ACISS 2009 Tutorial
Genetic Programming for Data Mining

Mengjie Zhang
School of Engineering and Computer Science
Victoria University of Wellington
New Zealand

Outline
- Data Mining
- Evolutionary Computing and Genetic Programming
- GP for Symbolic Regression
- GP for Classification
- Challenges and Issues
- Upcoming events

Data Mining and Knowledge Discovery
- KDD: the non-trivial process of identifying valid, novel, potentially useful and ultimately understandable patterns in data [Fayyad].
- DM: a step

Data Mining Tasks
- Classification (√)
- Regression (√)
- Prediction
- Time Series analysis
- Clustering
- Summarisation
- Association rules
- Sequence discovery

Evolutionary Computation
- Population based
- Multiple solutions
- Global search
- Many paradigms:
  - GAs, GP
  - ES, EP
  - PSO
  - DE, EDA, ...

EC Process
Genetic Programming

- GP inherits properties from EC techniques (GAs) and automatic programming
  - GP uses a similar evolutionary process to the general EC/GAs
  - bit strings chromosomes vs tree-like structures that can represent computer programs such as LISP (and C, Java)
  - fixed length representation vs Variable length
- Automatically learning a set of computer programs for a particular task is a dream of computer scientists
- GP is such a technique that helps achieve this goal

Programs as Tree Structures

\[(x - 1) - x^3\]
\[(- (- x 1) (* x (+ x x)))\]

GP Programs

- Terminal set: features/attributes from a task, constants (coefficients)
- Function set:
  - Standard functions: +, -, *, /, sin, log, exp, ...
  - Task/domain specific functions
- Sufficiency and closure
  - Selection of the functions and terminals is critical to success
  - Sufficient vs redundant

Program Generation

- For initialising a population or mutation.
- Maximum program depth: the maximum size permitted for a program,
- Program generation methods:
  - Full:
  - Grow, and
  - Ramped half-and-half:

Genetic Operators in GP

- Reproduction
- Crossover
- mutation

Fitness Cases and Fitness Function

- Fitness cases: instances, training/test
- Fitness is the measure of how well a program has learnt to predict the output from the input during simulated evolution
- The fitness of a program is calculated using the fitness function via program evaluation.
- The fitness function should be designed to give graded and continuous feedback
Fitness Function Examples
- Image matching: the number of matched pixels
- Robot learning obstacle avoidance: the number of wall hits for a robot
- Classification task: the number of correctly classified examples
- Prediction application: the deviation between prediction and reality
- GP-controlled agent in a betting game: the amount of money won
- Artificial life application: the amount of food found and eaten.

Selection
- Fitness selection determines which evolved program will be used by the genetic operators for evolution
  - Proportional selection
  - Tournament selection

Basic GP Algorithm
1. Initialise the population
2. Evaluate the individual programs in the current population. Assign a fitness to each program.
3. Until the new population is fully created, repeat the following:
   - Select programs in the current generation.
   - Perform genetic operators on the selected programs.
   - Insert the result of the genetic operations into the new generation.
4. If the termination criterion is not fulfilled, repeat steps 2-4 with the new generation.
5. Present the best individual in the population as the output.

Tackling a Problem with GP
If a GP package is available:
- Determine the terminal set, function set
- Determine the fitness function
- Determine the parameter values
  - Pop size, program size/depth, max generations, crossover/mutation/reproduction rates, etc.
- Determine the stopping criteria

Regression Analysis and Modelling
- In statistics, regression analysis examines the relation of a dependent variable (response variable) to specified independent variables (explanatory variables)
- The mathematical model of their relationship is the regression equation
- A regression equation contains estimates of one or more hypothesized regression parameters
- The estimates measure the relationship between the dependent variable and each of the independent variables

Statistical Regression Example
- Simple Linear Regression
  - Process:
    - Given data points
    - $y = a + bx + \epsilon$
    - $\epsilon$ is the error term
    - $\epsilon$ is normally distributed
    - Use some methods to estimate $a$ and $b$
Symbolic Regression

- Problems of statistical parameter regression
  - Need domain expertise to assume certain distribution of the given data, which is usually unknown in advance
  - Need statistical expertise to find an "appropriate" model, which is usually very hard

- Symbolic regression: the object to be found is a symbolic description of a model, not just a set of coefficients in a pre-specified model.
  - the model structure, with
  - the corresponding coefficients/parameters

GP for Symbolic Regression

- Objective: Find a program that produces the correct value of $x^2 - 2x^4 + x^6$ when given the value of $x$
- Terminal Set: X and a random number R in [-1.0, 1.0]
- Function Set: {+ , -, *, %}
- Fitness Cases: 50 random $x$ values in [-1.0, 1.0]
- Fitness Measure: Sum of the errors for the 50 cases
- Parameters: Population = 100. Generations = 51, ProgSize = 6, reproduction rate: 5%, crossover rate: 90%, mutation rate: 5%
- Success: The error for each of the 50 points is less than 0.01
- Termination criteria: satisfactory solutions found, or at generation 51.

