

2009 Special Issue

Adaptive dynamic programming approach to experience-based systems identification and control

George G. Lendaris*

Portland State University, Systems Science Graduate Program, Electrical & Computer Engineering and NW Computational Intelligence Laboratory, SySc, P.O. Box 751 Portland, OR 97207, United States

ARTICLE INFO

Article history:

Received 5 May 2009

Received in revised form 1 June 2009

Accepted 25 June 2009

Keywords:

Approximate Dynamic Programming (ADP)

Experience-based identification and control

Context discernment

Systems identification

Optimal control

Neural networks

ABSTRACT

Humans have the ability to make use of experience while selecting their control actions for distinct and changing situations, and their process speeds up and have enhanced effectiveness as more experience is gained. In contrast, current technological implementations slow down as more knowledge is stored. A novel way of employing Approximate (or Adaptive) Dynamic Programming (ADP) is described that shifts the underlying Adaptive Critic type of Reinforcement Learning method “up a level”, away from designing individual (optimal) controllers to that of developing on-line algorithms that efficiently and effectively select designs from a repository of existing controller solutions (perhaps previously developed via application of ADP methods). The resulting approach is called Higher-Level Learning Algorithm. The approach and its rationale are described and some examples of its application are given. The notions of context and context discernment are important to understanding the human abilities noted above. These are first defined, in a manner appropriate to controls and system-identification, and as a foundation relating to the application arena, a historical view of the various phases during development of the controls field is given, organized by how the notion ‘context’ was, or was not, involved in each phase.

© 2009 Elsevier Ltd. All rights reserved.

1. Introduction

In a recent paper, the notion called *Experience-Based Identification and Control* (EBIC) was put forth, with the objective of bringing to the attention of the controls community “. . . some hopefully seminal ideas that will inspire and guide even greater application of Adaptive Dynamic Programming (ADP) methods . . .” (Lendaris, 2008). In the present paper, this objective is further pursued, this time in the context of the neural networks community. While the underlying ideas of necessity are repeated, where the previous paper focused on some of the more abstract/mathematical aspects, this paper extends into neural network implementations of selected entry points into the proposed approach.

Development of the EBIC approach evolved out of the widely-held desire in the research community to achieve more **human-like capabilities** in identification and control technologies. It is generally understood (at least intuitively) that human performance levels for such tasks depend on effective and efficient use of *experiential* knowledge. While the control engineering field has indeed accumulated remarkable achievements, even exceeding human control capabilities for some applications, substantial additional progress is still needed towards building into machines the

ability to employ experiential knowledge (hereafter called **experience**) when performing system identification and when coming up with a good controller for a given situation (even novel ones), and importantly, to do so effectively and efficiently.

Two observations concerning human abilities are pertinent here:

1. After a human learns a set of related identification and/or control tasks, when presented with a novel task of the same genre, the human is able to quickly generate close-to-optimal performance on the new task, based on the previously learned skills (i.e., effective selection from experience).
2. The more knowledge a human attains, the speed and efficiency of performing tasks are improved (in a relevant environment) – in stark contrast to Artificial Intelligence systems thus far developed, wherein the more knowledge acquired (typically stored as “rules”) the *slower* the decision/action processing.

For computational Agents to achieve the above-noted kind of effectiveness and efficiency, we posit they will have to implement the equivalent of *experience*. The approach here assumes three components for such an Agent system:

- (1) A collection of models appropriate to a given engineering application (models are of plants or controllers, depending on whether they are doing control or system identification).
- (2) A characterization of this set of models in a form that facilitates accessing the models.

* Corresponding author. Tel.: +1503 725 4988.

E-mail address: lendaris@sysc.pdx.edu.

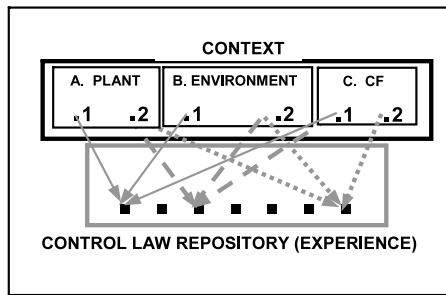


Fig. 1. Schematic of CONTEXT and corresponding control law REPOSITORY.

- (3) An Agent with an algorithm that effectively and efficiently selects a sequence of (good) models from this set as context-changes occur within the application.

These three components are here deemed fundamental to what is meant when humans are said to have attained *experience* related to a class of identification/control tasks, and note that implicit in these requirements is a *memory* property for the Agent.

To fix ideas, consider a control-engineering setting in which a plant, environment, and control objective are provided to a (human) designer, who is to design and implement a controller. If the designer is experienced and has “seen” the situation before, after obtaining context data (defined below), he/she pulls the appropriate design out of the archives and applies it to the current situation – perhaps with a little tailoring. *The more experienced the control engineer is, the process goes faster and with better results.*

1.1. Context

To craft a definition of experience, we first take note of another fundamental notion – **context**. Humans intuitively understand that as context changes, so do the decision rules and or control policies used to function within the given context.

