UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
ELE0606 - TÓPICOS ESPECIAIS EM INTELIGÊNCIA ARTIFICIAL

Docente:
José Alfredo Ferreira Costa

Autor:
Vinícius Venceslau Venancio da Penha

Redes Neurais MLPs

Natal - RN
1 de novembro de 2023

RESUMO

O relatório abordou tópicos essenciais, incluindo Redes Multilayer Perceptron (MLPs) e sua arquitetura, o Neurônio Artificial e seu papel nas redes neurais, uma comparação detalhada entre MLP, KNN e Árvore de Decisão como métodos de classificação, bem como a implementação prática em Python com as bases de dados Wine e Heart Disease.

Por fim, enfatizou-se a importância da compreensão desses conceitos para aplicações práticas e análise de dados.

Sumário

Lista de Figuras

1 Introdução

As **Redes Multilayer Perceptron (MLPs)** são uma classe de redes neurais artificiais amplamente utilizadas em tarefas de aprendizado supervisionado, como classificação e regressão.

Neste relatório, é explorado o uso de MLPs na análise da base de dados **Wine**, disponível no Scikit-Learn, bem como da base de dados **Heart Disease**, disponibilizada pelo docente responsável.

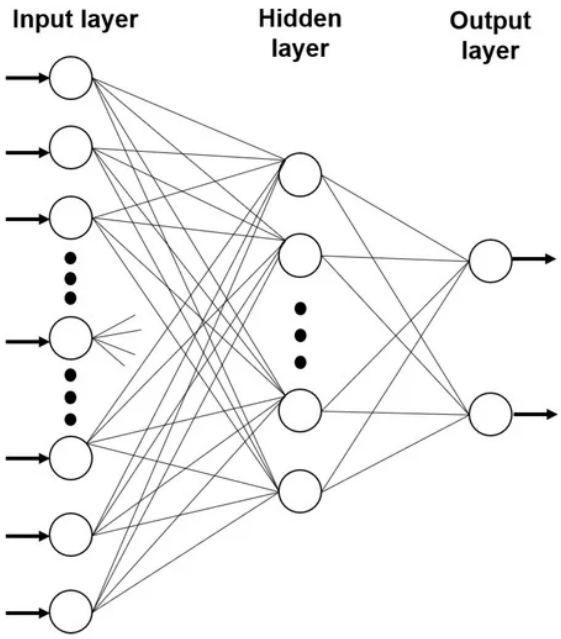

Figura 1: Estrutura de rede neural feedforward.

Feedforward é um termo que se refere a qualquer rede neural em que as informações fluem em uma direção, da camada de entrada para a camada de saída, sem ciclos ou realimentação.

Assim, a rede MLP (Multilayer Perceptron) é um tipo específico de rede neural feedforward que consiste em múltiplas camadas de neurônios (camadas ocultas) entre a camada de entrada e a camada de saída.

2 Desenvolvimento

2.1 O Neurônio Artificial

Uma rede neural é composta por neurônios artificiais, que são as unidades fundamentais de processamento. Cada neurônio recebe um conjunto de entradas ponderadas, realiza uma transformação linear dessas entradas e, em seguida, aplica uma função de ativação para produzir a saída.

A operação de um neurônio artificial pode ser descrita da seguinte maneira:

1. **Entradas:** Cada neurônio recebe um vetor de entradas, denotado, por exemplo, como:
$$X = [x_1, x_2, ..., x_n]$$
Dado que n se refere ao número de entradas no sistema.

2. **Pesos:** Cada entrada é atrelada a um peso correspondente, podendo ser representado da seguinte forma:
$$W = [w_1, w_2, ..., w_n]$$
Vale ressaltar que, neste caso, n ainda está associado ao número de entradas no sistema.

   Outrossim, os pesos representam a importância das entradas para o neurônio.

3. **Operação Linear:** O neurônio realiza uma soma ponderada das entradas multiplicadas pelos pesos, o que pode ser representado matematicamente como:
$$Z = \sum_{i=1}^{n} (x_i \cdot w_i)$$

4. **Função de Ativação:** Após a operação linear, o neurônio aplica uma função de ativação $f(Z)$ ao resultado Z para produzir a saída Y.

