



Aproximação de soluções de EDPs via Redes Neurais Fisicamente Guiadas (PINNs)

Samuel Kutz Paranhos

Universidade Federal do Paraná, CiDAMO

19/07/2024

Visão Geral



1. O que é uma rede neural?

2. O Problema

3. Physics Informed Neural Network (PINN)

4. Resultados

O que é uma rede neural?

Motivação: Aproximação de Soluções

No contexto de **Equações Diferenciais**

- **Série de Taylor:** Aproxima funções por polinômios.

Motivação: Aproximação de Soluções



No contexto de **Equações Diferenciais**

- **Série de Taylor:** Aproxima funções por polinômios.
- **Série de Fourier:** Aproxima funções por combinações lineares de senos e cossenos.

Motivação: Aproximação de Soluções



No contexto de **Equações Diferenciais**

- **Série de Taylor:** Aproxima funções por polinômios.
- **Série de Fourier:** Aproxima funções por combinações lineares de senos e cossenos.
- **Redes Neurais:** Aproximam funções através de combinações lineares de função de ativação.

O que é uma rede neural?

Definição Formal

- Dada uma função $f : \mathbb{R}^n \rightarrow \mathbb{R}$, uma rede neural pode ser entendida como uma aproximação F de f definida da seguinte forma:

$$F(x) \equiv \sum_{j=1}^N \alpha_j \sigma(\mathbf{w}_j^T \mathbf{x} + b_j),$$

onde

O que é uma rede neural?

Definição Formal

- Dada uma função $f : \mathbb{R}^n \rightarrow \mathbb{R}$, uma rede neural pode ser entendida como uma aproximação F de f definida da seguinte forma:

$$F(x) \equiv \sum_{j=1}^N \alpha_j \sigma(\mathbf{w}_j^T \mathbf{x} + b_j),$$

onde

- $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ é uma função não linear a qual chamamos de função de ativação.

O que é uma rede neural?

Definição Formal

- Dada uma função $f : \mathbb{R}^n \rightarrow \mathbb{R}$, uma rede neural pode ser entendida como uma aproximação F de f definida da seguinte forma:

$$F(x) \equiv \sum_{j=1}^N \alpha_j \sigma(\mathbf{w}_j^T \mathbf{x} + b_j),$$

onde

- $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ é uma função não linear a qual chamamos de função de ativação.
- $\mathbf{b} = (b_1, \dots, b_N)$ é chamado de vetor de vieses. b_j é o viés do j -ésimo neurônio.

O que é uma rede neural?

Definição Formal

- Dada uma função $f : \mathbb{R}^n \rightarrow \mathbb{R}$, uma rede neural pode ser entendida como uma aproximação F de f definida da seguinte forma:

$$F(x) \equiv \sum_{j=1}^N \alpha_j \sigma(\mathbf{w}_j^T \mathbf{x} + b_j),$$

onde

- $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ é uma função não linear a qual chamamos de função de ativação.
- $\mathbf{b} = (b_1, \dots, b_N)$ é chamado de vetor de vieses. b_j é o viés do j -ésimo neurônio.
- $\mathbf{w}_j = (w_{1j}, \dots, w_{nj})$ é vetor de pesos do j -ésimo neurônio.

Representação Gráfica



Estrutura de uma Rede Neural de 1 camada

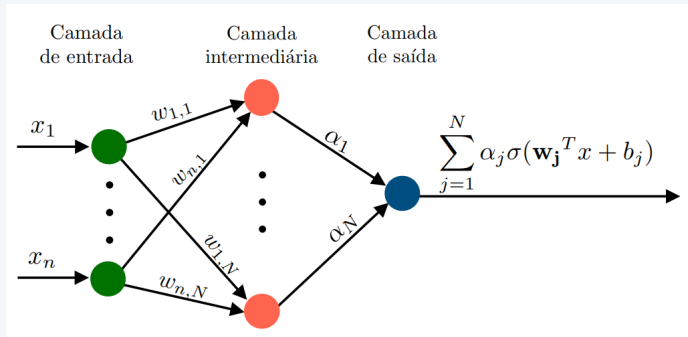


Figura: Representação gráfica de uma rede neural.