The notion of *context* is here formulated to comprise three components: (1) plant, (2) environment, and (3) objectives plus associated performance criteria (labeled **CF**). See Fig. 1. The specification of all three yields a specific context; a change in any of the components results in a different context. In this formulation, to each specific context there corresponds a particular control law.

As an intuitive entrée to this use of the term *context*, think of driving on a clear afternoon (1) on dry pavement, or, (2) on an icy pavement. The general driving skills in both scenarios are the same, but selected adjustments are needed to your control law and/or decision logic. If instead of a change in the environment (road conditions) there is a change in attributes of the car (plant) – e.g., a slightly flat tire – new adjustments are needed for the car to perform your desired maneuvers; driving your friend’s car that day rather than your own is another example. A third consideration involves performance criteria (CF); e.g., in a road race, a criterion is to minimize time, but for an elderly relative on an excursion, maximizing comfort is more likely. Each of these examples may be represented via a triplet of lines from the Context in Fig. 1, pointing to a corresponding control law in the Repository (and we note, this selected control law could be locally adaptive as well).

1.2. Experience

The term *experience* as used here entails a *collection* of designs that have already been developed for a set of contexts from a common application domain, and also, entails a *memory* about the collection. The collection is here called an **experience repository** for that domain. Clearly, a variety of existing methods of the controls field may be employed to generate components for the repository; those methods are here assumed available within the *experience-based* process as a means for growth of the repository.

1.3. Representation and mappings

The context-discernment and controller-selection processes are directly impacted by the representations crafted for context and repository, and in particular, by the indexing schemas defined for the various sets involved. Theoretical considerations related to such representation and mapping issues uncover a myriad of difficulties. A candidate tool deemed useful for delving into the associated issues is the formalism of *manifolds* from geometric topology, where the manifold comprises a *set* and a *coordinate system*. In the present setting, **the manifold’s set is to comprise the experience repository**, and the **manifold’s coordinate system is to be a searchable indexing mechanism with useful “nearness” properties**, more details in Lendaris (2008). In the author’s preliminary work, this approach provided the framework in which a novel concept for applying Reinforcement Learning (the previously mentioned HLLA, Higher Level Learning Algorithm – see Section 3) was developed for evolving the nascent experience-based ideas.

The key idea for HLLA is to **re-purpose** the Reinforcement Learning (RL) method so instead of performing the usual task of *designing* an optimal controller for a given context – the “level” at which the RL methods are typically applied – a collection (repository) of such designs for a variety of related contexts is provided, and the **new design task** for the RL is to develop a strategy for optimally *selecting* an existing solution from the repository (the focus for the RL is thus “one level up” – hence HLLA). A detailed set of definitions of the terms employed for the above framework are given in Lendaris (2008).

The *selection process* is to be triggered by the Agent becoming aware that a *change in context* may have occurred. This is followed by the Agent seeking information about what has changed – a process here called *context discernment*; the latter process typically entails a form of system identification (SID), also enhanced via *experience*. Two examples are given in Section 5.

1.4. Overview summary

In summary, it is posited that the following four aspects are fundamental to the Experience-Based (EB) notion: (1) *context*, (2) *discerning* current context, (3) *selecting* appropriate solution for the discerned context from an *experience repository*, and (4) doing the latter two in an effective and timely manner. Beyond this base level, it is further posited that context discernment is fundamental not only for the selecting aspect mentioned above, but also for deciding *what task(s)* to perform in a given situation; e.g., in a football game, do I throw the ball, kick it, or run it? Notions of hierarchy and optimization will no doubt be fundamental to such considerations; a concept that might be called *Context Space Hierarchy* would powerfully assist in this endeavor. Development of such a concept is deemed an important issue for future work.

2. Historical perspective of context and the controls field

While the existence of control devices dates back to antiquity (call it PHASE 1 for the controls field), Maxwell’s use of differential equations to analyze the dynamics of a flyball governor (ca. 1870) (Lewis, 1992) that had been invented by James Watt in 1788 (Watt, 2005) may be said to have ushered in a new phase for the controls field (call it PHASE 2). Whereas intuition and inventive genius may be said to characterize Phase 1, mathematics played a fundamental role in Phase 2, progressing through Fourier and Laplace transforms, state-space methods, stochastic methods, Hilbert space methods, and more recently, algebraic and geometric topological methods. The advent of modern computers has also been significant, not only relative to implementation, but also as

Download English Version:

<https://daneshyari.com/en/article/404647>

Download Persian Version:

<https://daneshyari.com/article/404647>

[Daneshyari.com](https://daneshyari.com)