Intelligible Neural Networks with BP-SOM

Ton Weijters, Antal van den Bosch, H. Jaap van den Herik

Department of Computer Science
Universiteit Maastricht
P.O. Box 616, NL-6200 MD Maastricht
The Netherlands
{weijters, antal, herik}@cs.unimaas.nl

Abstract

Back-propagation learning (BP) is known for its serious limitations in generalising knowledge from certain types of training material. BP-SOM is an extension of BP which overcomes some of these limitations. BP-SOM is a combination of a multi-layered feed-forward network (MFN) trained with BP, and Kohonen’s self-organising maps (SOMs). In earlier reports, it has been shown that BP-SOM improved the generalisation performance whereas it decreased simultaneously the number of necessary hidden units without loss of generalisation performance. These are only two effects of the use of SOM learning during training of MFNs. In this paper we focus on two additional effects. First, we show that after BP-SOM training, activations of hidden units of MFNs tend to oscillate among a limited number of discrete values. Second, we identify SOM elements as adequate organisers of instances of the task at hand. We visualise both effects, and argue that they lead to intelligible neural networks and can be employed as a basis for automatic rule extraction.

1 Background

Back-propagation learning [RHW86] (henceforth referred to as BP) is known for its serious limitations in (i) generalising knowledge from certain types of training material [Nor89, WK91] and (ii) inducing intelligible models [Nor89, SL97],

In earlier publications [Wei95, Wei96, WVVP97] experimental results are reported in which the generalisation performances of BP-SOM are compared to two other learning algorithms for MFNs, viz. BP, and BP augmented with weight decay [Hin86]. Henceforth, the latter is referred to as BPWD. The reported experiments show that (i) BP-SOM learning results in MFNs with a better generalisation performance, as compared to MFNs trained with BP and BPWD, and (ii) an increased amount of hidden units can be pruned without loss of generalisation performance. These are only two effects of the use of SOM learning when training BP-SOM networks. In this paper, we investigate two additional effects: (i) hidden-unit activations
tend to end up oscillating between a limited number of discrete values, and (ii) the SOM can be seen as an organiser of the instances of the task at hand, dividing them into a limited number of subsets. Furthermore, we visualise these effects, and show that they lead to better intelligibility of trained MFNs. Moreover, the division of the training and testing material into homogeneous subsets by SOM elements is shown to be a useful step in automatic rule extraction.

Section 2 summarises the BP-SOM architecture and learning algorithm. In Section 3, we present and visualise experimental observations of the BP-SOM architecture and learning algorithm. In section 4, we provide our conclusions.

2 BP-SOM

Below we give a brief characterisation of the functioning of BP-SOM. For details we refer to [Wei95,Wei96,WVVP97]. The aim of the BP-SOM learning algorithm is to establish a cooperation between BP-learning and SOM-learning in order to find adequate hidden-layer representations for learning classification tasks. To achieve this aim, the traditional MFN architecture [RHW86] is combined with self-organising maps (SOMs) [Koh89]: each hidden layer of the MFN is associated with one SOM (see Figure 1). During training of the weights in the MFN, the corresponding SOM is trained on the hidden-unit activation patterns.

After a number of training cycles of BP-SOM learning, each SOM, to a certain extent, develops self-organisation, and translates this self-organisation into classification information; i.e., each SOM element is provided with a class label (one of the output classes of the task). For example, let the BP-SOM network displayed in Figure 1 be trained on a classification task which maps instances to either output class A or B. If a SOM element is the best matching element to 4 hidden-unit activation patterns that map to class A, and to 2 hidden-unit activation patterns that map to class B, the class label of that SOM becomes ‘A’, with a reliability of $4/6 = 0.67$. As a result, we can visually distinguish areas in the SOM: areas containing elements labelled with class A and class B, and areas containing unlabelled elements (no winning class could be found). In Figure 1, we see four class labels A, four
The self-organisation of the SOM is used as an addition to the standard BP learning rule [RHW86]. Classification and reliability information from the SOMs is included when updating the connection weights of the MFN. The error of a hidden-layer vector is a accumulation of the error computed by the BP learning rule, and a SOM error. The SOM error is the difference between the hidden-unit activation vector and the vector of its best-matching element associated with the same class on the SOM (for more details, cf. [Wei95,Wei96]). The effect of including SOM information in the error signals is that clusters of hidden-unit activation patterns of instances associated with the same class tend to become increasingly similar to each other.

