University of Illinois at

Urbana-Champaign

**Scipy Optimizer Performance**

**---- Estimation of Functions**

SPRING 2023

![图形用户界面

描述已自动生成]()

**Feng Zhao**

March 22, 2023

Table of Contents

[1. General Table 3](#_Toc131329802)

[2. Output Constraint Setup 7](#_Toc131329803)

[2.1 Performance vs. Data Points Distribution & Functions 7](#_Toc131329804)

[2.2 Fit Output Bound in Scipy Optimizers 8](#_Toc131329805)

[2.2.1 Constraints Allowerance For Each Optimizer 8](#_Toc131329806)

[2.2.2 Norm Function 9](#_Toc131329807)

# General Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Iterations | Output Constraint | Jacobian | Hessian |
| SLSQP | / | Accept | Yes | No |
|
|
|
| trust-constr | Similar[[1]](#footnote-1)  (In comparison to SLSQP) | Accept | Yes | No |
|
|
|
| COBYLA | More  (In comparison to SLSQP) | Accept | No | No |
|
|
|
| Nelder-Mead | More  (In comparison to SLSQP) | N/A | No | No |
| BFGS | / | N/A | No | Yes |
| CG | More[[2]](#footnote-2)  (In comparison to BFGS) | N/A | Yes | No |
| Newton-CG | Similar  (In comparison to BFGS) | N/A | Yes | Yes |

\*From Perfect – Good – Medium – Bad – Terrible, it means the over-fitting is more serious.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Data Points Distribution | Function Choice | # Data Points (N) | Degree Level  (K) | Degree of Overfitting |
| SLSQP | Uniform:  Take the number of data points equal to N by averaging across the -10 to 10 interval. | Single Sigmoid  绿色的钟表  低可信度描述已自动生成 | 10 | 5 | good |
| 10 | terrible |
| 100 | 5 | good |
| 10 | terrible |
| Combined Sigmoid | 10 | 5 | terrible |
| 10 | terrible |
| 100 | 5 | medium |
| 10 | terrible |
| Rate Changing  图片包含 文本  描述已自动生成 | 10 | 5 | good |
| 10 | bad |
| 100 | 5 | perfect |
| 10 | good |
| Random:  Take the equal number of data points by randomly sampling from the -10 to 10 interval using np.random.uniform. | Single Sigmoid | 10 | 5 | good |
| 10 | terrible |
| 100 | 5 | good |
| 10 | terrible |
| Combined Sigmoid | 10 | 5 | medium |
| 10 | terrible |
| 100 | 5 | medium |
| 10 | terrible |
| Rate Changing | 10 | 5 | medium |
| 10 | terrible |
| 100 | 5 | medium |
| 10 | terrible |
| Adjusted(weighted) Random:  Under the premise of using random sampling, assign weights to the data points. Take 40% of the required data points from the -10 to -5 range, 40% from the 5 to 10 range, and the remaining 20% from the -5 to 5 range. | Single Sigmoid | 10 | 5 | good |
| 10 | terrible |
| 100 | 5 | good |
| 10 | terrible |
| Combined Sigmoid | 10 | 5 | good |
| 10 | bad |
| 100 | 5 | good |
| 10 | terrible |
| Rate Changing | 10 | 5 | medium |
| 10 | medium |
| 100 | 5 | good |
| 10 | terrible |
| trust-constr | Uniform | Single Sigmoid | 10 | 5 | perfect |
| 10 | bad |
| 100 | 5 | perfect |
| 10 | medium |
| Combined Sigmoid | 10 | 5 | perfect |
| 10 | bad |
| 100 | 5 | perfect |
| 10 | medium |
| Rate Changing | 10 | 5 | perfect |
| 10 | bad |
| 100 | 5 | perfect |
| 10 | bad |
| Random | Single Sigmoid | 10 | 5 | perfect |
| 10 | good |
| 100 | 5 | good |
| 10 | medium |
| Combined Sigmoid | 10 | 5 | perfect |
| 10 | bad |
| 100 | 5 | perfect |
| 10 | bad |
| Rate Changing | 10 | 5 | perfect |
| 10 | terrible |
| 100 | 5 | perfect |
| 10 | terrible |
| Adjusted(weighted) Random | Single Sigmoid | 10 | 5 | good |
| 10 | bad |
| 100 | 5 | good |
| 10 | terrible |
| Combined Sigmoid | 10 | 5 | perfect |
| 10 | bad |
| 100 | 5 | perfect |
| 10 | medium |
| Rate Changing | 10 | 5 | perfect |
| 10 | medium |
| 100 | 5 | perfect |
| 10 | medium |
| COBYLA | Uniform | Single Sigmoid | 10 | 5 | good |
| 10 | bad |
| 100 | 5 | good |
| 10 | medium |
| Combined Sigmoid | 10 | 5 | terrible |
| 10 | terrible |
| 100 | 5 | terrible |
| 10 | medium |
| Rate Changing | 10 | 5 | perfect |
| 10 | bad |
| 100 | 5 | perfect |
| 10 | medium |
| Random | Single Sigmoid | 10 | 5 | good |
| 10 | terrible |
| 100 | 5 | good |
| 10 | medium |
| Combined Sigmoid | 10 | 5 | terrible |
| 10 | bad |
| 100 | 5 | terrible |
| 10 | bad |
| Rate Changing | 10 | 5 | good |
| 10 | good |
| 100 | 5 | medium |
| 10 | bad |
| Adjusted(weighted) Random | Single Sigmoid | 10 | 5 | good |
| 10 | terrible |
| 100 | 5 | good |
| 10 | terrible |
| Combined Sigmoid | 10 | 5 | terrible |
| 10 | terrible |
| 100 | 5 | terrible |
| 10 | terrible |
| Rate Changing | 10 | 5 | perfect |
| 10 | perfect |
| 100 | 5 | perfect |
| 10 | bad |

