BRIEFINGS IN BIOINFORMATICS. VOL 9. NO 4. 307–316
Advance Access publication May 5, 2008
doi:10.1093/bib/bbn021

Computational intelligence approaches for pattern discovery in biological systems

Gary B. Fogel

Submitted: 21st January 2008; Received (in revised form): 29th March 2008

Abstract

Biology, chemistry and medicine are faced by tremendous challenges caused by an overwhelming amount of data and the need for rapid interpretation. Computational intelligence (CI) approaches such as artificial neural networks, fuzzy systems and evolutionary computation are being used with increasing frequency to contend with this problem, in light of noise, non-linearity and temporal dynamics in the data. Such methods can be used to develop robust models of processes either on their own or in combination with standard statistical approaches. This is especially true for database mining, where modeling is a key component of scientific understanding. This review provides an introduction to current CI methods, their application to biological problems, and concludes with a commentary about the anticipated impact of these approaches in bioinformatics.

Keywords: *computational intelligence; artificial neural networks; fuzzy logic; evolutionary computation; machine learning; bioinformatics*

INTRODUCTION

The exponential increase in the rate of biological data collection has mandated greater importance on the use of computational approaches for data analysis [1]. The discipline of bioinformatics grew out of this need, synthesizing informatics and computers for the organization, analysis, and interpretation of biological data. Concurrent to the development of bioinformatics, rapid advances in computer science, including the Internet, have helped to keep pace with the rate of biological data collection, organization and transfer. However, improved methods of data analysis will constantly be in demand, especially those that can more rapidly identify features of importance to the researcher and/or provide pattern recognition models that increase specificity and sensitivity. For this reason, it is important that biologists keep appraised of continuing advances in computer science, and identify appropriate tools for the problem at hand. Versatility over a wide variety of problems, data types and expression is therefore a key to successful data mining. In addition, resulting models must be easily interpreted by the end-user, so that key insight can be fed back iteratively into the data collection and modeling process. While it is not always easy to identify the right modeling approach for each problem at hand, identifying the appropriate model representation in light of these goals is often ignored in favor of speed to a solution, even if the approach is inferior.

As a solution to this issue, methods of computational intelligence (CI) (artificial neural networks (ANNs), fuzzy systems, evolutionary computation and other bio-inspired algorithms) are being applied more widely to bioinformatics problems [2, 3]. This review provides a short primer for each of these disciplines, followed by example applications in bioinformatics, specializing on phylogenetic analysis and transcription factor binding site identification.

ANNs

ANNs are mathematical constructs based loosely on neuronal structure [4–6]. They are transfer functions that accept input features and produce an

Corresponding author. Gary B. Fogel, Natural Selection, Inc., 9330 Scranton Rd., Suite 150, San Diego, CA 92121, USA. Tel: 858 455 6449; Fax: 858 455 1560; E-mail: gfogel@natural-selection.com

Gary B. Fogel is a Vice President at Natural Selection, Inc. He works on applications of computational intelligence approaches to bioinformatics, chemoinformatics and personalized medicine, with special interest in drug discovery, diagnostics and systems biology.

output decision. Typically, ANNs are trained over a set of examples such as features regarding nucleotide sequence information with an output being a decision concerning the likelihood that whether a genomic region is a coding region or not. ANN architectures are typically constructed so that they can include non-linear processing features interconnected by fixed or variable weights. Given sufficient complexity in the architecture, there exists an ANN that will map every input pattern to the appropriate output pattern so long as the mapping of inputs to outputs is not one-to-many. There are many types of ANNs including feed-forward neural networks (a common architecture), radial basis function networks, Kohonen self-organizing maps, recurrent networks and so forth. Each of these ANN representations is useful for particular problem sets and the researcher should familiarize themselves with their use before deciding which is best for the problem at hand (Figure 1).

Once an appropriate architecture (i.e. type of network, number of layers, number of nodes per layer, connections between nodes) is chosen, and a training set of input patterns is developed, the collection of variable weights on the ANN determines the output for each presented pattern. Each ANN can then be scored in light of a fitness metric that typically minimizes the squared error between the actual and the target values (other scoring functions can be used depending on the problem). There are three major learning paradigms associated with ANNs; supervised learning, unsupervised learning and reinforcement learning. Supervised learning requires the use of a set of training examples and actual outputs. The training examples are used to develop a model that relates features about the examples (inputs) to an output decision. This is often used in pattern recognition where there are target patterns in a database and for each target numerous possible features that could be used as input to an ANN. In unsupervised learning, the target patterns remain unknown. This makes the problem much harder in that the ANN must be tuned to make correct decisions in an absence of known truth. Clustering is an often cited type of unsupervised learning where the goal is to simply identify similarities between objects in a dataset (with the hope that properly clustered items will lead to intuitive classification). These approaches require a useful metric of what it means to be 'similar' and in some cases this can be difficult to define.

