Patterns of Life HMW 7 **Francesco Pancaldi**
"Heart shape" perceptron **11/18/2014**

Introduction

Signal transduction is an important mechanism present in many living organism. This mechanism controls the production of one or more specific proteins depending on a certain number of chemical or physical inputs. One specific type is regulated by a so called "perceptron". A perceptron is a network of reaction between different components divided in "layers". The first layers is composed of the inputs and the last layer defines the outputs. The intermediate layers constitute the additional step required to translate the signal of the inputs into the output response. Each layer is providing an output that is received as an input from the elements in the next layer through a certain connection strength (either positive or negative). These interaction determine for each set of input if the production of the final protein is ON or OFF.

When we have two (or three inputs) we can draw a picture to represent when we have production (indicated with black or dotted region) and when we have no production (indicated with white or empty regions), we will call this the "activation pattern" of the perceptron.
In this homework the goal was to "evolve" a perceptron to obtain an "heart" shaped "activation pattern". To do so we have first developed a Matlab code to generate the "activation pattern" of a given perceptron, code to generate the wanted "heart pattern" and a couple of algorithm to evolve a perceptron to be more adherent to the wanted pattern. We will discuss these in the next section.

Despite our work we were not able to reach the wanted pattern or even a decent approximation. In the last section we will try to provide possible explanations as well as suggestions for possible future work.

Discussion and Results

We would like to "evolve" a perceptron to achieve an "activation pattern that is 'heart" shaped as in Fig.1 (obtained using testheart.m).
In this picture the shaped is obtained using the following restrictions for x1 and x2:
$$\left(\left(4\left(x_1-0.5\right)\right)^2+\left(4\left(x_2-0.4\right)\right)^2-1\right)^3-\left(4\left(x_1-0.5\right)\right)^2\left(4\left(x_2-0.4\right)\right)^3<0$$
but similar shapes could be obtained using other definitions.

Our first step was to write a program that will generate the activation pattern of a given opteron.
In this homework we restricted ourselves to opteron with multiple layers with two nodes for each layer and only one output. These can be generalized (and may in fact need to be generalized for our problem) to many nodes per layer.
The Matlab file testperceptron.m generates the activation pattern for a specific perceptron using the steady state solution and the threshold assumption that w1*x1+w2*x2 needs to be more than 1 to have activation (as seen in class 11/11/2014).
The code generates the picture testing a certain number of pairs (x1,x2) depending on the parameter xpts of the function testperceptron (xpts is the number of intervals in which we divide [0,1] for both x1 and x2).
In this code the perceptron is defined by a n-by-2 matrix, with n number of nodes (including the output). Each row correspond to the connection of that node with the two nodes one layer below.
Lets look at some examples to understand how to write the matrix corresponding to a perceptron.
We want to reproduce the pictures seen for the examples seen in class (Fig.2-4)

