GECCO-2001 Tutorial on

Data Mining with
Evolutionary Algorithms

Alex A. Freitas

PUC-PR, Curitiba, Brazil
alex@ppgia.pucpr.br
http://www.ppgia.pucpr.br/~alex
Roadmap

• Introduction
  Classification and Clustering

• Genetic Algorithms for Data Mining
  Classification
  Clustering
  Attribute Selection

• Genetic Programming for Classification
  Constructive Induction

• Conclusions
Introduction

volume

intelligent decision

knowl.

information

data

value ($)

DM

DB
Information vs knowledge:
a simple example about a software house

Consulting low-level information in the DB:

How many videogames of type XYZ
were sold for customer ABC in 99/99/99?

Users - low managerial level
Now suppose we extract the following high-level knowledge from the database:

IF (Age < 18) AND (Job = student) THEN (Buy = videogame) (prob.=90%)  

We can ask: Which customers have a high probability of buying videogames?

Users - high managerial level
Desirable Properties of the discovered knowledge

* Accurate (as much as possible)

* Comprehensible by the human user

* Interesting (useful / new / surprising)
Data Mining Tasks

Types of problem to be solved:

Classification

Clustering

etc., etc.
Knowledge Discovery Paradigms

Type of method used to solve the task:

rule induction and decision trees

genetic algorithms

genetic programming

etc, etc;
Classification

Each example belongs to a predefined class

Each example consists of:
• a class (or goal) attribute
• a set of predicting attributes

The aim is to predict the class of an example, given its predicting attributes’ values
[Hand 97], [Michie et al. 94]
Data partitioning for the classification task.

<table>
<thead>
<tr>
<th>training data</th>
<th>test data</th>
</tr>
</thead>
<tbody>
<tr>
<td>(known class)</td>
<td>(unknown class)</td>
</tr>
<tr>
<td>. . .</td>
<td>goal</td>
</tr>
<tr>
<td>. . .</td>
<td>goal</td>
</tr>
<tr>
<td>c</td>
<td>. . .</td>
</tr>
<tr>
<td>b</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td></td>
</tr>
</tbody>
</table>
What is the next number in the sequence: [Bramer 96]

1, 4, 9, 16, ? (training data)
A possible answer is 20, based on the generator polynomium:

\((-5n^4 + 50n^3 - 151n^2 + 250n - 120) / 24\)

Both \(n^2\) and the complex polynomium are 100% consistent with the training data.
Classification example [Freitas & Lavington 98]:

Goal is to predict whether or not a customer will buy a product, given a customer’s Sex, Country and Age

<table>
<thead>
<tr>
<th>Sex</th>
<th>Country</th>
<th>Age</th>
<th>Buy? (goal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>France</td>
<td>25</td>
<td>yes</td>
</tr>
<tr>
<td>M</td>
<td>England</td>
<td>21</td>
<td>yes</td>
</tr>
<tr>
<td>F</td>
<td>France</td>
<td>23</td>
<td>yes</td>
</tr>
<tr>
<td>F</td>
<td>England</td>
<td>34</td>
<td>yes</td>
</tr>
<tr>
<td>F</td>
<td>France</td>
<td>30</td>
<td>no</td>
</tr>
<tr>
<td>M</td>
<td>Germany</td>
<td>21</td>
<td>no</td>
</tr>
<tr>
<td>M</td>
<td>Germany</td>
<td>20</td>
<td>no</td>
</tr>
<tr>
<td>F</td>
<td>Germany</td>
<td>18</td>
<td>no</td>
</tr>
<tr>
<td>F</td>
<td>France</td>
<td>34</td>
<td>no</td>
</tr>
<tr>
<td>M</td>
<td>France</td>
<td>55</td>
<td>no</td>
</tr>
</tbody>
</table>
Classification rules for the above data:

IF (Country = ‘Germany’) THEN (Buy = ‘no’)
IF (Country = ‘England’) THEN (Buy = ‘yes’)
IF (Country = ‘France’ & Age ≤ 25)  
   THEN (Buy = ‘yes’)
IF (Country = ‘France’ & Age > 25)  
   THEN (Buy = ‘no’)

Classification regarded as data separation

2 predicting attributes ($A_1$ and $A_2$)
2 classes (‘+’ and ‘-’)

