UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE FÍSICA DE SÃO CARLOS

Alisson Vinícius Gonsales

Modelos dirigidos por dados na física de sistemas dinâmicos

São Carlos

2022

Alisson Vinícius Gonsales

Modelos dirigidos por dados na física de sistemas dinâmicos

Trabalho de conclusão de curso apresentado ao Instituto de Física de São Carlos da Universidade de São Paulo para obtenção do título de Bacharel em Ciências Físicas e Biomoleculares.

Orientador: Prof. Dr. Fernando Fagundes Ferreira

Versão original

São Carlos

2022

1.1 Resumo

Com o avanço das técnicas de Machine Learning, há a possibilidade de encontrar as equações que descrevem a dinâmica de um sistema dinâmico de forma automática a partir dos dados. Os métodos propostos para essa tarefa devem ser capazes de reduzir o espaço de soluções possíveis a expressões com um pequeno número de termos. A resistência a ruído, geralmente proveniente das medidas em física é uma propriedade desejável. Neste trabalho alguns métodos disponíveis na literatura, incluindo um método de regressão esparsa (1) e métodos de Deep Learning (2) (3) são testados e comparados. Ao fim dos testes, concluiu-se que um dos métodos se mostra ineficaz, embora tenha suporte na literatura. Um método de Deep Learning se mostrou mais resistente a ruído em relação ao método de regressão esparsa. A comparação de métodos nesse campo de estudo é importante para indicar perspectivas para trabalhos futuros.

1.2 Introdução

No começo do século XVII, Johannes Kepler foi capaz de derivar leis para o movimento planetário a partir de um grande volume de dados vindos da observação de suas órbitas. Posteriormente, Isaac Newton foi capaz de deduzir a lei do inverso do quadrado da distância, que regia as órbitas celestes. O avanço de algoritmos de aprendizado de máquina hoje, quatro séculos depois, abre possibilidades para automatização de descobertas como a de Newton partindo de modelos indutivos como o de Kepler.

Com o objetivo de recuperar as equações da dinâmica de um sistema físico qualquer a partir dos dados, o primeiro passo é partir de sistemas bem conhecidos. (4) estudou o problema de objetos em queda e abordou as dificuldades a serem superadas como a presença de ruído em medidas e a existência de mecanismos complexos de segunda ordem em física. (5) utilizou uma técnica semelhante e aplicou a problemas de dinâmica não linear como osciladores e o sistema caótico de Lorenz, enfatizando a importância da descoberta de modelos com poucos termos. Em (6), foi utilizado uma rede neural de grafos para recuperar a lei do inverso do quadrado da distância e massas planetárias utilizando os princípios de equivariância translacional e rotacional.

(3), (7), (8) utilizaram uma abordagem na qual as funções de ativação de uma rede neural são dadas por operadores comuns em equações físicas, com aplicações em dinâmica, cinemática, estudo do clima e problemas de controle. (9) e (3) utilizam uma arquitetura semelhante baseada em dois passos: a extração de características dos dados espaço-temporais seguida por simulação de trajetórias a partir de uma condição inicial com métodos de Deep Learning. (10) também utilizou um método baseado em redes neurais explorando propriedades simplificativas nos problemas físicos, como simetrias, separabilidade e composição e em um conjunto de teste considerado mais difícil, foi capaz de melhorar o estado da arte de 15% para 90% de taxa de acerto. (11) e (2) utilizaram um método misto de redes

neurais com aprendizado por reforço e algoritmos genéticos para encontrar as melhores árvores de expressão que descreviam os dados. O mesmo método de regressão esparsa utilizado em (4), (5) é modificado em (12) para incorporar resultados de teoria de estabilidade de sistemas dinâmicos.
Há obstáculos envolvidos em encontrar expressões analíticas para dados em física e na capacidade de extrapolação. Embora haja muitos métodos já propostos com bons resultados, o avanço das técnicas de aprendizado de máquina incentiva o desenvolvimento de novos modelos que superem esses obstáculos. Nesse sentido, a comparação entre métodos propostos previamente se faz necessária para guiar o campo de estudo por vias mais promissoras.

1.3 Metodologia

A Metodologia do presente trabalho se divide em duas partes, sendo a primeira a descrição de alguns métodos empregados anteriormente e disponíveis na literatura e a segunda um conjunto de testes afim de avaliar e comparar o desempenho das soluções instanciadas na primeira parte.

1.3.1 Revisão da literatura

Na literatura já foram empregados diferentes métodos para problemas de regressão simbólica e construção de modelos dirigidos por dados. Nesta seção são descritos alguns desses métodos e técnicas envolvidas.

1.3.1.1 Esparsidade

Quando há diferentes modelos para um problema particular em física, opta-se pelo modelo mais simples. Um problema a ser considerado pelos métodos que buscam expressões analiticas para os dados é encontrar alternativas com um baixo número de termos(esparsas), previnindo overfitting e permitindo a extrapolação do modelo, que é um dos objetivos principais. Os métodos que garantem esparsidade são em geral regularizadores que forçam a solução a ter o maior número de parâmetros possível igual a zero.

1.3.1.2 Regularizadores L_p

A esparsidade pode ser obtida adicionando termos que penalizam parâmetros diferentes de zero. Normas L_p são regularizadores frequentemente utilizados (3) (13). Seja um modelo com uma coleção de parâmetros $\boldsymbol{\theta}$

$$L_p = \sum_{i=0}^{m} |\theta_i|^p, \tag{1.1}$$

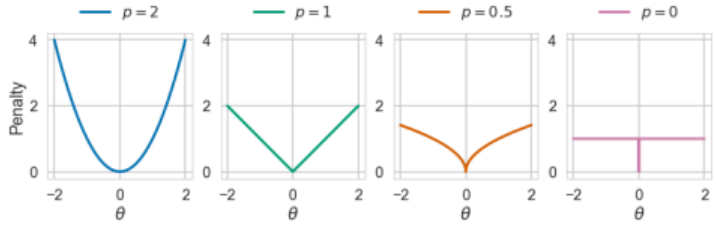
O treinamento do modelo é feito incluindo-se o termo regularizador à função objetivo. Isto é, dada um conjunto de pontos $\{(x_1, y_1), \ldots, (x_N, y_N)\}$ e uma hipótese h(.;$\boldsymbol{\theta}$), Definimos a função perda regularizada e o objetivo como:

$$\mathcal{R}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(h(x_i;\boldsymbol{\theta}), y_i) + \lambda||\boldsymbol{\theta}||_p, \tag{1.2}$$

$$\boldsymbol{\theta}^* = argmin\boldsymbol{\theta}\{\mathcal{R}(\boldsymbol{\theta})\},$$

Na qual λ controla a magnitude da penalidade. Exemplos conhecidos desse tipo de regularizador são o LASSO(p=1) e Weight Decay(p=2) (Figura 1)

Figura 1 – Normas L_p para p de 0 a 2.



