Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 61 (2015) 492 – 499

Procedia Computer Science

Complex Adaptive Systems, Publication 5
Cihan H. Dagli, Editor in Chief
Conference Organized by Missouri University of Science and Technology
2015-San Jose, CA

An Improved eXtended Classifier System for the Real-time-input Real-time-output (XCSRR) Stability Control of a Biped Robot

A. Sabzehzar[a]*, W. L. Shan[a], M. Shariat Panahi[b], O. Saremi[b]

[a] *Department of Mechanical Engineering, University of Nevada, Reno. Reno, NV 89557, USA*
[b] *Department of Mechanical Engineering, University of Tehran, Tehran 515-14395, Iran*

Abstract

In this paper a revised reinforcement learning method is presented for stability control problems with real-value inputs and outputs. The revised eXtended Classifier System for Real-input and Real-output (XCSRR) controller is designed, which is capable of working at fully real-value environment such as stability control of robots. XCSRR is a novel approach to enhance the performance of classifier systems for more practical problems than systems with merely binary behaviour. As a case study, we use XCSRR to control the stability of a biped robot, which is subjected to unknown external forces that would disturb the robot equilibrium. The external forces and the dynamics of the upper body of the biped robot are modelled in MATLAB software to train the XCSRR controller. Theoretical and experimental results of the learning behaviour and the performance of stability control on the robot demonstrate the strength and efficiency of the proposed new approach.

Peer-review under responsibility of scientific committee of Missouri University of Science and Technology

Keywords: eXtended Classifier Systems (XCS); reinforcement learning; real-value problem

1. Introduction

A classifier system is a set of if-then or condition-action rules that gain reinforcement form of the environment by on-line learning. Classifier systems are used to generate the behaviour of real system with unknown or changeable dynamics. A set of rules (or what we call “classifiers”) is trained to read the input condition of environment and to respond a dynamic model [1].

Learning Classifier System (LCS) is the very first generation of classifier systems and was introduced by John

* Corresponding author. Tel.: +1-775-409-2827
E-mail address: asabzehzar@unr.edu

Peer-review under responsibility of scientific committee of Missouri University of Science and Technology
doi:10.1016/j.procs.2015.09.198

Holland [2], which is a machine learning system with close links to reinforcement learning and genetic algorithms [3]. LCS consists of specified numbers of rules that compose population. A fitness value, based on a reinforcement learning technique, is allocated to each rule (classifier). In other words, LCS is an adaptive process with ability to choose the best action that improves with experience. Each rule has two parts: the if-part and the then-part. The if-part is the condition part of the rule that is examined to find out whether it is matched with the environment state. The then-part is the action part of the rule that apply actions on the environment. The environment states usually are extracted by sensors while actions are applied by actuators. In the training phase of LCS, a rule's fitness modifies; better rules get reward (i.e. higher fitness) and worse rules get punishment (i.e. lower fitness). The next phase is the testing phase in which the set of modified rules (population) after training is examined by unfamiliar examples.

For years LCS had not worked well for conditions out of training environment. Because the learning classifier system was complicated and difficult to understand, Zero Level Classifier (ZCS), a strength-based classifier system, which does not have temporary memory, was introduced [4]. Results show that ZCS can work properly if its parameters are set correctly, which unfortunately is a challenging task as the problems under investigation change. To address this issue, an accuracy based eXtended Classifier System (XCS) was introduced by Wilson [5]. XCS is the most popular implantation of LCS [6, 7], where classifiers' fitness is modified based on the accuracy of the rule that increases performance in proportion to LCS, especially for challenging problems [8, 9,10]. Later studies mostly focused on XCS and its modifications such as TCS [11], XCSF [12, 13, 14]. In XCS, similar to LCS, the environment states are read by sensors at time "t" and the controller processor generates proper response to the environment. At the training stage, computed payoff (reward or punishment) will modify the selected rule parameters based on the accuracy of the response.

