A clustering approach to incremental learning for feedforward neural networks

Conference Paper · February 2001

DOI: 10.1109/IJCNN.2001.938474 · Source: IEEE Xplore

Andries Engelbrecht
University of Pretoria

A Clustering Approach to Incremental Learning for Feedforward Neural Networks

AP Engelbrecht and R Brits
Department of Computer Science, University of Pretoria, South Africa
engel@driesie.cs.up.ac.za

Abstract

The sensitivity analysis approach to incremental learning presented in [4] is extended in this paper. The approach in [4] selects at each subset selection interval only one new informative pattern from the candidate training set, and adds the selected pattern to the current training subset. This approach is extended with an unsupervised clustering of the candidate training set. The most informative pattern is then selected from each of the clusters. Experimental results are given to show that the clustering approach to incremental learning performs substantially better than the original approach in [4].

1 Introduction

Active learning algorithms for training multilayer feedforward neural networks allow the neural network (NN) to select itself the most informative patterns from a candidate set of training patterns. The NN uses its own knowledge about the problem to determine which patterns will have maximum gain in reducing the approximation error of that network.

Cohn, Atlas and Ladner define active learning as any form of learning in which the learning algorithm has some control over which part of the input space it receives information [2]. From this definition, two main approaches to active learning can be defined, namely selective learning [3, 5, 7, 8] and incremental learning. Selective learning selects a completely new training subset from the candidate training set at each subset selection interval, based on some measure of pattern informativeness. Each original candidate pattern is eligible for selection at each subset selection interval, regardless of whether the pattern has been selected at a previous subset selection interval. Incremental learning follows a similar approach, but with the exception that selected patterns are removed from the candidate training set, and added to the actual training set for the duration of training. The training set therefor grows during training, while the candidate training set shrinks.

This paper concentrates on incremental learning, and proposes an adaptation of an existing incremental learning algorithm to first cluster the candidate training set. The most informative pattern of each cluster is then selected and removed at each subset selection interval. The sensitivity analysis incremental learning algorithm (SAILA), introduced in [4] is used for this purpose.

Several incremental learning algorithms have been developed, differing mainly in the measure of pattern informativeness. Most current incremental learning techniques have their roots in information theory, adapting Fedorov's optimal experiment design for NN learning [1, 6, 9, 10, 12]. The different information theoretic incremental learning algorithms are very similar, and differ only in whether they consider only bias, only variance, or both bias and variance terms to quantify pattern informativeness. Zhang [13] and Röbel [11] define informativeness as a function of the prediction error associated with a pattern: the larger the prediction error, the more informative the pattern. Zhang illustrated that information gain is maximized when a pattern is selected whose addition to the training subset leads to the greatest decrease in MSE. A different measure of pattern informativeness is defined by Engelbrecht and Cloete [3, 4, 5], where the sensitivity of the output units of the network with respect to perturbations of a pattern is used as selection criterion.

The remainder of this paper is organized as follows: Section 2 provides an overview of the sensitivity analysis incremental learning algorithm introduced in [4]. The new algorithm, cluster-SAILA (CSAILA) is presented in section 3. A comparison of CSAILA and SAILA is given in section 4.

2 Sensitivity Analysis Incremental Learning

The sensitivity analysis incremental learning algorithm defines pattern informativeness as the sensitivity of the NN output to perturbations in the input values of that pattern [4]. That is,

$$\Phi^{(p)} = ||\vec{S}_o^{(p)}||_\infty = \max_{k=1,\cdots,K}\{|S_{o,k}^{(p)}|\} \tag{1}$$

where $\Phi^{(p)}$ is the informativeness of pattern p, $\vec{S}_o^{(p)}$ is the output sensitivity vector for pattern p, and $S_{o,k}^{(p)}$ refers to the sensitivity of a single output unit o_k to changes in the input vector $\vec{z}$; K is the total number of output units. The output sensitivity vector is defined as

$$\vec{S}_o^{(p)} = ||S_{oz}^{(p)}||_2 \tag{2}$$

where $S_{oz}^{(p)}$ is the output-input layer sensitivity matrix. Each element $S_{oz,ki}^{(p)}$ of the sensitivity matrix is defined as (assuming differentiable activation functions)

$$S_{oz,ki}^{(p)} = \frac{\partial o_k}{\partial z_i^{(p)}} \tag{3}$$

Each element k of $\vec{S}_o^{(p)}$ is then computed as

$$S_{o,k}^{(p)} = \sqrt{\sum_{i=1}^{I}(S_{oz,ki}^{(p)})^2} \tag{4}$$

At each subset selection interval, SAILA selects only the most informative pattern p, i.e.

