UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

PEDRO JORGE OLIVEIRA CÂMARA

Neural network hyperparameter optimization using evolutionary algorithms

RIO DE JANEIRO
2025

PEDRO JORGE OLIVEIRA CÂMARA

Neural network hyperparameter optimization using evolutionary algorithms

Trabalho de conclusão de curso de graduação apresentado ao Instituto de Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientadores: Profa. Carolina Marcelino e Prof. Paulo Mann

RIO DE JANEIRO
2025

CIP - Catalogação na Publicação

C172n

Câmara, Pedro Jorge Oliveira
Neural network hyperparameter optimization using evolutionary algorithms / Pedro Jorge Oliveira Câmara. -- Rio de Janeiro, 2025.
51 f.

Orientadora: Carolina Gil Marcelino.
Coorientador: Paulo Roberto Mann Marques Júnior.
Trabalho de conclusão de curso (graduação) - Universidade Federal do Rio de Janeiro, Instituto de Computação, Bacharel em Ciência da Computação, 2025.

1. Inteligência artificial. 2. Redes neurais. 3. Otimização. 4. Algoritmos evolutivos. 5. Hiperparâmetros. I. Marcelino, Carolina Gil, orient. II. Marques Júnior, Paulo Roberto Mann, coorient. III. Título.

Elaborado pelo Sistema de Geração Automática da UFRJ com os dados fornecidos pelo(a) autor(a), sob a responsabilidade de Miguel Romeu Amorim Neto - CRB-7/6283.

PEDRO JORGE OLIVEIRA CÂMARA

Neural network hyperparameter optimization using evolutionary algorithms

Trabalho de conclusão de curso de graduação apresentado ao Instituto de Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 29 de agosto de 2025

BANCA EXAMINADORA:

Documento assinado digitalmente
gov.br
CAROLINA GIL MARCELINO
Data: 02/09/2025 15:43:56-0300
Verifique em https://validar.iti.gov.br

Carolina Gil Marcelino
D.Sc. (IC/UFRJ)

Documento assinado digitalmente
gov.br
PAULO ROBERTO MANN MARQUES JUNIOR
Data: 02/09/2025 15:50:30-0300
Verifique em https://validar.iti.gov.br

Paulo Roberto Mann Marques Júnior
D.Sc. (IC/UFRJ)

Documento assinado digitalmente
gov.br
JOAO LUIZ LAGOAS DE ALMEIDA BERTOLINO
Data: 02/09/2025 16:44:18-0300
Verifique em https://validar.iti.gov.br

João Luiz Lagôas de Almeida Bertolino
M.Sc. (Colégio Pedro II)

Documento assinado digitalmente
gov.br
SILAS PEREIRA LIMA FILHO
Data: 02/09/2025 16:49:46-0300
Verifique em https://validar.iti.gov.br

Silas Pereira Lima Filho
D.Sc.

Documento assinado digitalmente
gov.br
CARLA AMOR DIVINO MOREIRA DELGADO
Data: 02/09/2025 18:42:47-0300
Verifique em https://validar.iti.gov.br

Carla Amor Divino Moreira Delgado
D.Sc. (IC/UFRJ)

Dedicatória: à minha avó, que me possibilitou chegar até aqui.

AGRADECIMENTOS

O primeiro e mais importante agradecimento é direcionado à minha avó, Teresa, que me criou e tornou possível a minha formação no Colégio Pedro II e na Universidade Federal do Rio de Janeiro.

Agradeço ao professor João Lagôas, do Colégio Pedro II, que, desde o ensino médio, é minha inspiração acadêmica e profissional. Sou grato pela inspiração a entrar no Bacharelado em Ciência da Computação da UFRJ e pelo seu acompanhamento dessa trajetória, que abrange desde a notícia da aprovação no vestibular até os momentos finais da graduação e, certamente, nos próximos passos envolvendo o mestrado e a vida profissional.

Agradeço à professora Carolina e ao professor Paulo Mann, do Instituto de Computação, por aceitarem orientar esse trabalho, pela ajuda e pelas sugestões ao longo do desenvolvimento, e pela atenciosidade, empatia e apoio quanto às minhas questões burocráticas e pessoais.

Agradeço ao professor Rafael Dias Mesquita, do Laboratório de Bioinformática, do Instituto de Química, por disponibilizar a infraestrutura computacional do laboratório para ser usada no desenvolvimento deste trabalho e por me aceitar como aluno de iniciação científica.

Agradeço às agências de fomento CAPES, CNPq e FAPERJ pelo investimento aos projetos realizados pelo grupo de pesquisa, o que inclui este trabalho, e à UFRJ pela infraestrutura disponibilizada.

Por fim, um agradecimento a todos os amigos feitos na UFRJ, que, direta ou indiretamente, contribuíram para a minha formação. Em particular, agradeço aos amigos Matheus do Ó, Gabriel Stamato, Priscila Esteves, Christian Oliveira, João Pedro Souza e João Marcelo, com os quais compartilhei momentos importantes da graduação.

Muito obrigado! Meus mais sinceros agradecimentos a todos.

RESUMO

Questões envolvendo a geração e o consumo de energia são de interesse internacional. A energia elétrica, em particular, é um dos temas dos Objetivos de Desenvolvimento Sustentável da Organização das Nações Unidas, buscando, até 2030, estabelecer acesso confiável, sustentável e economicamente viável para todos. Atualmente, prever o consumo total ou parcial da eletricidade em um país pode ser de interesse para o planejamento estratégico. Há uma perspectiva de transição do atual sistema de fornecimento para malhas elétricas inteligentes, o que possibilitará práticas da distribuição mais sustentáveis, mas a flutuação do preço da malha depende da estabilidade do sistema físico em que ela se encontra. Uma possível abordagem para esses problemas é utilizar modelos de inteligência artificial para determinar o consumo de energia elétrica no atual sistema, como um problema de regressão, e a estabilidade econômica em uma malha inteligente, como um problema de classificação. As redes neurais artificiais são modelos consolidados na literatura, que apresentam desempenho satisfatório para uma ampla variedade de aplicações. No entanto, uma das etapas do seu desenvolvimento é a determinação, a priori, de hiperparâmetros, valores que determinam as características de sua arquitetura e seu funcionamento. Este trabalho de conclusão de curso se propõe a utilizar algoritmos evolutivos para encontrar os hiperparâmetros que constroem os melhores modelos possíveis para essas duas tarefas e compara o desempenho dessa abordagem com o Optuna, framework voltado para esse objetivo. Os resultados demonstram que, para a tarefa de regressão, os algoritmos evolutivos encontram valores robustos frente ao Optuna, enquanto que, para a classificação, ambas as formas são estatisticamente equivalentes.

Palavras-chave: inteligência artificial; redes neurais; otimização; algoritmos evolutivos; hiperparâmetros

ABSTRACT

Issues involving energy generation and consumption are of international interest. Electrical energy, in particular, is one of the topics of the United Nations Sustainable Development Goals, aiming, by 2030, to establish reliable, sustainable, and economically viable access for all. Currently, predicting the total or partial consumption of electricity in a country can be of interest for strategic planning. There is a perspective of transitioning from the current supply system to smart electrical grids, which will enable more sustainable distribution practices; however, the fluctuation in the grid's price depends on the stability of the physical system in which it operates. One possible approach to these problems is to use artificial intelligence models to determine electricity consumption in the current system, as a regression problem, and economic stability in a smart grid, as a classification problem. Artificial neural networks are well-established models in the literature, delivering satisfactory performance for a wide range of applications. However, one of the steps in their development is the prior determination of hyperparameters, values that define the characteristics of their architecture and operation. This final project proposes to use evolutionary algorithms to find the hyperparameters that build the best possible models for these two tasks and compares the performance of this approach with Optuna, a framework designed for this purpose. The results show that, for the regression task, evolutionary algorithms achieve robust values compared to Optuna, whereas for classification, both approaches are statistically equivalent.

Keywords: artificial intelligence; neural networks; optimization; evolutionary algorithms; hyperparameters.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

IA	Inteligência Artificial
MLP	Multilayer Perceptron - Perceptron Multicamadas
EA	Evolutionary Algorithm - Algoritmo Evolutivo
GA	Genetic Algorithm - Algoritmo Genético
DE	Differential Evolution - Evolução Diferencial
PSO	Particle Swarm Optimization - Otimização por Enxame de Partículas
ONU	Organização das Nações Unidas
RMSE	Root Mean Square Error - Raiz Quadrada do Erro Médio
ACC	Accuracy - Acurácia
kWh	Quilowatt-hora
EIA	U.S. Energy Information Administration
EUA	Estados Unidos da América
TP	True Positive - Verdadeiro positivo
TN	True Negative - Verdadeiro negativo
FP	False Positive - Falso positivo
FN	False Negative - Falso Negativo

LISTA DE SÍMBOLOS

$\mathbb{R}^d$	Conjunto dos números reais em dimensão d
$\forall$	Para todo
$\subset$	Contido
$\in$	Pertence
$\sum$	Somatório
α	Alfa
β	Beta
σ	Sigma
$\lvert \cdot \rvert$	Módulo

SUMÁRIO

1 INTRODUÇÃO

Os 193 Estados membros da Organização das Nações Unidas (ONU), incluindo a República Federativa do Brasil, comprometeram-se a adotar a chamada Agenda Pós-2015, considerada uma das mais ambiciosas da história da diplomacia internacional. A partir dela, a finalidade é cumprir os Objetivos de Desenvolvimento Sustentável (ODS). Os ODS representam um plano de ação global para eliminar a pobreza extrema e a fome, oferecer educação de qualidade ao longo da vida para todos, proteger o planeta e promover sociedades pacíficas e inclusivas até 2030 (ONU, 2025). Em particular, há uma preocupação crescente, a nível internacional, envolvendo questões energéticas, conforme elaborado em (ONU-ODS-7, 2025):

- Objetivo 7: Assegurar o acesso confiável, sustentável, moderno e a preço acessível à energia para todas e todos.
- 7.1 Até 2030, assegurar o acesso universal, confiável, moderno e a preços acessíveis a serviços de energia;
- 7.2 Até 2030, aumentar substancialmente a participação de energias renováveis na matriz energética global;
- 7.3 Até 2030, dobrar a taxa global de melhoria da eficiência energética;
- 7.a Até 2030, reforçar a cooperação internacional para facilitar o acesso a pesquisa e tecnologias de energia limpa, incluindo energias renováveis, eficiência energética e tecnologias de combustíveis fósseis avançadas e mais limpas, e promover o investimento em infraestrutura de energia e em tecnologias de energia limpa;
- 7.b Até 2030, expandir a infraestrutura e modernizar a tecnologia para o fornecimento de serviços de energia modernos e sustentáveis para todos os países em desenvolvimento, particularmente nos países menos desenvolvidos, nos pequenos Estados insulares em desenvolvimento e nos países em desenvolvimento sem litoral, de acordo com seus respectivos programas de apoio.

