128: Nature and Scope of AI Techniques

Ajith Abraham
Oklahoma State University, Stillwater, OK, USA

1 INTRODUCTION TO COMPUTATIONAL INTELLIGENCE

Machine intelligence dates back to 1936, when Alan Turing proposed the idea of a universal mathematics machine, a theoretical concept in the mathematical theory of computability. Turing and Emil Post independently proved that determining the decidability of mathematical propositions is equivalent to asking what sort of sequences of a finite number of symbols can be recognized by an abstract machine with a finite set of instructions.

Such a mechanism is now known as a Turing machine (Turing Machine, 2004). Turing's research paper addressed the question of machine intelligence, assessing the arguments against the possibility of creating an intelligent computing machine and suggesting answers to those arguments; it proposed the Turing test as an empirical test of intelligence (Turing, 1950).

The Turing test, called the *imitation game* by Alan Turing, measures the performance of a machine against that of a human being. The machine and a human (A) are placed in two rooms. A third person, designated the interrogator, is in a room apart from both the machine and the human. The interrogator cannot see or speak directly to either A or the machine, communicating with them solely through some text messages or even a chat window. The task of the interrogator is to distinguish between the human and the computer on the basis of questions he or she may put to both of them over the terminals. If the interrogator cannot distinguish the machine from the human, then, Turing argues, the machine may be assumed to be intelligent. In the 1960s, computers failed to pass the Turing test because of the low processing speed of the computers.

The last few decades have seen a new era of artificial intelligence (AI) focusing on the principles, theoretical aspects, and design methodology of algorithms gleaned from nature. Examples are artificial neural networks inspired by mammalian neural systems, evolutionary computation inspired by natural selection in biology, simulated annealing inspired by thermodynamics principles, and swarm intelligence inspired by the collective behavior of insects or microorganisms, and so on, interacting locally with their environment, therein causing coherent functional global patterns to emerge. These techniques have found their way into solving real-world problems in science, business, technology, commerce, and also to a great extent in measuring systems.

Computational intelligence is a well-established paradigm, where new theories with a sound biological understanding have been evolving. The current experimental systems have many of the characteristics of biological computers (brains, in other words) and are beginning to be built

© 2005 John Wiley & Sons, Ltd. ISBN: 0-470-02143-8.

to perform a variety of tasks that are difficult or impossible to do with conventional computers.

To name a few, we have microwave ovens, washing machines, and digital cameras that can figure out on their own what settings to use to perform their tasks optimally; they have a reasoning capability, make intelligent decisions, and learn from experience.

As usual, defining computational intelligence is not an easy task. In a nutshell, which becomes quite apparent in light of the current research pursuits, the area is heterogeneous with a combination of such technologies as neural networks, fuzzy systems, evolutionary computation, swarm intelligence, and probabilistic reasoning.

The recent trend is to integrate different components to take advantage of complementary features and to develop a synergistic system. Hybrid architectures like neuro-fuzzy systems, evolutionary-fuzzy systems, evolutionary-neural networks, evolutionary-neuro-fuzzy systems, and so on, are widely applied for real-world problem solving. In the following sections, the main functional components of computational intelligence are introduced along with their key advantages and application domains.

2 ARTIFICIAL NEURAL NETWORKS

Artificial neural networks (ANN) have been developed as generalizations of mathematical models of biological nervous systems.

In a simplified mathematical model of the neuron, the effects of the synapses are represented by connection weights that modulate the effect of the associated input signals, and the nonlinear characteristic exhibited by neurons is represented by a transfer function, which is usually the sigmoid, Gaussian, trigonometric function, and so on.

The neuron impulse is then computed as the weighted sum of the input signals, being transformed by the transfer function.

The learning capability of an artificial neuron is achieved by adjusting the weights in accordance to the chosen learning algorithm. Most applications of neural networks fall into the following categories:

- Prediction: Use input values to predict some output
- Classification: Use input values to determine the classification
- Data Association: Like classification, but it also recognizes data that contains errors
- Data conceptualization: Analyze the inputs so that grouping relationships can be inferred.