Teorema da Aproximação Universal

Seja $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ uma função contínua discriminatória (uma sigmóide), então dada qualquer $f : \Omega \subset \mathbb{R}^n$ **contínua**, e $\varepsilon > 0$, existem vetores $\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_N, \mathbf{w}_s, \mathbf{b}$ e uma função $F : \Omega \subset \mathbb{R}^n$ tais que

$$|F(\mathbf{x}) - f(\mathbf{x})| < \varepsilon, \quad \forall \mathbf{x} \in \Omega$$

onde

$$F(\mathbf{x}) = \sum_{j=1}^N \alpha_j \sigma(\mathbf{w}_j^T \mathbf{x} + b_j),$$

$$\mathbf{w}_s = (\alpha_1, \alpha_2, \dots, \alpha_N) \text{ e } \mathbf{b} = (b_1, b_2, \dots, b_N)$$

Prova: Veja [Hassoun, 1995] p.48

Camadas de uma Rede Neural



Forma Geral

- De maneira geral, a rede neural pode ser escrita na forma matricial como:

$$F(x) = \mathbf{w}_s^T \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}),$$

onde \mathbf{W} é uma matriz $N \times n$, com j -ésima linha sendo o vetor \mathbf{w}_j^T .

Camadas de uma Rede Neural



Forma Geral

- De maneira geral, a rede neural pode ser escrita na forma matricial como:

$$F(x) = \mathbf{w}_s^T \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}),$$

onde \mathbf{W} é uma matriz $N \times n$, com j -ésima linha sendo o vetor \mathbf{w}_j^T .

- Para redes neurais multicamadas, fazemos uma série de composições de funções afim com funções de ativação:

$$F(x) = \mathbf{W}^L \sigma^L (\mathbf{W}^{L-1} \sigma^{L-1} (\dots \sigma^1 (\mathbf{W}^0 \mathbf{x} + \mathbf{b}^0) \dots) + \mathbf{b}^{L-1}) + \mathbf{b}^L,$$

onde \mathbf{W}^l e \mathbf{b}^l são matrizes de pesos e vetores de vieses. A função de ativação σ é uma função de uma variável real, aplicada a cada entrada do vetor.

Funções de Ativação



Importância das Funções de Ativação

- Introduzem não-linearidade, permitindo à rede neural aprender relações complexas.
- Exemplos comuns incluem a função sigmoid, ReLU (Rectified Linear Unit), e a função **tangente hiperbólica**.

Processo de Treinamento



Como Treinar uma Rede Neural

- Utilizamos um conjunto de dados de treinamento para ajustar os pesos w e os vieses b .
- O objetivo é minimizar a **função de perda**, que mede o erro entre a saída prevista pela rede e a saída desejada.
- Algoritmos comuns de treinamento incluem Gradient Descent e o otimizador **ADAM**.

O que é uma rede neural?



Em resumo, uma rede neural consiste em

- Um aproximador universal de funções não-lineares

O que é uma rede neural?

Em resumo, uma rede neural consiste em

- Um aproximador universal de funções não-lineares
- Camadas e neurônios, que são *hiper-parâmetros*

O que é uma rede neural?

Em resumo, uma rede neural consiste em

- Um aproximador universal de funções não-lineares
- Camadas e neurônios, que são *hiper-parâmetros*
- A função ativação, que também deve ser escolhida

O que é uma rede neural?