$$p = \{p \in D_C|\Phi^{(p)} = \max_{q=1,\cdots,P_C}\{\Phi^{(q)}\}\} \tag{5}$$

where D_C is the current candidate training set and P_C is the number of patterns remaining in D_C. The pattern p is then removed from D_C and added to the current training subset D_T. Training continues on the training subset D_T until any one of the following subset selection criteria becomes true, upon which a new informative pattern is selected:

- The maximum number of training epochs on the current training subset has been exceeded.
- A sufficient decrease in training error has been achieved.
- The average decrease in error per epoch is too small.
- Overfitting of the current training subset has been detected.

3 Cluster-SAILA

The cluster sensitivity analysis incremental learning algorithm (CSAILA) is an extension of SAILA overviewed in the previous section. SAILA is extended to first cluster the patterns in the candidate training set. After the clustering process, training starts during which SAILA is applied to each of the clusters to find and remove the most informative pattern of each cluster. If the candidate training set is divided into C clusters, C informative patterns are selected and added to the training subset. Each selected pattern is removed from the corresponding cluster. When only one cluster is used, CSAILA and SAILA are equivalent.

The success of CSAILA lies in the number of clusters used. The cluster algorithm implemented for CSAILA dynamically grows the number of clusters based on the variance in Euclidean distance of all the patterns grouped in a cluster, from the cluster center. If the variance in Euclidean distance exceeds a user specified threshold, θ, a new cluster is added. The complete cluster algorithm is given below:

1. Initialization:
   (a) Find the minimum z_i^{min} and maximum z_i^{max} values of each input attribute z_i over the candidate training set D_C.
   (b) Initialize the cluster threshold θ and the initial number of clusters C.
   (c) Initialize the C cluster reference vectors $\vec{\rho}_1,\cdots,\vec{\rho}_C$. Each reference vector is initialized randomly such that each element $\rho_{c,i}$ of reference vector $\vec{\rho}_c$ falls within the minimum and maximum values of the corresponding input attributes. That is,

$$\rho_{c,i} \sim U(z_i^{min}, z_i^{max}) \tag{6}$$

2. For each pattern $p \in D_C$
   (a) Calculate the Euclidean distance

$$\varepsilon_c^{(p)} = \sqrt{\sum_{i=1}^{I}(z_i^{(p)} - \rho_{c,i})^2} \tag{7}$$

   to each cluster reference vector $\vec{\rho}_c$.
   (b) Find the closest cluster $\rho_c = \min_{c=1,\cdots,C}\{\varepsilon_c^{(p)}\}$ and add pattern p to cluster ρ_c
   (c) Adjust the reference vector $\vec{\rho}_c$ of cluster ρ_c:

$$\rho_{c,i}(t) = \rho_{c,i}(t-1) + \eta(z_i^{(p)} - \rho_{c,i}(t-1)) \tag{8}$$

   for all $i = i,\cdots,I$, with η is the learning rate.

3. Test for convergence

(a) Calculate the variance $\sigma^2_{\varepsilon_c}$ in Euclidean distance of each $p \in \rho_c$ for each cluster.

(b) If $\sigma^2_{\varepsilon_c} \leq \theta$ for all $c = 1, \cdots, C$, the clustering algorithm terminates.

(c) If $\sigma^2_{\varepsilon_c} > \theta$ for one of the clusters, then add an additional cluster with a randomly generated reference vector, and goto step 2.

After execution of the clustering algorithm, the incremental learning algorithm uses the SAILA approach to select at each subset selection interval the most informative pattern from each cluster.

4 Experimental Results

This section compares the performance of SAILA and CSAILA on two function approximation problems and two time series. Comparisons of SAILA with standard fixed set learning are given in [4]. The functions, defined below, are summarized in table 1 together with the NN architecture and parameters used.

- Function F1:

$$F1: \quad f(z) = \sin(2\pi z)e^{-z} + \zeta \tag{9}$$

where $z \in U(-1,1)$, and $\zeta \sim N(0,0.1)$. Output values were scaled to the range $[0,1]$.

- Function F2:

$$F2: \quad f(z_1,z_2) = \frac{1}{2}(z_1^2 + z_2^2) \tag{10}$$

where $z_1, z_2 \sim U(-1,1)$, and all outputs are in the range $[0,1]$.

