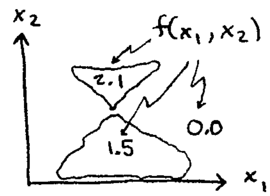
5 Apr 1990
Neil E Cotter

Approximation Theory - Universal Approximation - Perceptron ~~m~~ Computing arbitrary functions

ref: Lippmann article handed out in class

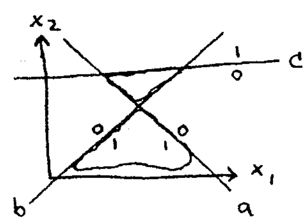
Suppose we wish to create a perceptron network to compute the following function (plotted as a topographic map):



Function $f(x_1, x_2)$ has two triangular mesas of height 1.5 and 2.1.

We use a three-layer network to approximate $f()$:

Layer 1) Use perceptrons (3 in this case) to outline polygons that approximate the boundaries of the mesas.



For greater accuracy, use more lines.

Layer 2) Use perceptrons (2 in this case) ~~to~~ as logic gates (usually AND gates) to determine if input (x_1, x_2) to net lies in a given polygon, (i.e. is on the correct side of all lines forming polygon). Thus, we have 1 logic gate for each polygon. Gate output is 1 if and only if point (x_1, x_2) is inside that gate's polygon. Only one gate output will be 1 at a given time.

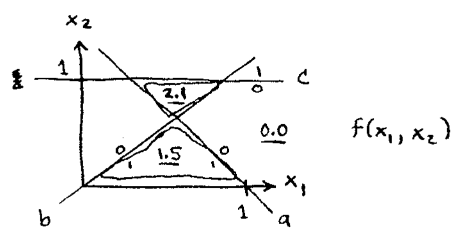
Layer 3) Use a perceptron from which the step function has been removed. This neuron is called a summing node or linear neuron. Its inputs are logic gate outputs which are 0 or 1. Set the synapse (receiving a given logic gate output) to the value or height of $f()$ in the corresponding polygon. (1.5 or 2.1 here)

5 Apr 1990

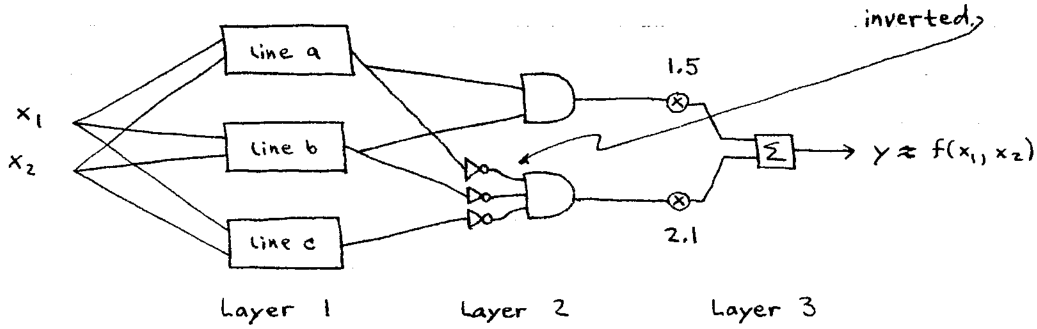
Neil E Cotter

Approximation Theory – Universal Approximation – Perceptron ~~An~~ Computing arbitrary functions (cont.)

ex:



Note that 2.1 region is on "0" side of all three lines. This is why AND gate inputs inverted.



Detailed view: