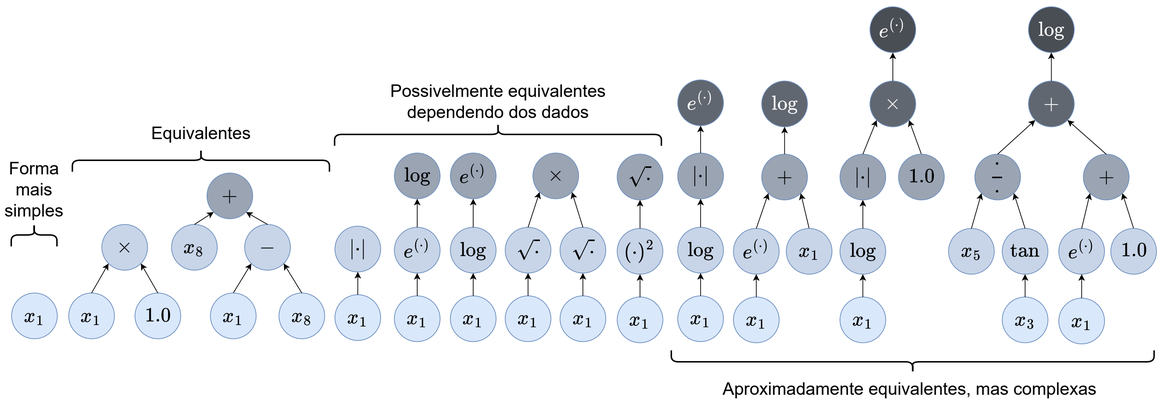
Workshop 2024 da PosComp/UFABC

Simplificação Inexata de expressões de Regressão Simbólica com hashing sensível à localidade

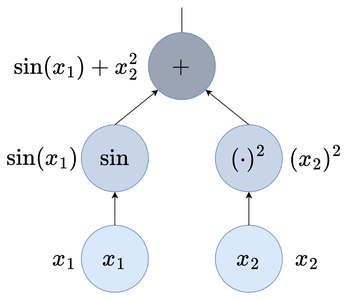
Guilherme Seidyo Imai Aldeia (UFABC), Fabrício Olivetti de França (UFABC), William G. La Cava (BCH)



Introdução

A regressão simbólica busca pela equação que melhor se ajusta à uma base de dados. O espaço de busca é enorme e contém equações com saídas equivalentes ou aproximadamente equivalentes.

A técnica é utilizada visando resultados interpretáveis e o mais simples possíveis, mas é custoso implementar um conjunto de regras de simplificação, e algumas simplificações são dependentes dos dados.

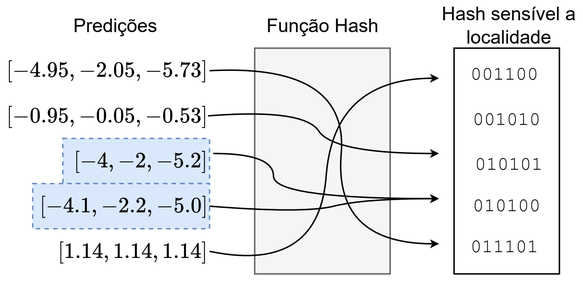


Esse trabalho propõe uma maneira simples e eficiente de simplificar equações durante a busca da regressão simbólica, em particular com o uso de Hash Sensível a Localidade (LSH) para mapear partes de equações aproximadamente equivalentes em coleções de substituições para simplificação.

Métodos

O Hash Sensível a Localidade encontra vizinhos aproximados dentro de um espaço de alta dimensionalidade, permitindo armazenamento e recuperação rápida de informações.

Isso naturalmente permite mapear expressões aproximadamente equivalentes visitadas durante a busca.



Ao armazenar as expressões ordenadas por tamanho, podemos utilizar a primeira delas para substituir todas as suas formas equivalentes.

