Neural networks in classifier systems (NNCS): An application to autonomous navigation.

Lubnen N. Moussi
DSCE-FEEC-UNICAMP
lubnen@dsce.fee.unicamp.br

Ricardo R. Gudwin
DCA-FEEC-UNICAMP
gudwin@dca.fee.unicamp.br

Fernando J. Von Zuben
DCA-FEEC-UNICAMP
vonzuben@dca.fee.unicamp.br

Marconi K. Madrid
DSCE-FEEC-UNICAMP
madrid@dsce.fee.unicamp.br

Abstract: We show in this paper an original version of a classifier system using neural networks as its classifiers. The main point is to determine whether neural networks, as universal approximators, can enrich classifier systems performance. To give a proper answer to this inquiry, the research was divided into two phases. The results presented here are related to the first one: an initial incursion characterized by the application of such classifier system to an autonomous vehicle control in a computational environment, and compare its performance with that produced by a conventional classifier system. This is done in a rather qualitative manner, where the main objective is to provide evidences that neural networks can really enrich a classifier system performance. We also present important aspects related to the second and final phase of the research: the exploration of well-developed learning methods and concepts of ensemble theory to improve the performance of the classifier system.

Key Words: Classifier systems, evolutionary computation, neural networks, autonomous navigation.

1 Introduction

Classifier systems (CS) [Booker et al., 1989] are learning systems based on propositional rules operating in parallel, and also on *credit assignment* and *rule discovery*. They were developed originally by Holland and his team [Holland, 1975] with the purpose of modeling natural evolutionary processes and presenting adaptation and learning characteristics.

However, propositional rules represents a limitation to classifier systems: many authors demonstrated their inefficacy in intelligent systems implementation [Simon, 1996]. Classifier systems present flexibility of operation and mechanisms of structural adaptation that few intelligent systems have shown till now.

Some relevant questions arise. Is it possible to avoid propositional rules and at the same time take advantage of all the qualities that these systems carry with them? Can neural networks substitute the propositional rules of the classifier systems? To answer these questions in a pragmatic way, we conceived a research project divided into two phases. First, we developed a prototype of a hybrid classifier system that substitutes propositional rules by neural networks. This prototype has to be tested in some practical application that permits a comparison with a conventional classifier system. After that, depending on the results, it will be possible to proceed a deeper analysis of this new version of classifier systems, trying to evaluate its application scope and also computational aspects involved. The present paper documents the execution of the first phase of this project and raises relevant questions to go towards the second one.

We conceived, as the practical application, an autonomous vehicle control in a computationally simulated environment. The results obtained were encouraging. The approach of this paper is original, at least to the extent we know. There are, of course, some applications that could be considered similar, as, for example, *Evolutionary Reinforcement Learning* (ERL) [Ackley & Littman, 1991], but the similarities are restricted to punctual aspects. As a matter of fact, we think to apply some of Akcley and Littman ideas related to their ERL in future NNCS developments.

In this article, we initially present basic concepts for conventional classifier systems, followed by the implementation of a new version with Neural Networks (NNCS). Afterwards, we discuss the details of the practical implementation, the simulations executed and finally our conclusions and relevant considerations for the studies to come.

2 Classifier Systems

There are few references to classifier systems in the literature. Some of the most important are: the book that launched its proposal [Holland, 1975], the article of Booker et al. [Booker et al., 1989], the book of Goldberg [Goldberg, 1989] and the thesis of Richards [Richards, 1995], which gives a detailed explanation of classifier systems.

Among other advantages, the classifier systems possess the following characteristics: they are indicated to operate in environments that typically exhibit new and successive events, accompanied by noise and irrelevant data; they are appropriate in situations were there exists the necessity to act in a continuous manner and frequently in real time; they possess the ability to be guided by implicit or non exact goals, as

well as they promote learning based on sparse rewarding. It is important to stand out that few other computational tools exhibit a so vast list of application possibilities.
These characteristics are possible because the classifier systems discover new categories and concepts by means of the regularities found in the environment, as they are relevant for the accomplishment of the desired goals. In the same manner, the classifier systems utilize the flow of information met in the way to the goal to refine their model of the environment and doing so they associate appropriated control actions with the situations met along the search.

2.1 Classifier Systems Architecture

The classifier system is based on a hierarquical architecture, where the lower level is presented in figure 1.

