ADAPTIVE DYNAMIC PROGRAMMING AND REINFORCEMENT LEARNING

Derong Liu and **Ding Wang**
The State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, PR China

Keywords: Adaptive dynamic programming, approximate dynamic programming, neural dynamic programming, neural networks, nonlinear systems, optimal control, reinforcement learning

Contents

Summary

In the present chapter, the mathematical formulations and architectural structures of reinforcement learning (RL) and a corresponding implementation approach known as adaptive dynamic programming (ADP) are introduced. The iterative ADP algorithm is developed to design the controller of nonlinear systems that both learn and exhibit optimal behavior. It is shown that the cost function and control law sequences of the iterative ADP algorithm can both converge to the optimal ones. Then, the globalized dual heuristic programming technique is employed to facilitate the iterative algorithm, where three neural networks are constructed to approximate the system dynamics, the cost function, and the control law, respectively. Moreover, the practical applications of ADP- and RL-related techniques are described. A simulation example is also provided to verify the effectiveness of the proposed approach. Finally, some topics about future development of ADP and RL are pointed out.

1. Introduction

Every living organism in the nature interacts with its environment and uses the interactions to improve its own actions to survive and increase. However, the limits within which the organisms can survive are often quite narrow and the resources available to most species are meager. Therefore, most organisms act in an optimal fashion in order to conserve resources yet achieve their goals. Optimal actions may be based on minimum fuel, minimum energy, minimum risk, maximum reward, and so on.

Learning is a process of acquiring new or modifying existing knowledge, behaviors,

skills, values, or preferences and may involve synthesizing different types of information. As Poggio and Girosi (1990) stated, the problem of learning between input and output spaces is in fact equivalent to that of synthesizing an associative memory that retrieves appropriate output when the input is present and generalizes when a new input is applied. With strong capabilities of self-learning and adaptivity, artificial neural networks (ANN or NN) are an effective tool to implement learning purpose (Haykin, 1999; Jagannathan, 2006). The ability to learn can be possessed by humans, animals, and some machines.

As a basic branch of artificial intelligence (Sigaud and Buffet, 2010), machine learning is a scientific discipline concerned with the design and development of various algorithms. Hence, the algorithms can allow the machine to learn via inductive inference based on observing data that represents incomplete information about statistical phenomenon and generalize it to rule and make predictions on missing attributes or future data. There are many types of learning including supervised learning, unsupervised learning, etc. We call modification of actions based on interactions with the environment reinforcement learning (RL). The combiner of learner and decision-maker is called the agent. The environment comprises everything outside the agent and also interacts with the agent. RL refers to an actor that interacts with its environment and modifies its actions based on stimuli received in response to its actions. RL implies a cause and effect relationship between actions and reward or punishment. RL concerns with how an agent ought to take actions in an environment so as to maximize some notion of cumulative reward (Lewis and Vrabie, 2009; Sutton and Barto, 1998; Wang et al., 2009).

The RL is highly related to dynamic programming (DP) technique, which is a very useful tool in solving optimization problem by employing the principle of optimality. Additionally, in control systems community, it is also an important approach to handle optimal control problem. Classical DP algorithms are of limited utility in RL both because of their assumption of a perfect model and because of their great computational expense, but they are still important theoretically. DP provides an essential foundation for understanding RL. Actually, most of the methods of RL can be viewed as attempts to achieve much the same effect as DP, with less computation and without assuming a perfect model of the environment.

It is often of interest to mimic nature and design control systems that are optimal in some sense of effectively achieving required performance without using undue amounts of resources.

As is known, the optimal control of nonlinear systems is a difficult and challenging area. Unlike the optimal control of linear systems, the optimal control of nonlinear systems often requires solving the nonlinear Hamilton-Jacobi-Bellman (HJB) equation instead of the Riccati equation (Lewis and Syrmos, 1995). For example, the discrete-time HJB (DTHJB) equation is more difficult to work with than the Riccati equation because it involves solving nonlinear partial difference equations. Fortunately, DP provides an effective avenue to deal with the problems. However, due to the well-known “curse of dimensionality” (Bellman, 1957), it is often computationally untenable to run DP to

obtain the optimal solutions. Moreover, the backward direction of the search obviously precludes the use of DP in real-time control.

