DiscoverDCP: A Data-Driven Approach for Construction of Disciplined Convex Programs via Symbolic Regression

Sveinung Myhre
University of California, Berkeley
Berkeley, CA, 94720
s.myhre@berkeley.edu

Abstract

DiscoverDCP is proposed as a data-driven framework that integrates symbolic regression with the rule sets of Disciplined Convex Programming (DCP) to perform system identification under the non-trivial but not uncommon assumption that the underlying system is convex. By enforcing that all discovered candidate model expressions adhere to DCP rules, it is ensured that the output expressions are guaranteed to be convex, circumventing the process of convexity verification. The advantage of this approach is that the discovered model expressions are permitted to exhibit more relaxed functional forms than typically used fixed parameter convex expressions (e.g. quadratic function with non-negative hessian). The proposed method is intended to produce interpretable, verifiable, and flexible convex surrogates models for various scientific applications.

1 Introduction

Convex optimization plays a central role in numerous applications including engineering, control problems, statistics, biology, finance, machine learning, and other quantitative disciplines, largely due to its strong theoretical guarantees for methods with tractable solutions [1]. Ensuring that an optimization problem is convex relies on having a convex objective and convex constraints on a convex domain. Traditionally, practitioners need to rely on models that are known to be convex, such as linear or quadratic functions with positive definite Hessians, typically of the form (1). A downside to restricting models to such simple families of functions is that it can limit the model expressiveness and ability to accurately capture non-linearities and complex system dynamics.

$$y = x^\top Ax + b^\top x + c. \tag{1}$$

The *Disciplined Convex Programming* (DCP) framework, introduced by [2], provides a set of compositional rules that guarantee convexity. If each building block of a function (e.g., sums, compositions with convex and monotone functions, pointwise maxima) respects certain rules, the resulting expression is also convex. It is noteworthy to mention that the set of DCP adhering convex functions is smaller than the set of all convex functions. While determining if a function is convex is in general a computationally intractable task, determining if a function is DCP is straightforward. There exist tools that can algorithmically determine this like [3].

Despite the promise of DCP, discovering suitable convex expressions from raw data remains challenging. Existing practice typically involves linearization or fitting parametric convex families, which may not capture the system accurately.

In parallel has symbolic regression emerged as a powerful technique for discovering interpretable analytic expressions directly from data. An example of one such tool is *PySR* [4]. Standard symbolic

regression algorithms do not inherently ensure that discovered expressions are convex. However, symbolic regression tools like *PySR* allow custom sets of operations, giving flexibility in the search space of the candidate convex expressions that can be constructed.

This paper proposes *DiscoverDCP*, a framework that leverages symbolic regression under the structural constraints imposed by DCP rulesets. By restricting the search space of symbolic regression to operations and compositions that preserve convexity, it becomes possible to algorithmically learn convex models from data. This approach yields interpretable mathematical expressions which are guaranteed to be convex.

2 Related Work

Work on convex modeling and optimization is extensive. Classic references include the textbook by Boyd and Vandenberghe [1], which provides a comprehensive treatment of convex sets, functions, and optimization problems. The notion of DCP was introduced to ensure that convexity verification is straightforward, laying the groundwork for other convex modeling frameworks and toolboxes such as CVXPY [5].

Symbolic regression has had recent developments with modern and highly performant tools like PySR [4] enabling more efficient and scalable expression search. While symbolic regression traditionally focuses on accuracy and parsimony, recent research suggests incorporating *shape constraints*, such as ensuring positivity, monotonicity or convexity [6,7].

This approach can also be related to attempts at enforcing structure in learned neural net models such as input convex neural architectures [8]. However, deep neural networks are difficult to interpret, whereas the proposed approach aims to enable clear mathematical interpretability and convex verification capabilities.

3 Method

The proposed methodology, *DiscoverDCP*, integrates symbolic regression with the rules of Disciplined Convex Programming:

1. Convexity-Preserving Operations. DCP provides a set of operations that preserve convexity. Examples include nonnegative weighted sums of convex functions, pointwise maximum, and composition with affine transformations or monotone convex functions. By restricting the symbolic search space to these building blocks and their allowed compositions, it is ensured that all candidate expressions are guaranteed to be convex [2]. Further DCP compliant rules can be found in [2].

The experiments conducted incorporated the binary operations "+", "*", "max" while ensuring only non-negative sums, as well as the unary operators "exp" and "square".