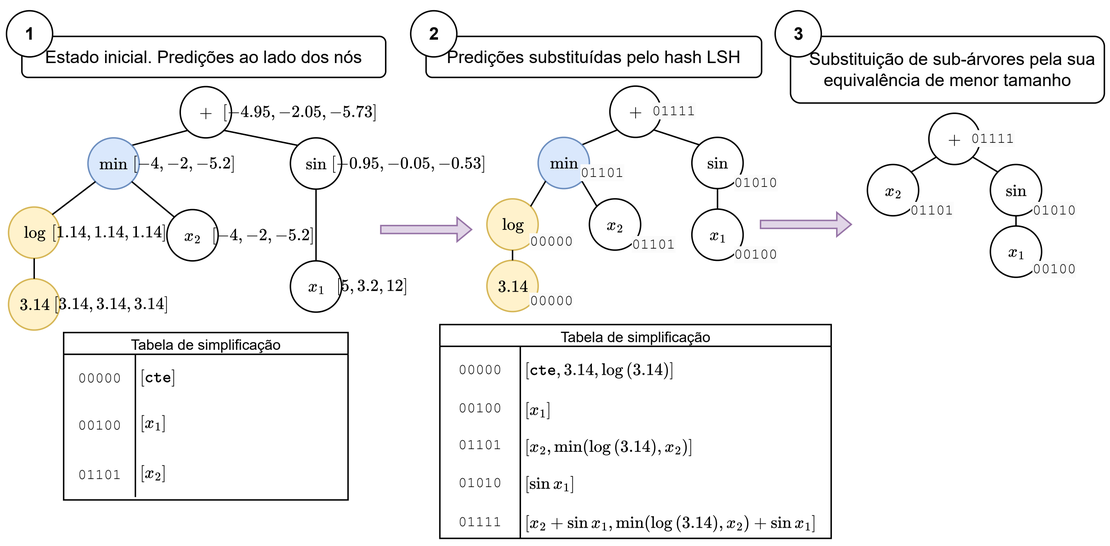


Tabela de simplificação

00000	[cte]
00100	$[x_1]$
01101	$[x_2]$

Tabela de simplificação

00000	[cte, 3.14, log (3.14)]
00100	$[x_1]$
01101	$[x_2, \min(\log(3.14), x_2)]$
01010	$[\sin x_1]$
01111	$[x_2 + \sin x_1, \min(\log(3.14), x_2) + \sin x_1]$

Experimentos

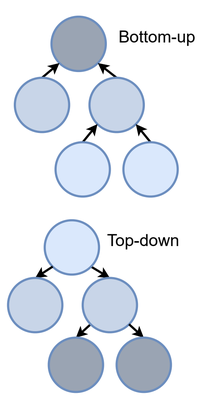
6 bases de dados de baixa dimensionalidade bastante utilizadas em benchmarks de regressão. 10 execuções para cada base. Parâmetros fixados e apenas a ordem de percorrer as expressões e o limiar de simplificação foram variados.

Os resultados focam em observar o tamanho, o erro, e a complexidade dos modelos finais.

Dataset	# samples	# features
Airfoil	1503	5
Concrete	1030	8
Energy Cooling	768	8
Energy Heating	768	8
Housing	506	13
Yacht	308	6

$$C(n) = c_n * (\sum_{a=1}^{k} C(a))$$

Parameter	Value
pop size (S)	80
max gen (G)	200
max depth ($\max_d$)	7
max size ($\max_s$)	128 (2^7)
tolerance (τ)	$1e-2$
hash_len	256 bits
probabilities	1/5 for each variation operator
objectives	[error (MSE), size (# nodes))]
Function set	$[+, -, *, \div, \lvert\cdot\rvert, \cos^{-1}, \sin^{-1}, \tan^{-1}, \cos, \sin, \tan, e^{(\cdot)}$, min, max, log, $\log(1+\cdot)$, $\exp(1+\cdot)$, $\sqrt{\lvert\cdot\rvert}$, $(\cdot)^2]$



Resultados

A maioria das simplificações são concentradas nas primeiras 10 hashs criadas, correspondendo a mais de 80% de todas as equivalências.

