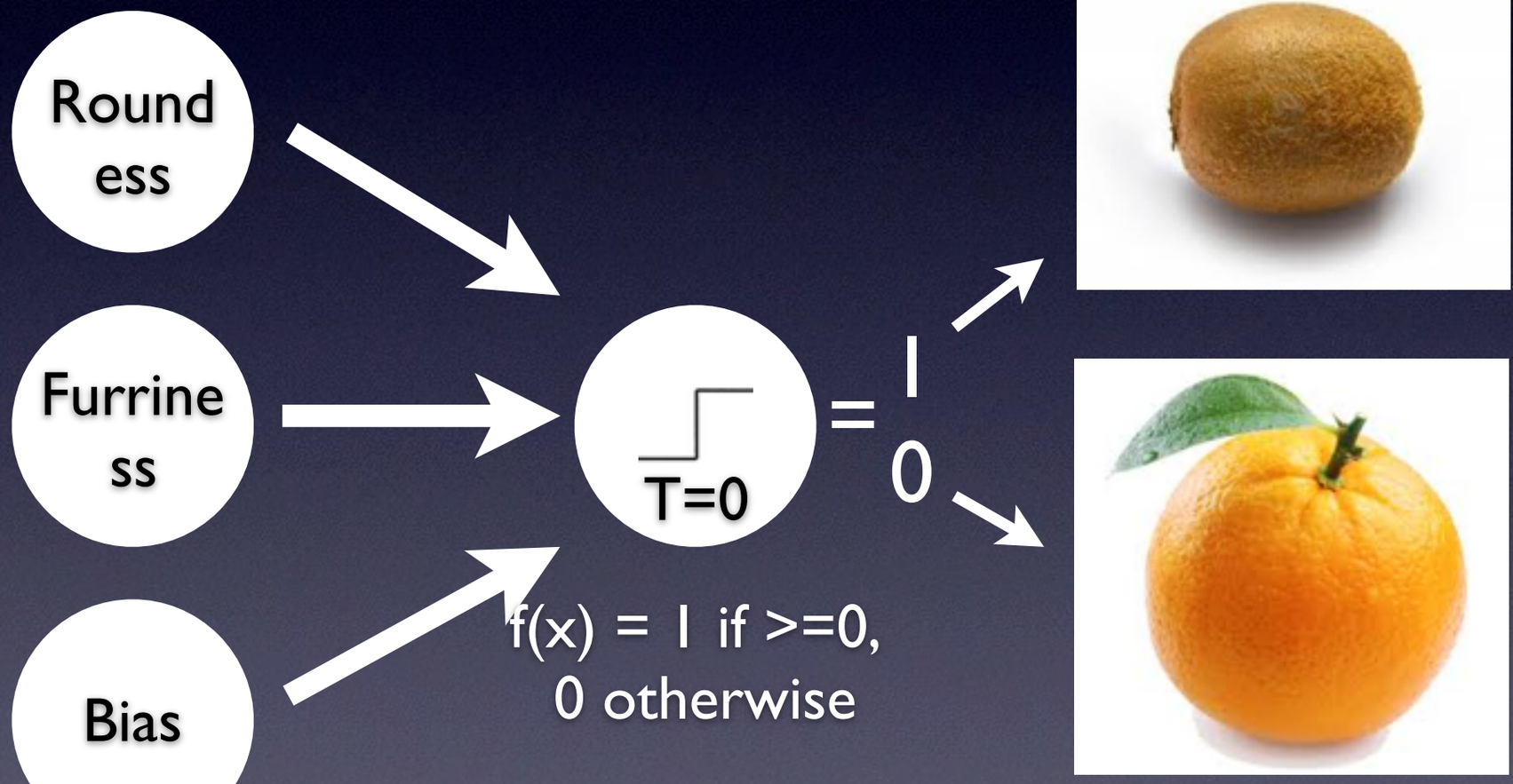
Evolutionary Robotics

UNIVERSITY OF WYOMING

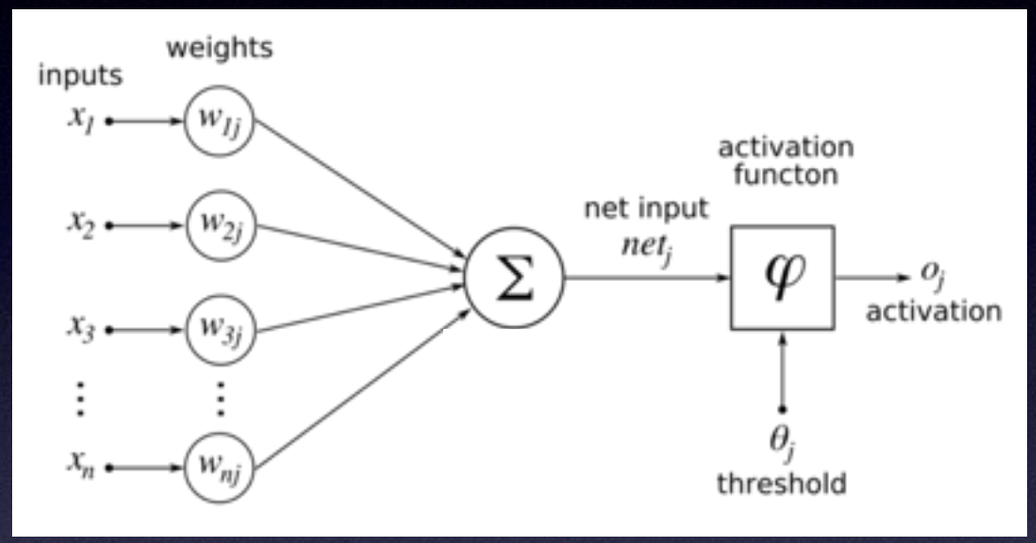
Neural Networks

- Simple example: kiwi classifier
  - binary inputs
  - weights = ??
  - bias = 1



