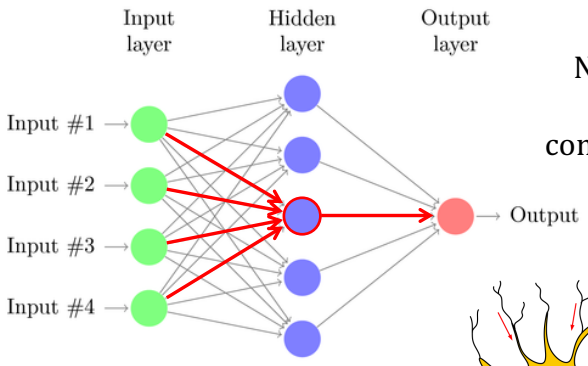
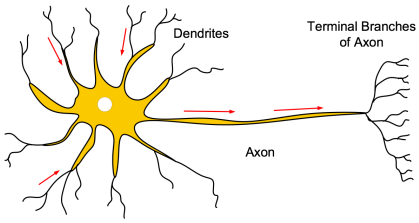
What is an artificial neural network?

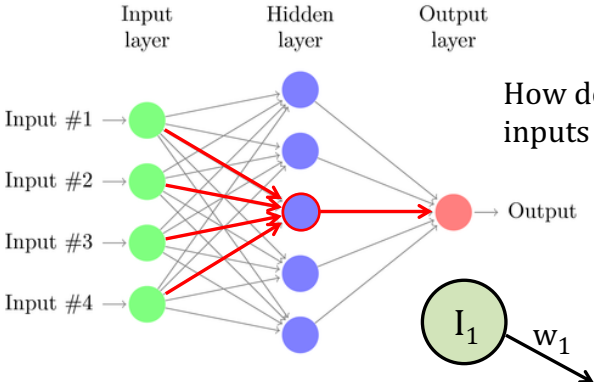


Network of simple *neuron-like* computing elements …

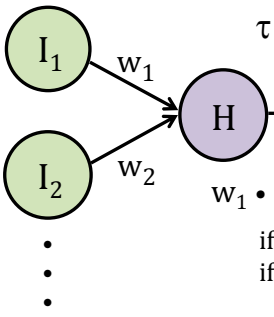


… that can *learn* to associate inputs with desired outputs

Computing in an artificial neural network



How does each unit integrate inputs to produce an output?



