De 07/10/2020 a 09/10/2020 ISSN - 2526-2769

DESENVOLVIMENTO DE UMA REDE NEURAL FEEDFORWARD GENÉRICA COM LINGUAGEM PYTHON

PENNING, Caio Henrique[1*]; ANGELIN, Igor[1]; AMADORI, Gabriel dos Santos[1]; SHUBEITA, Fauzi de Moraes[1]

[1] FAHOR, Curso de Engenharia de Controle e Automação, Campus Arnoldo Schneider, Avenida dos Ipês, 565, Horizontina, RS, Brasil.

*Autor Correspondente: cp002516@fahor.com.br

RESUMO

As redes neurais são sistemas de computação interconectados que funcionam de forma análoga a neurônios do cérebro humano. Utilizando-se de algoritmos, elas podem reconhecer padrões e realizar correlações em dados brutos, agrupá-los e classificá-los de forma a haver um aprendizado com o avanço de sua inteligência. Inicialmente as redes neurais eram algoritmos de uso exclusivo de organizações militares, universidades, grandes corporações e mercado financeiro. Contudo, atualmente sua implantação em larga escala e nos mais diversos setores vêm melhorando exponencialmente a análise das informações coletadas para os mais diversos fins. Levando-se em consideração o crescimento da importância de seu uso no meio acadêmico e industrial, buscou-se realizar o desenvolvimento de rede neural genérica do tipo feedforward através da linguagem de computação Python. Pode-se obter que o desenvolvimento de uma rede neural genérica é simples e pode agregar muito conhecimento aos envolvidos na programação.

Palavras-chave: Redes Neurais, Feedforward, Python, Sigmóide, Machine Learning.

DEVELOPMENT OF A GENERIC FEEDFORWARD NEURAL NETWORK WITH PYTHON LANGUAGE

ABSTRACT

Neural networks are interconnected computer systems that work in the same way as neurons in the human brain. Using algorithms, they can recognize patterns and correlate raw data, grouping and classifying them,so that there is a learning with the advance of their intelligence. Initially, neural networks were algorithms used exclusively by military organizations, universities, big corporations and financial markets. However, currently its implementation on a large scale and in the most diverse sectors has been exponentially improving the analysis of the information collected for the most diverse purposes. Considering the growing importance of its use in the academic and industrial context, the development of a generic neural network of the feedforward type was sought through the Python computing language.The development of a generic neural network is simple and can add a lot of knowledge to those involved in programming.

Keywords: Neural Network, Feedforward, Python, Sigmoid, Machine Learning.

1 INTRODUÇÃO

Segundo Russell et. al. (2018), "*Machine Learning"* é um método de programação que aprende a partir dos seus dados gerados. Como exemplo, o programa poderia distinguir se algo é importante ou "spam". Em *"Machine Learning"* os dados se referem a valores que retornam de treinamentos com valores pré-determinados chamados pesos.

Redes neurais artificiais são métodos para solucionar determinado problema se utilizando da tecnologia de inteligência artificial. Desse modo é mais fácil imaginar um computador que aprende sozinho, que se baseia em suas atividades e em seus erros para desenvolver soluções e se desenvolver, do que programa-lo e não estar preparado para possíveis falhas ou alteração de suas funções. Essa é a capacidade de uma rede neural, parece ficção cientifica, mas é uma tecnologia emergente que vem encontrando aplicações concretas. (BARRETO, 2002)

O desenvolvimento projeto tem como finalidade criar uma rede neural genérica, utilizando a linguagem "Python" e sua biblioteca "NumPy", onde o programa desempenhara somente o método "Feedforward", sendo assim os dados não serão realimentados, e os valores gerados serão aleatórios. O objetivo disso é demonstrar como pode ser simples programar essa tecnologia em poucas linhas de código, e que com mais alguns passos possam ser aplicados a alguns casos reais.

2 DESENVOLVIMENTO E DEMONSTRAÇÃO DOS RESULTADOS

2.1 REFERENCIAL TEÓRICO

O referencial teórico adicionado faz referência as Redes Neurais e suas funcionalidades, a descrição do modelo de funcionamento *Feedforward* e a linguagem Python e as bibliotecas necessárias para a implementação da mesma.