A função de ativação introduz não linearidade na operação do neurônio. Alguns exemplos de funções de ativação comuns incluem a função sigmóide, a função ReLU (Rectified Linear Unit) e a função tangente hiperbólica (tanh).

2.2 Arquitetura das Redes MLP

As MLPs são compostas por várias camadas de neurônios, incluindo uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída. Cada neurônio em uma camada está conectado a todos os neurônios na camada adjacente, tornando-as redes densamente conectadas.

No caso da base de dados Wine, a camada de entrada pode ter um número de neurônios igual ao número de características dos vinhos. Dessa maneira, as camadas ocultas são configuradas com base na complexidade da tarefa. Por fim, a camada de saída geralmente tem um neurônio por classe ou rótulo. Neste caso, tem-se três classes (tipos de vinho), então a camada de saída terá três neurônios.

2.2.1 Camada de Entrada

A primeira camada de uma rede neural é chamada de camada de entrada e consiste em nós que representam informações de entrada relevantes para o problema atual. Cada um desses nós está associado a uma característica ou atributo dos dados.

Para ilustrar este conceito, num cenário de classificação de dígitos manuscritos, os nós na camada de entrada podem ser entendidos como codificando a intensidade dos pixels em locais específicos da imagem.

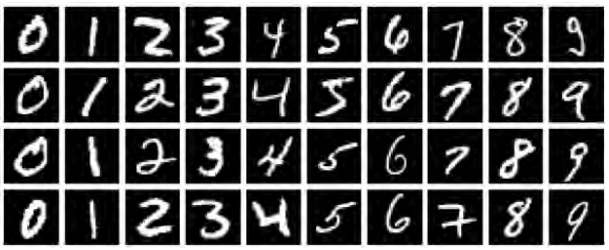

Figura 2: Amostras de imagens do banco de dados de dígitos manuscritos MNIST.

2.2.2 Camadas Ocultas

As camadas intermediárias estão posicionadas entre a camada de entrada e a camada de saída. Essas camadas são frequentemente denominadas "intermediárias"devido à ausência de uma conexão direta com as entradas ou saídas do sistema. Cada camada intermediária é composta por um número variável de unidades (também conhecidas como neurônios).

Essas unidades processam os dados que vêm das camadas anteriores, aplicam transformações não lineares e encaminham os resultados para a camada subsequente. É a integração de múltiplas camadas intermediárias que confere às MLPs a capacidade de aprender representações complexas dos dados.

2.2.3 Camada de Saída

A camada de saída é a última camada da rede neural. O número de unidades nessa camada varia de acordo com a natureza do problema que a rede está resolvendo. Por exemplo, em tarefas de classificação binária, haverá uma única unidade de saída para representar a probabilidade da classe positiva. Em problemas de classificação multiclasse, existirá uma unidade para cada classe possível.

Nesse sentido, essa camada é responsável por gerar as previsões ou resultados finais com base nas informações processadas pelas camadas intermediárias.

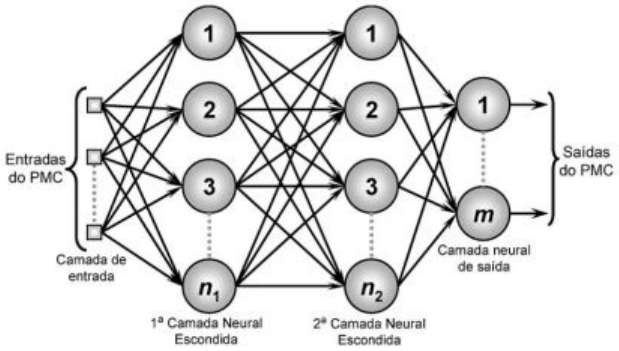

Figura 3: Redes Perceptron de Múltiplas Camadas (PMC).

