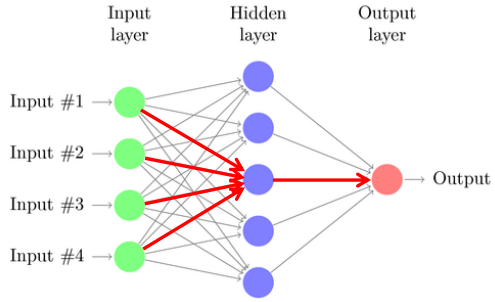
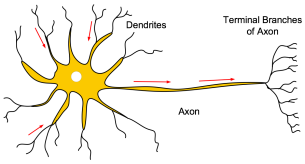
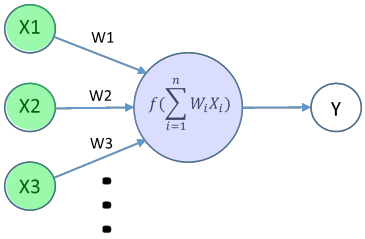
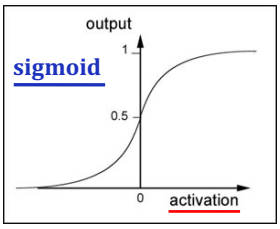
Artificial neural networks



network of simple *neuron-like* computing elements







$w_1 * I_1 + w_2 * I_2 + \ldots + w_n * I_n + b > 0$

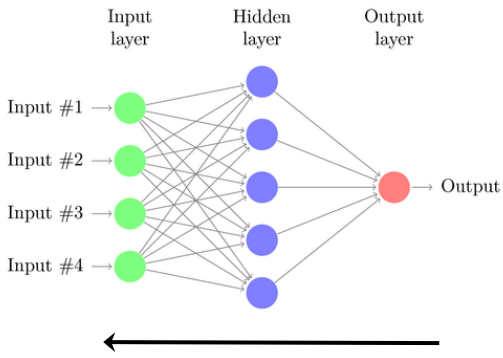
activation ***bias***

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

1

Learning to recognize input patterns

feedforward processing



backpropagation algorithm to learn network weights

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

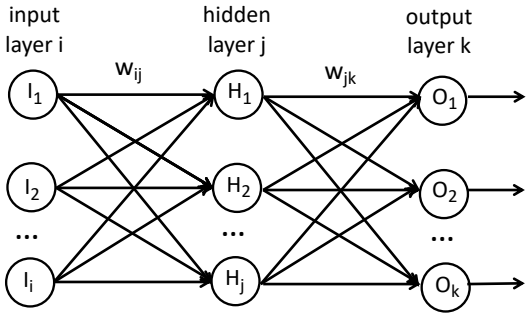
backpropagation:
iterative algorithm that progressively reduces error between computed and desired output until performance is satisfactory

on each iteration:

- compute output of current network and assess performance
- compute weight adjustments from hidden to output layer that can reduce output errors
- compute weight adjustments from input to hidden units that can enhance hidden layer
- change network weights, using *rate parameter*

2

Backpropagation algorithm



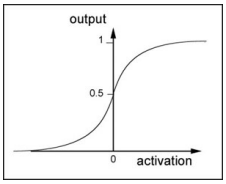
desired outputs

d_1

d_2

...

d_k



$$f(x) = \frac{1}{1+e^{-x}}$$

$f'(x) = f(x) * (1 - f(x))$

For each training sample, determine what weight changes would improve performance of the network:

Δw_{jk} = <rate parameter> * <current output of unit H_j> * <slope of O_k> * <benefit of adjusting O_k>

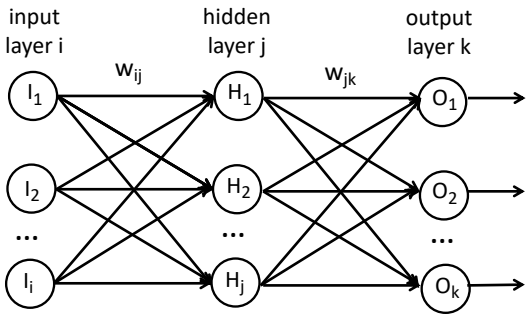
$\Delta w_{jk} = r * H_j * (O_k(1 - O_k)) * (O_k - d_k)$