One class of RL methods is built based on the actor-critic structure, namely, adaptive critic designs (Prokhorov and Wunsch, 1997), where an actor component applies an action or control policy to the environment, and a critic component assesses the value of that action. The combination of DP, NN and actor-critic structure results in the adaptive dynamic programming (ADP) algorithm, which was proposed by Werbos (1992) as a method to solve optimal control problems forward-in-time. During the last two decades, the ADP-related research has gained much progress in terms of theory and applications, especially in fields of artificial intelligence and control theory (Al-Tamimi et al., 2008; Balakrishnan et al., 2008; Bertsekas, 2011; Jagannathan and He, 2008; Lewis and Vrabie, 2009; Liu, 2005; Si and Wang, 2001; Vamvoudakis and Lewis, 2010; Venayagamoorthy, 2009; Wang et al., 2012; Werbos, 2011; Yang et al., 2008; Zhang et al., 2009). Now, the ADP approach has become the key direction for future research in understanding brain intelligence and building intelligent systems (Werbos, 2009). Moreover, it has becomes a main component of computational intelligence (Ruano, 2008).

2 Reinforcement Learning

In simple terms, the RL problem is meant to learn from interaction to achieve a goal. The interacting process between agent and environment consists of the agent selecting actions and the environment responding to those actions and presenting new situation to the agent. Besides, the environment gives rise to rewards, which are special numerical values that the agent tries to maximize (or minimize) over time.

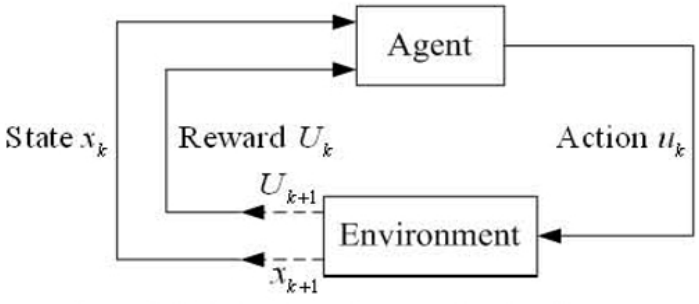

Figure 1. The interaction between agent and environment

More specifically, the agent and environment interact at each of a sequence of discrete time steps $k = 0,1,2,\ldots$. At each time step k, the agent receives some representation of environment's state $x_k \in \mathbb{R}^n$, where $\mathbb{R}^n$ is the set of possible states, and then selects an action $u_k \in \mathbb{R}^m$, where $\mathbb{R}^m$ is the set of actions available in state x_k. One time step later, the agent receives a numerical reward U_{k+1} and finds itself in a new state x_{k+1}. The schematic diagram of agent-environment interaction is depicted in Figure 1.

At each time step, the agent implements a mapping from states to probabilities of

selecting each possible action. This mapping is called the agent's policy and is denoted by p_k, where $p_k(x,u)$ is the probability that $u_k = u$ if $x_k = x$. RL methods specify how the agent changes its policy as a result of its experience. The agent's goal is to maximize (or minimize) the total amount of reward it receives over the long run.

3 Adaptive Dynamic Programming

The above RL problem can be taken as the basis of ADP-related problem. We consider nonlinear discrete-time dynamic (deterministic) systems given by

$$x_{k+1} = F(x_k, u(x_k)), k = 0,1,2,\ldots, \tag{1}$$

where $u(x_k)$ can be denoted by u_k for simplicity and $x_k \in \mathbb{R}^n$ and $u_k \in \mathbb{R}^m$ represent the state vector and control action of the controlled system respectively. The cost function associated with the controlled system is

$$J(x_k) = \sum_{i=k}^{\infty} \gamma^{i-k} U(x_i, u_i), \tag{2}$$

where U is called the utility function and γ is the discount factor with $0 < \gamma \le 1$. The function J is dependent on the initial time step k and the initial state x_k, and it is referred to as the cost-to-go of state x_k. The objective of optimal control problem is to choose a control sequence $u_k, u_{k+1}, \ldots$, so that the cost function J in (2) is minimized. According to Bellman's optimality principle, the optimal cost function from time step k

$$J^*(x_k) = \min_{u_k, u_{k+1}, \cdots} \sum_{i=k}^{\infty} \gamma^{i-k} U(x_i, u_i) \tag{3}$$

can be rewritten as

$$J^*(x_k) = \min_{u_k} \left\{ U(x_k, u_k) + \gamma \left(\min_{u_{k+1}, u_{k+2}, \cdots} \sum_{i=k+1}^{\infty} \gamma^{i-k-1} U(x_i, u_i) \right) \right\}. \tag{4}$$

