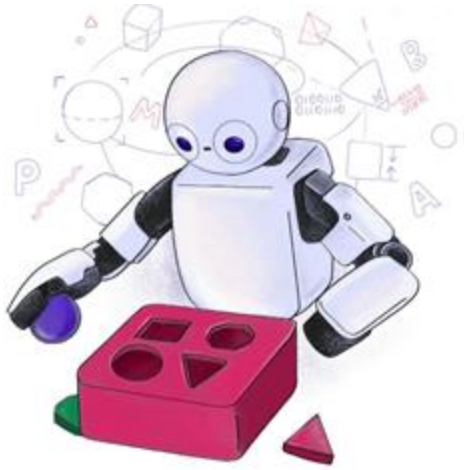
TP558 - Tópicos avançados em Machine Learning: ***Kolmogorov-Arnold Networks (KANs)***



Inatel

Paulo Otavio Luczensky de Souza
paulo.souza@mtel.inatel.br

Introdução

- Proposto pelos pesquisadores Ziming Liu, Yixuan Wang, Max Tegmark (2023).

- Uma alternativa às tradicionais Redes Neurais Artificiais, usando o teorema de representação de Kolmogorov-Arnold (1957).

- Ganhou destaque por unir **teoria matemática** e **aprendizado profundo**.

Inatel

Introdução

As KANs podem ser aplicadas em diversos cenários:

- **Finanças e economia**
  - Previsão de séries temporais e análise de risco com foco em interpretabilidade.
- **Engenharia e controle**
  - Identificação de sistemas, predição de falhas, controle adaptativo e otimização de processos.
- **Agricultura de Precisão**
  - **Otimização do uso da água:** aprender relações não lineares entre clima, solo e produtividade para recomendar volumes ideais de irrigação.

Inatel

Teorema de Kolmogorov–Arnold

Inatel

Fundamentação teórica

- Suponha uma **função muito complexa** com várias variáveis, por exemplo:

$$f(x_1, x_2, x_3, ..., x_n)$$

- Essa função pode representar **qualquer fenômeno real**. Por exemplo, o rendimento de uma lavoura em função da chuva, do solo, da luz, da temperatura etc.

- O desafio é: Como representar ou aproximar esta função complexa por uma função mais simples?

Inatel

Fundamentação teórica

- Segundo Kolmogrov e Arnold:
  - "Qualquer função contínua de várias variáveis pode ser construída a partir de funções de uma única variável e somas dessas funções."
- Desta forma, uma função complexa, como:

$$f(x_1, x_2, x_3)$$

- Poderia ser reescrita como uma **combinação de funções univariadas** (que dependem de só um número por vez), por exemplo:

$$f(x_1, x_2, x_3) = \sum_{i=1}^{m} g_i \left(\sum_{j=1}^{n} \phi_{ij}(x_j) \right)$$

- ϕ_{ij} são **funções de uma variável**,
- g_i também são **funções de uma variável**,
- e a soma combina tudo.

Inatel

Fundamentação teórica - Exemplo

- Realizar a previsão de crescimento de uma planta com base em:
  - x1 = chuva (mm)
  - x2 = temperatura (°C)
  - x3 = radiação solar (MJ/m²)

- Em vez de criarmos uma função complexa como: $f(x_1, x_2, x_3)$

- A fórmula assumiria o seguinte formato:

$$f(x_1, x_2, x_3) = g_1(\phi_{11}(x_1) + \phi_{12}(x_2) + \phi_{13}(x_3)) + g_2(...)$$

Inatel

Fundamentação teórica - Exemplo

- A função poderia ser aproximada como:

$$f(x_1, x_2, x_3) = g_1(\phi_{11}(x_1) + \phi_{12}(x_2) + \phi_{13}(x_3))$$

- Agora, adiciona-se funções simples com números:

Função	Fórmula
$\phi_{11}(x_1)$	0.5 * x1
$\phi_{12}(x_2)$	0.2 * x2
$\phi_{13}(x_3)$	0.1 * x3
$g_1(y)$	2 * y

Inatel

Fundamentação teórica - Exemplo

- Suponha que:
  - x1 = 10 mm
  - x2 = 25 °C
  - x3 = 15 MJ/m²
- 1º) Calcular a soma dentro de g1:

$$\text{soma}_1 = \phi_{11}(10) + \phi_{12}(25) + \phi_{13}(15) = (0.5 * 10) + (0.2 * 25) + (0.1 * 15) = 5 + 5 + 1.5 = 11.5$$

- 2º) Aplicar g1: $g_1(\text{soma}_1) = 2 * 11.5 = 23$

- Resultado: 23 representa a **pontuação de crescimento**, que poderia ser um **índice de biomassa** ou **quantidade de gramas de produção.**

Inatel

Benefícios do Teorema

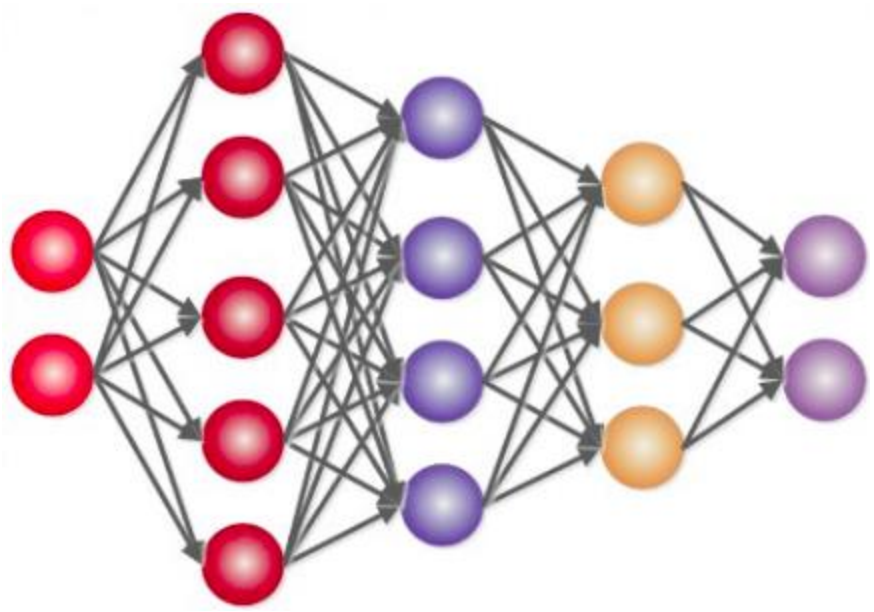
Aspecto	Sem o Teorema	Com o Teorema de Kolmogorov-Arnold
Modelagem	Depende de funções altamente complexas ou específicas	Qualquer função contínua pode ser representada como soma de funções unidimensionais
Interpretação	Difícil entender a contribuição de cada variável	Cada variável atua por meio de funções simples
Generalização	Pode superajustar os dados (overfitting)	Estrutura mais compacta e interpretável
Aplicação em ML	Exige redes profundas e não lineares fixas (MLPs)	Permite arquiteturas mais adaptativas, como as KANs