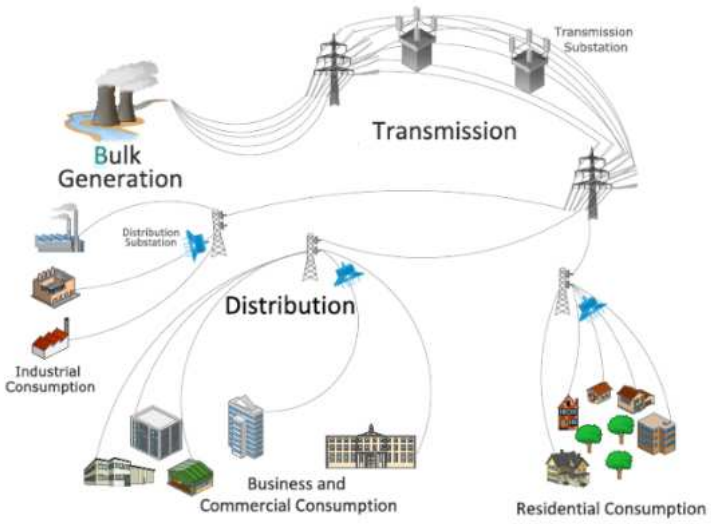
A rede elétrica, fundamental na produção, transmissão e distribuição de eletricidade, é essencial para o desenvolvimento econômico e social. Seu papel central está na alocação espacial da eletricidade. Apesar de ser considerada uma das maiores conquistas da engenharia do século XX e amplamente utilizada pelos consumidores, os métodos atuais para seu fornecimento são considerados rígidos e inflexíveis (POWELL et al., 2024).

Na média global, aproximadamente 30% da energia elétrica gerada em um país é consumida pelo setor residencial, com os valores variando entre 16-50%. Para incentivar

práticas sustentáveis, dado esse alto nível de consumo, é fundamental obter uma compreensão das características específicas desse setor. A necessidade de entender os padrões de consumo aumentou devido aos altos preços da energia, às mudanças climáticas e às variações na oferta e demanda. O consumo de energia elétrica pelo setor residencial, especificamente, é menos estudado do que os da indústria, comércio, agricultura e transporte (WANG et al., 2024).

Concomitantemente, existem tecnologias emergentes que buscam a transição para fontes energéticas renováveis e a produção eficiente de energia. O modelo atualmente consolidado de malhas elétricas possui comunicação unidirecional por natureza; ele converte apenas 1/3 da energia gerada em eletricidade, sem recuperar a energia dissipada em forma de calor. Quase 8% de sua produção é perdida ao longo das linhas de transmissão, enquanto 20% de sua capacidade de geração existe apenas para atender à demanda de pico (isto é, está em uso apenas 5% do tempo). Além disso, devido à topologia hierárquica de seus ativos, a rede elétrica existente sofre com falhas em efeito dominó (FARHANGI, 2010).

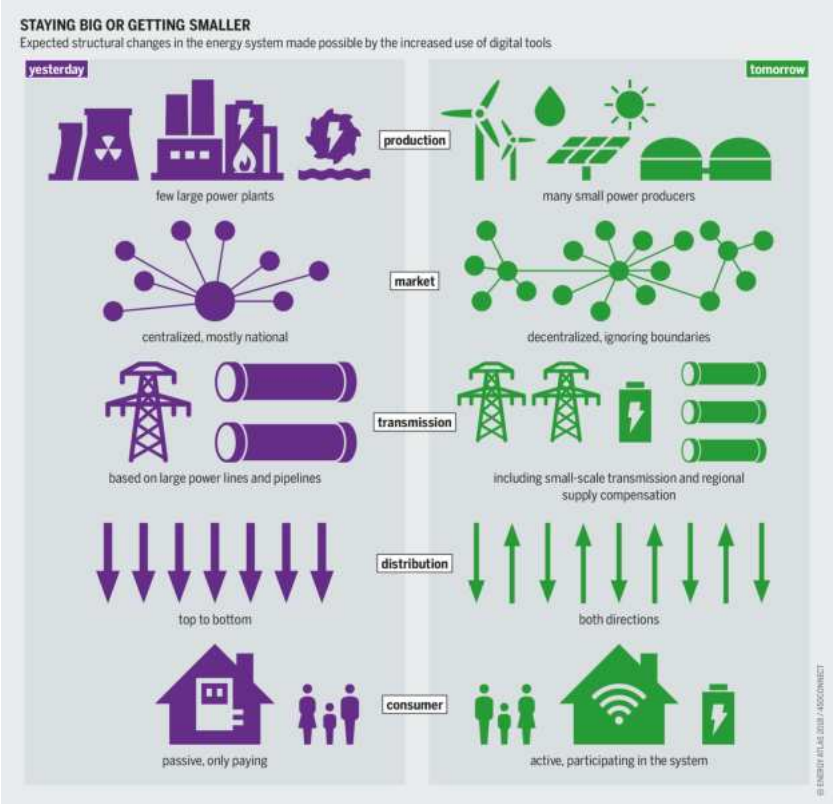
Figura 1 – Modelo da atual rede elétrica



Fonte: (TURNER; ULUDAG, 2015)

As redes elétricas inteligentes (Smart Grids) são consideradas a próxima geração do modelo de malhas elétricas, que se propõem a corrigir os problemas associados ao modelo atual, que é centralizado e unidimensional. A ideia é possibilitar uma comunicação bidimensional, utilizando uma topologia de rede distribuída com capacidade adaptativa. A transição para fontes de energia renováveis é facilitada devido à flexibilidade da Smart Grid em adotar fontes distintas de energia:

Figura 2 – Comparação entre os modelos de malhas elétricas



Fonte: (FOUNDATION et al., 2018)

Independentemente da malha, no contexto do consumo residencial, entender os padrões de uso em regiões particulares é essencial para inferir o consumo total desse setor, seja a nível nacional de um determinado país, seja a nível regional, para calcular, por exemplo, necessidades de maior fornecimento ou mudanças no consumo provocadas por fatores externos (WANG et al., 2024). Analogamente, para a transição para as Smart Grids, é necessário compreender a produção e o consumo de energia, para inferir a estabilidade econômica dessa rede (SCHäFER et al., 2015). Uma possível abordagem é utilizar algoritmos de inteligência artificial (IA) para essas tarefas.

1.1 OBJETIVOS

Este trabalho de conclusão de curso propõe a construção de modelos de inteligência artificial para:

- Regressão: utilizando informações geográficas, de demanda e de preço de um certo mês, determinar o consumo total de energia elétrica pelo setor residencial de um país naquele mês;
- Classificação binária: utilizando informações dos participantes da malha, determinar a estabilidade de uma Smart Grid (estável ou instável).

Em particular, uma etapa fundamental da construção de um modelo de IA é a escolha dos seus hiperparâmetros. É proposto, então, o uso de algoritmos evolutivos (Evolutionary Algorithms - EA), a saber, Genetic Algorithm (GA), Differential Evolution (DE) e Particle Swarm Optimization (PSO), para encontrar os melhores hiperparâmetros para modelos em cada tarefa. Essa abordagem é comparada a uma ferramenta mais consolidada de busca de hiperparâmetros no contexto de IA, o Optuna (AKIBA et al., 2019), que conta com implementações do estado da arte dessa área a partir do uso de técnicas de otimização bayesiana. O melhor conjunto de hiperparâmetros é considerado como aquele que minimiza o Root Mean Square Error (RMSE), para a regressão, e aquele que maximiza a Acurácia, para a classificação, que é são as funções objetivo em questão. O trabalho busca, então, uma resposta para a pergunta:

- O uso de algoritmos evolutivos para a busca de hiperparâmetros em modelos de inteligência artificial alcança um desempenho melhor ou, pelo menos, tão bom quanto o Optuna, ferramenta consolidada na área?

2 CONCEITOS TEÓRICOS

O objetivo deste capítulo é expor os conhecimentos prévios, relativos às áreas de otimização e inteligência artificial, necessários para a compreensão do escopo do projeto. São esclarecidos os principais conceitos relacionados aos algoritmos evolutivos, voltados para problemas de otimização não linear, e às redes neurais, subconjunto percentente à grande área de IA.

2.1 OTIMIZAÇÃO

> "A natureza otimiza: sistemas físicos tendem a um estado de energia mínima, as moléculas em um sistema químico isolado reagem entre si até que a energia potencial total de seus elétrons seja minimizada, raios de luz seguem caminhos que minimizam o seu tempo de percurso. As pessoas otimizam: investidores buscam criar portfólios que evitem riscos excessivos, enquanto alcançam uma alta taxa de retorno; fabricantes visam máxima eficiência no design e na operação dos seus processos produtivos, engenheiros ajustam parâmetros para otimizar a performance de seus projetos." (NOCEDAL; WRIGHT, 2006)

Todos esses fenômenos podem ser transformados em um problema matemático por meio de um processo chamado de modelagem. Primeiramente, identificamos uma certa função objetivo, uma métrica que quantifica a qualidade do estado do sistema em questão. Essa função depende de variáveis, valores que caracterizam o sistema. Em alguns casos, é possível que se deseje valores restritos para elas. Uma vez modelado, queremos encontrar nesse sistema uma disposição das variáveis que, convencionalmente na literatura, minimize a função objetivo, resolvendo assim um problema de otimização, alcançando uma solução ótima.

2.1.1 Formulação Matemática

Seja $f : X \to Y$ uma função objetivo, que mapeia os elementos $x \in X$ para $f(x) \in Y$. Queremos encontrar a solução ótima, um elemento $x^* \in X$ tal que $f(x^*) \leq f(x)\ \forall x \in X$:

$$x^* = arg\min_{x \in X} f(x). \tag{2.1}$$

Quando é o caso, podemos adicionar restrições para as variáveis:

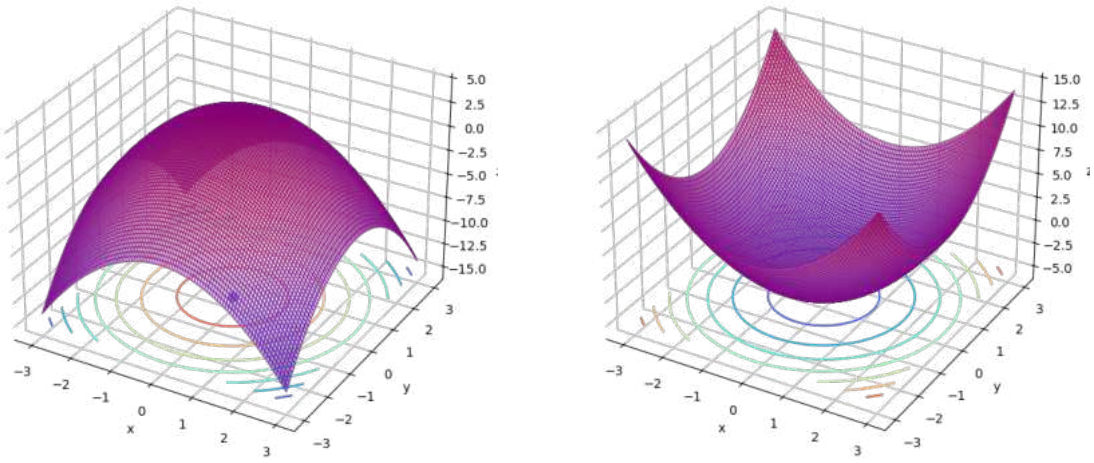
$$\begin{aligned} x^* &= \arg\min_{x \in X} f(x) \\ \text{s.a.} \quad & g(x) \leq 0 \\ & h(x) = 0. \end{aligned} \tag{2.2}$$

Embora a formulação do problema de otimização seja definida formalmente em termos da minimização de f, caso tenhamos uma modelagem voltada para a maximização, basta apenas minimizar $-f$ (NOCEDAL; WRIGHT, 2006):

$$\max_{x \in X} f(x) = \min_{x \in X} -f(x). \tag{2.3}$$

Figura 3 – Exemplo de uma função, $f : \mathbb{R}^2 \to \mathbb{R}$, que se deseja maximizar. Utilizando os métodos de minimização, basta minimizar $-f$ para encontrar o valor de x^*. Para esse caso particular, $x^* = (0, 0)$: $f(x^*) = 4$, valor máximo de f, e $-f(x^*) = -4$, valor mínimo de $-f$.

