UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Faculdade de Ciências e Tecnologia
Câmpus de Presidente Prudente

Aprendizado de máquina científico via regressão simbólica para descoberta de modelos interpretáveis

Daniel Henrique Serezane Pereira

Orientador: Prof. Dr. Cassio Machiaveli Oishi

Coorientador: Dr. Felipe Kesrouani Lemos

Programa: Matemática Aplicada e Computacional

Presidente Prudente, 29 de Junho de 2026

UNIVERSIDADE ESTADUAL PAULISTA

Faculdade de Ciências e Tecnologia de Presidente Prudente

Programa de Pós-Graduação em Matemática Aplicada e Computacional

Aprendizado de máquina científico via regressão simbólica para descoberta de modelos interpretáveis

Daniel Henrique Serezane Pereira

Orientador: Prof. Dr. Cassio Machiaveli Oishi

Coorientador: Dr. Felipe Kesrouani Lemos

Dissertação apresentada ao Programa de Pós-Graduação em Matemática Aplicada e Computacional da UNESP como parte dos requisitos para obtenção do título de Mestre em Matemática Aplicada e Computacional.

Presidente Prudente, 29 de Junho de 2026

P436a

Pereira, Daniel Henrique Serezane

Aprendizado de máquina científico via regressão simbólica para descoberta de modelos interpretáveis / Daniel Henrique Serezane Pereira. -- Presidente Prudente, 2026

105 p.

Dissertação (mestrado) - Universidade Estadual Paulista (UNESP), Faculdade de Ciências e Tecnologia, Presidente Prudente

Orientador: Cassio Machiaveli Oishi

Coorientador: Felipe Kesrouani Lemos

1. Ciência da computação Matemática. 2. Análise de regressão. 3. Programação genética (Computação). 4. Redes neurais (Computação). I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Dados fornecidos pelo autor(a).

CERTIFICADO DE APROVAÇÃO

TÍTULO DA DISSERTAÇÃO: Aprendizado de máquina científico via regressão simbólica para descoberta de modelos interpretáveis

AUTOR: Daniel Henrique Serezane Pereira
ORIENTADOR: Cassio Machiaveli Oishi

Aprovado como parte das exigências para obtenção do Título de Mestre em Matemática Aplicada e Computacional, pela Comissão Examinadora

Prof. Dr. Cassio Machiaveli Oishi (Participação Virtual)
Departamento de Matemática e Computação / Faculdade de Ciências e Tecnologia - FCT/UNESP

gov.br
Documento assinado digitalmente
CASSIO MACHIAVELI OISHI
Data: 30/07/2026 14:58:26-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Wallace Correa de Oliveira Casaca (Participação Virtual)
Departamento de Ciências de Computação e Estatística / Instituto de Biociências, Letras e Ciências Exatas - IBILCE/UNESP

Prof. Dr. Afonso Paiva Neto (Participação Virtual)
Departamento de Matemática Aplicada e Estatística / Instituto de Ciências Matemáticas e de Computação - ICMC/USP

Presidente Prudente, 29 de Junho de 2026

Agradecimentos

Agradeço aos meus orientadores Cassio Oishi e Felipe Lemos pelo constante apoio na realização do projeto, e pelas inúmeras oportunidades dentro e fora dele.

Agradeço ao colega Gabriel Yoshikazu Cortez Oukawa, mestre em engenharia ambiental pela UTFPR, pela enorme ajuda no levantamento e validação dos conjuntos de dados ambientais.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Resumo

Ao reconhecer o aprendizado de máquina como uma ferramenta central para a computação científica moderna, surge o aprendizado de máquina científico, com o principal objetivo de maximizar a contribuição destes modelos para o avanço do estado da arte em diversos domínios da ciência. Dentre as diretrizes de pesquisa propostas pelo conceito de aprendizado de máquina científico, está a produção de modelos interpretáveis, robustos e capazes de lidar com cenários de alta dimensionalidade, uma realidade decorrente da crescente abundância de dados, graças à revolução digital, e à necessidade de produzir conhecimento a partir deles. Este trabalho visa explorar a regressão simbólica como meio de produção de modelos interpretáveis, identificando seu estado da arte, suas limitações na produção de modelos escaláveis e resistentes a ruídos, formas de avaliação de métodos e perspectivas de evolução da área. Além do levantamento e estudo da literatura, também compõe o texto uma análise crítica do SRBench, principal benchmark para avaliação de métodos de regressão simbólica, acompanhada de uma proposta de evolução do mesmo por meio de novos conjuntos de dados voltados à predição da concentração de $MP_{2.5}$ a partir de variáveis meteorológicas e temporais, um problema real, complexo e compreendido dentro de um cenário ruidoso e de alta dimensão, dada a potencial influência de fatores imprevisíveis e a elevada quantidade de amostras e variáveis envolvidas.

Palavras-Chave: *Ciência da computação Matemática, Análise de regressão, Programação genética (Computação), Redes neurais (Computação).*

Abstract

By recognizing machine learning as a central tool for modern scientific computing, scientific machine learning emerges, with the main goal of maximizing the contribution of these models to advancing the state of the art across various domains of science. Among the research guidelines proposed by the concept of scientific machine learning is the production of interpretable, robust models capable of handling high-dimensional scenarios, a reality stemming from the growing abundance of data, thanks to the digital revolution, and the need to produce knowledge from it. This work aims to explore symbolic regression as a means of producing interpretable models, identifying its state of the art, its limitations in producing scalable and noise-resistant models, methods of evaluation, and prospects for the field's evolution. In addition to the survey and study of the literature, the text also includes a critical analysis of SRBench, the main benchmark for evaluating symbolic regression methods, accompanied by a proposal for its evolution through new datasets aimed at predicting $PM_{2.5}$ concentration from meteorological and temporal variables, a real, complex problem understood within a noisy, high-dimensional scenario, given the potential influence of unpredictable factors and the large number of samples and variables involved.

Keywords: *Computer science Mathematics, Regression analysis, Evolutionary programing (Computer science), Neural networks (Computer science).*

Lista de Figuras

Lista de Tabelas

Lista de Siglas

ML	aprendizado de máquina (*machine learning*)
SciML	aprendizado de máquina científico (*scientific machine learning*)
PRD	diretriz prioritária de pesquisa (*Priority Research Directions*)
SHAP	SHapley Additive exPlanations
IA	inteligência artificial
SR	regressão simbólica (*symbolic regression*)
GP	programação genética (*genetic programming*)
AE	algoritmo evolutivo
AG	algoritmo genético
RL	aprendizado por reforço (*reinforcement learning*)
BFGS	Broyden–Fletcher–Goldfarb–Shanno
MSE	erro quadrático médio (*mean squared error*)
PM	material particulado (*particulate matter*)
NLP	processamento de linguagem natural (*natural language processing*)
RNN	rede neural recorrente (*recurrent neural network*)
LSTM	*long short-term memory*
GRU	*gated recurrent unit*
FNN	rede neural *feedforward* (*feedforward neural network*)
LLM	*large language model*
PINN	*physics-informed neural network*
AUC	área sob a curva (*area under the curve*)
GAM	modelo aditivo generalizado (*generalized additive model*)
xAI	inteligência artificial explicável (*explainable artificial intelligence*)
BLH	altura da camada limite (*boundary-layer height*)

Sumário

Capítulo

1

Introdução

O uso de técnicas de aprendizado de máquina para resolução de problemas sofreu uma expansão sem precedentes com o surgimento, em 2006, do framework CUDA, desenvolvido pela NVIDIA para os seus dispositivos de processamento gráfico. O framework, juntamente com as bibliotecas de programação auxiliares também publicadas pela empresa, permitiam com um alto grau de controle o uso das GPUs para propósito geral, não relacionado com renderização gráfica [13].

Foi viabilizada, assim, a implementação otimizada de redes neurais e de seus componentes, como o algoritmo backpropagation, simbolizada pela criação da AlexNet em 2012 [14]. A alta capacidade de generalização [6] dos modelos desta categoria, aliada a uma crescente abundância de dados graças à revolução digital [15], desencadeou sua aplicação em uma série de áreas, inclusive na ciência, com a produção de modelos muito acurados.

Todavia, fatores limitantes no uso do aprendizado de máquina para avanço de linhas científicas podem ser identificados, especialmente: (1) sua capacidade de generalização permite que, com um volume moderado de dados suficientemente limpos, modelos acurados sejam produzidos mesmo com pouco ou nenhum conhecimento a respeito do fenômeno modelado, desta forma limitando muito o avanço na compreensão do mesmo, pois há precisão, mas não há clareza; e (2) a altíssima complexidade interna destes modelos os torna praticamente incompreensíveis sob um olhar humano, sendo assim, a confiabilidade de vários resultados é questionada à medida que a avaliação do modelo por um especialista do domínio de conhecimento de origem dos dados limita-se à observação da performance dos modelos e recolhimento de métricas.

De mesmo modo, resultados deste teor pouco contribuem para o avanço científico, pois pouco auxiliam com conhecimento adicional sobre o fenômeno estudado. Este último ponto leva ao uso do termo *black box* (caixa preta) para referir-se aos modelos de aprendizado de máquina cujo funcionamento interno é obscuro, e pode ser aplicado até mesmo a alguns modelos fora da categoria das redes neurais, tais como o *support vector machine* (SVM) não-linear, *random forest*, e processo gaussiano baseado em uma abordagem Bayesiana. Este cenário encontra-se ilustrado na Figura 1.1, baseada no trabalho de Brunton & Kutz [1], que compara uma rede neural feedforward a um modelo simbólico transparente obtido via SINDy [16], na tarefa de descobrir o sistema de Lorenz por meio de uma abordagem puramente orientada a dados.

Deve-se ressaltar, contudo, que o algoritmo SINDy foi criado especificamente para aprender sistemas dinâmicos como o do caso, enquanto a arquitetura de rede neural utilizada é completamente agnóstica e, inclusive, poderia ser aplicada em um conjunto de dados totalmente diferente e ainda produzir resultados satisfatórios.

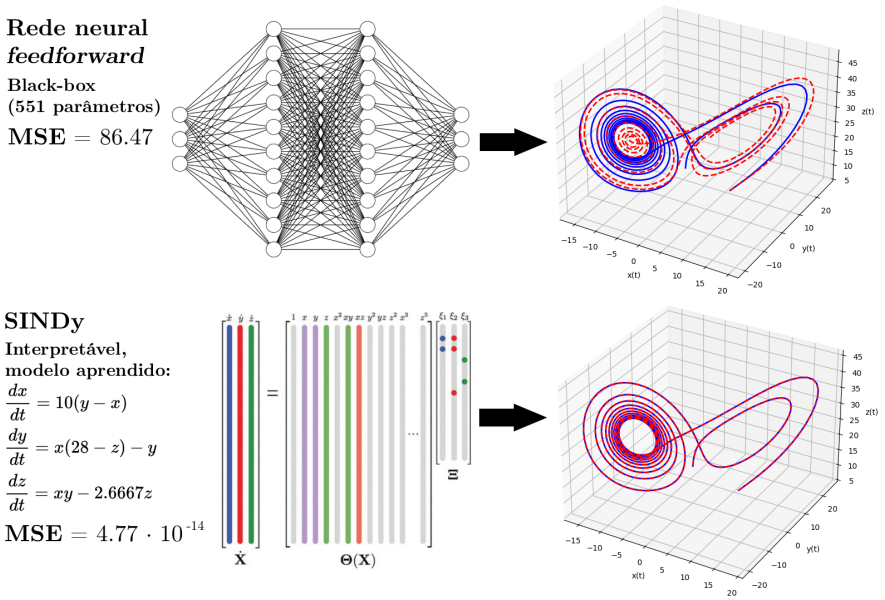

Figura 1.1: Ilustração de um experimento proposto em [1], onde compara-se uma rede neural *feedforward* ao método SINDy no mesmo objetivo de aprender o sistema de Lorenz. Adaptada de [1].

O termo "aprendizado de máquina científico" (SciML, de *scientific machine learning*) tem origem em um relatório técnico [15] do Programa de Computação Científica Avançada (ASCR, de *Advanced Scientific Computing Research*) do Departamento de Energia dos Estados Unidos.

O relatório, publicado em 2019, é resultado de um *workshop* realizado em 2018, unindo diversos pesquisadores com o objetivo de identificar necessidades básicas de pesquisa para melhoria do uso do aprendizado de máquina para computação científica avançada.

A iniciativa vem do reconhecimento da enorme capacidade dos modelos de inteligência artificial em avançar o estado da arte em diversos domínios do conhecimento, e visa assim maximizar este impacto.

Em suma, pode-se compreender o aprendizado de máquina científico como um movimento para melhoria do uso de aprendizado de máquina para contribuições científicas tangíveis ou, ainda, como uma combinação do aprendizado de máquina com a computação científica, em especial visando resultados mais transparentes e precisos.

Foram definidas assim 6 diretrizes prioritárias de pesquisa (PRDs, de *Priority Research Directions*), distintas entre fundamentos, desafios a serem enfrentados na elaboração de modelos para SciML, e oportunidades, linhas de aplicação nas quais o SciML tem grande possibilidade de impacto, ambas dadas a seguir na Tabela 1.1.

Tabela 1.1: Sumarização dos fundamentos e oportunidades do aprendizado de máquina científico.

Fundamentos do aprendizado de máquina científico		
Nro.	**Nome da diretriz**	**Sumarização da diretriz**
1	Consciência do domínio	Utilizar informações prévias sobre o fenômeno estudado, ou do domínio de conhecimento ao qual o mesmo pertence, para aprimorar o modelo de ML em termos de acurácia, transparência e confiabilidade.
2	Interpretabilidade e explicabilidade[1]	Adotar métodos que produzam modelos facilmente compreensíveis por um especialista no domínio dos dados, de forma que o mesmo possa atestar a qualidade e coerência do resultado com base no seu conhecimento. Quantificar com métricas objetivas a diferença entre modelos distintos.
3	Robustez	Os métodos produzidos devem ser estáveis e bem postos, passíveis de verificação, validação e reprodução, e resilientes a perturbações nos dados de entrada e erros computacionais.
Oportunidades para o uso do aprendizado de máquina científico		
4	Aplicações intensivas em dados	Aproveitar a flexibilidade dos métodos de aprendizado de máquina para atacar problemas com dados ruidosos, de alta dimensionalidade ou cuja saída esperada é mal-supervisionada. Também envolve abordagens estatísticas para melhoria da compreensão científica de dados.
5	Modelagem e simulação aprimoradas por ML	Combinar abordagens numéricas tradicionais com aprendizado de máquina para simular e prever cenários complexos, especialmente aqueles com mudanças drásticas de regime, em escala e comportamento.
6	Automação inteligente e apoio à tomada de decisão	Tornar a avaliação de cenários complexos e de alta dimensionalidade factível; produzir resultados validáveis e quantificar confiança e incertezas; permitir um balanço entre decisões automáticas e humanas.

Na literatura recente de SciML, foram feitos grandes avanços nas PRDs 1-5 com o uso de *physics-informed neural networks* (PINNs), como nos trabalhos [17] e [18], onde conhecimentos físicos prévios a respeito dos fenômenos (PRD 1) analisados foram utilizados para melhorar a acurácia de redes neurais através, em principal, da incorporação de equações diferenciais na função loss utilizada em treinamento.

Em particular, o trabalho [18] trata de uma aplicação em cardiologia e hemodinâmica, cenários complexos devido a diversos fatores como a dificuldade de sensoriamento (por tratar-se de um fenômeno interno do corpo humano) e mudanças severas de regime (PRDs

[1]No relatório técnico original, a explicabilidade é dada como um subtópico da interpretabilidade, em decorrência de alguns autores tratarem os termos como sinônimos ou correlatos. Como neste texto estas noções são distintas, optou-se por descrever esta diretriz com os tópicos separados.

3, 4 e 5). A abrangência ao PRD 2 vem pela explicabilidade proporcionada ao supervisionar a porção *physics-informed* do modelo, pois um especialista é capaz de verificar se a porção da loss referente às equações diferenciais conhecidas está sendo respeitada, aumentando a confiabilidade do modelo. Adicionalmente, o trabalho [17] também abrange o PRD 2 ao utilizar redes Kolmogorov-Arnold, capazes de destilar o modelo treinado em uma expressão interpretável.

Dados estes desenvolvimentos, o SciML tornou-se sinônimo do PINN, sendo apresentado deste modo inclusive em [18]. Contudo, a literatura, especialmente em [15], mostra que os objetivos da iniciativa vão muito além desta aplicação. Assim sendo, o projeto neste texto relatado encaixa-se ao aprendizado de máquina científico através de uma unificação dos PRDs 2, 3 e 4, como ilustra a Figura 1.2 ao propor o uso do aprendizado de máquina interpretável, através da regressão simbólica, aplicado em problemas reais, por vezes com dados ruidosos, e com um grande volume de amostras e variáveis.

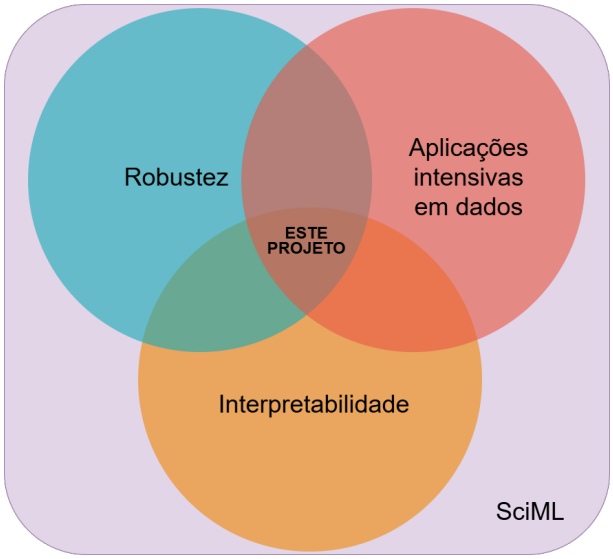

Figura 1.2: Diagrama ilustrativo do contexto deste projeto no aprendizado de máquina científico, visando unir resistência a dados ruidosos (robustez), capacidade de lidar com elevados volumes e bandas de dados (aplicações intensivas em dados), ao passo em que o modelo produzido é facilmente compreensível por um humano (interpretabilidade). Elaboração própria.

Este posicionamento está justificado pelo fato da regressão simbólica compor o estado da arte do aprendizado de máquina interpretável [19, 20].

Em contraste com outros modelos, por exemplo, métodos geradores de árvores de decisão, a SR é uma das, senão a principal alternativa para produção de expressões matemáticas que modelem um determinado fenômeno. Por este motivo, é possível encontrar aplicações bem sucedidas da regressão simbólica em diferentes áreas da ciência, citadas detalhadamente no capítulo 2.

Como também definido no capítulo 2, todavia, a regressão simbólica é uma tarefa, e não um modelo predefinido, e portanto admite uma miríade de soluções. A abordagem tradicional é a programação genética [5, 19–21], porém diversas propostas alternativas foram feitas ao decorrer da evolução da área, com variados graus de sucesso, a depender da aplicação [11, 22].

Por este fator, também foi julgada essencial para composição deste trabalho uma análise detalhada dos principais *benchmarks* para regressão simbólica. Em uma área onde as abordagens são potencialmente muito distintas, e o sucesso varia muito conforme o

conjunto de dados analisados, é importante uma forma de avaliar métodos a partir de uma perspectiva diversa e justa, a fim de fornecer à comunidade científica uma síntese do estado da arte da regressão simbólica [5, 11, 22].

Compuseram então os objetivos do projeto:

1. A compreensão básica do aprendizado de máquina interpretável e seu papel no aprendizado de máquina científico, com enfoque em explorar o diferencial da regressão simbólica em meio a este cenário.

2. O estudo da regressão simbólica como conceito, e das principais abordagens para resolução do problema por ela proposto.

3. Explorar as principais limitações apresentadas pelos métodos conhecidos de regressão simbólica quando aplicados em cenários desafiadores, onde há dados ruidosos, faltantes e/ou de alta dimensionalidade.

4. Compreender detalhadamente o funcionamento dos principais *benchmarks* da área de regressão simbólica, suas formas de avaliação para os métodos e tipos de dados utilizados.

5. Produzir alguma contribuição científica com base na regressão simbólica, seja na forma de uma aplicação de um método em um problema real, seja uma contribuição para evolução da área.

Quanto ao objetivo 5, a principal contribuição trazida por este trabalho se dá na forma de uma análise crítica do SRBench [11, 22], identificado consistentemente como principal *benchmark* para regressão simbólica. A análise traz diversos apontamentos de possíveis anomalias identificadas nos dados *black box* do *benchmark*, indicando pontos que requerem atenção, preferencialmente através de uma análise especialista acerca dos dados.

Trata-se de um ponto onde a evolução do SRBench é necessária, como indicado pelos próprios autores da bateria de testes em sua última publicação [11]. Neste sentido, também compõe este trabalho uma sugestão de possível melhoria dos dados *black box* do SRBench por meio de uma substituição e uma correção em dados de modelagem de poluentes presentes na bateria de testes. O objetivo é aproximar o SRBench da realidade científica atual, que vai ao encontro tanto dos objetivos deste trabalho, quanto dos objetivos do próprio SRBench.

Este texto está organizado da seguinte forma: a seção 1.1, sequência desta introdução, discute a distinção na literatura de aprendizado de máquina entre interpretabilidade e explicabilidade. O Capítulo 2 define a tarefa da regressão simbólica e detalha os principais métodos agnósticos para sua implementação: a seção 2.1 traz uma visão geral acerca de diferentes implementações para soluções do problema da regressão simbólica, a seção 2.2 discute limitações apresentadas por essas abordagens, e a seção 2.3 apresenta um exemplo mínimo, ilustrativo, para o processo de regressão simbólica. O Capítulo 3 aprofunda a discussão sobre algoritmos evolutivos para regressão simbólica por meio de uma breve descrição da literatura clássica de programação genética, em especial [21], e de uma revisão detalhada do algoritmo PySR com base em [5] e [23]. O Capítulo 4 explora a regressão simbólica via transformers à luz do modelo E2E [10]. O Capítulo 5 trata da avaliação de métodos para regressão simbólica, com foco no SRBench [22], seu principal benchmark, apresentando seus resultados e competições, uma análise crítica de seus conjuntos de dados, e uma proposta de evolução do mesmo por meio de novos conjuntos voltados à predição da concentração de $MP_{2.5}$ a partir de variáveis meteorológicas e temporais. Por fim, o Capítulo 6 apresenta considerações finais sobre o trabalho realizado, e as possibilidades para trabalhos futuros.

1.1 Aprendizado de máquina explicável e interpretável

Na literatura de aprendizado de máquina, os termos "interpretabilidade" e "explicabilidade" são frequentemente tidos como sinônimos [24, 25], com o último dando nome à linha de pesquisa de inteligência artificial explicável (xAI, *explainable artificial intelligence*), também catapultada pelo governo dos Estados Unidos [26, 27].

Contudo, para clareza na definição dos objetivos deste projeto, é mais adequado tomar a perspectiva estabelecida por Rudin em [28] que distingue explicitamente os termos.

Essa abordagem é justificada pois a regressão simbólica, principal objeto de estudo do projeto, é uma tarefa que se encaixa perfeitamente na produção de modelos interpretáveis, mas não plenamente na explicação de modelos *black box*, mesmo que aplicações conjuntas existam [29]. A distinção é concisa [25]:

- O objetivo do aprendizado de máquina interpretável é produzir modelos imediatamente compreensíveis ao olhar humano;
- O aprendizado de máquina explicável visa criar métodos para tornar os resultados produzidos por modelos *black box*, cujo funcionamento interno é complexo demais para o entendimento humano, mais transparentes, por meio de explicações adicionais posteriores à definição do modelo.

Lipton [30] especifica ainda mais a definição de interpretabilidade, estabelecendo que um modelo interpretável deve ser entendido em uma quantidade razoável de tempo, e que seus dados de entrada e parâmetros sejam intuitivos.

Trata-se de uma visão admitidamente subjetiva, a qual Cranmer [5] adiciona que a interpretabilidade de um modelo varia conforme o usuário, exemplificando que um modelo dilogarítmico pode ser muito intuitivo para um especialista em física de partículas, porém muito obscuro sob o olhar de outros destinatários. Nota-se que definições de explicabilidade e interpretabilidade para SciML dadas em [15] têm este cuidado, ao relacionar a interpretabilidade com a familiaridade do usuário a um determinado domínio do conhecimento. São exemplos de modelos interpretáveis [25]:

- Os modelos baseados em regras, onde condições do tipo "se, então" a respeito do problema analisado são utilizadas para tomada de decisões de forma transparente e enxuta. Um representante desta categoria são as árvores de decisão, passíveis de geração automática via ML através de um algoritmo tal qual RuleFit [31];
- Modelos baseados em pontuação, como o SLIM [32], baseado em programação inteira e otimização esparsa via penalização com norma-l_0, um modelo orientado a dados que aprende um sistema de pontuação que define o risco de apendicite em pacientes pediátricos;
- Modelos aditivos generalizados (GAMs), uma extensão da regressão linear clássica via mínimos quadrados ordinários baseada na soma de funções suaves, tornando possível a não-linearidade;
- A regressão simbólica, uma generalização do processo clássico de regressão, mas ainda objetivando produzir uma saída analítica, isto é, uma expressão matemática.

O conceito de explicabilidade, por sua vez, contrasta com o de interpretabilidade ao admitir uma gama maior de abordagens. A definição em [25] entende que qualquer método que provenha uma explicação *post hoc* útil sobre um modelo *black box* pode ser entendido

como uma direção para explicabilidade, mesmo se for baseada em apenas um único ponto de dados.

A principal variação entre as abordagens é, neste sentido, decorrente de uma mudança de escopo: ela pode ser global, abrangendo todos os dados analisados, ou local, focada em apenas algumas *features* ou resultados; pode ser específica a uma classe de modelos, ou agnóstica ao modelo. Das categorias estabelecidas na revisão [25], são mais aderentes aos objetivos deste projeto os métodos de atribuição, tipo mais frequentemente aplicado de método para explicação, e os metamodelos simbólicos.

Seguindo a definição do trabalho [33], os métodos de atribuição objetivam quantificar a relevância das variáveis independentes de um problema para os resultados fornecidos pelo modelo.

Definição 1 (Modelo de atribuição) *Sejam $i, s \in \mathbb{N}^*$ e $f\colon \mathbb{R}^i \to \mathbb{R}^s$ um modelo black box já treinado. Dado o vetor de variáveis independentes $u \in \mathbb{R}^i$ utilizado pelo modelo, um* modelo de atribuição *é uma aplicação que, a partir de n amostras $x_1, \ldots, x_n \in \mathbb{R}^i$, associa a cada variável independente $j \in \{1, \ldots, i\}$ e a cada amostra $k \in \{1, \ldots, n\}$ uma importância $a_{jk} \in \mathbb{R}$ para a saída produzida, de modo que*

$$A_f(u, x_k) = (a_{1k}, \ldots, a_{ik}), \qquad k = 1, \ldots, n.$$

Um dos principais [25] representantes desta categoria é o já citado SHAP (SHapley Additive exPlanations) [34], um método baseado em teoria dos jogos com o objetivo de unificar as abordagens propostas dos métodos [35–41].

Os metamodelos simbólicos, representados por [42] e [43], objetivam a geração de uma aproximação analítica interpretável, com algum grau de imprecisão aceitável, de um modelo *black box*. Estes trabalhos propõem uma abordagem de regressão simbólica passível de otimização via gradiente descendente, também abrangendo a produção de modelos interpretáveis e os comparando diretamente a expressões produzidas pelo algoritmo gplearn [44] em mesmas aplicações.

Nota-se contudo, nos resultados apresentados, uma aproximação destes modelos com a ideia de destilação simbólica [45] por meio da demonstração de casos onde o erro numérico dos metamodelos é desprezível e, portanto, tornam-se alternativas viáveis ao modelo *black box* original.

Pode-se, com algum rigor, distinguir os metamodelos simbólicos de trabalhos como o de Cranmer [29] através de seu objetivo original: enquanto os metamodelos ambicionam explicações, Cranmer já parte da finalidade de obter um modelo simbólico.

Todavia, trata-se de conceitos notavelmente análogos, e particularmente úteis para aplicações com dados da vida real, passíveis de ruído, onde a robustez das redes neurais pode aliar-se com a regressão simbólica para produção de um modelo acurado e interpretável.

Explicações para aumentar a transparência de modelos *black box* têm se provado muito úteis para demonstrar confiabilidade e explicitar vieses adquiridos no treinamento, a exemplo de [46], que explicitou a existência de um viés racial em um modelo para predição de reincidência criminal.

Em alguns casos, todavia, explicações não são suficientes, sendo necessária uma interpretação completa do caminho para as conclusões do modelo, especialmente em cenários de alto risco para tomada de decisões, como aplicações de IA na medicina, justiça, infraestrutura pública e risco financeiro, casos onde modelos interpretáveis são preferíveis, como argumenta Lipton em [30].

Mesmo Lipton, contudo, admite a utilidade de modelos altamente complexos em casos específicos dado, por exemplo, sua capacidade de encontrar padrões ocultos. Justifica-se

assim uma abordagem unificadora, como a de [29], onde o resultado produzido é interpretável, porém apoiado em uma base de aprendizado profundo dada a alta complexidade do problema.

Capítulo

2

A regressão simbólica

Semanticamente, observa-se na literatura algumas definições para a regressão simbólica (SR, de *symbolic regression*), a variar conforme o quão generalizada é a descrição. Pode ser entendida como um problema [47], uma sub-área do aprendizado de máquina [48], uma família de métodos [19], ou até mesmo como um método específico pertencente à classe dos algoritmos evolutivos [20], dado que este é o método mais popular de implementação da SR, e portanto se confunde com a mesma.

Neste texto, entende-se a regressão simbólica como uma tarefa de aprendizado de máquina [10, 48] cujo objetivo pode ser compreendido de forma composta: efetuar regressão (i) sobre um conjunto de dados, cujo resultado deve ser uma função analítica $f : \mathbb{R}^n \to \mathbb{R}$ (ii), ou seja, expresso simbolicamente.

Esta descrição é bastante ampla, e é usada por [19], que entende até mesmo a regressão linear clássica via mínimos quadrados ordinários e redes neurais *feedforward* como caminhos para regressão simbólica, pois os modelos obtidos podem ser expressos analiticamente.

Contudo, uma definição tão aberta fere uma das principais vantagens que pode ser trazida pela regressão simbólica: a interpretabilidade. Afinal, um modelo obtido via mínimos quadrados com centenas de coeficientes e uma rede neural com milhares de parâmetros não são razoáveis ao olhar humano [25].

Neste âmbito, alguns autores adicionam [5,10,25,47,48] então a regra de que o resultado produzido pela regressão simbólica deve ser interpretável (iii), mesmo que essa noção carregue alguma subjetividade, como já discutido anteriormente neste texto. Essas três sub-definições, todavia, ainda não compreendem o objetivo central dos principais métodos para regressão simbólica, por ignorar um complicador: f não tem forma predefinida (iv).

Ao considerar apenas os três primeiros princípios, pode-se considerar regressão linear clássica sobre um número suficientemente pequeno de elementos, ou uma rede neural suficientemente pequena como métodos para regressão simbólica, reduzindo a tarefa a um problema de otimização, meramente. Levando em conta, por outro lado, que não se pode definir uma forma prévia para f, expande-se a tarefa para um problema de busca por uma expressão cujo espaço funcional é potencialmente infinito. Para os fins deste projeto, portanto, assume-se a regressão simbólica como uma tarefa definida pelos princípios

(i) efetuar regressão de forma a obter um modelo $f : \mathbb{R}^n \to \mathbb{R}$, ou seja, definir um método para obtenção de um modelo f via aprendizado de máquina supervisionado [11] sobre d amostras das variáveis independentes e dependentes de um problema, de forma que

(ii) o modelo obtido f seja simbólico, dado por uma expressão matemática analítica;

(iii) o modelo obtido f seja interpretável, passível de compreensão humana imediata;

(iv) f não possui forma funcional predefinida.

É razoável, porém, admitir duas ressalvas ao ponto (iv). A primeira é limitar o espaço funcional F a qual pertence f ao defini-lo com um subconjunto de um conjunto primitivo P, predefinido pelo usuário da regressão simbólica.

São pertencentes usualmente a P operações e funções matemáticas básicas como adição, subtração, multiplicação, divisão, seno, cosseno, logaritmo, funções de potência e exponenciais, além de escalares e funções constantes (ex.: $c(x) = \pi$). Usando um exemplo de [48], tem-se $P = \{+(\cdot,\cdot), -(\cdot,\cdot), \times(\cdot,\cdot), x_1, x_2, \mathbb{R}\}$, com $F \subseteq P$, ou seja, F contém todos os polinômios de um grau qualquer envolvendo as variáveis independentes x_1 e x_2.

Nota-se que, mesmo desconsiderando o caráter contínuo dos escalares pertencentes a P, o espaço de busca F continua infinito, ao compreender qualquer composição possível entre os elementos de P. Portanto, é comum [5, 48, 49] aos algoritmos de regressão simbólica a inclusão de um "tamanho máximo" para a expressão.

Este limitante superior para F pode ser dado de diversas formas, podendo estar associado a um fator chamado de complexidade, explorado mais a frente, como no PySR [5], ou ainda ser referente a profundidade máxima da árvore binária que representa a expressão descoberta, como no Operon [49]. Dadas essas considerações, apresenta-se a definição de regressão simbólica utilizada neste projeto a seguir, baseada na de [48].

Definição 2 ***(Regressão simbólica)*** *Dado um conjunto P de funções e variáveis, uma métrica $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$, vetores $x_i = (x_{1i}, \ldots, x_{di}) \in \mathbb{R}^d$ e escalares $y_i \in \mathbb{R}$, para $i = 1, \ldots, n$, a tarefa da regressão simbólica é encontrar uma função f^* tal que*

$$f^* = \arg\min_{f \in F} \mathcal{L}(y, f(x)) \tag{2.1}$$

onde F é um espaço de busca definido por P, (x_i, y_i) é uma amostra vinda de um conjunto de dados de treinamento, x_{ji} é a j-ésima variável independente da amostra i, y_i é a variável dependente da amostra i, e $\mathcal{L}$ é uma métrica que define o quão bem ajustada a f descoberta pelo processo de busca está ao conjunto de treinamento.

