DUALITY IN NEUROCOMPUTATIONAL INDUCTIVE INFERENCE: A SIMULATIONIST PERSPECTIVE

Kevin G. Kirby
Computer Science and Engineering Department
Wright State University
Dayton, OH 45435

ABSTRACT

Inductive inference is the process of inferring a description a function from finite subset of its graph. Connectionist inductive inference typically involves gradient descent algorithms in weight space. When inferring functions of unbounded sequences such algorithms run on recurrent nets and become computationally expensive. In this paper we present a broader framework framework for inductive inference, and show that such problems admit a "dual" approach, which can be phrased in terms of the simulation-as-homomorphism perspective in systems theory. Whereas the usual approach adapts the dynamics of the net to match the dynamics of the target system, the dual approach keeps the dynamics fixed and learns a homomorphism from the net to the target. The latter technique is promising because of its efficiency and its direct applicability to learning by continuous non-connectionist systems, such as neural fields.

1. INTRODUCTION

Neural networks have become widely known in the past decade primarily because they embody simple parallel heuristics for approximating functions $f : A \to B$. In most supervised learning approaches, elements of the function domain are pushed through a feedforward network, and subject to an interleaved sequence of linear transformation and componentwise squashing. The actual value of the function on these points is compared to the net output, and updates to net parameters are made to lower the error on subsequent iterations. In many applications, the domain consists of fixed-length vectors; $A = X^n$. Alternatively, when the domain consists of sequences of arbitrary length, $A = X^*$, each item of the sequence may be passed through the net in a time sequence. For non-trivial functions of sequences, the net will need to keep state information, and will thus often have recurrent connections. Generalizations of feedforward gradient algorithms to the recurrent case are straightforward but expensive. (See [1] for a recent perspective.)

One problem often tackled by recurrent networks is that of learning a regular language $L \subseteq X^*$, where X is a finite set of irreducible symbols. This may be phrased as learning a function $f : X^* \to \{0,1\}$. The everyday "RMS/max-error" approach to supervised learning is too crude for a careful analysis of this problem, so in this paper we turn to the new field of *distribution-free learning theory* [2], also known as *Probably Approximately Correct (PAC) learning*. We want to explore algorithms that can learn f approximately with high probability. We will imagine that there is an unspecified probability measure μ over X^* according to which training and test examples are drawn. Since this is a countably infinite set, a uniform distribution cannot be defined. Realistic distributions would tend to favor short "real-world" length patterns. Let $\mu(L)$ denote the measure of a subset $L \subseteq X^*$. Essentially this assigns a number to each of the uncountably many languages over X. In PAC-learning, we wish to find an algorithm such that, for given accuracy ε and confidence δ, will process the set of samples and, with probability less than δ, construct a representation for a subset L' which differs from L on a set of measure greater than ε.

The representation of the concept class is crucial. We may attempt to learn regular languages by searching spaces symbolic representations, such as regular expressions, transition graphs, and so on. Connectionism attempts to use the weight spaces of recurrent networks as its representation space, performing gradient descent to minimize error on small training sets in the hope of capturing the entire correct, typically infinite, regular language, within the dynamics of the net. In the following sections, we show that the actual mechanisms for such algorithms can be fruitfully analyzed in a simulationist perspective.

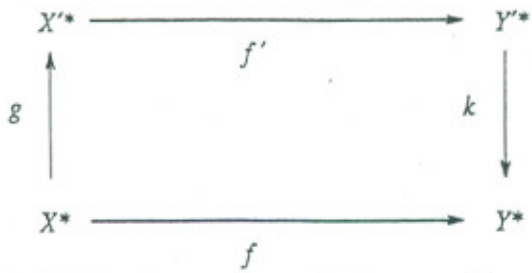
2. LEARNING AND SIMULATION

In sequential learning, as discussed above, we wish to learn a function of sequences. In the most general case, the range of the function is also a sequence. We feed in the terms of the sequence to the trained system, and, concurrently, expect the output sequence to be correct. We will speak of a *target* sequential behavior $f: X^* \to Y^*$, We wish to have a system (call it the "net") match this function. It is productive to view this as the learning *of* a dynamical system *by* a dynamical system. At the practical level, we only demand a crude behavioral match between the net and the

IEEE Int. Conf. on Sys. Eng., Aug 1991

CH3051-0/91/0000-0351 $1.00 © 1991 IEEE

target. This is characterized in the Zeiglerian simulation framework as "I/O Function Observation" [3]. We need to find a net behavior f', an encoding g, and a surjective decoding k such that following diagram commutes.



