The Neural Bucket Brigade

In R. Pfeifer, Z. Schreter, Z. Fogelman, and L. Steels, editors, Connectionism in Perspective, pages

439-446. Amsterdam: Elsevier, North-Holland, 1989.

Jürgen Schmidhuber*
Institut für Informatik
Technische Universität München
Arcisstr. 21, 8000 München 2, Germany
schmidhu@tumult.informatik.tu-muenchen.de

Abstract

Although there are straightforward methods for training neural nets to show certain types of sequential behavior, the general problem of structural and temporal credit assignment (appropriate weight changes) in networks existing in a changing environment remains unsolved. In this paper we argue that a learning mechanism for finding temporal input/output relations ought to depend solely on computations local in both space and time, and that no teacher should be required to indicate the starts and ends of relevant sequences to the network. We ask whether there are learning rules which do not depend on such external hints yet still can deal with 'hidden units' and with units whose past activations are 'hidden in time'. (Back propagation, for instance, is not even local in time.) We propose a discrete time version and a continuous time version of a simple on-line method local in space and time which is designed to deal with hidden states. The approach is inspired by Holland's idea of the bucket brigade for classifier systems, which is transformed to run on a neural network with fixed topology. The result is a recurrent 'neural' dissipative system which is consuming 'weight-substance' and permanently trying to distribute this substance onto its connections in an appropriate way. Simple experiments demonstrating the feasability of the algorithm are reported.

Keywords: recurrent networks, credit assignment, local computations, dissipative systems, autonomous agents, internal feedback, external feedback, reinforcement learning, supervised learning, neural bucket brigade.

1 Motivation

Many researchers in the field of neural network research are concerned with the study of static input/output mappings, emphasizing the parallel aspects of such networks. Standard methods for supervised learning of such mappings are Werbos' back propagation (BP)(Werbos, 1974) and Almeida's extension (Almeida, 1987) for networks with arbitrary feedback, where the activations flowing through the network have to reach a state of equilibrium. However, if we want a net to learn sequences of time-dependent outputs then an equilibrium is something we usually wish to avoid. Rumelhart, Hinton and Williams show how to correctly use BP

*Research supported by a scholarship of SIEMENS AG

for learning time-dependencies by utilizing Minsky and Papert's idea of unfolding a recurrent net into a feedforward net (Rumelhart et al., 1986). We will point out reasons why such an approach cannot give a satisfactory solution for all problems involved with time, and we want to present some constructive criticism.

A general setting for dynamic actions depending on dynamic inputs is given in the case of an autonomous agent existing in a changing environment. The agent, which shall be regarded preliminarily as a black box, is continuously receiving inputs from the environment by means of simple feature detectors. The agent is able to produce outputs (actions) by means of effectors which may influence the state of the environment. Since useful actions in general depend on previous temporal context the agent must be able to let future system states depend on the current state (i.e. short time memory storage in activations wandering around feedback loops). The agent is expected to learn succesful temporal behavior depending on time varying perceptions from the environment.

We can identify learning situations with different degrees of supervision: In a highly unsupervised situation only a global reinforcement signal may be available to the system at isolated points in time. In a highly supervised situation any part of the system is instructed by the environment at any time. In between these extremes there is a continuum of possibilities. Recurrent BP, for instance, requires teaching signals just for a subset of all units.

The system ought to find relevant structures in spatial/temporal input patterns by means of a memory that stores just the significant aspects of some particular input. Here 'relevance' and 'significance' is implicitly defined by the environment which rewards certain actions and punishes others. However, the system generally does not know in advance which spatial/temporal input patterns are relevant for fulfilling the tasks. It generally does not know how much time some particular relevant input sequence takes. It generally does not even know the beginning of such a sequence.

If the inside of the black box was a recurrent BP network the teacher would have to provide such information *a priori*: The teacher would have to tell the network when to start spreading activation, and he would have to tell it when to stop and start back-propagaton and weight changing. We identify one of the main problems of such BP networks in not being 'really' local. Although (Rumelhart et al., 1986) say that BP is a 'local' algorithm in so far as all relevant information needed to change some weight (error signal and activation signal) becomes available close to the weight (in a physical sense), BP is only what we informally call 'local in space'. It is not 'local in time', since in general the weight-changing pieces of information appear at very different, externally controlled points in time. [1]

If the inside of the black box was a network taught by an extended REINFORCE algorithm (Williams, 1988) then there would be no need of extensive book-keeping of past activations, as with BP. However, even in this case starts and ends of sequences would have to be indicated by an external teacher. Weights would not change permanently, but only after accumulation of activation information for a predefined number of time steps.