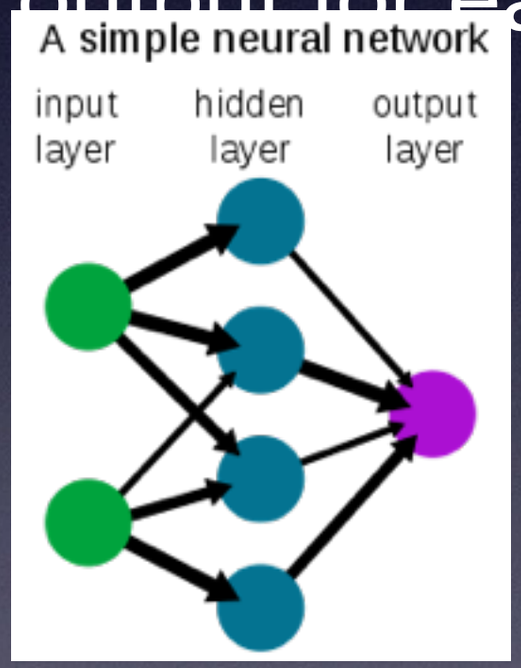
Non-Linear Activation Functions

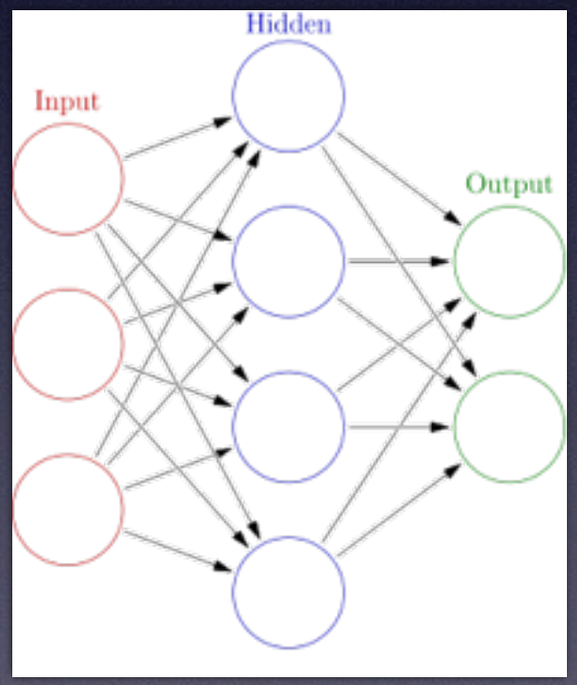
- Combining many of them builds an ANN
  - that can represent non-linear functions
    - because activation functions are non-linear



Outputs

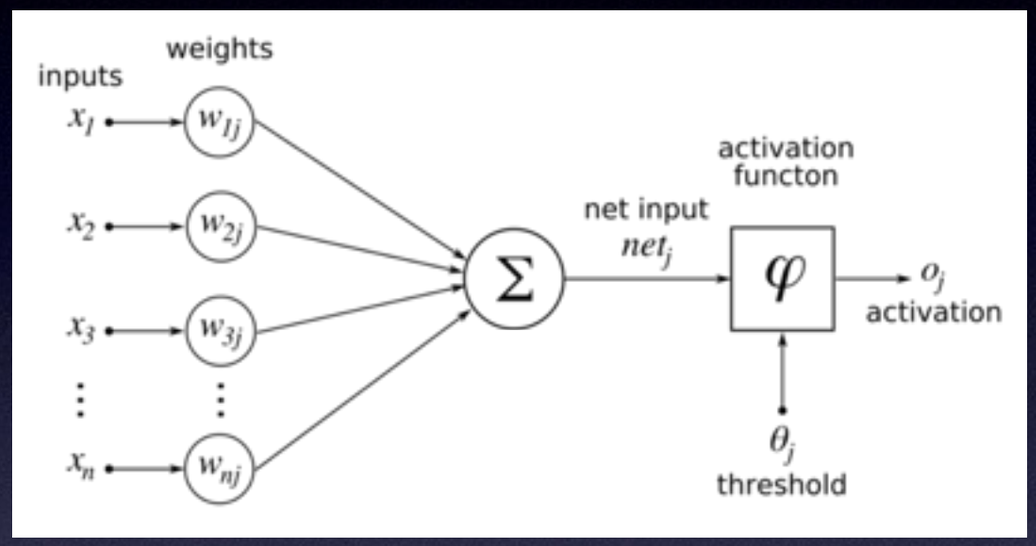
- Can have single or multiple outputs
- e.g. one output for each class