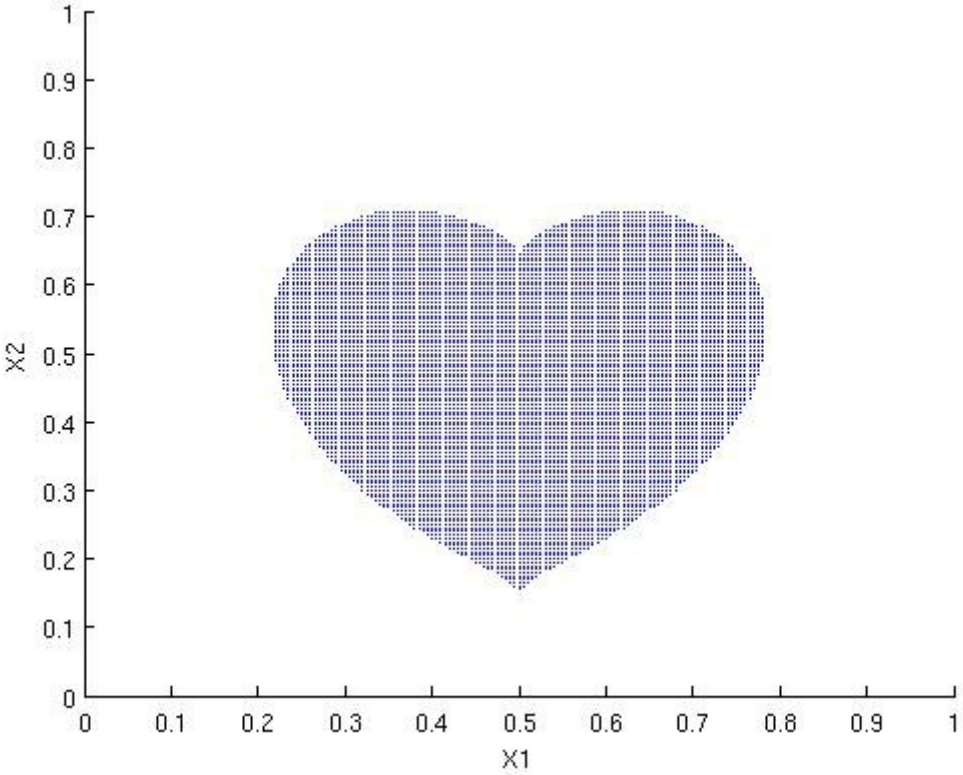
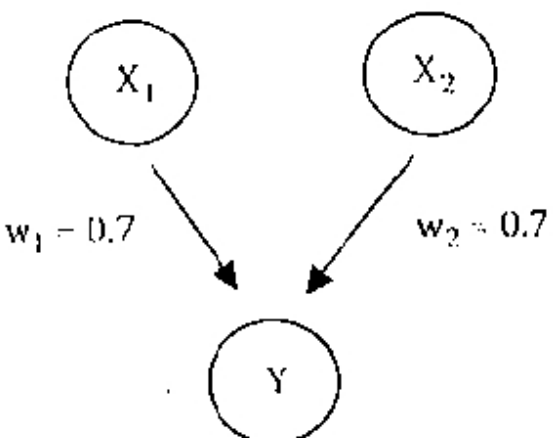
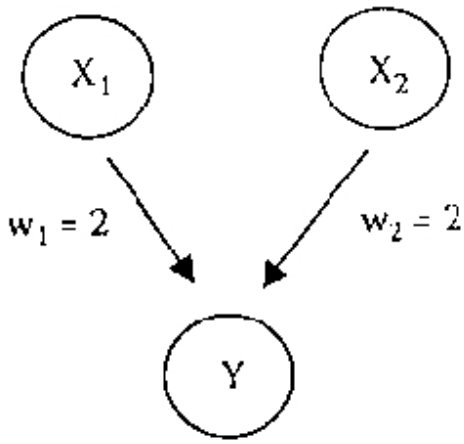
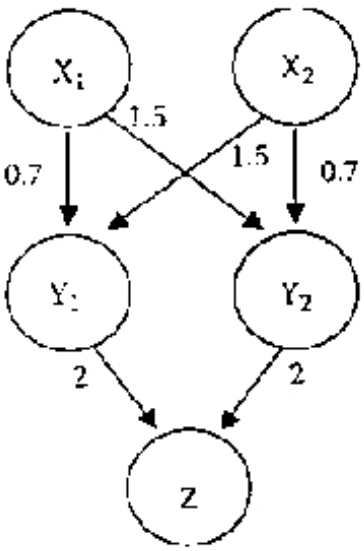
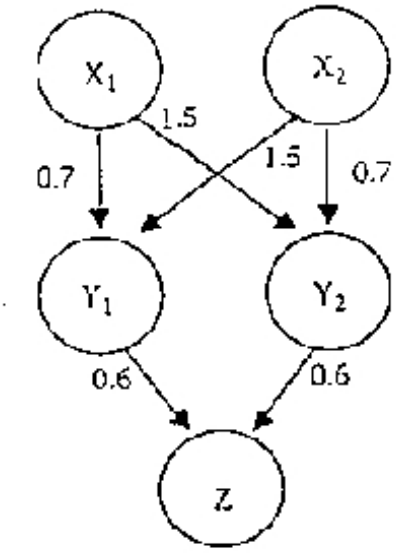


