Redes Neurais e Deep Learning

Aula 02 – Matemática para Redes Neurais

Prof. Érick T. Yamamoto

FIAP
GRADUAÇÃO

Motivação

Redes neurais são baseadas em operações matemáticas como matrizes e gradientes.

Conteúdo:

- Álgebra Linear (Matrizes e Vetores).
- Cálculo Diferencial (Derivadas, Gradiente Descendente).
- Implementação de gradiente descendente.

Álgebra Linear para Redes Neurais

O que são vetores e matrizes?

Vetores são listas ordenadas de números que representam magnitudes e direções no espaço. Matematicamente, um vetor pode ser expresso como:

$$v = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$$

onde v é um vetor de dimensão n, contendo n elementos $v1,v2,\ldots,vn$.

Álgebra Linear para Redes Neurais

O que são vetores e matrizes?

Matrizes são tabelas de números organizadas em linhas e colunas, utilizadas para armazenar grandes volumes de dados e realizar operações matemáticas fundamentais em redes neurais. Exemplo de matriz:

$$A_{mxn} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$$

Onde os elementos contidos na tem dois números representados pela linha m e colunas n.

Álgebra Linear para Redes Neurais

Operações Matriciais - Soma de Matrizes

A soma de duas matrizes A e B só é possível se elas possuírem as mesmas dimensões $m \times n$. A soma é feita somando elemento a elemento:

$$A + B = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11} + b_{11} & a_{12} + b_{12} \\ a_{21} + b_{21} & a_{22} + b_{22} \end{bmatrix}$$

Exemplo:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

$$A + B = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} + \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 1 + 5 & 2 + 6 \\ 3 + 7 & 4 + 8 \end{bmatrix} = \begin{bmatrix} 6 & 8 \\ 10 & 12 \end{bmatrix}$$

Álgebra Linear para Redes Neurais

Operações Matriciais - Soma de Matrizes na Linguagem Python

```python
import numpy as np
A = np.array([[1, 2], [3, 4]])
B = np.array([[5, 6], [7, 8]])
C = A + B
print(C)
```

Álgebra Linear para Redes Neurais

Operações Matriciais – Multiplicação Escalar e Matricial

A operação de produto/ multiplicação representada por (·) ou (*) , pode ser feito entre vetores, entre matrizes ou matriz e vetor:

1. Produto Escalar:

$$u \cdot v = u_1 \cdot v_1 + u_2 \cdot v_2 + \cdots + u_n \cdot v_n$$

2. Produto Matricial: Se A é uma matriz de dimensão m x n e B uma matriz n x p, o produto $C = A \cdot B$ *resulta em uma matriz m x p .*

$$C = A \cdot B, \qquad C_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj}$$

Álgebra Linear para Redes Neurais

Operações Matriciais – Multiplicação Escalar e Matricial

A operação de produto/ multiplicação representada por (·) ou (*) , pode ser feito entre vetores, entre matrizes ou matriz e vetor:

3. Produto entre vetor e matriz:

$$\lambda \cdot B = \lambda \cdot \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} \lambda b_{11} & \lambda b_{12} \\ \lambda b_{21} & \lambda b_{22} \end{bmatrix}$$

Álgebra Linear para Redes Neurais

Operações Matriciais – Multiplicação Escalar e Matricial no python

Exemplos:

1. Produto vetor com Matriz:

```
A = np.array([[1, 2], [3, 4]])
scalar = 2
C = scalar * A
print(C)
```

2. Produto Matricial:

```
A = np.array([[1, 2], [3, 4]])
B = np.array([[5, 6], [7, 8]])
C = np.dot(A, B)  # Ou A @ B
print(C)
```

Álgebra Linear para Redes Neurais

Operações Matriciais – Transposição de Matrizes

A transposição de uma matriz A é obtida trocando suas linhas por colunas:

$$A^T = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}^T = \begin{bmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \end{bmatrix}$$

Exemplo em python:

```
A = np.array([[1, 2], [3, 4]])
A_T = A.T
print(A_T)
```

Álgebra Linear para Redes Neurais

Vimos que as operações básicas de álgebra linear, como soma e multiplicação de matrizes, são essenciais para as redes neurais. No entanto, será que utilizaremos apenas essas operações elementares ou teremos algo um pouco mais complexo?

A introdução aos conceitos de autovalores e autovetores nos leva a um novo nível de compreensão, permitindo explorar temas como **redução de dimensionalidade** e **estabilidade** no aprendizado de redes neurais. Vamos descobrir juntos como essas ferramentas avançadas podem transformar nossa forma de entender os modelos de inteligência artificial!

Cálculo Diferencial

Derivadas e Gradientes Descendentes

Cálculo Diferencial para Redes Neurais

Revisão para as Derivadas: Taxa de Variação.

A derivada de uma função mede sua taxa de variação instantânea. Se tivermos uma função $f(x)$, sua derivada $f'(x)$ indica a inclinação da curva naquele ponto.

Exemplo - Seja a função:

$$f(x) = x^2 + 3x + 2$$

A derivada é calculada como: $f'(x) = \frac{d(x^2+3x+2)}{dx} = 2x + 3$

Em python

```python
import sympy as sp
x = sp.Symbol('x')
f = x**2 + 3*x + 2
derivada = sp.diff(f, x)
print(derivada)
```

Cálculo Diferencial para Redes Neurais

Revisão para as Derivadas: Derivadas Parciais.