The initial population of XCS classifiers is usually constant. In binary codes, '1' refers to the existence of a state and '0' the absence of that. Also, wildcard (#) is used to represent the state that can be replaced by '0' or '1'. In the then-part, '0' or '1' refers to turning the actuators on or off. In addition to the fitness in XCS, three other parameters were supplemented to initial LCS, which are error (E), experience (exp), and prediction (p). Error predicts the error of the classifier from the desired value; Experience defines the number of times that the classifier's action is applied to the environment, and Prediction predicts the value of the classifier.

LCS and XCS initially have taken binary values as input and output. To allow for continuous variables (real values) such as temperature for the input of the XSC, Stewart et al. introduced XCSR as a variation of XCS taking real value inputs [15]. In XCS the input values consist of {0, 1, #}, but in XCSR input values are represented as an interval predication. In XCSR, a classifier matches with an input x if and only if x is between the lower and upper limits of the interval predication. Although XCSR works more functional than XCS for real problems, it is still insufficient to solve problems with real value components for input and output vectors, for example, a system with temperature as the input and voltage as the output.

In this paper the revised version of XCSR, named eXtended Classifier System for Real-input Real-output (XCSRR), is introduced to be capable of working in real-value-input real-value-output environment. Here we report an experiment with XCSRR where its efficacy is tested in the stability control of a real biped robot. We first explain how XCS and XCSR work. We then define and illustrate our own approach XCSRR with more details and examples. We show the efficiency of our approach to control the stability of a biped robot using theoretical simulations by Matlab. Experimental results on the stability control of a simplified robot model are also shown to be as predicted by the theory, evidencing much improved learning performances. The XCSRR approach is thus shown to be advantageous in real-value machine learning systems, and can potentially find many more applications in this area.

2. Algorithm

2.1. Parameters definition

There are many parameters that can be defined by the user to control the training and testing phases of the classifier system. For the XCSRR, these parameters are the same as the XCS definition. The appropriate values of these parameters used in this study are shown in Table 1.

Table 1 Definition and values of parameters for XCSRR

symbol	definition	value in this study
N	maximum number of population rules (i.e. the number of the whole rules)	7000
β	learning rate, which is used to update rules' parameters (P, ε, and F)	0.15
α, υ, ε0	Used for calculation of the rules' fitness	α=0.1 υ=5 ε0=0.2
ϒ	Used to update the classifier's fitness	0.71
θGA	Genetic algorithm threshold; genetic algorithm applied to the selected classifiers once every θGA time. (this parameter is defined based on specific problems and number of steps)	60 (every two epoch)
X	probability of applying crossover in genetic algorithm	100%
μ	probability of applying mutation in generic algorithm	0.03
θdel	deletion threshold; if the experience of the classifier is greater than θdel, it is candidate for deletion (i.e., the classifiers with high number of experience and low fitness will be removed)	20
P#	the probability of using wildcard (#) in the process of generating rules (covering)	0.08
P1, ε1,f1	initial value of prediction, error, and fitness of classifiers*	P1=3 ε1=0.02 f1=0.01
θmna	minimum action-size; minimum number of classifiers actions, which is needed to be matched with the environment **	25

* initial value of P should be in the order of actions in real-value output

** θmna should be in the order of all different sets of actions

2.2. Loop formation

Like XCS, after defining parameters, the main loop of XCSRR should be formed during the training process. The main loop will be run for many times in order to modify the random rules. In the first stage of the main loop, the environmental states are read. The rules, whose if-part is matched with the environment, will be chosen to compose the Match set ([M]). Then, the classifiers in the match set with the same action part form the fraction (sub-groups); the set of all fractions is called the Prediction Array ([PA]). We allocate a value to each fraction that defines its relative strength. The prediction value for each action is calculated as below, where i represents the index of summation that is equivalent to the number of fractions, and j is equivalent to the number of classifiers at each fraction. The classifiers related to the action that is opted to apply to the environment are called the Action set [A].

$$PA_i = \frac{\sum_{i=1}^{n} P_j \times f_j}{\sum_{j=1}^{n} f_j} \tag{1}$$

Using a roulette wheel, one of the actions will be selected and applied to the environment. The parameters of two classifiers in the action set will be modified corresponding to the value of payoff that they receive from the reinforcement program. Table 2 shows the way of modifying classifier's parameters in this study. The experience of rules in a selected fraction will be added by 1. Genetic algorithm acts between two classifiers (chromosomes) in the selected fraction of action set after θ_{GA} steps and helps to compose new rules with relatively high accuracy. In this study, composed classifiers are added to the population with no deletion of parents. When the number of classifiers meets the maximum size of population, weak classifiers with experience greater than θ_{del} and low fitness will be removed by the use of a roulette wheel. In the case where there is no classifier with experience greater than θ_{del}, a classifier will be removed based on its fitness using a roulette wheel.

