Statistical Theories for Brain and Parallel Computing

Assignment No.2

17M18819

ZHU LICHENG

## Description of Assignment

In assignment 1, we need to construct a probabilistic and binary recurrent neural network to implement “the winner takes all model”.

## Solution 1

***Deterministic and Binary Model***

I used python to implement this task. I created one class called **ass2\_determin.py** where we can call the function **determin\_rec(init)** to calculate the result with various initial values for [x1,x2,x3,x4,x5]. I called **determin\_rec** 4 times with different initial value of x and the result is as follows. Each time I call this function, it will reset the order of updating by randomness 4 times and output the results.

(The following parameters can be changed by hand when calling function **determin\_rec()**)

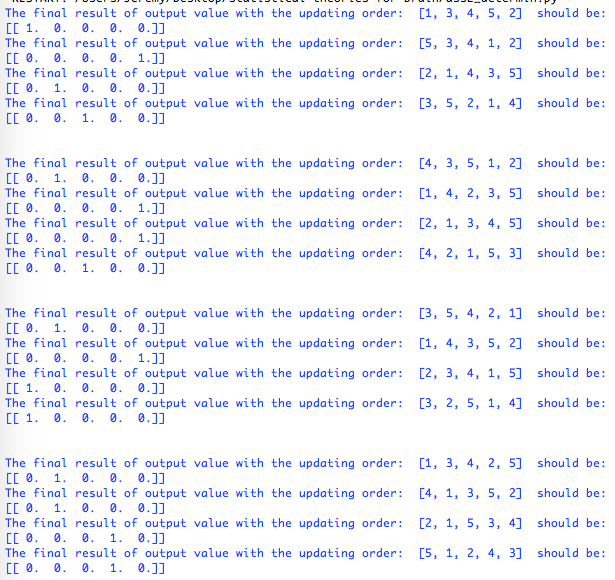
*determin\_rec([1,0,0,0,0,0])*

*determin\_rec([1,1,1,1,1,1])*

*determin\_rec([1,0.5,0.5,0.5,0.5,0.5])*

*determin\_rec([1,0.5,1,0.5,1,0.5])*

The running result is shown as follows:



***Probabilistic and Binary Model***

I also used python to implement this task in **ass2\_prob.py.** The main function **prob\_rec(init)** is very similar to the deterministic one except there is probability for a neuron to be updated instead of deterministic. I applied sigmoid function to it and randomly choose a neuron for updating for 10000 times. I need to calculate the distance between the 10000 results and (1,0,0,0,0). If the distance is smaller than 2, then we can increase the number of vectors that come close to (1,0,0,0,0) by 1. I called **prob\_rec()** two times and run the program also two times the result is as follows.

(The following parameters can be changed by hand when calling function **prob\_rec()**)

prob\_rec([1,1.3,1.2,0.7,1.2,1.0])

prob\_rec([1,1.3,0,0,0,1.0])

The running result is shown as follows:

