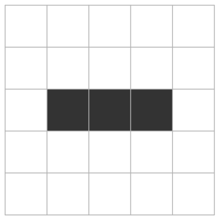
University of Helsinki
Department of Mathematics and Statistics
Neuronetworks February 2014
Exercises for week 1

1 Exercise. *Game of Life* is a dynamical system developed by J. Conway. The universe is an infinite two-dimensional orthogonal grid of square cells, each of which is in one of two possible states, alive or dead. Every cell interacts with its eight neighbours, which are the cells that are horizontally, vertically, or diagonally adjacent. At each step in time, the following transitions occur:

1. Any live cell with fewer than two live neighbours dies, as if caused by under-population.
2. Any live cell with two or three live neighbours lives on to the next generation.
3. Any live cell with more than three live neighbours dies, as if by overcrowding.
4. Any dead cell with exactly three live neighbours becomes a live cell, as if by reproduction.

Game of life can be made finite by replacing the infinite grid by a finite one, for example a square $n \times n$ or a toroidal lattice which is like $n \times n$ but the opposite walls are glued to each other. Show that in any case the Game of Life *cannot* be simulated by a (finite) Hopfield network no matter synchronous or not. Hint:



2 Exercise. Let n be any natural number greater than 1. Design a Hopfield network of n units whose only stable states are such that all neurons are in the same state (all 1 or all -1).

3 Exercise. Consider the network of fig. 1. All thresholds are equal to 0 and the states of each unit is either -1 or 1. The weights of the invisible connections are all 0. What are the stable states of the network? (No precise proof needed, heuristic explanation suffices.) Note the global behaviour despite only localised connections.

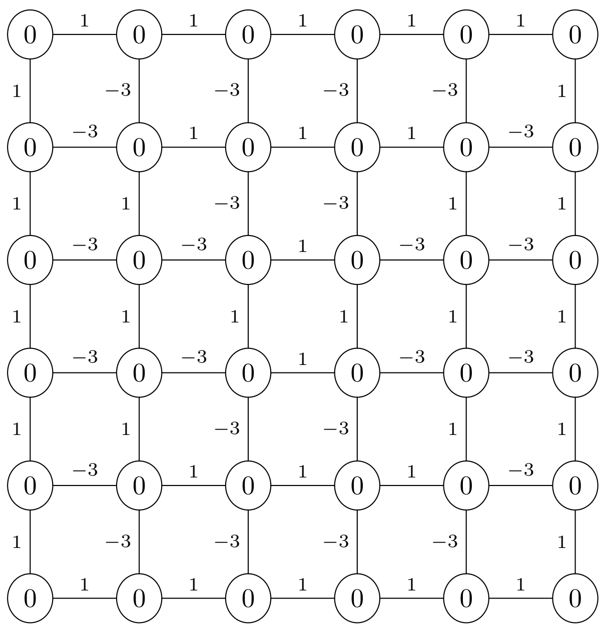


Figure 1: Network for Exercise 3

4 Exercise. Complete the code in the python file downloadable from the course web page. The code implements and visualises a Hopfield network which stores three pattern in a network with 100 units. Your part is to provide the code for the learning algorithm and the unit update function. The code requires a python package called *pygame* in order for the visualisation to work. There are more instructions in the file.