**UFOP – Universidade Federal de Ouro Preto Instituto de Ciências Exatas e Aplicadas** Departamento de Computação e Sistemas

**Dupla:**

**Aluno:** Marcos Vinicius Timoteo Nunes **Matricula:** 16.2.8388

**Aluno:** Eufrasio Junior **Matricula:** 17.1.8131

**Curso:** Sistemas de Informação

**Professora:** Talles **Disciplina**: Inteligencia Artificial

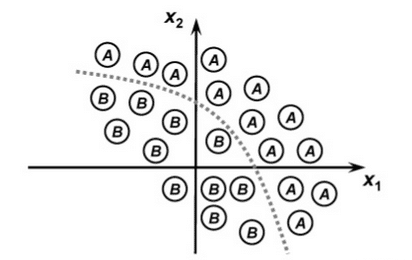
**Relatório:**

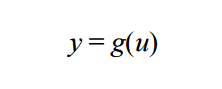
**Trabalho sobre Redes Neurais Perceptron e aplicações usando linguagem Python**

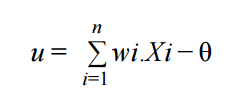
O Perceptron foi criado em 1958 por Rosenblatt, sendo a forma mais simples da configuração de uma rede neural artificial, uma vez que é constituída de uma única camada neural e de um único neurônio. Perceptron de única camada é um dos exemplos de implementações do algoritimo, nele podemos ter N entradas, mas apenas uma única saída com um valor de 0 e 1 ou de -1 e 1.Hoje, é mais comum usar outros modelos de neurônios artificiais, mas o Perceptron permite uma compreensão clara de como funciona uma rede neural em termos matemáticos, sendo uma excelente introdução.

o Perceptron possui três entradas: x1, x2, x3. Rosenblatt propôs uma regra simples para calcular a saída. Ele introduziu pesos, w1, w2, …, números reais expressando a importância das respectivas entradas para a saída. A saída do neurônio, 0 ou 1, é determinada pela soma ponderada, Σjwjxj, menor ou maior do que algum valor limiar (threshold). Assim como os pesos, o threshold é um número real que é um parâmetro do neurônio.

O funcionamento da rede Perceptron é muito simples, as entradas (Xi) representam as informações do processo que desejamos mapear, sendo que cada uma das entradas terá um peso sináptico ponderado (Wi) que representa a importância de cada entrada em relação ao valor de saída desejado (y). O resultado da somatória das entradas ponderadas será somado ao limiar de ativação (θ) e então repassado como argumento da função de ativação g(.), a qual terá como resultado a saída desejada. Normalmente a função de ativação costuma ser do tipo função degrau ou degrau bipolar. Representando matematicamente teremos:



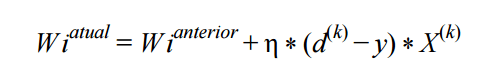




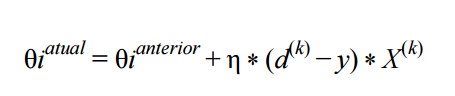
Simbologia:

* Xi - Entradas - Valor real ou binário
* Wi - Pesos Sinápticos - Valor Real aleatório
* θ - Limiar de Ativação - Valor Real aleatório
* y - Saída - Valor Binário
* g(.) - Função de Ativação - Degrau ou Degrau Bipolar

Os pesos sinápticos, assim como o limiar do Perceptron, serão corrigidos através do processo de treinamento supervisionado por meio da regra de aprendizado de Hebb, para cada amostra dos sinais de entrada. Se a saída produzida coincidir com a saída desejada, os pesos sinápticos e os limiares serão incrementados proporcionalmente aos valores de seus sinais de entrada. Do contrário, caso a saída seja diferente, seus valores serão decrementados. Este processo irá se repetir sequencialmente para todas as amostras de treinamento, até que a saída final seja similar à desejada, conforme as equações a seguir:



e



Onde:

* η  representa a taxa de aprendizado ou a velocidade com que o processo de treinamento converge rumo à estabilidade, sendo que o valor de η deverá estar compreendido entre 0 < η < 1;
* K representa a k-ésima amostra.

Já a saída do Perceptron assumirá dois valores possíveis. Supondo que desejemos trabalhar com a função sinal, teríamos:



Ou

Perceptron-formula6

Resultados Obtidos:

Taxa de aprendizagem = 0.01 – Teste 1:

Matriz de confusão:

1120 25

455 241

Especificidade: 99%

Sensibilidade: 37%

Taxa de aprendizagem = 0.01 – Teste 2:

Matriz de confusão:

920 180

65 676

Especificidade: 84%

Sensibilidade: 92%

Taxa de aprendizagem = 0.1 – Teste 3

Matriz de confusão:

878 143

73 683

Especificidade: 71%

Sensibilidade: 89%

Taxa de aprendizagem = 0.1 – Teste 4

Matriz de confusão:

669 371

11 579

Especificidade: 59%

Sensibilidade: 88%

Taxa de aprendizagem = 0.1 – Teste 5

Matriz de confusão:

659 366

9 556

Especificidade: 56%

Sensibilidade: 87%