Perceptron

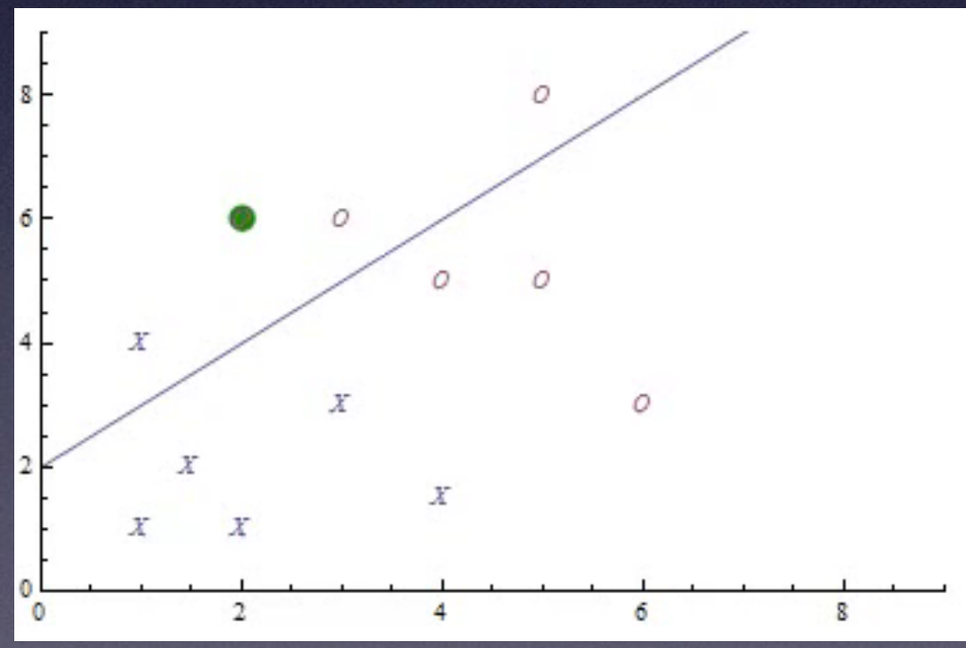
- Perceptron: Single unit
  - Default activation function: step
  - Sigmoid: “sigmoid perceptron”
- Perceptron Network:
  - single-layer network



McCullough & Pitts (1943)

Perceptron Training Rule

- Training:
  - step activation function: perceptron learning rule
  - logistic activation function: gradient descent
  - both work for linearly separable data
    - we already new that…just a new metaphor now



Perceptron Training Rule vs. Decision Tree

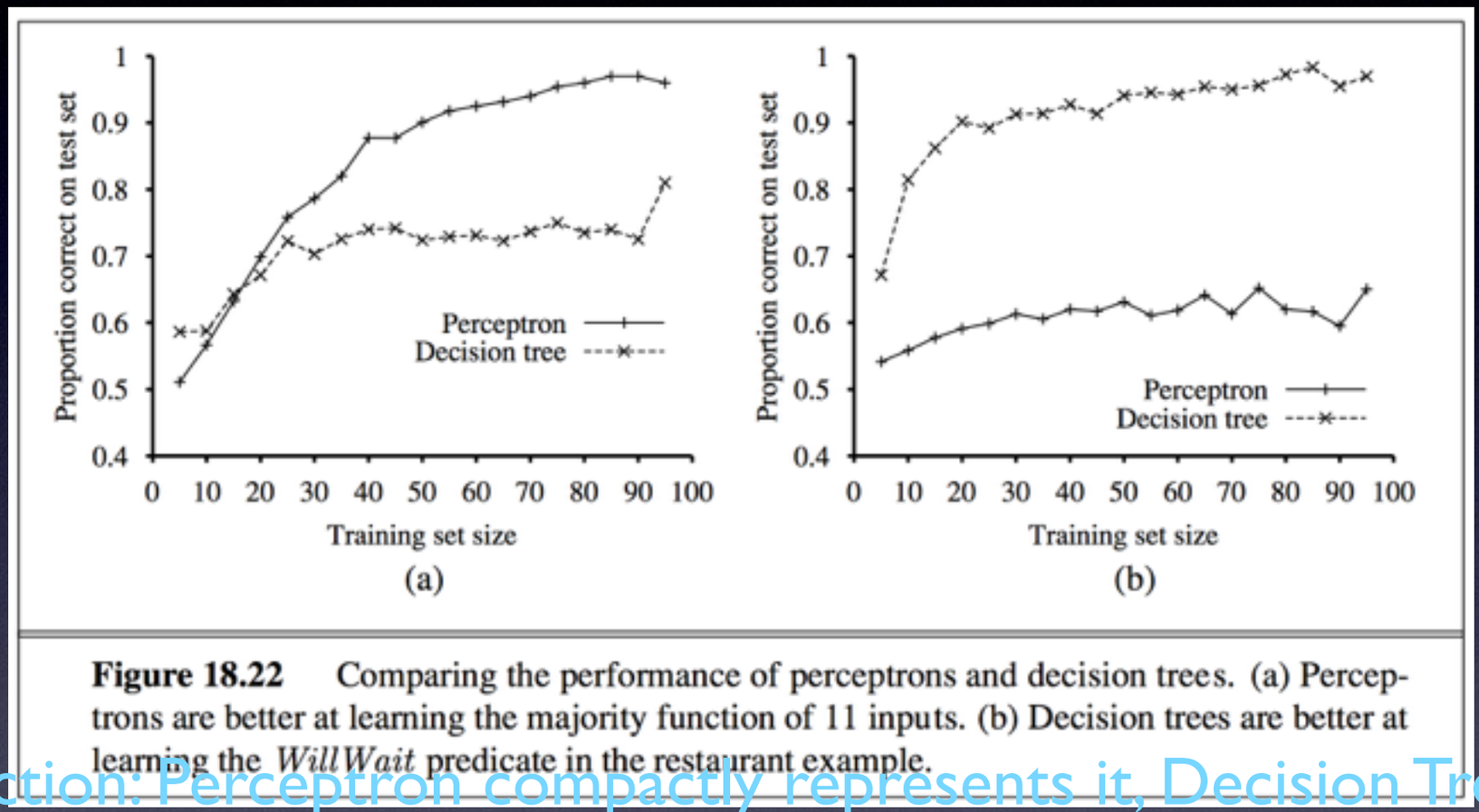


Figure 18.22 Comparing the performance of perceptrons and decision trees. (a) Perceptrons are better at learning the majority function of 11 inputs. (b) Decision trees are better at learning the *WillWait* predicate in the restaurant example.

