Chapter 1
Computational Intelligence Techniques and Applications

Xuan Zhu

Abstract Computational intelligence is a group of computational models and tools that encompass elements of learning, adaptation, and/or heuristic optimization. It is used to help study problems that are difficult to solve using conventional computational algorithms. Neural networks, evolutionary computation, and fuzzy systems are the three main pillars of computational intelligence. More recently, emerging areas such as swarm intelligence, artificial immune systems (AIS), support vector machines, rough sets, chaotic systems, and others have been added to the range of computational intelligence techniques. This chapter aims to present an overview of computational intelligence techniques and their applications, focusing on five representative techniques, including neural networks, evolutionary computation, fuzzy systems, swarm intelligence, and AIS.

Keywords Computational intelligence • Neural networks • Evolutionary computation • Fuzzy systems • Swarm intelligence • Artificial immune systems

1.1 Introduction

The earth and environmental systems are dynamic. They are shaped and changed continuously by complex and interrelated physical, biological, and chemical processes. For example, the physical process of weathering breaks down the rocks and soils into their constituent substances. When it rains, the water droplets absorb and dissolve carbon dioxide from the air, which causes the rainwater to be slightly acidic. The released sediment and chemicals then take part in chemical

X. Zhu (✉)
School of Geography and Environmental Science, Monash University, Building 11, Clayton Campus, Clayton, VIC 3800, Australia
e-mail: xuan.zhu@monash.edu

T. Islam et al. (eds.), *Computational Intelligence Techniques in Earth and Environmental Sciences*, DOI 10.1007/978-94-017-8642-3_1,

reactions that erode the earth's surface. The sediment may be transformed by geological forces into other rocks and soils. Living organisms also play a dynamic role through respiration, excretion, death, and decay. They recycle their constituent elements through the environment. Therefore, earth and environmental systems encompass numerous biological, physical, and chemical processes, which interact with each other and which are difficult to model and analyze. In addition, many earth and environmental systems present complex spatial and temporal patterns and behaviors. The interactions between these systems are often ill-defined and their relationships are generally nonlinear. Many earth and environmental problems have no strong theoretical understanding and, therefore, there are no full numerical models. The complexity of the earth and environmental systems has led to the need for effective and efficient computational tools to analyze and model highly nonlinear functions and can be trained to accurately generalize when presented with new, unseen data. The emerging computational intelligence techniques have some or all of these features. They provide an attractive alternative to developing numerical models to conventional statistical approaches, from which the new insights and underlying principles of earth and environmental sciences can be derived.

Computational intelligence is a group of computational models and tools devoted to solution of problems for which there are no effective computational algorithms (Konar 2005; Madani 2011). It is an offshoot of artificial intelligence, which focuses on heuristic algorithms such as neural networks, evolutionary computation, and fuzzy systems. Computational intelligence also involves adaptive mechanisms and learning ability that facilitate intelligent behavior in complex and changing environment. In addition to the three main pillars (neural networks, evolutionary computation, and fuzzy systems), computational intelligence includes swam intelligence, artificial immune systems (AIS), support vector machines, rough sets, chaotic systems, probabilistic methods, and other techniques or a combination of these techniques (Engelbrecht 2007). They have been successfully applied in a wide range of applications in the earth and environmental sciences, for example, species distribution modeling (Watts et al. 2011), site quality assessment of forests (Aertsen et al. 2010), prediction of water quality (Areerachakul et al. 2013), and forecasting of air pollution (Antanasijevia et al. 2013). This chapter aims to present an overview of computational intelligence techniques and their applications, focusing on the following five representative techniques, including neural networks, evolutionary computation, swam intelligence, AIS, and fuzzy systems.

1.2 Neural Networks

1.2.1 Basic Principles

Neural networks here refer to artificial neural networks (ANNs). They are developed to emulate biological neural systems. The basic building blocks of biological neural systems are neurons. A neuron consists of a cell body, an axon, and dendrites.

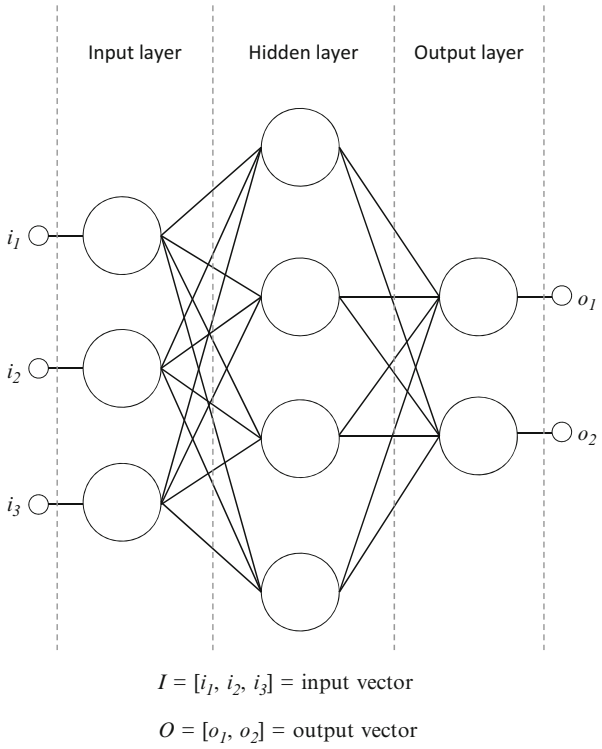

Fig. 1.1 An artificial neural network (ANN) with one hidden layer

A neuron is connected through its axon with a dendrite of another neuron. The connection point between neurons is called a synapse. Basically, a neuron receives signals from the environment. When the signals are transmitted to the axon of the neuron, i.e., the neuron is fired, the cell sums up all the inputs, which may vary by the strength of the connection or the frequency of the input signals, processes the input sum, and then produces an output signal, which is propagated to all connected neurons. An artificial neuron is a simplified computational model of a biological neuron, performing the above basic functions.

In an artificial neuron, the cell body is modeled by an activation or transfer function. The artificial neuron receives one or more inputs (representing signals received through dendrites that are excited or inhibited by positive or negative numerical weights associated with each dendrite), calculates the weighted sum of the inputs, and passes the sum to the activation function. The activation function is a nonlinear function, and its output represents an axon, which propagates as an input to another neuron through a synapse. An activation function can be sigmoid, hyperbolic tangent, or linear (Konar 2005).

An ANN is a layered network of artificial neurons. It typically consists of an input layer, one or more hidden layers, and an output layer (Fig. 1.1). Each layer is composed of a number of artificial neurons, also called nodes. The artificial neurons

in one layer are connected by weights to the artificial neurons in the next layer. It is essentially a model representing a nonlinear mapping between an input vector and output vector. As the output from an artificial neuron or node is a function of the sum of the inputs to the node modified by a nonlinear activation function (e.g., a logistic function), an ANN superposes many simple nonlinear activation functions used by the nodes constituting the network, which enables it to approximate highly nonlinear functions, thus introducing complex nonlinear behavior to the network (Principe et al. 2000). These functions can be trained to accurately generalize with new data. The adaptive property is embedded within the network by adjusting the weights that interconnect the nodes during the training phase. After the training phase the ANN parameters are fixed and the system is deployed to solve the problem at hand. Therefore, an ANN is an adaptive, nonlinear system that learns to perform a function from data.

There are several types of ANN, including multilayer feed-forward, recurrent, temporal, probabilistic, fuzzy, and radial basis function ANNs (Haykin 1999). The most popular one is the multilayer feed-forward neural network as shown in Fig. 1.1. It is also referred to as the multilayer perceptron network. In this type of ANN, the output of a node is scaled by the connection weights and fed forward as an input to the nodes in the next layer. That is, information flow starts from the nodes in the input layer, and then moves along weighted links to the nodes in the hidden layers for processing. The input layer plays no computational role, but provides the inputs to the network. The connection weights are normally determined through training. Each node contains an activation function that combines information from all the nodes in the preceding layer. The output layer is a complex function of the outcomes resulted from internal network transformations.