Fonte: (LOUIZOS, C. et al. 2018)

Problemas em utilizar a norma L_0 envolvem sua diferenciabilide, requesito necessário para métodos guiados por otimização do gradiente, e a natureza combinatória das 2^m possibilidades, que torna o problema intratável computacionalmente. Utilizar essa norma é desejável pois ela garante que os termos convirgam diretamente para zero. Formas de suavizar essa norma foram empregados anteriormente (13) e são descritos aqui. Considera-se uma reparametrização do vetor de parâmetros $\boldsymbol{\theta}$ inserindo variáveis latentes $\boldsymbol{z_i}$:

$$\begin{aligned} \theta_j &= z_j\tilde{\theta}_j, \\ \tilde{\theta}_j &\neq 0, \\ z_j &\in \{0, 1\}, \\ ||\boldsymbol{\theta}||_0 &= \sum_{j=1}^{m} z_j, \end{aligned} \tag{1.3}$$

As variáveis z_j são chamados "gates"e aplicar a norma L_0 consiste em contar quantos gates estão abertos. É possivel tomar z_j de uma distribuição q($z_j|\pi_j$). Em particular, fazendo $z_j \sim$ Bernoulli(π_j) é possível reformular o problema de otimização:

$$\mathcal{R}(\tilde{\boldsymbol{\theta}}, \boldsymbol{\pi}) = \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{\pi})}\left[\frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(h(x_i;\tilde{\boldsymbol{\theta}}\odot\boldsymbol{z}), y_i)\right] + \lambda\sum_{i=1}^{m}\pi_j, \tag{1.4}$$

$$\boldsymbol{\theta}^*, \boldsymbol{\pi}^* = argmin\tilde{\boldsymbol{\theta}}, \boldsymbol{\pi}\{\mathcal{R}(\tilde{\boldsymbol{\theta}}, \boldsymbol{\pi})\},$$

O primeiro termo de 1.3.1.2 pode ser difícil de diferenciar devido a natureza binária de $\boldsymbol{z}$. Uma forma de contornar isso é tomando uma variável $\boldsymbol{s}$ tal que:

$$\begin{aligned} &\boldsymbol{s} \sim q(s|\phi), \\ &\boldsymbol{z} = g(s) = min\{\mathbf{1}, max(\mathbf{0}, \boldsymbol{s})\}, \end{aligned} \tag{1.5}$$

A variável aleatória contínua $\boldsymbol{s}$ permite que a probabilidade da variável $\boldsymbol{z}$ ser não-zero seja calculada através da acumulada:

$$q(\boldsymbol{z} \neq 0|\boldsymbol{\phi}) = 1 - Q(\boldsymbol{s} \leq 0|\boldsymbol{\phi}),$$

Com isso, o problema de otimização se torna:

$$\begin{aligned} \mathcal{R}(\tilde{\boldsymbol{\theta}}, \boldsymbol{\phi}) = \mathbb{E}_{q(\boldsymbol{s}|\boldsymbol{\phi})} \left[\frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(h(x_i; \tilde{\boldsymbol{\theta}} \odot \boldsymbol{g}(\boldsymbol{s}), y_i) \right] + \\ \lambda \sum_{j=1}^{m} \left(1 - Q(\boldsymbol{s_j} \leq 0|\boldsymbol{\phi_j}) \right), \\ \boldsymbol{\theta}^*, \boldsymbol{\phi}^* = argmin\tilde{\boldsymbol{\theta}}, \boldsymbol{\phi}\{\mathcal{R}(\tilde{\boldsymbol{\theta}}, \boldsymbol{\phi})\}, \end{aligned} \tag{1.6}$$

Com essa nova norma temos a penalização da probabilidade dos parâmetros serem diferentes de zero. A distribuição q($\boldsymbol{s}$) permite reparametrização. Ao fim, teremos uma expressão para $\mathcal{R}(\tilde{\boldsymbol{\theta}}, \boldsymbol{\phi})$ dada por:

$$\mathcal{R}(\tilde{\boldsymbol{\theta}}, \boldsymbol{\phi}) = \mathcal{L}_E(\tilde{\boldsymbol{\theta}}, \boldsymbol{\phi}) + \lambda \mathcal{L}_C(\phi), \tag{1.7}$$

O primeiro termo reflete em quão bem a hipótese descreve os dados, ou seja, a acurácia do modelo, enquanto o segundo diz sobre sua esparsidade.

1.3.1.3 SINDy

SINDy (Sparse Identification of Nonlinear Dynamics)(1) é um método de regressão esparsa que busca reconstruir a dinâmica de um sistema a partir dos dados utilizando um número pequeno de termos. A princípio, a aplicação do método se dá para sistemas regidos por:

$$\frac{d}{dt}\mathbf{x(t)} = \mathbf{f}(\mathbf{x}(t)), \tag{1.8}$$

Em que $\mathbf{x}(\mathrm{t}) \in \mathbb{R}^n$ é o estado do sistema com dinâmica dada por uma função não-linear $\mathbf{f}$. Para encontrar $\mathbf{f}$, um conjunto de medidas do estado do sistema é coletado em diferentes instantes de tempo $t_1, t_2, ...t_m$ e a derivada $\dot{x}$ é obtida por medidas diretas ou diferenciação numérica. Assim, é obtida a matriz:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x^T}(t_1) \\ \mathbf{x^T}(t_2) \\ \vdots \\ \mathbf{x^T}(t_m) \end{bmatrix} = \begin{bmatrix} x_1(t_1) & x_2(t_1) & \dots & x_n(t_1) \\ x_1(t_2) & x_2(t_2) & \dots & x_n(t_2) \\ \vdots & \vdots & \vdots & \vdots \\ x_1(t_m) & x_2(t_m) & \dots & x_n(t_m) \end{bmatrix}, \tag{1.9}$$

$$\dot{\mathbf{X}} = \begin{bmatrix} \dot{\mathbf{x}}^{\mathbf{T}}(t_1) \\ \dot{\mathbf{x}}^{\mathbf{T}}(t_2) \\ \vdots \\ \dot{\mathbf{x}}^{\mathbf{T}}(t_m) \end{bmatrix} = \begin{bmatrix} \dot{x}_1(t_1) & \dot{x}_2(t_1) & \dots & \dot{x}_n(t_1) \\ \dot{x}_1(t_2) & \dot{x}_2(t_2) & \dots & \dot{x}_n(t_2) \\ \vdots & \vdots & \vdots & \vdots \\ \dot{x}_1(t_m) & \dot{x}2(t_m) & \dots & \dot{x}_n(t_m) \end{bmatrix}, \tag{1.10}$$

Em seguida, é construído um dicionário de funções não lineares candidatas aplicadas às colunas de $\mathbf{X}$ podendo incluir termos polinomiais, trigonométricos, constantes, etc.

$$\Theta(\mathbf{X}) = \begin{bmatrix} | & | & | & & | & | & \\ \mathbf{1} & \mathbf{X} & \mathbf{X}^2 & \dots & \mathbf{sin}(\mathbf{X}) & \mathbf{cos}(\mathbf{X}) & \dots \\ | & | & | & & | & | & \end{bmatrix}, \tag{1.11}$$

O problema de regressão esparsa consiste em identificar os coeficientes $[\xi_1\ \xi_2\ \dots\ \xi_n]$ da matriz de coeficientes $\boldsymbol{\Xi}$ tais que:

$$\dot{\mathbf{X}} = \Theta(\mathbf{X})\boldsymbol{\Xi}, \tag{1.12}$$

Ou seja, temos o problema de otimização:

$$\boldsymbol{\Xi} = argmin_{\boldsymbol{\Xi}}\{||\dot{\mathbf{X}} - \Theta(\mathbf{X})\boldsymbol{\Xi}||_2^2 + \lambda||\boldsymbol{\Xi}||_1\}, \tag{1.13}$$

No qual o segundo termo é o regularizador, usualmente a norma L_1(LASSO)