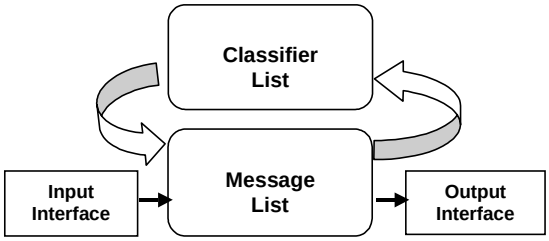

Figure 1: Basic Components of the Classifier System

This architectural level operates in the following way: the input interface collects information from the environment through sensors and codifies them into messages that will be put in the message list. Observe that not only the input interface sends message to the message list, but also the classifiers. One message in the messages list is a string of characters composed of two parts: the initial characters identify who had sent the message and the others identify occurrences in the environment, detected by the sensors.
The message example shown bellow corresponds to an arbitrary situation extracted from the application environment.

1	0	0	1	1	0	1	0	0	1	0

The first two bits '**10'** identify the input interface as the message's actor. The following nine bits describe the existence of obstacles inside the vehicle's visual field, each bit corresponding to a sub-region. Bit **0** corresponds to absence of obstacles, and bit **1** indicates presence of obstacle.
The classifier list is composed of *classifiers* (rules), which are string of characters divided in three parts. In the first part there are the characters that identify the type of *consequent* (it will be shown in the sequence). In the second there are the characters that constitute its *condition* (or *antecedent*). In the third there are the characters that correspond to the *consequent* (or *action*) of the classifier, witch is the message that will be sent to the message list, if the classifier is selected to become active.

0	1	#	1	#	0	#	1	0	#	#	0	1

In the example above, the first two bits, by an arbitrary convention, indicate that the consequent is a message for the output interface. The following nine bits correspond to the condition, where the character **"#"**, means *don't care*. It occurs a *matching* when the condition of a classifier equals a message of the message list. Each symbol # can take either value 0 or 1, favoring the matching.
In the third part, the consequent may be of two types: a message that will be used by the classifiers in the next cycle, or a message for the output interface to provide a determined action.
The classifiers that match one or more messages from the message list have the chance to post new messages. Following the example, the message posted is shown bellow:

0	1	0	1

This message is then send to the message list that receives messages from the input interface and from the classifier list. The output interface monitors constantly the message list, and, when it finds output messages, it takes those messages from the list executing the instructions determined by them. And more, the output interface is able to solve possible impasses created by conflicting messages.
This initial loop differs in nothing from the propositional rules mechanism. It assumes that the classifier list represents a set of rules that possesses the required knowledge to process the information from the environment. However, differing from a usual rules base, where the knowledge is acquired from one expert, in this technique the rules (*classifiers*) in the classifier list are usually determined by an evolutionary mechanism.
The classifiers can be generated initially by a random process, not discharging the possibility of introducing initial knowledge. So, the task is to discover which classifiers can produce appropriate behavior, generate new classifiers and discharge those that are or come to be inappropriate, considering the environment variability. This mechanism of rule learning is implemented by the joint application of the *credit assignment algorithm* and the *genetic algorithm* (GA), that actuate in a higher hierarchical level.

2.2 Credit Assignment

For each classifier, stipulate a value of *strength*, which is the measure of confidence in a classifier in face of a given situation. Classifiers that *match* some message in the message list will make a *bid* to acquire them, and the one that offers greater bid will have greater probability to post its message. The following lines illustrate the details of the implementation.

Bid of classifier C:
B(C,t) = b.R(C).s(C,t)
R(C) - *specificity* of the classifier condition (condition's length minus number of symbols #, divided by condition's length)
b - constant lesser then 1 (for example, 1/8 or 1/16)
s(C,t) - *strength* of classifier C at instant t
B - *bid*, it determines the probability that the classifier posts its message
Strength of classifier C:
s(C,t) = s(C,t) - B(C,t)
Reward: If the action generated by the classifier turns to be appropriated, it will receive its bid back, plus a reward R:
s(C,t+1) = s(C,t) + B(C,t) + R

One of the most famous algorithms for the implementation of credit assignment is the so called *Bucket Brigade* [Booker et al., 1989]. It utilizes the idea that classifiers can post messages that will generate new messages, composing a sequence of messages to generate an action and, because of this, all those messages contributing to the action have their credit back plus a reward if the action turned to be successful, or nothing otherwise.
In this work, we'll not use the Bucket Brigade in all its complexity, but a simpler version based on the same principles.

2.3 Genetic Algorithm (GA)

The process of rules discovery in the classifier systems utilizes a genetic algorithm.
Basically, the GA selects the classifiers with greater strengths to reproduce, generating new individuals by their recombination and mutation. The new classifiers generated take the places of the weaker ones, modifying the classifier set of the system [Holland, 1975].