2.3 Comparação entre os métodos de classificação

2.3.1 MLPs e KNN

Quando se trata de escolher entre MLP (Redes Multilayer Perceptron) e KNN (k-vizinhos mais próximos), é essencial ponderar as nuances de cada abordagem. O MLP, como uma rede neural feedforward, revela-se uma opção robusta para tarefas que demandam a captura de relações complexas nos dados. No entanto, a sofisticação do MLP traz consigo a necessidade de um treinamento cuidadoso e do ajuste de hiperparâmetros para atingir seu potencial máximo.

O MLP brilha particularmente em problemas complexos com grandes conjuntos de dados, mas sua sofisticação o torna suscetível ao overfitting se não for adequadamente regularizado.

Por outro lado, o KNN opera sob uma filosofia diferente. Em vez de criar um modelo explícito, o KNN toma decisões com base na proximidade entre os pontos de dados. Essa abordagem simples torna o KNN altamente interpretável e, portanto, uma escolha valiosa quando a interpretabilidade é uma prioridade.

Em resumo, é evidente que o desempenho do KNN é sensível à escala e à dimensionalidade dos dados, destacando-se em conjuntos de dados pequenos a moderados e quando os dados estão devidamente normalizados. Cada método tem seu próprio lugar no contexto de problemas de aprendizado de máquina, e a escolha depende das necessidades específicas de cada cenário.

2.3.2 MLPs e Árvore de Decisão

A seleção entre o MLP e a Árvore de Decisão está atrelada às demandas específicas da problemática em questão. O MLP é a escolha preferível em cenários intrincados, dotados de um grande volume de dados, onde a simplicidade interpretativa não figura como critério fundamental.

Por sua vez, as Árvores de Decisão revelam-se valiosas em situações mais simplificadas e naquelas em que a compreensibilidade assume papel central. Diante disso, este método constitue modelos governados por princípios de segregação, onde determinações são tomadas em cada ponto de decisão com base em atributos específicos. Essa configuração outorga às Árvores de Decisão um alto entendimento, uma vez que cada subdivisão representa uma determinação explícita.

2.4 Algoritmo em Python

2.4.1 Base de Dados - Wine

```python
import time
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, confusion_matrix

# Numero de vezes que voce deseja repetir o treinamento para calcular a media
num_execucoes = 10
acuracias = []
tempo_treinos = []

for n in range(num_execucoes):
    # Passo 1: Carregando os dados
    data = load_wine()
    X, y = data.data, data.target

    # Passo 2: Pre-processamento dos dados (normalizacao)
    scaler = StandardScaler()
    X = scaler.fit_transform(X)

    # Passo 3: Divisao dos dados em conjuntos de treinamento e teste
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=None)

    # Passo 4: Criacao do Modelo MLP
    mlp = MLPClassifier(hidden_layer_sizes=(100, 50), activation='relu', max_iter=1000)

    # Passo 5: Treinamento do Modelo
    inicio_tempo = time.time()
    mlp.fit(X_train, y_train)
    final_tempo = time.time()
    tempo_treino = final_tempo - inicio_tempo
    tempo_treinos.append(tempo_treino)

    # Passo 6: Avaliacao da Acuracia
    y_pred = mlp.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    acuracias.append(acc)

# Calculando a media das acuracias e do tempo de treinamento
media_acuracias = sum(acuracias) / num_execucoes
media_tempo_treino = sum(tempo_treinos) / num_execucoes

# Exibindo resultados
print("Media da Acuracia do Modelo MLP:", media_acuracias)
print("Media do Tempo de Treinamento:", media_tempo_treino, "segundos")
```

Listing 1: Redes Multilayer Perceptron (MLP) - **Wine**.

```
Media da Acuracia do Modelo MLP: 0.9722222222222223
Media do Tempo de Treinamento: 0.4033689260482788 segundos
```

Listing 2: Saída 1 - **Wine**.

```
# Criacao da matriz confusao

cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='coolwarm', annot_kws={"size": 14})

plt.xlabel('Previsoes')
plt.ylabel('Valores Verdadeiros')
plt.title('Matriz de Confusao')
plt.show()
```

Listing 3: Desenvolvimento da Matriz Confusão - **Wine**.