1.3.1.4 Redes Neurais EQL

As redes neurais de regressão simbólica Equation Learning(EQL)[(3)] possuem arquiteturas com camadas totalmente conectadas. A i-ésima camada escondida é descrita por:

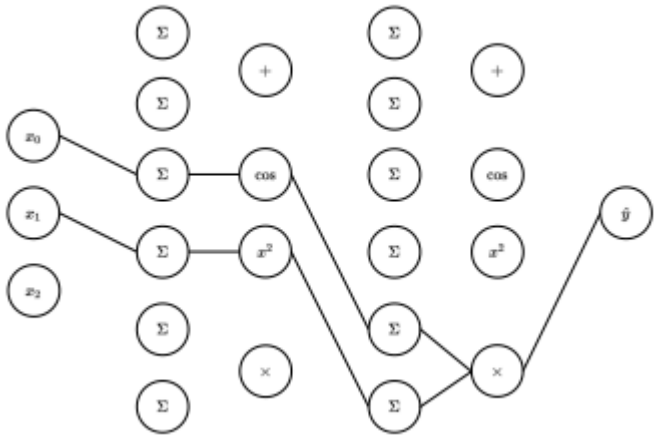
$$u_i = \boldsymbol{W_i h_{i-1}}, \tag{1.14}$$

$$\boldsymbol{h_i} = \mathbf{f}(u_i),$$

Em que $\boldsymbol{W_i}$ é a matriz de pesos na i-ésima camada, $\boldsymbol{h_0}$ são os dados de entrada. Diferente de outras redes neurais conhecidas, nas quais a função $\boldsymbol{f(.)}$ está entre um conjunto de funções comuns como a ReLU, sigmóide, etc. As redes EQL possuem funções de ativação dadas por operadores que vão desde a funções trigonométricas a polinômios e operadores binários aritméticos como soma. A camada final (output-layer) terá sáida dada por:

$$y = h_{m+1} = \boldsymbol{W_{m+1} h_m}, \tag{1.15}$$

Figura 2 – Rede EQL e caminho percorrido para encontrar uma função: $x_1^2 cos(x_0)$

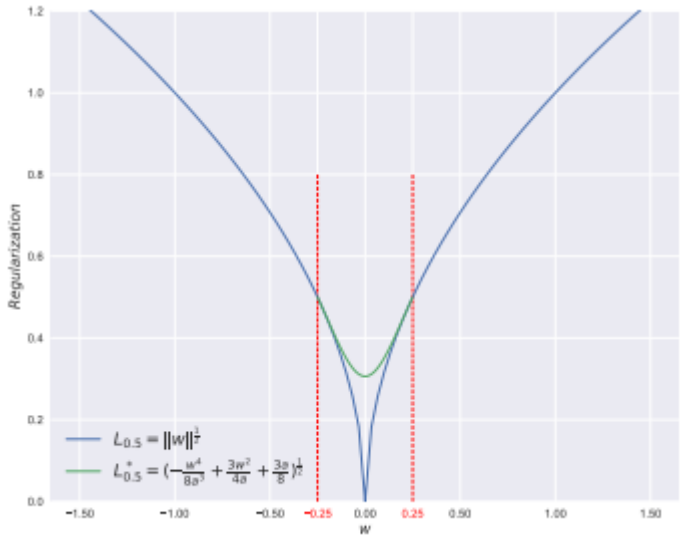


Fonte: (ABDELLAOUI, I. et al. 2021)

É permitido que haja duplicação de funções de ativação em uma mesma camada, visto que isso reduz a sensividade do sistema a inicializações aleatórias e evita mínimos locais durante a convergência (3). O número de camadas L é um parâmetro que controla a complexidade da expressão permitindo uma variedade maior de combinações e composições e é análogo ao comprimento máximo de árvores em regressão simbólica típica. As redes EQL podem ser treinadas usando o algoritmo backpropagation, o que permite sua integração a outros modelos de redes neurais para treinamentos do tipo end-to-end. A esparsidade é obtida utilizando-se regularização L_p. Em particular, em (14), (13), (7) é utilizado uma versão suavizada da norma $L_{0.5}$:

$$\begin{cases} |w| & |w| \geq a \\ \left(-\dfrac{w^4}{8a^3} + \dfrac{3w^2}{4a} + \dfrac{3a}{8}\right) & |w| < a \end{cases}, \tag{1.16}$$

Figura 3 – Norma $L_{0.5}$ e versão suavizada com parâmetro a=0.25



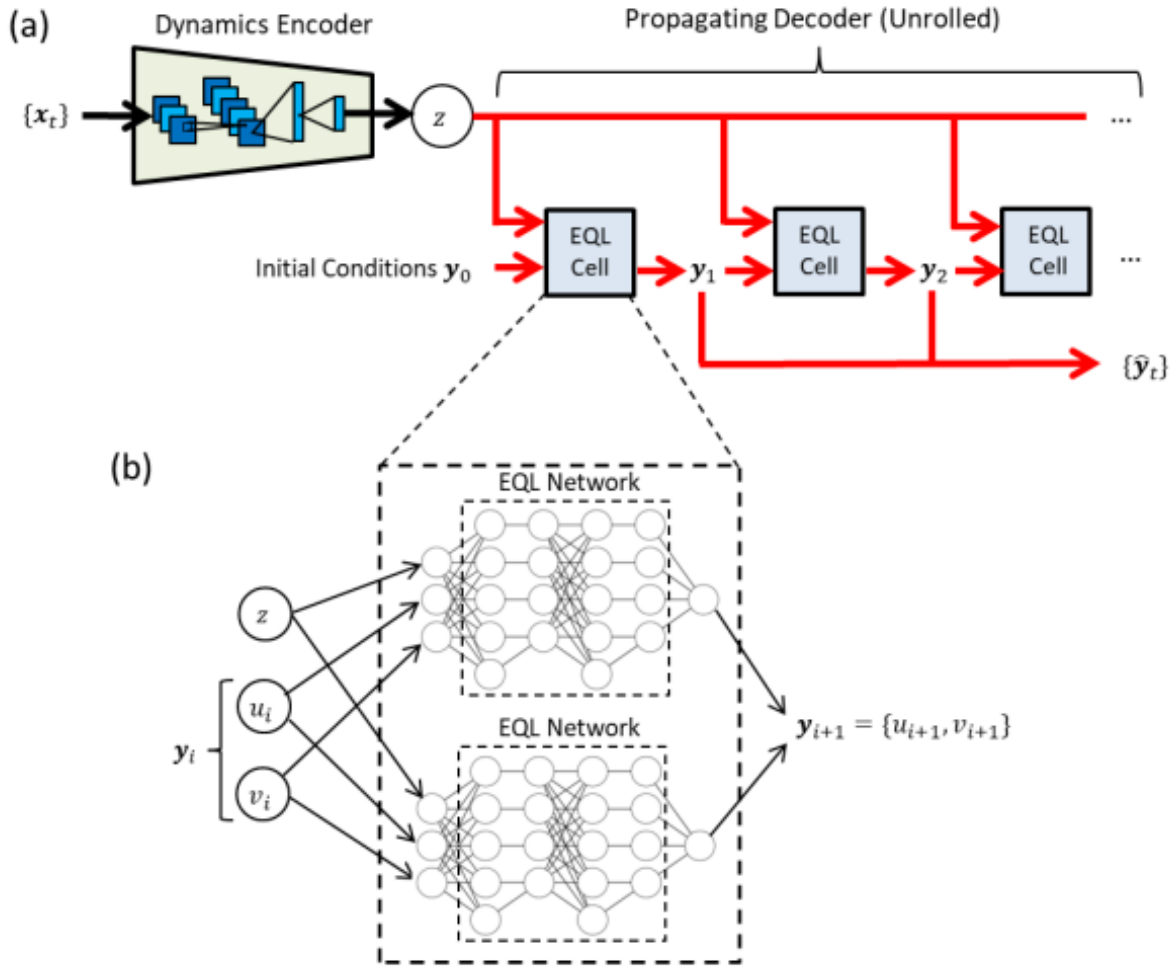
Fonte: (ABDELLAOUI, I. et al. 2021)

Em (7), essas redes são utilizadas em problemas de forecasts para dados climáticos de três cidades próximas utilizando uma base de dados do National Climatic Data Center (NCDC, EUA) com dados de 2000 a 2010 e resolução temporal horária.

