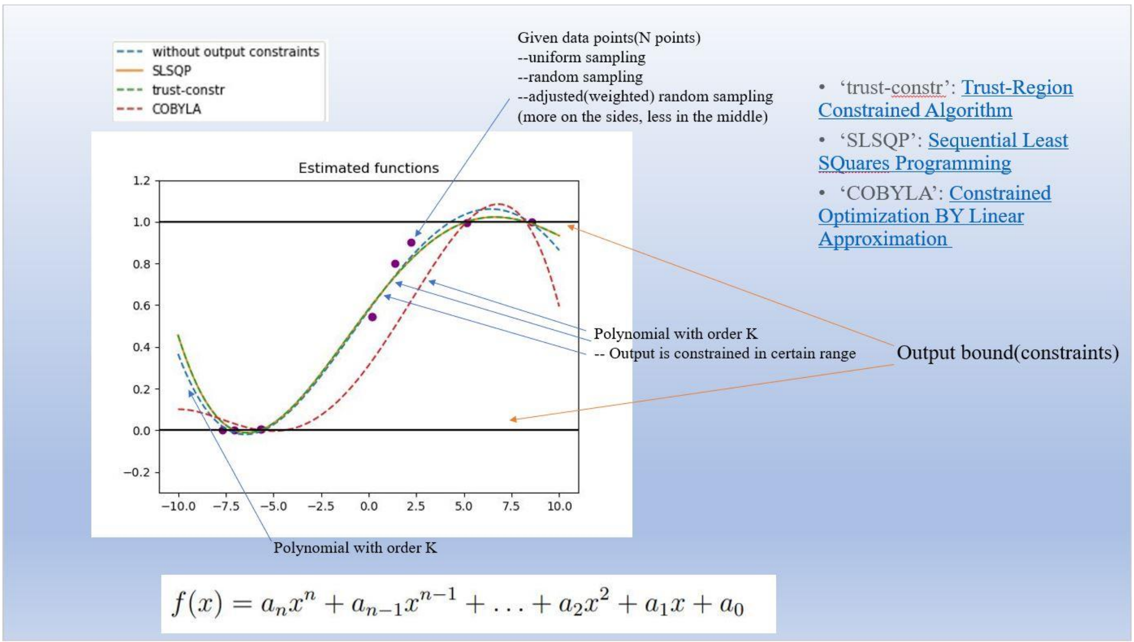
UNIVERSITY OF ILLINOIS AT

URBANA-CHAMPAIGN

Scipy Optimizer Performance

---- Estimation of Functions

SPRING 2023



Feng Zhao

March 22, 2023

Table of Contents

1. General Table

	Iterations	Output Constraint	Jacobian	Hessian
SLSQP	/	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 SLSQP)	N/A	No	No
BFGS	/	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

*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 $S(x) = \frac{1}{1+e^{-x}}$	10	5	good
				10	terrible
			100	5	good
				10	terrible
		Combined Sigmoid $S(x+5) + S(5-x) - 1$	10	5	terrible
				10	terrible
			100	5	medium
				10	terrible

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

		Rate Changing $\frac{1}{3}\left(e^{\left(\mathrm{abs}(x)-9\right)}\right)$	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.

2. Output Constraint Setup

2.1 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 -	Medium	Good	Good

Random			
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

2.2 Fit Output Bound in Scipy Optimizers

2.2.1 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.

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

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 1.1 Nonlinear Constraints Mathmatical Expression

2.2.2 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).