Раздел «Язык Си» . ExtrapolPython :

Экстраполяция

Постановка задачи

У нас есть много экспериментальных точек. Через них надо провести кривую, которая как можно ближе проходила к этим точкам.

Почему нельзя провести интерполяционный полином?

Это другая задача.

- Точки получены с некоторой погрешностью;
- интерполяционный полином будет либо очень большой степени (будет долго считаться; возникнут биения ближе к краям интервала, на котором проведена интерполяция; резкий рост за пределами интервала интерполяции)
- сплайн-интерполяция будет учитывать только несколько точек на краях, почти игнорируя другие точки (а нам хочется, чтобы все точки внесли вклад в построение кривой).
- задача построения значений вне отрезка интерполяции не решается интерполяционными полиномами.
- точки получены с ошибкой (измерения), поэтому прямо через них не обязательно проводить кривую, достаточно, чтобы она проходила через некоторую окрестность точки.
- количество уравнений, определяющих точки, больше, чем количество неизвестных. Т.е. система переопределена, точное решение невозможно.

Что значит "ближе подходит"?

90-60-90

- сумма отклонений
- сумма модулей отклонений
- сумма квадратов отклонений

Какие пакеты могут решать эту задачу?

- numpy - numpy.linalg.lstsq
- scipy - scipy.linalg

    - scipy.linalg содержит все функции из numpy.linalg плюс часть новых функций, которых нет в numpy.linalg
    - Еще одним преимуществом использования scipy.linalg вместо numpy.linalg является обязательная компиляция с поддержкой scipy.linalg over numpy.linalg, в то время как в numpy это не обязательная опция. То есть, в зависимости от того, какой именно пакет numpy установлен, эти же функции из scipy будут считаться быстрее.
- LMFIT? - устанавливается отдельно, тут не описана.

Приближение линейной функцией с помощью numpy.linalg.lstsq

Проведем прямую y = mx + c через экспериментальные точки. В примере только 4 точки, чтобы было меньше писать.

```
>>> import numpy as np
>>> x = np.array([0, 1, 2, 3])
>>> y = np.array([-1, 0.2, 0.9, 2.1])
```

Перепишем линейное уравнение y = mx + c как y = Ap, где A = [[x 1]] и p = [[m], [c]]

Построим A по x :

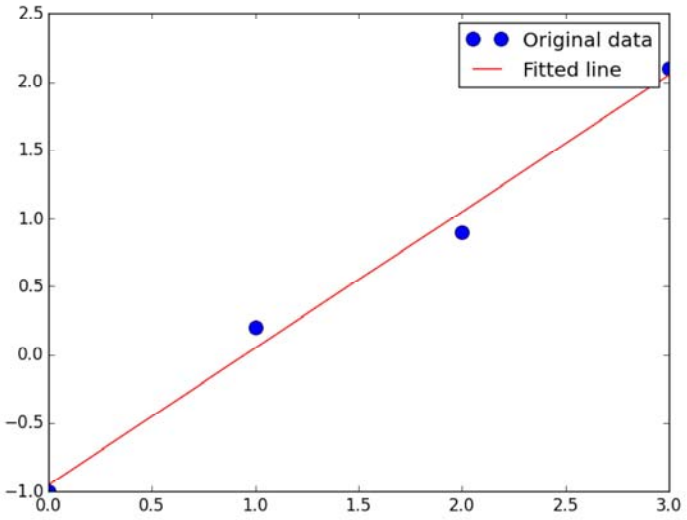
```
>>> A = np.vstack([x, np.ones(len(x))]).T
>>> A
array([[ 0.,  1.],
       [ 1.,  1.],
       [ 2.,  1.],
       [ 3.,  1.]])
```

Используем lstsq для решения его относительно вектора p.

```
>>> m, c = np.linalg.lstsq(A, y)[0]
>>> print m, c
1.0 -0.95
```

Построим график полученной прямой и укажем на нем точки.

```
>>> import matplotlib.pyplot as plt
>>> plt.plot(x, y, 'o', label='Original data', markersize=10)
>>> plt.plot(x, m*x + c, 'r', label='Fitted line')
>>> plt.legend()
>>> plt.show()
```



Подгонка методом наименьших квадратов в scipy (парабола)

lstsq(m,v) – приближенное решение системы линейных уравнений по методу наименьших квадратов.

Например: пусть x,y – вектора длиной n > 3 (точек > 3).

Задача: найти такие a, b, c, чтобы было $y = ax^2 + bx + c$ (аппроксимация параболой). Задача переопределена (n уравнений, 3 неизвестных) и точного решения не имеет.

Генерация файла данных (используем модуль numpy)

Данные должны быть получены в результате измерений, но мы показываем пример, поэтому сделаем эти данные сами. Возьмем функцию и добавим случайные отклонения в координаты x и y.

```
from numpy import *
from numpy.random import *
delta=1.0
x=linspace(-5,5,11)
y=x**2+delta*(rand(11)-0.5)
x+=delta*(rand(11)-0.5)
x.tofile('x_data.txt', '\n')
y.tofile('y_data.txt', '\n')
```

Получили данные x в файле x_data.txt

```
-4.53349565537
-4.09365239181
-3.46164907362
-2.27536520922
-0.799518640116
0.258848610464
0.678142088827
1.67746174919
3.33979664705
3.87971285728
4.61988761783
```

Получили данные y в файле y_data.txt

```
24.9934393435
16.0864373629
9.35597227962
4.40895322612
1.05652519476
0.0593537295658
1.07912259931
4.22402583041
8.77775630747
15.9549972433
24.9164389595
```

Нахождение коэффициентов функции вида $y = ax^2 + bx + c$ методом наименьших квадратов

