Polish Japanese School of Information Technology

Subject: Artificial Intelligence Tools

Autor: Mateusz Gruszka

Index: s16251

Date: 30.10.18

Exercise 1

Analyze capability of a neuron to split surface. Write a program that calculate output signal from neuron that have two inputs and also implement two activation functions:

- threshold bipolar
- sigmoid bipolar (λ = 1).

Experiment plan:

- a) Take two random values that will be weights synaptyic of a neuron w1, w2 iand one bias value.
- b) Take 100 point from square [–5, 5] × [–5, 5].
- c) For every point calculate output signal.
- d) In case with threshold bipolar function:

  > - if output signal y = –1 then mark that point as yellow, in other cases - red.

- e) In case with sigmoid bipolar function:

  > - if output signal y ∈ (–1.0, –0.5), mark that point as yellow.
  > - if output signal y ∈ [–0.5, 0.0), mark that point as green.
  > - if output signal y ∈ [0.0, 0.5), mark that point as blue.
  > - if output signal y ∈ [0.5, 1.0), mark that point as red.

What is an observation?

SOLUTION

Step 1. Prepare random data for experiment

In [35]:

```
%matplotlib inline
from IPython.display import set_matplotlib_formats
set_matplotlib_formats('png', 'pdf')
import random
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import math

w1 = 0
w2 = 0
bias = 0

def prepareRandomData():
    w1 = random.uniform(-1, 1)
    w2 = random.uniform(-1, 1)
    bias = random.uniform(-1, 1)

#     print(w1)
#     print(w2)
#     print(bias)
```

Step 2. Prepare hundred points

In [36]:

```
class Point:

    def __init__(self, x, y):
        self.x = x
        self.y = y
```

In [37]:

```
def preparePoints(pointsNumber):
    pointsArr = []

    for x in range(0, pointsNumber+1):
        randomX = random.uniform(-5, 5)
        randomY = random.uniform(-5, 5)
        newPoint = Point(randomX, randomY)
        pointsArr.append(newPoint)
#         print(newPoint.__dict__)

    return pointsArr
```

Step 3. Prepare output

- For threshold bipolar function:

In [38]:

```
def bipolarFunc(value):
    if value >= 0:
        return 1
    else:
        return -1
```

- For sigmoid bipolar function:

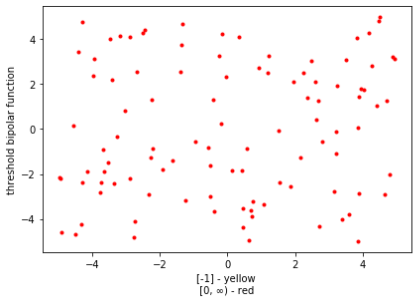
In [39]:

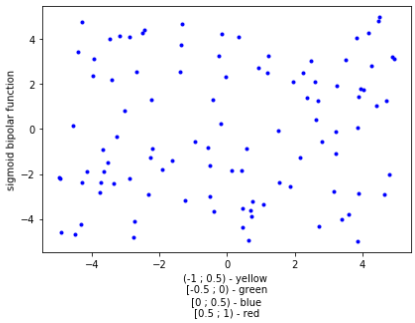
```
def sigmoidBipolarFunc(value):
    result = (2 / (1 + math.exp(-value*1))) - 1    # λ = 1
    return result
```

Step 4. Calculate

In [40]:

```
prepareRandomData()
pointsArr = preparePoints(100)

plt.rcParams['figure.figsize'] = [20, 10]
plt.subplot(2,2,1)

for point in pointsArr:
    resultFromActivation = 0
    singlePointResult = w1 * point.x + w2 * point.y + bias
    resultsFromActivation = bipolarFunc(singlePointResult)
    if(resultsFromActivation == -1):
        plt.plot(point.x, point.y, 'y.')
    elif(resultsFromActivation >= 0):
        plt.plot(point.x, point.y, 'r.')
plt.ylabel('threshold bipolar function')
plt.xlabel('[-1] - yellow \n [0, ∞) - red ')

plt.subplot(2,2,2)
plt.subplots_adjust(wspace=1)

for point in pointsArr:
    resultFromActivation = 0
    singlePointResult = w1 * point.x + w2 * point.y + bias
    resultsFromActivation = sigmoidBipolarFunc(singlePointResult)
    if(resultsFromActivation > -1 and resultsFromActivation < -0.5):
        plt.plot(point.x, point.y, 'y.')
    elif(resultsFromActivation >= 0.5 and resultsFromActivation < 0):
        plt.plot(point.x, point.y, 'g.')
    elif(resultsFromActivation >= 0 and resultsFromActivation < 0.5):
        plt.plot(point.x, point.y, 'b.')
    elif(resultsFromActivation >= 0,5 and resultsFromActivation < 1):
        plt.plot(point.x, point.y, 'r.')

plt.ylabel('sigmoid bipolar function')
plt.xlabel('(-1 ; 0.5) - yellow \n [-0.5 ; 0) - green \n [0 ; 0.5) - blue \n [0.
5 ; 1) - red ')

plt.show()
```





OBSERVATIONS

For the same data different activation functions give different outputs. We have to be accurate picking suitable activation function for the problem as well as gentle choosing other properties, like i.e. bias. The sigmoid function is able to output values between 0 and 1.