In other words, $J^*(x_k)$ satisfies the DTHJB equation

$$J^*(x_k) = \min_{u_k} \left\{ U(x_k, u_k) + \gamma J^*(x_{k+1}) \right\}. \tag{5}$$

The corresponding optimal control $u^*(x_k)$ is the u_k which achieves this minimum, i.e.,

$$u^*(x_k) = \arg\min_{u_k} \left\{ U(x_k, u_k) + \gamma J^*(x_{k+1}) \right\}. \tag{6}$$

Equation (5) is the principle of optimality for discrete-time systems. Its importance lies in the fact that it allows one to optimize over only one control vector at a time by working backward in time.

In nonlinear continuous-time case, the controlled systems can be given by

$$\dot{x}(t) = F(x(t), u(x(t))), t \geq t_0 . \tag{7}$$

In this case, the cost is defined by

$$J(x(t)) = \int_t^{\infty} U(x(\tau), u(x(\tau)))\mathrm{d}\tau . \tag{8}$$

For continuous-time systems, the optimal cost $J^*(x_0)$ will satisfy the HJB equation

$$\begin{aligned} -\frac{\partial J^*(x(t))}{\partial t} &= \min_{u \in U} \left\{ U(x(t), u(t)) + \left(\frac{\partial J^*(x(t))}{\partial x(t)} \right)^{\mathrm{T}} F(x(t), u(t)) \right\} \\ &= U(x(t), u^*(t)) + \left(\frac{\partial J^*(x(t))}{\partial x(t)} \right)^{\mathrm{T}} F(x(t), u^*(t)). \end{aligned} \tag{9}$$

Equations (5) and (9) are called the optimality equations of DP which are the basis for implementation of DP. However, as stated in Section 4.1, the theoretical solution of HJB equation is very different to obtain. Thus, the idea of ADP has been engendered to circumvent the "curse of dimensionality" by building a critic system to approximate the cost function of DP. Solution to the ADP formulation is obtained through NN based actor-critic approach. Except for the aforementioned "adaptive critic designs" and RL, there are several other synonyms used for ADP, including "approximate dynamic programming", "neuro-dynamic programming", and "neural dynamic programming". The main idea of ADP is shown in Figure 2.

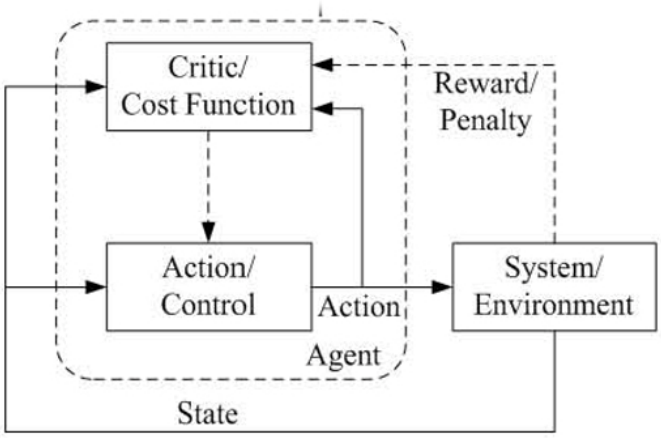

Figure 2. The main idea of ADP

3.1 Basic structures

According to Werbos (1992) and Prokhorov and Wunsch (1997), ADP approaches were classified into several main schemes: heuristic dynamic programming (HDP), action-dependent HDP (ADHDP), also known as Q-learning, dual heuristic dynamic programming (DHP), ADDHP, globalized DHP (GDHP), and ADGDHP. Werbos (1992) originally proposed two basic versions of ADP, i.e., HDP and DHP. The basic components are three NNs, which are model network, critic network and action network. The structure diagram of HDP is shown in Figure 3.

