CENTRO UNIVERSITÁRIO
UNI METROCAMP

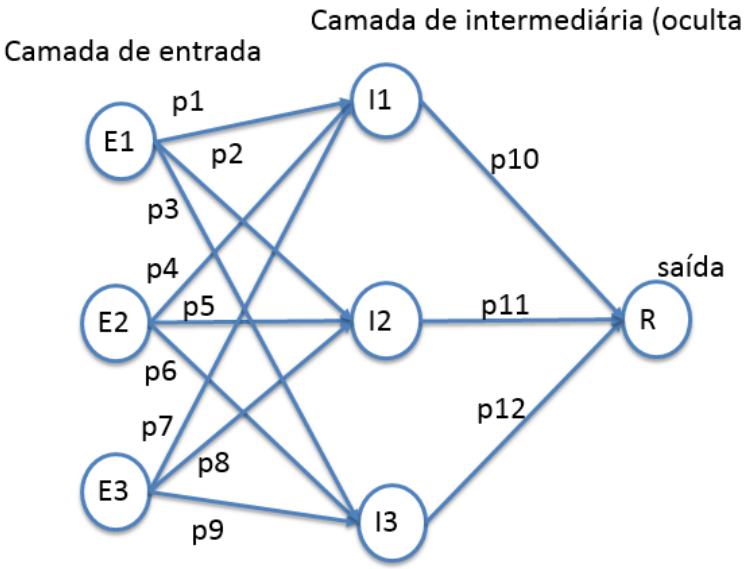
Inteligência Artificial

Prof Eduardo Nunes

Aluno: Marcelo Pedroni da Silva

RA: 202051855029

Prova AV2 – 5 pontos



E1	E2	E3	saída
0	0	0	0
1	0	0	1
0	1	0	0
1	1	0	0
0	0	1	1
1	0	1	1
0	1	1	0
1	1	1	0

A tabela verdade de um sistema está descrita na figura acima, com as 3 entradas (E1, E2 e E3) e respectiva saída. Uma rede neural com uma camada intermediária de 3 neurônios foi modelada como na figura. Nas redes neurais artificiais (RNA) o conhecimento é expressado nos pesos (sinapses) e treinar uma RNA é realizar o melhor ajuste dos pesos para a resposta que desejamos.

Baseado no algoritmo em PYTHON a seguir, realize as seguintes atividades:

1 – faça o treinamento desta RNA com 100 repetições (épocas) e taxa de aprendizagem de 1. Anote os valores do erro médio e os valores da camada de saída calculada. (1 ponto)

R:

main.py

```
1
2
3   import numpy as np
4
5
6   entradas = np.array([[0,0,0],
7                        [1,0,0],
8                        [0,1,0],
9                        [1,1,0],
10                       [0,0,1],
11                       [1,0,1],
12                       [0,1,1],
13                       [1,1,1]])
14
15  saidas = np.array([[0],[1],[0],[0],[1],[1],[0],[0]])
16
17
18  pesos0 = 2*np.random.random((3,3)) - 1
19  pesos1 = 2*np.random.random((3,1)) - 1
20
21  epocas = 100
22  taxaAprendizagem = 1
23
```

```
camada saida calculada:
[[0.25012256]
 [0.84188897]
 [0.00552345]
 [0.01247745]
 [0.84084177]
 [0.94401995]
 [0.01276695]
 [0.09742673]]
Erro: 0.09394580569690145
```

2 – faça o treinamento desta RNA com 10.000 repetições (épocas) e taxa de aprendizagem de 1. Anote os valores do erro médio e os valores da camada de saída calculada. (1 ponto)