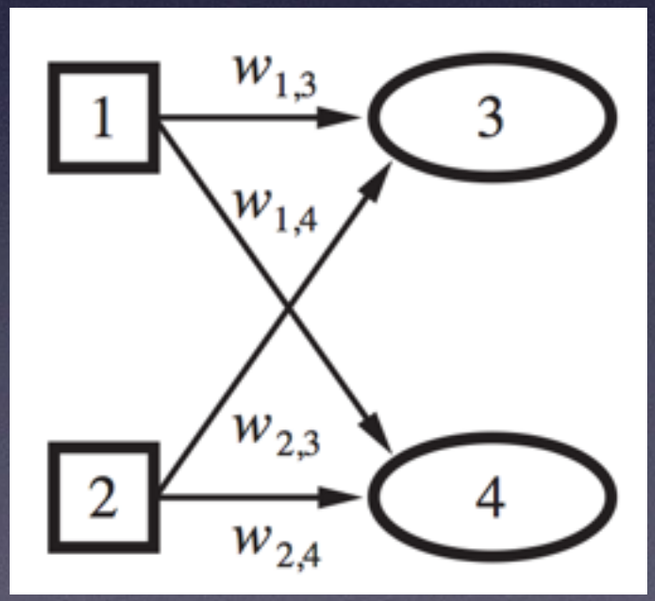
Majority Function: Perceptron compactly represents it, Decision Trees requires exponentially many nodes

Restaurant Wait Function: Not linearly separable, so DT works well, Perceptron does not

Adder Problem

- Binary Adder Function

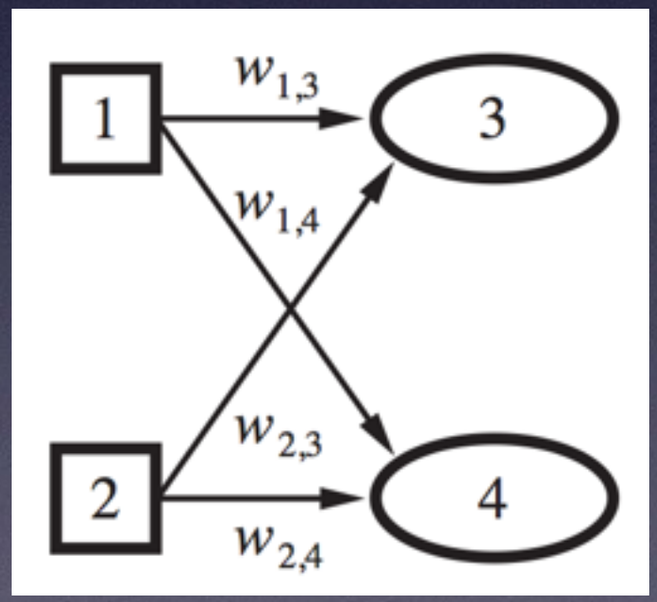
x_1	x_2	y_3 (carry)	y_4 (sum)
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



Adder Problem

- Binary Adder Function
- Note that the two outputs mean two separate learning processes
  - weights only affect one output

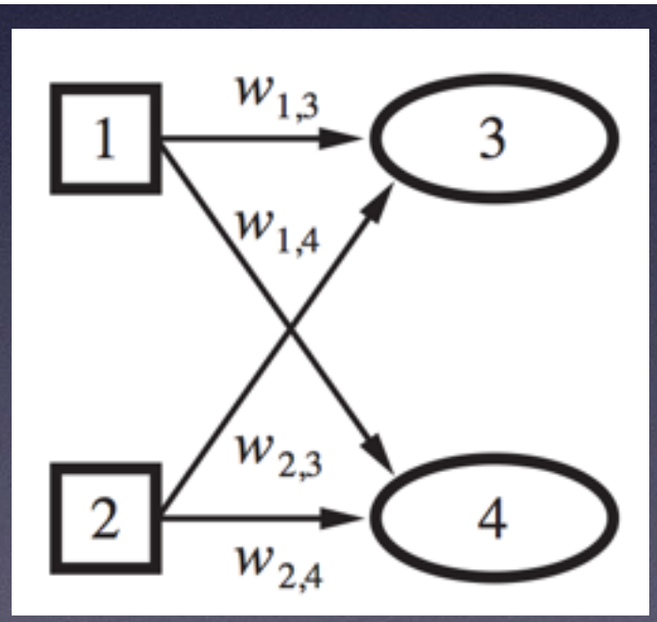
x_1	x_2	y_3 (carry)	y_4 (sum)
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



Adder Problem

- Run learning
  - Neuron 3 is perfect
  - Neuron 4 fails
  - Why?

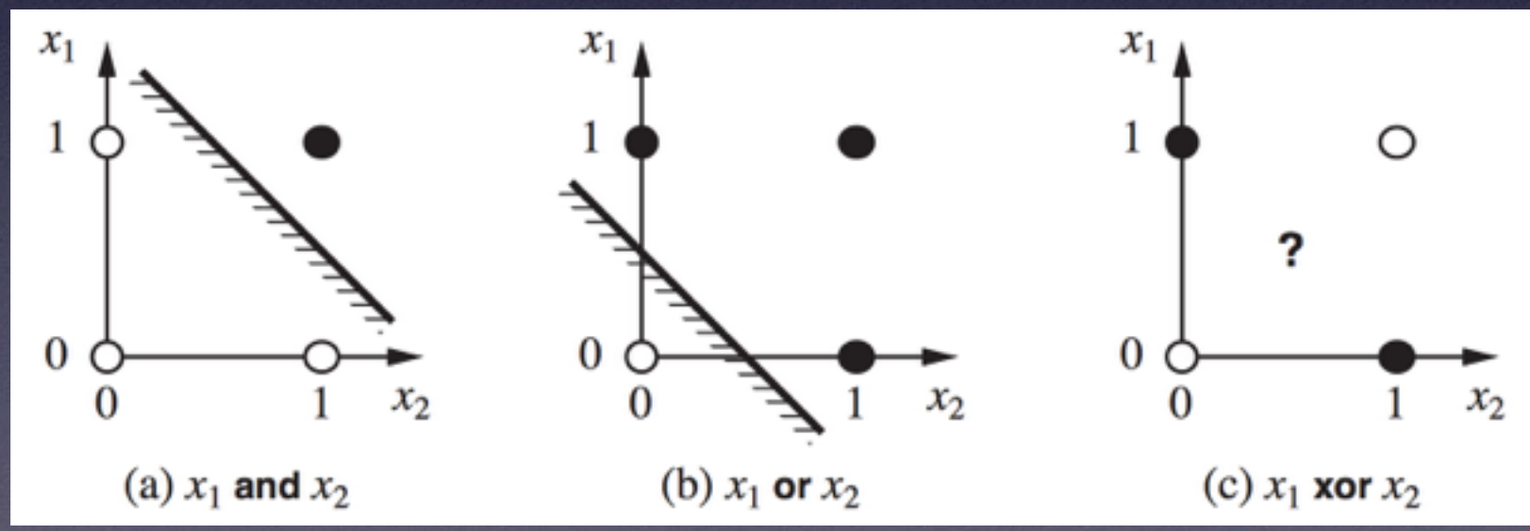
x_1	x_2	y_3 (carry)	y_4 (sum)
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



