Genetic Programming 1:
Introduction

Andrew Kusiak
Intelligent Systems Laboratory
2139 Seamans Center
The University of Iowa
Iowa City, Iowa 52242 – 1527

andrew-kusiak@uiowa.edu
http://www.icaen.uiowa.edu/~ankusiak

Some parts of this presentation is based on the material provided by Professor W. Pedrycz, ECE Department, University of Alberta, Canada)

November 28th, 2000

Introduction

• Learning
• Search Strategies
• Genetic Algorithms
• Genetic Programming

Reasoning Process: Typology

<table>
<thead>
<tr>
<th>Reasoning Process</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abduction</td>
<td>Data mining</td>
</tr>
<tr>
<td>Induction</td>
<td>Discovery of models</td>
</tr>
<tr>
<td>Deduction</td>
<td>Data fusion</td>
</tr>
<tr>
<td></td>
<td>Detection using models</td>
</tr>
</tbody>
</table>

Abduction: Create a model (hypothesis) for a specific data set to explain that data set

Induction: Extend model (hypothesis) for representative data sets to make a general assertion or explanation

Deduction: Apply a model to generate a hypothesis for detecting and classifying (explaining) the existence of a target

E. Waltz 1998

Reasoning Process Implementation

Types of Learning

• Supervised: Training examples with known inputs and outputs

• Unsupervised: No outputs are specified

• Reinforcement: Falls between the previous two types; A notion for the output quality is fed back to the learning algorithm.

Search Strategies in Learning Systems

• Blind (uninformed) search, e.g., tree search
  - breadth-first-search strategy
  - depth-first-search strategy

• Hill climbing,
  e.g., simulated annealing

• Beam search: limited number of solutions at each vertex
Genetic Algorithm: Basic Terms

POPULATION – a set of individuals which evolve according to rules of selection and genetic operators

FITNESS – a measure of ‘goodness’ assigned to each individual

SELECTION – a process of choosing high fitness individuals

GENETIC OPERATORS – used to perturb high fitness individuals

Genetic Operators

- Mutation

Before: 1 1 1 1 1 1 1
After: 1 1 1 0 1 1 1

Mutation usually happens with probability \( p_m \) for each gene

- Crossover

Crossover Types

- Single-point crossover
- Double-point crossover
- Uniform crossover
- Weighted (arithmetic) crossover
- Analytical crossover

Crossover Types

Single-Point Crossover

Uniform Crossover

The uniform crossover is works on individual locus rather than segments of a chromosome. The probability of selecting a locus for exchange is called the mixing rate. A mixing rate of 0.5 implies that each locus in the chromosome has an equal chance of being selected for replacement.

Double-Point Crossover

The double-point crossover cuts the chromosome into two segments at two different points and exchanges these segments between the two parents to create two new children.
**Weighted (Arithmetic) Crossover (a)**

*Weighted (arithmetic) crossover* modifies rather than exchanges genetic material. It works at the chromosome level rather than individual loci. Weight $w$ is selected before each crossover and then loci are randomly selected and exchanged.

- The expressions (1) and (2) represent the crossover process.

\[
\begin{align*}
    c_1 &= wp_1 + (1-w)p_2 \\
    c_2 &= wp_2 + (1-w)p_1
\end{align*}
\]

**Analytical Crossover**

*Analytical crossover* works at a chromosome level. It considers the best and the worst fitness of the two selected parents (see (3) - (4))

\[
\begin{align*}
    c_1 &= p_b + s(p_b - p_w) \\
    c_2 &= p_w
\end{align*}
\]

where:

- $s$ = scaling factor in $[0, 1]$
- $c_1$ = child 1
- $c_2$ = child 2
- $p_b$ = parent with the best fitness
- $p_w$ = parent with the worst fitness

**Genetic Algorithm: Process**

1. Initialize $P(0)$ to random individuals from the set \{1, 0\}
2. WHILE termination condition is false
   - Select individuals for re-production based on fitness
   - Those not selected die
   - Apply genetic operators to produce offspring
   - Produce $P(t+1)$ by adding offspring population to parent population
3. END

**Drawbacks**

- Some problems are difficult to represent as a binary string.
- Computation time can be long.
- Does not accurately reflect evolution.
- Relatively new concept.
Genetic Algorithm

Representation

Binary genome

0 1 0 0 1 1 1

Fixed size

Genetic Programming

Representation:

No constraints on the representation

Genetic Algorithm: Steps

Create random initial population

Evaluate population

Select individuals for variation

Vary

E.g., Crossover, mutation

Genetic Programming: Steps

Create random initial population

Evaluate population

Insert into population

Vary

Evaluate population

Select

GP vs GA

Genetic Programming: Population

Population: a set of individuals composed recursively from two sets:

- Set of $N_{fuc}$ functions
  - arithmetic operations (+, -, *, …)
  - mathematical functions (sin, cos, exp, …)
  - Boolean operations (AND, OR, NOT)
  - subroutines or other domain-specific functions
- Set of $N_{tem}$ terminals
  - variable atoms (inputs, sensor information, state variables)
  - constant atoms (numbers)
  - functions without arguments
Comment: Analogy to Machine Learning

- A feature (input) in the training data set becomes part of the terminal set in GP.

Thus, the features of the learning domain become some of the primitives used by GP to build program structures.

Genetic Programming: Population

The function set:
\[ F = \{ \text{AND, OR, NOT} \} \]

The terminal set:
\[ T = \{ d_0, d_1 \} \]

Genetic Programming: Population

Methods of generating initial population:
- **Full**: functions are chosen until the node reaches the maximum depth, therefore each branch of the tree has the same depth = max depth
- **Grow**: nodes are randomly selected from the function and terminal sets
- **Ramped half-and-half**

Generating Initial Population

Ramped half-and-half method: enhances population diversity

- Assume the maximum depth parameter = 6
- The population is divided equally among individuals to be initialized with trees having depths 2, 3, 4, 5, and 6
- For each depth group half of the trees are initialized with full method and half with grow method

Fitness Function

- Fitness function is a metric
- Fitness function is problem specific
- Fitness function provides feedback to the algorithm which individuals should reproduce
- Fitness function measures how well a program has learned to predict outputs from inputs
Fitness Function

\[ f_p = \sum_{i} |p_i - o_i| \]

\( p_i \) = (predicted) output from the GP program
\( o_i \) = (actual) output from the training set

Genetic Programming: Fitness

Types of fitness functions:
- raw fitness: not transformed
- standardized fitness: zero fitness value is always assigned to the fittest individual
- normalized fitness: all values are between 0 and 1

Genetic Programming: Selection

Methods of selection:
- fitness-proportional selection
- truncation selection
- ranking selection
- tournament selection

Fitness Proportional Selection

Probability of the individual \( i \) to be given a chance passing offspring to the next generation \( p_i \)

\[ p_i = f_i / \sum_{i} f_i \]

\( f_i \) = fitness of individual \( i \)

Truncation Selection

Known also as (\( \mu, \lambda \)) selection

- \( \mu \) parents are allowed to breed \( \lambda \) offspring, out of which the fittest \( \mu \) are used as parents for the next generation
- (\( \mu + \lambda \)) selection is also used, where offspring and parents participate in the selection

Remark

- For the selection scheme with \( m < \lambda \) the genetic algorithm becomes an evolutionary strategy (another type of evolutionary computation)
- The selection in EA is over-production selection, not mating selection as in GA. It is closer to what Darwin called “natural selection”
Ranking Selection
- Selection probability is assigned to an individual as a function of its rank
- Linear and exponential ranking functions are most often used

Tournament Selection
- Selection based on a competition within a subset of the population
- Tournament = the number of individuals is selected randomly

Genetic Programming: Genetic Operators
CROSSOVER
Two individuals

MUTATION: within one individual

Comment: Generalized Crossovers
- "Intelligent" crossover: Selection of a crossover point that is less destructive to the offspring
- Crossover operator that learns
- Heuristic guided crossover
- Context sensitive crossover
Genetic Programming: Parameters

- Major parameters
  - population size
  - number of generations
- Minor parameters
  - probability of crossover
  - selection of crossover points
  - size of S-expressions
  - probability of mutation
- Different ways of executing the runs
  - initial population
  - selection method
  - elitist strategy

Genetic Programming

- Representation of the problem
  - coding of individuals in the population
- Fitness function
  - evaluation of individuals in their capability to solve the problem

Genetic Programming: Fuzzy Rule-Based System Design

- Coding
  - each individual represents a single model, an array of floating-point numbers

References

- Koza, J. R. et al. (2003), Genetic Programming IV, Kluwer, Norwell, MA.