What Connectionists should know about

Genetic Algorithms and Classifiers

Chris McConnell
Department of Computer Science

Carnegie Mellon University

Nov. 9, 1988

What is a genetic algorithm?

- Genetic algorithms are adaptive algorithms that are motivated by looking at the mechanisms of biological evolution.
- They are characterized by a genotype, a set of idealized genetic operators and a fitness function for evaluating genotypes. (Possibly by translating to a phenotype.)
- The goal is to solve problems by specifying what is to be done, rather than how to do it while being able to exploit the best solution found so far.
- They have been applied in a wide variety of domains including ecomomics, control theory, game theory, AI, psychology and political science.

Example structures and evaluation functions

- **Genetics:** Chromosomes, fitness.
- **Economics:** Activites, utility.
- **Control:** Policies, error functions.
- **Game theory:** Strategies, payoff.
- **AI:** Programs, comparative performance.

Obstacles to adaptation

- There is a very large set of potentially interesting structures.
- The structures are complicated so that it is difficult to determine which substructures or components are responsible for good performance.
- The performance measure is a complicated function with many interdependent parameters. (High dimensionality, non-linearity, local optima, discontinuity.)
- The performance measure varies over space and time.
- There is a lot of relevant non-payoff information.

History

- Early attempts in the 50's and 60's made use of reproduction with emphasis and mutation and did not work very well.
- Mutation alone boils down to random search. Reproduction with emphasis and mutation works better, but is still basically an enumerative scheme.
- Theoretical analysis by Holland and others in the 70's indicates the importance of crossover and the roles of other genetic operators.
- Through the late 70's and 80's a multitude of theoretical and practical results.

Schemata

- A combination of symbols from some alphabet Σ and a wildcard symbol. With an alphabet of m symbols, and strings of length l, there are $(m+1)^l$ different schemata. Each schema is a hyperplane of dimension l or less.
- Each string over an alphabet is an instance of 2^l different schemata. String 101 is an instance of the schemata 101, 10*, 1*1, *01, 1**, *0*, **1, and ***.
- A string with many locations of few possibilities at each location will sample more schemata than a string with few locations and many possibilities.

Allocation of trials

- Each instance of a schema samples how well the explicit features found in that schema work together.
- We would like to generate different instances of a schema in proportion to its relative fitness. That way we have the best chance of finding the best combinations while still exploring other alternatives.
- A naive approach would explicitly manipulate schemas to give this distribution. Luckily this is not neccessary.

Genetic operators provide variation

- **Crossover:** Generates new samples of existing schemata and samples of new schemata.
- **Mutation:** A background operator that prevents a population from losing too much diversity.
- **Inversion:** Increases linkage between coadapted alleles so that they are less likely to be disrupted by crossover.
- **Dominance:** Mutation rate can be the square root of its value without dominance for the same diversity. It improves the long term memory of an individual, so that it can more rapidly adapt to a changing environment.
- **Extension:** Provides additional information carrying capacity.

Implicit Parallelism

- A population of M individuals where each generation replaces n members of the population in accord with their fitness will sample at least M^3 schemata every M/n generations.
- Above average, short, low-order schemata are given exponentially increasing numbers of trials in each generation.
- The population serves as a compact database of the relative rankings of many more schemata than the size of the population.

Fundamental theorem of GAs

- The fundamental theorem of GA under fitness proportinate reproduction, simple crossover and mutation.

H: a schema
$m(H,t)$: number of copies of H at time t
$f(H)$: fitness of H
$\overline{f}$: average fitness of all schemas
p_c: probability of crossover
p_m: probabililty of mutation
$\delta(H)$: distance between defining positions of H
$o(H)$: order of H = number of fixed positions
$m(H,t+1) \geq m(H,t)\frac{f(h)}{\overline{f}}\left[1 - p_c\frac{\delta(h)}{l-1} - p_m o(H)\right]$

Other biological ideas

- **Crowding:** Prevents the premature convergence of a GA by maintaining diversity. New individuals replace individuals that are most like them.
- **Speciation:** Is induced when benefactors are forced to share rewards from the environment. Will form subpopulations that are sized in relation to the size of the niche providing rewards.
- **Assortative mating:** Performance can be improved by having individuals mate with individuals that are in the same niche.

Genetic Algorithm Parameters

- Problem coding.
- Evaluation function.
- Reproductive strategy.
- Population size.
- Crossover rate.
- Mutation rate.
- Inversion rate.
- They can all be brought into the genotype and selected for.

Characteristics of Genetic Algorithms

- Each schema in the population increases or decreases independently of what is happening to ther schemata in the population.
- Each structure samples many different schemata. If a population is selected properly, it compactly ranks the relative merit of a much larger population of schemata.
- The distributed database allows GA's to handle problems with high dimensional, non-linear, discontinuous functions with local optima.
- Allows enfolding the specification of detectors, operators and parameters into the same representation, so that the space of search procedures can be searched at the same time.

When to use them

- The evaluation function has some combination of high dimensionality, non-linearity, discontinuity or local minima so that other techniques like hill climbing will not work well.
- There is a coding such that short, low-order, above average schemata will combine to form near optimal strings, or there is reason to believe that such a coding might exist.

