LEARNING IV

Overview

- Statistical machine learning
- Brains
- Neural networks
- Perceptrons
- Multilayer perceptrons

Statistical learning

- View learning as Bayesian updating of probability distribution over the *hypothesis space*
- Prior $\mathbf{P}(H)$, data $\mathbf{e} = e_1, \ldots, e_N$
- Given the data so far, each hypothesis has a posterior probability:

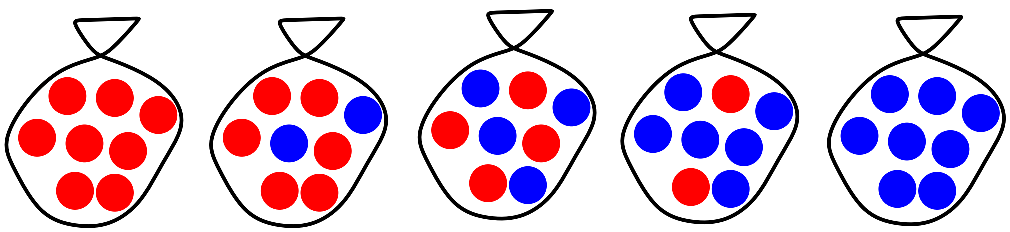
$$P(h_i|\mathbf{e}) = \alpha P(\mathbf{e}|h_i)P(h_i)$$

- Predictions use a likelihood-weighted average over the hypotheses:

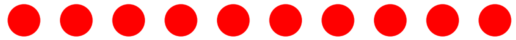
$$\mathbf{P}(X|\mathbf{e}) = \sum_i \mathbf{P}(X|\mathbf{e}, h_i)P(h_i|\mathbf{e}) = \sum_i \mathbf{P}(X|h_i)P(h_i|\mathbf{e})$$

Example

- Suppose there are five kinds of bags of marbles:
  - 10% are h_1: 100% blue marbles
  - 20% are h_2: 75% blue marbles + 25% red marbles
  - 40% are h_3: 50% blue marbles + 50% red marbles
  - 20% are h_4: 25% blue marbles + 75% red marbles
  - 10% are h_5: 100% red marbles

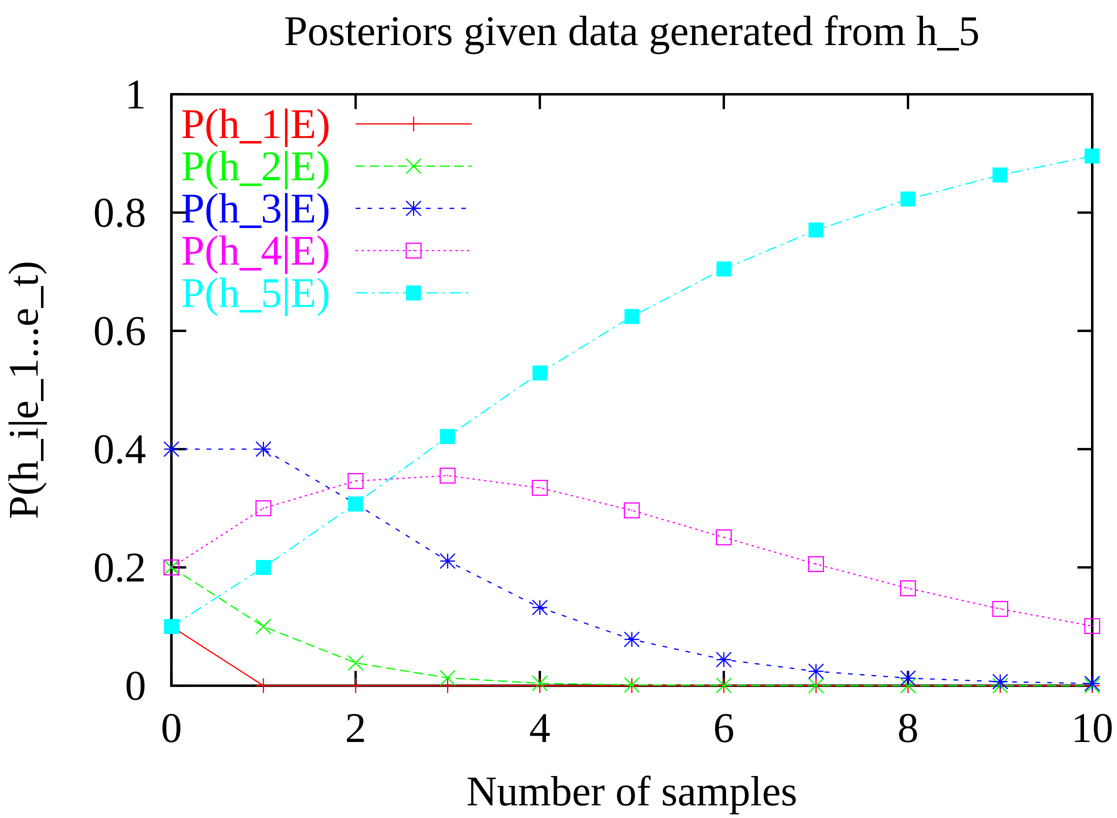


- Then we observe marbles drawn from some bag:



- What kind of bag is it? What color will the next marble be?

Posterior probability of hypotheses



Prediction probability

Sample	$P(red\|e_1 \ldots e_t)$
0.	0.5
1.	0.65
2.	0.7307692
3.	0.79605263
4.	0.8471074
5.	0.8859756
6.	0.9150034
7.	0.9364893
8.	0.9523869
9.	0.96420145
10.	x 0.97303146

MAP approximation

- Summing over the hypothesis space is often intractable
  (e.g., 18,446,744,073,709,551,616 Boolean functions of 6 attributes)
- *Maximum a posteriori* (MAP) learning: choose h_{MAP} maximizing $P(h_i|\mathbf{e})$

  I.e., maximize $P(\mathbf{e}|h_i)P(h_i)$ or $\log P(\mathbf{e}|h_i) + \log P(h_i)$
- Log terms can be viewed as (negative of) *bits to encode data given hypothesis* + *bits to encode hypothesis*

  This is the basic idea of *minimum description length* (MDL) learning
- For deterministic hypotheses, $P(\mathbf{e}|h_i)$ is 1 if consistent, 0 otherwise

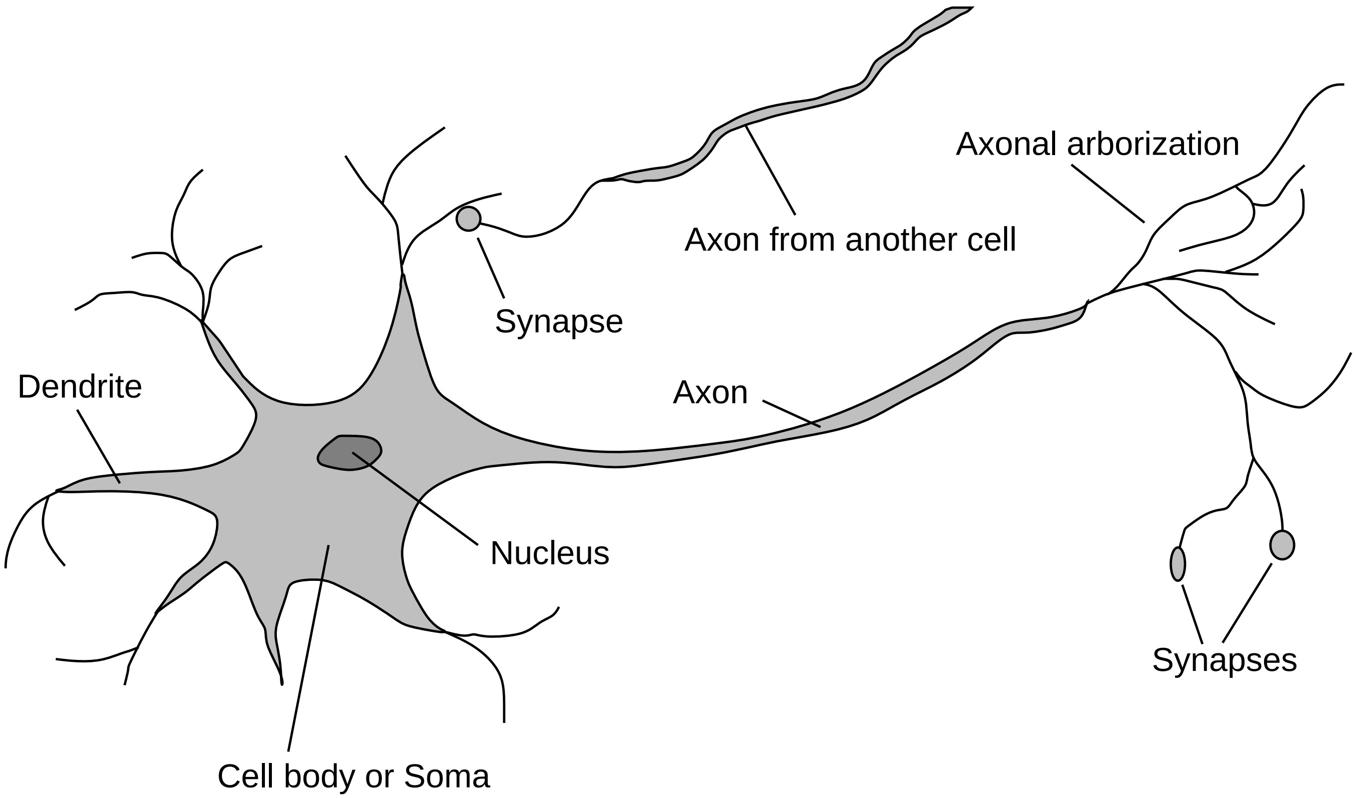
  $\Rightarrow$ MAP = simplest consistent hypothesis (cf. science)

