# Atividade Prática

| | |
|---|---|
| **Disciplina** | **MAP Bootcamp** |
| **Atividade** | **Rede Perceptron** |

## 1. Objetivos

Vamos rodar o programa em Python 3.x para treinar uma rede Perceptron. A ideia é ter uma rede que responda às entradas para:

- quadrupede = 1
- bipede = -1

Valores para treinamento da rede:

cao = [-1,-1,1,1] | resposta = 1

gato = [1,1,1,1] | resposta = 1

cavalo = [1,1,-1,1] | resposta = 1

homem = [-1,-1,-1,1] | resposta = -1

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# aplicativo para verificar se o ser vivo eh quadrupede ou bipede
# quadrupede = 1, bipede = -1
# cao = [-1,-1,1,1] | resposta = 1
# gato = [1,1,1,1] | resposta = 1
# cavalo = [1,1,-1,1] | resposta = 1
# homem = [-1,-1,-1,1] | resposta = -1

# pesos (sinapses)
w = [0,0,0,0]
# entradas
x = [[-1,-1,1,1],
     [1,1,1,1],
     [1,1,-1,1],
     [-1,-1,-1,1]]
# respostas esperadas
t = [1,1,1,-1]
# bias (ajuste fino)
b = 0
#saida
y = 0
# numero maximo de interacoes
max_int = 10
# taxa de aprendizado
taxa_aprendizado = 1
#soma
soma = 0
#theshold
threshold = 1
# nome do animal
animal = ""
# resposta = acerto ou falha
resposta = ""

# dicionario de dados
d = {'-1,-1,1,1' : 'cao',
     '1,1,1,1' : 'gato',
     '1,1,-1,1' : 'cavalo',
     '-1,-1,-1,1' : 'homem' }

print("Treinando")
```

```
44
45      # funcao para converter listas em strings
46      def listToString(list):
47          s = str(list).strip('[]')
48          s = s.replace(' ', '')
49          #print ("funcao" + s)
50          return s
51
52      # inicio do algoritmo
53      for k in range(1,max_int):
54          acertos = 0
55          print("INTERACAO "+str(k)+"------------------------")
56          for i in range(0,len(x)):
57              soma = 0
58
59              # pega o nome do animal no dicionário
60              if (listToString(x[i])) in d:
61                  animal = d[listToString(x[i])]
62              else:
63                  animal = ""
64
65              # para calcular a saida do perceptron, cada entrada de x eh multiplicada
66              # pelo seu peso w correspondente
67              for j in range(0,len(x[i])):
68                  soma += x[i][j] * w[j]
69
70              # a saida eh igual a adicao do bias com a soma anterior
71              y_in = b + soma
72              #print("y_in = ",str(y_in))
73
74              # funcao de saida eh determinada pelo threshold
75              if y_in > threshold:
76                  y = 1
77              elif y_in >= -threshold and y_in <= threshold:
78                  y = 0
79              else:
80                  y = -1
```

```
81
82              # atualiza os pesos caso a saida nao corresponda ao valor esperado
83              if y == t[i]:
84                  acertos+=1
85                  resposta = "acerto"
86              else:
87                  for j in range (0,len(w)):
88                      #    Peso ant + (tx aprendizado * valor esperado * valor do x no vetor)
89                      w[j] = w[j] + (taxa_aprendizado * t[i] * x[i][j])
90                      print ("Novo peso "+ str(j)+" "+str(w[j]))
91                  # bias ou erro = classe anterior - classe atual
92                  b = t[i] - y
93                  resposta = "Falha - Peso atualizado "+"Bias = "+str(b)
94              #imprime a resposta
95              if y == 1:
96                  print(animal+" = quadrupede = "+resposta)
97              elif y == 0:
98                  print(animal+" = padrao nao identificado = "+resposta)
99              elif y == -1:
100                 print(animal+" = bipede = "+resposta)
101
102         if acertos == len(x):
103             print("Funcionalidade aprendida com "+str(k)+" interacoes")
104             break;
105         print("")
106     print("Finalizado")
```

## 1. Conclusão

Ainda testando o notebook no Google Colab rodamos um programa que faz as interações em uma rede Perceptron até que tenhamos os pesos calibrados e a rede treinada.