Multilayer perceptron networks are able to learn through training. Training involves the use of a set of training data with a systematic step-by-step procedure to optimize a performance criterion or to follow some implicit internal constraint, which is commonly referred to as the learning rule. Training data must be representative of the entire data set. A neural network starts with a set of initial connection weights. During training the network is repeatedly fed with the training data (a set of input–output pattern pairs obtained through sampling) and the connection weights in the network are modified until the learning rule is satisfied. One performance criterion could be a threshold value of an error signal, which is defined as the difference between the desired and actual output for a given input vector. Training uses the magnitude of the error signal to determine how much the connection weights need to be adjusted so that the overall error is reduced. The training process is driven by a learning algorithm, such as back-propagation (Rumelhart et al. 1986) and scaled conjugate gradient algorithms (Hagan et al. 1996). Once trained with representative training data, the multilayer perceptron network gains sufficient generalization ability and can be applied to new data.

1.2.2 Applications

ANNs are applicable when a relationship between the independent variables (outputs) and dependent variables (inputs) exists. They are able to learn the relationship from a given data set without any assumptions about the statistical distribution of the data. In addition, ANNs perform a nonlinear transformation of input data to approximate output data, learning from training data and exhibiting the ability for generalization beyond training data. This makes them more practical and accurate in modeling complex data patterns than many traditional methods that are linear. They are also able to deal with outlying, missing, and noisy data due to their ability to generalize well on unseen data. The capability of learning from data, modeling nonlinear relationships, and handling noises in data makes ANN particularly suitable for pattern classification, function approximation, and prediction in most earth and environmental applications. Pattern classification is to classify data into the predetermined discrete classes. Functional approximation is to formulate a function from a given set of training data to model the relationship between variables. Prediction involves the estimation of output from previous samples or forecasting of future trends in a time series of data given previous and current conditions.

ANNs have been employed to predict and assess air quality and forecast severe weather. For example, Gardner and Dorling (1999) applied multilayer perceptron networks to predict hourly NO_x and NO_2 concentrations in urban air in London. The neural networks used in this study have two hidden layers, each containing 20 nodes. They used five variables (low cloud amount, visibility, dry bulb temperature, vapor pressure, and wind speed) to predict the pollutant concentration. The activation function used in the models was the hyperbolic tangent function (Engelbrecht 2007). The networks were trained using the scaled conjugate gradient algorithm based on hourly meteorological data on the five variables. Their results suggest that ANNs outperform regression models, and can be used to resolve complex patterns of source emissions without any explicit external guidance. Sousa et al. (2007) used feed-forward ANNs to predict hourly ozone concentrations with principal components as inputs, which effectively combined principal component analysis and ANN techniques. This combination reduces the collinearity of the data sets, determines the relevant independent variables for the prediction of ozone concentrations, and thus leads to a less complex architecture of the ANN due to the decrease of input variables. It also eliminates the overfitting problem, so that mean square errors of both validation and training continuously decline. Data on precursor concentrations and meteorological variables on ozone formation were used. The principal component-based ANN was compared with multiple linear regression, principal component regression, and feed-forward ANNs based on the original data. The results suggested that the use of principal components as inputs improved the ANN prediction. Krasnopolsky (2013) provided a representative set of ANN applications in meteorology, oceanography, numerical weather prediction, and climate studies.

In geology ANNs have been used for sediment prediction, permeability prediction, simulation of chemical concentrations in minerals, and geological hazard assessment. For example, Yang and Rosenbaum (2003) built a multilayer perceptron network, integrated with a GIS, to predict distributions of sediments (sand, silt, and clay) in Gothenburg harbor, Sweden. The developed network has four input nodes (water depth, distance from the river mouth, bank, and shipping lanes) and three output nodes (one for each sediment grade) as well as two hidden layers. The network was trained using the data from 139 sample stations. Fegh et al. (2013) applied ANN to determine rock permeability of a gas reservoir from well logs at well locations, and used the results from the ANN modeling as an input to several geostatistical models through the structural model to construct 3D geological reservoir permeability models. Permeability prediction using ANN used the data sets derived from four wells of the studied gas field. Two of the wells have core permeability data, one used for constructing the ANN model and the other for evaluating the reliability of the ANN model. The model was then applied to predict permeability at the other un-cored wells. Torkar et al. (2010) used ANN to model nonlinear dependency of radon concentrations in soil gas on five environmental variables (air and soil temperature, barometric pressure of air and soil, and rainfall). A four-layer perceptron network was developed in this study with five input nodes (the environmental variables) and one output node (radon concentration) and it was trained using the back-propagation algorithm. The ANN model correctly predicted 10 seismic events out of 13 within the 2-year period. Bui et al. (2012) investigated the potentials of ANN in landslide susceptibility mapping at the Hoa Binh province of Vietnam. They built two multilayer feed-forward ANNs with back-propagation training algorithms, Levenberg–Marquardt and Bayesian regularization. Ten landslide conditioning factors were used as input nodes: slope, aspect, relief amplitude, lithology, land use, soil type, rainfall, and distance to roads, rivers, and faults. A landslide inventory over 10 years derived from satellite images, field surveys, and existing literature was utilized as training data. The connection weights obtained in the training phase were applied to the entire study area to produce landslide susceptibility indexes. The prediction accuracy of landslide susceptibility mapping by the Bayesian regularization neural network and the Levenberg–Marquardt neural network was 90.3 % and 86.1 %, respectively. The study suggested that the ANNs have good predictive capability, but the Bayesian regularization network model appears more robust and efficient.

ANNs have also found to outperform traditional or classic modeling methods in ecological modeling (Lek and Guegan 1999). There have been numerous applications in ecological modeling in various fields of ecology since the early 1990s. For example, Ranković et al. (2010) built a feed-forward ANN model to predict the dissolved oxygen in the Gruža Reservoir, Serbia. The input variables of the neural network include water pH, water temperature, chloride, total phosphate, nitrites, nitrates, ammonia, iron, manganese, and electrical conductivity. The Levenberg–Marquardt algorithm was used to train the network. The results were compared with the measured data on the basis of correlation coefficient, mean absolute error, and mean square error, which indicated that the neural

network model provided accurate results. Other examples include using ANN to relate flow conditions to fish community diversity (Chang et al. 2013), determine factors that influence the dispersal of invasive species (Pontin et al. 2011), predict water quality indicators (Kuo et al. 2007; Huo et al. 2013), predict biodiversity (Yoo et al. 2012), estimate tree height (Özçelik et al. 2013), and simulate denitrification rates in wetlands (Song et al. 2013). Zhang (2010) offered an overall and in-depth knowledge on algorithms, programs, and case studies of ANNs in ecology.

Examples of ANN applications in hydrology include hydrological time series modeling (Lohani et al. 2012), groundwater salinity forecasting (Banerjee et al. 2011), rainfall–runoff modeling (Wu and Chau 2011), reservoir inflow prediction (Okkan 2012), and suspended sediment load prediction (Kakaei et al. 2013).

Pattern recognition and image classification are among the most common applications of ANN in remote sensing. The major advantage of ANN to image classification over conventional statistical approaches, such as maximum likelihood and Mahalanobis distance classifiers, is that ANN is essentially nonparametric and nonlinear, and has no assumptions regarding the underlying distribution of values of the explanatory variables and the training data. ANNs are found to be accurate in the classification of remotely sensed data (Mas 2004; Bao and Ren 2011; Dobreva and Klein 2011; Cruz-Ramírez et al. 2012).

1.3 Evolutionary Computation

1.3.1 Basic Principles

Evolutionary computation simulates natural evolution. It was born out of the idea of evolutionary programming introduced in the 1960s (Eiben and Smith 2003). Evolutionary computation includes evolutionary algorithms for solving search and optimization problems. It was thought of as a model for machine learning in which a population of randomly created individuals goes through a process of evolution mimicking the process of natural selection and natural genetics (Yu and Gen 2010). In every generation, a new set of artificial creatures is created using bits and pieces of the fittest of the old. An artificial creature is an individual representing a point in the problem's solution search space.