Para $\mathcal{L}$ há, na maioria das implementações [48], uma particularidade da regressão simbólica: é usual que o erro de um modelo definido via aprendizado de máquina seja avaliado por uma função *loss* tal como o erro médio absoluto ou o erro quadrático médio [6] porém, para aderir ao princípio (iii) aqui definido da regressão simbólica, muitos métodos também incorporam uma regularização que penaliza expressões por sua complexidade. Esta penalização geralmente está associada a um fator λ de controle, conhecido como parcimônia [5]. Seja $C(f)$ a complexidade de uma expressão avaliada, a métrica de avaliação $\mathcal{L}$ é dada então por

$$\mathcal{L}(y, f(x)) = l(y, f(x)) + \lambda C(f), \tag{2.2}$$

onde l é uma função *loss* tradicional do aprendizado de máquina, tais quais as citadas anteriormente, e a definição de C varia conforme o método. Para o PySR [5], por exemplo, C é dada pelo número de nós da árvore unária-binária equivalente a expressão avaliada.

A noção de complexidade leva a um outro conceito muito importante, utilizado dentro da regressão simbólica, e no *machine learning* de modo geral: a melhoria de Pareto [50]. Trata-se de uma ideia originada na socioeconomia, pelo cientista Vilfredo Pareto: uma melhoria socioeconômica é considerada como "de Pareto" quando a sua aplicação não gera

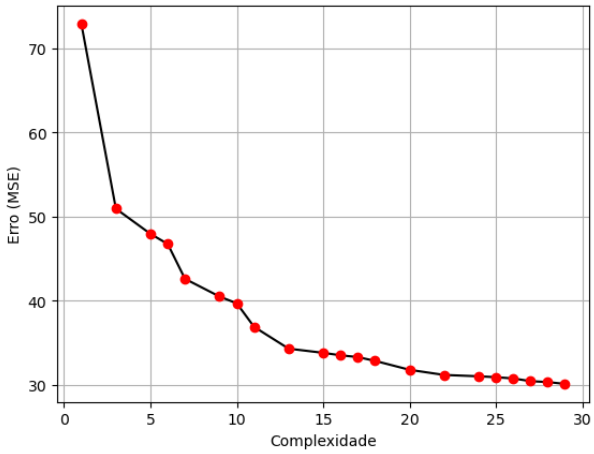

Figura 2.1: Ilustração de uma fronteira de Pareto ideal, na qual cada ponto representa um modelo. Elaboração própria.

detrimento para nenhum componente da sociedade. Um cenário socioeconômico pode ser considerado como "ótimo de Pareto" quando não há mudanças possíveis a serem feitas para melhorar um determinado indicador sem piorar outro.

Na regressão simbólica, a aplicação da eficiência de Pareto é direta: para cada complexidade de expressão C trabalhada por um algoritmo, uma solução é considerada ótima de Pareto se não existe outra solução com complexidade menor ou igual que apresente melhor desempenho. Assim, a busca por uma solução mais precisa tipicamente está associada a um incremento na complexidade. Contudo, uma solução só é considerada como "ótima de Pareto" se for mais precisa que todas as outras de complexidade inferior.

É comum que algoritmos de SR, como PySR [5] e Operon [49], forneçam seus resultados na forma de uma "fronteira de Pareto", uma coleção de soluções composta pela melhor solução para cada complexidade, desde que esta seja melhor que todas as outras de complexidade inferior, ou seja, todas as soluções ótimas de Pareto. Na literatura, essa coleção usualmente vem ilustrada em um gráfico como o da Figura 2.1, mostrando em seus eixos a complexidade contra alguma métrica de acurácia utilizada, e comumente recebe o nome de "*hall* da fama" [5].

Nota-se então, salvo em casos onde há um bom pré-condicionamento do problema, que a tarefa proposta pela regressão simbólica não é trivial, com o trabalho [48] fornecendo uma prova de que se trata de um problema NP-hard. Compreende-se então a predominância de métodos heurísticos [19] no estado da arte das implementações, dado que mesmo casos pequenos (fórmulas simples, com poucos aninhamentos e variáveis independentes) podem ser desafiadores [5, 11, 22].

A definição de regressão simbólica apresentada pode ser ampliada para outros conjuntos além dos reais $\mathbb{R}$ (ex.: complexos $\mathbb{C}$), porém poucos métodos assumem essa generalização, e têm seu foco somente no âmbito dos valores reais [48], e portanto a formulação apresentada é suficiente para o escopo deste projeto.

A seguir, são apresentadas e discutidas algumas das principais abordagens para solução do problema posto pela regressão simbólica. Neste sentido, é apresentada também a metodologia utilizada neste projeto para descoberta dos métodos.

2.1 Métodos para regressão simbólica

Dada a já exposta não trivialidade do problema proposto pela regressão simbólica, foi natural a proposta de diversas soluções ao decorrer do desenvolvimento da área [11]. Para tomar conhecimento das soluções empregadas ao longo do tempo e, especialmente, das que atualmente constituem o estado da arte da regressão simbólica, foi efetuado no decorrer deste projeto um extenso e contínuo processo de pesquisa. Em particular, as principais fontes consultadas foram:

- *Reviews* e *surveys*, em especial [19, 20, 51].
- Aplicações recentes de SR, isto é, foram analisadas publicações cujo objetivo era resolver um problema específico utilizando SR, e coletado o método aplicado, a exemplo de [29, 52–59].
- Benchmarks (comparativos entre soluções), particularmente o SRBench em suas diversas edições [11,22,60], e o EmpiricalBench [5], cujos dados foram posteriormente incorporados ao SRBench em [11]. Em todas as edições do SRBench, é associada uma nota a cada método baseado na simplicidade e acurácia do modelo obtido, para os desafios com dados sintéticos, e adicionalmente à confiabilidade dos modelos para os desafios com dados reais, enquanto o EmpiricalBench avalia binariamente se o modelo acertou ou não a expressão esperada, dado que todas têm uma *ground truth*.
- As edições de 2022 [61] e 2023 [62] da competição do SRBench. Estas são especialmente relevantes ao projeto por incluírem tracks ("desafios") utilizando dados reais e ruidosos [63].

É evidente, ao analisar os métodos provenientes destas fontes, que algoritmos evolutivos, popularizados pela programação genética [21] (GP, de *genetic programming*), representam a principal via de implementação para SR, afirmação verificável na literatura [11, 19, 20, 62], onde é possível até mesmo encontrar definições de SR como um subconjunto da GP, tamanha a dominância da categoria de algoritmos na área ao longo de décadas [5].

Por muito tempo, o principal algoritmo [5] desta categoria foi o Eureqa, uma solução privada baseada no conceito de *Age-Fitness Pareto Optimization* [64].

Essa prevalência não é apenas quantitativa, um senso comum, porém também qualitativa, pois em muitos casos a GP ainda é superior às suas alternativas como evidenciado em [11] e [12]. Como ressalta a discussão do SRBench em 2025 [11], contudo, não há via única na regressão simbólica, e existem contextos onde outras categorias de métodos se destacam, como na competição de 2022 [61], onde um método baseado em aprendizado profundo venceu o desafio com dados reais. Em suma, tornou-se claro que um dos objetivos secundários deste projeto compreenderia um estudo aprofundado dos algoritmos evolutivos no contexto da regressão simbólica, tomando ao menos um exemplo como objeto de estudo.

Destacam-se também os métodos baseados em redes neurais. Mesmo podendo ser categorizados em um mesmo grupo, são evidentemente bastante distintos em abordagem, com modelos baseados em aprendizado por reforço (RL, de *reinforcement learning*) [47,65], transformers [10, 66] e *shallow neural networks* [67].

Esforços neste sentido surgiram de modo a tentar aproveitar a ampla capacidade de generalização, robustez e aptidão para lidar com alta dimensionalidade trazidas pelo aprendizado profundo [5,10,12], em contraste com a dominante GP, uma abordagem sabidamente lenta, portanto escalando mal conforme a dimensão dos dados aumenta, e muito suscetível

a ruídos [19]. Contudo, não há prova concreta [11] de que as redes neurais realmente são uma alternativa preferível nestes casos.

Em [12], os autores evidenciam diversas aplicações científicas onde os algoritmos Operon [49], HeuristicLab [68] e ScSR [69] superam o modelo baseado em arquitetura transformer E2E [10]. No SRBench de 2025 [11], observa-se que as redes neurais, como NeSymRes [66], a própria E2E [10] e até mesmo modelos agregados como uDSR [70] e TPSR [71] estão no grupo das piores performances obtidas na bateria de testes.

Todavia, o mesmo trabalho reforça que, no estado da arte da regressão simbólica, ainda não há ganho fácil, isto é, um modelo, algoritmo ou abordagem específica que consistentemente supere todas as outras. A exemplo, na competição de 2022 [61], o modelo uDSR, composto em sua maior parte por modelos de aprendizado profundo foi campeão no desafio com dados reais, superando diversos algoritmos genéticos, como Operon, PS-Tree [72] e PySR, enquanto em [11], estes métodos obtiveram algumas das melhores métricas, mas o uDSR teve algumas das piores.

Mesmo que a programação genética e redes neurais cubram boa parte dos métodos utilizados para solução da SR [11, 19], a dificuldade do problema é convidativa para uma miríade de abordagens, algumas de difícil categorização dada a singularidade de sua proposta, como os métodos Rils-rols [73], um método iterativo que combina ideias de algoritmos de busca e otimização via mínimos quadrados ordinários; QLattice [74], baseado em amostragem de distribuições; BSR [75], com uma abordagem Bayesiana; e SINDy [16], que aplica otimização esparsa sobre a combinação linear de funções candidatas a fazer parte do modelo final, uma ideia similar a de FFX [76], ainda que menos automatizada, visto que este último traz por padrão uma quantidade massiva de candidatos para poda posterior. Existem propostas de categorização dos métodos para SR, como o *review* [19], que traz categorias dispostas em um fluxograma, e a apresentação [51], que resume os métodos em "baseados em programação genética" (ex: PySR [5]), "baseados em sequência" (ex.: ODEFormer [77]), "baseados em dicionário" (ex.: SINDy [16]) e "baseados em gramática" (ex.: ProGED [78]). Contudo, a diversidade do cenário de SR prejudica uma categorização tão absoluta, e não é claro onde pertencem casos como o BSR, cuja abordagem estatística não se encaixa bem em nenhuma das categorias.

Nota-se também a presença de métodos híbridos, que combinam abordagens de diferentes categorias (ex.: GP e redes neurais), ou até mesmo diferentes métodos de uma mesma categoria. Este último é o caso do citado TPSR [71], campeão da competição de 2023 [62] no desafio de dados sintéticos, cuja proposta é pré-condicionar a solução do problema da SR utilizando cadeias de Markov, e posteriormente resolvê-lo por meio de uma combinação dos modelos E2E [10] e NeSymRes [66].

O modelo uDSR [70] também entra nesta categoria, sendo uma combinação dos modelos AI Feynman [79], um modelo também híbrido para pré-condicionamento de SR, mesmo que possa ser usado independentemente [70], DSR [47], NeSymRes, E2E, SymFormer [80] e regressão linear.

Mais recentemente, com o advento dos modelos de linguagem de larga escala (LLMs), também surgiram esforços em sua utilização na SR, em especial para pré-condicionamento de problemas, com o LaSR [81], que utiliza um modelo LLM[1] como condicionador de buscas utilizando o algoritmo PySR.

Ressalta-se, neste sentido, que mesmo métodos categorizados como GP ou *deep learning* ainda podem possuir algum grau de hibridismo, como é o caso do PySR, que incorpora a têmpera simulada, um algoritmo de otimização inspirado em física, e o PS-Tree, que associa árvores de decisão à GP.

[1] O modelo de LLM em específico é mutável. Foram testados pelos autores o GPT 3.5 [82] e LLama 3 8b [83].

A Tabela 2.1 traz uma visão geral dos principais métodos descobertos no desenvolvimento deste projeto. A categorização provida é baseada na predominância da técnica utilizada pelo método, a exemplo do PS-Tree que, mesmo incorporando ideias de árvores de decisão, ainda é majoritariamente um algoritmo evolutivo. Dados os objetivos do projeto, essa tabela leva em consideração apenas métodos cuja ambição é agnóstica, ou seja, aplicáveis em diferentes contextos, ignorando portanto soluções desenvolvidas para sanar um problema específico.

As categorias criadas foram baseadas, em principal, na forma na qual o SRBench de 2025 [11] apresenta os métodos, estabelecendo então as categorias de "métodos baseados em programação genética", "métodos baseados em redes neurais", "métodos híbridos", e "outros métodos". No SRBench 2025, todavia, há uma especificidade ligeiramente maior, apresentando o BSR como um "método bayesiano" e os modelos [84, 85] como "métodos exaustivos".

Tabela 2.1: Sumarização categorizada de métodos para regressão simbólica.

Categoria	**Métodos descobertos**
Métodos baseados em programação genética	PySR [5], Operon [49], PS-Tree [72], gplearn [44], HeuristicLab [68], AFP/Eureqa [64], Brush [86], EPLEX [87], FEAT [88], GPGomea [89], SR-Forest [90], StackGP [91].
Métodos baseados em redes neurais	E2E [10], NeSymRes [66], EQL [67], DSR [47], Sym-Q [65], SymbolicGPT [92], SymFormer [80].
Métodos híbridos	TPSR [71], uDSR [70], LaSR [81], LLM-SR [93], AI Feynman [79].
Outros métodos	FFX [76], QLattice [74], BSR [75], ESR [84, 85], SINDy [16].

Dado este ecossistema tão amplo e diverso de abordagens, aliado ao caráter orientado a método deste projeto, onde objetiva-se explorar a regressão simbólica como instrumento para o SciML, e não necessariamente uma aplicação específica, foi traçada uma linha incremental, onde foi realizado um estudo inicial de um método de programação genética, dada sua dominância no cenário, seguindo para um método baseado em redes neurais com pouco ou nenhum hibridismo.

O aprofundamento teórico foi, então, iniciado com a análise do PySR [5], apresentado a seguir juntamente de sua descrição em detalhes, incluindo um panorama geral de algoritmos evolutivos, e de uma justificativa de sua escolha como objeto de estudo desta dissertação.

Posteriormente, o aprofundamento seguiu a partir do estudo detalhado do modelo E2E [10], incluindo sua base como rede neural pertencente à arquitetura transformer [9].

Esta escolha está justificada na crescente adoção de LLMs para formação de modelos híbridos de regressão simbólica. Como investigar LLMs diretamente extrapolaria o escopo deste projeto, optou-se por abordar os modelos transformers, base dos LLMs, mas ainda dentro do contexto da regressão simbólica.

Finalmente, métodos de benchmark para regressão simbólica também foram detalhadamente investigados, em especial o SRBench [11, 22, 60] em todas as suas edições, dada sua relevância para a área.

2.2 Limitações da regressão simbólica

A partir deste detalhamento do estado da arte da regressão simbólica, é possível compreender algumas limitações ainda observadas na área. Em [19], são citadas uma série de limitações em diferentes contextos, podendo ser entendidas como: limitações de conceito, inerentes a proposta da regressão simbólica; limitações de método, especificas às diferentes abordagens para resolução da tarefa proposta pela regressão simbólica; e limitações de avaliação, relativas a forma na qual os métodos são qualitativamente avaliados, em especial via *benchmarks*.

Conceitualmente, a regressão simbólica está limitada à produção de expressões matemáticas. Entende-se então que, em alguns contextos, a solução aproximada produzida pela SR não se adequará bem aos dados de entrada se os mesmos não forem compatíveis com a expressividade simbólica.

Por exemplo, no caso de predição de poluentes trabalhado posteriormente neste texto, a variável de dia juliano, estabelecida em um intervalo $j \in [1, 366]$, tem sua influência na concentração de um determinado poluente atmosférico de maneira condicional: nos dias representantes do inverno no hemisfério sul ($j \in [172, 264]$) espera-se aumento nas concentrações, e diminuição caso contrário. Tal condicional específica é dificilmente modelada com sucesso por um modelo de SR sem um condicionamento prévio, enquanto é possível que um modelo naturalmente condicional (ex.: árvores de decisão) fosse capaz de modelar esse fenômeno. Pode-se, todavia, contornar esta limitação com transformação de variáveis, como também exemplificado na seção 5.3.

As distintas abordagens descritas na seção 2.1 têm suas limitações particulares, por sua vez. A programação genética, abordagem tradicional para SR, é um processo conhecidamente lento [5, 11, 19]. Essencialmente, essa lentidão advém de uma necessidade deste tipo de método de gerar uma quantidade massiva de expressões, explorando formas funcionais diferentes através de múltiplas populações. Alguns métodos, como o PySR [5] e o Operon [49], buscam superar parte desta lentidão fazendo uso pesado de computação paralela, porém a condição de busca se mantém.

Este fato faz com que métodos baseados em GP não escalem bem com a dimensão do conjunto de dados. Aumentar o número de amostras tem impacto direto no tempo de execução. Ademais, dada uma quantidade fixa de iterações evolutivas, aumentar a quantidade de variáveis do problema, bem como aumentar a variabilidade numérica, prejudica muito a convergência dos métodos, com a chance de que o mesmo encontre famílias de formas funcionais bem ajustadas ao problema diminuindo conforme a complexidade do problema aumenta.

Em suma, mesmo que a quantidade de iterações seja alta, a capacidade de modelagem de métodos baseados em GP diminui conforme a dimensionalidade do conjunto [19], seja em número de amostras, seja em número de variáveis. O PySR, por exemplo, exibe por padrão um aviso ao usuário quando um conjunto com mais de 10 mil amostras é inserido [5], indicando que a partir desta volumetria não se espera melhoria de convergência e, portanto, é recomendando trabalhar com uma quantidade menor de dados.

Os métodos baseados em redes neurais também têm suas limitações particulares, porém essas variam muito conforme a arquitetura escolhida. Uma análise da *review* [19] cita que o método EQL [67] apresenta instabilidades numéricas dada a combinação do algoritmo *backpropagation* com a presença de operações de divisão e logaritmo nos modelos. O E2E [10], por sua vez, tem um processo de inferência muito custoso computacionalmente, dada a sua extrapolação do processo de regressão em uma representação linguística, e portanto é limitado à apenas 10 variáveis independentes e 200 amostras, fazendo uso de *bagging* para lidar com volumes maiores de dados.

É um empecilho comum à performance dos métodos baseados em redes neurais, contudo, a necessidade de treinamento, especialmente se comparados à GP, que não apresenta a mesma limitação. Quando alimentados diretamente com os dados que são o alvo da regressão, os modelos podem apresentar resultados imprecisos para conjuntos pouco comportados. Em outros casos, como o do E2E, onde o modelo é pré-treinado, a inferência é extremamente limitada às expressões vistas em treinamento e, portanto, a capacidade de extrapolação do modelo é drasticamente reduzida.

Uma outra limitação, sendo esta simultaneamente computacional e conceitual, é a necessidade de definir as operações possíveis dentro das expressões, isto é, restringir o espaço funcional. Essa limitação usualmente é manual [5, 49], onde o usuário seleciona operadores unários e binários passíveis de presença na expressão. Porém, é possível, como no E2E [10], que a limitação seja advinda de um conjunto de treinamento. Operadores ausentes do conjunto de treinamento jamais serão inferidos pelo modelo. Esta limitação é especialmente problemática em métodos como o SINDy [16], muito dependentes de uma seleção prévia adequada de operandos por parte do usuário.

Por fim, é notada também em [19] uma "limitação de *benchmarks*". Comenta-se que a maioria dos métodos analisados no *review*, na época de sua elaboração (2023), fazia uso apenas de conjuntos de dados sintéticos ao apresentar métricas de performance. Atualmente, é comum que novos métodos propostos sejam avaliados por meio do SRBench [11, 22], que tornou-se padrão da área com o decorrer de sua existência. Todavia, o próprio SRBench também carrega limitações, apontadas na sua própria revisão de 2025 [11], e detalhadas no capítulo 5. Mesmo que tente avaliar os métodos utilizando problemas reais, alguns dos dados propostos pelo SRBench apresentam indícios de anomalia, e são de difícil rastreabilidade.

2.3 Exemplo de regressão simbólica

O experimento de Johannes Kepler [94] que culminou em sua descoberta das leis do movimento planetário é considerado um claro exemplo de regressão simbólica [5,11,79,95] implementada manualmente, devido ao processo empregado pelo astrônomo. Utilizando dados produzidos observacionalmente e sem qualquer tipo de assistência [5] por Tycho Brahe, Kepler obteve sua fórmula após um processo de tentativa e erro baseado em sua intuição geométrica. A ambição de automatizar a descoberta científica é uma das principais forças motrizes por trás da busca por métodos cada vez melhores, ao passo que, na apresentação do SRBench 2025 [11], o processo de aplicar regressão simbólica para modelagem de fenômenos físicos é descrito como "automatizar Kepler".

Portanto, é tratado nesta seção um exemplo mínimo deste processo, implementado por Cranmer como parte de seu *benchmark* EmpiricalBench, para reprodução do experimento de Kepler, visando obter como resultado uma expressão equivalente a sua terceira lei postulada, dada por

$$\frac{T^2}{r^3} = k, \tag{2.3}$$

onde T é o período de translação de um planeta, em dias, r é o raio da órbita de um planeta, e k é um escalar constante para um mesmo sistema estelar. Os dados utilizados foram os exatos mesmos produzidos por Brahe e usados por Kepler em 1618: valores de seis planetas do sistema solar, dispostos na Tabela 2.2, sendo bastante imprecisos, onde nota-se que k, mesmo que pouco variante, não é constante. O alvo escolhido para

regressão foi o período de translação T. Portanto, foi necessária a reescrita

$$\begin{aligned} \frac{T^2}{r^3} &= k \\ \Rightarrow T^2 &= r^3 k \\ \Rightarrow T &= \sqrt{r^3 k}, \end{aligned}$$

onde r é, deste modo, o único preditor do problema.

$$T = f(r) = \sqrt{r^3 k}, \tag{2.4}$$

sendo este o resultado esperado para o experimento.

Planeta	**r**	**T**	$\mathbf{k}^{-1}$
Mercúrio	0.389	87.77	$7.64 \cdot 10^{-6}$
Vênus	0.724	224.70	$7.52 \cdot 10^{-6}$
Terra	1	365.25	$7.50 \cdot 10^{-6}$
Marte	1.524	686.95	$7.50 \cdot 10^{-6}$
Júpiter	5.20	4332.62	$7.49 \cdot 10^{-6}$
Saturno	9.510	10759.20	$7.43 \cdot 10^{-6}$

Tabela 2.2: Dados para o experimento visando redescobrir a terceira lei de Kepler via regressão simbólica.

Mesmo sendo extremamente simples, encontrar a terceira lei de Kepler automaticamente não é um problema trivial, dadas as dificuldades já apresentadas neste texto inerentes ao processo de SR. O EmpiricalBench indica que os algoritmos Operon [49], EQL [67] e QLattice [74] falharam em aprender a fórmula a partir das poucas amostras, enquanto o PySR [5] e o DSR [47] tiveram sucesso. Cada algoritmo foi executado 5 vezes, com sementes aleatórias distintas. Para maior clareza, dado que o objetivo desta seção é apenas fornecer um exemplo ilustrativo, será tratada apenas uma das execuções do PySR, dado que este é um dos objetos de estudo deste projeto e teve sucesso em todas as cinco tentativas. A configuração utilizada para o PySR é dada na Tabela 2.3. O *hall* da fama obtido é dado na Tabela 2.4, onde a expressão correta está em destaque.

Função *loss* customizada	$l(y, f(x)) = \lvert y - f(x) \rvert$
Número de iterações	1.000.000
Operadores binários	$+,-,\times,/$
Operadores unários	$\cdot^2,\cdot^3,e,\log,\sqrt{\cdot}$
Complexidade máxima da expressão	30
Profundidade máxima da árvore binária	20

Tabela 2.3: Parâmetros fornecidos ao PySR para reprodução do experimento de Kepler.

Compreende-se assim um exemplo mínimo onde, mesmo com dados ruidosos e pouquíssimas amostras, um algoritmo de regressão simbólica foi capaz de expressar uma relação física importante. Trata-se de um exemplo clássico, bastante simples, e onde o resultado esperado (*ground truth*) é conhecido, mas que apresenta o procedimento usual da regressão simbólica para descoberta científica.

C(E)	**l(E))**	**Expressão (E)**
1	$1.99 \cdot 10^{07}$	531.383073971584
2	$4.23 \cdot 10^{06}$	e^{r}
3	$7.57 \cdot 10^{05}$	$1131.3564r$
4	$2.51 \cdot 10^{05}$	$118.9649r^{2}$
5	$\mathbf{5.46 \cdot 10^{01}}$	$\mathbf{366.8675\sqrt{(r^{3})}}$
6	$5.46 \cdot 10^{01}$	$366.8675r^{1.5}$
7	$2.73 \cdot 10^{01}$	$368.0311\left(\sqrt{r} - 0.0033\right)^{3}$
8	$1.11 \cdot 10^{01}$	$\left(\sqrt{r} + 363.7836\right)\sqrt{(r^{3})}$
9	$9.93 \cdot 10^{-01}$	$r^{1.5}\left(0.3451r + 363.5857\right)$
10	$1.87 \cdot 10^{-01}$	$\left(0.0233r^{2} + 364.7563\right)\sqrt{(r^{3})}$
11	$1.30 \cdot 10^{-01}$	$0.0068r^{4} + 364.9577\sqrt{(r^{3})}$
12	$8.65 \cdot 10^{-02}$	$364.6557\sqrt{(r^{3})} + e^{0.4408r} - 1.2982$
13	$8.65 \cdot 10^{-02}$	$364.6556r^{1.5} + \sqrt{(e^{0.8816r})} - 1.3$
14	$3.53 \cdot 10^{-02}$	$r + 364.2348r^{1.5} + e^{0.446r} - 1.787$
15	$1.27 \cdot 10^{-02}$	$364.4604r^{1.5} + e^{0.4456r} + \log\left(r\right) - 0.8809$
19	$1.27 \cdot 10^{-02}$	$364.4604r^{1.5} + e^{0.4456r} + \log\left(r\right) - 0.8809$
20	$1.96 \cdot 10^{-03}$	$364.4603r^{1.5} + e^{0.4456r} + \log\left(r\right) - 0.8667 - \frac{0.0397}{r - 1.3621}$
22	$2.45 \cdot 10^{-04}$	$364.4603r^{1.5} + e^{0.4456r} + \log\left(r - 0.0279\right) - 0.8668 - \frac{0.0401}{r - 1.3622}$
25	$2.45 \cdot 10^{-04}$	$364.4603r^{1.5} + e^{0.4456r} + \log\left(r - 0.0279\right) - 0.8668 - \frac{0.0401}{r - 1.3622}$
29	$8.70 \cdot 10^{-05}$	$-\frac{0.0548\log\left(r^{3} + 0.64\right)}{\sqrt{r}(r - 1.2857)} + 364.4603r^{1.5} + e^{0.4456r} + \log\left(r\right) - 0.8667$

Tabela 2.4: *Hall* da fama para o experimento de Kepler com PySR.

Capítulo

3

Algoritmos evolutivos para regressão simbólica

PySR (abreviação baseada em "Python Symbolic Regression") é uma biblioteca escrita por Miles Cranmer para a linguagem de programação Python a fim de prover uma interface para a biblioteca SymbolicRegression.jl, do mesmo autor, para a linguagem Julia. A tecnologia e o algoritmo que ela implementa são apresentados na publicação [5], onde o nome "PySR" é utilizado concomitantemente para se referir tanto à biblioteca, quanto ao algoritmo de regressão simbólica. Neste texto, exceto em casos explicitamente relatados, o nome PySR refere-se ao algoritmo, não à biblioteca.

No âmbito dos algoritmos de regressão simbólica baseados em programação genética, o estado da arte foi dominado, por um período considerável [5,19], pela solução proprietária Eureqa (2011), tendo sua implementação fechada, e não permitindo [22] controle de diversos parâmetros do processo de otimização. Apenas um dos componentes do algoritmo é abertamente conhecido, por ser baseado no trabalho publicado em [64].

A mudança desta realidade é recente, em especial com o surgimento dos algoritmos gplearn [44] (2015) e Operon [49] (2019) que demonstraram resultados competitivos à Eureqa [22], fazendo-a cair em desuso para aplicações científicas [5].

A proposta do PySR decorre deste movimento: em um nível algorítmico, o trabalho traz algumas propostas originais, como a aplicação de uma nova técnica denominada "parcimônia adaptativa".

Tecnologicamente, como ferramenta, a biblioteca traz um conjunto de funcionalidades para aplicações científicas, como a possibilidade de customizar a função de aptidão e operadores utilizados na busca, e do estabelecimento de restrições às expressões encontradas.

Este aspecto vem fomentando a adoção do PySR para novos projetos e experimentos [96], dado que o Operon, seu principal "competidor" da mesma categoria de soluções, não possui tais facilitadores [5], como pode ser visto na Tabela 3.1.

A flexibilidade do PySR se dá em grande parte graças ao seu *backend* em Julia, uma linguagem compilada *just-in-time*, o que permite eficiência computacional, essencial a processos de programação genética, naturalmente lentos [19,97], ao passo em que ajustes podem ser facilmente concretizados sem necessidade de recompilação completa.

Ademais, o PySR também faz uso pesado de paralelismo real e pseudoparalelismo para uma otimização da eficiência ainda maior, por exemplo, com o uso de instruções SIMD para cálculo de operações binárias e unárias, e paralelização da evolução (real via núcleos ou falsa via threads, a depender da disponibilidade do sistema de execução) de populações.

Este cenário, e considerando que o PySR e Operon apresentam-se consistentemente como estado da arte para regressão simbólica puramente baseada em programação genética [12, 22, 61, 62, 98, 99] levou à adoção do PySR como objeto de estudo deste projeto, como representante da programação genética para regressão simbólica.

Tabela 3.1: Comparação entre os algoritmos PySR e Operon, adaptada de [5].

Funcionalidade	**PySR**	**Operon**
Escrito em linguagem compilada	Sim (Julia)	Sim (C++)
Aproveita múltiplos núcleos de processamento	Sim	Sim
Executa em GPU	Não	Não
Suporte à alta dimensionalidade	Não	Não
Apresenta a fronteira de Pareto completa	Sim	Sim
API em Python	Sim	Sim
Código aberto	Sim	Sim
Suporte à execução em clusters	Sim	Não
Tratamento de ruído incluso (requer acionamento)	Sim	Não
Operadores customizados	Sim	Não
Operadores descontínuos	Sim	Não
Função loss customizada	Sim	Não
Definição de restrições	Sim	Não
Escolha da complexidade máxima	Sim	Não

Neste capítulo, discute-se o algoritmo PySR sob a luz de seu pertencimento à categoria dos algoritmos evolutivos, bem como sua inspiração na popular abordagem de programação genética de Koza [21], por sua vez inspirada no algoritmo genético canônico de Holland [2]. Para fins de aderência ao foco do projeto, especificidades de implementação serão detalhadas apenas no âmbito do algoritmo PySR. Isto é, dadas alternativas possíveis para cada componente de um algoritmo evolutivo, discussões aprofundadas são tecidas apenas no contexto das abordagens propostas por Cranmer [5].

Sempre que possível, a publicação [5], relativa ao algoritmo, foi utilizada como referência. Contudo, a mesma oculta diversas complexidades internas do algoritmo, seja por assumir conhecimento prévio do leitor a respeito de programação genética para regressão simbólica, seja por motivos de brevidade textual. Para compreensão plena do algoritmo foi necessário, desta forma, estender a revisão literária para abranger textos a respeito dos fundamentos dos algoritmos evolutivos, bem como analisar o código [23] da biblioteca, incluindo a tecnologia correlata DynamicExpressions.jl [100], desenvolvida por Cranmer para geração e modificação de expressões matemáticas.

3.1 Introdução aos algoritmos evolutivos

Em seus estudos a respeito da origem das espécies biológicas encontradas na natureza [101], Darwin atribui as características dos indivíduos como consequência de um processo de seleção natural ao qual sobrevivem os indivíduos mais aptos ao ambiente, processo que denominou de evolução. Assim como ocorrido em diversos outros sistemas inspirados por princípios biológicos ou da natureza, como radares e redes neurais, o darwinismo provocou a origem dos algoritmos evolutivos (AEs), visando efetuar processos de otimização e aprendizado a partir de uma heurística baseada na evolução de indivíduos mais aptos de diversas populações [21].

Especialmente a partir dos anos 1960, diversas abordagens inspiradas nesses princípios foram propostas [102, 103], em principal: a programação evolutiva [104]; as estratégias evolutivas [105,106]; e os algoritmos genéticos (AGs) inicialmente criados por Holland [2].

Os algoritmos genéticos viriam a ser a principal base para a programação genética [107], que aplica seus princípios para geração automática de programas de computador, e se tornaria o principal catalisador da popularização dos algoritmos evolutivos ao decorrer do tempo [108]. No âmbito deste texto, destacam-se os algoritmos genéticos e a programação genética, estando estes diretamente relacionados ao algoritmo PySR [5], objeto de estudo desta dissertação.

Mesmo com diferentes abordagens, os algoritmos evolutivos possuem fundamentos comuns [103, 108, 109]:

- A existência de ao menos uma população, composta de indivíduos. Cada indivíduo representa uma solução para o problema que objetiva-se resolver.
- Alguma técnica de avaliação da aptidão (*fitness*) dos indivíduos ao ambiente. Neste contexto algorítmico, a aptidão frequentemente representa a proximidade do indivíduo avaliado com a resposta esperada. Por exemplo, a avaliação do erro de uma aproximação pode ser uma forma de avaliação de aptidão em casos supervisionados onde isso é possível.
- Alguma técnica para selecionar indivíduos de uma população. É comum a simples escolha dos indivíduos mais aptos, porém é possível seguir por outra abordagem, que incentive maior variabilidade genética, como feito no PySR [5]. Também é possível que a seleção para recombinação e a seleção de sobrevivência para a próxima geração, mecanismos descritos posteriormente, assumam lógicas diferentes [110].
- Um operador de recombinação (também chamado de cruzamento ou *crossover*) para gerar novos indivíduos a partir dos selecionados. Em algoritmos onde a genética dos indivíduos é bem definida, este operador visa emular a reprodução sexuada observada na natureza [21], sendo aplicado em um par de indivíduos.
- Na maioria dos casos, um operador de mutação, aplicado em um único indivíduo para geração de um novo indivíduo, modificado de alguma forma.
- Um critério de parada (comumente relacionado com a aptidão) para encerrar o algoritmo.

