Creating Learning Sets for Control Systems Using an Evolutionary Method

Marcin Gabryel[1], Marcin Woźniak[2], and Robert K. Nowicki[1]

[1] Department of Computer Engineering, Czestochowa University of Technology, Al. Armii Krajowej 36, 42-200 Czestochowa, Poland
[2] Institute of Mathematics, Silesian University of Technology, ul. Kaszubska 23, 44-100 Gliwice, Poland
{Marcin.Gabryel,Robert.Nowicki}@kik.pcz.pl, Marcin.Wozniak@polsl.pl

Abstract. The acquisition of the knowledge which is useful for developing of artificial intelligence systems is still a problem. We usually ask experts, apply historical data or reap the results of mensuration from a real simulation of the object. In the paper we propose a new algorithm to generate a representative training set. The algorithm is based on analytical or discrete model of the object with applied the k–nn and genetic algorithms. In this paper it is presented the control case of the issue illustrated by well known truck backer–upper problem. The obtained training set can be used for training many AI systems such as neural networks, fuzzy and neuro–fuzzy architectures and k–nn systems.

Keywords: genetic algorithm, control system, training data acquisition.

1 Introduction

Very important phase in the process of designing solution based on artificial intelligence, e.g. artificial neural networks, fuzzy and neuro-fuzzy architectures [1], [2], [3], type-2 neuro-fuzzy systems [4], [5], as well as its ensambles [6] is the acquisition of knowledge. The expert, fuzzy, neuro-fuzzy, rough systems and it's hybrids [7], [8] can apply the knowledge that come from human experts. In many projects this is a main source which determines the prototypes of rules. However, usually it is insufficient. These systems require also the other type of knowledge - the set of examples of proper operation of the system. This type of knowledge is necessary for tuning and evaluating the solution. In the case of developing neural networks and often even neuro-fuzzy architectures the set of examples, i.e. training set, is the one and only form of knowledge used both for training and evaluating [9], [10]. Moreover, the set of examples can be used to obtain other forms of knowledge including rules [11], [12], [13], [14], [15], [16]. As we see the set of examples is quite versatile knowledge form. The common practice in training and evaluating new AI systems is to use available to the public sets - benchmarks [17]. Such proceedings are obviously unsuitable in a real problem. During the building of medical diagnostic system the source of the case is historical diagnosis of real patients. During the development of the control

L. Rutkowski et al. (Eds.): SIDE 2012 and EC 2012, LNCS 7269, pp. 206–213, 2012.

or diagnostic system the samples come from measurement of the real objects or from model simulation. When we neglect the cost of first method and problem with imperfection of model in a second one, we have still two problems. The first one is poor representativity of obtained set. The second one is more serious. To proceed the work or simulation, in one of structures depicted in Fig. 1 or 2, we have to know how to control the object or detect the threat of damage. We can use the past controller or human operator. The training set of Truck backer–upper control problem [18], [19] (see Section 2) used in many experiments and publications comes from registration the trajectories when the truck was being controlled by the driver.

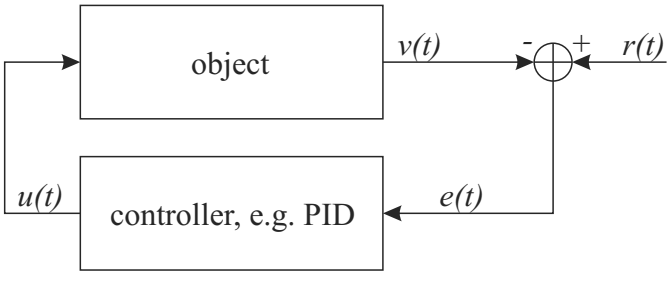

Fig. 1. Classical control system

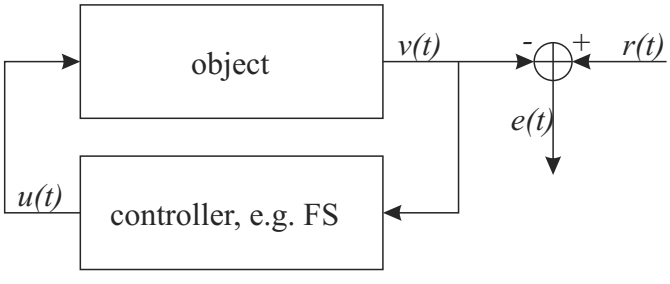

Fig. 2. AI control system

In the next part of the paper we will present the new method to generate a representative training set without the proper knowledge about control procedure. The algorithm is based on an analytical or discrete model of object with applied the k–nn [20] and genetic algorithms.

