Explorations on System Identification via Higher-Level Application of Adaptive-Critic Approximate Dynamic Programming

Joshua G. Hughes and George G. Lendaris

Abstract—In previous work it was shown that Adaptive-Critic-type Approximate Dynamic Programming could be applied in a “higher-level” way to create autonomous agents capable of using experience to discern context and select optimal, context-dependent control policies. Early experiments with this approach were based on full a priori knowledge of the system being monitored. The experiments reported in this paper, using small neural networks representing families of mappings, were designed to explore what happens when knowledge of the system is less precise. Results of these experiments show that agents trained with this approach perform well when subject to even large amounts of noise or when employing (slightly) imperfect models. The results also suggest that aspects of this method of context discernment are consistent with our intuition about human learning. The insights gained from these explorations can be used to guide further efforts for developing this approach into a general methodology for solving arbitrary identification and control problems.

I. INTRODUCTION

This paper reports on extensions of the approach described in [1][2][3] that apply Adaptive-Critic-type Approximate Dynamic Programming (ADP) in a novel way (at what is called a “higher level”) to arrive at a strategy for an autonomous agent to select an optimal controller from a collection of previously designed controllers, each optimal for a specified context. This is in contrast to the more usual application of ADP to design an optimal controller for a given context. The term context is defined to comprise three dimensions: the plant that is being controlled, its environment, and the optimality/performance criterion function. The approach presupposes that an agent has previously encountered a variety of distinct contexts for a given problem domain and for each, has gone through a process of developing an optimal control strategy (called ‘control policy’ in the Dynamic Programming literature). It is assumed that the agent has a way of saving these optimal control policies, with each indexed by the specific context for which it is optimal. It is next assumed that in the interest of efficiency, rather than designing a control strategy anew each time a previously experienced context is encountered, the agent saves each control policy as it is developed into a kind of library/repository and then selects a control policy from this repository each time the context changes. For this approach to be useful, the process of selecting the appropriate previously designed control policy is itself to be optimized, defined in terms of speed of selection (efficiency) and effectiveness of the selected control policy. The Adaptive-Critic-type of ADP is employed to develop such an optimal selection strategy.

A key precursor to the act of selecting a control policy is that the agent must be aware of the present context and discern when the context changes (such changes may occur in the plant itself, the environment, the criterion function, or any combination thereof). There are clearly multi-level aspects to such a decision environment. (See [2] for further discussion of this aspect.) At its current state, the present research focuses on what is deemed the first level up from the “usual” one, and in particular, focuses on the plant component of the context, with a fixed environment and criterion function assumed. Two tasks are addressed: system identification (by which the agent discerns the current plant status) [covered in the present paper], and representation of control policies in a way that facilitates selection once the system identification has been accomplished [to be covered in a future paper]. An important aspect in each task is whether the method allows for reasonable “interpolation”—i.e., if the presently discerned context is “near” a previously encountered one, might the corresponding selection be “near” an existing item in the repository, and if so, how to interpolate the selection, be it a model for the systems identification process, or a policy for the control application?

Space limitations preclude very much review of prior work. To summarize, however, the journal papers [1][2] contain the underlying ideas and rationale for the proposed approach to development of autonomous agents capable of performing context discernment and context-dependent control. Guiding motivation is provided by two features of humanlike control: 1) making use of experience while selecting a control policy for distinct situations, and 2) the ability to do so faster and faster as more experience is gained (in contrast to current technological implementations that slow down as more knowledge is stored). The above references include results of early experiments and cite the fact that those early experiments were based on full a priori knowledge of the structure of the plant/model. Future explorations were suggested wherein the process be subject to less precise knowledge of the plant/model structure, that the process be subject to noise, and that adequate representation schemes are developed for indexing and accessing items.
from the library/repository, both for systems identification and for controller selection. This paper describes experiments that explore some aspects of the suggested issues as they relate to the context discernment (specifically, system identification) part of the process.

A high-level representation of the process is depicted in Fig. 1. The plant that is being controlled is monitored to determine whether its input/output characteristics have changed, and if so, system identification (SysID) is undertaken to determine what it has changed to. In Fig 1, the SysID activity is depicted as a Self-Adjustable Model. From the perspective of the controller (not shown in Fig. 1), the plant status is one of the components of the context in which the controller is operating; it is therefore important to monitor changes in the plant, which entails a system identification (SysID) process, here called Context Discernment.