R:

main.py

```
1
2
3   import numpy as np
4
5
6   entradas = np.array([[0,0,0],
7                        [1,0,0],
8                        [0,1,0],
9                        [1,1,0],
10                       [0,0,1],
11                       [1,0,1],
12                       [0,1,1],
13                       [1,1,1]])
14
15  saidas = np.array([[0],[1],[0],[0],[1],[1],[0],[0]])
16
17
18  pesos0 = 2*np.random.random((3,3)) - 1
19  pesos1 = 2*np.random.random((3,1)) - 1
20
21  epocas = 10000
22  taxaAprendizagem = 1
23
```

```
camada saida calculada:
[[1.70904652e-02]
 [9.90266892e-01]
 [1.67665822e-06]
 [2.29699365e-05]
 [9.90280402e-01]
 [9.96780498e-01]
 [2.24025769e-05]
 [3.96942962e-03]]
Erro: 0.00547239401042236
```

3 – compare e discuta os valores obtidos no treinamento de 100 e 10.000 repetições. (2 pontos)

R: Com apenas 100 iterações, podemos perceber que os resultados obtidos são relativamente próximos aos esperados, com erro médio de 0,09 arredondando para 0,1. Se colocarmos em taxas, representaria 10%, embora seja um erro alto, se pensarmos que o custo computacional foi extremamente baixo é um resultado aceitável. A saída esperada de [0, 1, 0, 0, 1, 1, 0, 0] teve como resultado [0.250, 0.841, 0.005, 0.012, 0.840, 0.944, 0.012, 0.097].

Com 10000 iterações, nossa rede neural teve a oportunidade de aprimorar muito os pesos e alcançar resultados muito mais interessantes sob uma mesma taxa de aprendizado (1). Ao observarmos o erro médio em 0,005 ou seja 0,5% conseguimos compreender que tal correlação entre taxa de aprendizado e número de iterações (epocas) é muito mais apropriada do que com apenas 100 iterações. A saída foi:

[$1.70*10^{-2}$, $9.90*10^{-1}$, $1.67*10^{-6}$, $2.29*10^{-5}$, $9.90*10^{-1}$, $9.96*10^{-1}$, $2.24*10^{-5}$, $3.96*10^{-3}$]

(Números extremamente próximos aos esperados).

Mais uma vez cabe ressaltar que dado o exemplo didático ser pouco complexo, o custo computacional de “rodar” o algoritmo com ambas as condições é extremamente baixo, logo para esta realidade o resultado mais acurado é sempre a melhor escolha.

Mas, se transpormos para uma situação do cotidiano profissional, muitas vezes há de se avaliar os limites que são consideráveis uma relação de custo computacional e resultados esperados, pois algoritmos complexos de redes neurais podem necessitar de milhares de iterações e são processos longos e complexos, portanto esta é uma questão ainda muito importante a ser decidida em projetos de IA.

4 – O treinamento desta RNA está sendo feito pelo método de *backpropagation*, com base nos resultados obtidos e na teoria, explique como este método funciona. (1 ponto)

R: O treinamento desta rede neural realizado com o método de backpropagation nos revela primeiramente que sob a mesma taxa de aprendizado, os resultados serão mais acurados quanto mais vezes a rede for treinada. Faz sentido, uma vez que a taxa de aprendizado ao se manter constante, será aprimorada mais e mais com as interações.

O método de backpropagation se baseia em corrigir o sistema de pesos a cada iteração, visando se aproximar sempre mais do resultado esperado, avaliando os resultados, calculando os erros e melhorando à uma determinada taxa de aprendizagem determinada.

Este método de algoritmo é útil para solucionar problemas que não são lineares, onde não se consegue determinar padrões lineares de soluções.

Uma vantagem deste método é conseguir empregar multicamadas de “neurônios” com pesos diferentes, podendo ser pré-determinados ou não (aleatórios de início).

Outra vantagem é o seu custo computacional bastante reduzido frente a outros algoritmos que utilizam outros métodos. Seu custo computacional (de forma muito reducionista) é calculado basicamente sobre 2 “iterações” na rede, uma para frente e uma para “trás”, de tal forma que uma multiplica-se pela matriz de peso determinada no algoritmo e a outra multiplica-se pela transposta desta mesma matriz.