A typical multilayered neural network and an artificial neuron are illustrated in Figure 1. Each neuron is characterized by an activity level (representing the state of polarization of a neuron), an output value (representing the firing rate of the neuron), a set of input connections (representing synapses on the cell and its dendrite), a bias value (representing an internal resting level of the neuron), and a set of output connections (representing a neuron's axonal projections). Each of these aspects of the unit is represented mathematically by real numbers. Thus, each connection has an associated weight (synaptic strength), which determines the effect of the incoming input on the activation level of the unit. The weights may be positive or negative. Referring to Figure 1, the signal flow from inputs $x_1, \ldots, x_n$ is considered to be unidirectional, indicated by arrows, as is a neuron's output signal flow (O). The neuron output signal O is given by the following relationship:

$$O = f(net) = f\left(\sum_{j=1}^{n} w_j x_j\right) \quad (1)$$

where w_j is the weight vector and the function *f(net) i*s referred to as an *activation (transfer) function*. The variable net is defined as a scalar product of the weight and input vectors

$$net = w^{\mathrm{T}} x = w_1 x_1 + \cdots + w_n x_n \quad (2)$$

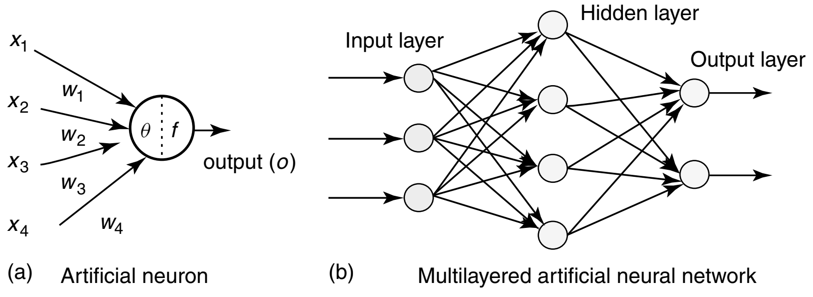

Figure 1. Architecture of an artificial neuron and a multilayered neural network.

where T is the transpose of a matrix and in the simplest case the output value O is computed as

$$O = f(net) = \begin{cases} 1 & if\ w^{\mathrm{T}}x \geq \theta \\ 0 & otherwise \end{cases} \tag{3}$$

where θ is called the *threshold level*, and this type of node is called a *linear threshold unit.*

3 NEURAL NETWORK ARCHITECTURES

The behavior of the neural network depends largely on the interaction between the different neurons. The basic architecture consists of three types of neuron layers:

1. Input
2. Hidden
3. Output.

In feed-forward networks, the signal flow is from input to output units, strictly in a feed-forward direction. The data processing can extend over multiple (layers of) units, but no feedback connections are present, that is, connections extending from outputs of units to inputs of units in the same layer or previous layers.

Recurrent networks contain feedback connections. Contrary to feed-forward networks, the dynamical properties of the network are important. In some cases, the activation values of the units undergo a relaxation process such that the network will evolve into a stable state in which these activations do not change anymore. In other applications, the changes of the activation values of the output neurons are significant, such that the dynamical behavior constitutes the output of the network.

There are several other neural network architectures (Elman network, adaptive resonance theory maps, competitive networks, etc.) depending on the properties and requirement of the application. The reader may refer to Bishop (1995) for an extensive overview of the different neural network architectures and learning algorithms.

A neural network has to be configured such that the application of a set of inputs produces the desired set of outputs. Various methods to set the strengths of the connections exist. One way is to set the weights explicitly, using a priori knowledge.

Another way is to train the neural network by feeding it, teaching patterns and letting it change its weights according to some learning rule.

The learning situations in neural networks may be classified into three distinct sorts of learning:

1. Supervised
2. Unsupervised
3. Reinforcement.

In supervised learning, an input vector is presented at the inputs together with a set of desired responses, one for each node, at the output layer. A forward pass is done and the errors or discrepancies, between the desired and actual response for each node in the output layer, are found. These are then used to determine weight changes in the net according to the prevailing learning rule.

The term ‘supervised’ originates from the fact that the desired signals on individual output nodes are provided by an external teacher. The best-known examples of this technique occur in the backpropagation algorithm, the delta rule, and perceptron rule.

In unsupervised learning (or self-organization), a (output) unit is trained to respond to clusters of pattern within the input. In this paradigm, the system is supposed to discover statistically salient features of the input population (Kohonen, 1988). Unlike the supervised learning paradigm, there is no a priori set of categories into which the patterns are to be classified; rather, the system must develop its own representation of the input stimuli.

