Making Universal Induction Efficient by Specialization

AGI @ Quebec
Alexey Potapov, Sergey Rodionov
{potapov, rodionov}@aideus.com
2014
(General) intelligence is an agent’s ability to efficiently achieve goals in a wide range of environments with insufficient knowledge and resources.
Gap between Universal and Pragmatic Methods

• Universal methods
  • can work in arbitrary computable environment
  • computationally infeasible
  • approximations are either inefficient or not universal

• Pragmatic methods
  • work in non-toy environments
  • set of environments is highly restricted

=> Bridging this gap is necessary
Key Idea

• Humans create narrow methods, which efficiently solve arbitrary recurring problems
• Generality should be achieved not by a single uniform method solving any problem in the same fashion, but by automatic construction of (non-universal) efficient methods
• Program specialization is the appropriate concept*, which relates general and narrow intelligence methods
• However, no analysis of possible specialization of concrete models of universal intelligence has been given yet.

Program Specialization

• Let $p_L(x,y)$ be some program (in some language $L$) with two arguments
• Specializer $spec_R$ is such program (in some language $R$) accepting $p_L$ and $x_0$ that
  \[(\forall y)spec_R(p_L, x_0)(y) = p_L(x_0, y)\]
• $spec_R(p_L, x_0)$ is the result of deep transformation of $p_L$ that can be much more efficient than $p(x_0, .)$

Futamura-Turchin projections

\[(\forall x)spec_R(intL, p_L)(x) = intL(p_L, x)\]
\[(\forall p_L, x)spec_R(spec_R, intL)(p_L)(x) = intL(p_L, x)\]
\[(\forall intL)spec_R(spec_R, spec_R)(intL) = comp_{L\rightarrow R}\]
Universal Mass Induction

- Let \( \{x_i\}_{i=1}^n \) be the set of strings
- An universal method cannot be applied to mass problems since typically
  \[ K_U(x_1x_2...x_n) \ll \sum_{i=1}^n K_U(x_i) \]
  where \( K \) is Kolmogorov complexity on universal machine \( U \)
- However, \( K_U(x_1x_2...x_n) \approx \min_S \left( l(S) + \sum_{i=1}^n K_U(x_i \mid S) \right) \) can be true
- One can search for models \( y_i^* = \arg \min_{y: S(y) = x_i} l(y) \) for each \( x_i \) independently
  within some best representation \( S^* = \arg \min_S \left( l(S) + \sum_{i=1}^n l(y_i^*) \right) \)

If \( S \) is not an universal program than this search can be made (much) more efficient than exhaustive search
Specialization of Universal Induction

• Universal mass induction consists of two procedures
  • Search for models
  \[ MSearch(S, x_i) \rightarrow y_i^* = \arg \min_{y : S(y) = x_i} l(y) \]
  • Search for representations
  \[ RSearch(x_1, \ldots, x_n) \rightarrow S^* = \arg \min_S \left( l(S) + \sum_{i=1}^{n} l(y_i^*) \right) \]
  • \( MSearch(S, x) \) is executed for different \( x \) with same \( S \)
  • This search cannot be non-exhaustive for any \( S \), but it can be efficient for some of them
  • One can consider computationally efficient projection
    \[ spec(MSearch, S): (\forall x)spec(MSearch, S)(x) = MSearch(S, x) \]
Approach to Specialization

- Direct specialization of $MSearch(S, x)$ w.r.t. some given $S^*$
- No general techniques for exponential speedup exists
- And how to get $S$? $RSearch$ is still needed
- Find $S' = spec(MSearch(S, x), S^*)$ simultaneously with $S^*$
- Main properties of $S, S'$: $$(\forall x) S(S'(x)) = x$$
  $$l(S) + \sum_i l(S'(x_i)) \rightarrow \text{min}$$

- $S$ is a generative representation (decoding)
- $S'$ is a descriptive representation (encoding)
  - $S'$ is also the result of specialization of the search for generative models, so in general it can include some sort of optimized search
- Simultaneous search for $S$ and $S'$ will be referred to as $SS'$-search
Combinatory Logic

• $K x y \to x$  
  ($((K x) y)$)

• $S x y z \to x z (y z)$  
  (‘(((S x) y) z))

