Introduction to Machine Learning

Useful tools: Python, NumPy, scikit-learn

Antonio Sutera and Jean-Michel Begon

September 29, 2016


python™

How to install Python?

Download and use the Anaconda python distribution `https://store.continuum.io/cshop/anaconda/`. It comes with all the scientific python stack.

Alternatives: linux packages, pythonxy, canopy, ...

Using the python interpreter

Interactive mode

1. Start a python shell

   ```
   $ ipython
   ```

2. Write python code

   ```
   >>> print("Hello World!")
   Hello World!
   ```

Script mode

1. hello.py

   ```
   print("Hello World!")
   ```

2. Launch the script

   ```
   $ ipython hello.py
   Hello world!
   ```

Basic types

Integer

```
>>> 5
5
>>> a = 5
>>> a
5
```

Float

```
>>> pi = 3.14
```

complex

```
>>> c = 1 - 1j
```

boolean

```
>>> b = 5 > 3  # 5 <= 3
>>> b
True  # False
```

string

```
>>> s = 'hello!'  # Also works with "hello!"
>>> s
'hello !'
```

Python is a dynamic program language

Variable types are implicitly inferred during the assignment.
Variables are not declared.

```
>>> # In python
>>> a = 1
```

By contrast in statically typed language, you must declared the type.

```
// In java, c, c++
int a = 1
```

Numbers and their arithmetic operations (+,-,/,//,*,**,%)

```
>>> 1 + 2
3
>>> 50 - 5 * 6
20
>>> 2 / 3  # with py3 0.66...
0
>>> 2. / 3  # float division in py2 and py3
0.6666666666666666
>>> 4 // 3  # Integer division with py2 and py3
1
>>> 5 ** 3.5  # exponent
279.5084971874737
>>> 4 % 2  # modulo operation
0
```

Playing with strings

```
>>> s = 'Great day!'
>>> s
'Great day!'
>>> s[0]  # strings are sequences
'G'
>>> """A very

very long string
"""
'A very\nvery long string\n'
>>> 'i={0} f={2} s={1}'.format(1, 'test', 3.14)
'i=1 f=3.14 s=test'
```

list, an ordered collection of objects

Instantiation

```
>>> l = []  # an empty list
>>> l = ['spam', 'egg', ['another list'], 42]
```

Indexing

```
>>> l[1]
'egg'
>>> l[-1]  # n_elements - 1
42
>>> l[1:2]  # a slice
["egg", ['another list']]
```

Methods

```
>>> len(l)
4
>>> l.pop(0)
'spam'
>>> l.append(3)
>>> l
['egg', ['another list'], 42, 3]
```

dict, an unordered and associative data structure of key-value pairs

Instantiation

```
>>> d = {1: "a", "b": 2, 0: [4, 5, 6]}
>>> d
{0: [4, 5, 6], 1: 'a', 'b': 2}
```

Indexing

```
>>> d['b']
2
>>> 'b' in d
True
```

Insertion

```
>>> d['new'] = 56
>>> d
{0: [4, 5, 6], 1: 'a', 'b': 2, 'new': 56}
```

Deletion

```
>>> del d['new']
>>> d
{0: [4, 5, 6], 1: 'a', 'b': 2}
```

dict, an unordered and associative data structure of key-value pairs

Methods

```
>>> len(d)
3
>>> d.keys()
[0, 1, 'b']
>>> d.values()
[[4, 5, 6], 'a', 2]
```

Control flow: if / elif / else

```
>>> x = 3
>>> if x == 0:
...     print("zero")
... elif x == 1:
...     print("one")
... else:
...     print("A big number")
...
'A big number'
```

Each indentation level corresponds to a block of code.

Note: Indentation in Python replaces "{}" and "()" of other languages.

Control flow: for loop

```
>>> l = [0, 1, 2, 3]
>>> for a in l:  # Iterate over a sequence
...     print(a ** 2)
0
1
4
```

Iterating over sequence of numbers is easy with the range built-in.

```
>>> range(3)
[0, 1, 2]
>>> range(3, 10, 3)
[3, 6, 9]
```

Control flow: while

```
>>> a = 0
>>> b = 1
>>> while b < 50:   # while True do ...
...     a, b = b, a + b
...     print(a)
...
1
1
2
3
5
8
13
21
34
```

Control flow: functions

```
>>> def f(x, e=2):
...     return x ** e
...
>>> f(3)
9
>>> f(5, 3)
125
>>> f(5, e=3)
125
```

Function arguments are passed by reference in python. Be aware of side effects: mutable default parameters, inplace modifications of the arguments.

Classes and object

```
>>> class Counter:
...     def __init__(self, initial_value=0):
...         self.value = initial_value
...     def inc(self):
...         self.value += 1
...
>>> c = Counter()  # Instantiate a counter object
>>> c.value    # Access to an attribute
0
>>> c.inc()    # Call a method
>>> c.value
1
```

Import a package

```
>>> import math
>>> math.log(3)
 1.0986122886681098
>>> from math import log
>>> log(4)
1.3862943611198906
```

You can try "import this" and "import antigravity".

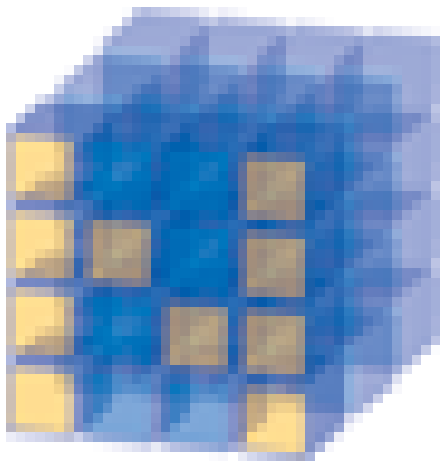
Python reference and tutorial

- Python Tutorial : `http://docs.python.org/tutorial/`
- Python Reference : `https://docs.python.org/library/`

How to use the "?" in ipython?

```
In [0]: d = {"a": 1}

In [1]: d?
Type:        dict
String Form:{'a': 1}
Length:      1
Docstring:
dict() -> new empty dictionary
dict(mapping) -> new dictionary initialized from a mapping object's
    (key, value) pairs
dict(iterable) -> new dictionary initialized as if via:
    d = {}
    for k, v in iterable:
        d[k] = v
dict(**kwargs) -> new dictionary initialized with the name=value pairs
    in the keyword argument list.  For example:  dict(one=1, two=2)
```



NumPy

NumPy

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object,
- sophisticated (broadcasting) functions,
- tools for integrating C/C++ and Fortran code,
- useful linear algebra, Fourier transform, and random number capabilities

With SciPy, it's a replacement for MATLAB(c).

1-D numpy arrays

Let's import the package.

```
>>> import numpy as np
```

Let's create a 1-dimensional array.

```
>>> a = np.array([0, 1, 2, 3])
>>> a
array([0, 1, 2, 3])
>>> a.ndim
1
>>> a.shape
(4,)
```

2-D numpy arrays

Let's import the package.

```
>>> import numpy as np
```

Let's create a 2-dimensional array.

```
>>> b = np.array([[0, 1, 2], [3, 4, 5]])
>>> b
array([[ 0, 1, 2],
       [ 3, 4, 5]])
>>> b.ndim
2
>>> b.shape
(2, 3)
```

Routine to create array: np.ones, np.zeros,...

Array operations

```
>>> a = np.ones(3) / 5.
>>> b = np.array([1, 2, 3])
>>> a + b
array([ 1.2, 2.2, 3.2])
>>> np.dot(a, b)
1.200000
>>> ...
```

Many functions to operate efficiently on arrays : np.max, np.min, np.mean, np.unique, ...

Indexing numpy array

```
>>> a = np.array([[1, 2, 3], [4, 5, 6]])
>>> a[1, 2]
6
>>> a[1]
array([4, 5, 6])
>>> a[:, 2]
array([3, 6])
>>> a[:, 1:3]
array([[2, 3],
       [5, 6]])
>>> b = a > 2
>>> b
array([[False, False, True],
       [ True, True, True]], dtype=bool)
>>> a[b]
array([3, 4, 5, 6])
```

Reference and documentation

- NumPy User Guide:
  `http://docs.scipy.org/doc/numpy/user/`
- NumPy Reference:
  `http://docs.scipy.org/doc/numpy/reference/`
- MATLAB to NumPy:
  `http://wiki.scipy.org/NumPy_for_Matlab_Users`


scikit
learn

scikit-learn Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

A bug or need help?

- Mailing-list: `scikit-learn-general@lists.sourceforge.net`;
- Tag scikit-learn on Stack Overflow.

How to install?

- It's shipped with Anaconda.
- `http://scikit-learn.org/stable/install.html`

Digits classification task

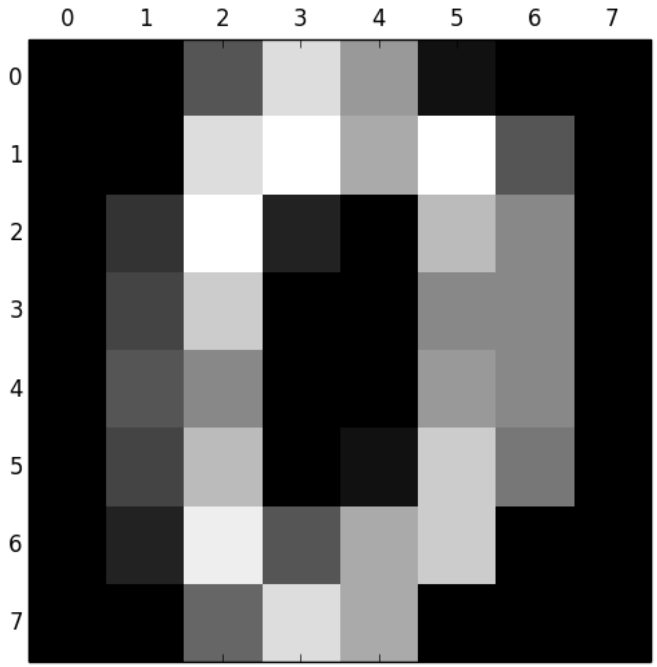
```
# Load some data
from sklearn.datasets import load_digits
digits = load_digits()
X, y = digits.data, digits.target
```

How can we build a system to classify images?
What is the first step?

