Optimization for fitting the best curve using Differential Evolution

Illia Tsiporenko, Dmytro Shvetsov

Summary

Background

Differential Evolution (DE) is a population-based optimization algorithm that is widely used for solving non-linear and non-differentiable problems. DE is a stochastic algorithm that can be used for global optimization, and it has been successfully applied to a wide range of optimization problems.

One such problem is the curve fitting problem, where the goal is to find the coefficients of a polynomial that best fit a set of data. The curve fitting problem is commonly encountered in various fields, such as physics, engineering, and finance. The polynomial coefficients are typically found by minimizing the mean errors between the polynomial and the data points. This is a non-linear optimization problem, and can be challenging to solve using traditional optimization methods. Our project aims to use DE to solve this problem and find the best polynomial fit for a given set of data points.

Methods

To implement DE in our project, we have used Python as the programming language. The algorithm was implemented from scratch, which allowed us to have full control over the implementation and fine-tune it to our specific problem. The polynomial coefficients are found by minimizing the mean errors between the polynomial and the data points. This is done by iteratively updating the coefficients according to the DE algorithm.

The DE algorithm is based on the concept of a population of candidate solutions, also known as individuals. In each generation, the algorithm creates new individuals by combining the traits of existing individuals in the population. The combination of traits is done through a process known as crossover and mutation. The way that the mutation was implemented in our project is fairly simple but effective - pick three agents from current population, one of them is the target vector, then apply the formula to mutate it

$$mutated = target + F \cdot (agent_1 - agent_2)$$

Crossover combines the traits of two individuals to create new offspring, while mutation makes small random changes to the traits of an individual.

The important parameter that defines the crossover probability is CP. For each component of the mutated vector the random probability is generated. Then the current algorithm is applied - if current component probability is less then CP, the mutated vector component is assign to the offspring. Else, the parent component is assign to the offspring.

$$\textit{if component probability} < CP\textit{:}$$
$$\textit{offspring component} = \textit{mutated vector component}$$
$$\textit{if component probability} > CP\textit{:}$$
$$\textit{offspring component} = \textit{parent vector component}$$

The new best individuals are selected to continue to the next generation. The selection is done by simply comparing the mean errors of the agents, if the parents ME is less then it‘s offspring, the parent survives or vice versa.

$$\textit{if parent ME} < \textit{offspring ME:}$$
$$\textit{return parent vector}$$
$$\textit{if parent ME} > \textit{offspring ME:}$$
$$\textit{return offspring vector}$$

The process is repeated until a stopping criterion is met, such as a maximum number of generations.(Figure1)

In addition to the DE implementation, we also used PyQt to create a user interface for the project.(Figure 2) The user interface allows users to easily experiment with different hyper-parameters such as population size, crossover rate, and mutation rate. This allows users to observe how changes in these parameters affect the fit of the polynomial to the data. The user interface also provides visualization tools to help users understand the results of the algorithm. These visualization tools can help users understand the optimization process, such as how the polynomial coefficients change over time, and the final polynomial fit.

One of the key advantages of DE is its ability to handle a large number of variables and constraints. DE can be easily applied to polynomial curves of any degree, and is not limited to a specific type of polynomial. Additionally, DE is robust to the presence of local optima, which can be a common problem in other optimization algorithms.

Results

The results of our project depend on the specific data set and the specific values of the hyper-parameters used. However, in general, DE is expected to find a set of polynomial coefficients that provide a good fit to the data. The user interface allows users to easily experiment with different hyper-parameter settings and see how they affect the fit of the polynomial. The visualization tools also make it easy for users to understand the results of the algorithm and how it is performing.

Conclusion

In conclusion, our project has successfully implemented the Differential Evolution algorithm for fitting polynomial curves to data. The implementation of DE from scratch and the PyQt user interface make it easy for users to experiment with different hyper-parameter settings and understand the results. This project can be useful in various scientific fields.

It can also be used as a starting point for other curve fitting problems and optimization tasks. The ability to tweak the algorithm to the specific problem will be beneficial in many real-world applications. The implementation of the algorithm from scratch and the user interface are the key features of this project, it allows users to understand the optimization process and make adjustments accordingly. DE's robustness to local optima and ability to handle a large number of variables make it a powerful tool for solving non-linear optimization problems.

Overall, this project demonstrates the effectiveness of DE in solving the curve fitting problem and its potential for use in other optimization tasks.