w_1, w_2 : weights

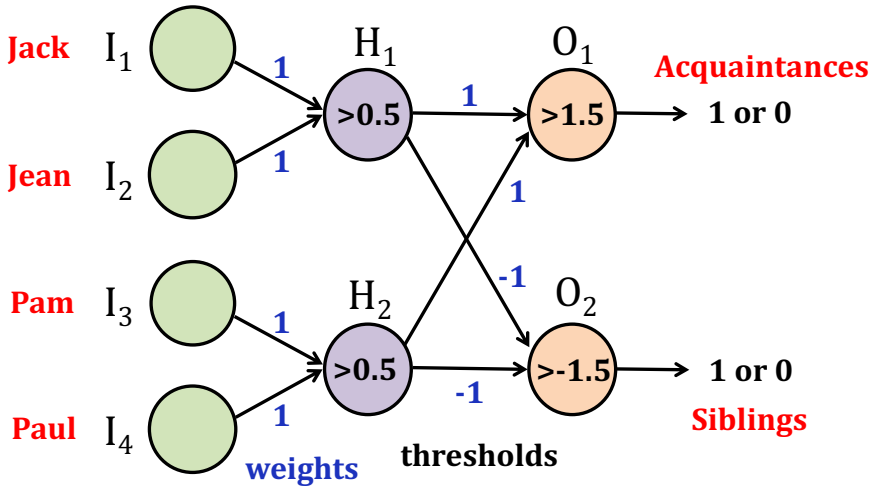
τ : threshold

$w_1 \bullet I_1 + w_2 \bullet I_2 > \tau$

if true: H = 1
if false: H = 0

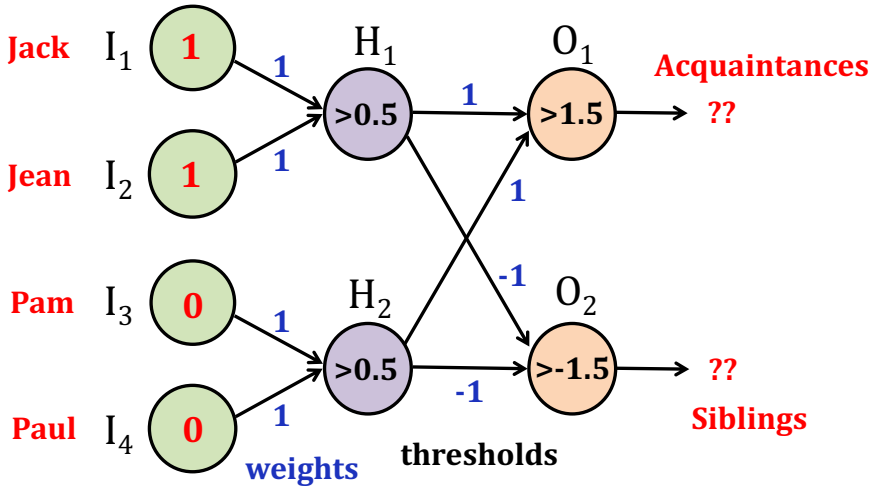
Can such a simple network perform a useful function?

A (very!) simple neural network



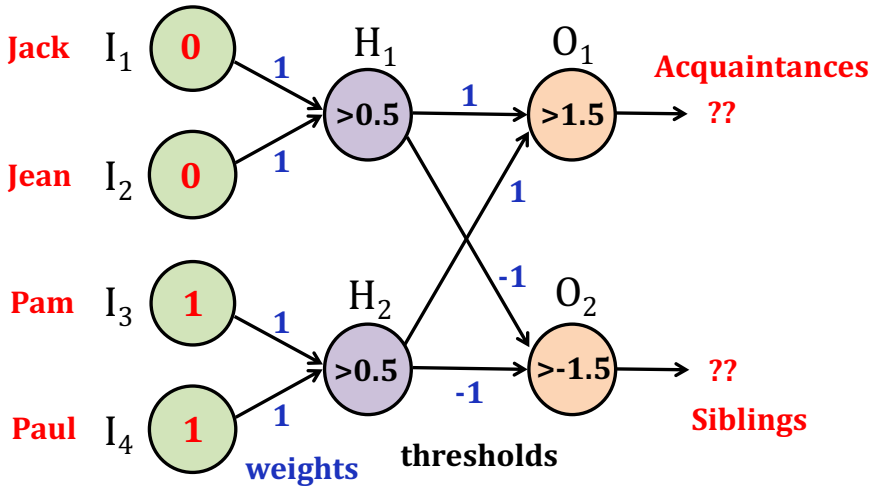
network inputs: 1 or 0, valid input combinations have exactly two 1s
network outputs: 1 or 0

A (very!) simple neural network



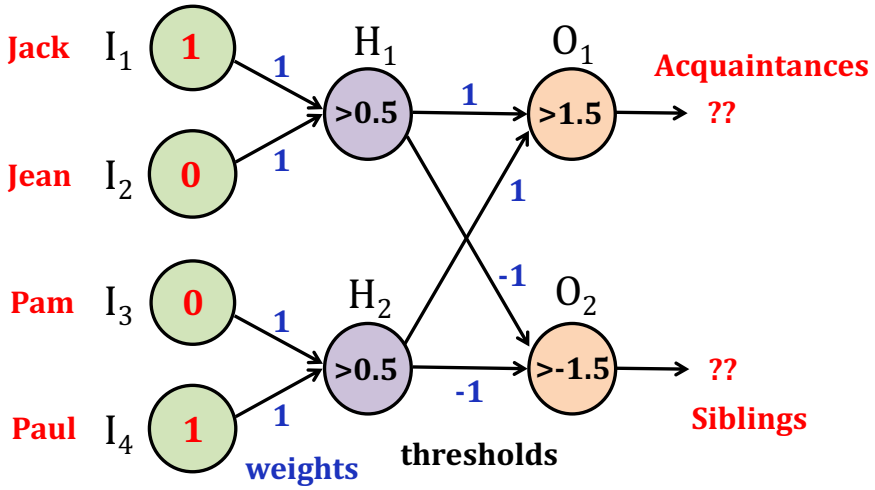
network inputs: 1 or 0, valid input combinations have exactly two 1's
network outputs: 1 or 0