3 The application and its environment

The mobile vehicle, as in figure 2, has a circular form, with a diameter of 50 pixels and visual field divided into 9 regions. The objective of the navigation is to avoid collision against obstacles. Notice that, as the initial composition of the classifier list is defined at random, no initial knowledge about how to attain the objective is available.

The environment is a reticulate of 700x700 pixels. The circular marks represent pilasters

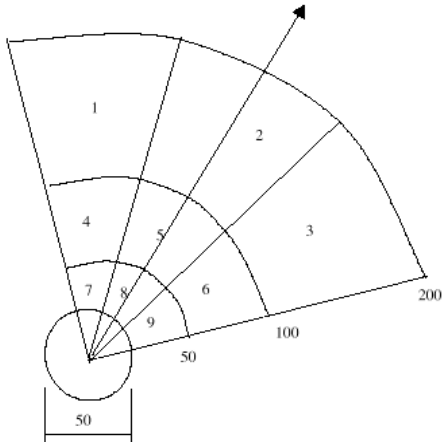

Figure 2: Vehicle and its visual field.

without dimension (zero diameter), as seen in figure 3.

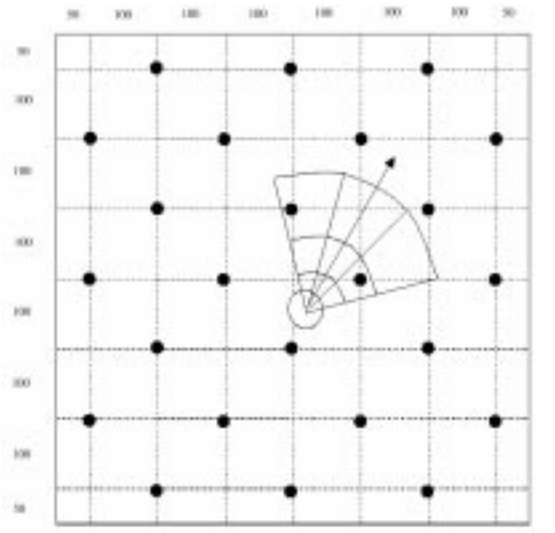

Figure 3: Simulation Environment

4 NNCS –Neural Networks in Classifier System

4.1 Why to use neural networks

The inclusion of neural networks to compose the classifiers has implications associated with their characteristics of universal approximators and the possibility to manipulate real numbers. In other words, it represents more than a mere substitution of numeric methods because neural networks provide new possibilities to enrich CS performance, by means of more sophisticated behaviors then the normally got without them.
The neural network architecture adopted is the multiplayer perceptron [Haykin, 1999], with a proper number of neurons at the hidden layer to provide the necessary approximation capability.

4.2 Neural networks as classifiers

Each classifier will be taken as an array that codifies the weights of two neural networks: one for *evaluation* and the other for *action*, as in figure 4. The evaluation neural network substitutes the matching in the conventional CS. Its hole is to determine the degree of interest for a message in the message list. The action neural network generates a new message that is posted in the message list. Both networks process the same input, which comes from the input interface.

The remaining aspects of the mechanism are exactly the same as in the normal CS.

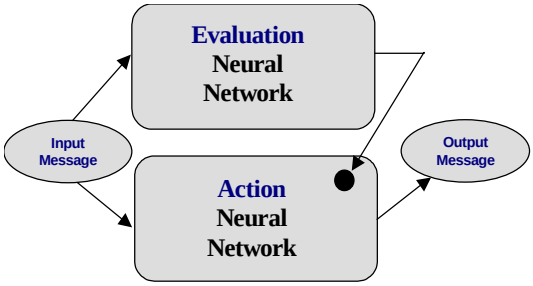

Figure 4: **Neural networks for evaluation and action.**

4.3 Configuration of the NNCS

The neural networks are multiplayer perceptrons with one intermediate layer of two neurons. The output layer of the evaluation network has only one neuron, which provides the specificity level. The output layer of the action network has one or two neurons, depending whether we consider only the vehicle orientation or also its speed. The activation function for the intermediate layer is the hyperbolic tangent and the output layer has linear activation.

The input to the neural networks are messages from the input interface plus polarization. Each message is an array of 9 bits. So, Evaluation Neural Network, and Action Neural Network will have 23 or 26 weights.