1.3.1.5 AutoEncoders integrados com redes EQL

O método descrito em (9) inspirou uma arquitetura semelhante em (3) na qual o módulo do Encoder variacional é substituído por um Encoder convolucional comum e o módulo do Decoder Propagante utiliza redes EQL ao invés de convolucionais. O objetivo era, além de conseguir extrapolar a dinâmica, conseguir uma expressão analítica que descrevesse o comportamento dos dados. As duas redes EQL de cada célula da estrutura recorrente objetivam aprender os termos posição e velocidade da trajetória e têm como entrada o par (posição, velocidade) gerados anteriormente e um vetor de parâmetros z. O esquema da arquitetura proposta é mostrado em (Figura 4)

Figura 4 – Arquitetura semelhante a descrita em (9) que utiliza decoders baseados em redes EQL. Em a) temos a etapa do Encoder que utiliza camadas convolucionais e em b) O propagating Decoder com células de redes EQL.



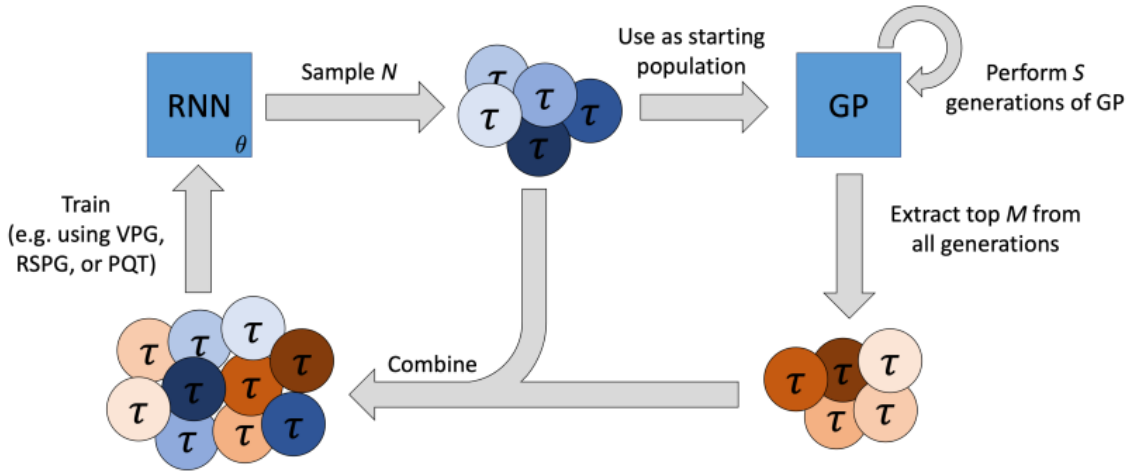
Fonte: (KIM. S et al. 2021)

1.3.1.6 Regressão simbólica utilizando RNN e aprendizado por reforço

O método híbrido proposto em (2) mistura redes neurais recorrentes, programação genética e aprendizado por reforço. Neste método, dado um conjunto de pontos, busca-se encontrar a expressão com o melhor fitting. Cada expressão matemática a ser considerada é representada por uma árvore em que os nós internos são operadores (ex: sin) e os terminais são as variáveis independentes(ex: x, y). Cada árvore é representada por um vetor transversal com seus nós, $\tau = [\tau_1, \ldots, \tau_{|\tau|}]$. A cada ciclo sucessivo de treinamento, uma amostra inicial é tomada de uma distribuição parametrizável de expressões matemáticas $p(\tau|\theta)$, representada pelo modelo da rede neural autoregressiva, em que a verossimilhança o i-ésimo token é tal que $p(\tau_i|\tau_{j\neq i}, \theta) = p(\tau_i|\tau_{j<i}, \theta)$. A RNN, composta por uma única camada LSTM com 32 hidden nodes, funciona como um gerador de sequências. O treinamento da rede é feito utilizando-se aprendizado por reforço buscando otimizar a

amostragem da rede neural, ou seja, gerando um batch $\mathcal{T}$ contendo N expressões, calulando uma função recompensa $\mathcal{R}(\tau)$ e utilizando o método do gradiente em uma função perda.

Figura 5 – Esquema do método híbrido que utiliza redes neurais recorrentes com algoritmos genéticos.



Fonte: (MUNDHENK, T. et al. 2021)

Em (2) são utilizados três métodos diferentes para treinar a RNN:

- **Vanilla policy gradient(VPG)** - O treinamento é feito com o batch $\mathcal{T}$, com uma função perda $\mathcal{L}(\theta) = \frac{1}{|\mathcal{T}|}\sum_{\tau\in\mathcal{T}}(\mathcal{R}(\tau) - b)\nabla_\theta log(p(\tau|\theta))$. Em que b é uma variável de controle.
- **Risk-seeking policy gradient(RSPG)** - Foi desenvolvido como uma alternativa ao VPG para otimizar caso a caso ao invés de uma recompensa média. A função perda é dada por $\mathcal{L}(\theta) = \frac{1}{\epsilon|\mathcal{T}|}\sum_{\tau\in\mathcal{T}}(\mathcal{R}(\tau) - \hat{\mathcal{R}}_\epsilon)\nabla_\theta log(p(\tau|\theta))\mathbf{1}_{\mathcal{R}(\tau)>\hat{\mathcal{R}}_\epsilon}$. Em que ϵ é um hiperparâmetro de controle e $\hat{\mathcal{R}}_\epsilon$ o (1 - ϵ) quantil das recompensas de $\mathcal{T}$
- **Priority queue training(PQT)** - Abordagem que não utiliza aprendizado por reforço. Amostras de cada batch gerado são adicionadas a uma fila de prioridade de máxima recompensa (MRPQ) e o treinamento é feito utilizando as amostras do MRPQ de forma supervisionada com função objetivo $\mathcal{L}(\theta) = \frac{1}{k}\sum_{\tau\in MRPQ}\nabla_\theta log(p(\tau|\theta))$. Em que k é o tamanho da MRPQ

Após a geração de uma população inicial de árvores de expressões $\mathcal{T}_\mathcal{G}\mathcal{P}$ pela RNN, a próxima etapa do método consiste em utilizar algoritmos genéticos para fazer uma série de "operações evolutivas"que alteram o estado dessa população. As operações utilizadas aqui são:

- **Mutações** - Alterações randomicas na estrutura de um dos indivíduos, ex: substituições de uma sub-árvore por outra sub-árvore gerada aleatóriamente.
- **Crossover** - Troca de conteúdo entre indivíduos, ex: Inserção de uma sub-árvore de um indivíduo da população em uma sub-árvore de outro indivíduo.
- **Seleção** - Decisão sobre quais indivíduos gerados pelos processos anteriores devem permanecer na próxima população. Um tipo comum de seleção é a amostragem aleatória de k indivíduos de uma população e escolha dos indivíduos com melhor fitting.