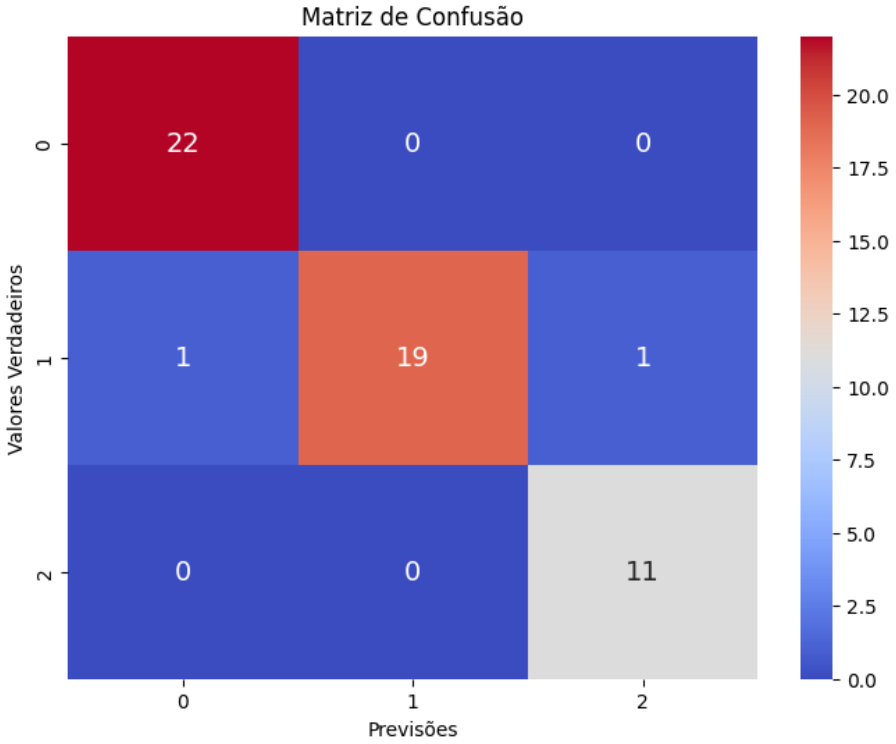

Figura 4: Saída 2 - **Wine**.

2.4.2 Base de Dados - Heart Disease

```python
import time
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, confusion_matrix

num_execucoes = 10
acuracias = []
tempo_treinos = []

# Passo 1: Carregando os dados
data = pd.read_csv("/content/heart.csv")

# Passo 2: Criar um Dataframe
heart_df = data

# Passo 3: Preparar os dados
X = heart_df.drop('target', axis=1).to_numpy()
y = heart_df['target'].to_numpy()

# Passo 4: Normalizar os dados:
scaler = StandardScaler()
X = scaler.fit_transform(X)

for n in range(num_execucoes):
    # Passo 5: Divisao dos dados em conjuntos de treinamento e teste
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=None)

    # Passo 6: Criacao do Modelo MLP
    mlp = MLPClassifier(hidden_layer_sizes=(32, 16), activation='relu', max_iter=1000)

    # Passo 7: Treinamento do Modelo
    inicio_tempo = time.time()
    mlp.fit(X_train, y_train)
    final_tempo = time.time()
    tempo_treino = final_tempo - inicio_tempo
    tempo_treinos.append(tempo_treino)

    # Passo 8: Avaliacao da Acuracia
    y_pred = mlp.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    acuracias.append(acc)

media_acuracias = sum(acuracias) / num_execucoes
media_tempo_treino = sum(tempo_treinos) / num_execucoes

print("Media da Acuracia do Modelo MLP:", media_acuracias)
print("Media do Tempo de Treinamento:", media_tempo_treino, "segundos")
```

Listing 4: Redes Multilayer Perceptron (MLP) - **Heart Disease**.

```
Media da Acuracia do Modelo MLP: 0.9736585365853658
Media do Tempo de Treinamento: 3.26777446269989 segundos
```

Listing 5: Saída 1 - **Heart Disease**.

```
# Criacao da matriz confusao

cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='coolwarm', annot_kws={"size": 14})

plt.xlabel('Previsoes')
plt.ylabel('Valores Verdadeiros')
plt.title('Matriz de Confusao')
plt.show()
```

Listing 6: Criação da Matriz Confusão - **Heart Disease**.