Sample Applications

- They have been used for job-shop scheduling, gas pipeline control, network design, board layout, image registration, travelling salesman problem, bin packing, simulating cortical pattern recognition, and playing poker.

Motivations for Classifiers

- Conventional expert systems have impressive performance, but are too brittle.
- Maintain flexibility by having a framework that does not build in knowledge about the domain into the general mechanisms. Input from the environment and feedback on actions.
- Characterize a situation by the parallel activation of building block rules rather than a single specific rule.
- Organize rules into parsimonius default hierarchies with associations between them.
- Generate and verify models to improve the rate of learning.
- Use competition and confirmation so that the system evolves in a graceful manner.

Classifier Systems

- Classifier systems are sub-symbolic probabilistic parallel production systems.
- A global message list has the state of the system at each point. A relatively small size focuses the action of the rules.
- They have a collection of rules with strengths that are modified by a bucket brigade mechanism.
- New rules are generated through the action of a genetic algorithm. The strength of rules are used for identifying fitness.

Classifiers

- **Message:** String of elements from {0, 1}.
- **Classifier:** A production rule with conditions and actions expressed in terms of 0, 1, #.
- **Condition:** Constructed using bit strings from {0, 1, #} and AND or NOT, usually with just two terms. A message matches a condition if it matches all of the 0's or 1's in the condition.
- **Action:** Constructed using bit strings from {0, 1, #}. The action is to post a message where the # bits come from the message that matches the first condition. Sometimes extended to other operations.

Tags

- A tag is a subset of bits in a message that serve to link classifiers together.
- The other bits can pass on information that might be useful in later processing.
- They serve as a means of control and sequencing.
- Implement synchronic and diachronic pointers and default hierarchies.

Basic Classifier Cycle

- Add input messages to the message list.
- Match messages to classifier conditions.
- Calculate bids for matching classifiers, select winning classifiers probabilitically and post winning messages.
- Process output messages and return any rewards.
- Apply the bucket brigade to modify strength of classifiers.
- Use a genetic algorithm to generate new rules.

Bidding

- **Strength-S**: A measure of the past success of a rule.
- **Risk-R**: A number between 0.0 and 1.0 that indicates the amount of strength a rule will risk on one bid.
- **Specificity-P**: The number of fixed positions in the antecedent divided by the length of the string. Biases the chance of winning a bid.
- **Support-U**: The combined strength of messages that trigger the rule. Biases the chance of winning a bid.
- The bid is the amount of strength that must be given up by the rule if its is successful. The effective bid is used to bias the selection of bids so that more specific and better supported rules will win.

$bid = RSPU$

$$ebid = bid^b P^p U^u$$

Strength Revision

- A classifiers strength is decreased by its bid each time it successfully posts a message.
- A classifiers strength is increased by a reward that comes from the environment, or when a classifier that matches its message is fired.
- There is a tax associated with each cycle. This puts pressure on classifiers to maintain relevance.

Discovery

- Select pairs of classifiers probabilistically in proportion to strength.
- Apply crossover and mutation to create new classifiers.
- Replace the weakest classifiers probablistically in proportion to strength.

Implementation

- Suitable for implementation on SIMD machines. They have been implemented on the Connection Machine.
- Suitable for implementation on MIMD machines. Each node contains a population. The environments can be different, or the populations as a whole can compete and take over each others nodes.

Example Coding

- Performance in predicting the next letter in a four letter sequence with 200 classifiers.
- **Input:** ### ppp #### tt lllll
- **Output:** 10 ######## tt lllll
- **ppp:** 000 current, 001 previous, 010 two before, 011 three before.
- **tt:** 01 vowel, 10 consonant.
- **Reward:** +50 for correct letter. +10 for letter type. -10 for wrong prediction.

Example Result

```
Overall performance rate:   90.6
10, 20%:  50, 400
30, 40%: 400, 450
50, 60%: 450, 500
70, 80%: 550, 750
90,100%: 750, 800
At equilibrium (3750)
Mean:    99.9
SD:       0.39
High:   100.0 (3750)
Low:     98.0 (5000)
```

Connections to production systems

- Default hierarchies instead of partial matching.
- Parallel rule competition.
- Improves performance by modifying weights and generating new rules.
- Emphasis on having many rules apply at the same time.

Similarities to Connectionism

- Integration of multiple sources of evidence.
- Revision of strength parameters based on feedback.
- Emergence of higher order structures.
- Inhibitory effects are like those in sigma-pi units.
- Similar to competitive learning.

Differences from Connectionism

- Compact way to have many 'virtual' neurons since one rule can fire many times in the same time step.
- Each rule firing sums only two inputs and then multiplies the result and applies a global threshold.
- Emphasizes exploring the architecture of connections.
- Many different variant architectures like Edelman's NGS.
- Generation of new rules, ie. new connections rather than incremental modification. Can lead to more radical shifts in behavior.

Conclusions

- System that shares some of the same commitments as connectionism, but with a different consistent paradigm.

References

- Holland, "Adaptation in Natural and Artificial Systems"
- "Proceedings of the 1st, 2nd and 3rd International Conferenence on Genetic Algorithms"
- Holland, et. al "Induction"
- Davis, "Genetic Algorithms and Simulated Annealing"