Em resumo, uma rede neural consiste em

- Um aproximador universal de funções não-lineares
- Camadas e neurônios, que são *hiper-parâmetros*
- A função ativação, que também deve ser escolhida
- Para encontrar w e b devemos resolver o problema de otimização dado pela função de perda, também chamada **loss function**

O Problema

Equação de Benjamin-Bona-Mahony (BBM)



A Equação Diferencial Parcial que trataremos consiste em:

$$\begin{aligned}u_t + u_x + u u_x - u_{xxt} &= 0, & (x, t) &\in (-10, 20) \times (0, 4) \\u(x, 0) &= u_0(x), & x &\in (-10, 20), \\u(-10, t) &= g_1(t) \\u(20, t) &= g_2(t) & t &\in (0, 4),\end{aligned}$$

Onde

Equação de Benjamin-Bona-Mahony (BBM)



A Equação Diferencial Parcial que trataremos consiste em:

$$\begin{aligned}u_t + u_x + u u_x - u_{xxt} &= 0, & (x, t) &\in (-10, 20) \times (0, 4) \\u(x, 0) &= u_0(x), & x &\in (-10, 20), \\u(-10, t) &= g_1(t) \\u(20, t) &= g_2(t) & t &\in (0, 4),\end{aligned}$$

Onde

- $u = u(x, t)$ é a função que queremos aproximar

Equação de Benjamin-Bona-Mahony (BBM)



A Equação Diferencial Parcial que trataremos consiste em:

$$\begin{aligned}u_t + u_x + u u_x - u_{xxt} &= 0, & (x, t) &\in (-10, 20) \times (0, 4) \\u(x, 0) &= u_0(x), & x &\in (-10, 20), \\u(-10, t) &= g_1(t) \\u(20, t) &= g_2(t) & t &\in (0, 4),\end{aligned}$$

Onde

- $u = u(x, t)$ é a função que queremos aproximar
- $u_0(x)$ é a **condição inicial** em $t = 0$

Equação de Benjamin-Bona-Mahony (BBM)



A Equação Diferencial Parcial que trataremos consiste em:

$$\begin{aligned}u_t + u_x + u u_x - u_{xxt} &= 0, & (x, t) &\in (-10, 20) \times (0, 4) \\u(x, 0) &= u_0(x), & x &\in (-10, 20), \\u(-10, t) &= g_1(t) \\u(20, t) &= g_2(t) & t &\in (0, 4),\end{aligned}$$

Onde

- $u = u(x, t)$ é a função que queremos aproximar
- $u_0(x)$ é a **condição inicial** em $t = 0$
- g_1, g_2 são **condições de contorno** que nos dizem como u se comporta nas bordas do domínio de x .

Escolha do problema

Para um u_0 específico, a BBM tem solução exata

- Se $u_0(x) = A \operatorname{sech}^2(kx)$, então

$$u(x, t) = A \operatorname{sech}^2(k(x - ct))$$

Onde

Escolha do problema

Para um u_0 específico, a BBM tem solução exata

- Se $u_0(x) = A \operatorname{sech}^2(kx)$, então

$$u(x, t) = A \operatorname{sech}^2(k(x - ct))$$

Onde

- A é dado e chamado de **amplitude**

Escolha do problema

Para um u_0 específico, a BBM tem solução exata

- Se $u_0(x) = A \operatorname{sech}^2(kx)$, então

$$u(x, t) = A \operatorname{sech}^2(k(x - ct))$$

Onde

- A é dado e chamado de **amplitude**
- $k = \sqrt{\frac{A}{12 + 4A}}$ e chamado de **frequência**

Escolha do problema

Para um u_0 específico, a BBM tem solução exata

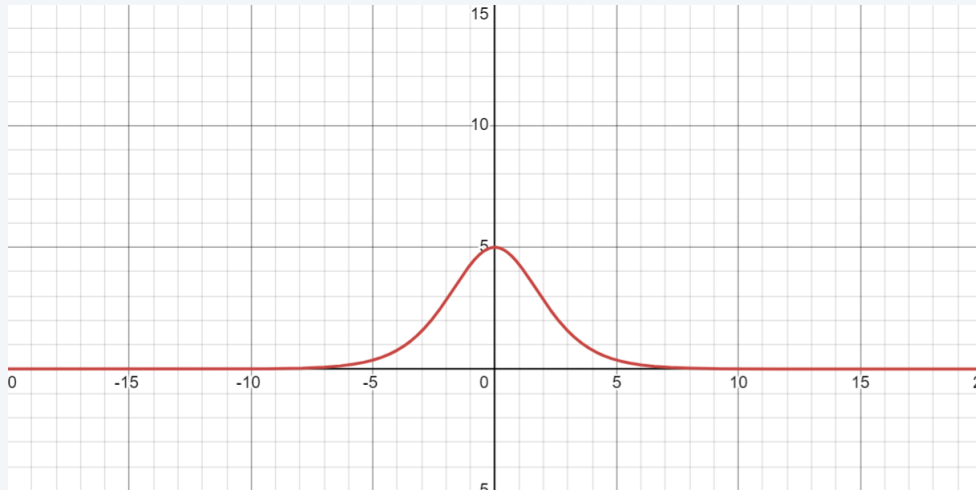
- Se $u_0(x) = A \operatorname{sech}^2(kx)$, então

$$u(x, t) = A \operatorname{sech}^2(k(x - ct))$$

Onde

- A é dado e chamado de **amplitude**
- $k = \sqrt{\frac{A}{12 + 4A}}$ e chamado de **frequência**
- $c = 1 + \frac{A}{3}$ e chamado de **velocidade**

Gráfico da u_0



Escolha do problema



- Escolhendo $A = 5$, vamos determinar o domínio (bastante arbitrário) como $x \in (-10, 20)$ e $t \in (0, 4)$.

Escolha do problema

- Escolhendo $A = 5$, vamos determinar o domínio (bastante arbitrário) como $x \in (-10, 20)$ e $t \in (0, 4)$.
- As condições de contorno g_1 e g_2 serão a própria solução exata $u(x,t)$ em $x = -10$ e $x = 20$, respectivamente

Escolha do problema

- Escolhendo $A = 5$, vamos determinar o domínio (bastante arbitrário) como $x \in (-10, 20)$ e $t \in (0, 4)$.
- As condições de contorno g_1 e g_2 serão a própria solução exata $u(x,t)$ em $x = -10$ e $x = 20$, respectivamente
- Isso nos dá um bom **problema teste**, pois temos como comparar a solução da rede neural com a solução exata.

Physics Informed Neural Network (PINN)

$$\begin{aligned}u_t + u_x + u u_x - u_{xxt} &= 0, & (x, t) &\in (-10, 20) \times (0, 4) \\u(x, 0) &= u_0(x), & x &\in (-10, 20), \\u(-10, t) &= g_1(t) \\u(20, t) &= g_2(t) & t &\in (0, 4),\end{aligned}$$

Vamos determinar uma função $u(x, t)$ que resolve simultaneamente os seguintes problemas de otimização:

1.

$$\min |f(x, t)| \quad \text{s.a.} \quad (x, t) \in (-10, 20) \times (0, 4),$$

onde

$$f := u_t + u_x + u u_x - u_{xxt}$$

$$\begin{aligned}u_t + u_x + u u_x - u_{xxt} &= 0, & (x, t) &\in (-10, 20) \times (0, 4) \\u(x, 0) &= u_0(x), & x &\in (-10, 20), \\u(-10, t) &= g_1(t) \\u(20, t) &= g_2(t) & t &\in (0, 4),\end{aligned}$$

Vamos determinar uma função $u(x, t)$ que resolve simultaneamente os seguintes problemas de otimização:

2.

$$\min |u(x, 0) - u_0(x)| \quad \text{s.a.} \quad x \in (-10, 20).$$

$$\begin{aligned}u_t + u_x + u u_x - u_{xxt} &= 0, & (x, t) &\in (-10, 20) \times (0, 4) \\u(x, 0) &= u_0(x), & x &\in (-10, 20), \\u(-10, t) &= g_1(t) \\u(20, t) &= g_2(t) & t &\in (0, 4),\end{aligned}$$

Vamos determinar uma função $u(x, t)$ que resolve simultaneamente os seguintes problemas de otimização:

3.

$$\min |u(-10, t) - g_1(t)| + |u(20, t) - g_2(t)| \quad \text{s.a.} \quad t \in (0, 4).$$

