Slightly modified version of the publication presented at the Fourteenth European Meeting on Cybernatics and Systems Research (EMCSR’98), Vienna, pp. 599-604, 1998.

A lifelong learning approach for incremental neural networks[1]

Fred Henrik Hamker, Horst-Michael Gross
Technische Universität Ilmenau, Neuroinformatik, D-98684 Ilmenau, Germany
http://cortex.informatik.tu-ilmenau.de/~fred
e-mail: fred@informatik.tu-ilmenau.de

Abstract

The aim of this approach is to reveal new lifelong learning algorithms, based on incremental neural networks with a local insertion criterion. Lifelong learning i.e., learning throughout the entire lifetime of a neural network, has to cope with the stability-plasticity dilemma. Our solution, a Lifelong Growing Neural Gas learns when to stop insertion inside of overlapping decision areas and adapts to changing environments while preserving old prototype patterns. Furthermore, a general removal criterion is introduced.

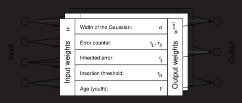
1 Introduction

Learning is one of the main issues of artificial neural network design. It describes a mechanism by which a system obtains a representation of its environment. Recent research addresses the topic of on-line learning, incremental learning and lifelong learning, which all discuss the same problem but emphasize different aspects. The necessity for on-line learning, in which the couplings of the network are updated after the presentation of each example, arises if not all training patterns are available all the time [Freeman and Saad, 1997; Heskes and Kappen, 1993]. Most publications referring to on-line learning focus on the role of the learning rule and the convergence of the learning process, but stop learning when a performance criterion is reached. For systems, like robots, which are faced with patterns during their entire lifetime, studying on-line learning in contexts such as a changing environment [Heskes and Kappen, 1993] encounters the problems of stability and plasticity. Incremental learning addresses the ability of repeatedly training a network with new data, without destroying the old prototype pattern. Lifelong learning emphasizes learning throughout the entire lifetime and has to cope with changing environments and overlapping decision areas. It is not sufficient to only follow a non-stationary input distribution like [Fritzke, 1997], lifelong learning has to solve the stability-plasticity dilemma, which demands the adaption to new patterns and the preservation of old patterns. Networks with a local or distributed representation of knowledge appear to be good candidates for lifelong learning scenarios. One type of a local representation of knowledge utilized in recent literature of on-line learning are RBF’s [Freeman and Saad, 1997] or similar networks [Gaussier and Zrehen, 1994]. Nevertheless, they have a fixed number of nodes which has to be determined by the designer. One way to tackle this problem, is to spend a huge number of fixed randomly initialized RBF units [Salomon, 1997]. Although learning is very fast [Salomon, 1997], due to the fixed vector quantization, the required number of nodes increases exponentially with the dimension of the input space. At least, the problem remains the same – the designer has to determine the density of RBF units in the input space.

Incremental networks have the advantage that the number of nodes is also a result of learning by doing. The most important question in lifelong learning incremental networks concerns the rule of insertion. ART networks, like FAM [Carpenter et al., 1992] insert new nodes based on a similarity measure. Other families of incremental networks use an error measure to insert new nodes. They can be subdivided into local error based insertion rules like Growing Cell Structures (GCS) [Fritzke, 1994], Growing Neural Gas (GNG) [Fritzke, 1995], Dynamic Cell Structures (DCS) [Bruske and Sommer, 1995] and global error based insertion rules like Cascade-Correlation [Fahlman and Lebiere, 1989]. Inserting new nodes solely depending on the similarity of the input pattern leads to a purely sensor-based representation, which does not reflect the requirements of further processing stages. In contrast, an error-based insertion adapts the representation depending on the task and therefore leads to a task-based representation [Hamker and Gross, 1997]. Compared to a global insertion criterion, a local criterion has the important advantage that insertion can be controlled locally. Summarizing, incremental networks with a local error based insertion rule are optimal candidates for lifelong learning – but only if the insertion of new nodes can be managed properly.

---

[1] partially supported by the BMBF, Grant No. 413-5839-01 M 3014C - MIRIS-Project.


Input
Input weights
w
Width of the Gaussian: σ
Error counter: τ_L, τ_S
Inherited error: τ_I
Insertion threshold: τ_ϑ
Age (youth): Y
Output weights
w^{out}
Output

comparison to a ML

$$d_b = \min_{i \in G}(d_i) \;; \quad d_s = \min_{i \in G, i \neq b}(d_i) \;;$$
$$d_i = \|\boldsymbol{x} - \boldsymbol{w}_i\| \qquad \forall i \in \boldsymbol{G}$$

$$y_i = e^{-\frac{\|\boldsymbol{x}-\boldsymbol{w}_i\|^2}{\sigma_i^2}} \;; \quad \sigma_i = \frac{1}{\|N_i\|} \sum_{j \in N_i} \|\boldsymbol{w}_i - \boldsymbol{w}_j\| \qquad \forall i \in \boldsymbol{G}$$

$$B^L_{(b/c)} = \frac{\tau_{S(b/c)} + 1}{\tau_{L(b/c)} + 1} \qquad \forall c \in N_b$$

