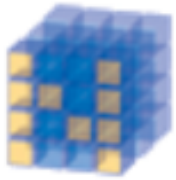


ANACONDA
Powered by Continuum Analytics

pandas
$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

jupyter

16.02.2017

Вычислительные модели с использованием научных библиотек Python

Введение

Цели курса

- Демонстрация возможностей Python для научных расчетов
- Изучение особенностей реализации численных методов
- Практикум по решению вычислительных задач

Темы курса

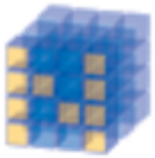
- Символьные вычисления
- Линейная алгебра. Разложение матриц, СЛАУ. Нелинейные уравнения.
- Численное интегрирование, ОДУ
- Условная, безусловная, глобальная оптимизация
- Алгоритмы на графах.
- Статистический анализ, интерполяция, регрессия.
- Спектральный анализ, фильтрация, обработка аудио.
- Обработка изображений и видео данных.
- Многопоточные вычисления, параллельные численные методы.

Инструменты курса

Anaconda – набор библиотек для научных и инженерных расчетов, интерактивная оболочка IPython.

ANACONDA
Powered by Continuum Analytics

NumPy



SciPy

Pandas

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$

Matplotlib

Scikit-Learn

scikit learn

Jupiter Notebook

jupyter

python

Критерии выставления оценки

Наименование	Баллы
Присутствие на семинарах	1
Решение задач на семинарах	2
Реализация проекта	3
Презентация проекта	4
Доп. ответы на вопросы по курсу	1

Проект – исследовательская, вычислительная задача, презентация 7-10 мин.

Основы языка Python

Основные особенности

- Динамическая типизация
- Автоматическое управление памятью
- Модульное программирование
- Сценарии компилируются в байт-код
- Байт код выполняется на виртуальной машине PVM.

Преимущества

- Качество ПО (в т.ч. читаемость кода)
- Скорость разработки
- Переносимость программ
- Разнообразие библиотек
- Интеграция с другими ЯП

Основы языка Python, lists

#1

```
>>> colors = ['red', 'blue', 'green', 'black', 'white']

>>> colors[2]
'green'

>>> colors[-1]
'white'
>>> colors[-2]
'black'

>>> colors[2:4]
['green', 'black']


>>> colors[0] = 'yellow'
>>> colors
['yellow', 'blue', 'green', 'black', 'white']
>>> colors[2:4] = ['gray', 'purple']
>>> colors
['yellow', 'blue', 'gray', 'purple', 'white']
```

#2

```
>>> colors.append('pink')
>>> colors
['red', 'blue', 'green', 'black', 'white', 'pink']

>>> colors.pop() # removes and returns the last item
'pink'

>>> colors.extend(['pink', 'purple']) # extend colors, in-place
```

#3

```
>>> rcolors = colors[::-1]
>>> rcolors
['white', 'black', 'green', 'blue', 'red']

>>> rcolors.reverse() # in-place
```

#4

```
>>> rcolors + colors
['white', 'black', 'green', 'blue', 'red', 'red', 'blue', 'green', 'black', 'white']
>>> rcolors * 2
['white', 'black', 'green', 'blue', 'red', 'white', 'black', 'green', 'blue', 'red']
```

#5

```
>>> rcolors.sort() # in-place
>>> rcolors
['black', 'blue', 'green', 'red', 'white']
```

Основы языка Python, dictionary, tuple, set

#1

```
>>> tel = {'emmanuelle': 5752, 'sebastian': 5578}
>>> tel['francis'] = 5915
>>> tel
{'sebastian': 5578, 'francis': 5915, 'emmanuelle': 5752}
>>> tel['sebastian']
5578

>>> tel.keys()
['sebastian', 'francis', 'emmanuelle']

>>> tel.values()
[5578, 5915, 5752]
>>> 'francis' in tel
True
```

#2

```
>>> t = 12345, 54321, 'hello!'
>>> t[0]
12345
>>> t
(12345, 54321, 'hello!')
>>> u = (0, 2)
```

#3

```
>>> s = (('a', 'b', 'c', 'a'))
>>> s
set(['a', 'c', 'b'])

>>> s.difference(('a', 'b'))
set(['c'])
```

Python коллекции: https://habrahabr.ru/post/319164/

Основы языка Python, control flow

#1

```
>>> a = 10
>>> if a == 1:
...         print(1)
...     elif a == 2:
...         print(2)
...     else:
...         print('A lot')
A lot
```

#2

```
>>> a = [1, 0, 2, 4]
>>> for element in a:
...         if element == 0:
...             continue
...         print(1. / element)
1.0
0.5
0.25
```

#3

```
>>> d = {'a': 1, 'b':1.2, 'c':1j}

>>> for key, val in (d.items()):
...         print('Key: %s has value: %s' % (key, val))
Key: a has value: 1
Key: b has value: 1.2
Key: c has value: 1j
```