ML approximation

- For large data sets, prior becomes irrelevant
- *Maximum likelihood* (ML) learning: choose h_{ML} maximizing $P(\mathbf{e}|h_i)$
- Simply get the best fit to the data; identical to MAP for uniform prior
  (which is reasonable if all hypotheses are of the same complexity)
- ML is the "standard" (non-Bayesian) statistical learning method

Brains

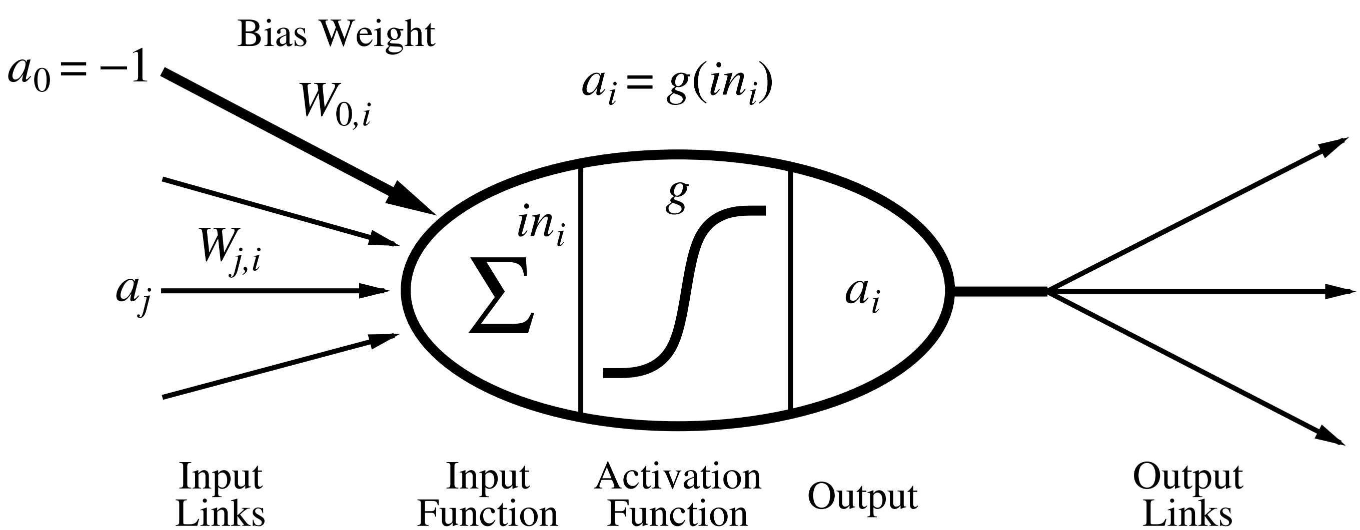
- 10^{11} neurons of > 20 types, 10^{14} synapses, 1ms–10ms cycle time
- Signals are noisy "spike trains" of electrical potential



