous slides)?
- Reported in `solution`

Second method (`day1-ML/k-means/ver4-daal4py-patch`): minimal changes to monkey-patch the script

- Again: it was introduced on passing, but you can check in the documentation
- Please go to `ver4-daal4py-patch/solution` to check your work and run the script
- **In both methods**: what about the performance of your script?

K-means with daal4py

Go to `day1-ML/k-means/ver5-daal4py`
and open `kmeans-daal4py.py`

- The example is taken from the daal4py documentation, slightly adapted to resemble the previous one.
- No hands-on here, but you can read the code and the documentation

Which execution model is more convenient?

- Running scikit-learn uses TBB under the hood
- Patching it with daal4py keeps the code more similar in terms of algorithms, and boosts the performance
- Using the daal4py classes explicitly allows distributing multiple nodes with Intel MPI.

E.g.: `mpirun -prepend-rank -genv I_MPI_DEBUG=5 -n 2 python -u kmeans-daal4py.py`

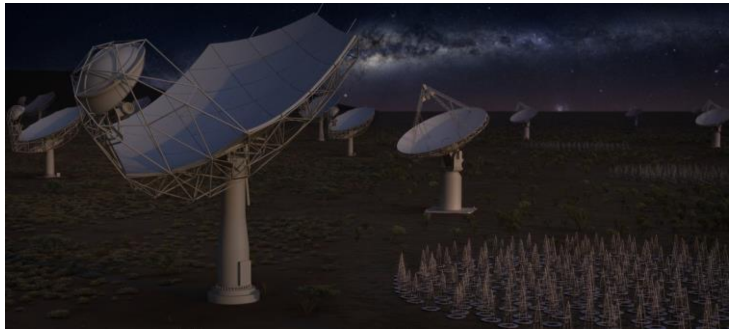
ML algorithm II: PCA

Dimensionality reduction algorithms

High-dimensional datasets are more and more common in data science

Their analysis often involves a **dimensionality reduction**:

- selecting a subset of features which best describes the dataset
- constructing a new set of features which provides a good description



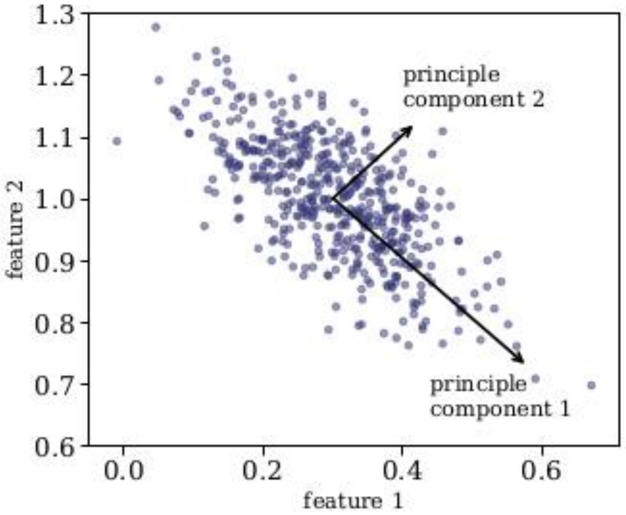
Why to reduce data dimensionality?

- Removing noise
- Making the results easier to understand
- Making the dataset easier to be used (**data handling and movement**)
- Reducing computational cost of algorithms (**data processing**)

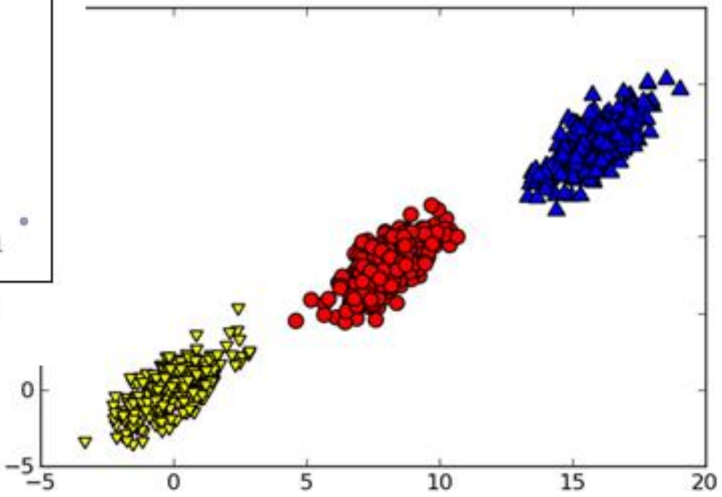
These points are crucial for upcoming instruments like SKA, where retaining the full complexity of the raw data will be infeasible.

