Concept learning and the language of thought

Noah D. Goodman
Stanford University
IPAM graduate summer school
July 15, 2011
Statistics and composition

Probabilistic language of thought hypothesis

Thought is useful in an uncertain world

Thought is productive: “the infinite use of finite means”

Belief  Desire  Action

Probabilistic inference  Generative models  Compositional representations
The probabilistic language of thought hypothesis:

- Mental representations are compositional,
- Their meaning is probabilistic,
- They encode generative knowledge,
- Hence, they support thinking and learning by probabilistic inference.
PLoT

- The probabilistic language of thought hypothesis: Mental representations are functions in a stochastic process calculus (e.g. $\psi\lambda$-calculus / Church).
- Intuitive framework theories.
- Flexible reasoning and language use.
- Learning structured concepts.
If concepts are probabilistic programs, then concept learning is *probabilistic program induction*.
Outline

If concepts are probabilistic programs, then concept learning is *probabilistic program induction*.

(query
  (define concept (sample-PLoT-expression))
  concept
  (and (= (noisy (sample concept)) obs1)
       (= (noisy (sample concept)) obs2)
       ...
  ))
If concepts are probabilistic programs, then concept learning is *probabilistic program induction*.

- Boolean categories
- Quantified concepts
- Natural number concepts
- Generative kinds
- Program induction
Categorization

Medin & Schaffer (1978):
Categorization

Medin & Schaffer (1978):

“These are Feps”
Categorization

Medin & Schaffer (1978):

“These are Feps”

“These are not Feps”
Categorization

Medin & Schaffer (1978):

“These are Feps”

“Is this a Fep?”

“These are not Feps”
Categorization

- Rule-based category learning:
  - Infinitely many concepts formed compositionally.

- Statistical category learning:
  - Graded inferences from sparse, noisy evidence.
Categorization


```
% Fep

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.5</td>
<td>1.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
"These are Feps"
"These are not Feps"
"Is this a Fep?"
```
Categorization


- Graded judgements

- "These are Feps"
- "These are not Feps"
- "Is this a Fep?"

- % Fep

• Graded judgements
Categorization


- Graded judgements
- Typicality

% Fep

“These are Feps”

“Is this a Fep?”

“These are not Feps”

• Graded judgements

• Typicality
Categorization


- Graded judgements
- Typicality
- Prototype enhancement

% Fep

“These are Feps”

“These are not Feps”

“Is this a Fep?”
Generating rules

“It’s a Fep if it has flat head and round wings”
(define fep? (λ (x) (and (flat-head x) (round-wings x))))
(define \texttt{fep?} \\
(\lambda (x) \\
 (and (\texttt{flat-head} x) \\
 (\texttt{round-wings} x))) \\
(fep? ) \\
=> \texttt{true}
(define \texttt{fep?} \\
(\lambda \ (x) \\
 (\text{and} \ (\text{flat-head} \ x) \ \\
 (\text{round-wings} \ x))))

(fep? \\
 => \ true

(\texttt{rule-generator} \\
 (\lambda () \\
 (if (\text{flip} \ 0.3) \\
 (\texttt{sample-feature}) \\
 (\texttt{combine-rules} \ (\texttt{sample-feature}) \ \\
 (\texttt{rule-generator}))))

(\texttt{combine-rules} \\
 (\lambda \ (r1 \ r2) \\
 (\lambda \ (x) \ (\text{and} \ (r1 \ x) \ (r2 \ x))))
(define fep? (λ (x) (and (flat-head x) (round-wings x))))

(fep? )
=> true

(define rule-generator (λ ()
  (if (flip 0.3)
      (sample-feature)
      (combine-rules (sample-feature)
                     (rule-generator)))))

(define combine-rules (λ (r1 r2)
                         (λ (x) (and (r1 x) (r2 x)))))
(define fep? (\(x\) (and (flat-head x) (round-wings x))))

(fep? ) => true

(define rule-generator (\() 
  (if (flip 0.3) 
    (sample-feature) 
    (combine-rules (sample-feature) (rule-generator))))

(define combine-rules (\(r1 \, r2\) 
  (\(x\) (and (r1 x) (r2 x)))))
Generating rules

(define \texttt{fep?} \\
(\lambda (x) \\
 \hspace{1em} \text{(and (flat-head } x) \\
 \hspace{2em} \text{(round-wings } x))))

(fep? )

=> true

• Longer rules have lower probability (Occam’s razor).

