Eric Christopher / 15M53531

I use deterministic recurrent neural network. Last time, I used parameter that the looping will stop after the first loop. And with this experience, I think that is not enough, so I modify it to :

> the looping will stop after get (size of node+1 or 1 looping) times of same value.

Just to make sure.

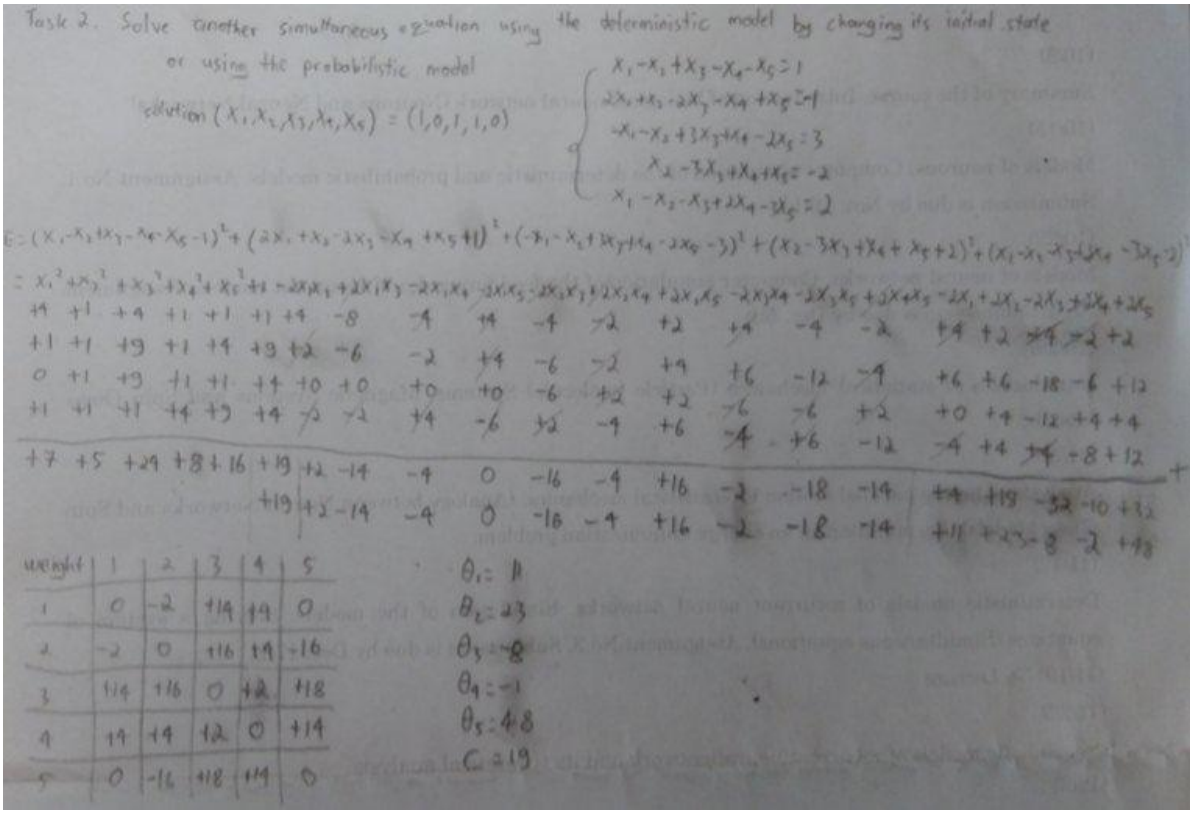
In this report I use input that opposite to the result to see how it works. At first I assume that the input with opposite of the result will has the longest step to become stable (when node are not changing anymore), but it seems my assumption is wrong. Look at task 1, input (0,0,1) (66,67% wrong) has more step than input (0,0,0) (100% wrong). So the step count is not determined by how many differences between the input and result. If it was a probabilistic recurrent neural network, we don't know the step count, because it is depend on our "luck". It can be faster but it can be slower. But with probabilistic model, we can avoid the local minimum.