Some simulations may also have velocity and orientation as input, adding two positions to the input array. At the beginning, the weights of the neural networks are generated randomly, and then evolved based on a genetic algorithm.

The GA uses two-point crossover. Since they are real numbers, it's considered the sum, difference or mean of the values between the points, to get a larger range of values. Mutation for the initial tests is the inversive, however inductive mutation should provide more diversity for the values of weights.

The GA selects the **'n'** better classifiers and the *Roulette Wheel* algorithm [Goldberg, 1989] is applied to form the pars. Two criteria are adopted to start the GA: number of iterations or number of collisions, which occurs first.

- **Number of collisions:** a lot of collisions may indicate the necessity of better classifiers, and the GA will implement the search.
- **Number of iterations:** the absence of collisions doesn't mean that the behavior can't be made better.

It's important to observe that the configuration chosen for this first prototype of NNCS is the simplest possible, developed solely with the intention of attesting competence to achieve the goals.

5 Simulation results

To verify the functionality of NNCS, we considered just orientation or orientation and velocity as control variables. The objective is: to run without colliding with any obstacle.

So, the objective will be fulfilled if, after a learning stage, the vehicle could move without collisions for a great number of iterations. This behavior normally will be characterized by the vehicle movement describing orbits of a great variety of types, surrounding one or more pilasters.

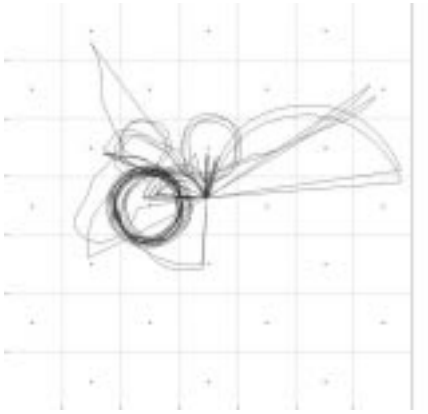

Figure 5: Vehicle Trajectory

Figure 5 illustrates vehicle movement, where the vehicle finally finds its equilibrium in an almost circular orbit around the pilaster to the left of the environment center. The straight lines represent de vehicle being placed back to the center after a collision.

The simulation results are divided into 7 steps.

5.1 Initial Step: Parameters adjustment

Here, the main objective is to get the parameters adjusted to obtain velocities and orientation deviations with values appropriate for the environment. Only one classifier was allowed to post message, the one with higher evaluation. It was observed the occurrence of elliptical trajectories with a center displacement, resulting in helicoids. With the conventional CS, for the same circumstances, only circular orbits were

observed. This difference may be consequence of using real numbers but also of substituting classifiers by neural networks. We noticed difficulties in learning, fact attributed to the use of only one winner to bid, reducing the exposition of the classifiers.

5.2 Step 2: 5 winners

The 5 classifiers with better matching were selected to bid. We used the following formula for calculating the bid in the initial step:

B = 0.1**strength*

And here we changed it to:

B = *matching***strength*

were *matching* corresponds to the output of the evaluation neural network and will be utilized to measure the classifier specificity; greater its value, more specific the classifier. Bids and strengths can have positive and negative values. Figure 6 presents the results of one test. It can be noticed positive and negative velocities, and also with very high values, fact not of concern at this time. By iteration 1100, the vehicle enters a central orbit, running backwards. The vehicle has no sensor looking backwards, but it can learn, even so, by the environment symmetry.

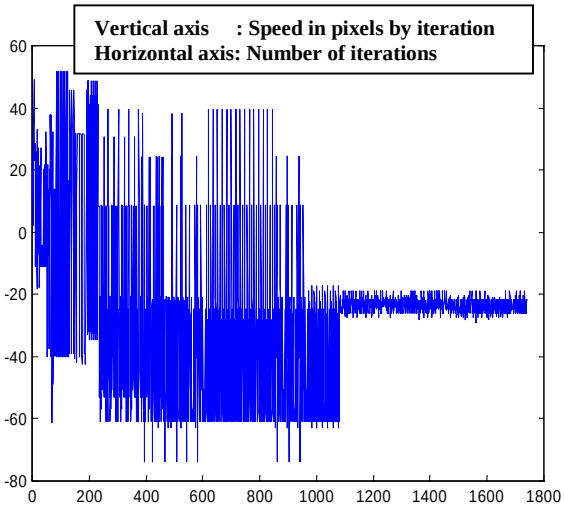

Figure 6: Speed versus number of iterations.

