Innate concepts as specialized programs?

Chung-chieh Shan
Rutgers University

Cornell workshop on grammar induction
Commentary on Noah Goodman’s talk
‘Concept learning as probabilistic program induction’
May 16, 2010
Marr

I represent knowledge in (probabilistic) programming languages for human communication and machine execution.

➤ Separate what from how
➤ Reconcile generality with specialization

Question
How to base algorithmic accounts of human performance on Noah’s computational models?

➤ Initial hypothesis: Church’s general inference
➤ Eventual hypotheses: hand-coded special inference

Complaint
Why not discard Church model eventually?
Especially if special inference is approximate…

Suggestion
Custom code generation—compile model into inference!
I represent knowledge in (probabilistic) programming languages for human communication and machine execution.

- Separate what from how
- Reconcile generality with specialization

**Question**

How to base algorithmic accounts of human performance on Noah’s computational models?

- Initial hypothesis: Church’s general inference
- Eventual hypotheses: hand-coded special inference

**Complaint**

Why not discard Church model eventually?
Especially if special inference is approximate…

**Suggestion**

Custom code generation—compile model into inference!
I represent knowledge in (probabilistic) programming languages for human communication and machine execution.

- Separate what from how
- Reconcile generality with specialization

**Question**
How to base algorithmic accounts of human performance on Noah’s computational models?

- Initial hypothesis: Church’s general inference
- Eventual hypotheses: hand-coded special inference

**Complaint**
Why not discard Church model eventually?
Especially if special inference is approximate…

**Suggestion**
Custom code generation—compile model into inference!
I represent knowledge in (probabilistic) programming languages for human communication and machine execution.

- Separate what from how
- Reconcile generality with specialization

**Question**

How to base algorithmic accounts of human performance on Noah’s computational models?

- Initial hypothesis: Church’s general inference
- Eventual hypotheses: hand-coded special inference

**Complaint**

Why not discard Church model eventually? Especially if special inference is approximate...

**Suggestion**

Custom code generation—compile model into inference!
‘Domain-general’, ‘language-specific’ are properties of modules.  

A module is a part of a description of a system.  

- Modularity should be invariant under physically entangled emulation with dye pack.  
- Modularity makes a theory more concise, comprehensible.  

Organizing principle: reuse in the face of change
‘Domain-general’, ‘language-specific’ are properties of modules. **A module is a part of a description of a system.**

- Modularity should be invariant under physically entangled emulation with dye pack.
- Modularity makes a theory more concise, comprehensible.

Organizing principle: reuse in the face of change

\[
\begin{align*}
\text{general description} & \xrightarrow{\text{specialize}} \text{specific machinery} \\
\begin{array}{c}
\varepsilon : \varepsilon \\
\varepsilon : \varepsilon \\
\varepsilon : \varepsilon \\
\end{array} & \times abb = \\
\begin{array}{c}
b : \varepsilon \\
b : b \\
b : \varepsilon \\
\end{array} & \begin{array}{c}
a \\
\varepsilon \\
a \\
\end{array} & \begin{array}{c}
b \\
\varepsilon \\
b \\
\end{array}
\end{align*}
\]
Computation: $\lambda x. x^8$
Computation: $\lambda x. x^8$

Algorithm: $\lambda x. ((x^2)^2)^2$
Futamura

Computation: $\lambda x. x^8$
Algorithm: $\lambda x. ((x^2)^2)^2$
Algorithm: $\lambda x. f(3)$ where $f(0) = x$

$$f(k + 1) = f(k)^2$$
Computation: \( \lambda x. \ x^8 \)

Algorithm: \( \lambda x. \ ((x^2)^2)^2 \)

Algorithm: \( \lambda x. \ f(3) \) where \( f(0) = x \)
\[
f(k + 1) = f(k)^2
\]

Algorithm generator: ‘\( \lambda x. \)’ \( f(3) \) where \( f(0) = 'x' \)
\[
f(k + 1) = f(k)^2'
\]
Futamura

Computation: $\lambda x. \ x^8$

Algorithm: $\lambda x. \ ((x^2)^2)^2$

Algorithm: $\lambda x. \ f(3)$ where $f(0) = x$

$f(k + 1) = f(k)^2$

Algorithm generator: $\lambda x. \ f(3)$ where $f(0) = 'x'$

$f(k + 1) = f(k)^2$

Computation: $\lambda x. \ x^{10}$

Algorithm: $\lambda x. \ ((x^2)^2 \times x)^2$

Algorithm generator: $\lambda x. \ g(10)$ where $g(1) = 'x'$

$g(2n) = g(n)^2$

$g(2n + 1) = g(2n) \times x$
A module is a part of a description of a system.

specialize

general description → specific machinery

‘Compile time’ includes evolution.