In PAC-learning, we expect this diagram will not strictly commute; the chances of perfect learning may be nil. Bringing in the PAC-learning terminology, given any two functions $r,s:A \rightarrow B$, define

$$Er(r,s) = \{x \in A \mid r(x) \neq s(x)\}. \qquad (1)$$

We would like an algorithm which, when given a function f in a certain concept class (e.g. characteristic functions of regular sets), will find maps f', g and h such that

$$Prob\left\{\mu[\,Er(f,\ k\cdot f'\cdot g)\,] > \varepsilon\right\} \ < \ \delta\ . \qquad (2)$$

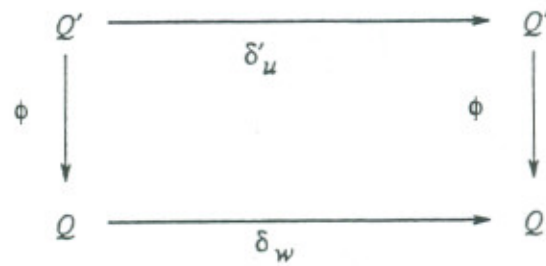
For small ε, δ, this inequality says that the diagram above "probably approximately commutes".

In the Zeiglerian approach, a modeler/scientist would posit a state space for the target system. It is at this point that we introduce the specific nature of the dynamics. Since we wish to learn a regular language, we will interpret this as learning a finite-state automaton.* The adaptive system which is doing the learning may be a discrete recurrent net of threshold gates, or a nonlinear high-dimensional continuous dynamical system. In this section, we will merely speak of a dynamics δ' on a phase space, with trajectories perturbed by inputs.

In Zeiglerian terminology, we have now moved to the level of the "I/O System Morphism". It is useful at this juncture to adopt the language of transformation monoids, which has been used to study representation change in classical artificial intelligence [4]. View δ^* as a collection of maps $Q \rightarrow Q$. Let us call this collection M, where we have used the identification

$$\delta_w(q) \equiv \delta^*(q,\ w). \qquad (3)$$

Under this identification, we can view the DFA as a *transformation monoid* (Q, M). Let $A = (Q, X, Y, \delta, \omega)$ and $B = (Q', X, Y, \delta', \omega')$ be two DFAs, sharing the same input and output sets for simplicity. We say A is simulated by B if there exists a partial surjection $\phi:Q' \rightarrow Q$ such that for each $w \in X^*$ there exists a $u \in X^*$ making the following diagram semi-commute:



The "semi-commutativity" means that whenever $\delta_w \cdot \phi\,(q')$ is defined, it equals $\phi \cdot \delta'_u\,(q')$ for all $q' \in Q'$. Intuitively, think of the "big" DFA Q' simulating the "small" DFA Q, by using a many-one representation. A schematic view of this idea, when ϕ is total, is shown in Figure 1.

The conditions in the simulation definition yield three requirements. (*i*) A state in Q' cannot represent more than one state in Q. This follows from ϕ being a function. The reverse is legal, however, since ϕ need not be injective. (*ii*) The simulation should not lose information. This means that two trajectories cannot converge on a single state in Q' unless they do so in Q. The reverse is legal, and is the essence of "quotient learning" described below. (*iii*) The simulation should preserve "relevance". In other words, a trajectory inside the domain of ϕ cannot move out of it later. Again, the reverse is legal. It should be noted that even though we are trying to keep the mathematics simple by presenting the definitions in automata theory, the ideas can be translated into an arbitrary theory of dynamical systems. In this paper, the state space Q' will belong to our learning machine; the state space Q will be a finite space holding the the behavior our machine is attempting to learn.