Inatel

Redes Neurais Multi-Camadas

Inatel

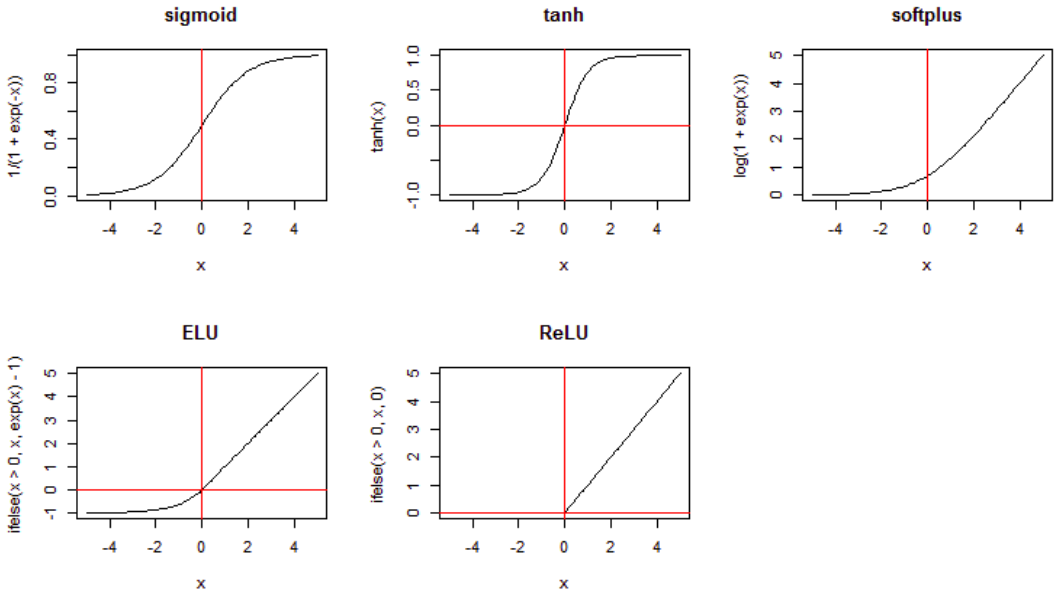
Fundamentação teórica - Multi Layer Perceptron



- **Estrutura em camadas**
  - Compostas por uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída, onde cada neurônio da camada atual se conecta a todos da camada seguinte.
- **Aprendizado supervisionado**
  - Ajustam os pesos das conexões com base no erro entre a saída prevista e a real, usando algoritmos como *backpropagation*.
- **Funções de ativação não lineares**
  - Aplicadas em cada neurônio (ReLU, Sigmoid, Tanh) para permitir que a rede aprenda relações complexas entre as variáveis.
- **Generalização de padrões**
  - Capazes de aprender e representar relações complexas nos dados, permitindo prever resultados para entradas nunca vistas durante o treinamento.

Inatel

Fundamentação teórica - Funções de Ativação



- **Filtro Inteligente**
  - Controlam quanto a saída de um neurônio influencia a próxima camada.
- **Entrada transformada**
  - Recebem a soma ponderada ($z=\sum wi * xi + b$) e produzem f(z)
    - wi → peso que indica a importância relativa
    - b → bias, permite deslocar a função para melhor se adequar aos dados.
- **Não linearidade**
  - Permitem que a rede aprenda **padrões complexos**, como relações **não proporcionais** ou **interações entre variáveis**
- **Exemplos**
  - ***Sigmoid*:** saída entre 0 e 1 → útil para classificação binária.
  - ***Tanh*:** saída entre -1 e 1 → acelera a convergência.
  - ***ReLU*:** mantém valores positivos e zera negativos → eficiente em redes profundas.

Inatel

Limitações das Funções de Ativação em MLPs

- **Saturação e gradientes próximos de zero**
  - Funções como **sigmoid** e **tanh** param de variar muito quando a entrada é muito alta ou muito baixa. Nessas regiões, suas derivadas ficam quase zero, fazendo com que o **gradiente "desapareça"** — ou seja, os neurônios deixam de aprender durante o treinamento.

- **Unidades "mortas" (dead units)**
  - No caso da ***ReLU***, neurônios podem "morrer" se a entrada for sempre negativa e a derivada for zero, então, esse neurônio deixa de aprender.

- **Dependência de inicialização e profundidade da rede**
  - Quanto mais camadas, mais a propagação do gradiente sofre, podendo impedir que camadas iniciais aprendam de forma eficaz.

- **Não-linearidade fixa e estrutura rígida**
  - Nas MLPs, cada neurônio aplica sempre a mesma função de ativação (como ReLU), independentemente dos dados. Essa não-linearidade fixa limita a capacidade da rede de se adaptar, motivando o surgimento de arquiteturas mais flexíveis, como as KANs.

Inatel

Redes Neurais Kolmogorov–Arnold

Inatel

Fundamentação teórica - Motivações para as KANs

- **Limitações das MLPs**
  - Nas redes MLP tradicionais, as funções de ativação são fixas nos nós, o que pode impedir a rede de capturar relações complexas entre variáveis contínuas.

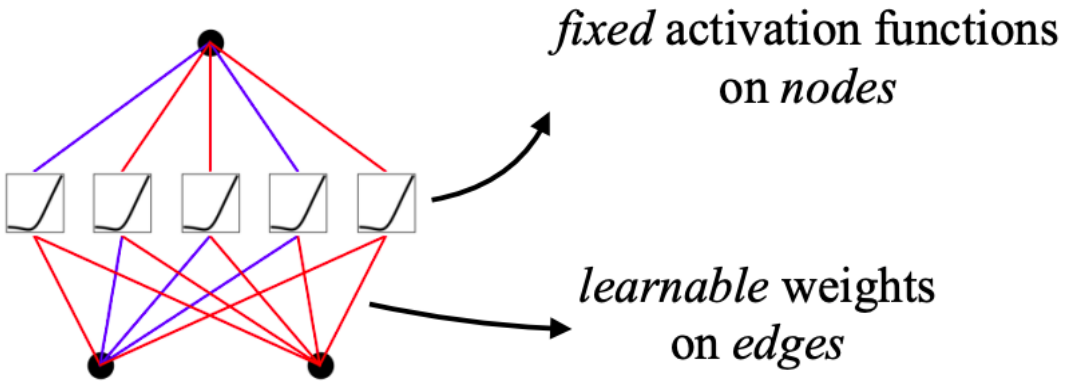
- **Pesos lineares restringem expressividade**
  - Os pesos lineares ajustam apenas a intensidade das conexões, sem permitir modelar transformações complexas de maneira eficiente.

- **Necessidade de maior interpretabilidade**
  - Pesquisadores buscam redes que permitam entender como cada variável contribui para a saída, algo que MLPs não fornecem de forma clara.

- **Inspiração no Teorema de Kolmogorov-Arnold**
  - O teorema mostra que qualquer função contínua multivariada pode ser representada como composição de funções univariadas, sugerindo uma estrutura mais poderosa para redes neurais.