Reinforcement learning is learning what to do – how to map situations to actions – so as to maximize a numerical reward signal. The learner is not told which actions to take, as in most forms of machine learning, but instead must discover which actions yield the most reward by trying them. In the most interesting and challenging cases, actions may affect not only the immediate reward but also the next situation and, through that, all subsequent rewards. These two characteristics, trial-and-error search and delayed reward, are the two most important distinguishing features of reinforcement learning.

4 FUZZY LOGIC

Zadeh (1965) introduced the concept of fuzzy logic to represent vagueness in linguistics and to further implement and express human knowledge and inference capability in a natural way. Fuzzy logic starts with the concept of a fuzzy set.

A fuzzy set is a set without a crisp, clearly defined boundary. It can contain elements with only a partial degree of membership.

A Membership Function (MF) is a curve that defines how each point in the input space is mapped to a membership value (or degree of membership) between 0 and 1. The input space is sometimes referred to as the *universe of discourse*.

Let X be the universe of discourse and x be a generic element of X. A classical set A is defined as a collection of elements or objects $x \in X$, such that each x can either belong to or not belong to the set A.

By defining a characteristic function (or membership function) on each element x in X, a classical set A can be represented by a set of ordered pairs $(x, 0)$ or $(x, 1)$, where 1 indicates membership and 0, nonmembership.

Unlike the conventional set mentioned above, the fuzzy set expresses the degree to which an element belongs to a set. Hence, the characteristic function of a fuzzy set is allowed to have a value between 0 and 1, denoting the degree of membership of an element in a given set. If X is a collection of objects denoted generically by x, then a fuzzy set A in X is defined as a set of ordered pairs:

$$A = \{(x, \mu_A(x)) | x \in X\} \tag{4}$$

$\mu_A(x)$ is called the *membership function* of linguistic variable x in A, which maps X to the membership space M, $M = [0,1]$, where M contains only two points 0 and 1, A is crisp and μ_A is identical to the characteristic function of a crisp set.

Triangular and trapezoidal membership functions are the simplest membership functions, formed using straight lines. Some of the other shapes are Gaussian, generalized bell, sigmoidal, and polynomial-based curves. Figure 2 illustrates the shapes of two commonly used MFs. The most important thing to realize about fuzzy logical reasoning is the fact that it is a superset of standard Boolean logic.

4.1 Fuzzy logic operators

It is interesting to note the correspondence between two-valued and multivalued logic operations for the **AND, OR**, and **NOT** logical operators.

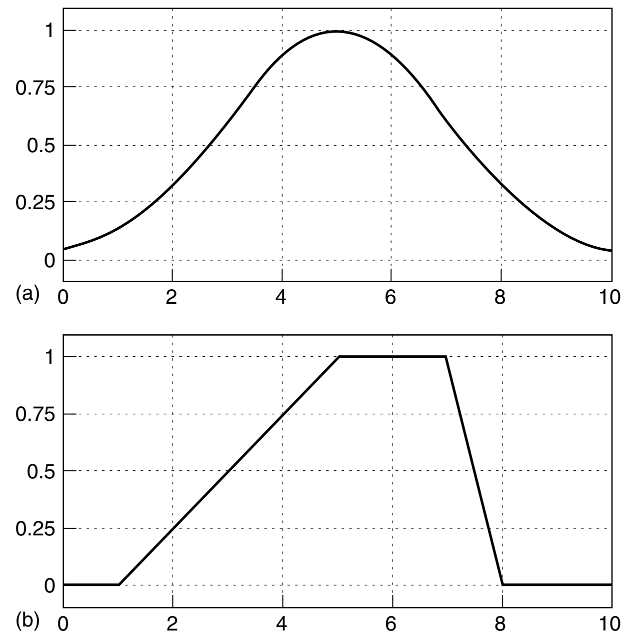

Figure 2. Membership functions; (a) Gaussian and (b) trapezoidal.

It is possible to resolve the statement A **AND** B, where A and B are limited to the range (0, 1) by using the operator minimum (A, B). Using the same reasoning, we can replace the **OR** operation with the maximum operator, so that A **OR** B becomes equivalent to maximum (A, B). Finally, the operation **NOT** A becomes equivalent to the operation 1 – A.

