

**Goal Directed Model Inversion:
A Characterization of Learning Behavior**

SILVANO COLOMBANO
AMES RESEARCH CENTER

MICHAEL COMPTON
HELEN STEWART
MARTHA DEL ALTO
RECOM TECHNOLOGIES

ARTIFICIAL INTELLIGENCE RESEARCH BRANCH
NASA AMES RESEARCH CENTER
MAIL STOP 269-2
MOFFETT FIELD, CA 94035-1000

*Submitted a revised version to the Artificial Neural Networks in Engineering (ANNIE'93)
November 14-17, 1993, St. Louis, MO.*

NASA Ames Research Center

Artificial Intelligence Research Branch

Technical Report FIA-93-16

August, 1993

**Goal Directed Model Inversion:
a Characterization of Learning Behavior**

Silvano P. Colombano, Michael Compton*,
Helen Stewart* and Martha DelAlto*
Information Sciences Division
NASA Ames Research Center
MS 269-2
Moffett Field CA 94035

*RECOM Technologies

Abstract

Goal Directed Model Inversion (GDMI) is an algorithm designed to generalize

learning system. The output of the learning system becomes the input to some external device or transformation, and only the output of this device or transformation can be compared to a desired target.. The fundamental driving mechanism of GDMI is to learn from success. Given that a wrong outcome is achieved, one notes that the action that produced that outcome "would have been right if the outcome had been the desired one". The algorithm makes use of these intermediate "successes" to achieve the final goal. One issue for an earlier version of GDMI was referred to as "the priming problem" and relates to the relative difficulty in achieving the very first external target. In this paper we report modifications to the GDMI algorithm which eliminate the priming problem.

of the neural network controller should have been. This error is in "robot goal space" not "action space" and, in terms of a neural controller, there is no meaningful way to apply this error to the output units of the controller. These units would, in general, code for values of joint angles, currents to motors etc. while the output error might simply be a distance between the target and the end effector.

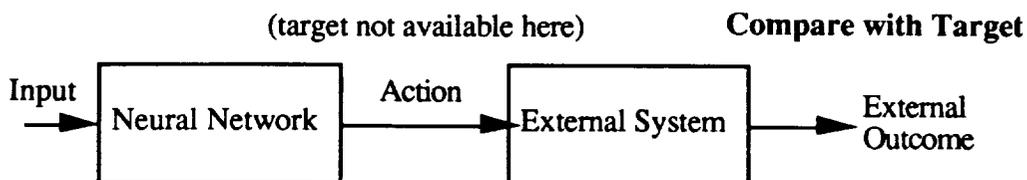


Figure 1. *Generalized Supervision. The target is not available at the neural net output.*

Our problem is how to make use of the information provided by that "distant" error. This has also been called learning with a "distal teacher" (Jordan and Rumelhart, 1989). Three approaches are possible: unsupervised learning (e.g. Kohonen 1990, Jorgensen 1990), reinforcement learning (e.g. Barto *et. al.* 1983, Williams 1988, Berenji and Khedkar 1992) and model inversion. A discussion of the first two approaches is beyond the scope of this work and we leave it to the references cited. Here we briefly discuss the third approach and suggest a new technique based on the model inversion concept. We have called this technique "Goal Directed Model Inversion" (GDMI).

A preliminary version of GDMI was presented in previous publications (Colombano *et. al.* 1992a,b) in the context of the inverse kinematic problem for a simulated robotic arm with three degrees of freedom. One problem noted with that version of GDMI was that the first goals presented to the system had to be very "easy". This lack of generality in the initial goal seeking behavior was called "priming problem". In this paper we show how this problem can be overcome and we begin studying the system. with systematic experimentation.

2. Model Inversion

Two techniques have been utilized for model inversion. One has been called "Direct Inverse Modeling", and the other "Forward Modeling" (Jordan and Rumelhart, *ibid.*). For the latter we prefer to use the name "Indirect Inverse Modeling". The reason for this preference will become evident when we describe the technique.