(a) $f(x, y) = x^2 + y^2 + 4$ (b) $-f(x, y) = -x^2 - y^2 - 4$



Fonte: o autor

Diversas técnicas clássicas foram desenvolvidas para resolver casos especiais dos problemas (2.1) e (2.2), de maneira a garantir-se analiticamente a convergência para a solução exata. Alguns exemplos (CUNHA; TAKAHASHI; ANTUNES, 2012) são:

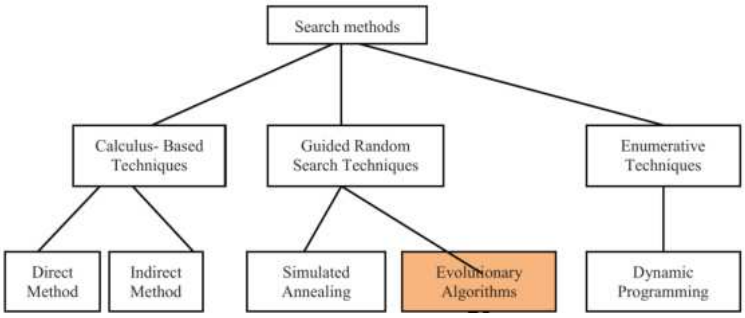
- Simplex ou métodos de pontos interiores: desenvolvidos para casos em que a função objetivo e as possíveis restrições são funções lineares;
- Métodos de Newton e Quasi-Newton: desenvolvidos para quando a função é contínua, unimodal e possui segunda derivada;
- Método do Gradiente: desenvolvido para funções contínuas, diferenciáveis e unimodais.

2.1.2 Algoritmos Evolutivos

Apesar da diversidade de abordagens existentes para problemas de otimização, nem sempre a função objetivo se encaixa nos casos particulares exigidos por esses métodos

(CUNHA; TAKAHASHI; ANTUNES, 2012). Em cenários de problemas reais mais complexos, como no caso das redes elétricas, é possível que sequer exista uma expressão analítica para f, o que impossibilita a análise teórica de suas propriedades. Nesses casos, em geral sabemos receber um valor $f(x)$ para qualquer ponto x no espaço de variáveis do problema. Com essa única fonte de informação, é possível utilizar algoritmos evolutivos e encontrar soluções suficientemente boas.

Figura 4 – Algoritmos evolutivos como um dos métodos de busca da solução ótima. Os métodos clássicos em geral são baseados em técnicas de cálculo.



Fonte: (VIKHAR, 2017)

Essa abordagem explora o espaço de soluções do problema simulando a evolução de uma população ao longo do tempo. Ela é inspirada pelos conceitos biológicos de evolução, propostos por Charles Darwin, que descrevem a sobrevivência dos indivíduos de uma população de acordo com sua adaptação com o ambiente (VIKHAR, 2017), podendo também utilizar-se de fenômenos naturais como o comportamento social da migração de pássaros (KENNEDY; EBERHART, 1995).

A inspiração biológica, de forma simplificada, está baseada na ideia de que, na natureza, para sobreviver, indivíduos precisam se adaptar ao ambiente em que vivem. Aqueles com características melhores adaptadas possuem maior chance de sobreviver e de se reproduzir, passando-as adiante para a próxima geração e perpetuando-as, enquanto os demais são eliminados. É possível que, em determinado momento nasça um indivíduo com características únicas, ao acaso, não herdadas de seus progenitores, que podem ou não ser adaptativas, fenômeno chamado de mutação. Todo esse processo biológico é chamado de seleção natural.

2.1.3 Genetic Algorithm

Algoritmo atribuído normalmente a John Holland e desenvolvido nos anos de 1970, que se baseia nas ideias do Darwinismo (LAMBORA; GUPTA; CHOPRA, 2019). Um conjunto $P = \{x_1, x_2, ..., x_n\} \subset X$ de pontos do domínio, representando indivíduos, sendo n é o tamanho da população, é gerado aleatoriamente, com o objetivo de minimizar a função objetivo $f : X \to Y$. A métrica da adaptação de x_i é dada por $f(x_i)$ $\forall i \in$

$\{1, 2, ..., n\}$: quanto menor $f(x_i)$, mais adaptado é x_i naquela população. A cada iteração, que representa uma geração, os indivíduos são recombinados, com a inspiração biológica da reprodução, gerando novos pontos, e podem sofrer uma mutação ao acaso. Após essa etapa, teremos $n + p$ indivíduos, onde p é o número de pontos gerados na etapa de recombinação. São selecionados então os n mais aptos, de acordo com seu valor f, e o processo é repetido até que se alcance o número máximo de gerações ou uma solução suficientemente boa seja alcançada.

Os processos de seleção, recombinação e mutação são os responsáveis pela criação das novas gerações, conduzindo assim a população para uma solução:

- A seleção escolhe os melhores indivíduos, de acordo com seu valor de f, para recombiná-los;
- A recombinação acontece com probabilidade $p_{\text{crossover}}$ e junta características de indivíduos, gerando filhos que mantém as características de seus pais;
- A mutação acontece com probabilidade p_{mutation} mudando alguma característica do indivíduo x.

Algorithm 1 Exemplo de uma implementação do Algoritmo Genético

```
1: P ← generate_population()                    ▷ Gera uma população aleatória
2: P.evaluate_fitness()                         ▷ Calcula f para cada indivíduo da população
3: for 1 : n_generations do                     ▷ Para cada geração...
4:     P.selection()                            ▷ Seleciona os n melhores indivíduos
5:     for x ∈ P do                             ▷ Para cada indivíduo da população...
6:         child ← crossover(x)                 ▷ Gera um filho de x com probabilidade p_crossover
7:         mutated ← mutate(x)                  ▷ Muta x com probabilidade p_mutation
8:         if f(child) < f(x) then              ▷ Se o valor de f for mais adaptado
9:             P.update_population(child)       ▷ Adiciona o indivíduo na população
10:        end if
11:    end for
12: end for
```

2.1.4 Differential Evolution

Proposto em (STORN; PRICE, 1997). O algoritmo de Evolução Diferencial é particularmente semelhante ao Algoritmo Genético. A diferença encontra-se principalmente na forma de realizar os procedimentos de recombinação e mutação. Nenhum indivíduo sofre mutações; ao invés disso, gera-se um indivíduo x mutado a partir da equação

$$x = x_{base} + F(x_{r_1} - x_{r_2}). \tag{2.4}$$

O elemento x_{base} depende da estratégia adotada pelo algoritmo, podendo ser um indivíduo aleatório ou o melhor indivíduo até a iteração atual. Já x_{r_1} e x_{r_2} são indivíduos

escolhidos aleatoriamente da população e $F \in (0, 1)$ é um parâmetro usado para ponderar a diferença e controlar o ritmo de evolução da população. Sendo assim, indivíduos mutados são gerados a partir de uma perturbação dos membros daquela geração.

A recombinação, por sua vez, é feita combinando-se as componentes de um indivíduo x_i da população original e um indivíduo x_j gerado por mutação, de maneira a introduzir uma maior diversidade. Dessa forma, gera-se um indivíduo *trial*, que combina as características de seus pais. A decisão de adicionar ou não a característica de um indivíduo mutado é dada pelo parâmetro $Cr \in (0, 1)$:

Figura 5 – Ilustração da recombinação no Differential Evolution. Da esquerda para a direita: um indivíduo da população, um indivíduo gerado por mutação e um indivíduo resultado da recombinação.

Fonte: o autor

Selecionam-se então os melhores indivíduos, dentre todos os existentes, para compor a próxima geração.

Algorithm 2 Exemplo de uma implementação do Evolução Diferencial

```
P ← generate_population()                          ▷ Gera uma população aleatória
P.evaluate_fitness()                               ▷ Calcula f para cada indivíduo da população
for 1 : n_generations do                           ▷ Para cada geração...
    for x ∈ P do                                   ▷ Para cada indivíduo da população...
        mutated ← mutate(x)                        ▷ Muta x com probabilidade p_mutation
        random ← select_random_vector_from_population(P)
        crossover_vector ← crossover(mutated, random)
        if f(crossover_vector) < f(x) then
            update_Fit(crossover_vector)
            update_P(crossover_vector)
        end if
    end for
end for
```

2.1.5 Particle Swarm Optimization - PSO

Proposto em (KENNEDY; EBERHART, 1995). Baseia-se na inspiração biológica do comportamento social de bandos de pássaros ou cardumes de peixes. O PSO diferencia-se do GA e do DE pelo fato de não simular um contexto de sobrevivência dos mais aptos, mas na troca de informação entre os indivíduos da população, que transitam pelo ambiente e comunicam-se entre si.

Cada indivíduo ou *partícula* $x \in P$ possui posição, dada pelas próprias componentes de x, e velocidade, que direciona seu movimento no domínio de f. A cada iteração, representando um instante de tempo, as partículas ajustam suas trajetórias com base em duas informações: sua própria melhor posição já encontrada, usando como métrica $f(x)$, e a melhor posição já encontrada levando em consideração toda a população.

O movimento da partícula é regido pela sua velocidade, que é atualizada a cada iteração de acordo com a equação

$$v_x^{(t+1)} = w v_x^{(t)} + c_1 r_1 (p_{\text{best}} - x^{(t)}) + c_2 r_2 (g_{\text{best}} - x^{(t)}), \tag{2.5}$$

na qual:

- $v_x^{(t+1)}$ é a velocidade da partícula x na iteração $t+1$;
- w é o termo de inércia;
- $v_x^{(t)}$ é a velocidade da partícula x na iteração t;
- c_1 é o termo cognitivo;
- r_1 e r_2 são números reais aleatórios no intervalo $(0, 1)$;
- p_{best} é a melhor posição encontrada pela partícula x até o momento;
- c_2 é o termo de aprendizado social;
- g_{best} é a melhor posição encontrada por toda a população até o momento;
- $x^{(t)}$ é a posição da partícula x na iteração t.

Após a atualização da velocidade, a nova posição da partícula é calculada por

$$x^{(t+1)} = x^{(t)} + v_x^{(t+1)}, \tag{2.6}$$

fazendo com que as partículas "voem" pelo domínio de f.

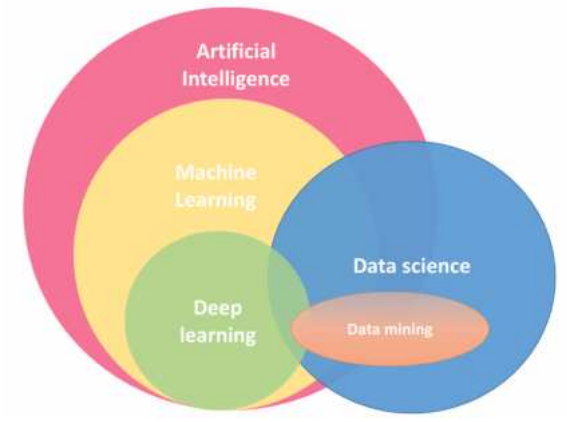
Algorithm 3 Exemplo de uma implementação do Particle Swarm Optimization

```
P ← generate_population()        ▷ Gera uma população com posições e velocidades aleatórias
P.evaluate_fitness()             ▷ Calcula f para cada partícula da população
for 1 : n_iters do               ▷ Para cada iteração...
    for x ∈ P do                 ▷ Para cada partícula...
        if f(x) < x.best_fit then    ▷ Se f(x) for melhor do que sua atual melhor f
            x.best_position = x      ▷ Atualiza a melhor posição encontrada por x
            x.best_fit = f(x)        ▷ Atualiza o melhor valor de f encontrado por x
            if f(x) < P.g_best then  ▷ Se f(x) for melhor do que a atual g_best
                P.g_best_position = x    ▷ Atualiza a melhor posição encontada pela população
                P.g_best_fit = f(x)      ▷ Atualiza o melhor valor de f encontrado pela população
            end if
        end if
        x.update_vel()           ▷ Calcula a nova velocidade de x
        x.update_pos()           ▷ Calcula a nova posição de x
    end for
end for
```

2.2 REDES NEURAIS

Redes neurais são um caso particular de algoritmos de aprendizado de máquina (machine learning), que, por sua vez, são um subconjunto da grande área de inteligência artificial. Portanto, para entender o funcionamento das redes neurais, é preciso uma familiarização com os conceitos elementares de aprendizado de máquina (**??**).

Figura 6 – Ilustração da área de inteligência artificial e áreas correlatas



Fonte: (KULIN et al., 2021)

2.2.1 Algoritmos de Aprendizado

Um algoritmo de aprendizado de máquina é capaz de, dado um certo conjunto de dados, aprender a realizar alguma tarefa: jogar jogos eletrônicos (OpenAI et al., 2019; VINYALS et al., 2019) ou de tabuleiro (SILVER et al., 2016), reconhecer objetos ou pessoas em uma foto ou vídeo (ZAIDI et al., 2022), conversar via texto ou áudio com uma pessoa (ROUMELIOTIS; TSELIKAS, 2023), dentre muitas outras aplicações. Mais formalmente, "um programa de computador 'aprende' pela experiência E em relação a alguma classe de tarefas T e uma métrica de performance P se seu desempenho nas tarefas em T, medido por P, melhora com E" (MITCHELL, 1997).

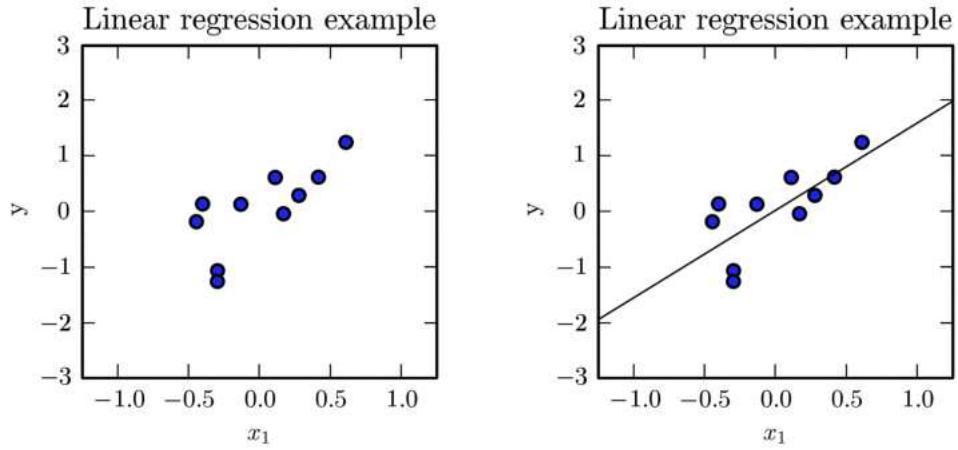
Classicamente, os algoritmos de aprendizado são classificados em tipos, como, por exeplo, supervisionado, não supervisionado, por reforço e auto-supervisionado. No escopo deste trabalho, levamos em consideração apenas o aprendizado supervisionado.

2.2.2 Aprendizado Supervisionado

Os modelos construídos utilizando o paradigma supervisionado recebem um conjunto $S = \{(x_1, y_1), (x_2, y_2), ...(x_n, y_n)\}$, chamado de conjunto de treinamento, no qual:

- n é o número de amostras;
- $x \in \mathbb{R}^d$ é um vetor que representa o dado, chamado de vetor de atributos ou de features, com d atributos;
- y_i é a saída esperada para uma entrada x_i, $\forall i \in \{1, 2, ..., n\}$.

Em geral, essa abordagem é utilizada para tarefas de regressão, quando y é um valor contínuo, e de classificação, quando y é um valor discreto representando as classes.

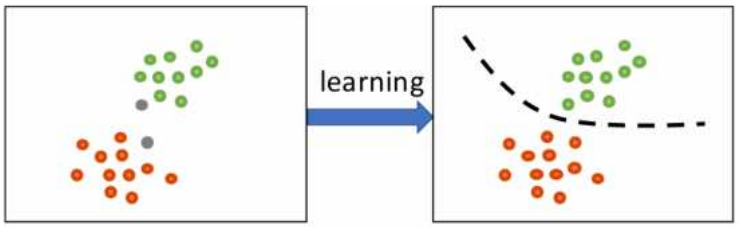


(a) Conjunto (x_i, y_i) de pontos, que representam os dados

(b) Função $y = f(x)$ que aproxima os pontos

Fonte: adaptado de (LECUN; BENGIO; HINTON, 2015)

Figura 7 – Exemplo de uma regressão. São conhecidos diversos exemplos (x_i, y_i) que relacionam as d features de $x_i \in \mathbb{R}^d$ de uma certa instância i com o valor de saída y_i. O modelo precisa encontrar uma função $y = f(x)$ que melhor aproxime os pontos dados, de maneira que, com uma nova entrada não conhecida, seja possível produzir uma saída.



Fonte: (CHEN et al., 2024)

Figura 8 – Exemplo de uma classificação. São conhecidos diversos exemplos (x_i, y_i) que relacionam as d features de $x_i \in \mathbb{R}^d$ de uma certa instância i com a classificação y_i. O modelo precisa encontrar uma função $y = f(x)$ de maneira que, com uma nova entrada não conhecida, seja possível produzir uma classificação.

2.2.3 Perceptron

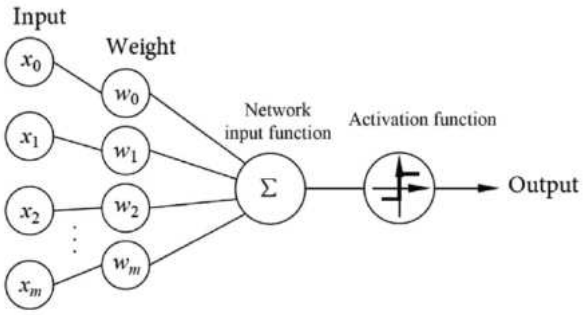
O Perceptron é uma rede neural simples (LIANG, 2020). Um dado, representado como um vetor x, é recebido por nós de entrada, chamados de neurônios, de tal maneira que a componente x_i seja associado ao neurônio i. Cada nó de entrada dispara para um único neurônio de saída um valor $w_i x_i$, onde w_i é o peso do nó i. É feito, então, um somatório dos valores de saída de cada nó, seguido de uma função de ativação, produzindo o valor de saída.

Mais formalmente, podemos definir o Perceptron da seguinte maneira: sejam $i \in \{1, 2, ..., n\}$ os neurônios de entrada, onde n é o tamanho do vetor x, w_i o peso associado

ao nó i, f uma função de ativação e o o neurônio de saída. Então a saída do algoritmo é:

$$o(x) = f(\sum_{i=1}^{n} w_i x_i). \tag{2.7}$$

Figura 9 – Representação da arquitetura do Perceptron



Fonte: (LIANG, 2020)

O Perceptron pode ser usado, por exemplo, para uma tarefa de classificação binária. "Treinar" essa rede significa simplesmente encontrar valores satisfatórios para os pesos w, de maneira a obter-se valores satisfatórios para o. Nesse caso, suponha:

- Um conjunto de treinamento $S = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$, com $y = 1$ ou $y = -1$;
- Uma função $L(y, o(x))$ de perda ou custo, que dê uma métrica de erro entre o valor verdadeiro y e o valor $o(x)$ produzido pela rede;
- Uma taxa de aprendizado $\alpha \in (0, 1)$, que rege a atualização dos valores dos pesos. Valores próximos de 0 fazem com que os pesos mudem de forma pouco significativa a cada iteração, enquanto que valores mais próximos de 1 causam uma mudança expressiva.

É possível utilizar-se de um algoritmo de aprendizado de máquina para encontrar valores ideais para os pesos:

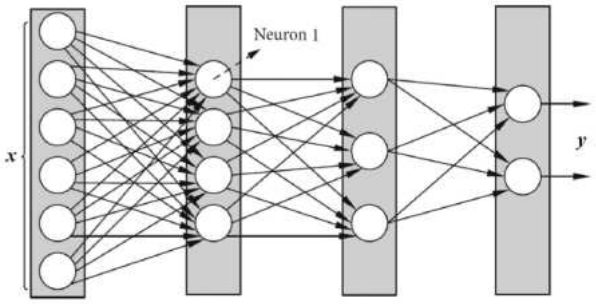
Algorithm 4 Treinamento de uma rede neural Perceptron

1: **for** $i \in \{1, 2, ..., m\}$ **do** ▷ Para cada neurônio de entrada...
2: $w_i \leftarrow$ random() ▷ Inicializa aleatoriamente o peso daquele neurônio
3: **for** $(x, y) \in S$ **do** ▷ Para cada instância dos dados de treinamento...
4: **for** $i \in \{1, 2, ..., m\}$ **do** ▷ Para cada neurônio...
5: $w_i^{(t+1)} \leftarrow w_i^{(t)} + \alpha \cdot L(y, o(x)) \cdot x_i$ ▷ Atualiza o peso do neurônio i para a iteração $t + 1$
6: **end for**
7: **end for**
8: **end for**

2.2.4 Multilayer Perceptron - MLP

Partindo da estrutura do Perceptron, é possível adicionar neurônios entre os nós de entrada e o nó de saída, criando-se assim as camadas ocultas ou escondidas, gerando uma arquitetura multicamadas, em contrapartida à camada única com entrada e saída (LIANG, 2020). O uso de camadas ocultas adiciona a noção de profundidade na rede, gerando o termo aprendizado profundo.

Figura 10 – Multilayer Perceptron com uma camada de entrada, duas camadas ocultas e uma camada de saída.