In fuzzy logic terms, these are popularly known as fuzzy intersection or conjunction (AND), fuzzy union or disjunction (OR), and fuzzy complement (NOT). The intersection of two fuzzy sets A and B is specified, in general, by a binary mapping T, which aggregates two membership functions as follows.

$$\mu_{A\cap B}(x) = T(\mu_A(x), \mu_B(x)) \tag{5}$$

The fuzzy intersection operator is usually referred to as *T-norm (Triangular norm) operator*. The fuzzy union operator is specified in general by a binary mapping S.

$$\mu_{A\cup B}(x) = S(\mu_A(x), \mu_B(x)) \tag{6}$$

This class of fuzzy union operators are often referred to as *T-conorm (S-norm) operators*.

5 IF-THEN RULES AND FUZZY INFERENCE SYSTEMS

The fuzzy rule base is characterized in the form of *if-then* rules in which preconditions and consequents involve linguistic variables. The collection of these fuzzy rules forms the rule base for the fuzzy logic system. Owing to their concise form, fuzzy *if-then* rules are often employed to capture the imprecise modes of reasoning that play an essential role in the human ability to make decisions in an environment of uncertainty and imprecision. A single fuzzy *if-then* rule assumes the form

$$if\ x \text{ is } A\ then\ y \text{ is } B$$

where A and B are linguistic values defined by fuzzy sets in the ranges (universes of discourse) X and Y, respectively.

The *if* part of the rule 'x is A' is called the *antecedent (precondition)* or premise, while the *then* part of the rule 'y is B' is called the *consequent* or conclusion. Interpreting an *if-then* rule involves evaluating the antecedent (*fuzzification of* the input and applying any necessary *fuzzy operators*) and then applying that result to the consequent (known as *implication*). For rules with multiple antecedents, all parts of the antecedent are calculated simultaneously and resolved to a single value using the logical operators. Similarly, all the consequents (rules with multiple consequents)

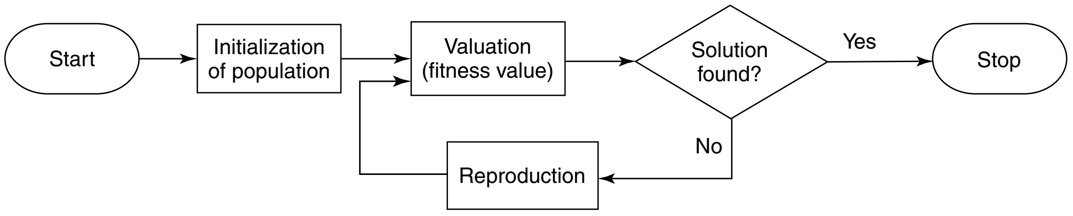

Figure 3. Flowchart of genetic algorithm iteration.

are affected equally by the result of the antecedent. The consequent specifies a fuzzy set be assigned to the output. The implication function then modifies that fuzzy set to the degree specified by the antecedent. For multiple rules, the output of each rule is a fuzzy set. The output fuzzy sets for each rule are then aggregated into a single output fuzzy set. Finally, the resulting set is defuzzified, or resolved, to a single number.

The defuzzification interface is a mapping from a space of fuzzy actions defined over an output universe of discourse into a space of nonfuzzy actions, because the output from the inference engine is usually a fuzzy set, while for most practical applications, crisp values are often required.

The three commonly applied defuzzification techniques are max-criterion, center-of-gravity, and the mean-of-maxima. The max-criterion is the simplest of these three to implement. It produces the point at which the possibility distribution of the action reaches a maximum value.

The reader can refer to Nguyen and Walker (1999) for more information related to fuzzy systems. It is typically advantageous if the fuzzy rule base is adaptive to a certain application. The fuzzy rule base is usually constructed manually or by automatic adaptation by some learning techniques using evolutionary algorithms and/or neural network learning methods (Abraham, 2001).

6 EVOLUTIONARY ALGORITHMS

Evolutionary algorithms (EA) are adaptive methods, which may be used to solve search and optimization problems, based on the genetic processes of biological organisms. Over many generations, natural populations evolve according to the principles of natural selection and 'survival of the fittest', first clearly stated by Charles Darwin in his work '*The Origin of Species*'. By mimicking this process, evolutionary algorithms are able to 'evolve' solutions to real-world problems, provided they have been suitably encoded (Fogel, 1999).

Usually grouped under the term evolutionary algorithms or evolutionary computation, we find the domains of genetic algorithms (GA) (Holland, 1975), (Goldberg, 1989); evolution strategies (Rechenberg, 1973), (Schwefel, 1977); evolutionary programming (Fogel, Owens and Walsh, 1967); genetic programming (Koza, 1992); and learning classifier systems. They all share a common conceptual base of simulating the evolution of *individual* structures via processes of *selection, mutation*, and *reproduction*. The processes depend on the perceived performance of the individual structures as defined by the environment (problem).

EA's deal with parameters of finite length, which are coded using a finite alphabet, rather than directly manipulating the parameters themselves. This means that the search is unconstrained neither by the continuity of the function under investigation nor by the existence of a derivative function. Figure 3 depicts the functional block diagram of a genetic algorithm, and the various aspects are discussed below. It is assumed that a potential solution to a problem may be represented as a set of parameters. These parameters (known as *genes*) are joined together to form a string of values (known as a *chromosome*). A gene (also referred to as a *feature, character*, or *detector*) refers to a specific attribute that is encoded in the chromosome. The particular values that the genes can take are called its *alleles.*

Encoding issues deal with representing a solution in a chromosome and, unfortunately, no one technique works best for all problems. A fitness function must be devised for each problem to be solved.

Given a particular chromosome, the fitness function returns a single numerical fitness or figure of merit, which will determine the ability of the individual that the chromosome represents.

Reproduction is the second critical attribute of GAs, where two individuals selected from the population are allowed to mate to produce an offspring, which will comprise the next generation. Having selected the parents, the offsprings are generated, typically using the mechanisms of crossover and mutation.

Selection is the survival of the fittest within GAs. It determines which individuals are to survive to the next generation. The selection phase consists of three parts.

The first part involves determination of the individual's fitness by the fitness function. A fitness function must be devised for each problem; given a particular chromosome, the fitness function returns a single numerical fitness value, which is proportional to the ability, or utility, of the individual represented by that chromosome.

The second part involves converting the fitness function into an expected value, followed by the last part where the expected value is then converted to a discrete number of offspring.

Some of the commonly used selection techniques are roulette wheel and stochastic universal sampling. If the GA has been correctly implemented, the population will evolve over successive generations so that the fitness of the best and the average individual in each generation increases toward the global optimum.

7 INTELLIGENT PARADIGMS

7.1 Probabilistic computing

Probabilistic models are viewed as being similar to that of a game; actions are based on expected outcomes. The center of interest moves from the deterministic to probabilistic models using statistical estimations and predictions.

In the probabilistic modeling process, risk means uncertainty for which the probability distribution is known. Therefore, risk assessment means a study to determine the outcomes of decisions along with their probabilities. Decision makers often face a severe lack of definite information. Probability assessment quantifies the information gap between what is known and what needs to be known for an optimal decision. Probabilistic models are used for protection against adverse uncertainty and exploitation of propitious uncertainty (Pearl, 1997).

A good example is the probabilistic neural network (Bayesian learning) in which probability is used to represent uncertainty about the relationship being learned. Before we have seen any data, our *prior* opinions about what the true relationship might be can be expressed in a probability distribution over the network weights that define this relationship.

After we look at the data, our revised opinions are captured by a *posterior* distribution over network weights. Network weights that seemed plausible before, but which do not match the data very well, will now be seen as being much less likely, while the probability for values of the weights that do fit the data well will have increased. Typically, the purpose of training is to make predictions for future cases in which only the inputs to the network are known. The result of conventional network training is a single set of weights that can be used to make such predictions.

7.2 Swarm intelligence

Swarm intelligence is aimed at collective behavior of intelligent agents in decentralized systems. Although there is typically no centralized control dictating the behavior of the agents, local interactions among the agents often cause a global pattern to emerge (Eberhart, Shi and Kennedy, 2001).

Most of the basic ideas are derived from the real swarms in nature, which includes ant colonies, bird flocking, honeybees, bacteria and microorganisms, and so on. Ant Colony Optimization (ACO) has already been applied successfully to solve several engineering optimization problems.