Inatel

Arquitetura das MLPs

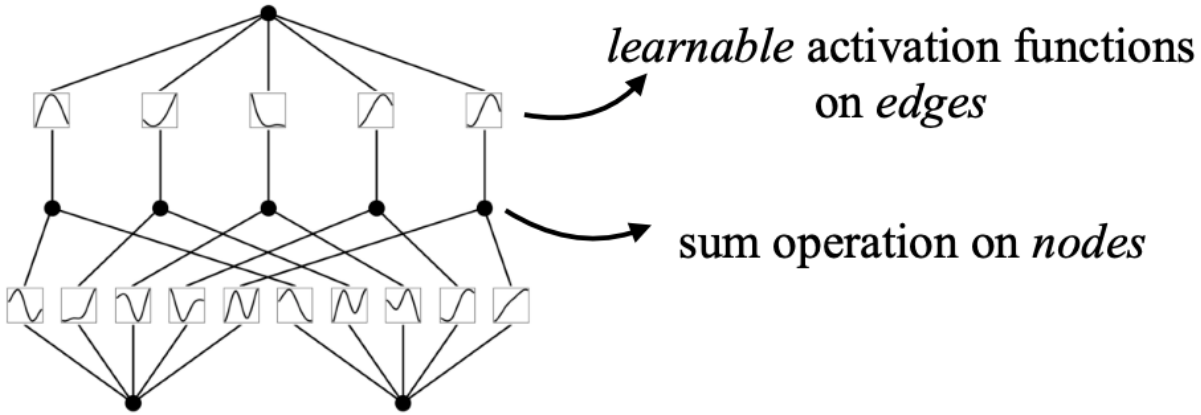


- **Nós (*nodes*):** Funções de ativação **fixas** aplicadas nos nós.
- **Conexões (*edges*):** Pesos **aprendidos** durante o treinamento, ajustando a força de cada conexão.

Inatel

- **Conectividade Total (Densa)**
  - Cada neurônio recebe entradas de todos os neurônios da camada anterior (**totalmente conectado**).
- **Funções de Ativação Fixas (Nós/Nodes)**
  - As funções de ativação (e.g., *ReLU*, *Sigmoid*) são **fixas** e definem como a soma ponderada das entradas será transformada.
- **Pesos Aprendíveis (Conexões/Edges)**
  - As conexões possuem **pesos lineares aprendíveis** que são ajustados via *backpropagation*.
- **Mecanismo de Aprendizado**
  - O ajuste dos pesos permite que a rede aprenda padrões complexos nos dados, ajustando a **força** de cada conexão.

Arquitetura das KANs



• **Nós (nodes)**: Representam somatórios de sinais de entrada. Cada nó realiza a **operação de soma** dos valores que recebe.

• **Conexões(edges)**: Cada conexão aplica **funções de ativação aprendíveis**, permitindo que a rede capture transformações complexas das entradas.

Funções de Ativação nos Pesos (Conexões/Edges)

- Cada conexão possui uma **função univariada aprendível**, geralmente modelada por **splines**.
- Essas funções ajustam **como cada variável é transformada**, substituindo os pesos fixos das MLPs por **relações não lineares adaptativas**.

Composição de Funções (Nós/Nodes)

- Inspirada no **Teorema de Kolmogorov-Arnold**, cada nó combina as saídas das funções de conexão por **somas**.
- Isso permite representar **qualquer função contínua**, decompondo o problema em transformações simples.

Estrutura Flexível

- Como não depende de pesos lineares fixos, a rede pode **modelar formas mais complexas** com **menos parâmetros**.
- Essa flexibilidade melhora o **ajuste aos dados** e a **eficiência de aprendizado**.

Interpretabilidade e Interatividade

- As funções aprendidas em cada conexão podem ser **visualizadas e ajustadas manualmente**, tornando o modelo **mais transparente** e permitindo **análises explicativas** do comportamento da rede.

Inatel

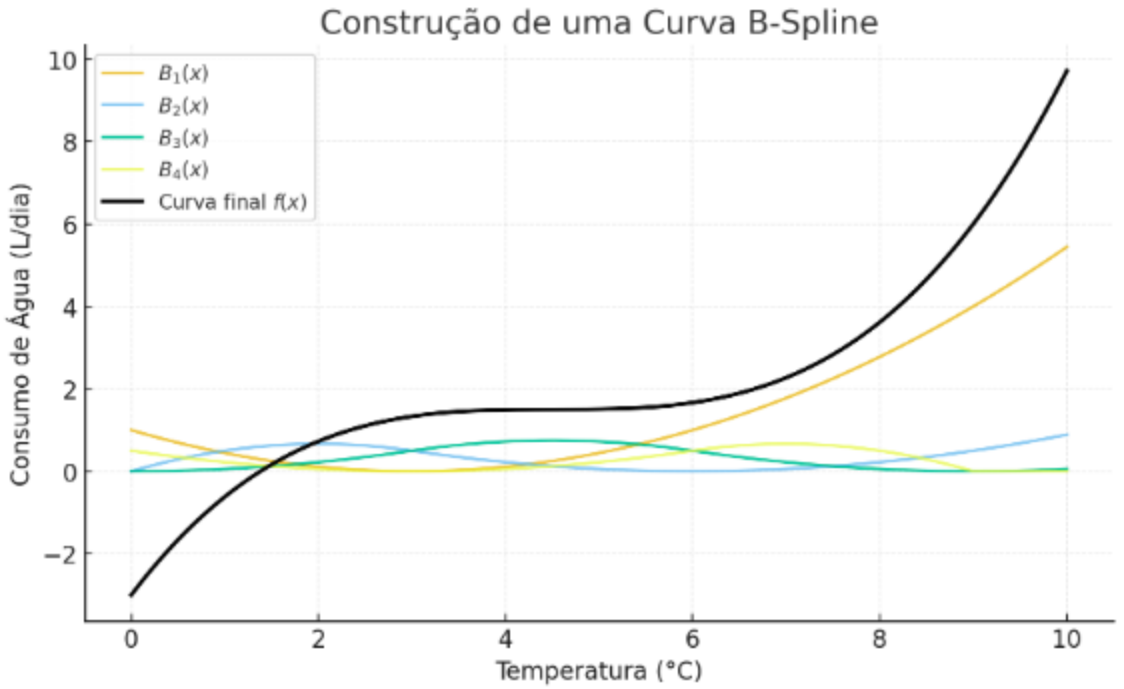
1) Funções de Ativação nos Pesos (Conexões/Edges)

Característica	MLP	KAN
Representação do Edge (Peso)	Um **número único e fixo** *w*. O mesmo valor sempre multiplica a entrada.	Uma **função univariada aprendível** $\phi(x_i)$ que transforma a entrada de forma adaptativa.
Cálculo da contribuição	**Linear:** $w \cdot x_i$. Cada peso atua como multiplicador constante.	**Não-linear:** $\phi(x_i)$. Cada conexão pode curvar a entrada de forma personalizada.
Onde está a não-linearidade?	Apenas no **nó** (neurônio), via função de ativação fixa (σ).	**Na própria conexão (edge)**, através de uma função **flexível e aprendida**, permitindo ajustes finos antes de chegar ao nó.
Expressão matemática da saída do nó	$\sigma(\sum w_i * x_i + b)$ — combinação linear das entradas seguida de ativação.	• $\sum \phi_i(x_i)$ — soma das saídas das funções não-lineares das conexões; • O nó agrega essas contribuições.
Analogia didática	**"Régua Reta"**: aplica sempre a mesma inclinação ao sinal de entrada.	**"Régua Flexível"**: curva a entrada de forma ideal para cada situação, modelando relações complexas.
Mecanismo de aprendizado	Ajuste **global** do peso único *w* usando backpropagation.	Ajuste **local e fino** de cada função $\phi(x)$, geralmente via **splines B**, permitindo maior flexibilidade e interpretação.
Vantagem principal	Simples e rápido, fácil de implementar.	Permite modelar **relações não-lineares complexas** com menos camadas e mais interpretabilidade.