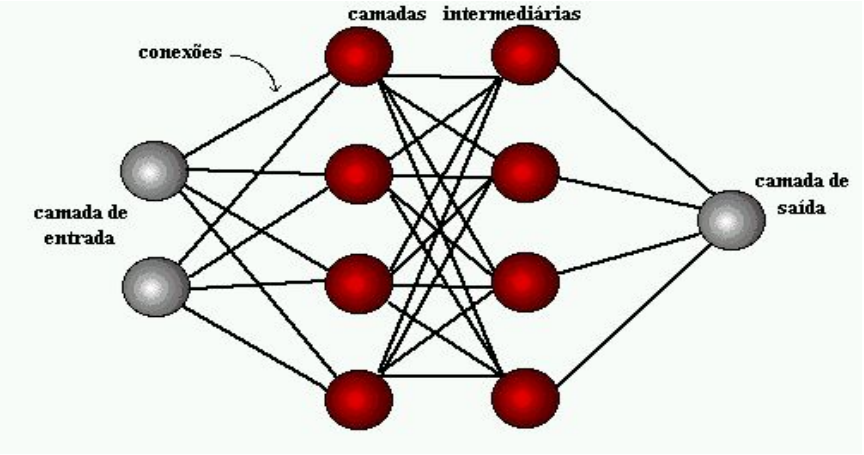
2.1.1 Redes Neurais

Rede neural é uma técnica para criar um programa de computador que aprende com os dados. Primeiro, uma coleção de "neurônios" de *software* é criada e conectada, permitindo que eles enviem mensagens um para o outro. Em seguida, solicita-se à rede que resolva um problema, o que ela tenta resolver repetidamente, sempre fortalecendo as conexões que levam ao sucesso e diminuindo as que levam ao fracasso.

Carvalho (2020) afirma que usualmente as camadas das redes neurais artificiais são classificadas em três grupos, observados na Figura 1:

- Camada de Entrada: onde os padrões são apresentados à rede;
- Camadas Intermediárias ou Escondidas: onde é feita a maior parte do processamento, através das conexões ponderadas; podem ser consideradas como extratoras de características;
- Camada de Saída: onde o resultado final é concluído e apresentado.

Figura 1 - Camadas de uma rede neural



Fonte: CARVALHO, 2020.

Carvalho (2020) declara que a propriedade mais importante das redes neurais é a habilidade de aprender de seu ambiente e com isso melhorar seu desempenho. O treinamento

é feito através de um processo iterativo de ajustes aplicado a seus pesos. O aprendizado ocorre quando a rede neural atinge uma solução generalizada para uma classe de problemas.

Denomina-se algoritmo de aprendizado a um conjunto de regras bem definidas para a solução de um problema de aprendizado. Existem muitos tipos de algoritmos de aprendizado específicos para determinados modelos de redes neurais, estes algoritmos diferem entre si principalmente pelo modo como os pesos são modificados.

2.1.2 Algoritmo *Feedforward*

De acordo com Araújo (2020), uma rede de algoritmo *feedforward* é uma espécie de rede dividida em camadas, cada camada se conecta à próxima camada, porém não há caminho de volta. Todas as conexões, portanto, têm a mesma direção, partindo da camada de entrada rumo a camada de saída.

2.1.3 Python

A linguagem de programação Python foi desenvolvida no final da década de 1980 por Guido van Rossum, um programador holandês enquanto trabalhava no CWI (*CentrumWiskunde&Informatica, em Amsterdã, Holanda*). De acordo com Perkovic (2016), o Python é uma linguagem de uso geral, projetada especificamente para tornar os programas bastante legíveis. Python também possui uma rica biblioteca, tornando possível criar aplicações sofisticadas usando código de aparência relativamente simples. Por esses motivos, Python tornou-se uma linguagem de desenvolvimento de aplicações popular e também uma preferência como "primeira" linguagem de programação.

2.1.4 NumPy

Conforme Santiago (2020), o NumPy é uma poderosa biblioteca Python que é usada principalmente para realizar cálculos em *Arrays* Multidimensionais. O NumPy fornece um grande conjunto de funções e operações de biblioteca que ajudam os programadores a executar facilmente cálculos numéricos. As principais aplicações dessa biblioteca são modelos de *Machine Learning*, processamento de imagem e computação gráfica e algumas tarefas matemáticas com cálculos simples e complexos.

