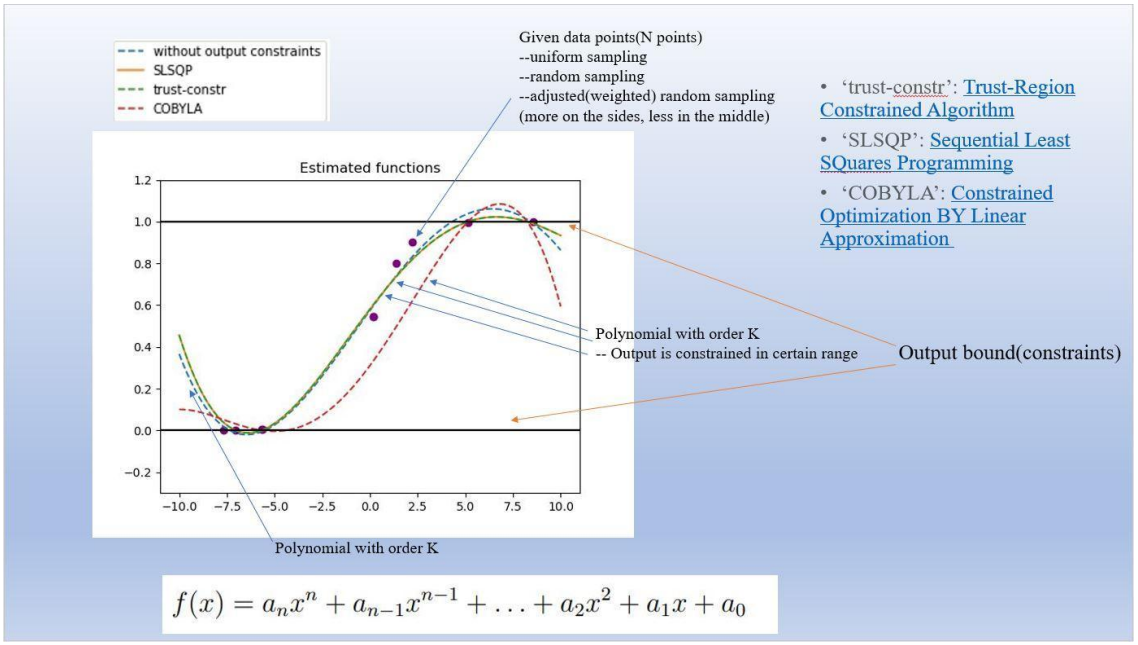
UNIVERSITY OF ILLINOIS AT
URBANA-CHAMPAIGN

Scipy Optimizer Performance

---- Estimation of Functions

SPRING 2023



Feng Zhao

May 12, 2023

Table of Contents

1. General Table

1.1 Tables

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

Table 1.1: General Table

1.2 Three Test Functions Used

The following are the three test functions I used for evaluating the output constraints of the SLSQP, trust-constr, and COBYLA optimizers during my experimentation:

1. Single Sigmoid

$$S(x) = \frac{1}{1 + e^{-x}}$$

2. Combined Sigmoid

$S(x + 5) + S(5 - x) - 1$

3. Rate Changing

[1] From high level to low for # iterations: More – Similar – Less

[2] From high level to low for # iterations: More – Similar – Less

$$\frac{1}{3}\left(e^{(\text{abs}(x)-9)}\right)$$

The following figure is a schematic diagram of the comparison of the three functions when plotted together.

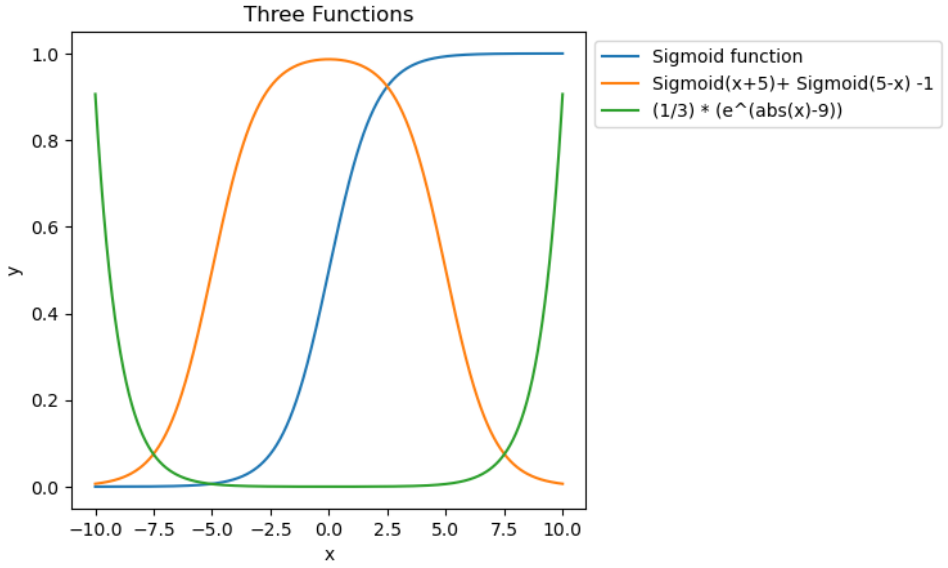


Figure 1.1: Three Test Functions

1.3 Data Points Sampling Methods:

I also added the variable of data point sampling methods in the tests, because different sampling methods can produce biases, and overly random points may lead to an excessive concentration of data points in areas with little change in the function, while not having enough points in areas with dramatic changes in the function. This could potentially lead to a high degree of overall overfitting. Therefore, in addition to Uniform and Random sampling, I also incorporated a weighted random sampling method with artificial control of sampling weight for each region, to conduct more reasonable tests.