For problems with real input and real output values, the chromosome consists of real value genes. In the if-part of the rule, each state of the classifier is expressed by the Lower-upper Bound (or Center-Spread) representation [10], and the if-part of the rule is also of real value. To compose an action set, a method of clustering is used to compose fractions with approximately the same actions.

Table 2 Mechanisms of updating classifiers' experience, prediction, error, and fitness [5]

expi = expi +1	
If expi < 1/β	Pi=Pi+(R-Pi)/ expi
	εi= εi+(|R-Pi|- εi)/ expi
If expi ≥ 1/β	Pi=Pi+ β (R-Pi)
	εi= εi+ β (|R-Pi|- εi)
If εi < ε0 ki=1	fi=fi+ β [(ki/∑kj) – fi]
If εi ≥ ε0 ki= β (εi/ε0) –γ	j refers to classifiers related to the selected action

3. XCSRR, eXtended Classifier System for Real-value-input Real-value-output

The proposed XCSRR method is capable of controlling the system in real environment with real-input real-output values. There are two approaches to show the input condition of the system: Center-Spread approach and Lower-Upper bound approach. In the Center-Spread approach, the environment's conditions, i.e. the input value for the classifier system, should be within the interval between Center plus and minus Radius of convergence. For the Lower-Upper bound approach, the interval is between the lower and upper values of the if-part.

For the operation system in real value environment, match set is composed as with XCS. To compose an action set, we can assume that each action is a point with one, two, or more dimensions (dimensions are the number of actions that act on the environment). We aim to classify the nearest actions (points) in one group of action set. To do this, we use a clustering method. In our clustering method, firstly the distances between any two points are calculated. For N classifiers in a match set, N_D distances are calculated, where N_D is equal to $\frac{N(N-1)}{2}$. For example, suppose that we have ten matched classifiers in a match set with the real value then-part as shown in Table 3, then we will have 45 distances between every two of the ten available points in the if-part, as shown in Table 4.

Table 3 Sample match set with classifiers with real-value input and output

Classifier number in match set	If-part	Then-part
1	0.85, 0.125, 0.875, 0.00	**2.01**
2	0.50, 0.35, 0.625, 0.11	**9.35**
3	0.625, 0.37, 0.125, 0.02	**0.29**
4	0.50, 0.50, 0.68, 0.25	**1.10**
5	0.625, 0.33, 0.18, 0.125	**7.31**
6	0.25, 0.17, 0.125, 0.125	**1.80**
7	0.375, 0.375, 0.875, 0.05	**6.83**
8	0.125, 0.10, 0.75, 0.25	**5.12**
9	0.45, 0.125, 0.50, 0.30	**6.02**
10	0.56, 0.125, 0.375, 0.375	**0.79**

In the training step the first two points with the minimum distance compose the first action group and are assumed as one point with the value equal to the mean value of two points. This procedure repeats recursively for the N-1 numbers of points (actions) again and continues till all points compose one group of classifiers. As shown in Table 4, the minimum distance is between points 1 and 6. This composes the first (nearest) group. By continuing the procedure, we have the results plotted in Fig. 1.