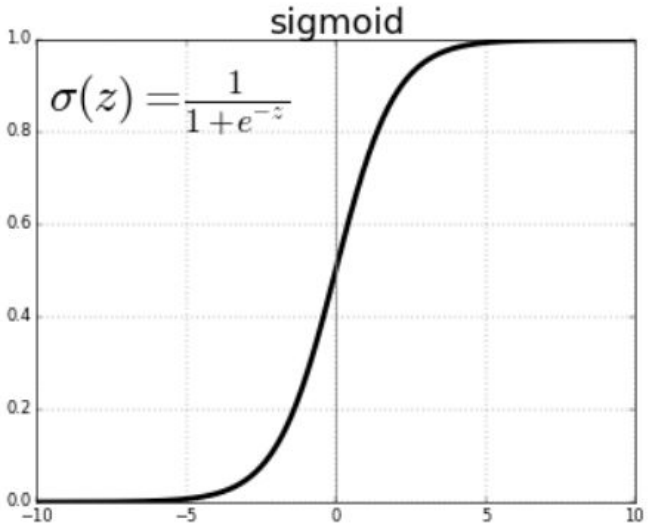
2.1.5 Funções de ativação

De acordo com Soares (2020), as funções de ativação são funções que recebem um tipo de sinal de entrada e o converte para um sinal de saída. Podem ser lineares ou

não-lineares, contudo seu principal potencial está na introdução de não linearidade nas redes neurais.

Elas permitem que pequenas mudanças nos pesos e bias causem apenas uma pequena alteração no output, onde basicamente decidem se um neurônio deve ser ativado ou não. Dessa forma, se a informação que o neurônio está recebendo é relevante para a informação fornecida ou deve ser ignorada. Um exemplo de função de ativação é a de tipo sigmoide, observada na Figura 2.

Figura 2 – Representação gráfica da função sigmoidal



Fonte: SOARES, 2020.

2.2 MATERIAL E MÉTODOS

O desenvolvimento deste projeto, de modo genérico, de uma rede neural utiliza se da linguagem de programação Python, que possui certas operações simplificadas perante outras linguagens e que dispõe de um grande número de bibliotecas e materiais complementares. A rede neural desenvolvida segue a ideia de "*FeedForward*", não sendo desenvolvido a parte correspondente a "*BackPropagation*" ou qualquer parâmetro que torne esse código um algoritmo genético.

O código é composto por "neurônios", sendo eles divididos em três classes: entrada, ocultos e de saída. As entradas são tratadas por meio de multiplicação de matrizes e entre esses neurônios foram aplicados pesos aleatórios e uma função de ativação. Assim pressupõe que o código será capaz de identificar valores e trazer resultados baseados nos pesos colocados entre as camadas.

2.2.1 Visual Studio

Como ferramenta de desenvolvimento e análise do algoritmo optou-se por utilizar a IDE (ambiente de desenvolvimento integrado) do Microsoft Visual Studio em virtude da facilidade ao encontrar os complementos e bibliotecas necessárias para executar o algoritmo na linguagem Python.

2.2.2 Desenvolvimento da Rede Neural

Para realizar os cálculos de matrizes que serão necessários para o funcionamento do código fora importada a biblioteca Numpy, como observado na Figura 3.

Figura 3 – Importando a biblioteca Numpy

```
import numpy as np #numpy é uma biblioteca utilizada
#para fazer cálculos matemáticos de matrizes
# - np é o atalho para esta função
```

Fonte: Autores (2020)

Em um segundo momento foi foram definidas duas matrizes, sendo elas uma matriz, 4 x 2 para guardar os valores de entrada, bem como uma matriz 3 x 1 para armazenar os dados de saída. Os valores do teste podem ser verificados na Figura 4.

Figura 4 – Definição de matrizes

```
# X = (horas estudando, horas dormindo), y = Pontuação no teste
xAll = np.array(([2, 9],
                 [1, 5],
                 [3, 6],
                 [5, 10]), dtype=float) # xALL = dados de entrada - np.array cria uma matriz 4x2
y = np.array(([92],
              [86],
              [89]), dtype=float) # y = saída - np.array cria uma matriz 3x1 para a saída desejada
```

Fonte: Autores (2020)

O conjunto de comandos observado na Figura 5 é composto por cálculos que realizam a transformação de números inteiros em decimal, e reorganizam a matriz que vai ser utilizada como entrada dividindo-a em diversas submatrizes.