2 Truck Backer–Upper Control Problem

The problem of truck parking has been proposed and used as an example issue of non-linear control of the object by Nguyen and Widrow [18] and also used by Kong'a, Kosko [19]. It has become quite popular in experiments of control systems.

Truck goes to the back of a constant speed, and its goal is to reach the ramp. The parameter controlling the steering angle θ. State of the vehicle in the parking determine four parameters: coordinates x and y – determine the location of the parking lot, ϕ – angle to the axis Y of the vehicle, θ – turn the wheels of the vehicle. Truck moves backwards in the following steps in order to reach axis of the ramp (point $x = 0$, $\phi = 0$). Distance from the dock is ignored, since goal is assumed that any further driving in a straight line is not a problem. Problem posed in the article is to generate a learning set based on the model describing the motion of the vehicle in the following steps. The individual data within the learning set should be chosen in such a way that for a given position in which the vehicle is, in the next step to bring the vehicle to the ramp turning the wheels. The next steps of the simulation (vehicle's motions) describe the following formulas:

$$\begin{aligned} x(k+1) &= x(k) + \delta t v \cos\left(\phi(k)\right), \\ y(k+1) &= y(k) + \delta t v \sin\left(\phi(k)\right), \\ \phi(k+1) &= \phi(k) + \frac{\delta t v \sin(\theta(k))}{L}, \end{aligned} \tag{1}$$

where ϕ – angle to the Y axis, L – length of the vehicle, θ – steering angle, v – vehicle speed, δt – time interval, k – iteration in the simulation, (x, y) – vehicle position. The range of variation of the variables is as follows: $x \in (-150, 150)$, $y \in (0, 300)$, $\phi \in (-45, 45)$, $\theta \in (-180, 180)$. The problem is shown in Fig. 3.

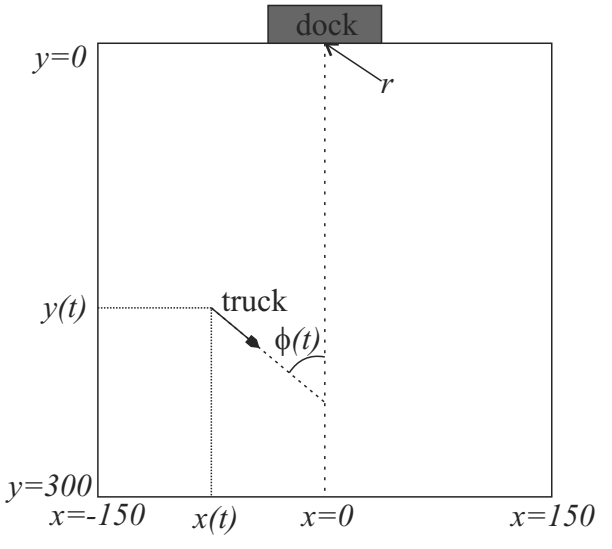

Fig. 3. Model of vehicle parking

3 *k* Nearest Neighbor Controller

To check the quality of the individual training sets we could build a Controller on each of them. This solution has one serious defect. Constructing the controller is time-consuming. The results can depend on the type of controller and above all it do not allow an individual assessment of the samples. Rating would apply to all drivers and so the entire training set.

In the proposed algorithm, there was proposed a special Controller, based on the algorithm of k–nn [20]. The driver will be the used knowledge contained in a single set of learning samples. The set is composed of M samples, each of them has two parts — the value in the input space $\mathbf{v}_i \in \mathbf{V}$ and the corresponding baseline values in the output space of $u_i \in \mathbf{U}$. The fitness function $f_i = F(X_i)$ is assigned to each sample (see Section 4).