Swarm models are population based and the population is initialized with a population of potential solutions. These individuals are then manipulated (optimized) over several iterations using several heuristics inspired from the social behavior of insects in an effort to find the optimal solution.

Ant colony algorithms are inspired by the behavior of natural ant colonies, in the sense that they solve their problems by multiagent cooperation using indirect communication through modifications in the environment. Ants release a certain amount of pheromone (hormone) while walking, and each ant prefers (probabilistically) to follow a direction that is rich in pheromone. This simple behavior explains why ants are able to adjust to changes in the environment, such as optimizing the shortest path to a food source or a nest. In ACO, ants use information collected during past simulations to direct their search and this information is available and modified through the environment. Recently, ACO algorithms have also been used for clustering data sets.

8 HYBRID INTELLIGENT SYSTEMS

Several adaptive hybrid intelligent systems have, in recent years, been developed for model expertise, image, and video segmentation techniques, process control, mechatronics, robotics, complicated automation tasks and so on.

Many of these approaches use the combination of different knowledge representation schemes, decision-making models, and learning strategies to solve a computational task. This integration aims at overcoming limitations of individual techniques through hybridization or fusion of various techniques.

These ideas have led to the emergence of several different kinds of intelligent system architectures. Most of the current Hybrid Intelligent Systems (HIS) consist of three essential paradigms: artificial neural networks, fuzzy inference systems, and global optimization algorithms (example, evolutionary algorithms). Nevertheless, HIS is an

Table 1. Hybrid intelligent system basic ingredients.

Methodology	Advantage
Artificial neural networks	Adaptation, learning, and approximation
Fuzzy logic	Approximate reasoning
Global optimization algorithms	Derivative free optimization

open, instead of conservative, concept, that is, it is evolving those relevant techniques together with the important advances in other new computing methods. Table 1 lists the three principal ingredients together with their advantages (Abraham, 2002).

Experience has shown that it is crucial for the design of HIS to primarily focus on the integration and interaction of different techniques rather than merge different methods to create ever-new techniques. Techniques already well understood should be applied to solve specific domain problems within the system. Their weakness must be addressed by combining them with complementary methods.

Neural networks offer a highly structured architecture with learning and generalization capabilities. The generalization ability for new inputs is then based on the inherent algebraic structure of the neural network. However, it is very hard to incorporate human a priori knowledge into a neural network. This is mainly due to the fact that the connectionist paradigm gains most of its strength from a distributed knowledge representation.

In contrast, fuzzy inference systems exhibit complementary characteristics, offering a very powerful framework for approximate reasoning as it attempts to model the human reasoning process at a cognitive level. Fuzzy systems acquire knowledge from domain experts and this is encoded within the algorithm in terms of the set of *if-then* rules. Fuzzy systems employ this rule-based approach and interpolative reasoning to respond to new inputs. The incorporation and interpretation of knowledge is straightforward, whereas learning and adaptation constitute major problems.

Global optimization is the task of finding the absolutely best set of parameters to optimize an objective function. In general, it may be possible to have solutions that are locally optimal but not globally optimal. Evolutionary computing (EC) works by simulating evolution on a computer. Such techniques could be easily used to optimize neural networks, fuzzy inference systems, and other problems.

Owing to the complementary features and strengths of different systems, the trend in the design of hybrid system is to merge different techniques into a more powerful integrated system to overcome their individual weaknesses.

9 MODELS OF HYBRID SYSTEMS

The various HIS architectures could be broadly classified into four different categories based on the systems overall architecture:

1. Stand alone
2. Transformational
3. Hierarchical hybrid
4. Integrated hybrid.

9.1 Stand-alone architecture

Stand-alone models of HIS applications consist of independent software components, which do not interact in any way. Developing stand-alone systems can have several purposes. First, they provide direct means of comparing the problem-solving capabilities of different techniques with reference to a certain application. Running different techniques in a parallel environment permits a loose approximation of integration. Stand-alone models are often used to develop a quick initial prototype while a more time-consuming application is developed. Some of the benefits are simplicity and ease of development using commercially available software packages.