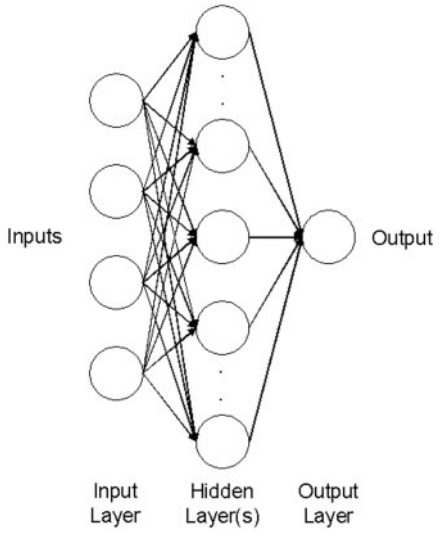

Figure 1: A typical artificial neural network representation consisting of an input layer, hidden layer and output layer. The representation may have one or more hidden layers in the case of a non-linear transform of the input nodes to output node, or may have no hidden layer whatsoever, directly connecting the input nodes to the output nodes, resulting in a linear model. During evolutionary optimization of the neural networks, not only can the weight assignment for each connection be optimized, but the entire topology of the neural network can be made free to vary, including number of layers, number of nodes per layer (including feature reduction of the input layer), mathematical transforms representing each node, the addition or deletion of connections, etc., providing incredible flexibility during model development.

Reinforcement learning, an approach that allows the machine or software to learn appropriate behavior based on feedback from a given environment, can be used to assist with both supervised and unsupervised approaches. However, each of these requires optimization of a model (in the case of supervised learning, quite often a classifier such as an ANN) in light of a fitness function. A common strategy for this is for example in the case of supervised learning, a backpropagation method [7, 8] that uses gradient descent to minimize the average error (or squared error) between the ANN output and the actual target value. This approach guarantees convergence but only to a locally optimal solution and there are methods to avoid this pitfall (see below). Once sufficiently trained (or in some cases as part of the training process), the best ANN is assayed on held-out testing samples (a validation set) for a measure of true predictive accuracy. In the case where a validation set is used, an additional test set is

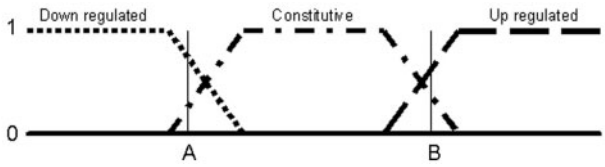

Figure 2: Mapping gene expression levels to three classes (down regulated, constitutive and up regulated) using a fuzzy transformation, despite gene expression (the x-axis) being a continuous variable. Point A in the graph represents the condition of 'not-likely to be up regulated', 'slightly constitutive' and 'mostly down regulated'. Similarly, Point B on the graph represents the condition of 'not-likely to be down regulated', 'slightly constitutive' and 'mostly up regulated'.

also used to assay model specificity and sensitivity in light of data that was not used at all during model development. For additional information the reader is directed to [9–12] for a broader review of ANNsand their application to bioinformatics.

FUZZY SYSTEMS

The concept of fuzzy sets was introduced in the 1960s by Zadeh [13, 14] as a generalization of conventional set theory. Fuzzy models attempt to quantify imprecision and uncertainty that is not easily captured by standard mathematical models. This notion fits very well with many pattern recognition problems where the classes to be separated do not have precisely defined membership criteria (Figure 2). For example, in bioinformatics problems, membership of a particular gene to a gene cluster may not be precisely defined (and may indeed be improperly defined based on an arbitrary threshold of expression required for classical approaches). Fuzzy sets can be used for clustering or classification [15] and, perhaps most importantly for bioinformatics research, used to manage uncertainty in rule-based representations, and rule conflict resolution [16] where the underlying logic of the representation is important to the end-user. For example, a microarray analysis system might have rules such as:

IF the expression of gene A is HIGH
THEN the predicted cancer prognosis is LOW
or
IF the expressions of gene A and gene B are both MOSTLY ON
THEN the decision of cancer is TRUE

For additional information the reader is directed to [17–23] for a broader review of fuzzy systems and

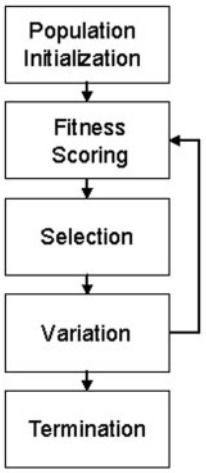

Figure 3: A standard flowchart for evolutionary algorithms. A population of solutions is constructed for the problem at hand, taking care to consider the appropriate representation for the problem. Each solution is scored with respect to a fitness function. In the case of model optimization this could be mean squared error between predicted and actual values. In the condition of other problems such as transcription factor binding site discovery, this could be a complex equation of many terms and associated weights representing important facets of the problem. Once all solutions have been scored, a method of selection is used to remove inadequate solutions from the population. The remaining solutions serve as 'parents' for the next generation of 'offspring' solutions. In order to generate offspring solutions, variation operators are applied (again developed specifically in light of the problem at hand). This process is iterated until a termination criteria is satisfied (typically either convergence of the population on a useful solution or lack of remaining CPU time).

their application to bioinformatics. Seminal papers in the field can be found in Bezdek and Pal [24].

EVOLUTIONARY COMPUTATION

Natural evolution can be viewed as a population-based optimization process. Simulation of this process on a computer results in a robust method for optimization (Figure 3). Historically there are several subdivisions of evolutionary computation (evolutionary programming (EP) [25], evolution strategies (ES) [26] and genetic algorithms (GA) [27–29]). Evolutionary programming and evolution strategies were conceived as abstractions of Darwinian evolution at the phenotypic level, whereas GA were conceived as abstractions of evolution at the genotypic level. More recent derivations and similar approaches include genetic

programming (GP) [30] which typically represents individuals as tree structures of mathematical expressions, particle swarm optimization (PSO) [31] wherein the populations of solutions is abstracted as a swarm of interacting particles with relative motion or velocity through the search space guided by the worth of each particle and the particles in a neighborhood, ant-colony optimization (ACO) [32] which abstracts the individual solutions at ants that migrate through the solution space based on the trails left by other ants in the population, and others such as differential evolution (DE) [33, 34] a simple and efficient method of global search. In DE, a population of vectors is initialized at random, and at each generation, new vectors are generated by randomly combining vectors in the population (mutation) and by mixing with other predetermined vectors in the space (recombination). The new vector is accepted if it generates a useful increase in fitness (selection).

