Artificial neural networks

A network of simple *neuron-like* computing elements

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

Learning to recognize input patterns

feedforward processing

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

\[
\Delta w_{jk} = \text{rate parameter} \times \text{current output of unit } H_j \times \text{slope of } O_k \times \text{benefit of adjusting } O_k
\]

\[
\Delta w_{ij} = r \times I_i \times (H_j (1 - H_j)) \times b_j
\]

where \( b_j \) is the benefit of adjusting \( H_j \):

\[
b_j = \sum_k w_{jk} \times (O_k (1 - O_k)) \times (O_k - d_k)
\]

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

\[
f'(x) = f(x) \times (1 - f(x))\]

(1) for each training sample, determine all the weight changes \( \Delta w_{jk} \) and \( \Delta w_{ij} \) 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 = \( \sum_k (O_k - d_k)^2 \)

\[\text{Note about rate parameter:}\]

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

**MNIST database:**
3000 28x28 images of handwritten digits

28

Sample image from MNIST database

25 hidden units

Input layer

Hidden layer

Output layer

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

Results: learning handwritten digits

Correct examples  Wrong examples  Confusion matrix

Overall classification accuracy: 87.1%
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!))