3.1 Direct Model Inversion

Direct Model Inversion is illustrated in figure 2. The neural network is trained to create an inverse model of the system which is to produce a desired output. The space of possible "actions" is sampled at random, and the external system produces the corresponding outputs. Each output becomes the input to a neural network which is then trained to produce the corresponding action. More succinctly: input-output pairs are obtained from the real system and are used, respectively, as the output-input pairs of the neural net (*i.e.* they are reversed).

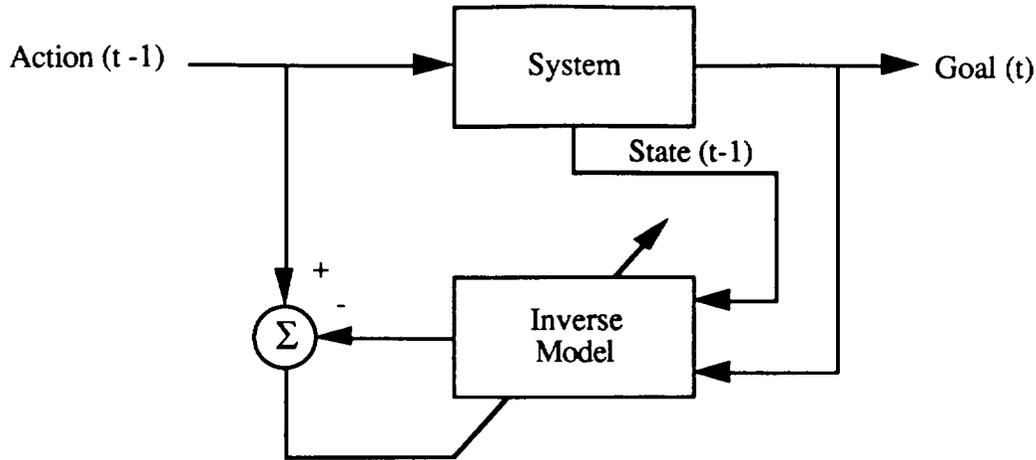


Figure 2. *Direct Model Inversion*

This approach can be successful in some situations, but it may not be able to achieve correct results in "many to one" mappings, *i.e.* when different actions achieve the same goal. The reverse mapping of goals to actions will, in general, produce an average of these correct actions. But averaging over correct actions will not necessarily produce a correct action (Jordan and Rumelhart *ibid.*). Another disadvantage of this technique is that sampling to obtain the input-output pairs must be done in action space. It is often more desirable to sample in the areas of interest to the external system, *i.e.* in goal space.

3.2 Indirect Inverse Modeling

The Indirect Inverse Modeling technique involves the following steps:

- 1) A forward neural network model of the system is produced. This model is simply a neural network simulation of the external system. It is trained by obtaining input

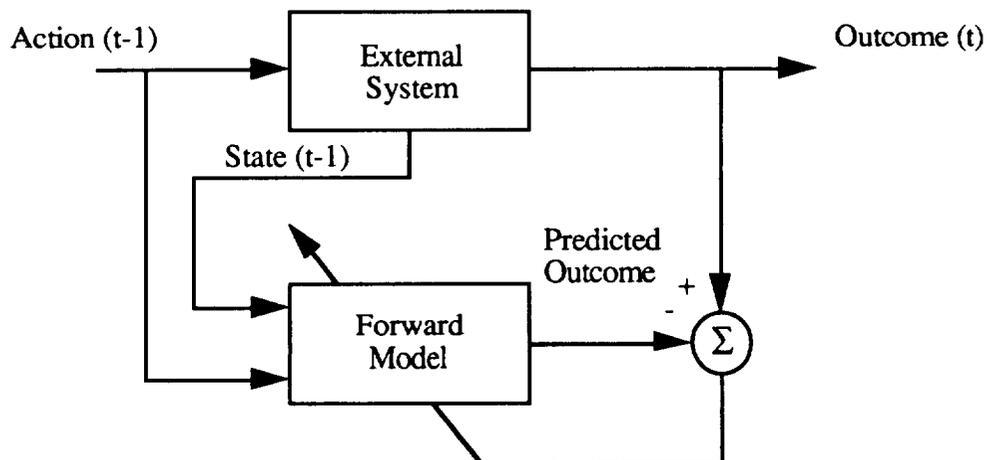


Figure 3. *Forward System Modeling. This is the first step required in the creation of an indirect inverse model.*

