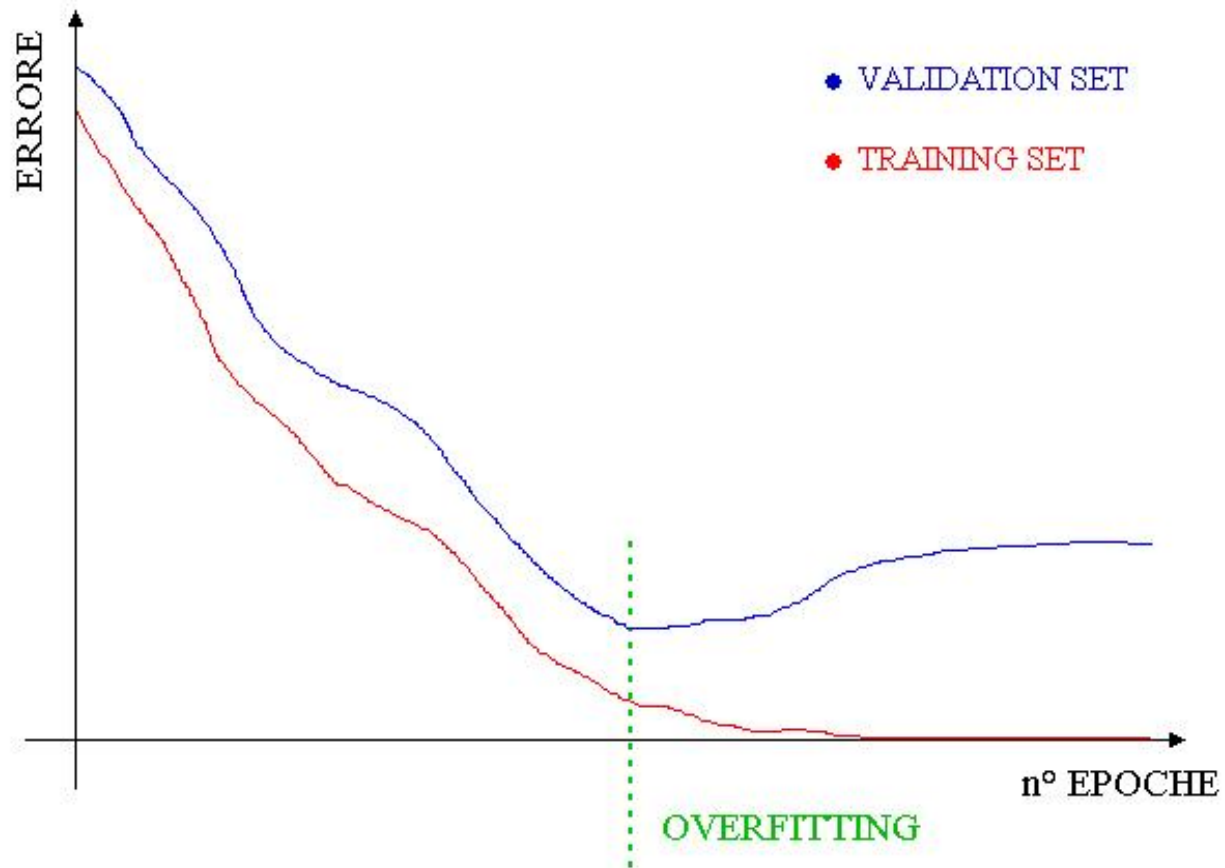
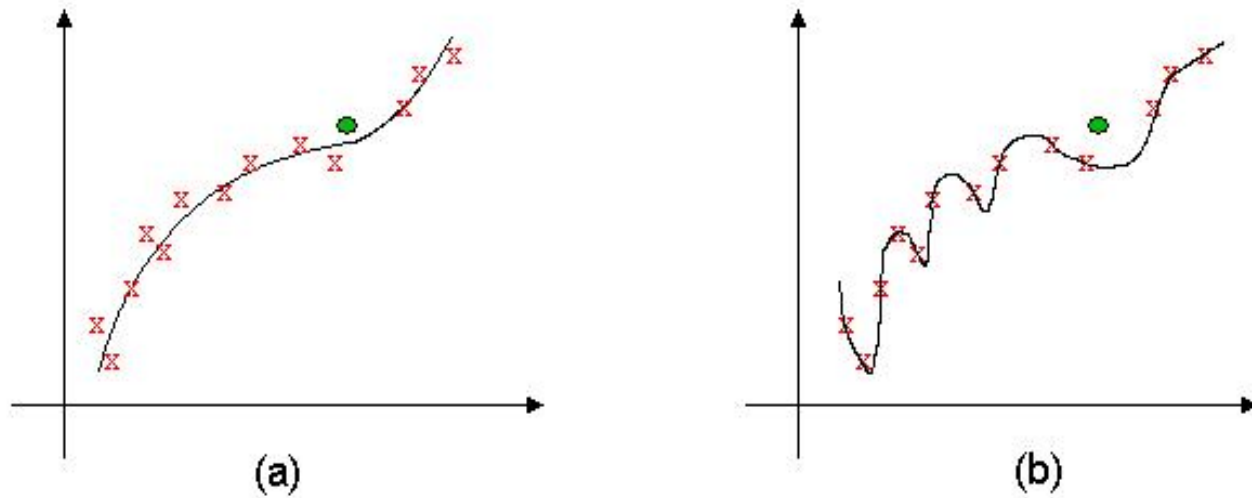