As derivadas parciais são uma extensão do conceito de derivadas para funções de múltiplas variáveis. Em redes neurais, essas derivadas são fundamentais para entender como cada variável de entrada afeta a saída da rede e são amplamente utilizadas no **gradiente descendente** e no **backpropagation**.

Cálculo Diferencial para Redes Neurais

Seja uma função $f(x, y)$, de duas variáveis:

$$f(x, y) = x^2 + 3xy + y^2$$

As derivadas parciais de $f(x, y)$ em relação a x e y são calculadas diferenciando f apenas em relação a uma variável, mantendo a outra constante.

- Derivada Parcial em relação a x:

$$\frac{\partial f}{\partial x} = \frac{d(x^2 + 3xy + y^2)}{dx} = 2x + 3y$$

$\frac{\partial f}{\partial x}$ mede como f(x,y) muda se apenas x for alterado, mantendo y constante.

COMO SERÁ QUE FICA PARA $\frac{\partial f}{\partial y}$?

Cálculo Diferencial para Redes Neurais

A função $f(x, y) = x^2 + 3xy + y^2$, de duas variáveis em python, ficaria:

```python
import sympy as sp

# Definindo as variáveis
x, y = sp.symbols('x y')

# Definindo a função
f = x**2 + 3*x*y + y**2

# Cálculo das derivadas parciais
df_dx = sp.diff(f, x)
df_dy = sp.diff(f, y)

print("Derivada parcial em relação a x:", df_dx)
print("Derivada parcial em relação a y:", df_dy)
```

Gradiente Descendente e Função de Custo

Otimizando as Redes Neurais Artificiais.

Gradiente Descendente e Função de Custo

O **gradiente descendente** é um dos algoritmos mais importantes para otimização de funções, especialmente em aprendizado de máquina e redes neurais. Ele permite encontrar os valores ótimos dos parâmetros de um modelo minimizando uma **função de erro ou custo**.

O QUE É UMA FUNÇÃO DE CUSTO?

A **função de custo** mede o erro entre as previsões de um modelo e os valores reais. O objetivo do treinamento de uma rede neural é minimizar essa função para melhorar a precisão do modelo.

Gradiente Descendente e Função de Custo

Tipos comuns de funções de custo:

1. Erro Quadrático Médio (MSE):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y_i})^2$$

2. Erro Absoluto Médio (MAE):

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \widehat{y_i}|$$

Gradiente Descendente e Função de Custo

Tipos comuns de funções de custo:

3. Entropia Cruzada (Cross-Entropy Loss):

$$L = -\sum_{i=1}^{n} y_i \log(\hat{y}_i)$$

EXISTEM OUTRAS FUNÇÕES QUE PODEM SER APLICADAS, PORÉM ESSAS QUE FORAM APRESENTADAS SÃO AS PRINCIPAIS E MAIS UTILIZADAS.

Gradiente Descendente e Função de Custo

Dada uma função de custo $L(w)$, onde w representa os parâmetros do modelo, o gradiente descendente ajusta iterativamente esses parâmetros na direção negativa do gradiente:

$$w = w - \alpha \nabla L(w)$$

Onde:

- α é a taxa de aprendizado, que controla o tamanho do passo da atualização;
- $\nabla L(w)$ é o gradiente da função de custo, que indica a direção de maior crescimento.

Iremos implementar e entender com o python.

Gradiente Descendente e Função de Custo

Iremos implementar e entender com o python.

```python
import numpy as np

def mse(y_real, y_pred):
    return np.mean((y_real - y_pred) ** 2)

def gradient(y_real, y_pred, x):
    return -2 * np.mean((y_real - y_pred) * x)

# Inicialização
w = 0
alpha = 0.1
num_steps = 10
y_real = np.array([3, 5, 7, 9])
x = np.array([1, 2, 3, 4])

for i in range(num_steps):
    y_pred = w * x
    grad = gradient(y_real, y_pred, x)
    w = w - alpha * grad
```

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$\frac{\partial MSE}{\partial w} = -\frac{2}{n}\sum_{i=1}^{n}(y_i - wx_i) * x_i$$

Descanso

Do professor =)

Exercícios

Álgebra Linear e Matrizes

1. Dada a matriz $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ e a matriz $B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$, calcule a soma $A + B$ e a multiplicação $A \times B$ manualmente e usando Python.
2. Calcule a transposta da matriz $C = \begin{bmatrix} 2 & 4 & 6 \\ 1 & 3 & 5 \end{bmatrix}$ e implemente a operação em Python.
3. Encontre o produto escalar dos vetores $\mathbf{u} = [1, 2, 3]$ e $\mathbf{v} = [4, 5, 6]$ e implemente em Python.

Cálculo Diferencial e Gradiente Descendente

4. Encontre a derivada da função $f(x) = 3x^2 + 2x + 1$ manualmente e implemente o cálculo usando SymPy.
5. Calcule a integral indefinida da função $g(x) = 4x^3 - 2x + 5$ manualmente e usando Python.
6. Para a função $h(x, y) = x^2 + xy + y^2$, calcule as derivadas parciais $\frac{\partial h}{\partial x}$ e $\frac{\partial h}{\partial y}$.