Fonte: (LIANG, 2020)

O incremento em relação ao Perceptron tradicional é que cada neurônio da camada de entrada está conectado com todos os nós da camada oculta seguinte, que por sua vez também possuem todos os nós interconectados com os da seguinte, sucessivamente até a camada de saída. Mais formalmente, podemos definir a saída do MLP como

$$o(x) = f_k(W_k...f_2(W_2(f_1(W_1x))). \tag{2.8}$$

Nesse caso, W_j é a matriz de pesos dos neurônios da camada $j-1$ para a camada j e f_j é uma função de ativação, $\forall j \in \{1, 2, ..., k\}$, com k sendo o número de camadas da rede. A adição dessa maior complexidade, envolvendo matrizes de pesos e funções de ativação em cada camada, produz a performance característica das redes neurais artificiais, em particular pela introdução da não linearidade no modelo, permitindo que o MLP capture relações complexas entre os dados de entrada e de saída (WANG et al., 2024). "Treinar" uma rede neural artificial multicamadas é simplesmente encontrar valores satisfatórios para as matrizes W.

O treinamento de um MLP é um processo mais sofisticado do que o descrito no Algoritmo 4, envolvendo, normalmente, o algoritmo de backpropagation, a retropropagação do erro. A intuição é semelhante à arquitetura básica: compara-se a saída da rede com a saída esperada, utilizando uma função $L(x, o(x))$; nesse caso, atualiza-se os pesos de acordo com a derivada de L, visando-se sua minimização.

2.2.5 Hiperparâmetros

Uma etapa importante no desenvolvimento de uma rede neural é a escolha de seus hiperparâmetros. Esses valores controlam tanto a arquitetura do modelo quanto o seu comportamento e não fazem parte do escopo de treinamento. Por um lado, um MLP encontra valores ideais para suas matrizes W de peso, por outro, os hiperparâmetros que o caracterizam precisam ser escolhidos de antemão. A escolha desses valores pode ser crucial para a performance do modelo:

- Número de camadas: embora tenhamos, de forma fixa, uma camada de entrada, para receber os dados, e uma camada de saída, que produz a resposta, o número de camadas ocultas é um inteiro arbitrário;
- Número de neurônios em cada camada: na camada de entrada, o número de nós é igual ao tamanho do vetor x, e na camada de saída, o número de nós é 1, para problemas de regressão, ou c, para problemas de classificação, com c sendo o número de classes possíveis. No caso das camadas escondidas, pode-se adicionar um número inteiro arbitrário de neurônios em cada camada;
- Funções de ativação: escolher a função de ativação ideal é vital para a performance do modelo, visto que ela é o que adiciona a captura de relações não lineares nos dados. Pode-se ainda escolher diferentes funções para diferentes camadas;
- Taxa de aprendizado: um número real $\alpha \in (0, 1)$ arbitrário. Se muito pequeno, pode fazer com que o treinamento seja extremamente lento; se muito grande, pode deixar o modelo instável e impedir a convergência;
- Número de épocas: a quantidade de vezes que o conjunto de treinamento é processado. Poucas vezes podem levar ao fenômeno de underfitting, quando o modelo é muito simples para capturar a relação entre os dados, e muitas vezes podem levar ao overfitting, quando o modelo aprende somente os dados de treinamento, não conseguindo generalizar para novas instâncias;
- Taxa de regularização: um número real $r > 0$ que previne o overfitting e melhora a generalização do modelo. Se muito alta, o modelo sofre um underfitting, se muito baixa, um overfitting.

Sendo assim, determinar bons valores para os hiperparâmetros de uma rede neural artificial é de extrema importância no processo de sua construção.

3 PROPOSTA

Neste capítulo é descrita e caracterizada a implementação da proposta do trabalho. São utilizados o Optuna e os algoritmos evolutivos Genetic Algorithm, Differential Evolution e Particle Swarm Optimization para buscar hiperparâmetros para uma rede neural, dentro de um certo espaço de busca, para duas tarefas, uma de regressão e outra de classificação, com o objetivo de alcançar as configurações de melhor performance.

3.1 HIPERPARÂMETROS

Os hiperparâmetros utilizados para compor o espaço de busca tanto no Optuna quanto nas metaheurísticas são:

- h_1: Função de ativação $\in$ {ReLU, Tangente Hiperbólica, Sigmoide}: a função de ativação única que é utilizada nas camadas escondidas. Não é possível escolher uma função de ativação diferente para cada camada;
- h_2: Otimizador $\in$ {Adam, Limited-memory BFGS}: o algoritmo de otimização responsável pela função de perda e pela atualização dos pesos da rede;
- h_3: Alpha $\in [0, 1]$: a força da regularização L2;
- h_4: Taxa de aprendizado $\in$ {Constante, Escala inversa, Adaptativa}: se Constante, o valor inicial é mantido durante todo o treinamento, se Escala Inversa, a taxa de aprendizado decai a cada passo do algoritmo, se Adaptativa, mantém o valor constante enquanto a função de perda continua sendo minimizada, quando isso não acontece mais, o valor é dividido por 5;
- h_5: Inicialização da taxa de aprendizado $\in [10^{-6}, 0.1]$: o valor da taxa de aprendizado no início do algoritmo;
- h_6: Tamanho do lote $\in [32, 512]$: o tamanho do lote visto pelo modelo em cada iteração de atualização dos pesos;
- h_7: $\beta_1 \in [0, 1]$: o decaimento exponencial para a estimativa do vetor do primeiro momento, utilizado somente no otimizador Adam;
- h_8: $\beta_2 \in [0, 1]$: o decaimento exponencial para a estimativa do vetor do segundo momento, utilizado somente no otimizador Adam.

Os números de camadas escondidas e de neurônios foram mantidos constantes durante todo o experimento tanto para as heurísticas quanto para o Optuna. Ambos possuem 5

camadas com, respectivamente, 20, 15, 10, 5 e 2 neurônios. Sendo assim, um indivíduo $h \in P$ é da forma $h = (h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8)$.

3.2 FUNÇÃO OBJETIVO

A função objetivo f é dada pela média do resultado de um treinamento feito em validação cruzada, utilizando KFold, com $k = 5$:

$$f(h) = \text{mean}(\text{cross\_validation}(X, y, m_h)). \tag{3.1}$$

X é o conjunto de treinamento, y são os valores associados a cada instância $x \in X$ e m_h é um modelo construído com o conjunto h de hiperparâmetros. A métrica do treinamento em validação cruzada é o Root Mean Square Error (RMSE) para a regressão e a acurácia para a classificação. Basta então selecionar um conjunto de treinamento e utilizar tanto os algoritmos evolutivos quanto o Optuna para buscar os hiperparâmetros que construam o modelo que melhor performa em relação a cada métrica.

4 TRABALHOS RELACIONADOS

A caracterização do problema de otimização de hiperparâmetros em modelos em aprendizado de máquina é explorada em (FRANCESCHI et al., 2024): seja $A(\lambda)$ um algoritmo de aprendizado, configurado pelo hiperparâmetro λ, e M uma função que avalia sua performance em alguma tarefa. A composição $f(\lambda) = M \circ A(\lambda)$ é chamada de função de resposta. A otimização de hiperparâmetros consiste em otimizar f com relação a λ. Para os autores, o que distingue a busca por λ^* de um problema de otimização convencional é o fato de que avaliar f significa executar um algoritmo de aprendizado, o que, na prática, pode ser extremamente custoso em termos computacionais.

O uso do algoritmo evolutivo Genetic Algorithm é feito para encontrar hiperparâmetros de um modelo de regressão logística para classificação de dados do MNIST em (NURMAMATOV; SARIYEV; ESHONKULOV, 2025). Nesse experimento, o espaço de busca consiste em buscar bons valores para a taxa de aprendizado [dentre 0.001, 0.01 e 0.1], o tamanho do lote [dentre 16, 32, 64 e 128] e o número de camadas [dentre 2, 3 e 4]. É feita uma comparação do uso do GA em relação ao Grid Search e ao Random Search, com um ganho modesto na acurácia. No entanto, levando em consideração o tempo de processamento e o número de combinações testadas, o uso do GA se mostrou satisfatório:

Optimization method	Highest accuracy	Calculation time (minutes)	Number of combinations tested
Genetic algorithm	94.6	50	50
Grid Search	94.3	120	108
Random Search	93.2	30	20

Fonte: (NURMAMATOV; SARIYEV; ESHONKULOV, 2025)

No caso de problemas envolvendo energia elétrica, é feita uma relação entre certos atributos de entrada e o consumo total pelo setor residencial em (WANG et al., 2024) utilizando dados dos Estados Unidos. A proposta é supõe ser possível determinar esse valor utilizando-se de informações geográficas e de preço da energia. Os autores utilizam então redes neurais para essa tarefa. Nesse caso, o uso de algoritmos evolutivos encontra-se não no encontro de hiperparâmetros suficientemente bons, mas na substituição do gradiente descendente para o treinamento. São utilizadas três heurísticas para encontrar os pesos do MLP implementado: Heap-Based Optimizer (HBO), Multiverse Optimizer (MVO), Whale Optimization Algorithm (WOA), atingindo um RMSE de 68.44 e R^2 de 0.98 no melhor caso nos dados de teste:

	Swarm size	Training dataset		Testing dataset	
		RMSE	R^2	RMSE	R^2
HBOMLP	150	90.25319	0.97150	96.67639	0.96449
MVOMLP	500	48.55082	0.99184	68.44517	0.98236
WOAMLP	400	63.19792	0.98613	75.10985	0.97872

Fonte: (WANG et al., 2024)

O problema da determinação da estabilidade em uma Smart Grid é descrito e explorado em (ARZAMASOV; BOHM; JOCHEM, 2018). Os autores modelam a rede inteligente utilizando descrições e equações da física, descrevendo a rede inteligente em termos de uma equação diferencial ordinária de segunda ordem. A equação em questão possui infinitas soluções, das quais uma parcela finita assume valores reais positivos. Essas parcelas caracterizam a estabilidade do sistema em questão.

Como caracterizar uma Smart Grid em termos de equações diferenciais pode se tornar um problema conforme o número de participantes da malha aumenta. Há diversas abordagens que utilizam aprendizado de máquina para essa finalidade, e encontram bons resultados, como, por exemplo: (MASSAOUDI et al., 2021), a partir do uso de Redes Neurais Recorrentes (Recurrent Neural Networks - RNN), (ALAZAB et al., 2020) ou implementando um modelo de Long short-term memory (LSTM).

Em particular, existem estudos comparativos entre diversos modelos para determinar a estabilidade de uma malha inteligente, demonstrando que, em geral, a performance é satisfatória: (YERROLLA.; RAMESH, 2023) compara regressão logística (81% de acurácia), árvore de decisão (99%) e Support Vector Machine (SVM) (93%), com todos apresentando boas matrizes de confusão; (IBRAR et al., 2022) compara mais algoritmos e abordagens de aprendizado de máquina, como Light Gradient Boosting Machine (LightGBM), Locally Deep SVM e XGBoost, e apresenta mais métricas: acurácia, precisão, recall e ROC. Nesse estudo, todos apresentaram valores acima de 80%.

5 EXPERIMENTOS

Todos os experimentos realizados foram conduzidos utilizando-se de:

- Processador: 4th Gen Intel Core i7-4790 x8
- RAM: 16GB
- Sistema Operacional: Linux Ubuntu 24.04
- Pacotes e biliotecas: [1]
  - Python (3.13.0)
  - Scikit-Learn (1.6.1)
  - Pymoo (0.6.1)
  - Optuna (4.4.0)
  - Scipy (1.15.3)

As redes neurais artificiais foram construídas utilizando as implementações MLPRegressor e MLPClassifier do Scikit-Learn. Os algoritmos evolutivos foram utilizados a partir da Pymoo.