|  |  |  |
| --- | --- | --- |
|  | Pros | Cons |
| SLSQP | * Can handle problems with both linear and nonlinear constraint conditions. * Can effectively handle large-scale problems (large size of data points). | * For complex problems, the algorithm may require multiple iterations to find the optimal solution, which may cause over-fitting seriously. |
| trust-constr | * It can generally handle linear and nonlinear constraint conditions. * It can quickly find the optimal solution and balance between global and local convergence. | * May encounter limitations when handling large-scale problems if Jacobian and Hessian matrices are given. |
| COBYLA | * Can perform calculations without requiring Jacobian and Hessian matrices. * Typically requires a bit more iterations to find the optimal solution. | * Can only handle inequality constraint conditions. |
| All Three |  | * May converge to local minimum that is not the optimal solution. |

# Output Constraint Setup

## Performance vs. Data Points Distribution & Functions

**Data Points Distribution:**

Uniform:

Take the number of data points equal to N by averaging across the -10 to 10 interval.

Random:

Take the equal number of data points by randomly sampling from the -10 to 10 interval using np.random.uniform.

Adjusted(weighted) Random:

Under the premise of using random sampling, assign weights to the data points. Take 40% of the required data points from the -10 to -5 range, 40% from the 5 to 10 range, and the remaining 20% from the -5 to 5 range. **By allocating more data points to the smoother region of the sigmoid function, we can indirectly force the optimizer to fit more precisely within that region, thus avoiding overfitting.**

* For the Sigmoid function, COBYLA performs well with uniform and random data point distributions, but not with adjusted random data point distribution; Trust\_Constr works well with random and adjusted random data point distributions, but not with uniform data point distribution; SLSQP is suitable for all types of data point distributions.
* For the Combined Sigmoid function, COBYLA works well with all types of data point distributions; Trust\_Constr works well with uniform and adjusted random data point distributions, but not with random data point distribution; SLSQP is suitable for all types of data point distributions.
* For the Rate-changing function, COBYLA performs well with random and adjusted random data point distributions, but not with uniform data point distribution; Trust\_Constr is suitable for all types of data point distributions; SLSQP is suitable for all types of data point distributions.

Here is the overall performance in table format:

|  |  |  |  |
| --- | --- | --- | --- |
|  | COBYLA | Trust\_Constr | SLSQP |
| Sigmoid - Uniform | Medium | Bad | Good |
| Sigmoid - Random | Good | Good | Perfect |
| Sigmoid - Adjusted Random | Bad | Perfect | Perfect |
| Combined Sigmoid - Uniform | Good | Good | Perfect |
| Combined Sigmoid - Random | Medium | Good | Good |
| Combined Sigmoid - Adjusted Random | Good | Good | Perfect |
| Rate-changing - Uniform | Medium | Good | Perfect |
| Rate-changing - Random | Good | Perfect | Good |
| Rate-changing - Adjusted Random | Good | Perfect | Perfect |

## Fit Output Bound in Scipy Optimizers

### Constraints Allowerance For Each Optimizer

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Optimizer | Search Space Definition | Inequality Constraints | Equality Constraints | Explicit/Implicit Constraints | Linear/Non-Linear  Constraints |
| SLSQP | Bounds | Yes | Yes | Explicit | Both |
| Trust-Constr | Bounds | Yes | Yes | Inequality only | Both |
| COBYLA | Constraints | Yes | No | Both explicit and implicit | Both |

Trust-Constr, SLSQP, and COBYLA all accept both linear and nonlinear constraints. However, Trust-Constr is more suited for problems with nonlinear constraints, while SLSQP and COBYLA are better suited for problems with linear constraints.

SLSQP and COBYLA optimizers limit the search space by using bounds as a parameter, while the trust-constr optimizer uses constraints as a parameter to define the search space.

SLSQP can handle explicit equality and inequality constraints, while COBYLA can only handle inequality constraints. Trust-constr optimizer can handle both explicit and implicit equality and inequality constraints.

**Therefore, the main difference between the optimizers are the SLSQP and COBYLA optimizer prefer to use bounds to limit the search space, while the trust-constr optimizer uses constraints to define the search space. The difference between them lies in whether the constraints are explicitly expressed.**

It should be pointed out that the SLSQP optimizer prefer to use bounds to limit the search space, but it can also handle explicit equality and inequality constraints.

![文本

中度可信度描述已自动生成]()

Figure 2.1 Constrained Minimization Bound Set For SLSQP

![文本, 信件

描述已自动生成]()

Figure 2.2 Constrained Minimization Bound Set For trust-constr

![图片包含 图形用户界面

描述已自动生成]()

Figure 1.1 Nonlinear Constraints Mathmatical Expression

### Norm Function

The norm2\_sq function calculates the squared Euclidean distance between the linear function A.dot(x\_choose) and a target vector b. It can be written as:

|| A.x\_choose - b ||2 = (A.x\_choose - b)T (A.x\_choose - b)

where || . || denotes the L2 norm, ^T denotes transpose, and A is a matrix determined by the input vector x\_choose.

In summary, the norm2\_sq function calculates the distance between a linear function A.dot(x\_choose) and a target vector b, which is used as the objective function in the optimization problem. The optimization seeks to find the x\_choose that minimizes the distance subject to constraints, using various optimization methods and constraints. The optimization is subject to constraints specified by the constraints parameter, which can be either inequality constraints on x\_choose or nonlinear constraints on the output of A.dot(x\_choose).

1. From high level to low for # iterations: More – Similar – Less [↑](#footnote-ref-1)
2. From high level to low for # iterations: More – Similar – Less [↑](#footnote-ref-2)