A (very!) simple neural network



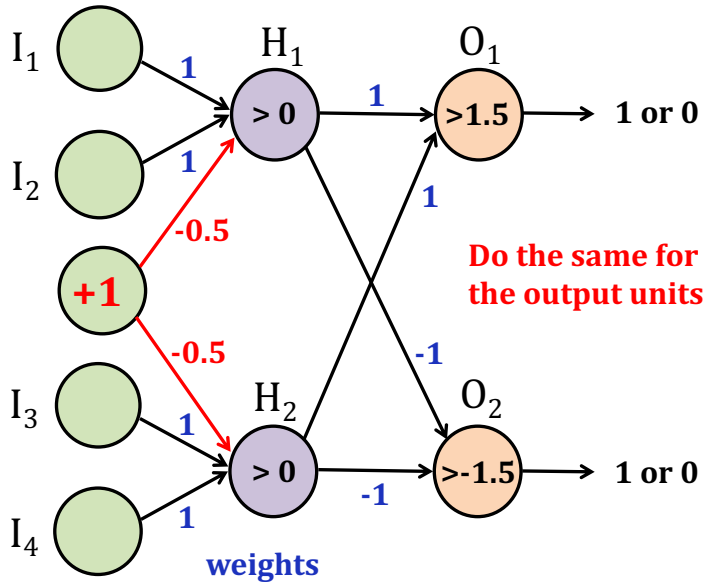
network inputs: 1 or 0, valid input combinations have exactly two 1's
network outputs: 1 or 0

A (very!) simple neural network

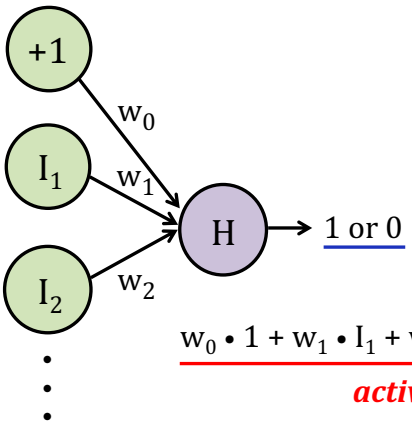


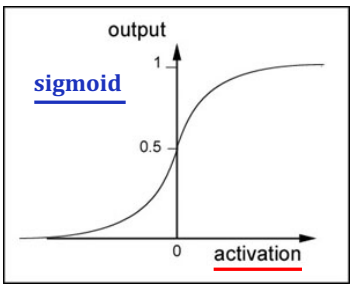
network inputs: 1 or 0, valid input combinations have exactly two 1's
network outputs: 1 or 0

Add "bias" units to simplify thresholds



Computing in a "typical" neural network





$$w_0 \bullet 1 + w_1 \bullet I_1 + w_2 \bullet I_2 + \ldots + w_n \bullet I_n > 0$$

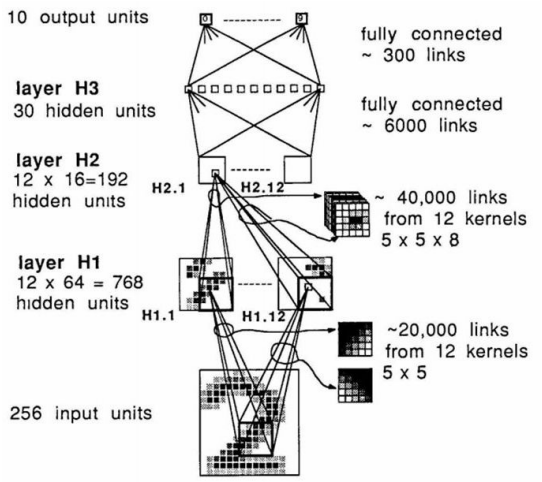
activation

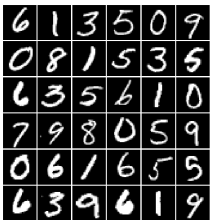
How does each unit integrate its inputs to produce an output?

sum of weighted inputs → sigmoid function → output *between 0 and 1*

Learning to recognize handwritten zip codes

LeCun et al. (1989)



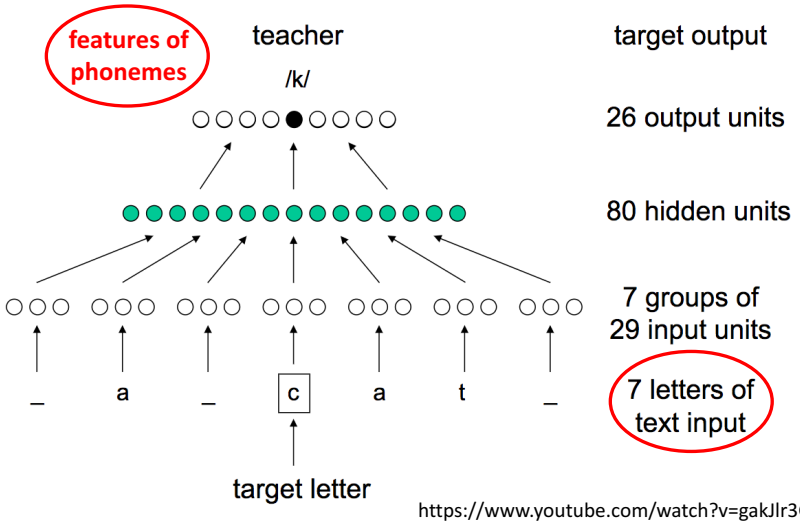


System could recognize image samples provided by the US postal service, with high accuracy