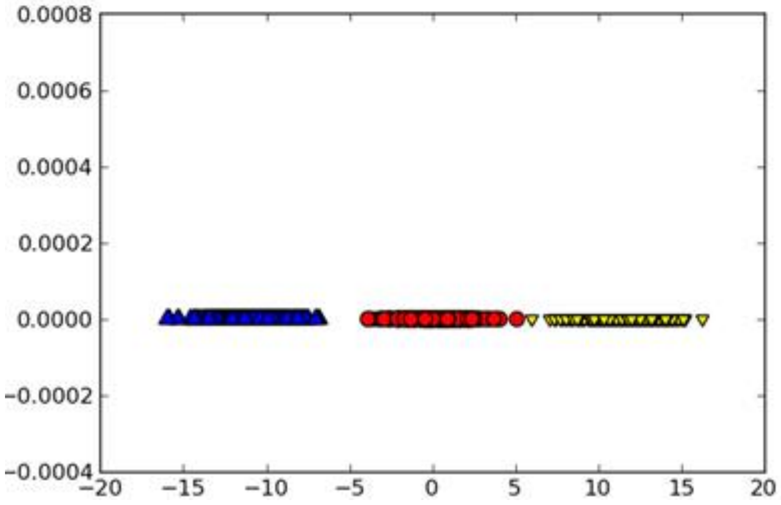
Principal component analysis (PCA)

- PCA: popular method for dim. reduction
- In essence, it is a **coordinate transformation** in the dataset
- In the new coordinate system, the first coordinate (component) is chosen so to have the largest variance in the data.
- The second component is orthogonal and has the second largest variance.
- Number of components = number of features of the data.
- Lower number of components → dimensionality reduction.



Baron 2019: PCA as new coordinate system





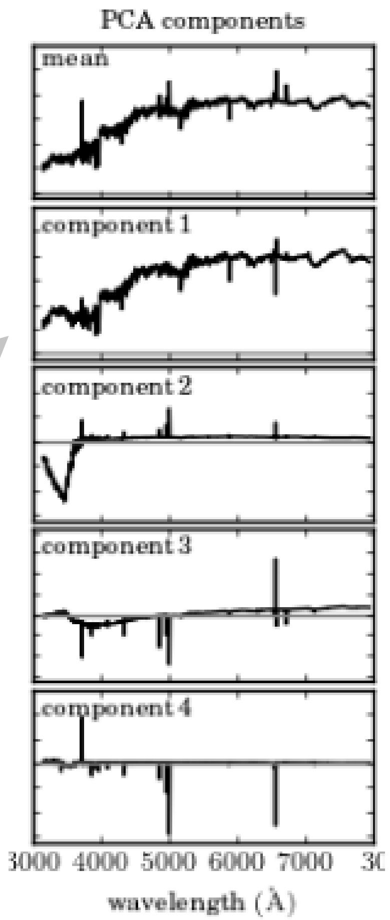
Harrington 2012: dimensionality reduction

PCA for a formal viewpoint

- Consider the $n \times m$ matrix of n data points with m features each
- Compute the **covariance matrix** C of it
- Diagonalise C
- Sort its (positive) eigenvalues by size
- Identify in this way the eigendirections where the data have maximal variance → the principal components

Examples of astronomical applications:

- Decomposition of quasar spectra (Boroson & Green 1992)
- Physical parameters of stellar atmospheres (Zhang et al. 2006)
- Properties of SDSS galaxy spectra (Vandeplas et al. 2012)
- Optimal base for cosmological observables (Maturi & Mignone 2009)
- …

