1 – O funcionamento do modelo McCulloch e Pitts pode ser dividido em quatro parte sendo entrada, combinação das entradas, ativação e a saída. A entrada são os valores externos inseridos na rede neural artificial, a combinação das entradas também conhecido como “net” é o somatório das entradas multiplicado pelo *w* (peso) com isso temos a próxima etapa que é chamada de ativação no qual o valor da combinação das entradas passa por uma função que irar falar se o valor do neurônio atual será 0 ou 1 no modelo McCulloch e Pitts essa função de ativação é conhecida com função degrau sendo dada se o net for maior que 0 então o valor é 1 e se for menor que 0 então é 0, por último temos a saída que irá predizer a resposta para aquele conjunto de entrada.

2 – Nas redes neurais artificiais a aprendizagem é feita durante o processo de treinamento no qual o a rede neural artificial tenta predizer a saída e caso tenha predito o valor errado os pesos sofrem um ajuste assim realizando a aprendizagem, no caso do modelo McCulloch e Pitts o ajuste é dado pela seguinte formula Wij(n+1) = Wij + n \* Xi(yd -y).

As redes neurais possuem basicamente três tipos de aprendizagem, supervisionado, não supervisionado e por reforço, a aprendizagem supervisionada é quando a rede neural aprende a partir de dados definidos com entrada e saída assim o treinamento é feito através do erro entre o valor esperado e o valor predito pela rede neural, já o ultimo o aprendizado não supervisionado os dados não possui a saída esperada assim o modelo tenta separar os dados em categorias diferente do supervisionado que possui o valor de saída esperado por fim o aprendizado por reforço no qual o treinamento é feito para o sistema tomar uma sequência de decisões durante o seu aprendizado a rede neural artificial recebe feedback sobre o erro cometido durante a sequência assim a rede neural tenta maximizar os erros.

3 – O perceptron é uma rede neural muito similar ao modelo McCulloch e Pitts podem aceitar valores não binários na entrada assim tornando um pouco mais flexível, o perceptron é um modelo matemático que prever um valor binário na sua saída assim classificando os dados da entrada. A função de ativação utilizada no perceptron é chamada de função degrau onde se o valor for maior que 0 então a saída vai ser 1 e se o valor for menor que 0 a saída vai ser 0 ou também a função degrau bipolar -1 a 0. O aprendizado do perceptron é feito através do seu treinamento essa etapa quando o valor da saída previsto for diferente do valor esperado é feito um ajuste no peso dado pela seguinte formula Wij(n+1) = Wij + n \* Xi(yd -y) sendo (novo peso = peso atual + taxa de aprendizado \* a entrada (valor de saída esperado – valor de saída previsto)), esse ajuste acontece para todos os pesos logo após é passado para próxima previsão e assim ajustando os pesos caso necessário durante o ciclo.