2. Symbolic Regression with Constrained Search Space. A symbolic regression engine is employed, and for this paper it being PySR [4]. The symbolic regression tool represents mathematical expressions as a composition of operations stored in a tree-structure and evolves these expressions using genetic programming. Here, the set of available operations to the engine is restricted to DCP-compliant operations. The search begins from a base set of atomic convex functions (e.g., affine terms, norms, exponentials in nonnegative domains, and maximum operations). At each evolutionary step, candidate solutions are mutated or combined, with only those expressions that remain DCP-compliant retained. A *complexity measure* is used to guide the search towards simpler, interpretable forms. This *complexity measure* can be customized but for our purposes uses the default measure set by PySR which is proportional to the number of nodes in such an expression tree [4].

4 Experiments

Experiments were conducted by creating synthetic datasets where the ground-truth functions are known, allowing controlled testing of whether the proposed method can recover or approximate these target functions to a satisfactory degree.

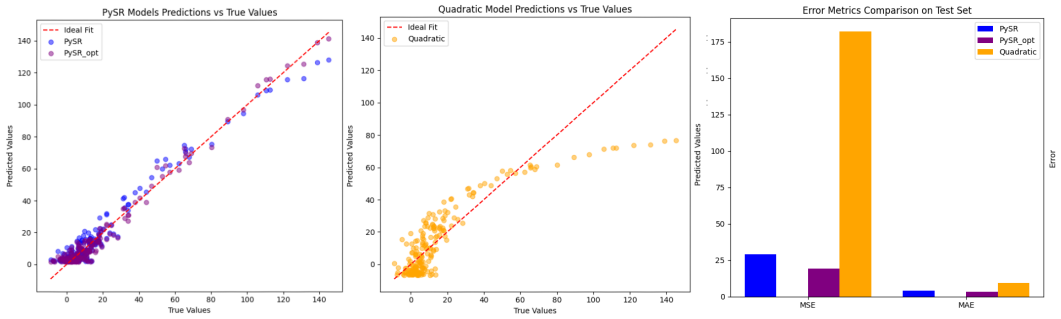

Figure 1: Ideal fit indicates the ground truth of values from the hidden model, which is $y = \exp\left(x + \max(x, -5x) + x^2\right) + 4x$. PySR indicates objective values assumed by the discovered function with *complexity score* of 8 with the equation $exp(3.2308085 * x) + exp(-4.8683963 * x)$. PySR_opt indicates objective values assumed by the discovered function with complexity of 49 (expression not shown). Quadratic indicates objective values assumed by the quadratic function Eq. (1) with the equation $61.5890x^2 - 20.3866x - 4.9182$.

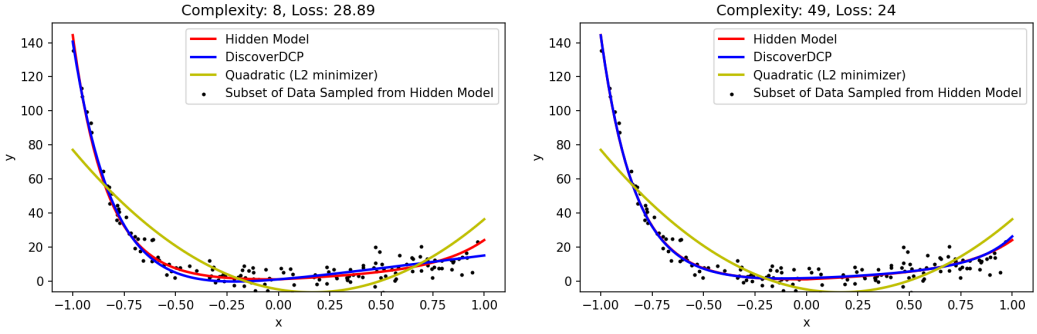

Figure 2: The figure indicates discovered approximations of the hidden model at two different "complexity scores". It is showing the same equations as in Figure 1. Hidden model: $y = \exp\left(x + \max(x, -5x) + x^2\right) + 4x$ Quadratic: $y = 61.5890x^2 - 20.3866x - 4.9182$ Left plot DiscoverDCP: $y = exp(3.2308085 * x) + exp(-4.8683963 * x)$, Right plot DiscoverDCP: $y = ((\exp(-0.33)+((-1.12+\exp(-0.89))+((((\exp(x^2-1.11)+x^2)+(0.46)^2)+\max(-0.57, x+(-0.10+x))) + ((x+x)+\exp(-1.22+(-6.06\cdot x))) + \exp((x\cdot(-1.70))^2)))) + x) + x^2$

Experiments were conducted on a series of "hidden models" by sampling data points x_i evenly from a uniform random distribution over a specified closed domain (in this instance a polytope). Using polytopes given by $Ax \leq b$ is practical since they are already commonplace in constructing convex programs, making it straightforward to create DCPs. Then Gaussian white noise was added to the data points sampled from the "hidden equation".