<table>
<thead>
<tr>
<th>$A_2$</th>
<th>$A_2$</th>
<th>$A_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>++ - - - -</td>
<td>++ - - - -</td>
<td>++ - - - -</td>
</tr>
<tr>
<td>+ - + - -</td>
<td>+ - + - -</td>
<td>+ - + - -</td>
</tr>
<tr>
<td>+ - + - -</td>
<td>+ - + - -</td>
<td>+ - + - -</td>
</tr>
<tr>
<td>+ - + - -</td>
<td>+ - + - -</td>
<td>+ - + - -</td>
</tr>
<tr>
<td>+ - + - -</td>
<td>+ - + - -</td>
<td>+ - + - -</td>
</tr>
<tr>
<td>+ - + - -</td>
<td>+ - + - -</td>
<td>+ - + - -</td>
</tr>
<tr>
<td>+ - + - -</td>
<td>+ - + - -</td>
<td>+ - + - -</td>
</tr>
<tr>
<td>+ - + - -</td>
<td>+ - + - -</td>
<td>+ - + - -</td>
</tr>
</tbody>
</table>

original data separating by $A_1$ values separating by $A_2$ values

Which classifier will be more accurate on unseen test data?
The system must “invent” classes, by grouping similar examples
After clustering, we can apply classification methods
Criteria for finding good clusters

Minimize within-cluster distance

Maximize between-cluster distance

Favor a small number of clusters
Induction of Classification Rules

Basic idea: improve candidate rules, via generalization and specialization operations

Example of specialization:

IF (Country = ‘France’) THEN ...

\[\text{specialization}\]

IF (Country = ‘France’ & Age \leq 25) THEN ...

Generalization is the opposite operation
Decision Trees

internal nodes: predicting attributes
leaf nodes: predicted class

To classify a new example, push it down the tree, until reaching a leaf node

Country?

Germany no France Age yes England yes

≤ 25 > 25 no
Tree is built by selecting one-attribute-at-a-time (local search)

**LOOP**
Select attribute that best separates classes; Partition the set of examples in the current node according to selected attribute’s values; Repeat this process, recursively;
Drawback of Local search  
(select one-attribute-at-a-time)

Problems with attribute interaction

Exclusive OR (XOR) problem:

<table>
<thead>
<tr>
<th>A₁</th>
<th>A₂</th>
<th>XOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Looking only at one attribute gives us no useful information for predicting XOR
Example of Simpson’s Paradox

Renewal of magazine subscriptions, by month and subscription category
[Wagner 82], [Newson 91]

<table>
<thead>
<tr>
<th>subscription category</th>
<th>month</th>
<th>gift</th>
<th>previous</th>
<th>direct</th>
<th>sub.</th>
<th>catalog</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Jan</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>total</td>
<td>3,954</td>
<td>18,364</td>
<td>2,986</td>
<td>20,862</td>
<td>149</td>
<td>45,955</td>
</tr>
<tr>
<td></td>
<td>renew</td>
<td>2,918</td>
<td>14,488</td>
<td>1,783</td>
<td>4,343</td>
<td>13</td>
<td>23,545</td>
</tr>
<tr>
<td></td>
<td>rate</td>
<td>0.812</td>
<td>0.789</td>
<td>0.597</td>
<td>0.208</td>
<td>0.087</td>
<td>0.512</td>
</tr>
<tr>
<td></td>
<td>Feb</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>total</td>
<td>884</td>
<td>5,140</td>
<td>2,224</td>
<td>864</td>
<td>45</td>
<td>9,157</td>
</tr>
<tr>
<td></td>
<td>renew</td>
<td>704</td>
<td>3,907</td>
<td>1,134</td>
<td>122</td>
<td>2</td>
<td>5,869</td>
</tr>
<tr>
<td></td>
<td>rate</td>
<td>0.796</td>
<td>0.760</td>
<td>0.510</td>
<td>0.141</td>
<td>0.044</td>
<td>0.641</td>
</tr>
</tbody>
</table>
Genetic Algorithms for Data Mining

In data mining, GA can be used to:

(a) optimize parameters for other kinds of data mining algorithm

(b) discover knowledge by itself
Using GAs for parameter optimization

(a) finding a good set of attribute weights for nearest-neighbor algorithms

(b) finding a good set of weights and/or a good topology for a neural network

(c) selecting a good set of attributes to be given to another algorithm
GAs can also be used for rule discovery

Why should we consider using GAs rather than rule induction?