Figure 7 presents a graphic considering the number of iterations between collisions, displayed by each bar. It can be seeing, at the beginning, about 53 iterations without collisions. It's obvious that it was not learned, illustrating the possibility of occurring innate knowledge, which anyway is just casual and even not desired, because it prevents the system to learn new abilities.

This experiment was observed until iteration 6000, maintaining a central orbit, sometimes with some modifications on its trajectory, describing more complex orbits, combining ellipses with helicoids.

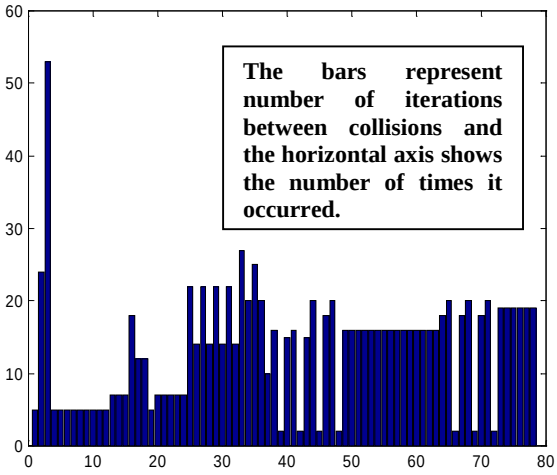

Figure 7: Iterations between collisions. After this learning phase, no additional collision was observed.

5.3 Step 3: Velocity as input and orientation controlled by displacement

As velocity varies with time, we decided to feed it back as an additional input to the classifiers. On the other hand, orientation is not required to be considered as input to the classifiers, because we use the control action associated with displacement of the current orientation, and not orientation itself.

5.4 Step 4: Velocity (*v*) and orientation displacement (*dteta*) inside a range of values

We verified wether it's possible to make parameters adjustment to obtain absolute values of *v* and *dteta* inside a range, without degrading the performance. We tried to obtain both of them with their maximum around 25, that we considered appropriate for this phase. We got, without great difficulties, *v* between –30 and 40, with its major part between –10 and 20. It doesn't look easy to get better values by this approach. See figure 8.

Another approach, to be considered in the next phase, is to pennalize the classifiers giving values outside the required range. That is, to consider the obtention of values inside the range will be another goal of the system.

Now, how to avoid negative values for velocity, preventing the use of back sensors? One of the possible solutions is the adoption of a post-processing device to impose a specific range of values to the neural network output.

Another possibility, instead of velocity we could consider acceleration, and negative acceleration being implemented as the brakes of the vehicle. It's a quite more sophisticated mathematical model that is not considered in this introductory study.

5.5 Step 5: Only *dteta* as a control variable

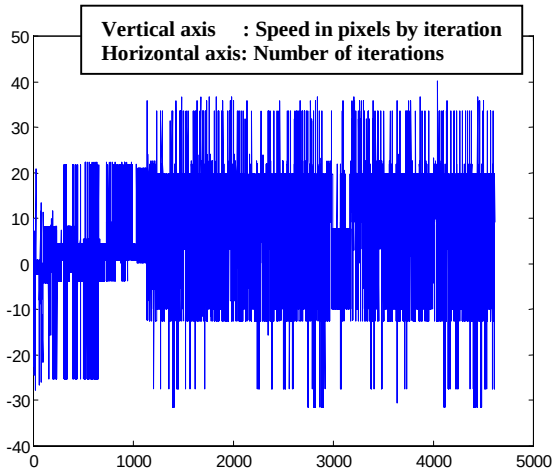

Figure 8: Speed versus iterations.

To get simpler situations to observe and so being able to get deeper understanding of what is fundamental for the learning process, we eliminated velocity variations. So, the vehicle mission is just to run with fixed velocity.
With that, the actuator will have just one neuron in its output layer.
Learning becomes more evident, but the performance varies significantly along training. We have two points to consider.

- The learning mechanism must be desigend do discover and maintain appropriate classifiers. The method in use based on bid (penalization) and reward looks like not being sufficient.
- Also, the GA gives results in opposing directions: new knowledge generates new behaviors that have to be evaluated and classified.

In the next step we introduce reinforcement learning, trying to solve the first point. For the second, the solution is related to finding more adequate periods to put the GA in action.

5.6 Step 6: Additional reinforcement in learning

It would be ideal to implement the *Bucket Brigade* now, but we decided for a structurally more simplified algorithm, which would point to its future implementation viability. The principle is to penalize a sequence of actions that leads to a collision.
In average, the vehicle at 15 pixels per iteration requires 5 iterations to reach a pilaster. We will consider the last three actions as responsible for the collision, because in this meantime the vehicle had the chance to avoid it.
Making this, it was observed improvement in learning.