(define \texttt{rule-generator} \\
(\lambda () \\
 \hspace{1em} \text{(if } (\text{flip } 0.3) \\
 \hspace{2em} \text{(sample-feature)} \\
 \hspace{3em} \text{(combine-rules } (\text{sample-feature} \\
 \hspace{4em} \text{(rule-generator)))) \)))

(define \texttt{combine-rules} \\
(\lambda (r1 r2) \\
 \hspace{1em} \text{(and } (r1 x) \hspace{1em} (r2 x))))
Generating rules

Put uncertainty over rule probabilities:

```scheme
(define rule-prob (uniform 0 1))
(define rule-generator
  (λ ()
    (if (flip rule-prob)
      ...
      ...
  )
```

Generate disjunctive normal form (DNF) rules:

```scheme
(define fep? (λ (x)
  (λ (x)
    (and (or (flat-head x) ...)
      (or (round-wings x) ...)
      ...
    )))
```

The general idea:
grammar-based induction.
Inducing rules

Inference:

(query
  (define rule (rule-generator))
  (rule
    (and (= (rule true) true)
         (= (rule false) false)
         ...
  ))

Hypotheses

Data
Inducing rules

Inference:

(query
  (define rule (rule-generator))
  (rule
    (and (= (noisy (rule )) true)
        (= (noisy (rule )) false)
        ...
  ))

Observation noise:

(define noisy
  (λ (bit) (if (flip b) bit (not bit))))
Categorization

Human data

Rule-induction model

Goodman, et al. (2007, 2008a, 2008b)
Categorization

\[ r = 0.99 \]

(one free param.)

Goodman, et al. (2007, 2008a, 2008b)
Graded judgments

\[ r = 0.99 \]
(one free param.)

Human data vs. Rule-induction model

Goodman, et al. (2007, 2008a, 2008b)
Categorization

- Graded judgments
- Typicality

Goodman, et al. (2007, 2008a, 2008b)

\[ r = 0.99 \] (one free param.)
Categorization

- Graded judgments
- Typicality
- Prototype enhancement

```
\[ r = 0.99 \]
(one free param.)
```

Goodman, et al. (2007, 2008a, 2008b)
Categorization

- Graded judgments
- Typicality
- Prototype enhancement
- Selective attention

\[ r = 0.99 \]
(one free param.)

Human data vs. Rule-induction model

- Fep
- non-Fep
- transfer

Goodman, et al. (2007, 2008a, 2008b)
• Model assumes individuals **sample** a (few) rule(s).
Complexity shift

- With more exposure to training examples, subjects shift from simple to complex categorization patterns. (Medin, et al., 1982)

- Tradeoff between observation noise and simplicity bias (Occam’s razor).
With more exposure to training examples, subjects shift from simple to complex categorization patterns. (Medin, et al., 1982)

Tradeoff between observation noise and simplicity bias (Occam’s razor).

Single feature:

```scheme
(define fep? (λ (x) (flat-head x)))
```
Complexity shift

- With more exposure to training examples, subjects shift from simple to complex categorization patterns. (Medin, et al., 1982)

- Tradeoff between observation noise and simplicity bias (Occam’s razor).

```scheme
(define fep? (\(x\) (\(\lambda\) (flat-head x))))
```

Single feature:

```scheme
(define fep? (\(x\) (or (and (flat-head x) (round-wings x)) (and (round-head x) (square-wings x)))))
```

Multiple feature:
Complexity shift

- With more exposure to training examples, subjects shift from simple to complex categorization patterns. (Medin, et al., 1982)

- Tradeoff between observation noise and simplicity bias (Occam’s razor).

Model inferences:

- Better rules (Log posterior)
  - One-feature rules.
  - Four-feature rules.