Correlation based learning rules like Hebb's rule, or e.g. Grossberg's and Kohonen's modifications, are local in both space and time and are not concerned with starts and ends of sequences. Such rules make external control of the global system behavior unnecessary. However, these methods seem to lack the ability to deal with hidden units and with unit activations hidden in time. We now ask whether there are methods local in space and time

[1] Recently Williams and Zipser have introduced a procedure for 'supervised' learning which is local in time but not in space (Williams and Zipser, 1988).

that require a minimum of external teaching information (in the sense described above) and *still* allow credit assignment to hidden units and to units whose past activations are hidden in time.

2 Classifier Systems and the Bucket Brigade.

Holland (Holland, 1985) has proposed the meanwhile well-known bucket brigade algorithm for classifier systems, which in principle incorporates the desirable properties mentioned above. In this section we shortly review the main idea of this algorithm.

Messages in form of bitstrings of size n can be placed on a global message list either by the environment or by entities called classifiers. Each classifier consists of a condition part and an action part defining a message it might send to the message list. Both parts are strings out of $\{0, 1, \_\}^n$ where the '_' serves as a 'don't care' if it appears in the condition part. (Less important for our purposes, the '_' serves as a 'pass-through' if it appears in the action part.) A non-negative real number is associated with each classifier indicating its 'strength'.

During one cycle all messages on the message list are compared with the condition parts of all classifiers of the system. Each matching classifier computes a 'bid' by multiplying its specificity (the number of non-don't cares in its condition part) with the product of its strength and a small factor. The highest bidding classifiers may place their message on the message list of the next cycle, but they have to pay with their bid which is distributed among the classifiers active during the last time step which set up the triggering conditions (this explains the name bucket brigade).

Certain messages result in an action within the environment (like moving a robot one step). Because some of these actions may be regarded as 'useful' by an external critic who can give payoff by increasing the strengths of the currently active classifiers, learning may take place. The central idea is that classifiers which are not active when the environment gives payoff but which had an important role for setting the stage for directly rewarded classifiers can earn credit by participating in 'bucket brigade chains'. The success of some active classifier recursively depends on the success of classifiers that are active at the following time ticks.

As an additional means for improving performance Holland introduces a genetic algorithm to construct new classifiers from old successful ones. This feature will not be important for our purposes.

3 The Neural Bucket Brigade

In this section we propose a combination of principles of the bucket brigade algorithm with principles of neural networks. Competition can be introduced naturally into neural networks by a mechanism of lateral inhibition. What we still need is a mechanism analogous to the process of bidding and paying in classifier systems. This mechanism must establish recursive dependencies 'through time'. We introduce a local method for shifting 'weight substance' (initially provided by the environment) from weights that are allowed to transport activation information at a certain time to those weights that were 'setting the stage' one time tick earlier. We assume the following scenario for the inside of the black box:

The basic structure is some rather arbitrary cyclic directed graph, where the nodes are quite familiar processing units. Some units are used for input purposes, others serve as

outputs and may be coupled with effectors that may change the environment, which in turn may change the current input. Thus we have external and internal feedback.

The set of non-input units is partitioned into predefined 'competitive subsets'. All non-input units synchronously try to get activated by summing their weighted inputs at each time tick. All members of a predefined competitive subset laterally inhibit each other (by some 'winner-take-all' mechanism) thus competing for being active. All weights are randomly initialized with a positive real value, and are modifiable. Initially we will assume that there is instant decay: A unit active at time t manages to send its contributions to connected units that try to get activated at $t+1$, then the sender is switched off instantly.

