Descrição formal das funções de ativação de modelos de aprendizado de máquina

Formal description of activation functions of machine learning models

Julia Assunção Leal[1], Hellena Christina Fernandes Apolinário[1] e Rogério Azevedo Rocha[1]

[1] *Curso Bacharelado em Ciência da Computação, Universidade Federal do Tocantins (UFT), Palmas/TO*

Data de recebimento do manuscrito: 29/11/2024
Data de aceitação do manuscrito: 25/02/2025
Data de publicação: 13/03/2025

Resumo— Modelos de inteligência artificial são cada vez mais comuns em vários aspectos do dia a dia. Não só nos casos mais emblemáticas, mas também para casos corriqueiros como em sistemas de recomendação em sites de compras. Nesse sentido, é muito importante o entendimento de como esses modelos funcionam por parte dos desenvolvedores. Contudo, o uso massivo de bibliotecas para utilização desses modelos pode desfavorecer esse entendimento. Assim, esse trabalho traz a definição e demonstração formal das funções de ativação em modelos de aprendizado de máquina. Esse é um ponto fundamental para a introdução do assunto à novos desenvolvedores e cientistas que trabalharão na área. A descrição formal se aplica às clássicas funções ReLU, Sigmóide, tangente hiperbólica, softmax e gradiente descendente. Além disso, também são discutidos o impacto dessas funções no modelo LeNet-5 aplicado à base de dados MNIST.

Palavras-chave—Inteligência Artificial, Matemática Aplicada, Funções de Ativação, ReLU, Sigmóide, TanH, Softmax, Método do gradiente

Abstract— *Artificial intelligence models are increasingly common in many aspects of everyday life, not only in the most emblematic cases, but also in everyday cases such as recommendation systems on shopping websites. In this sense, developers need to understand how these models work. However, the massive use of libraries to use these models can hinder this understanding. Therefore, this work defines and formally demonstrates activation functions in machine learning models. This is a fundamental point for introducing the subject to new developers, and scientists, who will work in the area. The formal description applies to the classic ReLU, Sigmóide, hyperbolic tangent, softmax and gradient descent functions. In addition, the impact of these functions on the LeNet-5 model applied to the MNIST database is also discussed.*

Keywords—*Artificial Intelligence, Applied Mathematics, Activation Functions, ReLU, Sigmoid, TanH, Softmax, Gradient Descent*

I. Introdução

Com o avanço da tecnologia nos campos da computação e da matemática, o desenvolvimento da Inteligência Artificial (IA) tornou-se mais eficiente e próximo da realidade. A matemática, nesse contexto, configura-se como a base que fundamenta a IA, fornecendo teorias e ferramentas essenciais para o aprendizado das máquinas (ML ou *Machine Learning*). De algoritmos complexos a modelos preditivos, a matemática transforma grandes volumes de dados brutos em informações compreensíveis, contribuindo para o progresso em diversas áreas [1, 2, 3, 4].

Esse artigo foca na base matemática para compreender o funcionamento de uma IA, com interesse em apoiar pesquisadores iniciantes na área. Entre os conceitos matemáticos indispensáveis para a IA, destacam-se as funções de ativação, fundamentais para a arquitetura das redes neurais. Essas funções transformam a entrada linear de cada neurônio em uma saída não-linear, permitindo que as redes aprendam padrões complexos nos dados. Sem elas, uma rede neural seria apenas um modelo linear, incapaz de realizar tarefas mais elaboradas, como o reconhecimento de fala, o processamento de imagens ou a interpretação de linguagem natural [5]. Assim, o estudo das funções é crucial para tornar os sistemas mais eficientes e funcionais.

Outro conceito relevante é a otimização por gradientes, que ajusta os parâmetros dos modelos de IA. Por meio do cálculo do gradiente (a derivada do erro de uma função em relação aos pesos e vieses) é possível minimizar o erro e aprimorar o aprendizado do modelo. O método do gradiente descendente, por exemplo, permite encontrar o mínimo de uma função de custo de forma iterativa, sendo

Open Data

Dados de contato: Julia Assunção Leal, leal.julia@uft.edu.br

essencial no treinamento de redes neurais [6]. Além disso, a probabilidade desempenha um papel vital, possibilitando que modelos lidem com incertezas e façam inferências sobre dados variáveis. Redes bayesianas e processos gaussianos, por exemplo, são amplamente utilizados em áreas como diagnósticos médicos e sistemas de recomendação [7].

Portanto, a matemática não apenas sustenta a Inteligência Artificial, mas também potencializa seu impacto em diversas áreas, como a medicina e a indústria. Para garantir o contínuo avanço nessa área, é fundamental que instituições de ensino ampliem os investimentos em disciplinas que integrem matemática aplicada e computação.

II. Explicação geral sobre o funcionamento de uma IA

Segundo [1], o desenvolvimento de uma Inteligência Artificial (IA) envolve diversas etapas cruciais. Inicialmente, ocorre a coleta e preparação dos dados, etapa na qual os dados são limpos, normalizados e organizados para serem utilizados no treinamento do modelo. Durante o treinamento, o modelo aplica funções de ativação, que introduzem não linearidade e permitem que ele se adapte a funções mais complexas, mas essas funções não processam os dados diretamente, mas transformam os resultados intermediários gerados por uma combinação linear dos dados e dos pesos do modelo. Em cada neurônio, após a multiplicação dos dados de entrada pelos seus respectivos pesos e a adição de um viés, a função de ativação entra em ação.

Em seguida, o modelo faz previsões com base nos dados de entrada, e essas previsões são comparadas aos resultados esperados. A diferença entre eles é mensurada por meio de uma função de perda, que calcula o erro. Esse erro é então utilizado no processo de retropropagação, onde ele é propagado para trás pelas camadas do modelo, ajustando os pesos dos neurônios para melhorar a precisão. Esse ciclo se repete até que o erro seja minimizado ou atinja um nível aceitável, otimizando o desempenho do sistema.

III. Definição e Propriedades da função ReLU

A função de ativação da Unidade Linear Rectificada (ReLU), amplamente conhecida e adotada em redes neurais profundas e no contexto de aprendizado de máquina [1] [5], foi introduzida por [8]. Ela se destaca pela simplicidade e eficácia na modelagem da ativação de neurônios artificiais.