#4

```
>>> z = 1 + 1j
>>> while abs(z) < 100:
...         if z.imag == 0:
...             break
...         z = z**2 + 1
```

Основы языка Python, functions

#1

```
>>> def try_to_modify(x, y, z):
...         x = 23
...
...         y.append(42)
...         z = [99] # new reference
...         print(x)
...         print(y)
...         print(z)
...
>>> a = 77 # immutable variable
>>> b = [99] # mutable variable
>>> c = [28]
>>> try_to_modify(a, b, c)
23
[99, 42]
[99]
>>> print(a)
77
>>> print(b)
[99, 42]
>>> print(c)
[28]
```

#2

```
>>> def variable_args(*args, **kwargs):
....         print 'args is', args
....         print 'kwargs is', kwargs
....
>>> variable_args('one', 'two', x=1, y=2, z=3)
args is ('one', 'two')
kwargs is {'y': 2, 'x': 1, 'z': 3}
```

Основы языка Python, OOP

#1

```
>>> class Student(object):
...     def __init__(self, name):
...         self.name = name
...     def set_age(self, age):
...         self.age = age
...     def set_major(self, major):
...         self.major = major
>>> anna = Student('anna')
>>> anna.set_age(21)
>>> anna.set_major('physics')
```

#2

```
>>> class MasterStudent(Student):
... internship = 'mandatory, from March to June'
...
>>> james = MasterStudent('james')
>>> james.internship
'mandatory, from March to June'
>>> james.set_age(23)
>>> james.age
23
```

Библиотека NumPy

#1

```
>>> import numpy as np

>>> a = np.array([0, 1, 2, 3])


>>> b = np.array([[0, 1, 2], [3, 4, 5]]) # 2 x 3 array
>>> b
array([[0, 1, 2],
[3, 4, 5]])

>>> b.ndim
2

>>> b.shape
(2, 3)
>>> len(b) # returns the size of the first dimension
2
```

#2

```
>>> a = np.arange(10) # 0 .. n-1 (!)
>>> a
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

>>> b = np.arange(1, 9, 2) # start, end (exclusive), step
>>> b
array([1, 3, 5, 7])
```

#3

```
>>> c = np.linspace(0, 1, 6) # start, end, num-points
>>> c
array([ 0. , 0.2, 0.4, 0.6, 0.8, 1. ])
```

#4

```
>>> a = np.ones((3, 3)) # reminder: (3, 3) is a tuple

>>> b = np.zeros((2, 2))

>>> c = np.eye(3)
>>> c
array([[ 1., 0., 0.],
[ 0., 1., 0.],
[ 0., 0., 1.]])

>>> d = np.diag(np.array([1, 2, 3, 4]))
>>> d
array([[1, 0, 0, 0],
[0, 2, 0, 0],
[0, 0, 3, 0],
[0, 0, 0, 4]])
```

Библиотека NumPy

#1

```
>>> a = np.random.rand(4) # uniform in [0, 1]
>>> a
array([ 0.95799151,  0.14222247,  0.08777354,
0.51887998])

>>> b = np.random.randn(4) # Gaussian
>>> b
array([ 0.37544699, -0.11425369, -0.47616538,
1.79664113])

>>> np.random.seed(1234) # Setting the random

>>> np.random.seed(3)

>>> a = np.random.random_integers(0, 20, 15)
>>> a
array([10, 3, 8, 0, 19, 10, 11, 9, 10, 6, 0, 20, 12, 7,
14])
```

#2

```
>>> x = np.array([[1, 1], [2, 2]])
>>> x.sum(axis=0) # columns (first dimension)
array([3, 3])

>>> x[:, 0].sum(), x[:, 1].sum()
(3, 3)

>>> x[0, :].sum(), x[1, :].sum()
(2, 4)
```

#3

```
>>> a = np.array([1, 2, 3, 4])
>>> a + 1
array([2, 3, 4, 5])

>>> 2**a
array([ 2, 4, 8, 16])

>>> b = np.ones(4) + 1

>>> a - b
array([-1., 0., 1., 2.])

>>> a * b
array([ 2., 4., 6., 8.])
```

#4

```
>>> a = np.arange(5)

>>> np.sin(a)
array([ 0. , 0.84147098, 0.90929743, 0.14112001,
-0.7568025 ])

>>> np.log(a)
array([ -inf, 0. , 0.69314718, 1.09861229,
1.38629436])

>>> np.exp(a)
array([ 1. , 2.71828183, 7.3890561 ,
20.08553692, 54.59815003])
```