5.1 CONJUNTO DE DADOS UTILIZADOS

Como temos duas tarefas, uma de regressão e outra de classificação, foram utilizados duas bases de dados, uma para cada ocasião:

5.1.1 Regressão: Consumo de energia

A tarefa de regressão consiste em, dadas certas informações geográficas e relativas a preço, determinar o consumo total de energia pelo setor residencial de um país. Para tal, foi utilizado um conjunto de dados reais, medidos pela Administração de Informação de Energia dos Estados Unidos (U.S. Energy Information Administration - EIA), uma agência do Departamento de Energia dos EUA (EIA, 2025). Essa base é interpretada e modelada em (WANG et al., 2024), de maneira a relacionar esses determinados atributos de entrada com um valor de saída. Como essas correlações foram feitas pelos autores, houve uma preferência para utilizar esses dados já compreendidos. De acordo com essa perspectiva, os seguintes fatores impactam nesse consumo:

[1] Código fonte disponível em https://github.com/pedro-jorge/TCC, incluindo o .yml com todas as dependências

- Graus-dia de aquecimento: índice que mede a demanda de energia para aquecimento de edificações (STENSJö; FERREIRA; LOURA, 2017);
- Graus-dia de resfriamento: índice que mede a demanda de energia para resfriamento de edificações (STENSJö; FERREIRA; LOURA, 2017);
- Preço médio da eletricidade para consumidores finais no setor residencial;
- Preço médio da eletricidade para todos os consumidores finais.

A base que conta com essas informações foi colhida do website oficial da EIA (EIA-MONTHLY, 2025), utilizando os arquivos de:

- "Average prices of electricity to ultimate customers", da categoria Energy Prices;
- "Cooling degree days by census division", da categoria Energy overview;
- "Heating degree days by census division", da categoria Energy overview;
- "Residential sector energy consumption", da categoria Energy consumption by sector.

Todos os arquivos contam com uma métrica mensal, desde 01/07/1976, de, respectivamente, o preço médio da eletricidade para todos os setores de consumo nos EUA, os graus-dia de resfriamento por região, os graus-dia de aquecimento por região e o consumo total do setor de energia no país. Ao juntar todas as informações utilizando como chave a data, cada instância possui os seguintes atributos:

Tabela 1 – Atributos do dataset de regressão

Atributo
Graus-dia de aquecimento - Nova Inglaterra
Graus-dia de aquecimento - Meio Atlântico
Graus-dia de aquecimento - Centro-Leste
Graus-dia de aquecimento - Centro-Oeste
Graus-dia de aquecimento - Atlântico Sul
Graus-dia de aquecimento - Centro-Sul
Graus-dia de aquecimento - Centro-Sul Oest
Graus-dia de aquecimento - Região Montanhosa
Graus-dia de aquecimento - Pacífico
Graus-dia de aquecimento - Estados Unidos
Graus-dia de resfriamento - Nova Inglaterra
Graus-dia de resfriamento - Meio Atlântico
Graus-dia de resfriamento - Centro-Leste
Graus-dia de resfriamento - Centro-Oeste
Graus-dia de resfriamento - Atlântico Sul
Graus-dia de resfriamento - Centro-Sul
Graus-dia de resfriamento - Centro-Sul Oeste
Graus-dia de resfriamento - Região Montanhosa
Graus-dia de resfriamento - Pacífico
Preço médio da eletricidade para consumidores finais - Total
Preço médio da eletricidade para consumidores finais - Residencial

A unidade de medida dos preços é em centavos de dólar por quilowatt-hora (kWh), incluindo taxas. Cada instância possui um valor numérico de saída associado, representando o total de energia consumido pelo setor residencial (em Trilhão de BTUs). A divisão das regiões segue o padrão do U.S. Census Bureau (EIA-MONTHLY, 2025):

Figura 11 – Regiões dos EUA conforme o U.S. Census Bureau

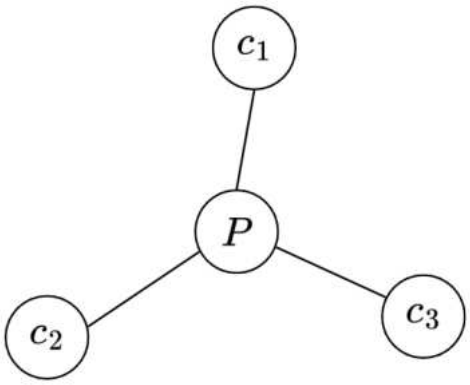
Fonte: (FLA-SHOP, 2022)

O número de instâncias é de 585, mas algumas entradas possuem valores não disponíveis em algum atributo. Após selecionarmos somente os dados que possuem todos os valores conhecidos, o número total para o experimento é de 514. Utilizando esses atributos, que caracterizam a demanda e o preço da energia elétrica para o mês em particular, assumimos que é possível produzir uma saída numérica que represente o consumo por parte do setor residencial para aquele mês.

5.1.2 Classificação: Estabilidade de Smart Grids

Para a tarefa de classificação, foi utilizado um conjunto de dados de simulação de Smart Grids, gerado por (SCHäFER et al., 2016) e disponibilizado em (ARZAMASOV; BOHM; JOCHEM, 2018). Esse dataset é frequentemente utilizado em estudos voltados para predição da estabilidade das malhas elétricas inteligentes, como em (SATU; KHAN, 2024), (EFATINASAB et al., 2025), (ALSIRHANI et al., 2023); por esse motivo, foi escolhida a sua adoção. A topologia da Smart Grid é da seguinte forma:

Figura 12 – Topologia da Smart Grid. P é o produtor de energia e c_1, c_2 e c_3 são os consumidores.



Fonte: o autor

Com base em informações envolvendo o produtor e os consumidores de energia elétrica, a classificação caracteriza a Smart Grid como estável ou instável. Cada instância conta com os seguintes (ARZAMASOV, 2018):

- τ: Tempo de reação do produtor para alterar a geração de energia com base na demanda, medido em segundos e com valores no intervalo $[0.5, 10]$;
- τ_i: Tempo de reação do consumidor i para alterar a demanda com base em perturbações na rede ou mudanças no preço, $\forall i \in \{1, 2, 3\}$, medido em segundos e com valores no intervalo $[0.5, 10]$;
- p: Energia gerada pelo produtor, com $p = |p_1 + p_2 + p_3|$;
- p_i: Energia gerada ou consumida pelo consumidor i (valores negativos para consumo, valores positivos para geração), $\forall i \in \{1, 2, 3\}$, variando no intervalo $[-0.5, -2]s^{-2}$;
- g: Coeficiente proporcional à elasticidade do preço para o produtor, com valores entre $[0.05, 1]s^{-1}$;
- g_i Coeficiente proporcional à elasticidade do preço para o consumidor i, $\forall i \in \{1, 2, 3\}$ com valores entre $[0.05, 1]s^{-1}$.

A elasticidade se refere ao quanto a demanda ou fornecimento de energia se altera conforme o preço muda. Com base nesse atributo, em conjunto com o tempo de reação dos participantes para ocorrências na rede elétrica e a quantidade de energia gerada e consumida, assumimos que é possível determinar a estabilidade de uma Smart Grid. A ideia é que, na implementação da rede inteligente, há sensores medindo esses valores, de maneira que, a cada determinado instante de tempo, seja possível produzir essa classificação.

A cada instância, com seus 12 atributos, está relacionada uma saída binária: instável ou estável. O número total utilizado é de 1000 instâncias, com 638 classificadas como instáveis e 362 como estáveis.

5.2 MÉTRICAS

5.2.1 Root Mean Square Error - RMSE

Para o caso de regressão, a métrica utilizada é o RMSE, dada por

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}, \tag{5.1}$$

com y_i e $\hat{y}_i$ sendo, respectivamente, o valor de saída da rede e o valor esperado associado à instância x_i e n o número de instâncias avaliadas. Quanto menor o RMSE, mais performático é o modelo.

5.2.2 Acurácia - ACC

Para o caso de classificação, a métrica utilizada é a Acurácia, dada por

$$\text{ACC} = \frac{\text{Número de predições corretas}}{\text{Número total de predições}} = \frac{TP + TN}{TP + TN + FP + FN}, \tag{5.2}$$

com:

- TP (True Positives) - Verdadeiros positivos
- TN (True Negatives) - Verdadeiros negativos
- FP (False Positives) - Falsos positivos
- FN (False Negatives) - Falsos negativos

Ambos os modelos são treinados utilizando validação cruzada, utilizando Kfold com $k = 5$, então a métrica o valor da função objetivo é a média do RMSE ou da ACC, a depender do caso, para uma entrada.

5.2.3 Parâmetros dos Algoritmos Evolutivos

Os EAs também dependem de parâmetros determinados à priori. Eles foram definidos, de forma arbitrária, como:

- Quantidade de iterações/gerações: 100;
- Tamanho da população: 10;

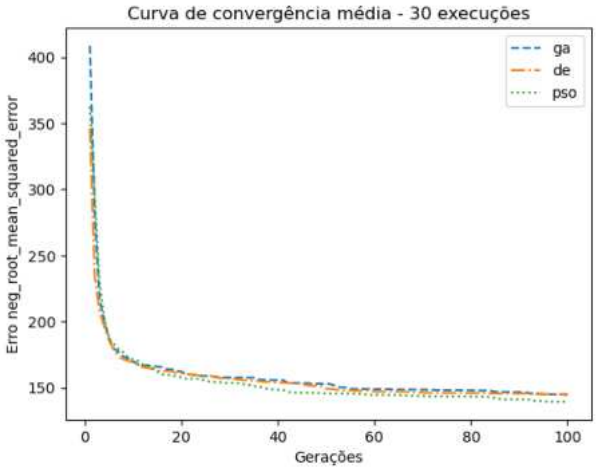
- Genetic Algorithm
  - Taxa de reprodução $p_{\text{crossover}} = 0.8$;
  - Taxa de mutação $p_{\text{mutation}} = 0.5$;
- Differential Evolution
  - Taxa de reprodução $Cr = 0.8$;
  - Taxa de mutação $F = 0.5$;
- Particle Swarm Optimization
  - Termo de inércia $w = 0.5$;
  - Termo de cognição $c_1 = 0.5$;
  - Termo de aprendizado social $c_2 = 0.3$.

5.3 RESULTADOS

5.3.1 Regressão

Foram realizadas 30 execuções dos algoritmos GA, DE, PSO e Optuna para encontrar os melhores hiperparâmetros para a tarefa de regressão ao longo de 100 iterações. É feita uma curva de convergência média, mostrando os resultados de cada geração em cada uma das execuções. O objetivo era minimizar o RMSE:

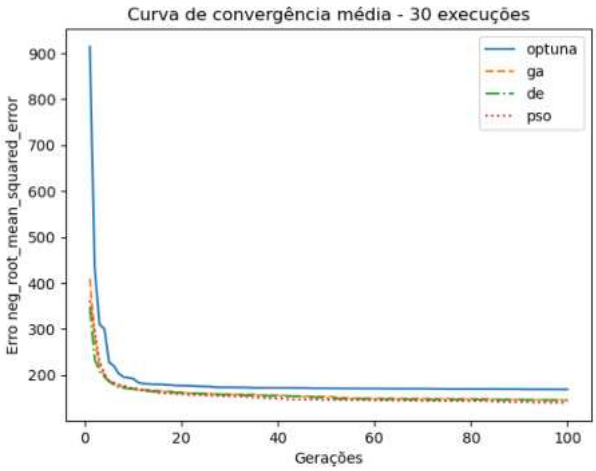
Figura 13 – Resultado da otimização dos hiperparâmetros utilizando GA, DE e PSO.