To establish a benchmark to compare *DiscoverDCP* to, conventional approaches that use fixed parametric expressions (e.g., quadratic functions of the form Eq. (1)) were implemented. The quadratic expressions were fit to the data (minimizing L2 loss) with the constraint that the matrix A is PSD (Positive Semidefinite) by using CVXPY [5].

Figure 1 and 2 shows an experiment comparing a model identified by *DiscoverDCP* at different complexity levels and quadratic approximation to the true underlying hidden model. One can notice the equations discovered by *DiscoverDCP*. These figures were produced by sampling 1000 points from a uniform random distribution over the domain $[-1, 1]$ of $x \in \mathbf{R}$.

The author would like to add more examples showing the full extent of applications this method can be applied to, however the author believes that Figures 1 and 2 effectively captures the essence of the proposed method. Interested readers are encouraged to experiment with the code, which has been tested on higher-dimensional systems, under noisy conditions and tested with over 20 different convex and non-convex 'hidden models'. A link to the code is found in section 6.

5 Discussion

It is worthy to note that even "simple" quadratic expressions like Eq. (1) inhibits significant *complexity scores* (as defined by number of nodes in an expression tree) as the number of features of the dataset increases.

Since A is assumed PSD the matrix A has at least $\frac{n(n+1)}{2}$ unique parameters, b has n and the constant term c is 1.

$$\#\textit{unique parameters in PSD quadratic} = \frac{n(n+1)}{2} + n + 1. \tag{2}$$

A proposed benchmark to compare complexity of expressions found by PySR and complexity of quadratic expressions would be Eq. (2). As the number of features n in the dataset increases one should expect at least the complexity score given by eq. (2) to achieve similar accuracy to the quadratic models.

When performing symbolic regression, it is crucial to balance the increased model accuracy with the complexity of the model expression. The optimal balance between these factors will vary depending on the specific application and its requirements. Instead of evaluating the complexity of resulting models in absolute terms, it may be advantageous to use the complexity score of quadratic expressions as a comparative benchmark. This approach allows for a more meaningful assessment of whether more complex models provide improvements over "simple" quadratic models.

6 Conclusion

This paper has proposed *DiscoverDCP*, a data-driven framework for automatically constructing convex models by fitting an analytic expression to a given data set while adhering to the compositional rules of DCP. This approach can open up new avenues for interpretable, verifiable, and flexible convex modeling, preserving the applicability of convex optimization techniques to real-world systems. The method was demonstrated to match or improve model accuracy compared to commonly used convex quadratic models, while providing the end user control in balancing model complexity and accuracy.

Future work includes detailed experiments under diverse and noisy datasets, exploring extensions of the method to higher-dimensional input spaces and incorporating more classes of DCP-compliant operations.

The code used for experiments is available at: `https://github.com/svemyh/DiscoverDCP`.

References

[1] Boyd, S., and Vandenberghe, L. "Convex Optimization." Cambridge University Press, 2004.

[2] Grant, M., Boyd, S. and Ye, Y. "Disciplined Convex Programming" Springer, 2006.

[3] Diamond, S., Chu, E., & Boyd, S. (2013) "DCP Analyzer." Available at: `https://dcp.stanford.edu/analyzer`

[4] Cranmer, M. (2023) "Interpretable Machine Learning for Science with PySR and SymbolicRegression.jl." arXiv:2305.01582 [astro-ph.IM], `https://arxiv.org/abs/2305.01582`

[5] CVXPY: A Python-embedded modeling language for convex optimization problems. `https://www.cvxpy.org/`.

[6] Błądek, I., and Krawiec, K. "Solving Symbolic Regression Problems with Formal Constraints." in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 977–984, 2019. `https://doi.org/10.1145/3321707.3321743`

[7] Aubin-Frankowski, P.-C., and Szabo, Z. "Hard Shape-Constrained Kernel Machines." in *Advances in Neural Information Processing Systems*, NeurIPS-2020. `https://doi.org/10.48550/arXiv.2005.12636`

[8] Amos, B., Xu, L., and Kolter, J.Z. "Input Convex Neural Networks." in *Proceedings of the 34th International Conference on Machine Learning*, Vol. 70, pp. 146–155, 2017. `https://dl.acm.org/doi/10.5555/3305381.3305397`