Para implementação completa de um algoritmo evolutivo, especialmente em um contexto computacional, é necessário ainda considerar outras características importantes [21]: o tipo de estrutura usada para representar os indivíduos (números, strings, árvores, etc.); e quaisquer parâmetros de controle relevantes para o processo, como tamanho da população e valores probabilísticos [21], por exemplo, a probabilidade de recombinação para um par de indivíduos. Nota-se que, no processo evolutivo, é praticamente imperativa [21, 103] a presença de alguma aleatoriedade por meio de operações probabilísticas, pois um comportamento elitista, onde apenas os "melhores" indivíduos são selecionados limita o processo de busca pelo indivíduo mais apto e incentiva o algoritmo a encontrar mínimos locais.

Também muito relevante ao processo de busca é a quantidade de indivíduos distintos que podem ser formados a partir das estruturas, pois definem o tamanho do espaço de busca. No algoritmo genético de Holland [2], os indivíduos são definidos por uma cadeia de caracteres de tamanho fixo. O tamanho da cadeia, bem como a quantidade de caracteres válidos, afeta diretamente o tamanho do espaço de busca.

Figura 3.1: Ilustração de algoritmo genético, baseada no AG Canônico de Holland [2]. Extraída de [3].

Combinam-se então estes conceitos na elaboração do algoritmo evolutivo. Um AE usual começa pela geração de uma população inicial aleatória de tamanho m, isto é, preenchida por m indivíduos gerados aleatoriamente, seguindo alguma regra determinada pela estrutura.

Os indivíduos são então avaliados e, com base em sua aptidão, são selecionados pares de pais para sofrer a recombinação, gerando n filhos.

Este novo amálgama de $m + n$ indivíduos é novamente avaliado, e dentre eles, são selecionados os m indivíduos mais aptos para comporem uma nova geração. O processo se repete até que surja um indivíduo apto o suficiente, de acordo com o critério de parada (ex.: aptidão desejada atingida ou número máximo de gerações atingido). Essa lógica está disposta no Pseudocódigo 1, adaptado de [103] e [110]. Analogamente, a Figura 3.1 ilustra o funcionamento de um algoritmo baseado no AG de Holland [2], também chamado de "AG canônico" [103].

Pseudocódigo 1: Forma geral de um algoritmo evolutivo.

Entrada: parâmetros de controle, como tamanho m da população, tamanho n de filhos gerados e fatores probabilísticos.
Saída: solução do problema.

```
1 inicialize a população com m indivíduos gerados aleatoriamente;
2 avalie cada indivíduo;
3 enquanto critério de parada não atingido faça
4 |   selecione pais;
5 |   recombine pares de pais para gerar n filhos;
6 |   mute os filhos;
7 |   avalie os m + n indivíduos;
8 |   selecione m indivíduos para compor a próxima geração;
9 fim
```

Algoritmos mais avançados introduzem a ideia de diferentes populações evoluindo paralelamente [21] e eventualmente interagindo entre si. É o caso do algoritmo PySR, que também exemplifica diversos outros conceitos apresentados superficialmente nesta seção.

Por isso, a descrição do processo evolutivo seguirá concomitantemente a descrição do PySR, suas técnicas escolhidas para implementação dos fundamentos dos algoritmos evolutivos e novas propostas para melhoria da abordagem, incluindo métodos auxiliares, como a têmpera simulada.

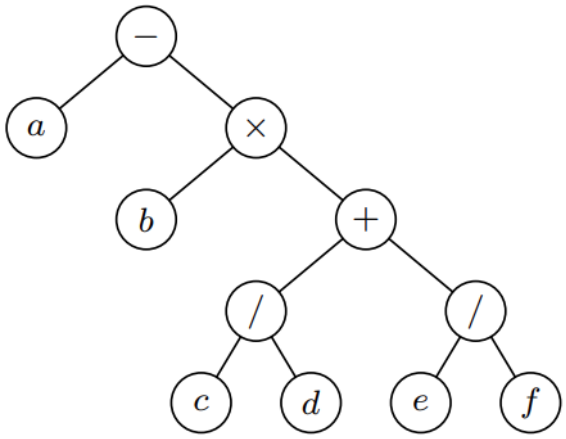

Figura 3.2: Árvore binária correspondente à expressão $a - (b(\frac{c}{d} + \frac{e}{f}))$. Extraída de [4].

3.2 O algoritmo PySR

A popularização da programação genética e, consequentemente, de seu uso como meio para execução da regressão simbólica, se deve em boa parte aos trabalhos de Koza, como seu livro [21] que, além de trazer um panorama geral do estado da arte na época, e descrições detalhadas a respeito da programação genética, seus antecessores e formas de implementação, ainda examina uma série de aplicações, dentre elas a regressão simbólica.

Em teoria, a tarefa da regressão simbólica via algoritmos evolutivos não converge necessariamente com a ideia de gerar um programa de computador, porém, na prática, as ideias avançaram em conjunto. Aplicações práticas da regressão simbólica começam por representar as expressões como estruturas computacionais, tornando o processo, essencialmente, em programação genética, mesmo que em um domínio diferente do inicialmente apresentado por Koza, que pretendia escrever programas em LISP.

Dada a já citada popularidade do trabalho, algumas ideias propostas por Koza em [21], mesmo que não originais do texto, viriam a se tornar padrões em suas respectivas áreas. Dentre as principais para o entendimento do PySR está a representação de expressões matemáticas em árvores binárias (chamadas em alguns textos de árvores unárias-binárias [19]), isto é, árvores com no máximo dois filhos por nó. Neste contexto, árvores são estruturas de dados, porém também podem ser vistas como grafos [111].

A relação entre operadores, variáveis e escalares se dá através das conexões da árvore. A precedência das operações está diretamente relacionada com o nível da árvore: nós em níveis maiores têm precedência sobre nós em níveis menores, sendo assim desnecessário o uso de parênteses na árvore. Um exemplo dessa questão, bem como da representação de expressões em árvores binárias, pode ser visto na Figura 3.2, que mostra uma árvore representando a expressão $a - (b(\frac{c}{d} + \frac{e}{f}))$.

A implementação da geração de populações na biblioteca SymbolicRegression.jl, backend do PySR, é bastante direta. Inicialmente, gera-se um nó com um valor escalar aleatório que, no contexto do algoritmo, representa uma expressão mínima válida. Depois, a depender da complexidade desejada para a população, gera-se operações unárias ou binárias até que ela seja atingida.

Para evitar a geração de uma expressão inválida, as operações são geradas de forma completa: é gerada tanto a operação, quanto os elementos dela, que podem ser tanto valores escalares, quanto variáveis do problema em resolução.

A anexação das operações à árvore se dá por meio da substituição de um nó folha. Todas as decisões são feitas aleatoriamente: qual nó folha sofrerá anexação; se a operação

anexada será unária ou binária; e se os elementos da operação serão escalares ou variáveis do problema.

A não ser que alteradas pelo usuário, todas essas decisões são uniformemente distribuídas, ou seja, têm chance igual de ocorrer. Os escalares, por outro lado, são gerados aleatoriamente a partir de amostras de uma distribuição normal $N(0, 1)$. Este procedimento é repetido até que a população seja preenchida, como dado nos Pseudocódigos 2, 3, 4 e 5, adaptados de [23].

Pseudocódigo 2: Geração de uma população.

Entrada: tamanho da população m; complexidade das expressões c.
Saída: população P de indivíduos.

repetir m **vezes**
  $i \leftarrow$ gerar indivíduo de complexidade c;
  adicionar i em P;
fim

Pseudocódigo 3: Geração de uma expressão aleatória.

Entrada: complexidade do indivíduo c.
Saída: expressão em forma de árvore binária a.

$a \leftarrow$ nó$(N(0, 1))$;
se $c = 1$ **então**
  retorna a;
senão
  enquanto *(*$C(a) < c$*) OU (*$C(a) < c - 1$ *E não há mais operações unárias disponíveis)* **faça**
    adicionar uma operação em a;
  fim
fim

Pseudocódigo 4: Geração de um terminal.

Entrada: variáveis independentes $x_1, \ldots, x_i$ do problema.
Saída: nó com valor terminal v.

decidir aleatoriamente se v é escalar ou variável;
se v *é escalar* **então**
  retorna nó$(N(0, 1))$
fim
se v *é variável* **então**
  retorna nó$(x_j \in x_1, \ldots, x_i)$
fim

Pseudocódigo 5: Adição de uma operação a uma árvore de expressão.

Entrada: árvore a que sofrerá alteração; complexidade c da expressão final.
Saída: árvore a, modificada.

escolher nó folha aleatório k de a para ser substituído;
escolher aleatoriamente se a operação *op* é unária ou binária;
se *op é unária ou* $C(a) = c - 1$ **então**
    $v \leftarrow$ gerar terminal;
    op $\leftarrow$ operação unária aleatória;
    $k \leftarrow op(v)$ em a;
fim
se *op é binária* **então**
    $v_1 \leftarrow$ gerar terminal;
    $v_2 \leftarrow$ gerar terminal;
    op $\leftarrow$ operação binária aleatória;
    $k \leftarrow op(v_1, v_2)$ em a;
fim

Com apenas essa conceituação inicial, já é possível compreender o chamado "laço externo" do PySR (Pseudocódigo 6). Inicialmente, são criadas n_p populações P_i, $i = 1 \ldots, n_p$ de complexidade mínima c_{min}, preenchidas por z expressões aleatórias, definida pelo usuário, e conjuntos M_i associados, para armazenar as expressões mais acuradas vistas em Pi, em cada complexidade.

É criado também um conjunto H, o já mencionado hall da fama, ou fronteira de Pareto, onde são armazenadas as expressões mais acuradas encontradas em toda a execução do algoritmo. Inicia-se então o laço, executado por n_{iter} iterações onde, a cada iteração, uma população é evoluída a partir do laço interno do PySR, caracterizado por "evoluir-simplificar-otimizar" [5]. A evolução das n_p populações ocorre paralelamente, se aproveitando de múltiplos núcleos de processamento da máquina onde o algoritmo está sendo executado, porém ainda com uma contingência para pseudoparalelismo caso não seja possível acomodar todas as populações nos núcleos do processador.

Ao final das iterações, o algoritmo conclui sua execução retornando ao usuário o conjunto H de melhores expressões encontradas.

Pseudocódigo 6: Loop externo do PySR.

Entrada: matriz de *features* X e vetor de *targets* y para regressão.
Saída: expressão em forma de árvore binária a.

para cada $i \in [1, n_p]$ **faça**
    $P_i \leftarrow$ gerar população com z expressões aleatórias de complexidade c_{min};
    criar conjunto vazio M_i;
fim
criar conjunto vazio H;
repetir n_{iter} **vezes**
    para cada $i \in [1, n_p]$ **faça**
        evoluir P_i;
    fim
fim
retorna H apenas com expressões dominantes;

Uma das mudanças propostas pelo PySR em relação a uma abordagem evolucionária clássica é dada pelo controle dinâmico das mutações. Em um algoritmo mais tradicional, a etapa de mutação é sempre aplicada, mesmo que este processo piore a aptidão dos

indivíduos que o sofrem. Uma abordagem mais sofisticada e comum [103] é utilizar uma taxa de mutação, isto é, uma probabilidade fixa de aplicação da mutação.

Contudo, encontrar uma taxa que produza bons resultados não é uma tarefa trivial: se muito baixa, há um incentivo para convergência precoce, onde o algoritmo foca sua busca em um tipo muito específico de expressão, dada a pouca variabilidade decorrente de pouca mutação.

Uma taxa muito alta, todavia, deixaria o processo demasiadamente aleatorizado, prejudicando sua eficiência devido à potencial avaliação de muitas expressões pouco aptas [112, 113]. O PySR propõe uma solução para este dilema através da aplicação da têmpera simulada [97] para definição da taxa de mutação.

O processo de têmpera simulada é um algoritmo de busca local. Trata-se de uma categoria de algoritmos de busca onde o caminho até o objetivo (melhor solução para um determinado problema) é irrelevante [97], em contraste de outros processos, como o algoritmo de Dijkstra ou o A* [97, 114], onde busca-se, além da melhor solução possível para um determinado problema, o melhor caminho até essa solução, como na busca de caminhos mínimos em grafos.

Contudo, em casos onde o espaço de busca é muito grande ou contínuo, não é possível operar com algoritmos que requerem conhecimento prévio dos caminhos possíveis a serem explorados. Os algoritmos de busca local são guiados, desta forma, por um único estado: a solução atual do problema.

O algoritmo de têmpera simulada apresenta uma melhoria em relação a outro algoritmo clássico de busca local: a subida de encosta. A subida de encosta também é conhecida como busca local gulosa, pois simplesmente vai em direção à melhor solução vizinha. Para demonstração do conceito, considera-se um exemplo ilustrativo onde deseja-se encontrar o mínimo global de uma função $f : \mathbb{R} \to \mathbb{R}$.

O algoritmo de subida de encosta, sem quaisquer modificações, inicia seu processamento iterativo com um valor inicial $x = x_0$. Em sequência, são explorados valores vizinhos $x + \beta$ e $x - \beta$, com $\beta \in \mathbb{R}$ fixo [97]. O algoritmo então avalia a função nos pontos vizinhos. Caso um dos vizinhos apresente melhor ajuste ao objetivo, o valor x é substituído por este vizinho. Caso contrário, o algoritmo encerra, com o x atual sendo o resultado, isto é, o mínimo global heuristicamente definido.

Pode-se observar através do ponto destacado na Figura 3.3, no entanto, que o algoritmo de subida de encosta, devido a seu comportamento guloso, falha em encontrar o mínimo global, caindo ao invés disso em um dos mínimos locais da função $f(x) = x^2 - 10\cos(\frac{\pi x}{2}) + 20$. Observa-se, também, que a mudança proposta pela têmpera simulada, com as mesmas condições iniciais, foi suficiente para sanar este problema.

A têmpera simulada propõe uma abordagem gradual e estocástica: ao invés de se explorar o melhor vizinho, explora-se um vizinho aleatório e, mais importante, mesmo que este novo vizinho seja pior para a solução, objetivamente, ainda há uma probabilidade dele ser aceito. Essa probabilidade, guiada por um fator chamado temperatura T, começa alta, e vai decaindo a cada iteração do algoritmo.

Assim, cria-se um algoritmo mais arrojado inicialmente, prevenindo o aprisionamento dos valores em mínimos locais, porém que se estabiliza em valores seguros nas iterações finais. O quão mais adequada for a gradatividade do escalonamento de T, mais próxima será de 1 probabilidade do algoritmo convergir em uma solução ótima global [97]. Esta concepção é inspirada no processo físico de têmpera, na qual um metal é aquecido e, em seguida, resfriado, portanto dando origem ao nome "têmpera simulada". A lógica está disposta nos Pseudocódigos 7 e 8, que trazem o algoritmo de subida de encosta (versão encosta menos íngreme, dado o exemplo fornecido que objetiva mínimos globais) e o algoritmo de têmpera simulada, respectivamente.

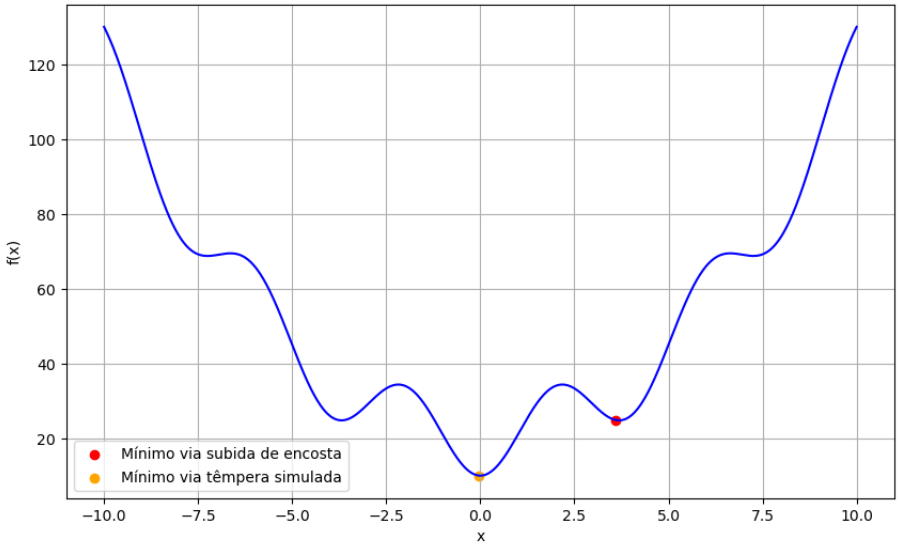

Figura 3.3: Comparação demonstrativa dos algoritmos de subida de encosta e têmpera simulada para busca do mínimo global de uma função $f(x) = x^2 - 10\cos(\frac{\pi x}{2}) + 20$. Elaboração própria.

Pseudocódigo 7: Algoritmo de subida de encosta.

Entrada: o problema a ser resolvido, com um estado inicial definido.
Saída: o estado que melhor se ajusta, localmente, ao objetivo do problema.

```
atual ← estado inicial do problema;
repetir ∞ vezes
    vizinho ← vizinho de atual melhor ajustado ao objetivo;
    se vizinho melhor que atual, então atual ← vizinho;
    senão, retorna atual;
fim
```

Pseudocódigo 8: Algoritmo de têmpera simulada.

Entrada: o problema a ser resolvido, com um estado inicial definido; estratégia de escalonamento; função p_a de probabilidade de aceitação
Saída: o estado que melhor se ajusta, localmente, ao objetivo do problema.

```
atual ← estado inicial do problema;
repetir ∞ vezes
    T ← escalonamento(T);
    se T = 0 então retorna atual;
    candidato ← vizinho aleatório de atual;
    se candidato melhor que atual, então atual ← candidato;
    senão atual ← candidato apenas com probabilidade p_a;
fim
```

Como já descrito, a têmpera simulada é utilizada como instrumento de controle da probabilidade de uma expressão sofrer uma mutação no PySR. Dada uma temperatura T, controlada pelo processo geral de evolução de uma população P, descrito posteriormente, aplica-se uma mutação em uma expressão E, gerando uma expressão E^*, inicialmente uma cópia de E. A modificação via têmpera simulada se dá então por um fator

$$q_{\text{têmpera}} = e^{-\frac{l(E^*)-l(E)}{\alpha T}}, \tag{3.1}$$

onde l é a função *loss* usada para medir a acurácia das expressões descobertas, e α é um fator ajustável pelo usuário para balanceamento da influência de T. No experimento mostrado na Figura 3.3, foi utilizada uma taxa de resfriamento fixa $\alpha = 0.95$ como estratégia de escalonamento, ou seja, uma redução de 5% em T a cada iteração.

Seguindo a discussão, deste modo, com a forma na qual o PySR avalia as soluções, já foi estabelecido pela Equação 2.2 que os métodos de SR usualmente levam em consideração tanto a acurácia em si, utilizando alguma função *loss* clássica do aprendizado de máquina, quanto a complexidade da expressão, a fim de incentivar soluções mais simples. Para o PySR, a complexidade é o número de nós na árvore binária que representa a expressão,

$$C(E) = \text{número de nós em } E, \tag{3.2}$$

e a parte "tradicional" da função *loss* é o erro quadrático médio, dado então por

$$l(E) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{3.3}$$

sendo n o número de amostras na regressão, y_i o valor verdadeiro da amostra i, e $\hat{y}$ o valor predito pelo modelo avaliado.

Todavia, para avaliar as expressões utilizando tanto sua acurácia, quanto sua complexidade, o PySR propõe uma nova abordagem, diferente da tradicional apresentada na Equação 2.2, chamada "parcimônia adaptativa", também aplicada no processo de mutação juntamente à têmpera simulada.

A parcimônia adaptativa leva em conta a frequência e a recência das expressões de uma determinada complexidade. O PySR mantém em memória as expressões mais recentes observadas dentre todas as populações, com este *buffer* sendo atualizado ao final de cada iteração de evolução de uma população.

Caso o acúmulo de expressões ultrapasse 100.000, as mais antigas são apagadas até que a quantidade volte a estabelecida, porém com o cuidado de manter ao menos 1 representante para cada complexidade observada (remove-se alternativamente uma outra expressão, mais recente, de uma complexidade que tenha mais que uma única expressão representante). A parcimônia adaptativa é aplicada nos processos de mutação e seleção. Na mutação, ela surge através de um fator

$$q_{\text{parcimônia}} = \frac{v_{\text{atual}(C(E))}}{v_{\text{atual}(C(E^*))}}, \tag{3.4}$$

onde $v_{\text{atual}(C(E))}$ é a frequência[1] da complexidade da expressão pré-mutação, e $v_{\text{atual}(C(E^*))}$ é a frequência da expressão E^*, isto é, E mutada. É dada assim a probabilidade de aceitação de uma mutação através do fator γ. Caso $\gamma \in [0, 1) \leq q_{\text{têmpera}} \cdot q_{\text{parcimônia}}$, a mutação é aceita. Caso contrário, é rejeitada, e E^* descartada. Neste sentido, é obtida também a versão final da função *loss* padrão[2] do PySR, dada por

$$\mathcal{L}(E) = l(E) \cdot e^{v_{\text{atual}}(C(E))}. \tag{3.5}$$

Há 9 possíveis mutações, com chances de ocorrência balanceadas por pesos $w_1, \ldots, w_{10}$, sendo a $10^{\underline{a}}$ possibilidade a de nenhuma mutação ocorrer. Os pesos padrão estão listados, juntamente com o significado de cada mutação, na Tabela 3.2, baseada em [5, 23]. A probabilidade de escolha para cada uma das mutações é calculada ponderadamente via

$$p_i = \frac{w_i}{\sum_{j=1}^{10} w_j} \tag{3.6}$$

[1]Chamada por Cranmer [5] de "frecência" (*frecency*) por se tratar da frequência regularizada pela recência, como explicado.

[2]Pois o PySR permite customização da função.

permitindo assim o ajuste por parte do usuário de um dos pesos sem a necessidade de rebalancear o restante dos valores. Os valores em si foram definidos empiricamente por Cranmer de forma a provocar variabilidade, porém sem que alterações muito drásticas na forma funcional da expressão sejam excessivamente frequentes [5].

Tabela 3.2: Mutações possíveis no algoritmo PySR.

i	**Operação**	w_i	**Descrição**
1	Mutar constante	0.0353	Um valor escalar aleatório em E^* é multiplicado por um fator a, com $a = (1 + sT + \epsilon)^r$, onde $s \in \mathbb{R}$, $s = 1$ (padrão) é uma constante de perturbação, T é a temperatura atual da têmpera simulada, $\epsilon \in \mathbb{R}$, $\epsilon = 0.1$ (padrão) uma perturbação mínima, e $r \in \mathbb{R}$, $-1 \leq r \leq 1$ é um fator aleatório. Ainda há uma chance de 50% de inversão de a, isto é, de $a \leftarrow -a$.
2	Mutar operador	3.63	Substitui aleatoriamente um operador de E^* por outro operador de mesmo grau.
3	Permutar operandos	0.00608	Troca operandos de lugar em uma operação binária aleatória de E^* sensível a essa mudança (ex.: potenciação $a^b \rightarrow b^a$ ou divisão $\frac{a}{b} \rightarrow \frac{b}{a}$).
4	Rotacionar árvore	1.42	Rotaciona a árvore binária que representa E^*. É escolhido um filho pivô aleatório do nó raiz, e um filho aleatório deste filho (ou seja, um neto da raiz), e uma troca é feita entre o nó raiz e o nó neto. A depender dos nós escolhidos, isto pode representar uma rotação a direita ou a esquerda [115]. Aqui, há uma série de verificações para que a rotação não resulte em uma expressão inválida [23].
5	Adicionar operação	0.0771	Adiciona uma nova operação aleatória (Pseudocódigo 5) a E^*. A nova operação pode ser prefixada (adicionada antes da raiz da árvore, com o operador tornando-se a nova raiz), ou sufixada (nó folha aleatório escolhido para sofrer a sufixação). A chance de prefixação ou sufixação é aleatória, porém igual (50%). Em caso de prefixação, a raiz se torna operando da nova operação, com um novo operando sendo gerado apenas em caso de uma operação binária.
6	Inserir operação	2.44	Um único nó interno (não raiz e não folha) aleatório de E^* é envolto em uma nova operação, na qual ele é um dos operandos. Em caso de operação binária, um novo operando é criado (Pseudocódigo 5).

Continua na próxima página

Continuação da Tabela 3.2 — Mutações possíveis no algoritmo PySR

i	**Operação**	w_i	**Descrição**
7	Apagar sub-árvore	0.369	Um nó aleatório qualquer de E^* é escolhido. Se o nó é folha, ele é apagado e substituído por um novo nó folha (Pseudocódigo 5). Se o nó é operador unário, ele é substituído por seu filho. Se o nó é operador binário, ele é substituído por um de seus filhos (escolha aleatória com chance igual para ambos).
8	Simplificar expressão	0.00148	Simplifica E^* através de equivalências matemáticas, de acordo com um algoritmo de [100]. Exemplo: $x^2 - x^2 \to 0$. Essa mutação independe da simplificação proposta pelo laço de evolução, descrito na sequência deste texto.
9	Nova árvore aleatória	0.00695	Gera uma expressão completamente nova, porém de complexidade igual à de E^*.
10	Nenhuma mutação	0.431	

Uma última característica do PySR que afeta a mutação é a possibilidade de definição de restrições às expressões descobertas. A primeira delas, imposta naturalmente em alguns sublaços do algoritmo, mas que pode ser definida à força pelo usuário, é a complexidade máxima (quantidade de nós na árvore) de uma expressão. Pode-se também restringir:

- a profundidade máxima de uma expressão dada pela altura/profundidade da árvore que a representa, muito útil quando deseja-se evitar aninhamento excessivo de expressões.

- a complexidade máxima de sub-expressões por operador. Por exemplo [5], definir as complexidades máximas do operador de potência "^" como $(-1, 3)$ significa que, em uma expressão ou sub-expressão $S = A^B$, onde A e B são sub-expressões de S, A pode ter qualquer complexidade ("qualquer" denotado por -1), porém B tem complexidade máxima de 3. Este é um mecanismo importante para casos onde é imprescindível o controle sobre a complexidade das expressões.

- o número de aninhamentos utilizando operadores específicos. Trata-se de uma estrutura de microgerenciamento bastante específica onde, por exemplo, a instrução `sin: { sin: 0, cos: 1}` proíbe aninhamentos do tipo $\sin(\sin(x))$, porém permite um único aninhamento do tipo $\sin(\cos(x))$. Neste mesmo caso, $\sin(\cos(\cos(x)))$ já não seria permitido, dado o limitante 1 para este tipo de aninhamento.

Caso qualquer restrição estabelecida seja desrespeitada, o algoritmo faz outra mutação, seguindo as possibilidades da Tabela 3.2. Estas restrições também são aplicadas no processo de recombinação, explorado posteriormente neste texto, com o mesmo sendo refeito caso a verificação falhe. Estabelece-se, assim, a lógica de mutação do PySR, disposta no Pseudocódigo 9.

Resultados empíricos apresentados por Cranmer [5] mostram que a parcimônia adaptativa aumentou a diversidade das expressões exploradas pelo algoritmo, afinal, há um incentivo para que diversas complexidades sejam analisadas, encontrando assim expressões menos complexas, porém que apresentam uma acurácia reduzida, e expressões mais acuradas, porém mais complexas, prevenindo um hiperfoco precoce do algoritmo em algumas poucas formas funcionais que apresentem uma acurácia minimamente satisfatória.

Ao mesmo tempo, a aplicação em conjunto da têmpera simulada aparenta ter reduzido consideravelmente o tempo de convergência do algoritmo, dado que o comportamento mais explorador é desencorajado ao longo do tempo, sendo mais presente nas primeiras iterações, onde há maior risco de convergência precoce. Em suma, no processo evolutivo, há um incentivo inicial de variabilidade populacional, mas previne-se explorações desnecessárias nos estágios mais avançados do algoritmo, visto que provavelmente as melhores formas funcionais já foram encontradas por meio do processo evolutivo.

Pseudocódigo 9: Mutação de uma expressão no PySR.

Entrada: expressão E.
Saída: expressão mutada E^*.

$i \leftarrow$ índice de mutação aleatória g_i com chance ponderada w_i de escolha;
$E^* \leftarrow g_i(E)$;
$q_{\text{têmpera}} \leftarrow e^{-\frac{l(E^*)-l(E)}{\alpha T}}$;
$q_{\text{parcimônia}} \leftarrow \frac{v_{\text{atual}}(C(E))}{v_{\text{atual}}(C(E^*))}$;
se *mutação com chance* $q_{têmpera} \cdot q_{parcimônia}$ **então**
    retorna E^*;
fim
retorna E;

Outro mecanismo central do processo evolutivo é o critério para seleção de indivíduos, aplicado tanto para escolha de quais membros da população sofrerão mutação, quanto para decidir quais pares de expressões sofrerão recombinação. O PySR utiliza um método comum [21] para seleção: o torneio.

No torneio, uma amostra de $n_s = 12$ expressões participantes é selecionada aleatoriamente, uma das principais características desta abordagem, pois a torna menos tendenciosa em relação aos indivíduos mais aptos globalmente [103]. A partir dessa seleção, inicia-se a competição, até que reste apenas uma expressão: a cada iteração $i = 0, \ldots, n_s - 1$, seleciona-se a melhor expressão dentre $n_s - i$ disponíveis, tendo ela $p_{\text{torneio}} = 0.9$ de chance de vencer.

Caso vença, o torneio se encerra, mas caso perca, essa expressão é removida do conjunto Q de expressões disponíveis, havendo assim uma chance, mesmo que muito pequena, de que as piores expressões selecionadas sobrevivam. Esta lógica está disposta nos Pseudocódigos 11, acerca do torneio, e 10, detalhando a lógica de avaliação de aptidão das expressões apresentada.

Pseudocódigo 10: Seleção de indivíduo mais apto no PySR.

Entrada: população P de expressões.
Saída: uma expressão vencedora E.

$\mathcal{L}_{\text{melhor}} \leftarrow \infty$;
$E_{\text{melhor}} \leftarrow$ primeira expressão de P;
para cada $E \in P$ **faça**
    se $\mathcal{L}(E) < \mathcal{L}_{melhor}$ **então**
        $\mathcal{L}_{\text{melhor}} \leftarrow \mathcal{L}(E)$;
        $E_{\text{melhor}} \leftarrow E$;
    fim
fim
retorna E_{melhor};

Pseudocódigo 11: Torneio de expressões no PySR.

Entrada: população P de expressões.
Saída: uma expressão vencedora E.

$Q \leftarrow$ conjunto de n_s expressões aleatoriamente selecionadas de P;
se $n_s = 1$ **então**
  retorna única expressão de Q;
fim
enquanto *tamanho(Q) > 1* **faça**
  $E \leftarrow$ expressão mais apta de Q;
  se E *é vencedora com chance* $p_{torneio}$ **então**
    quebrar laço;
  fim
  remover E de Q;
fim
retorna E;

A lógica de recombinação do PySR, baseada no cruzamento de subárvores, é bastante simples e está disposta no Pseudoalgoritmo 12, bem como na Figura 3.4. Duas expressões pais selecionadas e distintas E_1 e E_2 são clonadas, criando expressões E_1^* e E_2^*. Então, nós aleatórios são escolhidos em E_1^* e E_2^*, e tornam-se a raiz de subárvores A_1 e A_2, bem como pontos de corte para o cruzamento entre os indivíduos. A_1 é substituída por A_2 em E_1^*, e A_2 é substituída por A_1 em E_2^*, finalizando assim o cruzamento, com o surgimento de duas novas expressões.

Pseudocódigo 12: Recombinação entre indivíduos no PySR.

Entrada: expressões pais E_1 e E_2.
Saída: expressões filhas E_1^* e E_2^*.

$E_1^*, E_2^* \leftarrow$ cópias de E_1, E_2, respectivamente;
$o_1, o_2 \leftarrow$ nós aleatórios de E_1^*, E_2^*, respectivamente;
$A_1, A_2 \leftarrow$ subárvores a partir de o_1, o_2, respectivamente;
trocar A_1 por A_2;
retorna E_1^*, E_2^*;

Este amálgama de técnicas culmina no chamado "laço interno" do PySR, no qual o processo evolutivo é construído. Para cada população em $i = 1, \ldots, n_p$, padrão $n_p = 40$, são executadas $k = 1, \ldots, n_c$, padrão $n_c = 300000$, iterações de evolução, em cada qual há uma chance $p_{\text{rec}} = 0.01$ de recombinação, caso contrário haverá mutação.

No caso de recombinação, são selecionadas duas expressões via torneio, e a reprodução entre elas dá origem a novas expressões E_1^* e E_2^*, como visto anteriormente. E_1^* e E_2^* então substituem as duas expressões mais antigas em P_i.

No caso de mutação, uma expressão candidata E é escolhida via torneio, e sua versão mutada $E*$, com temperatura de têmpera simulada $T = 1 - \frac{k}{n_c}$ (nota-se que a temperatura decai ao longo do processo iterativo, como dita o processo de têmpera) substitui a expressão mais antiga em P_i.

Ressalta-se que, inerente ao processo de mutação é a possibilidade de rejeição, neste caso $E^* = E$. Então, para cada expressão em P_i, é aplicado um processo de "simplificação e otimização". A simplificação se dá no mesmo sentido da mutação de simplificação mostrada na Tabela 3.2: a expressão E em questão é simplificada por meio de equivalências matemáticas.

Isto só é feito após as iterações de mutação pois objetiva-se utilizar algumas expressões não simplificadas como intermédio para descoberta de outras a exemplo de [5], $x_1 x_1 -$

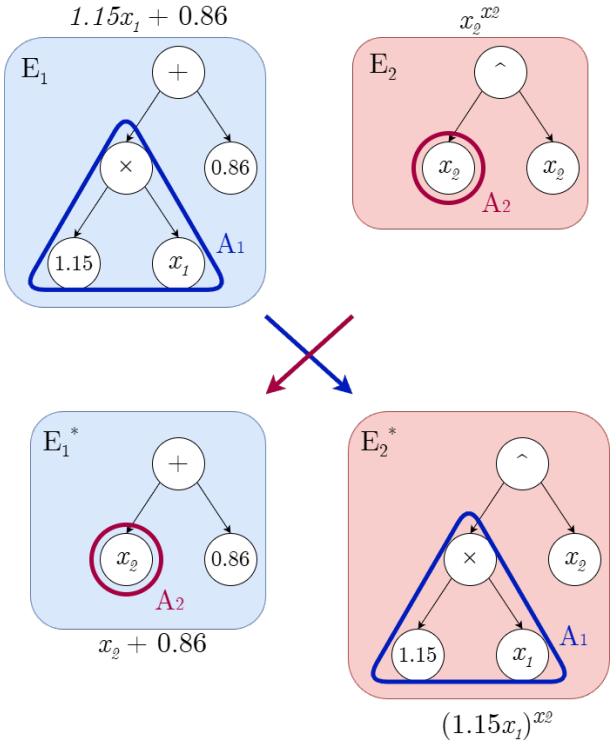

Figura 3.4: Exemplo de recombinação no PySR via cruzamento de duas expressões $E_1 = 1.15x_1 + 0.86$ e $E2 = x_2^{x_2}$. Baseada em [5].

x_1x_1, que claramente pode ser simplificada para zero, mas pode ser usada para descobrir $x_1x_1 - x_1x_2$ via mutação.