These approaches are broadly similar in that they are all nature-inspired, maintain a population of solutions for the problem at hand, impose some set of (typically random) variations to those solutions, and use a method of selection to determine which solutions are to be removed from the current population, leaving the remainder to serve as 'parents' for the next generation of 'offspring' solutions. Evolutionary algorithms have been shown to possess asymptotic global convergence properties, and in some cases geometric rates of error convergence [35, 36] and thus they are very attractive methods for function optimization.

Evolutionary algorithms require the user to define a cost function so that alternative solutions can be scored appropriately, and for many real-world problems, defining a suitable cost function requires its own significant expertise and for some problems this can require its own research and development. In addition, the representation and variation operators should be developed specifically for the problem at hand (see notes below). For additional material the reader is directed to [37] for a broader review of applications of evolutionary computation in bioinformatics. Seminal papers in the field can be found in [38].

Combinations of CI approaches

Evolutionary computation cannot only be applied as an optimization process to identify useful solutions from a solution space (i.e. searching the space of possible docking conformations for a small molecule of best fit to a protein pocket), but it can be used to optimize neural networks or other pattern recognition representations. In this case, each solution in the population is itself a model representation (i.e. an ANN with a different weight assignment or topology) and can be scored in light of predictive accuracy. Or evolutionary computation can be used to optimize fuzzy classifiers, or even optimize ANNs that use fuzzy inputs. In short, the combination of approaches presents incredible flexibility to the user. It is this flexibility that makes CI approaches so attractive, but at the same time requires the user to understand where particular representations are most useful, how to best describe variation operators in light of the chosen fitness function, and how to describe a fitness function that accurately reflects the problem to be solved. Broad expertise in CI approaches applied to a wide range of problem areas is a key to successful implementation. For further information on combining evolutionary optimization with other CI approaches, the reader is directed to [39, 40]. The remainder of this review will focus on two current application areas for these technologies in bioinformatics and highlight commercial software using the technology.

Problem representation

It is important to note that an appropriate representation of the problem can be critical to the success of any CI approach. Very often this requires experience in identifying the right representation for the problem at hand rather than attempting to fit one particular representation to all problems. For example, in the case where pattern recognition is required, ANNs are commonly applied. ANNs might be particularly well suited to application areas that do not require easy human interpretation. The resulting 'black box' approaches may be difficult to understand, especially when the number of inputs and/or connections is large. Thus, if the problem of classification requires no additional human interpretation, an ANN may be a perfectly reasonable representation. However, if the researcher is very interested in understanding how or why particular inputs may be related in model, then alternate approaches such as representing the problem as a set of rule bases, or a decision tree, or other logical representation may be more appropriate. If evolutionary algorithms are to be used to optimize these representations, the user should develop

representation and/or problem-dependent variation operators. This also requires skill in understanding the solution space and thinking critically about what sorts of operators will work, as well as attempting to understand the probabilities associated with each operator. Methods of self-adaptation can be used to tune the parameters of concern automatically concurrent with the evolutionary process: a 'meta-level' evolutionary algorithm. This works well especially when the problem itself is a dynamic process.

Feature selection

A critical issue with many biological datasets is the overwhelming number of possible features that could be used as input to a pattern recognition model. Standard statistical approaches can be used to reduce the number of features, but in many cases thousands of features may still remain. And it may also be the case that the researcher is most interested in identifying the key non-linear relationships between features that are useful in an output decision, and reducing the feature space using linear regression methods may not be well suited. In such cases, the researcher can use the evolutionary process to evolve the selection of features to be used as input to the model concurrent with the optimization of the model itself. For example, when using evolutionary computation to optimize an ANN, a population of ANNs is maintained, each with alternate weight assignment on the connections, number of connections, processing functions internal to nodes in the ANN, possible recurrence and even the number of hidden and/or input nodes. Everything about the ANN, including the weight assignment and topology is optimized simultaneously. This includes the input layer, which can be bounded by the user to reflect a reasonable range of inputs for the problem at hand, sampled at random (or with expertise) from the set of all possible features. Such a method rapidly identifies useful collections of features while simultaneously producing an optimized model. The following sections provide applications of CI methods to biological problems as examples to the reader.

Phylogenetic analysis

The four main approaches for phylogenetic reconstruction from character information include distance matrix, parsimony, invariants and maximum likelihood methods [41]. As the number of characters and/or sequences used in the model construction increase, the number of possible tree topologies increases as a factorial. Three main methods to search for the best topology from the set of possible trees include: (i) exhaustive methods, (ii) branch and bound methods and (iii) heuristic methods. Exhaustive methods are most useful only when the space of tree topologies is small, which is rarely the case for meaningful phylogenetic comparisons, especially in light of increased interest in genome comparisons and the speed at which they are being generated. Branch and bound methods exclude trees that do not meet specific criteria, reducing the search space to a more reasonable size. This increases the probability of identifying a useful solution, but places an upper bound on the number of character states that can be used. Heuristic searches are used to build tree topologies either by changing the order in which the trees are built or via branch swapping in combination with a scoring metric (e.g. parsimony). These approaches have all led to significant advances in our understanding of the history of life on Earth; however, as the number of characters/sequences has increased dramatically, algorithms that can rapidly explore large spaces of possible phylogenetic trees are required.