Adder Problem

- Run learning
  - Neuron 3 is perfect
  - Neuron 4 fails
    - It's not linearly separable
    - It's an XOR problem
    - classic #fail for perceptrons
      - Minsky & Papert 1969
      - Though McCullough & Pitts (1943) knew it

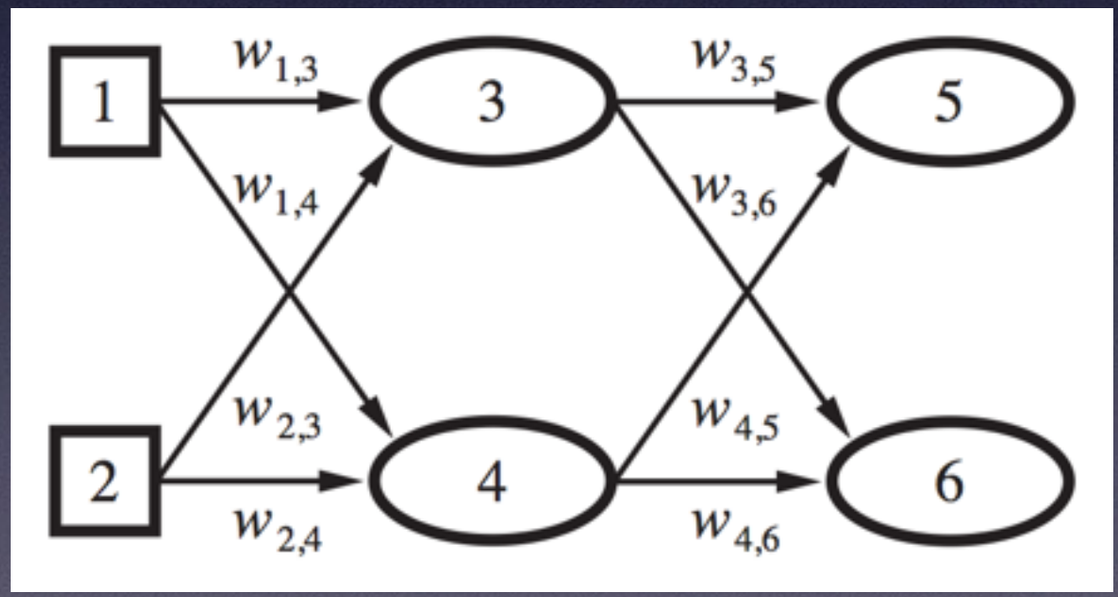
x_1	x_2	y_3 (carry)	y_4 (sum)
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



XOR Problem

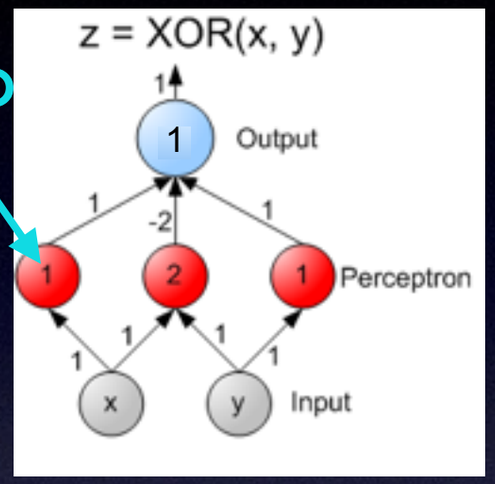
- Solution?
- Multiple layers
  - aka “multi-layer percept

x_1	x_2	y_3 (carry)	y_4 (sum)
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

