gradient_descent

April 10, 2016

1 Градиентный спуск. Метод наискорейшего спуска.

Иванычев Сергей, 376 группа

Python 2/3

Будем оптимизировать функцию

$$f(x) = ||Ax - b||^2 \to \min$$

Приведем функцию к квадратичному виду (как во второй задаче)

$$f(x) = \frac{1}{2}\langle x, 2A^\intercal Ax\rangle - \langle x, 2A^\intercal b\rangle$$

Условие на шаг:

$$\alpha_k = \frac{\langle \nabla f(x_k), \nabla f(x_k)\rangle}{\langle \nabla f(x_k), A\nabla f(x_k)\rangle}$$

In [1]:
```
import numpy as np
import scipy as sc
from functools import reduce
from scipy.optimize import minimize
from scipy.optimize import check_grad
from scipy.optimize import approx_fprime
from math import sqrt
```

In [2]:
```
PHONE_NUMBER = "89104577995"
NUMSUM = sum((int(x) for x in PHONE_NUMBER))
print("Вариант %d" % NUMSUM)
```

Вариант 64

Здесь реализуем функцию, генерирующую случайные $f(x)$

In [3]:
```
def generate_f(ndim):
    A = np.random.rand(ndim, ndim)
    b = np.random.rand(ndim, 1)
    def f(x, *args):
        return np.linalg.norm(np.dot(A, x) - b)
    return f, A
```

f_min требуется в моем варианте в условии остановки. Вычислим его стандартными методами с помощью модуля scipy.optimize

In [4]:
```
ndim = NUMSUM % 6 + 2
f, A = generate_f(ndim)
f_min = minimize(f, np.zeros(ndim)).fun
eps = 10**(-ndim % 3)
print(f_min)
print(ndim)
```

```
1.36024685691
6
```

Здесь напишем функцию-обертку, вычисляющую градиент произвольной функции $f(x)$

In [5]:
```
def gradient(x, f, eps=sqrt(np.finfo(float).eps)):
    return approx_fprime(x, f, eps)
```

In [6]:
```
x = np.zeros(ndim)
check_grad(f, gradient, x, f)
```

Out[6]: 0.0

1.1 Реализация алгоритма

In [7]:
```
def gradient_descent(f, x_0, f_min, A, eps, iter_max=1000000, verbose=True):
    """
    Input:
    f(x)          callable, x is np.ndarray
    x_0            zero point
    f_min          aka f^*, known minimum of the function f(x)
    A             matrix from the denaminator for alpha_k
    eps           interruption condition
    iter_max       treshold for number of iterations
    verbose        debug output enabled if True

    Return: (minimum value, minimum argument)
    """
    x = x_0
    iters = 0
    while np.abs(f(x) - f_min) > eps and iters < iter_max:
        s = -gradient(x, f)
        alpha = np.dot(s, s) / np.dot(s, np.dot(A, s))
        x = x + alpha * s
        iters += 1
    print("Iterations: ", iters)
    return f(x), x
```

В качестве аргумента A подставляем матрицу преобразования в виде задачи 2 из задания 2

```
In [8]: y, x = gradient_descent(f, np.zeros(ndim) + 3, f_min, np.dot(2*np.transpose(A), A), eps)
        print(y, x)
```

```
('Iterations: ', 20)
(2.089999537453318, array([ 0.26282517,  0.20130258, -0.43759258,  0.2553671 , -0.01573846,
       -0.32319269]))
```