

## Lecture 15 of 42

# Genetic and Evolutionary Computation 1 of 3: The Simple Genetic Algorithm

Friday, 16 February 2007

William H. Hsu

Department of Computing and Information Sciences, KSU

<http://www.kddresearch.org>

<http://www.cis.ksu.edu/~bhsu>

Readings:

Sections 9.1-9.4, Mitchell

Chapter 1, Sections 6.1-6.5, Goldberg

Section 3.3.4, Shavlik and Dietterich (Booker, Goldberg, Holland)



CIS 732: Machine Learning and Pattern Recognition

Kansas State University  
Department of Computing and Information Sciences

## Lecture Outline

- **Readings**
  - Sections 9.1-9.4, Mitchell
  - Suggested: Chapter 1, Sections 6.1-6.5, Goldberg
- **Paper Review: “Genetic Algorithms and Classifier Systems”, Booker *et al***
- **Evolutionary Computation**
  - Biological motivation: process of natural selection
  - Framework for search, optimization, and learning
- **Prototypical (Simple) Genetic Algorithm**
  - Components: selection, crossover, mutation
  - Representing hypotheses as individuals in GAs
- **An Example: GA-Based Inductive Learning (GABIL)**
- **GA Building Blocks (*aka* Schemas)**
- **Taking Stock (Course Review): Where We Are, Where We’re Going**



CIS 732: Machine Learning and Pattern Recognition

Kansas State University  
Department of Computing and Information Sciences

## Simple Genetic Algorithm (SGA)

- **Algorithm Simple-Genetic-Algorithm (Fitness, Fitness-Threshold,  $p$ ,  $r$ ,  $m$ )**  
 //  $p$ : population size;  $r$ : replacement rate (aka generation gap width),  $m$ : string size
  - $P \leftarrow p$  random hypotheses // initialize population
  - FOR each  $h$  in  $P$  DO  $f[h] \leftarrow \text{Fitness}(h)$  // evaluate Fitness: hypothesis  $\rightarrow R$
  - WHILE ( $\text{Max}(f) < \text{Fitness-Threshold}$ ) DO
    - 1. **Select**: Probabilistically select  $(1 - r)p$  members of  $P$  to add to  $P_S$ 

$$P(h_i) = \frac{f[h_i]}{\sum_{j=1}^p f[h_j]}$$
    - 2. **Crossover**:
      - Probabilistically select  $(r \cdot p)/2$  pairs of hypotheses from  $P$
      - FOR each pair  $\langle h_1, h_2 \rangle$  DO
 
$$P_S += \text{Crossover}(\langle h_1, h_2 \rangle) \quad // P_S[t+1] = P_S[t] + \langle \text{offspring}_1, \text{offspring}_2 \rangle$$
    - 3. **Mutate**: Invert a randomly selected bit in  $m \cdot p$  random members of  $P_S$
    - 4. **Update**:  $P \leftarrow P_S$
    - 5. **Evaluate**: FOR each  $h$  in  $P$  DO  $f[h] \leftarrow \text{Fitness}(h)$
  - RETURN the hypothesis  $h$  in  $P$  that has maximum fitness  $f[h]$



## GA-Based Inductive Learning (GABIL)

- **GABIL System [Dejong et al, 1993]**
  - Given: concept learning problem and examples
  - Learn: disjunctive set of propositional rules
  - Goal: results competitive with those for current decision tree learning algorithms (e.g., C4.5)
- **Fitness Function:  $\text{Fitness}(h) = (\text{Correct}(h))^2$**
- **Representation**
  - Rules: IF  $a_1 = T \wedge a_2 = F$  THEN  $c = T$ ; IF  $a_2 = T$  THEN  $c = F$
  - Bit string encoding:  $a_1 [10] \cdot a_2 [01] \cdot c [1] \cdot a_1 [11] \cdot a_2 [10] \cdot c [0] = 10011 11100$
- **Genetic Operators**
  - Want variable-length rule sets
  - Want only well-formed bit string hypotheses



## Crossover: Variable-Length Bit Strings

- **Basic Representation**

- Start with

	$a_1$	$a_2$	$c$		$a_1$	$a_2$	$c$
$h_1$	1[0	01	1		11	1]0	0
$h_2$	0[1	1]1	0		10	01	0

- Idea: allow crossover to produce variable-length offspring

- **Procedure**

- 1. Choose crossover points for  $h_1$ , e.g., after bits 1, 8
- 2. Now restrict crossover points in  $h_2$  to those that produce bitstrings with well-defined semantics, e.g., <1, 3>, <1, 8>, <6, 8>

- **Example**

- Suppose we choose <1, 3>

- Result

$h_3$	11	10	0				
$h_4$	00	01	1		11	11	0
					10	01	0



## GABIL Extensions

- **New Genetic Operators**

- Applied probabilistically
- 1. AddAlternative: generalize constraint on  $a_i$  by changing a 0 to a 1
- 2. DropCondition: generalize constraint on  $a_i$  by changing every 0 to a 1