3. STATE-SPACE LEARNING

A simple recurrent network for the discrete sequential associations $\mathbf{x}(t) \Rightarrow \mathbf{y}(t)$ can be represented by the equations

$$\mathbf{q}(t) \ = \ \mathbf{s}[\ \mathbf{A}\mathbf{x}(t-1) + \mathbf{B}\mathbf{q}(t-1) + \mathbf{C}\mathbf{y}(t-1) - \theta\ ] \qquad (4)$$

$$\mathbf{y}(t) \ = \ \mathbf{s}[\ \mathbf{U}\mathbf{q}(t) + \mathbf{W}\mathbf{x}(t-1) + \mathbf{V}\mathbf{y}(t-1) - \chi\ ] \qquad (5)$$

Here the vector $\mathbf{q}$ represents the states of a recurrent hidden subnetwork, which mediates between the input and output vectors. Note the similarity to the state space equations for a linear system, the essential difference being the interposition of a function $\mathbf{s}$. This nonlinearity is typically a hard (e.g., *sgn*) or soft (e.g., *tanh*) step. The vectors θ and χ serve as thresholds. (In many presentations, the affine transformation inside this dynamics is made linear by a homogeneous transformation, raising the dimensions by one.) The components of the matrices are interpreted as connection weights. A, B, and C are afferent to the hidden recurrent subnet, and U, V, and W are afferent to the output units.

In the familiar supervised learning paradigm, we calculate the difference $\mathbf{e}(t)$ between the net output $\mathbf{y}(t)$ and a

* A deterministic finite-state automaton (DFA) $A = (Q, X, Y, \delta, \omega)$ consists of a finite set of states Q, a finite set of inputs X, a finite set of outputs Y, a transition function $\delta: Q \times X \rightarrow Q$, and an output function $\omega: X \rightarrow Y$. It is convenient to use the transitive closure of the transition function $\delta^*: Q \times X^* \rightarrow Q$, which gives us the state to which an input sequence drives a given state.

teacher signal $\mathbf{y}^*(t)$. Typically the weights are modified by descent along a gradient of the form

$$\mathbf{g} = \nabla \sum_{t=t_1}^{t=t_2} |\mathbf{e}(t)|^2 \ . \qquad (6)$$

Whereas the expression for this gradient neatly unfolds to outer products for feedforward nets (giving rise to the backpropagation rule), with recurrent nets the updates are nonlocal, "three-subscript" quantities. As such, the recurrent extensions of backpropagation are much more expensive in computation space and time. Nevertheless, the research field is exceedingly active, and new results are promising.

Let us examine the state space equations more carefully. Adaptation of the matrices A,B,C is on a different footing than the others. The state space of the q's may be of vastly greater dimensions than the input output space. (A,B,C) determines the dynamics on this space. This dynamics tells us how the net remembers its input history. These are precisely the matrices that are computationally expensive to train. We can identify them with the transition function δ' in the previous section. Recurrent backpropagation and similar algorithms try to make the diagram above (semi-)commute by adapting this state space dynamics δ'. We shall call this *state-space learning*.

On the other hand, the matrix U defines the mapping from the state space of q's to the outputs. U is trained to function as a vector of feature detectors which look at the raw state space, and carve it up to capture the correct behavior. If we pretend that q is like an external input, the processing by U is the same as in a single-layer feedforward net, the most tractable of all neural network architectures. Again referring to the diagram above, by adapting U we are indirectly adapting the morphism ϕ between the state spaces. Adapting the homomorphism is thus "dual" to adapting the dynamics.

Since the equivalence classes of elements of Q' that are mapped to the same point in Q form a quotient space under the homomorphism, we will call this technique *quotient-space learning*. Here one learns a quotient structure on Q', rather than a dynamics δ'. This is analogous to "lumping" in the simulationist framework.

4. QUOTIENT-SPACE LEARNING

That state space learning is possible, if not efficient, is attested to by the volume of recurrent network learning research, of which we have cited only one [1] as a pointer to the rest of the field. Conversely, because learning need only involve a adapting a single layer, quotient space learning is efficient, but is it possible? It not intuitive that any reasonable learning can be done by changing only a homomorphism, rather than directly manipulating the dynamics.