Inatel

B-Spline

- Um **B-spline (Basis Spline)** é uma função suave composta pela soma de várias **funções base**.
- Cada base é ativada apenas em uma parte do domínio, o que dá **flexibilidade local**.
- Em vez de pesos fixos como nas MLPs, as KANs aprendem os **coeficientes** dessas curvas.
- Assim, o modelo se adapta melhor a padrões complexos e suaves (como variações de irrigação ou temperatura).

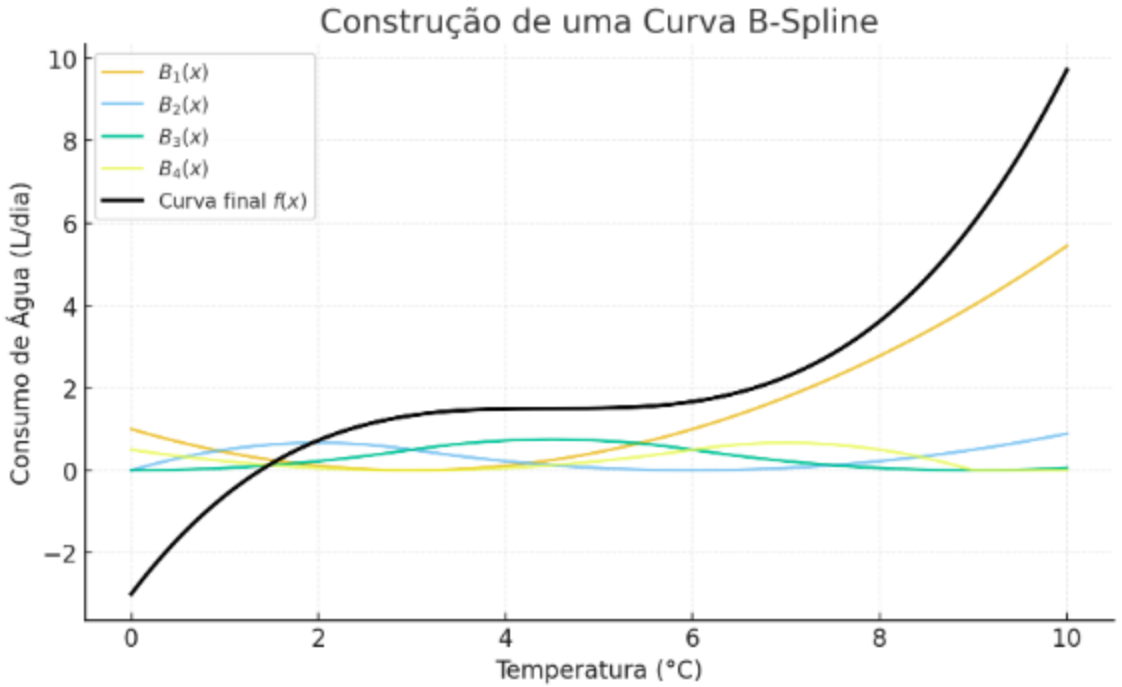


B-Spline - Exemplo

• Suponha que queremos representar a relação entre temperatura e consumo de água:

$$f(x) = \sum_{i=1}^{k} c_i B_i(x)$$

- $B_i(x)$ = funções base do B-spline
- c_i = coeficientes aprendidos (ajustam a forma da curva)



A figura mostra 4 funções base B1(x),B2(x),B3(x),B4(x).

Cada uma atua em uma faixa do eixo x.

A **curva preta final** é a soma ponderada dessas funções representando, por exemplo, a relação entre **temperatura e consumo de água**.

Como as KANs escolhem as funções univariadas

Etapas do processo

1. Inicialização suave

- Cada função univariada f(x) é iniciada de forma **quase linear**, geralmente com f(x) ≈ x.
- Isso garante que, no começo, a KAN **se comporte como uma MLP tradicional**, evitando instabilidades.
- É uma forma de "partida segura": a rede só se torna não-linear quando há evidência nos dados para isso.

2. Aprendizado adaptativo

- Durante o treinamento, **os parâmetros das splines** (pontos de controle) são atualizados pelo **gradiente descendente**.
- Cada spline funciona como uma **pequena curva flexível** que pode se ajustar localmente, permitindo que a função mude de forma conforme o erro de predição.
- Isso faz com que **cada conexão "descubra" a melhor forma** de transformar sua entrada — linear, convexa, côncava, saturante, etc.

3. Seleção natural das formas

- Ao longo do treinamento, as funções que **mais contribuem para reduzir o erro** acabam sendo **refinadas e preservadas**.
- Outras permanecem quase lineares — o que mostra que a rede **aprende onde precisa ser complexa** e **onde pode ser simples**.
- Assim, surge uma **distribuição adaptativa de não-linearidades**, onde apenas partes do modelo se tornam altamente não-lineares.

Inatel

2) Composição de Funções (Nós)

Modelo	Base Matemática / Teorema	Implementação do Nó	Divisão de Tarefas / Estrutura	Exemplo Matemático	Resultados / Benefícios
MLP	• Não segue diretamente o Teorema K-A; • Cada neurônio aplica soma ponderada e função de ativação fixa.	Cada nó realiza: $y = \sigma\left(\sum_i w_i x_i + b\right)$	• Complexidade centralizada no nó; • Cada neurônio precisa capturar não-linearidades da função.	Soma ponderada + ativação não-linear (sigmoid, ReLU, tanh).	• Difícil interpretação; • Precisa de redes largas e profundas para funções complexas.
KAN	• Parametriza diretamente o Teorema K-A: • Qualquer função multivariada contínua pode ser composta por funções univariadas somadas.	Cada nó realiza apenas: $y = \sum_i \phi_i(x_i)$	• Complexidade absorvida pelos edges; • O nó é um agregador simples, tornando a rede mais clara e modular.	• Funções ϕi(xi) aplicadas nos edges; • Soma simples no nó.	• Estrutura simplificada; • Interpretável; • Captura relações complexas com menos parâmetros; • Facilita análise científica.

Inatel

3) Estrutura Flexível

Modelo	Flexibilidade / Estrutura	Adaptabilidade	Interpretação Matemática	Resultados
MLP (Rígida)	• Necessita redes muito largas e profundas (milhões ou bilhões de parâmetros) para modelar funções complexas. • Transformação não-linear fixa em cada nó.	Limitada; toda a rede precisa aprender mudanças complexas.	• Difícil interpretar; • Os pesos não têm significado direto.	• Alta complexidade; • Difícil explicação; • Menos eficiente para capturar relações complexas com poucos dados.
KAN (Flexível)	• Pesos fixos substituídos por funções flexíveis (splines), permitindo aproximação de funções complexas com redes mais rasas.	Adaptação local: cada spline ajusta-se às variações dos dados sem alterar toda a estrutura.	• Interpretação matemática clara; • Cada função $\phi(x)$ é visualizável e tem significado explícito.	• Capta mudanças sutis; • Promove transições suaves, eficiente em parâmetros, • Fácil explicação e validação científica.