Aplica-se então uma otimização de constantes via minimização de $l(E)$ através de algumas iterações do algoritmo BFGS [116], ou alternativamente por algum outro algoritmo clássico de otimização disponível na biblioteca Optim.jl [117].

A cada iteração evolutiva $k \in [1, n_c]$ de P_i, é atualizada a lista M_i de expressões mais acuradas vistas na população em questão (vide Pseudocódigo 6) para cada complexidade, isto é, apenas um representante de cada complexidade é mantido em M_i. Ao final do processo de otimização de escalares, atualiza-se também o hall da fama H, que mantém a expressão geral mais acurada, dentre todas as populações para cada complexidade, ou seja, as expressões mais acuradas em $M_i \cup H$.

Por fim, o PySR aplica estocasticamente um fenômeno de migração, ideia que advém de alguns trabalhos anteriores [118, 119], e que consiste basicamente em, com alguma frequência, transportar algumas expressões de uma "ilha" (população P_i) para outra, provocando maior variabilidade genética entre as populações, de forma similar a fenômenos de migração de espécies no mundo real.

No PySR, para cada expressão E de uma população, há chances concorrentes $\alpha_H = 0.05$ e $\alpha_M = 0.05$ de que E seja substituída por uma expressão aleatória vinda do hall da fama global H ou do ranking de melhores expressões de uma outra população $M_{j \neq i}$, respectivamente. A substituição por H toma precedência sobre a por M_j, pois ocorre posteriormente, como pode ser visto no Pseudocódigo 13, que dispõe o laço interno evolutivo do PySR.

Pseudocódigo 13: Laço interno do PySR.

Entrada: população P_i.
Saída: população P_i, evoluída.

para cada $k \in [1, n_c]$ **faça**
  se *recombinação com chance* p_{rec} **então**
    $E_1 \leftarrow$ resultado de um torneio em P_i;
    $E_2 \leftarrow$ resultado de um torneio em P_i;
    $E_1^*, E_2^* \leftarrow$ resultado do cruzamento entre E_1 e E_2;
    substituir as duas expressões mais velhas de P_i por E_1^*, E_2^*;
  fim
  senão
    $E \leftarrow$ resultado de um torneio em P_i;
    $T \leftarrow 1 - \frac{k}{n_c}$;
    $E^* \leftarrow$ mutação de E com temperatura T;
    substituir a expressão mais velha de P_i por E^*;
  fim
fim
para cada $E \in P_i$ **faça**
  simplificar E;
  otimizar constantes de E;
  atualizar E em P;
fim
$M_i \leftarrow$ melhor expressão de P_i para cada complexidade;
$H \leftarrow$ melhor expressão de $M_i \cup H$ para cada complexidade;
para cada $E \in P_i$ **faça**
  se *substituição com chance* α_H **então**
    substituir E em Pi por uma expressão aleatória de H;
  fim
  se *substituição com chance* α_M **então**
    substituir E em Pi por uma expressão aleatória de $M_{j \neq i}$;
  fim
fim

Capítulo

4

Regressão simbólica via transformers

Os transformers compõem atualmente o estado da arte para tarefas de processamento de linguagem natural (NLP, de *natural language processing*), tais como tradução [120] e geração de texto [121]. Desde sua proposta original por Vaswani et al. [9], a arquitetura e seus vários desdobramentos [121–124] vêm consistentemente estabelecendo resultados excelentes em tarefas linguísticas, ao passo em que seu tempo de inferência é muito baixo, dada a alta capacidade de paralelização das operações matriciais que a compõem [125].

Com sua eficácia nesses casos, surgiram tentativas de aplicar a arquitetura para cumprir a tarefa da regressão simbólica. Tais esforços são motivados principalmente [10,66,71] por uma busca por maior eficiência computacional: enquanto os métodos tradicionais de programação genética iniciam do zero seu processo de inferência a cada novo problema analisado, tornando lenta a experimentação, um modelo transformer pré-treinado pode inferir expressões em um tempo significativamente menor.

Este capítulo dedica-se a investigar uma dessas tentativas, o modelo E2E [10]. A escolha justifica-se pela boa performance do TPSR, que utiliza o E2E e o NeSymRes [66] como base, em uma das competições do SRBench [62], ao passo em que é mais recente que o NeSymRes.

Antes de tratar sobre o E2E, especificamente, disserta-se a respeito de como o mecanismo de atenção, base dos transformers, surge no contexto das redes neurais recorrentes, originalmente como uma extensão da arquitetura encoder-decoder proposta por Bahdanau et al. [126] para a tradução de máquina.

Posteriormente, também é explorado o transformer original [9], cuja arquitetura é muito similar à do E2E e, portanto, esclarecedora na compreensão do mesmo.

4.1 Modelos linguísticos e o problema do contexto

Redes neurais recorrentes (RNNs, de *recurrent neural networks*) [127] podem ser entendidas como uma arquitetura especializada na modelagem de sequências [6]. Em comparação com uma rede feedforward tradicional, cuja camada de entrada é disposta em um único vetor x, a entrada de uma RNN é dada por uma sequência de vetores $x^{(1)}, \ldots, x^{(T)}$.

Um dos principais diferenciais das RNNs é o compartilhamento de parâmetros: enquanto em uma FNN cada neurônio da camada inicial possui um peso associado a si, em uma RNN os pesos são compartilhados entre vários passos da sequência. A ideia central é que, ao invés de construir a rede como uma composição de combinações lineares, onde a tarefa da mesma é aprender os escalares destas combinações, uma RNN parte do princípio

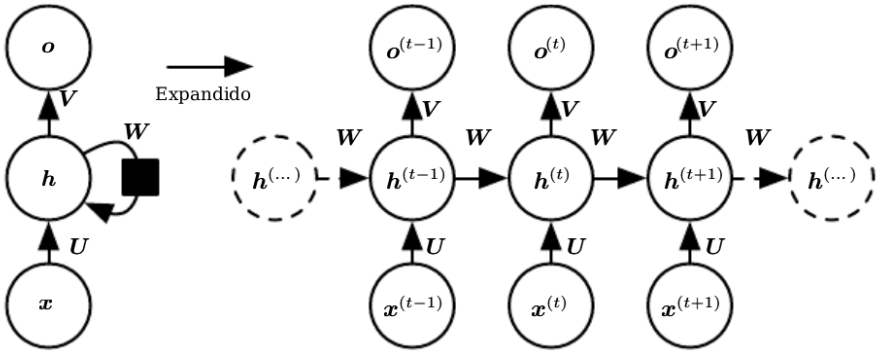

Figura 4.1: Exemplo simplificado (à esquerda) e expandido (à direita) de uma rede neural recorrente e suas principais matrizes de pesos otimizadas durante o processo de treinamento. Adaptada de [6].

que cada passo da sequência pode ser representado por um sistema dinâmico onde cada estado $s^{(t)}$ no passo t depende do estado anterior $s^{(t-1)}$, e pode ser calculado por meio de uma função de transição f:

$$s^{(t)} = f(s^{(t-1)}; \theta),$$

onde θ são os pesos da rede, escalares arbitrários a serem aprendidos por meio de um processo de otimização. Percebe-se então diretamente a aplicação na modelagem de fenômenos sequenciais, tais quais geração e tradução de textos, sequências de palavras, e séries temporais, estas que induzem o uso da notação t para um passo da sequência. Decorre também o nome dado a redes deste tipo, onde "recorrente" indica justamente o uso de um mesmo estado na representação de estados subsequentes: $s^{(t)}$ é dado em função de $s^{(t-1)}$, que é dado em função de $s^{(t-2)}$, e assim por diante.

A partir deste conceito, é possível definir as unidades ocultas, nome dado especificamente aos neurônios da camada oculta de uma RNN, mais especificamente, o chamado "estado oculto" carregado por estas unidades

$$h^{(t)} = f(h^{(t-1)}, x^{(t)}; \theta). \tag{4.1}$$

A arquitetura RNN está ilustrada na Figura 4.1. Nela, é possível observar o resultado do processo de treinamento, isto é, os pesos aprendidos pela rede, bem como as conexões diretas e recorrentes, detalhadas anteriormente.

A tarefa de uma RNN se dá portanto em aprender os pesos genericamente representados por θ na Equação 4.1, estabelecendo assim

$$\begin{aligned} h^{(t)} &= a\left(b + Wh^{(t-1)} + Ux^{(t)}\right) \\ o^{(t)} &= c + Vh^{(t)} \end{aligned}$$

onde a é uma função de ativação não linear, como tanh ou ReLU; b e c são vieses; U é a matriz de pesos entre a camada de entrada e a camada oculta; W é a matriz de pesos da camada oculta com si mesma (recorrências); e V é a matriz de pesos entre a camada oculta e a camada de saída. Em uma RNN comum [6, 127], os pesos são otimizados por meio de uma modificação do algoritmo backpropagation, o *backpropagation through time* (BPTT).

Decorrente destas noções, pesquisadores do campo de processamento de linguagem natural (NLP, de natural language processing) desenvolveram a família de redes neurais sequence-to-sequence (seq2seq) com o objetivo de efetuar traduções entre diferentes idiomas [128, 129].

Neste tipo de modelagem, a rede aprende a receber uma sequência de palavras como entrada e devolver outra como saída, neste exemplo, representando a tradução da frase. Para representar as sentenças numericamente, é criado um vocabulário onde palavras e símbolos (ou pequenos agrupamentos destes) são mapeados em números inteiros, compondo um vetor $x = (x_1, ..., x_T), x \in \mathbb{Z}^T$.

Estas unidades de uma sequência são conhecidas como "tokens". É usual [130] a existência de um token representando o fim de uma sequência, bem como um token indicando um espaço vazio, pelo simples fato de que a rede deve permitir tradução de frases de tamanhos distintos, porém representados em uma mesma dimensionalidade vetorial, como mostra a Figura 4.2, adaptada de [130], onde $\langle \text{EOS} \rangle = 3$ (fim de frase) e $\langle \text{PAD} \rangle = 4$ (espaço em branco), e o tamanho máximo do vetor é $T = 5$.

Essa não é a única forma de representar uma frase numericamente, havendo múltiplas alternativas, tais como representar cada letra como um inteiro e, portanto, cada palavra como um vetor e cada frase como uma matriz [131]; ou, ainda, treinar uma outra rede neural com o único objetivo de transformar palavras ou trechos de frases em um vetor de números inteiros ou reais [122].

```
"go ."          → [1, 2, 3, 4, 4]   ← preenchido com <PAD>
"i lost ."      → [5, 6, 2, 3, 4]   ← preenchido com <PAD>
"he's calm ."   → [9, 2, 10, 2, 3] ← sem preenchimento
```

Figura 4.2: Processo de tokenização para representação de um texto em um vetor de números inteiros. Elaboração própria.

A abordagem tradicional para cumprir esta tarefa é utilizar duas redes neurais recorrentes, usualmente do tipo LSTM [132] ou GRU [128], uma agindo como encoder e a outra como decoder.

A tarefa do encoder é aprender a representar a sequência recebida na camada de entrada como um vetor $c \in \mathbb{R}^H$, comumente chamado de "variável de contexto", onde H é a dimensionalidade da camada oculta da rede (quantidade de unidades ocultas em cada estado),

$$c = q(h^{(1)}, \ldots, h^{(T)}),$$

e onde q, em casos mais simples, meramente retorna o último estado oculto $h^{(T)}$ ou, ainda, é um outro tipo de função, como média entre estados ocultos, média ponderada, *max pooling* [130] e até mesmo uma outra rede neural.

Mesmo que, na camada de entrada, já houvesse uma representação numérica via tokens para o texto inserido, o processo de encoding é essencial ao funcionamento do seq2seq.

O contexto não apenas considera o valor de cada token isoladamente, mas a sequência e contexto como um todo, dado o funcionamento natural de uma RNN, isto é, a dependência recorrente dos estados.

Neste sentido, a rede encoder é treinada juntamente com a decoder para melhor otimizar a representação da frase original na tarefa de tradução para o idioma de saída. Esta representação aprendida pelas redes em seus espaços latentes, tanto encoder para a frase original, quanto no decoder para a frase de saída, é chamada de "embedding".

Em outros contextos, pode haver um método exclusivamente dedicado à geração do embedding, como ocorre nos transformers [9]. Caberia então ao encoder criar a melhor representação possível do encadeamento de embeddings para a rede decoder.

A tarefa da rede decoder decorre deste processo: ler a representação c produzida pelo encoder e produzir a frase traduzida no idioma alvo desejado, neste exemplo, também representada através de um vetor de números inteiros. Dada a variável de contexto c, e

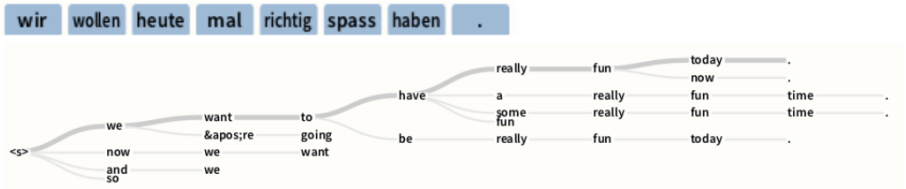

Figura 4.3: Visualização do fluxo preditivo realizado por um modelo seq2seq na tradução de uma frase do Alemão para o Inglês. Em azul, vê-se a frase codificada pelo encoder. No fluxo abaixo, vê-se os tokens preditos a cada passo, ordenados verticalmente do mais provável ao menos provável. Adaptado de [7].

seja $y_1, ..., y_T$ a sequência traduzida esperada, o decoder calcula, a cada passo t', as probabilidades de ocorrência de cada *token* do vocabulário de saída, isto é, $P(y_{t'+1}|y_1, ..., y_{t'})$, como visto na Figura 4.3, que ilustra o fluxo de um modelo seq2seq. O cálculo é feito a partir da aplicação de uma função softmax [130] na camada de saída. O token mais provável é então escolhido como $y_{t'+1}$ para predizer a sequência, e é alimentado de volta na rede, que prossegue com o cálculo de $y_{t'+2}$, e assim sucessivamente até o token de fim de sequência ser predito.

Uma fragilidade conhecida [128] de modelos seq2seq está na representação da frase de entrada em um único vetor c de tamanho fixo H. A compressão de toda a semântica do texto nesta única variável nem sempre é plenamente possível, como na tradução de textos muito longos e, especialmente, na tradução de textos maiores do que os vistos pela rede em seu treinamento [126].

Visando solucionar esse problema, Bahdanau et al. [126] propuseram uma extensão da arquitetura encoder-decoder padrão por meio de um mecanismo de alinhamento suave.

4.2 Redes neurais recorrentes com mecanismo de atenção

A ideia de alinhamento para tradução é mapear quais os tokens mais importantes da sequência de origem para tradução da sequência final. Se feito de forma rígida, o mecanismo apontaria diretamente para um único token de origem. Contudo, a proposta de Bahdanau et al. é de um alinhamento suave, onde o modelo aprende a conferir um grau de importância, quantificado em um número real, a cada um dos tokens da frase original, para cada um dos tokens traduzidos.

Em seu texto, Bahdanau et al. comparam este alinhamento com o conceito de atenção: "...implementa um mecanismo de atenção. O decoder decide em quais partes da sentença original ele deve prestar atenção". Por esta razão, o nome atenção tornou-se comum para se referir ao mecanismo de alinhamento suave [9].

Para implementar este mecanismo, a primeira mudança substancial está na variável de contexto c, a maior fragilidade do modelo seq2seq tradicional. Ao invés dela, o modelo com alinhamento suave propõe uma sequência de vetores c_i, $i = 1, ..., T$, cada qual associado a um token de origem $(x_1, ..., x_{T_x})$. Analogamente, o encoder agora carrega T_x estados ocultos, que recebem nesta circunstância o nome de "anotações", $(h_1, ..., h_{T_x})$. A arquitetura proposta para o encoder, detalhada a seguir, induz que o i-ésimo estado oculto carregue, principalmente, informação dos tokens vizinhos do i-ésimo token de entrada.

Define-se então o vetor de contexto c_i por meio de uma média ponderada das anotações,

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j, \tag{4.2}$$

onde α_{ij} é o peso da j-ésima anotação no i-ésimo token de entrada,

$$\alpha_{ij} = \frac{e^{\beta_{ij}}}{\sum_{k=1}^{T_x} e^{\beta_{ik}}}. \tag{4.3}$$

Na Equação 4.3, β representa o cálculo resultante da aplicação do modelo a de atenção,

$$\beta_{ij} = a(s_{i-1}, h_j), \tag{4.4}$$

onde s_i é um estado oculto da rede decoder. A rede decoder prossegue então da mesma forma que no seq2seq tradicional, produzindo predições $y'_1, ..., y'_{T'}$ por meio de uma abordagem probabilística. A principal diferença nesta parte jaz, deste modo, na entrada da rede, que agora recebe uma sequência de vetores de contexto $(c_1, ..., c_T)$.

Na parte encoder do modelo, contudo, há uma maior diferença, pois o mesmo conta não só com uma, mas duas redes neurais recorrentes. Trata-se de uma BiRNN [130], cujo uso é motivado pelo objetivo de que os estados ocultos da rede carreguem não somente as informações atuais e passadas, como também as futuras. A BiRNN é composta por uma rede progressiva, que lê a sequência entrada na sua forma original $(x_1, ..., x_{T_x})$ e produz os estados ocultos progressivos $(\overrightarrow{h_1}, ..., \overrightarrow{h_{T_x}})$; e uma rede regressiva, que lê a sequência de forma reversa $(x_{T_x}, ..., x_1)$, e portanto produz estados ocultos regressivos $(\overleftarrow{h_1}, ..., \overleftarrow{h_T})$.

Os estados oculto definitivos, isto é, as anotações h_j utilizadas para cálculo dos vetores de contexto c_i são simplesmente uma concatenação vetorial dos equivalentes estados progressivo e regressivo, respectivamente,

$$h_j = [\overrightarrow{h_j}; \overleftarrow{h_j}]. \tag{4.5}$$

A Figura 4.4 ilustra o treinamento do modelo, com os estados progressivos e regressivos, mecanismos de atenção, estados do decoder e saídas esperadas $(y_1, ..., y_{T'})$.

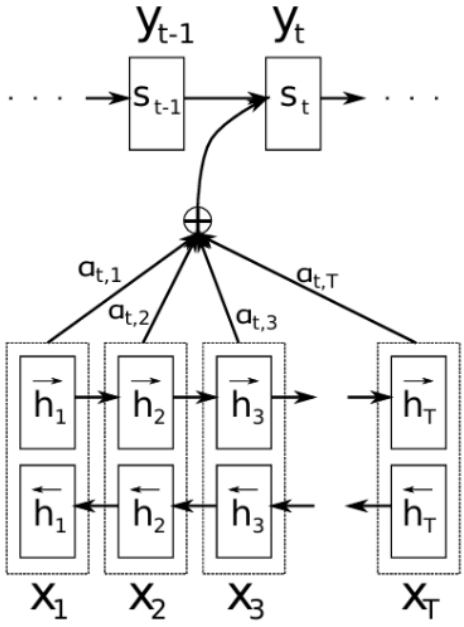

Figura 4.4: Modelo encoder-decoder (seq2seq) com encoder bidirecional e mecanismo de atenção. Adaptado de [6].

O modelo de alinhamento, isto é, o mecanismo de atenção, é uma simples rede neural feedforward com uma única camada oculta:

$$a(s_{i-1}, h_j) = v_a^\top \tanh(W_a s_{i-1} + U_a h_j), \tag{4.6}$$

sendo $W_a \in \mathbb{R}^{n \times n}$ e $U_a \in \mathbb{R}^{n \times 2n}$ as matrizes de peso associadas às entradas vindas do decoder (s_{i-1}) e do encoder (h_j), respectivamente; e $v_a \in \mathbb{R}^n$ é o vetor de pesos associado à camada de saída.

Esta escolha é justificada pelo fato de que cada par (x_i, y_i), $x_i \in \mathbb{R}^T$, $y_i \in \mathbb{R}^{T'}$ de frases de entrada e saída precisa realizar $T \times T'$ cálculos de alinhamento, então é ideal manter a rede a o mais simples possível. No trabalho original [126] as redes foram treinadas utilizando os algoritmos SGD e Adadelta em conjunto [133].

4.3 A arquitetura transformer

A partir da proposta de Bahdanau et al., e de trabalhos subsequentes explorando mecanismos de atenção, de diferentes formas, em redes neurais recorrentes, surge a ideia central à arquitetura de redes neurais transformer [9]: se o mecanismo de atenção for capaz de representar suficientemente bem o contexto carregado por cada parte de uma sequência (ex.: palavra em uma frase), então a rede neural recorrente não é necessária. Este conceito dá nome ao trabalho que o propõe [9]: "*Attention Is All You Need*", "você só precisa de atenção", em tradução livre. Novamente, essas ideias surgem na área de processamento de linguagem natural, objetivando o avanço do estado da arte da tradução de máquina.

Este esforço vem de dois fortes motivadores numérico-computacionais [9, 134]:

- A distância linear entre estados: dada a natureza composta dos estados ocultos de uma RNN, estados distantes $h^{(i)}, h^{(i+n)}$, interagem somente após n passos. Dependências distantes entre palavras podem ser, deste modo, difíceis de representar devido ao problema do desaparecimento de gradientes, mesmo em redes LSTM e GRU. Ademais, no contexto isto induz uma representação linear de frases, o que não é ideal para NLP, onde diferentes idiomas possuem estruturas de ordem distintas;

- A falta de paralelismo: só é possível computar estados futuros $h^{(i+1)}$ uma vez que todos os estados passados $h^{(i)}, h^{(i-1)}, \ldots, h^{(1)}$ foram computados, o que restringe muito o tempo de treinamento e inferência dos modelos recorrentes.

Vaswani et al. [9] propõem uma substituição do mecanismo de recorrência por um de auto-atenção. Ao invés de modelar cada estado uma relação de dependência $h^{(t)} \circ h^{(t-1)} \circ \ldots \circ h^{(1)}$, agora cada vetor dos espaços latentes produzidos pela rede tem uma relação quantificada com todos os vetores do espaço, dentro de uma mesma camada, incluindo a si próprio.

Essa noção está ilustrada na Figura 4.5 onde é mostrado que cada palavra de uma frase tem uma relação mais ou menos forte com palavras distintas da mesma frase, sendo esta a auto-atenção. Apesar dessa ser precisamente a intuição desejada pelos pesquisadores de NLP ao propor este mecanismo, ressalta-se que a auto-atenção é aplicada no espaço latente da rede neural, onde não necessariamente cada membro de um vetor representa uma palavra, dado que os embeddings são aprendidos durante o treinamento, como já estabelecido anteriormente.

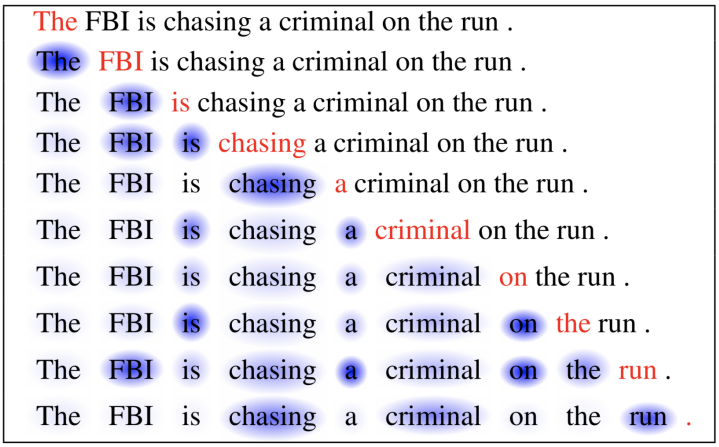

Figura 4.5: Exemplo de atenção (gradiente azul) para cada token (em vermelho) de uma frase. Extraída de [8].

Mesmo que muitíssimo similar conceitualmente à atenção de Bahdanau et al. [126], a auto-atenção difere-se pelo fato de ser aplicada em um mesmo espaço, e não apenas entre o encoder e o decoder.

A implementação proposta por [9] também é significativamente distinta, em principal devido a não precisar mais ater-se a uma lógica recorrente. Considere vetores exemplo $x_1 \in \mathbb{R}^{d_m}$ e $x_2 \in \mathbb{R}^{d_m}$, e que desejamos calcular a atenção que x_1 deve prestar em x_2, e d_m é a dimensionalidade do modelo transformer, isto é, a dimensionalidade do espaço latente do mesmo[1]. Projetamos linearmente o vetor x_1 em um query $q_1 \in \mathbb{R}^{d_k}$, $d_k < d_m$,

$$q_1 = x_1 W_Q, \tag{4.7}$$

onde $W_Q \in \mathbb{R}^{d_m \times d_k}$ é a matriz de pesos de query, aprendida pelo modelo. Conceitualmente, o objetivo do query é representar o tipo de conteúdo mais relevante para x_1, em outras palavras, "o que x_1 busca" [130].

Calculamos então o vetor "chave" (key) $k_2 = x_2 W_K$, onde, novamente, $W_K \in \mathbb{R}^{d_m \times d_k}$ é uma matriz de pesos, e o vetor chave objetiva quantificar o contexto oferecido pelo vetor projetado (no caso, x_2). Quantificamos então a intensidade da atenção de x_1 em x_2, $c_{1\to 2}$, pelo simples produto escalar entre os vetores query e chave, normalizado pela raiz da dimensão vetorial de ambos, $\sqrt{d_k}$,

$$c_{1\to 2} = \frac{q_1^\top k_2}{\sqrt{d_k}}, \tag{4.8}$$

processo denominado de atenção por produto escalar escalonado. Existem ainda outras formas de quantificar atenção [9], porém opta-se pelo produto escalar por motivos de performance computacional.

Considerando ainda este exemplo, se x_1 e x_2 forem suficientemente compatíveis, então x_2 deve contribuir significativamente na representação contextual de x_1. A informação carregada pelo vetor x_2 é quantificada por um vetor valor, v_2, também resultante de uma projeção linear em uma matriz $W_V \in \mathbb{R}^{d_m \times d_v}$ aprendida pelo modelo,

$$v_2 = x_2 W_V. \tag{4.9}$$

Expandindo a partir deste exemplo, como visto anteriormente, a atenção de um vetor x aos vetores de uma determinada camada, incluindo a si próprio, é computada de forma

[1] Em [9], todos os diferentes espaços de embedding usam uma mesma dimensionalidade $d_m = 512$.

completa. Isto é feito através da compactação dos vetores chave em uma matriz K, onde cada chave é uma linha, e de seus respectivos valores em uma matriz V onde, analogamente, cada valor é uma linha. Trata-se de uma versão suavizada do conceito de chave-valor visto classicamente na computação, aplicado em hashmaps e bancos de dados [135].

Para maior eficiência, os queries também são compactados em linhas em uma matriz Q, e todo "bloco" de atenção é calculado de uma só vez. Desta forma, é possível entender a razão dos transformers conhecidamente serem muito performáticos quando processados em GPUs [125], dada a alta capacidade de paralelização destas operações matriciais.

No cálculo matricial da atenção, é aplicada uma ativação softmax, de forma a controlar a estabilidade numérica da rede. Temos então o cálculo de atenção como proposto por Vaswani et al.:

$$a(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V. \tag{4.10}$$

Adicionalmente, considerando que a arquitetura transformer ainda carrega uma estrutura de encoder-decoder, é aplicado um mascaramento na auto-atenção da camada decoder. O mascaramento visa impedir que exista atenção entre um vetor anterior e um seguinte no decoder pois, no treinamento, isso significaria "espiar" o futuro. A ideia é traduzir token a token, sequencialmente, portanto uma posição do decoder deve trabalhar apenas com informações passadas. O mascaramento é feito simplesmente definindo a intensidade da atenção c_{ij} como $-\infty$, pois $\text{softmax}(-\infty) = 0$.

Ao invés de aplicar a auto-atenção uma única vez por componente ou, ainda, uma única vez a cada camada do componente, Vaswani et al. ainda propõem uma estrutura com múltiplas cabeças de atenção. A ideia é capturar, paralelamente, diferentes aspectos da sentença [134]: uma cabeça pode capturar fortemente relações entre substantivos, enquanto outra mapeia um encadeamento de ações (verbos). Isto, contudo, é apenas um efeito desejado, e o comportamento real de uma cabeça de atenção é fortemente dependente do treinamento da rede, e difícil de ser previsto, dada a natureza *black box* das redes neurais.

Para cada bloco de atenção, usa-se paralelamente h cabeças de atenção independentes, e seu resultado é concatenado e linearmente projetado via uma matriz W_O, representando assim a saída final da camada, de dimensão d_m,

$$\text{Multi}(Q, K, V) = [\text{cabeça}_1, ..., \text{cabeça}_h]W_O, \tag{4.11}$$

onde $\text{cabeça}_i = a(X_i^q W_Q^i, X_i^k W_K^i, X_i^v W_V^i)$, e $W^O \in \mathbb{R}^{hd_v \times d_m}$, pois para manter o custo computacional das multi-cabeças controlado, usa-se $d_k = d_v = \frac{d_m}{h}$. No caso do trabalho original [9], usa-se $d_m = 512$ e $h = 8$, portanto, $d_k = d_v = \frac{512}{8} = 64$. Aqui, $X \in \mathbb{R}^{n \times d_m}$, onde n é o tamanho das sequências tratadas pelo modelo.

O conteúdo de X varia conforme onde está o bloco de atenção: pode ser o resultado dos embeddings de entrada ou saída da rede ou, ainda, a saída de uma camada anterior, conforme detalhado a seguir. A estrutura final de um bloco de atenção multi-cabeças está ilustrada na Figura 4.6.

Os resultados do bloco de atenção, em seguida, passam por um processo de conexão residual e normalização. A conexão residual é uma ideia emprestada das redes neurais convolucionais, mais precisamente a ResNet [136]: adicionar a entrada da camada em sua saída. Esta técnica traz alguns benefícios conhecidos, como auxiliar na prevenção do desaparecimento de gradientes e permitir redes muito mais profundas (grande número de camadas), mas ainda estáveis [136]. Benefícios similares são observados para a normalização [137], justificando também sua aplicação.

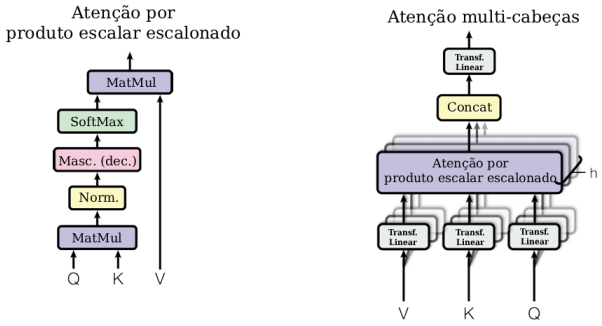

Figura 4.6: Componentes de atenção de um modelo transformer tradicional. Adaptada de [9].

A normalização aplicada é a proposta por Ba, Kiros & Hinton [137], dado um vetor $x \in \mathbb{R}^{d_m}$, representando a saída de um bloco,

$$\text{Normaliza}(x) = \frac{x - \mu}{\sqrt{\sigma + \varepsilon}} \odot \gamma + \beta, \tag{4.12}$$

onde $\mu = \frac{1}{d_m} \sum_{i=1}^{d_m} x_i$ e $\sigma^2 = \frac{1}{d_m} \sum_{i=1}^{d_m} (x_i - \mu)^2$,
e γ e β são, respectivamente, parâmetros de ganho e viés aprendidos pela rede.

No caso do transformer, a adição e normalização (abreviada como "Add & Norm" no artigo) é feita a cada transição de bloco (subcomponentes da rede), como ilustrado na Figura 4.7. Ou seja, a saída de um bloco nunca é puramente Bloco(x), mas sim

$$\text{BlocoResidual}(x) = \text{Normaliza}(x + \text{Bloco}(x)). \tag{4.13}$$

Além disso, introduz-se não-linearidade no transformer através de pequenas redes feedforward adicionadas ao final do encoder e decoder da rede, como também ilustra a Figura 4.7. A arquitetura feedforward utilizada conta apenas com duas transformações lineares com ativação ReLU:

$$\text{FNN}(x) = \max(0, xW_1 + b_1)W_2 + b_2, \tag{4.14}$$

com $W_1 \in \mathbb{R}^{d_m \times d_{ff}}$, $W_2 \in \mathbb{R}^{d_{ff} \times d_m}$, $b_1 \in \mathbb{R}^{d_{ff}}$, $b_2 \in \mathbb{R}^{d_m}$, sendo d_{ff} a dimensionalidade da camada oculta de FNN. Vaswani et al. utilizaram $d_{ff} = 2048$.

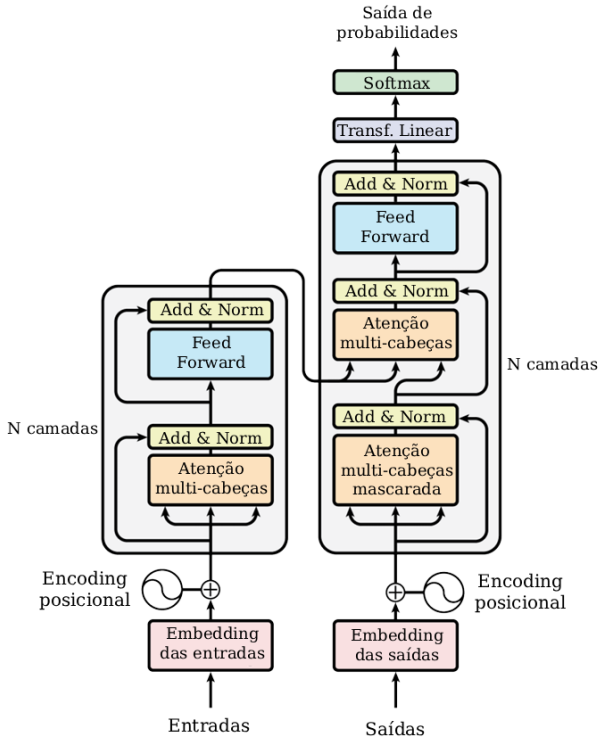

Figura 4.7: A arquitetura transformer. Adaptada de [9].