One effort that can be viewed as a step in this direction is the experimental study of Gallant and King [5]. They use an architecture we can describe here by equations (4,5) above. This amounts to having an internal recurrent network with fixed random weight matrices A, B, C; and a single output layer with adaptable weights consisting of the concatenation of W, U, and V. Gallant and King present the fixed random recurrent subnet as a kind of temporal version of the Rosenblatt technique of randomly boosting input patterns into a higher dimensional space to increase the chance of linear separability. The output processing can then employ only one layer, and use a perceptron-like algorithm to associate the state of the internal recurrent net with input patterns.

The work of Kirby [6] and Day [7] has expanded on this approach of using a fixed, random subnetwork to hold representations of input histories. One key discovery is that the global $O(n^2)$ connectivity inside the random nets is not necessary, and that local $O(n)$ connectivity slows down the learning time only slightly, while reducing space requirements by at least an order of magnitude. In fact, the matrix B in equation 4, can be reduced to a narrow band matrix, in which each unit has from 3 or 4 nearest neighbors. Further, one can set C=V=0. Systematic studies showed learning a mapping of finite sequences (the "robotic control task" of Gallant and King) proved simple, whereas learning 2- and 3-state finite automata proved considerably difficult, in that generalization off a training set was rarely better than 90%. Nevertheless, the generalization was present and significant, which was surprising, in that one might expect the perceptron layer to merely memorize a finite number of associations.

The implication of these studies was to show that one could move beyond a connectionist net to a system with only local connectivity. Instead of a lattice of threshold units, we studied a neural field, with excitation propagating according to reaction-diffusion equations. This is isomorphic to the intraneuronal dynamics model of Kirby, Conrad and Kampfner [8]. Figure 2 shows the architecture. By employing direct analog devices built on artificial membranes, one could realize this system cheaply, perhaps. This suggests the irony that an uncountable number of neurons may be cheaper to implement than a large finite number.

Other research outside the supervised learning paradigm may also be usefully analyzed from this perspective. For example, the active perception model of Whitehead and Ballard [9] constructs an implicit morphism between external and internal DFAs. The problem they identify as *perceptual aliasing*, the representation of two external states by one internal state, is exactly what one prevents when one demands ϕ be a (partial) function rather than a relation.

5. CONCLUSIONS

This perspective opens up many fundamental research questions, of which we name only a few: First, a mathematical study of when a random DFA can admit a quotient structure isomorphic to a given DFA must be undertaken. This is an interesting complement to the approach of Zimmer, *et al.* [4], who study the boosting of search spaces into larger but computationally more tractable spaces that are wreath products of elementary ones. Second, in learning a function on X^*, our apparent quotient structure may collapse for very long strings. We may need to investi-

gate fuzzy quotient spaces. Third, the output layer functions as a feature detector whose input space may be a manifold, rather than the vector spaces typical of pattern recognition. What can we prove about feature detectors on manifolds?

Finally, the curious message of this approach is that a machine learner, as an adaptive simulator, might just try some "creative lumping" in order to model the external world. There is a fascinating analogy to a recent theorem of Putnam [10], stating *Every open system is a realization of every finite automaton.* (This is used to refute the philosophical position of computational functionalism.) The theorem is proved by pointing out how *any* physical state space can be lumped *any* way we like, so that we can make it resemble any given system. We believe that "Putnamizing" complex dynamical systems may in fact be a realistic technology for machine learning.

ACKNOWLEDGEMENTS

This work was sponsored by the Air Force Office of Scientific Research. The author wishes to thank Louis Tamborino for his encouragement and insight.

REFERENCES

[1] R.J. Williams and J. Peng, "An Efficient Gradient-Based Algorithm for On-Line Training of Recurrent Neural Network Trajectories," *Neural Computation* Vol. 2, pp. 490-501: 1990.

[2] M.J. Kearns, *The Computational Complexity of Machine Learning*, MIT Press, Cambridge, 1990.

[3] B. Zeigler, *Theory of Modelling and Simulation*, Wiley, New York, 1976.

[4] R. Zimmer, A. MacDonald, and R. Holte, "Reasoning about Representations: Towards the Automation of Representation Change," *Proc. Florida AI Research Symposium*, pp. 201-205: April 1991.