Both paradigms can discover high-level “IF-THEN” rules, but:
Most rule induction algorithms select one-attribute-at-a-time (*local* search)

GAs perform a *global* search that copes better with attribute interaction

Rules are evaluated as a whole by the fitness function

Genetic operators can modify many-attributes-at-a-time
Hybrid decision tree/GA
[Carvalho & Freitas 2000a, 2000b]

- decision tree used to classify “easy” examples
  (it exploits simplicity and efficiency of decision tree algorithms)

- GA used to classify “difficult” examples
  (it exploits GA ability to cope better with attribute interaction)
Identifying easy/difficult examples in a decision tree

A_1

A_2

“easy”

“difficult”
Basic ideas of GAs for rule discovery:

(a) Candidate rules are represented as individuals of a population

(b) Rule quality is computed by a fitness function

(c) Using task-specific knowledge
Classification with Genetic Algorithms

1) Each individual represents a rule set, i.e. an independent candidate solution

2) Each individual represents a single rule
   A set of individuals (or entire population) represents a candidate solution (rule set)
Individual Representation

In GABIL, an individual is a rule set, encoded as a bit string [DeJong et al. 93]

It uses $k$ bits for the $k$ values of a categorical attribute

If all $k$ bits of an attribute are set to 1 the attribute is not used by the rule
Goal attribute: Buy furniture (y/n)
Marital_status: Single/Married/Divorced
House: Own/Rented/University

<table>
<thead>
<tr>
<th>Marital_status</th>
<th>House</th>
<th>Buy?</th>
</tr>
</thead>
<tbody>
<tr>
<td>011</td>
<td>100</td>
<td>1</td>
</tr>
</tbody>
</table>

The string represents the rule
IF (Marital_status = M or D) and (House = O)
THEN (Buy furniture = y)
An individual is a variable-length string representing a set of fixed-length rules

\[
\begin{array}{c|c}
\text{rule 1} & \text{rule 2} \\
011 & 101 \\
100 & 110 \\
1 & 0 \\
\end{array}
\]

Mutation: traditional bit inversion

Crossover: corresponding crossover points in the two parents must semantically match
Example of crossover in GABIL

if a parent is “cut” in the second bit of a rule, the other parent must also be cut in the second bit of a rule, e.g.:

```
011 100 1 101 110 0
010 101 1 111 101 1 101 111 0
```
Individual representation

In GIL an individual is a set of rules, using a high-level encoding [Janikow 93]

<table>
<thead>
<tr>
<th>rule 1</th>
<th>rule 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A=1) and (B=2 or 3)</td>
<td>(C=2)</td>
</tr>
</tbody>
</table>

This kind of high-level encoding is more efficient for continuous attributes
Representing the predicted class

• included in the genome (and evolved)

• not included in the genome
  • all rules predict the same class
  • for a given rule antecedent, choose class maximizing rule quality

[Greene & Smith 93], [Noda et al. 99]
Task-specific genetic operators

generalizing/specializing mutation
[Janikow 93], [Liu & Kwok 2000]

Example of specializing mutation:
IF (Age ≤ 30) ... AND ... THEN ...

specialization

IF (Age ≤ 25) ... AND ... THEN ...
generalizing/specializing crossover
[Giordana et al 94], [Anglano et al. 1998]

generalizing crossover - logical OR
specializing crossover - logical AND

<table>
<thead>
<tr>
<th></th>
<th>generalizing</th>
<th>specializing</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 0 1</td>
<td>0 1 1 1</td>
<td>0 0 0 1</td>
</tr>
<tr>
<td>1 0 1 0</td>
<td>1 1 1 0</td>
<td>1 0 0 0</td>
</tr>
</tbody>
</table>
GIL has special genetic operators for handling: [Janikow 93]

- rule sets
- rules
- rule conditions

Operators can perform generalization, specialization or other operation
Generalization at the rule-set level in GIL:

Given two individuals, copy a rule from an individual to the other

Indiv$_1$: rule (or) rule

Indiv$_2$: rule (or) rule (or) rule

copy
Relevance-Based Rule Pruning

remove some conditions from a rule
(simplifies and generalizes the rule)

Basic idea: the less relevant a rule
condition is, the higher the probability
of removing that condition