Para compreender o comportamento dessa função, que é constante em $x = 0$, até o ponto $y = 0$ onde os valores aumentam,tem-se a Figura 1, feita a partir do modelo proposto pelo estudo.

Matematicamente, a função ReLU é definida como:

- **Fórmula**: $f(x) = \max(0,x)$
- **Domínio**: $\mathbb{R}$ (todos os números reais)
- **Imagem**: $[0,\infty)$

ReLU pode ser expresso de diferentes maneiras equivalentes. Formalmente, temos:

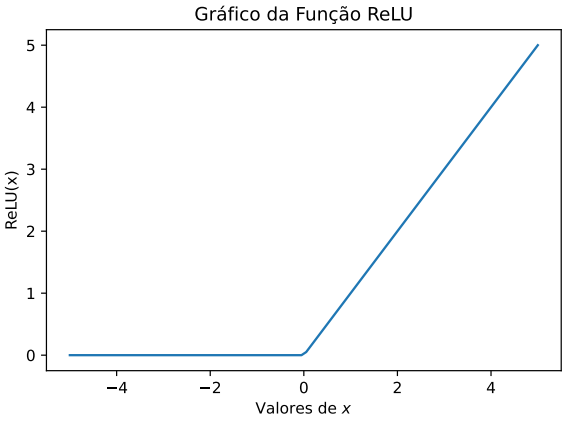

Figura 1: Função ReLU testada com valores do MNIST.

$$f(x) = x^+ = \max(0,x) = \frac{x+|x|}{2}$$

A função atua como uma unidade linear para valores de entrada positivos, enquanto zera os valores de entrada negativos:

$$f(x) = \begin{cases} x & \text{se } x > 0, \\ 0 & \text{se } x \leq 0. \end{cases}$$

a. Derivada da função ReLU:

$$f'(x) = \begin{cases} 1 & \text{se } x > 0, \\ 0 & \text{se } x < 0. \end{cases}$$

Observa-se que em $x = 0$, a derivada não existe, pois:

$$\lim_{h\to 0^+} \frac{f(a+h)-f(a)}{h} \neq \lim_{h\to 0^-} \frac{f(a+h)-f(a)}{h}$$

Se a variável é positiva ($x > 0$):

$$f(x) = \frac{x+|x|}{2} = \frac{x+x}{2} = \frac{2x}{2} = x$$

E se a variável é negativa ($x < 0$):

$$f(x) = \frac{x+|x|}{2} = \frac{x-x}{2} = \frac{0}{2} = 0$$

A função de ativação ReLU, devido à sua simplicidade e capacidade de diminuir o problema do desaparecimento do gradiente [1]. tornou-se uma escolha normal em redes neurais simples. Este estudo destacou suas propriedades, representação gráfica e aplicação em contextos de aprendizado de máquina.

b. Aplicação de ReLU em redes neurais

A classificação de imagens é uma tarefa fundamental no campo da visão computacional, e o dataset MNIST, composto por dígitos manuscritos, consiste em imagens de 28x28 pixels em tons de cinza, divididas em conjuntos de treinamento e teste. Neste estudo, foi implementada a arquitetura LeNet-5, uma rede neural convolucional clássica, para realizar a classificação dos dígitos do MNIST [9]. O objetivo principal foi comparar o desempenho do modelo

ISSN: 2675-3588

ao utilizar diferentes funções de ativação, analisando sua influência na precisão e na capacidade de generalização do modelo.

A arquitetura LeNet-5 foi escolhida por sua simplicidade em tarefas de classificação de imagens. Ela é composta por camadas convolucionais, seguidas por camadas de pooling para redução de dimensionalidade, e finalizada com camadas totalmente conectadas. A sua última camada utiliza a função Softmax, que é ideal para problemas de classificação multiclasse, pois produz uma distribuição de probabilidade sobre as classes. Para investigar o impacto das funções de ativação, foram testadas três opções: ReLU, Sigmóide e TanH. Cada uma dessas funções de ativação tem características distintas, influenciando a capacidade do modelo de aprender padrões complexos nos dados.

O treinamento foi realizado por 10 epochs (uma passagem completa pelo conjunto de dados de treinamento), com um *batch size* (número de amostras que são processadas pelo modelo antes que os pesos da rede sejam atualizados) 128. Durante o treinamento, o modelo foi avaliado no conjunto de validação, permitindo monitorar seu desempenho e evitar **overfitting** (ajuste excessivo a um conjunto de dados ao ponto de não generalizar para novos dados). Após o treinamento, o modelo foi avaliado no conjunto de teste de dados, onde foram calculadas a perda e a precisão.

A função de ativação ReLU (*Rectified Linear Unit*) destacou-se como a mais eficiente na classificação dos dígitos do dataset MNIST utilizando a arquitetura LeNet-5. Durante o treinamento, a ReLU proporcionou uma convergência rápida e estável, alcançando uma precisão final de 97,76% por cento no conjunto de teste, com uma perda de 0,0706. A característica principal da ReLU é sua capacidade de ativar apenas valores positivos, o que evita o problema de gradientes desaparecendo, comum em outras funções de ativação. Isso permitiu ao modelo capturar padrões complexos nos dados de forma eficaz, resultando em uma generalização robusta.

Além disso, para uma análise mais detalhada, foram geradas matrizes de confusão 2, 4 e 6 .

Na Figura 2 tem-se a matriz referente à função ReLU:

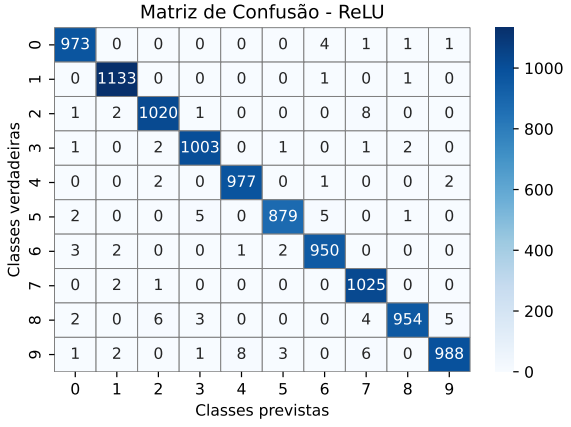

Figura 2: Matriz de confusão testada com valores do MNIST.

