KSCHOOL

Working with Python

Jupyter, numpy, pandas, matplotlib…

Máster en Data Science

Antonio Almagro

1.1 Introduction to numpy and matplotlib

A complete first class on Numpy and Matplotlib

- **Python scientific stack and Jupyter notebook**
- **Matplotlib**
- **Numpy**
  - indexing.
  - ndarrays and matrices.
  - Linear regression implementation using numpy to minimize cost function.

Python scientific stack

- **NumPy:** Base N-dimensional array package
- **SciPy**: Fundamental library for scientific computing
- **Matplotlib**: Comprehensive 2D Plotting
- **Sympy:** Symbolic mathematics
- **Pandas:** Data structures & analysis
- **IPython:** Enhanced Interactive Console
  - *ipython notebook —> now Jupyter —>* ***JupyterLab***

Python cheatsheet:

[*http://yogen.io/assets/pdfs/Python3Fundamentals.pdf*](http://yogen.io/assets/pdfs/Python3Fundamentals.pdf)

Quick Start
jupyter notebook

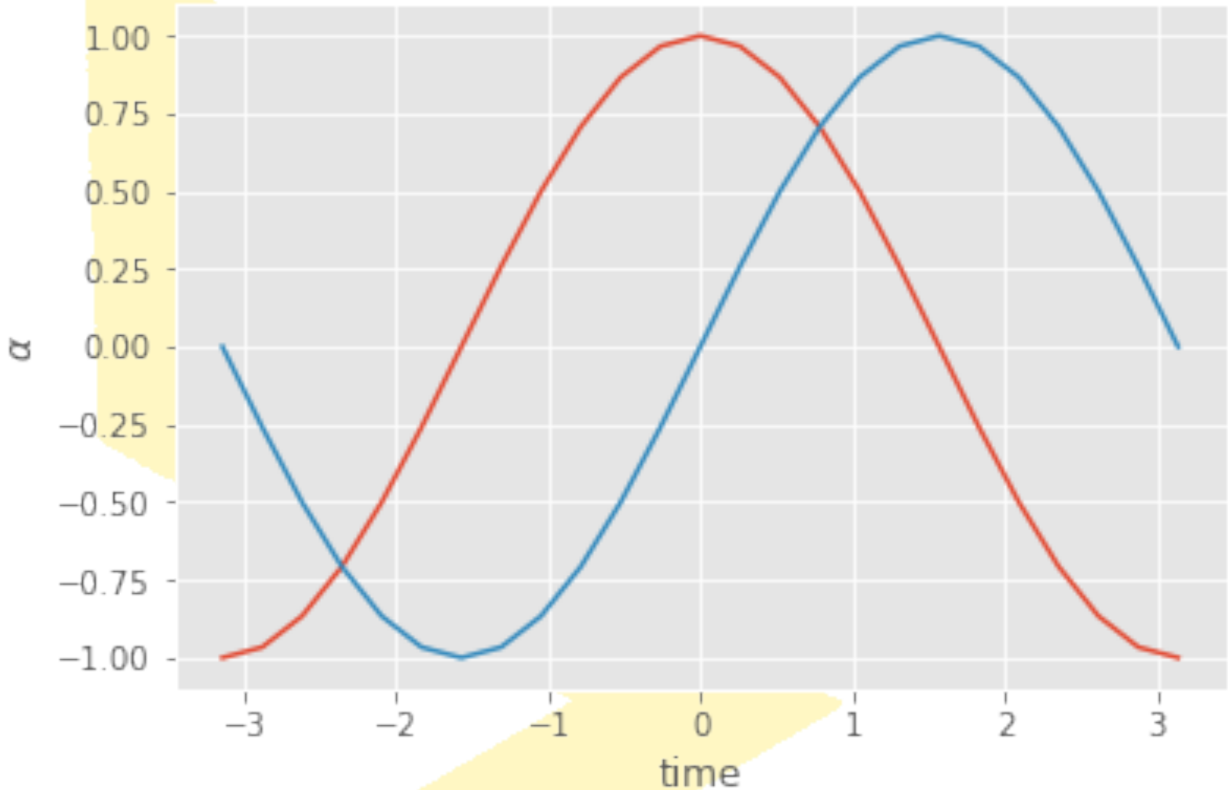
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib

matplotlib.style.use('ggplot')
%matplotlib inline
```

Matplotlib

Example: plot sin and cosine functions for one period using matplotlib.

X = np.linspace(-np.pi, np.pi, 25)
C,S = np.cos(X), np.sin(X)

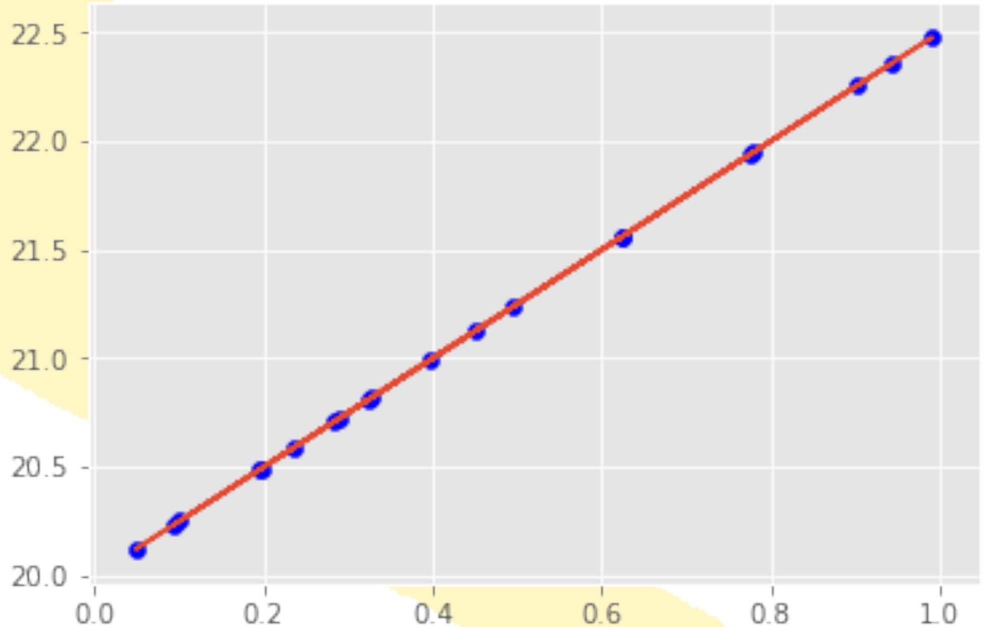


help(plt.plot)

Numpy

Example: Create a sample of points that follow the equation \$Y = AX + B\$, where A = 2.5 and B = 20.

Now, plot it as either a cloud of points or a line.



Numpy

Example: represent the logistic, or sigmoid, function between -10 and 10. Remember its formula is:

(*) Latex
$$\displaystyle S(x)={\frac {1}{1+e^{-x}}}={\frac {e^{x}}{e^{x}+1}}$$

$$S(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}$$

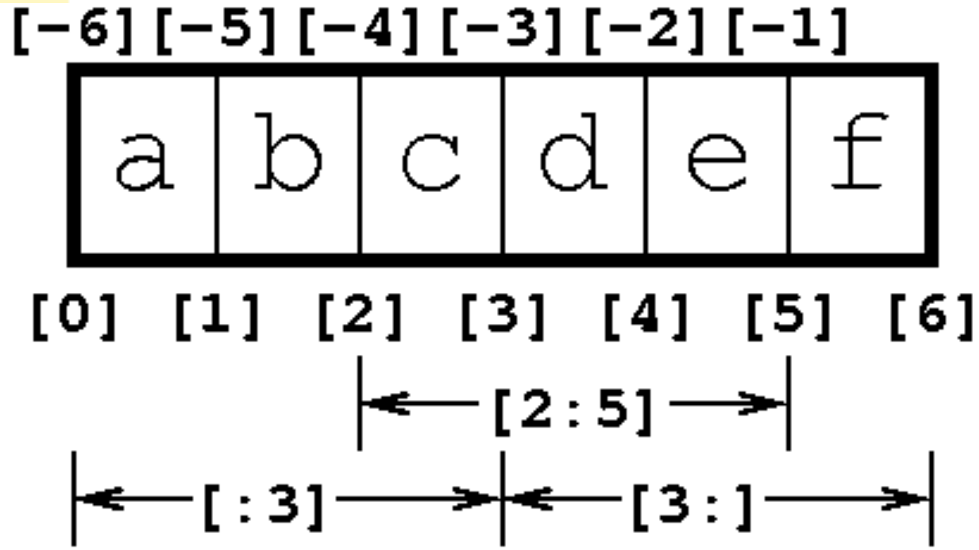
Hint: you will need an X and a Y to plot against it.

Hint: check out the function np.exp

(*) https://www.cs.princeton.edu/courses/archive/spr10/cos433/Latex/latex-guide.pdf

Numpy

Indexing and slicing



Copying arrays!

Numpy

Element wise operations
Matrix operations
ndarrays vs matrix

Lineal Algebra: linalg:

- Trace, determinant, inverse.
- Norm

Exercise: in a chicken and rabbit farm, there are 35 heads and 94 legs. How many chickens and how many rabbits do we have?