This basic idea was used e.g. in:
[Liu & Kwok 2000], [Carvalho & Freitas 2000a]
Fitness Functions for Classification

• at least a measure of predictive accuracy

• possibly, also a measure of comprehensibility
  (the fewer the number of rules and
  rule conditions, the better)

E.g.: complexity = 2 #rules + #conditions
[Janikow 93]
• possibly, also a measure of rule interestingness [Noda et al. 1999]

motivation for interestingness measures:
IF (pregnant) THEN (sex = ´female´)
(accurate, comprehensible, *uninteresting*)
• Standard approach to combine accuracy, comprehensibility and interestingness: weighted fitness functions

• Problem: non-commensurate objectives

• Solution: multi-objective EAs
  [Bhattacharyya 2000a, 2000b], [Kim et al. 2000]
  [Emmanouilidis et al. 2000]
Parallel GA

- Parallelize fitness computation
- Fitness of many individuals can be computed in parallel (each processor evaluates a subset of individuals)
- Fitness of a single individual can be computed in parallel by p processors (data distributed across p processors)

See [Freitas & Lavington 98], [Flockhart & Radcliffe 95], [Giordana & Neri 95], [Anglano et al 97], [Araujo et al 99]
Clustering with GA

Simple representation:

- one gene per object to be clustered

- each gene = id of the cluster to which the object belongs

E.g. - 10 objects, 4 clusters:
    B C B B A D D C A D
Advantage: fixed-length individual

Disadvantages [Falkenauer 98]:

• high redundancy
e.g. (A B B A) and (B A A B)

• crossover and mutation problems

• is not scalable w.r.t. No. of objects
Graph-based clustering [Park & Song 98]

Each data instance is a node in a graph.
A cluster is a group of connected nodes.
Adjacency-based representation:
A vector of N integer elements, where 
N = No. of instances to be clustered.

i-th gene with value j means that the nodes i 
and j are connected by a link in the graph

E.g. the above clustering could be represented 
by:  \( I = < 2, 1, 5, 3, 3 > \)

Search space size: \((N-1)^N\)
Extending adjacency representation with task-specific knowledge:

The i-th gene can take on a value j only if j is one of the k nearest neighbours of the i-th instance

search space size: $k^N$
Advantages of this representation:

• does not require prespecified number of clusters

• does not require special genetic operators to produce valid offspring

• knowledge-based representation reduces the size of the search space
Disadvantages of this representation:

• not scalable for large data sets (genome length is N, where N is the No. of instances)

• redundancy - several genotypes correspond to the same phenotype
Clustering with hybrid GA/K-means
[Hall et al. 99]

GA optimizes location of cluster centroids

Individual representation:
a matrix of $c \times n$ cluster centers
($c =$ No. of clusters, $f =$ No. of features)

\[
M_{11} \ldots \ldots M_{1f} \quad \text{fitness based on}
\]
\[
\ldots \ldots \ldots \ldots \ldots \quad \text{distances from}
\]
\[
\ldots \ldots \ldots \ldots \ldots \quad \text{centroids}
\]
\[
M_{c1} \ldots \ldots M_{cf} \quad \text{(K-means)}
\]
Attribute selection with GA

Simple Individual Representation: One gene for each predicting attribute

Each gene can take on 2 values: 0 (attrib. ignored), or 1 (attrib. selected)

Example: 0 1 1 0 1 0 0 0
(attributes 2, 3, and 5 are selected)

See [Vafaie & DeJong 93], [Bala et al. 95]
Fitness depends on the performance of a data mining algorithm with the selected attributes (GA is a wrapper)

Fitness function can include a penalty for individuals with many attributes
More elaborated individual encoding:  
[Cherkauer & Shavlik 97]

Each gene can contain an attribute \((A_i)\) or be empty \((0)\)

Ex.: 0 0 A7 A2 0 A7 A5 0 0
(slected attributes: A_2, A_5, A_7)
Advantages of the elaborated encoding:

Repeated attributes reduces loss of genetic diversity

Individual’s length does not depend on the number of attributes being mined

Individual contains info about relative importance of selected attributes
GA for selecting attributes for an ensemble of classifiers
[Guerra-Salcedo & Whitley 99]

Each GA run selects an attribute subset

Each selected attribute subset is used to build one classifier of the ensemble
Genetic Programming for Classification