IV. Definição e propriedades da função Sigmóide

A função de ativação Sigmóide é comumente usada em redes neurais devido às suas propriedades de suavização e transformação não linear [1, 5]. Essa função mapeia a entrada em um intervalo entre 0 e 1, o que é útil para a modelagem de probabilidades [7, 10]. Além disso, a Sigmóide facilita a interpretação dos resultados em tarefas de classificação binária, onde as saídas podem ser interpretadas como probabilidades de pertencer a uma determinada classe [2, 6].

Da mesma forma que a função ReLU, para compreender o comportamento dessa função, a Figura 3 foi criada utilizando-se da inteligência artificial proposta pelo artigo.

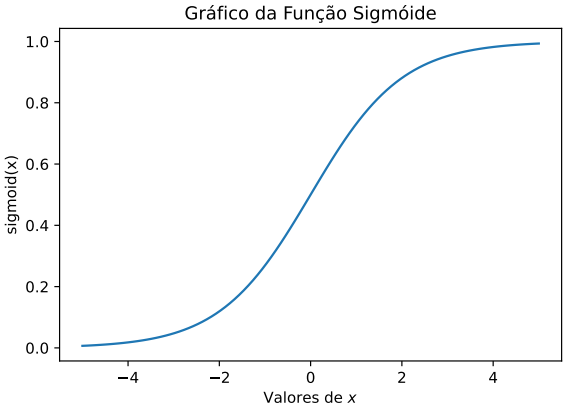

Figura 3: Como a função Sigmóide testada com valores do MNIST.

A expressão matemática para a função Sigmóide é dada por:

- **Fórmula**: $f(x) = \frac{1}{1+e^{-x}}$
- **Domínio**: $\mathbb{R}$
- **Imagem**: $(0,1)$

a. Derivada da função Sigmóide

Considerando a função Sigmóide $f(x) = \frac{1}{1+e^{-x}}$, a sua derivada em relação a x, é da seguinte forma:

$$f'(x) = \frac{0.(1+e^{-x}) - 1.(1+e^{-x})'}{(1+e^{-x})^2} = \frac{-(e^{-x}.(-1))}{(1+e^{-x})^2} \quad (1)$$

Segue daí que:

$$f'(x) = \frac{e^{-x}}{(1+e^{-x})^2} \quad (2)$$

Finalmente, expressamos a derivada em termos da própria função Sigmóide:

$$f'(x) = f(x) \cdot (1 - f(x)) \quad (3)$$

b. Aplicação de Sigmóide em redes neurais

A função de ativação Sigmóide destacou-se como a menos eficiente na classificação dos dígitos do dataset MNIST

utilizando a arquitetura LeNet-5. Durante o treinamento, ela alcançou uma precisão final de 96,60% no conjunto de teste, com uma perda de 0.1131. Foi gerada a matriz de confusão para ela na Figura 4, que permite visualizar a distribuição dos erros de classificação entre as diferentes classes.

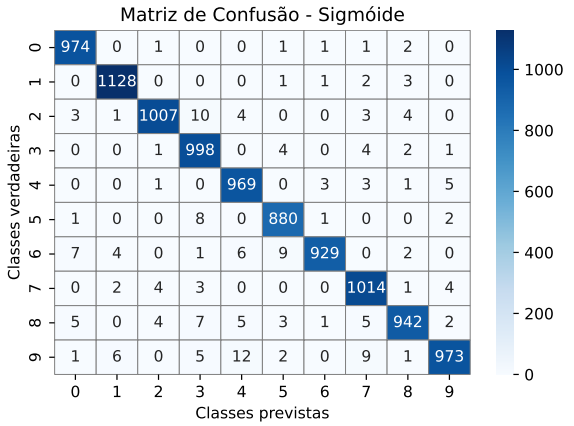

Figura 4: Matriz de confusão testada com valores do MNIST.

V. Definição e propriedades da função TanH

A função tangente hiperbólica (tanh) é uma função de ativação comum em redes neurais, definida matematicamente por:

- **Fórmula**: $f(x) = \tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
- **Domínio**: $\mathbb{R}$
- **Imagem**: $(-1, 1)$

Esta função mapeia os valores de entrada para o intervalo entre -1 e 1, o que é bom para a modelagem de características não lineares em modelos de aprendizado de máquina [1], [5].

Da mesma forma das outras funções, para compreender o comportamento dessa função, a Figura 5 foi criada utilizando-se da inteligência artificial proposta pelo artigo.

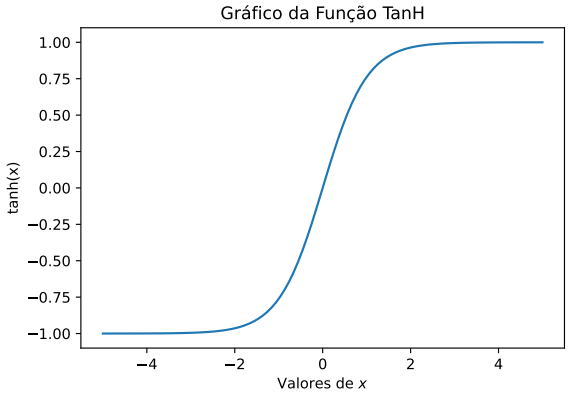

Figura 5: Como a função Tahn testada com valores do MNIST.

a. Derivada da função TanH

A derivada, usando a regra da divisão do quociente, pode ser expandida como:

$$f'(x) = \frac{(e^x + e^{-x})(e^x - e^{-x})' - (e^x - e^{-x})(e^x + e^{-x})'}{(e^x + e^{-x})^2} \quad (1)$$

$$f'(x) = \frac{(e^x + e^{-x})(e^x + e^{-x}) - (e^x - e^{-x})(e^x - e^{-x})}{(e^x + e^{-x})^2} \quad (2)$$

$$f'(x) = \frac{(e^x + e^{-x})^2 - (e^x - e^{-x})^2}{(e^x + e^{-x})^2} \quad (3)$$