**Figure 1.** High-level representation of the system identification process. The Self-Adjustable Model monitors the input and output of the Plant to determine whether or not the Plant has changed and, if it has, what it has changed to.

**Figure 2.** Conceptual representation of the Self-Adjustable Model of Fig 1. The context discerner (CD) provides the parameter values \( p \) (‘selector input’) that instantiate a specific mapping in the parameterized-model box. After the CD has learned a family of mappings, it selects a specific mapping based on a measure of the difference between model’s output with that of the plant being observed. The CD is trained via an Adaptive-Critic-type of Approximate Dynamic Programming approach (not shown).

There are two key components of the Self-Adjustable Model of Fig. 1. With reference to Fig. 2, there is a component “box” with inputs and outputs that performs a mapping (function) from its inputs to the outputs. This box contains a Parameterized Model that performs the mapping and has an additional input called a ‘selector input’ \( p \) that sets values of the Model’s parameters, with the effect of specifying the particular input-output mapping being implemented by the box. In addition, there is a “box” whose purpose is to provide the selector input \( p \) to the parameterized model (the first box) and is called the Context Discerner (CD). Its role is to discern the current input/output characteristics of the Plant being controlled and to select the mapping within the parameterized model box that represents those characteristics. The CD is trained to perform its discerning task via an Adaptive-Critic-type of Approximate Dynamic Programming. Details of this training process are given in [3], and a more general description of ADP operation in [7].

### II. OUTLINE OF CURRENT EXPLORATIONS

The objective of the work reported here has been to perform preliminary explorations into the effects of noise on the Context Discernment process, and in addition, the effect of employing a (slightly) inaccurate parameterized model to represent the plant being controlled.

To provide for an orderly exploration, the Plant was set up as a feedforward neural network (NN) of a given connection pattern, and the parameterized model (the Model) was set up with a NN of identical structure and weight assignments. For each set of experiments a different instantiation of weight assignments was made. The weights on the biases were employed as adjustable parameters (selector inputs), while all other weights were fixed. Conceptually, to each instantiation of fixed weight values there corresponds a family of mappings that can be performed by the NNs, where different members of the family are instantiated by assigning different values for the adjustable parameters (in this case the bias weights, which have the effect of shifting the hidden-node activation functions). Thus, the task of the Context Discerner box of Fig. 2 was to determine the correct values for the Adjustable Weight Vector (AWV) \( p \) to enable the CD to select the correct model from the parameterized model box.

A number of different experiments were run via different instantiations of the fixed weights, thus providing different families of mappings to explore. Two example families are given in Figs. 3 and 4. The Plant for these experiments had two inputs and one output. This choice was made to facilitate the representation employed in Figs. 3 and 4, namely, the bottom plane representing the two inputs \( x_1 \) and \( x_2 \), and the vertical plane representing the NN’s output \( y \). (At this stage of exploration, visualization of the discernment process has been very useful.) Each surface depicted in the two figures corresponds to different values in the AWV. Some of the qualitative differences are easy to notice in Figs. 3 and 4. These are discussed in the next section.

The process of context discernment for a trained CD is illustrated in Figs. 3 and 4. (Both figures are based upon results from experiments discussed in the following sections). Given inputs \( x_1 \) and \( x_2 \) and an initial (random) AWV \( p \) to the Model, the Model generates an output \( \hat{y} \) (the black square in both figures). The target output, \( y^* \), for a given input-vector value is calculated via the Plant, which is instantiating a member of the family of mappings associated with its current weight settings, determined by the Plant’s current AWV. The target output for the given input-vector value is a point on this surface. When the error measure between the Model and Plant outputs exceeds a specified threshold the CD updates its guess of the AWV, employing i) this error-measure value, ii) the CD’s last guess for the AWV (context), and iii) the Plant’s inputs. Within a few time steps the error-measure value becomes close enough to zero (according to some threshold value) that the context discernment...
process stops, and the CD is considered to have identified the Plant. After a few more time steps in the experiments reported in the next sections, the Plant is changed again, i.e., a new adjustable weight vector (AWV) is instantiated. Since the CD has not yet received any information to signal that the context has changed, it does not update the AWV and so the Model output is (momentarily) incorrect; however, at the next time step the CD receives an error-measure value which triggers the discernment process once again, and within a few steps the CD successfully identifies the new Plant.

