PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Instituto de Ciências Exatas e de Informática

Marcos Antonio Lommez Candido Ribeiro[1]

# Lista #7

**Inteligência Artificial**

[1]Aluno de Graduação em Ciência da Computação – tonilommez@hotmail.com

## Todo o codigo gerado pode ser encontrado no link a seguir

Questão 1

Questão 5

**Implemente o algoritmo Percetron para resolver as funções AND e OR com n entradas booleanas. Ou seja, o usuário poderá selecionar se ele deseja resolver um problema AND com 2 ou 10 entradas, por exemplo.**

**A sua lista deverá conter todas as explicações da implementação e os resultados dos testes realizados.**
**Coloque prints das entradas e saídas e explique o que foi observado.**
**Ao final, disponibilize o código desenvolvido.**
**Mostre também que o Perceptron não resolve o XOR.**

### Definição da classe Perceptron

A classe `Perceptron` é uma implementação das formas mais simples de um classificador linear. Este modelo é baseado no conceito de um neurônio artificial e é capaz de realizar classificações binárias. A classe consiste em várias partes principais:

- **weights**: Lista que armazena os pesos associados a cada entrada, onde cada peso determina a importância de cada entrada na decisão.
- **threshold**: Número de iterações sobre o conjunto de treinamento para ajustar os pesos.
- **learning_rate**: Taxa de aprendizado que controla o quanto os pesos são ajustados durante o treinamento.
- **bias**: Coeficiente da dimensão Y da fórmula linear utilizada no perceptron.

A predição é feita realizando o somatório dos pesos multiplicados pelas entradas mais o bias, o que pode ser simplificado na fórmula da equação linear

$$\text{Output} = \sum_{i=1}^{n} (w_i \cdot x_i) + b$$

onde:

- Output é a saída prevista pelo Perceptron (antes da aplicação da função de ativação).

- $w_i$ são os pesos.
- $x_i$ são as entradas.
- $b$ é o bias.
- $n$ é o número de entradas.

Por fim, o treinamento é feito com a iteração entre os valores de treinamento. Para cada valor de entrada, realiza-se a predição, e com base no erro, os pesos são ajustados.

```python
class Perceptron:
    weights = 0
    threshold = 0
    learning_rate = 0
    bias = 0

    def __init__(self, num_inputs, threshold=100000, learning_rate=0.01):
        # Inicializando os pesos e o bias
        self.weights = [0.0 for _ in range(num_inputs)]
        self.bias = 0.0
        self.threshold = threshold
        self.learning_rate = learning_rate

    def predict(self, inputs):
        # Calcula a soma ponderada das entradas e adiciona o bias
        summation = sum(w * i for w, i in zip(self.weights, inputs)) + self.bias
        # Aplica a função de ativação (step function)
        return 1 if summation > 0 else 0

    def train(self, training_inputs, labels):
        for _ in range(self.threshold):
            for inputs, label in zip(training_inputs, labels):
                prediction = self.predict(inputs)
                # Ajusta os pesos e o bias com base no erro (label - prediction)
                for i in range(len(self.weights)):
                    self.weights[i] += self.learning_rate * (label - prediction) * inputs[i]
                self.bias += self.learning_rate * (label - prediction)

    def print_formula(self):
        terms = [f"({w:.2f} * x{i})" for i, w in enumerate(self.weights, start=1)]
        formula = " + ".join(terms) + f" + {self.bias:.2f}"
        print(f"\nformula = {formula}")
```

A função `generate_inputs` é usada para gerar todas as combinações possíveis de entradas booleanas para um dado número de entradas $n$ e assim foi feito o treinamento das instancias de perceptron para as formulas logicas. Aqui está o código da função:

```
def generate_inputs(n):
    return [[(i >> j) & 1 for j in range(n)] for i in range(2 ** n)]
```

Por fim após realizado o treinamento das instancias de perceptron podemos validar o treinamento dos modelos, e assim estas foram as saidas:

**AND**

```
Input: [0, 0] - Expected: 0, Predicted: 0
Input: [1, 0] - Expected: 0, Predicted: 0
Input: [0, 1] - Expected: 0, Predicted: 0
Input: [1, 1] - Expected: 1, Predicted: 1
Accuracy: 100.0%
formula = (0.01 * x1) + (0.02 * x2) + -0.02
```

**OR**

```
Input: [0, 0] - Expected: 0, Predicted: 0
Input: [1, 0] - Expected: 1, Predicted: 1
Input: [0, 1] - Expected: 1, Predicted: 1
Input: [1, 1] - Expected: 1, Predicted: 1
Accuracy: 100.0%
formula = (0.01 * x1) + (0.01 * x2) + 0.00
```

**XOR**

```
Input: [0, 0] - Expected: 0, Predicted: 1
Input: [1, 0] - Expected: 1, Predicted: 1
Input: [0, 1] - Expected: 1, Predicted: 0
Input: [1, 1] - Expected: 0, Predicted: 0
Accuracy: 50.0%
formula = (0.00 * x1) + (-0.01 * x2) + 0.01
```

**O problema do xor**

Redes Perceptrons como concebidas inicialmente não são capazes de computar a equação de xor, devido a natureza de implementarem suas decisões baseadas em uma formula linear wx+b. Como o plot do XOR não pode ser separado em uma simples linha então consequentemente o perceptron não ira conseguir resolve-lo. Isso pode ser contornado utilizando duas estrategias, primeiro adicionando camadas e transformando o perceptron em um multi layer, segundo modificando a formula para quadrática e possivelmente usando outra politica de atualização de erro, mas esta solução se torna overkill além de adicionar camadas de complexidade na utilização do modelo.

**Considere as seguintes afirmações sobre redes neurais artificiais:**
**I. Um perceptron elementar só computa funções linearmente separáveis.**
**II. Não aceitam valores numéricos como entrada.**
**III. O "conhecimento"é representado principalmente através do peso das conexões.**
**São corretas:**
**(a) Apenas III**
**(b) Apenas I e II**
**(c) Apenas I e III**
**(d) Apenas II e III**
**(e) I, II e III**

Resposta correta: (c) Apenas I e III

1: A formula do perceptron elementar é baseada formula da reta sum(wx)+b
2: Perceptrons precisam que o valor seja numérico.
3: O conhecimento do perceptron é colocado dentro dos seus pesos e eventualmente no bias que também é um peso.

**Considere as funções booleanas abaixo:**
**I.** $p \wedge q$ **(conjunção)**
**II.** $p \equiv q$**(equivalência)**
**III.** $p \implies q$ **(implicação)**
**Quais destas funções podem ser implementadas por um perceptron elementar?**
**a) Somente I;**
**b) Somente I e II;**
**c) Somente I e III;**
**d) Somente II e III;**
**e) I, II e III.**

Resposta correta: (c) Somente I e III

A equivalência equivale a um xnor, que precisa de pelo menos 2 retas ou uma curva para diferenciar

**Dado um perceptron simples de duas entradas e um bias, cujos pesos são w1=0,5, w2=0,4 e w0=-0,3 respectivamente, assinalar a resposta correta:**
**(a) o perceptron realiza a função NOR**
**(b) o perceptron realiza a função AND**
**(c) o perceptron realiza a função OR**
**(d) o perceptron realiza a função XOR**
**(e) nenhuma das alternativas**

(0.5*0) + (0.4*0) + (-0.3) = -0.3 = 0
(0.5*0) + (0.4*1) + (-0.3) = 0.1 = 1
(0.5*1) + (0.4*0) + (-0.3) = 0.2 = 1
(0.5*1) + (0.4*1) + (-0.3) = 0.6 = 1