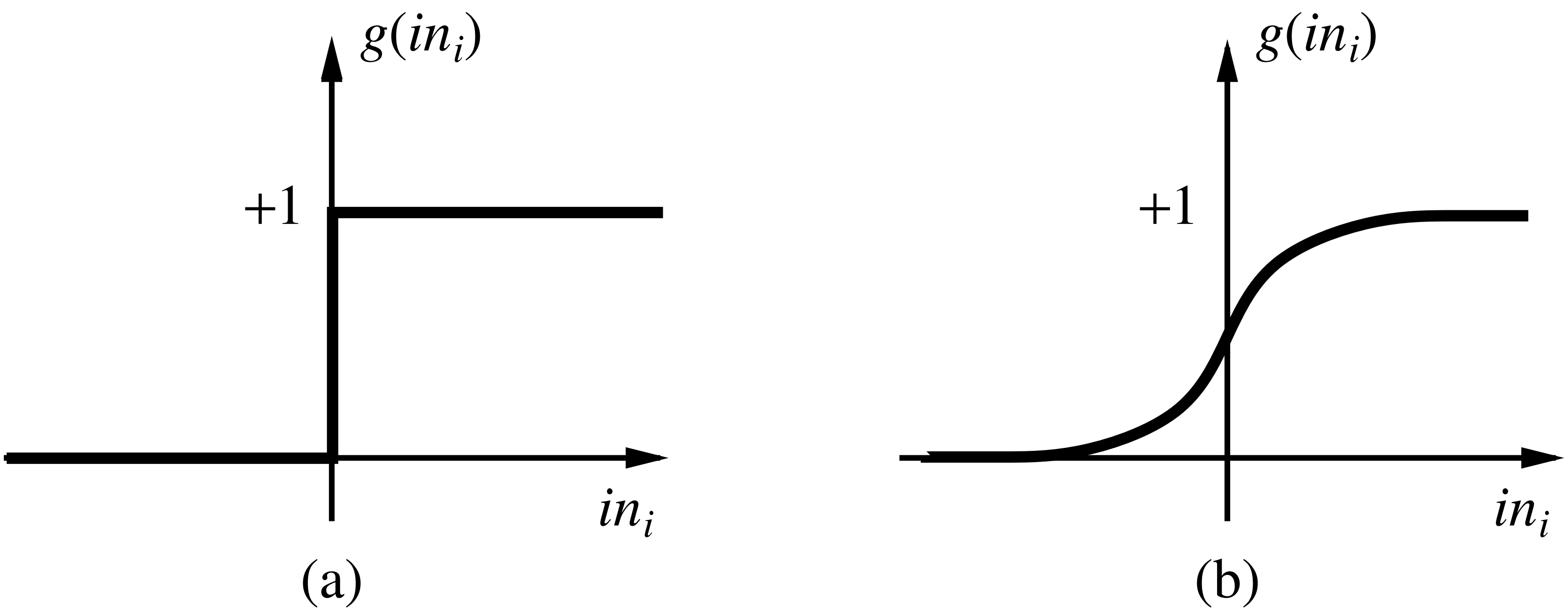
McCulloch–Pitts "unit"

- Output is a "squashed" linear function of the inputs:

$$a_i \leftarrow g(in_i) = g\left(\sum_j W_{j,i} a_j\right)$$

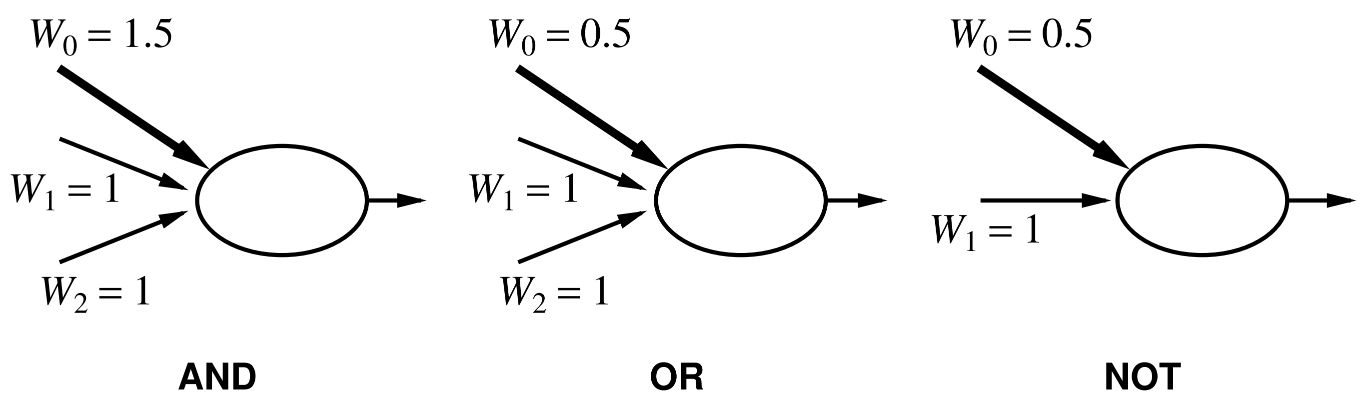


Activation functions



- (a) is a *step function* or *threshold function*
- (b) is a *sigmoid* function $1/(1+e^{-x})$
- Changing the bias weight $W_{0,i}$ moves the threshold location