The control system shown in Fig. 2 will be used in this case. State of the controlled object is described by the vector $\mathbf{v}(t) \in \mathbf{V}$, which is passed to the input driver. In the collection of samples used by the driver there is no sample for which $\mathbf{v}_i = \mathbf{v}(t)$ (omitting digitizing measurement and representation of samples, this situation is infinitely improbable). To design the control value $u(t)$ there will be used all samples contained in a set, each depending on the distance

$$d_i(t) = ||\mathbf{v}(t) - \mathbf{v}_i|| \tag{2}$$

and fitness function for each sample f_i according to

$$u(t) = \frac{\sum\limits_{i=1}^{M} g\left(d_i(t)\right) f_i u_i}{\sum\limits_{i=1}^{M} g\left(d_i(t)\right) f_i}, \tag{3}$$

where g is a not increasing function for positive values in the space of variations $d_i(t)$ defined by the formula (2).

We can also consider the inclusion $k < M$ the nearest $\mathbf{v}$ samples, however this requires a sort to the distance d_i. Controll value willbe calculated by the formula

$$u(t) = \frac{\sum\limits_{i:\ d_i \in \Omega_k(t)} f_i u_i}{\sum\limits_{i:\ d_i \in \Omega_k(t)} f_i}, \tag{4}$$

where $\Omega_k(t)$ is a set of k lowest values of $d_i(t)$. Hence the name of the proposed controller.

4 Testing Procedure and Results

Implemented system was tested using truck. Due to discrete nature of the model used in the model integrals were replaced by the sums of the successive steps of the simulation. Described problem will be solved using the evolutionary strategy

(μ, λ) (see [21], [22]). It is well known that evolution strategies are distinguished by self-adaptation of additional strategy parameters, which enable them to adapt the evolutionary optimization process to the structure of the fitness [23]. It is assumed that the chromosome of an individual is formed by a pair of real-valued vectors (X, σ). The strategy vector σ is a subject to a random mutation according to the formula

$$\sigma_i' = \sigma_i \cdot e^{\tau' \cdot N(0,1) + \tau \cdot N(0,1)}, \tag{5}$$

where $\tau' = \frac{1}{\sqrt{2L}}$, $\tau = \frac{1}{\sqrt{2\sqrt{L}}}$, $i = 1, \ldots, L$ and L is the length of the chromosome.

The mutation in the algorithm is based on the formula

$$X_i' = X_i + \sigma_i' \cdot N_i(0,1), \tag{6}$$

replaces the parent X' with the descendant X. The standard evolution strategy based on mutation is extended by using of a uniform recombination operator [21] . In a single chromosome is encoded $M = 50$ possible samples (triplets of numbers (X, ϕ, θ)). The length of the chromosome is therefore $L = 3M = 150$. The proposed algorithm uses an evolutionary algorithm in addition to the calculation algorithm of the additional k-nearest neighbor algorithm (k–nn) [20]. The algorithm consists of several steps:

1. Initialize the algorithm – Enter the number of steps N.
2. For $k = 1, \ldots, N$, repeat steps 3-6.
3. Random selection of the initial position of truck:
   (a) $x_k = N(0,1) \cdot (x_{max} + x_{min}) + x_{min}$.
   (b) $\phi_k = N(0,1) \cdot (\phi_{max} + \phi_{min}) + \phi_{min}$.
4. Initiation of an evolutionary strategy (μ, λ).
   (a) Determination of parameters of an evolutionary strategy.
   (b) Random the vectors X_j of initial population for $j = 1, \ldots, \mu, i = 1, \ldots, M$.
      i. $X_{j,i\cdot 3} = N(0,1) \cdot (x_{max} + x_{min}) + x_{min}$.
      ii. $X_{j,i\cdot 3+1} = N(0,1) \cdot (\phi_{max} + \phi_{min}) + \phi_{min}$.
      iii. $X_{j,i\cdot 3+2} = N(0,1) \cdot (\theta_{max} + \theta_{min}) + \theta_{min}$.
5. Commissioning strategy (μ, λ) for 100 generations of evolution and the calculation of fitness function according to algorithm:
   (a) $F(X_j) = 0$.
   (b) Perform a full simulation of motion for the point (x_k, ϕ_k):
      i. $t = 0$.
      ii. Find the turning angle θ of wheels for your vehicle from all samples using the algorithm k–nn.
      iii. Move the vehicle to a new position $x(t+1), y(t+1)$ according to equations (1), $t = t + 1$.
      iv. $F(X_j) = F(X_j) + x(t) + \phi(t)$ (see Fig. 4).
      v. Finish the simulation of T steps if the vehicle approaches the ramp, otherwise go to step ii.
6. The introduction of all the samples with the winning chromosome which participated in the k–nn algorithm and adding them to the learning set Ω.

