New GP Techniques for Classification

Mengjie Zhang

mengjie@ecs.vuw.ac.nz
Outline

• Structure of standard GP
• Binary classification
• Multiclass classification
  – Class translation from the program output
  – Multiple outputs
  – Probability based GP
• Object detection — Fitness function
• Search techniques
  – gradient descent on constants
Standard GP for Binary Classification

\[
\text{if } \text{ProgOut} > 0 \text{ then } \text{Class1}; \\
\text{else } \text{Class2};
\]
Classification Map

Static Class Boundary Determination

- Boundaries are **fixed** at locations on the real number line of the program output
- These boundaries are **predefined**
- A class is determined from the **fixed regions**
- Classes are in a **fixed** order

![Diagram of Classification Map]

Genetic Program
Dynamic Class Translation

Centred Dynamic Class Boundary Determination

- Boundaries are **dynamically** determined based on the **centres** of different classes, each of which is calculated as the average output of all the programs for training examples of that class.

- Boundaries are set **halfway between adjacent centres**.

![Diagram showing program output results for all programs on all training patterns, with centres and boundaries indicated.](image-url)
Dynamic Class Translation

Slotted Dynamic Class Boundary Determination

- Real number line divided into 200 slots in [-25, 25]
- Slots are assigned the class labels during evolution
  - the class with the most programs that fall into the slot.

Program output results for all programs on all training patterns

<table>
<thead>
<tr>
<th>Slots</th>
<th>Class 1</th>
<th>Class 2</th>
<th>Class 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Values in slots</td>
<td>2 0 1 1 0 2 0 3 0 0 0 0</td>
<td>(Array[slot][class] += 1)</td>
<td></td>
</tr>
</tbody>
</table>
Multiple Outputs

Program Structure

- A genetic program can produce more than one output value, each for a class
- A voting strategy (WTA) is applied
- Program Structure:
Multiple Outputs

- Program Evaluation
Multiple Outputs

- Program Simulation
Probability GP

- Fitness function: replace classification accuracy with probability
- Fitness Measures: area vs distance

- Area Measure:
  \[ P(x) = \frac{\exp\left(\frac{-x^2}{2}\right)}{\sqrt{2\pi}} \]
  \[ A(x) = \sum_{i=0}^{\alpha} \alpha P(\alpha_i) \]
  \[ A(\mu, \sigma, x) = A\left(\frac{x - \mu}{\sigma}\right) \]
  \[ A_o = 1 - A(\mu_1, \sigma_1, m) - A(\mu_2, \sigma_2, m) \]
• Distance measure: weighted distribution distance \[ d = 2 \times \frac{|\mu_1 - \mu_2|}{\sigma_1 + \sigma_2} \]
• Standardised distribution distance measure \[ d_s = \frac{1}{1+d} \]
• Fitness function:

\[
fitness = \sum_{i=1}^{C_n^2} M_i
\]

• Classification with multiple programs:

\[
Prob_c = \prod_{i=1}^{l} P(\mu_{i,c}, \sigma_{i,c}, r_i)
\]

\[
P(\mu, \sigma, x) = \frac{\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)}{\sigma \sqrt{2\pi}}
\]
Object Detection — Fitness Function

- Fitness function is based on detection rate and false alarm rate:
  \[ \text{fitness}(DR, FAR) = W_d \times (1 - DR) + W_f \times FAR \]
- One problem: could not reflect the small improvement

- Second problem: programs are often very long
- A new fitness function:
  \[ fit = K_1 \cdot (1 - DR) + K_2 \cdot FAR + K_3 \cdot FAA + K_4 \cdot \text{ProgSize} \]
- Research question: how to set parameter \( K_i \)
Gradient Descent in GP

- Gradient descent search/hill climbing search is widely used in many techniques, including neural networks.

- Gradient descent search has two “problems”:
  - one run only has one potential solution;
  - it often has the local minuma.

- Gradient descent search can use the heuristics; local minuma is not the ideal/best solution, but often meets the requirements of many applications.

- Genetic beam search can solve/improve the local minuma problem, but it does not use the heuristic sufficiently.

- Can we combine them together?
Gradient Descent in GP

• Apply gradient descent locally on the numeric terminals

\[ \Delta O_i = -\alpha \cdot \frac{\partial C}{\partial O_i} = -\alpha \cdot \frac{\partial C}{\partial y} \cdot \frac{\partial y}{\partial O_i} \]

\[ C = \frac{(y - Y)^2}{2} \]

\[ \frac{\partial C}{\partial y} = \frac{\partial (\frac{(y-Y)^2}{2})}{\partial y} = y - Y \]

\[ Y = \text{class} - \frac{\text{numclass} + 1}{2} \]
Gradient Descent in GP

<table>
<thead>
<tr>
<th>Function $f$</th>
<th>meanings</th>
<th>$\frac{\partial f}{\partial a_1}$</th>
<th>$\frac{\partial f}{\partial a_2}$</th>
<th>$\frac{\partial f}{\partial a_3}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(+ $a_1$ $a_2$)</td>
<td>$a_1 + a_2$</td>
<td>1</td>
<td>1</td>
<td>–</td>
</tr>
<tr>
<td>(- $a_1$ $a_2$)</td>
<td>$a_1 - a_2$</td>
<td>1</td>
<td>-1</td>
<td>–</td>
</tr>
<tr>
<td>(* $a_1$ $a_2$)</td>
<td>$a_1 \times a_2$</td>
<td>$a_2$</td>
<td>$a_1$</td>
<td>–</td>
</tr>
<tr>
<td>(/ $a_1$ $a_2$)</td>
<td>$a_1 \div a_2$</td>
<td>$a_2^{-1}$</td>
<td>$-a_1 \times a_2^{-2}$</td>
<td>–</td>
</tr>
<tr>
<td>(if $a_1$ $a_2$ $a_3$)</td>
<td>if $a_1 &lt; 0$ then $a_2$ else $a_3$</td>
<td>0</td>
<td>1 if $a_1 &lt; 0$</td>
<td>0 if $a_1 &lt; 0$</td>
</tr>
</tbody>
</table>

\[
\frac{\partial y}{\partial O_2} = \frac{\partial (O_2 O_3)}{\partial O_2} = O_3
\]

\[
\frac{\partial y}{\partial O_5} = \frac{\partial (O_2 O_3)}{\partial O_5} = \frac{\partial (O_2 O_3)}{\partial O_3} \cdot \frac{\partial O_3}{\partial O_5}
\]

\[
= \frac{\partial (O_2 O_3)}{\partial O_3} \cdot \frac{\partial (O_4 + O_5)}{\partial O_5}
\]

\[
= O_2 \cdot 1 = O_2
\]
Gradient Descent in GP

\[ \alpha = \eta \cdot \frac{1}{\sum_i^N \left( \frac{\partial y}{\partial O_i} \right)^2} \]

- New value of the Numeric terminal

\[ (O_i)_{new} = O_i + \Delta O_i = O_i - \eta \cdot \frac{1}{\sum_i^N \left( \frac{\partial y}{\partial O_i} \right)^2} \cdot \frac{\partial C}{\partial O_i} \]

- This algorithm is only **locally** applied to individual programs.
- Genetic beam search is still **globally** applied to the evolutionary process.
- online learning vs offline learning
- every generation vs every five generations