Implementing logical functions

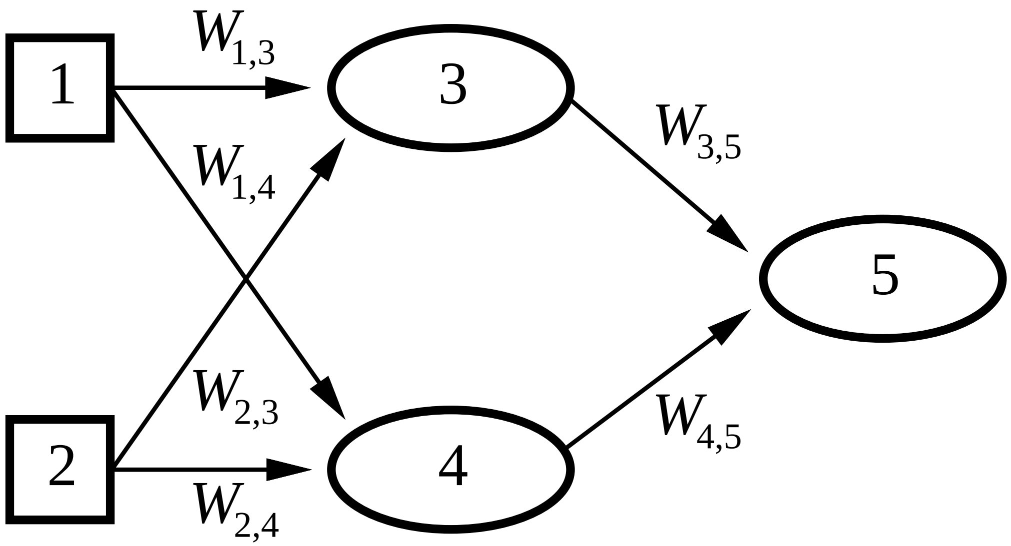


- McCulloch and Pitts: every Boolean function can be implemented

Network structures

- Feed-forward networks:
  - single-layer perceptrons
  - multi-layer perceptrons
- Feed-forward networks implement functions, have no internal state
- Recurrent networks:
  - Hopfield networks have symmetric weights ($W_{i,j} = W_{j,i}$) $g(x) = \text{sign}(x)$, $a_i = \pm 1$; *holographic associative memory*
  - Boltzmann machines use stochastic activation functions,
  - recurrent neural nets have directed cycles with delays
    $\Rightarrow$ have internal state (like flip-flops), can oscillate etc.

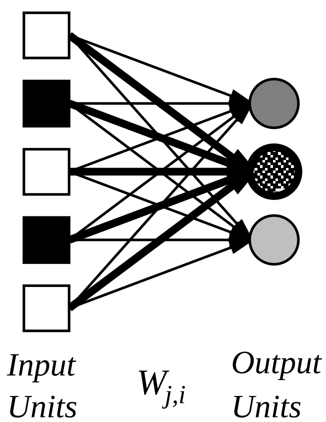
Feed-forward example

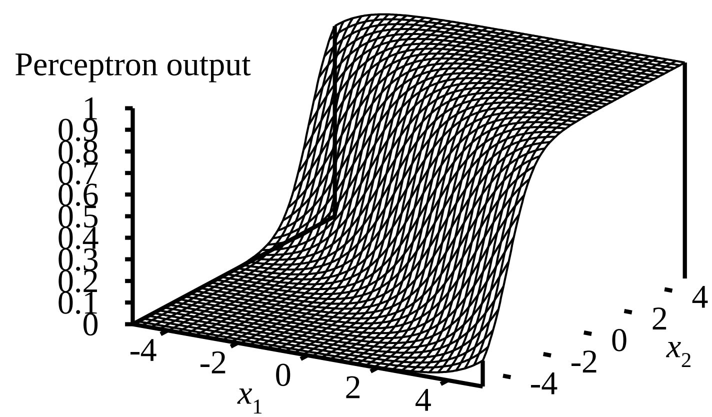


- Feed-forward network = a parameterized family of nonlinear functions:

$$\begin{aligned} a_5 &= g(W_{3,5} \cdot a_3 + W_{4,5} \cdot a_4) \\ &= g(W_{3,5} \cdot g(W_{1,3} \cdot a_1 + W_{2,3} \cdot a_2) + W_{4,5} \cdot g(W_{1,4} \cdot a_1 + W_{2,4} \cdot a_2)) \end{aligned}$$