Uniform:
Take the number of data points equal to N by averaging across the -10 to 10 interval.
Random:
Take an 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.**

1.4 Heatmaps

The following are three heatmaps illustrating the relationship between different optimizer methods, data point sampling methods, the number of data points, and the order of the generating functions, based on the results obtained from testing each of the test functions separately.

Figure 1.2 Heatmaps for Single Sigmoid


Single Sigmoid
[N=100, K=5], Uniform
[N=100, K=5], Random
[N=100, K=5], Adjusted Random
[N=100, K=10], Uniform
[N=100, K=10], Random
[N=100, K=10], Adjusted Random
[N=10, K=5], Uniform
[N=10, K=5], Random
[N=10, K=5], Adjusted Random
[N=10, K=10], Uniform
[N=10, K=10], Random
[N=10, K=10], Adjusted Random
good
perfect
good
good
good
good
good
good
good
bad
medium
medium
terrible
bad
terrible
medium
bad
medium
good
perfect
good
medium
good
bad
good
good
good
terrible
bad
terrible
terrible
bad
terrible
terrible
bad
terrible
SLSQP
trust-constr
COBYLA
Combined Sigmoid
[N=100, K=5], Uniform
good
perfect
medium

Figure 1.3 Heatmaps for Combined Sigmoid

Figure 1.4 Heatmaps for Rate Changing

* The severity of overfitting increases in the following order: Perfect, Good, Medium, Bad, and Terrible.

1.4.1 Conclusion for Heatmaps:

1. We have observed that for any specific combination of N and K where K is not less than N, using a randomly sampled dataset leads to increased overfitting, irrespective of the optimization method selected.

2. With respect to the three types of optimizer methods, in most situations, the trust-constr optimizer achieves the best model performance. The SLSQP optimizer provides comparable or slightly superior performance in some instances. The COBYLA optimizer generally results in the poorest model performance.

3. Concerning three types of data point sampling: For the Single Sigmoid model and Combined Sigmoid model, the Adjusted Random sampling generally performs the best, which is usually one level higher than the other two, while the performance of Uniform sampling is a good option among the three. Random sampling tends to have poor performance.

Although in certain situations, such as the Single Sigmoid model with N=100 and K=5, the performance of Uniform sampling is better than Adjusted Random sampling. However, in most cases, the latter still outperforms the former. The reason for this, as I have considered, is that for irregular functions, Adjusted Random sampling is more closely aligned with the actual situation compared to pure Uniform sampling. This is because the former intentionally distributes data points closer to the real function's behavior.

The following table provides a comparison of the strengths and weaknesses of the three optimizers:

	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 a local minimum which is not the optimal solution.

Table 1.2: Pros and Cons

2. Output Constraint Setup

2.1 Fit Output Bound in Scipy Optimizers

2.1.1 Constraints Allowance 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

Table 2.1 Constraints Allowance

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 is the SLSQP and COBYLA optimizers 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 prefers to use bounds to limit the search space, but it can also handle explicit equality and inequality constraints.

$$\begin{aligned} \min_x \quad & f(x) \\ \text{subject to:} \quad & c_j(x) = 0, \quad j \in \mathcal{E} \\ & c_j(x) \geq 0, \quad j \in \mathcal{I} \\ & \text{lb}_i \leq x_i \leq \text{ub}_i, \, i = 1, \ldots, N. \end{aligned}$$

Figure 2.1 Constrained Minimization Bound Set For SLSQP

$$\begin{aligned} \min_x \quad & f(x) \\ \text{subject to:} \quad & c^l \leq c(x) \leq c^u, \\ & x^l \leq x \leq x^u. \end{aligned}$$

Figure 2.2 Constrained Minimization Bound Set For trust-constr

Defining Nonlinear Constraints:

The nonlinear constraint:

$$c(x) = \begin{bmatrix} x_0^2 + x_1 \\ x_0^2 - x_1 \end{bmatrix} \leq \begin{bmatrix} 1 \\ 1 \end{bmatrix},$$

with Jacobian matrix:

$$J(x) = \begin{bmatrix} 2x_0 & 1 \\ 2x_0 & -1 \end{bmatrix},$$

and linear combination of the Hessians:

$$H(x, v) = \sum_{i=0}^{1} v_i \nabla^2 c_i(x) = v_0 \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix} + v_1 \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix},$$

is defined using a `NonlinearConstraint` object.

Figure 2.3 Nonlinear Constraints Mathematical Expression

2.1.2 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).