E portanto, obtemos que:

$$f'(x) = \frac{(e^x + e^{-x})^2 - (e^x - e^{-x})^2}{(e^x + e^{-x})^2} = 1 - \tanh^2(x) \quad (4)$$

E assim a derivada de $\tanh(x)$ em relação a x é crucial para o processo de retropropagação em redes neurais e é dada por:

$$f'(x) = 1 - \tanh^2(x) \quad (5)$$

b. Aplicação da função TanH em redes neurais

A função de ativação TanH ficou entre as duas anteriores em termos de eficiência na classificação dos dígitos do dataset MNIST utilizando a arquitetura LeNet-5. Durante o treinamento, ela alcançou uma precisão final de 98,54% no conjunto de teste, com uma perda de 0.0501. Foi gerada a matriz de confusão para ela na Figura 6, que permite visualizar a distribuição dos erros de classificação entre as diferentes classes:

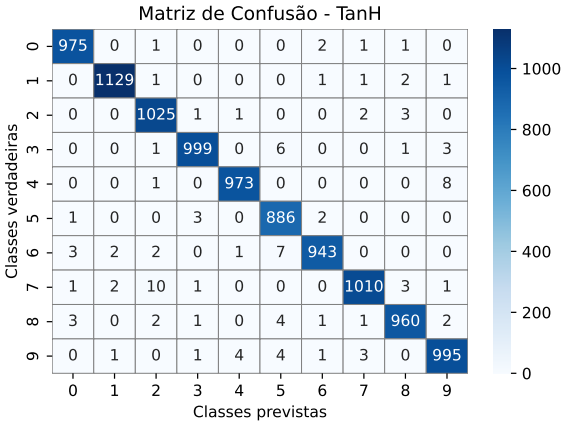

Figura 6: Função Tahn testada com valores do MNIST.

VI. Definição e propriedades da função SoftMax

A função Softmax também é utilizada para interpretar os resultados como probabilidades,mas não é convencionalmente uma função de ativação, pois é mais utilizada para normalizar as saídas dos neurônios. Isto é especialmente útil em problemas de classificação onde precisamos determinar a probabilidade de uma instância pertencer a uma das várias classes possíveis, ao contrário da função Sigmóide [1, 7]. A função Softmax transforma as pontuações (também conhecidas como logits) produzidas pelos modelos

ISSN: 2675-3588

de aprendizado de máquina em probabilidades que somam 1, facilitando a interpretação dos resultados [1], [5]. Essa propriedade é fundamental para tarefas como classificação multiclasse, onde cada classe é mutuamente exclusiva, permitindo que o modelo selecione a classe mais provável para uma dada entrada [7].

Da mesma forma que a função anterior, para melhor compreender o comportamento dessa função, utilizando os dados do modelo testado, foi feita a Figura 7:

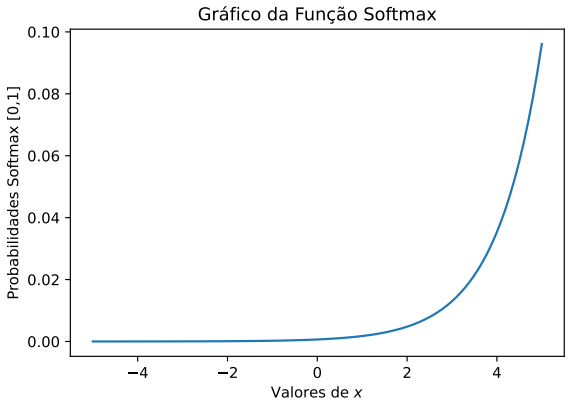

Figura 7: Função Softmax testada com valores do MNIST.

a. Conceito matemático da função SoftMax

A função Softmax é uma função matemática que transforma um vetor de números reais em um vetor de probabilidades, onde a soma total é 1. Ela é utilizada em modelos de classificação multiclasse em aprendizado de máquina, especialmente em redes neurais.

Definição Formal:

Dado um vetor de entrada $\mathbf{z} = (z_1, z_2, \ldots, z_K) \in \mathbb{R}^K$, a função Softmax é definida para cada componente z_i como:

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

onde:

- e é denominado número de Euler.
- K é o número de classes ou categorias.

Domínio e Contradomínio:

- **Domínio**: $\mathbb{R}^K$, o espaço de todos os vetores reais de dimensão K.
- **Contradomínio**: Vetores $\mathbf{p} = (p_1, p_2, \ldots, p_K)$ tais que $p_i \in (0,1)$ e $\sum_{i=1}^{K} p_i = 1$.

Propriedades Importantes:

1. Normalização:

$$\sum_{i=1}^{K} \text{Softmax}(z_i) = 1$$

Isso significa que a saída da função Softmax forma uma distribuição de probabilidade sobre as classes.

2. Positividade:

$$\text{Softmax}(z_i) > 0 \quad \text{para todo } z_i \in \mathbb{R}$$

3. Monotonicidade Relativa:
A ordem relativa das entradas é preservada nas saídas. Se $z_i > z_j$, então $\text{Softmax}(z_i) > \text{Softmax}(z_j)$.

b. Derivados da função SoftMax

Vamos considerar duas derivadas principais da função softmax: a derivada em relação ao mesmo elemento ($i = j$) e a derivada em relação a outro elemento ($i \neq j$).

c. Derivada em relação ao mesmo elemento (i = j)

Queremos calcular a derivada da função softmax em relação a z_i:

$$\frac{\partial}{\partial z_i}(\text{softmax}(z_i)) = \frac{\partial}{\partial z_i}\left(\frac{e^{z_i}}{\sum_k e^{z_k}}\right) \tag{1}$$

Aplicamos a regra do quociente, onde:

- $f(z_i) = e^{z_i}$
- $g(z_i) = \sum_k e^{z_k}$

As derivadas são:

- $f'(z_i) = e^{z_i}$
- $g'(z_i) = \frac{\partial}{\partial z_i}(\sum_k e^{z_k}) = e^{z_i}$