Descritos na Figura 4.7 estão todos os sub-componentes, ou blocos, da rede transformer. Estes blocos estão compostos como partes de dois componentes maiores: o encoder e o decoder, ambos alimentados por embeddings das entradas e saídas da rede, respectivamente.

Essas entradas e saídas são pré-processadas, onde é aplicada a tokenização através da técnica *byte-pair encoding* [138], que unifica as palavras dos idiomas fonte e destino em um único vocabulário compartilhado, representado por índices inteiros. Este vocabulário inclui tokens de termos auxiliares como ⟨EOS⟩, marcando o fim de uma sentença, e ⟨PAD⟩, marcando um espaço em branco.

O processo de embedding no transformer é governado por uma matriz $W_E \in \mathbb{R}^{d_{vocab} \times d_m}$, onde d_{vocab} é o tamanho do vocabulário e d_m, notação já utilizada, é a dimensão geral do modelo transformer[2]. A matriz W_E, aprendida pelo modelo, age essencialmente como uma tabela de consulta. A frase a sofrer embedding passa antes pela tokenização, que a transforma em um vetor de índices. O vetor de índices é transformado em uma matriz via *one-hot encoding*, onde cada linha da mesma é um vetor unitário, cujo elemento 1 está posicionado exatamente no índice equivalente da sequência original. Essa matriz é então multiplicada por W_E, gerando a matriz final $E \in \mathbb{R}^{n \times d_m}$, sendo a conversão da frase em embeddings. A seguir está um exemplo deste processo:

$$\begin{bmatrix} 1 \\ 3 \\ 4 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} W_E = E. \tag{4.15}$$

Dado que o vocabulário é compartilhado entre os diferentes componente do transformer, a mesma matriz W_E é utilizada na entrada do encoder, para transformar os dados

[2]Para facilitar a transferência de informações entre blocos, o transformer padroniza a mesma dimensão para todas as ocorrências.

de entrada; na entrada do decoder, para transformar os dados de saída; e na saída do decoder, para transformar as predições geradas. Nas entradas do encoder e decoder, a matriz utilizada é multiplicada por $\sqrt{d_m}$.

Mesmo que uma das grandes vantagens da arquitetura transformer para tradução de textos seja sua formulação não-linear, visto que diferentes línguas produzem ordens de palavras diferentes, isto não significa que o conhecimento sobre a ordem da sequência não é valioso. Portanto, Vaswani et al. incluem esta informação posicional como um par de senoides alternantes,

$$\text{posEncode}(i,j) = \begin{cases} \text{sen}\left(i \cdot 10^{-4\frac{j}{d_m}}\right) & \text{se } j \text{ é ímpar} \\ \cos\left(i \cdot 10^{-4\frac{j}{d_m}}\right) & \text{se } j \text{ é par} \end{cases}$$

onde $i \in 1, ..., n$ é o índice do embedding e $j \in 1, ..., d_m$ é o índice do j-ésimo elemento do embedding. Os números gerados por essa fórmula são simplesmente somados ao seu embedding equivalente, menos na saída do decoder, onde não são utilizados.

O encoder do transformers recebe os embeddings em sua entrada, os encaminha para o bloco de auto-atenção multi-cabeças, que então encaminha seus resultados ao bloco da rede neural feedforward, que produz as saídas do encoder. O encoder é composto por $N = 6$ camadas empilhadas desta estrutura.

O decoder, por sua vez, é alimentado pelos embeddings das saídas já produzidas pelo transformer, que na primeira iteração contém apenas um token de início de sentença ⟨BOS⟩, além da saída do encoder. As saídas já geradas passam por um bloco de auto-atenção multi-cabeças com máscara, de forma a não olhar para o futuro durante o treinamento, e são encaminhadas para um outro bloco de atenção. Este bloco, por sua vez, trata os embeddings do decoder como queries, mas as chaves e valores vêm da saída do encoder, de forma a imitar a atenção encoder-decoder visto nos modelos recorrentes.

O resultado deste último bloco passa também por um bloco de FNN, como já estabelecido, gerando as saídas finais do decoder. O processo é empilhado sequencialmente para todas as camadas do decoder, também em quantidade $N = 6$.

A saída produzida finalmente pelo decoder é uma matriz $S \in \mathbb{R}^{n_{atual} \times d_m}$, onde n_{atual} é o número de predições geradas até o momento, incluindo o token inicial. A matriz S é projetada nos pesos do embeddings, gerando uma matriz de logits $L = SW_E^\top$, $L \in \mathbb{R}^{n_{atual} \times d_{vocab}}$. Cada linha é uma posição predita pelo decoder, e cada coluna representa a chance daquela posição ser ocupada pelo token de índice equivalente ao número da coluna. Ou seja, L_{ij} guarda a chance, em um valor bruto, da i-ésima posição ser ocupada pelo j-ésimo token.

A matriz L é então normalizada por uma função softmax aplicada ao eixo do vocabulário (linha a linha), gerando probabilidades no intervalo $[0, 1]$. Essencialmente, essa é a saída final produzida pelo modelo transformer, como indica a Figura 4.7.

Resta então a escolha de um token para ser o próximo da sequência. A sequência com o novo token é retroalimentada no decoder, que gerará o próximo token, igualmente retroalimentado, e assim por diante até a geração de um token de fim de sentença ⟨EOS⟩.

Seria natural a escolha simplesmente do elemento com maior probabilidade de ocorrência (argmax), porém Vaswani et al. optam pelo uso da operação *beam search* [139] para seleção do token. Isto é justificado pois um melhor token localmente (mais provável dada a sequência já gerada até o momento) não necessariamente é o token que dará origem a melhor sequência.

O beam search é uma técnica heurística de exploração de grafos. No transformer, ele é utilizado para tentar encontrar a sequência mais promissora de tokens. A ideia é explorar, iterativamente, os k tokens mais prováveis (chamados usualmente de top k) até

que uma sentença inteira seja gerada, como ilustra a Figura 4.8. A seleção dos top k é feita iterativamente via poda. Isto é, a cada iteração, poda-se o grafo para manter apenas as top k sequências mais prováveis. A saída do transformer é, simplesmente, a sequência concluída mais provável.

Para calcular a sequência mais provável na iteração atual, soma-se os logaritmos das probabilidades para cada token da sequência. A mais provável é, simplesmente, a sequência com maior somatório. Contudo, aplica-se uma penalização diretamente ao tamanho da sentença, ideia extraída de [139], de forma a não favorecer injustamente frases que são simplesmente maiores e, portanto, têm maior somatório naturalmente.

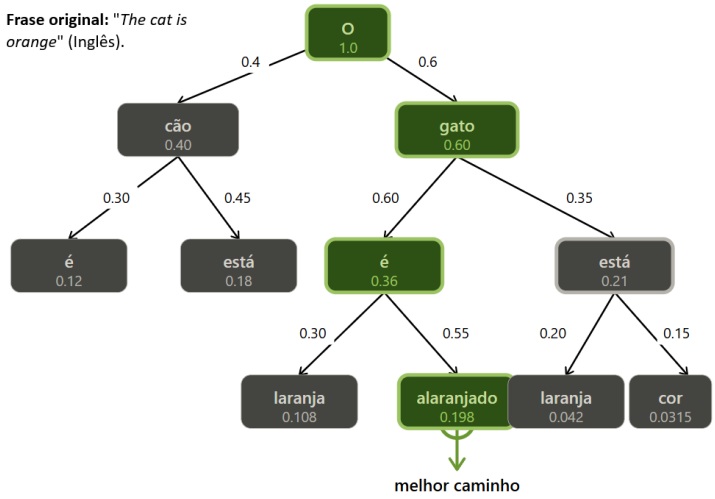

Figura 4.8: Exemplo meramente ilustrativo de aplicação de beam search com $k = 2$. Tokens especiais e penalização de tamanho omitidos para simplificação. Elaboração própria.

O modelo demonstrado em [9] foi treinado com o otimizador Adam. A função *loss* utilizada no transformer é a entropia cruzada. Seja $i = 1, ..., d_{atual}$ as posições geradas continuamente pelo transformer, e $P_i^{correto}$ a probabilidade estimada pelo decoder para o token correto na posição i, a *loss* é dada por

$$\mathcal{L} = -\sum_{i=1}^{d_{atual}} \log(P_i^{correto}). \tag{4.16}$$

Se a probabilidade estimada para o token correto é baixa, $-\log(P^{correto})$ torna-se grande, e a *loss* aumenta. Caso contrário, o valor aproxima-se de 0, e a *loss* é baixa.

4.4 O modelo E2E: regressão simbólica *end-to-end* com transformers

O trabalho original dos transformers [9] e seus vários desdobramentos [121–124] e aplicações da arquitetura hoje dominam o estado da arte do processamento de linguagem natural, em tarefas como tradução [120] e geração [121] de texto.

Para a tarefa de regressão simbólica, existem atualmente duas abordagens dominantes [11, 140]:

1. Adaptar o problema de SR para que se torne próximo a um problema de NLP, como feito pelo E2E [10], NeSymRes [66] e SymbolicGPT [92].

2. Unir uma outra abordagem de SR, usualmente GP, a um modelo de linguagem de larga escala (LLM, de *large language model*), como no LLM-SR [93] e LaSR [81].

Como já estabelecido, para evitar que o escopo desta dissertação estenda-se até os LLMs, que por si só possuem uma série de particularidades não relacionadas à regressão simbólica, a abordagem (1) foi escolhida como objeto de estudo. O método escolhido para investigação foi o E2E, dada a boa performance do TPSR (que utiliza o E2E, juntamente do NeSymRes, como base) em uma das competições do SRBench, ao passo que é mais recente que o NeSymRes.

O uso da arquitetura transformer para regressão simbólica é motivado principalmente [10, 66, 71] por uma busca por maior eficiência computacional. Dada a natureza exploradora dos métodos de programação genética, e o fato de que estes iniciam do zero seu processo de inferência a cada novo problema analisado, naturalmente torna-se lenta a experimentação. Em contrapartida, modelos transformer pré-treinados vêm consistentemente estabelecendo resultados excelentes em tarefas linguísticas, ao passo em que o tempo de inferência é muito baixo [9].

O E2E, sigla derivada do nome do artigo [10], *End-to-end symbolic regression with transformers*, representa uma dessas tentativas de criar um modelo qualitativamente competitivo, porém com um tempo de inferência significativamente menor do que as alternativas via GP.

Com este foco, o E2E adota uma estratégia diferente da utilizada por métodos conhecidos de SR via redes neurais até então: a predição da expressão completa, e não apenas de seu esqueleto.

Neste contexto, o "esqueleto" de uma função seria sua expressão completa, porém sem escalares definidos. Por exemplo, $f(x) = \cos(ax)$ é o esqueleto de $f(x) = \cos(3x)$. No DSR [47] e no NeSymRes [66], citados por Kamienny et al., o modelo usa uma combinação de predição do esqueleto via redes neurais, e predição de escalares via BFGS.

Segundo Kamienny et al. [10], o problema de predição de esqueletos é mal-posto e de difícil supervisão, e é um dos motivos para os métodos baseados em redes neurais não se aproximarem da precisão dos métodos baseados em GP. Por isso, propõem a predição numérica e estrutural das expressões em uma única passagem, justificando o nome do método, "*end-to-end*".

Ao invés de propor uma adaptação da arquitetura transformer original, cujo objetivo é linguístico, ao contexto da regressão simbólica, Kamienny et al. optam por transformar o problema de regressão de expressões em um problema linguístico. A arquitetura proposta, detalhada a seguir, é, neste sentido, extremamente próxima à original de Vaswani et al. [9].

A primeira mudança de paradigma em relação à GP é que as variáveis independentes $y_i \in R,\ i = 1, ..., N$ não são mais o alvo da otimização. Trazendo um ponto de vista linguístico, a entrada do E2E é um conjunto de variáveis completo, dependentes e independentes, e a saída alvo é a expressão matemática que as governa. O aprendizado profundo depende de treinamento e, dado o objetivo dos transformers e o objetivo de gerar uma expressão matemática, muda-se a perspectiva para receber os dados e transformá-los diretamente em expressões.

É necessário, portanto, uma forma de transformar valores contínuos e operadores em tokens. O método proposto apoia-se em uma abordagem clássica [141] da computação: a decomposição do número em sinal, mantissa e expoente.

Por padrão, o E2E limita o intervalo da mantissa em $[0, ..., 9999] \in \mathbb{Z}^+$, e do expoente em [E-100, ..., E100] (representação em caracteres do intervalo $\{10^k \mid k \in \mathbb{Z},\ -100 \leq k \leq 100\} \subset \mathbb{R}$). De forma similar ao expoente, a mantissa é prefixada por um caractere "N". Por exemplo, nesse sistema, o número 10.25 seria ["+", "N1025", "E-2"], onde todos estes são cadeias de caracteres, e cada cadeia é um token. No tempo de inferência, os

tokens são gerados um a um, mesmo que um número precise dos três componentes para ser representado.

Nota-se, portanto, uma limitação na quantidade de números que podem ser representados. Para contornar essa limitação, os dados de entrada são normalizados. Toma-se v_j, $j = 1, ..., D+1$ representando uma das variáveis do conjunto (isto é, uma das colunas), e faz-se

$$\tilde{v}_j = \frac{v_j - \mu_j}{\sigma_j}, \tag{4.17}$$

e o modelo encaminha $\tilde{v}_j$ para tokenização ao invés de v_j. Para gerar a expressão final inferida, a normalização é revertida para cada uma das variáveis presentes na expressão final.

A abordagem descrita é suficiente para representar as entradas, puramente numéricas, sendo utilizada pelo encoder do transformer. Contudo, para representar as expressões geradas no decoder, é necessário tratar a representação dos operadores, variáveis (de forma analítica) e da sequência e ordem de precedência da expressão em si.

Os operadores são manualmente mapeados no código do modelo. Por padrão, há suporte para os unários `abs` (valor absoluto), `inv` ($\frac{1}{x}$), `sqrt` (raiz quadrada), `logn`, `exp`, `sin`, `arcsin`, `cos`, `arccos`, `tan`, `arctan`, `pow2` (x^2), `pow3` (x^3); e para os binários `add` (adição), `sub` (subtração), `mul` (multiplicação), `div` (divisão[3]) e `pow` (potenciação). Nota-se a representação das operações clássicas de adição e subtração como palavras para diferenciá-las dos sinais dos tokens numéricos.

As variáveis também são adicionadas manualmente ao vocabulário. Por padrão, o E2E trabalha com apenas $D = 10$ variáveis independentes, e uma variável dependente, mapeadas sintaticamente para [`"x0"`, ..., `"x9"`, `"y"`]. O vocabulário também conta com tokens especiais ⟨EOS⟩, agindo para marcar tanto o fim quanto o início de uma expressão; e ⟨PAD⟩, para preencher espaços em branco.

O token ⟨PAD⟩ é particularmente usado quando $D < 10$, pois como a rede tem um tamanho de entrada fixo, é preciso preencher os espaços de variáveis independentes vazios com alguma informação.

Faz-se então a conversão do vocabulário para números inteiros, assim como visto em [9], na seguinte ordem: token especial ⟨EOS⟩, operadores e variáveis em ordem alfabética, tokens numéricos em ordem alfabética. Como estabelecido, o encoder tem visibilidade apenas dos membros numéricos.

Neste sentido, nota-se que a presença destes mapeamentos manuais implica que o modelo E2E como originalmente dado por Kamienny et al. é limitado a estes operadores e a 10 variáveis independentes, e qualquer modificação decorreria no re-treinamento do modelo.

Por fim, resta uma abordagem para dar um tratamento frasal às expressões, transformando-as em sentenças para o modelo linguístico. Para tanto, o E2E faz uso da notação polonesa direta [141] , assim como feito em [93]. Trata-se de uma forma não ambígua de representação de expressões, e que permite sua resolução com uma única leitura da esquerda para direita, sendo uma abordagem usual em calculadoras, juntamente da notação polonesa reversa. Dá-se então um exemplo de uma expressão tokenizada pelo sistema do E2E:

$$f(x) = cos(5.273x) \Rightarrow [\langle\text{EOS}\rangle, \text{cos}, \text{mul}, +, \text{N5273}, \text{E-3}, \text{x}].$$

[3]Em [10], é mencionado que o modelo não foi treinado com o operador de divisão, sendo este "substituído" pela multiplicação e pelo operador unário `inv`. Ao inspecionar o código do modelo, contudo, vê-se que `div` ainda faz parte do vocabulário da rede, mesmo que sua aparição seja praticamente impossível dado o treinamento.

A partir deste alicerce, a arquitetura do E2E [10] segue quase identicamente a proposta no transformer original [9]. Um encoder composto por um bloco de auto-atenção, seguido de uma rede FNN pequena; e um decoder com bloco de auto-atenção mascarado, seguido de um bloco de inter-atenção, seguido de uma pequena FNN de mesma arquitetura que no encoder. Novamente, assim como em [9], a saída de cada bloco é pós-processada por um componente de adição residual e normalização. A arquitetura de FNN utilizada também é a mesma de Vaswani et al. [9].

Há diferenças notáveis em termos de hiperparâmetros dimensionais das camadas e blocos, todavia. São utilizadas $h = 16$ cabeças de atenção, e a quantidade de camadas é assimétrica: 4 camadas no encoder, e 16 camadas no decoder. A assimetria é uma característica emprestada de [142], mas também observada empiricamente como a melhor opção para o E2E [10]. Demais dimensões são as mesmas de [9], tais como $d_m = 512$, a dimensão dos espaços latentes, e $d_{ff} = 2048$, a dimensão das redes feedforward internas do encoder e decoder.

Outra diferença principal está no processo de embedding. Ao invés de uma simples matriz de pesos W_E, o E2E emprega uma rede feedforward para fazer o embedding dos tokens. Isto se mostrou necessário [10] dada a notação decomposta em sinal, mantissa e expoente, empregada pelo modelo, que faz com que cada amostra de um conjunto custe $3(D + 1)$ tokens para representar.

Também no processo de embedding, outra distinção é o não uso da codificação posicional na entrada do encoder. Por se tratarem de valores puramente numéricos, a ordem não importa e, portanto, utilizar uma codificação posicional apenas adicionaria ruído. A variável dependente, contudo, é sempre posicionada à direita.

O custo $3(D + 1)$ de representação numérica torna o modelo muito custoso computacionalmente: mesmo que por padrão ele esteja limitado para $N = 200$ entradas e $D = 10$ variáveis independentes, isto já representaria um custo de 6600 tokens. Sem um hardware extremamente poderoso, o tempo de processamento seria alto, e o modelo perderia seu principal apelo.

Portanto, ao invés de criar um embedding para cada token, o E2E emprega uma FNN, de mesma arquitetura e dimensionalidade da interna do modelo, e a aplica para criar um único embedding para toda uma amostra, mapeando simultaneamente variáveis dependentes e independentes de uma amostra em um único vetor de dimensão d_m.

No mesmo sentido de otimização de eficiência computacional, o modelo por padrão é limitado para $N \leq 200$ amostras para inferência de uma expressão, fato já citado anteriormente. Aumentar esse limite acarretaria em impactos computacionais notáveis, especialmente em uso de memória primária [10].

Alternativamente, para conjuntos com $N > 200$, o E2E emprega *bagging* [143] para dividir os dados em B *bags* ($B = 100$ máximo por padrão) de 200 amostras, aleatoriamente. Cada *bag* é individualmente inserida no modelo, e a geração de expressões prossegue.

O E2E, contudo, não aplica o beam search para geração de expressões, por padrão. Ao invés disso, ele utiliza as probabilidades geradas pelo transformer como uma distribuição, e escolhe aleatoriamente um token seguinte de acordo com a mesma. Isto é feito paralelamente $C = 10$ vezes (execuções paralelas do decoder), gerando C funções candidatas para cada *bag*.

Este processo cria um arcabouço de BC funções candidatas. Das BC funções, eliminam-se funções inválidas e com esqueleto repetido. Destas, selecionam-se as $K = 10$ melhores via comparação do erro quadrático médio. As K funções passam então por um refinamento, onde seus escalares são otimizados via BFGS. Por fim, os escalares são denormalizados, produzindo assim a saída final do E2E, como ilustra a Figura 4.9.

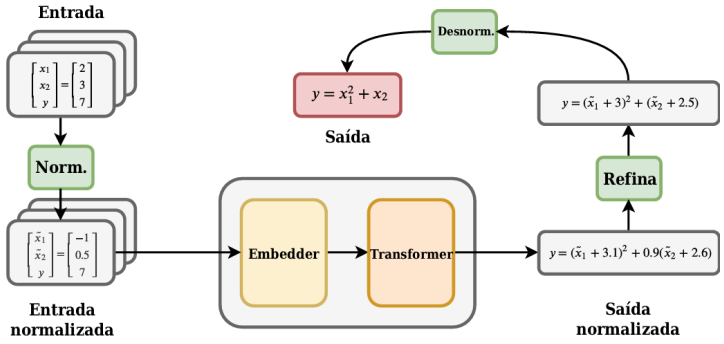

Figura 4.9: Visão geral do modelo E2E para regressão simbólica via transformers. Adaptada de [10].

O modelo foi treinado por Kamienny et al. utilizando o otimizador Adam, onde a função *loss* é a entropia cruzada, detalhada anteriormente, também usada no transformer original.

Os dados de treinamento foram gerados de forma puramente sintética. Primeiramente, geram-se funções f usando o seguinte processo:

1. Escolhe-se aleatoriamente uma dimensão D para a função f, $D \sim \mathcal{U}(1, D_{max})$, com $D_{max} = 10$.

2. Escolhe-se aleatoriamente um número b de operadores binários para compor a função, $b \sim \mathcal{U}(D-1, 2D+5)$. Depois, escolhe-se b operadores aleatórios de $\mathcal{U}\{\texttt{add}, \texttt{sub}, \texttt{mul}\}$.

3. Usa-se essas seleções para construir uma árvore binária representando a expressão, a partir de um método dado por [144].

4. Insere-se nas folhas da árvore variáveis aleatórias x_d, $d \in \mathbb{N}$, $1 \leq d \leq D$.

5. Escolhe-se um número u de operadores unários, $u \sim \{0, u_{max}\}$ ($u_{max} = 10$, por padrão). Então, seleciona-se aleatoriamente u operadores unários, dadas as chances dadas pelos seguintes pesos: `inv` $= 5$, `abs` $= 1$, `pow2` $= 3$, `sqrt` $= 3$, `sin` $= 1$, `cos` $= 1$, `tan` $= 0.2$, `log` $= 0.2$, `exp` $= 1$. Os operadores selecionados são inseridos em posições aleatórias válidas na árvore.

6. Para cada variável x_d e operador unário u, aplica-se uma transformação afim aleatória, isto é, substitui-se x_d por $ax_d + b$, e u por $au + b$. Os escalares a e b são gerados aleatoriamente, de forma decomposta: sinal $\sim \mathcal{U}\{-1, 1\}$, mantissa $\sim \mathcal{U}(0, 1)$ e expoente $\sim \mathcal{U}(-2, 2)$.

A cada passo do treinamento, o modelo gera um conjunto de 256 funções. O modelo foi treinado com 50 épocas de 3000 passos cada. Teoricamente, isto faz o modelo ser capaz de observar até 38.4 milhões de funções, mesmo que sua diversidade esteja limitada ao algoritmo.

Para cada uma das funções geradas, gera-se um conjunto de $N \in \mathcal{U}(10D, N_{max})$ (padrão $N_{max} = 200$, como dado anteriormente) dados. Para incentivar uma diversidade nos valores observados, a geração de variáveis independentes segue o algoritmo:

1. Escolhe-se um número de $k \sim \mathcal{U}(1, 10)$ clusters. A partir desta quantidade, gera-se $w_i \sim \mathcal{U}(0, 1)$, normalizados para que $\sum_i w_i = 1$.

2. Para cada cluster i, escolhe-se aleatoriamente um centroide $\mu_i \sim \mathcal{N}(0,1)^D$, um vetor de variâncias $\sigma_i \sim \mathcal{U}(0,1)^D$ e uma forma de distribuição (gaussiana ou uniforme) $\mathcal{D}_i \in \{\mathcal{N}, \mathcal{U}\}$.

3. Para cada cluster, são gerados $N = w_i N$ amostras, com $N \in \mathbb{Z}_*^+$ arredondado para baixo. A geração se dá aleatoriamente pela distribuição definida no passo anterior, $\mathcal{D}_i(\mu_i, \sigma_i)$.

4. Gera-se a variável dependente $y = f(x)$.

5. Concatenam-se as variáveis dependentes e independentes, formando uma única matriz X. A matriz é então normalizada colunarmente, por razões já descritas.

O primeiro resultado discutido a partir do uso do E2E, apresentado em [10], é uma aplicação dentro do domínio de treinamento, com um conjunto de testes gerado seguindo os exatos mesmos algoritmos descritos acima, para 100.000 funções.

Nestes testes, o modelo obteve performance razoável, com $R^2 = 0.68$. Além de demonstrar a performance, Kamienny et al. também apresentam uma análise de algumas cabeças de atenção para um teste feito especificamente com a função $f(x) = \frac{\text{sen}(x)}{x}$, dado na Figura 4.10. Nele, é possível ver que as cabeças capturam o comportamento periódico da função, um indicativo positivo de que o E2E é capaz, ao menos em alguns cenários, de capturar o comportamento das funções a partir do vocabulário simbólico-numérico.

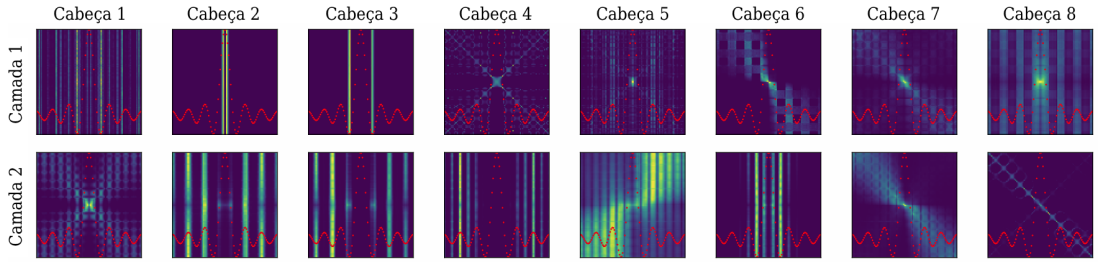

Figura 4.10: Projeção das 8 primeiras cabeças de atenção das 2 primeiras camadas do encoder do E2E ao receber a função $f(x) = \frac{\text{sen}(x)}{x}$. Adaptada de [10].

Além deste teste intra-domínio, Kamienny et al. também aplicaram o E2E ao SRBench, versão de 2021 [22]. Contudo, a apresentação de resultados dada em [10] é parcial: nenhum dos resultados é apresentado numericamente, apenas figuras. O código para execução destas avaliações também está disponível apenas parcialmente no repositório do modelo e, portanto, a reprodução dos resultados não é possível.

Em uma das figuras do trabalho [10], adaptada na Figura 4.11, observa-se que, nos problemas *black box*, o E2E obteve métricas R^2 próximas, mesmo que não superiores, ao Operon [49], um dos métodos considerados estado da arte para regressão simbólica [11], porém em um tempo de inferência muito inferior.

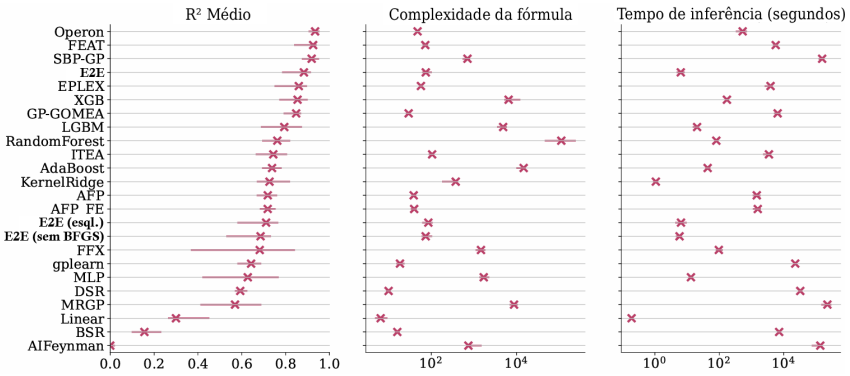

Figura 4.11: Resultados do E2E aplicado a uma seleção de problemas *black box* do SRBench 2021. Adaptada de [10].

Contudo, o trabalho [12] mostra um cenário muito diferente quando o E2E é aplicado em problemas físicos reais. As Tabelas 4.1 e 4.2, adaptadas a partir dos resultados do trabalho, e que comparam o E2E com o Operon, mostram que não somente o Operon obteve resultados mais precisos (via erro quadrático médio normalizado), como também o tempo de inferência foi consideravelmente menor na maioria dos casos.

Este último fato é particularmente surpreendente, dado que o Operon é baseado em programação genética e, portanto, precisa executar todo a sua lógica evolutiva antes de inferir suas expressões de saída. Os testes foram realizados em CPU (Intel Core i5-10400) para o Operon, e em GPU (RTX 2060 GPU, 6GB VRAM) para o E2E, um cenário ideal para ambos os métodos [5, 9].

Dados	**Operon**	**E2E**
Chemical Tower	0.057	55.99
Chemical Comp.	0.270	1.229
Flow Stress	0.001	0.491
Friction (dyn.)	0.070	1.087
Friction (stat.)	0.104	0.996
Battery 1	0.024	0.347
Battery 2	0.100	0.151
Nikuradse 1	0.054	0.905
Nikuradse 2	0.021	0.129

Tabela 4.1: Comparação de erro quadrático médio normalizado entre Operon e E2E. Adaptada de [12].

Dados	**Operon**	**E2E**
Chemical Tower	148	351
Chemical Comp.	29	92
Flow Stress	114	30
Friction (dyn.)	18	229
Friction (stat.)	21	230
Battery 1	18	79
Battery 2	35	171
Nikuradse 1	5	35
Nikuradse 2	6	50

Tabela 4.2: Comparação de tempo de inferência (em segundos) entre Operon e E2E. Adaptada de [12].

Os resultados obtidos em [12] corroboram os obtidos na competição de 2022 do SRBench [61] e na edição de 2025 do SRBench [11]. Na competição, o E2E ocupa a posição 6/9 nos testes simbólicos, e 8/8 nos testes *black box*. Nos problemas *black box* da edição de 2025 do benchmark, o E2E falha em obter um AUC Score maior que 0.6 em 11 dos 12 problemas avaliados.

Contudo, não é possível afirmar que este cenário descarta o uso do E2E em aplicações práticas. Na competição de 2023, o TPSR, que utiliza o E2E e o NeSymRes [66] internamente, foi o vencedor de um dos testes.

Tal fato reforça afirmações vistas em diversos trabalhos [11, 19, 22]: não existe um método absolutamente superior para regressão simbólica, e a performance varia conforme o caso. Ainda assim, nas comparações observadas nesta dissertação, os métodos baseados em programação genética tendem a apresentar os melhores resultados na maior parte dos cenários avaliados.

Capítulo

5

Avaliação de métodos para regressão simbólica

Mesmo já tendo, a esse ponto, cerca de 30 anos de existência, popularizada pelos trabalhos de Koza [21], a regressão simbólica ainda carecia, em meados de 2018, de um método de avaliação, um benchmark, adotado consensualmente entre os trabalhos [22].

Ainda era, neste período, comum observar o uso de problemas de baixíssima dimensionalidade, de fácil resolução, e de caráter altamente sintético [22], distante da realidade, como forma de comparação entre métodos. Em contraste, a comunidade de redes neurais já havia evoluído muito neste sentido, a exemplo da adoção dos conjuntos de dados ImageNet [145], VOC [146] e COCO [147] como padrões na detecção de objetos.

Ao longo do tempo, alguns trabalhos chamam atenção para este problema, como McDermott et al. em [148] (2012) e [149] (2013), que destacam dificuldades específicas ao processo de regressão simbólica, motivadoras para esta falta de consenso no período, como a falta de uma interface unificada para os métodos, requisitando escrita individual de código para cada método testado em um benchmark, desencorajando os autores de métodos a fazê-lo. Em contrapartida, as redes neurais já haviam se acomodado, em sua maioria, ao uso de *frameworks* em comum como Tensorflow [150] e Torch [151].

McDermott et al. também documentam tentativas falhas de estabelecer novos padrões de avaliação de métodos na área, como o GP-Beagle [152], definido pelos autores como "excessivamente ambicioso", onde apontam que o mesmo sofria da mencionada alta complexidade, isto é, sua interface entre método e conjuntos de dados do benchmark exigia muito preenchimento de informações e escrita de código por parte dos contribuintes. Deste modo, o GP-Beagle não obteve grande adoção pela comunidade científica.

Mesmo que reconhecidos [22] por propor sugestões para melhoria do estado da arte, como o banimento de problemas extremamente simples[1] de benchmarks, e definam características de um bom benchmark, como facilidade de modificação, variedade de problemas e eficiência computacional, os trabalhos [148] e [149] não representavam um esforço concreto para a produção de uma nova proposta de avaliação de métodos de regressão simbólica.

Em vista deste cenário, La Cava et al. propuseram o SRBench, apresentado de forma incipiente em uma conferência, em 2018 [60], e oficialmente publicado em sua primeira versão completa em 2021 [22] (a versão de 2018 era limitada a, apenas, alguns métodos baseados em programação genética).

[1] e.g. regressão de um polinômio de quarto grau [149]

Beneficiando-se da ascensão da linguagem de programação Python como língua franca da ciência de dados, o SRBench utiliza alguns padrões estabelecidos pela comunidade de aprendizado de máquina da linguagem para definir seus fundamentos.

A interface proposta pelo SRBench é, essencialmente, uma versão simplificada, mas ainda compatível, da API utilizada pelo scikit-learn [143], uma das bibliotecas mais conhecidas de ML para Python, com documentação bem estruturada. Deste modo, não é necessário adotar um padrão exclusivo ao SRBench, com o objetivo único de viabilizar a avaliação do método pela bateria de testes, pois implementar a API do scikit-learn também significa facilitar a adoção do método pela comunidade científica, ao torná-lo compatível com um padrão muito conhecido.

