Introduction to Computer Science
HW #6
Due: 2018/06/20

Homework Rules:

Hand-written homework can be handed in **before lecture starts**. Otherwise, you may contact the TA in advance and then bring the hardcopy to the TA in MD-631 (please send e-mail in advance).

As for the programming part, you need to upload it to CEIBA before the deadline. The file you upload must be a **.zip** file that contains the following files:

README.txt

HW06_b04901XXX (a folder that contains all .cpp & .h as required),

1. Do not submit executable files (.exe) or objective files (.o, .obj). Files with names in wrong format will not be graded. You must **remove any system calls**, such as system ("pause"), in your code if any.

In README.txt, you need to describe which compiler you used in this homework, how to compile it (if it is in a "project" form) and **how do you generate training data.**

2. In your .cpp files, we suggest you write comments as detailed as you can. If your code does not work properly, code with comments earns you more partial credits.

Chapter 11 Review Problems (25%):

25, 44, 55.

Chapter 12 Review Problems (25%):

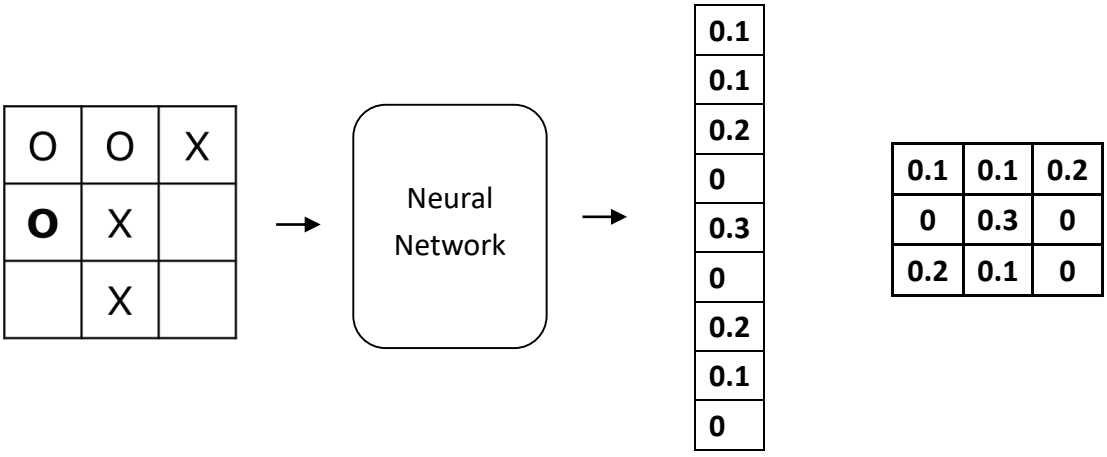
13, 17, 31

Introduction to Computer Science
HW #6
Due: 2018/06/20

Programming Problem I (50%):

In this homework, we are going to implement a neural network to paly tic-tac-toe. Here we already have a framework written in C++. What you need to do is to **design the network architecture** and to **provide training data.**

Input **Output**



What to do?

(1) Design network architecture

(2) Provide training data

(3) Set learning rate, epoch and batch size

How to start?

(1) Table representation:

State	One-hot encoding
Empty	1 0 0
Player	0 1 0
Opponent	0 0 1

Introduction to Computer Science
HW #6
Due: 2018/06/20

Example:

O	O	X
O	X	
	X	

Assume:
X: player
O: opponent

Assume:
X: player
O: opponent

The right table can be represented as:

0 0 1 0 0 1 0 1 0 0 0 1 0 1 0 1 0 0 1 0 0 0 1 0 1 0 0

O	O	X	**O**	X			X	

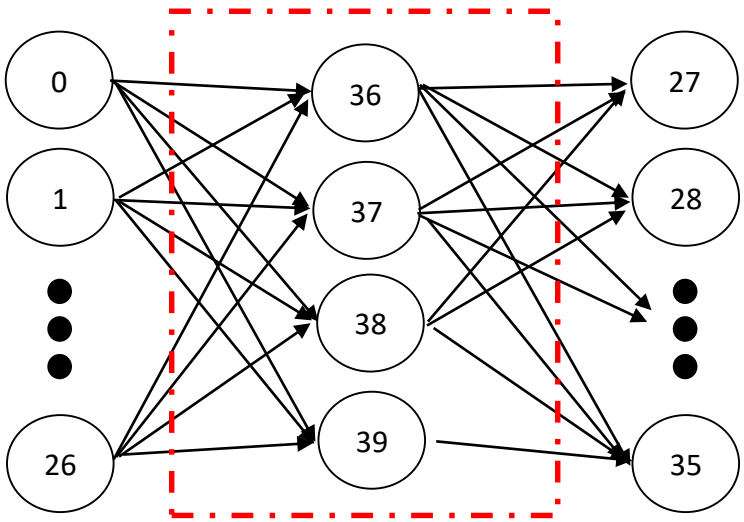
(2) Network architecture

Your network is composed of neurons. Each neuron has its own id.