# Theoretical / Practical Questions

- 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





(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-dimensional net and then remove redundant nodes and / or connections:

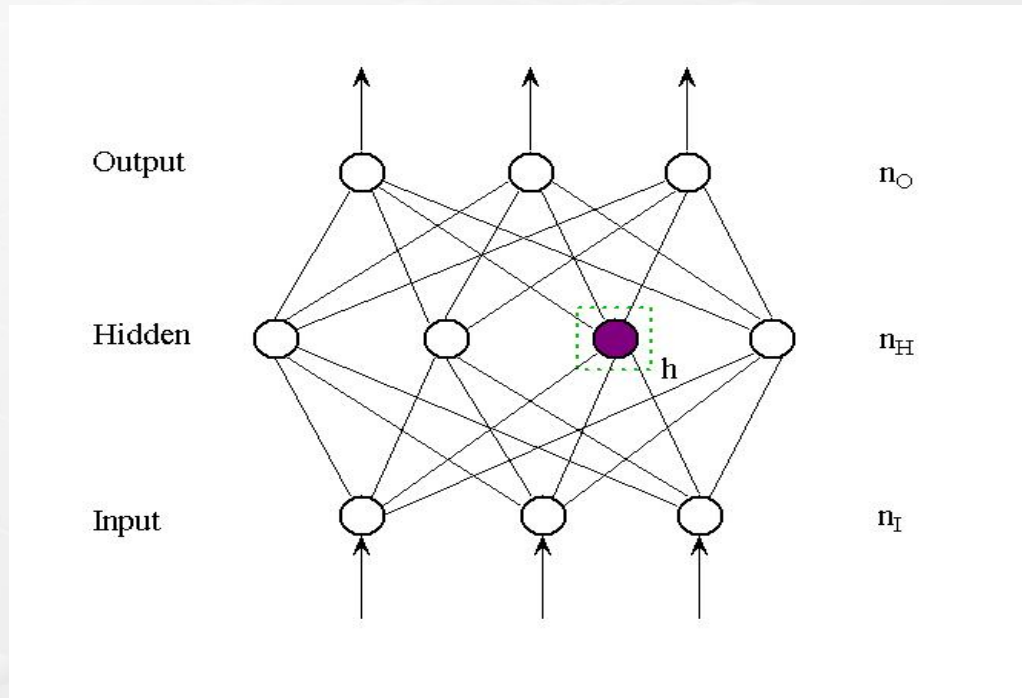
- Sietsma & Dow (1988, 1991)
- Mozer & Smolensky (1989)
- Burkitt (1991)

Advantages:

- 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:

$$\forall i = 1 \dots n_o, \forall \mu = 1 \dots P \quad \sum_{j=1}^{nh} w_{ij} y_j^{(\mu)} = \sum_{\substack{j=1 \\ j \neq h}}^{nh} (w_{ij} + \delta_{ij}) y_j^{(\mu)}$$

which is equivalent to:

$$\sum_{j \neq h} \delta_{ij} y_j^{(\mu)} = w_{ih} y_h^{(\mu)}$$

In a more compact notation:

$$A \bar{x} = b$$

with  $A \in \mathfrak{R}^{P n_o \times n_o (nh-1)}$

LS solution :

$$\min_x \left\| A \bar{x} - b \right\|$$

## 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:  $x_0 = 0$  ( $\Rightarrow r_0 = \|b\|$ )
- 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  $h$

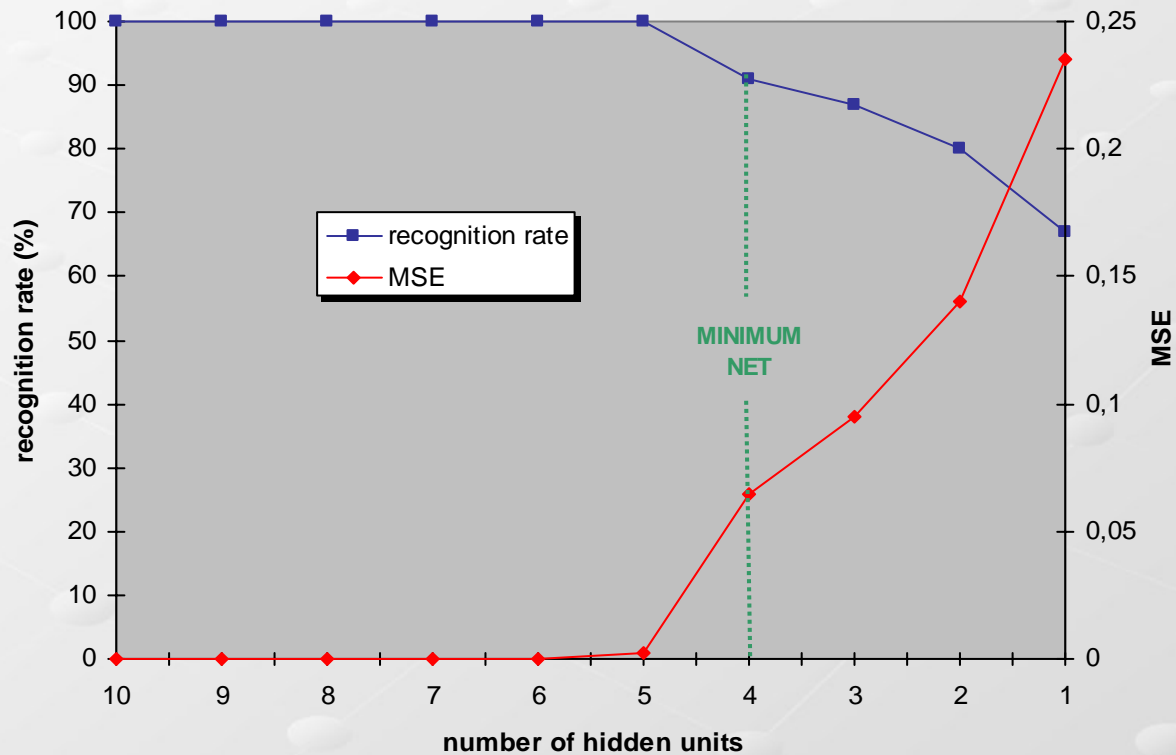
Until  $\text{Perf}(\text{pruned}) - \text{Perf}(\text{original}) < \text{epsilon}$
- 3) Reject the last reduced network

