ESANN'1996 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-25-26 April 1996, D-Facto public., ISBN 2-9600049-6-5, pp. 123-128

Towards Constructive and Destructive Dynamic Network Configuration

Stefan Wermter, Manuela Meurer

Department of Computer Science, University of Hamburg
22765 Hamburg, Federal Republic of Germany

Abstract. In this paper we describe a combined constructive/destructive approach for dynamic network configuration. Different from previous approaches we address dynamic network configuration for recurrent networks and for a real-world natural language task of semantic categorization. We show that dynamic network configuration can lead to architectures which are smaller and faster than static networks and which perform slightly better on generalization than static networks.

1. Introduction

Most connectionist - and also symbolic - architectures use a predetermined architecture based on the known and experienced properties of the task. For instance, for supervised connectionist learning there are not yet generally applicable rules for finding the necessary number of hidden units for arbitrary tasks. Theoretical results have provided important proofs about the learnability of functions within certain architectures [5] but unfortunately such results often contain assumptions which are worst case upper bounds (like exponential number of hidden units) and therefore cannot be used directly for a particular task. Because of this lack of theoretical guidelines, typically different architecture variations are tested in order to determine a good representative architecture. However, especially for larger real-world data sets, training connectionist architectures can be very time-consuming. Therefore, given the lack of current general theoretical guidelines for network configuration, it would be desirable to design network configurations dynamically during task learning.

In the past several approaches for dynamic network configuration have been examined, in particular for feedforward connectionist networks and for small tasks. *Constructive approaches* start with a small number of hidden units and add more units as long as training does not fulfill a desired performance criterion (e.g. [1, 6]). *Destructive approaches* start with a number of hidden units which is supposed to be too large for the given task and prune units as long as the network performance does not deteriorate under a specified performance

This research was supported in part by the Federal Secretary for Research and Technology under contract #01IV101A0.

ESANN'1996 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-25-26 April 1996, D-Facto public., ISBN 2-9600049-6-5, pp. 123-128

criterion (e.g. [8, 7, 2]). *Combined approaches* switch between constructive and destructive phases (e.g., [4]). In general, starting with a constructive approach is more efficient since the network starts with a small number of units. On the other hand, it might be useful to have a destructive phase after a constructive phase since finding a good architecture might need more units than performing the task itself. Therefore, it seems to be a promising dynamic strategy to start with a constructive phase followed by a destructive phase.

In spite of these approaches the experience with dynamic network configuration has been rather limited compared with the number of static network architectures. All work cited above concentrates on small problems (like learning x-or, encoders, etc.) using only feedforward architectures. In this paper we describe a combined constructive/destructive approach for recurrent network architectures using a real-world size data set from semantic categorization.

2. Dynamic Network Configuration

In order to *modify the architecture during learning* we need a measurement of the current performance. Since the total sum squared error (also called global error) is often used for estimating the overall error we also use the dynamics of this global error (the learning curve) for possible dynamic changes of the network architecture. If the learning curve shows a steep descent the architecture learns well, if the learning curve shows a flat asymptotic form the architecture does not learn well. That is, the gradient is taken as an indicator for the possible dynamic changes of the network architecture. We will now describe the overall algorithm in more detail.

2.1. Preprocessing: Computing Sliding Averages

In feedforward networks typical learning curves show a monotonically descending shape. In this case the gradient of the global error can be used for determining zones of slow learning. However, we would like to examine dynamic network configuration for *recurrent* networks since only recurrent networks can represent potentially unrestricted sequence patterns. Unfortunately, the introduction of recurrent connections within a connectionist architecture causes a deviation from the true gradient descent, and learning curves do not have to be monotonically descending anymore. Therefore, instead of using the direct value of the current global error the sliding average error E_{sa} is computed over the last n values of the global errors $E(t-1)...E(t-n)$. In general this procedure leads to a smoothing of the learning curve while keeping the main shape of the learning curve.

$$E_{sa}(t) = \frac{E(t-1) + ... + E(t-n)}{n}$$

All phases of the algorithm described below will use this smoothed learning curve.