Fonte: o autor

Com a inclusão do Optuna, temos a seguinte curva:

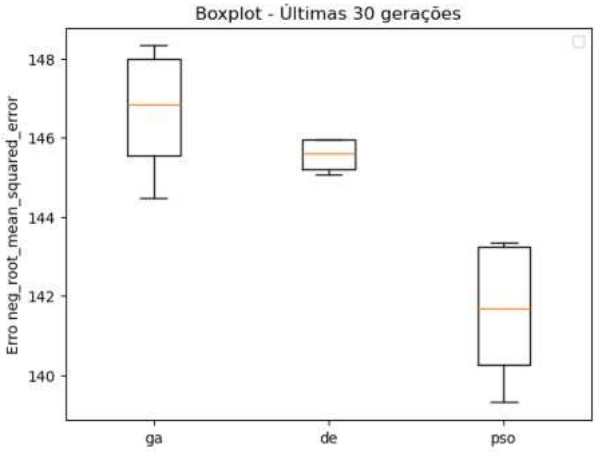
Figura 14 – Resultado da otimização dos hiperparâmetros utilizando GA, DE, PSO e Optuna



Fonte: o autor

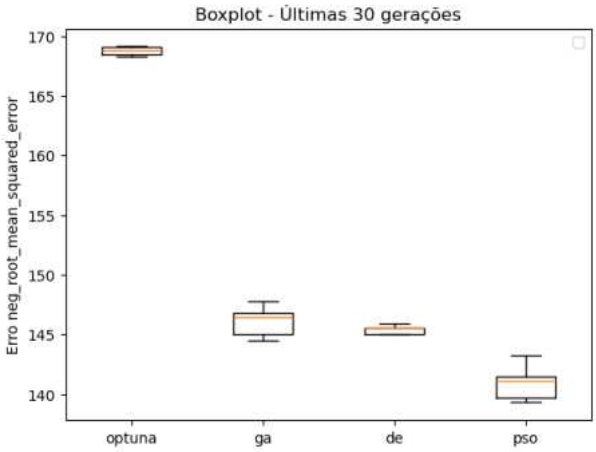
Apesar de ambas as abordagens, EAs e Optuna, alcançarem valores similares, pouco abaixo de 200, é possível notar uma modesta superioridade dos Algoritmos Evolutivos. Abaixo, podemos ver o boxplot dos resultados; os EAs aparentam apresentar resultados bem próximos entre si, mas com uma diferença em relação ao Optuna:

Figura 15 – Boxplot do resultado da otimização dos hiperparâmetros utilizando GA, DE, PSO



Fonte: o autor

Figura 16 – Boxplot do resultado da otimização dos hiperparâmetros utilizando GA, DE, PSO e Optuna



Fonte: o autor

Por fim, uma tabela numérica com valores de minimo, máximo, média e desvio padrão para cada um dos algoritmos:

Tabela 2 – Resumo dos resultados obtidos para o RMSE para cada algoritmo em 30 execuções

População	**Algoritmo**	**Min**	**Max**	**Mean**	**Std**
10	GA	144.46	408.58	158.93	30.34
	DE	145.05	346.19	156.36	23.34
	PSO	139.30	362.42	154.54	28.82
	Optuna	168.32	914.14	186.27	80.20

Repare que os valores encontrados pelo GA, DE e PSO, tanto mínimos quanto médios, estão bem próximos entre si e com uma certa distância em relação ao Optuna. É feito, então, um teste de hipótese: consideramos como H_0, hipótese nula, que a média dos melhores valores encontrados em cada algoritmo, para as 30 execuções, é igual. O método utilizado é o Teste de Friedman, que verifica a consistência de amostras que foram obtidas de maneiras diferentes. Para cada algoritmo, temos um vetor da forma

$$[f(h_a^{(1)}), f(h_a^{(2)}), ..., f(h_a^{(30)})]^T, \tag{5.3}$$

no qual $f(h_a^{(i)})$ é o valor otimizado encontrado pelo algoritmo a na execução i, $\forall i \in \{1, 2, ..., 30\}$. Caso o p-valor < 0.05, é possível afirmar, com 95% de confiança, que há uma diferença significativa entre os resultados. Utilizando o Scipy, encontramos o valor de $p = 6.1 \cdot 10^{-12}$, o que significa que a hipótese nula H_0 foi rejeitada.

Sendo significativamente diferente, é necessário utilizar um método post-hoc para encontrar onde está essa diferença. É utilizado o teste de Nemenyi para responder quais pares de algoritmos estão divergindo:

Tabela 3 – Test post-hoc de Nemenyi: p-valores e significância estatística para cada par de algoritmos. Repare que a tabela é simétrica, dado que comparar a_1 com a_2 é o mesmo que comparar a_2 com a_1.

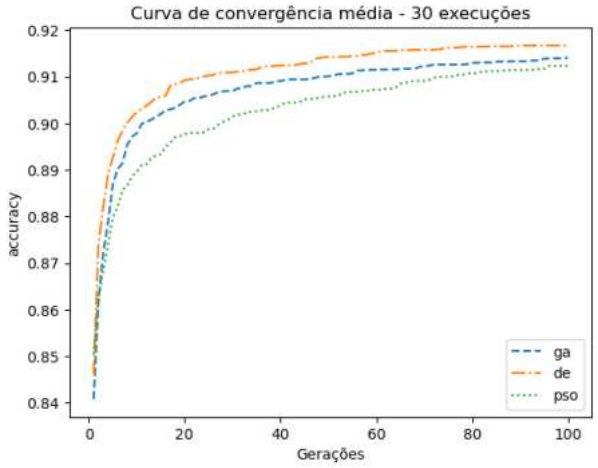
	Optuna	**GA**	**DE**	**PSO**
Optuna	–	$1.28 \cdot 10^{-7}$ (Sim)	$1.19 \cdot 10^{-6}$ (Sim)	$9.31 \cdot 10^{-10}$ (Sim)
GA		–	0.97 (Não)	0.85 (Não)
DE			–	0.62 (Não)
PSO				–

É possível concluir que, para essa tarefa de regressão, o uso dos algoritmos evolutivos para encontrar os melhores hiperparâmetros tem desempenho robusto, nessa métrica, frente ao Optuna: para todos os três casos estudados, existe uma diferença significativa na média dos melhores resultados das 30 execuções. Apesar da performance ser superior em termos de encontro dos hiperparâmetros otimizados dentro do problema proposto, o Optuna vence no quesito de número de avaliações de função. Enquanto esse framework faz exatamente n avaliações, com n sendo o número de iterações, os EAs fazem $n \cdot |P|$, onde $|P|$ é o tamanho da população. Uma forma de contornar essa diferença seria, ao longo da execução, utilizar a técnica de programação dinâmica de memoização, guardando todos os indivíduos h já avaliados em uma estrutura de hashset H e, com o surgimento de um indivíduo $\bar{h}$ com as mesmas componentes que algum $h \in H$, não seria necessário avaliar $f(\bar{h})$.

5.3.2 Classificação

Foram realizadas 30 execuções dos algoritmos GA, DE, PSO e Optuna para encontrar os melhores hiperparâmetros para a tarefa de classificação ao longo de 100 iterações. É feita uma curva de convergência média, mostrando os resultados de cada geração em cada uma das execuções. O objetivo era maximizar a acurácia.

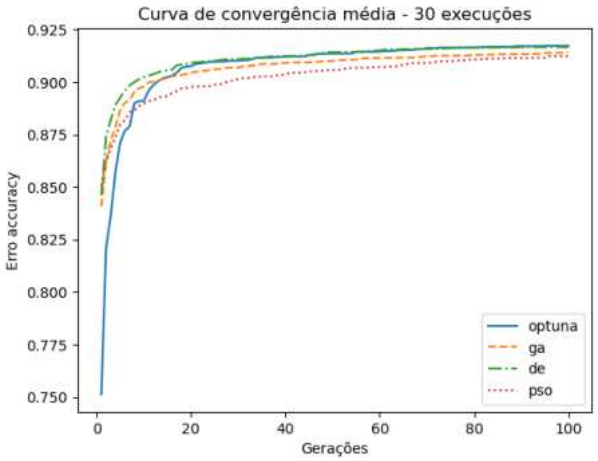
Figura 17 – Resultado da otimização dos hiperparâmetros utilizando GA, DE e PSO



Fonte: o autor

Com a inclusão do Optuna:

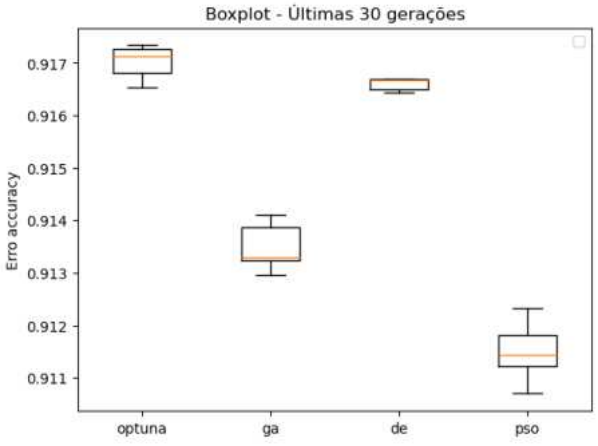
Figura 18 – Resultado da otimização dos hiperparâmetros utilizando GA, DE, PSO e Optuna



Fonte: o autor

Utilizando o boxplot, podemos notar que todos os resultados encontrados são extremamente próximos entre si:

Figura 19 – Boxplot do resultado da otimização dos hiperparâmetros utilizando GA, DE, PSO e Optuna



Fonte: o autor

Por fim, uma tabela numérica com mínimo, máximo, média e desvio padrão dos valores encontrados:

Tabela 4 – Resumo dos resultados obtidos para a acurácia para cada algoritmo

População	**Algoritmo**	**Min**	**Max**	**Mean**	**Std**
10	GA	0.84	0.91	0.90	0.01
	DE	0.84	0.91	0.91	0.009
	PSO	0.85	0.91	0.90	0.01
	Optuna	0.75	0.91	0.90	0.02

Diferentemente da tarefa de regressão, os valores encontrados durante a maximização da acurácia são próximos o suficiente para, a priori, já ser possível intuir que não há diferença estatística entre os algoritmos nesse caso. Ainda que no boxplot haja uma diferença visual, a escala mostra que ela acontece já na terceira casa decimal.

Analogamente à tarefa de regressão, ao fazer o teste de Friedman, utilizando os melhores valores de cada uma das 30 execuções dos algoritmos, encontramos um p-valor< 0.05, com $p = 0.01$. Prosseguimos então para o teste de Nemenyi, obtendo:

Tabela 5 – Test post-hoc de Nemenyi: p-valores e significância estatística para cada par de algoritmos. Repare que a tabela é simétrica, dado que comparar a_1 com a_2 é o mesmo que comparar a_2 com a_1.