# Example I : 4-bit parity

Ten initial 4-10-1 networks


Pruned nets {  
  nine 4-5-1  
  one 4-4-1

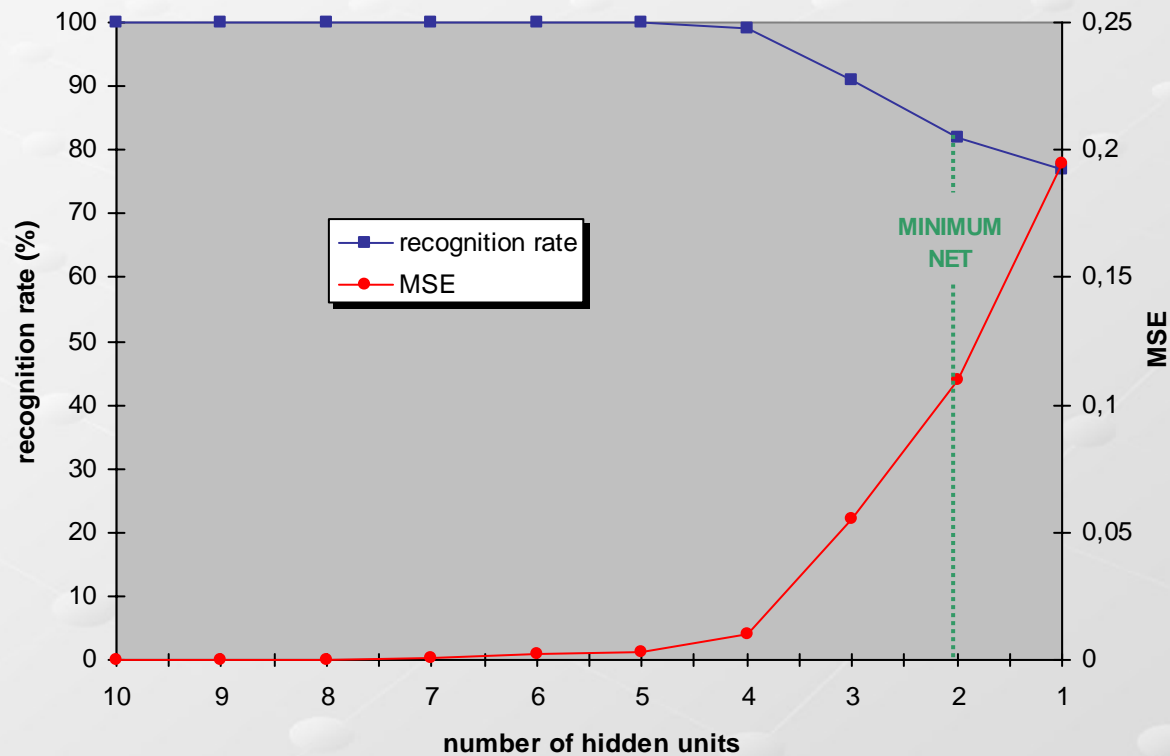
5 hidden nodes (average)



## Example II : 4-bit simmetry

Ten initial 4-10-1 networks

Pruned nets  4.6 hidden nodes (average)



# Optimal Brain Surgeon (OBS)

The question then becomes, which weights should be eliminated? How should we adjust the remaining weights (if at all) for best performance? How can this be done in a computationally efficient way?

Magnitude based methods [Hertz, Krogh and Palmer, 1991] eliminate weights that have the smallest magnitude. This simple and naively plausible idea unfortunately often leads to the elimination of the wrong weights — small weights can be necessary for low error. Both Optimal Brain Damage [Le Cun, Denker and Solla, 1990] and our Optimal Brain Surgeon use instead a criterion of minimal increase in error on the training data. The superiority of OBS lies in great part to the fact that it makes no restrictive assumptions about the form of the network's Hessian that OBD does — assumptions that we find are invalid for many networks. Moreover, unlike OBD, OBS does not demand (typically slow) retraining.