Evolutionary computation has been applied to the problem of phylogenetic reconstruction for over 10 years [42–44]. More recently, Congdon *et al*. [45–47] presented a method called 'Gaphyl' for phylogenetic reconstruction with evolutionary algorithms. Gaphyl can outperform Phylip [48] in terms of CPU time to discover equally parsimonious solutions for datasets with large numbers of species (63 species, 1588 nucleotides each) Other approaches using evolutionary algorithms for phylogenetic reconstruction [49–53] also include parallelization [52]. In this case, the decreased search time was roughly linear with respect to the number of processors. This is only one of many similar application areas that utilize CI methods for optimization and discovery.

Transcription factor binding site identification

Computational assistance with genome annotation has recently become an important concern, especially in light of rapidly decreasing costs for human genome sequencing [54]. Identification or prediction of key regulatory features can be accomplished through homology searching algorithms such as BLAST and PSI-BLAST [55], however, this works well only in the cases where sequence similarity is preserved in evolution. Small regions such as

intron/exon boundaries, transcription start sites, even mature non-coding RNAs may either not contain sufficient length to be useful in BLAST or may contain sufficient variability even between closely related species or between tissues. While BLAST is an important and fast tool for similarity searching, in many cases the community requires algorithms that are specifically trained to identify novel functional elements with few known examples, where the novel elements may be highly variable, and yet where these elements might play key roles in cellular regulation.

The identification of *cis*-regulatory features such as enhancers and promoters represents one area of critical importance to genome annotation. Computational identification and experimental validation of transcription factor binding sites upstream of genes known to be co-expressed has resulted in databases of transcription factor binding sites such as TRANSFAC and COMPEL. Unfortunately, experimental validation of these TFBSs is still costly, and TFBSs may be located kilobases in either direction from the promoter, and may even exist within introns, making their discovery and analysis more difficult. Very often these binding sites are on the order of 12 nucleotides or less, with a well-conserved 'core' of roughly 4 nucleotides. Identification of novel binding sites therefore, requires inferring statistical knowledge from these databases rather than BLAST, as BLAST will be unable to use such small conserved regions for meaningful *e*-values.

Several researchers have made use of CI approaches for TFBS identification. These approaches can generally accept motifs of $n > 7$, are not organism specific, do not require exhaustive calculation or enumeration, and can report putative TFBSs in terms of either a nucleotide likelihood matrix or n-mer alignment. ANNs [56–58] and evolutionary computation have been applied in this regard, either on their own or combined for the optimization of pattern recognition models for TFBS discovery [59–63] or through combinations of evolutionary optimization approaches such as a combination of GA and PSO. Researchers have also used fuzzy k-means clustering to identify TFBSs [64].

When using evolutionary computation for this purpose, researchers generally must make several assumptions including either searching for a fixed or variable length TFBS, a proper calculation of similarity, compositional complexity and other metrics that might be included as part of a fitness function, establishing reasonable weights of importance associated with each of the fitness function parameters, and generating specialized variation operators that can search the solution space most effectively. With these approaches it is possible to examine huge possible search spaces at only a fraction of the time, and the inherently parallel nature of the evolutionary process is perfectly suited to parallelization on clusters of PCs.

Commercial software

CI approaches are becoming increasingly incorporated into commercialized software for bio and chemoinformatics. This is especially true in the area of drug discovery, where evolutionary algorithms have been used since the early 1990s and are included with several docking algorithms such as AutoDock [65], pso@autodock [66], GOLD [67] and MolDock [68, 69]. In addition, specialized companies such as Natural Selection, Inc. (San Diego, CA) offer their own set of CI tools (e.g. Connect® tools) and a history of expertise for problem solving in bioinformatics, including pattern recognition for novel microRNAs and biomarkers for cancer diagnosis.

Other application areas

CI is being used in a wide variety of other bioinformatics problem areas including microarray analysis [70–77] (mainly for feature selection and model development), modeling gene regulatory networks [78–81], RNA structure comparison and prediction [82–91], quantitative structure-activity/property relationships [92–97], sequence alignment [98–103], drug dissolution [104], spectroscopy [105–107], single nucleotide polymorphisms [108–110], CpG island methylation [111] and many other areas.

CONCLUSIONS AND PERSPECTIVES

Biological systems are inherently non-linear and dynamic. Datasets resulting from the analysis of biological systems typically include additional noise (in light of the experimental conditions and/or methods used to generate the data). The proper interpretation of such systems demands interpretive methods that do not rely strictly on linearity and yet

can provide reasonable solutions to the researcher in fast time. CI represents a burgeoning field in computer science that has broad, largely underappreciated, utility in biochemistry and medicine. However, in recent years there has been a dramatic increase in the use of these approaches, broadly, over many disciplines of biomedicine and biochemistry. Such methods can be complementary to previous approaches (such as local search or dynamic programming), and can be used to search very large solution spaces efficiently. The breadth of recent successful application makes it all the more apparent that the need for these methods will continue to grow in the near future.

Some of these approaches are considered 'black box' models, where it is difficult for the researcher to understand the underlying logic of the optimized pattern recognition model. For some problems, this is of little importance. However, there are problems where an understanding of the logic is of great importance. In these cases, more appropriate model representations can be developed that optimize the logical transform in a manner that is more easily understood by the end-user. Such approaches are very useful for automated feature reduction and pattern recognition.

Far too often, researchers fail to identify the right method or computational approach for the right problem and impose the use of one favored method over all problems. In reality, each problem requires its own model representation in light of input data types (e.g. categorical or numerical values, fuzzy terms or discrete variables) and end-user constraints and interpretation. The No Free Lunch theorem [112] suggests that there is likely not to be one best representation or model optimization method for all problems. Thus the informed bioinformatician will do well to explore the potential of CI approaches in providing a unique and powerful versatility to overcome many hurdles, simultaneously, with model development and optimization in achieving the right method for the right problem.

Key Points

- CI paradigms offer a unique and underappreciated advantage to challenging, non-linear, dynamic problems in bioinformatics.
- Hybridization of methods is possible and very useful for some problems.
- These approaches can be used not only for the discovery of solutions in large search spaces but the optimization of pattern recognition models themselves in light of sufficient data.

References

1. Reichhardt T. It's sink or swim as a tidal wave of data approaches. *Nature* 1999;**399**:517–20.
2. Fogel GB, Corne DW, Pan Y (eds). *Computational Intelligence in Bioinformatics*. Piscataway, NJ: Wiley-IEEE Press, 2008.
3. Seiffert U, Jain LC, Schweizer P (eds). *Bioinformatics Using Computational Intelligence Paradigms*. Heidelberg, Germany: Springer Verlag, 2005.
4. McCulloch WS, Pitts W. A logical calculus of ideas immanent in nervous activity. *Bull Math Biophys* 1943: **5**:115–33.
5. Rosenblatt F. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Washington, DC: Spartan Books, 1962.
6. Haykin S. *Neural Networks: A Comprehensive Foundation*. Upper Saddle River, NJ: Prentice Hall, 1998.
7. Werbos P. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. Harvard: Doctoral dissertation, 1974.
8. Werbos P. *The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting*. New York, NY: Wiley-Interscience, 1994.
9. Wu CH, McLarty JW. *Neural Networks and Genome Identification*. New York, NY: Elsevier, 2000.
10. Baldi P, Brunak S. *Bioinformatics: The Machine Learning Approach*. New York, NY: MIT Press, 1998.
11. Wood MJ, Hirst JD. Recent applications of neural networks in bioinformatics. In: Apolloni B, Marinaro M, Tagliaferri R (eds). *Biological and Artificial Intelligence Environments*. New York, NY: Springer, 2005.
12. Fiesler E, Beale R (eds). *Handbook of Neural Computation*. Bristol, UK: Oxford University Press, 1997.
13. Zadeh LA. Fuzzy sets. *Information and Control* 1965;**8**:338–53.
14. Zadeh LA. Fuzzy algorithms. *Information and Control* 1968; **12**:94–102.
15. Bezdek JC, Castelaz PF. Prototype classification and feature selection with fuzzy sets. *IEEE Trans Sys Man Cybern* 1977; **7**:87–92.
16. Keller JM, Tahani H. Implementation of conjunctive and disjunctive fuzzy logic rules with neural networks. *Int J Approx Reason* 1992;**6**:221–40.
17. Cox E. *The Fuzzy Systems Handbook: A Practitioner's Guide to Building, Using, Maintaining Fuzzy Systems*. San Diego, CA: AP Professional, 1994.
18. Zimmerman H. *Fuzzy Set Theory and its Applications*. Hingham, MA: Kluwer Academic, 2001.
19. Torres A, Nieto JJ. Fuzzy logic in medicine and bioinformatics. *J Biomed Biotechnol* 2006;**2**:91908.
20. Mordeson JN, Malik DS, Cheng S-C. *Fuzzy Mathematics in Medicine*. Physica, 2000.
21. Szczepaniak PS, Lisoba PJG, Kacprzyk J. *Fuzzy Systems in Medicine*. Physica, 2000.
22. Dong X, Bondugula R, Popescu M, *et al*. Bioinformatics and fuzzy logic. *2006 IEEE Int Conf Fuzzy Syst* 2006, 817–24.
23. Ruspini EH, Bonissone PP, Pedrycz W (eds). *Handbook of Fuzzy Computation*. Bristol, UK: Oxford University Press, 1998.

24. Bezdek JC, Pal SK (eds). *Fuzzy Models for Pattern Recognition: Methods that Search for Structures in Data*. Piscataway, NJ: IEEE Press, 1992.
25. Fogel LJ, Owens A, Walsh MJ. *Artificial Intelligence Through Simulated Evolution*. New York, NY: Wiley, 1966.
26. Rechenberg I. *Evolutionsstrategie: Optimerung technischer Systeme nach Prinzipien der biologischen Evolution*. Stuttgart, Germany: Fromman-Holzboog, 1973.
27. Bremmerman HJ. Optimization through evolution and recombination. In: Yovits MC, Jacobi GT, Goldstein GD (eds). *Self-Organizing Systems*. Washington DC: Spartan Press, 1962.
28. Holland JH. *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: University of Michigan Press, 1975.
29. Michalewicz Z. *Genetic Algorithms + Data Structures = Evolution Programs*. 3rd edn. Berlin, Germany: Springer, 1996.
30. Koza J. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: MIT Press, 1992.
31. Eberhart RC, Shi Y, Kennedy J. *Swarm Intelligence*. San Francisco, CA: Morgan Kaufmann, 2001.
32. Dorigo M, Gambardella LM. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Trans Evol Comput* 1997;**1**:53–66.
33. Storn R, Price K. *Differential Evolution: A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces*. Technical Report TR-95-012, ICSI, March 1995.
34. Storn R. System design by constraint adaptation and differential evolution. *IEEE Trans Evol Comput* 1999;**3**:22–34.
35. Bäck T, Fogel DB, Michalewicz Z (eds). *Handbook on Evolutionary Computation*. Bristol, UK: Oxford University Press, 1997.
36. Fogel DB. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. 3rd edn. Piscataway, NJ: IEEE Press, 2005.
37. Fogel GB, Corne DW (eds). *Evolutionary Computation in Bioinformatics*. San Francisco, CA: Morgan Kaufmann, 2002.
38. Fogel DB (ed). *Evolutionary Computation: The Fossil Record*. Piscataway, NJ: IEEE Press, 1998.
39. Fogel DB, Fogel LJ, Porto VW. Evolving neural networks. *Biol Cybern* 1990;**63**:487–93.
40. Yao X. Evolving artificial neural networks. *Proc IEEE* 1999; **87**:1423–47.
41. Swofford DL, Olsen GJ. Phylogeny reconstruction. In: Hillis DM, Moritz C (eds). *Molecular Systematics*. Sunderland, MA: Sinauer Associates, 1990, 411–501.
42. Matsuda H. Construction of phylogenetic trees from amino acid sequences using a genetic algorithm. In: Hagiya M, Suyama A, Takagi T, *et al*., (eds). *Proceedings of Genome Informatics Workshop*, Vol. 6. Tokyo, Japan: Universal Academy Press, 1995, 19–28.
43. Matsuda H. Protein phylogenetic inference using maximum likelihood with a genetic algorithm. *Pac Symp Biocomput* 1996, 512–23.
44. Lewis PO. A genetic algorithm for maximum-likelihood phylogeny inference using nucleotide sequence data. *Mol Biol Evol* 1998;**15**:277–83.
45. Congdon CB, Greenfest EF. Gaphyl: a genetic algorithm approach to cladistics. In: Freitas AA, Hart W, Krasnogor N, *et al*. (eds). *Data Mining with Evolutionary Algorithms*. Heidelberg, Germany: Springer-Verlag, 2000, 85–8.
46. Congdon CB. Gaphyl: a genetic algorithms approach to cladistics. In: DeRaedt L, Siebes A (eds). *Principles of Data Mining and Knowledge Discovery*. Berlin: Springer-Verlag, 2001, 67–78.
47. Congdon, CB, Septor, KJ. Phylogenetic trees using evolutionary search: initial progress in extending gaphyl to work with genetic data. In: *Proceeding of the 2003 IEEE Congress on Evolutionary Computation*. Piscataway NJ: IEEE Press, 2003, 320–6.
48. Felsenstein J. Phylip source code and documentation. 1995. http://evolution.genetics.washington.edu/phylip.html.
49. Catanzaro D, Pesenti R, Milinkovitch MC. An ant colony optimization algorithm for phylogenetic estimation under the minimum evolution principle. *BMC Evol Biol* 2007; **15**:228.
50. Perretto M, Lopes HS. Reconstruction of phylogenetic trees using the ant colony optimization paradigm. *Genet Mol Res* 2005;**4**:581–9.
51. Qin L, Chen Y, Pan Y, *et al*. A novel approach to phylogenetic tree construction using stochastic optimization and clustering. *BMC Bioinformatics* 2006;**12**:S24.
52. Lemmon AR, Milinkovitch MC. The metapopulation genetic algorithm: an efficient solution for the problem of large phylogeny estimation. *Proc Natl Acad Sci USA* 2002; **99**:10516–21.
53. Brauer MJ, Holder MT, Dries LA, *et al*. Genetic algorithms and parallel processing in maximum-likelihood phylogeny inference. *Mol Biol Evol* 2002;**10**:1717–26.
54. Bentley DR. Whole-genome re-sequencing. *Curr Opin Genet Dev* 2006;**16**:545–52.
55. Lipman DJ. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res* 1997;**25**:3389–402.
56. Workman CT, Stormo GD. ANN-Spec: a method for discovering transcription factor binding sites with improved specificity. *Pac Symp Biocomput* 2000, 467–78.
57. Knudsen S. Promoter 2.0: for the recognition of PolII promoter sequences. *Bioinformatics* 1999;**15**:356–61.
58. Ho LS, Rajapakse JC. Splice site detection with higher-order markov model implemented on a neural network. *Genome Inform* 2003;**14**:64–72.
59. Fogel GB, Weekes DG, Varga G, *et al*. Discovery of sequence motifs related to coexpression of genes using evolutionary computation. *Nucleic Acids Res* 2004; **32**:3826–35.
60. Congdon CB, Fizer CW, Smith NW, *et al*. Preliminary results for GAMI: a genetic algorithms approach to motif inference. In: *Proceedings of the 2005 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*. Piscataway, NJ: IEEE Press, 2005.
61. Congdon CB, Aman J, Nava G, *et al*. An evaluation of information content as a metric for the inference of putative conserved noncoding regions in DNA sequences using a genetic algorithms approach. *IEEE/ACM Trans Comput Biol Bioinform* 2008;**5**:1–14.
62. Lones M, Tyrell A. Regulatory motif discovery using a population clustering evolutionary algorithm. *IEEE/ACM Trans Comput Biol Bioinforms* 2007;**4**:403–14.

63. Aerts S, Van Loo P, Moreau Y, *et al*. A genetic algorithm for the detection of new cis-regulatory modules in sets of coregulated genes. *Bioinformatics* 2004;**20**:1974–76.
64. Gasch AP, Eisen MB. Exploring the conditional coregulation of yeast gene expression through fuzzy k-means clustering. *Genome Biol* 2002;**10**:3.
65. Morris GM, Goodsell DS, Halliday RS, *et al*. Automated docking using a Lamarckian genetic algorithm and an empirical binding free energy function. *J Comput Chem* 1998;**19**:1639–62.
66. Namasivayam V, Gunther R. Pso@autodock: a fast flexible molecular docking program based on swarm intelligence. *Chem Biol Drug Res* 2007;**70**:475–84.
67. Jones G, Willett P, Glen RC. Molecular recognition of receptor sites using a genetic algorithm with a description of desolvation. *J Mol Biol* 1995;**245**:43–53.
68. Thomsen R, Christensen MH. MolDock: a new technique for high-accuracy molecular docking. *J Med Chem* 2006; **49**:3315–21.
69. Thomsen R. Protein-ligand docking with evolutionary algorithms. In: Fogel GB, Corne DW, Pan Y (eds). *Computational Intelligence in Bioinformatics*. Piscataway, NJ: IEEE Press, 2008, 169–95.
70. Ooi CH, Tan P. Genetic algorithms applied to multi-class prediction for the analysis of gene expression data. *Bioinformatics* 2003;**19**:37–44.
71. Kim K-J, Cho S-B. Prediction of colon cancer using an evolutionary neural network. *Neurocomputing* 2004; **61**:361–79.
72. Khan J, Wei JS, Ringnér M, *et al*. Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nat Med* 2001; **7**:673–79.
73. Ma PCH, Chan KCC, Yao X, *et al*. An evolutionary clustering algorithm for gene expression microarray data analysis. *IEEE Trans Evol Comput* 2006;**10**:296–314.
74. Shen Q, Shi WM, Kong W. Hybrid particle swarm optimization and tabu search approach for selecting genes for tumor classification using gene expression data. *Comput Biol Chem* 2008;**32**:52–9.
75. Xu R, Anagnostopoulous GC, Wunsch DC. Hybrid of neural classifier and swarm intelligence in multiclass cancer diagnosis with gene expression signatures. In: Fogel GB, Corne DW, Pan Y (eds). *Computational Intelligence in Bioinformatics*. Piscataway, NJ: IEEE Press, 2008, 3–20.
76. Chuang LY, Chang HW, Tu CJ, *et al*. Improved binary PSO for feature selection using gene expression data. *Comput Biol Chem* 2008;**32**:29–37.
77. Robbins KR, Zhang W, Bertrand JK, et al. The ant colony algorithm for feature selection in high-dimension gene expression data for disease classification. *Math Med Biol* 2008;Feb 22 [Epub ahead of print].
78. Xu R, Wunsch D, Frank R. Inference of genetic regulatory networks with recurrent neural network models using particle swarm optimization. *IEEE/ACM Trans Comput Biol Bioinform* 2007;**4**:681–92.
79. Xu R, Venayagamoorthy GK, Wunsch DC. Modeling of gene regulatory networks with hybrid differential evolution and particle swarm optimization. *Neural Netw* 2007;**20**:917–27.
80. Ho SY, Hsieh CH, Yu FC, *et al*. An intelligent two-stage evolutionary algorithm for dynamic pathway identification from gene expression profiles. *IEEE/ACM Trans Comput Biol Bioinform* 2007;**4**:648–60.
81. Ressom HW, Zhang Y, Xuan J, *et al*. Inferring network interactions using recurrent neural networks and swarm intelligence. *Conf Proc IEEE Eng Med Biol Soc* 2006; **1**:4241–4.
82. Fogel GB, Porto VW, Weekes DG, et al. Discovery of RNA structural elements using evolutionary computation. *Nucleic Acids Res* 2002;**30**:5310–17.
83. Lesnik EA, Fogel GB, Weekes D, *et al*. Identification of conserved regulatory RNA structures in prokaryotic metabolic pathway genes. *BioSystems* 2004;**80**:145–54.
84. Shapiro BA, Wu JC, Bengali D, *et al*. The massively parallel genetic algorithm for RNA folding: MIMD implementation and population variation. *Bioinformatics* 2001;**17**:137–48.
85. Chen J-H, Le S-Y, Maizel JV. Prediction of common secondary structures of RNAs: a genetic algorithm approach. *Nucleic Acids Res* 2000;**28**:991–9.
86. Wiese KC, Deschênes A, Hendriks A. RnaPredict – an evolutionary algorithm for RNA secondary structure prediction. *IEEE/ACM Trans Comp Biol Bioinf* 2008;**5**:25–41.
87. Wiese KC, Glen E. A permutation-based genetic algorithm for the RNA folding problem: a critical look at selection strategies, crossover operators, and representation issues. *BioSystems* 2003;**72**:29–41.
88. Wiese KC, Hendricks A. Comparison of P-RnaPredict and mfold-algorithms for RNA secondary structure prediction. *Bioinformatics* 2006;**22**:934–42.
89. Wiese KC, Deschênes A, Hendriks A. RNA secondary structure prediction employing evolutionary algorithms. In: Fogel G, Corne D, Pan Y (eds). *Computational Intelligence in Bioinformatics*. Piscataway, NJ: Wiley-IEEE Press, 2008, 197–223.
90. Wiese KC, Hendriks, A, Deschênes, A, Youssef BB. P-RnaPredict – a parallel evolutionary algorithm for RNA folding: effects of psudorandom number quality. *IEEE Trans NanoBiosci* 2005;**4**:219–27.
91. Wiese KC, Hendriks A, Deschênes A. Analysis of thermodynamic models and performance in RnaPredict – an evolutionary algorithm for RNA folding. In: *Proceedings of the 2006 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*. Piscataway, NJ: IEEE Press, 2006, 343–51.
92. Luke BT. Evolutionary programming applied to the development of quantitative structure-activity relationships and quantitative structure-property relationships. *J Chem Information Comput Sci* 1994;**34**:1279.
93. So S-S, Karplus M. Evolutionary optimization in quantitative structure-activity relationships: an application of genetic neural networks. *J Med Chem* 1996;**39**:1521–30.
94. Weekes D, Fogel GB. Evolutionary optimization, backpropagation, and data preparation issues in QSAR modeling of HIV inhibition by HEPT derivatives. *BioSystems* 2003; **72**:149–58.
95. Buyukbingol E, Sisman A, Akyildiz M, *et al*. Adaptive neuro-fuzzy inference system (ANFIS): a new approach t predictive modeling in QSAR applications: a study of neuro-fuzzy modeling of PCP-based NMDA receptor antagonists. *Bioorg Med Chem* 2007;**15**:4265–82.
96. Mwense M, Wang XZ, Buontempo FV, *et al*. QSAR approach for mixture toxicity prediction using independent