- Time series TS1 (the Henon-map):

$$\begin{aligned} TS1: o_t &= z_t \\ z_t &= 1 + 0.3z_{t-2} - 1.4z_{t-1}^2 \end{aligned} \tag{11}$$

where $z_1, z_2 \sim U(-1,1)$. The output values o_t were scaled such that $o_t \in [0,1]$.

- Time series TS2:

$$\begin{aligned} TS2: o_t &= z_t \\ z_t &= 0.3z_{t-6} - 0.6z_{t-4} + 0.5z_{t-1} \\ &\quad +0.3z_{t-6}^2 - 0.2z_{t-4}^2 + \zeta_t \end{aligned} \tag{12}$$

where $z_t \sim U(-1,1)$ for $t = 1, \cdots, 10$, and $\zeta_t \sim N(0,0.05)$ is zero-mean noise sampled from a normal distribution. All output values were scaled to the range $[0,1]$.

This section compares the performance of CSAILA with that of SAILA with reference to generalization, convergence and complexity. For this purpose, 50 simulations have been executed for each problem. One simulation consists of a SAILA and CSAILA run, using the same data sets, initial weights and training parameters.

The training error and generalization performance of the training algorithms are compared in table 2. The second and third columns respectively shows the average training error $\overline{E}_T$ and average generalization error $\overline{E}_G$ over the 50 simulations (these errors are reported as the MSE over the respective data sets), together with a 95% confidence interval. Both SAILA and CSAILA consistently showed better performance than FSL. Furthermore, CSAILA performed better than SAILA for functions F1 and F2 (with substantial improvements for the latter two functions). For TS1, CSAILA and SAILA achieved the same training accuracy, but a much better generalization was achieved by CSAILA. For TS2, FSL and CSAILA had approximately the same training error, but FSL substantially overfitted with a bad generalization. SAILA performed the best for TS2. However, even though CSAILA reached a higher training error than SAILA, a generalization performance comparable to that of SAILA has been reached.

The improvements in performance by CSAILA are attributed to the larger set of informative patterns selected for training. The most informative patterns as used by CSAILA, span a wider area of input space compared to that of SAILA.

The convergence characteristics of CSAILA and SAILA are compared in figure 1, which plots the percentage of simulations that did not reach specific generalization errors. For all the problems studied in this paper, CSAILA and SAILA exhibited consistently better convergence characteristics than FSL. For all the functions, except for TS2, CSAILA had more converged simulations than SAILA. It is for low generalization levels of less than 0.004 that CSAILA improved on SAILA for TS2.

The complexity of CSAILA and SAILA is directly related to the number of patterns selected from the candidate training set. The less patterns selected, the less weight updates need to be made. Even with the little added complexity of the sensitivity analysis pattern selection approach (all information used by the selection approach is already available from the training equations), substantial reductions in training set sizes resulted in large reductions in computational complexity. Table 3 summarizes for selected epochs the percentage of the original candidate set that was used by CSAILA and SAILA for training, in comparison with the original size of the candidate training set (as given in the last column). Table 3 shows that both CSAILA and SAILA used substantially less patterns, therefor less pattern presentations. The second last column lists the percentage re-

Table 1: Summary of test functions and network parameters

Problem	**F1**	**F2**	**TS1**	**TS2**
Training/Test/Validation Set Sizes	600/200/200	600/200/200	600/200/200	180/60/60
Learning Rate	0.1	0.1	0.05	0.05
Momentum	0.9	0.9	0.9	0.9
Maximum Epochs	2000	2000	4000	1000
Architecture	1-10-1	2-5-1	2-5-1	10-10-1
Cluster Variance Threshold	0.01	0.05	0.01	0.01
Initial Clusters	3	3	3	3

Table 2: Error performance measures

Problem		$\overline{E}_T$	$\overline{E}_G$	E_G^{best}
F1:	*SAILA*	0.013 ± 0.019	0.012 ± 0.015	0.001
	CSAILA	0.0053 ± 0.0065	0.0062 ± 0.0043	0.0012
F2:	*SAILA*	0.0034 ± 0.01	0.0014 ± 0.002	0.00036
	CSAILA	0.00073 ± 0.0004	0.00073 ± 0.0004	0.00038
TS1:	*SAILA*	0.00061 ± 0.0002	0.00076 ± 0.0004	0.00019
	CSAILA	0.00068 ± 0.0002	0.00053 ± 0.0002	0.00026
TS2:	*SAILA*	0.0047 ± 0.009	0.0045 ± 0.012	0.00026
	CSAILA	0.0087 ± 0.011	0.005 ± 0.0079	0.00029

