Applied Intelligence 17, 7–10,2002
© Kluwer Academic Publishers

Guest Editorial for Special Issue on Scalable Applications of Neural Networks to Robotics

By Patrick van der Smagt and Daniel Bullock

The goal of this special issue is to present a set of papers focused on scalable applications of neural networks to robotics. The field of neural networks is very broad, with a biological side and an engineering side. On the biological side, the field spans from studies of highly specialized neuronal circuits in primitive animals to studies of the kind of general-purpose, adaptive neuronal circuits that are exemplified by the human cerebral cortex and the cerebellum. On the engineering side, the field spans from adaptive function approximation, through machine vision to pattern recognition and classification, and on to adaptive control systems for processes of all kinds. Robotics is an area in which the full range of neural networks ideas could find a natural home, but for many applications, adaptive, self-learning methods will become practical and preferred solutions only if they are implemented in a way that scales well with the dimensionality of the problem.

Adaptive movement control and applications of cerebellar models

A natural place to begin any survey of possible applications of ideas from biological neural networks to robotics is with the cerebellum. Virtually every movement-controlling signal generated within the brains of terrestrial vertebrates is subject to learned modification by the cerebellum. This implies that the basic cerebellar circuit has repeatedly proven able to provide a service that has been valuable to movement generators arising in other parts of the central nervous system. Two of the papers appearing in this issue focus exclusively on scalable applications of ideas about the cerebellum, and these papers are largely complementary. The first paper, by Peters & van der Smagt, notes that there are several distinct formal models of what and how the biological cerebellum contributes to movement control in animals. It then surveys a number of these formal models. In addition to reviewing variations on the well-known CMAC (cerebellar model articulation controller) design, which was first introduced by Albus [1], the authors review recent proposals that the cerebellum may perform both inverse and forward modeling. Whatever the merits of these formalizations qua models of a major part of the brain, they can be independently assessed as sources of viable ideas for adaptive robotic controllers. In this context, the authors present an introduction to the general problem of robot motion control, with an emphasis on what any cerebellar model would have to provide to be regarded as a robust and scalable solution in this area of robotics. Of particular interest to the authors are applications to control of a newer generation of very light weight and flexible robots.

Whether the cerebellum is performing inverse or forward modeling, or both, it needs to learn associations between state representations and outputs. The second paper, by Rhodes & Bullock, begins with a review of data indicating that the cerebellum has the ability to adaptively time its outputs, so that they follow the pertinent state inputs by any required interval in the range from around 90 ms to several seconds. Prior models of this adaptive timing competence have been far from optimal in their scalability. In particular, virtually all prior models have required an entire population of computational elements in association with each unique state input to achieve adaptive timing. This dependence on populations implies poor scalability as the state space grows larger. Rhodes & Bullock propose instead that the biological cerebellum may be using a temporal "slide and latch" process that requires only a single adaptive element to learn the time after which detection of a given state should be followed by the output. Within the existing universe of cerebellar adaptive timing models, such a mechanism may be said to exhibit optimal scalability. When used in tandem with a recurrent architecture, in which the outputs feedback to become part of the evolving state representation, this mechanism can be used for adaptive timing of single actions and to learn sequences of timed outputs as responses to state detections. Adaptive timing is very important in those cases where many of the state events upon which responses need to be conditioned are transient events that lead the time when action is needed by a significant interval.

Efficient on-line function approximation

Forward and inverse modeling of robot dynamics requires approximation of non-linear functions, and this was one of the original attractions of CMAC. Both of the next two papers advocate neural networks that can be adapted incrementally, using online training data, to provide piecewise approximation of non-linear functions. In each case, the authors avoid multi-layer perceptrons (MLPs) for two reasons. According to the authors, MLPs in their basic form are more suitable for offline learning, and in MLPs adaptation in one part of the input space traditionally has undesirable effects on information stored in association with another part of the input space. Although neither paper is presented as being in the cerebellar tradition, both in fact adopt the cerebellar strategy of highly local adaptation that allows what is learned in one part of the state space to be largely insulated from learning pertinent to another part.

Schaal, Atkeson, & Vijayakumar treat two kinds of locally weighted learning (LWL). In memory-based LWL, all training data (input-output pairs) are stored in memory, and efficient lookup and interpolation methods are utilized to process this stored data to make an appropriate output prediction for new inputs on the basis of locally linear models. Non-memory-based LWL is statistically similar, but instead of storing the training data, it simply uses each new input-output sample to incrementally update the internal models. Whereas this loses the flexibility associated with storing whole data sets, the time needed to generate predictions for new inputs is greatly reduced, as are the memory demands of the method. Gaussian kernels are used to compute the region of validity, or receptive field, of each locally linear model. The authors take these ideas through several iterations to arrive at what they call locally weighted projection regression (LWPR), which is a non-memory based LWL technique that uses a kind of partial regression to allow it to be practical with high dimensional input spaces. The LWPR variant of LWL is applied to learning an inverse dynamics model of a 7 DOF anthropomorphic robot arm.

Krabbes & Doeschner also present an approach to learning inverse dynamics, with an application to control of a 6 DOF industrial robot. Their approach is based on an architecture that uses functions similar to radial basis functions (RBF) to define receptive fields of local neurons that contribute to piecewise function approximation. In this respect, the approach is similar to that of Schaal et al. However, Krabbes and & Doeschner take a different approach to avoiding the curse of dimensionality. They use an analytic investigation of the robot dynamics to transform the problem from state space to configuration space, and then to show that the dimensionality can be reduced sufficiently in that space to permit an efficient but sufficiently accurate representation of the robot's inverse dynamics in their RBF-like architecture.