O erro de predição entre a expressão original é no máximo o valor do limiar definido, e a direção das predições varia dentro de um intervalo pequeno, sendo dependente do problema.

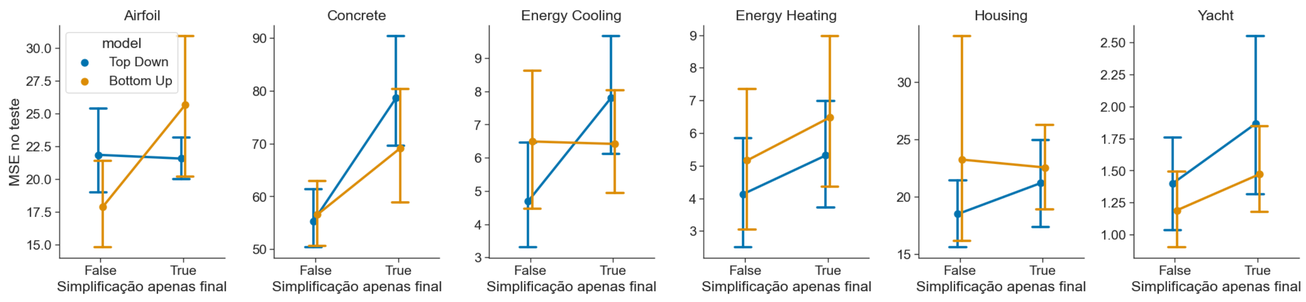
Melhorias significativas acontecem na complexidade e erro médio quadrático, mas pouca diferença é observada no tamanho final.

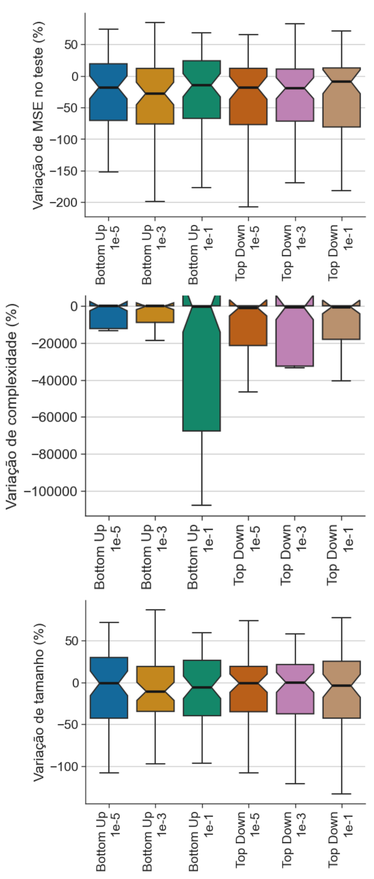
Simplificar ao longo da execução mostra um desempenho superior do que simplificar apenas as últimas expressões.

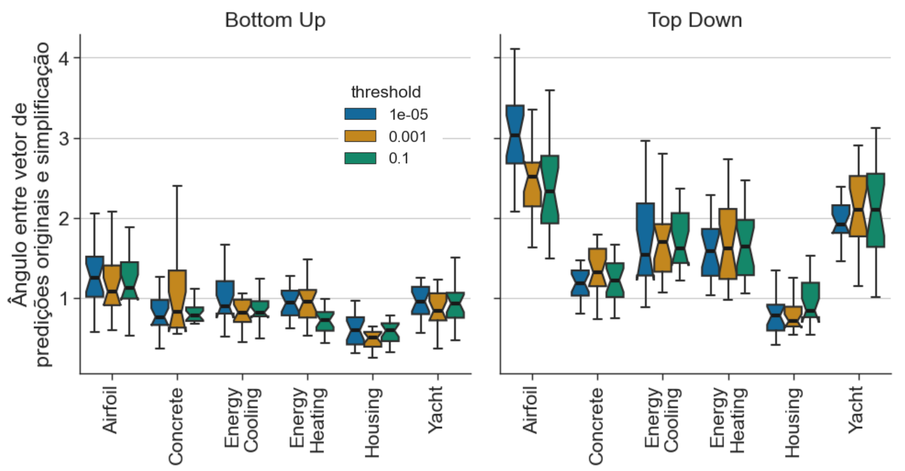
```
1010101111001001...
 - x_2
 - absolute(x_2)
 - minimum(x_2, x_2)
 - minimum(x_2, x_6)
 - minimum(x_2, x_5)
 - add(0.0, x_2)
 - log(exp(x_2))
 - sqrtabs(square(x_2))
 - maximum(-523.249, x_2)
```