output pairs from the external system and by using them to train the neural net (Fig. 3). The algorithm typically used for training is Backpropagation (e.g. Rumelhart and McClelland, 1986). This simulation is not a reverse model: inputs and outputs of the system are also, respectively, inputs and target outputs of the neural net.

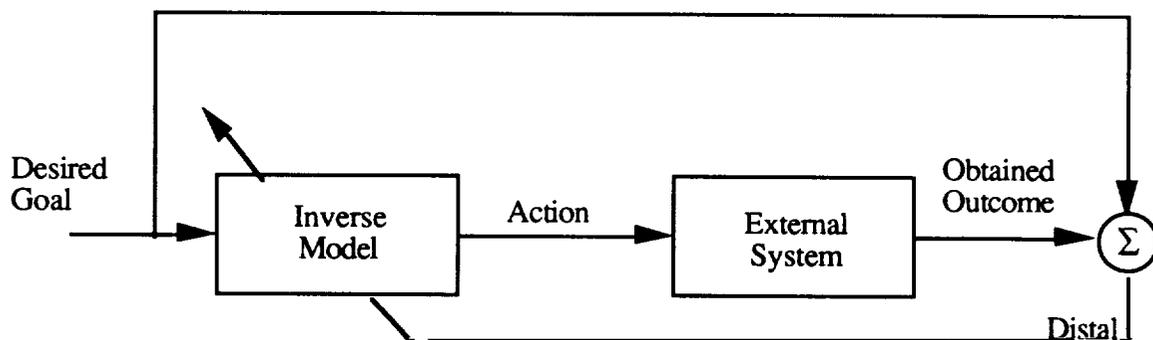
2) After the forward model has been adequately trained, another neural net, the one that will be trained to become an inverse model, is attached to the first one, so that its outputs become the inputs of the forward model.

3) Any desired system goals G are presented to this second net (inverse model). The output of this net is then presented to the external system to produce the actual outcome.

4) The actual outcome of the system is compared with the desired goal and an error is obtained. As pointed out earlier, this error is not meaningful to the model inversion net, which needs error signals in action space, but it is consistent with the output of the forward model, which is also in goal space.

5) Since the error determined in step 4 is in goal space instead of in action space, as required by the inverse model, the error signal is fed back through the forward model with techniques such as Backpropagation, until it reaches the output units of the inverse model. At this point the error is "meaningful" to the output units of the inverse modeling network and Backpropagation can proceed through this network in order to change the weights. Note that the weights of the forward model are not changed during this phase.

The Indirect Inverse Modeling process is shown in figure 4 and is explained in greater detail in Jordan and Rumelhart (*ibid*) and Jordan (1989). A version of this approach was used to train the "truck backer-upper" controller (Nguyen and Widrow, 1989). Training this controller required the evaluation of a sequence of



control actions rather than a single one, and the solution was obtained by unfolding steps 4 and 5 through time.

The advantages of Indirect Inverse Modeling are twofold: learning can proceed in a Goal Directed manner since the goals to be learned are presented at the input of the combined neural net, and the many-to-one mapping problem is obviated by the fact that a particular action will be chosen to achieve a desired goal.

A drawback is that a forward model must be created before training of the inverse model can take place. This drawback, however, is not as severe as it might seem because an approximate model is, generally, sufficient. In a sense, the role of the forward model is to act as an inverse transformation of the error from goal space to action space. From this point backwards, the inverse model learns to compensate for inaccuracies of the forward model.