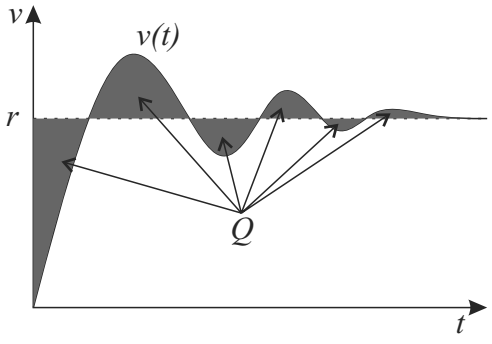

Fig. 4. Method of fitness function calculate

The algorithm uses the following designations: N – the number of subjects of the vehicle states, $x_{\max}, x_{\min}, \phi_{\max}, \phi_{\min}, \theta_{\max}, \theta_{\min}$ – maximum and minimum values are defined in Section 2, $N(0,1)$ – random number generated from the range $(0,1)$, M – number of samples encoded in the chromosome, t – iteration in the simulation, $F(X_j)$ – value of fitness function for the j-th chromosome.

Generated samples are collections of three numbers (x, ϕ, θ), where for input data x and ϕ is adjusted steering angle θ to make the vehicle closer to the ramp in the next move. Algorithm in the steps satisfies the conditions $x = 0$, $y = 0$ and $\theta = 0$. To simplify the operations it is assumed that the y position of the truck will not be taken into account.

The idea behind the algorithm is to generate many of the initial states of the model, and then evolutionary selection of parameters affecting its performance taking into account his current state. After finishing the simulation the

Table 1. The results obtained in the algorithm

No.	x	ϕ	θ	No.	x	ϕ	θ
1	-98.19	-14.16	60.00	16	26.36	66.86	-5.37
2	-76.53	-57.65	-3.38	17	31.28	15.75	19.96
3	-53.56	-15.10	56.88	18	37.56	48.22	0.03
4	-36.71	-34.93	60.00	19	39.83	54.27	2.79
5	-33.48	7.49	60.00	20	40.56	51.23	5.63
6	-30.20	-2.35	36.72	21	45.28	61.24	-20.88
7	-16.42	48.89	0.33	22	49.29	39.68	-38.75
8	-16.35	34.85	29.16	23	50.51	61.57	-41.48
9	-9.61	32.19	-12.92	24	57.68	24.44	-2.31
10	-9.51	81.89	-16.01	25	57.69	30.45	-60.00
11	-6.11	41.96	-5.69	26	73.51	117.22	-1.39
12	-2.96	118.15	-4.17	27	79.76	110.37	-10.74
13	-1.07	33.32	-9.31	28	90.79	31.61	-2.13
14	3.43	95.29	-4.33	29	98.52	57.67	-19.50
15	4.09	7.98	-21.92	30	105.12	-43.77	-60.00

best chromosomes selected are those samples that have been generated by the algorithm k–nn with the operation of fitness function. The algorithm has been implemented in Java with the following parameters of the algorithm: $N = 10, \mu = 10, \lambda = 50, M = 50, k = 5, T = 500$. The generated sequence of learning states can be found in Table 1. Analyzing the samples can be seen that the generated sequence is appropriately diverse and individual states (X, ϕ) corresponds to the appropriate response to the steering wheel θ.

5 Final Remarks

The article proposed a new method to generate a collection of representative samples, which can be used in the preparation of the target driver based on various methods of artificial intelligence, but also other, using the knowledge in the form of examples [24]. This method can be useful when we have a model of controlled object, and we have no knowledge of it's proper control. Conducted experiments confirm it's usefulness. An important restriction only need to carry out a large number of simulation control process to determine the assessment of individual sets of samples and the same samples. It is therefore time-consuming procedure. Further work should therefore be carried out in the direction of reducing time-consuming solution, eg. by using some knowledge prior to generating the initial population.