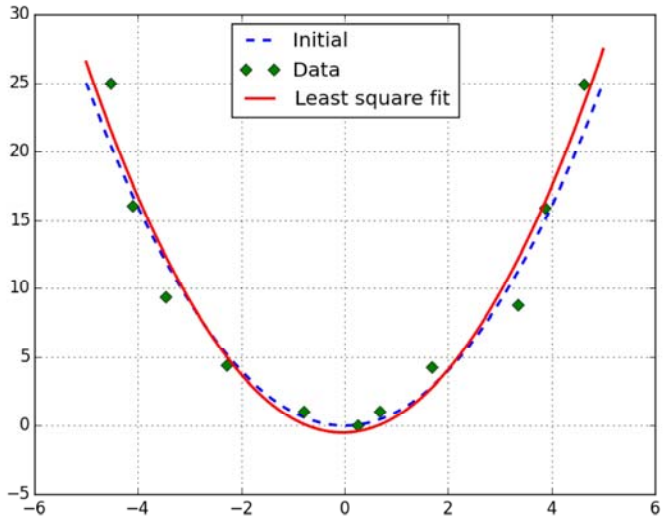
```
from pylab import *
from scipy.linalg import *

# читаем данные из файлов
x=fromfile('x_data.txt',float,sep='\n')
y=fromfile('y_data.txt',float,sep='\n')

# задаем вектор m = [x**2, x, E]
m=vstack((x**2,x,ones(11))).T
# находим коэффициенты при составляющих вектора m
s=lstsq(m,y)[0]

# на отрезке [-5,5]
x_prec=linspace(-5,5,101)
# рисуем теоретическую кривую x<sup>2</sup>
plot(x_prec,x_prec**2,'--',lw=2)
# рисуем точки
plot(x,y,'D')
# рисуем кривую вида y = ax<sup>2</sup> + bx + c, подставляя из решения коэффициенты s[0], s
plot(x_prec,s[0]*x_prec**2+s[1]*x_prec+s[2],'-',lw=2)

grid()
legend(('Initial','Data','Least square fit'),9)
savefig('plot4.png')
```



Еще один пример для scipy (произвольная функция)

Пусть мы проверяем гипотезу, что наши точки ложатся на кривую вида f (x,b) = b0 + b1*exp(-b2*x**2)

Генерация тестовых данных

Добавим шума в данные, сделанные по функции f(x,b) с коэффициентами b = (0.25, 0.75, 0.5)

```
beta = (0.25, 0.75, 0.5)
def f(x, b0, b1, b2):
    return b0 + b1 * np.exp(-b2 * x**2)
# зададим массив точек xi
xdata = np.linspace(0, 5, 50)
# создаем теоретически правильные значения точек yi (без шума)
y = f(xdata, *beta)
# зашумляем эти данные
ydata = y + 0.05 * np.random.randn(len(xdata))
```

Решение

Используем функцию для получения решения в виде коэффициентов функции f (x) для указанных xdata и ydata

```
beta_opt, beta_cov = optimize.curve_fit(f, xdata, ydata)
beta_opt
array([ 0.25733353, 0.76867338, 0.54478761])
```

Вычислим, насколько велико линейное отклонение

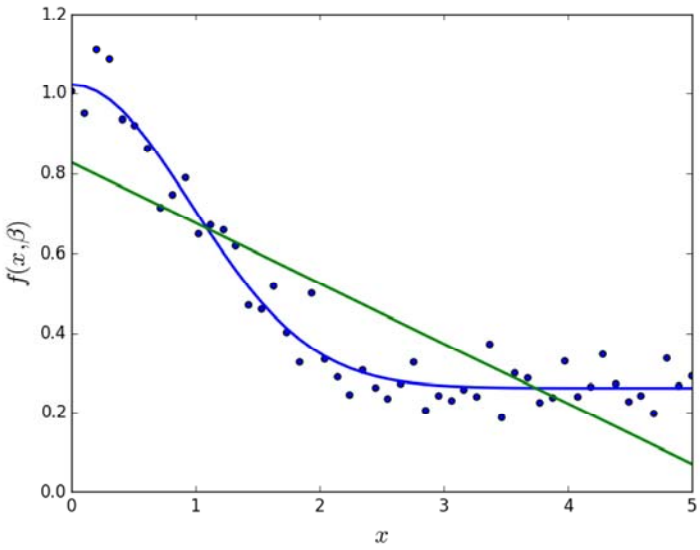
```
lin_dev =  sum(beta_cov[0])
print lin_dev
```

Вычислим, насколько велико квадратичное отклонение

```
residuals = ydata - f(xdata,*beta_opt)
fres = sum(residuals**2)
print fres
```

Нарисуем полученное решение

```
fig, ax = plt.subplots()
ax.scatter(xdata, ydata)
ax.plot(xdata, y, 'r', lw=2)
ax.plot(xdata, f(xdata, *beta_opt), 'b', lw=2)
ax.set_xlim(0, 5)
ax.set_xlabel(r"$x$", fontsize=18)
ax.set_ylabel(r"$f(x, \beta)$", fontsize=18)
plt.show()
```



- optimize.curve_fit:

Attachment	Action	Size	Date	Who	Comment
figure_1.png	manage	25.0 K	23 Apr 2016 - 18:36	TatyanaDerbysheva	np.linalg.lstsq
plot4.png	manage	39.4 K	23 Apr 2016 - 21:03	TatyanaDerbysheva	scipy.linalg.lstsq
figure_3.png	manage	35.8 K	27 Apr 2016 - 22:32	TatyanaDerbysheva	optimize.curve_fit