Além disso, algumas mudanças foram introduzidas na parte genética do método As mutações, que usualmente são uniformes, foram substituídas por três tipos: Inserção, Substituição de um nó e Encurtamento de ramos. Adição de restrições, como por exemplo em funções trigonométricas compostas. Se uma alteração viola alguma dessas restrições, a expressão gerada passa a ser uma cópia da anterior.
A população inicial da fase GP sempre é gerada pela RNN e a população gerada ao fim dessa fase é utilizada para treinar a rede neural. O algoritmo geral do funcionamento do método é mostrado abaixo.

Algorithm 1 Algoritmo do método. Fonte: (MUNDHENK, T. et al. 2021)

Input Tamanho da população N; Número de gerações na fase GP por alteração da RNN S; restrições da fase GP Ω; Probabilidades de Crossover e Mutação P_c, P_m; Número de candidatos para o operador de seleção k; Tamanho da população selecionada M($\leq$ N)
Output Melhor amostra τ^*
Inicializar função recompensa $\mathcal{R} : \tau \Longrightarrow \mathbb{R}$
Inicializar distribuição de expressões da RNN p(.$|\ \theta, \Omega$)
Inicializar operador GP(P_c,P_m,k,R) := Γ: $\mathcal{T} \longrightarrow \mathcal{T}$
$\tau^* \leftarrow 0$
while total de amostras menor que o número de estimativas **do**
  $\mathcal{T}_{RNN} \leftarrow \{\tau^{(i)} \sim \text{p}(.|\ \theta, \Omega)\}_{i=1}^{N}$ ▷ Gerando amostras da distribuição
  $\mathcal{T}_{GP}^{(0)} \longleftarrow \mathcal{T}_{RNN}$ ▷ Iniciando a fase GP com população inicial vinda da RNN
  for $s = 1, \ldots, S$ **do**
    $\mathcal{T}_{GP}^{(s)} \longleftarrow \Gamma(\mathcal{T}_{GP}^{(s-1)})$ ▷ Aplicando o operador GP
  end for
  $\mathcal{T}_{TRAIN} \longleftarrow$ Top-M($\mathcal{T}_{GP}^{(0)} \bigcup \mathcal{T}_{GP}^{(1)} \bigcup \ldots \bigcup \mathcal{T}_{GP}^{(S)}$) ▷ Selecionando os top M indivíduos
  $\mathcal{T}_{TRAIN} \longleftarrow \mathcal{T}_{TRAIN} \bigcup \mathcal{T}_{RNN}$ ▷ Somando as amostras iniciais com os selecionados da GP
  $\mathcal{R} \longleftarrow \{R(\tau)$ for $\tau \in \mathcal{T}\}$ ▷ Calculando recompensas
  $\theta \longleftarrow \theta + \nabla_\theta \mathcal{L}(\theta)$ ▷ Aplicando método do gradiente
  if max($\mathcal{R}$)>R(τ^*) **then** $\tau^* \leftarrow$ argmax($\mathcal{R}$) ▷ Atualizando estimativa
  end if
end while
return τ^*

O método conseguiu superar o estado-da-arte na recuperação das equações para

vários problemas diferentes e o melhor método de treinamento obtido foi o PQT.

1.3.2 Metodologia experimental

Nesta seção estão os testes realizados para avaliar algumas soluções descritas na seção anterior. Com a finalidade de um estudo comparativo entre os métodos apresentados em (2), (7), (1) foram conduzidos experimentos para verificar sua eficácia em problemas de regressão, utilizando a própria implementação disponível de cada artigo.

1.3.2.1 Redes EQL

Nos testes envolvendo redes EQL, foi utilizado a arquitetura proposta em (7), com duas camadas escondidas EQL e uma camada Densa com um único neurônio e ativação linear. Em todos os testes, os pesos de cada camada foram inicializados tomando valores de uma distribuição normal com variâncias (0.1,0.5,1.0) respectivamente, visando diminuir a sensibilidade à inicializações aleatórias como mencionado em (3). As funções de ativação eram as mesmas para todas as camadas, com possibilidade de repetição dentro de uma mesma camada. A função perda utilizada foi o erro quadrático médio com regularização $L_{0.5}$*. O otimizador RMSProp foi utilizado em todos os testes.

1.3.2.2 Redes Recorrentes e programação genética

Nos testes utilizando a solução proposta em (2) uma rede LSTM com uma camada e 32 unidades foi utilizada. Os estados de cada unidade foram inicializados como zeros. O método utilizado de treinamento foi o PQT. Os outputs das células armazenados no estado do loop da RNN foram utilizados como logits em uma camada Softmax para calcular a distribuição $p(\tau|\theta)$ de probabilidades associada com os batches de expressões. O treinamento foi efetuado seguindo o método PQT com otimização por gradiente com o otimizador Adam. A recompensa utilizada para formar MPRQ foi obtida a partir da erro quadrático médio normalizado (NMRSE).

$$R(\tau) = \frac{1}{1 + NMRSE(\tau)},$$

1.3.2.3 Validação e comparação

Na primeira etapa a arquitetura baseada em redes EQL foi testada para o problema do oscilador harmônico:

$$\frac{d^2}{dt^2}x = -\omega_0^2 x, \tag{1.17}$$

Dados foram simulados para esse problema por integração numérica da equação de movimento com 1000 passos e Δt=0.01. O dicionário de ativações utilizado foi $\Theta = [1, x, x^2, Sin, Sigmoid]$, além de um operador binário produto. Os parâmetros associados a esparsidade foram variados para esse problema com ordens próximas aos parâmetros utilizados

em (7) em triplicata, e a frequência de cada operador na expressão final foi calculada junto com o percentual de esparsidade de cada camada e o comprimento da expressão final. O treinamento foi feito em duas fases:

- Fase I: 100 épocas com taxa de aprendizado 1e-4 com regularização.
- Fase II: 100 épocas com taxa de aprendizado 1e-5 sem regularização.

O batch size utilizado foi 200.
O trabalho (2) aplicou seu método em um conjunto de problemas de regressão propostas em (15). Essa análise foi repetida aqui considerando os seis primeiros problemas para comparar os métodos (7) (2). Esses experimentos também foram conduzidos em triplicata devido a sensibilidade das redes EQL a inicialização dos pesos.100 batches de 20 observações foram geradas para cada problema. Para as redes EQL, a frequência dos termos almejados no conjunto total de termos recuperados foi utilizada para o cálculo da taxa de recuperação de cada problema. Para a arquitetura em (2) as expressões recuperadas juntamente com as recompensas obtidas foram utilizadas e a taxa de aprendizado utilizada foi de 0.0025.