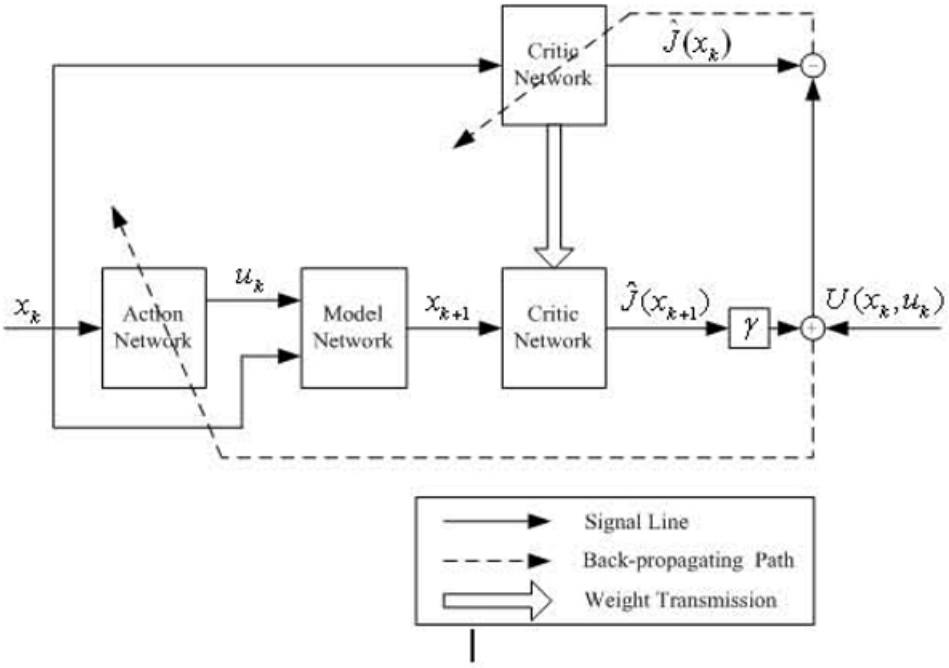

Figure 3. The HDP structure

In HDP, the output of the critic network is $\hat{J}$, which is the estimate of J in (2). The training process is done by minimizing the following error measure over time

$$\|E_{\mathrm{H}}\| = \sum_k E_{\mathrm{H}k} = \frac{1}{2}\sum_k \left\{\hat{J}(x_k) - U(x_k,u_k) - \gamma \hat{J}(x_{k+1})\right\}^2. \qquad (10)$$

When $E_{\mathrm{H}} = 0$ for all k, (10) implies that

$$\hat{J}(x_k) = U(x_k,u_k) + \gamma \hat{J}(x_{k+1}). \qquad (11)$$

Then, we can further obtain that

$$\hat{J}(x_k) = \sum_{i=k}^{\infty} \gamma^{i-k} U(x_i,u_i), \qquad (12)$$

which is the same as (2).

In DHP, the action network remains the same as the one for HDP, while the critic network outputs the derivative of the cost function, namely, the costate function. The structure diagram is depicted in Figure 4.