![Figure 3](image3.png)

**Figure 3.** Representation of a family of mappings implemented by one instantiation of fixed weight values. The three indicated surfaces correspond to three different adjustable weight vector (AWV) values. This family of mappings is characterized as “high volume, high tilt.” This characterization and the trajectories on the surfaces are described in Section III.

![Figure 4](image4.png)

**Figure 4.** Representation of a family of mappings implemented by an instantiation of weight values different from those instantiated for the mappings in Fig. 3. This family of mappings is characterized as “low volume, low tilt.” The three indicated surfaces were generated with the same three adjustable weight vectors (AWVs) used to generate the surfaces in Fig. 3. This characterization and the trajectories on the surfaces are described in Section III.

### III. EXPERIMENTS

**A. Experimental Design**

In crafting the experimental design, a variety of considerations are important and comprise at least the following: i) the overall system structure (including that of the Model and the Context Discerner), ii) the inputs provided to the context discerner, iii) whether or not the available observations of the system are noisy, iv) the training and testing syllabi, and v) the measures of performance. In this endeavor we are guided by our understanding and intuition about how humans learn and develop experience. While the experimental results that follow are based on a relatively simple formulation of the system identification component of the context discernment problem, formulation of the experiments was guided by the associated desire to have this methodology eventually applicable to arbitrary control problems, of which system identification is the first component.

A first step is to decide what types of plants/systems to focus on. Given that certain neural network paradigms are universal function approximators, these were deemed a good starting point. Although the context discernment methodology is intended to be able to identify any type of input-output system, neural network models are used in the present experiments because they provide known plant structures on which models can be based. A model may be capable of representing a system perfectly, but generally it only need be good enough to provide the information needed to perform the task at hand. In humans, it is believed that models are continuously improved as learning progresses, up to a degree required by the problem domain. For example, initial driving lessons may be given on an isolated driving course to provide the new driver a safe environment in which to develop a starting model, which is then continuously improved as the student driver moves out of the driving course and encounters “real world” driving conditions. The actual model that is developed depends on whether the driving is to be performed in a rural area, a congested metropolitan area, or a race track.

For the experiments reported here, the models employed had structures either identical or nearly identical to the plants being identified.

Mathematically, a feedforward neural network provides a mapping from input to output. Once the type of element is selected, the NN’s structure is fixed, and the weights and biases are specified, the mapping performed by the NN is determined. The mapping can be changed by changing any of these aspects. For example, varying the biases can change the mapping performed by the NN while maintaining the same structure and weights. For each instantiation of the adjustable weight vector, which in the present case specifies the bias values, there corresponds a particular mapping that is performed by the given NN structure. This is illustrated by the multiple surfaces (mappings) in Figs. 3 and 4. The surfaces depicted here represent three distinct mappings from the family of mappings the given NN is capable of performing; the particular instantiations were determined by three different values for the adjustable weight vector (AWV).

The Plant is represented by such an NN for the experiments reported here. The NN employed comprised two hidden nodes, and specification of their biases is given via a 2-dimensional AWV. Different values for the AWV result in different plant instantiations, and hence different contexts from the perspective of the agent whose task is to select an appropriate controller as changes in plant I/O characteristics (changes in context) are detected. For simplicity, the two hidden nodes were assigned identical adjustable weights.

With this particular formulation of the Plant, the system identification task (context discernment) is one of determining the appropriate AWV value for the plant instantiation.
that is currently generating the input/output mapping. In the ideal case the context-discriminating agent will select the correct AWV from the experience repository and do so in an efficient manner. In the present line of research, the agent is designed (trained) via the Adaptive-Critic-type of Approximate Dynamic Programming (ADP) method to implement a (near-optimal) selection strategy.

One of the key challenges in setting up this context discernment process is representation. How can the different context components be represented to create a natural “indexing” schema that the agent can use to effect the selection of context, and further, that includes the notion of “nearness” mentioned earlier? Another challenge in developing this methodology is the fact that, in practice, whether or not the agent can learn to optimally select a mapping from the repository (represented here as a family of mappings) is affected not only by the particular mappings but also by the training process, in particular, the “training syllabus” employed. When a good selection performance is not achieved, it is not always clear whether this failure was due to the difficulty of the problem or to sub-optimal selection of ADP values for the training process.