The technique we propose, Goal Directed Model Inversion, extends this concept further by making a forward model unnecessary while retaining the desirable feature of being goal directed.

3. Goal Directed Model Inversion

We start by presenting an untrained multilayer network (for generality) with the desired goal G and the present state S of the external system, coded in some convenient way to provide the activation of the input layer. The output of the network is decoded into an action A that, when executed by the system, will produce the desired goal. Generally, the first action produced by an untrained network will not produce the desired goal, but an alternate outcome, G' .

Here is where our scheme differs from both Direct and Indirect Inverse Modeling: no error is fed back from goal space, instead, the goal G' that was actually reached is presented to the same network for a training pass. The output of this second pass, A' , is not executed by the external system and is used only to train the network in the following way: an error is determined by comparing A' with A (which takes on the role of a target output), and is used to modify the weights by some convenient algorithm such as Backpropagation. After the weights have been modified, the original G is presented to the network again, a new action A is produced and the cycle continues.

Learning ceases when $A = A'$. This is clearly the case when $G = G'$, but, unfortunately, it is also rather easy for the system to produce a net that will output $A = A'$ when G differs from G' . In our first version of GDMI this situation could be avoided in an "ad hoc" fashion by making the first goals very easy. The user had to note the position of the arm which was produced by the untrained system and had to place the very first goal close to it. After that goal was reached a second goal could be placed a bit farther, and so on. This process made it difficult to conduct a systematic study of the algorithm and we referred to this shortcoming as a "priming problem".

We have now found a better solution. When the above situation is encountered we simply do not allow the system to stop learning. If G differs from G' and $A = A'$ then A is perturbed to produce a new action A^* . Typically the second pass-through does

not produce the same A^* and the learning cycle begins again. Within the limits explained below, any goal for which a solution exists will be reached. This makes it possible to start systematic studies of this process.

We have no mathematical proof for the convergence of this system, but we are gathering empirical evidence that non-trivial solutions can usually be found. In these cases driving A' towards A drives G' towards G . Note that G remains constant, but A does not and provides a slowly shifting target. This process is illustrated in figure 5.