$\Delta w_{ij} = r * I_i * (H_j(1 - H_j)) * b_j$ < b_j is benefit of adjusting H_j>

$b_j = \sum_k w_{jk} * (O_k(1 - O_k)) * (O_k - d_k)$

3

Backpropagation algorithm



desired outputs

d_1

d_2

...

d_k

note about rate parameter:

- if too small, network may take a very long time to converge
- if too large, network behavior oscillates around best solution

(1) for each training sample, determine all the weight changes Δw_{jk} and Δw_{jk} that would improve performance of the network

(2) add up the weight changes for all training examples and change all the weights at once

(3) repeat steps (1) and (2) until overall performance is satisfactory e.g. *small cost* = $\sum_k (O_k - d_k)^2$

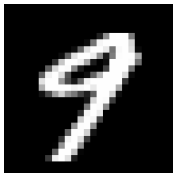
4

Example: learning handwritten digits

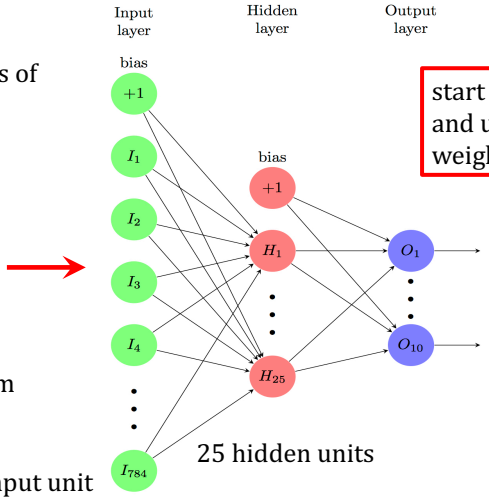
MNIST database:
3000 28x28 images of handwritten digits

28

28



sample image from MNIST database



25 hidden units

one input unit for each pixel

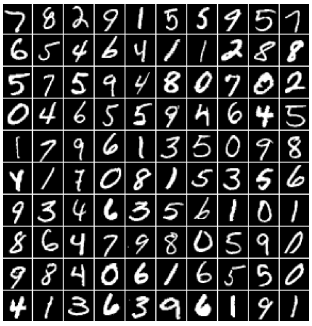
start with random initial weights and use backpropagation to learn weights to recognize digits

one output unit for each digit

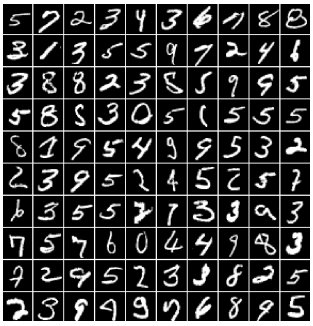
select output unit with maximum response
e.g. 9

5

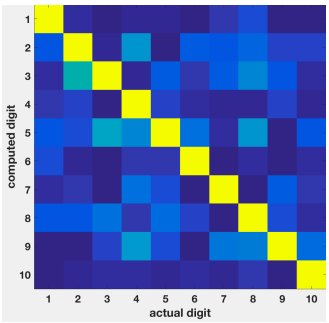
Results: learning handwritten digits



correct examples



wrong examples



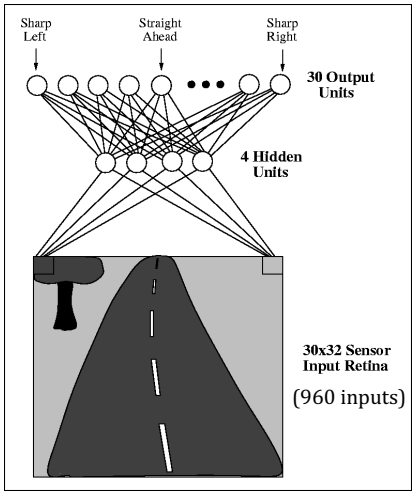
confusion matrix

overall classification accuracy: 87.1%

6

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!))

7