To explore such issues, it would be useful to develop a method for characterizing a family of mappings. There no doubt exists a large variety of possible representations and measures that might be useful for such purposes. In our explorations, two simple measures and corresponding representation approaches emerged. The measures are here called “volume” and “tilt.” A vector space is defined via the Plant’s inputs, outputs, and AWV values. Outputs are plotted in this space for all possible values of inputs and AWVs. The portion of the space that is circumscribed by the resulting plot is said to have a “volume” and a “tilt” in that vector space. The volume measure characterizes how much of this space is occupied by the family of mappings; the tilt measure characterizes how the family of mappings is oriented within this space. (Refer to Figs. 3 and 4 for examples.) Both these measures are crude; it is not suggested here that these measures are universal or applicable to all problems. Rather they serve to characterize the specific problems explored here in a way that allowed us to gain insight into the performance of the context discerner and how we might adjust the training syllabus to improve performance. For other families of mappings, other means and measures may be more useful.

That context (plant characteristics) can change without having a significant effect on the output is an important consideration for performance. This brings to light the idea that it is not so much an issue of the Plant characteristics changing, per se, but rather an issue of magnitude of divergence between the Plant’s output and the Model’s output (recall that the agent employs the Model to select the controller). The Plant characteristics have to have changed “enough” to appreciably affect the output mapping to be of interest. To this end, three measures of performance were developed for the present experiments: i) operation within threshold, ii) root mean square (RMS) error, and iii) recovery speed. Operation within threshold represents the percentage of time during an observation window that the current model gene-

rates an output within some tolerance—an arbitrary, acceptable threshold. RMS error is the square root of the squared errors averaged over the entire test. Recovery speed is the amount of time (number of time steps) it takes for the discerner to recognize that the context has changed and select a new Model that operates within threshold once again.

Based on these measures, different approaches can be used to determine whether or not to select a new context. Having set forth the idea that for human-like control the agent must “detect large enough changes” before selecting a new policy, for the current experiments context was updated only when the difference between Model and Plant outputs was outside a set threshold.

We again drew upon our intuition and knowledge of human learning to inform design of the “training syllabus.” In general, it is not possible to train for all possible contexts; nevertheless, the goal is to create agents that generalize well, that is, perform well in contexts never before encountered. Humans can often perform near-optimally in new contexts by drawing upon experience from similar contexts.

Two main methods for exploring different mappings/surfaces during training of the Context Discerner exist: random or organized. In the random case, some subset of contexts (adjustable weight vectors AWVs) is selected randomly; each is instantiated as the Plant to be modeled and is employed for some number of training iterations. Each AWV may be selected multiple times during the training process. In the organized case, the AWVs (contexts) are selected methodically and purposefully. Within this organized case, two possibilities exist: either a defined sequence of different AWVs is explored, or after each is explored for a designated number of training iterations the next AWV to be explored is based on the previous performance. In terms of human learning, one could imagine an instructor employing a strategy of presenting a pupil with a pre-defined set of exemplars, or one of presenting “easy” exemplars first and then increasing and/or decreasing the difficulty over time depending on the pupil’s performance. Other organized approaches are also possible. Both random and organized approaches to training were used in the experiments discussed below. As with any neural network training algorithm, the learning rate, momentum, and other learning parameters must also be considered.

The experiments reported here were run on the software platform called ONPAK (Ordered Network Package), a neural network package developed at the Northwest Computational Laboratory (NCWCL) at Portland State University. The ONPAK Matlab code set and additional scripts were used to create the necessary components and implement the context discernment process described in [7][8]. Multilayer Perceptrons filled the Plant, Model, CD, and Critic roles. The actual architecture used for all system identification experiments is shown in Figure 5. The Plant and Model both have two inputs, \(x_1 \) and \(x_2 \), and a single output node, plus a single selector input \(p^* \) or \(p \) (adjustable weight vector AWV) used to tune the biases of the two hidden nodes. Both the plant and Model have identical, random, fixed weights for all connections except those on the connections from the selec-
tor input to the hidden nodes, which are fixed at a value of 1. The Model is created by making an exact copy of the Plant. The CD and Critic each have a single hidden node, and the connection weights are adjusted during the Adaptive-Critic-type ADP training process. The Critic trains both the CD and itself based on the Critic’s outputs calculated from the input, discerned context, an output error measure, and the CD output. This architecture was used for all experiments. A bias node (not shown) with a constant input of 1 is also connected to the output node in all four networks. In the Plant and Model the output bias weight is random but fixed; in the CD and Critic the output bias weight is adjusted until training is completed.

\[ x_1, x_2 \]

\[ p^* \text{ or } p \]

\[ \lambda \]

\[ AC-type \]

\[ \text{ADP} \]

\[ \text{CD} \]

\[ \text{Plant} \]

\[ \text{Model} \]

\[ \text{Critic} \]

\[ \text{input} \]

\[ \text{output} \]

\[ \text{bias} \]

\[ \text{weights} \]

\[ \text{epochs} \]

\[ \text{learning rate} \]

\[ \text{noise} \]

\[ \text{AWV} \]

\[ \text{threshold} \]

\[ \text{RMS} \]

\[ \text{speed} \]

\[ \text{data} \]

\[ \text{weights} \]

\[ \text{bias weight} \]

\[ \text{context} \]

\[ \text{parameters} \]

\[ \text{training} \]

\[ \text{AWVs} \]

\[ \text{represents standard neurons that sum inputs and apply a hyperbolic tangent activation function. The Plant and Model both have two inputs, } x_1 \text{ and } x_2, \text{ a single selector input (the Plant or discerned context), } p^* \text{ or } p, \text{ used to tune the biases of the two hidden nodes, and a single output node. Both the Plant and Model have identical, random, fixed weights for all connections; the weights for the selector inputs are set equal to 1. The CD and Critic each have a single hidden node. The Critic’s outputs, the } \lambda \text{-vector, calculated from the input, discerned context, output error, and the CD output, are used to train the CD and the Critic. An output bias node (not shown) with a constant input of 1 is connected to the output units in all four networks. In the Plant and Model this output bias weight is random but fixed; in the CD and Critic this output bias weight is adjusted until training is completed. The “cloud” indication in the upper right is a proxy for the ADP process, not discussed herein.} \]

A number of preliminary experiments were run to confirm some general principles about training such systems prior to setting out the final experiments. In particular, preliminary experiments focused on determining reasonable values for the learning rate and number of training epochs. It is generally acknowledged in the neural network literature that, for a given problem, there exists a “sweet spot” at which the best results are consistently obtained. For these preliminary experiments, learning rates of 0.01, 0.05, and 0.25 were used, with training epochs of 500, 2,500 and 10,000, along with variable amounts of adjustable weight vectors (AWVs), } x\text{-vector (Plant input) presentations, and measurement noise. The results of these preliminary experiments suggested that training with a learning rate of 0.05 for 2,500 epochs consistently produced good results for a wide range of two-input, one-output Plants with varying amounts of noise. With the lower and higher learning rates, comparable results were sometimes obtained, but often results were worse. With a learning rate of 0.05 for 10,000 epochs slightly superior results were sometimes obtained, but not often enough to warrant the much higher computational expense. It is not here implied that a learning rate of 0.05 for 2,500 epochs is the ideal configuration for the context discernment methodology (much more training will likely be necessary for more complex problems), but rather that these training parameters worked well for the present experiments. For all experiments reported here, the learning rate was 0.05 and the number of training epochs was 2,500. For each experiment, either 10 or 50 settings of the AWVs, } p^* \in [-1,1], \text{ were used during training. In experiments with 10 settings, each setting was instantiated in the Plant for either 10 or 50 learning iterations (input-output presentations); for experiments with 50 settings, each setting was instantiated for only 10 consecutive learning iterations at a time. In other words, an AWV setting was selected, either randomly or according to some organized training scheme, and instantiated in the Plant for a specified number of learning iterations, then another setting was selected (with replacement) and presented for the same number of iterations, and so on. At each iteration, a random input pair } (x_1, x_2), x_1, x_2 \in [-1,1] \text{ was used to generate an output from the Plant based on its current (fixed) value of the AWV, and that same input pair was also presented to the Model. The CD received i) the same inputs as the Plant, ii) the current discerned context, and iii) the absolute error between the Plant and Model outputs. The Plant always received noise-free inputs, but for the experiments with noise both the Model and CD received the inputs with noise added (to represent measurement noise); in some cases the CD also received error-measure values with noise added. The noise distribution had a mean of 0, a standard deviation of 0.33, and was multiplied by a factor of 0.10, 0.25, or 0.50 before being added to the input or error values. Because the input and output values were scaled to (-1,1), this noise distribution could significantly alter the data received by the CD. After training, the Context Discerners were tested on a subset of five settings used during training to see if they had been adequately trained. The ability of the CD to discern context (i.e., select the correct model via its context parameter AWV) was then measured using two generalization tests: 1) performance relative to a specified sequence of five discrete context changes and 2) performance relative to a slowly varying context (a sine wave). These two tests are here referred to as “fast” and “slow” tests. Performance was evaluated quantitatively in terms of the percentage of time the Model operated within threshold, RMS error, and speed of recovery. The threshold was set at a squared error of 0.01 for all experiments.}
The experiments were designed with the intent that the results would be able to provide insights useful for development of the context discernment methodology. So while the results can be used to evaluate the performance of the methodology for the present system identification tasks, they also provide guidance for how to best apply the methodology and what general approaches might be used to improve its performance. In the discussion that follows, we focus on these broader methodological issues.

