Towards compositional learning with dynamic neural networks

J. H. Schmidhuber

Report FKI-129-90
April 1990

TUM
TECHNISCHE UNIVERSITÄT MÜNCHEN
Institut für Informatik, Arcisstr. 21, 8000 München 2, West Germany
Towards Compositional Learning in Dynamic Networks

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

Abstract

None of the existing learning algorithms for neural networks with internal and/or external feedback addresses the problem of learning by composing subprograms, of learning 'to divide and conquer'. In this work it is argued that algorithms based on pure gradient descent or on temporal difference methods are not suitable for large scale dynamic control problems, and that there is a need for algorithms that perform 'compositional learning'. Some problems associated with compositional learning are identified, and a system is described which attacks at least one of them. The system learns to generate sub-goals that help to achieve its main goals. This is done with the help of 'time-bridging' adaptive models that predict the effects of the system's sub-programs. An experiment is reported which demonstrates the feasibility of the method.

Introduction

Terminology

External feedback. Consider a neural network receiving inputs from a non-stationary environment and being able to produce actions that may have an influence on the environmental state. Since the new state may cause new inputs for the network we speak of external feedback.

Internal feedback. If the network topology is cyclic, then input activations from a given time may alter the way that inputs from later times are processed. In this case there is a potential for the 'representation of state', or 'short term memory', and we speak of internal feedback.

Dynamic Learning Algorithms and Networks. A problem that requires credit assignment to past activation states is called a dynamic problem. Learning algorithms for handling dynamic problems are called dynamic learning algorithms. Learning algorithms that are no dynamic algorithms are called static algorithms. For instance, all algorithms that require settling into equilibria while the inputs have to remain stationary are considered to be static algorithms, although the settling process is a dynamic one based on internal feedback.

If a given network type can be employed for dynamic problems, and if there exists a corresponding learning algorithm, then we sometimes speak of a dynamic network.

The credit assignment problem. If a neural network is supposed to learn externally posed tasks then it faces Minsky's fundamental credit assignment problem: If performance is not sufficient, then which component of the network at which time did in which way contribute to the failure? How should critical components change behavior to increase future performance?

*This work was supported by a scholarship from SIEMENS AG
Supervised Learning. A learning task is a supervised learning task if there are externally defined desired outputs at certain times, but the network never needs to discover output actions on its own. Supervised learners have to consider only the internal feedback for performing credit assignment.

Reinforcement Learning. A learning task is a reinforcement learning task if the teacher only indicates once a while whether the system is in a desirable state or not, without giving information about how to reach desirable states. Usually an evaluative (non-instructive) teaching mechanism sometimes provides a scalar signal, the reinforcement, whose value indicates success or failure. During training the network is supposed to discover on its own outputs that eventually lead to desirable states. In contrast to supervised learning, there can be something like undesired inputs caused by former output actions. In general the external unknown dynamics have to be taken into consideration to perform credit assignment.

Reinforcement learning is strongly related to control tasks. With many control tasks more information is available about goal states than just a simple reinforcement signal. However, just as with reinforcement learning, the (sequential) outputs necessary to achieve the goal states in general are not known.

Existing Algorithms for Dynamic Networks and some Major Weaknesses

Various algorithms for attacking the fundamental credit assignment problem with dynamic learning algorithms in more or less complex non-stationary environments have been proposed. These algorithms can be classified according to various criteria. Some algorithms are suited for ‘pure’ reinforcement learning with internal feedback [25] [14], some are suited only for supervised learning [10] [9] [26] [12] [3]. Some are based on pure gradient descent [25] [9] [10] [26], some are inspired by temporal difference methods (Sutton’s TD(λ) with small λ [21]) [2] [1] (only external, but no internal feedback), and [15] [17] [19] [13] (internal feedback possible). Some algorithms involve the construction of a model of certain aspects of the environment, e.g. by using TD-methods [2] [1] [15] or Werbos’ heuristic dynamic programming [23] [6] or by using pure gradient descent methods [11] [4] [8] [18] [16] (using the approach of system identification).

All these algorithms have at least one thing in common: They show significant drawbacks when the credit assignment process has to bridge long time gaps between past actions and later consequences. Consider the following example (which is just meant to illustrate some fundamental problems):

A robot controlled by a dynamic neural network arrives at home and detects that it can not open the door because it has not got the key with it. It left the key on a table at the university (where it teaches an undergraduate course on neural networks). A wise action (to be detected by the credit assignment process) would have been to grasp the key from the table before leaving the university.

What would happen if the robot used a gradient descent method [11] [8] [10] [18] for credit-assigning its past behavior?

Essentially all past activations of all units in the system would contribute to the computation of an error gradient for the network weights. (The error may be given by negative reinforcement, or by the difference between the desired input (the view of an open door, say) and the actual input to the robot.) Traces of every past action of the robot, of every step that it made between its work place and its home would be taken into consideration for credit assignment. However, most of these steps are totally irrelevant in the context of the present task (which is to modify the network such that something like the current undesirable state will not occur again). In many cases there will have been only a few events in the past that might have contributed to the current failure, including the state corresponding to the decision to leave the university without the key.