Illustration 1: "heart" shaped "activation pattern"









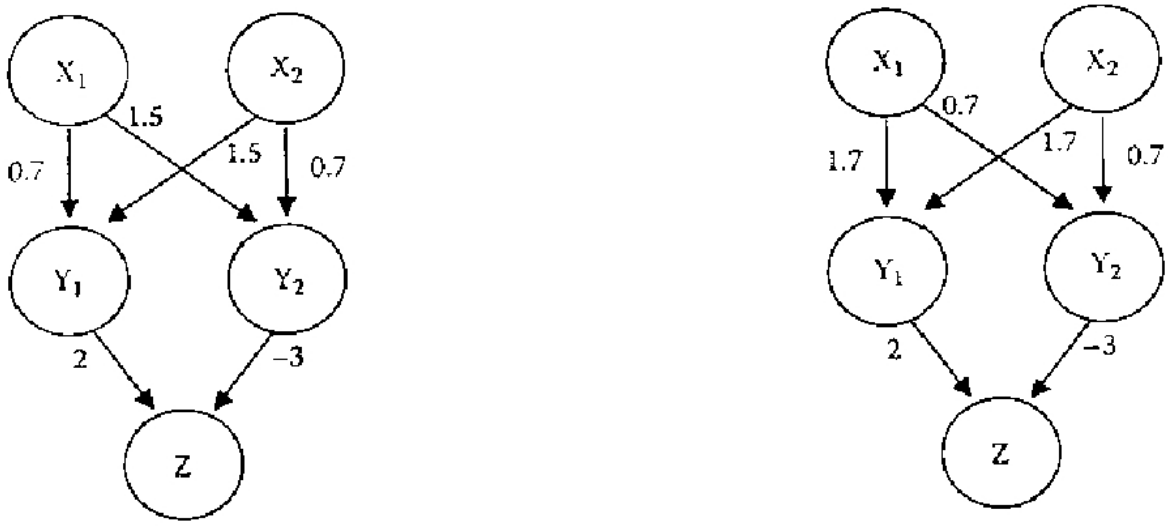


Illustration 4

This will correspond to the following matrices (from figure 2 to figure 4 from left to right).

|0.7 0|	|2 0|	|0.7 1.5|	|0.7 1.5|	|0.7 1.5|	|1.7 0.7|
|0.7 0|	|2 0|	|1.5 0.7|	|0.7 1.5|	|1.5 0.7|	|1.7 0.7|
|0 0|	|0 0|	|2 0 |	|0.6 0|	|2 0|	|2 0|
		|2 0 |	|0.6 0|	|-3 0|	|-3 0|
		|0 0 |	|0 0|	|0 0|	|0 0|