Standard approach:

terminal set: predicting attributes, random constant generator

function set: mathematical, comparison and logical operators

each individual is a “rule”
Classification: compare the output of the root node against a threshold

For an m class problem, run GP m times: each time we solve a 2-class problem

(one class is the “+” class and all other classes are grouped into a “-” class)
Genetic Programming for Classification

Main problems with standard approach

Closure property requires that all tree nodes return the same data type

Size and complexity of GP trees make them difficult to understand
Non-standard approaches for Classification with GP

‘Booleanize” attributes and function set

Constrained-synta x GP

Grammar-based GP
Booleanize attributes and use logical operators (and, or) in the function set
[Hu 98], [Eggermont et al 99], [Bojarczuk et al 99]

Property of closure is satisfied

\[
\begin{align*}
\text{OR} & \quad \text{AND} & \quad \text{Age} > 65 \\
\text{AND} & \quad \text{Age} < 18 & \quad \text{Sex} = \text{`M'}
\end{align*}
\]
Constrained-syntax GP

For each function used in the function set, specify the type of its arguments and the type of its result

Crossover and mutation are modified to respect the defined restrictions

See e.g. [Bhattacharyya et al. 98]
<table>
<thead>
<tr>
<th>Functions</th>
<th>datatype of input arguments</th>
<th>datatype of output</th>
</tr>
</thead>
<tbody>
<tr>
<td>+, -, *, /</td>
<td>(real, real)</td>
<td>real</td>
</tr>
<tr>
<td>≤, &gt;</td>
<td>(real, real)</td>
<td>boolean</td>
</tr>
<tr>
<td>AND, OR</td>
<td>(boolean, boolean)</td>
<td>boolean</td>
</tr>
<tr>
<td>IF</td>
<td>(boolean, boolean or real, boolean or real)</td>
<td>boolean</td>
</tr>
</tbody>
</table>

```
IF
  ≤
  9.6 Var1
OR
  ≤
  Var5 20
  >
  Var2 1.1
```
Grammar-based GP

Basic idea: use a grammar to define the format of rules [Wong & Leung 2000]

The placement of a symbol in the tree must be allowed by the grammar

To create an individual, a complete derivation is performed from the start symbol of the grammar
Ex.: predicting attributes: sex, age, x-ray
    goal attributes: disease_1, disease_2

Rule → Rule1 | Rule2
Rule1 → if Antec1 then Cons1
Rule2 → if Antec2 then Cons2
Antec1 → Sex_cond and Age_cond
Antec2 → Age_cond and X_ray_cond
Sex_cond → ∅ | ‘sex = M’ | ‘sex = F’
    .
    .
Cons1 → Disease1
Disease1 → ‘disease1=yes’ | ‘disease1=no’
    .
    .
Example: Rule

Rule1

if Antec1 then Cons1

Sex_cond and Age_cond “disease1=no”

“sex = F” ∅
Advantages of grammar-based GP:
• uses domain knowledge
• avoids the need for closed function set

Disadvantages of grammar-based GP:
• grammar is domain-specific
• reduces the autonomy of the algorithm to discover novel, surprising knowledge
Constructive Induction

Motivation - generate ‘higher-level’ attributes, such as:

‘Income > Expenditure?’

which can be used to generate rules such as:

IF (‘Income > Expenditure?’ = ‘yes’) ...
Note that the condition:
(“Income > Expenditure?” = “yes”)

corresponds to an “infinite” number
of conditions of the form:
(Income > value) AND (Expenditure < value)
Constructive induction with GP [Hu 98]

1st step: ‘booleanize’ all the attributes

E.g.: values of attribute Age can be divided into 2 groups: Age ≤ v, Age > v
(v is automatically chosen)

2nd step:
apply GP to construct new attributes
Each individual represents an attribute

Terminals: booleanized attributes
Functions: logical operators (and, not)

All terminal and function symbols return boolean values
(meets the closure requirement)
Conclusions

• Motivation for data mining with GA/GP: to cope with attribute interaction better than local, greedy rule induction methods

• Need for task-specific genetic operators
• GP’s representational power is more useful to construct new attributes and discover novel knowledge

• Fitness function should consider accuracy, comprehensibility, and interestigness – which suggests multi-objective algorithms
References