In deriving our method we begin, as do Le Cun, Denker and Solla [1990], by considering a network trained to a local minimum in error. The functional Taylor series of the error with respect to weights is:

$$\delta E = \underbrace{\left( \frac{\partial E}{\partial \mathbf{w}} \right)^T \cdot \delta \mathbf{w}}_{\approx 0} + \frac{1}{2} \delta \mathbf{w}^T \cdot \mathbf{H} \cdot \delta \mathbf{w} + \underbrace{O(\|\delta \mathbf{w}\|^3)}_{\approx 0} \quad (1)$$

where  $\mathbf{H} = \frac{\partial^2 E}{\partial \mathbf{w}^2}$  is the Hessian matrix (containing all second order derivative information) and the superscript T denotes vector transpose. For a network trained to a local minimum in error, the first (linear) term vanishes; we also ignore the third and all higher order terms. Our goal is then to set one of the weights to zero (which will be called  $w_q$ ) to minimize the increase in error given by Eq. 1. Eliminating  $w_q$  can be expressed as:

$$\delta w_q + w_q = 0 \quad \text{or} \quad \mathbf{e}_q^T \cdot \delta \mathbf{w} + w_q = 0 \quad (2)$$

where  $\mathbf{e}_q$  is the unit vector in weight space corresponding to (scalar) weight  $w_q$ . Our goal is then to solve:

$$\text{Min}_q \{ \text{Min}_{\delta \mathbf{w}} \{ \frac{1}{2} \delta \mathbf{w}^T \cdot \mathbf{H} \cdot \delta \mathbf{w} \} \text{ such that } \mathbf{e}_q^T \cdot \delta \mathbf{w} + w_q = 0 \} \quad (3)$$

To this end we form a Lagrangian from Eqs. 1 and 2:

$$L = \frac{1}{2} \delta \mathbf{w}^T \cdot \mathbf{H} \cdot \delta \mathbf{w} + \lambda (\mathbf{e}_q^T \cdot \delta \mathbf{w} + w_q) \quad (4)$$

where  $\lambda$  is a Lagrange undetermined multiplier. We take functional derivatives, employ the constraints of Eq. 2, and use matrix inversion to find that the optimal weight change and resulting change in error are:

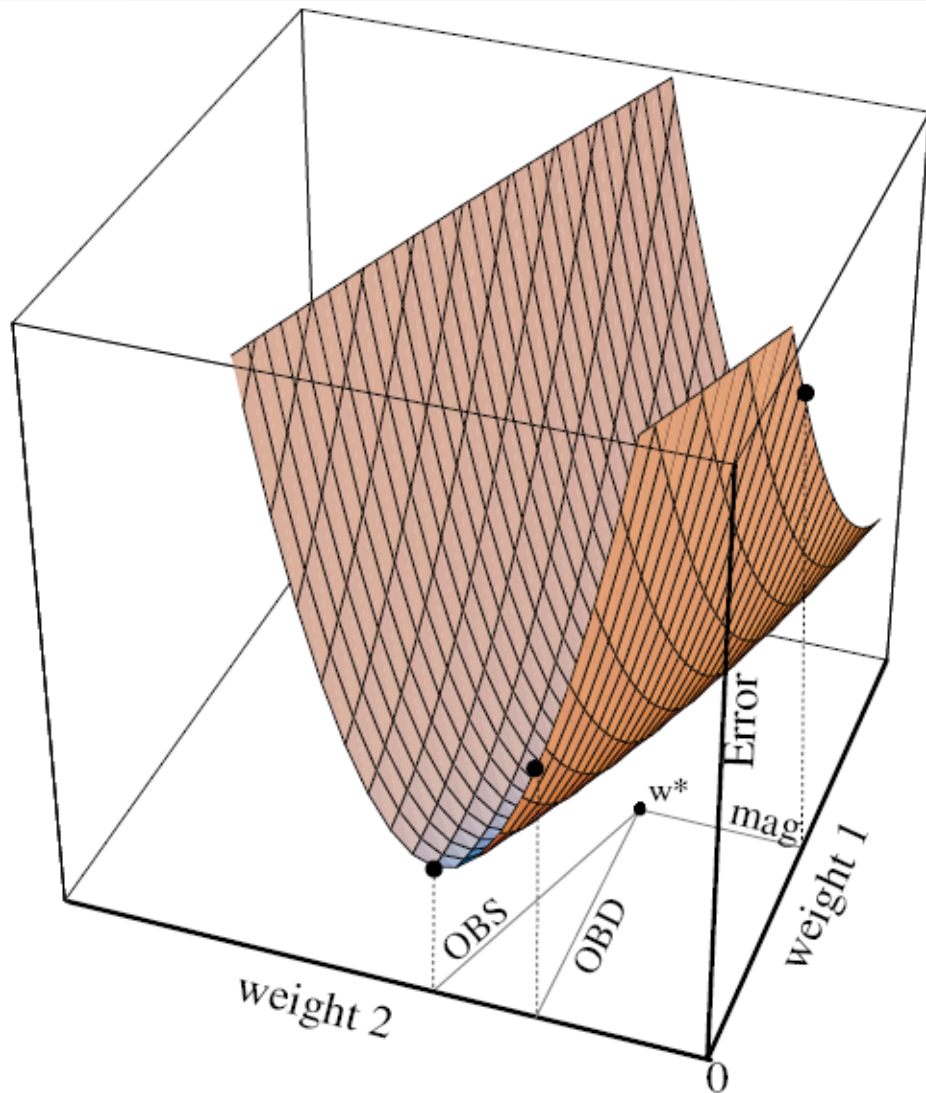
$$\delta \mathbf{w} = -\frac{w_q}{[\mathbf{H}^{-1}]_{qq}} \mathbf{H}^{-1} \cdot \mathbf{e}_q \quad \text{and} \quad L = \frac{1}{2} \frac{w_q^2}{[\mathbf{H}^{-1}]_{qq}} \quad (5)$$

