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...
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. 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
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 ANNs and 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 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

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'.

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).
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].

Combination 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 under-appreciated, 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


96. Mwense M, Wang XZ, Buontempo FV, et al. QSAR approach for mixture toxicity prediction using independent