	Optuna	**GA**	**DE**	**PSO**
Optuna	–	0.53 (Não)	0.89 (Não)	0.08 (Não)
GA		–	0.16 (Não)	0.74 (Não)
DE			–	0.01 (Sim)
PSO				–

Nesse caso, a diferença se encontra entre os resultados do Differential Evolution e do Particle Swarm Optimization, não havendo diferença estatística entre o uso dos EAs e do Optuna. Portanto, é possível dizer ambas das abordagens, algoritmos evolutivos e Optuna, são competitivas entre si.

6 CONCLUSÃO

Neste trabalho, foi proposto o uso de algoritmos evolutivos para buscar hiperparâmetros em redes neurais artificiais. Em particular, essa abordagem foi utilizada em duas tarefas distintas, de regressão e de classificação. Ambas abordam problemas relacionados à questão energética, preocupação emergente do presente momento do século XXI:

- Regressão: dados atributos relacionados à demanda e ao preço da energia elétrica, inferir o valor do consumo no setor residencial;
- Classificação: dadas informações sobre produtores e consumidores em uma Smart Grid, decidir se ela é ou não estável economicamente.

A escolha das metaheurísticas Genetic Algorithm, Differential Evolution e Particle Swarm Optimization se dá pelo fato se serem algoritmos clássicos da área, que já demonstraram desempenho satisfatório na literatura. Para verificar se há algum ganho no uso desses métodos, foi feita uma comparação com o framework Optuna, consolidado na área de IA e especializado em busca de hiperparâmetros. Os detalhes de implementação dos EAs poderiam interferir no seu desempenho, então foi escolhido o uso da biblioteca Pymoo, desenvolvida para problemas de otimização, que já conta com boas implementações.

Com os experimentos executados, podemos concluir que pode haver situações em que o uso da computação evolutiva pode trazer um melhor resultado ou, pelo menos, uma performance equiparável às técnicas mais consolidadas no estado da arte dessa área. No caso da tarefa de regressão escolhida, todas as metaheurísticas encontraram melhores hiperparâmetros, enquanto que para a classificação, o desempenho foi estatisticamente igual.

Apesar de ser possível utilizar essa abordagem, houve a constatação do problema que envolve o grande número de avaliações de função. Nos casos estudados, em especial, como o espaço de busca é relativamente pequeno, seria possível armazenar um conjunto com todos os indivíduos já avaliados e, quando surgisse um novo indivíduo já nesse conjunto, não seria necessário calcular f novamente. No entanto, em espaços de buscas maiores, essa solução poderia não ser satisfatória, já que o número de pontos distintos no espaço seria maior. Aumentando o número de indivíduos ou o número de gerações, o custo computacional poderia escalar de forma a se tornar um problema, ainda que houvesse uma maior exploração do espaço em relação ao Optuna.

Há alguns caminhos que podem ser explorados em trabalhos futuros:

- Utilização de um conjunto de dados de consumo de energia do Brasil, disponibilizado pela Empresa de Pesquisa Energética (EPE). Para tanto, há a necessidade

de modelar e intepretar os dados disponíveis, ou consultar estudos que tenham realizado essa tarefa, de maneira a relacionar certos atributos para, posteriormente, construir um conjunto de dados viável para o aprendizado supervisionado;

- Utilização de conjuntos de dados maiores e não necessariamente voltados para energia, para obter uma perspectiva do desempenho dos EAs contextos mais generalizados;
- Utilização de outras metaheurísticas, como o Artificial Bee Colony (ABC) e Canonical Differential Evolutionary Particle Swarm Optimization (C-DEEPSO), que podem performar bem para esse problema;
- Implementação em Pytorch, que possibilita um número maior de hiperparâmetros (como mais funções de ativação e otimizadores) e o uso de GPU. Essa implementação pode aumentar significativamente o tamanho do espaço de busca, o que gera uma demanda por uma implementação mais sofisticada dos EAs em termos de custo computacional e em número de avaliações de função;
- Paralelização dos algoritmos evolutivos, visando reduzir o tempo de processamento de cada geração;
- Ajuste fino dos parâmetros dos algoritmos, que também são dependentes dessa decisão a priori.

REFERÊNCIAS

AKIBA, T. et al. Optuna: A next-generation hyperparameter optimization framework. In: **Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. [S.l.: s.n.], 2019.

ALAZAB, M. et al. A multidirectional lstm model for predicting the stability of a smart grid. **IEEE Access**, v. 8, 2020. ISSN 21693536.

ALSIRHANI, A. et al. A novel approach to predicting the stability of the smart grid utilizing mlp-elm technique. **Alexandria Engineering Journal**, v. 74, p. 495–508, 2023. ISSN 1110-0168. Disponível em: https://www.sciencedirect.com/science/article/pii/S1110016823004234.

ARZAMASOV, V. **Electrical grid stability simulated data**. [S.l.]: UCI Machine Learning Repository, 2018.

ARZAMASOV, V.; BOHM, K.; JOCHEM, P. Towards concise models of grid stability. In: **2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids, SmartGridComm 2018**. [S.l.: s.n.], 2018.

CHEN, Y. et al. Semi-supervised and unsupervised deep visual learning: A survey. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 46, 2024. ISSN 19393539.

CUNHA, A. G.; TAKAHASHI, R.; ANTUNES, C. H. **Manual de computação evolutiva e metaheurística**. [S.l.: s.n.], 2012.

EFATINASAB, E. et al. Towards robust stability prediction in smart grids: Gan-based approach under data constraints and adversarial challenges. **Internet of Things**, v. 33, p. 101662, 2025. ISSN 2542-6605. Disponível em: https://www.sciencedirect.com/science/article/pii/S2542660525001763.

EIA. **U.S. Energy Information Administration - EIA - Independent Statistics and Analysis — eia.gov**. 2025. https://www.eia.gov/about/mission_overview.php). [Acessado em 10/08/2025].

EIA-MONTHLY. **Total Energy Monthly Data - U.S. Energy Information Administration (EIA) — eia.gov**. 2025. https://www.eia.gov/totalenergy/data/monthly/. [Acessado em 10/08/2025].

FARHANGI, H. The path of the smart grid. **IEEE Power and Energy Magazine**, v. 8, 2010. ISSN 15407977.

FLA-SHOP. **United States Region Maps — fla-shop.com**. 2022. https://www.fla-shop.com/resources/us-regions/. [Acessado em 10/08/2025].

FOUNDATION, G. E. et al. Energy atlas 2018. **Green European Foundation European Renewable Energies Federation Friends of the Earth Europe Heinrich Böll**, 2018.

FRANCESCHI, L. et al. Hyperparameter optimization in machine learning. out. 2024.

IBRAR, M. et al. A machine learning-based model for stability prediction of decentralized power grid linked with renewable energy resources. **Wireless Communications and Mobile Computing**, Hindawi Limited, v. 2022, 2022. ISSN 15308677.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: **IEEE International Conference on Neural Networks - Conference Proceedings**. [S.l.: s.n.], 1995. v. 4. ISSN 2334-4563.

KULIN, M. et al. A survey on machine learning-based performance improvement of wireless networks: Phy, mac and network layer. **Electronics (Switzerland)**, MDPI AG, v. 10, p. 1–64, 2 2021. ISSN 20799292.

LAMBORA, A.; GUPTA, K.; CHOPRA, K. Genetic algorithm- a literature review. In: **Proceedings of the International Conference on Machine Learning, Big Data, Cloud and Parallel Computing: Trends, Prespectives and Prospects, COMITCon 2019**. [S.l.: s.n.], 2019.

LECUN, Y.; BENGIO, Y.; HINTON, G. **Deep learning**. 2015.

LIANG, X. **Ascend AI Processor Architecture and Programming: Principles and Applications of CANN**. [S.l.: s.n.], 2020.

MASSAOUDI, M. et al. Accurate smart-grid stability forecasting based on deep learning: Point and interval estimation method. In: **2021 IEEE Kansas Power and Energy Conference, KPEC 2021**. [S.l.: s.n.], 2021.

MITCHELL, T. M. **Machine Learning**. [S.l.: s.n.], 1997.

NOCEDAL, J.; WRIGHT, S. J. **Numerical optimization**. [S.l.]: Springer, 2006. ISSN 21971773.

NURMAMATOV, M.; SARIYEV, S.; ESHONKULOV, B. Application of evolutionary algorithms to enhance the efficiency of neural networks and machine learning algorithms. In: **2025 International Russian Smart Industry Conference (SmartIndustryCon)**. [S.l.]: IEEE, 2025. p. 533–537.

ONU. **Objetivos de Desenvolvimento Sustentável — unicef.org**. 2025. https://www.unicef.org/brazil/objetivos-de-desenvolvimento-sustentavel. [Acessado em 09/08/2025].

ONU-ODS-7. **Sustainable Development Goal 7: Energia limpa e acessível — brasil.un.org**. 2025. https://brasil.un.org/pt-br/sdgs/7. [Acessado em 09/08/2025].

OpenAI et al. Dota 2 with Large Scale Deep Reinforcement Learning. **arXiv e-prints**, p. arXiv:1912.06680, dez. 2019.

POWELL, J. et al. Smart grids: A comprehensive survey of challenges, industry applications, and future trends. **Energy Reports**, Elsevier BV, v. 11, p. 5760–5785, jun. 2024. ISSN 2352-4847. Disponível em: http://dx.doi.org/10.1016/j.egyr.2024.05.051.

ROUMELIOTIS, K. I.; TSELIKAS, N. D. **ChatGPT and Open-AI Models: A Preliminary Review**. 2023.

SATU, M.; KHAN, M. I. Machine learning approaches to predict the stability of smart grid. 01 2024.

SCHäFER, B. et al. Taming instabilities in power grid networks by decentralized control. **European Physical Journal: Special Topics**, v. 225, 2016. ISSN 19516401.

SCHäFER, B. et al. Decentral smart grid control. **New Journal of Physics**, v. 17, 2015. ISSN 13672630.

SILVER, D. et al. Mastering the game of go with deep neural networks and tree search. **Nature**, v. 529, 2016. ISSN 14764687.

STENSJö, I. P.; FERREIRA, C. C.; LOURA, R. M. Classificação e agrupamento das cidades brasileiras em graus-dia de aquecimento e resfriamento: 1960 a 2013. **urbe. Revista Brasileira de Gestão Urbana**, FapUNIFESP (SciELO), v. 9, p. 286–300, 10 2017.

STORN, R.; PRICE, K. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. **Journal of Global Optimization**, v. 11, 1997. ISSN 09255001.

TURNER, S.; ULUDAG, S. Towards smart cities: Interaction and synergy of the smart grid and intelligent transportation systems. In: _____. [S.l.: s.n.], 2015.

VIKHAR, P. A. Evolutionary algorithms: A critical review and its future prospects. In: **Proceedings - International Conference on Global Trends in Signal Processing, Information Computing and Communication, ICGTSPICC 2016**. [S.l.: s.n.], 2017.

VINYALS, O. et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. **Nature**, v. 575, 2019. ISSN 14764687.

WANG, G. et al. Application and evaluation of the evolutionary algorithms combined with conventional neural network to determine the building energy consumption of the residential sector. **Energy (Oxf.)**, Elsevier BV, v. 298, n. 131312, p. 131312, jul. 2024.

YERROLLA., C.; RAMESH, D. Electrical grid stability prediction using machine learning algorithms. In: **2023 4th International Conference on Electronics and Sustainable Communication Systems (ICESC)**. [S.l.]: IEEE, 2023.

ZAIDI, S. S. A. et al. **A survey of modern deep learning based object detection models**. 2022.