All units active at time t *take away a fraction of the positive weights of their outgoing connections* (if there are any) *that lead to winners active at time t+1, and distribute this 'weight-substance' proportionally to the respective contributions among the incoming connections* (if there are any) *coming from winners (or input units) active at time t-1.* Since the weights determine the context-dependent strength of a unit, winners 'get paid' for setting the stage for their successors. Input units do not have any incoming connections that they could strengthen, they get activated by the environment thus representing holes through which the weight-substance of the system is leaking. The environment's influence is completed by sometimes rewarding (or punishing) the connections to currently active units in the case of useful output behavior. (An external critic decides what kind of behavior is useful.) The sum of all positive weights in the system remains constant, except for the weight-substance that is leaking through the input units and the new substance that is entering the system in the case of payoff. Thus we have a dissipative system which is consuming weight-substance provided by the environment.

More formally, at time t we denote the activation of the jth unit by $x_j(t)$, the weight on the directed connection between units i and j by $w_{ij}(t)$, and the contribution of some connection by $c_{ij}(t) = x_i(t-1)w_{ij}(t-1)$.

The activation rule works as follows: Unit j gets activated at time t if it is an input unit and receives a perception, or if it wins the competition between the units in the competitive subset it belongs to by having the largest positive net input $net_j(t) = \sum_i c_{ij}(t)$. We assume the simplest case: $x_j(t)$ equals 1 if unit j is active, and 0 otherwise.

If unit j is active then its weights change according to

$$\Delta w_{ij}(t) = -\lambda c_{ij}(t) + \frac{c_{ij}(t-1)}{\sum_i c_{ij}(t-1)} \sum_{k\ wins} \lambda c_{jk}(t) + Ext_{ij}(t)$$

where $0 < \lambda < 1$ determines how much of its weight some particular connection has to pay to those connections that were responsible for setting the stage at the previous time step. $Ext_{ij}(t)$ is the 'external payoff' that the environment gives to w_{ij} at time t, and may be computed like this: If the external critic does not know at time t whether useful behavior took place then $Ext_{ij}(t) = 0$. Else, if the critic notices a useful action , and if unit j was active at time t, then $Ext_{ij}(t) = \eta c_{ij}(t)$ with η being a proportionality factor. There is much room for more or less supervised strategies to determine Ext_{ij}: Every unit might get instructed at every time step, or just a few units at certain isolated time steps, etc.

The *weights* of the system (as opposed to the activations in Hopfield-networks or feedback-BP) have reached a stable state when every connection at any time is giving back as much weight-substance as it is receiving during the next time step. This means that (parallel) chains of units and connections cooperating in time have evolved.

Figure 1: Weight substance given by the environment in case of successful behavior is flowing through an agent living in a changing environment. The direction of weight flow is opposite to the direction of activation flow originating from perceptions. Credit assignment (appropriate weight changes) is done by local computations only. (See text for full explanation).

It is important to see the local character of this method. No book-keeping of past activations is required, not even the accumulative computation of, say, a weighted sum of past activations. Each weight and each unit in principle performs the same operation at each time tick. No such things as 'epoch boundaries' are required during training.

However, the method introduced above still has elements of global control: There is the clock for synchronous updates, for instance. To come closer to asynchronous models from biology we now give up the assumption of predefined competitive subsets and of instant decay. To save the concept of winning units we explicitly introduce fixed inhibitory connections (e.g. an on-center-off-surround structure (see (Kohonen, 1988) and (Grossberg, 1976)). We assume that the output x_j of unit j and the transmission properties of the connections are governed by differential equations that say that x_j does not change significantly during the time needed to transport activation information from one unit to its successors. Then we may write down a continuous time version of the weight changes caused by the neural bucket brigade in case of net_j being greater than zero:

$$\frac{\partial w_{ij}}{\partial t} = -\lambda x_i w_{ij} x_j + \frac{x_i w_{ij}}{\sum_i x_i w_{ij}} \sum_k \lambda x_j w_{jk} x_k + Ext_{ij}$$

Only positive weights appear in this formula, the inhibitory connections have to remain fixed. Tentatively denoting $\sum_k w_{jk} x_k$ by $back_j$ we find (by letting $\frac{\partial w_{ij}}{\partial t} = 0$) that the weight-flow through a positive weight w_{ij} that does not receive external payoff has reached a dynamic equilibrium if net_j equals $back_j$ all the time.