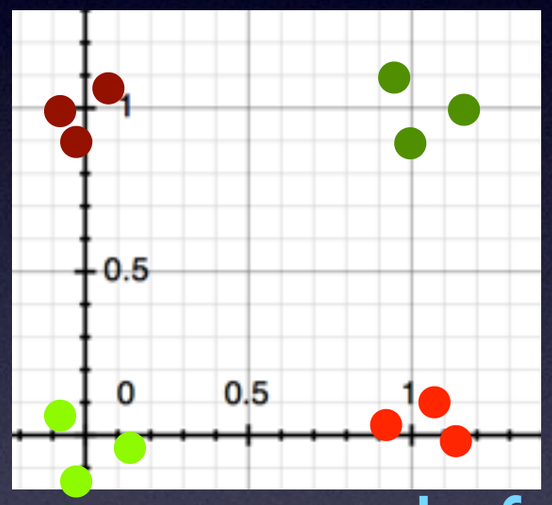


Neural Networks

utputs 1 if >= number in no



2D



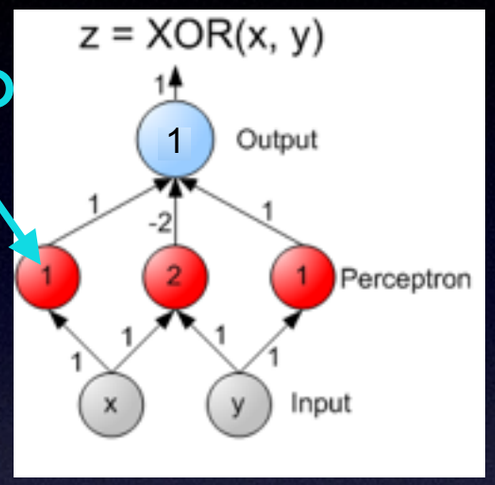
x	y	Left	Center	Right	Output
1	1				
1	0				
0	1				
0	0				

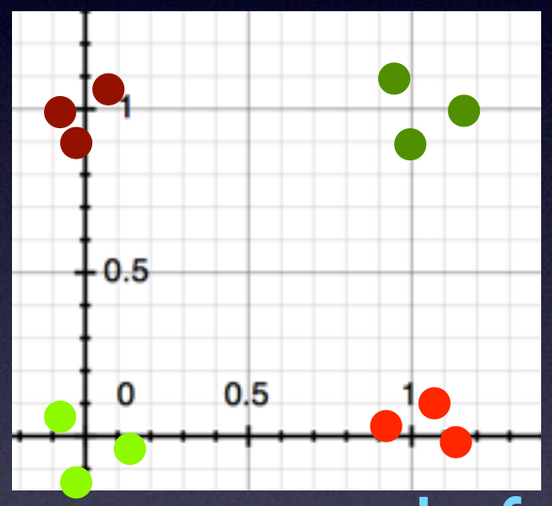
Outputs of each node

Fill in! (Groups)