Evolutionary algorithms use random choice as a tool to guide a highly exploitive search toward regions of the search space with likely improvement. A single point in the solution search space is an individual, represented by a chromosome, which consists of genes. For example, a land use pattern is a single point in the search space of a land use optimization problem. A land use pattern can be seen as a chromosome. Genes are essentially the parameters of the problem to be solved. A gene can take many forms depending on the problem definition. For the land use allocation problem, a gene can be a land parcel with a given land use and land

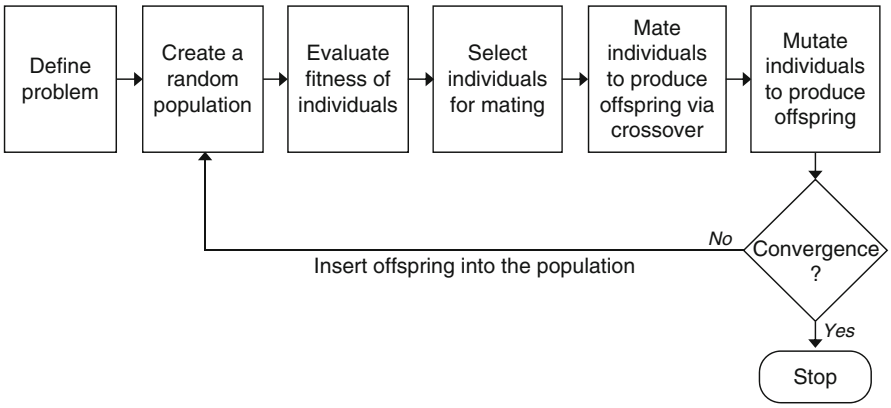

Fig. 1.2 General procedure of an evolutionary algorithm

attributes. The fitness (survival strength) of a chromosome is measured by a fitness or an objective function. A fitness function is some measure of profit, utility, or goodness to be maximized. A population is a collection of all the chromosomes being evolved. As new chromosomes are created and reinserted into the population, less fit chromosomes are replaced and only the fittest survive into the next generation. It is here that the process of evolution occurs, as the fitness of the competing chromosomes is compared in order to select parent chromosomes to reproduce.

Evolutionary algorithms generally follow the basic procedure as depicted in Fig. 1.2. The first step is to define the problem, which includes defining genes to encode the information needed for problem solving, specifying chromosomes to represent single solutions, and defining a fitness function. After the problem is defined, an initial population is randomly created as the first generation. All the chromosomes in the generation are then evaluated using the fitness function. After that, chromosomes are selected from the population according to their fitness function values to ensure that only the fittest chromosomes can survive into the next generation. The selected chromosomes are then combined in a process called crossover to create a set of children. The children are randomly mutated to create a new set of chromosomes to be reinserted into the population. Once enough children chromosomes have been created to replace a population, a generation is said to have passed. For the new generation, the evaluation, selection, crossover, mutation, and insertion process starts again. After a number of generations have elapsed, an optimal solution is converged and the process stops. The best chromosome is selected from the final population and represents the optimal solution to the problem being solved. Essentially what's happening is that a random set of solutions to a problem within a given search space is created and evolves over an amount of time to find an optimal solution.

There are different categories of evolutionary algorithms, including genetic algorithms, genetic programming, evolutionary programming, and evolution strategies. Genetic algorithms model genetic evolution (Goldberg 1989); genetic programming optimizes a population of computer programs based on genetic algorithms, where each individual is a computer program (Langdon and Poli 2002); evolutionary programming is similar to a genetic algorithm, but simulates only adaptive behavior in evolution (Yao et al. 1999); and evolution strategies are used to minimize functions of real variables, which are individuals, each having a "strategy variable" that determines the degree of mutation to be applied to the corresponding variable (Auger and Hansen 2011). They all share a basic principle of simulating the evolution of individuals through the process of evaluation, selection, crossover, mutation, and reproduction.

1.3.2 Applications

Evolutionary computation has been used successfully for optimization, scheduling, and time series approximation in earth and environmental applications.

Downing (1998) illustrated three applications of evolutionary computation in environmental modeling: optimal foraging strategies, temporal ideal-free distributions of larval emergence dates, and evolution of microscopic aquatic ecosystems. In nature, foraging animals often face difficult decisions on where to search for food. They need to make tradeoffs based on their knowledge of the relative abundance of food resources and predatory dangers as well as their current energetic condition (e.g., starvation, strong, weak). A foraging strategy consists of a list of the best foraging areas for each of the discrete energetic conditions. In optimization of animal foraging strategies, a genetic algorithm was used. It starts with an initial population of foragers that have randomly generated foraging strategies. Each forager with the initial middle energy level implements its strategy over a season. Every day over the season, the forager selects the area determined by its energy level and strategy, and it is then charged the daily metabolic cost and possibly fed and/or killed in accordance with the death and feeding-success probabilities associated with that area. The fitness of the forager and its associated strategy is measured as the average number of days it survives over the season. After all foragers and their associated strategies in a generation have obtained a fitness value, the fittest foragers are selected to produce the new generation of foragers by crossing over and mutating the strategy chromosomes of the parents. The new generation of strategies is then evaluated and the generational cycle continues until the convergence criterion is met and near-optimal strategies are found. Other examples of optimization with evolutionary computation include eco-design (Lim et al. 2013), mine planning (Riff et al. 2008), optimization of power factor and power output of wind turbines (Kusiak and Zheng 2010), and determination of the optimum pumping rates of coastal aquifers (Mantoglou et al. 2004).

The case of temporal ideal-free distributions of larval emergence dates described in Downing (1998) is an example of time series approximation. An ideal-free distribution refers to the situation where organisms diversify over space and/or time so that each gets about the same amount of resources. For species for which males often mate with several females, when male and female insects emerge from dormant stages before mating, male emergence distributions often match those of the females due to the competition for females (i.e., resources). This phenomenon is known as protandry. In the representation of a genetic algorithm, each individual is a chromosome encoding emergence times and sex. Males mate several times over several days and have either fixed active periods or age-independent mortality. Females mate only once on the emergence day. The life cycle of an insect commences with a dormant phase, which lasts until its emergence date. The insect becomes active for a fixed number of time steps, during which it shares the available food resources with other active insects. An insect's fitness is directly proportional to the amount of resources it acquires during the active period. The genetic algorithm was run on ten different cases. Each case has a different initial distribution of emergence times (the number of insects emerging on each day of the season), different frequency of the resource curve, and different postemergence life-span of the simulated insects. In all cases, the emergence distributions evolved to closely match the computed ideal-free distributions for the situation. The protandry simulations support one of the tacit assumptions of research on ideal-free distributions; that is, emergence times are genetically controlled. Other examples of time series approximation using genetic algorithms include the simulation of copepod population dynamics (Record et al. 2010) and modeling of long-term hydrological time series (Wang et al. 2009).

Timber harvest scheduling is a typical scheduling problem that can be solved using evolutionary computation. For instance, Ducheyne et al. (2004) applied genetic algorithms to develop forest harvesting plans. In the genetic algorithms, each chromosome represents a harvesting plan, encoding the felling period of each stand (gene). The fitness function maximizes the present value and minimizes deviations between successive cutting periods. It suggested that using multiple objective genetic algorithms to solve the harvest scheduling problem speeds up the optimization process and distributes the solutions evenly along the Pareto front.

1.4 Swarm Intelligence

1.4.1 Basic Principles

Swarm intelligence models the social behavior of organisms living in swarms or colonies. It is a form of agent-based modeling aiming at collective behavior of intelligent agents in decentralized, self-organized systems. Basic principles of

swarm intelligence are derived from real swarms in nature, including ant colonies, bees, bird flocking, animal herding, fish schooling, and bacterial growth. A swarm intelligence system typically consists of a population of simple agents (individuals) interacting locally with one another and with their environment. These agents follow simple rules without centralized control dictating how individuals should behave. Local interactions of agent-to-agent or agent-to-environment lead to the emergence of intelligent global behavior or pattern which is unknown to the individual agents. In addition, the behavior of agents may change when the local environment is modified, which is referred to as stigmergy (i.e., the trace left in the environment by an action incites the performance of a next action by the same or different agents).