As soon as BP-SOM learning has finished, the SOMs are redundant, since they do not contribute to the activation feed-forward within the BP-SOM network. Therefore all SOMs are removed from a trained BP-SOM network, resulting in a stand-alone MFN.

3 Visualising the effects: experiments

In this section, we present two experimental observations with the BP-SOM architecture and the BP-SOM learning algorithm. The observations are made by training BP-SOM on two benchmark classification tasks, viz. the parity-12 task, and the monks-1 task. The parity-12 task is hard to learn for many learning algorithms, since it often leads to overfitting [Sch93,Tho95]. The reason is that the output classification depends on the values of all input features. The monks-1 is a well-known benchmark task for automatic rule extraction [Thr91].

For all experiments reported in this paper we have used a fixed set of parameters for the learning algorithms. The BP learning rate was set to 0.15 and the momentum to 0.4. In all SOMs a decreasing interaction strength from 0.15 to 0.05, and a decreasing neighbourhood-updating context from a square with maximally 25 units to only 1 unit (the winner) was used [Koh89]. BP-SOM was trained for a fixed number of cycles \( m = 2000 \); class labelling was performed at each 5th cycle \( n = 5 \). Early stopping, a common method to prevent overfitting, was used in all experiments: the performance of a trained network was calculated in percentages of incorrectly-processed test instances at the cycle where the classification error on validation material was minimal [Pre94]. For further details on the experiments, we refer to [Wei96].

3.1 Parity-12

BP, BPWD, and BP-SOM have been applied to the parity-12 task, i.e., the task to determine whether a bit string of 0’s and 1’s of length 12 contains an even number of 1’s. The training set contained 1,000 different instances selected at random out of the set of 4,096 possible bit strings. The test set and the validation set contained 100 new instances each. The hidden layer of the MFN in all three algorithms contained 20 hidden units, and the SOM in BP-SOM contained \( 7 \times 7 \) elements. The algorithms were run with 10 different random weight
Table 1: Average generalisation performances of BP, BPWD, and BP-SOM, on the parity-12 task (plus standard deviation, after ‘±’; averaged over ten experiments) in terms of incorrectly-processed training and test instances.

<table>
<thead>
<tr>
<th>Task</th>
<th>BP: % incorrect</th>
<th>BPWD: % incorrect</th>
<th>BP-SOM: % incorrect</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Train</td>
<td>Test</td>
<td>Train</td>
</tr>
<tr>
<td>parity-12</td>
<td>14.1 ± 18.8</td>
<td>27.4 ± 16.4</td>
<td>21.6 ± 24.2</td>
</tr>
</tbody>
</table>

Figure 2: Graphic representation of a 7 × 7 SOM: associated with a BP-trained MFN (left), with a BPWD-trained MFN (middle), and with a BP-SOM network (right); all are trained on the parity-12 task. White squares represent class ‘even’; black squares represent class ‘odd’. The width of a square represents the reliability of the element; a square of maximal size represents a reliability of 100%.

Table 1 displays the classification errors on training instances and test instances. The results indicate that BP-SOM performed significantly better than BP and BPWD on test material (with p-values $p < 0.01$ and $p < 0.05$, respectively), and that BP-SOM was able to avoid overfitting better than BP. Although BPWD was also able to avoid overfitting, it was at the cost of a low performance on both training and test material.

To visualise the differences among the representations developed at the hidden layers of the MFNs trained with BP, BPWD, and BP-SOM, respectively, we also trained SOMs with the hidden-layer activities of the trained BP and BPWD networks. Figure 2 visualises the class labelling of the SOMs: the left part is an MFN after BP-training; the middle part visualises the SOM after a BPWD-training of an MFN, and the right part displays the SOM of the BP-SOM network after training on the same material. The SOM of the BP-SOM network is much more organised and clustered than that of the SOMs corresponding with the BP-trained and BPWD-trained MFNs. The reliability values of the elements of all three SOMs are represented by the width of the black and white squares. It can be seen that the overall reliability of the
Figure 3: Standard deviations of the activations of the 20 hidden units of a BP-trained MFN (left), a BPWD-trained MFN (middle), and a BP-SOM network (right), trained on the parity-12 task (1,000 instances).