Aplicando a regra do quociente:

$$\frac{\partial}{\partial z_i}\left(\frac{f(z_i)}{g(z_i)}\right) = \frac{f'(z_i)\cdot g(z_i) - f(z_i)\cdot g'(z_i)}{[g(z_i)]^2} \tag{2}$$

$$\tag{3}$$

$$= \frac{e^{z_i}\cdot(\sum_k e^{z_k}) - e^{z_i}\cdot e^{z_i}}{(\sum_k e^{z_k})^2} \tag{4}$$

$$\tag{5}$$

$$= \frac{e^{z_i}(\sum_k e^{z_k} - e^{z_i})}{(\sum_k e^{z_k})^2} \tag{6}$$

Reconhecendo que:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_k e^{z_k}} \tag{7}$$

Portanto:

$$\frac{\partial}{\partial z_i}(\text{softmax}(z_i)) = \text{softmax}(z_i)\cdot\left(\frac{\sum_k e^{z_k} - e^{z_i}}{\sum_k e^{z_k}}\right) \tag{8}$$

$$= \text{softmax}(z_i)\cdot(1 - \text{softmax}(z_i)) \tag{9}$$

d. Derivada em relação a um outro elemento $(i \neq j)$

Agora, calculamos a derivada de $\text{softmax}(z_i)$ em relação a z_j, onde $i \neq j$:

$$\frac{\partial}{\partial z_j}\left(\text{softmax}(z_i)\right) = \frac{\partial}{\partial z_j}\left(\frac{e^{z_i}}{\sum_k e^{z_k}}\right) \tag{1}$$

Aplicando novamente a regra do quociente. Neste caso:

- $f(z_j) = e^{z_i}$ (não depende de z_j, logo $f'(z_j) = 0$)
- $g(z_j) = \sum_k e^{z_k}$
- $g'(z_j) = \frac{\partial}{\partial z_j}\left(\sum_k e^{z_k}\right) = e^{z_j}$

Aplicando a regra:

$$\frac{\partial}{\partial z_j}\left(\frac{f(z_j)}{g(z_j)}\right) = \frac{f'(z_j)\cdot g(z_j) - f(z_j)\cdot g'(z_j)}{[g(z_j)]^2} \tag{2}$$

$$= \frac{0\cdot\left(\sum_k e^{z_k}\right) - e^{z_i}\cdot e^{z_j}}{\left(\sum_k e^{z_k}\right)^2} \tag{3}$$

$$= -\frac{e^{z_i}\cdot e^{z_j}}{\left(\sum_k e^{z_k}\right)^2} \tag{4}$$

Novamente, reconhecendo as expressões de softmax:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_k e^{z_k}}$$

$$\text{softmax}(z_j) = \frac{e^{z_j}}{\sum_k e^{z_k}}$$

Portanto:

$$\frac{\partial}{\partial z_j}\left(\text{softmax}(z_i)\right) = -\text{softmax}(z_i)\cdot\text{softmax}(z_j) \tag{5}$$

e. Exemplo prático

Seja $\mathbf{z} = (2,1,0)$ um vetor do $\mathbb{R}^3$

1. Subtraímos o máximo (2) de cada componente para estabilidade:

$$z_{\max} = 2 \quad \Rightarrow \quad \mathbf{z}' = (0,-1,-2) \tag{1}$$

2. Calculamos as exponenciais:

$$e^0 = 1 \quad e^{-1} \approx 0{,}3679 \quad e^{-2} \approx 0{,}1353 \tag{2}$$

3. Calculamos a soma:

$$S = 1 + 0{,}3679 + 0{,}1353 \approx 1{,}5032 \tag{3}$$

4. Calculamos as probabilidades:

$$\text{Softmax}(z_1) = \frac{1}{1{,}5032} \approx 0{,}6652 \tag{4}$$

$$\text{Softmax}(z_2) = \frac{0{,}3679}{1{,}5032} \approx 0{,}2447 \tag{5}$$

$$\text{Softmax}(z_3) = \frac{0{,}1353}{1{,}5032} \approx 0{,}0900 \tag{6}$$

As probabilidades resultantes indicam a propensão de cada classe.

VII. Função de custo quadrática (erro quadrático médio)

Em Inteligência Artificial, especialmente em aprendizado de máquina e aprendizado profundo, a função de custo (também conhecida como função de perda ou função objetivo) é uma medida que quantifica a diferença entre as previsões do modelo e os valores reais esperados(Goodfellow et al., 2016; Bishop, 2006). Ela serve como um guia para o processo de treinamento do modelo, indicando o quão bem o modelo está performando e como ele pode ser melhorado. O objetivo para melhorar a predição do modelo é minimizar essa função de custo, uma vez que ao fazer isso, a diferença entre o valor previsto pela IA e o valor real se torna mínima.

Existem várias funções de custo, esse artigo abordará a quadrática:

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(\text{predição}_i - \text{valor real}_i\right)^2$$

onde:

- m: é o número de exemplos.
- predição_i: é a previsão do modelo para o i-ésimo exemplo.
- valor real_i: é o valor verdadeiro do i-ésimo exemplo.

Com variáveis:

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

onde $h_\theta(x)$ é a função do modelo, $x^{(i)}$ são as entradas de treinamento e $y^{(i)}$ são as saídas.

Ao calcular a diferença entre o valor real e o previsto $(h_\theta(x^{(i)}) - y^{(i)})$, obtemos o erro para cada exemplo. Elevando esse erro ao quadrado, garantimos que todos os valores sejam positivos, o que evita que erros negativos e positivos se cancelem ao somar. Além disso, o erro ao quadrado penaliza erros maiores de forma mais significativa, incentivando o modelo a minimizar grandes discrepâncias.[1].

O coeficiente $\frac{1}{2}$ é incluído por conveniência matemática. Ao derivar a função de custo durante o processo de otimização (por exemplo, usando gradiente descendente), o expoente 2 do termo quadrático cancela com o $\frac{1}{2}$, simplificando os cálculos das derivadas.

Dividindo a soma total pelo número de exemplos n, obtemos a média dos erros quadráticos. Isso normaliza o custo, permitindo comparações consistentes, independentemente do tamanho do conjunto de dados.