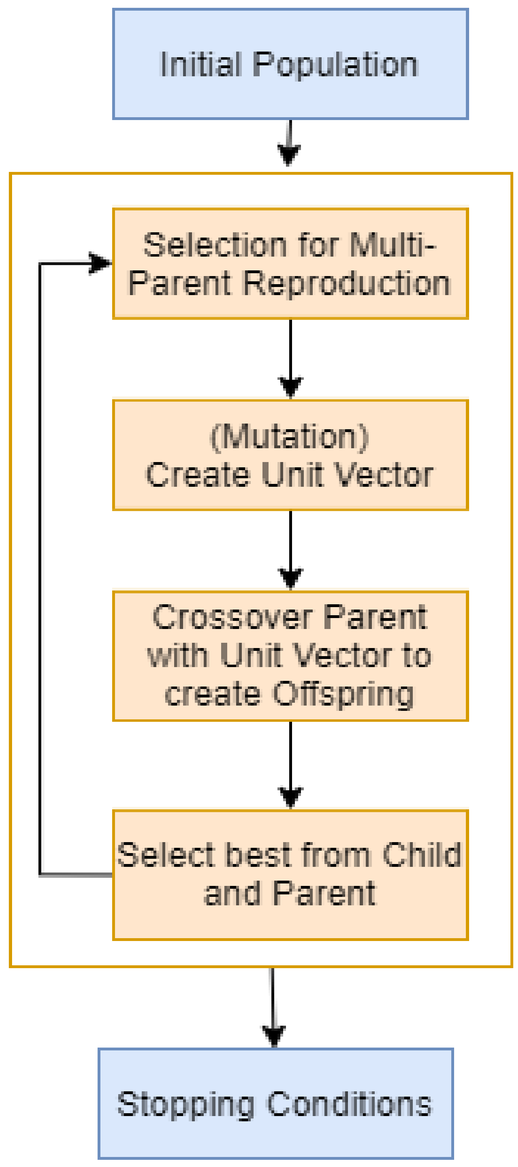

Figure 1: Differential algorithm.

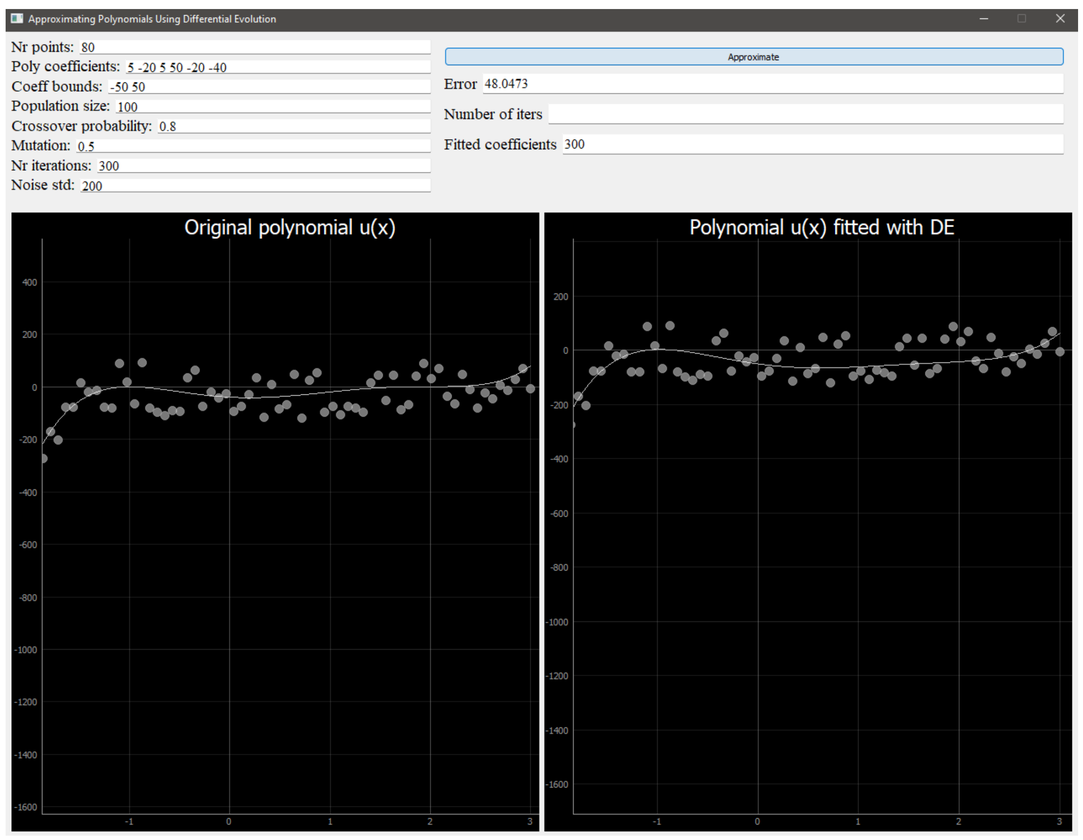

Figure 2: Program execution example.

References

[1] Mishra, SK; Performance of Differential Evolution Method in Least Squares Fitting of Some Typical Nonlinear Curves. North-Eastern Hill University, Shillong (India).29 August 2007.Available from: https://mpra.ub.uni-muenchen.de/4656/1/MPRA_paper_4656.pdf

[2] Matts Bjorck; Fitting with differential evolution: an introduction and evaluation. Paul Scherrer Institut, Villigen, Switzerland. 2011 International Union of Crystallography Printed in Singapore. Available from: https://www.researchgate.net/publication/263492029_Fitting_with_differential_evolution_An_introduction_and_evaluation

[3] C. Venkateswaran, M. Ramachandran, M. Amudha, T. Vennila, M. Manjula; A Review on Differential Evolution Optimization Techniques.Available from: https://www.researchgate.net/publication/358090789_A_Review_on_Differential_Evolution_Optimization_Techniques/link/61efe4fac5e3103375bd6f44/download

[4] Hongwei Liu, Xiang Li, Wenyin Gong; Rethinking the differential evolution algorithm. Available from: https://www.researchgate.net/publication/339672733_Rethinking_the_differential_evolution_algorithm