4 Simple Experiments

Any algorithm for learning sequential tasks also should allow the learning of static pattern association, since static learning tasks can be viewed as sequential tasks where inputs and desired outputs do not change over time.

In a preliminary experiment we tested whether the NBB is capable of adjusting a network such that it solves a static non-linearily separable task. The classical example for such a task is the XOR-problem.

The network was of the feed-forward type: A layer of three input units was connected to a predefined competitive subset of three hidden units and a predefined competitive subset of two output units. The subset of hidden units also was connected to the subset of output-units.

The discrete time version of the algorithm described above was employed. 32-bit floating point arithmetic was used for the simulations. In the beginning all weights were randomly initialized between 0.999 and 1.001. At the beginning of each cycle all unit activations were reset to 0, and one of the four binary XOR input patterns was randomly chosen. During the cycle this pattern was presented to the first two input units for a period of 6 time ticks. The activation of the remaining input unit was always set to 1, in order to provide a modifiable bias for every non-input unit in the network.

The task for the network was to switch on the first output unit if the XOR of the input pattern was 1, and to switch on the second output unit otherwise. The task was formulated as a reinforcement learning task: At each time tick the environment gave a payoff (playing a role similar to the role of a reinforcement signal) $Ext_{ij}(t) = \eta c_{ij}(t)$ to w_{ij} if unit j was an output unit and if it was switched on correctly at time t. In all other cases $Ext_{ij}(t)$ was set equal to 0. (Recall that payoff can be considered as a bit of weight substance which has to

be distributed in an appropriate way by the NBB algorithm.) Both η and λ were set equal to 0.005.

The network was said to correctly classify a single pattern if it switched on the corresponding output unit during the last three time ticks of a cycle, without the weight changing mechanism being employed. The network was said to have solved the problem if it correctly classified the four input patterns.

During 20 test runs the network needed an average of 619 pattern presentations to find a solution. So each of the four patterns had to be presented for about 155 times, which corresponds to the notion of 155 'epochs'. No systematic attempt has been undertaken to optimize this performance.

Most of the 20 solutions were brittle in the sense that further training did not necessarily stabilize them. For instance, after a solution had been found, another 5 training cycles could lead to worse performance, again. So we measured the number of cycles needed to achieve a stable solution.

A solution was considered to be stable if 100 additional pattern presentations did not disturb the performance of the network. During 10 test runs it was found that each pattern had to be presented for about 674 times in order to reach this criterion.

We also conducted experiments where the input patterns for successive cycles were not chosen randomly but in periodical sequential order. Here we found that the average time to find a solution increased, and that the solutions tended to be more brittle in the sense that it took much longer to achieve stable solutions. This suggests that the random element in the process of pattern selection introduces a stabilizing effect. One might suppose that similar stabilizing effects could be achieved by using stochastic activation rules. However, this has not been tested.

Where can instabilities arise from? The brittleness of the first solutions was attributed to the empirically observed fact that competing units within a competitive subset often had very similar net inputs. This in turn was attributed to a property of the NBB algorithm: Weights of connections leading to units that loose a competition remain the same. Consider a unit j that does not participate in a bucket brigade chain causing a correct classification of pattern A. This means that j's weights do not change. If the weights of j are slightly increased during the presentation of another correctly classified pattern, this modification may also lead to a winning situation for j during the next presentation of A, if the net input of the competitor of j who usually won during A's presentation was only slightly larger than j's net input. This may cause an incorrect classification of A. The interplay of these effects may lead to instabilities.

The *raisons d'être* for the NBB are non-stationary environments. One of the simplest tasks involving non-stationary environments may be to recognize different kinds of motion. We conducted a simple experiment with varying perceptions. A one dimensional 'retina' consisting of 5 input units (plus one additional unit which was always turned on) was fully connected to a competitive subset of two ouput units. This subset of output units was completely connected to itself, in order to allow recurrency. The task for the network was to switch on the first output unit after an illumination point has wandered across the retina from the left to the right (within 5 time ticks), and to switch on the first output unit after the illumination point has wandered from the right to the left.

During one cycle one of the two sequences (which had been chosen randomly) was presented to the network twice. Payoff was given as described for the stationary experiment, the same initialization and the same parameters η and λ were used. In 1 out of 10 test runs the

network did not find a stable solution within 3000 cycles (according to a criterion analogue to the one used for the stationary experiment). In the remaining 9 test runs an average of 223 cycles per sequence were needed to achieve a stable solution.