Amount of evidence
• With more exposure to training examples subjects shift from simple to complex categorization patterns. (Medin, et al., 1982)

• Tradeoff between observation noise and simplicity bias (Occam’s razor).
With more exposure to training examples, subjects shift from simple to complex categorization patterns. (Medin, et al., 1982)

Tradeoff between observation noise and simplicity bias (Occam’s razor).
Evaluating languages

• Induction to the language generated by the DNF grammar explains important phenomena (and fits relevant data).

• But is this the right LoT?

• Test on wider data set?

• Compare to other propositional languages?
Broader test

- 7 Boolean features.
- 43 randomly generated concepts (3-6 pos. + 2 neg. exs)
- 128 judgements (~122 transfer questions)

**Figure 13**

Human categorization response frequency against model posterior generalization probability. Error bars represent standard error of frequency assuming binomial distributions. Frequencies are computed by first binning responses according to model predictions. The mean of response frequencies binned according to model predictions is computed for each run separately. Error bars represent standard error of the mean over runs. Bars below each data point indicate the number of runs contributing to that bin.

Model by Love et al. than they are to representations in RULX. Conjunctive blocks of R

(F

formulae are analogous to the clusters that SUST"IN learn sv with features that are omitted from a conjunctive clause analogous to features that receive low attentional weights in SUST"IN. Three of these models—RULX, SUST"IN, and RR

—navigate similar issues of representational flexibility. Trade-offs between conceptual complexity and ease of learning and generalization under uncertainty. The main advantages that Rational Rules offer over the other two models come from its focus on the computational theory level of analysis and the modeling power we gain at that level, the ability to work with a minimal number of free parameters and still achieve strong quantitative data fits, the ability to separate out the effects of representational commitments and inductive logic from the search and memory processes that implement inductive computations, and the ability to seamlessly extend the model to work with different kinds of predicate-based representations, such as those appropriate for learning concepts in continuous spaces, concepts defined by causal implications (see Nx, Goodman et al., In Press), or concepts defined by relational predicates (see below).

The central theme of our work is the complementary nature of rule-based representations and statistical inference and the importance of integrating these two capacities in a model of human concept learning. Other authors have written about the need for both rule-based and statistical abilities—or of often rules and similarity—in concept learning and cognition more generally. Sloman, Pinker, and Pothos. The standard approach to combining these notions employs a "separate but equal" hybrid approach, endowing a model with two modules or systems of representation—one specialized for rule-based representations and one for statistical or similarity-based representations, and then letting these two modules compete or cooperate to solve some learning task. The "TRIUM model of Brock and Kruschke is a good example of this approach, where a rule module and a similarity module are trained in parallel, and a gating module arbitrates between their predictions at decision time.

We argue here for a different, more unified approach to integrating rules and statistics. Rules expressed in a flexible concept language provide a single unitary representation, statistics provides not a complementary form of representation, but the rational inductive mechanism that maps from observed data to the concept language. We thus build on the insights of Shepard and Tenenbaum that the effects of similarity and rules can both emerge from a single model, one with a single representational system of rule-like hypotheses, learned via a single rational inductive mechanism that operates according to the principles of Bayesian statistics.

Evaluating languages

• High-throughput MTurk experiment.

• 108 concepts,
  • Boolean (circle or red)
  • Context-dependent (“Determiners”)
    (unique largest, exists another with same shape)

• 2 orders per concept,

• 1596 participants.

Piantadosi, Goodman, Tenenbaum (in prep)
Evaluating languages

• High-throughput MTurk experiment.
  
• 108 concepts,
  
• Boolean (circle or red)
  
• Context-dependent (“Determiners”)  
  (unique largest, exists another with same shape)

• 2 orders per concept,

• 1596 participants.  

Piantadosi, Goodman, Tenenbaum (in prep)
Evaluating languages

- High-throughput MTurk experiment.
- 108 concepts,
  - Boolean (circle or red)
  - Context-dependent (“Determiners”) (unique largest, exists another with same shape)
- 2 orders per concept,
- 1596 participants.

Piantadosi, Goodman, Tenenbaum (in prep)
Evaluating languages

- High-throughput MTurk experiment.
- 108 concepts,
  - Boolean (circle or red)
  - Context-dependent ("Determiners") (unique largest, exists another with same shape)
- 2 orders per concept,
- 1596 participants.