4 –

|  |  |  |  |
| --- | --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82 | *# -\*- coding: utf-8 -\*-*  *"""*  *Created on Sun Mar 14 15:37:30 2021*  *@author: Ezequiel*  *"""*  **import** **numpy** **as** **np**  taxa\_apredizado = 0.1  entradas\_saidas = np.array([  [1, 1, 0, 1],  [1, 0, 1, 1],  [1, 1, 1, 1],  [1, 0, 0, 0],  ])  peso = np.array([-0.5, 0, 0])  **def** entrada(x):  xx = entradas\_saidas[x][0:-1]  **return** xx  **def** resposta(x):  xx = entradas\_saidas[x][-1:]  **return** xx  **def** ajuste\_pesos(i, ii, saida\_encontrada):  novo\_w = float(peso[ii]) + taxa\_apredizado \* (entrada(i)  [ii] \* (resposta(i) - round(float(saida\_encontrada), 2)))  novo\_w = round(novo\_w[0], 2)  **return** novo\_w  **def** funcao\_ativacao(valor):  **if**(valor > 0):  **return** 1  **else**:  **return** 0  **print**('Peso inicial: '+str(peso))  **def** treinamento():  **global** peso  nao\_teve\_ajuste = 0  **for** i **in** range(0, len(entradas\_saidas)):  somatorio = np.dot(entrada(i), peso)  saida\_encontrada = funcao\_ativacao(somatorio)  **if**(saida\_encontrada != resposta(i)):  novos\_pesos = []  nao\_teve\_ajuste = 1  **for** ii **in** range(0, len(entrada(i))):  novos\_pesos.append(ajuste\_pesos(i, ii, saida\_encontrada))  peso = novos\_pesos  **print**('Novo peso: ' + str(peso))  **if**(nao\_teve\_ajuste):  treinamento()  treinamento()  **print**('**\n** Rede treinada: ' + str(peso))  **print**('**\n** Teste **\n**')  **for** i **in** range(0, len(entradas\_saidas)):  valor\_encontrado = np.dot(entrada(i), peso)  saida\_encontrada = funcao\_ativacao(valor\_encontrado)  **if**(saida\_encontrada == resposta(i)):  **print**('Acertou - saida encontrada: ' + str(saida\_encontrada))  **else**:  **print**('err')  5 –   |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102 | **import** **numpy** **as** **np**  taxa\_apredizado = 0.4  entradas\_saidas = np.array([  [1, 1, 1, 0, 1, 0],  [1, 0, 0, 1, 0, 1],  [1, 1, 1, 0, 0, 1],  [1, 1, 0, 1, 1, 0],  [1, 1, 0, 0, 1, 1],  [1, 0, 0, 1, 1, 0],  ])  peso = np.array([0.1, 0.2, -0.2, -0.2, -0.3])  **def** entrada(x):  xx = entradas\_saidas[x][0:-1]  **return** xx  **def** resposta(x):  xx = entradas\_saidas[x][-1:]  **return** xx  **def** ajuste\_pesos(i, ii, saida\_encontrada):  novo\_w = float(peso[ii]) + taxa\_apredizado \* (entrada(i)[ii] \*  (resposta(i) - round(float(saida\_encontrada), 2)))  novo\_w = round(novo\_w[0], 2)  **return** novo\_w  **def** funcao\_ativacao(valor):  **if**(valor > 0):  **return** 1  **else**:  **return** 0  **print**('Peso inicial: '+str(peso))  **def** treinamento():  **global** peso  nao\_teve\_ajuste = 0  **for** i **in** range(0, len(entradas\_saidas)):  somatorio = 0  **for** jj **in** range(0, len(entrada(i))):  somatorio = somatorio + entrada(i)[jj] \* peso[jj]  saida\_encontrada = funcao\_ativacao(somatorio)  **if**(saida\_encontrada != resposta(i)):  novos\_pesos = []  nao\_teve\_ajuste = 1  **for** ii **in** range(0, len(entrada(i))):  novos\_pesos.append(ajuste\_pesos(i, ii, saida\_encontrada))  peso = novos\_pesos  **print**('Novo peso: ' + str(peso))  **if**(nao\_teve\_ajuste):  treinamento()  treinamento()  entradas\_saidas = np.array([  [1, 1, 1, 0, 1, 0],  [1, 0, 0, 1, 0, 1],  [1, 1, 1, 0, 0, 1],  [1, 1, 0, 1, 1, 0],  [1, 1, 0, 0, 1, 1],  [1, 0, 0, 1, 1, 0],  [1, 1, 1, 1, 1, 0],  [1, 0, 0, 0, 0, 1],  ])  **print**('**\n** Rede treinada: ' + str(peso))  **print**('**\n** Teste **\n**')  paciente = 'XXX'  **for** i **in** range(0, len(entradas\_saidas)):  valor\_encontrado = 0  **for** jj **in** range(0, len(entrada(i))):  valor\_encontrado = valor\_encontrado + entrada(i)[jj] \* peso[jj]  saida\_encontrada = funcao\_ativacao(valor\_encontrado)  **if**(saida\_encontrada == 0):  paciente = 'Doente'  **else**:  paciente = 'Saudável'  **if**(saida\_encontrada == resposta(i)):  **print**('Acertou - saida encontrada: ' +  str(saida\_encontrada) + ' - ' + paciente)  **else**:  **print**('err') | |

6 –

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 | **import** **numpy** **as** **np**  taxa\_apredizado = 0.3  entradas\_saidas = np.array([  [1, 0, 0, 1, 0],  [1, 1, 1, 0, 1],  ])  peso = np.array([0.5, 0.4, -0.6, 0.6])  **def** entrada(x):  xx = entradas\_saidas[x][0:-1]  **return** xx  **def** resposta(x):  xx = entradas\_saidas[x][-1:]  **return** xx  **def** ajuste\_pesos(i, ii, saida\_encontrada):  novo\_w = float(peso[ii]) + taxa\_apredizado \* (entrada(i)  [ii] \* (resposta(i) - round(float(saida\_encontrada), 2)))  novo\_w = round(novo\_w[0], 2)  **return** novo\_w  **def** funcao\_ativacao(valor):  **if**(valor > 0):  **return** 1  **else**:  **return** 0  **print**('Peso inicial: '+str(peso))  **def** treinamento():  **global** peso  nao\_teve\_ajuste = 0  **for** i **in** range(0, len(entradas\_saidas)):  somatorio = np.dot(entrada(i), peso)  saida\_encontrada = funcao\_ativacao(somatorio)  **if**(saida\_encontrada != resposta(i)):  novos\_pesos = []  nao\_teve\_ajuste = 1  **for** ii **in** range(0, len(entrada(i))):  novos\_pesos.append(ajuste\_pesos(i, ii, saida\_encontrada))  peso = novos\_pesos  **print**('Novo peso: ' + str(peso))  **if**(nao\_teve\_ajuste):  treinamento()  treinamento()  **print**('Rede treinada: ' + str(peso))  valore\_teste = np.array([  [1, 1, 1, 1],  [1, 0, 0, 0],  [1, 1, 0, 0],  [1, 0, 1, 1],  ])  **print**('**\n** **\n** A) **\n**')  **for** i **in** range(0, len(entradas\_saidas)):  valor\_encontrado = np.dot(entrada(i), peso)  saida\_encontrada = funcao\_ativacao(valor\_encontrado)  **if**(saida\_encontrada == resposta(i)):  **print**('Acertou - saida encontrada: ' + str(saida\_encontrada))  **else**:  **print**('errou')  **def** entrada\_teste(x):  xx = valore\_teste[x]  **return** xx  **def** resposta\_teste(x):  xx = valore\_teste[x][-1:]  **return** xx  **print**('**\n** **\n** B) **\n**')  **for** i **in** range(0, len(valore\_teste)):  valor\_encontrado = np.dot(entrada\_teste(i), peso)  saida\_encontrada = funcao\_ativacao(valor\_encontrado)  **print**('Saida encontrada: ' +  str(valore\_teste[i][1:]) + ' ' + str(saida\_encontrada)) |