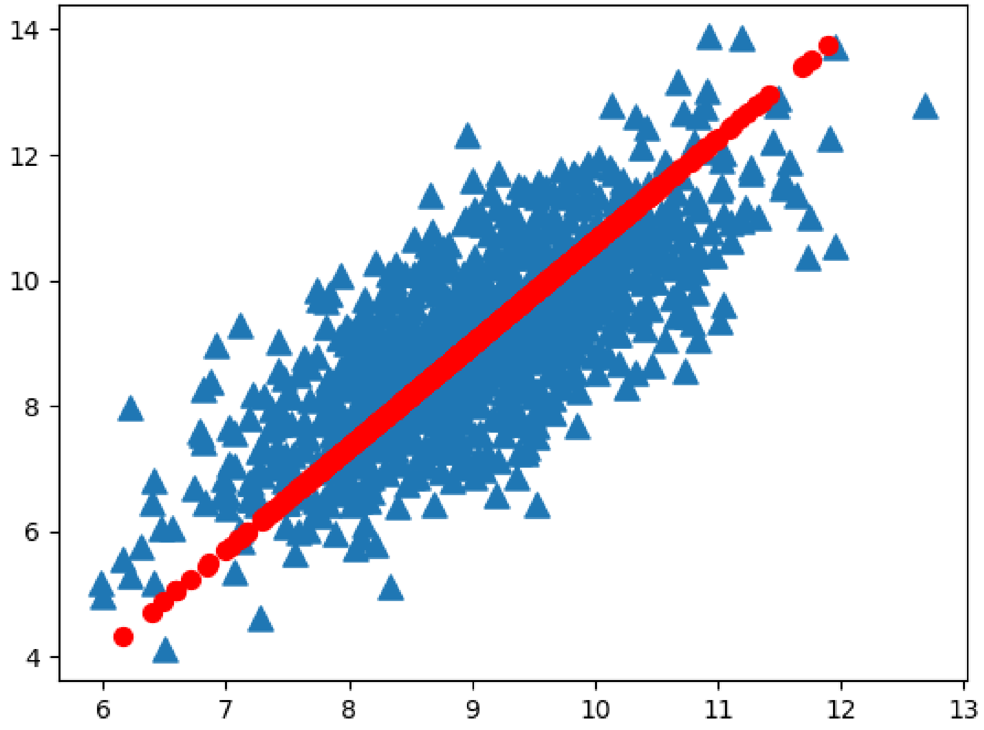


Description of the hands-on

Very simple test:

- Reading data from a file (1000 x 2)
- Applying PCA to find the principal component
- Applying a transformation to the "collapsed" data point, to overplot them with the original data

This procedure will be applied in three different versions of the script, using **Numpy**, **scikit-learn** and **daal4py**



Plot from the original Numpy test (Harrington et al. 2012)

Useful resources

The hands-on Python script (Numpy version) comes originally from the book *Machine Learning in Action* (Harrington 2012), Manning Publications
`https://www.manning.com/books/machine-learning-in-action`

PCA documentation on scikit-learn:
`https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html`

PCA documentation on daal4py:
`https://intelpython.github.io/daal4py/algorithms.html#principal-component-analysis-pca`

Concluding remarks

HPC is quickly changing its workloads from traditional compute to data-intensive applications

Python has become a full-fledged language for HPC

Performance at scale depends on the same features for all problems:

- Concurrency and parallelism at all levels
- Access to data in memory

Using optimized libraries addresses the above points and is crucial in ML on all system scales, up to HPC

The HPC environment is striving for fully supporting ML and data analytics applications.
Is your code ready?

References and other resources

https://software.intel.com/en-us/distribution-for-python

https://scikit-learn.org/stable/index.html

https://intelpython.github.io/daal4py/contents.html

Schäfer & Bartelmann 2017, *Statistics: the logic of science*, online document

Harrington 2012, *Machine Learning in Action*, Manning Publications

Baron 2019, *Machine Learning in Astronomy: a practical overview.* ArXiv: 1904.07248