Perceptrons

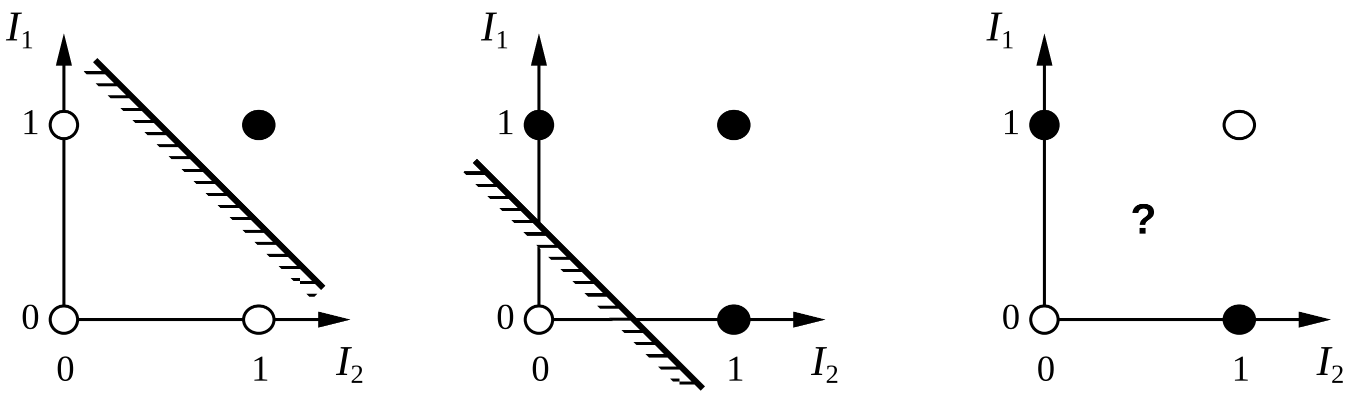




Expressiveness of perceptrons

- Consider a perceptron with g = step function (Rosenblatt, 1957, 1960)
- Can represent AND, OR, NOT, majority, etc.
- Represents a *linear separator* in input space:

$$\sum_j W_j x_j > 0 \quad \text{or} \quad \mathbf{W} \cdot \mathbf{x} > 0$$



(a) I_1 **and** I_2 (b) I_1 **or** I_2 (c) I_1 **xor** I_2

Perceptron learning

- Learn by adjusting weights to reduce *error* on training set
- The *squared error* for an example with input **x** and true output y is

$$E = \frac{1}{2}Err^2 \equiv \frac{1}{2}(y - h_{\mathbf{W}}(\mathbf{x}))^2 \ ,$$

- Perform optimization search by gradient descent:

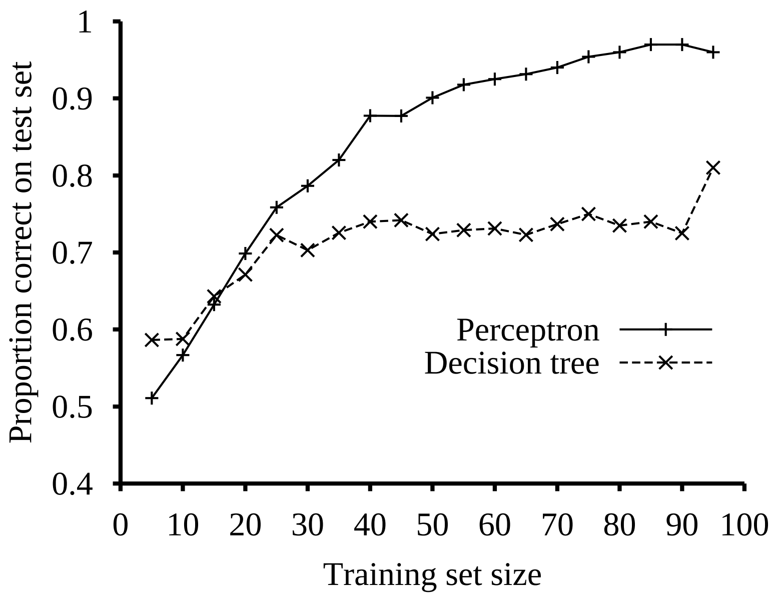
$$\begin{aligned}\frac{\partial E}{\partial W_j} &= Err \times \frac{\partial Err}{\partial W_j} = Err \times \frac{\partial}{\partial W_j}\left(y - g(\sum_{j=0}^{n} W_j x_j)\right) \\ &= -Err \times g'(in) \times x_j\end{aligned}$$