Piantadosi, Goodman, Tenenbaum (in prep)
Evaluating languages

- High-throughput MTurk experiment.
- 108 concepts,
- Boolean (circle or red)
- Context-dependent (“Determiners”) (unique largest, exists another with same shape)
- 2 orders per concept,
- 1596 participants.

Piantadosi, Goodman, Tenenbaum (in prep)
Evaluating languages

- High-throughput MTurk experiment.
- 108 concepts,
  - Boolean (circle or red)
  - Context-dependent (“Determiners”)
    (unique largest, exists another with same shape)
- 2 orders per concept,
- 1596 participants.

Piantadosi, Goodman, Tenenbaum (in prep)
• High-throughput MTurk experiment.

• 108 concepts,

• Boolean (circle or red)

• Context-dependent (“Determiners”)
  (unique largest, exists another with same shape)

• 2 orders per concept,

• 1596 participants.

Piantadosi, Goodman, Tenenbaum (in prep)
• **High-throughput MTurk experiment.**

• 108 concepts,

• Boolean (*circle* or *red*)

• Context-dependent (“Determiners”)  
  (*unique largest*, exists another with *same shape*)

• 2 orders per concept,

• 1596 participants.  

Piantadosi, Goodman, Tenenbaum (in prep)
Boolean concepts

- Circle or blue
- Not [circle or blue]
- Size 2
- [Circle or triangle] implies blue
Boolean concepts

Circle or blue

Accuracy

Response number

0.0

0.2

0.4

0.6

0.8

0.0

0.2

0.4

0.6

0.8

FullBoolean grammar

Human

0.96

Not [circle or blue]

Accuracy

Response number

0.0

0.2

0.4

0.6

0.8

0.0

0.2

0.4

0.6

0.8

FullBoolean grammar

Human

0.94

Size 2

Accuracy

Response number

0.0

0.2

0.4

0.6

0.8

0.0

0.2

0.4

0.6

0.8

FullBoolean grammar

Human

0.97

[Circle or triangle] implies blue

Accuracy

Response number

0.0

0.2

0.4

0.6

0.8

0.0

0.2

0.4

0.6

0.8

FullBoolean grammar

Human

0.93
Boolean concepts

Best model performance on Boolean concepts:
Comparing languages

- **DNF**
  disjunctions of conjunctions

- **Horn clauses**
  conjunctions of implications

- **Full boolean**
  any combinations of AND, OR, NOT, IF, IFF

- **Nand**
  combinations of NAND

\[ (\lambda (x) (or (and (red? x) (circle? x)) (and (red? x) (triangle? x)))) \]

\[ (\lambda (x) (and (implies (not (red? x)) false) (implies (not (triangle? x)) (circle? x)))) \]

\[ (\lambda (x) (and (red? x) (or (circle? x) (triangle? x)))) \]

\[ (\lambda (x) (nand false (nand (red? x) (nand false (circle? x)) (nand false (triangle? x))))) \]
Comparing languages

- Fit hyper-parameters (dirichlet on each NT) for each language.
- Evaluated against held out data.

<table>
<thead>
<tr>
<th>Grammar</th>
<th>H.O. LL</th>
</tr>
</thead>
<tbody>
<tr>
<td>FULL BOOLEAN</td>
<td>-16315.27</td>
</tr>
<tr>
<td>CNF</td>
<td>-16333.59</td>
</tr>
<tr>
<td>DNF</td>
<td>-16368.31</td>
</tr>
<tr>
<td>BICONDITONAL</td>
<td>-16385.01</td>
</tr>
<tr>
<td>IMPLIES</td>
<td>-16442.40</td>
</tr>
<tr>
<td>HORNCLAUSE</td>
<td>-16487.25</td>
</tr>
<tr>
<td>SIMPLE BOOLEAN</td>
<td>-16490.51</td>
</tr>
<tr>
<td>NAND</td>
<td>-16902.68</td>
</tr>
<tr>
<td>NOR</td>
<td>-16917.49</td>
</tr>
<tr>
<td>UNIFORM</td>
<td>-19482.72</td>
</tr>
<tr>
<td>EXEMPLAR</td>
<td>-23645.13</td>
</tr>
<tr>
<td>ONLY FEATURES</td>
<td>-31662.08</td>
</tr>
<tr>
<td>RESPONSE-BIASED</td>
<td>-37906.77</td>
</tr>
</tbody>
</table>
Outline

If concepts are probabilistic programs, then concept learning is *probabilistic program induction*.