SOM of the BP-SOM network is considerably higher than that of the SOM of the BP-trained and BPWD-trained MFNs.

**Simplified hidden-unit activations**

By including the SOM in BP-SOM learning, clusters of hidden-unit activation patterns associated with the same class tend to become more similar [Wei96]. When analysing the hidden-unit activations in BP-SOM networks, we observed a valuable additional effect, viz. that hidden-unit activations culminated in a stable activity with a very low variance or that they resulted in oscillating between a limited number of discrete values. This clearly contrasts with hidden units in MFNs trained with BP, of which all activations usually display high variance.

Figure 3 displays the standard deviation of the 20 hidden-unit activations of an MFN trained with BP (left), and MFN trained with BPWD (middle) and a BP-SOM network (right), each of them trained on the parity-12 task (1,000 instances). The standard deviations are computed after presentation of all training instances without further training. As can be seen from Figure 3, the standard deviations of ten out of twenty units in the BP-SOM network are equal to 0.01 or lower.

Whenever a unit has a stable activation with a low standard deviation for all training instances, it is redundant in the input-output mapping. In that case, the unit can be pruned from the network, and its effect on the following layer (i.e., its mean activation times its weights to units in the following layer) can be included into the weights from the bias unit associated with the same layer, to units of the following layer. This pruning can be performed both during and after training.

We trained BP, BPWD, and BP-SOM on the parity-12 task, in an attempt to prune hidden units according to a threshold criterion during training. We introduced a stability threshold parameter $s$, denoting the standard deviation of the unit’s activation below which it is pruned. After a number of pilot experiments with different values for $s$, we performed experiments with $s = 0.01$. All three algorithms were trained on the parity-12 task with ten different random initialisations. We found that BP-SOM was able to prune 12 out of 20 hidden units.
(averaged over 10 experiments), without loss of generalisation performance. With the same setting of \( s \), trained on the same tasks, no hidden units could be pruned with BP, nor with BPWD. Only with \( s = 0.1 \) hidden units could be pruned during BP and BPWD learning; however, this led to seriously worse generalisation performance of these networks.

Since BP-SOM learning results in a trained MFN, the increased ability to prune hidden units, as illustrated with the parity-12 task, is an advantage over fixed-size networks. The latter may be chosen as small as the MFN end product, but the risk exists that a learning algorithm, such as BP, is faced with too few degrees of freedom to learn the task. We performed a series of ten experiments with BP, using an MFN with one hidden layer of eight units, and obtained an average classification error on test material of 36.5% (± 12.6): setting the number of hidden units explicitly to eight did not help BP in learning the task.

To illustrate the second effect, viz. the oscillating of hidden-unit activations between a limited number of discrete values, one experiment with an MFN trained with BP-SOM on the parity-12 task was chosen. In this experiment, an extremely large number of hidden units was pruned (16 out of 20 hidden units), while the performance of the trained MFN on test material was still acceptable (classification error 4%). Figure 4 displays the activations of the four hidden units of the BP-SOM-trained MFN (displayed on the y-axis), measured for each of the 100 test instances (displayed on the x-axis). The test instances are grouped on the basis of their respective SOM classification: we collected for each labelled SOM element all associated test instances.

It can be seen from Figure 4 that the four hidden units end up oscillating between a discrete
number of values, depending on the SOM element to which a test instance belongs. This effect is witnessed in all BP-SOM-trained MFNs. The less hidden units are pruned, the less discrete activations are used by the remaining hidden units. The apparent discretisation of activations offers a potentially interesting basis for rule extraction from MFNs: “In order to extract rules from (MFNs), it is better that (hidden-unit) activations be grouped into a small number of clusters while at the same time preserving the accuracy of the network” [SL97]. Current approaches to rule extraction from MFNs have to rely on a procedure which can only be applied after having trained the MFN [SL97]. In contrast, BP-SOM yields the discretisation automatically during learning; no additional discretisation method need to be performed. Future work should elaborate on exploiting the discretisation of hidden-unit activations.