Inatel

4) Interpretabilidade e Interatividade

Modelo	Interpretabilidade	Interatividade	Exemplos	Resultados
MLP	• Extremamente difícil; • Impossível entender a contribuição de cada feature individual olhando apenas para pesos.	Limitada: • Pouca ou nenhuma colaboração humana possível.	Pesos e biases em uma rede profunda.	• Difícil explicar decisões do modelo. • Pouco útil para descoberta científica.
KAN	• Direta: cada função univariada $\phi(x)$ pode ser visualizada; • Funções simples indicam relações simples; • Funções curvas indicam relações complexas.	Alta, especialistas podem: • Simplificar funções complexas • Incorporar conhecimento prévio ajustando $\phi(x)$ • Colaborar para interpretação científica	Funções lineares, quadráticas ou senoidais $\phi(x)$ para cada feature.	• Permite descobrir relações científicas interpretáveis nos dados. • Facilita explicação e validação por especialistas.

Inatel

Exemplo de Cálculo

- Um sistema IoT coleta medições climáticas:
  - Temperatura (T) = 30 °C
  - Umidade relativa (RH) = 60%
  - Radiação solar (R_s) = 800 W/m²
- O modelo deve prever **ET_o (mm/h),** a perda de água pela lavoura.

- 1º) MLP (pesos fixos + ReLU)
- **Parâmetros escolhidos**
- Camada oculta:
  - neurônio 1: $w11=0.3,\ w12=-0.1,\ w13=0.002,\ b1=-2$
  - neurônio 2: $w21=0.1,\ w22=0.05,\ w23=0.001,\ b2=0.5$
- Saída: w31=0.6, w32=0.4, b3=0.2
- Ativação oculta: ReLU(z) = max(0,z)

Inatel

Exemplo de Cálculo

- **Cálculo**

$$
\begin{aligned}
z_{h1} &= 0.3 \cdot 30 + (-0.1) \cdot 60 + 0.002 \cdot 800 - 2 \\
&= 9 - 6 + 1.6 - 2 = 2.6 \\
h_1 &= \mathrm{ReLU}(2.6) = 2.6
\end{aligned}
$$

$$
\begin{aligned}
z_{h2} &= 0.1 \cdot 30 + 0.05 \cdot 60 + 0.001 \cdot 800 + 0.5 \\
&= 3 + 3 + 0.8 + 0.5 = 7.3 \\
h_2 &= \mathrm{ReLU}(7.3) = 7.3
\end{aligned}
$$

$$
\begin{aligned}
\hat{y}_{\mathrm{MLP}} &= 0.6 \cdot h_1 + 0.4 \cdot h_2 + 0.2 \\
&= 0.6 \cdot 2.6 + 0.4 \cdot 7.3 + 0.2 = 1.56 + 2.92 + 0.2 = \mathbf{4.68}
\end{aligned}
$$

- **Resposta: 4.68** (valor estimado de ET_o numa escala do modelo)

Inatel

Exemplo de Cálculo

- 2º) KAN (funções aprendíveis por conexão)
- Cada conexão aplica uma função ϕ reaprendida; somam-se as ϕs por neurônio (camada oculta) e depois por saída.

- Para neurônio oculto 1:
  - $\phi_{11}(T) = 1.2\sin(0.05T)$
  - $\phi_{12}(RH) = 0.01\,RH$
  - $\phi_{13}(R_s) = 0.0005\,R_s$
  - $b_1 = 0.1$

- Para neurônio oculto 2:
  - $\phi_{21}(T) = 0.05\,T$
  - $\phi_{22}(RH) = 0.0005\,RH^2$
  - $\phi_{23}(R_s) = 0.0008\sqrt{R_s}$
  - $b_2 = -0.2$

- Saída (funções sobre h_1, h_2):
  - $\phi_{31}(h_1) = 0.5\,h_1$
  - $\phi_{32}(h_2) = 0.9\ln(1 + h_2)$
  - $b_3 = 0.3$

OBS: Essas funções **não vieram de um treinamento real**; foram escolhidas manualmente **apenas para mostrar que o KAN consegue modelar relações não-lineares** entre entrada e saída.

Exemplo de Cálculo

- **Cálculo**

$$\phi_{11}(30) = 1.2\sin(0.05 \cdot 30) = 1.2\sin(1.5) \approx 1.2 \cdot 0.997 = 1.196$$
$$\phi_{12}(60) = 0.01 \cdot 60 = 0.60$$
$$\phi_{13}(800) = 0.0005 \cdot 800 = 0.40$$
$$h_1 = 1.196 + 0.60 + 0.40 + 0.1 = \mathbf{2.296}$$

$$\phi_{21}(30) = 0.05 \cdot 30 = 1.5$$
$$\phi_{22}(60) = 0.0005 \cdot 60^2 = 0.0005 \cdot 3600 = 1.8$$
$$\phi_{23}(800) = 0.0008\sqrt{800} \approx 0.0008 \cdot 28.284 = 0.0226$$
$$h_2 = 1.5 + 1.8 + 0.0226 - 0.2 = \mathbf{3.1226}$$

$$\phi_{31}(h_1) = 0.5 \cdot 2.296 = 1.148$$
$$\phi_{32}(h_2) = 0.9\ln(1 + 3.1226) = 0.9\ln(4.1226) \approx 0.9 \cdot 1.416 = 1.2744$$
$$\hat{y}_{\text{KAN}} = 1.148 + 1.2744 + 0.3 = \mathbf{2.7224}$$

- **Resposta: 2.72** (valor estimado de ET_o numa escala do modelo)

Inatel

Exemplo de Cálculo

- **Conclusão**

Modelo	Camada oculta	Cálculo (exemplo)	Saída	Observação didática
MLP	2 neurônios	$h_1 = \text{ReLU}(0.5 * 1.0 + 0.3 * 0.8) = 0.74$ $h_2 = \text{ReLU}(0.7 * 1.0 + 0.2 * 0.8) = 0.86$ $y = 1.2 * 0.74 + 2.0 * 0.86 = 4.68$	4.68	• Combina entradas de forma linear + ReLU → saída alta; • Não considera limite físico.
KAN	2 neurônios	$h_1 = \phi_{11}(T) + \phi_{21}(RH) = 1.2 * 0.6 + 0.0005 * 80^2 = 1.92$ $h_2 = \phi_{12}(T) + \phi_{22}(RH) = 0.9 * 0.7 + 0.0005 * 80^2 = 1.81$ $y = \phi_{31}(h_1) + \phi_{32}(h_2) = 0.9\ln(1 + 1.92) + 0.9\ln(1 + 1.81) = 2.72$	2.72	• Cada entrada tem função própria → captura saturação, limite físico; • Saída mais realista.