B. Experimental Results

1) General Observations

The initial runs comprised 1,350 experiments with 50 different Plant “sets” (50 unique sets of weight assignments) and served to explore questions related to training schemes and noisy inputs. Context discerners were successfully trained to perform system identification for the 50 different Plant sets under a variety of conditions. Additional experiments, performed after reviewing results of the initial experiments, showed good performance could also be achieved when noise was introduced into the error-measure calculations and when (in the present exploration, only slightly) imperfect models were used for context discernment.

Training performance was a very good predictor of generalization. Performance on both generalization tests (the fast and slow tests) was similar to training performance for all measures (operation within threshold, RMS error, and recovery speed); if training performance was poor (as it was in a few cases), generalization was also poor, and when training was successful, generalization was as well. Results on the fast test were usually slightly better than on the slow test. This was likely due to the nature of the tests themselves: the fast tests included only five context changes, while the slow test used a continuously varying context; due to what may be called quantization (via the threshold employed for the error measure) the context discernment lags behind. Specific details regarding performance are discussed below.

In terms of average performance (both operation within threshold and RMS error) and the number of performances exceeding 90% within threshold, use of the 10-50 scheme (10 AWV settings each instantiated for 50 consecutive iterations) for training achieved the best-performing context discerners. Based on the results it appears that having more consecutive presentations for each setting during training (equivalent to more sample points on each selected surface, if visualized as in Figs. 3 and 4) typically improved performance. In both the 10-10 scheme and the 10-50 scheme, each specific mapping is sampled, on average, for 250 of the 2,500 total training iterations. The only difference between these two training schemes is how many input/output exemplars are presented before a new AWV setting is selected. Using an insufficient number of exemplars can be characterized as “under-training”—the context discerner does not learn enough about the specific mapping before the trainer moves it onto another one. Results for the 50-10 scheme were very similar to the results for the 10-10 scheme. This suggests that given limited training time (in this case just 2,500 total training iterations), increasing the number of consecutive training presentations may be more fruitful than increasing the number of different settings.

On the other hand, examples of the worst performances also occurred with the 10-50 training scheme. This suggests that in some cases the Context Discerner was “over-trained” using an inadequate set of training examples. As with many other machine learning methods, just what constitutes a good set of exemplars and how long and how often they should be presented is unclear. A few supplemental experiments with organized presentation of non-random, purposely selected AWV settings presented in either particular or random orders generated much worse results than the main experiments with randomly generated settings presented in random order. Choosing a “good” set of exemplars is a non-trivial task—even for the present seemingly straightforward, low-dimensional system-identification task. Future research must necessarily explore this aspect of the methodology.

![Figure 6](image-url)

**Figure 6.** Performance compared to the characteristics of the system, volume and tilt, for all 50 target networks and for the (a) 10-10 and (b) 10-50 training schemes. The sizes of the circles represent the percent operation outside threshold for the experiments—smaller circles represent better performance—with both noisy and noise-free inputs. In the noisy cases shown, the testing noise was equal to the training noise. Performance is related to volume and tilt, the training scheme, and amount of noise. Note that, in general, the error increases with increases in noise.
Performance is also affected by characteristics of the plants themselves, here characterized as volume and tilt. Plots of test performance with respect to volume and tilt are shown in Fig. 6 for the (a) 10-10 and (b) 10-50 training schemes. (As noted above, results for the 50-10 scheme were very similar to those for the 10-10 scheme; in the interest of space a plot for these is omitted.) Circle size represents the percentage of operation outside threshold based on a squared error of 0.01 between the Plant and Model outputs; smaller circles represent better performance, larger circles worse performance.