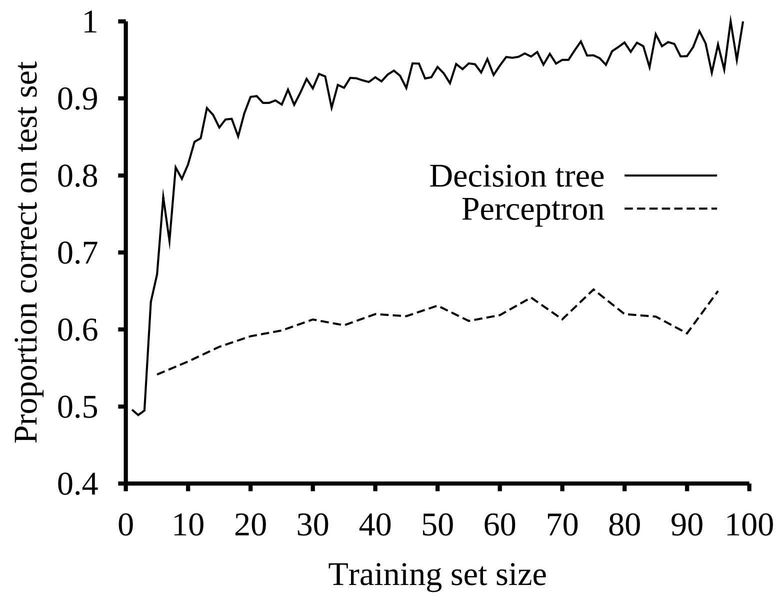
- Simple weight update rule:

$$W_j \leftarrow W_j + \alpha \times Err \times g'(in) \times x_j$$

Perceptron learning II

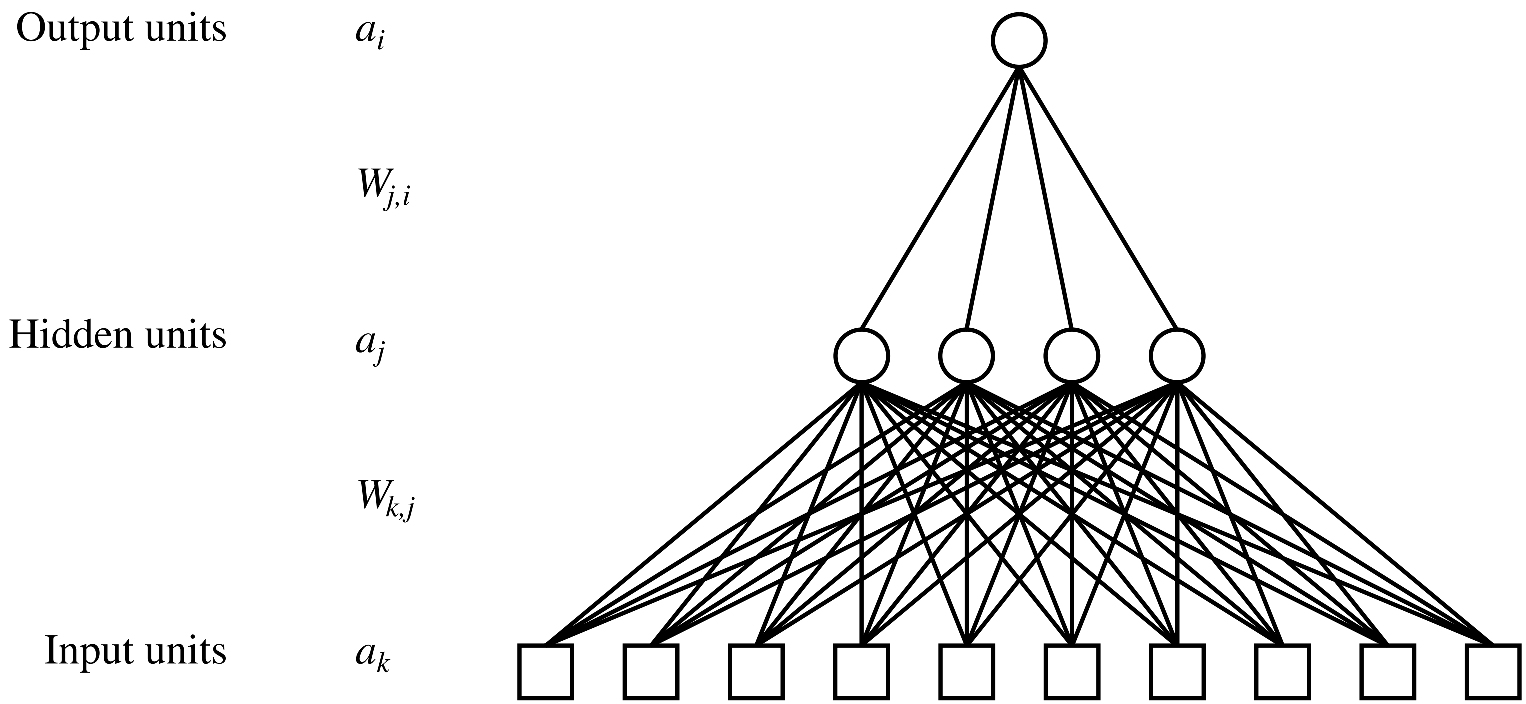
- E.g., +ve error $\Rightarrow$ increase network output
  $\Rightarrow$ increase weights on +ve inputs, decrease on -ve inputs
- Perceptron learning rule converges to a consistent function
  for any linearly separable data set