A função $u(x, t)$ será aproximada via Redes Neurais por meio da **minimização** do **Erro Médio Quadrático (EMQ)** da função de perda que é dada por

$$\text{EMQ} = \text{EMQ}_f + \text{EMQ}_0 + \text{EMQ}_b$$

Resíduo da EDP

$$\text{EMQ}_f = \frac{1}{N_f} \sum_{i=1}^{N_f} |f(x_i, t_i)|^2,$$

- $(x_i, t_i), i = 1, \dots, N_f$, representa N_f pontos de colocação escolhidos aleatoriamente em $(-10, 20) \times (0, 4)$

Resíduo da Condição Inicial

$$\text{EMQ}_0 = \frac{1}{N_0} \sum_{i=1}^{N_0} |u(x_i, 0) - u_0(x_i)|^2,$$

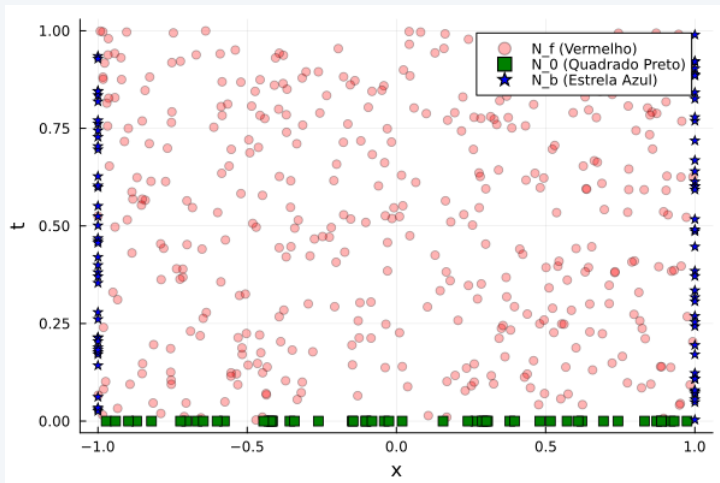
- $x_i, i = 1, \dots, N_0$ representa N_0 pontos de colocação no intervalo $(-10, 20)$

Resíduo da Condição de Contorno

$$\text{EMQ}_b = \frac{1}{N_b} \sum_{i=1}^{N_b} |u(x_i, t_i)|^2,$$

- $(x_i, t_i), i = 1, \dots, N_b$ corresponde à N_b pontos de colocação ao longo do conjunto cartesiano $\{-10, 20\} \times (0, 4)$

0 "Dataset"



Resultados

Resultados



Seguindo esses valores:

Parâmetro	Valor
Amplitude inicial (A)	5
Domínio espacial (x)	$[-10, 20]$
Domínio temporal (t)	$[0, 4]$
Pontos de colocação no domínio (N_f)	15 000
Pontos de colocação na fronteira (N_b)	100
Pontos de colocação no valor inicial (N_0)	100
Neurônios por camada	50
Camadas intermediárias	3
Iterações do otimizador	15 000

Resultados



- `exactVSmodel.gif`

Utilizando a biblioteca de Python chamada **DeepXDE** [Lu et al., 2021], que tem como objetivo exatamente a implementação de Redes Neurais em Equações Diferenciais, conseguimos obter resultados convincentes do potencial das PINNs.

Resultados



- `exactVSmodel.gif`
- Obtivemos um erro relativo global de $5.3e-03$

Utilizando a biblioteca de Python chamada **DeepXDE** [Lu et al., 2021], que tem como objetivo exatamente a implementação de Redes Neurais em Equações Diferenciais, conseguimos obter resultados convincentes do potencial das PINNs.





LABFluid

Laboratório de Dinâmica
dos Fluidos Computacional



Referências I



-  Hassoun, M. (1995).
Fundamentals of artificial neural networks.
MIT Press.
-  Lu, L., Meng, X., Mao, Z., and Karniadakis, G. E. (2021).
DeepXDE: A deep learning library for solving differential equations.
SIAM Review, 63(1):208–228.



Obrigado pela atenção!

Samuel Kutz Paranhos

Universidade Federal do Paraná, CiDAMO

19/07/2024