$$\eta^i_{(b/c)} = \begin{cases} 0 & if \quad \alpha^i_{(b/c)} \prec 0 \\ \eta_{(b/n)} & if \quad \alpha^i_{(b/c)} \succ 1 \\ \alpha^i_{(b/c)} \cdot \eta_{(b/n)} & else \end{cases}$$
$$\alpha^i_{(b/c)} = \frac{B^L_{(b/c)}}{1 + \vartheta^i_L} + Y_{(b/c)} - 1 \qquad \forall c \in N_b$$

$$\eta^{i'}_{(b/c)} = \max(\eta_{M(b/c)}, \eta^i_{(b/c)}) \;;$$
$$\eta_{M(b/c)} = \eta^i_{(b/c)} \cdot (1 - y_{(b/c)}) \cdot \vartheta_M \qquad \forall c \in N_b$$

$$\Delta \boldsymbol{w}_b = \eta^{i'}_b(\boldsymbol{x} - \boldsymbol{w}_b)$$
$$\Delta \boldsymbol{w}_c = \eta^{i'}_n(\boldsymbol{x} - \boldsymbol{w}_c) \qquad \forall c \in N_b$$

$$B^I_i = \tau_{Li} - \tau_{\vartheta i} \cdot (1 + \vartheta_{ins}) \qquad \forall i \in \boldsymbol{G}$$

$$0 \prec K_{ins,q} = \max_{i \in G}(K_{ins,i}) \;; \quad B^I_f = \max_{i \in N_q}(B^I_i) \;;$$
$$K_{ins,i} = B^I_i - Y_i \qquad \forall i \in \boldsymbol{G}$$

$$\tau_{Li} \succ \tau_{Ii} \cdot (1 - \vartheta_{ins}) \qquad \forall i \in \{q, f, r\}$$

$$\tau_{\vartheta i} := \tau_{\vartheta i} + \eta_\vartheta \cdot (\tau_{Li} - \tau_{\vartheta i} \cdot (1 - \vartheta_{ins}))$$
$$\forall i \in \{k \mid \tau_{Lk} \succ \tau_{Ik} \cdot (1 - \vartheta_{ins}); q, f, r\}$$

$$\tau_{Ii} = \tau_{Li} \qquad \forall i \in \{q, f, r\}$$

$$\vartheta_{del} \prec K_{del,d} = \min_{i \in G}(K_{del,i}) \;\wedge\; \|N_d\| \geq 2 \;\wedge\; Y_d \prec \vartheta_{delY}$$

$$K_{del,i} = \frac{\overline{\Delta w_i}}{\bar{l}} \cdot \overline{\Delta w_i^{out}} \qquad \forall i \in \boldsymbol{G}$$

$$\overline{\Delta w_i} = \frac{1}{\|N_i\|} \sum_{j \in N_i} \|w_i - w_j\|$$

$$\bar{l} = \frac{1}{n_N} \sum_{j=1}^{n_N} \overline{\Delta w_j}$$

$$\overline{\Delta w_i^{out}} = \frac{1}{\|N_i\|} \sum_{j \in N_i} \|w_i^{out} - w_j^{out}\|$$

In case of the error-driven example discussed here, determine the squared error.

$$\mathbf{E}_{task}(\boldsymbol{x}) := \mathbf{E}_{\substack{squared\\error}}(\boldsymbol{x}) = \|\zeta - \boldsymbol{o}\|^2$$

$$\eta_i^o = \begin{cases} 0 & if \quad \alpha_i^o \prec 0 \\ \eta_o & if \quad \alpha_i^o \succ 1 \\ \alpha_i^o \cdot \eta_o & else \end{cases}$$

$$\alpha_i^o = \frac{1 + B_i^L}{1 + \vartheta_L^o} + Y_i - 1 \qquad \forall i \in \boldsymbol{G}$$

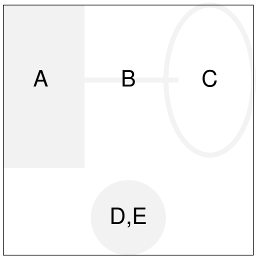
$$\Delta w_{ji} = \eta_i^o(\zeta_j - o_j)y_i \quad ; \qquad \forall j \in \{1 \ldots \boldsymbol{m}\}, \quad \forall i \in G$$

$$\tau_{(L/S)b} := e^{-\frac{1}{\mathrm{T}_{(L/S)}}} \cdot \tau_{(L/S)b} + (1 - e^{-\frac{1}{\mathrm{T}_{(L/S)}}}) \cdot \mathbf{E}_{task}(\boldsymbol{x})$$

$$Y_b := e^{-\frac{1}{T_Y}} \cdot Y_b$$

$$\tau_{\vartheta b} := \left(1 - \Lambda(\alpha_b)\right) \cdot e^{-\frac{1}{T_\vartheta}} \cdot \tau_{\vartheta b} \qquad\qquad if \quad \Lambda(\alpha_b) \succ 0$$

$$\alpha_b = \frac{1 + |B_b^L - 1|}{1 + \vartheta_L^i} - 1 \; ; \quad \Lambda(x) = \begin{cases} 0 & if & x \prec 0 \\ 1 & if & x \succ 1 \\ x & else & \end{cases}$$


A
B
C
D,E