Combining Symbolic Expressions and Black-box Function Evaluations in Neural Programs

Forough Arabshahi*  Sameer Singh*  Animashree Anandkumar†

*University of California, Irvine  †California Institute of Technology

Neural Programming
- Learning black-box functions
- Observations:
  - black-box function evaluations (fEval)
  - program execution traces (eTrace)
- Challenges: Lack of generalization due to:
  - fEval: Insufficient structural information
  - eTrace: Computational issues affecting the domain coverage
- Solution:
  - Most problems have access to symbolic representations (sym)
  - Combine sym and fEval data:
    - sym preserve problem’s structure
    - fEval enable function evaluation

Mathematical Equation Modeling
- Grammar rules:
  - I → = (E, E), ≠ (E, E)
  - E → T, F₁(E), F₂(E, E)
  - F₁ → sin, cos, tan, . . .
  - F₂ → +, ∧, x, . . .
  - T → −, 1, 0, 1, 2, π, x, y, . . . any number in [-3.14, +3.14]
- Covered domain:

<table>
<thead>
<tr>
<th>Unary functions, F₁</th>
<th>Terminal, T</th>
<th>Binary, F₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>sin cos sec tan</td>
<td>0 1 +</td>
<td></td>
</tr>
<tr>
<td>cot arsin arccos arccsc</td>
<td>2 3 x</td>
<td></td>
</tr>
<tr>
<td>arctan arcsinh csc csc arcsin</td>
<td>4 10 ∧</td>
<td></td>
</tr>
<tr>
<td>sech tanh coth sinh</td>
<td>0.5 −1</td>
<td></td>
</tr>
<tr>
<td>arcoth arctanh arcoth</td>
<td>0.4 0.7</td>
<td></td>
</tr>
</tbody>
</table>

- Examples of equation trees:

Generating symbolic equations
- Generate possible equations valid in the grammar
  - Start from a small initial set of axioms
  - For each axiom, choose a random tree node
  - Make local random changes to the node
  - Problem: More incorrect equations than correct
  - Solution: Sub-tree matching

Generating function evaluation equations
- Function Evaluation:
  - Range of floating point numbers of precision 2: [-3.14, 3.14]
  - For each unary function: draw a random number and evaluate
  - For each binary function: draw two random numbers and evaluate
- Representation of numbers:
  - For all numbers in the dataset, form the decimal tree expansion
  - E.g. 2.5 = 2 × 10^0 + 5 × 10^-1

Tree LSTMs for Modeling Equations
- Tree LSTM whose structure mirrors the input equation
  - Function blocks are LSTM cells
  - Weight sharing between occurrences of the same function
  - Symbol block is a 1-layer feed-forward net for embedding terminals
  - Number block is a 2-layer feed-forward net for embedding numbers

Experiments and Results

Complexity of an equation: its expression tree depth
- Equation Verification: Generalization to unseen identities

<table>
<thead>
<tr>
<th>Approach</th>
<th>Sym</th>
<th>F Eval depth 1</th>
<th>depth 2</th>
<th>depth 3</th>
<th>depth 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test set size</td>
<td>3527</td>
<td>401</td>
<td>5+2</td>
<td>542+158</td>
<td>2416+228</td>
</tr>
<tr>
<td>Majority Class</td>
<td>50.24</td>
<td>50.00</td>
<td>20.00</td>
<td>45.75</td>
<td>52.85</td>
</tr>
<tr>
<td>Sympy</td>
<td>81.74</td>
<td>-</td>
<td>80.00</td>
<td>89.11</td>
<td>82.98</td>
</tr>
<tr>
<td>RNN</td>
<td>66.37</td>
<td>-</td>
<td>50</td>
<td>62.93</td>
<td>65.13</td>
</tr>
<tr>
<td>LSTM</td>
<td>81.71</td>
<td>-</td>
<td>80.00</td>
<td>79.49</td>
<td>80.81</td>
</tr>
<tr>
<td>TreeNN</td>
<td>92.06</td>
<td>-</td>
<td>100</td>
<td>95.37</td>
<td>94.16</td>
</tr>
<tr>
<td>TreeLSTM</td>
<td>93.15</td>
<td>-</td>
<td>80.00</td>
<td>96.50</td>
<td>95.07</td>
</tr>
<tr>
<td>TreeLSTM + data</td>
<td>93.38</td>
<td>92.81</td>
<td>87.5</td>
<td>94.43</td>
<td>92.32</td>
</tr>
<tr>
<td>TreeLSTM + data</td>
<td>97.11</td>
<td>97.17</td>
<td>75.00</td>
<td>98.14</td>
<td>97.01</td>
</tr>
</tbody>
</table>

Equation Verification: Extrapolation to unseen depths
- Table: Extrapolation Evaluation to measure capability of the model to generalize to unseen depth. Acc: Accuracy, Prec: Precision, Rec: Recall