Neural-guided equation discovery

Jannis Brugger, David Richter, Mattia Cerrato, Mira Mezini, Stefan Kramer
Symbolic regression

- Problem of learning a symbolic expression from numerical data
Symbolic regression

- Problem of learning a symbolic expression from numerical data
Why equations?

• Most common form of knowledge in science
• Captures the relationships between features
• Understandable for humans

$$E_{kinetic} = \frac{1}{2}m \cdot v^2$$
Why equations?

- Most common form of knowledge in science
- Captures the relationships between features
- Understandable for humans

\[
\frac{1}{\pi} = \frac{2\sqrt{2}}{9801} \cdot \sum_{n=0}^{\infty} \frac{(4n)! \cdot (1103+26390n)}{(n!)^4 \cdot 396^{4n}}
\]

[J. Borwein, P. Borwein, and Bailey 1989]
Contribution

Combining AlphaZero [Silver et al. 2018] and Dreamcoder [Ellis et al. 2021] for equation discovery

Important aspects

• Equation transfer learning
• Usage of context-free grammar
• Embedding of syntax trees and datasets
Overview

- Input into system
- Input into discovery of one equation
- Expand one node in syntax tree
- Output of discovery of one equation
- Output of system
## System view

<table>
<thead>
<tr>
<th>Input into system</th>
<th>Dataset 1</th>
<th>Dataset n</th>
<th>Untrained equation finder</th>
<th>Grammar:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>3</td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output of system</th>
<th>k complete syntax tree</th>
<th>Reward for k complete syntax trees</th>
<th>Trained equation finder</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>S</td>
<td>Reward = 1 - L2 (Ydataset, Yprediction)</td>
<td></td>
</tr>
</tbody>
</table>

- **S**: Start symbol
- **Number**: Represents a number
- **Mul**: Multiplication
- **Product**: Product symbol
- **Variable**: Represents a variable
- **x**: Input value
Example datasets

Processed by
- Multilayer perceptron
- LSTM
- Bidirectional LSTM

Raw or normalized data

\[ y = x^2 \]

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>0.25</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
</tr>
</tbody>
</table>

\[ y = \log(x) \]

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>-0.3</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ y = \frac{1}{x} \]

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>10</td>
<td>0.1</td>
</tr>
</tbody>
</table>
Grammar

- Usage of domain knowledge
- Start symbol \( \langle S \rangle \)
- Production rules
  - left ::= right

Generated equations:

\[
\begin{align*}
(3 + x) \\
8 \\
(9 + (1 * x))
\end{align*}
\]
Syntax tree \((9 + (1 \times x))\)

```
< S > ::= ( < S > < OP > < S > ) | < T > | < CONST >
< T > ::= x | y
< OP > ::= * | / | + | −
< CONST > ::= 0 | 1 | ... | 9
```
Representation of syntax tree

- Hashed path [Alon et al. 2019]
  
  Pattern:
  left symbol, Hash(path), right symbol

  Path A:
  $\langle \text{CONST}, \text{Hash}(\langle S \rangle, \langle S \rangle \langle \text{OP} \rangle) \rangle, +$

  Path B:
  $\langle S \rangle, \text{Hash}(\langle S \rangle \langle S \rangle), \langle \text{OP} \rangle$

  Path C:
  $\langle S \rangle, \text{Hash}(\langle S \rangle, \langle S \rangle, \langle S \rangle, \langle T \rangle), x$

  Path …
Representation of syntax tree

- Hashed path [Alon et al. 2019]
- Full tree

Pattern:
- Node [Child Nodes]

Example tree
\[
\langle S \rangle \left[ \langle S \rangle \ldots \langle OP \rangle \langle + \rangle \langle S \rangle \ldots \rangle \right]
\]
Representation of syntax tree

- Hashed path [Alon et al. 2019]
- Full tree
- Fringe
  
  Pattern:
  Fringe nodes

Example tree

\[(9 + (1 \times \langle S \rangle))\]
### Input into discovery of one equation

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Unfinished syntax tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

Grammar:
- $S \rightarrow \text{Number Plus Sum}$
- $S \rightarrow \text{Number Mul Product}$
- $\ldots$
- $\text{Variable} \rightarrow x$

### Grammar view

- **Grammar:** $S \rightarrow \text{Number Plus Sum}$
- $S \rightarrow \text{Number Mul Product}$
- $\ldots$
- $\text{Variable} \rightarrow x$

### Output of discovery of one equation

#### Complete syntax tree

- $S$
- $\text{Number Mul Product}$
- $1 \ast \text{Variable}$
- $\times x$

#### Reward for complete syntax tree

$$\text{Reward} = 1 - L2 (\text{ydataset, Yprediction})$$

#### Intermediate syntax trees

- $S$
- $\text{Number Mul Product}$

#### MCTS visit counts

- Histogram of visit counts for MCTS
Search tree
Monte-Carlo tree search

Probability for child node depends on:

- Prior
  - Uniform
  - Grammar
  - Recognition module
- Visit counts
- Expected reward
Recognition module

Expand one node in syntax tree

1. Select new node to expand
2. Run Monte-Carlo tree search
3. Sample rule to apply
4. Get visit counts for child nodes

Prior from recognition module:
Input:
- Syntax tree
- Dataset
Output:
- Probability for rules in grammar

Syntax tree to be expanded:
S

Number Mul Product

Dataset:

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

Run MCTS for n steps
Use prior from recognition module

Until equation is completed

Monte-Carlo tree search:
S

Number Mul Product

Visit counts for child nodes

Grammar rules
Experiment: Sanity check

Grammar

\[ S ::= \text{Number Plus } S \]
\[ S ::= \text{Number} \]
\[ \text{Number} ::= "1" \]
\[ \text{Plus} ::= " + " \]

Equations to discover

- 1
- 1 + 1
- ...
- 1 + 1 + 1 + 1 + 1

True formula: 1

\[ "S" : [0.09 0.91] \]

True formula: \(1 + 1 + 1 + 1\)

\[ "S" : [0.93 0.07] \]
\[ "1 + S" : [0.84 0.16] \]
\[ "1 + 1 + S" : [0.73 0.27] \]
\[ "1 + 1 + 1 + S" : [0.46 0.54] \]
\[ "1 + 1 + 1 + 1 + S" : [0.38 0.62] \]
Experiment: Discovering equations

Grammar [1/2]:
- \( S \rightarrow \text{Number Plus Sum} \) [0.2]
- \( S \rightarrow \text{Number Mul Product} \) [0.15]
- \( S \rightarrow \text{Number Div OP Product CP} \) [0.15]
- \( S \rightarrow \text{Number Minus OP Sum CP} \) [0.1]
- \( S \rightarrow \text{Number Mul Trigonometric} \) [0.1]
- \( S \rightarrow \text{Number Mul Logarithm} \) [0.1]
- \( S \rightarrow \text{Number Mul Root} \) [0.1]
- \( S \rightarrow \text{Number} \) [0.1]
- \( \text{Sum} \rightarrow \text{Variable Plus Sum} \) [0.3]
- \( \text{Sum} \rightarrow \text{Variable} \) [0.7]
- \( \text{Number} \rightarrow '1' \) [1.0]
- \( \text{Number} \rightarrow '2' \) [0]
- \( \text{Number} \rightarrow '3' \) [0]
- \( \text{Number} \rightarrow '4' \) [0]
- \( \text{Plus} \rightarrow '+' \) [1.0]
- \( \text{Product} \rightarrow \text{Variable Mul Product} \) [0.3]
- \( \text{Product} \rightarrow \text{Variable} \) [0.7]
- \( \text{Mul} \rightarrow '*' \) [1.0]
- \( \text{Div} \rightarrow '/' \) [1.0]
- \( \text{Minus} \rightarrow '-' \) [1.0]
Experiment: Discovering equations

Grammar [2/2]:

- Trigonometric → Trigonometric_operator OP Number Mul Variable CP [1.0]
- Trigonometric_operator → ‘sin’ [0.5]
- Trigonometric_operator → ‘cos’ [0.5]
- Logarithm → Logarithm_operator OP Number Mul Variable CP [1.0]
- Logarithm_operator → ‘log’ [1.0]
- Root → OP Number Plus Sum CP Power RootOperator [1.0]
- RootOperator → ‘0.5’ [1.0]
- Power → ‘**’ [1.0]
- OP → ‘(’ [1.0]
- CP → ‘)’ [1.0]
- Variable → ‘x’ [1.0]
Experiment: Discovering equations

\( x \sim \mathcal{N}(\mu = \{0.3, 2.24, 4.18, 6.12, 8.06, 10\}, \sigma^2 = 0.1) \)

Equations:

\begin{align*}
\bullet \quad y &= \sqrt{1 + x} \\
\bullet \quad y &= \sqrt{1 + x + x} \\
\bullet \quad y &= \sqrt{1 + x} \\
\bullet \quad y &= \cos(x) \\
\bullet \quad y &= \sin(x) \\
\bullet \quad y &= \log(x) \\
\bullet \quad y &= x \\
\bullet \quad y &= x + 1 \\
\bullet \quad y &= x^2 \\
\bullet \quad y &= x^3 \\
\bullet \quad y &= x + x + x \\
\bullet \quad y &= \cos(x) \\
\bullet \quad y &= \sin(x) \\
\bullet \quad y &= \log(x) \\
\bullet \quad y &= x \\
\bullet \quad y &= x + 1 \\
\bullet \quad y &= \frac{1}{x} \\
\bullet \quad y &= \frac{1}{x^2} \\
\bullet \quad y &= \frac{1}{x^3} \\
\bullet \quad y &= 1 - x \\
\bullet \quad y &= 1 - (x + x) \\
\bullet \quad y &= 1 - (x + x + x) \\
\bullet \quad y &= 1
\end{align*}
Experiment: Discovering equations

100 MCTS steps
Experiment: Discovering equations

100 MCTS steps

![Graph showing the performance of different neural network architectures over time. The x-axis represents time in hours, and the y-axis represents the ratio of the loss. The graph compares prior neural net, MLP Norm Tree structure, MLP Tree structure, MLP Norm Fringe, and MLP Fringe architectures.]
Experiment: Discovering equations

100 MCTS steps
Outlook

- Dataset processing
- Find pattern in solution and add them to grammar
Outlook

- Dataset processing
- Find pattern in solution and add them to grammar
Thank you for your attention

Conclusion:

• Symbolic regression
• Context-free grammar
• Representation of syntax tree
• System for neural-guided equation discovery
• Experiments
Contact

• Jannis Brugger
  • TU Darmstadt, Germany
  • jannis.brugger@tu-darmstadt.de

• Mattia Cerrato
  • JGU Mainz, Germany
  • mcerrato@uni-mainz.de

• Stefan Kramer
  • JGU Mainz, Germany
  • kramer@informatik.uni-mainz.de
Bibliography