Inatel

Evolução das Redes KANs

Ano	Trabalho / Versão	Principais Avanços	Limitações / Observações
2023	KAN (Liu et al.)	Introdução da arquitetura com funções adaptativas (splines) no lugar de pesos fixos. Alta interpretabilidade e capacidade de extrapolação.	Alto custo computacional e instabilidade no treinamento.
2024	FastKAN	Versão otimizada para maior eficiência em GPU, reduzindo drasticamente o tempo de treinamento.	Ainda sensível à escolha de hiperparâmetros (nós, regularização).
2024	EfficientKAN / TinyKAN	Estruturas mais leves e modulares, adaptadas a tarefas tabulares e de visão.	Pequena perda de precisão em relação à KAN completa.
2025	HybridKAN (KAN + Transformers)	Combina KANs com mecanismos de **atenção** (*que permitem ao modelo focar nas partes mais relevantes da entrada, aprendendo dependências contextuais*) e **embeddings**, aprimorando o aprendizado contextual.	Aumenta a complexidade e reduz a interpretabilidade.
2025	PyKAN	Implementação estável em PyTorch, facilitando uso em pipelines de ML.	Ainda em fase experimental; requer ajuste fino para evitar explosão de gradientes.

Inatel

Exemplo de Aplicação

Inatel

Exemplo de Aplicação

- Prever a evapotranspiração (ET0) horária em áreas agrícolas.
- Comparar desempenho e interpretabilidade entre **MLP** e **KAN**.

Inatel

Exemplo de Aplicação

Dataset Utilizado

Base de dados: ETSojaPerHourDataset

Características:

- Total de registros: 8760
- Divisão dos dados:
    - 6132 para treinamento
    - 1314 para validação
    - 1314 para teste
- Tipo de dado: Variáveis meteorológicas e temporais para predição de ET0

Principais variáveis:

- `lat`, `lon`, `temp`, `rh`, `wind_spd`, `solar_rad`, `ET0_KC`
- Além de outras como: `ts`, `timestamp_local`, `dewpt`, `clouds`, `precip`, `uv`

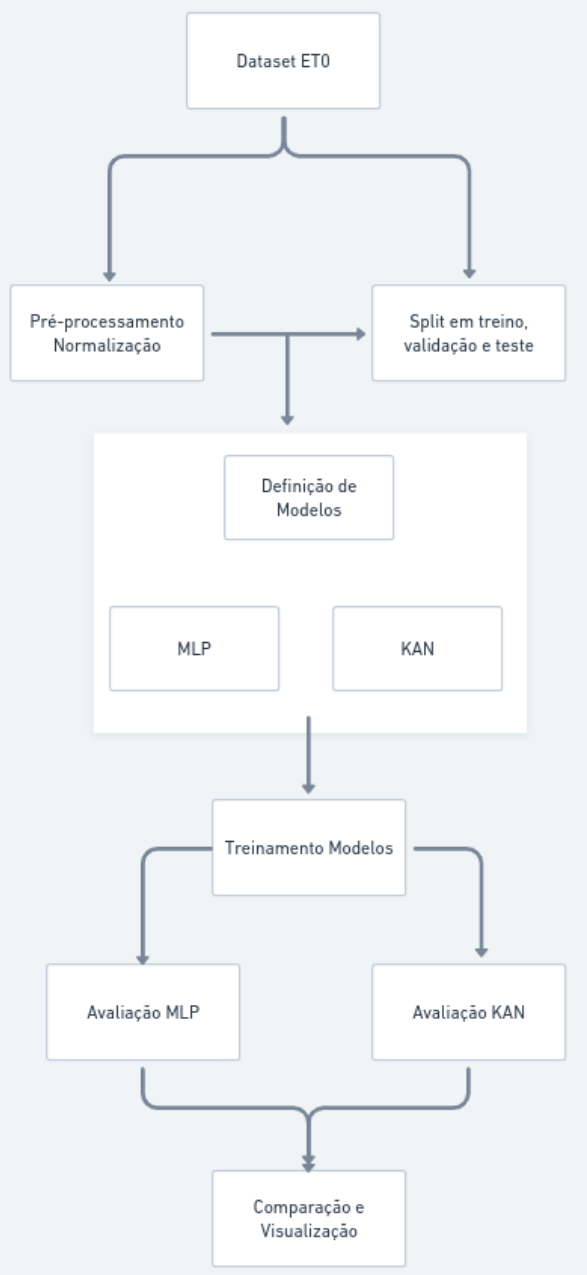
Inatel

Configurações de Treinamento e Avaliação

Parâmetro	MLP	KAN	Descrição
Tipo de modelo	Rede Neural Feedforward	Kolmogorov-Arnold Network	Arquitetura utilizada
Estrutura/camadas	32 → 16 → 8 → 4 → 2 → 1	32 → 16 → 8 → 4 → 2 → 1	Número de neurônios por camada
Função de ativação	ReLU		Transformação não-linear aplicada
Otimizador	Adam	Adam	Algoritmo de atualização de pesos
Taxa de aprendizado (lr)	0.01	0.01	Velocidade de ajuste dos pesos
Épocas máximas	100	100	Número máximo de iterações de treinamento
Critério de parada	Early stopping (patience=30)	Early stopping (patience=30)	Condição para interromper o treino
Função de perda	MSE	MSE	Métrica usada para calcular erro

Inatel

Arquitetura e funcionamento



- **Dataset ET0:**
  - Dados ETSojaPerHour.csv com medições horárias de variáveis ambientais e de solo.

- **Pré-processamento:**
  - Normalização das features e target (StandardScaler) para facilitar o treinamento.

- **Divisão dos dados:**
  - Treino (70%), Validação (15%), Teste (15%) para ajuste de hiperparâmetros e avaliação.

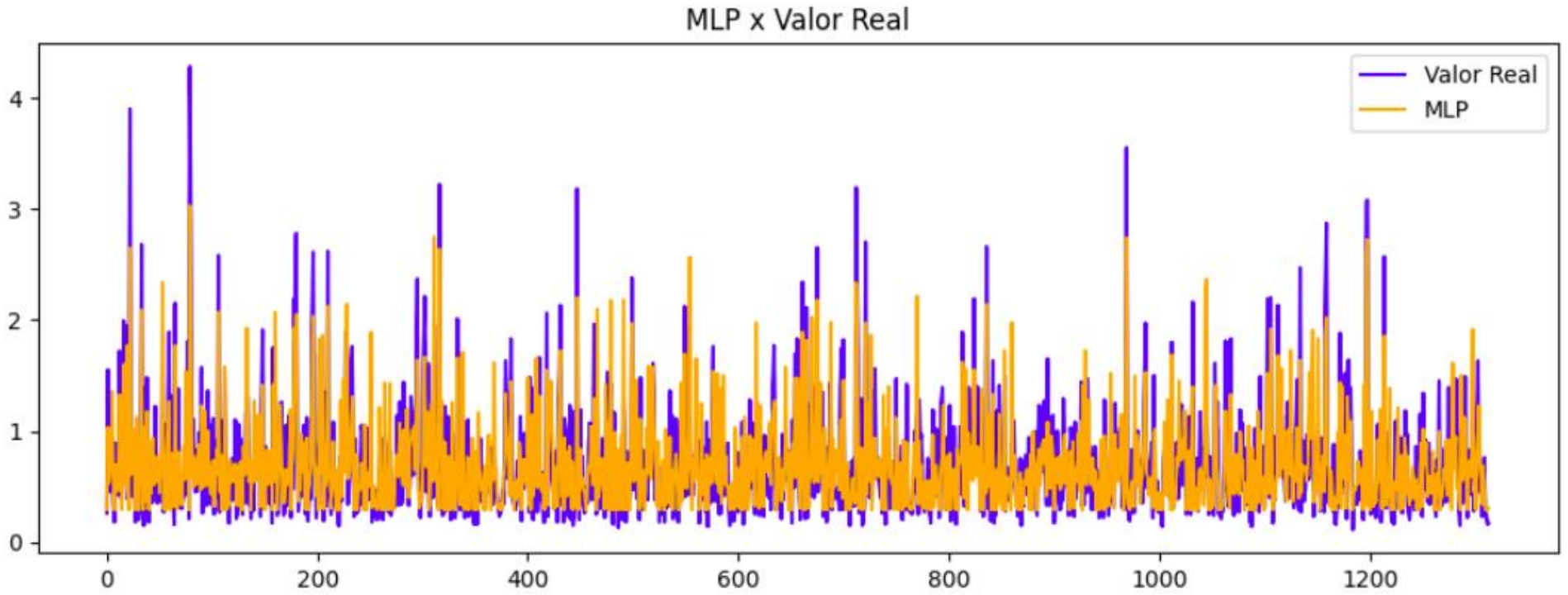
- **Modelos (MLP/KAN):**
  - **MLP:** Rede 32 → 16 → 8 → 4 → 2 → 1, ReLU, captura relações lineares e não-lineares.
  - **KAN:** Rede adaptativa com funções splines, mesma arquitetura, interpretável por feature.