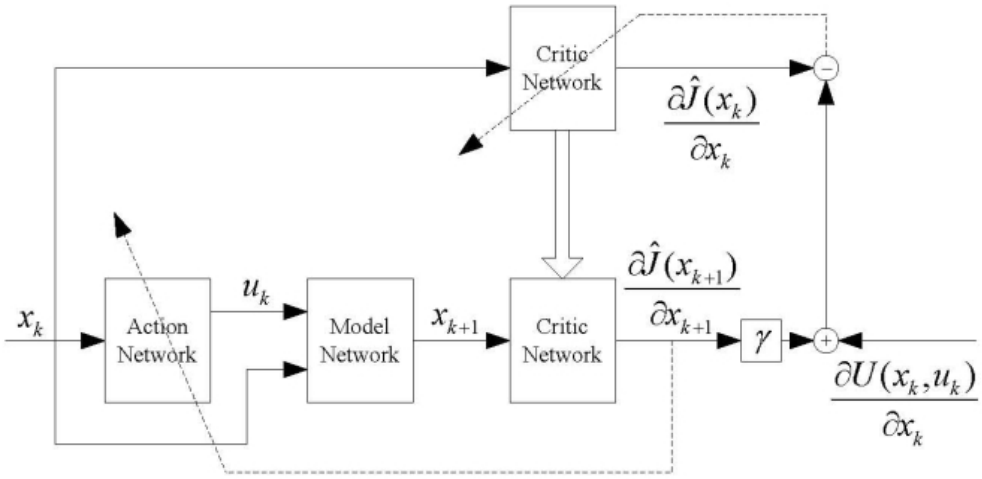

Figure 4. The DHP structure

Here, the critic network is trained to minimize

$$\|E_{\mathrm{D}}\| = \sum_k E_{\mathrm{D}k} = \frac{1}{2}\sum_k \left\{ \frac{\partial \hat{J}(x_k)}{\partial x_k} - \frac{\partial U(x_k,u_k)}{\partial x_k} - \gamma \frac{\partial \hat{J}(x_{k+1})}{\partial x_k} \right\}^2 \qquad (13)$$

over time. Note that when $E_{\mathrm{D}} = 0$ for all k, we have

$$\frac{\partial \hat{J}(x_k)}{\partial x_k} = \frac{\partial U(x_k,u_k)}{\partial x_k} + \gamma \frac{\partial \hat{J}(x_{k+1})}{\partial x_k}. \qquad (14)$$

Though the training process of the critic network in DHP is somewhat complicated, the resulting behavior is expected to be superior to HDP. This is because the action network is adapted based on the value of $\partial \hat{J}/\partial x$, which is a direct output of DHP. However, if we employ HDP, we have to compute $\partial \hat{J}/\partial x$ from backpropagation, which will introduce approximation error inevitably.

3.2. Improved Structures

Prokhorov and Wunsch (1997) presented some new improvements to the design of GDHP. The most remarkable feature of GDHP is that the critic network outputs both the cost function $\hat{J}$ and its derivative $\partial \hat{J}/\partial x$. This can be schematically depicted in Figure 5.

Padhi et al. (2006) proposed the single network adaptive critic structure, which eliminated the use of the action network and offered three potential advantages: a

simpler architecture, lesser computational load and elimination of the approximation error associated with the action network. This structure is shown in Figure 6 and the training details can be referred to Padhi et al. (2006).

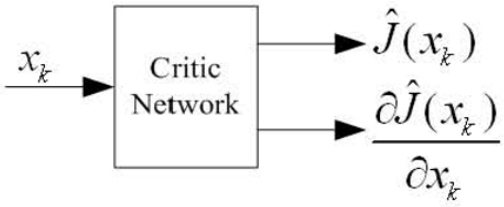

Figure 5. The critic network of GDHP

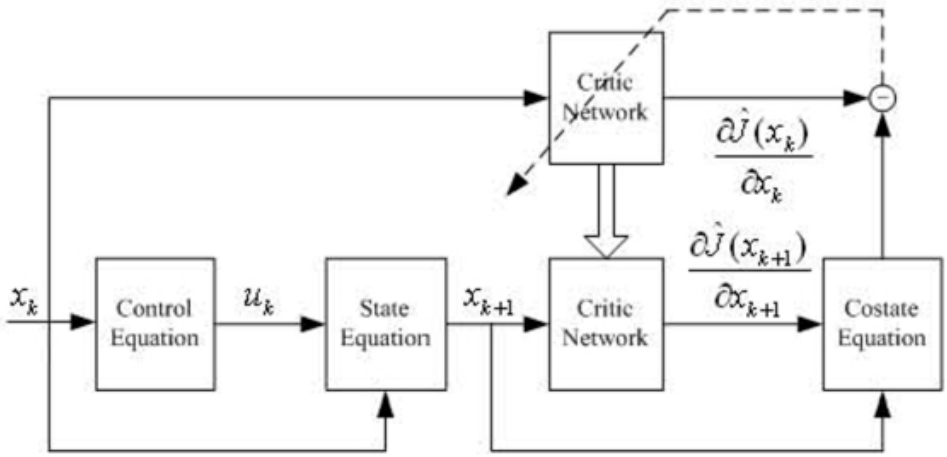

Figure 6. The single network adaptive critic structure

In Si and Wang (2001), the direct HDP technique was developed for the design of model-free adaptive critic, as shown in Figure 7. Compared with the traditional HDP technique, the model network is eliminated, which simplifies the structure and reduces the computation burden as well.