Symbolic Regression Example

- One run gave:
  \[
  \text{One run gave:} \quad \left( \left( \left( \left( \left( x \times x \right) \times x \right) \times x \right) - 0.395493 \right) - 0.4665 \right) \ldots
  \]

- This example: one input variable ($x$), training set only
- Real world applications: usually multiple input variables, can have a separate test set, but use the same principle

Symbolic Regression Example

- One run gave:
  \[
  \left( \left( \left( \left( \left( x \times x \right) \times x \right) \times x \right) - 0.395493 \right) - 0.4665 \right) \ldots
  \]

- This example: one input variable ($x$), training set only
- Real world applications: usually multiple input variables, can have a separate test set, but use the same principle

GP for Regression Applications

- Economic prediction, e.g. stock market prediction, GDP prediction.
- Industrial prediction, e.g. prediction of containers handling capacity at a particular sea port, short-term, medium-term and long-term prediction of power load at a region
- Experiential formula modeling in Engineering, e.g. formulating the amount of gas emitted from Coal surface
- Time series projection, e.g. CPI projection for a country or a region
- Selection/Choice of Equipments, e.g. equipment choice for work platform in mine industry
- Fault diagnosis, e.g. find optimal strategy in fault isolation, fault analysis in combustion system for diesel engine
- Robot self-adaptive behaviour
- GIS systems, e.g. projection transformation
- Electronic circuit design (GP IV, Koza)
GP for Binary Classification

- Terminal set: features, constants
- Function set: standard + specific
- Fitness function: classification accuracy or error rate on the training example
- Program class translation rule: how to translate/convert the single program output to one of class labels

Tree-based GP for Classification

Example: ObjectTracking Task

Approach

- Use GP to track an object in low-quality webcam footage, at a real-time speed.
- Test the GP method on two object tracking problems of varying difficulty.

Training

- Specify target object position
- Evaluate tracker program at a set of training points around target producing refined estimates.
- Fitness of program = avg. distance from target

Tracking the left eye

- Tracks well, even when the face was quite blurry due to fast movement.
Tracking the head

- Tracks well, even when the face was quite blurry due to fast movement and when the head looks up.

GP for English Stress Detection

- English becomes more and more important as a communication tool in the world.
- Provide P2P training to ESL students is very expensive. Therefore, software is desirable.
- Correct rhythmic stress in ESL students' speech is a key point to make the speech sound like native. Therefore, to accurately detect rhythmic stress in spoken English becomes an important functionality in this kind of software.

Known stress classifiers

- Bayesian classifier
- Support vector machine classifier
- Decision tree classifier
- Neural networks classifier

The best accuracy is around 85%. It is not high enough for a commercial use.

Overview of the whole project

The Speech Analyser

- Text
- Sound
- Stress Classifier
- Stress Detector
- Match

Identified Stress or Rhythm Errors

The Stress Detector

- Sound (Phoneme Labelled)
- Feature Extraction & Normalisation
- Feature Vectors
- Stress Classifier
- Stress Pattern ... 1 –1 ...

Pedagogic
Component

Speech
Analyser

Analysis

Feedback

Speech

Dialogues

Learner
Classifier Learning Procedure

Experiments

- Data set: 703 vowels in 60 utterances of ten distinct sentences produced by 6 female speakers – 340 stressed and 363 unstressed
- Scaled feature values in the range [-1, 1] are also used.
- Three experiments are conducted on the three terminal sets respectively.
- 10 times 10-fold cross validation for training and testing
- Comparing with
  - DT -- C4.5
  - SVM -- LibSVM (with Radial Basis Function kernel and C = 1)
  - GP: Discipulus

Detection Accuracy (%)