1.3.2.4 Resistência a ruído

Os métodos com melhor propriedades de esparsidade da etapa anterior da metodologia foram utilizados nesta etapa para verificar se as equações obtidas sofriam grandes mudanças com adição de ruído. O problema considerado foi um sistema de Lorenz para convecção:

$$\begin{aligned}\frac{dx}{dt} &= \sigma(y-x),\\ \frac{dy}{dt} &= x(\rho-z)-y,\\ \frac{dz}{dt} &= xy-\beta z,\end{aligned}$$

Com parâmetros $\sigma = 10$, $\rho = 28$, $\beta = \frac{8}{3}$.
Foram simulados dados a partir desse modelo partindo de uma condição inicial (1,-1,20) para t(tempo) entre 0 e 100 s com um $\Delta t = 0.01$. As derivadas foram obtidas a partir dos dados simulados utilizando o método de Diferença Finita. Ruídos de uma distribuição normal com diferentes variâncias foram adicionados aos dados e os métodos de regressão simbólica foram testados para encontrar as equações do modelo.
Para o método de regressão esparsa((1)) foi utilizado um otimizador Base, padrão do método. O dicionário de funções utilizado possuía termos identidade e termos quadráticos nas variáveis disponíveis. Para o método utilizando Deep Learning ((2)) foi utilizado um tamanho de população $N = 2000000$. Para a fase GP foi utilizado um número de gerações $S = 50$ com probabilidade de crossover P_c=0.5 e probabilidade de mutação P_m=0.7 e

10 candidatos para o operador de seleção. O método PQT de treinamento foi adotado com um tamanho da fila de máxima de prioridade(MRPQ) $m = 10$ e um batch size de 5. O conjunto de funções candidatas utilizado contém operadores de adição, subtração e multiplicação.

1.4 Resultados e Discussão

Para o problema do oscilador harmônico utilizando as redes EQL, os parâmetros de esparsidade, próximos aos utilizados em (7) foram variados. O erro quadrático médio e o percentual de esparsidade obtidos após a segunda fase para cada um dos parâmetros é sumarizado na tabela 1.

Tabela 1 – Percentual de esparsidade em cada uma das três camadas da rede EQL utilizada, número de termos na expressão final L e erro quadrático médio(MSE) para o problema do oscilador harmônico.

Parâmetros			**Esparsidade(%)**				**MSE**
λ	a	$Threshold$	$S1$	$S2$	$S3$	L	
5	5e-3	7e-3	94.44	88.88	37.5	144	1.93e-05
5	7e-3	8e-4	88.88	45.13	18.75	229	1.77e-6
3	7e-4	8e-3	75.00	34.02	31.25	269	1.53e-5
3	7e-4	8e-4	66.66	38.54	18.75	254	2.3e-5

Fonte: Elaborado pelo autor

Todos os parâmetros testados renderam expressões finais com centenas de termos. A maior esparsidade obtida foi para os parâmetros (λ,a,$Threshold$)=(5,5$e-$3,7$e-$3) com 144 termos na expressão final. O valor final do erro quadrático médio da validação mostra que para todos os testes a rede foi capaz de aprender o necessário para aproximar a função mas não sem incluir um grande número de termos. A frequência de cada termo na expressão final é listada na tabela 2

Tabela 2 – Valores dos parâmetros de esparsidade testados na primeira coluna. Na segunda, a frequência dos termos encontrados na expressão final.

Parâmetros			**Frequência de termos(%)**					
λ	a	$Threshold$	$Sigmoid$	Sin	x	1	x^2	Product
5	5e-3	7e-3	20.16	11.53	30.31	11.27	18.57	8.16
5	7e-3	8e-4	13.10	13.64	26.86	13.3	22.15	10.95
3	7e-4	8e-3	12.85	13.66	24.93	11.91	24.07	12.55
3	7e-4	8e-4	13.50	14.02	28.31	13.62	21.10	9.45

Fonte: Elaborado pelo autor

Observa-se que os termos de maior frequência na expressão final são termos identidade, embora termos quadráticos também tenham aparecido. Os parâmetros com maior percentual de esparsidade pela tabela 1 apresentam a maior frequência de termos identidade e menor frequência do operador produto. Isso mostra que embora a escolha dos parâmetros garanta soluções menores e aumente a frequência dos termos da expressão analítica real do modelo, a regularização utilizada ainda não é suficiente para levar a rede a convergir para uma solução com poucos termos.

1.4.1 Conjunto de problemas de Nguyen

Quando testadas no conjunto de problemas de regressão contendo expressões polinomiais, trigonométricas, as redes EQL novamente não atingiram a expressão final almejada, mas a razão entre os termos dessas expressões finais (Polinomiais ou Trigonométricos) e o número total de termos foi grande.(tabela 3)

Tabela 3 – Problemas de regressão propostos em (15) com os parâmetros de regularização λ=5, a=5e-3,Threshold=7e-3 e Taxa de recuperação (Razão entre o número de termos entre os termos da solução e número total de termos)

Problema de regressão	**Expressão**	**Taxa de recuperação**		
		EQL1	**EQL2**	**EQL3**
Nguyen-1	$x^3 + x^2 + x$	0.69	0.63	0.67
Nguyen-2	$x^4 + x^3 + x^2 + x$	0.77	0.81	0.80
Nguyen-3	$x^5 + x^4 + x^3 + x^2 + x$	0.72	0.72	0.69
Nguyen-4	$x^6 + x^5 + x^4 + x^3 + x^2 + x$	0.70	0.70	0.71
Nguyen-5	$sin(x^2)cos(x) - 1$	0.60	0.59	0.59
Nguyen-6	$sin(x) + sin(x + x^2)$	0.79	0.79	0.80

Fonte: Elaborado pelo autor

O teste dos mesmos problemas utilizando a solução descrita em (2) encontrou todas as expressões testadas (tabela 4).

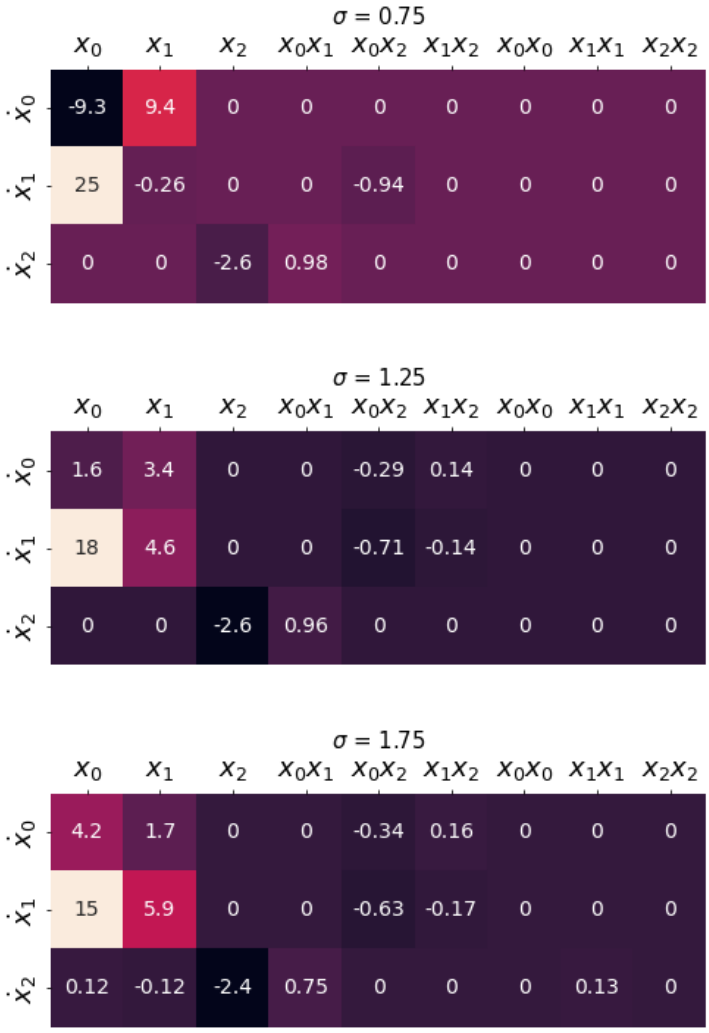
Tabela 4 – Expressões recuperadas utilizando a solução baseada em redes neurais recorrentes combinado com algoritmos genéticos. Na terceira coluna temos a recompensa calculada para o fitting da expressão final com a real.