- Boolean categories
- Quantified concepts
- Natural number concepts
- Generative kinds
- Program induction
Role-governed concepts

“Key”

Goodman, et al. (2007)
Role-governed concepts

“Key”

Goodman, et al. (2007)
Role-governed concepts

“There is no logic to the shape of a key. Its logic is: it turns the lock.” - Chesterton

Goodman, et al. (2007)
Role-governed concepts

“There is no logic to the shape of a key. Its logic is: it turns the lock.” - Chesterton

• Key, poison, passenger...

Goodman, et al. (2007)
Role-governed concepts

“There is no logic to the shape of a key. Its logic is: it turns the lock.” - Chesterton

• Key, poison, passenger...

• Go beyond features to relations and roles.

Goodman, et al. (2007)
Role-governed concepts

“Key”

- Key, poison, passenger...
- Go beyond features to relations and *roles*.
- Extend language by allowing relations and quantifiers.

“There is no logic to the shape of a key. Its logic is: it turns the lock.” - Chesterton

Goodman, et al. (2007)
Role-governed concepts

“Key”

• Key, poison, passenger...

• Go beyond features to relations and roles.

• Extend language by allowing relations and quantifiers.


“There is no logic to the shape of a key. Its logic is: it turns the lock.” - Chesterton

Goodman, et al. (2007)
Role-governed concepts

“Key”

- Key, poison, passenger...
- Go beyond features to relations and roles.
- Extend language by allowing relations and quantifiers.

“There is no logic to the shape of a key. Its logic is: it turns the lock.” - Chesterton

∀x ℓ(x) ⇔ (∃y r_1(x, y) = 1)

Expected weight

Number of observed labels

---

Goodman, et al. (2007)
Non-Boolean concepts

• Big experiment included context-dependent (determiner-like) concepts.

• What languages explain inductive bias for these non-boolean concepts?

Piantadosi, Goodman, Tenenbaum (in prep)
Non-Boolean concepts

- Best language is full boolean plus quantifiers.

<table>
<thead>
<tr>
<th>FOL</th>
<th>One-Or-Fewer</th>
<th>Small-Cardinalities</th>
<th>2nd-Ord.-Quan.</th>
<th>H.O. LL</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>-79279.95</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>-79560.90</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-79642.46</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-79972.75</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-80198.75</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-80267.46</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-80285.38</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-80300.00</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-80614.35</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-80942.77</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-81138.27</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-81289.85</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-81596.68</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-81651.36</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-81773.43</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-81967.68</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-82144.71</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-82219.08</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-82685.21</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-82752.82</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-82853.59</td>
</tr>
</tbody>
</table>
Other work

- Quantifying over objects/features
  (Kemp and Jerns, 2010)

- Learning a relation (by learning a theory)
  (Kemp, Goodman, Tenenbaum, 2008a, 2008b)

- Learning intuitive theories
Outline

If concepts are probabilistic programs, then concept learning is *probabilistic program induction*.

- Boolean categories
- Quantified concepts
- Natural number concepts
- Generative kinds
- Program induction
## Learning number

<table>
<thead>
<tr>
<th>Number knowledge</th>
<th>Age</th>
<th>Level</th>
<th>How many duckies?</th>
</tr>
</thead>
<tbody>
<tr>
<td>No word meanings</td>
<td>&lt;24m</td>
<td>No-knower</td>
<td>Can you give me two duckies?</td>
</tr>
</tbody>
</table>

See, e.g. Spelke 2003; Wynn 1990, 1992
# Learning number

<table>
<thead>
<tr>
<th>Number knowledge</th>
<th>Age</th>
<th>Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>No word meanings</td>
<td>&lt;24m</td>
<td>No-knower</td>
</tr>
<tr>
<td>“one”</td>
<td>24-30m</td>
<td>One-knower</td>
</tr>
</tbody>
</table>

- How many duckies?
  - Can you give me two duckies?