(MNIST database)

NETtalk learned phonemes from text

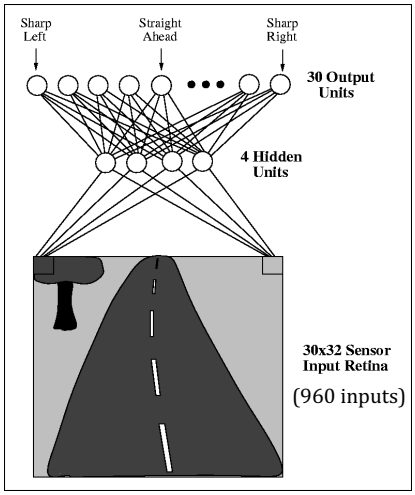
Sejnowski & Rosenberg (1989)



https://www.youtube.com/watch?v=gakJlr3GecE

ALVINN learned to control steering actions

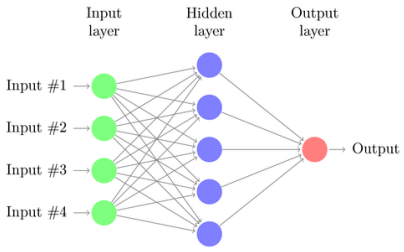
Pomerleau (1991)



- ALVINN learned to steer by *observing a human driver*
- Multiple networks for different roads (e.g. dirt road, two-lane road, highway (up to 70mph!))

Learning to recognize input patterns

feedforward processing



back-propagation algorithm to learn network weights

network weights can be ***learned*** from training examples (map inputs to correct outputs)

back-propagation:
iterative algorithm progressively reduces error between computed and desired output until performance is satisfactory

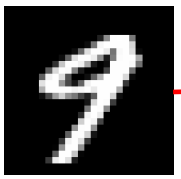
on each iteration:

- compute output of current network and assess performance
- compute weight adjustments from hidden to output layer that reduce output errors
- compute weight adjustments from input to hidden units that improve hidden layer
- change network weights, incorporating a rate parameter

Learning handwritten digits

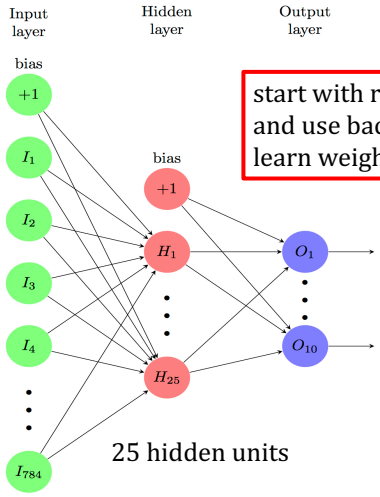
MNIST database:
3000 28x28 images
of handwritten digits

28



28

sample image



one input unit
for each pixel

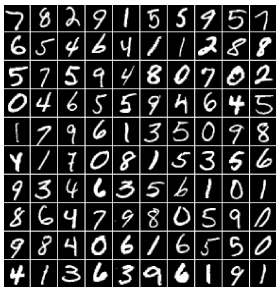
25 hidden units

start with random initial weights and use back-propagation to learn weights to recognize digits

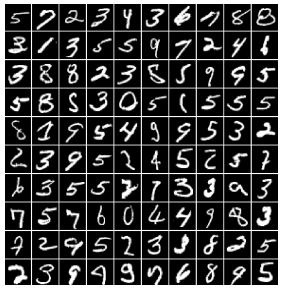
one output unit for each digit

select output unit with maximum response
e.g. 9

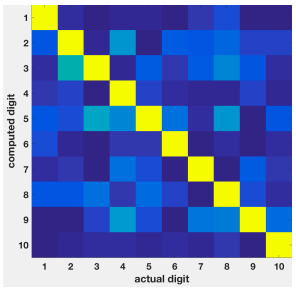
Results: Learning handwritten digits



correct examples



wrong examples



confusion matrix

overall classification accuracy: 93.1%