What would happen if the robot used a TD-based method in the style of [2] [1] [14] [15] for credit-assigning its past behavior?

Only the most recent state(s) that the robot went through would be associated with a modified prediction of the undesirable event. The robot would have to repeat the same mistake again and again to allow credit assignment to relevant decisions that have been taken somewhere in the beginning of each of the many unsuccessful trials.

Obviously both approaches show awkward performance in the case where the robot already has learned a lot of action sequences in the past (like ‘walking home’ or ‘grasping the key’). Both approaches
Figure 1: A program executor, an evaluator, and a sub-goal generator are shown. (See text for full explanation.)
Figure 2: Two (or more) copies of the evaluator serve to compute a gradient for the sub-goal generator. (See text for full explanation.)
A conventional algorithm was used to train an artificial 'animal' to march from a certain point in a one-dimensional world to another one. Both start and goal states were indicated by coordinates of corresponding points. None of the programs taught to the animal took more than ten time steps. The maximal stepsize of the animal was limited in such a way that there were combinations of start and goal states where the animal could not know a corresponding program after training.

The evaluator was trained in a second phase to predict for given combinations of start and goal states whether there was a corresponding program or not. One reason for choosing a simple environment was to isolate the sub-goal generation process from effects that could be introduced by an adaptive on-line evaluation function (an adaptive critic, say). For our simple environment it was easy to define a prewired evaluation function: The 'goodness' of some program indexed by a start and a goal state was given by the difference between the position of the animal after program execution and the desired goal state. (Of course, future research will focus on parallel on-line learning of all components of the system, however, as always, it is preferable to proceed incrementally from small problems to bigger ones.)

In the final phase the sub-goal generator was trained: Combinations of start and goals states that did not have a working program associated with them were given to the sub-goal generation process described in the last section.

The sub-goal generator actually learned to generate appropriate sub-goals for the animal.

Currently some ongoing experiments are being conducted with two-dimensional environments (where there are obstacles).

The 'dividing problem' and 'causality detectors'

What is a good sub-program that is worth being memorized via a corresponding start/end combination? Similarly: Which sub-sequences that are visible in the environment 'belong together'? These questions certainly have to do with unsupervised temporal regularity detection.

To represent the causal structure of the environment in an efficient way, the following basic idea is proposed.

A sub-sequence that 'belongs together' is a sequence during which it is easy for an adaptive predictor to predict the input at a given time step from previous time steps. The adaptive predictor can be trained by any learning algorithm for dynamic recurrent networks. Whenever there is a mismatch between expectation and reality, there is a reason to memorize the situation in a separate associative device which only memorizes the unexpected events. There is also a reason for generating a new name for a sub-sequence (one can simply take the start and the end state of the sub-sequence) that bridges the time between the last unexpected events. This is very natural and also efficient in a certain sense, since the expected events do not have to be memorized, they can be deduced by considering what the system already knows.

What we get is some sort of causality detection: Causality detection aims at the reduction of the representation of the external dynamics such that a 'minimal' description of the causal temporal structure of the environment is obtained. Of course, 'minimality' here is relative to the current knowledge of the system: An environment may have some deep causal structure, but the learning system may be far away from being able to detect it.

Note that this kind of causality detection is reminiscent of what you yourself do: You like to remember especially the unexpected events. Those events that always repeat themselves do not call your attention.

Clearly, for goal-directed learners there are other interesting candidates for situations worth to be memorized. Regularity detection should be heavily influenced by selective attention [20][24] and current system goals. Currently research is being done on integrating 'causality detectors' and sub-goal generators into a coherent whole.
Future research

In the near future it is intended to apply sub-goal generators to problems of attentive vision as described in [20]. There the goal is to generate (without a teacher) fovea-trajectories involving translations and rotations which lead a non-stationary ‘fovea’ to focus on a part of the plane corresponding to a certain target. For the most interesting applications this requires to consider the interplay of effects introduced by parallel on-line learning of all system components.

Sub-goal generation is an essential but in general also very complex process. With many problems there is a need for a hierarchy of goals and sub-goals, and there also is a need for mechanisms stepping through that hierarchy in order to find appropriate sub-goals. I am currently working on an architecture where there are internal actions that allow the learning system itself to influence the way it builds associations, shifts attention, and triggers sub-goal generation. Of course, the goal is to make the internal actions themselves adaptive. It is intended to use causality detectors as described in the last section for credit assignment for both external and internal actions.

I suggest that by considering systems that can dynamically manipulate the way how they learn (to a certain well-defined degree), we will enter a very promising and exciting field which will help to bridge the gap between so-called ‘sub-symbolic’ and so-called ‘symbolic’ computation. This will open the door to self-introspective neural systems and to all the benefits that usually are associated with introspection and meta-learning.

References