[5] S.I. Gallant and D.J. King, "Experiments with Sequential Associative Memories," *Cognitive Science Society Conference*, Montreal: 1988.

[6] K. Kirby, "Context Dynamics in Neural Sequential Learning," *Proc. Florida AI Research Symposium*, pp. 66-70: April 1991.

[7] N. Day, *Inductive Inference of Regular Languages By Low-Connectivity Neural Networks*. MS Thesis, Department of Computer Science and Engineering, Wright State University: 1991.

[8] K. Kirby, M. Conrad, and R. Kampfner, "Evolutionary Learning in Reaction-Diffusion Neurons," *Applied Mathematics and Computation*, Vol. 41, pp. 233-263: 1991.

[9] S.D. Whitehead and D.H. Ballard, "Active Perception and Reinforcement Learning." *Neural Computation* Vol. 2, pp.409-419: 1990.

[10] H. Putnam, *Representation and Reality*, MIT Press, Cambridge, 1988.

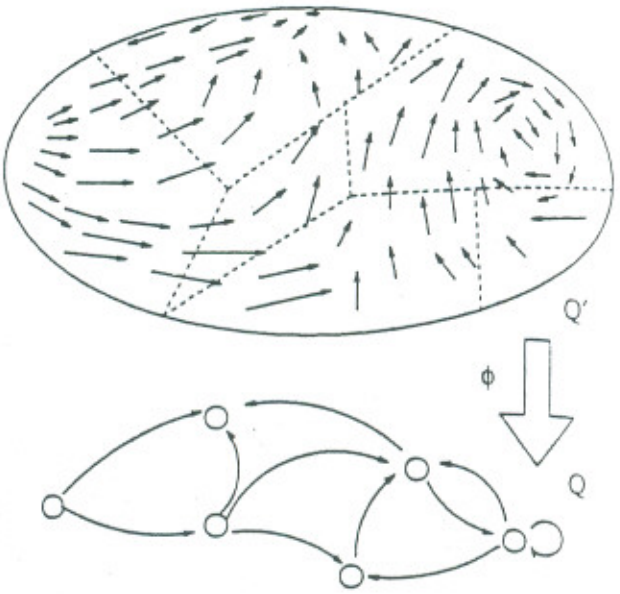


Figure 1. A high-dimensional dynamical system (top) simulating a finite-state automaton. This simulation relation can be *learned* in two dual senses: (1) the time evolution of the simulating system (top) can be adapted under a fixed transparent homomorphism ϕ; or (2) the dynamics of the simulating system can be fixed, and the homomorphism ϕ can be adapted to partition its state space so that the quotient dynamics is isomorphic to the finite automaton. The former approach is typical of connectionist inductive inference.

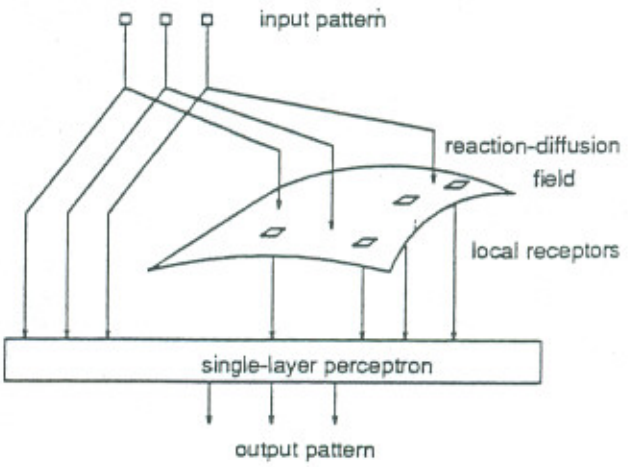


Figure 2. A neural field serving as a context-reverberation subsystem, replacing a random recurrent net. Neural field states hold compressed, scrambled representations of the input histories. In inductive inference of regular languages, the neural field state space Q' is mapped to the target finite automaton's state space Q by a homomorphism learned at the perceptron layer. This amounts to forming a quotient map by slicing images of the field's state space with hyperplanes.