Table 3: Summary of training set sizes and total number of pattern presentations

		D_T After Epoch					**% Reduction in**	
Problem		**50**	**200**	**600**	**1000**	**2000**	**Pattern Presentations**	P_C
F1:	*SAILA*	1.1%	2.5%	7.5%	12.8%	23.8%	83.8%	600
	CSAILA	2.0%	8.5%	16.0%	21.0%	66.5%	58.7%	
F2:	*SAILA*	1.6%	5.0%	19.3%	34.5%	71.0%	53.3%	600
	CSAILA	6.7%	10.8%	21.2%	27.7%	70.8%	53.9%	
TS1:	*SAILA*	1.3%	7.3%	26.7%	45.0%	85.5%	57.3%	600
	CSAILA	7.0%	7.0%	16.7%	39.9%	69.0%	53.5%	
TS2:	*SAILA*	5.0%	16.1%	22.5%	97.2%	-	49.5%	180
	CSAILA	7.8%	25.0%	56.7%	80.0%	-	41.2%	

duction in the total number of pattern presentations (i.e. the total number of backward propagations to update weights) at the end of training, with reference to the total number of pattern presentations if training would have been on all patterns in the candidate training set. Both CSAILA and SAILA resulted in large reductions in the total number of pattern presentations. With the exception of function F1, the two incremental learning algorithms had approximately the same reductions. For F1, SAILA used much less patterns than CSAILA, but with the disadvantage of having a worse generalization performance (refer to table 2).

5 Conclusions

This paper presented an extension to the sensitivity analysis approach to incremental learning introduced in [4]. Instead of just selecting the most informative pattern from the candidate training set, the candidate set is clustered before training, and the most informative pattern is selected from each one of the clusters. In doing so, the neural network uses more information about the most informative regions of input space at each subset selection interval. The clustering approach to incremental learning (CSAILA) consistently showed better convergence results than the original incremental learning algorithm (SAILA). CSAILA also resulted in better generalization performance than SAILA.

Further research is necessary to investigate the influence of the number of clusters on performance. The more clusters there are, the more patterns are selected per subset selection interval, but the more the increase in computational complexity. Also, too many clusters approximate the performance of standard fixed set learning (i.e. training on all the patterns), while too few patterns approximate the performance of SAILA.

References

[1] DA Cohn, “Neural Network Exploration using Optimal Experiment Design”, AI Memo No 1491, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1994.

[2] D Cohn, L Atlas, R Ladner, “Improving Generalization with Active Learning”, Machine Learning, Vol. 15, pp 201-221, 1994.

[3] AP Engelbrecht, I Cloete, “Selective Learning using Sensitivity Analysis”, IEEE World Congress on Computational Intelligence, International Joint Conference on Neural Networks, Anchorage, Alaska, pp 1150-1155, 1998.

[4] AP Engelbrecht, I Cloete, “Incremental Learning using Sensitivity Analysis”, IEEE International Joint Conference on Neural Networks, Washington DC, USA, 1999, paper 380.

[5] AP Engelbrecht, *Sensitivity Analysis for Selective Learning by Feedforward Neural Networks*, accepted for Fundamenta Informaticae, IOS Press, 2001

[6] K Fukumizu, *Active Learning in Multilayer Perceptrons*, DS Touretzky, MC Mozer, ME Hasselmo (eds), Advances in Neural Information Processing Systems, Vol 8, 1996, pp 295-301.

[7] JB Hampshire, AH Waibel, “A Novel Objective Function for Improved Phoneme Recognition using Time-Delay Neural Networks”, IEEE Transactions on Neural Networks, 1(2), 1990, pp 216-228.

[8] SD Hunt, JR Deller (jr), “Selective Training of Feedforward Artificial Neural Networks using Matrix Perturbation Theory”, Neural Networks, 8(6), 1995, pp 931-944.