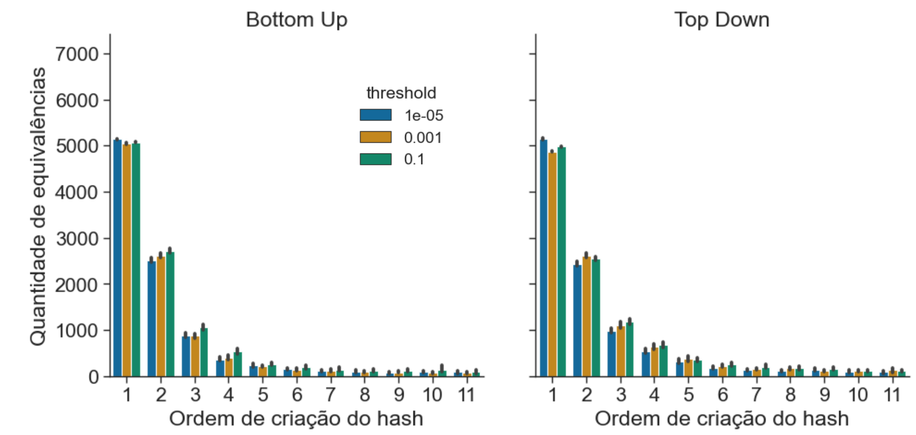
```
1010110111001101...
 - square(x_5)
 - multiply(x_5, x_5)
 - absolute(square(x_5))
 - maximum(square(x_5), x_0)
 - maximum(add(-15.455, x_1), square(x_5))
```

```
1110001011011111...
 - multiply(x_1, x_7, x_0, x_4)
 - multiply(x_0, x_4, x_1, x_7)
 - multiply(x_1, x_0, multiply(x_4, x_7))
```









Conclusões

O método de simplificação encontra expressões com menor erro e menor complexidade, e funciona sem a necessidade de definir regras de simplificação.

Dado que a simplificação é inexata, ela pode realizar substituições que afetam a predição das expressões, ainda que de maneira mínima.

A eficiência do método é dependente das colisões geradas pelo hash, e o processo de busca consegue efetivamente construir uma coleção de modelos equivalentes.

Referências

[1] Bogdan Burlacu, Lukas Kammerer, Michael Affenzeller, and Gabriel Kronberger. 2021. Hash-Based Tree Similarity and Simplification in Genetic Programming for Symbolic Regression. https://doi.org/10.1007/2F978-3-030-45093-9_44arXiv:2107.10640 [cs]

[2] Omid Jafari, Preeti Maurya, Parth Nagarkar, Khandker Mushfiqul Islam, and Chidambaram Crushev. 2021. A Survey on Locality Sensitive Hashing Algorithms and their Applications. http://arxiv.org/abs/2102.08942 arXiv:2102.08942 [cs]

[3] William La Cava, Patryk Orzechowski, Bogdan Burlacu, Fabricio de França, Marco Virgolin, Ying Jin, Michael Kommenda, and Jason Moore. 2021. Con temporary Symbolic Regression Methods and their Relative Performance. In Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks, J. Vanschoren and S. Yeung (Eds.), Vol. 1. Curran.

[4] Imai Aldeia, Guilherme Seidyo, Fabrício Olivetti De França, and William G. La Cava. "Inexact Simplification of Symbolic Regression Expressions with Locality-sensitive Hashing." Proceedings of the Genetic and Evolutionary Computation Conference. 2024.