and this will generate the following pictures

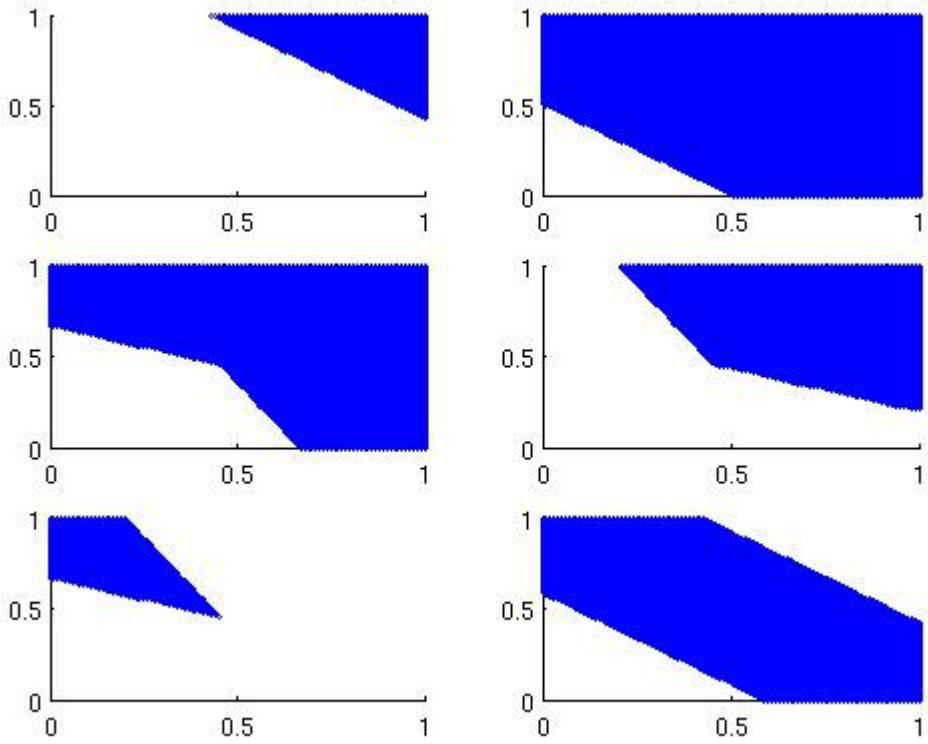


Illustration 5: activation patterns

To evolve a perceptron and obtain an activation pattern like in fig.1 we defined a fitness function that increases for each active point inside the heart that is active and decreases for each active point outside the heart region (fitness=#active_in/#heart_in - #active_out/#heart_out).

This fitness function has values between -1 and 1 (where -1 would correspond to the complementary picture of fig 1 and 1 correspond to figure 1).
The fitness is caluclated using testfitness.m, and the actual "evolution" is carried out through testevo.m and testevo2.m. The first file goes through each node, add or subtract a small quantity to the connection strength, compare the fitness with the previous one and adopts the changes if the fitness increase, this is repeated for that node a fixed number of time or until we have no more improvement, then the same is repeated with opposite sign, and then we do the same for the next node, and we keep going till we finish the nodes.
The second algorithm produce a parameter search for the optimal fitness. The way we do this is layer by layer, starting with the lower one and building up the perceptron adding new layer to the previously optimized one. For example if we want two layers like in fig.4 we will start optimizing the 2 connection in a perceptron like in fig 2 (for instance finding best combination of connections with values between -2 and 2) then would add a new layer and will optimize the 4 new connection keeping the previous two fixed.

Each of this methods are not providing the results we hoped for, in fact using a single method we were able to get with both a fitness around 0.2, while combining them it was possible to increase this value to 0.25 (testevo2 then testevo) or 0.27 (testevo2 1 layer then testevo then add new layer with testevo2 and then testevo again).
However, this indicates an adherence varying from 1/5th to 1/4th of the wanted pictures, and plotting the activation patterns results in pictures similar to the top right in fig.5 (more than significantly different from fig.1).

Conclusions

We were not able to achieve an activation pattern as in Fig.1 through our procedures. However, the algorithm used are able to improve the fitness of the starting network compared to the desired one. Reasons for our failure and possible improvement points include

- network topology: here we consider only two nodes per layer, but more may be required to reach good results
- fitness function: the fitness function we used is not the only possible one and probably not the best possible one
- "evolution" algorithms: the algorithms we are using while seem to be working (at least partially) are not optimal solution and have big space for improvement.

However, we were able to establish an algorithm to determine the activation pattern of a perceptron and evaluate it compared to a preferential pattern.

References:

[1] notes in class and slides 11/11/2014
[2] discussion with Christopher Eshcb regarding evolution algorithms