$$A \cdot X = B$$

$$A^{-1} \cdot A \cdot X = I \cdot X = A^{-1} \cdot B$$

$$X = A^{-1} \cdot B$$

$$\begin{pmatrix} A11 & A12 \\ A21 & A22 \end{pmatrix} \begin{matrix} x1 \\ x2 \end{matrix} = \begin{matrix} b1 \\ b2 \end{matrix}$$

x1 = chicken #
x2 = rabbit #
b1 = head #
b2 = legs #

A Linear Regression Example using Numpy

Hypothesis function: $$h_\theta(x) = \theta_0 + \theta_1 x$$

Cost function: $$J(\theta_0, \theta_1) = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2 = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i)^2$$

Generating sample data

theta_0 = 2
theta_1 = 5
X = (np.random.randn(100) + 1) * 50
jitter = 50 * np.random.randn(100)
Y = theta_0 + theta_1 * X + jitter

(a) from scipy.optimize import fmin

(b) Gradient descent

$$\frac{\partial}{\partial\theta_0} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i)$$

$$\frac{\partial}{\partial\theta_1} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i)\cdot x_i$$

- Initialize variables
- Compute cost function
- Compute gradients
- Update variables: gradient times learning rate (alpha)
- Repeat until convergence: cost at iteration n-1 approx. cost at iteration n

Add comments to the following code:

```
theta_0 = np.random.randn()
theta_1 = np.random.randn()
J_prime_0 = derivative_theta_0(X, Y)
J_prime_1 = derivative_theta_1(X, Y)
convergence_criterion = 1e-1
converged = False
alpha = 10e-5
trace = []

for _ in range(100):
    trace.append([theta_0, theta_1])

    J_0 = J([theta_0, theta_1])

    diff_theta_0 = J_prime_0(theta_0, theta_1)
    diff_theta_1 = J_prime_1(theta_0, theta_1)

    theta_0 = theta_0 - alpha * diff_theta_0
    theta_1 = theta_1 - alpha * diff_theta_1

    J_1 = J([theta_0, theta_1])

    converged = abs(J_0 - J_1) < convergence_criterion
```

1.2 Introduction to pandas

- Data structures
- Indexing and reindexing
- Dropping
- Selection and filtering
- Function application and mapping
- Sorting and ranking
- Summarizing, unique values, value counts, and membership
- Missing data

Data Structures

Series

One-dimensional array-like object containing an array of data (any numpy data type) and an associated array of data labels (index). **Can be created from dictionaries.**

Ex. s = pd.Series([4, 7, -5, 3]) —> index, values

Dataframes

Represents a tabular, spreadsheet-like data structure containing an ordered collection columns, each of which can be a different value type. It has both row and column index (a dict of Series).

Ex. dfdata = { 'province' : ['M', 'M', 'M', 'B', 'B'],
'population': [1.5e6, 2e6, 3e6, 5e5, 1.5e6],
'year' : [1900, 1950, 2000, 1900, 2000]
}; df = pd.DataFrame(dfdata)

Data inputs to DataFrame

Type	Notes
2D ndarray	A matrix of data, passing optional row and column labels
dict of arrays, lists, or tuples	Each sequence becomes a column in the DataFrame. All sequences must be the same length.
NumPy structured/record array	Treated as the "dict of arrays" case
dict of Series	Each value becomes a column. Indexes from each Series are unioned together to form the result's row index if no explicit index is passed.
dict of dicts	Each inner dict becomes a column. Keys are unioned to form the row index as in the "dict of Series" case.
list of dicts or Series	Each item becomes a row in the DataFrame. Union of dict keys or Series indexes become the DataFrame's column labels
List of lists or tuples	Treated as the "2D ndarray" case
Another DataFrame	The DataFrame's indexes are used unless different ones are passed
NumPy MaskedArray	Like the "2D ndarray" case except masked values become NA/missing in the DataFrame result

Indexing and reindexing

- panda’s Index objects are responsible for holding the axis labels and names.
- Any sequence of labels used when constructing a Series/DF is converted to an index.
- Index objects are immutable (can’t be modified by the user).
- *reindex* creates a new object with data conformed to a new index (rearranging and filling if not existing).

Index

Methods and properties

Method	Description
append	Concatenate with additional Index objects, producing a new Index
diff	Compute set difference as an Index
intersec-tion	Compute set intersection
union	Compute set union
isin	Compute boolean array indicating whether each value is contained in the passed collection
delete	Compute new Index with element at index i deleted
drop	Compute new index by deleting passed values
insert	Compute new Index by inserting element at index i
is_monoto-nic	Returns True if each element is greater than or equal to the previous element
is_unique	Returns True if the Index has no duplicate values
unique	Compute the array of unique values in the Index

Functionality

- Drop
- Selection and filtering

Type	Notes
`obj[val]`	Select single column or sequence of columns from the DataFrame. Special case conveniences: boolean array (filter rows), slice (slice rows), or boolean DataFrame (set values based on some criterion).
`obj.ix[val]`	Selects single row of subset of rows from the DataFrame.
`obj.ix[:, val]`	Selects single column of subset of columns.
`obj.ix[val1, val2]`	Select both rows and columns.
`reindex` method	Conform one or more axes to new indexes.
`xs` method	Select single row or column as a Series by label.
`icol`, `irow` methods	Select single column or row, respectively, as a Series by integer location.
`get_value`, `set_value` methods	Select single value by row and column label.

Function application and mapping

- Numpy element-wise array methods (ufuncs) work fine with pandas objects. (**apply** method)
- Element-wise Python functions can be used too: **applymap.**
- **Note that** Series already have **map** method for applying element-wise functions.

Sorting and ranking

- **sort_index** method: to sort lexicographically by row or column index
- **order** method: to sort a Series by its values.
- **rank** method: assigning ranks from one through N in an array. Breaks ties by mean rank but can be assigned according to the different methods.

Method	Description
'average'	Default: assign the average rank to each entry in the equal group.
'min'	Use the minimum rank for the whole group.
'max'	Use the maximum rank for the whole group.
'first'	Assign ranks in the order the values appear in the data.

Summarizing, unique values, value counts, and membership

- **describe** method: general info about the df.
- Statistics to understand the df: **sum, mean, cumsum, std**
- **unique** method: return unique elements.
- **value_counts**
- **isin**

Missing data

- Very important to know **how to deal with missing data,** ideal data frames (with no NaN) only exist on films and internet tutorials.
  - 1) detecting missing values: not trivial with big (unknown) df.
  - 2) filling missing values or dropping them.

Did you know…?

… Jupyter can be used for presentations

```
jupyter nbconvert youripnb.ipynb --to slides --post serve
```

Bibliography:

Python for Data Analysis. Data Wrangling with Pandas, NumPy, and IPython
By William McKinney

2. Loading and saving data

A class on loading csv, excel and database data. Practical example on reading using open data (~2h)

3. Merge, concatenate and transform

A class on joins (~3h)

- **Joins and concatenations.**
- **Data transformation:** string operations, function application.