- **New Field**

- Add fields to bit string to decide whether to allow the above operators

$a_1$	$a_2$	$c$		$a_1$	$a_2$	$c$	<u>AA</u>	<u>DC</u>
01	11	0		10	01	0	1	0

- So now the learning strategy also evolves!
- aka genetic wrapper



## GABIL Results

- **Classification Accuracy**
  - Compared to symbolic rule/tree learning methods
    - C4.5 [Quinlan, 1993]
    - ID5R
    - AQ14 [Michalski, 1986]
  - Performance of GABIL comparable
    - Average performance on a set of 12 synthetic problems: 92.1% test accuracy
    - Symbolic learning methods ranged from 91.2% to 96.6%
- **Effect of Generalization Operators**
  - Result above is for GABIL without AA and DC
  - Average test set accuracy on 12 synthetic problems with AA and DC: 95.2%



## Building Blocks (Schemas)

- **Problem**
  - How to characterize evolution of population in GA?
  - Goal
    - Identify basic building block of GAs
    - Describe *family of individuals*
- **Definition: Schema**
  - String containing 0, 1, \* (“don’t care”)
  - Typical schema: 10\*\*0\*
  - Instances of above schema: 101101, 100000, ...
- **Solution Approach**
  - Characterize population by number of instances representing each possible schema
  - $m(s, t) \equiv$  number of instances of schema  $s$  in population at time  $t$



## Selection and Building Blocks

- **Restricted Case: Selection Only**

- $\bar{f}(t)$   $\equiv$  average fitness of population at time  $t$
- $m(s, t)$   $\equiv$  number of instances of schema  $s$  in population at time  $t$
- $\hat{u}(s, t)$   $\equiv$  average fitness of instances of schema  $s$  at time  $t$

- **Quantities of Interest**

- Probability of selecting  $h$  in one selection step

$$P(h) = \frac{f(h)}{\sum_{i=1}^n f(h_i)}$$

- Probability of selecting an instance of  $s$  in one selection step

$$P(h \in s) = \sum_{h \in (s, p_t)} \frac{f(h)}{n \cdot \bar{f}(t)} = \frac{\hat{u}(s, t)}{n \cdot \bar{f}(t)} \cdot m(s, t)$$

- Expected number of instances of  $s$  after  $n$  selections

$$E[m(s, t+1)] = \frac{\hat{u}(s, t)}{\bar{f}(t)} \cdot m(s, t)$$



## Schema Theorem

- **Theorem**

$$E[m(s, t+1)] \geq \frac{\hat{u}(s, t)}{\bar{f}(t)} \cdot m(s, t) \cdot \left(1 - p_c \frac{d_s}{l-1}\right) \cdot (1 - p_m)^{\alpha(s)}$$

- $m(s, t)$   $\equiv$  number of instances of schema  $s$  in population at time  $t$
- $\bar{f}(t)$   $\equiv$  average fitness of population at time  $t$
- $\hat{u}(s, t)$   $\equiv$  average fitness of instances of schema  $s$  at time  $t$
- $p_c$   $\equiv$  probability of single point crossover operator
- $p_m$   $\equiv$  probability of mutation operator
- $l$   $\equiv$  length of individual bit strings
- $\alpha(s)$   $\equiv$  number of defined (non “\*”) bits in  $s$
- $d(s)$   $\equiv$  distance between rightmost, leftmost defined bits in  $s$

- **Intuitive Meaning**

- “The expected number of instances of a schema in the population tends toward its relative fitness”
- A fundamental theorem of GA analysis and design



## Terminology

- **Evolutionary Computation (EC): Models Based on Natural Selection**
- **Genetic Algorithm (GA) Concepts**
  - **Individual:** single entity of model (corresponds to hypothesis)
  - **Population:** collection of entities in competition for survival
  - **Generation:** single application of selection and crossover operations
  - **Schema aka building block:** descriptor of GA population (e.g.,  $10^{**0*}$ )
  - **Schema theorem:** *representation of schema proportional to its relative fitness*
- **Simple Genetic Algorithm (SGA) Steps**
  - **Selection**
    - **Proportionate reproduction (aka roulette wheel):**  $P(\text{individual}) \propto f(\text{individual})$
    - **Tournament:** let individuals compete in pairs or tuples; eliminate unfit ones
  - **Crossover**
    - **Single-point:**  $11101001000 \times 00001010101 \rightarrow \{ 11101010101, 00001001000 \}$
    - **Two-point:**  $11101001000 \times 00001010101 \rightarrow \{ 11001011000, 00101000101 \}$
    - **Uniform:**  $11101001000 \times 00001010101 \rightarrow \{ 10001000100, 01101011001 \}$
  - **Mutation:** single-point (“bit flip”), multi-point



## Summary Points

- **Evolutionary Computation**
  - **Motivation:** process of natural selection
    - Limited population; individuals compete for membership
    - Method for parallelizing and stochastic search
  - **Framework for problem solving:** *search, optimization, learning*
- **Prototypical (Simple) Genetic Algorithm (GA)**
  - **Steps**
    - **Selection:** reproduce individuals probabilistically, in proportion to fitness
    - **Crossover:** generate new individuals probabilistically, from pairs of “parents”
    - **Mutation:** modify structure of individual randomly
  - **How to represent hypotheses as individuals in GAs**
- **An Example: GA-Based Inductive Learning (GABIL)**
- **Schema Theorem: Propagation of Building Blocks**
- **Next Lecture: Genetic Programming, The Movie**