- **Treinamento:**
  - Adam, LR=0.01, MSE, early stopping.
  - Objetivo: Minimizar o erro entre predição e ET0 real.

- **Avaliação:**
  - Teste final, gráficos comparativos.

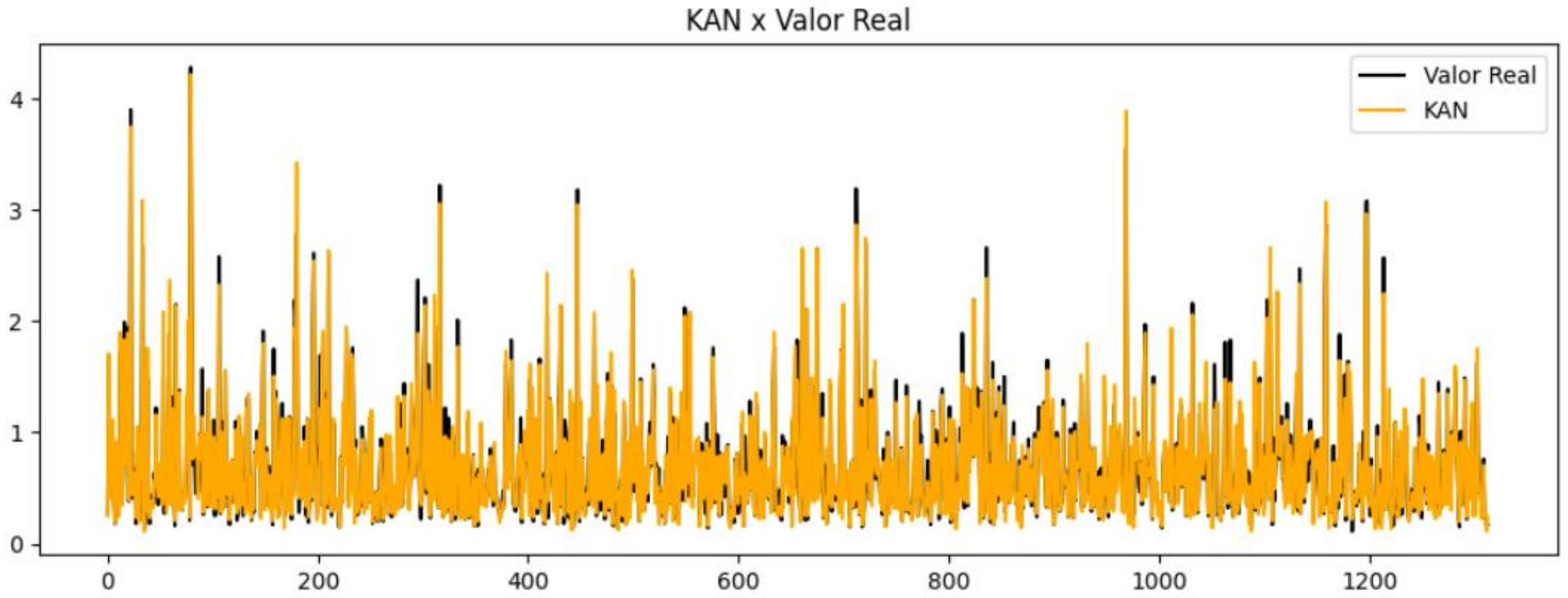
Resultados do MLP

Época	Treino Loss	Val Loss
0	1.4138	1.1956
20	0.5780	0.4920
40	0.4394	0.4071
60	0.4047	0.3726
80	0.3706	0.3481
99	0.3308	0.3481