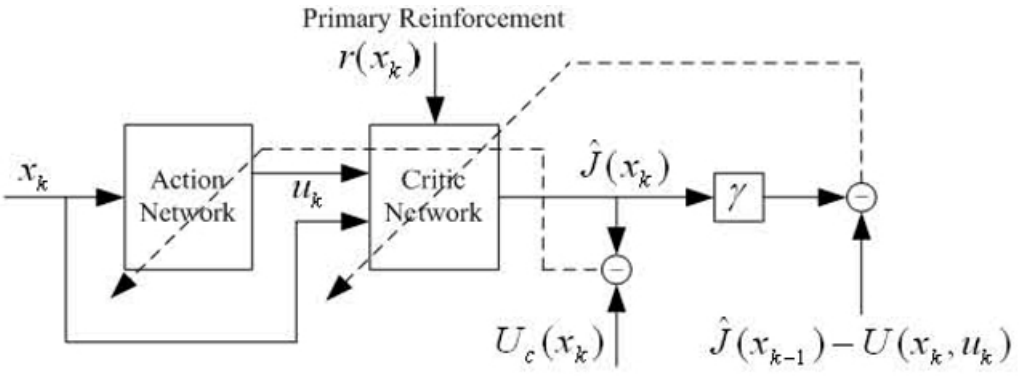

Figure 7. The direct HDP structure

The binary reinforcement signal $r(x_k)$ is provided from the external environment and

may be as simple as either a "0" or "-1" corresponding to "success" or "failure", respectively. The prediction error for training the critic network is $\gamma \hat{J}(x_k) - \hat{J}(x_{k-1}) + U(x_k, u_k)$. The principle in adapting the action network is to indirectly backpropagate the error between the desired ultimate objective, denoted by $U_c(x_k)$, and the approximate cost function $\hat{J}(x_k)$. The value of $U_c(x_k)$ can be set to "0", which has be defined as the reinforcement signal for "success".

Liu et al. (2001) further studied the model-free adaptive critic designs and provided two approaches for training the critic network. Figure 8 shows the diagram of forward-in-time approach.

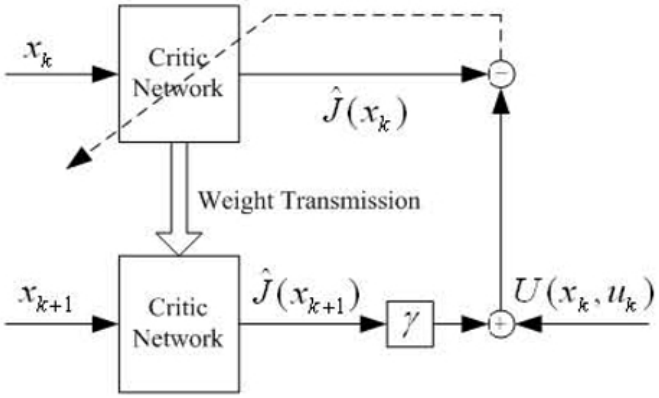

Figure 8. The forward-in-time approach

In this approach, we view $\hat{J}(x_k)$ as the output of the critic network to be trained and choose $U(x_k, u_k) + \gamma \hat{J}(x_{k+1})$ as the training target. Obviously, $\hat{J}(x_k)$ and $\hat{J}(x_{k+1})$ are obtained using state variables at different time instances.

Another method for training the critic network is backward-in-time approach described in Figure 9.

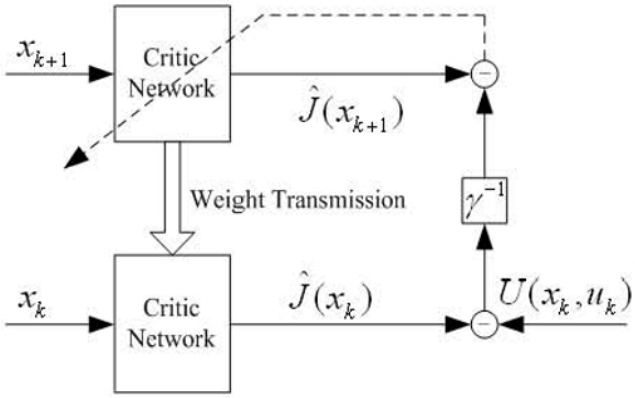

Figure 9. The backward-in-time approach

-
-
-

Bibliography

[1] Siegelmann, HT (April 2013). "Turing on Super-Turing and Adaptivity". *Journal Progress in Biophysics & Molecular Biology*. pii: S0079-6107(13)00027-8. [A review paper that describes the Super-Turing model of computation and its possible effect in Artificial Intelligence and Biology].