Table 4 Distances between then-part of classifiers

10	9	8	7	6	5	4	3	2	points
1.22	4.01	3.11	4.82	0.21	5.3	0.91	1.72	7.34	**1**
8.56	3.33	4.23	2.52	7.55	2.04	8.25	9.06	_	**2**
0.5	5.73	4.83	6.54	1.51	7.02	0.81	_	_	**3**

0.31	4.92	4.02	5.73	0.7	6.21	_	_	_	**4**
6.52	1.29	2.19	0.48	5.51	_	_	_	_	**5**
1.01	4.22	3.32	5.03	_	_	_	_	_	**6**
6.04	0.81	1.71	_	_	_	_	_	_	**7**
4.33	0.9	_	_	_	_	_	_	_	**8**
5.23	_	_	_	_	_	_	_	_	**9**

One critical parameter in composing action sets is the threshold number. Fig. 2 shows how the threshold number defines the maximum distance between points (actions) within one group of action set. For a specific number of classifiers, by having a greater threshold, the number of action sets decreases, but the number of rules in each action set increases. Similarly, by having a smaller threshold, the standard deviation of classifiers in one group of action set is smaller and the then-actions are closer to each other. On the other hand, training phase needs more iterations and classifiers to cover whole states of environment.

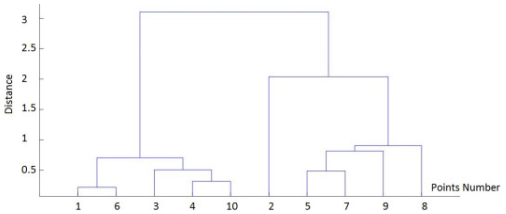

Fig. 1 Distances between then-part of classifiers

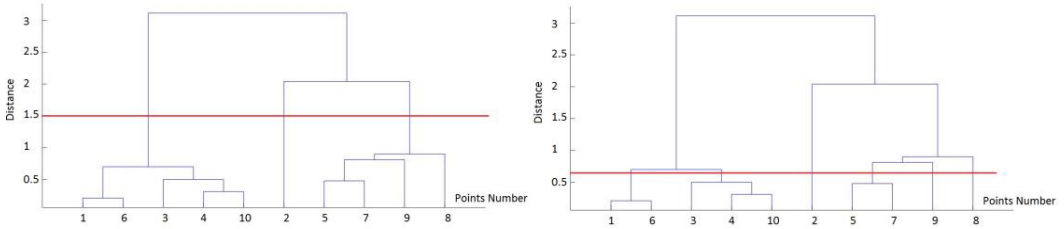

Fig. 2 Effect of threshold value on action set configuration

Using a roulette wheel, the random action is chosen as in XCS and the mean value of the classifiers' then-part, $Action_j$, in the chosen action set acts on the environment, as Eqn. 2 shows:

$$Action_j = \frac{\sum_{j=1}^{n_j} action_{ji}}{n_j} \tag{2}$$

where i is the index of summation within a specific action set, j is the index of the action sets, and n_j is the number of actions within a specific action set.

Implementation of XCSRR for stability control

How to maintain the stability of humanoid robots during walking and standing is the most important challenge in designing a controller for these robots. Zero Moment Point (ZMP) theory is a common method to control the robot stability, which is based on the resultant of gravity and inertia forces moment [16]. Bio-inspired architectures or artificial intelligence methods can help the controller correct its parameters with the aim of controlling the robot equilibrium. For this purpose, many neural network controllers, such as Cerebellar Model Articulation Controllers (CMAC) [17,18,19], recurrent neural networks [20,21], and fuzzy logic [22] for self-constructing fuzzy neural controller [23], have been developed. Moreover, Continuous-Time Recurrent Neural Networks (CTRNN) have been developed to generate adaptive behaviour in which the controller is trained with the Back-Propagation Through Time (BPTT) algorithm [24-26].

In this study, we build up a XCSRR controller with continues-valued input and output to keep the online balance of the bipedal robot with one arm in the standing posture during the disturbance of external forces. This controller analyses the environment states on-line and updates its classifier parameters in real time. In Ref. [27] a simplified mechanical system was proposed for the torso, which is capable of applying forces in different directions (i.e. F_x, F_y, F_z, M_x, M_y ,and M_z) to legs using locomotion systems. The proposed model is a 4 doffs R3P mechanism with one rotational joint (R) and three prismatic joints (3P). Fig. 3a illustrates the modelling of the R3P torso subjected to external forces. There are five mass points of which three are connected to three linear motors (q_x, q_y, q_z) and able to move in the main directions of X, Y, and Z, respectively. All Three linear motors are connected to the vertical rotational axis and the whole system rotates on Y axis. Fig. 3b shows the simplified model that has a moveable mass connected to a motor. The total force, which is measured with a force sensor, is displayed in an oscilloscope. The total forces will be used to compute the reward factors to train the classifier system controller in the training stage.