Data exploration and visualization

```
# Data visualization
import matplotlib.pyplot as plt
plt.gray()
plt.matshow(digits.images[0])
plt.show()
```

What else can be done?



Fit a supervised learning model

```
from sklearn.svm import SVC
clf = SVC() # Instantiate a classifier

# API The base object, implements a fit method
# to learn from data, either:
clf.fit(X, y) # Fit a classifier with the learning samples

# API Exploit the fitted model to make prediction
clf.predict(X)

# API Get a goodness of fit given data (X, y)
clf.score(X, y) # accuracy=1.
```

What do you think about this score of 1.?

Cross validation

```
from sklearn.svm import SVC
from sklearn.cross_validation import KFold
scores = []
for train, test in KFold(len(X), n_folds=5, shuffle=True):
    X_train, y_train = X[train], y[train]
    X_test, y_test = X[test], y[test]
    clf = SVC()
    clf.fit(X_train, y_train)
    scores.append(clf.score(X_test, y_test))

print(np.mean(scores)) # 0.44... !
```

What do you think about this score of 0.44?

Tip: This could be simplified using the cross_val_score function.

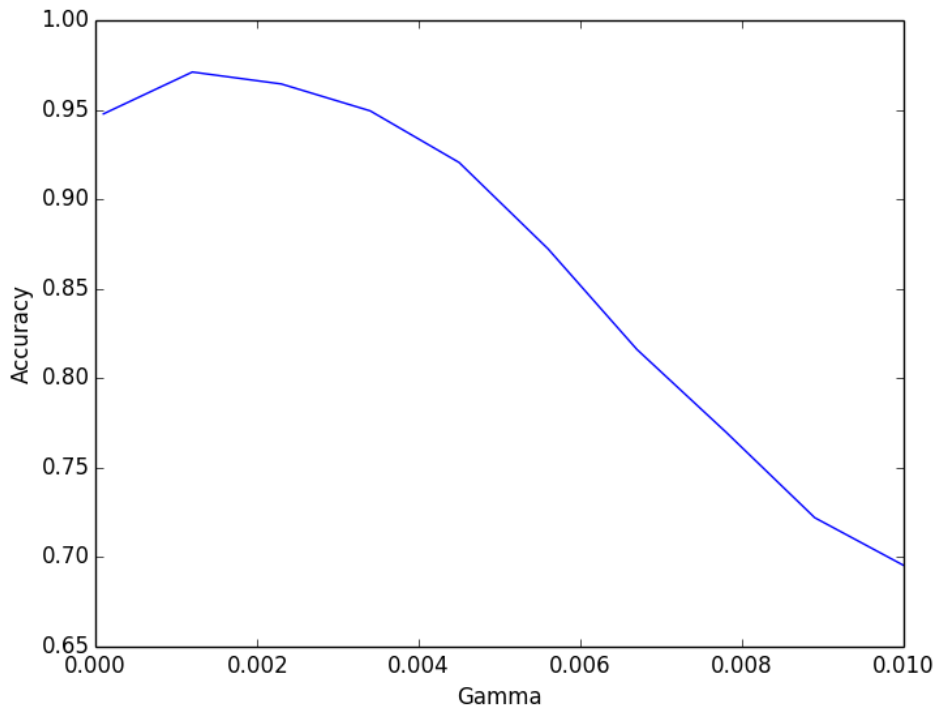
Hyper-parameter optimization

```
from sklearn.svm import SVC
from sklearn.cross_validation import cross_val_score
parameters = np.linspace(0.0001, 0.01, num=10)
scores = []
for value in parameters:
    clf = SVC(gamma=value)
    s = cross_val_score(clf, X, y=y, cv=5)
    scores.append(np.mean(s, axis=0))
print(np.max(scores))  # 0.97... !
```

Tip: This could be simplified using the GridSearchCV meta-estimator.

Visualizing hyper-parameter search

```
import matplotlib.pyplot as plt
plt.figure()
plt.plot(parameters, scores)
plt.xlabel("Gamma")
plt.ylabel("Accuracy")
plt.savefig("images/grid.png")
```



Estimator cooking: transformer union and pipeline

```
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline

# API Transformer has a transform method
clf = make_pipeline(StandardScaler(),
                    # More transformers here
                    SVC())

from sklearn.pipeline import make_union
from sklearn.preprocessing import PolynomialFeatures

union_transformers = make_union(StandardScaler(),
                                # More transformers here
                                PolynomialFeatures())
clf = make_pipeline(union_transformers, SVC())
```

Model persistence

```python
from sklearn.externals import joblib

# Save the model for later
joblib.dump(clf, "model.joblib")

# Load the model
clf = joblib.load("model.joblib")
```

Reference and documentation

- User Guide: `http://scikit-learn.org/stable/user_guide.html`
- Reference: `http://scikit-learn.org/stable/modules/classes.html`
- Examples: `http://scikit-learn.org/stable/auto_examples/index.html`