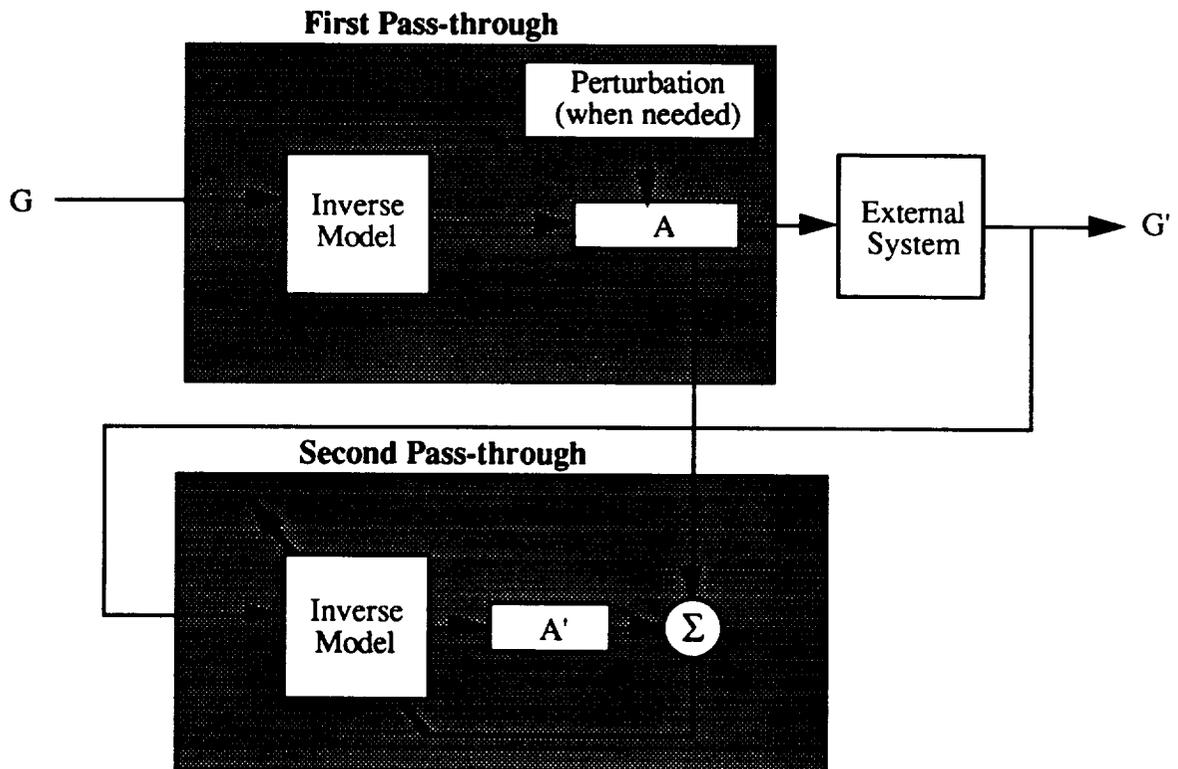


Figure 5. Goal Directed Model Inversion. The achieved output G' is presented

Typical "on-line" supervised learning makes similar assumptions. Error corrections due to incoming input-output pairs generally have the cumulative effect of lowering the error on future input-output pairs.

4. Preliminary experiments

We tested the GDMI process on a simulated planar robotic arm with three degrees of freedom. The goal G is specified as the position of the end effector in x, y coordinates and the action A specifies a set of angles for the arm joints. These angles are executed by the arm which moves the end effector to the corresponding position G' and, ultimately G .

4.1 Learning Behavior

Given any goal G in the space that is reachable by the arm, the system succeeds "almost always" (figure 6). By this we mean that typically the number of iterations required for success has some variability which depends on the number of hidden units (more on this below), but, occasionally (about one in one hundred times for this problem), the system appears to get trapped in a limit cycle or in a very long spiral towards the goal. This suggests including a "patience" parameter for providing a "jolt" in the same fashion as when the system stops learning before reaching the goal.

4.2 Effect of hidden layer

The number of units in the hidden level, not surprisingly, has considerable effect on the system's ability to learn. This was quantified in an experiment where goals A, B and then A again were given to the system consecutively, without changing the set of weights. This was designed to see 1) the effect of having achieved goal A on the ability to achieve goal B and 2) how much the system "forgot" about goal A in the process of achieving goal B .

The experiment shows that the system takes the longest and shows most variability while achieving the first goal. The second goal is then achieved more readily and with less variability. Finally, returning to the first goal shows that some "forgetting" occurs but a new set of correct weights is found again, with the least number of iterations and the least variability. The effect of increasing numbers of hidden units is to increase the speed with which the system achieves its goals and to decrease the variability (figure 7). Note that the graph for 45 hidden units shows an apparent contradiction of this trend in the standard deviation for the "return to A " (single asterisk in the figure). This large standard deviation was actually due to a single case in which the system was trapped in a very long spiral towards the goal, as explained above. The only other limit cycle was encountered in reaching goal A for the first time, with 15 hidden units (double asterisk in the figure). In this case the runs was interrupted and the point was not included in the calculations.

The general trend shown by the experiment was not surprising, but it is interesting to note that within the limits explored, the number of hidden units is not crucial to the ability to attain any particular goal. There is a trade-off between the number of hidden units and the speed and variability with which the goals are attained. This