The experiments described above share a rather simple nature. It remains to be seen how well the NBB can deal with more difficult problems, like the learning of motor control for autonomous agents in a changing environment. We currently do not have sufficient empirical and mathematical results to decide under which environmental circumstances solutions represent attractors, and how fast convergence could be.

5 Concluding Remarks

There is an analogy between the NBB and competitive learning (Grossberg, 1976)(Kohonen, 1988)(Rumelhart and Zipser, 1986). Competitive learning also can be interpreted as a shifting of weight substance. However, here it is the weakly contributing incoming connections to a unit that have to pay to the strongly contributing incoming connections. In contrast, the NBB causes weight shifts from outgoing to incoming connections. This is the key feature used for relating present system states to past states.

Due to the local nature of all computations, the discrete time version can easily be implemented such that the time complexity of one update cycle (activation changes and weight changes) is $O(n)$ where n is the number of weights in the system. For some particular connection all information needed at a given time is its current weight, its contribution during the current time step and its contribution during the last time step. For some particular unit all information needed at a given time is its current activation, the summed contributions it receives during the current time step, and the summed contributions it received during the last time step.

Short term memory can be identified in activations wandering around feedback loops. Such loops may even become stable: A competitive subset of units that is permanently referencing itself can lead to a local dynamic equilibrium of weight flow (and of activation flow running in the opposite direction). Such equilibria may get perturbed by new inputs from the environment or from other competitive subsets that do not participate in the loop.

One difference to Holland's bucket brigade algorithm is that there is no analogue to the creation of new classifiers at run time: The number of connections in an NBB system remains fixed. The justification for this is given by the fact that weights are modifiable, while the 'specificity' of a classifier is not. (In (Compiani et al., 1989) Compiani, Montanari, Serra and Valastro consider more relationships between classifier systems and neural networks.)

Finally we would like to address the question of biological plausibility. We certainly do not want to suggest that the brain uses a weight shifting mechanism for e.g. physically transporting transmitter substance from synapses of outgoing connections to synapses of incoming connections. However, we do not want to exclude the possibility that some kind of feedback mechanism exists whose effects on the synapses are similar to the effects caused by the NBB.

A major property of the brain seems to be that the motoric actions which it causes depend on local computations only. The major contribution of this paper is to propose at least one possibility for how completely local computations within a neural network-like system may lead to goal directed parallel/sequential behavior.

The NBB represents a general credit assignment scheme for neural network-like structures.

'General' often seems to imply 'weak'. How 'weak' is the NBB? It remains to be seen whether the NBB can be successfully applied to difficult control tasks.

References

Almeida, L. B. (1987). A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. In *IEEE 1st International Conference on Neural Networks, San Diego*, volume 2, pages 609–618.

Compiani, M., Montanari, D., Serra, R., and Valastro, G. (1989). Classifier systems and neural networks. In Caianello, E. R., editor, *1st Workshop on Parallel Architectures and Neural Nets*.

Grossberg, S. (1976). Adaptive pattern classification and universal recoding, 1: Parallel development and coding of neural feature detectors. *Biological Cybernetics*, 23:187–202.

Holland, J. H. (1985). Properties of the bucket brigade. In *Proceedings of an International Conference on Genetic Algorithms*. Hillsdale, NJ.

Kohonen, T. (1988). *Self-Organization and Associative Memory*. Springer, second edition.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In *Parallel Distributed Processing*, volume 1, pages 318–362. MIT Press.

Rumelhart, D. E. and Zipser, D. (1986). Feature discovery by competitive learning. In *Parallel Distributed Processing*, pages 151–193. MIT Press.

Werbos, P. J. (1974). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University.

Williams, R. J. (1988). Toward a theory of reinforcement-learning connectionist systems. Technical Report NU-CCS-88-3, College of Comp. Sci., Northeastern University, Boston, MA.

Williams, R. J. and Zipser, D. (1988). A learning algorithm for continually running fully recurrent networks. Technical Report ICS Report 8805, Univ. of California, San Diego, La Jolla.