VIII. Método dos mínimos quadrados

O método dos mínimos quadrados é uma técnica estatística utilizada para encontrar a melhor aproximação linear entre um conjunto de dados observados (Montgomery et al., 2012). Por exemplo, pode ser utilizado para minimizar a diferença entre os valores previstos pelo modelo e os valores reais observados, ou seja, o erro do modelo (Draper & Smith, 1998). No caso da IA, o objetivo da utilização do método

ISSN: 2675-3588

dos mínimos quadrados seria minimizar a função de custo discutida anteriormente.

a. Definição

Dado um conjunto de n pontos (x_i, y_i), $i = 1, 2, 3, \ldots, n$, queremos encontrar os parâmetros a e b para a reta $r(x) = a + bx$ de tal forma que minimize a soma das distâncias verticais (d) ao quadrado entre cada ponto y_i e a reta ajustada. Essa soma das distâncias ao quadrado é representada por:

$$M(a,b) = \sum_{i=1}^{n} [d]^2 \tag{1}$$

Que pode ser substituído por:

$$M(a,b) = \sum_{i=1}^{n} [y_i - (a + bx_k)]^2 \tag{2}$$

onde $[y_i - (a+bx_i)]^2$ representa a distância ao quadrado entre o valor observado y_i e o valor estimado $r(x_i)$ na reta.

b. Expansão da função objetivo

Substituímos d_i por $y_i - (a+bx_i)$. Assim, a função $M(a,b)$ é dada por:

$$M(a,b) = \sum_{i=1}^{n} (y_i - a - bx_i)^2 \tag{3}$$

c. Derivação para encontrar o ponto crítico

Para minimizar $M(a,b)$, precisamos encontrar os valores de a e b que tornam $M(a,b)$ mínimo. Segundo o método, isso é feito tomando as derivadas parciais de $M(a,b)$ em relação a a e b e igualando a zero.

Derivada em Relação a a

$$\frac{\partial M}{\partial a} = -2 \sum_{i=1}^{n} (y_i - a - bx_i) \tag{4}$$

Igualando a zero para minimizar:

$$\sum_{i=1}^{n} (y_i - a - bx_i) = 0 \tag{5}$$

Derivada em Relação a b

$$\frac{\partial M}{\partial b} = -2 \sum_{i=1}^{n} x_i (y_i - a - bx_i) \tag{6}$$

Igualando a zero para minimizar:

$$\sum_{i=1}^{n} x_i (y_i - a - bx_i) = 0 \tag{7}$$

Essas duas equações formam um sistema de equações lineares nos parâmetros a e b. Podemos resolver esse sistema para encontrar os valores de a e b que minimizam $M(a,b)$.

A minimização se dá ao derivar $S(a,b)$ em relação a a e b utilizando a regra da cadeia e, então, igualar a zero:

$$\frac{\partial S}{\partial a} = \frac{\partial S}{\partial x} \cdot \frac{\partial x}{\partial a} \tag{8}$$

$$\frac{\partial S}{\partial x} = 2 \sum_{i=1}^{n} (y_i - a - bx_i) \cdot (-1) \tag{9}$$

$$\frac{\partial S}{\partial a} = -2 \sum_{i=1}^{n} (y_i - a - bx_i) = 0 \tag{10}$$

$$\frac{\partial S}{\partial b} = -2 \sum_{i=1}^{n} x_i (y_i - a - bx_i) = 0 \tag{11}$$

Distribuindo e dividindo a primeira expressão por $2n$ temos:

$$-2 \sum_{i=1}^{n} y_i + 2 \sum_{i=1}^{n} a + 2 \sum_{i=1}^{n} bx_i = 0 \tag{12}$$

$$\frac{-2\sum_{i=1}^{n} y_i}{2n} + \frac{2\sum_{i=1}^{n} a}{2n} + \frac{2\sum_{i=1}^{n} bx_i}{2n} = \frac{0}{2n} \tag{13}$$

$$-\frac{\sum_{i=1}^{n} y_i}{n} + a + b\frac{\sum_{i=1}^{n} x_i}{n} = 0 \tag{14}$$

$$-\bar{y} + a + b\bar{x} = 0 \tag{15}$$

$$\bar{y} - b\bar{x} = a \tag{16}$$

onde $\bar{y}$ é a média amostral de y e $\bar{x}$ é a média amostral de x.

Substituindo esse resultado na segunda expressão, temos:

$$-2 \sum_{i=1}^{n} x_i (y_i - \bar{y} + b\bar{x} - bx_i) = 0 \tag{17}$$

$$\sum_{i=1}^{n} [x_i (y_i - \bar{y}) + x_i b(\bar{x} - x_i)] = 0 \tag{18}$$

$$\sum_{i=1}^{n} x_i (y_i - \bar{y}) + b \sum_{i=1}^{n} x_i (\bar{x} - x_i) = 0 \tag{19}$$

$$\frac{\sum_{i=1}^{n} x_i (y_i - \bar{y})}{\sum_{i=1}^{n} x_i (x_i - \bar{x})} = b \tag{20}$$

Existe uma fórmula diferente que gera o mesmo resultado:

$$b = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2} \tag{21}$$

IX. GRADIENTE DESCENDENTE

Sobre o método do gradiente descendente, é uma técnica de otimização utilizada em aprendizado de máquina e estatística para minimizar funções (Goodfellow et al., 2016; Bishop, 2006). A ideia é ir na direção inversa da apontada pelo gradiente, uma vez que essa aponta para o máximo da função (Haykin, 2009). Vamos detalhar esse método em termos matemáticos, que foi o tema inicial do estudo. Será ilustrado o método, e colocado um exemplo.

a. Inicialização

Comece com valores iniciais para os parâmetros $\theta = [\theta_1, \theta_2, \ldots, \theta_n]$.

b. Função de custo

Utilizando a função de custo:

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

onde $h_\theta(x)$ é a função do modelo, $x^{(i)}$ são as entradas de treinamento e $y^{(i)}$ são as saídas.

c. Cálculo do gradiente

O gradiente da função de custo em relação aos parâmetros θ é um vetor de derivadas parciais:

$$\nabla J(\theta) = \left[\frac{\partial J(\theta)}{\partial \theta_1}, \frac{\partial J(\theta)}{\partial \theta_2}, \ldots, \frac{\partial J(\theta)}{\partial \theta_n}\right] \tag{1}$$

Cada componente do gradiente é calculado como:

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)} \tag{2}$$

d. Atualização dos parâmetros

Atualize os parâmetros na direção oposta ao gradiente:

$$\theta_j := \theta_j - \alpha\frac{\partial J(\theta)}{\partial \theta_j} \tag{3}$$

onde α é a taxa de aprendizado.

e. Repetição

Repita os passos 2 e 3 até que a função de custo convirja, ou seja, até que as mudanças nos valores de θ sejam menores que um determinado limiar.