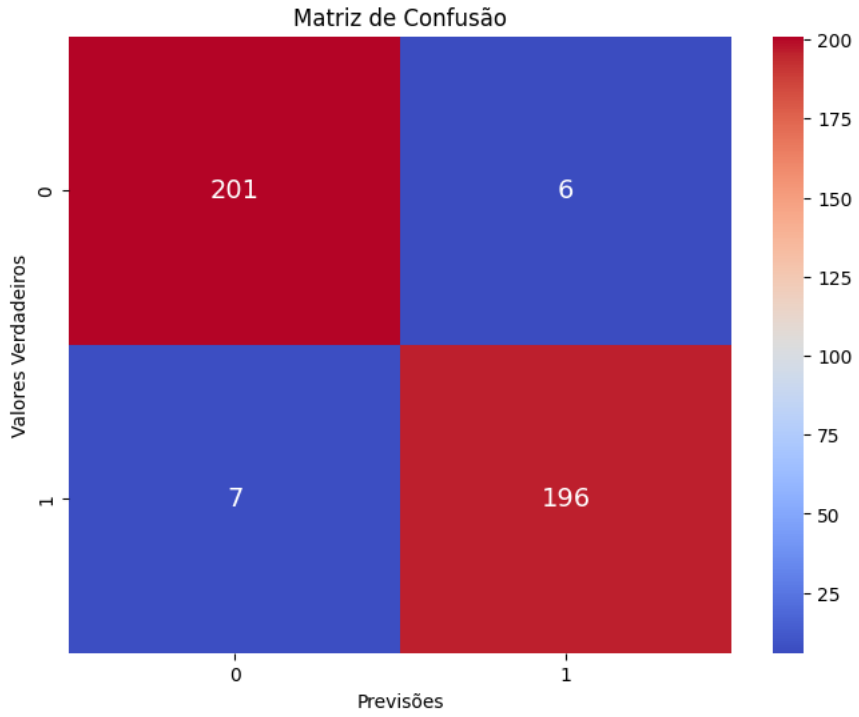

Figura 5: Saída 2 - **Heart Disease**

3 Conclusão

As Redes Multilayer Perceptron (MLPs) são uma poderosa ferramenta para tarefas de aprendizado supervisionado, incluindo a classificação de dados, como a base de dados Wine que exploramos neste relatório. Elas são altamente configuráveis em termos de arquitetura e hiperparâmetros, o que permite que se adaptem a uma variedade de problemas.

No contexto da análise da base de dados Wine, uma MLP pode ser treinada para classificar vinhos em três categorias com bom desempenho. É fundamental considerar a divisão apropriada dos dados em treinamento, validação e teste, bem como a escolha de funções de ativação e técnicas de regularização.

Em suma, as MLPs são uma ferramenta valiosa na caixa de ferramentas de aprendizado de máquina, permitindo a modelagem de relacionamentos complexos em dados e demonstrando sua eficácia na classificação de vinhos no conjunto de dados Wine. O entendimento dos neurônios artificiais e de suas operações é fundamental para compreender o funcionamento das MLPs e redes neurais em geral.

4 Referências

[1] NUNES DA SILVA, I. **Redes Neurais Artificiais AULA 05 - Perceptron Multicamadas - Aspectos de Treinamento - Aplicação em Aproximação de Funções**. [s.l: s.n.]. Disponível em: `https://edisciplinas.usp.br/pluginfile.php/7677182/mod_resource/content/3/RNA_Aula05.pdf`. Acesso em: 31 de outubro de 2023.

[2] **José Alfredo Costa - Aula 25-10-2023 - Introdução a redes neurais**. Disponível em:`https://www.youtube.com/watch?v=UNy2TCaR9ik`. Acesso em: 31 de outubro de 2023.

[4] **Scikitlearn - MLP Classifier**. Disponível em: `https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html`. Acesso em: 1 de novembro de 2023.

[4] OPENAI. **ChatGPT**. 2023. Disponível em: `https://openai.com/`. Acesso em: 1 de novembro de 2023.