5.7 Step 7: Pilasters moving

Moving the pilasters determines a more varied information flow coming from the environment. This resulted in the necessity of a greater number of iterations to acquire a good level of performance.

6 NNCS – Final considerations

The results of this work, by its nature, are qualitative.
We achieved the objective of implementing all the functions of conventional CS utilizing NNCS. As expected, the process of weight adjustment is time-consuming, but not superior to the requirements to adjust parameters in conventional CS.
The results are surprising, as we utilized very simple techniques in all the steps. It's clear that doing so the solution was very dependent on the initial conditions.
In the cases where the environment was maintained stationary, only the vehicle moving, or when the pilasters were moving periodically, we observed, in a great part of the experiments, the search for an state of equilibrium.
With CS, the mentioned equilibrium manifests itself in very simple periodic movements. With NNCS the equilibrium is rather sophisticated, because, besides utilizing real values for the actuators outputs, the neural networks may be producing non-linear soft mappings.
So, orbits of many types were observed: with variable eccentricity; helicoids; periodic or apparently periodic, with long periods.

7 NNCS – Next steps

In what follows, we enumerate important points to better understand the results of utilizing NNCS:

- To verify a manner of not suddenly altering vehicles velocity and orientation.
- As seeing in step 1, it can occur negative matching. It's worth to study a more adequate manner to make the bid in this case and to deduct it from the strength. One possible solution for the previous point is to consider not occurring matching if the evaluation is negative.
- To implement variations based on the *Bucket Brigade* algorithm.
- To study how to explicity apply apply *Evolutionary Reinforcement Learning* in NNCS.
- To develop specific genetic operators, as the chromosomes are weights of neural networks, and so providing better results for the GA actuation.
- Better genetic operators may facilitate the criteria definition for the frequency of GA activation, based on the necessity of new

knowledge as well as to improve the quality of the knowledge basis just acquired.

- To verify what are in common with # (don't care) on the CS with the ever occurrence of matching on the NNCS.
- Is there any kind of *building blocks* in the NNCS?
- To consider the vehicle movement dynamics.
- To vary acceleration instead of velocity, with negative acceleration being considered as the brakes of the vehicle.

8 Bibliography

[Ackley & Littman 1991] Interactions Between Learning and Evolution, David Ackley and Michael Littman. Artificial Life II, SFI Studies in the Sciences of Complexity, vol. X, edited by C. G. Langton, C. Taylor, J. D. Farmer, & S. Rasmussen, Addison-Wesley, 1991.

[Belew et al. 1991] "Evolving Networks: Using the Genetic Algorithm with Connectionist Learning", Richard K. Belew, John McInerney, and Nicol N. Schraudolph. Artificial Life II, SFI Studies in the Sciences of Complexity, vol. X, edited by C. G. Langton, C. Taylor, J. D. Farmer, & S. Rasmussen, Addison-Wesley, 1991

[Booker et al. 1989] "Classifier Systems and Genetic Algorithms" - L. B. Booker, D. E. Goldberg and J. H. Holland - Artificial Intelligence 40 (1989) pp. 235-282.

[Goldberg 1989] "Introduction to Genetics-Based Machine Learning" - Chapter 6 of "Genetic Algorithms in Search, Optimization and Machine Learning" - David. E. Goldberg - Addison-Wesley 1989.

[Haykin 1999] Neural Networks – A Comprehensive Foundation. Symon Haykin, 2nd. edition, Prentice-Hall, 1999.

[Michalewicz 1996] "Genetic Algorithms + Data Structures = Evolution Programs", Zbigniew Michalewicz, 3rd, Revised and Extend Edition, Springer. 12.1 The Michigan approach.

[Poli 1996] "Introduction to Evolutionary Computation", Riccardo Poli - Web Page at the School of Computer Science - The University of Birmingham:
http://www.cs.bham.ac.uk/~rmp/slide_book/.

[Richards 1995] "Classifier Systems & Genetic Algorithms" Robert A. Richards - Chapter 3 of Richards, Robert A.; 1995; Zeroth-order Shape Optimization Utilizing a Learning Classifier System Ph.D. Dissertation, Mechanical Engineering Department, Stanford University.
http://www.stanford.edu/~buc/SPHINcsX/book.html

[Simon 1996] The Sciences of the Artificial. H. A. Simon, MIT Press, 1996.