X. EXEMPLO DE GRADIENTE DESCENDENTE

Para ilustrar o método do gradiente descendente, vamos aplicar os conceitos apresentados a um exemplo prático de regressão linear simples. Suponha que desejamos ajustar uma reta a um conjunto de dados para prever valores futuros.

a. Dados de exemplo

Considere o seguinte conjunto de dados:

x	y
1	2
2	4
3	6
4	8

(1)

Nosso objetivo é encontrar os parâmetros θ_0 e θ_1 da reta $h_\theta(x) = \theta_0 + \theta_1 x$ que melhor se ajustam aos dados.

b. Inicialização

Inicializamos os parâmetros com valores arbitrários:

$$\theta_0 = 0, \quad \theta_1 = 0 \tag{2}$$

c. Uso da função de custo

Utilizamos a função de custo de mínimos quadrados:

$$J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 \tag{3}$$

d. Cálculo do gradiente

Calculamos as derivadas parciais da função de custo em relação a θ_0 e θ_1:

$$\frac{\partial J}{\partial \theta_0} = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right) \tag{4}$$

$$\frac{\partial J}{\partial \theta_1} = \frac{1}{m}\sum_{i=1}^{m}\left(\left(h_\theta(x^{(i)}) - y^{(i)}\right)x^{(i)}\right) \tag{5}$$

e. Atualização dos parâmetros

Atualizamos os parâmetros na direção oposta ao gradiente:

$$\theta_0 := \theta_0 - \alpha\frac{\partial J}{\partial \theta_0} \tag{6}$$

$$\theta_1 := \theta_1 - \alpha\frac{\partial J}{\partial \theta_1} \tag{7}$$

Escolhemos uma taxa de aprendizado $\alpha = 0{,}01$.

f. Iterações do gradiente descendente

Realizaremos três iterações para observar o ajuste dos parâmetros.

Iteração 1

Passo 1: Calculamos as previsões iniciais:

$$h_\theta(x^{(i)}) = \theta_0 + \theta_1 x^{(i)} = 0 \tag{8}$$

Passo 2: Calculamos os erros:

i	$h_\theta(x^{(i)})$	Erro $= h_\theta(x^{(i)}) - y^{(i)}$
1	0	-2
2	0	-4
3	0	-6
4	0	-8

(9)

Passo 3: Calculamos as derivadas:

$$\frac{\partial J}{\partial \theta_0} = \frac{1}{4}(-2-4-6-8) = -5 \tag{10}$$

$$\frac{\partial J}{\partial \theta_1} = \frac{1}{4}((-2)(1)+(-4)(2)+(-6)(3)+(-8)(4)) = -15 \tag{11}$$

Passo 4: Atualizamos os parâmetros:

$$\theta_0 = 0 - 0{,}01 \times (-5) = 0{,}05 \tag{12}$$

$$\theta_1 = 0 - 0{,}01 \times (-15) = 0{,}15 \tag{13}$$

ISSN: 2675-3588

1. Iteração 2

Passo 1: Novas previsões:

$$h_\theta(x^{(i)}) = 0{,}05 + 0{,}15x^{(i)} \tag{14}$$

Passo 2: Calculamos os erros:

i	$h_\theta(x^{(i)})$	Erro
1	0,20	−1,80
2	0,35	−3,65
3	0,50	−5,50
4	0,65	−7,35

(15)

Passo 3: Calculamos as derivadas:

$$\frac{\partial J}{\partial \theta_0} = \frac{1}{4}(-1{,}80 - 3{,}65 - 5{,}50 - 7{,}35) \tag{16}$$

$$\frac{\partial J}{\partial \theta_0} = \frac{1}{4}(-18{,}30) \tag{17}$$

$$\frac{\partial J}{\partial \theta_0} = -4{,}575 \tag{18}$$

$$\frac{\partial J}{\partial \theta_1} = \frac{1}{4}(-1{,}80 + (-3{,}65)(2) + (-5{,}50)(3) + (-7{,}35)(4)) \tag{19}$$

$$\frac{\partial J}{\partial \theta_1} = \frac{1}{4}(-1{,}80 - 7{,}30 - 16{,}50 - 29{,}40) \tag{20}$$

$$\frac{\partial J}{\partial \theta_1} = \frac{1}{4}(-55{,}00) \tag{21}$$

$$\frac{\partial J}{\partial \theta_1} = -13{,}75 \tag{22}$$

Passo 4: Atualizamos os parâmetros:

$$\theta_0 = 0{,}05 - 0{,}01 \times (-4{,}575) = 0{,}09575 \tag{23}$$

$$\theta_1 = 0{,}15 - 0{,}01 \times (-13{,}75) = 0{,}2875 \tag{24}$$

2. Iteração 3

Passo 1: Novas previsões:

$$h_\theta(x^{(i)}) = 0{,}09575 + 0{,}2875x^{(i)} \tag{25}$$

Passo 2: Calculamos os erros:

i	$h_\theta(x^{(i)})$	Erro
1	0,38325	−1,61675
2	0,67075	−3,32925
3	0,95825	−5,04175
4	1,24575	−6,75425

(26)