See, e.g. Spelke 2003; Wynn 1990, 1992
# Learning number

<table>
<thead>
<tr>
<th>Number knowledge</th>
<th>Age</th>
<th>Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>No word meanings</td>
<td>&lt;24m</td>
<td>No-knower</td>
</tr>
<tr>
<td>“one”</td>
<td>24-30m</td>
<td>One-knower</td>
</tr>
<tr>
<td>“one”, “two”</td>
<td>30-39m</td>
<td>Two-knower</td>
</tr>
</tbody>
</table>

See, e.g. Spelke 2003; Wynn 1990, 1992

How many duckies?

Can you give me two duckies?
# Learning number

<table>
<thead>
<tr>
<th>Number knowledge</th>
<th>Age</th>
<th>Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>No word meanings</td>
<td>&lt;24m</td>
<td>No-knower</td>
</tr>
<tr>
<td>“one”</td>
<td>24-30m</td>
<td>One-knower</td>
</tr>
<tr>
<td>“one”, “two”</td>
<td>30-39m</td>
<td>Two-knower</td>
</tr>
<tr>
<td>“one”-”three”</td>
<td>39-42m</td>
<td>Three-knower</td>
</tr>
</tbody>
</table>

See, e.g. Spelke 2003; Wynn 1990, 1992
# Learning number

<table>
<thead>
<tr>
<th>Number knowledge</th>
<th>Age</th>
<th>Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>No word meanings</td>
<td>&lt;24m</td>
<td>No-knower</td>
</tr>
<tr>
<td>“one”</td>
<td>24-30m</td>
<td>One-knower</td>
</tr>
<tr>
<td>“one”, “two”</td>
<td>30-39m</td>
<td>Two-knower</td>
</tr>
<tr>
<td>“one”-“three”</td>
<td>39-42m</td>
<td>Three-knower</td>
</tr>
<tr>
<td>All number words</td>
<td>&gt;42m</td>
<td>CP-knower</td>
</tr>
</tbody>
</table>

See, e.g. Spelke 2003; Wynn 1990, 1992

How many duckies?

Can you give me two duckies?
Central questions

- In a way that doesn’t presuppose integers?
- Explaining the abrupt CP-transition?
- What is the role of language?
Learning number

- Sample a **lexicon**: a mapping from situations to descriptions.

- Lexicons expressed in (limited) $\lambda$-calculus plus primitives:
  - Set primitives: difference, union, select, singleton?, doubleton?, ...
  - Count-list operations: `prev / next` move between *words* on the list.
  - Recursion: $(\_ \_ \_)$.
  - *if, and, ...*

Piantadosi, Tenenbaum, Goodman (subm.)
Learning number

A two-knower lexicon:

(define L
  (λ (S)
    (if (singleton? S)
      "one"
      (if (doubleton? S) "two" undef)))
Learning number

A two-knower lexicon:

\[
\begin{align*}
\text{(define } & L \\
& (\lambda \ (S) \\
& \quad \text{(if } (\text{singleton? } S) \\
& \quad \quad \text{"one"} \\
& \quad \quad \text{(if } (\text{doubleton? } S) \text{ "two" undefined)}) \\
& \end{align*}
\]

A CP-knower lexicon:

\[
\begin{align*}
\text{(define } & L \\
& (\lambda \ (S) \\
& \quad \text{(if } (\text{singleton? } S) \\
& \quad \quad \text{"one"} \\
& \quad \quad \text{(next } \text{(L \ (set-difference } S \text{ \ (select } S)))\text{)))}
\end{align*}
\]
Learning number

- Large space of hypotheses contains many potentially useful lexica, as well as very silly ones.
Learning number

- Large space of hypotheses contains many potentially useful lexica, as well as very silly ones.