Problema de regressão	**Expressão recuperada**	**Recompensa**
Nguyen-1	$x(x^2 + x + 1)$	1.0
Nguyen-2	$x(x^2(x + 1) + x + 1)$	1.0
Nguyen-3	$x(x^2 + 1)(x^2 + x) + x$	1.0
Nguyen-4	$x(x^3 + x(x^2(x^2 + x) + x) + x) + x$	0.99
Nguyen-5	$sin(x^2)cos(x) - 1$	1.0
Nguyen-6	$sin(x) + sin(x + x^2)$	1.0

O tratamento das expressões como árvores em (2) é mais eficaz em controlar a esparsidade do que o regularizador utilizado em (7) pois resulta em expressões com número de termos semelhantes aos da expressão real. Além disso, o método (2) encontrou todas as expressões testadas.

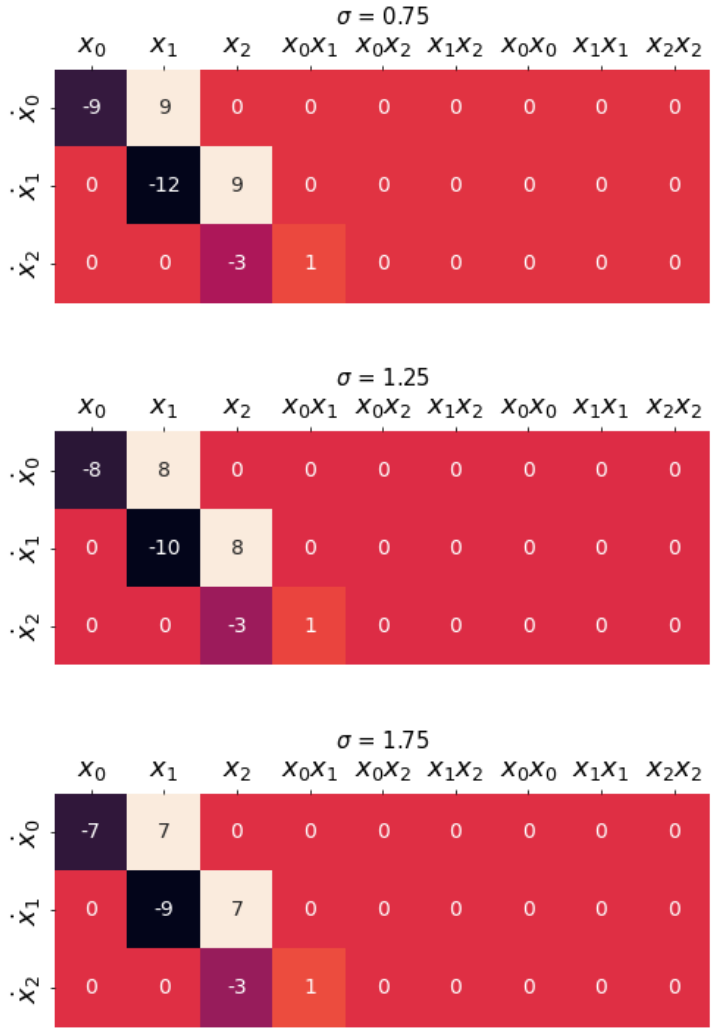
1.4.2 Resistência ao ruído

Figura 6 – Para uma das variâncias do ruído testadas no método SINDy, vemos as colunas com termos candidatos de funções e nas linhas as três derivadas. O valor dos coeficientes de cada termo é anotado em cada célula



Fonte: Elaborado pelo autor

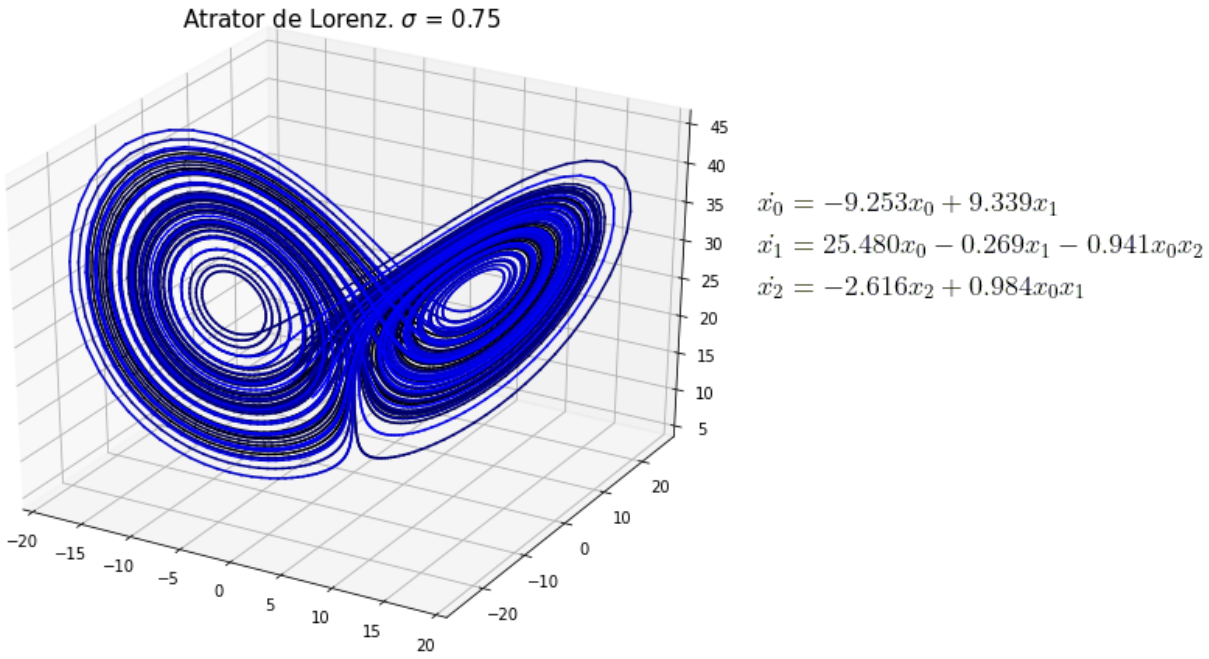
Figura 7 – Para uma das variâncias do ruído testadas no método descrito em (2), vemos as colunas com termos candidatos de funções e nas linhas as três derivadas. O valor dos coeficientes de cada termo é anotado em cada célula



Fonte: Elaborado pelo autor

A figura 6 mostra os valores dos coeficientes encontrados para cada candidato no dicionário de funções utilizado pelo método SINDy e a figura 7 para o método de Deep Learning. Na figura 8 é possível observar a correspondência do atrator gerado pelo modelo e a tabela representada na figura 6.

Figura 8 – Atrator de Lorenz para o modelo encontrado pelo método Sindy(Figura 6) com ruído de variância ($\sigma = 0.75$) adicionado aos dados



Fonte: Elaborado pelo autor

Vemos que a adição da perturbação diminui a esparsidade no método SINDy, enquanto no método de Deep Learning apenas os coeficientes dos termos originais são alterados. Foi observado que ao limitar o número de funções candidatas no método de Deep Learning a inicialização tendia a ser mais consistente, no sentido que o processo chegava a um mínimo mais rapidamente. Com ruído fraco, o método SINDy consegue identificar os termos corretos em todas as derivadas, enquanto o método de Deep Learning falha na derivada da variável x_1.