Note that neither  $\mathbf{H}$  nor  $\mathbf{H}^{-1}$  need be diagonal (as is assumed by Le Cun et al.); moreover, our method recalculates the magnitude of *all* the weights in the network, by the left side of Eq.5. Figure 1 illustrates the basic idea.

## Optimal Brain Surgeon procedure

1. Train a “reasonably large” network to minimum error.
2. Compute  $\mathbf{H}^{-1}$
3. Find the  $q$  that gives the smallest  $L = w_q^2 / (2[\mathbf{H}^{-1}]_{qq})$  (cf. Eq.5).  
If this candidate error increase is much smaller than  $E$ , then the  $q^{\text{th}}$  weight should be deleted, and we proceed to step 4; otherwise go to step 5.
4. Use the  $q$  from step 3 to update *all* weights (Eq.5). Go to step 2.
5. No more weights can be deleted without large increase in  $E$ . (At this point it may be desirable to retrain the network.)





**Figure 1:** Error as a function of two weights in a network. The (local) minimum occurs at weight  $w^*$ , found by gradient descent learning. In this illustration, a magnitude based pruning technique (mag) then removes the smallest weight, weight 2; Optimal Brain Damage before retraining (OBD) removes weight 1. In contrast, our Optimal Brain Surgeon method (OBS) removes weight 1 but *also* automatically adjusts the value of weight 2 to minimize the error. The error surface here is general in that it has different curvatures (second derivatives) along different directions, a minimum at a non-special weight value, and a non-diagonal Hessian (i.e., principal axes are *not* parallel to the weight axes). The relative magnitudes of the error after pruning (before retraining, if any) depend upon the particular problem, but to second order obey:  $E(\text{mag}) \geq E(\text{OBD}) \geq E(\text{OBS})$ , which is the key to the superiority of OBS. We call our method Optimal Brain *Surgeon* because in addition to cutting out weights, it calculates and *changes* the strengths of other weights without the need for retraining.