Swarm theory has led to the development of a number of algorithms for routing and optimization (Blum and Merkle 2008). Ant colony optimization (ACO) and particle swarm optimization (PSO) are two popular swarm intelligence techniques. ACO is a probabilistic technique for solving computational problems which involve finding optimum paths through graphs. It is inspired by the behavior of ants in finding paths toward food sources or their colonies. In the real world, ants initially roam randomly. Once they find foods, they return to their colony while leaving pheromone trails. When other ants find such a trail, they are likely to follow it and reinforce it with their own pheromone if they find foods. However, the pheromone along the trail evaporates over time, reducing its attractive strength. The longer it takes for an ant to travel along the trail, the weaker and less attractive the pheromone it laid down becomes. Over a short path, the strength of the pheromone remains high as it is reinforced as fast as or quicker than it can evaporate or decay. Ants tend to choose their trails with stronger pheromone concentrations. Therefore, when one ant finds a shorter path, other ants are more likely to follow that trail, and positive feedback eventually leads all the ants choose the shortest trail.

ACO mimics the behavior of ants with artificial ants walking along a graph representing the problem to solve. The graph consists of edges linking nodes. Each edge is a path from one node to another, representing a potential solution. Each edge is also assigned a pheromone value. Figure 1.3 shows a general procedure of ACO. First, the graph is initialized by assigning the same initial pheromone value to each edge and randomly selecting a node to place an artificial ant. Then the ant selects an edge to move at the current node with a probabilistic transition rule. This rule is commonly expressed as the probability (Dorigo et al. 1996)

$$p\left(e_{i,j}\right) = \frac{\left[\tau_{i,j}\right]^{\alpha}\left[f_{i,j}\right]^{\beta}}{\sum_{k\in\theta_i}\left[\tau_{i,k}\right]^{\alpha}\left[f_{i,k}\right]^{\beta}} \tag{1.1}$$

where $p(e_{i,j})$ is the probability of $e_{i,j}$ (the edge from node i to node j) being selected at node i, $\tau_{i,j}$ is the pheromone value associated with $e_{i,j}$, $f_{i,j}$ is a value of a weighting

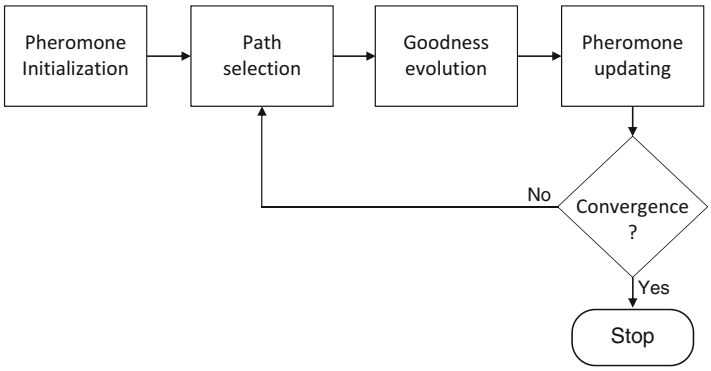

Fig. 1.3 A general procedure of ant colony optimization (ACO)

function (called a heuristic information) of $e_{i,j}$ measuring the desirability of the edge, θ_i is the set of edges available at node i, and α and β are parameters defining the relative importance of the pheromone strength and desirability. After a solution is obtained by the ant, its goodness is evaluated using an objective function. The pheromone value of each edge is then updated by uniformly decreasing all the pheromone values (pheromone evaporation) and increasing the pheromone values of one or more better solutions. The above process is repeated by applying a number of ants per iteration until a given convergence criterion (e.g., a time limit) is satisfied or all artificial ants follow the same path.

PSO simulates the social behavior of bird flocking and fish schooling. It is a search and optimization technique, similar to evolutionary computation techniques. It is initialized with a population of random solutions and searches for the optimal solutions through updating generations. However, unlike evolutionary computation techniques, it does not use evolution operators such as crossover and mutation. In PSO, individuals, referred to as particles, are grouped into a swarm. Each particle in the swarm represents a potential solution to the optimization problem. The particles are "flown" through the multidimensional search space by following the current "best" particle called guide. In each iteration, each particle is evaluated using the fitness function, and its velocity and position in the search space are updated using the following two equations (Kennedy and Eberhart 1995):

$$v(t+1) = v(t) + c_1 \times R_1 \times [\text{pbest} - p(t)] + c_2 \times R_2 \times [\text{gbest} - p(t)] \quad (1.2)$$

$$p(t+1) = p(t) + v(t+1) \quad (1.3)$$

where $v(t)$ is the current velocity of the particle, $p(t)$ is its current position, pbest and gbest are the best solutions achieved so far by the particle and population, respectively, R_1 and R_2 are the random numbers in the range [0, 1], and c_1 and c_2 are acceleration coefficients. pbest is also called the personal best, and gbest the global

best. The process terminates when a time limit is reached or gbest cannot be improved further. While ACO is mainly used for combinatorial optimization, PSO is primarily employed for continuous optimization.

1.4.2 Applications

The study of swarm intelligence is providing insights that can help humans manage complex systems, from truck routing to military robots. ACO has many successful applications in discrete optimization problems such as travelling salesman problem. For example, Afshar (2010) used ACO for optimal design of sewer networks. In this study, the nodal elevations of the network were used as the decision variables of the optimization problem. The pheromone concentration over the allowable range of each decision variable was modeled with a Gaussian probability density function. It tried two alternative approaches to the implementation of ACO: constrained and unconstrained. The unconstrained approach did not take into account the minimum slope and other constraints. In the constrained approach, the elevation at downstream node of a pipe is used to define new bounds on the elevation of the upstream node, which represents the constraints on the pipe slopes. Other constraints include hydraulic radius, pipe diameter, average excavation depth, velocity, and flow depth of each link. The results from the constrained approach were compared with those of the unconstrained one. The constrained ACO was shown to be very effective in locating the optimal solution and efficient in terms of the convergence. It was also found to be relatively insensitive to the initial colony and its size when compared to the unconstrained algorithm.

PSO has few or no assumptions about the problem being optimized. It can search large solution spaces. Ma et al. (2011) applied PSO for land use optimization. In their study, each land parcel is abstracted to a particle by its centroid. Particles constantly fly to adjust their positions and velocities according to their personal best and the global best assessed in terms of the cost of land use transformation, biophysical suitability, and compactness of landscape. Chou (2012) used PSO in modeling rainfall–runoff relationships. The study compared PSO and a simple linear model in simultaneous identification of system structure and parameters of the rainfall–runoff relationship. The simple linear model combines classic models typically used in hydrology to simulate the subsystems, and transforms the system structure identification problem into a combinational optimization problem. The PSO was employed to select the optimized subsystem model with the best data fit. It found that the PSO simulates the time of peak arrival more accurately compared to the simple linear model, and it also accurately identifies the system structure and parameters of the rainfall–runoff relationship. PSO has also been applied for forecasting river stage (Chau 2007), modeling turbidity intrusion processes in flooding season (Wang et al. 2012), simulating soil moisture (Alizadeh and Mousavi 2013), optimizing greenhouse climate model parameters (Hasni et al. 2011), scheduling electric power production at a wind farm, predicting penetration rate of hard rock tunnel boring machines (Yagiz and Karahan 2011), etc.

1.5 Artificial Immune Systems

1.5.1 Basic Principles

AIS model the human immune system. The human immune system is a robust, decentralized, error-tolerant, and adaptive system. It is composed of a great variety of molecules, cells, and organs spreading all over the body. The main function of the human immune system is to search for malfunctioning cells from its own body (such as cancer cells) and foreign disease causing elements (such as bacteria and viruses). The elements that can be recognized by the immune system are referred to as antigens. The cells belonging to the body are called self or self-antigens, while the foreign cells entering the body are termed nonself or nonself-antigens. The immune system can distinguish between self and nonself. The field of AIS encompasses a spectrum of algorithms. Different algorithms mimic the behavior and properties of different immunological cells (specifically B-cells, T-cells, and dendritic cells). There are three main categories of AIS algorithms derived from the simplified immune systems: negative selection, clonal selection, and immune networks.

Negative selection algorithms simulate the negative selection process that occurs during the maturation of T-cells in the thymus. T-cells originate in the bone marrow, but pass on to the thymus to mature before they circulate the body in the blood and lymphatic vessels. Negative selection refers to the identification and elimination of those T-cells that may recognize and attack the self-antigens presented in the thymus. All T-cells that leave the thymus to circulate throughout the body are said to be tolerant to self. A typical negative selection algorithm involves a self-set S which defines the self-elements in a problem space (i.e., representative samples of self-antigen), and a detector set F which contains all elements that have been identified as nonself-antigens and do not belong to S (Dasgupta and Forrest 1999; de Castro and Timmis 2002). Basically, the algorithm first generates a random set of candidate elements C, and then compares the elements in C with the elements in S. If an element in C matches an element in S, it will be discarded; otherwise, it will be added to F. D'haeseleer et al. (1996) proposed a more efficient approach that tries to minimize the number of generated detector elements while maximizing the coverage of the nonself-space. After F is generated, it is used to detect nonself-elements in a data set (feature set) S^*, which may be composed of the set S plus other new features or a completely new set. The elements in S^* are checked against those in F. A match indicates a nonself-element is identified and an action will be followed. The follow-up action of detecting nonself varies according to the problem under investigation. The efficiency and complexity of a negative selection algorithm depend on the type of problem space (continuous, discrete, mixed, etc.), the problem representation scheme, and the matching rules. Most of the research works on the negative selection algorithm have used the binary matching rules like r-contiguous (Forrest et al. 1994). Negative selection algorithms have been applied to pattern recognition and classification.

Clonal selection algorithms are inspired by the clonal selection theory that explains how the immune system reacts when a nonself-antigen is detected by a B-cell. B-cells produce antibodies. When an antibody strongly matches an antigen, the corresponding B-cell is stimulated to generate clones of itself, which then produce more antibodies. The binding of an antibody to a nonself-antigen is a signal to destroy the invading organism on which the antigen is found. This process is called clonal selection. Clonal selection algorithms are most commonly applied to optimization and pattern recognition. They generally evolve candidate solutions by means of selection, cloning, and mutation. de Castro and von Zuben (2002) proposed a clonal selection algorithm for pattern recognition, which includes the following steps:

1. Create a random population of individuals (B-cells or antibodies).
2. Given a set of patterns (antigens) to be recognized, for each pattern, determine its affinity with each element of the population.
3. Select a number of the highest affinity individuals from the population and clone these individuals to a certain number of copies proportional to their affinity with the antigen. The greater the affinity, the larger the number of copies, and vice versa.
4. Mutate all the clones with a rate inversely proportional to their affinity with the input pattern. The higher the affinity, the smaller the mutation rate, and vice versa.
5. Add the mutated individuals to the population, and reselect the maturated (optimized) individuals to be kept as memories of the system. Delete other superfluous clones and replace them with new randomly generated individuals.
6. Repeat steps 2–5 until a stop condition is met, e.g., a minimum pattern recognition or classification error.

There are several variants of the clonal selection algorithm and clonal selection-based hybrid algorithms, which are reviewed in Berna and Sadan (2011).

Immune network algorithms assume that B-cells form a network. When a B-cell is activated as a response to an antigen, it stimulates all other B-cells to which it connects in the network. These algorithms are similar to clonal selection algorithms in that they both measure the goodness of B-cells by affinities and both involve a process of selection, cloning, and mutation. The main difference is that the immune network algorithms consider that B-cells are not isolated, but communicate with each other via collective dynamic network interactions, while clonal selection algorithms only care about the interactions between B-cells and antigens. An immune network algorithm develops a population of individuals (B-cells) that interact with data (antigens) and with each other. The interactions with antigens and between B-cells fire up the B-cells. Highly stimulated B-cells undertake cloning and mutation as they do in a clonal selection algorithm. The number of clones and mutation rate also depend on the affinity of the cell with the current stimulating antigen. This process is regulated by the interaction between B-cells, which can stimulate them in order to create a memory of observed antigens, or suppress them, in order to control the immune response. It also includes natural

death of unstimulated B-cells and addition of new random B-cells to the population. A working procedure of an artificial immune network can be found in de Castro and von Zuben (2001). Immune network algorithms perform unsupervised learning. They have been typically used for clustering, but have also been adapted to optimization and classification.

1.5.2 Applications

AIS have been successfully used for optimization, classification/clustering, and pattern recognition. For example, Liu et al. (2012) applied AIS for optimizing multi-objective allocation of water resources in river basins. They integrated the macroevolution algorithm (Marin and Sole 1999), clonal selection, and an entropy-based density assessment scheme (EDAS) to perform a global optimal search. The clonal selection was based on the diversity in the evolving population and applied for solution exploitation. EDAS was used to distribute non-dominated individuals uniformly along the discovered Pareto-frontier, and the macroevolution algorithm is employed to preserve the diversity of individuals and form part of the pool solution.

AIS have been widely used for remote sensing image classification and pattern recognition (Xu and Wu 2008; Zhang et al. 2004; Zheng and Li 2007; Zhong et al. 2007). Gong et al. (2011) developed an improved artificial immune network algorithm for land cover classification from remote sensing images. It involves creation of land cover class representatives as antibodies or B-cells. Basically, an initial population of antibodies is randomly generated so that the possibility of successful recognition of a land cover class is maximized without prior knowledge of antigens. The initialized population of antibodies is then optimized by cloning and mutating the antibodies that can best recognize the antigens. The system then evolves antibodies over a number of generations until stopping criteria are met. In this study, the best antibodies for each land cover class were preserved in each generation. An adaptive mutation rate was used to adjust the learning speed in response to the difference between the classification accuracies of the current and previous generation. In addition, the Euclidean distance and spectral angle mapping distance are used as affinity measures. A genetic algorithm was also used to identify optimal weights representing contributions from different affinity measures. The artificial immune network algorithm was applied to classify land covers in a residential area in Denver, CO, with high-spatial resolution QuickBird image and LiDAR data and in a suburban area in Monticello, UT, with HyMap hyperspectral imagery. The method was compared with a decision tree, a multilayer feed-forward back-propagation neural network, and another artificial immune networks algorithm from de Castro and von Zuben (2001). The results showed that it outperformed the other methods with higher accuracy and more spatially cohesive land cover classes with limited salt-and-pepper effect.

1.6 Fuzzy Systems

1.6.1 Basic Principles

Fuzzy systems make use of fuzzy sets or fuzzy logic. Traditional set theory requires objects to be either part of a set or not. For example, the set of rice paddy fields is distinct from the set of forest stands. If a piece of land belongs to one of the two sets, it cannot belong to the other. Such sets are called crisp. Crisp sets have well-defined boundaries with no ambiguity about an object's membership. However, in earth and environmental studies, our observations and reasoning are often not this exact and usually include a measure of membership. For instance, many data collected in the field survey are often described in ambiguous words: soils can be recorded as being poorly drained, slightly susceptible to soil erosion, and marginally suitable for maize. Such sets are fuzzy. With fuzzy sets, an object belongs to a set to a certain degree of membership. Mathematically, a crisp set is described by a characteristic function whose value is always either 1 for elements of the set or 0 for those outside the set (Fig. 1.4a). A fuzzy set is defined by a membership function that takes values in the range of 0 and 1 (Fig. 1.4b). An element belongs to a fuzzy set if the value of the set's membership function at that element is nonzero. A nonzero value of the membership function indicates the degree to which an element belongs to the set.