ESANN'1996 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-25-26 April 1996, D-Facto public., ISBN 2-9600049-6-5, pp. 123-128

2.2. An Overview about the 3-Phase Dynamic Configuration Algorithm

Before we will explain and motivate the algorithm in detail we give an overview about the dynamic network configuration. The algorithm can be summarized as follows:

```
%Constructive learning phase
WHILE learning slope decreasing DO
        IF learning slope only slightly decreasing and last
           addition long ago
        THEN add hidden unit and learn further
        ELSE learn further

%Destructive learning phase
WHILE NOT learning curve increasing DO
        IF  last removal long ago
        THEN remove hidden unit and learn further
        ELSE learn further

%Post-learning phase
WHILE epoch limit not yet reached DO
        learn further
```

The construction phase starts with a minimal number of hidden units and adds more units until the performance cannot be improved anymore. Then, in the destructive phase hidden units are eliminated from the hidden layer. Finally, there is a post-learning phase for optimizing the found configuration.

2.3. Constructive Learning

Since it is more efficient to grow small networks than to shrink large networks we first start with an architecture which does not have a sufficiently large number of hidden units. In all our experiments we start with one hidden unit. Then, learning proceeds until the descent of the smoothed learning curve is too small to expect further performance improvement. At that point an additional hidden unit is added. The descent is measured using the global smoothed error E_{sa} at the start and end of a window w. If the absolute difference between the error $E_{sa}(t)$ at time t and the error $E_{sa}(t-w)$ at time t-w divided by the error at the last unit addition $E_{sa}(t_{add})$ gets smaller than a certain value Δ_{adding} a new unit will be added. The value of the *learningslope* will be high and units will not yet be added if the difference between $E_{sa}(t)$ and $E_{sa}(t-w)$ is large (that is if learning proceeds successfully) and if $E_{sa}(t_{add})$ is relatively small (that is a bias for longer training with increasing time since $E_{sa}(t_{add})$ will get smaller over time with increasing additions and generally decreasing learning curves). Below we show this addition condition:

ESANN'1996 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-25-26 April 1996, D-Facto public., ISBN 2-9600049-6-5, pp. 123-128

$$Learning\ slope = \frac{|E_{sa}(t) - E_{sa}(t-w)|}{E_{sa}(t_{add})} < \Delta_{adding}$$

In order to prevent networks from growing too fast, additions will only be allowed after at least w epochs through the training set.

2.4. Destructive Learning

After learning has been started the learning slope will be rather steep and decreasing. Each time when learning slows down ($learningslope < \Delta_{adding}$) after a number of learning epochs (time $t > t_{add} + w$) a new hidden unit is added. Then the learning slope typically increases briefly due to the new unit and finally decreases again. After a number of constructive additions of hidden units the learning slope will not decrease significantly anymore. This is taken as an indicator that the constructed network will have learned the task as good as possible for the current configuration. If the learning slope falls under a value $\Delta_{removing}$ where $\Delta_{removing}$ is smaller than Δ_{adding} the constructive phase is terminated and the destructive learning phase begins.

During this destructive learning phase the network is sequentially pruned from previously added hidden units. The underlying motivation is that for learning the task and for determining an architecture more hidden units may have been added than required for learning the task. Therefore a hidden unit is removed as long as the current global error E_{sa} is smaller than the error at the time when the last preceding unit was removed. Below we show this removing condition:

$$Learning\ slope = \frac{|E_{sa}(t) - E_{sa}(t-w)|}{E_{sa}(t_{add})} < \Delta_{removing} < \Delta_{adding}$$

In order to prevent networks from shrinking too fast, unit elimination will only be allowed after at least w epochs through the training set, similar as in the constructive phase.

2.5. Post-learning

The destructive phase is terminated as soon as the global error E_{sa} of the current epoch is bigger than the error at the removal of the last hidden unit. In that case only the last removed unit led to a performance decrease. Therefore this single unit is added again. After the dynamic construction and destruction of the network we let the network settle and optimize for the current found network configuration.