Task 1

Initial input: (0,0,0) – opposite to the result	Initial input (0,0,1)
Result: --(0,0,0) - E=0.0 ========== -change node 1 --S=0.0 --(1,0,0) - E=0.0 ========== -change node 2 --S=6.0 --(1,1,0) - E=0.0 ========== -change node 3 --S=4.0 --(1,1,1) - E=-1.0 ========== -change node 1 -- 1st same value --S=2.0 --(1,1,1) - E=-1.0 ========== -change node 2 -- 2nd same value --S=14.0 --(1,1,1) - E=-1.0 ========== -change node 3 -- 3rd same value --S=4.0 --(1,1,1) - E=-1.0 ========== -change node 1 -- 4th same value --S=4.0 --(1,1,1) - E=-1.0	Result: --(0,0,1) - E=3.0 ========== -change node 1 --S=-4.0 --(0,0,1) - E=3.0 ========== -change node 2 --S=8.0 --(0,1,1) - E=1.0 ========== -change node 3 --S=8.0 --(0,1,1) - E=1.0 ========== -change node 1 --S=2.0 --(1,1,1) - E=-1.0 ========== -change node 2 -- 1st same value --S=14.0 --(1,1,1) - E=-1.0 ========== -change node 3 -- 2nd same value --S=4.0 --(1,1,1) - E=-1.0 ========== -change node 1 -- 3rd same value --S=2.0 --(1,1,1) - E=-1.0

========== ==========END========== ==========	========== -change node 2 -- 4th same value --S=2.0 --(1,1,1) - E=-1.0 ========== ==========END========== ==========

Task 2



Initial input: (0,1,0,0,1) – opposite to the result	Initial input (1,1,1,1,1)
Result: --(0,1,0,0,1) - E=87.0 ========== -change node 1 --S=-2.0 --(0,1,0,0,1) - E=87.0 ========== -change node 2 --S=-16.0 --(0,0,0,0,1) - E=48.0 ========== -change node 3 --S=18.0 --(0,0,1,0,1) - E=22.0	Result: --(1,1,1,1,1) - E=18.0 ========== -change node 1 --S=16.0 --(1,1,1,1,1) - E=18.0 ========== -change node 2 --S=2.0 --(1,0,1,1,1) - E=-3.0 ========== -change node 3 --S=34.0 --(1,0,1,1,1) - E=-3.0

========== -change node 4 --S=16.0 --(0,0,1,1,1) - E=4.0 ========== -change node 5 --S=32.0 --(0,0,1,1,0) - E=-12.0 ========== -change node 1 --S=18.0 --(1,0,1,1,0) - E=-19.0 ========== -change node 2 – 1st same value --S=18.0 --(1,0,1,1,0) - E=-19.0 ========== -change node 3 – 2nd same value --S=16.0 --(1,0,1,1,0) - E=-19.0 ========== -change node 4 – 3rd same value --S=6.0 --(1,0,1,1,0) - E=-19.0 ========== -change node 5 – 4th same value --S=32.0 --(1,0,1,1,0) - E=-19.0 ========== -change node 1 – 5th same value --S=18.0 --(1,0,1,1,0) - E=-19.0 ========== -change node 2 – 6th same value --S=18.0 --(1,0,1,1,0) - E=-19.0 ========== ==========END========== ==========	========== -change node 4 --S=20.0 --(1,0,1,1,1) - E=-3.0 ========== -change node 5 --S=32.0 --(1,0,1,1,0) - E=-19.0 ========== -change node 1 – 1st same value --S=18.0 --(1,0,1,1,0) - E=-19.0 ========== -change node 2 – 2nd same value --S=18.0 --(1,0,1,1,0) - E=-19.0 ========== -change node 3—3rd same value --S=16.0 --(1,0,1,1,0) - E=-19.0 ========== -change node 4 – 4th same value --S=6.0 --(1,0,1,1,0) - E=-19.0 ========== -change node 5 – 5th same value --S=32.0 --(1,0,1,1,0) - E=-19.0 ========== -change node 1 – 6th same value --S=32.0 --(1,0,1,1,0) - E=-19.0 ========== ==========END========== ==========