Passo 3: Calculamos as derivadas:

$$\frac{\partial J}{\partial \theta_0} = \frac{1}{4}(-1{,}61675 - 3{,}32925 - 5{,}04175 - 6{,}75425) \tag{27}$$

$$= -4{,}1855 \tag{28}$$

$$\frac{\partial J}{\partial \theta_1} = \frac{1}{4}\big((-1{,}61675)(1) + (-3{,}32925)(2) \tag{29}$$

$$+ (-5{,}04175)(3) + (-6{,}75425)(4)\big) \tag{30}$$

$$= -12{,}604375 \tag{31}$$

Passo 4: Atualizamos os parâmetros:

$$\theta_0 = 0{,}09575 - 0{,}01 \times (-4{,}1855) = 0{,}137605 \tag{32}$$

$$\theta_1 = 0{,}2875 - 0{,}01 \times (-12{,}604375) = 0{,}41354375 \tag{33}$$

g. Convergência

Continuando esse processo por várias iterações, os parâmetros θ_0 e θ_1 convergirão para os valores que minimizam a função de custo. No nosso exemplo, como os dados seguem uma relação linear perfeita $y = 2x$, os valores ideais são $\theta_0 = 0$ e $\theta_1 = 2$.

h. Visualização

Utilizando duas classes de dígitos selecionadas do conjunto de dados MNIST (neste caso, os números 2 e 7, a título de exemplo), foi gerado um gráfico que apresenta uma superfície tridimensional, e ele representa a função de custo em relação aos dois primeiros parâmetros do modelo de regressão logística.

A superfície exibe a variação da função de custo conforme os valores dos parâmetros (Theta 0 e Theta 1) mudam, permitindo uma análise visual do comportamento da otimização. A coloração no gráfico, obtida a partir do mapa de cores viridis, destaca as regiões de maior e menor custo, onde áreas mais elevadas indicam valores de erro maiores, enquanto áreas mais baixas representam regiões de menor erro.

Além disso, um ponto vermelho destacado no gráfico representa o mínimo encontrado pela descida do gradiente, evidenciando a convergência do algoritmo para a melhor solução dentro do espaço de parâmetros. Esse ponto é acompanhado por um contorno preto que enfatiza sua posição na superfície. Dessa forma, o gráfico auxilia na compreensão do processo de ajuste dos pesos do modelo, demonstrando como a otimização ocorre para minimizar a função de custo e melhorar a capacidade preditiva da regressão logística.

XI. Considerações finais

Este estudo evidenciou, de maneira detalhada, o papel central da matemática no desenvolvimento da Inteligência Artificial (IA), ressaltando conceitos fundamentais como funções de ativação, o método do gradiente descendente e os mínimos quadrados. A análise destacou que funções de ativação, como ReLU, Sigmóide, TanH e Softmax, são determinantes para a introdução de não-linearidades nas redes neurais. Essa característica permite que essas estruturas matemáticas reconheçam padrões complexos e ofereçam soluções sofisticadas para uma ampla variedade de problemas. Este

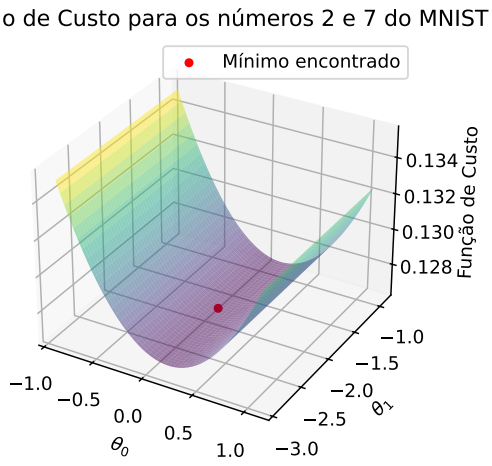

Figura 8: Plano do ajuste dos dados utilizando gradiente descendente

estudo evidenciou, de maneira detalhada, o papel central da matemática no desenvolvimento da Inteligência Artificial (IA), ressaltando conceitos fundamentais como funções de ativação, o método do gradiente descendente e os mínimos quadrados. A análise destacou que funções de ativação, como ReLU, Sigmóide, TanH e Softmax, são determinantes para a introdução de não-linearidades nas redes neurais. Essa característica permite que essas estruturas matemáticas reconheçam padrões complexos e ofereçam soluções sofisticadas para uma ampla variedade de problemas. A otimização baseada em gradientes também foi enfatizada como uma das principais técnicas de treinamento de redes neurais, demonstrando como o cálculo do gradiente da função de custo viabiliza ajustes iterativos precisos nos pesos e vieses dos modelos. Esse processo minimiza o erro, acelera a convergência e melhora o desempenho geral dos algoritmos. Além disso, o estudo destacou a importância do método dos mínimos quadrados, ferramenta estatística em modelos preditivos que requerem alta precisão. Embora esses conceitos matemáticos sejam cruciais para o avanço da IA, são raros os artigos que exploram profundamente essa base teórica, frequentemente focando apenas nos resultados práticos ou técnicos. No entanto, estudar mais a fundo essa base matemática oferece vantagens significativas, pois permite a criação de modelos mais robustos, eficientes e inovadores. Assim, ficou evidente que a matemática não apenas melhora o desempenho dos algoritmos, mas também expande o potencial de criação de soluções tecnológicas. Por fim, o estudo reforça que o progresso contínuo da IA depende de uma sólida compreensão matemática e de esforços interdisciplinares. Ao aprofundar o conhecimento nessa área, é possível não apenas elevar o desempenho técnico das redes neurais, mas também maximizar o impacto transformador dessas tecnologias na sociedade contemporânea.

REFERÊNCIAS

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA: MIT Press, 2016.

[2] F. Rosenblatt, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. New York: Spartan Books, 1986.

[3] N. R. Draper and H. Smith, *Applied Regression Analysis*, 3rd ed. New York: Wiley, 1998.

[4] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to Linear Regression Analysis*, 5th ed. New York: Wiley, 2012.

[5] S. Haykin, *Neural Networks and Learning Machines*. Pearson Education: Pearson, 2009.

[6] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[7] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York: Springer, 2006.

[8] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010. [Online]. Available: https://www.cs.toronto.edu/~hinton/absps/reluICML.pdf

[9] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Proceedings of the 3rd International Conference on Neural Information Processing Systems*, ser. NIPS'89. Cambridge, MA, USA: MIT Press, 1989, p. 396–404.

[10] M. A. Nielsen, *Neural Networks and Deep Learning*. Available online: Determination Press, 2015. [Online]. Available: http://neuralnetworksanddeeplearning.com/

ISSN: 2675-3588