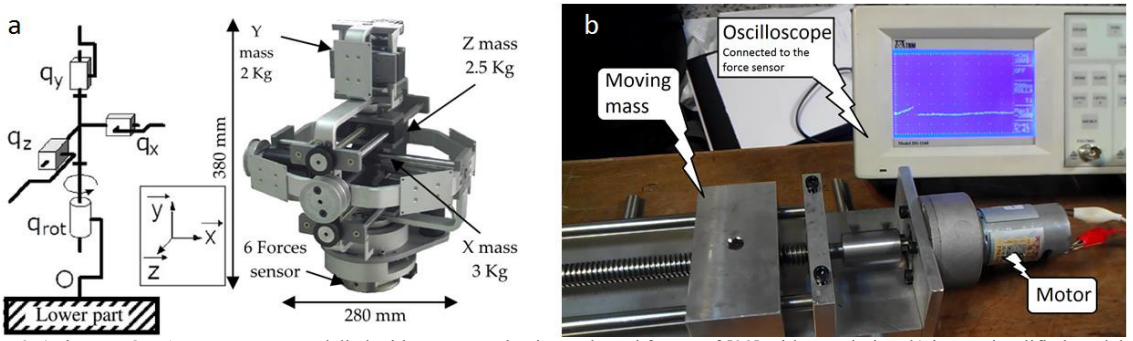

Fig. 3 a) shows ROBIAN upper part, modelled with RPPP mechanism, adapted form Ref [28] with permission. b) is our simplified model of ROBIAN robot which works in x-direction.

4. Results and Discussions

4.1. Theoretical Results

For this study, XCSRR is used with Lower-upper Bound representation. Classifiers of this system consist of 13 genes, 6 if-part, 3 then-part and 4 bits for prediction, fitness, error, and experience. The training process lasts for 165 seconds when external disturbances are applied to the X and Z directions simultaneously; the robot is in the standing posture and the only motor works for this experiment is in the X-direction; the moment components in X and Z directions are not required for classifiers training because their effects can be deducted.

The time constant (i.e. the time step that is used for external force disruptions) was selected by trial and error. The dynamics of ROBIAN biped and the external forces, which act as disturbances, are modelled using MATLAB software. Fig. 4 shows the random external forces that are applied to the X-axis between time t=0s to t=10s. The magnitude of the external force varies between zero and a maximum of 10 N. External forces disturb the robot stability during the training phase and the controller is trained to compensate for these random disturbances.

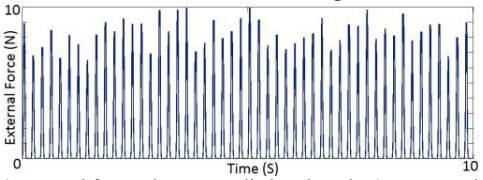

Fig. 4 External forces that are applied to the robot's torso randomly

To control the stability of ROBIAN robot, the classifiers are trained based on the calculated moment between the torso and the locomotion system (i.e. 6-component force sensor). The classifier controller is trained to keep these forces as close to zero as possible for X and Z axes. The training process is continued until the mean of the calculated forces at the hip for a set of unfamiliar testing data does not change noticeably. These testing data are dissimilar to the training data that are applied to the ROBIAN every 25 epochs (every 3.75 seconds). When applied the testing data, classifiers do not receive any reward or punishment and consequently do not update. Fig. 5 shows

the effect of training on the performance on testing data. It demonstrates the XCSRR controller's capacity to control the torso actuators to reduce the impact of external forces disturbances. XCSRR controller brings a decrease of about 65% in the calculated force at the robot's hip, which is much better than the previously reported 50% decrease achieved by a neural network controller [28].

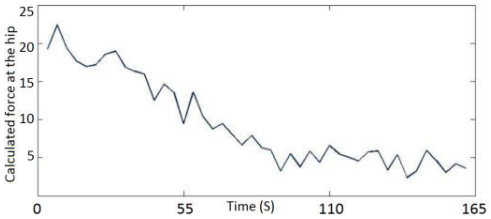

Fig. 5 On-line learning process for X and Z-axis, The XCS controller learned to compensate for external forces.