Esta exigência do uso da linguagem Python e suas convenções é potencialmente problemática ao passo em que vários métodos utilizam outra linguagem em sua implementação. Contudo, dada a já citada massiva adoção do Python, muitos dos principais métodos já nascem capazes de interfacear com a linguagem, como foi o caso do Operon [49], escrito em C++, e do PySR [5], cujo núcleo é feito em Julia.

A compatibilidade com a API do scikit-learn viabiliza a testagem massiva de métodos, pois padroniza a inserção de dados e colheita de resultados e métricas. Contudo, ainda há a particularidade da regressão simbólica, onde é desejável uma transparência e interpretabilidade dos modelos obtidos, isto é, das expressões matemáticas produzidas.

Novamente baseando-se nas convenções da linguagem Python, o SRBench exige que os métodos avaliados produzam saídas compatíveis com a biblioteca para matemática simbólica sympy [153], que tem mecanismos bastante robustos, capazes de representar uma grande quantidade de operações algébricas, variáveis e funções e, portanto, expressões matemáticas.

Adicionalmente, o SRBench também exige a adoção de versões específicas das bibliotecas requisitadas, para evitar possíveis disparidades entre os testes, fruto de, por exemplo, instabilidade numérica presente em uma versão de uma biblioteca, mas não em outra.

Com exceção de sua versão *preprint*, o SRBench traz em sua bateria de testes dois tipos de conjuntos de dados: problemas *black box*, cuja solução é desconhecida, objetivando conseguir uma melhor aproximação possível; e problemas *ground truth*, cuja solução é conhecida, ou seja, sabe-se a expressão matemática que o método avaliado deveria obter.

Para este segundo caso, nota-se que não necessariamente os dados de treinamento fornecidos aos métodos são extremamente precisos. Pelo contrário, alguns casos, como o experimento de Kepler neste texto demonstrado, trazem amostras imprecisas, as mesmas obtidas no século XVI por Brahe. Desta forma, objetiva-se também, de forma indireta, avaliar o quão bem os métodos avaliados lidam com ruídos e erros de medida.

Em 2021 [22], o benchmark contava com 122 conjuntos de dados *black box* e 130 conjuntos *ground truth*. A fonte dos dados *black box* é o PMLB [154], descrito a seguir. Já os dados *ground truth* vêm do Feynman Symbolic Regression Database [79] e do repositório ODE-Strogatz [155], conjuntos de dados majoritariamente compostos por problemas físicos conhecidos.

Na revisão de 2025, contudo, o número de testes foi drasticamente reduzido para 12 de cada tipo. Com o intuito de eliminar problemas triviais, foram removidos conjuntos de dados onde (1) todos os algoritmos obtiveram métrica $R^2 > 0.99$ (2) ou se $R^2 > 0.99$ foi atingido por uma regressão linear clássica via mínimos quadrados [11], restando 52 conjuntos. Por fim, foi utilizado o método k-means [156] após uma redução dimensional via t-SNE [157] aplicada sobre os metadados[2] dos conjuntos, criando 12 clusters, dos quais foram escolhidos os centroides como membros da seleção final.

[2]Número de amostras, número de variáveis, e performance no SRBench de 2021.

Ademais, foi notado que 62 dos 122 conjuntos *black box* originais eram variações da função de Friedman, uma fórmula para geração de dados sintéticos de regressão. Nesta nova versão, foi feita uma restrição manual para que no máximo 25% dos conjuntos viessem dessa família, caso passassem pelos filtros anteriormente mencionados. Vê-se na relação final que a restrição provavelmente foi necessária, pois a mesma contém 3 conjuntos (exatamente 25%) de Friedman.

Os dados *black box* originam-se diretamente da PMLB 1.0 (*Penn Machine Learning Benchmarks*) [154], uma grande coleção curada de conjunto de dados, criada com o objetivo de facilitar o acesso a benchmarks para avaliação de modelos. O repositório conta com 450 problemas, sendo 179 de classificação e 271 de regressão, obtidos através de diferentes fontes. Dentre os de regressão, os autores do SRBench selecionaram 12 considerados bons representantes de "problemas do mundo real", através do processo descrito anteriormente.

Quanto aos problemas *ground truth*, onde as expressões matemáticas são conhecidas, foram escolhidos 12 problemas a partir de seleções feitas por Cranmer [5] e Russeil et al. [158]. Tratam-se de equações clássicas das ciências da natureza, como a equação do gás ideal e a terceira lei de Kepler, adaptadas para tornarem-se problemas de regressão. Como já dito, mesmo trazendo expressões relativamente simples, estes problemas são considerados desafiadores dada a pequena quantidade de amostras e, em alguns casos, imprecisão nos dados.

De modo geral, essa redução foi motivada pelos objetivos de (1) eliminar problemas triviais e/ou pouquíssimo desafiadores; (2) dar maior foco em diversidade de problemas do que em quantidade; (3) reduzir o viés devido a uma super-representação dos conjuntos de Friedman; (4) diminuir o tempo de execução do benchmark, viabilizando iterações mais rápidas para melhoria do mesmo; e (5) permitir uma análise mais aprofundada dos resultados, dada a quantidade limitada de conjuntos.

A Tabela 5.1 traz uma relação detalhada dos conjuntos de dados presentes na edição de 2025 do SRBench, incluindo a fonte de cada um dos problemas trazidos pelo benchmark.

Para elaboração deste detalhamento complementar, a origem dos dados foi rastreada com base no nome do conjunto, nos nomes das variáveis, no tamanho dos conjuntos de dados e, após a filtragem com base nesses parâmetros, a equivalência foi comprovada através dos valores das amostras em si.

Este trabalho foi necessário pois, dos 12 conjuntos, apenas 2, 1089_USCrime e 192_vineyard, possuíam metadados (descrição do conjunto, documentação da origem dos dados e descrição das variáveis) bem documentados no repositório do PMLB. Dois dos conjuntos, ambos de prefixo "BNG", não puderam ter sua origem comprovada integralmente, dada uma provável modificação feita nos dados, explicada a seguir.

Nos conjuntos 1199_BNG_echoMonths e 1193_BNG_lowbwt, não há equivalência completa dos dados. As fontes citadas trazem exatamente os mesmos nomes de coluna dos dados do PMLB, porém os valores diferem. Os conjuntos do PMLB também são bem maiores, em número de linhas, do que os supostos originais, nestes casos.

Dado o prefixo "BNG", trazido pelo PMLB, porém ausente nos conjuntos supostamente originais, e o fato da suposta origem ser o software Weka [159], a hipótese mais provável é que esses dados tenham sido expandidos a partir dos originais utilizando redes bayesianas aleatórias, algo presente dentro do próprio Weka com o nome de "BayesNetGenerator" (BNG).

Os próprios autores do PMLB reconhecem o problema de falta de metadados em diversos conjuntos incluídos, e solicitam ajuda da comunidade para preenchimento [154], dado que trata-se de um projeto de código livre. Neste sentido, nenhuma das edições do SRBench [11, 22, 60] traz qualquer tipo de reflexão a respeito do conteúdo dos conjuntos de dados trazidos do PMLB.

Essa noção, todavia, é muito importante considerando que, com a inclusão dos dados *black box*, o SRBench ambiciona "trazer problemas do mundo real" [11, 22] para avaliar os modelos. Sem conhecimento a respeito do que as variáveis representam ou, no mínimo, deveriam representar, não é possível avaliar se os dados são aderentes a essa proposta, algo muito importante ao considerar a aplicação da regressão simbólica em contextos de aprendizado de máquina científico [15].

Conjunto de dados	**Nro. de linhas**	**Nro. de colunas**	**Descrição**	**Fonte**
Black box				
1028_SWD	1000	11	Originado do Weka, um software de mineração de dados da Universidade de Waikato, Nova Zelândia. Segundo sua descrição, o conjunto traz avaliações de funcionários do serviço social a respeito do risco de crianças permanecerem com suas famílias, avaliações estas que são usadas em casos judiciais a respeito de guarda de menores. Contudo, nenhuma das variáveis, incluindo a dependente, é documentada em nenhuma das fontes. Sabe-se apenas que tratam-se de valores categóricos, o que indica um problema de classificação.	[159, 160]
1089_USCrime	47	14	Um conjunto de variáveis socioeconômicas, relacionadas à taxa criminal de 47 estados dos EUA em 1960. A fonte secundária é um livro de estatística [161] e a fonte primária é um relatório do FBI [154, 161].	[161]
1193_BNG_lowbwt	31104	10	Variáveis relativas à gestação humana, cujo alvo de regressão é o peso do recém-nascido em gramas. A fonte secundária é o livro [162], e a primária são dados do Baystate Medical Center, em Springfield/MA, EUA.	[162]
1199_BNG_echoMonths	17496	10	Originado do Weka. Dados de pacientes que sofreram ataques cardíacos, onde os preditores são informações sobre os pacientes e seu sistema cardiovascular, com objetivo de predizer quantos meses o indivíduo sobreviverá após o ataque.	[159]

Conjunto de dados	**Nro. de linhas**	**Nro. de colunas**	**Descrição**	**Fonte**
192_vineyard	52	3	Dados de um pequeno vinhedo no lago Erie, EUA. Cada linha representa a quantidade anual, em caixotes, de uvas colhidas em uma mesma fileira de videiras. Cada coluna é um ano de colheita: 1989, 1990, o alvo sendo 1991. Trata-se portanto de um problema auto-regressivo.	[163]
210_cloud	108	6	Dados de um experimento de semeadura de nuvens feito em 1974. Utiliza-se a estação do ano, o fato da nuvem estar semeada ou não, e a pluviosidade nas regiões norte, sul e noroeste da área tratada para predizer a pluviosidade da região leste.	[164]
522_pm10	500	8	Quantidade de carros por hora, temperatura a 2m do solo, diferença de temperatura entre 2m e 25m do solo, velocidade do vento, direção do vento, hora do dia e dia a partir de Outubro de 2001, utilizados para predizer a concentração de MP_{10}. Dados coletados em Oslo, Noruega, por uma entidade pública.	[165]
557_analcatdata_apnea1	475	4	Notas de dois profissionais que avaliaram a qualidade do sono de 19 indivíduos a partir de resultados de diferentes polissonografias. As notas são variáveis categóricas, mapeadas para $\mathbb{Z}$. Os indivíduos também são distintos por um número inteiro. As variáveis são utilizadas para predizer a duração, em segundos, de apneia nos pacientes.	[166]
579_fri_c0_250_5 606_fri_c2_1000_10 650_fri_c0_500_50	250 1000 500	6 11 51	Dados para regressão completamente sintéticos gerados com a fórmula de Friedman (vide fonte).	[167]

Conjunto de dados	**Nro. de linhas**	**Nro. de colunas**	**Descrição**	**Fonte**
678_visualizing_environmental	111	4	Médias diárias de concentração de ozônio (em ppb), radiação solar (Langleys), temperatura (F) e velocidade do vento (mph) medidas de Maio a Setembro de 1973 em New York, NY, EUA. A variável predita é a velocidade do vento.	[168, 169]
Ground truth				
first_principles_absorption	14	2	$f(x;\mu,\epsilon) = \log\left(\frac{1}{\mu + e^{-\epsilon x}}\right)$	[158]
first_principles_bode	8	2	$f(n) = 0.4 + 0.3 \cdot 2^n$	[170]
first_principles_hubble	32	2	$f(D) = H_0 D$	[171]
first_principles_ideal_gas	30	4	$f(n,T,V) = \frac{nRT}{V}$	[172][3]
first_principles_kepler	6	2	$f(a) = \sqrt{\theta}\, a^3$	[94]
first_principles_leavitt	26	2	$f(P) = \alpha \log_{10}(P) + \delta$	[173]
first_principles_newton	30	4	$f(m_1,m_2,r) = \frac{G m_1 m_2}{r^2}$	[174][3]
first_principles_planck	100	3	$f(\nu,T) = \frac{2h\nu^3}{c^2} \cdot \frac{1}{e^{h\nu/(k_B T)} - 1}$	[175][3]
first_principles_rydberg	50	3	$f(n_1,n_2) = \frac{1}{\lambda} = R_H\left(\frac{1}{n_1^2} - \frac{1}{n_2^2}\right)$	[176][3]
first_principles_schechter	27	2	$f(L) = \frac{\phi^*}{L^*}\left(\frac{L}{L^*}\right)^{\alpha} e^{-L/L^*}$	[177][3]
first_principles_supernovae_zr	236	2	$f(t) = e^{-At}\left(B - e^{-Ct}\right)$	[158]
first_principles_tully_fisher	18	2	$f(V) = AV^{\alpha}$	[178]

Tabela 5.1: Conjuntos de dados presentes no SRBench 2025.

Ao analisar minuciosamente os metadados dos conjuntos presentes no SRBench 2025, percebe-se algumas possíveis incoerências, além da própria falta de descritores nas publicações e repositórios, tanto do PMLB, como do SRBench:

[3]Dados do SRBench gerados sinteticamente, com ruído, a partir da fórmula.

- Nenhuma das variáveis do 1028_SWD é descrita. Contudo, todas são explicitadas como categóricas, caracterizando um problema de classificação, não de regressão. Existem *issues* em aberto no repositório do PMLB que indicam conjuntos de dados mal caracterizados [154]. Porém, sem conhecimento dos dados, é impossível afirmar com certeza se a denominação está errada. Caso este erro eventualmente se confirme, contudo, o conjunto seria impróprio para inclusão do SRBench.

- De acordo com a própria descrição do conjunto de dados, a variável dependente para o 1089_USCrime aparenta estar incorreta: a descrição indica o uso de variáveis socioeconômicas para predição de taxa criminal (R), porém o alvo escolhido pelo PMLB foi o número de famílias com baixa renda (X), enquanto a taxa criminal entra como preditor. Se de fato foi um erro, este foi possivelmente motivado pela coluna X ser a última (esq. para dir.) do conjunto.

- Não há clareza sobre a origem dos conjuntos "BNG", possivelmente semi-sintéticos.

- 1193_BNG_lowbwt objetiva uma predição do peso do recém nascido, porém utilizada o fato do bebê estar ou não abaixo do peso (< 1.5kg) como um dos preditores, o que é incoerente por ser uma análise *post hoc* à própria regressão.

- 1199_BNG_echoMonths objetiva predizer quantos meses o indivíduo sobreviverá após um ataque cardíaco, porém utiliza o fato do indivíduo estar ou não vivo como um dos preditores.

Sem a avaliação de um especialista, não é possível fazer afirmações a respeito da qualidade dos conjuntos selecionados. Contudo, os pontos levantados, especialmente o fato de nenhuma das publicações do SRBench fazer reflexões qualitativas a respeito dos dados integrantes do benchmark, indicam que foi dedicada pouca atenção à avaliação e compreensão dos conjuntos, talvez por assumir que essa curadoria já tivesse sido feita pela equipe do PMLB. Segundo os próprios autores do PMLB, contudo, a curadoria foi incompleta [154].

É possível que, ao filtrar os conjuntos com objetivo de selecionar 12 representantes de problemas desafiadores, tenha havido um efeito colateral de seleção de conjuntos particularmente problemáticos. Contudo, ao filtrar os 122 conjuntos do SRBench 2021, observa-se questões semelhantes: retirando os sintéticos de Friedman e os 12 da versão 2025, nota-se que 45 dos 51 conjuntos restantes também não possuem documentação, 2 destes inclusive tendo sido oficialmente aposentados pelos autores do PMLB, reforçando a noção de que pouca ou nenhuma curadoria adicional foi feita nos dados pelos autores do SRBench.

Em [12] e [11], este segundo sendo o próprio texto do SRBench 2025, são desenvolvidas críticas de que o SRBench, mesmo em sua versão mais recente, não é suficientemente representativo de problemas do mundo real, e reconhece os pontos acima citados, como a falta de conhecimento acerca dos conjuntos, como questões a serem melhoradas.

> "[...] o benchmark deve desafiar os algoritmos de regressão simbólica contemporâneos, permanecendo computacionalmente viável e representativo de desafios do mundo real, garantindo que os resultados se estendam além de casos de teste artificiais. O SRBench fica aquém nesses aspectos, pois depende de mais de 200 conjuntos de dados sem uma categorização detalhada de seus desafios específicos." [11]

Neste sentido, nota-se uma evolução na proximidade do SRBench 2025 com problemas reais, se comparada à versão anterior. Em 2024, Matsubara et al. [179] identificaram problemas com os dados *ground truth* utilizados pelo SRBench 2021, advindos do Feynman

dataset [79], indicando que os mesmos não representavam os fenômenos físicos que deveriam modelar. Em vista disso, a versão 2025 substituiu esses dados pelos testes propostos por Russeil et al. [158] e Cranmer [5], amplamente documentados e com citações diretas já no texto do SRBench 2025.

Ademais, já existem propostas de expansão para o SRBench, tanto no campo *ground truth*, como o GeoBench [180], que propõe a inclusão de 71 funções geométricas ao benchmark; e, no campo *black box*, o SRBench++ [181], que traz 16 funções sintéticas, distintas e bem documentadas, e três conjuntos reais, sendo estes séries temporais relativas à COVID-19 em Nova Iorque, EUA.

Existem ainda outras propostas de benchmarking para regressão simbólica, tais quais o LLM-SRBench [140], à luz de diversas propostas recentes visando unificar LLMs a métodos de SR, novos ou preexistentes; e um trabalho recente que avalia diversos métodos na tarefa específica de recuperar sistemas dinâmicos [95].

Contudo, nenhum trabalho é tão amplamente citado e tido como referência quanto o SRBench, a exemplo de [5, 10, 19, 71]. Desta forma, vê-se que a melhoria contínua do SRBench é instrumental para manutenção e evolução do estado da arte da regressão simbólica, em específico

1. Para compreensão de quais são os melhores modelos de regressão simbólica na resolução de problemas reais, científicos e práticos, orientando assim a evolução do desenvolvimento de métodos.

2. Para que seja um alicerce na geração de dados sintéticos, essenciais para modelos de regressão simbólica que exigem treinamento, tais quais [10, 71] e que, portanto, tem sua qualidade diretamente associada com a qualidade dos dados de treinamento, preferencialmente em alta volumetria.

Mesmo havendo diversos pontos passíveis de crítica e melhorias no SRBench, é importante ressaltar que sua criação e ampla adoção ainda representam um grande avanço para a comunidade científica de regressão simbólica. Antes dele não havia consenso para benchmarking e, mesmo que os testes contidos precisem ser aperfeiçoados, todo o trabalho de interfaceamento via Python para os métodos e conjuntos de dados já é um grande auxílio para avaliação em massa.

Neste mesmo sentido, é importante ressaltar que propor alternativas ao SRBench (criação de um novo benchmark para substituí-lo) pode não ser o caminho mais promissor, ao menos não neste momento da regressão simbólica. É comum que áreas em evolução sofram com descentralização [182–184]. Até que uma solução muito madura exista, o que ainda não é o caso do SRBench, a história do desenvolvimento de software mostra que pode ser mais vantajoso para a comunidade científica que os esforços se concentrem na melhoria da solução mais comumente adotada.

5.1 Resultados do SRBench

O SRBench 2025 avaliou os modelos em duas principais métricas: coeficiente de determinação do modelo (R^2) e consumo energético em kWh. Para calcular as métricas, cada método foi avaliado 30 vezes em cada conjunto de testes. Os conjuntos foram divididos em 75% de dados para treino e 25% para teste. Foram utilizados hiperparâmetros ótimos definidos via busca em grade [11, 143].

A Figura 5.1 mostra os resultados de consumo energético na forma da mediana dos valores em kWh juntamente com o tempo de treinamento de cada modelo. São pontos

destacados pelos autores: (1) os métodos mais onerosos têm grande parte de sua implementação feita na linguagem de programação Python, interpretada e conhecidamente lenta e computacionalmente custosa, uma exceção sendo o Operon que, mesmo implementado em C++, tem passos internos de ajustes de hiperparâmetros, estes sendo bastante custosos; e (2) não há correlação clara entre uso de GPU (métodos TPSR, E2E e NeSymRes) e um aumento no consumo energético. Ademais, é destacado que, por mais que isso não necessariamente cause grande impacto em termos energéticos, adotar estratégias de parada precoce podem auxiliar em menores tempos de treinamento e, portanto, em menor consumo.

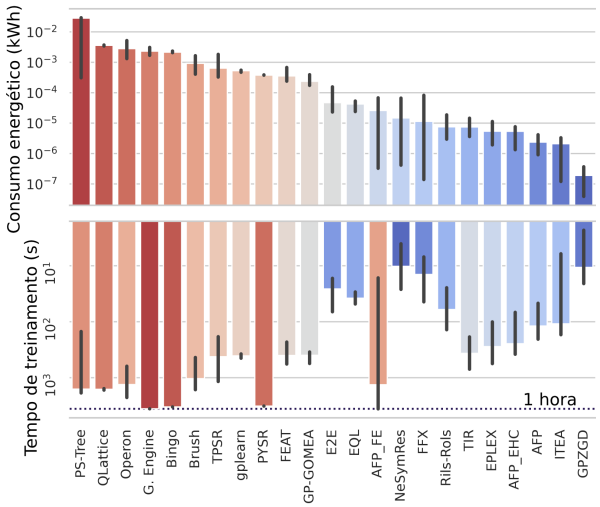

Figura 5.1: Consumo energético por modelo no SRBench 2025. Adaptada de [11].

Já os resultados referentes ao R^2 dos modelos foram exibidos em forma de um *plot* de perfil de performance [185]. Essencialmente os valores mostram, em um intervalo $AUC \in [0, 1]$, a probabilidade de $R^2 = 1$ para o método avaliado. Os resultados estão dispostos da Figura 5.2, onde a cor e o número em cada quadrado indicam a performance do método, e o tamanho do quadrado a complexidade[4] da expressão descoberta.

Os métodos mais performáticos foram, de modo geral, os baseados em programação genética, abordagem clássica para a regressão simbólica: PySR, Operon, Brush, PS-Tree, GPZGD e GP-Gomea figuram entre os líderes dos resultados e pertencem a esta categoria. Por outro lado, percebe-se que métodos bastante recentes, baseados em redes neurais, figuram entre os piores colocados, a exemplo do NeSymRes, uDSR, TPSR e E2E. Estudos como [12] já haviam sinalizado essa lacuna de performance entre os métodos baseados em redes neurais, que figuram entre as abordagens mais recentes [10] para SR, e as abordagens mais clássicas, onde estas se sobressaem em aplicações reais.

5.2 Competições do SRBench

Em 2022 e 2023, os principais autores do SRBench, em colaboração com outros pesquisadores, realizaram durante a Conferência de Computação Genética e Evolucionária (GECCO), dos respectivos anos, competições entre diferentes métodos de regressão simbólica.

Na competição de 2022, foram propostos dois desafios: o Desafio 1, com um conjunto de dados sintéticos; e o Desafio 2, com um conjunto real de COVID-19. Cada aplica-

[4]Mesma métrica utilizada pelo PySR: número de nós na árvore binária que representa a expressão.

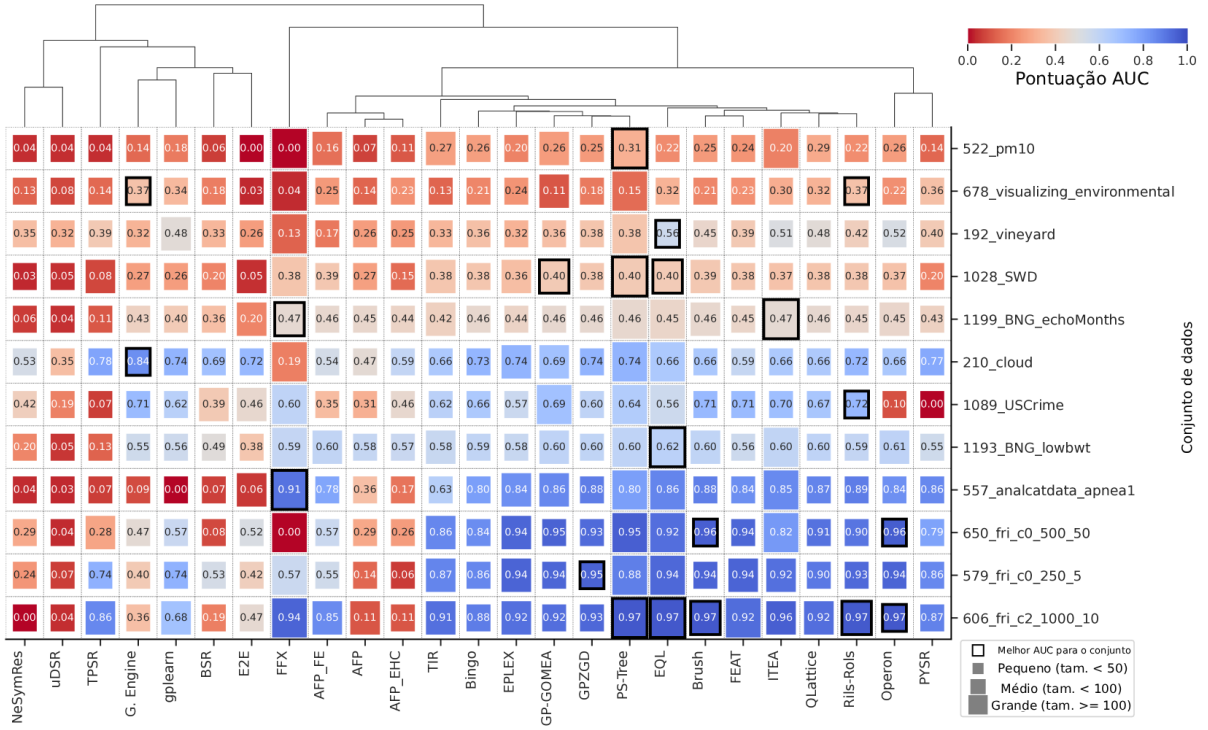

Figura 5.2: Resultados do SRBench 2025. AUC por modelo e conjunto de dados. Adaptada de [11].

ção de método aos dados foi avaliada em complexidade e acuracidade, sendo associada a estas uma nota final. Ademais, no Desafio 1, foram avaliadas também proximidade das expressões descobertas às expressões esperadas; variáveis escolhidas (a presença de variáveis mais importantes dava ao método maior pontuação); resistência a ótimos locais; capacidade de generalização; e sensibilidade a ruído. No Desafio 2, além de complexidade e acuracidade, um especialista avaliou as expressões descobertas, conferindo notas às soluções.

Em 2023, foram propostos novamente dois desafios, porém cada um com múltiplos conjuntos de dados. O Desafio 1 era composto por três conjuntos sintéticos; e o Desafio 2 por um conjunto real. A mudança entre os desafios estava no critério de avaliação.

No Desafio 1, avaliou-se apenas a complexidade e acurácia dos modelos. Já no desafio 2, além destes fatores, as expressões foram avaliadas também em termos de interpretabilidade, onde um júri deu uma nota para este quesito para o modelo mais acurado obtido por cada método.

A maioria dos métodos que figuraram entre os melhores, em ambas competições, são baseados em programação genética. Isto é esperado, até por estes comporem a maioria dos métodos para SR e, deste modo, possuírem maior chance de serem adotados. Todavia, o campeão do Desafio 2 de 2022 foi o uDSR, um método baseado em redes neurais; bem como o do Desafio 1 de 2023, o TPSR, baseado em arquitetura transformer. Desta forma, nota-se que a pouca performance de métodos baseados em redes neurais quando aplicados em casos reais, afirmada por [12] e evidenciada em [11], não pode ser considerada um fato absoluto.

Estes resultados reforçam a afirmação vista em [5, 11, 19], dentre outros trabalhos: no estado da arte da regressão simbólica, não existe uma abordagem unânime, método ou categoria de métodos que domine todas as outras. Cada experimento é um caso individual, e testar distintas abordagens e até mesmo propor novas soluções (sejam completamente

diferentes, ou variações de métodos conhecidos) não só é recomendável, como é comum (e.g. [86]).

5.3 Proposta de evolução ao SRBench

Ao analisar os dados presentes na edição de 2025 do SRBench, nota-se que os dois conjuntos *black box* que obtiveram menor performance geral advêm de um contexto de modelagem de poluentes: 522_pm10 e 678_visualizing_environmental.

Como já detalhado anteriormente deste texto, estes são, respectivamente, dados que relacionam a concentração de MP_{10} com covariáveis meteorológicas (ex.: temperatura) e espaciais (ex.: carros por hora); e dados que correlacionam a concentração de O_3 (gás ozônio) com variáveis meteorológicas. Tratam-se, portanto, de problemas muito semelhantes.

A predição de concentração de poluentes puramente a partir de covariáveis é um problema reconhecidamente difícil, com mesmo métodos robustos e resistentes a erros, baseados em redes neurais, atingindo métricas como $R^2 \approx 0.5$ [186]. Mesmo que espere-se, portanto, métricas razoavelmente baixas vindas destes experimentos, alguns detalhes devem ser ressaltados, especialmente acerca do conjunto 522_pm10:

- O mesmo possui apenas 500 amostras de média horária das variáveis, aleatórias a partir de Outubro de 2001. Mesmo que fossem contíguas, representariam apenas cerca de 21 dias de informação, o que não é suficiente para capturar fenômenos meteorológicos cuja variação é sazonal [187].

- Estão ausentes variáveis como umidade relativa do ar (r_h) e pressão atmosférica (p_a), demonstradas por outros experimentos [188] terem forte relação com a concentração de MP_{10} e $MP_{2.5}$.

- Sendo o SRBench uma bancada de testes específica para regressão simbólica, nota-se que a forma absoluta na qual algumas variáveis são postas não auxilia os métodos a atingirem convergência. Exemplo notável é a direção do vento (w_d), em graus, onde valores radicalmente diferentes têm efeitos similares (0° e 360°), e a variável "dia", cuja influência na concentração de poluentes é não-linear, e ainda assim é dada no 522_pm10 como uma variável codificada a partir do início da coleta de amostras, o que foge do usual [189, 190] para este tipo de experimento, onde observa-se o uso do dia juliano, objetivando modelagem de fenômenos sazonais.

Este último ponto não é necessariamente um problema para o SRBench, pois é possível argumentar que espera-se que os métodos de SR aprendam a transformar variáveis de forma a melhor ajustá-las – por exemplo, no caso do 522_pm10, seria possível decompor a direção e velocidade do vento em componentes vetoriais horizontal e vertical, como mostrado a seguir.

Contudo, deve-se também lembrar da proposta do SRBench de ser um benchmark vivo, atualizado e renovado conforme o estado da arte avança [11,22]. Neste sentido, seria menos valiosa a presença de um problema tão difícil quanto "prever as concentrações de poluentes do zero", enquanto a barreira de resolver o mesmo problema em uma versão simplificada, porém cujo sucesso ainda teria muito impacto científico, ainda não foi superada.

Propomos então a substituição do 522_pm10 por um conjunto mais rico em amostras e covariáveis, e também cujo processo de coleta é claro e rastreável, contrastando o feito pelo PMLB [154], onde ressalta-se a ausência de metadados sobre o 522_pm10 no repositório. O novo conjunto proposto foi batizado de "PM25_IBP", e é detalhado a seguir. Uma

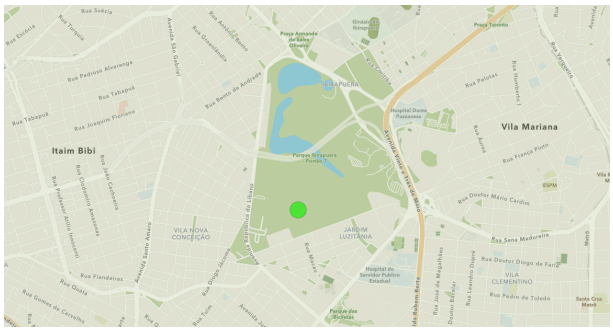

Figura 5.3: Localização da estação medidora de $MP_{2.5}$ no parque do Ibirapuera em São Paulo/SP. Extraída do mapa de estações do INMET.

outra versão deste conjunto de dados, "PM25_IBP_SR", também é apresentada, sendo esta mais ajustada à realidade atual do estado da arte da regressão simbólica.

A razão pela escolha do $MP_{2.5}$ ao invés do MP_{10}, visto no SRBench, é devido ao seu impacto na saúde. O $MP_{2.5}$ representa um material cujas partículas têm 2.5 µm ou menos de diâmetro aerodinâmico [191], como o *black carbon* e alguns tipos de hidrocarbonetos. Esta condição faz com que ele seja particularmente nocivo à saúde humana [192,193] pois, em contraste com algumas das maiores partículas de MP_{10}, o $MP_{2.5}$ consegue passar pelos filtros naturais do corpo humano e adentrar a corrente sanguínea facilmente [191].

Ademais, propomos também uma mudança ao conjunto 678_visualizing_environmental. Como notado neste capítulo, o experimento presente no PMLB e, consequentemente, no SRBench 2025, utiliza a radiação solar (em Langleys), a temperatura (em graus Fahrenheit) e a concentração de ozônio (em ppb) para predizer a velocidade do vento (em mph).

A utilização destas covariáveis para regressão da velocidade do vento não tem precedentes na literatura. A predição da concentração de O_3 a partir da radiação solar, temperatura e velocidade do vento, contudo, é usual, pois há uma fortíssima relação direta entre radiação e temperatura e a concentração, fatores envolvidos na formação deste poluente [187,191]. O vento, por sua vez, age como dispersor do gás, tendo relação inversa.

É muito provável, assim sendo, que os responsáveis pelo PMLB tenham selecionado a coluna incorreta como alvo da predição. Os resultados apresentados na seção 5.3.1 reforçam essa hipótese, pois a troca do ozônio, agora variável predita, com a velocidade do vento, agora preditor, produziram métricas muito superiores às apresentadas no SRBench 2025.

Todas as mudanças propostas foram validadas por um especialista da área de engenharia ambiental, creditado na seção de agradecimentos desta dissertação. A validação dos dados por um especialista também é um fator ausente no SRBench, sendo este um dos pontos de melhoria apresentados pelo SRBench++ [181].