1.4.3 Discussão

Na elaboração dos experimentos realizados, foram considerados trabalhos que tinham suporte na literatura como funcionais. Uma outra arquitetura (3) utilizando redes EQL foi testada, porém, como os resultados não foram satisfatórios nas primeiras tentativas, foi excluída deste trabalho. Embora o desempenho das redes EQL tenha atingido resultados satisfatórios em trabalhos precedentes (7), (3), a conclusão dos testes é que estas redes se mostram incapazes de buscar expressões garantindo esparsidade e os resultados só puderam ser trazidos na forma de frequência de termos na expressão buscada. O mal desempenho das redes EQL em problemas de regressão simbólica já foi constatado em (11), no qual esse método foi excluído das baselines de comparação por ser incapaz de encontrar as soluções exatas para problemas de regressão simples. A arquitetura das

camadas totalmente conectadas perde em esparsidade quando comparada a árvores binárias de expressões.
O método SINDy tem sido amplamente utilizado na descoberta de equações físicas, enquanto o método em (2) reproduz o estado da arte da área de regressão simbólica((11)).
A estabilidade do método diante de ruídos é uma propriedade desejável. O método em (2) utiliza a Frente de Pareto para determinar a equivalência simbólica no espaço de recompensas atingidas. Vemos que o resultado no sistema de Lorenz é que este método supera o SINDy com otimizador base. Um trabalho em (12) já incorporou previamente resultados de teoria de estabilidade para sistemas dinâmicos na forma de um novo otimizador para o SINDy. No entanto, sua aplicação se reduzia a um tipo específico de sistemas com termos quadráticos nas derivadas e incluía mais parâmetros.
Ainda para o sistema de Lorenz, foi observado que o método em (2) falhou em recuperar a expressão para a segunda derivada e chegou a um mínimo local. É possível que a restrição no espaço de funções candidatas combinado com método de treinamento escolhido tenham previnido o overfitting, enquanto o SINDy recuperou todas as expressões no caso sem ruído. Futuramente, outros testes com os outros métodos disponíveis de aprendizado por reforço aqui listados deveriam ser considerados para a comparação desses dois métodos.

1.5 Conclusão

Contornar os obstáculos envolvidos em encontrar expressões analíticas para dados em física exige a capacidade de delimitar um espaço de soluções possíveis, mantendo esparsidade. Nesse sentido, os testes feitos aqui levaram a conclusão de que arquiteturas de redes neurais baseadas em camadas EQL não são suficientes para recuperação de expressões a partir dos dados e há um desacordo na literatura sobre os resultados desse método. Os outros dois métodos testados foram suficientes para a recuperação das equações.
Sobre o teste de resistência a ruído, concluiu-se que o método em (2) era o método mais resistente em sentido de manter os mesmos termos, embora não tenha encontrado a expressão para uma das derivadas, possivelmente devido ao tamanho do dicionário de operadores utilizado. Futuramente, para os métodos de Deep Learning, o uso de estruturas de árvores para representar expressões pode se mostrar útil em garantir esparsidade. Métodos envolvendo aprendizado competitivo e dropout de neurônios durante o treinamento podem ser estratégias para melhorar o método em (3).
No futuro, soluções que incorporem conhecimentos de problemas físicos serão necessárias para reduzir a complexidade da tarefa. com o avanço das técnicas descritas aqui espera-se que seja possível a redescoberta de princípios físicos assim como a automatização de descobertas da dinâmica de sistemas de interesse.

REFERÊNCIAS

1 BRUNTON, S. L.; PROCTOR, J. L.; KUTZ, J. N. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. **Proceedings of the National Academy of Sciences**, Proceedings of the National Academy of Sciences, v. 113, n. 15, p. 3932–3937, mar 2016. Available at: https://doi.org/10.1073%2Fpnas.1517384113.

2 MUNDHENK, T. N. *et al.* **Symbolic Regression via Neural-Guided Genetic Programming Population Seeding**. arXiv, 2021. Available at: https://arxiv.org/abs/2111.00053.

3 KIM, S. *et al.* Integration of neural network-based symbolic regression in deep learning for scientific discovery. **IEEE Transactions on Neural Networks and Learning Systems**, Institute of Electrical and Electronics Engineers (IEEE), v. 32, n. 9, p. 4166–4177, sep 2021. Available at: https://doi.org/10.1109%2Ftnnls.2020.3017010.

4 SILVA, B. M. de *et al.* Discovery of physics from data: Universal laws and discrepancies. **Frontiers in Artificial Intelligence**, Frontiers Media SA, v. 3, apr 2020. Available at: https://doi.org/10.3389%2Ffrai.2020.00025.

5 BRUNTON, S. L.; PROCTOR, J. L.; KUTZ, J. N. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. **Proceedings of the National Academy of Sciences**, Proceedings of the National Academy of Sciences, v. 113, n. 15, p. 3932–3937, mar 2016. Available at: https://doi.org/10.1073%2Fpnas.1517384113.

6 LEMOS, P. *et al.* **Rediscovering orbital mechanics with machine learning**. arXiv, 2022. Available at: https://arxiv.org/abs/2202.02306.

7 ABDELLAOUI, I. A.; MEHRKANOON, S. **Symbolic regression for scientific discovery: an application to wind speed forecasting**. arXiv, 2021. Available at: https://arxiv.org/abs/2102.10570.

8 SAHOO, S. S.; LAMPERT, C. H.; MARTIUS, G. **Learning Equations for Extrapolation and Control**. arXiv, 2018. Available at: https://arxiv.org/abs/1806.07259.

9 LU, P. Y.; KIM, S.; ć, M. Soljači. Extracting interpretable physical parameters from spatiotemporal systems using unsupervised learning. **Phys. Rev. X**, American Physical Society, v. 10, p. 031056, Sep 2020. Available at: https://link.aps.org/doi/10.1103/PhysRevX.10.031056.

10 UDRESCU, S.-M.; TEGMARK, M. Ai feynman: a physics-inspired method for symbolic regression. arXiv, 2019. Available at: https://arxiv.org/abs/1905.11481.

11 PETERSEN, B. K. *et al.* Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. arXiv, 2019. Available at: https://arxiv.org/abs/1912.04871.

12 KAPTANOGLU, A. A. *et al.* Promoting global stability in data-driven models of quadratic nonlinear dynamics. **Physical Review Fluids**, American Physical Society (APS), v. 6, n. 9, sep 2021. Available at: https://doi.org/10.1103%2Fphysrevfluids.6.094401.

13 LOUIZOS, C.; WELLING, M.; KINGMA, D. P. **Learning Sparse Neural Networks through L_0 Regularization**. 2018.

14 WU, W. *et al.* Batch gradient method with smoothing l1/2 regularization for training of feedforward neural networks. **Neural Networks**, v. 50, p. 72–78, 2014. ISSN 0893-6080. Available at: https://www.sciencedirect.com/science/article/pii/S0893608013002700.

15 UY, N. Q. *et al.* Semantically-based crossover in genetic programming: Application to real-valued symbolic regression. **Genetic Programming and Evolvable Machines**, Kluwer Academic Publishers, USA, v. 12, n. 2, p. 91–119, jun 2011. ISSN 1389-2576. Available at: https://doi.org/10.1007/s10710-010-9121-2.