# Comparison with OBD

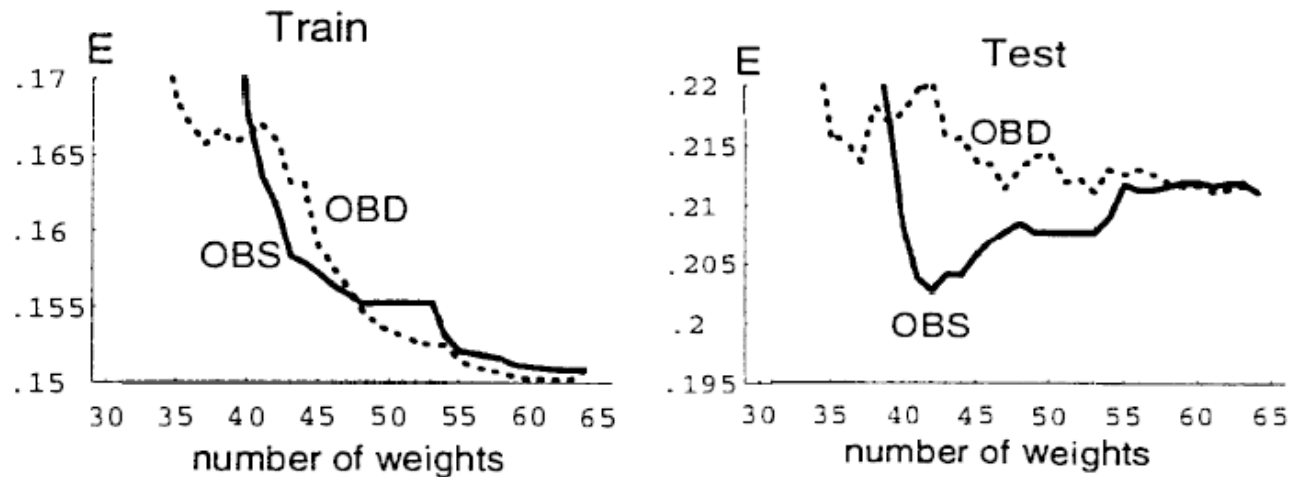


Figure 1: *OBS* and *OBD* training error on a sum of Gaussians prior pattern classification task as a function of the number of weights in the network. (Pruning proceeds right to left.) *OBS* pruning employed  $\alpha = 10^{-6}$  (cf., Eq. 5); *OBD* employed 60 retraining epochs after each pruning.

# OBS/OBD comparison on “stopped” networks

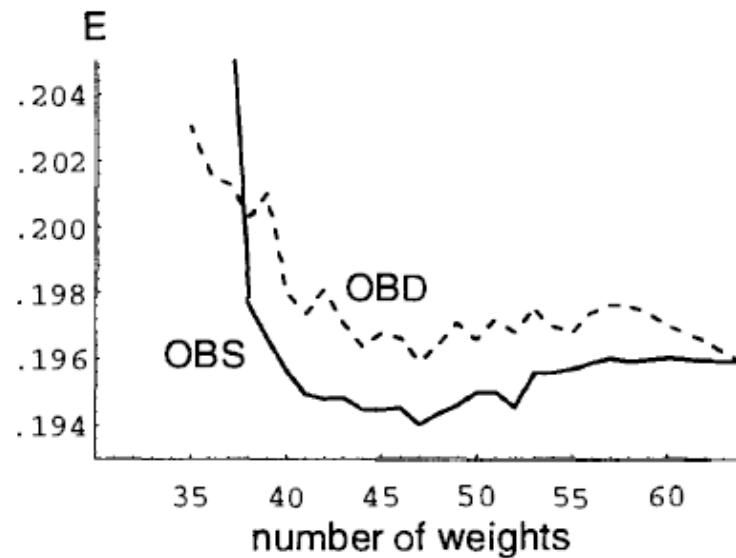


Figure 2: A 64-weight network was trained to minimum validation error on the Gaussian problem — *not*  $w^*$  — and then pruned by *OBD* and by *OBS*. The test error on the resulting network is shown. (Pruning proceeds from right to left.) Note especially that even though the network is far from  $w^*$ , *OBS* leads lower test error over a wide range of prunings, even through *OBD* employs retraining.

# Riferimenti bibliografici

- G. Castellano, A. M. Fanelli, M. Pelillo. A method of pruning feed-forward neural networks. *IEEE Transactions on Neural Networks*, 1997.
- B. Hassibi, D. G. Stork. Second-order derivatives for network pruning: Optimal Brain Surgeon. In: *NIPS* 1993.