Fuzzy logic allows approximate reasoning based on fuzzy sets. It usually uses IF-THEN rules. The following is an example of a fuzzy rule:

> IF the soil depth is shallow and accumulated temperature is moderate
> THEN the suitability for maize is marginal

The AND, OR, and NOT logic operators of traditional Boolean logic are also used in fuzzy logic. The AND operation is the intersection of fuzzy sets, given by the minimum of the membership functions. OR is the union of fuzzy sets, defined as the

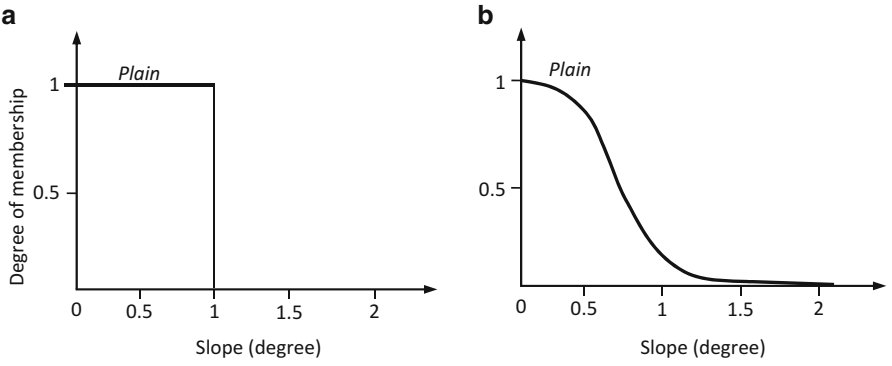

Fig. 1.4 Crisp set vs. fuzzy set defining plain. (**a**) Characteristic function, (**b**) fuzzy membership function

maximum of the membership functions. NOT gives the complement of a fuzzy set. Other operators exist in fuzzy logic (Zimmermann 2001).

A fuzzy system can be defined as a collection of IF-THEN rules with fuzzy predicates, or an algebraic or differential equation in which parameters are fuzzy numbers expressing the vagueness in the parameter values, or a system with fuzzy inputs (such as readings from unreliable sensors or quantities relating to human perception) and fuzzy outputs. In the earth and environmental systems, the majority of the phenomena are understood only partially and therefore cannot be modeled using mathematical models. A significant portion of knowledge about the earth and environmental systems is available as the heuristic knowledge or rule of thumb from experts or practitioners. Fuzzy rule-based systems can be used to represent such knowledge and make approximate reasoning using the heuristic knowledge. From this point of view, fuzzy systems can be considered as a type of expert systems or knowledge-based systems (Patterson 1990). In addition, fuzzy systems provide an alternative approach to dealing with uncertainty as not all types of uncertainty can be handled by traditional stochastic modeling framework. The uncertainty in fuzzy systems is non-statistical, based on vagueness, imprecision, and/or ambiguity. Non-statistical uncertainty is an inherent property of a system and cannot be changed or resolved by observations. Stochastic methods deal with statistical uncertainty, which is based on the laws of probability and can be resolved through observations. Moreover, like other computational intelligence techniques such as ANN, fuzzy systems can be used to model nonlinear systems and approximate other functions or measurement data with a desired accuracy. They provide a more transparent representation of the system under investigation, mainly due to the linguistic model interpretation in the way close to the one humans use for reasoning about the real world.

1.6.2 Applications

Fuzzy systems are often a choice when dealing with non-statistic uncertainties in applications including classification/clustering, function approximation, and prediction. Rezaei et al. (2013) provides an example of classification using fuzzy logic in evaluation of groundwater pollution. In this study, groundwater vulnerability was assessed by using linguistic variables to describe hydrogeological characteristics and linguistic terms to define vulnerability ratings. The Boolean logic cannot reflect the actual differences between the points in a hydrogeological setting; thus, regions that should have different vulnerability indices may be characterized by the same index. In contrast, a fuzzy system is able to adjust itself with the range of variation of input indices. The study demonstrated that fuzzy logic allows better and more logical ratings to be obtained for the values located near the classification boundaries. Other examples include using fuzzy clustering methods to characterize the physicochemical properties of groundwater and assess the impact of anthropogenic activities on the groundwater hydrology

and chemistry (Güler et al. 2012), and to cluster forests according to their forest fire risk (Iliadis et al. 2010).

An example of function approximation is the use of fuzzy systems to downscale the regional wind climate to local wind conditions, considering the surrounding topography, in order to assess the wind potential available in an area for wind farms (de la Rosa et al. 2011). Wind exhibits high local-scale variability caused by the local topography, roughness, obstacles, etc. Although the regional wind climate can be extrapolated from the wind measurements at the meteorological stations, it does not reflect the local variability. The real wind is the result of local conditions. Due to the vagueness in the terrain description (e.g., down slope, up slope, and plain) in the local scale, the low quality of meteorological data (without measuring local topographic effects) recorded at the stations, and the chaotic dynamics inherent to atmospheric events, de la Rosa et al. (2011) proposed to use a fuzzy system to transform the regional wind distribution into the real one, which takes into account topographic parameters. The fuzzy system was designed to establish a link between the local wind conditions and the terrain features. It calculates the probability of possible changes in the wind direction based on the analysis of the terrain in those directions. The membership distributions of the fuzzy system were also optimized using a genetic algorithm. The fuzzy system effectively approximated the local wind conditions by transforming a regional wind climate model.

In the study by Kayastha (2012), a fuzzy system approach was used to predict landslide susceptibility. Eight causative factors were used in the predictive modeling, including slope, aspect, slope shape, relative relief, distance from drainage, land use, geology, and distance from active faults. Likelihood ratios were calculated for each class of the causative factors by comparing with past landslide occurrences. The likelihood ratios were then normalized to fuzzy membership values between 0 and 1. The study compared several different fuzzy operators and found that the fuzzy gamma (λ) operator with a λ-value of 0.70 produced the best prediction accuracy. Other instances of prediction with fuzzy systems include prediction of air quality (Fisher 2003; Carbajal-Hernández et al. 2012), ocean wave energy (Özger 2011), and habitat quality (Mocq et al. 2013).

1.7 Conclusions

The complexity and nonlinearity of most earth and environmental problems have led to the increased use of computational intelligence techniques. This chapter reviewed five representative methods of computational intelligence and their applications. The characteristics of these techniques reveal that each technique has its own merits and limitations. Fuzzy systems, for instance, are good at approximate reasoning, but do not have learning and optimization ability, while ANN is capable of adaptive learning and evolutionary computation is efficient in intelligent search

and optimization. Fusion of different computational intelligence techniques thus may provide better computational models. For example, coupling ANN and fuzzy systems provides capabilities of learning from approximate data or knowledge and approximate reasoning using the knowledge derived through adaptive learning (Azar 2010). Attempts to combine evolutionary computation and ANN have a long history. For example, evolutionary computation techniques can be used to determine the weights of an ANN with the mean square error sum of the neurons at the output layer as the fitness function (Piotrowski and Napiorkowski 2011). Evolutionary computation can also be used to optimize parameters of a fuzzy system and adapt the membership distribution to optimize the performance of a fuzzy system (Sanchez et al. 1997; de la Rosa et al. 2011). An ANN can be trained with fuzzy membership distributions that have been optimized by an evolutionary computation technique. Moreover, evolutionary computation may be used to determine the best set of training data in an ANN-fuzzy system (Azar 2010). Chapter 2 of this book provides a more extensive overview of computational intelligence applications in earth and environmental sciences. Other chapters in this book offer more technical discussions on some computational intelligence techniques including those that were not reviewed in this chapter.

Computational intelligence is a collection of computational models and tools, whose classification, clusterization, optimization, prediction, reasoning, and approximation capabilities have been improved incrementally and continuously. There are already many computational intelligence techniques or combinations of the techniques. It is always possible to find alternative techniques to address a specific earth and environmental problem. However, domain-specific knowledge underlying the physical, chemical, or biological processes in an earth or environmental system under investigation should be incorporated into the problem formulation, selection of appropriate computational intelligence techniques, and evaluation of modeling results. There is also a need to incorporate computational intelligence techniques into existing modeling framework so that they can be widely accepted in the communities who are traditionally accustomed to process models. Computational intelligence provides practical tools for earth and environmental scientists to handle heterogeneous, incomplete, and noise data and to build models of uncertainty trained on the historical, current, and predicted data. We will see more advances in computational intelligence techniques and more innovative use of these techniques in earth and environmental applications in the future.