latent descriptors and fuzzy membership functions. *SAR QSAR Environ Res* 2006;**17**:53–73.

97. Hecht D, Fogel GB. High-throughput ligand screening via preclustering and evolved neural networks. *IEEE/ACM Trans Comput Biol Bioinformatics* 2007;**4**:476–84.
98. Chellapilla K, Fogel GB. Multiple sequence alignment using evolutionary programming. In: *Proceedings of the 1999 IEEE Congress on Evolutionary Computation*. Piscataway, NJ: IEEE Press, 1999, 452.
99. Thomsen R, Fogel GB, Krink T. A clustal alignment improver using evolutionary algorithms. In: *Proceedings of the 2002 IEEE Congress on Evolutionary Computation*. Piscataway, NJ: IEEE Press, 2002, 121–6.
100. Thomsen R, Fogel GB, Krink T. Improvement of clustal-derived sequence alignments with evolutionary algorithms. In: *Proceedings of the 2003 IEEE Congress on Evolutionary Computation*. Piscataway, NJ: IEEE Press, 2003, 312–9.
101. Notredame C. Recent progress in multiple sequence alignment: a survey. *Summary Pharmacogenomics* 2002;**3**: 131–44.
102. Notredame C. SAGA: sequence alignment by genetic algorithm. *Nucleic Acids Res* 1996;**24**:1515–24.
103. Nguyen HD, Yamamori K, Yoshihara I, *et al*. Aligning multiple protein sequences by parallel hybrid genetic algorithm. *Genome Inform* 2002;**13**:123–32.
104. Do DQ, Rowe RC, York P. Modelling drug dissolution from controlled release products using genetic programming. *Int J Pharm* 2008;**351**: 194–200.
105. Jarvis RM, Goodacre R. Genetic algorithm optimization for pre-processing and variable selection of spectroscopic data. *Bioinformatics* 2005;**21**:860–8.
106. Ressom HW, Varghese RS, Drake SK, *et al*. Peak selection from MALDI-TOF mass spectra using ant colony optimization. *Bioinformatics* 2007;**23**:619–26.
107. Kim B, Kwon MJ. Optimization of principal-component-analysis-applied in situ spectroscopy data using neural networks and genetic algorithms. *Appl Spectrosc* 2008; **61**:73–7.
108. Nunkesser R, Bernholt T, Schwender H, *et al*. Detecting high-order interactions of single nucleotide polymorphisms using genetic programming. *Bioinformatics* 2007;**23**:3280–8.
109. Xue D, Yin J, Tan M, *et al*. Prediction of function nonsynonymous single nucleotide polymorphisms in human G-protein-coupled receptors. *J Hum Genet* 2008;Feb 6 [Epub ahead of print].
110. Chang HW, Chuang LY, Ho CH, *et al*. Odds ratio-based genetic algorithms for generating SNP barcodes of genotypes to predict disease susceptibility. *OMICS* 2008; **12**:71–81.
111. Sjahputera O, Popescu M, Keller JM, et al. Fuzzy approaches for the analysis of CpG island methylation patterns. In: Fogel GB, Corne DW, Pan Y (eds). *Computational Intelligence in Bioinformatics*. Piscataway, NJ: IEEE Press, 2008, 141–65.
112. Wolpert DH, Macready WG. No free lunch theorems for optimization. *IEEE Trans Evol Comput* 1997;**1**:67–82.