5.3.1 Criação e avaliação dos conjuntos propostos

Foram obtidos dados de Janeiro de 2015 a Dezembro de 2025, provenientes do Sistema Qualar da CETESB [194], relativos ao Parque do Ibirapuera[5] em São Paulo/SP, coletados automaticamente de hora em hora. A razão para escolha deste conjunto de dados em específico é o seu relativo isolamento: mesmo estando inserido dentro da Cidade de São Paulo, a estação do Ibirapuera está localizada dentro do parque, vide Figura 5.3 e portanto relativamente distante dos principais emissores de poluentes via combustão.

[5]Estação em Lat. 23º 34' 55" S, Long. 46º 39' 25" W.

Deste modo, esperava-se que a concentração de $MP_{2.5}$ aferida pelo equipamento no local fosse majoritariamente influenciada por fatores meteorológicos, com pouca influência de variáveis espaciais, como quantidade de veículos e outros emissores ao redor da estação. O objetivo foi, neste quesito, propor um problema de resolução mais viável apenas a partir de variáveis meteorológicas e temporais.

No futuro, com o avanço dos métodos para regressão simbólica, seria factível a proposição da substituição do PM25_IBP por outros dados advindos, por exemplo, de estações da cidade de São Paulo mais afetadas por emissores, tais quais Pq. Dom Pedro II e Pinheiros, além da inclusão de variáveis espaciais.

Dada a falta de dados de precipitação dentre os disponibilizados pela CETESB, foi necessária a fusão dos dados do Ibirapuera com dados de precipitação da estação do Mirante de Santana [195], admitindo alguma imprecisão em decorrência desta extrapolação geográfica.

Os preditores escolhidos, bem como a respectiva justificativa e comportamentos esperados em literatura para sua escolha, foram:

- Umidade relativa do ar (h_r), em porcentagem (%), demonstrada ter relação inversa com a concentração de $MP_{2.5}$ por [188, 196], isto é, condições de seca favorecem a concentração do $MP_{2.5}$, enquanto condições de umidade média ou alta favorecem redução.

- Velocidade e direção do vento, em metros por segundo (m/s) e graus, respectivamente. Quanto maior a velocidade do vento, maior a dispersão de poluentes causada e, portanto, menor a concentração de $MP_{2.5}$ [191].

- Pressão atmosférica (p_a), em hectopascal (hPa), dado que altas pressões tendem a obstruir a dispersão vertical do $MP_{2.5}$, mantendo-o próximo à superfície [190, 197].

- Dia juliano (j), dado que o fenômeno da inversão térmica prejudica a dispersão de poluentes, e portanto maiores concentrações tendem a ser observadas no inverno, como indicam [198, 199] ao analisarem a região metropolitana de São Paulo.

- Dia da semana (d_s), dado o menor fluxo de veículos e atividade industrial nos fins de semana, espera-se uma menor concentração de poluentes [198, 199].

- Hora do dia (h_d), pois há um pico na hora do rush devido às maiores emissões veiculares [198, 199].

- Temperatura ($temp$), em graus Celsius (°C), onde existe uma relação inversa pois o aquecimento de superfícies gera turbulência, causando maior dispersão dos poluentes [199].

- Altura da camada limite (BLH, de *boundary layer height*), cujo aumento provoca dispersão [199].

- Precipitação (r), em milímetros (mm), devido a um fenômeno chamado "deposição úmida", no qual as gotas de chuva retiram material particulado do ar, sendo assim esperada uma diminuição nas concentrações medidas [187].

- Tempo absoluto (t), dado em Unix Time, de forma a modelar alguma tendência existente ao longo do tempo.

Os dados extraídos diretamente contavam com 67.382 amostras, um número fora dos padrões atualmente vistos em aplicação para algoritmos de SR. Relembra-se que o PySR recomenda uso de conjuntos de até 10 mil amostras, e o E2E foi treinado com cargas de 200 amostras. Portanto, destes, foram selecionadas aleatoriamente 10 mil amostras, divididas entre 75% para treino e 25% para teste, assim como visto no SRBench 2025 [11].

Além deste, foi criado o conjunto PM25_IBP_SR, com a motivação de criar um problema mais factível para resolução por algoritmos de regressão simbólica. As variáveis escolhidas foram exatamente as mesmas do PM25_IBP, porém com as seguintes mudanças:

- Uso da média diária dos dados, ao invés de horária, de forma a mitigar ruídos.
- Descarte da variável hora do dia, devido a alteração descrita acima.
- Transformação da variável de dia juliano em um fator contínuo, a partir da combinação de funções periódicas sen e cos na função suave $j_c = y(j) = \text{sen}\left(\frac{2\pi j}{365}\right) + \cos\left(\frac{2\pi j}{365}\right)$, seguindo a abordagem sugerida no livro [200]. Um *plot* desta curva pode ser visto na Figura 5.4, onde observa-se os picos de aproximadamente 1.4 e -1.4 no verão e inverno, respectivamente.
- Transformação da variável de dia da semana em um fator contínuo, utilizando a função $\cos\left(\frac{1.4\pi \cdot d_s}{14}\right) + \text{sen}\left(\frac{1.2\pi \cdot d_s}{7}\right)$, cujo *plot* também pode ser visto na Figura 5.4. Similarmente ao caso do dia juliano, objetivou-se criar uma função suave com vale nas abscissas equivalentes aos fins de semana.
- Decomposição da velocidade do vento em componentes vetoriais horizontal w_u e vertical w_v a partir da velocidade e direção em graus, criando assim fatores cujos valores absolutos são diretamente relacionados com a dispersão do $MP_{2.5}$.
- Transformação do Unix Time, que naturalmente produz números muito grandes, em uma variável normalizada $t \in [0, 1]$, com $t = 0$ no dia 01/01/2015, e $t = 1$ no dia 31/12/2025, sendo este o domínio do conjunto.

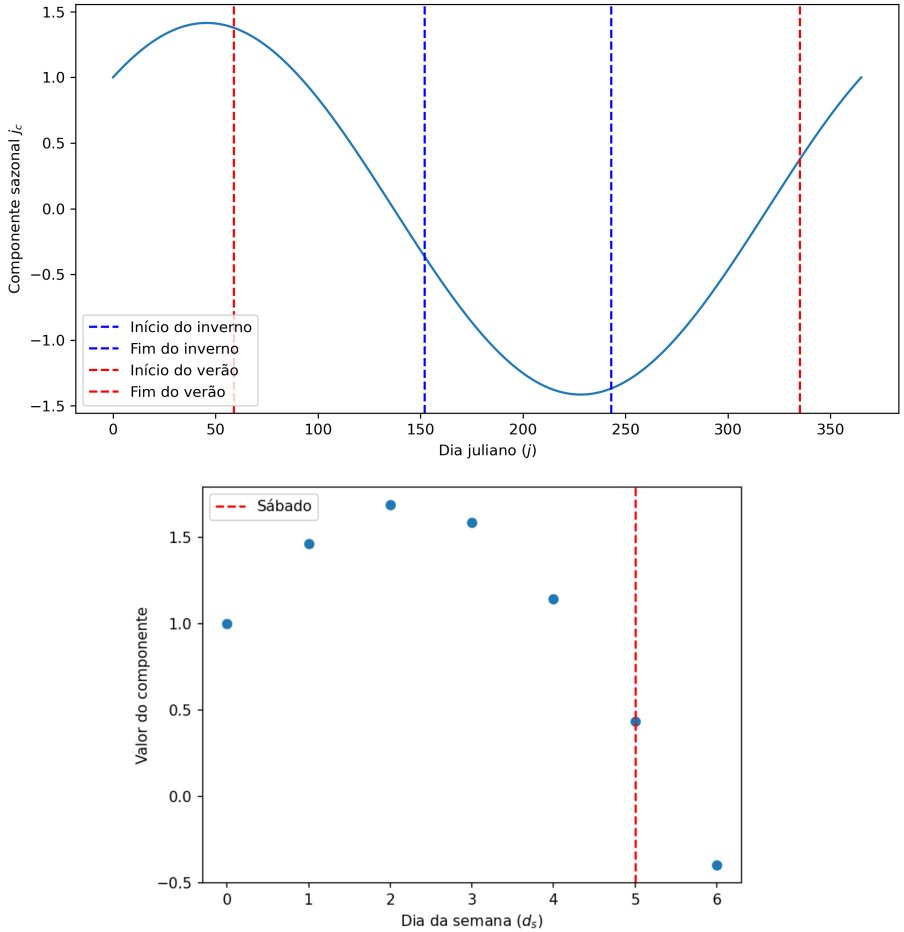

Figura 5.4: Componentes sazonais transformados de dia juliano e dia da semana. Elaboração própria.

Já quanto ao conjunto 678_visualizing_environmental, a alteração foi bem mais direta. A coluna "wind", representando a velocidade do vento, foi movida para ser um preditor, e a coluna "ozone", representando a concentração de O_3, passou a ser a variável predita. O conjunto foi então novamente dividido entre treino e teste, seguindo o padrão 75%/25% do SRBench 2025.

Os dados foram então fornecidos aos métodos PySR e E2E, objetos de estudo desta dissertação, e também presentes no SRBench 2025, e sua performance foi comparada entre os experimentos via métrica AUC. Os resultados obtidos, comparados aos do SRBench 2025, estão dispostos na Tabela 5.2.

Ressalta-se que o AUC calculado no SRBench 2025 é "a chance de um método obter $R^2 = 1$" [11]. Para tanto, são realizadas 30 rodadas de testes com sementes de aleatoriedade fixas, porém distintas. Este exato mesmo procedimento foi aplicado aos conjuntos novos, e a exata mesma métrica foi coletada, de forma a propor uma comparação justa. As métricas foram aferidas a partir da expressão mais acurada produzida por cada um dos métodos, em cada rodada.

	PySR - AUC	**E2E - AUC**
678_visualizing_environmental (SRBench)	0.36	0.03
678_visualizing_environmental (corrigido)	0.62	0.46
522_pm10 (SRBench)	0.14	0.00
PM25_IBP (novo)	0.22	0.01
PM25_IBP_SR (novo)	0.60	0.30

Tabela 5.2: Comparativo de perfil de performance entre os dados do SRBench e os novos dados propostos.

Observa-se um ganho modesto de performance para o PM25_IBP em relação ao 522_pm10, possivelmente atribuível a maior quantidade de dados, bem como a maior riqueza de variáveis. Ambos os métodos continuam apresentando um baixo coeficiente de determinação para o problema, contudo.

Neste contexto, destaca-se a limitação do E2E, que trabalha apenas com, no máximo, 10 variáveis independentes. Foi necessária, portanto, a remoção de uma das variáveis para realização dos testes. A variável escolhida foi a precipitação, por ser, dentre as disponíveis, a mais esparsa do conjunto, onde muitas amostras estão zeradas dada a ausência de chuva no respectivo instante horário.

O resultado obtido, de forma geral, é esperado e condizente com a literatura: como estabelecido no início desta seção, a predição de poluentes é um problema complexo, onde mesmo modelos muito mais resistentes a ruídos do que expressões matemáticas falham em obter acurácia elevada.

Independentemente dos resultados obtidos, ressalta-se que a utilização de um conjunto com mais amostras, mais variáveis conhecidamente importantes para o problema, mais recente, transparentemente rastreável e validado por especialista aproximaria mais o SRBench da realidade científica, em contraste com os conjuntos *black box* pouco documentados vistos atualmente.

O cenário é diferente para o PM25_IBP_SR. Observa-se um salto de performance em ambos os métodos, com destaque para o caso do E2E, que antes gerou pouquíssimos modelos capazes de atingir um $R^2 > 0$, produzindo uma métrica de performance próxima de zero, mas agora foi capaz de atingir uma pontuação de 0.3 (em escala de 0 a 1). Trata-se de um resultado ainda pouco expressivo, porém demonstra uma mudança qualitativa.

Deste modo, os resultados apontam que as transformações aplicadas para criação do PM25_IBP_SR auxiliaram na convergência dos métodos, e podem ser até mesmo um caminho para viabilizar a aplicação de regressão simbólica na predição de poluentes, trazendo interpretabilidade em uma área onde imperam os métodos *black box* [186, 189].

Os resultados com o 678_visualizing_environmental reforçam a hipótese de troca de colunas no conjunto de dados original. Houve aumento substancial nas métricas de ambos os métodos, onde destaca-se novamente a diferença para o E2E, que antes gerava poucas predições com $R^2 > 0$, com um perfil de performance próximo de zero, e com as correções atingiu AUC = 0.46, um aumento expressivo.

Seguindo o mesmo protocolo do SRBench 2025, foi coletada a performance do método dos mínimos quadrados ordinários quando aplicados nos novos conjuntos de dados, com o objetivo de verificar se os problemas propostos são "triviais". Relembra-se que, no SRBench 2025, um dos critérios para trivialidade era um $R^2 \geq 0.99$ atingido via mínimos quadrados ordinários.

Mesmo que o coeficiente R^2 obtido tenha sido relativamente alto para uma regressão linear tradicional, não é possível afirmar que esta abordagem resolve o problema e portanto o trivializa. O R^2 obtido para cada conjunto está disposto na Tabela 5.3, estando comparado contra os modelos mais precisos obtidos via PySR e E2E. Por se tratar de um método determinístico, não é possível calcular um perfil de performance para mínimos quadrados e, portanto, compara-se apenas o R^2 propriamente dito.

	Mín. Quadrados	**PySR**	**E2E**
678_visualizing_environmental (corrigido)	0.55	0.73	0.66
PM25_IBP	0.15	0.25	0.01
PM25_IBP_SR	0.56	0.63	0.49

Tabela 5.3: Comparativo de R^2 entre regressão linear e regressão simbólica para os novos conjuntos de dados sugeridos ao SRBench.

Assim como no SRBench 2025, os métodos também foram comparados em termos de sua complexidade, cujos resultados estão na Tabela 5.4. Novamente, os modelos com maior R^2 foram utilizados para comparação. Aqui, foi utilizada a definição adotada pelo PySR: contagem de nós na árvore binária que representa a expressão. Nestes testes, observa-se que as expressões geradas pelo E2E têm complexidade muitíssimo maior do que as geradas pelo PySR.

	Mín. Quadrados	**PySR**	**E2E**
678_visualizing_environmental (corrigido)	13	12	103
PM25_IBP	45	16	171
PM25_IBP_SR	41	28	138

Tabela 5.4: Comparativo de complexidade entre os modelos mais acurados obtidos para os novos conjuntos de dados sugeridos ao SRBench.

É importante destacar que, enquanto o método dos mínimos quadrados ordinários produz modelos de complexidade fixa $4n+1$, sendo n o número de variáveis independentes nos dados, e o PySR tem um limitante para complexidade por padrão[6] e a utiliza em sua função *loss*, o E2E não possui essa funcionalidade e tampouco incorpora a complexidade para cálculo da *loss*, justificando assim este comportamento.

Trata-se de um fato relevante pois, mesmo que interpretabilidade seja um conceito subjetivo, como já discutido anteriormente neste texto, com o aumento da complexidade das expressões, infere-se que também há uma diminuição significativa na facilidade de compreensão humana.

Os métodos foram também comparados em tempo de inferência médio, com resultados dispostos na Tabela 5.5. Não se trata de uma comparação isonômica, contudo. O método dos mínimos quadrados sequer pode ser caracterizado como um método de regressão simbólica, produzindo formas funcionais predefinidas. O E2E é um modelo pré-treinado e, portanto, realiza apenas inferência em tempo de execução. Já o PySR é essencialmente um algoritmo de busca e, sendo assim, não distingue treino de inferência, produzindo e avaliando expressões em tempo de execução. Todos os testes foram realizados em uma máquina portando um processador Intel Core I7-12700KF, 32GB de RAM DDR5, e uma placa de vídeo NVIDIA RTX 3090, com 24GB de VRAM GDDR6X.

[6]Na configuração utilizada no SRBench 2025, limita-se a complexidade para $C = 30$.

	Mín. Quadrados	**PySR**	**E2E**
678_visualizing_environmental (corrigido)	0.0012	3027.64	3.19
PM25_IBP	0.0019	3025.96	38.06
PM25_IBP_SR	0.0012	3025.71	42.59

Tabela 5.5: Comparativo de tempo de execução, em segundos, para os novos conjuntos de dados sugeridos ao SRBench.

Há, ainda, um agravante para o caso do PySR: a configuração utilizada pelo SRBench 2025 essencialmente força o algoritmo a rodar por 50 minutos, ao definir o número de iterações como $n_{\text{iter}} = 10^9$, e colocando um tempo limite de 50 minutos para finalização da busca. Uma comparação mais justa implicaria o abandono das configurações providas pelo SRBench, o que não é objetivo deste estudo.

Ressalta-se, ainda, que não foi ambição deste trabalho produzir modelos de predição ambiental. Neste mérito, um trabalho com este objetivo requereria mais experimentos, preferencialmente com o teste de outras configurações para os métodos que não as padrões escolhidas para o SRBench.

O objetivo foi destacar uma disparidade entre o perfil dos dados trabalhados no estado da arte da predição de poluentes e os contidos no SRBench. Os resultados são demonstrativos de uma mudança de cenário, no caso positiva. Em outros conjuntos, todavia, é possível que houvesse impacto negativo, como nos conjuntos onde há suspeita de uso de preditores *post hoc*, havendo possível sobreajuste.

Caracteriza-se assim a principal contribuição deste trabalho não apenas com a sugestão dos novos dados, mas sim com a análise crítica como um todo, identificando possíveis sintomas de anomalias da *track black box* do SRBench. Para amadurecimento do *benchmark*, fica claro que tais questões precisam ser minimamente esclarecidas e, se de fato reconhecidas como problemas, sanadas.

Capítulo

6

Considerações finais

A partir da análise detalhada dos métodos para regressão simbólica, explorados nesta dissertação, é possível compreender que a mesma representa uma ferramenta poderosa para o aprendizado de máquina interpretável, já tendo encontrado diversas aplicações práticas em diferentes áreas científicas [19, 29, 57].

Foi possível concluir que o estado da arte atual é dominado por métodos baseados em programação genética, onde os mesmos apresentam métricas mais ajustadas em baterias de testes amplas [11] e específicas [12]. Também compõem a grande maioria dos métodos descobertos, em função de serem a abordagem clássica para resolução da tarefa [21].

Este fato, todavia, não é um representativo absoluto, tampouco um desincentivo para outras abordagens. Atualmente, é unanimidade entre os principais autores da área que não existe solução única para a tarefa da regressão simbólica [5, 11, 19], onde já foram demonstrados casos onde métodos baseados em redes neurais [47, 67, 71] superaram os de programação genética em baterias de testes [11] e aplicações práticas [61, 62].

Observa-se também, recentemente, dada a explosão de adoção de LLMs em diferentes áreas [201], uma tendência crescente de incorporar transformers, modelos criados para o processamento de linguagem natural, na regressão simbólica.

Tentativas de adaptar a regressão simbólica em um problema linguístico [10, 66] apresentam resultados inconsistentes [11, 61, 62]. Todavia, abordagens mais recentes [81, 93] visam incorporar diretamente os LLMs, e seu potencial de inferência textual, com os métodos clássicos, baseados em programação genética, para produção de modelos.

A ideia central destas propostas é unir o potencial de análise crítica e de agregação de conhecimento dos LLMs com os algoritmos exploradores da programação genética, governando a descoberta de expressões. Tais tentativas ainda estão em fase incipiente, porém já despertaram interesse por parte da comunidade científica, notada através da criação recente de um benchmark específico para este tipo de modelo, o LLM-SRBench [140].

Neste sentido, também foi possível identificar a necessidade de evolução dos benchmarks para regressão simbólica, em especial do mais utilizado, o SRBench [11]. A existência de um benchmark sólido é central para a evolução de métodos de aprendizado de máquina, evidenciado pela história das redes convolucionais e seus conjuntos de teste [145, 147], muitas vezes diretamente relacionados com grandes avanços na área.

Esta demanda por evolução é carregada pelo próprio título da última edição do SRBench, "*call for action*", ou "chamado para ação", em tradução livre, onde os autores demonstram um refinamento da bateria de testes, ao passo em que pedem ajuda da comunidade para adição de mais dados, e a verificação de mais aspectos dos métodos.

Como possíveis trabalhos futuros decorrentes desta dissertação, é possível identificar duas frentes: a avaliação detalhada de métodos de regressão simbólica baseados em LLMs, como evolução natural da avaliação feita para métodos baseados em transformers, realizada neste texto; e a busca por aplicações práticas da regressão simbólica para predição de poluentes, dados os resultados positivos descobertos quando os dados são transformados de forma a adequarem-se ao funcionamento dos métodos de regressão simbólica.

Referências Bibliográficas

[1] Brunton, Steven L. e J. Nathan Kutz: *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2ª edição, 2022.

[2] Holland, John H.: *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1975.

[3] Lacerda, E. G. M. e André Carlos Ponce de Leon Ferreira Carvalho: *Introdução aos algoritmos genéticos*. Em *Anais do Congresso Nacional da Sociedade Brasileira de Computação*, Rio de Janeiro, 1999. EntreLugar.

[4] Knuth, Donald E.: *The Art of Computer Programming, Volume 1: Fundamental Algorithms*. Addison-Wesley, Reading, MA, 3ª edição, 1997, ISBN 978-0-201-89683-1.

[5] Cranmer, Miles: *Interpretable Machine Learning for Science with PySR and SymbolicRegression.jl*, 2023. https://arxiv.org/abs/2305.01582.

[6] Goodfellow, Ian, Yoshua Bengio e Aaron Courville: *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[7] Strobelt, Hendrik, Sebastian Gehrmann, Michael Behrisch, Adam Perer, Hanspeter Pfister e Alexander M. Rush: *Seq2Seq-Vis: A Visual Debugging Tool for Sequence-to-Sequence Models*, 2018. https://arxiv.org/abs/1804.09299.

[8] Weng, Lilian: *Attention? Attention!* lilianweng.github.io, 2018. https://lilianweng.github.io/posts/2018-06-24-attention/.

[9] Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser e Illia Polosukhin: *Attention Is All You Need*. CoRR, abs/1706.03762, 2017. http://arxiv.org/abs/1706.03762.

[10] Kamienny, Pierre Alexandre, Stéphane d'Ascoli, Guillaume Lample e François Charton: *End-to-end symbolic regression with transformers*, 2022. https://arxiv.org/abs/2204.10532.

[11] Imai Aldeia, Guilherme Seidyo, Hengzhe Zhang, Geoffrey Bomarito, Miles Cranmer, Alcides Fonseca, Bogdan Burlacu, William G. La Cava e Fabrício Olivetti de França: *Call for Action: towards the next generation of symbolic regression benchmark*. Em *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '25 Companion, página 2529–2538. ACM, julho 2025. http://dx.doi.org/10.1145/3712255.3734309.

[12] Radwan, Yousef A., Gabriel Kronberger e Stephan Winkler: *A Comparison of Recent Algorithms for Symbolic Regression to Genetic Programming*, 2024. https://arxiv.org/abs/2406.03585.

[13] Wael, Ayham e Amer Madi: *Accelerating Artificial Intelligence: The Role of GPUs in Deep Learning and Computational Advancements*. East Journal of Engineering, 1(1):31–46, Feb. 2025. https://eastpublication.com/index.php/eje/article/view/34.

[14] Krizhevsky, Alex, Ilya Sutskever e Geoffrey E Hinton: *ImageNet Classification with Deep Convolutional Neural Networks*. Em Pereira, F., C.J. Burges, L. Bottou e K.Q. Weinberger (editores): *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[15] Baker, Nathan, Frank Alexander, Timo Bremer, Aric Hagberg, Yannis Kevrekidis, Habib Najm, Manish Parashar, Abani Patra, James Sethian, Stefan Wild *et al.*: *Workshop Report on Basic Research Needs for Scientific Machine Learning: Core Technologies for Artificial Intelligence*. Relatório Técnico, USDOE Office of Science (SC), Washington, D.C. (United States), fevereiro 2019. https://www.osti.gov/biblio/1478744.

[16] Brunton, Steven L., Joshua L. Proctor e J. Nathan Kutz: *Discovering governing equations from data by sparse identification of nonlinear dynamical systems*. Proceedings of the National Academy of Sciences, 113(15):3932–3937, março 2016, ISSN 1091-6490. http://dx.doi.org/10.1073/pnas.1517384113.

[17] Faroughi, Salah A., Farinaz Mostajeran, Amin Hamed Mashhadzadeh e Shirko Faroughi: *Scientific Machine Learning with Kolmogorov-Arnold Networks*, 2025. https://arxiv.org/abs/2507.22959.

[18] Quarteroni, Alfio, Paola Gervasio e Francesco Regazzoni: *Combining physics-based and data-driven models: advancing the frontiers of research with scientific machine learning*. Mathematical Models and Methods in Applied Sciences, 35(04):905–1071, março 2025, ISSN 1793-6314. http://dx.doi.org/10.1142/S0218202525500125.

[19] Makke, Nour e Sanjay Chawla: *Interpretable scientific discovery with symbolic regression: a review*. Artificial Intelligence Review, 57(1):2, 2024, ISSN 1573-7462. https://doi.org/10.1007/s10462-023-10622-0.

[20] Angelis, Dimitrios, Filippos Sofos e Theodoros E. Karakasidis: *Artificial Intelligence in Physical Sciences: Symbolic Regression Trends and Perspectives*. Archives of Computational Methods in Engineering, 30(6):3845–3865, 2023, ISSN 1886-1784. https://doi.org/10.1007/s11831-023-09922-z.

[21] Koza, John R.: *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992, ISBN 0-262-11170-5.

[22] Cava, William La, Patryk Orzechowski, Bogdan Burlacu, Fabrício Olivetti de França, Marco Virgolin, Ying Jin, Michael Kommenda e Jason H. Moore: *Contemporary Symbolic Regression Methods and their Relative Performance*, 2021. https://arxiv.org/abs/2107.14351.

[23] Cranmer, Miles e Contributors: *SymbolicRegression.jl: Distributed High-Performance Symbolic Regression in Julia*, 2025. https://github.com/MilesCranmer/SymbolicRegression.jl.

[24] Carvalho, Diogo V., Eduardo M. Pereira e Jaime S. Cardoso: *Machine Learning Interpretability: A Survey on Methods and Metrics*. Electronics, 8(8), 2019, ISSN 2079-9292. https://www.mdpi.com/2079-9292/8/8/832.

[25] Marcinkevičs, Ričards e Julia E. Vogt: *Interpretable and explainable machine learning: A methods-centric overview with concrete examples*. WIREs Data Mining and Knowledge Discovery, 13(3):e1493, 2023. https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1493.

[26] Zeiler, Matthew D. e Rob Fergus: *Visualizing and Understanding Convolutional Networks*. Em Fleet, David, Tomas Pajdla, Bernt Schiele e Tinne Tuytelaars (editores): *Computer Vision – ECCV 2014*, páginas 818–833, Cham, 2014. Springer International Publishing, ISBN 978-3-319-10590-1.

[27] Choo, Jaegul e Shixia Liu: *Visual Analytics for Explainable Deep Learning*, 2018. https://arxiv.org/abs/1804.02527.

[28] Rudin, Cynthia: *Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead*. Nature Machine Intelligence, 1(5):206–215, 2019, ISSN 2522-5839. https://doi.org/10.1038/s42256-019-0048-x.

[29] Cranmer, Miles, Alvaro Sanchez-Gonzalez, Peter Battaglia, Rui Xu, Kyle Cranmer, David Spergel e Shirley Ho: *Discovering Symbolic Models from Deep Learning with Inductive Biases*, 2020. https://arxiv.org/abs/2006.11287.

[30] Lipton, Zachary C.: *The mythos of model interpretability*. Commun. ACM, 61(10):36–43, setembro 2018, ISSN 0001-0782. https://doi.org/10.1145/3233231.

[31] Friedman, Jerome H. e Bogdan E. Popescu: *Predictive learning via rule ensembles*. The Annals of Applied Statistics, 2(3), setembro 2008, ISSN 1932-6157. http://dx.doi.org/10.1214/07-AOAS148.

[32] Ustun, Berk e Cynthia Rudin: *Supersparse linear integer models for optimized medical scoring systems*. Machine Learning, 102(3):349–391, novembro 2015, ISSN 1573-0565. http://dx.doi.org/10.1007/s10994-015-5528-6.

[33] Sundararajan, Mukund, Ankur Taly e Qiqi Yan: *Axiomatic Attribution for Deep Networks*, 2017. https://arxiv.org/abs/1703.01365.

[34] Lundberg, Scott M e Su In Lee: *A Unified Approach to Interpreting Model Predictions*. Curran Associates, Inc., 2017. http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf.

[35] Ribeiro, Marco Tulio, Sameer Singh e Carlos Guestrin: *"Why Should I Trust You?": Explaining the Predictions of Any Classifier*, 2016. https://arxiv.org/abs/1602.04938.

[36] Shrikumar, Avanti, Peyton Greenside e Anshul Kundaje: *Learning Important Features Through Propagating Activation Differences*, 2019. https://arxiv.org/abs/1704.02685.

[37] Štrumbelj, Erik e Igor Kononenko: *Explaining prediction models and individual predictions with feature contributions*. Knowledge and Information Systems, 41:647–665, 2014. https://api.semanticscholar.org/CorpusID:2449098.

[38] Datta, Anupam, Shayak Sen e Yair Zick: *Algorithmic Transparency via Quantitative Input Influence: Theory and Experiments with Learning Systems*. páginas 598–617, maio 2016.

[39] Bach, Sebastian, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus Robert Müller e Wojciech Samek: *On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation*. PLOS ONE, 10(7):e0130140, 2015, ISSN 1932-6203. https://doi.org/10.1371/journal.pone.0130140.

[40] Lipovetsky, Stan e Michael Conklin: *Analysis of regression in game theory approach*. Applied Stochastic Models in Business and Industry, 17(4):319–330, 2001. https://onlinelibrary.wiley.com/doi/abs/10.1002/asmb.446.

[41] Saabas, Ando: *Interpreting random forests*. http://blog.datadive.net/interpreting-random-forests/, 2014. Acesso em 29/09/2025.

[42] Alaa, Ahmed M. e Mihaela van der Schaar: *Demystifying Black-box Models with Symbolic Metamodels*. Em Wallach, H., H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox e R. Garnett (editores): *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. https://proceedings.neurips.cc/paper_files/paper/2019/file/567b8f5f423af15818a068235807edc0-Paper.pdf.

[43] Crabbé, Jonathan, Yao Zhang, William Zame e Mihaela van der Schaar: *Learning outside the Black-Box: The pursuit of interpretable models*, 2020. https://arxiv.org/abs/2011.08596.

[44] Stephens, Trevor: *gplearn: Genetic Programming in Python*. https://gplearn.readthedocs.io/en/stable/, 2025. Acesso em 10/11/2025.

[45] Cranmer, Miles: *Interpretability via Symbolic Distillation*. Apresentação feita no Simons Institute, Berkeley, CA, 2023. Disponível em: https://simons.berkeley.edu/talks/miles-cranmer-flatiron-institute-2023-08-15, Acesso em 09/09/2025.

[46] Jeff Larson, Surya Mattu, Lauren Kirchner e Julia Angwin: *How We Analyzed the COMPAS Recidivism Algorithm*. ProPublica, 2016. https://www.propublica.org/article/how-we-analyzed-the-compas-recidivism-algorithm, Acesso em 10/11/2025.

[47] Petersen, Brenden K., Mikel Landajuela, T. Nathan Mundhenk, Claudio P. Santiago, Soo K. Kim e Joanne T. Kim: *Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients*, 2021. https://arxiv.org/abs/1912.04871.

[48] Virgolin, Marco e Solon P. Pissis: *Symbolic Regression is NP-hard*, 2022. https://arxiv.org/abs/2207.01018.

[49] Burlacu, Bogdan, Gabriel Kronberger e Michael Kommenda: *Operon C++: an efficient genetic programming framework for symbolic regression*. Em *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, GECCO '20, página 1562–1570, New York, NY, USA, 2020. Association for Computing Machinery, ISBN 9781450371278. https://doi.org/10.1145/3377929.3398099.

[50] Brisset, S. e F. Gillon: *Approaches for multi-objective optimization in the ecodesign of electric systems*. Em Bessède, Jean Luc (editor): *Eco-Friendly Innovation in Electricity Transmission and Distribution Networks*, páginas 83–97. Woodhead Publishing, Oxford, 2015, ISBN 978-1-78242-010-1. https://www.sciencedirect.com/science/article/pii/B9781782420101000045.

[51] Chatzi, Eleni, Georgios Kissas, Karin Yu e Orestis Oikonomou: *Formal Grammars in Scientific Discovery*. Em *SIAM Conference on Computational Science and Engineering (CSE25)*, 2025.

[52] Zhang, Zhen, Zongren Zou, Ellen Kuhl e George Em Karniadakis: *Discovering a reaction–diffusion model for Alzheimer's disease by combining PINNs with symbolic regression*. Computer Methods in Applied Mechanics and Engineering, 419:116647, 2024, ISSN 0045-7825. https://www.sciencedirect.com/science/article/pii/S0045782523007703.

[53] Kabliman, Evgeniya, Ana Helena Kolody, Johannes Kronsteiner, Michael Kommenda e Gabriel Kronberger: *Application of symbolic regression for constitutive modeling of plastic deformation*. Applications in Engineering Science, 6:100052, 2021, ISSN 2666-4968. https://www.sciencedirect.com/science/article/pii/S2666496821000182.

[54] Udrescu, Silviu Marian e Max Tegmark: *Symbolic pregression: Discovering physical laws from distorted video*. Physical Review E, 103(4), abril 2021, ISSN 2470-0053. http://dx.doi.org/10.1103/PhysRevE.103.043307.

[55] Sasaki, Yuji, Keito Tanemura, Yuki Tokuni, Ryohei Miyadera e Hikaru Manabe: *Application of Symbolic Regression to Unsolved Mathematical Problems*. Em *2023 International Conference on Artificial Intelligence and Applications (ICAIA) Alliance Technology Conference (ATCON-1)*, páginas 1–6, 2023.

[56] Abdellaoui, Ismail Alaoui e Siamak Mehrkanoon: *Symbolic regression for scientific discovery: an application to wind speed forecasting*, 2021. https://arxiv.org/abs/2102.10570.

[57] Ukachukwu, Christian Chinweike: *SYMBOLIC REGRESSION IN ENERGY ENGINEERING*, 2024. ISSN 0282-1990. Tese de graduação.

[58] Roland, Wolfgang, Michael Kommenda e Gerald R. Berger-Weber: *Application of Symbolic Regression in Polymer Processing*. Em *2022 24th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, páginas 311–318, 2022.