References

Aertsen W, Kint V, van Orshoven J, Özkan K, Muys B (2010) Comparison and ranking of different modelling techniques for prediction of site index in Mediterranean mountain forests. Ecol Model 221(8):1119–1130

Afshar MH (2010) A parameter free Continuous Ant Colony Optimization Algorithm for the optimal design of storm sewer networks: constrained and unconstrained approach. Adv Eng Softw 41(2):188–195

Alizadeh H, Mousavi SJ (2013) Coupled stochastic soil moisture simulation-optimization model of deficit irrigation. Water Resour Res 49(7):4100–4113

Antanasijevia DZ, Pocajt VV, Povrenovia DS, Ristia MA, Peria-Grujia AA (2013) PM.sub.10 emission forecasting using artificial neural networks and genetic algorithm input variable optimization. Sci Total Environ 443:511–519

Areerachakul S, Sophatsathit P, Lursinsap C (2013) Integration of unsupervised and supervised neural networks to predict dissolved oxygen concentration in canals. Ecol Model 261–262:1–7

Auger A, Hansen N (2011) Theory of evolution strategies: a new perspective. In: Auger A, Doerr B (eds) Theory of randomized search heuristics, foundations and recent developments. World Scientific, Singapore, pp 289–325

Azar AT (2010) Adaptive neuro-fuzzy systems. In: Azar AT (ed) Fuzzy systems. InTech, Croatia, pp 85–110

Banerjee P, Singh VS, Chatttopadhyay K, Chandra PC, Singh B (2011) Artificial neural network model as a potential alternative for groundwater salinity forecasting. J Hydrol 398(3):212–220

Bao YH, Ren JB (2011) Wetland landscape classification based on the BP neural network in DaLinor Lake Area. Procedia Environ Sci 10:2360–2366

Berna HU, Sadan KK (2011) A review of clonal selection algorithm and its applications. Artif Intell Rev 36(2):117–138

Blum C, Merkle D (2008) Swarm intelligence: introduction and applications. Springer, Berlin

Bui DT, Pradhan B, Lofman O, Revhaug I, Dick OB (2012) Landslide susceptibility mapping at Hoa Binh Province (Vietnam) using an adaptive neuro-fuzzy inference system and GIS. Comput Geosci 45:199–211

Carbajal-Hernández JJ, Sánchez-Fernández LP, Carrasco-Ochoa JA, Martínez-Trinidad JF (2012) Assessment and prediction of air quality using fuzzy logic and autoregressive models. Atmos Environ 60:37–50

Chang F, Tsai W, Chen H, Yam RS, Herricks EE (2013) A self-organizing radial basis network for estimating riverine fish diversity. J Hydrol 476:280–289

Chau KW (2007) A split-step particle swarm optimization algorithm in river stage forecasting. J Hydrol 346(3):131–135

Chou CM (2012) Particle swarm optimization for identifying rainfall-runoff relationships. J Water Resour Protect 4:115–126

Cruz-Ramírez M, Hervás-Martínez C, Jurado-Expósito M, López-Granados F (2012) A multi-objective neural network based method for cover crop identification from remote sensed data. Expert Syst Appl 39(11):10038–10048

D'haeseleer P, Forrest S, Helman P (1996) An immunological approach to change detection: algorithms, analysis, and implications. In: Proceedings of the IEEE symposium on computer security and privacy. IEEE Computer Society Press, Los Alamitos, CA, pp 110–119

Dasgupta D, Forrest S (1999) An anomaly detection algorithm inspired by the immune system. In: Dasgupta D (ed) Artificial immune systems and their applications. Springer, Berlin, pp 262–277

de Castro LN, Timmis J (2002) Artificial immune systems: a novel paradigm to pattern recognition. In: Corchado JM, Alonso L, Fyfe C (eds) Artificial neural networks in pattern recognition. SOCO-2002. University of Paisley, Paisley, England, pp 67–84

de Castro LN, von Zuben FJ (2001) aiNet: an artificial immune network for data analysis. In: Abbass HA, Sarker RA, Newton CS (eds) Data mining: a heuristic approach (Chapter XII). Idea Group Publishing, Hershey, PA, pp 231–259

de Castro LN, von Zuben FJ (2002) Learning and optimization using the clonal selection principle. IEEE Trans Evol Comput 6(3):239–251

de la Rosa JJG, Pérez AA, Salas JCP, Leo JGR, Muñoz AM (2011) A novel inference method for local wind conditions using genetic fuzzy systems. Renew Energy 36:1747–1753

Dobreva ID, Klein AG (2011) Fractional snow cover mapping through artificial neural network analysis of MODIS surface reflectance. Remote Sens Environ 115(12):3355–3366

Dorigo M, Maniezzo V, Colorni A (1996) Ant system: optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybern B Cybern 26(1):29–41

Downing K (1998) Using evolutionary computational techniques in environmental modelling. Environ Model Softw 13:519–528
Ducheyne EI, de Wulf RR, de Baets B (2004) Single versus multiple objective genetic algorithms for solving the even-flow forest management problem. For Ecol Manage 201(2):259–273
Eiben AE, Smith JE (2003) Introduction to evolutionary computing. Springer, New York
Engelbrecht AP (2007) Computational intelligence: an introduction. Wiley, Chichester, England
Fegh A, Riahi MA, Norouzi GH (2013) Permeability prediction and construction of 3D geological model: application of neural networks and stochastic approaches in an Iranian gas reservoir. Neural Comput Appl 23(6):1763–1770. doi:10.1007/s00521-012-1142-8
Fisher B (2003) Fuzzy environmental decision-making: applications to air pollution. Atmos Environ 37(14):1865–1877
Forrest S, Perelson AS, Allen L, Cherukuri R (1994) Self-nonself discrimination in a computer. In: Proceedings of the IEEE symposium on research in security and privacy. IEEE Computer Society Press, Los Alamitos, CA, pp 202–212
Gardner MW, Dorling SR (1999) Neural network modelling and prediction of hourly NO_x and NO_2 concentrations in urban air in London. Atmos Environ 33:709–719
Goldberg DE (1989) Genetic algorithm in search, optimization, and machine learning. Addison-Wesley, Reading, MA
Gong B, Im J, Mountrakis G (2011) An artificial immune network approach to multi-sensor land use/land cover classification. Remote Sens Environ 115:600–614
Güler C, Kurt MA, Alpaslan M, Akbulut C (2012) Assessment of the impact of anthropogenic activities on the groundwater hydrology and chemistry in Tarsus coastal plain (Mersin, SE Turkey) using fuzzy clustering, multivariate statistics and GIS techniques. J Hydrol 414–415:435–451
Hagan MT, Demuth HB, Beale MH (1996) Neural network design. PWS Publishing, Boston
Hasni A, Taibi R, Draoui B, Boulard T (2011) Optimization of greenhouse climate model parameters using particle swarm optimization and genetic algorithms. Energy Procedia 6:371–380
Haykin S (1999) Neural networks: a comprehensive foundation, 2nd edn. Prentice-Hall, Upper Saddle River, NJ
Huo S, He Z, Su J, Xi B, Zhu C (2013) Using artificial neural network models for eutrophication prediction. Procedia Environ Sci 18:310–316
Iliadis LS, Vangeloudh M, Spartalis S (2010) An intelligent system employing an enhanced fuzzy c-means clustering model: application in the case of forest fires. Comput Electron Agric 70(2):276–284
Kakaei LE, Moghaddam NA, Ahmadi A (2013) Daily suspended sediment load prediction using artificial neural networks and support vector machines. J Hydrol 478:50–62
Kayastha P (2012) Application of fuzzy logic approach for landslide susceptibility mapping in Garuwa sub-basin, East Nepal. Front Earth Sci 6(4):420–432
Kennedy J, Eberhart RC (1995) Particle swarm optimization. In: Proceedings of IEEE international conference on neural networks, vol 4. IEEE Press, Piscataway, NJ, pp 1942–1948
Konar A (2005) Computational intelligence: principles, techniques and applications. Springer, Berlin
Krasnopolsky VM (2013) The application of neural networks in the earth system sciences. Springer, The Netherlands
Kuo J, Hsieh M, Lung W, She N (2007) Using artificial neural network for reservoir eutrophication prediction. Ecol Model 200(1):171–177
Kusiak A, Zheng H (2010) Optimization of wind turbine energy and power factor with an evolutionary computation algorithm. Energy 35(3):1324–1332
Langdon WB, Poli R (2002) Foundations of genetic programming. Springer, Berlin
Lek S, Guegan JF (1999) Artificial neural networks as a tool in ecological modelling, an introduction. Ecol Model 120:65–73