In general, performance decreased as volume and tilt increased, regardless of whether measurement noise was present or not. For “low-volume” problems changes in context within that volume yield plants whose output values are very similar and thus lower corresponding error-measure values and operation outside threshold; for “high-volume” problems changes in context yield plants whose output values vary more and thus increase error-measure values and operation outside threshold. For plants with very low volumes and very low tilts, operation within threshold was (or was very close to) 100%—indicated by the very small circles in the lower left corner of both plots in Fig. 6—even with noisy inputs.

Comparison of the results using the 10-10 scheme with those of the 10-50 scheme provides additional insights into the relationship between training and the problem characteristics. Evident from comparison of the two-schemes is that performance on low-volume problems can be improved by increasing the number of consecutive x-vector presentations for each example mapping within the family of mappings. Because the individual mappings are so closely spaced, more refined training—e.g., more example points from each surface—is needed, otherwise it is difficult (if not impossible) for the CD to distinguish between very close mappings, i.e., under-training may be a greater concern for low-volume problems. The specifics of a particular problem will likely dictate how fine the resolution in the context discernment process is needed to generate adequate performance, and by which metric that performance will be evaluated.

2) Noisy Measurement Data

Performance on noise-free data was the best. When measurement noise was introduced, performance was degraded in proportion to the amount of noise. A selection of results using the 10-50 training scheme (the best-performing scheme) is summarized in Table I. In Fig. 6, the relationship between performance and noise is represented by the sets of three concentric circles, each set representing tests on the same family of mappings with different amounts of noise (0, 0.25, or 0.50) added to the Plant’s input values seen by the Model and CD. In most cases, the operation outside threshold increased as noise increased. This relationship also reinforces the likelihood that the (four) very poor performances observed when using the 10-50 scheme (Fig. 6b) were due, at least in part, to over-training with an inadequate set of AWV settings since in all four cases better performance was observed with higher amounts of noise.

Interestingly, test performance did not seem to be significantly affected by whether or not the x-vector (Plant input) values provided to the Model and CD were noisy during training. Inspection of the weights for the trained CDs revealed that the weights on the connections between the input nodes and hidden node were close to zero and several orders of magnitude smaller than the weights for the current-context and error-measure inputs to the CD. In other words, in most of these initial experiments (with or without noisy inputs), the CDs had learned to select the AWV values primarily via the information from the current context and error measure.

Table I: Summary of performance (operation within threshold) for selected experiments using the 10-50 training scheme.

<table>
<thead>
<tr>
<th>experiments</th>
<th>train</th>
<th>fast test</th>
<th>slow test</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>89.9</td>
<td>88.5</td>
<td>86.9</td>
</tr>
<tr>
<td>Train noise-free, test noise-free.</td>
<td>96.7</td>
<td>92.6</td>
<td>89.9</td>
</tr>
<tr>
<td>Train w/ noisy input, test noise-free.</td>
<td>93.8</td>
<td>91.2</td>
<td>88.3</td>
</tr>
<tr>
<td>Train noise-free, test w/ noisy input.</td>
<td>87.1</td>
<td>86.9</td>
<td>86.2</td>
</tr>
<tr>
<td>Train w/ noisy input, test w/ noisy input.</td>
<td>87.6</td>
<td>86.1</td>
<td>85.0</td>
</tr>
<tr>
<td>Train w/ low noisy output, test w/ low noisy output.</td>
<td>90.1</td>
<td>88.3</td>
<td>85.9</td>
</tr>
<tr>
<td>Train w/ high noisy output, test w/ high noisy output.</td>
<td>64.9</td>
<td>63.1</td>
<td>63.9</td>
</tr>
<tr>
<td>Imperfect model: train noise-free, test noise-free.</td>
<td>87.1</td>
<td>83.6</td>
<td>80.8</td>
</tr>
</tbody>
</table>

To investigate these results further, additional runs with noisy error measurements were made, and we observed that adding noise to the error measure had more negative effect on performance than did adding noise to the Plant-input observations. With 0.25 noise added to the latter, operation within threshold was still above 90% two-thirds to three-quarters of the time, yet not one CD was able to achieve 90% operation within threshold when the same amount of noise was added to the error-measure values. With just 0.10 noise added to the error-measure values, 90% operation within threshold was achieved more than half the time.