The paper by Doerschuk, Nguyen, & Li addresses the problem of designing a controller for the takeoff phase of running with a jointed leg. Although a CMAC controller is envisioned for swing and landing phases, both of which require rapid tracking of prescribed joint trajectories, a neurofuzzy controller is recommended for the takeoff phase. Although slower than CMAC, the memory demand of the neurofuzzy controller was found to scale much better in actual application. Here the principal controlled variable is stride length and the task for the adaptive takeoff controller is to learn the correct combination of torques to apply to each leg joint to achieve whatever stride length is desired. The neural system incrementally establishes prototypes for data clusters defined by similar values of input state (initial conditions) and output result (stride characteristics), and learns to associate with these prototypes the control actions that carried the system from the initial to the final state. The number of stored prototypes can be varied to achieve the desired level of accuracy. To compute a control action for a new input, a variable number, K, of nearest neighbor (KNN) prototypes are selected, and their contributions weighted by their distance from the current input. Several iterations arrived at a training protocol that avoided needless proliferation of prototypes while supporting accurate performance.

Variation and selection as an alternative to error-guided learning

Most cerebellum-mimetic controllers use error-guided learning to adjust multiplicative weights in the neural network. This is natural because in the biological cerebellum the teaching signal waxes and wanes respectively with the onset and correction of performance errors, and because most adaptive controllers are used in contexts where desired trajectories are prescribed, so it becomes easy to measure performance error relative to this desired trajectory. An alternative approach is to use random variation, recombination, and selection mechanisms that mimic Darwinian evolution in order to conduct a search for improved weight distributions (or even connectivities) in neural network based controllers. But in some applications, performance goals may be so generally specified that there are no desired trajectories with respect to which errors can be measured. Instead, a global fitness score is used to assess overall system performance. One problem with Darwinian variation and selection is that it can be very slow to converge to an adequate solution, and this problem tends to be greatly amplified as the complexity of the controller is scaled up. The paper by Salomon examines object avoidance in the Khepera robot and compares two types of neural network controllers that differ in their complexity: Braitenberg controllers (simple) and receptive field controllers (relatively complex). For both cases, he shows that the convergence time associated with a standard GA (genetic algorithm) can be reduced by more than an order of magnitude by an alternative evolutionary algorithm called ES (evolutionary strategy). Contrary to GAs, ESs encode real-valued parameters as floating point numbers, mutate all parameters simultaneously, and use the selection mechanism to search for an optimal mutation step-size.

Multiple overlapping sequence learning and recall

In the final paper, Araujo & Barreto construct an adaptive neural network that can learn to recall multiple long sequences even if those sequences have overlapping, i.e., shared, elements. This requires using temporal records (short term memories) of prior sequence states to provide a temporal context sufficient to disambiguate the sequence continuations that need to be produced upon exiting the shared states. The construction uses a combination of competitive learning [2-5], Hebbian learning and tapped delay lines to achieve its ability to incrementally recall a previously experienced sequence. Because the recall operation is incremental, i.e., requires sequential passage through the sequence of states to complete recall, the recall time grows with the length of the sequence. The applications discussed by the authors are applications in which the system learns and recalls a long sequence of postures, such as those that would be required for a robot to track a figure eight. However, the architecture could be applied much more generally, e.g., to learning and recall of symbol sequences.

Some general observations

One remarkable aspect of the contributions is the universal avoidance of batch-trained multi-layer perceptrons in favor of incremental learning approaches that can be used online. This represents a large shift relative to what might have been observed in such a collection of papers even 5 years ago and can be explained by the fact that such approaches are in large part understood and their limitations known. Another point worth remarking is that on a microscopic level, virtually all the approaches use piecewise linearization to accomplish function approximation. This can be applied by using stored data examples in flexible but slower and more memory intensive methods, or it can be applied by using data examples just once to modify one or another of the set of local functions. In the classical CMAC architecture, the input space receptive fields were predefined. In a majority of the more recent models considered in this set of contributions, some adaptivity of the receptive fields is utilized. This allows many choices to be made that change the nature of generalization to fit the problem to be solved. Finally, the adaptive timing competence studied by Rhodes & Bullock is almost orthogonal to the aspect of intelligence treated in the other contributions. It is better regarded as a possible supplement than as an alternative, although it might replace the need for tapped delay lines in the model of Araujo & Barreto.

References

1. Albus, J.S. "A new approach to manipulator control: The cerebellar model articulation controller (CMAC)." *Journal of Dynamic Systems, Measurement, and Control*, vol. 97, pp. 220—227, 1975.
2. Grossberg, S. "How does a brain build a cognitive code?" *Psychological Review*, vol. 87, pp. 1-51, 1980.
3. Grossberg, S. "Adaptive pattern classification and universal recoding. I. Parallel development and coding of neural feature detectors." *Biological Cybernetics*, vol. 23, pp. 121-134, 1976.
4. Malsburg, C. von der Self-organization of orientation sensitive cells in the striate cortex. *Kybernetik*, vol. 14, pp. 85-100, 1973.
5. Rumelhart, D.E. & McClelland, J.L. *Parallel distributed processing, volume 1: Foundations*. Cambridge, MA: MIT Press, 1986.