Figura 5 – Conversão e divisão dos dados

```
# unidades de escala
xAll = xAll/np.amax(xAll, axis=0) # escala dos dados de entrada -
#np.amax = retorna o maior valor dentro da matriz no eixo 0
y = y/100 # limita a escala dos dados de saída (o máximo do teste é 100)

# Divisão dos dados
X = np.split(xAll, [3])[0] # divide uma matriz em varias submatrizes - training data
```

Fonte: Autores (2020)

Os parâmetros da classe “Rede Neural” definem a quantidade de neurônios de entrada, bem como os neurônios ocultos e de saída que o sistema possuirá. Esses parâmetros podem ser observados na Figura 6.

Figura 6 - Parâmetros

```
class Rede_Neural(object):
  def __init__(self):
    #parâmetros
    self.inputSize = 2
    self.outputSize = 1
    self.hiddenSize = 3
```

Fonte: Autores (2020)

Nesse código os pesos são definidos por meio aleatório utilizando a biblioteca NumPy e são criados de acordo com a quantidade de neurônios estipulados anteriormente. As funções utilizadas no cálculo dos pesos podem ser observadas na Figura 7.

Figura 7 - Pesos

```
#Pesos
self.W1 = np.random.randn(self.inputSize, self.hiddenSize) #  Matriz de peso da entrada para camada oculta
self.W2 = np.random.randn(self.hiddenSize, self.outputSize) #  Matriz de peso da camada oculta para saida
    #np.randon.randn cria uma matriz randomica do tamanho de seus parametros
```

Fonte: Autores (2020)

O feedforward com a multiplicação dos neurônios e aplicação dos pesos, como pode ser observado na Figura 8. A variável “z” é a entrada pelo primeiro conjunto de pesos, “z2” é onde se aplica a função de ativação, nesse caso sigmoide, e “z3” recebe o produto da camada oculto pelo segundo conjunto de pesos. Por fim, a variável de saída “o” recebe o produto de z3 ativado com a função sigmoidal.

O motivo da escolha pela função de ativação de tipo sigmoide foi o alcance dos valores de saída, nesse caso variando de 0 a 1 e possuindo um formato S, ou seja, a saída não é linear. Essa função no algoritmo pode ser notada na Figura 9.

Figura 8 - Feedforward

```
def forward(self, X):
  #Propagação direta através da rede
  self.z = np.dot(X, self.W1) # produto pontual de X (entrada) e primeiro conjunto de pesos 3x2

  self.z2 = self.sigmoid(self.z) # Função de ativação no resultado de z

  self.z3 = np.dot(self.z2, self.W2) # produto pontual da camada oculta (z2) e o segundo conjunto de pesos 3x1

  o = self.sigmoid(self.z3) # Função de ativação no valor de saida
    #dot é a função de multiplicação de matrizes
  return o
```

Fonte: Autores (2020)

Figura 9 - Função de ativação

```
def sigmoid(self, s):
  # Função de Ativação
  return 1/(1+np.exp(-s)) #np.epx equivale a exponencial (e¹)
```

Fonte: Autores (2020)

Por fim, a variável NN é igualada a função Rede Neural () e executada. Como observado na Figura 10, a Saída Prevista é aquela originada pelos cálculos da Rede Neural, enquanto a Saída Atual é a meta estabelecida antes do uso do algoritmo.

Figura 10 - Definindo a saída

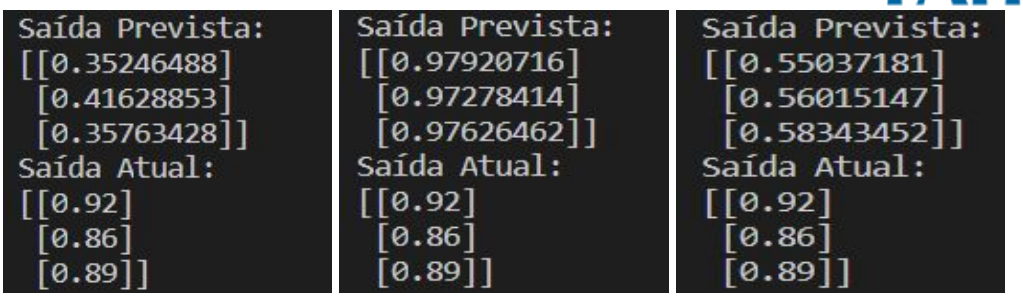
```
NN = Rede_Neural()

#Definindo a saída
o = NN.forward(X)

print ("Saída Prevista: \n" + str(o))
print ("Saída Atual: \n" + str(y))
```