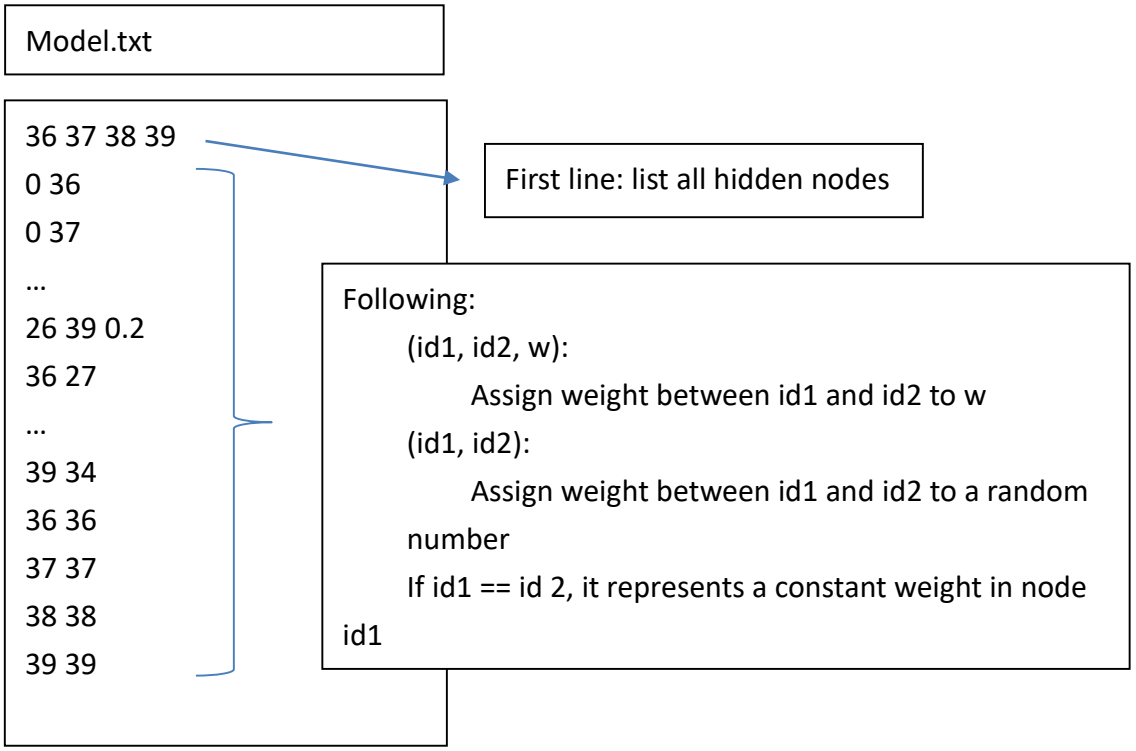
Input layer: id 0 ~ 26

Output layer: id 27~35

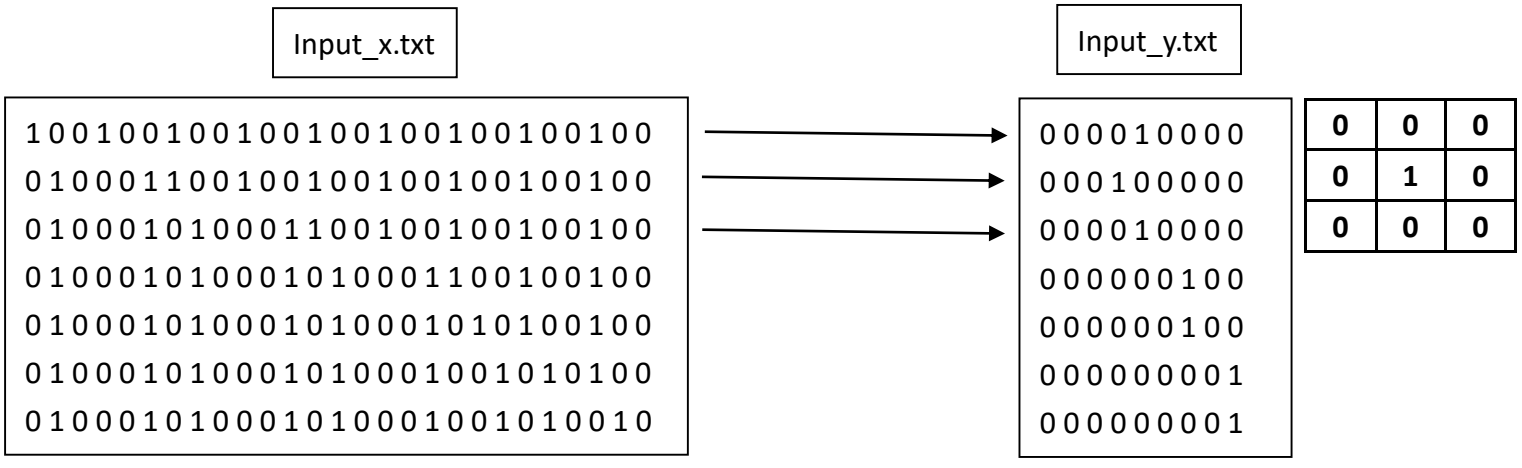
Please make sure that the network is a DAG (No cycle)