Neural Networks

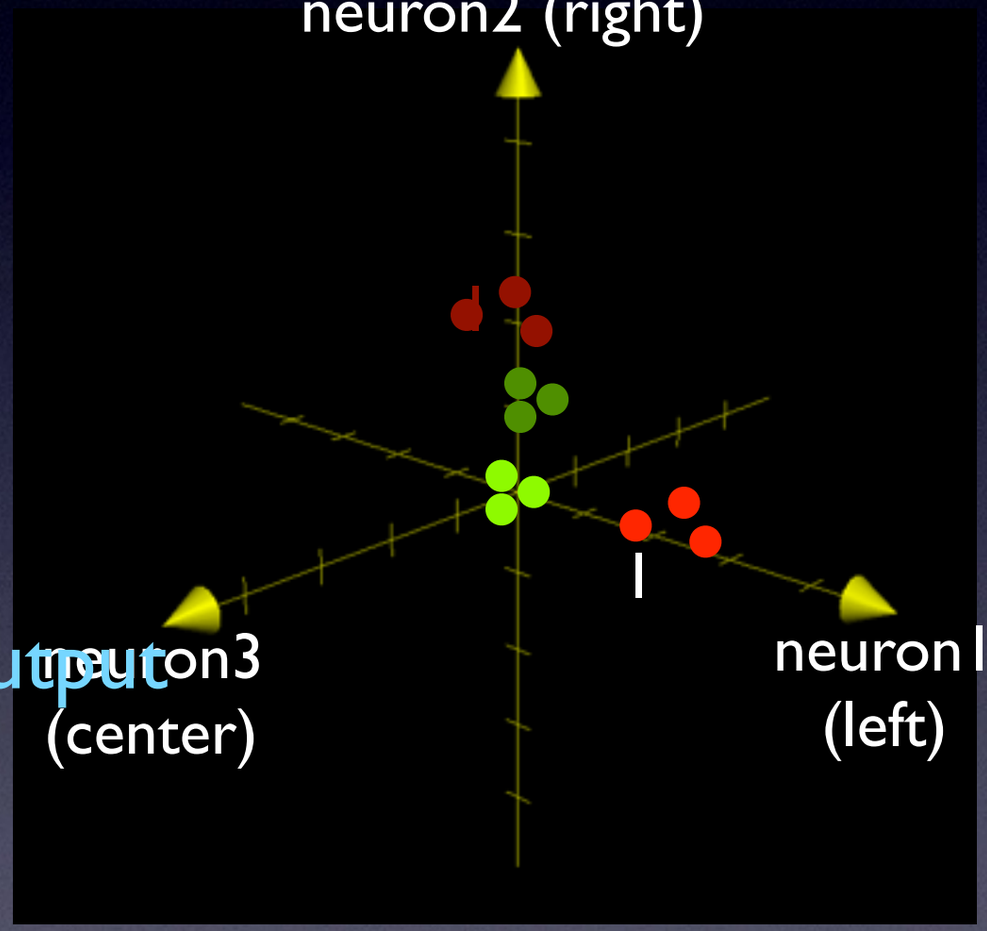
utputs 1 if >= number in no





2D

Non-Linear Mapping



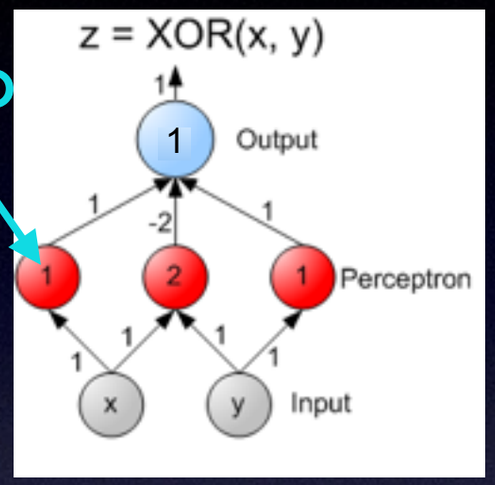
3D

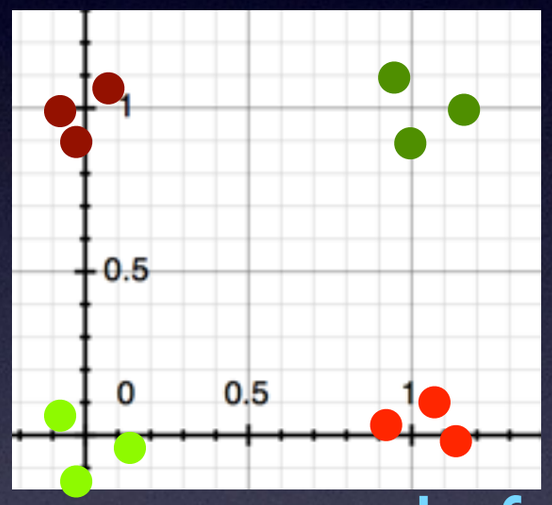
x	y	Left	Center	Right	Output
1	1				
1	0				
0	1				
0	0				