[2] Siegelmann, HT (April 1995) "Computation Beyond the Turing Limit," *Science* 238(28): 632-637. [A research paper introducing a particular dynamical system that has chaotic properties, and that computationally cannot be described by the Turing machine model and instead can be fully explained by the Super-Turing model].

[3] Hebb, D (1949). *The Organization of Behavior*. New York: Wiley & Sons [Hebb's seminal work describing his postulate of associative learning at the neural level].

[4] Markram H, Lübke J, Frotscher M, Sakmann B (January 1997). "Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs". *Science* 275 (5297): 213–5. [A research paper that building on results of previous researchers, suggests how synapses change plastically by the Spike Timing Dependence associative learning process].

[5] Hopfield JJ (April 1982). "Neural networks and physical systems with emergent collective computational abilities". *Proceedings of the National Academy of Sciences of the USA*, 79(8): 2554–2558. [Hopfield introduces a physics based symmetric network, that can keep memory traces in its attractors; An energy function is defined in which its minima are the attractors.].

[6] Kosko, B (Jan/Feb 1988). "Bidirectional Associative Memories". *IEEE transactions of systems, man and cybernetics*: 49-60. [A generalization of the Hopfield attractor networks to allow for input-output functions].

[7] Cohen MA and Grossberg S (1983). "Absolute stability of global pattern formation and parallel memory storage by competitive neural networks". *IEEE Transactions on Systems, Man, and Cybernetics*, 13: 815-826. [Grossberg's group early research - somewhat unique and separate from the main stream - includes other attractor networks of the time].

[8] Zemel R and Mozer, M (2001). "Localist attractor networks". *Neural Computation* 13: 1045–1064. [A paper that introduced a particular attractor network with three levels: Input, neural, and attractors: The dynamic flow is in the middle layer, while each neuron in the third level stands for a memory trace. This allows for great separation among attractors, a known problem in all Hopfield-like networks].

[9] Kaneko K (1990), "Clustering, coding, switching, hierarchical ordering, and control in network of chaotic elements". *Physica* D 41: 137-172. [Introduces chaos theory in neural networks].

[10] Aihara K (1994). "Chaos in neural response and dynamical neural network models: toward a new generation of analog computing". in *Towards the harnessing of chaos*, edited by M. Yamaguti, Elsevier, Science Publishers B.V., Amsterdam: 83–98. [Introduces analog computation and chaotic behavior in networks].

[11] Kozma R and Freeman WJ (2001). "Chaotic resonance: Methods and applications for robust classification of noisy and variable patterns". *International Journal of Bifurcation and Chaos* 10: 2307-2322. [Practical usages of chaotic dynamic in neural networks].

[12] Kaneko K and Tsuda I eds (2003). Focus issue on chaotic itinerancy. *Chaos* 13: 926-1164. [Special dynamical behavior in networks and memory].

[13] Nowicki D and Siegelmann HT (june 2010). "*Flexible Kernel Memory*". PLOS One 5: e10955. http://www.plosone.org/article/info%3Adoi%2F10.1371%2Fjournal.pone.0010955 [Memory model that includes unbounded number of memory traces and allows inputs of variation in shades – much more practical than previous model].

[14] Ben-Hur A, Horn D, Siegelmann HT and Vapnik V (2001). "Support vector clustering". *Journal of Machine Learning Research* 2: 125-137. [A very successful clustering method based on kernel functions].

[15] Sara SJ (2000). "Retreival and reconsolidation: Toward a neurobiology of remembering". *Learning and Memory*, 7(2): 73-84. [Coins the term "reconsolidation" as the process of re-inserting information after recall].

[16] Blumenfeld B, Preminger S, Sagi D and Tsodyks M (2006). "Dynamics of memory representations in networks with novelty-facilitated synaptic plasticity". *Neuron* 52(2): 383-394. [A version of Hopfield network that allows the update of synaptic strength over time – rather than only during original loading].

[17] Siegelmann HT (2008). "Analog-Symbolic Memory that Tracks via Reconsolidation". *Physica* D: Nonlinear Phenomena 237 (9): 1207-1214. [A generalization of the localist attractor network to acknowledge the process of reconsolidation].

