RELATÓRIO SOBRE PERCEPTRON - REDE NEURAL COM APRENDIZADO SUPERVISIONADO

CIC 260 - INTELIGÊNCIA ARTIFICIAL

Nome:	Willian Saymon da Silva
Matrícula:	33962
Linguagem:	Python 3

PERCEPTRON

Perceptron é um algoritmo de aprendizado supervisionado para classificações binárias, ou seja, decide se uma entrada faz parte de uma classe ou não. Ele é um classificador linear, logo apenas problemas lineares podem ser resolvidos por eles, todavia utilizando uma implementação de múltiplas camadas, camadas de neurônios com alimentação indireta, é possível criar hiperplanos, permitindo assim a solução de problemas não lineares. O perceptron foi uma das primeiras redes neurais produzidas.

Um perceptron aprende conceitos, elo pode aprender a responder como verdadeiro ou falso pelas entradas que são apresentadas a ele, “estudando” e analisando a taxa de erro repetidamente os exemplos que lhe são apresentados. Com o treino e o estudo o perceptron pode calcular valores dos pesos e inclinação do vetor, ou reta, que resolvem o problema linear.

IMPLEMENTAÇÃO

O trabalho consiste na implementação de um perceptron que classifique entradas para as portas lógicas E e OU. O algoritmo que resolve os dois problemas é o mesmo. Vale ressaltar que é possível resolver a porta lógica OU sem a utilização de um BIAS e utilizando 0 como entrada e resposta esperada.

Segue abaixo o algoritmo, orientado a objetos, para solução dos problemas:

```
classe PerceptronE:

    método PerceptronE:
        entrada_padrao =    [[-1, -1, -1],
                            [-1, 1, -1],
                            [1, -1, -1],
                            [1, 1, 1]]
        padrao = 0
        limiar_t = 0
        fator_correcao = 0.2
        peso_1 = aleatorio(-1,1)
        peso_2 = aleatorio(-1,1)
        treino_completo = Verdade
        bias = -1

    método treino(){
        num_epoca = 0
        Enquanto (Verdade){
            treino_completo = Verdade
            Para padrao de 0 à  3 {
                somatória = calcula_somatoria()
                valor_obtido = ativacao(somatória)

                Se valor_obtido diferente de entrada_padrao[padrao][2]{
                treino_completo = Falso
                    erro =calcula_erro(valor_obtido)
                    atualiza_pesos(erro)
                }
            }
            Se treino_completo{
                Parar
            }
            num_epoca += 1
            Se num_epoca > 100{
                Parar
            }
        }
    }

    método calcula_somatoria(){
        retorna (peso_1 * entrada_padrao[padrao][0] ) + ( peso_2 *
                                                entrada_padrao[padrao][1]) + bias
    }

    método ativacao(somatória){
        Se somatória > limiar_t {
            retorna  1
        }
```

```
49.                     Se não{
50.                             retorna -1
51.                     }
52.             }
53.
54.             método calcula_erro(obtido){
55.                     retorna entrada_padrao[padrao][2] - obtido
56.             }
57.
58.              método atualiza_pesos(erro){
59.                     peso_1 = peso_1 + (entrada_padrao[padrao][0] * fator_correcao  * erro)
60.                     peso_2 = peso_2 + (entrada_padrao[padrao][1] * fator_correcao  * erro)
61.                     bias = bias + erro
62.             }
```

O algoritmo é baseado na lógica apresentada no livro Inteligência Artificial de Ben Coppin. Podemos resumir o algoritmo aos seguintes passos:

1. Inicializa os pesos(peso_1, peso_2), o bias(bias) e a taxa de aprendizagem.
2. Realizar a somatória das entradas.
3. Aplica a somatória na função de ativação.
4. Caso haja erro, calcula taxa de erro.
    4.1. Atualiza os pesos e o bias com base na taxa de erro.
5. Retorna ao passo dois para a próxima entrada

Tomando Época como um ciclo de treino com todos os exemplos disponíveis ao perceptron, o algoritmo termina quando não houver erro em uma Época. Caso o algoritmo execute durante 100 épocas ele tem sua execução parada pois nenhum resultado será obtido a partir de então, este valor de 100 Épocas foi escolhido arbitrariamente e pode variar.

Para a solução da porta ou, basta substituir os valores da entrada padrão, lembrando sempre que -1 equivale a 0, logo teremos:

```
entrada_padrao = [[-1, -1, -1],
                  [-1, 1, 1],
                  [1, -1, 1],
                  [1, 1, 1]]
```

BIBLIOGRAFIA:

COPPIN, Ben. **Artificial intelligence illuminated**. Jones & Bartlett Learning, 2004.