Outputs of each node

Neural Networks

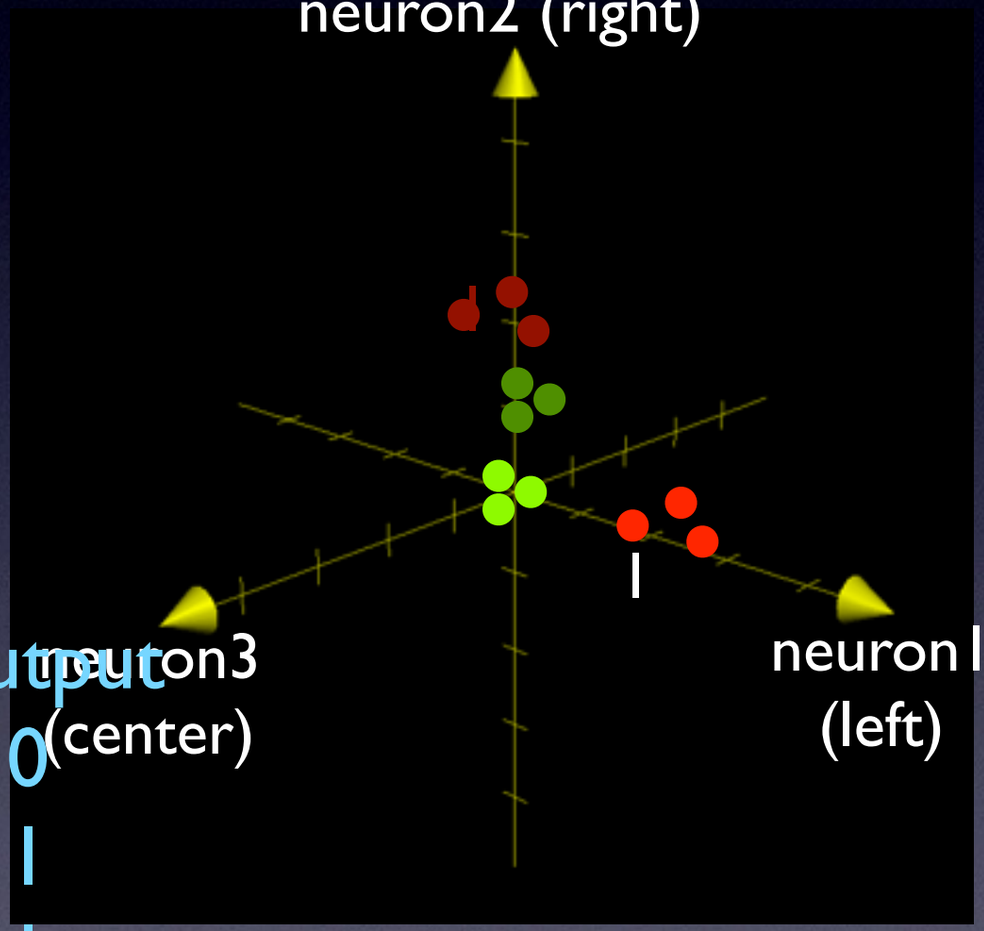
utputs 1 if >= number in no





2D

Non-Linear Mapping

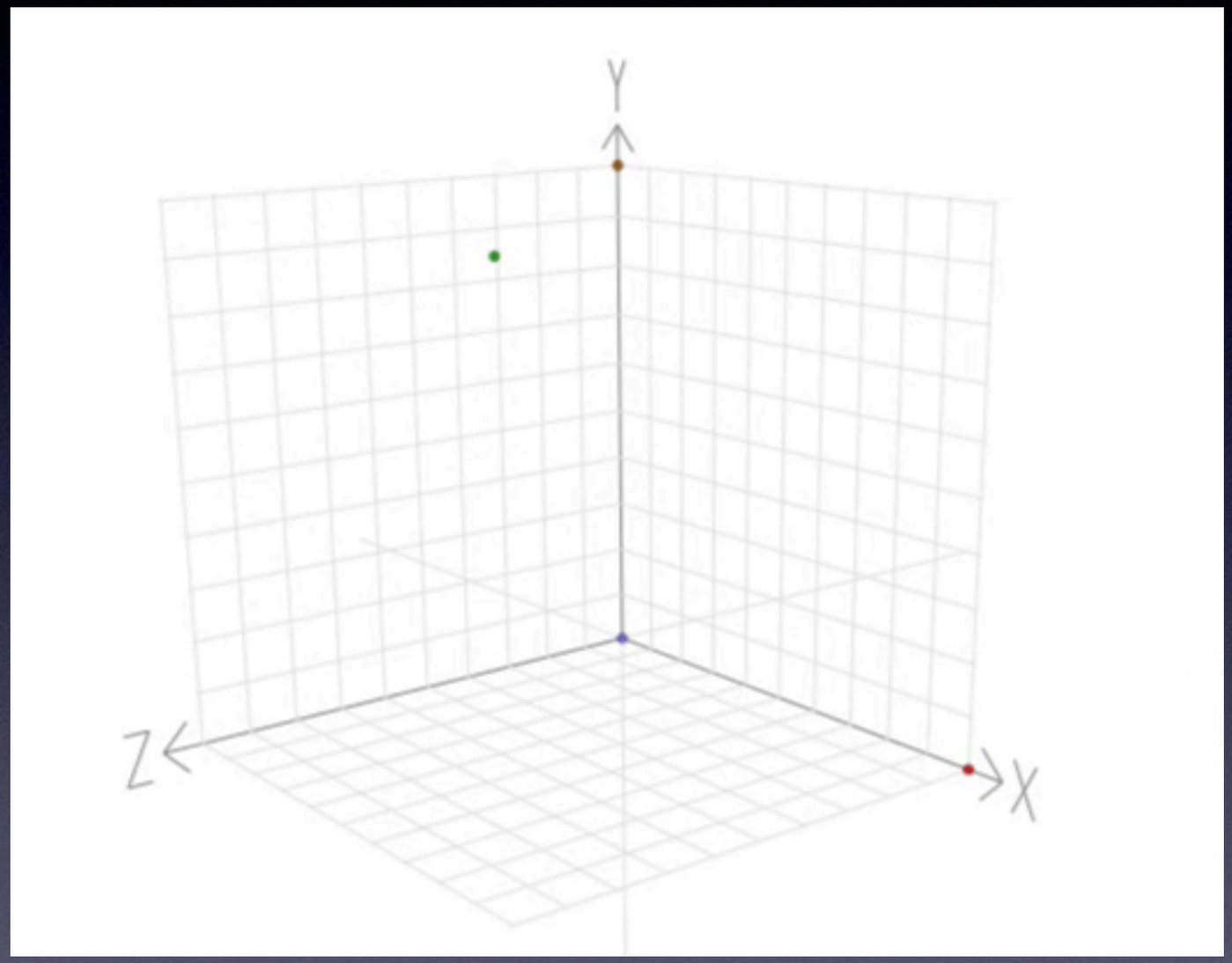


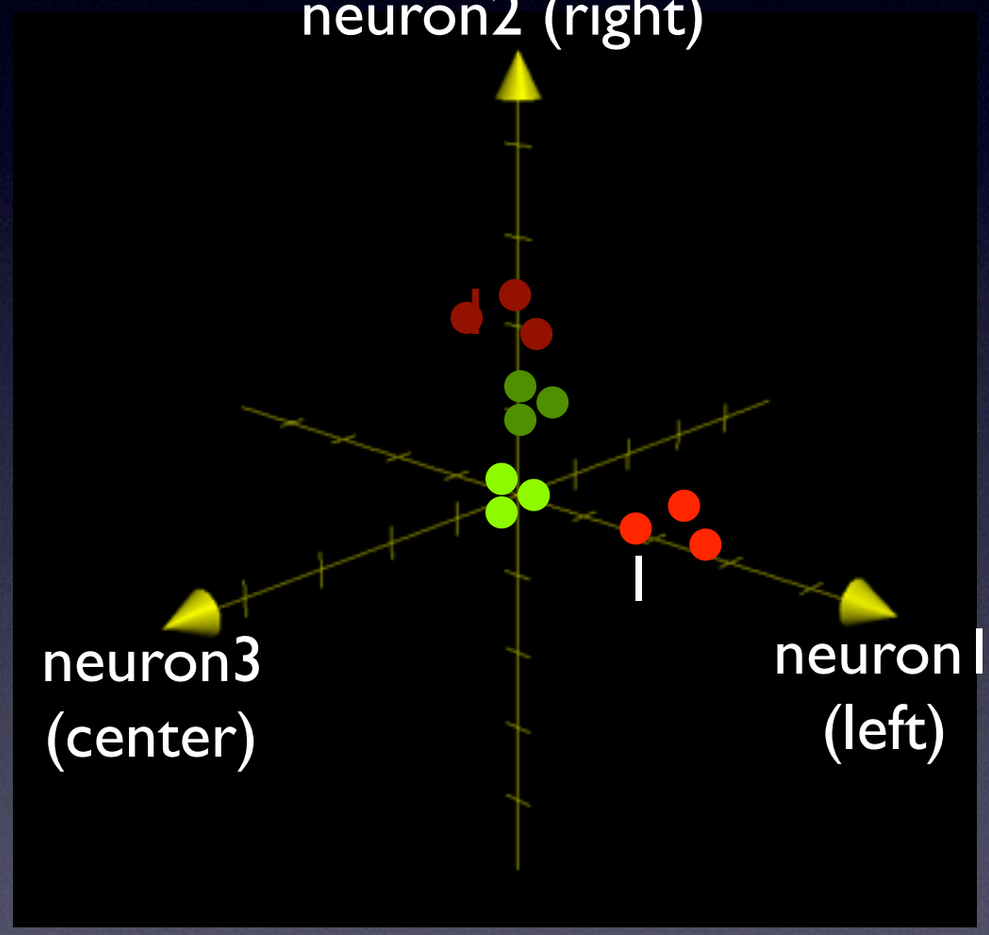
3D

x	y	Left	Center	Right	Output
1	1	1	1	1	0
1	0	1	0	0	1
0	1	0	0	1	1
0	0	0	0	0	0

Outputs of each node

Neural Networks





3D