Portanto a alternativa certa é (c) OR

**Considerando a base de dados (breast-cancer.csv) e utilizando-se o notebook "RNA.ipynb" (faça as adaptações necessárias) que está no CANVAS, pede-se:**
**1) Faça os pré-processamentos necessários para esta base: conversão de nominal para numérico, identificação de outlier, normalização, balanceamento, eliminação de redundância, dentre outros**
**2) Avalie o desempenho do modelo a partir de diferentes topologias: números de camadas e neurônios. Use diferentes heurísticas para isso, conforme discutido em sala**
**3) Avalie o parâmetro 'taxa de aprendizado' e veja sua relação com a quantidade de épocas.**
**4) Caso julgue necessário, investigue outros hiperparâmetros necessários para a rede neural a fim de melhorar os resultados obtidos. Compare os resultados obtidos.**
**5) Ajuste os hiperparâmetros automaticamente, usando métodos como Grid Search, CVParameterSelection e MultiSearch**

Primeiro analisei a existência de valores nulos e cheguei a um total de 9 valores, devido a sua baixa quantidade em relação a tabela original, decidi apenas remove-los para evitar viés.

A seguir fiz a discretização da tabela usando o seguinte mapa:

- `Age`: '20-29': 0, '30-39': 1, '40-49': 2, '50-59': 3, '60-69': 4, '70-79': 5.

- `Menopause`: 'premeno': 0, 'ge40': 1, 'lt40': 2.
- `Tumor Size`: '0-4': 0, '5-9': 1, '10-14': 2, '15-19': 3, '20-24': 4, '25-29': 5, '30-34': 6, '35-39': 7, '40-44': 8, '45-49': 9, '50-54': 10.
- `Inv Nodes`: '0-2': 0, '3-5': 1, '6-8': 2, '9-11': 3, '12-14': 4, '15-17': 5, '24-26': 6.
- `Node Caps`: 'no': 0, 'yes': 1.
- `Breast`: 'left': 0, 'right': 1.
- `Irradiat`: 'no': 0, 'yes': 1.
- `Breast Quad`: 'left_up': 0, 'right_up': 1, 'central': 2, 'left_low': 3, 'right_low': 4.
- `Class`: 'no-recurrence-events': 0, 'recurrence-events': 1.

Durante a analise de instancias repetidas 14 foram identificadas e removidas da base. Alem disso, devido a todos os dados terem sido colocados em um range que começa entre 0 e no máximo 10, a normalização dos atributos não se tornou necessária, mesmo com testes utilizando o valor final não mudou. Igualmente no z-score poucos valores foram identificados como outliers, apenas 10 na classe inv-nodes, e devido a natureza de redes neurais lidarem bem com esses dados foi decidido mantê-los, porque a natureza do significado desses dados poderia influenciar negativamente o modelo.

Partindo para o balanceamento dos dados, a base é dividida entre 187 valores de não e 77 valores de sim. Adotei uma estrategia de retirar proporcionalmente mais valores da classe majoritaria (40%) e menos da minoritaria (10%), e em seguida fazer uma aplicação de SMOTE para oversampling dos valores, levando a um total de 222 instancias de treinamento e 83 para validação.

O modelo foram ajustado com Grid Search utilizando CV=3 e os seguintes hiper parâmetros:

- `max_iter`: [1000]
- `random_state`: [42]
- `hidden_layer_sizes`: [(50, 50, 50), (50, 100, 50), (100)]
- `activation`: ['relu', 'identity', 'logistic', 'tanh']
- `solver`: ['sgd', 'adam']
- `alpha`: [0.0001, 0.05]
- `learning_rate`: ['constant','adaptive']

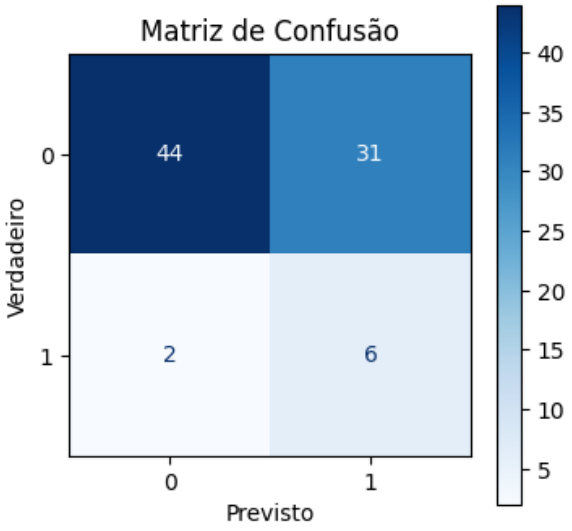
O algoritmo chegou a uma conclusão de que os melhores seriam a combinação de:

- `activation`: 'logistic'
- `alpha`: 0.0001
- `hidden_layer_sizes`: 100
- `learning_rate`: 'constant'
- `max_iter`: 1000
- `random_state`: 42
- `solver`: 'adam'

Assim sendo o modelo foi treinado, levando um total de 225 épocas para chegar a uma conversão menor do que 0.001 por 10 épocas consecutivas.

Com o modelo devidamente treinado e validado os seguintes valores foram obtidos:

Acurácia: 0.60
Recall: 0.75
Precisão: 0.16
Pontuação F1: 0.27



## Faça uma resenha do artigo que está no CANVAS: "A Survey Of Methods For Explaining Black Box Models"

O artigo apresentado faz uma análise sobre o estado da utilização de modelos que possuem a característica de serem "caixa-preta", ou seja, a falta de interpretabilidade existente nos modelos de aprendizado de máquina complexos. À medida que estes se tornam cada vez mais usados em diversas áreas, principalmente por serem muitas vezes o estado-da-arte, torna-se cada vez mais necessário explicar o motivo de suas saídas, para que decisões não sejam tomadas com viés inerente à sua base de dados.

É comum que problemas ocorram em modelos "caixa-preta". Por exemplo, ao analisar uma imagem e classificá-la, o modelo pode acabar se baseando na cor do fundo da imagem, ou em objetos próximos que estejam relacionados, em vez de se concentrar puramente no que deveria ser analisado. Esses vieses são difíceis de perceber e os modelos normalmente não possuem abertura para mostrar que isso está ocorrendo.

O artigo enfatiza que a interpretabilidade não deve ser vista como um objetivo secundário, mas como uma característica essencial que deve ser incorporada desde o início do desenvolvimento do modelo. A contribuição central do artigo é a apresentação de diversos métodos que podem ser aplicados para gerar alguma interpretabilidade dos modelos. Estes são divididos em diferentes categorias, com base no critério do modelo e na interpretabilidade global e local.

Métodos como LIME e SHAP são apresentados, os quais permitem a interpretação local de previsões, ajudando os usuários a entender quais características influenciaram uma decisão específica. Outro grande ponto é o trade-off que muitas vezes ocorre entre interpretabilidade e performance, onde é comum preferir sacrificar a acurácia do modelo em troca de sua interpretação, devido aos modelos mais complexos serem tendenciosamente mais opacos.

## Assista ao documentário "Coded Bias" da NETFLIX e faça um breve resumo do que você entendeu do assunto.
## Qual a relação deste documentário com o artigo anterior?

O documentário explora as consequências sociais da inteligência artificial e o algoritmo de reconhecimento fácil. A principal motivação dessa exploração nasce ao perceber que a maioria dos softwares de reconhecimento facial existentes não identifica corretamente rostos de pessoas de cor, em especial mulheres negras.

Isso desencadeia uma investigação sobre o viés incorporado em algoritmos, que muitas

vezes são desenvolvidos com dados não representativos, resultado em discriminação e injustiças. O documentário também mostra como estas tecnologias são implementadas sem regulamentação adequada, afetando os direitos civis, e assim mostra a necessidade urgente de responsabilidade e transparencia dentro da IA, bem como a implementação de leis para proteger os cidadãos contra preconceitos e erros automatizados.

Assim como explicado no artigo, ambos abordam o problema central da existência de viés e falta de transparência em modelos de aprendizado de máquina. Enquanto o documentário possui um foco maior sobre as consequências sociais e éticas o artigo oferece uma perspectiva técnica sobre tanto como estes modelos podem falhar, seja em viés de dados como em má supervisão daqueles que o treinaram, tanto quanto métodos que podem ser utilizados para entender como estes problemas aconteceram e fornecer insights de como resolve-los.