Fonte: Autores (2020)

2.3 RESULTADOS E DISCUSSÃO

Após o desenvolvimento do código realizou-se a execução do mesmo para que os resultados pudessem ser analisados contra a saída prevista e comparados com as hipóteses colocadas anteriormente. Dessa forma obteve-se os resultados apresentados na Figura 11 nas saídas da rede neural após três execuções do código:

Figura 11 - Resultados das saídas do código



Fonte: Autores (2020)

Pode-se observar que os resultados não seguem uma linha lógica, o que condiz com as hipóteses apresentadas, exibindo um resultado coerente com o desejado. O algoritmo busca através de tentativas com valores aleatórios aproximar-se do resultado desejado, mas não consegue "aprender" em virtude de ser apenas um *feedforward* genérico, sem *backpropagation*.

Dessa forma, o funcionamento da rede neural pode ser exposto como uma série de acertos e erros, onde as tentativas acertadas – ou mais aproximadas do resultado desejado – são utilizadas como alvo a ser superado na próxima tentativa realizada pelo algoritmo.Também se coloca que o uso da linguagem Python mostrou ser muito produtivo, pois com facilidade se pode encontrar informações e projetos para auxílio no desenvolvimento de um código de rede neural.

CONCLUSÃO

Após o projeto e desenvolvimento da rede neural, se pode afirmar que os objetivos foram alcançados e as hipóteses foram confirmadas, pois o algoritmo apresentou o comportamento aleatório previsto anteriormente.

Uma forma de melhorar este código seria a utilização de "*BackPropagation*" e algoritmos genéticos, onde existem inúmeras aplicações para os mesmos, variando de um sistema que analisa números a códigos capazes de identificação facial de seres humanos e aprendizados e melhorias em suas análises com o passar do tempo.

Com os resultados da rede neural genérica é possível observar que inúmeras aplicações podem ser feitas com redes neurais e que muitas ainda estão para ser desenvolvidas, explicitando a importância de aprender sobre redes neurais e desenvolver uma linha de pensamento concisa sobre o assunto, não só para manifestar uma ideia, como também para estar apto a trabalhar juntamente com essa tecnologia e se tornar mais produtivo de forma a possuir um diferencial no mercado cada vez mais competitivo.

REFERÊNCIAS

ARAÚJO, Joseana Macêdo Fechine Régis de. **Aprendizagem (Redes Neurais - Complementar).** Disponível em: http://www.dsc.ufcg.edu.br/~joseana/IAPos_NA15_Complementar.pdf. Acesso em: 05 abr. 2020.

BARRETO, Jorge M. **Introdução às Redes Neurais Artificais**. Florianópolis: UFSC -Departamento de Informática e de Estatística, 2002.

CARVALHO, André Ponce de Leon F. de. **Redes Neurais Artificiais**. Disponível em: https://sites.icmc.usp.br/andre/research/neural/. Acesso em 07 maio 2020.

PERKOVIC, Ljubomir. **Introdução à Computação Usando Python - Um Foco no Desenvolvimento de Aplicações.** Rio de Janeiro; LTC — Livros Técnicos e Científicos Editora Ltda., 2016. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788521630937/. Acesso em: 03 maio 2020.

RUSSELL, Rudolph. **MACHINE LEARNING:** Step-by-Step Guide To Implement Machine Learning Algorithms with Pytho: CreateSpace Independent Publishing Plataform, 2018. E-book.

SANTIAGO, Luiz. **O que é NumPy?**. Disponível em: https://medium.com/ensina-ai/entendendo-a-biblioteca-numpy-4858fde63355. Acesso em: 05 abr. 2020.

SOARES, Anderson da Silva. **Aula 2 - Redes Neurais Perceptron Multicamadas.** Disponível em: http://ww2.inf.ufg.br/~anderson/deeplearning/20181/Aula%202%20-%20Multilayer%20Perceptron.pdf. Acesso em: 05 abr. 2020.