Introduction to Computer Science
HW #6
Due: 2018/06/20



(3) Generate training data



(4) Execute

Please compile with c++11

Introduction to Computer Science
HW #6
Due: 2018/06/20

./a.out learn_rate epochs batch_size model input_x input_y model_out test

learn_rate: double
epochs: int. Epochs of training.
batch_size: int. Batch size
model: string. Filename of network architecture
input_x: string. Filename of input_x
input_y: string. Filename of input_y
model_out: string. Filename of stored model after training.
test: int. Test model after training. 1 imply yes. 0 imply no.

Example:
./a.out 0.01 10 100 model.txt input_x.txt input_y.txt model_out.txt 1

Grading:

Basic:

(45%): (# of win + # of tie) / # of games >= 85%

(5%): (# of win + # of tie) / # of games >= 95%

Bonus:

(2%): (# of win + # of tie) / # of games = 100%

Submit:

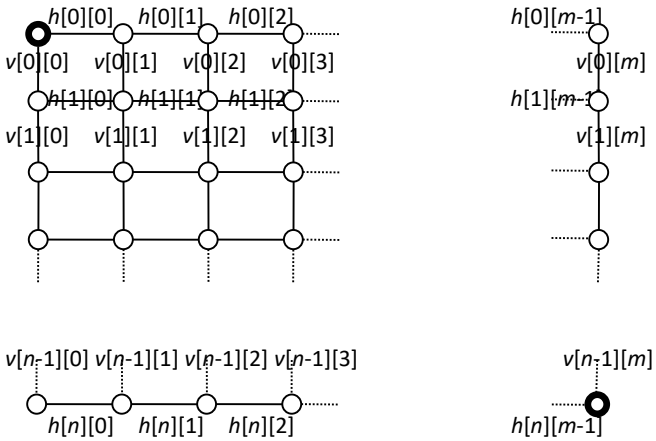
model.txt: your model
input_x.txt: the training data
input_y.txt: the training data

please describe how do you generate training data in readme file

Introduction to Computer Science
HW #6
Due: 2018/06/20

Bonus (3%): Minimax Search

Consider a graph of *m* by *n* grids:



The bonus problem is to write a minimax solver given the following situation: given a 2D grid with each edge having a positive weight. The goal of our agent is to move from one point to another designated point (to simplify the problem, from the lower left corner to the upper right corner) in the shortest sum of weight. However, there is another agent whose goal is to interfere our agent by removing one of our neighbor edges (change the weight to infinity) once we make a move given that we still can move to the goal. The routine would be move -> remove -> move -> remove -> ……

For the second part, you need to write a bonus.cpp/bonus.py and you need to take the depth of minimax search as a command line parameter and output the final decision by "u", "d", "l"or "r"along the heuristic score. The heuristic is the shortest path length from the current point to the goal. You can assume that the input depth would be even. The code helping to read the grid is given.

You can use **readParameters()** to read all parameters (*m*, *n*, *v*[][], and *h*[][]) from **input**. Remember to call release() when done. Check out **hw4.cpp / hw4.py** for more information.

Introduction to Computer Science
HW #6
Due: 2018/06/20

If you accomplished the bonus, save your code in **“bonus.cpp” (or bonus.py)** to hand in.

If you meet the bonus requirements, write “I finished the bonus part.” with further details (at least how to do the job) in the readme file to let TA know.