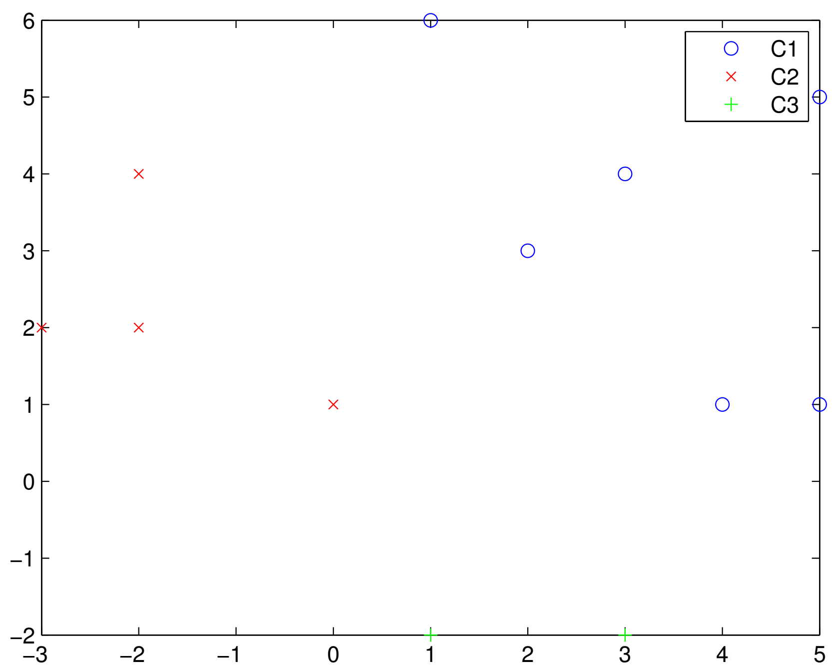
CS 540, HW4 Solution, Spring 2010

Problem 1

Can you argue (without explicitly solving for the weights) whether or not your perceptron can learn a hyperplane to separate the examples?

Yes, it is possible. Just look at the plot.



Now, what happens if you add (0, 7) to C2? Can the perceptron still learn a hyperplane to separate the examples? What happens if you add (7, 7) to C2?

Yes. No. Just add them to the plot.

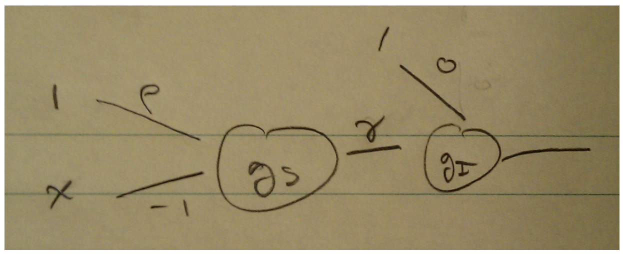
What kind of modifications/extensions to your implementation will you consider to address such situations?

Add slack variables or use nonlinear transformation.

Problem 2

(a) For user-provided/given integers γ and ρ (i.e., these values are fixed beforehand), can you use the activation functions above to construct a neural network to give the following function $out(\cdot)$: $out(x) = \gamma$ if $x < \rho$ and 0 otherwise.

I.e. we want $out(x) = \gamma g_S(\rho - x) = g_I(0 \cdot 1 + \gamma \cdot g_S(\rho \cdot 1 + (-1) \cdot x))$. Just read off the network from the preceeding expression and get:



(b) Assume now that there are two input units: x and y (each is a 1 bit binary value), construct a neural network that performs the operation of exclusive-OR. In other words, the neural network should output 1 if $x \neq y$ and 0 otherwise.

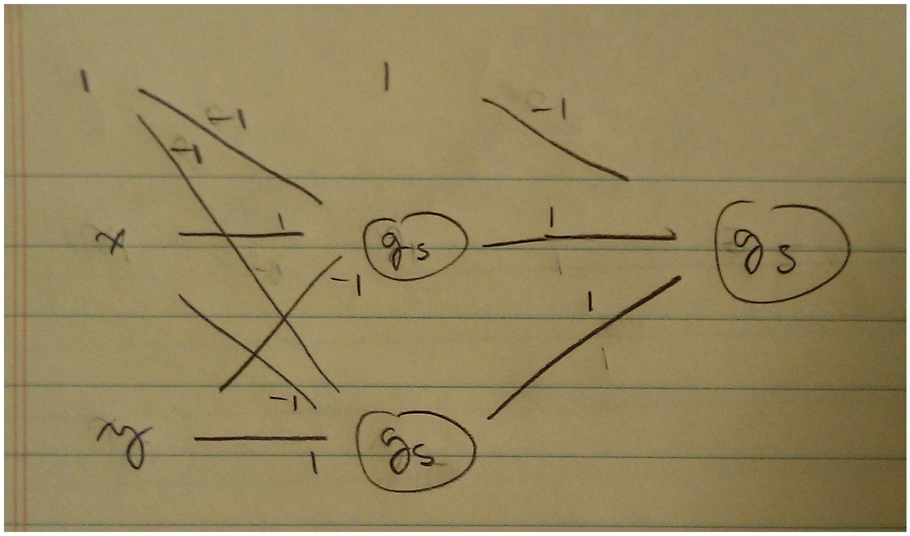
This map works:

$$f(x, y) = g_S(-1 + g_S(-1 - x + y) + g_S(-1 + x - y))$$

Check:

$$\begin{aligned}
f(0,0) &= g_S(-1 + g_S(-1 - 0 + 0) + g_S(-1 + 0 - 0)) \\
&= g_S(-1 + 0 + 0) \\
&= 0 \\
f(1,0) &= g_S(-1 + g_S(-1 - 1 + 0) + g_S(-1 + 1 - 0)) \\
&= g_S(-1 + 0 + 1) \\
&= 1 \\
f(0,1) &= g_S(-1 + g_S(-1 - 0 + 1) + g_S(-1 + 0 - 1)) \\
&= g_S(-1 + 1 + 0) \\
&= 1 \\
f(1,1) &= g_S(-1 + g_S(-1 - 1 + 1) + g_S(-1 + 1 - 1)) \\
&= g_S(-1 + 0 + 0) \\
&= 0
\end{aligned}$$

Get structure:

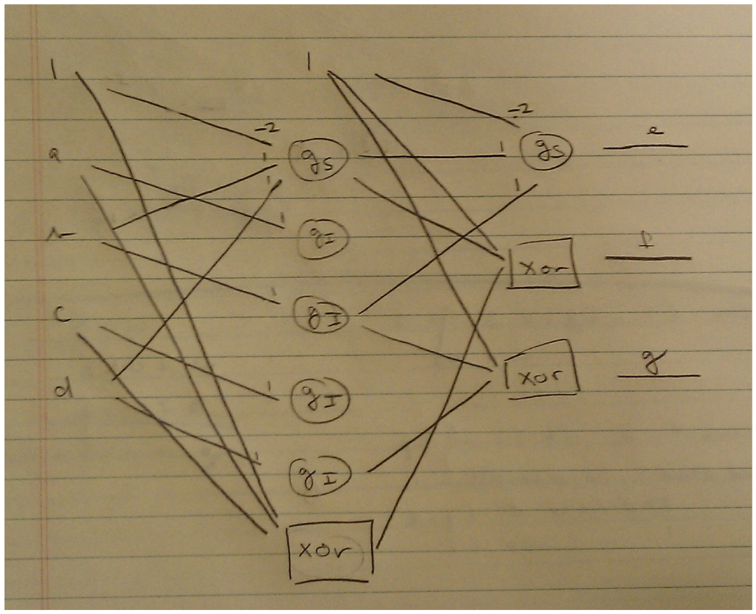


(c) Now, we have two binary numbers: ab and cd. Also, efg is the binary number (of 3 bits) which is the addition of ab and cd. Here a, c, e are the higher-order bits and b, d, g are the lower-order bit. That is, if ab is 01 and cd is 11, then efg should be 100. Similarly, if ab is 01, and cd is 10, then efg should be 011. Can you design a neural network to implement this add operation?

Want:

$$\begin{aligned}
g(a,b,c,d) &= \operatorname{mod}(b+d,2) \\
&= \operatorname{xor}(b,d) \\
x(a,b,c,d) &= \begin{cases} 1 & \text{if } b+d=2 \\ 0 & \text{o.w.} \end{cases} \\
&= g_S(-2+b+d) \\
f(a,b,c,d) &= \operatorname{mod}(x+a+c,2) \\
&= \operatorname{xor}(x,\operatorname{xor}(a,c)) \\
e(a,b,c,d) &= \begin{cases} 1 & \text{if } x+a+c \geq 2 \\ 0 & \text{o.w.} \end{cases} \\
&= g_S(-2+x+a+c)
\end{aligned}$$

Read off the following network:



Here XOR denotes a module of the form in part (b).