  – $S K K x \to K x (K x) \to x$  
    $I = S K K$  
    $I x \to x$

  – $(S (K (S I)) (S (K K) I) x y) \to \ldots \to y x$

  – and other combinators: $B$, $b$, $W$, $M$, $J$, $C$, $T$

• In lambda-calculus

  – $\lambda x.x == I$  
    $\lambda x. \lambda y.(y x) == S (K (S I)) (S (K K) I)$
Mass Induction in CL

- Data strings $x_i$ with common regularities
- One representation $S$
- Individual models $y_i$

- $MSearch$ enumerates all models to find the shortest appropriate model: $S y_i = x_i$
- $RSearch$ enumerates all $S$ and calls $MSearch$ for each $S$
**SS'-Search example**

- $S' = KC$
- $S$ and $S'$ are enumerated together
- $S'$ is used instead of $MSearch$ to obtain $y_i$

Data strings $x_i$ with common regularities

Individual models $y_i$

One representation $S$
Genetic programming for Mass Induction

- **RSearch+MSearch**
  - Genome is composed of $S$ and $\{y_i\}$ each of which corresponds to a separate chromosome
- **SS'-Search**
  - Genome is composed of two chromosomes – $S$ and $S'$
  - Each chromosome is subjected to crossover independently
  - Implementation of GP for CL is described in our previous paper

Experimental results

- Simple redundancy

**SS'-Search**

\[
\begin{align*}
1 & \\
1 & \\
1 & \\
\vdots & \\
1 & \\
\end{align*}
\]

\[S \rightarrow W110010 \rightarrow S' \rightarrow J(bMJK)T\]

**RSearch**

\[
\begin{align*}
1 & \\
1 & \\
1 & \\
\vdots & \\
1 & \\
\end{align*}
\]

\[11100101 \rightarrow 11100101 \rightarrow 11100101 \rightarrow 11100101 \rightarrow 0101\]

- *RSearch* fails to find optimal solution even in this simple case
- *SS'-Search* appears to be efficient; *S'* constructs correct models
- This can seem strange since *S'* is not simpler than \(y_i\), but *SS'-Search* allows for incremental improvement
Experimental results

• Poorly compressible data

\[
\begin{array}{c}
101101101010 \\
001101001011 \\
111111110011 \\
\vdots \\
011011010111 \\
\end{array}
\quad \begin{array}{c}
0101101101010 \\
000110100111 \\
011111110011 \\
\vdots \\
0011011010111 \\
\end{array}
\]

\textit{SS'-Search} extracts information from data to construct models, while \textit{RSearch} searches for models blindly.

• \textit{RSearch} fails to find any precise solution
Experimental results

- Simple common regularity

\[ \text{SS}'-\text{Search} \]

\[ \begin{array}{c}
0000 \\
0001 \\
0010 \\
\ldots \\
1111 \\
\end{array} \xrightarrow{\text{BBB(BM)}} \begin{array}{c}
00000000 \\
00010001 \\
00100010 \\
\ldots \\
11111111 \\
\end{array} \]

\[ \text{S'} \]

\[ \begin{array}{c}
00000000 \\
00010001 \\
00100010 \\
\ldots \\
11111111 \\
\end{array} \xrightarrow{\text{B(SJCK)}} \begin{array}{c}
0000 \\
0001 \\
0010 \\
\ldots \\
1111 \\
\end{array} \]

- Both methods successfully found good solutions
- \text{RSearch} requires low complexity from both representations and models
Experimental results

- More complex regularities

```
SS'-Search
S
B(S(BST))M
S'
JKK

RSearch
159951
248842
678876
...
179971
```
Experimental results

• More complex regularities

SS'-Search

S
KBb
W
S'
BK

RSearch

30718
01232
68956
...
78214

307718
012232
689956
...
782214
Conclusion

• Ideas of universal induction, representations, and program specialization were combined.
• Specialization of universal (mass) induction w.r.t. some (generative) representation yields descriptive representations.
• These descriptive representations being not Turing-complete can construct data models much more efficient than universal induction methods.
• Also, automatic simultaneous construction of generative and descriptive representations appeared to be more efficient than construction of generative representations and models, so explicit specialization seems to be not necessary here.
• Can $R$Search be more efficient than $SS'$-Search?
Thank you for attention

AGI @ Quebec

Alexey Potapov, Sergey Rodionov

{potapov, rodionov}@aideus.com

2014