Lim S, Kim YR, Woo SH, Park D, Park JM (2013) System optimization for eco-design by using monetization of environmental impacts: a strategy to convert bi-objective to single-objective problems. J Clean Prod 39:303–311

Liu D, Guo S, Chen X, Shao Q, Ran Q, Song X, Wang Z (2012) A macro-evolutionary multi-objective immune algorithm with application to optimal allocation of water resources in Dongjiang River basins, South China. Stoch Environ Res Risk Assess 26(4):491–507

Lohani AK, Kumar R, Singh RD (2012) Hydrological time series modeling: a comparison between adaptive neuro-fuzzy, neural network and autoregressive techniques. J Hydrol 442–443:23–35

Ma S, He J, Liu F, Yu Y (2011) Land-use spatial optimization based on PSO algorithm. Geo-spat Inf Sci 14:54–61

Madani K (2011) Computational intelligence. Springer, Berlin

Mantoglou A, Papantoniou M, Giannoulopoulos P (2004) Management of coastal aquifers based on nonlinear optimization and evolutionary algorithms. J Hydrol 297(1):209–228

Marin J, Sole RV (1999) Macroevolutionary algorithms: a new optimization method on fitness landscapes. IEEE Trans Evol Comput 3(4):272–286

Mas JF (2004) Mapping land use/cover in a tropical coastal area using satellite sensor data, GIS and artificial neural networks. Estuar Coast Shelf Sci 59:219–230

Mocq J, St-Hilaire A, Cunjak RA (2013) Assessment of Atlantic salmon (Salmo salar) habitat quality and its uncertainty using a multiple-expert fuzzy model applied to the Romaine River (Canada). Ecol Model 265:14–25

Okkan U (2012) Wavelet neural network model for reservoir inflow prediction. Sci Iran 19(6):1445–1455

Özçelik R, Diamantopoulou MJ, Crecente-Campo F, Eler U (2013) Estimating Crimean juniper tree height using nonlinear regression and artificial neural network models. For Ecol Manage 306:52–60

Özger M (2011) Prediction of ocean wave energy from meteorological variables by fuzzy logic modeling. Expert Syst Appl 38(5):6269–6274

Patterson DW (1990) Introduction to artificial intelligence and expert systems. Prentice-Hall, Englewood Cliffs, NJ

Piotrowski AP, Napiorkowski JJ (2011) Optimizing neural networks for river flow forecasting—evolutionary computation methods versus the Levenberg–Marquardt approach. J Hydrol 407:12–27

Pontin DR, Schliebs S, Worner SP, Watts MJ (2011) Determining factors that influence the dispersal of a pelagic species: a comparison between artificial neural networks and evolutionary algorithms. Ecol Model 222(10):1657–1665

Principe JC, Euliano NR, Lefebvre WC (2000) Neural and adaptive systems: fundamentals through simulations. Wiley, New York

Ranković V, Radulović J, Radojević I, Ostojić A, Čomić L (2010) Neural network modeling of dissolved oxygen in the Gruža reservoir, Serbia. Ecol Model 221(8):1239–1244

Record NR, Pershing AJ, Runge JA, Mayo CA, Monger BC, Chen C (2010) Improving ecological forecasts of copepod community dynamics using genetic algorithms. J Mar Syst 82(3):96–110

Rezaei F, Safavi H, Ahmadi A (2013) Groundwater vulnerability assessment using fuzzy logic: a case study in the Zayandehrood aquifers, Iran. Environ Manage 51:267–277

Riff MC, Alfaro T, Bonnaire X, Grandon C (2008) EA-MP: an evolutionary algorithm for a mine planning problem. In: Proceedings of IEEE congress on evolutionary computation, June 2008, pp 4011–4014

Rumelhart DE, Hinton GE, Williams RJ (1986) Learning internal representations by error propagation. In: Rumelhart DE, McClelland JL (eds) Parallel distributed processing: explorations in the microstructure of cognition, vol 1. MIT Press, Cambridge, MA, pp 318–362

Sanchez E, Shibata T and Zadeh LA (1997) Genetic algorithms and fuzzy logic systems: soft computing perspectives. World Scientific Pub., Singapore; River Edge, NJ

Song K, Park Y, Zheng F, Kang H (2013) The application of Artificial Neural Network (ANN) model to the simulation of denitrification rates in mesocosm-scale wetlands. Ecol Inform 16:10–16

Sousa SIV, Martins FG, Alvim-Ferraz MCM, Pereira MC (2007) Multiple linear regression and artificial neural networks based on principal components to predict ozone concentrations. Environ Model Softw 22:97–103

Torkar D, Zmazek B, Vaupotič J, Kobal I (2010) Application of artificial neural networks in simulating radon levels in soil gas. Chem Geol 270:1–8

Wang W, Xu D, Qiu L, Ma J (2009) Genetic programming for modelling long-term hydrological time series. In: Proceedings of the fifth international conference on natural computation, Aug 2009, vol 4, pp 265–269

Wang S, Qian X, Wang QH, Xiong W (2012) Modeling turbidity intrusion processes in flooding season of a canyon-shaped reservoir, South China. Procedia Environ Sci 13:1327–1337

Watts MJ, Li Y, Russell BD, Mellin C, Connell SD, Fordham DA (2011) A novel method for mapping reefs and subtidal rocky habitats using artificial neural networks. Ecol Model 222(15):2606–2614

Wu CL, Chau KW (2011) Rainfall–runoff modeling using artificial neural network coupled with singular spectrum analysis. J Hydrol 399(3–4):394–409

Xu S, Wu Y (2008) An algorithm for remote sensing image classification based on artificial immune B-cell network. In: The international archives of the photogrammetry, remote sensing and spatial information sciences, vol XXXVII, part B6b, pp 107−112

Yagiz S, Karahan H (2011) Prediction of hard rock TBM penetration rate using particle swarm optimization. Int J Rock Mech Mining Sci 48(3):427–433

Yang Y, Rosenbaum MS (2003) Artificial neural networks linked to GIS. In: Nikravesh M, Aminzadeh F, Zadeh LA (eds) Developments in petroleum science, vol 51, Soft computing and intelligent data analysis in oil exploration. Elsevier, The Netherlands, pp 633–650

Yao X, Liu Y, Lin G (1999) Evolutionary programming made faster. IEEE Trans Evol Comput 3(2):82–102

Yoo J, Lee Y, Lee C, Kim C (2012) Effective prediction of biodiversity in tidal flat habitats using an artificial neural network. Mar Environ Res 83:1–9. doi:10.1016/j. marenvres.2012.10.001

Yu X, Gen M (2010) Introduction to evolutionary algorithms. Springer, New York

Zhang W (2010) Computational ecology: artificial neural networks and their applications. World Scientific, Singapore

Zhang X, Shan T, Jiao L (2004) SAR image classification based on immune clonal feature selection. In: Mohamed SK, Aurélio CC (eds) Proceedings of image analysis and recognition, vol 3212, Lecture notes in computer science. Springer, Berlin, pp 504–511

Zheng H, Li L (2007) An artificial immune approach for vehicle detection from high resolution space imagery. Int J Comput Sci Network Security 7:67–72

Zhong Y, Zhang L, Huang B, Li P (2007) A resource limited artificial immune system algorithm for supervised classification of multi/hyper-spectral remote sensing imagery. Int J Remote Sens 28:1665–1686

Zimmermann H (2001) Fuzzy set theory and its applications. Kluwer Academic, Boston