<table>
<thead>
<tr>
<th>Terminal Set I (prosodic features):</th>
<th>SVM</th>
<th>DT</th>
<th>GP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unscaled</td>
<td>81.3</td>
<td>80.4</td>
<td>92.0</td>
</tr>
<tr>
<td>Scaled</td>
<td>91.6</td>
<td>80.6</td>
<td>93.2</td>
</tr>
<tr>
<td>Terminal Set II (vowel quality features):</td>
<td>SVM</td>
<td>DT</td>
<td>GP</td>
</tr>
<tr>
<td>Unscaled</td>
<td>85.4</td>
<td>79.7</td>
<td>91.1</td>
</tr>
<tr>
<td>Scaled</td>
<td>84.6</td>
<td>78.9</td>
<td>90.5</td>
</tr>
<tr>
<td>Terminal Set III (combination):</td>
<td>SVM</td>
<td>DT</td>
<td>GP</td>
</tr>
<tr>
<td>Unscaled</td>
<td>92.0</td>
<td>79.9</td>
<td>81.3</td>
</tr>
<tr>
<td>Scaled</td>
<td>92.0</td>
<td>80.1</td>
<td>82.0</td>
</tr>
</tbody>
</table>

GP for Multi-class Classification

- Classification map: Static method
- Boundaries are fixed
- These boundaries are predefined
- A class is determined from the fixed regions
- Classes are in a fixed order

Dynamic methods:

Linear GP for Multi-class Classification

\[ r[0] = 0.453 - cr[1]; \]
\[ r[1] = r[0] * 0.9; \]
\[ r[2] = 0.453 - cr[3]; \]
\[ r[3] = cr[4] - 0.8; \]
// instrs, src regs, dst regs, ops

We use multiple destination registers each corresponding to one class.

The winner-takes-all strategy is used for classification: the class represented by the register with the largest value is considered the class of the input object.

GP for Multi-class Classification

Program output results for all programs on all training patterns:

Class 1 | Class 2 | Class 3
---|---|---
Instructions, src regs, dst regs, ops

We use multiple destination registers each corresponding to one class.

The winner-takes-all strategy is used for classification: the class represented by the register with the largest value is considered the class of the input object.
Experiment Design

- Data sets: (Shape: 600 objects; digit15 and digit30: 1000)
- Terminal set: 8 features (shape), and 49 pixels (digits)
- Function set: /, x, +, -
- TGP length heuristic based on LGP
- Repeat 50 runs and mean/standard deviation of the results are reported

Classification Results

<table>
<thead>
<tr>
<th>Data set</th>
<th>Method</th>
<th>Training Accuracy % (µ ± σ)</th>
<th>Test Accuracy % (µ ± σ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>shape</td>
<td>LGP</td>
<td>100.00 ± 0.00</td>
<td>99.91 ± 0.17</td>
</tr>
<tr>
<td></td>
<td>TGP</td>
<td>85.04 ± 16.49</td>
<td>84.41 ± 17.17</td>
</tr>
<tr>
<td>digit15</td>
<td>LGP</td>
<td>68.62 ± 4.67</td>
<td>65.78 ± 5.20</td>
</tr>
<tr>
<td></td>
<td>TGP</td>
<td>52.60 ± 6.65</td>
<td>51.80 ± 6.85</td>
</tr>
<tr>
<td>digit30</td>
<td>LGP</td>
<td>55.22 ± 3.49</td>
<td>51.94 ± 4.26</td>
</tr>
<tr>
<td></td>
<td>TGP</td>
<td>41.15 ± 5.03</td>
<td>35.00 ± 6.17</td>
</tr>
</tbody>
</table>

Program Comprehensibility

// r[1] = r[1] / r[1];
// r[3] = cf[0] + cf[5];
// if (r[3] < 0.86539)
r[0] = 0.453 – cf[1];
r[1] = r[0] * 0.9;
if (cf[6] < cf[1])
r[2] = 0.453 – cf[3];

Major Challenges

- Program structures and representations
- Operators and search techniques
- High dimensional data
- Unbalanced data
- Conflict objectives
- Computation cost
- Understanding of evolved programs

Bibliography

Bibliography


Upcoming Conferences/Workshops

- Special Session on Evolutionary Computer Vision, CEC 2010: IEEE Congress on Evolutionary Computation
  - Organisers: Victor Ciesielski, Mario Koeppen, Mengjie Zhang
  - July 18-23, 2010, Barcelona
  - Paper Submission deadline: 31 Jan 2010
- Genetic and Evolutionary Computation Conference (GECCO 2010)
  - Time/Venue: Portland, 7-11 July 2010
  - Paper Submission deadline: 13 Jan 2010

Acknowledgements

- University Research Fund under the number of URF09 2399/85608/85808 at Victoria University of Wellington, New Zealand
- Marsden Fund council from the government funding (VUW0806), administrated by the Royal Society of New Zealand.
- Research students Jason Xie, Will Smart, Christo Fogelburg at Victoria University of Wellington, New Zealand
- ACISS 2009 organisers