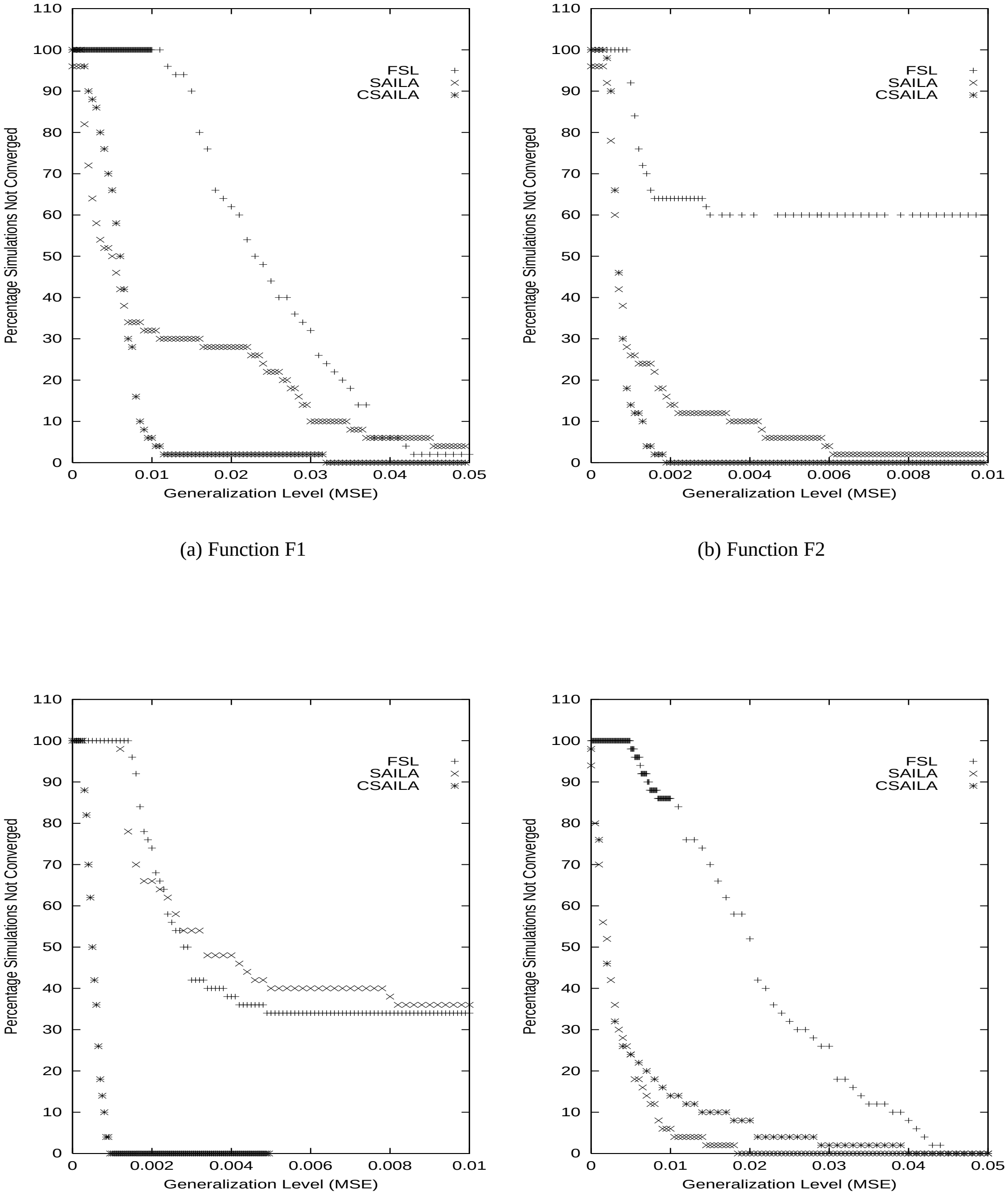
[9] DJC MacKay, *Bayesian Methods for Adaptive Models*, PhD Thesis, California Institute of Technology, 1992.

[10] M Plutowski, H White, *Selecting Concise Training Sets from Clean Data*, IEEE Transactions on Neural Networks, 4(2), March 1993, pp 305-318.

[11] A Röbel, *Dynamic Pattern Selection: Effectively Training Backpropagation Neural Networks*, International Conference on Artificial Neural Networks, Vol 1, 1994, pp 643-646.

[12] KK Sung, P Niyogi, “A Formulation for Active Learning with Applications to Object Detection”, AI Memo No 1438, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1996.

[13] B-T Zhang, “Accelerated Learning by Active Example Selection”, International Journal of Neural Systems, 5(1), March 1994, pp 67-75.



(a) Function F1

(b) Function F2

(c) Time Series TS1

(d) Time Series TS2

Figure 1: Percentage simulations that did not converge to generalization levels