Linear Regression and Least Squares

Method

We will be using least-squares minimization to fit the image data.

We could write the code from scratch, but it is tricky to find the minimum of a function in a high-dimensional space, so it is better to use well-tested code. The scipy library has an algorithm for fitting.

The first thing we want to do is learn to to use their algorithm. This might seem easy, but remember, that the data has to be passed in, the parameters have to be passed in, and the residuals have to be computed, and in order to do this completely generally, abstractions have to be introduced.

More specifically, the least_squares function takes the function to be minimized as its first argument. It is easiest to see this in an example.

Example

The authors of scipy.optimize.least_squares give an example uing the Rosenbrock function which is considered a challenging test for least squares fitting.

In their example, there is no data. Just two parameters are given, x[0] and x[1], and the function that the user supplies produces two residuals.

In [1]:
```
import numpy as np
from scipy.optimize import least_squares

def the_rosenbrock_function(parameter_vector, data=None):
    parameter_a = parameter_vector[0]
    parameter_b = parameter_vector[1]
    residual_1 = parameter_b - parameter_a**2
    residual_2 = 1 - parameter_a
    residual_vector = [residual_1, residual_2]
    return np.array(residual_vector)

initial_guess_for_parameter_a = 2
initial_guess_for_parameter_b = 2

initial_parameter_vector = [initial_guess_for_parameter_a, initial_guess_for_parameter_b]


result = least_squares(the_rosenbrock_function, np.array(initial_parameter_vector))

print(result.x)
```

```
[1. 1.]
```

Problem 1

Note that in the above example, the data argument wasn't used, and of course in our imaging problem data is essential. So let's do a problem that has some data.

Below is some data consisting of an array of 13 ordered pairs. You can think of these ordered pairs as x-y values, and we want to find the best values for the parameters m and b with y = m * x + b.

Write a function that produces the residuals, and then use scipy.optimize.least_squares to find the best fit.

In [2]:

```
# The data

some_data = [
    (24, 196),
    (26, 194),
    (28, 195),
    (29, 194),
    (31, 192),
    (32, 191),
    (34, 193),
    (36, 192),
    (38, 191),
    (40, 192),
    (41, 193),
    (42, 190),
    (43, 190)
]

# My solution

def make_my_function(data):
    def my_function(parameter_vector):
        m = parameter_vector[0]
        b = parameter_vector[1]
        return np.array([item[1] - (m * item[0] + b) for item in data])

    return my_function

initial_guess_for_m = 0.0
initial_guess_for_b = 0.0

initial_parameter_vector = [initial_guess_for_m, initial_guess_for_b]

my_function = make_my_function(some_data)

result = least_squares(my_function, np.array(initial_parameter_vector))

print(result.x)
```

```
[ -0.23391167 200.52744479]
```

Problem 2

The above problem was a linear regression problem. We know how to solve it exactly, so we can just compute the exact values of m and b from the data without using scipy.optimize.least_squares.

In [3]:
```
# My solution

# The least-squares minimization quickly yields two equations in the two unknowns m and b.
# The two equations are:
#
# sigma_xy = m * sigma_xx + b sigma_x
# sigma_y = m * sigma_x + b * sigma_1
#
# where

sigma_xx = sum([item[0]**2 for item in some_data])
sigma_xy = sum([item[0] * item[1] for item in some_data])
sigma_x = sum([item[0] for item in some_data])
sigma_y = sum([item[1] for item in some_data])
sigma_1 = len(some_data)

# The two unknowns are given by:

determinant = sigma_1 * sigma_xx - sigma_x**2

m = (sigma_1 * sigma_xy - sigma_x * sigma_y) / determinant
b = (sigma_x * sigma_xy - sigma_xx * sigma_y) / determinant

print([m, b])
```

```
[-0.23391167192429022, -200.5274447949527]
```

Problem 3

Compare and reconcile the answers to Problems 1 and 2.

My Solution

Well, they compare perfectly, to all decimal places shown in Problem 1, so there is no reconciling to do.