3.2 SOM-based rule extraction

In our experiments, we observed that the reliability of SOM elements of successfully trained BP-SOM networks is often very high. This implies that the SOM can be seen as an organiser of the learning material. It divides the material into a limited number of subsets (bounded by the number of SOM elements). Within subsets, all instances are associated with the same class assuming a reliability of the SOM element of 1.0. This automatic division into homogeneous subsets can be a useful step in automatic rule extraction.

As an example, we trained BP-SOM networks on the monks-1 task [Pre94], a well-known benchmark problem for rule extraction. The task is to classify an instance \((a1, a2, a3, a4, a5, a6)\) with six attributes. The possible values of the attributes are: \(a1 \in \{1, 2, 3\}\), \(a2 \in \{1, 2, 3\}\), \(a3 \in \{1, 2\}\), \(a4 \in \{1, 2, 3, 4\}\), \(a5 \in \{1, 2, 3, 4\}\), \(a6 \in \{1, 2\}\). An instance is mapped to a class ‘1’ when \((a1 \neq a2)\) or \((a5 = 1)\), and to class ‘0’ otherwise. The task is considered to be easily learnable. We used MFNs with one hidden layer of 10 units, \(5 \times 5\)-sized SOMs, and the same experimental settings as described earlier. After training (at which we obtained a generalisation accuracy of 100% on the test set), we collected for each labelled SOM element all associated training instances. Table 2 lists all the instances associated with the SOM element at SOM coordinates \((1,1)\) (see also Figure 5).

The subset of instances associated with SOM element \((1,1)\) are all classified as ‘1’. Moreover, two attributes display a constant value in the subset, viz. attributes \(a1\) (having value 1) and \(a2\) (having value 1); all other attributes display varying values. This regularity can be exploited to form a rule, which states that if \(a1 = 1\) and \(a2 = 1\), the corresponding class is ‘1’. This rule extraction procedure is formalised as follows. For each SOM element, an \(IF . . . THEN\) rule is extracted: it is composed of a conjunction of all attribute values, having a constant value throughout the instance subset associated (they are concatenated on the left-hand side of the rule), and of the classification of the instances (on the right-hand side). This procedure, when applied to SOM element \((1,1)\), results in the rule \(IF (a1 = 1) \& (a2 = 1) THEN class= 1\). Application of this procedure to all labelled SOM elements results in the rules visualised in Figure 5.

We find that class ‘1’ is covered by seven SOM elements, and that class ‘0’ is covered by four SOM elements. Although none of the SOM elements fully represent the underlying concept of the monks-1 problem, each extracted rule covers a part of the problem by focusing only on the relevant attributes \(a1, a2, \) or \(a5\). The simple rule extraction procedure given
above cannot produce rules expressing the dependency of attribute values (e.g., \( a1 = a2 \)), disjunctions, or nested conditions, nor does it express weighting and ordering of rules. However, we stress the fact that the clustering of subsets of instances on the SOM leads to intelligible trained MFNs and provides an important first step in rule extraction.

4 Conclusions

In this paper we focused on two effects displayed by BP-SOM learning. First, we showed that after BP-SOM training, activations of hidden units of MFNs tend to oscillate among a limited number of discrete values. The effect is illustrated on the parity-12 task. The discretisation ability makes BP-SOM-trained MFNs suited for rule extraction, without having to resort to
post-processing rule-extraction methods.

Second, we identified SOM elements as adequate organisers of instances of the task at hand. This effect is visualised by inspecting the organisation of instances in subsets on the elements of the SOM. When trained on the monks-1 task, the SOM of BP-SOM can be exploited straightforwardly as a first step in automatic rule extraction.

In sum, both effects lead to intelligible neural networks and can be employed as a basis for automatic rule extraction.

From the reported results and previous reports on BP-SOM we conclude that BP-SOM addresses essential issues in neural-network research. By letting supervised and unsupervised learning cooperate, BP-SOM aims at finding a combined solution to problems of overfitting, simplifying MFNs (either by pruning or by simplifying hidden-unit representations), and automatic rule extraction.

**Acknowledgement**

We wish to thank Eric Postma for stimulating discussions that provided part of the motivation for this work.
References