For example: a ‘mod 5’ lexicon:

```scheme
(define L
  (λ (S)
    (if (or (singleton? S)
             (equal? (L (set-diff S (select S)))
                              "five")
             "one"
        (next (L (set-diff S (select S)))))))))
```

Learning number

Hypotheses

Data

Learning data: number words paired with sets of objects (frequency of words matches CHILDES corpus).

Frequency of word + noun
Learning number

Figure 3 shows marginal posterior probability of exhibiting each type of behavior, as a function of amount of data. Figure 3b shows the same plot on a log y-axis demonstrating the large number of other numerical systems which are considered, but found to be unlikely given the data. For instance, the 2-knower line shows the sum of the posterior probability of all LOT expressions which map sets of size 1 to "one", sets of size 2 to "two", and everything else to undef. Intuitively, this marginal probability corresponds to the proportion of children who should look like subset- or CP-knowers at each point in time. This figure shows that the model exhibits the correct developmental pattern, first learning the meaning of "one", then "two", "three", and finally transitioning to a CP-knower who knows the correct meaning of all number words. That is, with very little data the "best" hypothesis is one which looks like a 1-knower, and as more and more data is accumulated, the model transitions through subset-knowers. Eventually, the model accumulates enough evidence to justify the CP-knower lexicon that recursively defines all number words on the count list. At that point, the model exhibits a conceptual re-organization, changing to a hypothesis in which all number word meanings are defined recursively as in the CP-knower lexicon in Table 1.
Figure 3: Figure 3a shows marginal posterior probability of exhibiting each type of behavior, as a function of amount of data. Figure 3b shows the same plot on a log y-axis demonstrating the large number of other numerical systems which are considered, but found to be unlikely given the data.

For instance, the 2-knower line shows the sum of the posterior probability of all LOT expressions which map sets of size 1 to "one", sets of size 2 to "two", and everything else to undef. Intuitively, this marginal probability corresponds to the proportion of children who should look like subset- or CP-knowers at each point in time. This figure shows that the model exhibits the correct developmental pattern, first learning the meaning of "one", then "two", "three", and finally transitioning to a CP-knower who knows the correct meaning of all number words. That is, with very little data the "best" hypothesis is one which looks like a 1-knower, and as more and more data is accumulated, the model transitions through subset-knowers. Eventually, the model accumulates enough evidence to justify the CP-knower lexicon that recursively defines all number words on the count list. At that point, the model exhibits a conceptual re-organization, changing to a hypothesis in which all number word meanings are defined recursively as in the CP-knower lexicon in Table 1.
Outline

If concepts are probabilistic programs, then concept learning is *probabilistic program induction*.

- Boolean categories
- Quantified concepts
- Natural number concepts
- Generative kinds
- Program induction
The complexity of real-world categories—using probabilistic programs at different levels of abstraction in the representation of events, such as a soccer match with its structured concepts, can be found in language.

Concept learning has traditionally been studied in the context of languages and learning these programs from data. The main advantage of using probabilistic programs is that they allow for handling models with non-tree-like topologies.

Consider the design of a castle, composed of a set of large towers and small towers connected by walls in a circular topology. This has semantic significance; the field of view and therefore the tactical effectiveness of towers occur at relatively straight sections.

The layout of chairs in a conference room; there are several rows of chairs. As shown in Figure 0, each castle generated satisfies the constraints that are hard to satisfy through grammar-based methods. The multi-stage factor graph, which is context sensitive, allows for constraint specification using the most perspicuous representation.

In grammar-based systems, a model is the interpretation of a string of tokens, while in procedural modeling, it is one of the earliest grammar-based methods. The L-system [Lindenmayer, 1968] used for procedural modeling of plants is one of the earliest grammar-based systems. As shown in Figure 1, this system allows for the dynamic constraint between type and position using declarative and procedural programming.

To express the dynamic constraint between type and position using declarative programming, we used two stages to synthesize the result: first sample the number of towers, and then assign types of adjacent towers. In the second stage, we can choose the type of each tower after them in stage one. This is illustrated in Figure 2.

To recover the circular topology, we must consider the type of each tower after them in stage one. For example, constraints between towers are context sensitive constraints whose application depends on tower positions. We used two stages to synthesize the result: sample the number of towers, and then assign types of adjacent towers. In the second stage, we can choose the type of each tower after them in stage one.