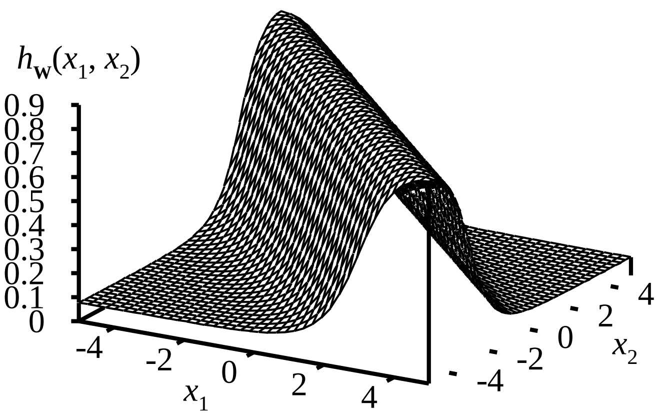
Multilayer perceptrons

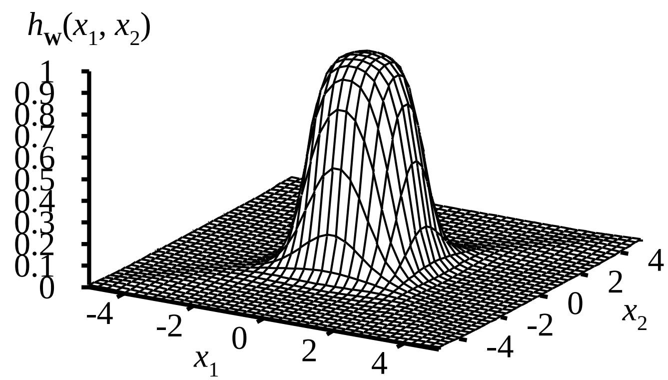
- Layers are usually fully connected
- Numbers of *hidden units* typically chosen by hand



Expressiveness of MLPs

- All continuous functions w/ 2 layers, all functions w/ 3 layers





Back-propagation learning

- Output layer: same as for single-layer perceptron,

$$W_{j,i} \leftarrow W_{j,i} + \alpha \times a_j \times \Delta_i$$

  where $\Delta_i = Err_i \times g'(in_i)$

- Hidden layer: *back-propagate* the error from the output layer:

$$\Delta_j = g'(in_j) \sum_i W_{j,i} \Delta_i \ .$$

- Update rule for weights in hidden layer:

$$W_{k,j} \leftarrow W_{k,j} + \alpha \times a_k \times \Delta_j \ .$$

- Most neuroscientists deny that back-propagation occurs in the brain

Summary

- Full Bayesian learning gives best possible predictions but is intractable
- MAP learning balances complexity with accuracy on training data
- Maximum likelihood assumes uniform prior, OK for large data sets
- ML for continuous spaces using gradient (etc.) of log likelihood
- Most brains have lots of neurons; each neuron $\approx$ linear–threshold unit (?).
- Perceptrons (one-layer networks) insufficiently expressive.
- Multi-layer networks are sufficiently expressive; can be trained by gradient descent, i.e., error back-propagation.