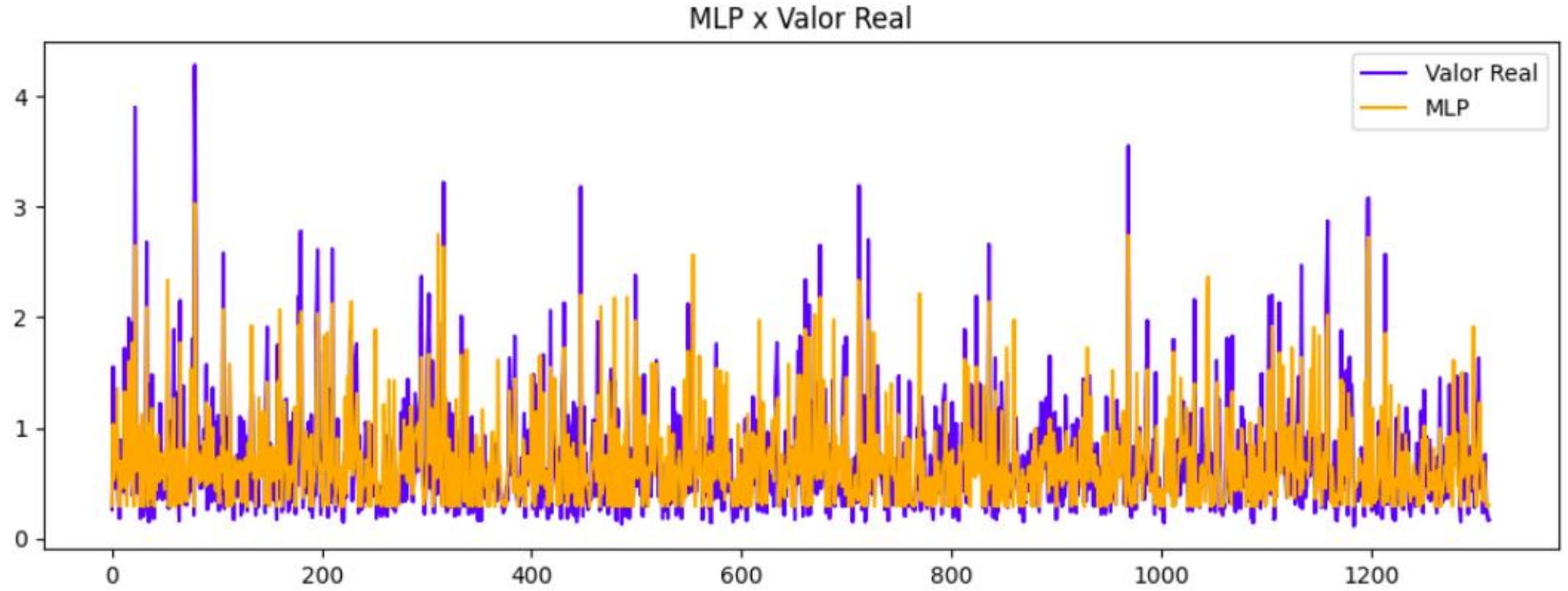


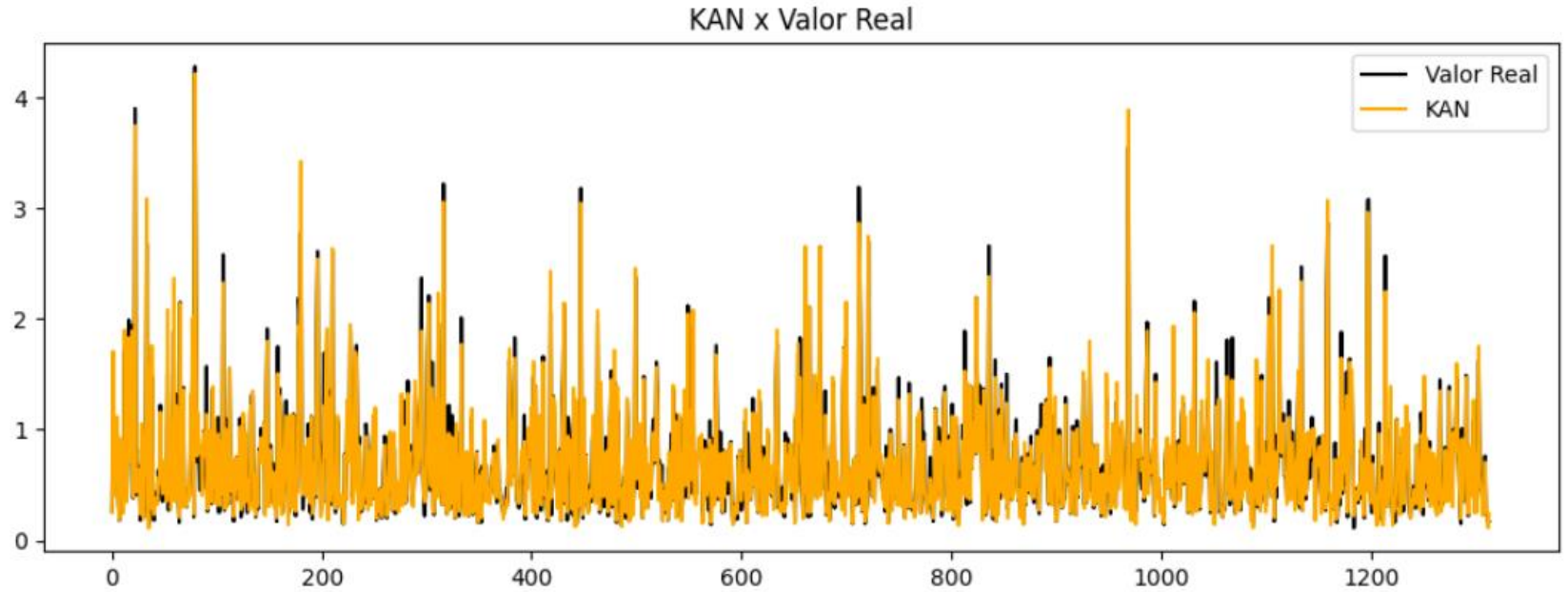
Resultados da KAN

Época	Treino Loss	Val Loss
0	1.0403	0.8607
20	0.3129	0.2551
40	0.1008	0.1164
60	0.0563	0.0837
80	0.0321	0.0588
99	0.0214	0.0479



Resultados MLP vs KAN





Vantagens e Desvantagens: *Kolmogorov-Arnold Networks*

Ponto de Comparação	Vantagem	Desvantagem
Interpretabilidade	Cada função univariada $\phi(x)$ é visualizável, facilitando análise e explicação científica.	Requer conhecimento para interpretar corretamente as funções;
Eficiência de Parâmetros	Representa funções complexas com menos parâmetros e redes mais rasas.	• Implementação mais complexa; • Custo computacional maior que MLPs.
Colaboração com Especialistas	Permite ajustes manuais nas funções $\phi(x)$ para incorporar conhecimento prévio.	Ajustes manuais podem demandar tempo e experiência.
Generalização / Suavidade	Funções contínuas promovem transições suaves e estabilidade na previsão.	Pode não generalizar tão bem em tarefas de visão ou NLP sem adaptações.
Escalabilidade Computacional	Melhor desempenho em tarefas de regressão simbólica e análise científica.	• Uso de recursos mais alto (BRAM, DSP, LUTs) e latência maior em hardware; • Até 100x mais lento que MLPs em alguns casos.

Inatel

Perguntas?

Referências

- [1] LIU, Ziming; WANG, Yixuan; VAIDYA, Sachin; RUEHLE, Fabian; HALVERSON, James; SOLJAČIĆ, Marin; HOU, Thomas Y.; TEGMARK, Max. *KAN: Kolmogorov-Arnold Networks.* arXiv preprint arXiv:2404.19756, 2024. Disponível em: https://arxiv.org/abs/2404.19756.

- [2] JAMALI, Ali; ROY, Swalpa Kumar; HONG, Danfeng; LU, Bing; GHAMISI, Pedram. *How to Learn More? Exploring Kolmogorov-Arnold Networks for Hyperspectral Image Classification.* Preprint, 2024. Disponível em: https://www.researchgate.net/publication/381667121_How_to_Learn_More_Exploring_Kolmogorov-Arnold_Networks_for_Hyperspectral_Image_Classification.

- [3] NEURAL NINJA. *Multi-Layer Perceptrons: Unlocking the Secrets of Neural Networks.* Let's Data Science, 7 mai. 2023. Disponível em: https://letsdatascience.com/multi-layer-perceptrons/.

- [4] BETHELL, Daniel. *Demystifying Kolmogorov-Arnold Networks: A Beginner-Friendly Guide with Code.* 13 maio 2024. Disponível em: https://daniel-bethell.co.uk/posts/kan/.

Repositório GitHub

- Link de acesso ao repositório:

- https://github.com/PauloLuczensky/tp558-959-Paulo/tree/main/seminarios/ciclo_3/c%C3%B3digo

Inatel

GitHub

Quizz

- Link de acesso ao Quizz:

- https://docs.google.com/forms/d/e/1FAIpQLSfzPosRxLRj48K7-ndTmc1sy5VESMS141n7OEKjane64V0HmA/viewform?usp=dialog

Inatel

Obrigado!