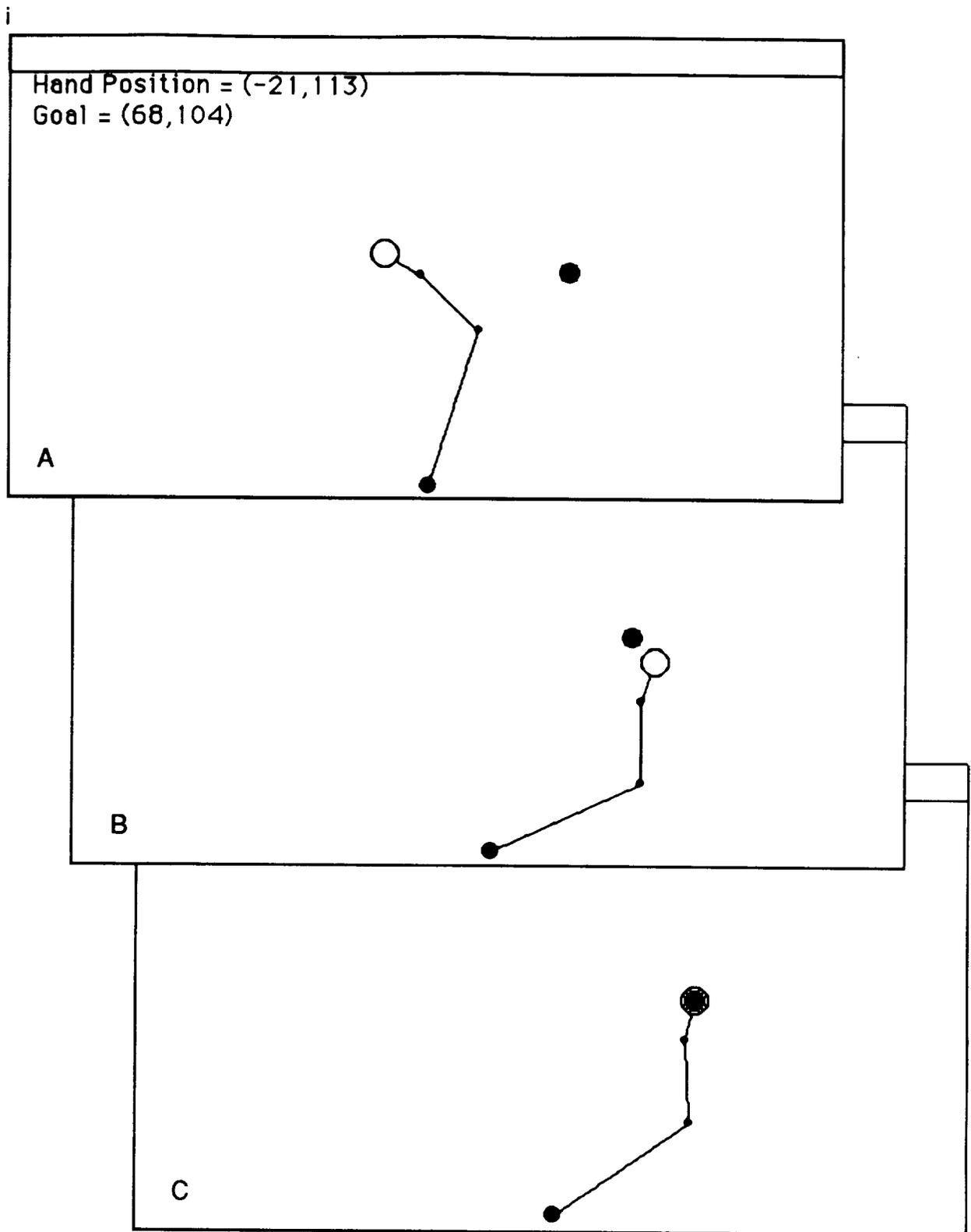


Figure 6. A typical learning sequence: A) starting condition, B) goal overshoot, C) goal achievement.