9.2 Transformational hybrid architecture

In a transformational hybrid model, the system begins as one type and ends up as the other. Determining which technique is used for development and which is used for delivery is based on the desirable features that the technique offers. Expert systems and neural networks have proven to be useful transformational models. Variously, either the expert system is incapable of adequately solving the problem, or it requires the speed, adaptability, and robustness of neural network. Knowledge from the expert system is used to set the initial conditions and training set for neural network. Transformational hybrid models are often quick to develop and ultimately require maintenance on only one system. Most of the developed models are just application oriented.

9.3 Hierarchical hybrid architectures

The architecture is built in a hierarchical fashion, associating a different functionality with each layer. The overall functioning of the model will depend on the correct functioning of all the layers. A possible error in one of the layers will directly affect the desired output.

9.4 Integrated hybrid architectures

These models include systems, which combine different techniques into one single computational model. They share data structures and knowledge representations. Another approach is to put the various techniques on a side-by-side basis and focus on their interaction in the problem-solving task. This method might allow integrating alternative techniques and exploiting their mutuality. The benefits of fused architecture include robustness, improved performance, and increased problem-solving capabilities. Finally, fully integrated models can provide a full range of capabilities such as adaptation, generalization, noise tolerance, and justification. Fused systems have limitations caused by the increased complexity of the intermodule interactions, and specifying, designing, and building fully integrated models is complex.

10 SUMMARY

Artificial intelligence is the study of intelligent behavior. Its ultimate goal is a theory of intelligence that accounts for the behavior of naturally occurring intelligent entities, and this guides the creation of artificial entities capable of intelligent behavior. The stagnation of artificial intelligence during the 1970s and 1980s does not have much bearing on the likelihood of artificial intelligence to succeed in the future, since we know that the cause responsible for stagnation (mainly due to insufficient hardware resources) is no longer present. More detail of the various methods introduced here is found in **Article 129, Artificial Neural Networks, Volume 2**; **Article 130, Rule-based Expert Systems, Volume 2**; and **Article 131, Evolutionary Computation, Volume 2**.

REFERENCES

Abraham, A. (2001) in *Neuro-Fuzzy Systems: State-of-the-art Modeling Techniques, Connectionist Models of Neurons, Learning Processes, and Artificial Intelligence, Lecture Notes in Computer Science*, Vol. 2084, (eds J. Mira and A. Prieto), Springer Verlag, Germany, (pp. 269–276).

Abraham, A. (2002) Intelligent Systems: Architectures and Perspectives, Recent Advances in Intelligent Paradigms and Applications, in *Studies in Fuzziness and Soft Computing*, Chapter 1, (eds A. Abraham, L. Jain and J. Kacprzyk), Springer Verlag Germany, (pp. 1–35).

Bishop, C.M. (1995) *Neural Networks for Pattern Recognition*, Oxford University Press, UK.

Eberhart, R., Shi, Y. and Kennedy, J. (2001) *Swarm Intelligence*, Morgan Kaufmann, San Francisco, CA.

Fogel, D.B. (1999) *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, 2nd edn, IEEE Press, Piscataway, NJ.

Fogel, L.J., Owens, A.J. and Walsh, M.J. (1967) *Artificial Intelligence Through Simulated Evolution*, John Wiley & Sons, New York.

Goldberg, D.E. (1989) *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing Corporation, Inc, Reading, MA.

Holland, J. (1975) *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Harbor, MI.

Kohonen, T. (1988) *Self-organization and Associative Memory*, Springer-Verlag, New York.

Koza, J.R. (1992) *Genetic Programming*, MIT Press, Cambridge, MA.

Nguyen, H.T. and Walker, E.A. (1999) *A First Course in Fuzzy Logic*, CRC Press, USA.

Pearl, J. (1997) *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann Publishers, San Francisco, CA.

Rechenberg, I. (1973) *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*, Fromman-Holzboog, Stuttgart.

Schwefel, H.P. (1977) *Numerische Optimierung von Computermodellen Mittels der Evolutionsstrategie*, Birkhaeuser, Basel.

Turing, A.M. (1950) Computing Machinery and Intelligence http://abelard.org/turpap/turpap.htm.

Turing Machine. (2004) http://www.turing.org.uk/turing/.

Zadeh, L.A. (1965) Fuzzy Sets. *Journal of Information and Control*, **8**, 338–353.