Библиотека NumPy

#1

```
>>> a.sort(axis=1)
>>> a
array([[3, 4, 5],
[1, 1, 2]])
```

#2

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

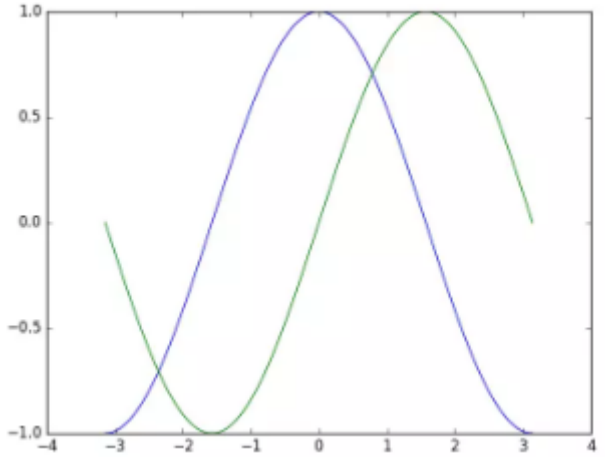
#3

```
>>> a = np.array([4, 3, 1, 2])
>>> j_max = np.argmax(a)
>>> j_min = np.argmin(a)
>>> j_max, j_min
(0, 2)
```

Библиотека Matplotlib

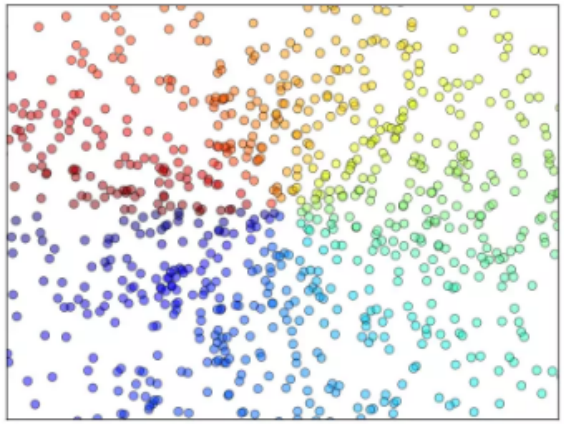
#1

```
import numpy as np
import matplotlib.pyplot as plt
%pylab inline

X = np.linspace(-np.pi, np.pi, 256, endpoint=True)

C, S = np.cos(X), np.sin(X)

plt.plot(X, C)

plt.plot(X, S)

plt.show()
```



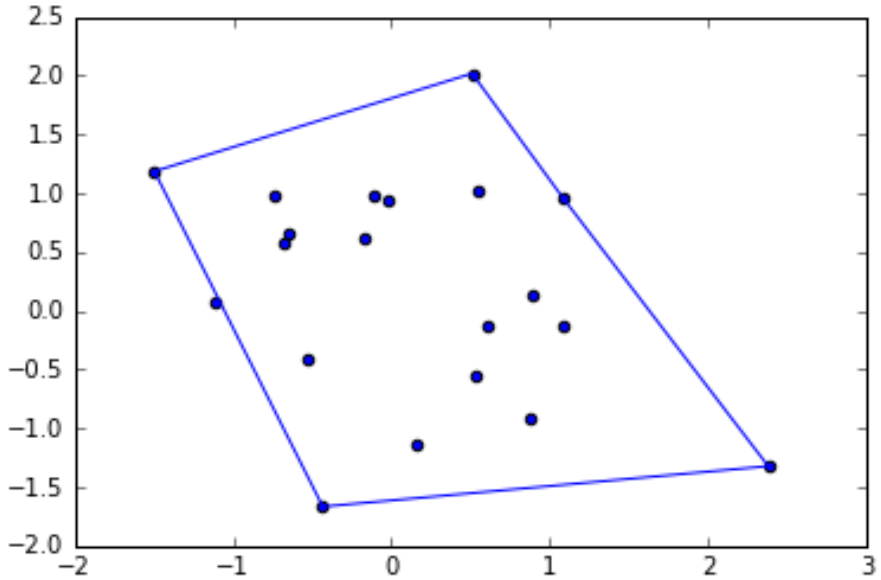
#2

```
n = 1024

X = np.random.normal(0,1,n)

Y = np.random.normal(0,1,n)

plt.scatter(X,Y)
```



Практика, постановка задачи

Задание: построить двумерную выпуклую оболочку случайного множества точек.




```
import numpy as np
import matplotlib.pyplot as plt
%pylab inline

def getConvex(x, y):
….

n =20
x = np.random.randn(n)
y = np.random.randn(n)
convex = getConvex(x, y)
xConv = x[convex]
yConv = y[convex]
plt.plot(xConv,yConv)
plt.scatter(x,y)
plt.show()
```

Практика, алгоритм giftwrapping

Задание: построить двумерную выпуклую оболочку случайного множества точек.

```
1) p[1] = самая левая нижняя точка множества P;
2) p[2] = соседняя точка от p[1] справа (находится через
   минимальный положительный полярный угол)
3) i = 2;
4) do: p[i+1] = любая точка из P (кроме уже попавших в
выпуклую оболочку, но включая p[1]);
  для каждой точки j от 1 до |P|, кроме уже попавших в
выпуклую оболочку, но включая p[1]
       p[i+1] = point_with_min_cos(p[i-1], p[i], P[j]);
//точка, образующая минимальный косинус с прямой p[i-
1]p[i],

 while p[i] != p[1]
5) return p;
```