References

1. Rutkowska, D., Nowicki, R.K.: Implication-based neuro–fuzzy architectures. International Journal of Applied Mathematics and Computer Science 10(4), 675–701 (2000)
2. Rutkowski, L., Cpałka, K.: A general approach to neuro - fuzzy systems. In: Proceedings of the 10th IEEE International Conference on Fuzzy Systems, December 2-5, vol. 3, pp. 1428–1431 (2001)
3. Rutkowski, L., Cpałka, K.: A neuro-fuzzy controller with a compromise fuzzy reasoning. Control and Cybernetics 31(2), 297–308 (2002)
4. Starczewski, J., Rutkowski, L.: Interval type 2 neuro-fuzzy systems based on interval consequents. In: Rutkowski, L., Kacprzyk, J. (eds.) Neural Networks and Soft Computing, pp. 570–577. Physica-Verlag, Springer-Verlag Company, Heidelberg, New York (2003)
5. Starczewski, J.T., Rutkowski, L.: Connectionist Structures of Type 2 Fuzzy Inference Systems. In: Wyrzykowski, R., Dongarra, J., Paprzycki, M., Waśniewski, J. (eds.) PPAM 2001. LNCS, vol. 2328, pp. 634–642. Springer, Heidelberg (2002)
6. Korytkowski, M., Rutkowski, L., Scherer, R.: From Ensemble of Fuzzy Classifiers to Single Fuzzy Rule Base Classifier. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2008. LNCS (LNAI), vol. 5097, pp. 265–272. Springer, Heidelberg (2008)
7. Nowicki, R.: Rough–neuro–fuzzy structures for classification with missing data. IEEE Trans. on Systems, Man, and Cybernetics—Part B: Cybernetics 39 (2009)
8. Nowicki, R.: On classification with missing data using rough-neuro-fuzzy systems. International Journal of Applied Mathematics and Computer Science 20(1), 55–67 (2010)

9. Scherer, R.: Boosting Ensemble of Relational Neuro-fuzzy Systems. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Żurada, J.M. (eds.) ICAISC 2006. LNCS (LNAI), vol. 4029, pp. 306–313. Springer, Heidelberg (2006)
10. Korytkowski, M., Scherer, R., Rutkowski, L.: On combining backpropagation with boosting. In: 2006 International Joint Conference on Neural Networks, Vancouver, BC, Canada, pp. 1274–1277 (2006)
11. Wang, L.X.: Adaptive Fuzzy Systems and Control. PTR Prentice Hall, Englewood Cliffs (1994)
12. Wang, L.X., Mendel, J.M.: Generating fuzzy rules by learning from examples. IEEE Transactions on Systems, Man, and Cybernetics 22(6), 1414–1427 (1992)
13. Grzymala-Busse, J.W.: LERS — a system for learning from examples based on rough sets. In: Sowiski, R. (ed.) Intelligent Decision Support: Handbook of Applications and Advences of the Rough Sets Theory, pp. 3–18. Kluwer, Dordrecht (1992)
14. Grzymala-Busse, J.W.: An overview of the LERS1 learning systems. In: Proceedings of the 2nd International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, pp. 838–844 (1989)
15. Nozaki, K., Ishibuchi, H., Tanaka, H.: A simple but powerful heuristic method for generating fuzzy rules from numerical data. Fuzzy Sets and Systems 86, 251–270 (1997)
16. Sugeno, M., Yasukawa, T.: A fuzzy-logic-based approach to qualitative modeling. IEEE Transactions on Fuzzy Systems 1(1), 7–31 (1993)
17. Mertz, C.J., Murphy, P.M.: UCI respository of machine learning databases, `http://www.ics.uci.edu/pub/machine-learning-databases`
18. Nguyen, D., Widrow, B.: The truck backer-upper: An example of self-learning in neural network. IEEE Control Systems Magazine 10(3), 18–23 (1990)
19. Kong, S.G., Kosko, B.: Comparison of fuzzy and neural truck backer upper control system. In: Proceedings of IJCNN 1990, vol. 3, pp. 349–358 (1990)
20. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Transactions on Information Theory 13(1), 21–27 (1967)
21. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer, Heidelberg (1998)
22. Eiben, A.E.: Introduction to Evolutionary Computing. Springer, Heidelberg (2003)
23. Back, T.: Evolutionary Algorithms in Theory and Practice. Oxford University Press, Oxford (1996)
24. Scherer, R.: Neuro-fuzzy Systems with Relation Matrix. In: Rutkowski, L., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2010. LNCS (LNAI), vol. 6113, pp. 210–215. Springer, Heidelberg (2010)