4.2. Experimental results

We also conducted experiments to verify the much improved performance predicted by our theory. In these experiments a ball-screw system is attached to a force sensor and a 12V DC motor (Fig. 6). The force sensor is capable of measuring the overall force (due to the disturbance force and the force of accelerated mass) acting on the system online. The disturbing force is applied as time passes and acts on the system in the same direction of the moving mass. An Oscilloscope is attached to the force sensor and displays the magnitude of the overall force.

Our goal is to keep the overall force near zero. To this end, the XCSRR controller is trained to control the motor's voltage such that the overall force is kept close to zero. In other words, the motor accelerates the attached mass in the direction opposite to the applied external disturbances to keep the overall force zero. As shown in Fig. 6, the XCSRR controller decreases the overall force and keeps the system in stability. The maximum overall force on the oscilloscope does not go over 3 N, which is much smaller when compared to the maximum disturbing force of 10 N (Fig. 4)

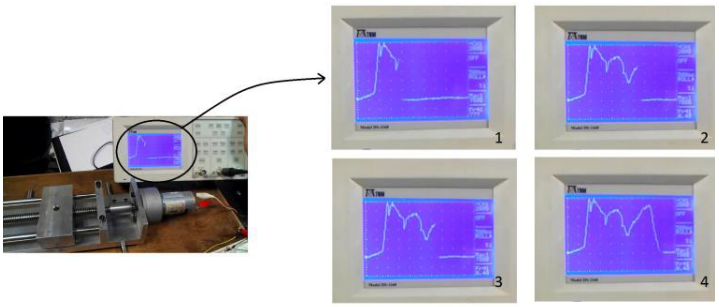

Fig. 6 Total force acting on a robot decreases (oscilloscope voltage) when using a XCSRR controller. Panels 1-4 show the efficiency of the controller to reduce the disturbing forces and keep the robot in stability.

4.3. Discussions

XSCRR is a great improvement in using classifier systems for real value problems. With this method of classifying, we are now capable of forming the dynamics of the systems that are measured and controlled with real value parameters such as voltage, temperature, and current. Although this method is developed for the use of XCS, it also can be utilized for other methods of classifying systems such as TCS. XCSRR is also comparable with other reinforcement methods in terms of computing cost, capability of linearity, and trainability. Our theory and experiments demonstrated that XCSRR keeps the stability of a humanoid robot, the ROBIAN biped, better than the neural networks method.

Choosing the optimum training parameters has been and still is a huge challenging issue in the training phase of classifiers systems to get the optimized training time and trainability. Table 1 showed the optimized parameters value for our problem that were obtained after a trial and error process. Although these parameters worked very well

for our problem, more studies are still needed in this area to get the best result especially for the threshold number (Fig. 2). Our suggestion is that one can use a changeable threshold number based on the action size and the standard deviation of the actions value.

5. Conclusions

This paper addresses the deficiency of XCSR for environment with real-value-input and real-value-input. Our novel approach, eXtended Classifier System for Real-Value-Input Real-Value-Output (XCSRR), is capable of working in fully real-value environment by composing action sets using a clustering method. XCSRR imitates the dynamic behaviour of the real system and updates its classifiers parameters during the training phase. Both theory and experiments point to a much improved performance when compared with other existing classifiers. In an exemplary biped robot stability control case, the performance of our method evidences a huge jump in reduction of disturbance forces, from previously reported 50% by neural networks method to the current 65% by XCSRR.