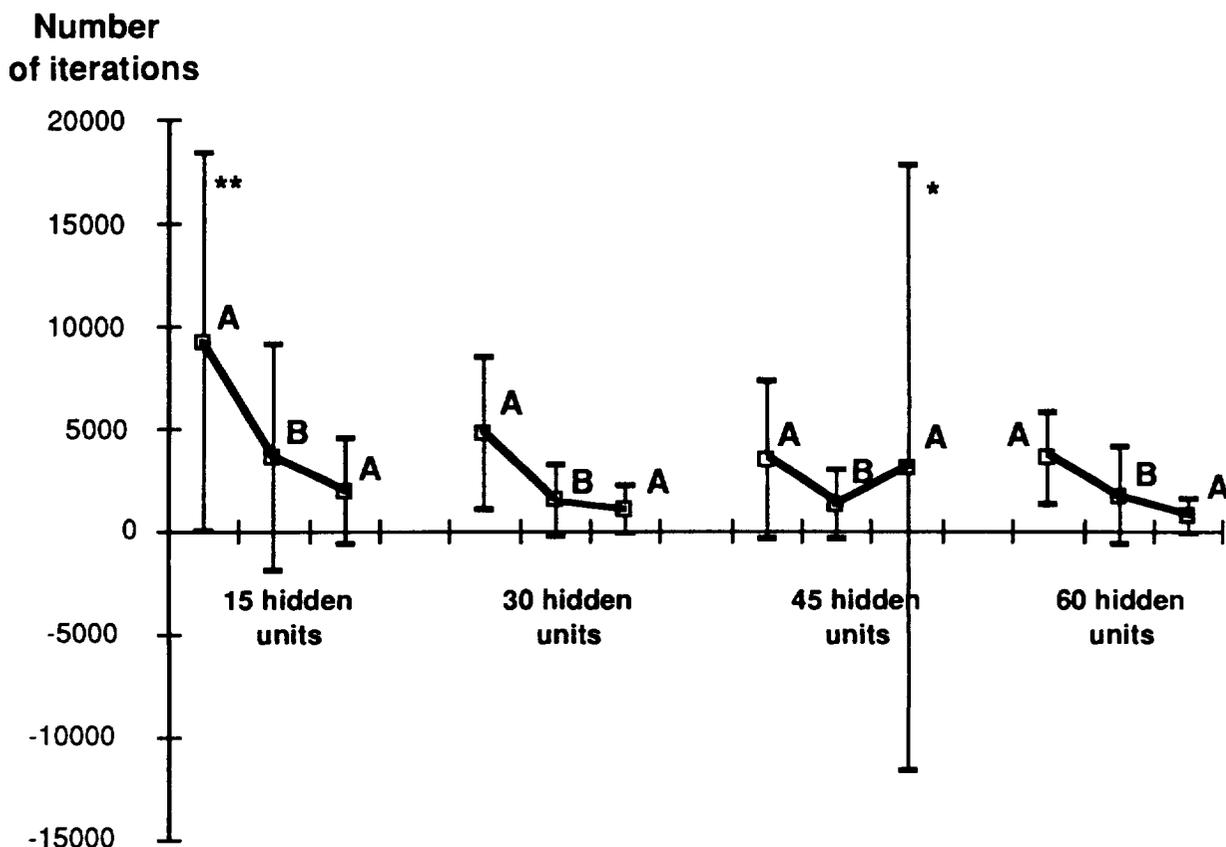


Figure 7. Learning to reach A, then B, then A again: Different numbers of hidden units affect the speed and variability of how quickly goals are reached. The asterisks are explained in the text.

ndicates that the systems is robust with respect to the ability to solve problems provided, at least for now, that time is not a crucial factor.

The simulations were run on a MacIntosh II computer, and the Backpropagation part of the GDMI system was adapted from the software provided in McClelland and Rumelhart 1989.

4.3 Adaptation to external system changes

GDMI requires no model of the external process. The implication of this fact is very powerful, since the external process is then allowed to change in unexpected ways. Under similar circumstances Indirect Model Inversion might instead require retraining of the forward model. We say "might" because Indirect Model Inversion only requires an approximate model of the external system, it can, therefore, cope with small changes. GDMI does not even need an approximate model. It builds a model of the process while it is learning to control it.

