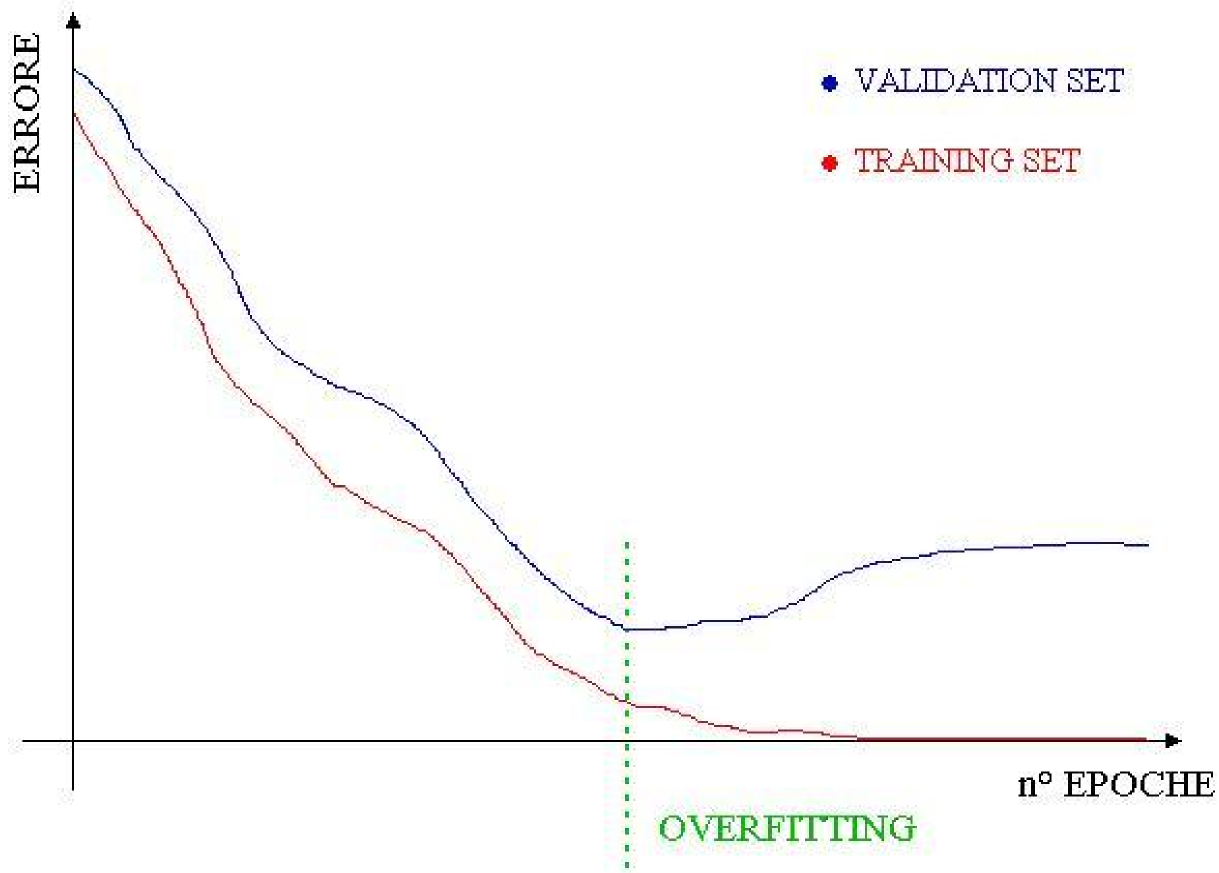
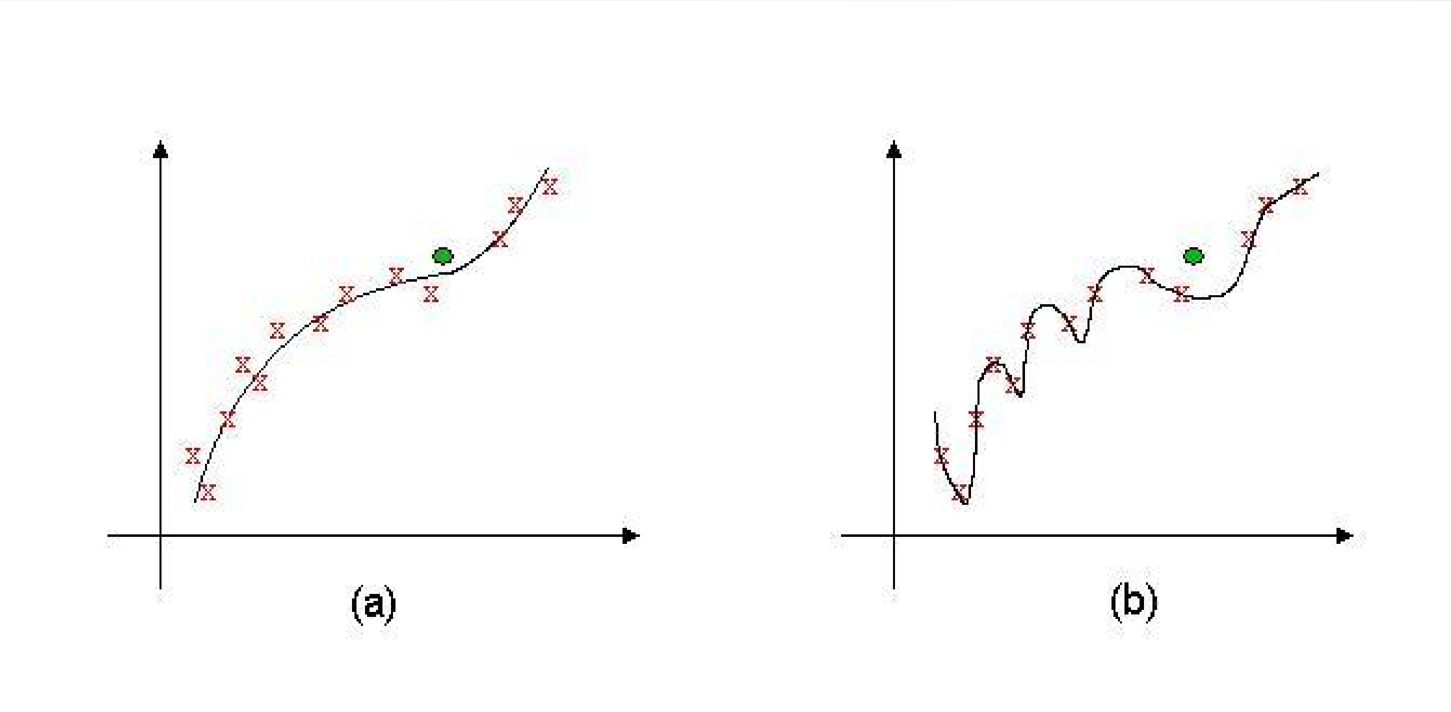
## Theoretical / Practical Questions

- How many layers are needed for a given task?
- How many units per layer?
- To what extent does representation matter?
- What do we mean by generalization?
- What can we expect a network to generalize?
  - Generalization: performance of the network on data not included in the training set
  - Size of the training set: how large a training set should be for “good” generalization?
  - Size of the network: too many weights in a network result in poor generalization


ERRORE
VALIDATION SET
TRAINING SET
n° EPOCHE
OVERFITTING




(a) A good fit to noisy data. (b) Overfitting of the same data: the fit is perfect on the "training set" (x's), but is likely to be poor on "test set" represented by the circle.

# Motivation

- The size (i.e. the number of hidden units) of an artificial neural network affects both its functional capabilities and its generalization performance

- Small networks could not be able to realize the desired input / output mapping

- Large networks lead to poor generalization performance

# The Pruning Approach

Train an over-dimensioned net and then remove redundant nodes and / or connections:

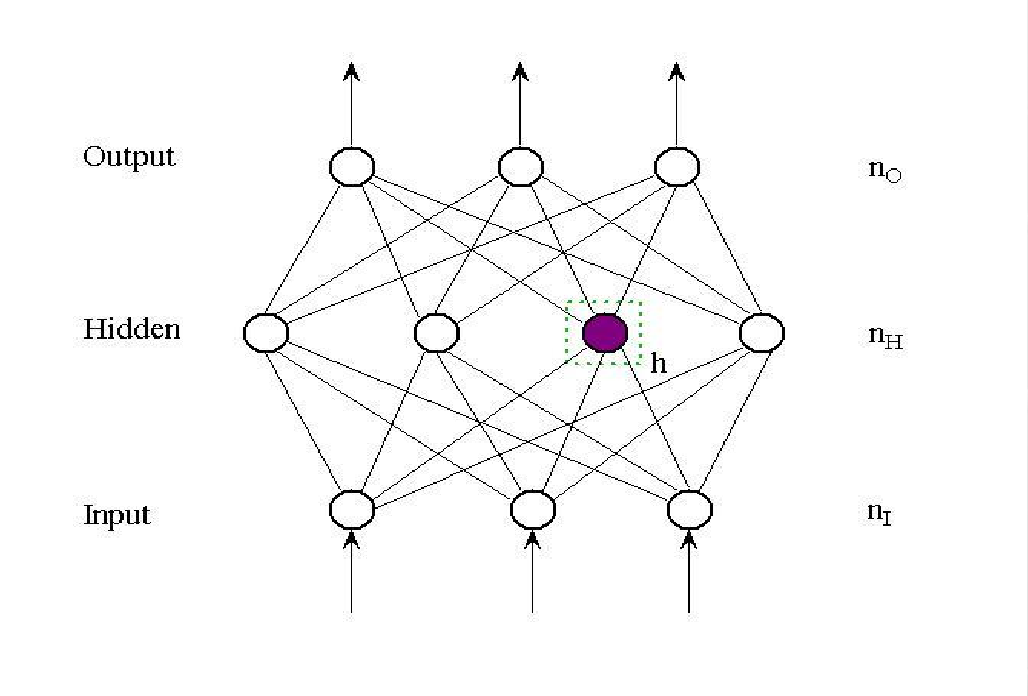
- Sietsma & Dow (1988, 1991)
- Mozer & Smolensky (1989)
- Burkitt (1991)

Adavantages:

- arbitrarily complex decision regions
- faster training
- independence of the training algorithm

# The Proposed Method

Consider (for simplicity) a net with one hidden layer:




Suppose that node h is to be removed:

Remove h (and its in/out connections) and adjust the remaining weights so that the I/O behavior is the same

This is equivalent to solving the system:

$$\forall i = 1 \ldots n_O \ , \ \forall m = 1 \ldots P \qquad \sum_{j=1}^{nh} w_{ij} \ y_j^{(m)} = \sum_{\substack{j=1 \\ j \neq h}}^{nh} \left( w_{ij} + d_{ij} \right) y_j^{(m)}$$

which is equivalent to:

$$\sum_{j \neq h} d_{ij} \ y_h^{(m)} = w_{ih} \ y_h^{(m)}$$

In a more compact notation:

$$A \, \overline{x} = b$$

with $A \in \Re^{P n_O \times n_O (nh - 1)}$

LS solution : $\min_{\overline{x}} \left\| A \overline{x} - b \right\|$

# Detecting Excessive Units

- Residual-reducing methods for LLSPs start with an initial solution $x_0$ and produces a sequences of points $\{x_k\}$ so that the residuals

$$\| Ax_k - b \| = r_k$$

decrease $r_k \leq r_{k+1}$

- Starting point: $x_0 = 0 \quad \left( \Rightarrow r_0 = \| b \| \right)$

- Excessive units can be detected so that $\| b \|$ is minimum

## The Proposed Approach

Instead of analyzing the consistency of the system, we directly solve it in the least squares sense:

$$FIND\ x\ such\ that\ \| Ax_k - b \|\ is\ minimum$$

The method chosen is a projection method developed by Bjorck&Elfving (BIT, 1979) called CGPCNE

# The Pruning Algorithm

1) Start with an over-sized trained network

2) Repeat

   2.1) find the hidden unit h for which $\|b\|$ is minimum
   2.2) solve the corresponding system
   2.3) remove unit h

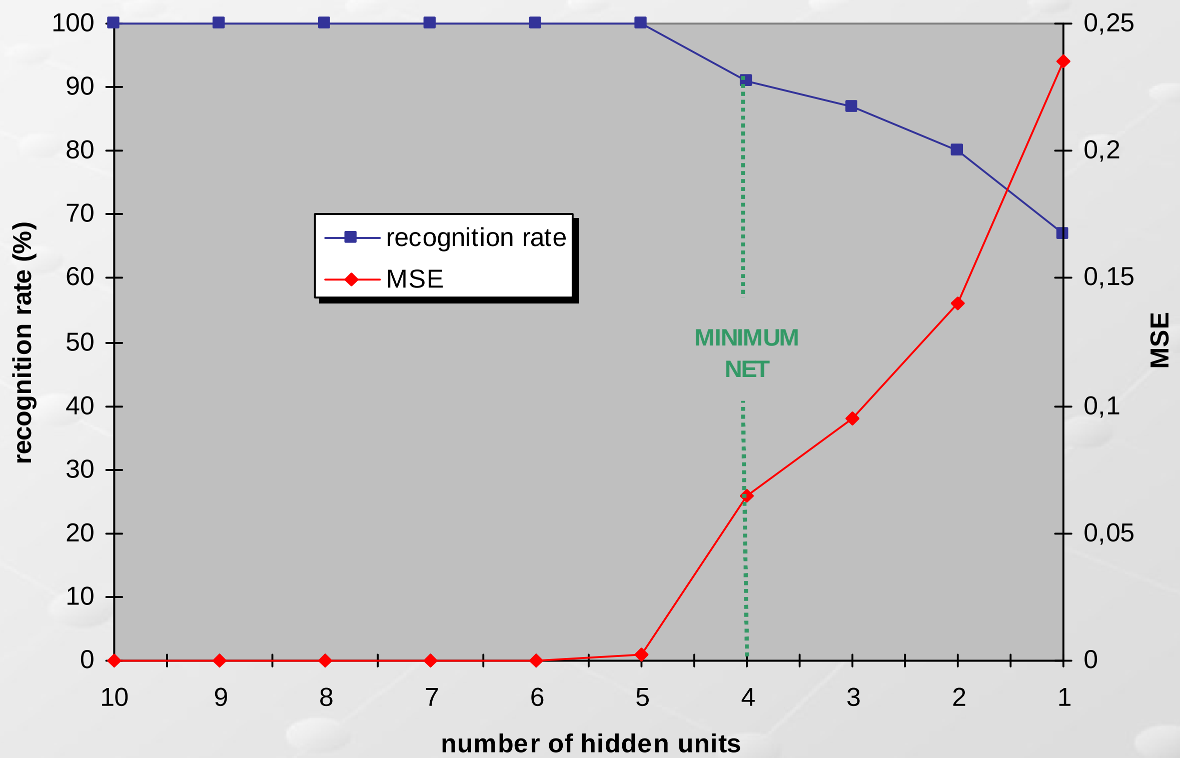
   Until Perf(pruned) – Perf(original) < epsilon

3) Reject the last reduced network

# Example I : 4-bit parity

Ten initial 4-10-1 networks

Pruned nets: nine 4-5-1, one 4-4-1 — 5 hidden nodes (average)



# Example II : 4-bit simmetry

Ten initial 4-10-1 networks

Pruned nets ⇒ 4.6 hidden nodes (average)