References

1. Holland, J.H., Hidden order: How adaptation builds complexity. 1995: Basic Books.
2. Holland, J.H., Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. 1975: U Michigan Press.
3. Holland, J.H., A mathematical framework for studying learning in classifier systems. Physica D: Nonlinear Phenomena, 1986. 22(1): p. 307-317.
4. Wilson, S.W., ZCS: A zeroth level classifier system. Evolutionary computation, 1994. 2(1): p. 1-18.
5. Wilson, S.W., Classifier fitness based on accuracy. Evolutionary computation, 1995. 3(2): p. 149-175.
6. Wilson, S.W., S. Wilson, and G. Xcs, Generalization in the XCS classifier system. 1998.
7. Wilson, S.W., State of XCS classifier system research, in Learning Classifier Systems. 2000, Springer. p. 63-81.
8. Wilson, S.W., Mining oblique data with XCS, in Advances in Learning Classifier Systems. 2001, Springer. p. 158-174.8.
9. Panahi, M. S., Yousefi, A. K., and Khorshidi, M. (2013). Combining accuracy and success-rate to improve the performance of eXtended Classifier System (XCS) for data-mining and control applications. Engineering Applications of Artificial Intelligence, 26(8): p. 1930-1935.
10. Butz, M.V., P.L. Lanzi, and S.W. Wilson, Function approximation with XCS: Hyperellipsoidal conditions, recursive least squares, and compaction. Evolutionary Computation, IEEE Transactions on, 2008. 12(3): p. 355-376.
11. Hurst, J., L. Bull, and C. Melhuish, TCS learning classifier system controller on a real robot, in Parallel problem solving from nature—PPSN VII. 2002, Springer. p. 588-597.
12. Wilson, S.W., Classifiers that approximate functions. Natural Computing, 2002. 1(2-3): p. 211-234.
13. Lanzi, P.L., et al. Prediction update algorithms for XCSF: RLS, kalman filter, and gain adaptation. in Proceedings of the 8th annual conference on Genetic and evolutionary computation. 2006. ACM.
14. Lanzi, P.L., et al., Generalization in the XCSF classifier system: Analysis, improvement, and extension. Evolutionary Computation, 2007. 15(2): p. 133-168.
15. Wilson, S.W., Get real! XCS with continuous-valued inputs, in Learning Classifier Systems. 2000, Springer. p. 209-219.15.
16. Vukobratović, M. and B. Borovac, Zero-moment point—thirty five years of its life. International Journal of Humanoid Robotics, 2004. 1(01): p. 157-173.
17. Lin, C.-M. and C.-H. Chen, Robust fault-tolerant control for a biped robot using a recurrent cerebellar model articulation controller. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 2007. 37(1): p. 110-123.
18. Kun, A. and W.T. Miller III. Adaptive dynamic balance of a biped robot using neural networks. in Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on. 1996. IEEE.
19. Sabourin, C. and O. Bruneau, Robustness of the dynamic walk of a biped robot subjected to disturbing external forces by using CMAC neural networks. Robotics and Autonomous Systems, 2005. 51(2): p. 81-99.
20. Marcu, T., B. Köppen-Seliger, and R. Stücher, Design of fault detection for a hydraulic looper using dynamic neural networks. Control Engineering Practice, 2008. 16(2): p. 192-213.
21. Song, E.-J. and M.-J. Tahk, Real-time neural-network midcourse guidance. Control Engineering Practice, 2001. 9(10): p. 1145-1154.
22. Su, S.-F., Z.-J. Lee, and Y.-P. Wang, Robust and fast learning for fuzzy cerebellar model articulation controllers. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 2006. 36(1): p. 203-208.
23. Kim, B.S. and A.J. Calise, Nonlinear flight control using neural networks. Journal of Guidance, Control, and Dynamics, 1997. 20(1): p. 26-33.
24. Beer, R.D., Parameter space structure of continuous-time recurrent neural networks. Neural Computation, 2006. 18(12): p. 3009-3051.
25. Pineda, F.J., Generalization of back-propagation to recurrent neural networks. Physical review letters, 1987. 59(19): p. 2229.
26. Rumelhart, D.E., G.E. Hinton, and R.J. Williams, Learning internal representations by error propagation. 1985, DTIC Document.
27. Mohamed, B., F. Gravez, and F. Ouezdou, Emulation of the dynamic effects of human torso during a walking gait. Journal of Mechanical Design, 2004. 126(5): p. 830-841.
28. Hénaff, P., et al., Real time implementation of CTRNN and BPTT algorithm to learn on-line biped robot balance: Experiments on the standing posture. Control Engineering Practice, 2011. 19(1): p. 89-99.