[18] Osan R, Tort A and Amaral O (2011). "A Mismatch-Based Model for Memory Reconsolidation and Extinction in Attractor Networks". *PLoS ONE*, 6(8):e23113. [A small update to the Hopfield model that allows it to remove memories].

[19] Nowicki D, Verga P, Siegelmann HT (2013). "Modeling Reconsolidation in Kernel Associative Memory". *PLoS ONE* 8(8): e68189. doi:10.1371/journal.pone.0068189 [Richest model of Reconsolidation in computational terms].

[20] Kohonen T and Honkela T (2007). "Kohonen Network". *Scholarpedia*. [Recent paper summarizing self-organizing neural network by Teuvo Kohonen and follows updates and applications].

[21] Tal A and Siegelmann HT (June 2013). "Conscience mechanism in neocortical competitive learning," ICCN2013, Sweden. [Recent work that utilizes a biological version of the SOM to explain recently found biological connection in layer 5].

[22] Roth F, Siegelmann HT and Douglas RJ (2007). "The Self-Construction and -Repair of a Foraging Organism by Explicitly Specified Development from a Single Cell," *Artificial Life* 13(4: 347-368. [Artificial life model that uses the concept of self organization – joining chemistry and physics]

Biographical Sketches

Dr. Siegelmann's research focuses on mathematical and computational studies of the brain, memory, biological computation, and systemic disorders. Her research into biologically inspired cognitive and neural processes has led to theoretical modeling and original algorithms of machine systems capable of superior computation and learning. A unifying theme underlying her research is the study of dynamical and complex systems. Siegelmann's seminal Turing machine equivalence of recurrent neural networks theorem and Super-Turing theory introduced a new subfield of computer science. Her modeling of geometric neural clusters resulted in the highly utile and widely used Support Vector Clustering algorithm with Vladimir Vapnik and colleagues, specializing in the analysis of high-dimensional, complex data. Her work is often interdisciplinary, combining complexity science, dynamical and computational theories with biological sciences, neurology, physics and medicine. Recent contributions include computational and dynamical system studies of reconsolidation, the circadian system, addiction, cancer, and genetic networks; applications in intelligent robotics and advanced human-robot interfaces for military and health applications are currently funded. Her engineering background has been central to her NSF funded development of analog hardware for brain-like intelligence. She remains active in supporting young researchers and encouraging minorities and women to enter and advance in STEM. She has years of experience consulting with industry, creating educational programs including interdisciplinary and international programs, fund raising, and in educational administration and organization.

Dr. Kozma's current research interests include spatio-temporal dynamics of neural processes, random graph approaches to large-scale networks, such as neural networks, computational intelligence methods for knowledge acquisition and autonomous decision making in biological and artificial systems; he also published in related fields including signal processing; and design, analysis, and control of intelligent systems. He has served since 2009 as a Professor of Computer Science, University of Memphis, Memphis, Tennessee, and Professor of Mathematical Sciences, University of Memphis, Memphis, Tennessee. He has also been the director of Computational Neurodynamics Laboratory, presently CLION, FedEx Institute of Technology of the University of Memphis, Memphis since 2001. Kozma serves on the AdCom of IEEE Computational Intelligence Society CIS (2009–2012) and on the Governing Board of the International Neural Network Society INNS (2004–2012). He is Chair of the Distinguished Lecturer Program, IEEE CIS. He has been a Technical Committee Member of IEEE Computational Intelligence Society since 1996, and IEEE Senior Member. He also served in leading positions at over 20 international conferences, including General Chair of IEEE/INNS International Joint Conference on Neural Networks IJCNN09 at in Atlanta; Program Co-Chair of International Joint Conference on Neural Networks IJCNN08/WCCI08 in Hong Kong; Program Co-Chair of IJCNN04, Budapest, Hungary; Chair for Finances of IEEE WCCI06, Vancouver, Canada. He is Associate Editor of 'Neural Networks (Elsevier),' 'IEEE Transactions on Neural Networks,' 'Neurocomputing' (Elsevier), 'Journal of Cognitive Neurodynamics' (Springer), Area Editor of 'New Mathematics and Natural Computation' (World Scientific), and 'Cognitive Systems Research.'