Motivations: Learning Dynamics

- Given a set of input/output states of a black-box system, learn its internal mechanics.
- Discrete system: input/output are vectors of the same size which contain discrete values.
- Dynamic system: input/output are states of the system and output is the next input.

Goal: produce an artificial system with the same behavior, i.e., a digital twin.
- Representation: propositional logic programs encoding multi-valued discrete variables.
- Method: learn the dynamics of systems from their state transitions.

Problem: Combinatorial Explosion

We introduce an heuristic algorithm PRIDE which trades the completeness of GULA for polynomial complexity. PRIDE learns a subset of P(T) sufficient to realize T.

Run Time

<table>
<thead>
<tr>
<th>System runs (s)</th>
<th>7</th>
<th>9</th>
<th>10</th>
<th>12</th>
<th>13</th>
<th>15</th>
<th>18</th>
<th>25</th>
</tr>
</thead>
<tbody>
<tr>
<td>GULA run time</td>
<td>0.047</td>
<td>0.13</td>
<td>0.19</td>
<td>0.26</td>
<td>0.48</td>
<td>0.75</td>
<td>1.0</td>
<td>1.3</td>
</tr>
<tr>
<td>PRIDE run time</td>
<td>0.005</td>
<td>0.02</td>
<td>0.06</td>
<td>0.28</td>
<td>0.56</td>
<td>0.76</td>
<td>0.93</td>
<td>1.2</td>
</tr>
</tbody>
</table>

Average run time of GULA and PRIDE when learning Boolean networks of PyBoolNet [2], at least 10,000 runs with a timeout of (T.O. of 1.000 seconds.

PRIDE performances allows to learn more complex systems and drastically reduce computation time of smaller ones.

Algorithm: PRIDE

Theorem 1 (Consistent Rule Always exists). Let \( T \subseteq S^T \times S^T \), \( (s, s') \in T \) and \( v^{\text{val}} \subseteq v' \). The rule \( R = v^{\text{val}} \rightleftharpoons s \) is consistent with \( T \) and realizes \( (s, s') \).

Theorem 2 (Irreducible Rules are Optimal). Let \( R \) be a rule consistent with a set of transitions \( T \subseteq S^T \times S^T \). If \( \forall R' \subseteq \text{body}(R') \), \( v^{\text{val}} \subseteq \text{body}(R') \), \( R' \) conflicts with \( T \), then \( R' \) is not consistent with \( T \) such that \( R' \subseteq R \) and thus, \( R \in P(T) \).

Idea:
- given positives/negatives examples of occurrence of a target atom \( v^{\text{val}} \) in \( T \),
- we can find a rule \( R \in P(T) \) to explain each positive example \( s \) starting from \( v^{\text{val}} \) - remove body atom until conflict is not avoidable.

Algorithmic properties:
- it is faster to start from \( v^{\text{val}} \) by specialise it until consistency and then generalise
- adding only the atoms of \( s \) ensure to matches \( s \), in worst case we reach \( v^{\text{val}} \) - \( s \) more variable in the system, more generalization is avoided.

Implementation: python library and user API

PyLFIT Library
- Open source python library: pip install pylfit
- Contain all LFIT algorithms and a simple user API
- Built-in data/model conversion/usage
- User API
  - Load raw data of different format into a Dataset object
  - Choose desired model type and run corresponding LFIT algorithm
  - Use model object for predictions, analysis or convert it to other format.

Predictions:
- DMVLP and CDMVLP (constraints) can be used for predicting possible target states
- WDVMVLP model both possibility and impossibility, it also adds weightings to rules w.r.t. observations to allow probabilistic predictions of target atom occurrence in a transition

Dataset Model

Summary

- The polynomiality of PRIDE is obtained at the cost of completeness over \( P(T) \).
- Still, the program learned can reproduce all observations and provides minimal explanation for each of them in the form of optimal rules.
- The source code is available as open source on github and pygi.org (see OR code).
- A user-friendly API allows to easily use LFIT algorithms on different kinds of datasets and is already being used in several research collaborations [4].