[59] Weng, Baicheng, Zhilong Song, Rilong Zhu, Qingyu Yan, Qingde Sun, Corey G. Grice, Yanfa Yan e Wan Jian Yin: *Simple descriptor derived from symbolic regression accelerating the discovery of new perovskite catalysts*. Nature Communications, 11(1):3513, 2020, ISSN 2041-1723. https://doi.org/10.1038/s41467-020-17263-9.

[60] Orzechowski, Patryk, William La Cava e Jason H. Moore: *Where are we now? a large benchmark study of recent symbolic regression methods*. Em *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '18, página 1183–1190, New York, NY, USA, 2018. Association for Computing Machinery, ISBN 9781450356183. https://doi.org/10.1145/3205455.3205539.

[61] La Cava, William: *Results files from the 2022 SRBench Competition: Interpretable Symbolic Regression for Data Science*, julho 2022. https://doi.org/10.5281/zenodo.6842176.

[62] França, Fabrício Olivetti de, Marco Virgolin, Pierre Alexandre Kamienny e Geoffrey Bomarito: *SRBench Competition 2023 - Datasets, Scripts, and Results*, agosto 2023. https://doi.org/10.5281/zenodo.8283005.

[63] Burlacu, Bogdan: *GECCO'2022 Symbolic Regression Competition: Post-Analysis of the Operon Framework*. Em *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, GECCO '23 Companion, página 2412–2419, New York, NY, USA, 2023. Association for Computing Machinery, ISBN 9798400701207. https://doi.org/10.1145/3583133.3596390.

[64] Schmidt, Michael e Hod Lipson: *Age-Fitness Pareto Optimization*, páginas 129–146. Springer New York, New York, NY, 2011, ISBN 978-1-4419-7747-2. https://doi.org/10.1007/978-1-4419-7747-2_8.

[65] Tian, Yuan, Wenqi Zhou, Michele Viscione, Hao Dong, David Kammer e Olga Fink: *Interactive Symbolic Regression through Offline Reinforcement Learning: A Co-Design Framework*, 2025. https://arxiv.org/abs/2402.05306.

[66] Biggio, Luca, Tommaso Bendinelli, Alexander Neitz, Aurelien Lucchi e Giambattista Parascandolo: *Neural Symbolic Regression that Scales*, 2021. https://arxiv.org/abs/2106.06427.

[67] Sahoo, Subham, Christoph Lampert e Georg Martius: *Learning Equations for Extrapolation and Control*. Em Dy, Jennifer e Andreas Krause (editores): *Proceedings of the 35th International Conference on Machine Learning*, volume 80 de *Proceedings of Machine Learning Research*, páginas 4442–4450. PMLR, 10–15 Jul 2018. https://proceedings.mlr.press/v80/sahoo18a.html.

[68] Heuristic and Evolutionary Algorithms Laboratory (HEAL): *HeuristicLab*. https://dev.heuristiclab.com/trac.fcgi/, 2009. Acesso em 10/11/2025.

[69] Keren, Liron Simon, Alex Liberzon e Teddy Lazebnik: *A computational framework for physics-informed symbolic regression with straightforward integration of domain knowledge*. Scientific Reports, 13(1), janeiro 2023, ISSN 2045-2322. http://dx.doi.org/10.1038/s41598-023-28328-2.

[70] Landajuela, Mikel, Chak Shing Lee, Jiachen Yang, Ruben Glatt, Claudio P Santiago, Ignacio Aravena, Terrell Mundhenk, Garrett Mulcahy e Brenden K Petersen: *A Unified Framework for Deep Symbolic Regression*. Em Koyejo, S., S. Mohamed, A. Agarwal, D. Belgrave, K. Cho e A. Oh (editores): *Advances in Neural Information Processing Systems*, volume 35, páginas 33985–33998. Curran Associates, Inc., 2022. https://proceedings.neurips.cc/paper_files/paper/2022/file/dbca58f35bddc6e4003b2dd80e42f838-Paper-Conference.pdf.

[71] Shojaee, Parshin, Kazem Meidani, Amir Barati Farimani e Chandan K. Reddy: *Transformer-based Planning for Symbolic Regression*, 2023. https://arxiv.org/abs/2303.06833.

[72] Zhang, Hengzhe, Aimin Zhou, Hong Qian e Hu Zhang: *PS-Tree: A piecewise symbolic regression tree*. Swarm and Evolutionary Computation, 71:101061, 2022.

[73] Kartelj, Aleksandar e Marko Djukanović: *RILS-ROLS: robust symbolic regression via iterated local search and ordinary least squares*. Journal of Big Data, 10(1):71, 2023, ISSN 2196-1115. https://doi.org/10.1186/s40537-023-00743-2.

[74] Broløs, Kevin René, Meera Vieira Machado, Chris Cave, Jaan Kasak, Valdemar Stentoft-Hansen, Victor Galindo Batanero, Tom Jelen e Casper Wilstrup: *An Approach to Symbolic Regression Using Feyn*, 2021. https://arxiv.org/abs/2104.05417.

[75] Jin, Ying, Weilin Fu, Jian Kang, Jiadong Guo e Jian Guo: *Bayesian Symbolic Regression*, 2020. https://arxiv.org/abs/1910.08892.

[76] McConaghy, Trent: *FFX: Fast, Scalable, Deterministic Symbolic Regression Technology*, páginas 235–260. Springer New York, New York, NY, 2011, ISBN 978-1-4614-1770-5. https://doi.org/10.1007/978-1-4614-1770-5_13.

[77] d'Ascoli, Stéphane, Sören Becker, Alexander Mathis, Philippe Schwaller e Niki Kilbertus: *ODEFormer: Symbolic Regression of Dynamical Systems with Transformers*, 2023. https://arxiv.org/abs/2310.05573.

[78] Brence, Jure, Ljupčo Todorovski e Sašo Džeroski: *Probabilistic grammars for equation discovery*. Knowledge-Based Systems, 224:107077, 2021, ISSN 0950-7051. https://www.sciencedirect.com/science/article/pii/S0950705121003403.

[79] Udrescu, Silviu Marian e Max Tegmark: *AI Feynman: a Physics-Inspired Method for Symbolic Regression*, 2020. https://arxiv.org/abs/1905.11481.

[80] Vastl, Martin, Jonáš Kulhánek, Jiří Kubalík, Erik Derner e Robert Babuška: *SymFormer: End-to-end symbolic regression using transformer-based architecture*, 2022. https://arxiv.org/abs/2205.15764.

[81] Grayeli, Arya, Atharva Sehgal, Omar Costilla-Reyes, Miles Cranmer e Swarat Chaudhuri: *Symbolic Regression with a Learned Concept Library*, 2024. https://arxiv.org/abs/2409.09359.

[82] OpenAI: *ChatGPT (Version 3.5) [Large language model]*. https://chat.openai.com, 2023. Acesso em 10/11/2025.

[83] Grattafiori, Aaron *et al.*: *The Llama 3 Herd of Models*, 2024. https://arxiv.org/abs/2407.21783.

[84] Kammerer, Lukas, Gabriel Kronberger, Bogdan Burlacu, Stephan M. Winkler, Michael Kommenda e Michael Affenzeller: *Symbolic Regression by Exhaustive Search: Reducing the Search Space Using Syntactical Constraints and Efficient Semantic Structure Deduplication*, páginas 79–99. Springer International Publishing, Cham, 2020, ISBN 978-3-030-39958-0. https://doi.org/10.1007/978-3-030-39958-0_5.

[85] Bartlett, Deaglan J., Harry Desmond e Pedro G. Ferreira: *Exhaustive Symbolic Regression*. IEEE Transactions on Evolutionary Computation, 28(4):950–964, 2024.

[86] Cava, William La e Joseph D. Romano: *Brush: An Interpretable Machine Learning Library for Training Symbolic Models*. `https://cavalab.org/brush/`, 2025. Acesso em 10/11/2025.

[87] La Cava, William, Thomas Helmuth, Lee Spector e Jason H. Moore: *A Probabilistic and Multi-Objective Analysis of Lexicase Selection and e-Lexicase Selection*. Evolutionary Computation, 27(3):377–402, setembro 2019, ISSN 1063-6560. `https://doi.org/10.1162/evco_a_00224`.

[88] Cava, William La, Tilak Raj Singh, James Taggart, Srinivas Suri e Jason Moore: *Learning concise representations for regression by evolving networks of trees*. Em *International Conference on Learning Representations*, 2019. `https://openreview.net/forum?id=Hke-JhA9Y7`.

[89] Virgolin, M., T. Alderliesten, C. Witteveen e P. A. N. Bosman: *Improving Model-Based Genetic Programming for Symbolic Regression of Small Expressions*. Evolutionary Computation, 29(2):211–237, junho 2021, ISSN 1063-6560. `https://doi.org/10.1162/evco_a_00278`.

[90] Zhang, Hengzhe, Aimin Zhou, Qi Chen, Bing Xue e Mengjie Zhang: *SR-Forest: A Genetic Programming-Based Heterogeneous Ensemble Learning Method*. IEEE Transactions on Evolutionary Computation, 28(5):1484–1498, 2024.

[91] Haut, Nathan, Wolfgang Banzhaf e Bill Punch: *Active Learning Improves Performance on Symbolic Regression Tasks in StackGP*. Em *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, páginas 1011–1019, 2022. `https://doi.org/10.1145/3520304.3528941`.

[92] Valipour, Mojtaba, Bowen You, Maysum Panju e Ali Ghodsi: *SymbolicGPT: A Generative Transformer Model for Symbolic Regression*, 2021. `https://arxiv.org/abs/2106.14131`.

[93] Shojaee, Parshin, Kazem Meidani, Shashank Gupta, Amir Barati Farimani e Chandan K Reddy: *LLM-SR: Scientific Equation Discovery via Programming with Large Language Models*, 2025. `https://arxiv.org/abs/2404.18400`.

[94] Kepler, J. e A.F.W. Sommer: *Harmonices Mundi (1619)*. Editiones Neolatinae. Selbstverl. Sommer, 2005, ISBN 9783902484963.

[95] Brum, Beatriz R., Luiza Lober, Isolde Previdelli e Francisco A. Rodrigues: *Discovering equations from data: symbolic regression in dynamical systems*, 2025. `https://arxiv.org/abs/2508.20257`.

[96] Brunton, Steve: *Python Symbolic Regression (PySR) [Physics Informed Machine Learning]*. YouTube, 2024. `https://www.youtube.com/watch?v=df43V4OjMVs`, Acesso em 10/11/2025.

[97] Norvig, Peter e Stuart Russell: *Inteligência Artificial: Uma Abordagem Moderna*. Elsevier, Rio de Janeiro, 3ª edição, 2013, ISBN 978-8535237016.

[98] Franca, Fabricio Olivetti de e Gabriel Kronberger: *Improving Genetic Programming for Symbolic Regression with Equality Graphs*, 2025. `https://arxiv.org/abs/2501.17848`.

[99] Scherk, Michael e Boyuan Chen: *SymMatika: Structure-Aware Symbolic Discovery*, 2025. `https://arxiv.org/abs/2507.03110`.

[100] Cranmer, Miles *et al.*: *DynamicExpressions.jl: Ridiculously fast symbolic expressions*, 2025. `https://github.com/SymbolicML/DynamicExpressions.jl`, Acesso em 10/11/2025.

[101] Darwin, Charles: *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*. John Murray, London, 1859.

[102] De Jong, Kenneth A.: *Evolutionary Computation: A Unified Approach*. MIT Press, Cambridge, MA, USA, 2016, ISBN 0262529602.

[103] Gabriel, Paulo Henrique Ribeiro e Alexandre Cláudio Botazzo Delbem: *Fundamentos de algoritmos evolutivos*. ICMC-USP, São Carlos, 2008.

[104] Fogel, Lawrence J.: *Autonomous Automata*. Industrial Research, 4(2):14–19, 1962.

[105] Rechenberg, Ingo: *Cybernetic Solution Path of an Experimental Problem*. Relatório Técnico 1122, Royal Aircraft Establishment, Farnborough, UK, 1965.

[106] Schwefel, Hans Paul: *Evolutionsstrategie und numerische Optimierung*. Tese de Doutoramento, janeiro 1975, ISBN 978-3-7643-0876-6.

[107] Koza, John R.: *Hierarchical Genetic Algorithms Operating on Populations of Computer Programs*. Em *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, páginas 185–234, San Mateo, CA, 1989. Morgan Kaufmann.

[108] Yu, Xinjie e Mitsuo Gen: *Introduction to Evolutionary Algorithms*. Decision Engineering. Springer, London, 1ª edição, 2010, ISBN 978-1-84996-128-8.

[109] Michalewicz, Zbigniew e David B. Fogel: *How to Solve It: Modern Heuristics*. Springer, Berlin, Heidelberg, 2ª edição, 2006, ISBN 978-3-7908-1771-3.

[110] Eiben, A.E. e J.E. Smith: *Introduction to Evolutionary Computing*. Springer, Berlin, Heidelberg, 2ª edição, 2015, ISBN 978-3-662-44874-8.

[111] Atkinson, Steven, Waad Subber, Liping Wang, Genghis Khan, Philippe Hawi e Roger Ghanem: *Data-driven discovery of free-form governing differential equations*, 2019. `https://arxiv.org/abs/1910.05117`.

[112] Grefenstette, John J.: *Optimization of Control Parameters for Genetic Algorithms*. IEEE Transactions on Systems, Man, and Cybernetics, 16(1):122–128, 1986.

[113] Hassanat, Ahmad, Khalid Almohammadi, Esra'a Alkafaween, Eman Abunawas, Awni Hammouri e V. B. Surya Prasath: *Choosing Mutation and Crossover Ratios for Genetic Algorithms—A Review with a New Dynamic Approach*. Information, 10(12), 2019, ISSN 2078-2489. `https://www.mdpi.com/2078-2489/10/12/390`.

[114] Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest e Clifford Stein: *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edição, 2009, ISBN 0262033844.

[115] Knuth, Donald E.: *The Art of Computer Programming, Volume 4A: Combinatorial Algorithms, Part 1*. Addison-Wesley Professional, Boston, MA, 2011, ISBN 9780201896831.

[116] Nocedal, Jorge e Stephen J. Wright: *Numerical Optimization*. Springer, New York, NY, 2ª edição, 2006, ISBN 978-0-387-30303-1.

[117] Mogensen, Patrick Kofod e Asbjørn Nilsen Riseth: *Optim: A mathematical optimization package for Julia*. Journal of Open Source Software, 3(24):615, 2018.

[118] Kommenda, Michael, Michael Affenzeller, Gabriel Kronberger, Bogdan Burlacu e Stephan Winkler: *Multi-Population Genetic Programming with Data Migration for Symbolic Regression*, páginas 75–87. Springer International Publishing, Cham, 2015, ISBN 978-3-319-15720-7. https://doi.org/10.1007/978-3-319-15720-7_6.

[119] Kommenda, Michael, Michael Affenzeller, Bogdan Burlacu, Gabriel Kronberger e Stephan M. Winkler: *Genetic programming with data migration for symbolic regression*. Em *Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation*, GECCO Comp '14, página 1361–1366, New York, NY, USA, 2014. Association for Computing Machinery, ISBN 9781450328814. https://doi.org/10.1145/2598394.2609857.

[120] Xu, Haoran, Kenton Murray, Philipp Koehn, Hieu Hoang, Akiko Eriguchi e Huda Khayrallah: *X-ALMA: Plug and Play Modules and Adaptive Rejection for Quality Translation at Scale*, 2025. https://arxiv.org/abs/2410.03115.

[121] OpenAI: *GPT-4 Technical Report*, 2024. https://arxiv.org/abs/2303.08774.

[122] Devlin, Jacob, Ming Wei Chang, Kenton Lee e Kristina Toutanova: *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, 2019. https://arxiv.org/abs/1810.04805.

[123] Touvron, Hugo, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave e Guillaume Lample: *LLaMA: Open and Efficient Foundation Language Models*, 2023. https://arxiv.org/abs/2302.13971.

[124] Guo, Daya et al.: *DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning*. Nature, 645(8081):633–638, 2025, ISSN 1476-4687. http://dx.doi.org/10.1038/s41586-025-09422-z.

[125] Ning, Emma, Nathan Yan, Jeffrey Zhu e Jason Li: *Microsoft open sources breakthrough optimizations for transformer inference on GPU and CPU*. Microsoft Open Source Blog, janeiro 2020. https://opensource.microsoft.com/blog/2020/01/21/microsoft-onnx-open-source-optimizations-transformer-inference-gpu-cpu/, Acesso em 14/03/2026.

[126] Bahdanau, Dzmitry, Kyunghyun Cho e Yoshua Bengio: *Neural Machine Translation by Jointly Learning to Align and Translate*. Em *3rd International Conference on Learning Representations (ICLR 2015)*, 2015. https://arxiv.org/abs/1409.0473.

[127] Rumelhart, David E., Geoffrey E. Hinton e Ronald J. Williams: *Learning representations by back-propagating errors*. Nature, 323(6088):533–536, 1986, ISSN 1476-4687. https://doi.org/10.1038/323533a0.

[128] Cho, Kyunghyun, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk e Yoshua Bengio: *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*, 2014. https://arxiv.org/abs/1406.1078.

[129] Sutskever, Ilya, Oriol Vinyals e Quoc V. Le: *Sequence to Sequence Learning with Neural Networks*, 2014. https://arxiv.org/abs/1409.3215.

[130] Zhang, Aston, Zachary C. Lipton, Mu Li e Alexander J. Smola: *Dive into Deep Learning*. arXiv preprint arXiv:2106.11342, 2021.

[131] Kim, Yoon, Yacine Jernite, David Sontag e Alexander M. Rush: *Character-Aware Neural Language Models*, 2015. https://arxiv.org/abs/1508.06615.

[132] Hochreiter, Sepp e Jürgen Schmidhuber: *Long Short-Term Memory*. Neural Computation, 9(8):1735–1780, novembro 1997.

[133] Zeiler, Matthew D.: *ADADELTA: An Adaptive Learning Rate Method*, 2012. https://arxiv.org/abs/1212.5701.

[134] Stanford Online: *Stanford CS224N: NLP with Deep Learning | 2023 | Lecture 8 - Self-Attention and Transformers*. YouTube, 2023. https://www.youtube.com/watch?v=LWMzyfvuehA, Aula da disciplina CS224N: Natural Language Processing with Deep Learning, Stanford University, https://web.stanford.edu/class/cs224n/. Acesso em 13/03/2026.

[135] Silberschatz, Abraham, Henry F. Korth e S. Sudarshan: *Sistema de Banco de Dados*. Elsevier/Campus, Rio de Janeiro, 6ª edição, 2012, ISBN 978-8535245356.

[136] He, Kaiming, Xiangyu Zhang, Shaoqing Ren e Jian Sun: *Deep Residual Learning for Image Recognition*, 2015. https://arxiv.org/abs/1512.03385.

[137] Ba, Jimmy Lei, Jamie Ryan Kiros e Geoffrey E. Hinton: *Layer Normalization*, 2016. https://arxiv.org/abs/1607.06450.

[138] Sennrich, Rico, Barry Haddow e Alexandra Birch: *Neural Machine Translation of Rare Words with Subword Units*, 2016. https://arxiv.org/abs/1508.07909.

[139] Wu, Yonghui, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes e Jeffrey Dean: *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*, 2016. https://arxiv.org/abs/1609.08144.

[140] Shojaee, Parshin, Ngoc Hieu Nguyen, Kazem Meidani, Amir Barati Farimani, Khoa D Doan e Chandan K Reddy: *LLM-SRBench: A New Benchmark for Scientific Equation Discovery with Large Language Models*, 2025. https://arxiv.org/abs/2504.10415.

[141] Knuth, Donald E.: *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. Addison-Wesley Professional, Reading, MA, 3ª edição, 1997, ISBN 978-0201896848.

[142] Charton, François: *Linear algebra with transformers*, 2022. https://arxiv.org/abs/2112.01898.

[143] Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot e E. Duchesnay: *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12:2825–2830, 2011.

[144] Lample, Guillaume e François Charton: *Deep Learning for Symbolic Mathematics*. ArXiv, abs/1912.01412, 2019. https://api.semanticscholar.org/CorpusID:208547770.

[145] Deng, Jia, Wei Dong, Richard Socher, Li Jia Li, Kai Li e Li Fei-Fei: *ImageNet: A large-scale hierarchical image database*. Em *2009 IEEE Conference on Computer Vision and Pattern Recognition*, páginas 248–255, 2009.

[146] Everingham, Mark, Luc Van Gool, Christopher K. I. Williams, John Winn e Andrew Zisserman: *The PASCAL Visual Object Classes (VOC) Challenge*. International Journal of Computer Vision, 88(2):303–338, 2010.

[147] Lin, Tsung Yi, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár e C. Lawrence Zitnick: *Microsoft COCO: Common Objects in Context*. Em *European Conference on Computer Vision (ECCV)*, páginas 740–755. Springer, 2014.

[148] McDermott, James, David R. White, Sean Luke, Luca Manzoni, Mauro Castelli, Leonardo Vanneschi, Wojciech Jaskowski, Krzysztof Krawiec, Robin Harper, Kenneth De Jong e Una May O'Reilly: *Genetic programming needs better benchmarks*. Em *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*, GECCO '12, página 791–798, New York, NY, USA, 2012. Association for Computing Machinery, ISBN 9781450311779. https://doi.org/10.1145/2330163.2330273.

[149] White, David R., James McDermott, Mauro Castelli, Luca Manzoni, Brian W. Goldman, Gabriel Kronberger, Wojciech Jaśkowski, Una May O'Reilly e Sean Luke: *Better GP benchmarks: community survey results and proposals*. Genetic Programming and Evolvable Machines, 14(1):3–29, 2013.

[150] Abadi, Martín, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete

Warden, Martin Wattenberg, Martin Wicke, Yuan Yu e Xiaoqiang Zheng: *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*, 2015. `https://www.tensorflow.org/`, Software available from tensorflow.org.

[151] Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai e Soumith Chintala: *PyTorch: An Imperative Style, High-Performance Deep Learning Library*, 2019. `https://arxiv.org/abs/1912.01703`.

[152] Feldt, Robert, Michael O'Neill, Conor Ryan, Peter Nordin e William B. Langdon: *GP-Beagle: A Benchmarking Problem Repository for the Genetic Programming Community*. Em Whitley, Darrell (editor): *Late Breaking Papers at the 2000 Genetic and Evolutionary Computation Conference (GECCO)*, páginas 90–97, Las Vegas, Nevada, USA, 2000.

[153] Meurer, Aaron, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman e Anthony Scopatz: *SymPy: symbolic computing in Python*. PeerJ Computer Science, 3:e103, janeiro 2017, ISSN 2376-5992. `https://doi.org/10.7717/peerj-cs.103`.

[154] Olson, Randal S., William La Cava, Patryk Orzechowski, Ryan J. Urbanowicz e Jason H. Moore: *PMLB: A Large Benchmark Suite for Machine Learning Evaluation and Comparison*, 2017. `https://arxiv.org/abs/1703.00512`.

[155] Strogatz, Steven H.: *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*. Studies in Nonlinearity. Westview Press, Boulder, CO, 2ª edição, 2014, ISBN 978-0813349107.

[156] MacQueen, James: *Some Methods for Classification and Analysis of Multivariate Observations*. Em Le Cam, Lucien M. e Jerzy Neyman (editores): *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, páginas 281–297, Berkeley, CA, 1967. University of California Press.

[157] Maaten, Laurens van der e Geoffrey Hinton: *Visualizing Data using t-SNE*. Journal of Machine Learning Research, 9:2579–2605, 2008.

[158] Russeil, Etienne, Fabricio Olivetti de Franca, Konstantin Malanchev, Bogdan Burlacu, Emille Ishida, Marion Leroux, Clément Michelin, Guillaume Moinard e Emmanuel Gangler: *Multiview Symbolic Regression*. Em *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '24, página 961–970, New York, NY, USA, 2024. Association for Computing Machinery, ISBN 9798400704949. `https://doi.org/10.1145/3638529.3654087`.

[159] Hall, Mark, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann e Ian H. Witten: *The WEKA Data Mining Software: An Update*. SIGKDD Explorations, 11(1):10–18, 2009.

[160] Cardoso, Jaime S. e Ricardo Sousa: *Measuring the Performance of Ordinal Classification*. International Journal of Pattern Recognition and Artificial Intelligence, 25(8):1173–1195, 2011.

[161] Hand, D. J., F. Daly, A. D. Lunn, K. J. McConway e E. Ostrowski: *A Handbook of Small Data Sets*. Chapman & Hall, London, 1994.

[162] Hosmer, David W., Stanley Lemeshow e Rodney X. Sturdivant: *Applied Logistic Regression*. Wiley Series in Probability and Statistics. John Wiley & Sons, Hoboken, NJ, 3ª edição, 2013, ISBN 978-0470582473.

[163] Simonoff, Jeffrey S.: *Smoothing Methods in Statistics*. Springer Series in Statistics. Springer-Verlag, New York, 1996, ISBN 978-0387947167.

[164] Miller, A. J., D. E. Shaw, L. G. Veitch e E. J. Smith: *Analyzing the results of a cloud-seeding experiment in Tasmania*. Communications in Statistics - Theory and Methods, 8(10):1017–1047, 1979.

[165] Aldrin, Magne: *Improved predictions penalizing both slope and curvature in additive models*. Computational Statistics and Data Analysis, 50(2):267–284, 2006, ISSN 0167-9473. `https://www.sciencedirect.com/science/article/pii/S0167947304002476`.

[166] Simonoff, Jeffrey S.: *Analyzing Categorical Data*. Springer Texts in Statistics. Springer, New York, 2003, ISBN 978-0387007496.

[167] Friedman, Jerome H.: *Multivariate Adaptive Regression Splines*. The Annals of Statistics, 19(1):1–67, 1991.

[168] Bruntz, S. M., W. S. Cleveland, T. E. Graedel, B. Kleiner e J. L. Warner: *Ozone Concentrations in New Jersey and New York: Statistical Association with Related Variables*. Science, 186(4160):257–259, outubro 1974.

[169] Chambers, John M., William S. Cleveland, Beat Kleiner e Paul A. Tukey: *Graphical Methods for Data Analysis*. Wadsworth, Belmont, CA, 1983, ISBN 978-0534980528.

[170] Bonnet, Charles: *Contemplation de la nature*, volume 2. Marc-Michel Rey, Amsterdam, 1764. `https://books.google.com/books?id=Sm8GAAAAQAAJ`.

[171] Hubble, Edwin: *A Relation between Distance and Radial Velocity among Extra-Galactic Nebulae*. Proceedings of the National Academy of Sciences, 15(3):168–173, março 1929.

[172] Clapeyron, Émile: *Mémoire sur la puissance motrice de la chaleur*. Journal de l'École Polytechnique, 14:153–190, 1834.

[173] Leavitt, Henrietta S. e Edward C. Pickering: *Periods of 25 Variable Stars in the Small Magellanic Cloud*. Harvard College Observatory Circular, 173:1–3, março 1912.

[174] Newton, Isaac: *Philosophiæ Naturalis Principia Mathematica*. Jussu Societatis Regiæ ac Typis Joseph Streater, London, England, 1687.

[175] Planck, Max: *The Theory of Heat Radiation*. P. Blakiston's Son & Co., Philadelphia, 2ª edição, 1914. Translated by Morton Masius.

[176] Rydberg, Johannes R.: *Recherches sur la constitution des spectres d'émission des éléments chimiques*. Kongliga Svenska Vetenskaps-Akademiens Handlingar, 23(11), 1889.

[177] Press, William H. e Paul Schechter: *Formation of Galaxies and Clusters of Galaxies by Self-Similar Gravitational Condensation*. The Astrophysical Journal, 187:425–438, fevereiro 1974.

[178] Tully, R. B. e J. R. Fisher: *A New Method of Determining Distances to Galaxies*. Astronomy and Astrophysics, 54(3):661–673, fevereiro 1977.

[179] Matsubara, Yoshitomo, Naoya Chiba, Ryo Igarashi e Yoshitaka Ushiku: *Rethinking Symbolic Regression Datasets and Benchmarks for Scientific Discovery*. Journal of Data-centric Machine Learning Research, 2024. https://openreview.net/forum?id=qrUdrXsiXX.

[180] Xu, Yilong, Kai Ruan, Mengzhen Liu e Hao Sun: *GeoBench: A new benchmark on Symbolic Regression with Geometric Expressions*, 2025. https://openreview.net/forum?id=TqzNI4v9DT.

[181] França, Fabrício Olivetti de, Marco Virgolin, Michael Kommenda, M. S. Majumder, Miles Cranmer, Guilherme Espada, Leon Ingelse, Alcides Fonseca, Mikel Landajuela, Brenden Petersen, Ruben Glatt, Nathan Mundhenk, Chak Shing Lee, Jacob D. Hochhalter, David L. Randall, Pierre Alexandre Kamienny, Hengzhe Zhang, Grant Dick, Andrew Simon, Bogdan Burlacu, Jaan Kasak, Meera Machado, Casper Wilstrup e William G. La Cava: *SRBench++: Principled Benchmarking of Symbolic Regression with Domain-Expert Interpretation*. IEEE Transactions on Evolutionary Computation, 29(4):1127–1134, agosto 2025.

[182] Mens, Tom e Alexandre Decan: *An Overview and Catalogue of Dependency Challenges in Open Source Software Package Registries*, 2024. https://arxiv.org/abs/2409.18884.

[183] Schueller, William e Johannes Wachs: *Modeling interconnected social and technical risks in open source software ecosystems*. Collective Intelligence, 3(1), 2024.

[184] Schueller, William, Johannes Wachs, Vito D. P. Servedio, Stefan Thurner e Vittorio Loreto: *Evolving collaboration, dependencies, and use in the Rust Open Source Software ecosystem*. Scientific Data, 9(1), novembro 2022, ISSN 2052-4463. http://dx.doi.org/10.1038/s41597-022-01819-z.

[185] Dolan, Elizabeth D. e Jorge J. Moré: *Benchmarking Optimization Software with Performance Profiles*, 2004. https://arxiv.org/abs/cs/0102001.

[186] Alrashidi, Huda, Fadi N. Sibai, Abdullah Abonamah, Mufreh Alrashidi e Ahmad Alsaber: *PM2.5: Air Quality Index Prediction Using Machine Learning: Evidence from Kuwait's Air Quality Monitoring Stations*. Sustainability, 17(20), 2025, ISSN 2071-1050. https://www.mdpi.com/2071-1050/17/20/9136.

[187] Sokhi, Ranjeet *et al.*: *Air Quality: Science, Impacts, and Management*. Elsevier, Amsterdam, Netherlands, 1ª edição, 2022, ISBN 978-0-12-822591-2.

[188] Lou, Cairong, Hongyu Liu, Yufeng Li, Yan Peng, Juan Wang e Lingjun Dai: *Relationships of relative humidity with PM2.5 and PM10 in the Yangtze River Delta, China*. Environmental Monitoring and Assessment, 189(11):582, Oct 2017, ISSN 1573-2959. https://doi.org/10.1007/s10661-017-6281-z.

[189] Lagesse, Brent, Shuoqi Wang, Timothy V. Larson e Amy Ahim Kim: *Performing indoor PM2.5 prediction with low-cost data and machine learning*. Facilities, 40(78):495–514, 2022, ISSN 0263-2772. https://www.sciencedirect.com/science/article/pii/S0263277222000277.

[190] Yang, Hongmei, Qin Peng, Jun Zhou, Guojun Song e Xinqi Gong: *The unidirectional causality influence of factors on PM2.5 in Shenyang city of China*. Scientific Reports, 10(1):8403, May 2020, ISSN 2045-2322. https://doi.org/10.1038/s41598-020-65391-5.

[191] Oke, T. R., G. Mills, A. Christen e J. A. Voogt: *Urban Climates*. Cambridge University Press, 2017.

[192] California Air Resources Board: *Inhalable Particulate Matter and Health (PM2.5 and PM10)*, n.d. https://ww2.arb.ca.gov/resources/inhalable-particulate-matter-and-health, Acesso em 01/10/2025.

[193] Xing, Yu Fei, Yue Hua Xu, Min Hua Shi e Yi Xin Lian: *The impact of PM2.5 on the human respiratory system*. Journal of Thoracic Disease, 8(1):E69–E74, janeiro 2016, ISSN 2072-1439. Review.

[194] Companhia Ambiental do Estado de São Paulo (CETESB): *Qualar – Qualidade do Ar*, n.d. https://cetesb.sp.gov.br/ar/qualar/, Acesso em 01/10/2025.

[195] Instituto Nacional de Meteorologia (INMET): *Tabela de Estações: A701*, n.d. https://tempo.inmet.gov.br/TabelaEstacoes/A701, Acesso em 01/10/2025.

[196] Wallace, John M. e Peter V. Hobbs: *7 - Atmospheric Dynamics*. Em Wallace, John M. e Peter V. Hobbs (editores): *Atmospheric Science (Second Edition)*, páginas 271–311. Academic Press, San Diego, second edition edição, 2006, ISBN 978-0-12-732951-2. https://www.sciencedirect.com/science/article/pii/B9780127329512500120.

[197] Liu, Yansui, Yang Zhou e Jiaxin Lu: *Exploring the relationship between air pollution and meteorological conditions in China under environmental governance*. Scientific Reports, 10(1):14518, Sep 2020, ISSN 2045-2322. https://doi.org/10.1038/s41598-020-71338-7.

[198] Albuquerque, Tatiana *et al.*: *Analysis of PM2.5 concentrations under pollutant emission control strategies in the metropolitan area of São Paulo, Brazil*. Environmental Science and Pollution Research International, 26(32):33216–33227, novembro 2019, ISSN 0944-1344. Epub 2019 Sep 13.

[199] CETESB - Companhia Ambiental do Estado de São Paulo: *Qualidade do ar no estado de São Paulo 2014*. CETESB, São Paulo, Brazil, 2015. https://repositorio.cetesb.sp.gov.br/handle/123456789/2285, Série Relatórios.

[200] Derryberry, DeWayne R.: *Basic Data Analysis for Time Series with R*. Wiley, Hoboken, NJ, 1ª edição, 2014, ISBN 978-1-118-42254-0.

[201] Maslej, Nestor, Loredana Fattorini, Raymond Perrault *et al.*: *The AI Index 2025 Annual Report*. Relatório Técnico, Stanford Institute for Human-Centered Artificial Intelligence (HAI), Stanford University, Stanford, CA, 2025. `https://hai.stanford.edu/ai-index/2025-ai-index-report`.