These results were not surprising since results of the prior experiments showed that the error measure provided more useful information to the CD than the input values did. However, when CDs were trained and tested with 0.25 noisy error measures, operation within threshold improved by more than 5% for more than a third of the experiments. This improvement was primarily observed for the higher level of error noise, and adding noise to the inputs reduced this improvement. Examination of the trained CDs showed that, at least in some cases, the weights connecting the Plant-input nodes to the hidden layer were often an order of magnitude higher than those in CDs that had been trained without noisy error measures. Based on these results, we offer the conjecture that when the CD was trained with high levels of error noise it learned to extract additional information from the input data, information that was not necessary when the error measure was only slightly noisy or noise-free.

3) Imperfect Model

Performance was also degraded by use of an imperfect model, in this case a Model that used a linear activation
function in its output node instead of the hyperbolic tangent activation function present in the Plant’s output node. For some problems, performance degraded little, but for others it degraded significantly. On average, operation within threshold was reduced by about 10%， although for some cases the reduction was much higher. However, the average increase in RMS error was only 33%， and only in a few experiments did the RMS error more than double. An example of results using an imperfect model is shown in Fig. 7. The Plant used for this experiment was identical to that used to produce Fig. 3. Comparison of the two figures confirms that the performance of the CD was degraded by use of an imperfect model, but the CD still performs reasonably well given this limitation and the difficulty of the problem. One might suggest that the CD performed “as best it could.”

Figure 7. Representation of the context discernment process for the same family of mappings as that shown in Fig. 3, but using an imperfect model. For the results shown in this figure, the Model employed a linear activation function instead of the hyperbolic tangent activation function present in the Plant (weights were still identical). Comparison of the two figures shows that use of an imperfect model degrades performance. However, although the error is greater, the CD still performs reasonably well. One might suggest that the CD performed “as best it could” with an imperfect model.

4) Insights gained from Experiments

Analysis of these results suggests at least four insights that can be gained from these experiments: 1) how exemplars are selected and presented during training can significantly affect performance; 2) the characteristics of the plant/system to be identified affects the context discerner’s ability to identify it and/or may require more refined training schemes; 3) while measurement noise degrades performance, performance may still be satisfactory for even large amounts of noise; and 4) using an imperfect model may also degrade performance for some plants/systems, but performance may be satisfactory. Regarding the last two items, when measurement noise is present or an imperfect model is employed it may be necessary to tolerate worse performance unless/until noise can be reduced or a better model can be developed.

It is interesting to compare these four insights to our intuition about human learning: 1) how a human learns a task affects the human’s ability to perform that task, 2) the characteristics of a task may make it more or less difficult to perform, 3) a human's performance on a task suffers when measurements (perceptions) are not accurate, and 4) when a human does not have an accurate model of a task, how well the task can be performed is limited. That these experiments confirm our intuition about human learning suggests that our context discernment methodology has at least some analogues with it. We certainly do not mean to imply that with these experiments we have identified the mechanisms by which humans learn, merely that these results deserve some thought and offer direction for further research.

IV. Conclusion

Adaptive-Critic-type Approximate Dynamic Programming is being applied in a “higher-level” way to create autonomous agents capable of using experience to discern context and select optimal, context-dependent control policies. The experimental results reported here show that this approach can be robust and adaptive when performing system identification tasks on small neural networks capable of representing different families of mappings. Agents trained using this approach can achieve high levels of performance when subject to even large amounts of noise and perform reasonably well when employing imperfect models. These results suggest that aspects of the proposed method of context discernment are consistent with our intuition about human learning, particularly with respect to how learning is affected by the models and measurements used to perform a task and the difficulty of the task itself. Future experiments must necessarily introduce more complex problems and less perfect knowledge representations if this approach is to be developed into a general methodology for creating agents capable of performing identification and control tasks for arbitrary control systems.

REFERENCES*


* Please see [1] and [2] for extended lists of References for this material.