3. Semantic Context Classification

The dynamic network configuration above was tested with a semantic context classification task. This task is based on a title corpus of several thousand

ESANN'1996 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-25-26 April 1996, D-Facto public., ISBN 2-9600049-6-5, pp. 123-128

phrases described in detail in [9]. Here we just use part of this corpus as a well-suited testbed for dynamic network configuration. 795 phrases belonged to 10 different semantic classes: theology, history, law, mathematics, chemistry, geology, electrical engineering, computer science, art, and music. The task was to learn the training phrase classification and to generalize to new test titles. 277 of the 795 phrases were used for training, the rest was kept for testing generalization.

Each word was represented with a 10-unit plausibility vector which represented the normalized frequencies with which this word occured in the semantic classes in the corpus. Each class was represented with a 10 unit vector with the single desired class unit represented by a 1 and all other units by a 0. Since the phrases can be of arbitrary length we used simple recurrent networks [3]. The input to this network is a sequence of words, one at a time represented by 10 units, the output is a corresponding sequence of one class, one at a time represented by 10 units.

As a benchmark comparison, *static* simple recurrent networks were tested with different architectures. The learning rates used were 10^{-2}, 10^{-3}, 10^{-4}, the number of hidden units was fixed to 4, 8, 12, and 16. For each of these combinations three training runs were performed with 4000 cycles. The best results were found for a learning rate of 10^{-3} and 12 hidden units. For this configuration the best network had a training and test set performance of 85% and 79% respectively (see table 1).

Network	Training	Test	Hidden Units	Time
Static	85%	79%	12	299:17
Dynamic	82%	81%	9	257:07

Table 1: Best training and generalization performance. Time is shown in minutes and seconds

For the dynamic networks we performed the same kind of experiments as for the static networks using the same learning rate of 10^{-3}.[1] The best results were 82% on the training set and 81% on the test set. The improvement on the test set is much more important than the deterioration on the training set since we are primarily interested in generalizing the learned behavior to new examples. Besides the slight improvement on the test set, we can see that the dynamically determined network used less memory and time to come to these results. Rather than 12 as in the best static network, the best dynamic network used only 9 hidden units. Furthermore, training in the dynamic network (257:07 minutes) was faster than in the static network (299:17 minutes).

[1]The learning slope for the constructive phase was 0.05, for the destructive phase 0.01.

ESANN'1996 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-25-26 April 1996, D-Facto public., ISBN 2-9600049-6-5, pp. 123-128

4. Conclusions

In general, we could observe slight generalization, memory and time improvements for the best dynamic recurrent networks compared with static networks. In contrast to previous work we have concentrated on dynamics in recurrent networks and we have applied this technique to a real-world problem of categorizing titles according to their semantic classes. The improvements found are particularly significant, if we consider that a dynamic network will determine the architecture automatically while learning the task. Furthermore, many static networks have to be trained for testing different architectures and we cannot be sure whether a certain number of hidden units is appropriate. In contrast, dynamic networks learn this task while determining the architecture and have the potential to arrive at smaller, faster and better performing networks.

References

[1] T. Ash. Dynamic node creation in backpropagation networks. Technical Report ICS Report 8901, University of California, San Diego, 1989.

[2] A. N. Burkitt. Optimization of the architecture of feed-forward neural networks with hidden layers by unit elimination. *Complex Systems*, 5:371–380, 1991.

[3] J. L. Elman. Finding structure in time. *Cognitive Science*, 14:179–221, 1990.

[4] Y. Hirose, K. Yamashita, and S. Hijiya. Back-propagation algorithm which varies the number of hidden units. *Neural Networks*, 4:61–66, 1991.

[5] K. Hornik, W. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[6] N. Indurkhya and S. M. Weiss. Heuristic configuration of hidden units for neural network classifiers. Technical Report TR-279, Department of Computer Science, Rutgers University, 1991.

[7] M. C. Mozer and P. Smolensky. Skeletonization: A technique for trimming the fat from a network via relevance assessment. Technical Report Tech. Report CU-CS-421-89, University of Colorado, Boulder, 1989.

[8] M. C. Mozer and P. Smolensky. Using relevance to reduce network size automatically. *Connection Science*, 1(1):3–16, 1989.

[9] S. Wermter. *Hybrid Connectionist Natural Language Processing.* Chapman and Hall, London, UK, January, 1995.