The potential importance of this property can be seen by "freezing" one of the joints of the simulated robot arm. If the frozen joint does not make a given goal

unreacheable, the system treats this case as anew problem and finds a solution as described above.

4.4 Multiple Constraints

There is also no limit (in principle) to the number of goals that can be simultaneously satisfied. This provides a mechanism for imposing any number of constraints. In our example these might relate to desired arm configurations and obstacle avoidance.

5. Future extensions

As already pointed out by Jordan and Rumelhart (*ibid*), Model Inversion is really a powerful generalization of supervised learning. Robotic arms provide good visual examples, but the concept applies to any type of transformation from the output of the network to the output of the external system. Guidance and control are obvious applications, but the concept can be extended even further. For example, the external system could be a simulation of a complex machine we wish to diagnose. Given specific faults it is relatively easy to determine, by simulation, what the effects or symptoms would be. The inverse is what we need for diagnosis: given the symptoms find the causes. With GDMI we could teach a net to produce causes given the symptoms. The symptoms become the system "goals", and the causes become the "actions". We are presently exploring these concepts.

In general this suggests a way to combine induction and deduction for problem solving. Note also that the supervised learning component of GDMI, which was implemented for expediency as a Backpropagation network, is certainly not limited to neural network implementations.

Another consideration is that learning in this sytem is "on line" , i.e. the learning component is never exposed to the same input data. This leads to the notion of learning by exploring the behavior of some unknown svstem without making

for the ability to reach the last given goal. GDMI was tested on a robotic simulation but its general applicability to a broad range of systems is suggested.

8. References

Barto, A.G., Sutton, R.S. and Anderson, C.W. "Neuronlike adaptive elements that can solve difficult learning control problems". *IEEE Transactions on Systems, Man and Cybernetics* SMC-13:834-846, 1983.

Berenji, H.R. and Khedar, P. "Learning and Tuning Fuzzy Logic Controllers Through Reinforcements". *IEEE Transactions on Neural Networks*, Vol 3, No 5:724-740, 1992

Colombano, S.P., Compton M. and Bualat, M. "Goal Directed Model Inversion: Adaptation to Unexpected Model Changes" *Proceedings to Neuro-Nimes '91*, 269-278, Nov. 1991.

Colombano, S.P., Compton M. and Bualat, M. "Goal Directed Model Inversion" *Proceedings to the International Joint Conference on Neural Networks* Vol III:2422-2427, Nov. 1991.

Jordan, M.I. and Rumelhart, D.E. "Forward models: Supervised learning with a distal teacher". *Occasional paper # 40, Center for Cognitive Science, MIT*, 1989.

Jordan, M.I. "Generic Constraints on Underspecified Target Trajectories". *Proceedings to the International Joint Conference on Neural Networks*, Vol I:217-225, 1989.

Jorgensen, C.C. "Distributed Memory Approaches for Robotic Neural Controllers". *RIACS Technical Report 90.29*. Research Institute for Advanced Computer Science, NASA-Ames, Moffett Field CA 94035, 1990.

Kohonen, T. "The Self-Organizing Map", *Proceedings of the IEEE*, Vol 78, No. 9:1464-1480, 1990.

McClelland, J.L. and Rumelhart, D.E. "Parallel Distributed Processing: Explorations in the Microstructure of Cognition, a Handbook of Models, Programs and Exercises", Vol 3, The MIT Press, 1989.

Nguyen, D. and Widrow, B. "The Truck Backer-Upper: An Example of Self-Learning in Neural Networks". *Proceedings to the International Joint Conference on Neural Networks*, Vol II:357-363, 1989.

Rumelhart, D.E. and McClelland, J.L. editors. "Parallel Distributed Processing: Explorations in the Microstructure of Cognition", Vol 1, Chapter 8. The MIT Press, 1986.

Williams, R.J. "Toward a Theory of Reinforcement - Learning Connectionist Systems". *Technical Report NU-CCS-88-3*, College of Computer Science, Northeastern University, 1988.