This allows the vine grammar to smoothly curve, but we cannot satisfy these constraints. Therefore, we used two stages to synthesize the result: first sample the number of towers, and then assign types of adjacent towers. In the second stage, we can choose the type of each tower after them in stage one. This is illustrated in Figure 2.
Learning generative kinds

Simple prototype

Sub-concepts

Single recursion

Prototype

Concept Type

Example Program and Observations

Table 1: Taxonomy of Generative Concepts. For each concept type, an example program and observations drawn from this study are shown.

- **Single recursion**
  - Example program: 
    ```lambda ()
    (node 'a
    (node 'b (node 'c (node 'c)) (node 'b))
    (node 'd (node 'e))))
    ```
  - Observations: The assumption that the true generating program is close to the model score estimation process. The model and human judgment is what an ideal observer that knows the generating grammar would infer given the training examples shown to the subjects may not be correct. Subconcepts with arguments are not part of the model may nonetheless have had an influence on the subjects' responses. Noise in the model score estimation process. The model and human judgment is what an ideal observer that knows the generating grammar would infer given the training examples shown to the subjects. Misleading visual representation of the stimuli. The colors may not correspond to the model and human judgment is what an ideal observer that knows the generating grammar would infer given the training examples shown to the subjects. Different background assumptions. The tree structure and the color cues may not correspond to the model and human judgment is what an ideal observer that knows the generating grammar would infer given the training examples shown to the subjects. Inference over programs with lambda abstraction requires the model to compare trees of the same type even though they are not part of the model. Subconcepts with arguments are not part of the model may nonetheless have had an influence on the subjects' responses.

- **Multiple recursion**
  - Example program: 
    ```begin
    (define (part) (node 'c (node 'c (node 'a)) (node 'c)))
    (lambda () (node 'a (node 'b (if (flip) (node 'c) (part))) (node 'd (part)) (node 'b (part) (node 'd)))))))
    ```
  - Observations: The assumption that the true generating program is close to the model score estimation process. The model and human judgment is what an ideal observer that knows the generating grammar would infer given the training examples shown to the subjects. Noise in the model score estimation process. The model and human judgment is what an ideal observer that knows the generating grammar would infer given the training examples shown to the subjects. Misleading visual representation of the stimuli. The colors may not correspond to the model and human judgment is what an ideal observer that knows the generating grammar would infer given the training examples shown to the subjects. Different background assumptions. The tree structure and the color cues may not correspond to the model and human judgment is what an ideal observer that knows the generating grammar would infer given the training examples shown to the subjects. Inference over programs with lambda abstraction requires the model to compare trees of the same type even though they are not part of the model. Subconcepts with arguments are not part of the model may nonetheless have had an influence on the subjects' responses.

- **Nested prototypes**
  - Example program: 
    ```begin
    (define (part) (node 'b (if (flip) (node 'c) (part))))
    (lambda () (node 'c (node 'd (node 'f (node 'f)) (node 'f))) (node 'd (node 'e) (node 'f (node 'f)) (node 'f)))))))
    ```
  - Observations: The assumption that the true generating program is close to the model score estimation process. The model and human judgment is what an ideal observer that knows the generating grammar would infer given the training examples shown to the subjects. Noise in the model score estimation process. The model and human judgment is what an ideal observer that knows the generating grammar would infer given the training examples shown to the subjects. Misleading visual representation of the stimuli. The colors may not correspond to the model and human judgment is what an ideal observer that knows the generating grammar would infer given the training examples shown to the subjects. Different background assumptions. The tree structure and the color cues may not correspond to the model and human judgment is what an ideal observer that knows the generating grammar would infer given the training examples shown to the subjects. Inference over programs with lambda abstraction requires the model to compare trees of the same type even though they are not part of the model. Subconcepts with arguments are not part of the model may nonetheless have had an influence on the subjects' responses.

Stuhlmüller, Tenenbaum, Goodman (2010)
Learning generative kinds

<table>
<thead>
<tr>
<th>Prototype</th>
<th>Nested Prototype</th>
<th>Parts</th>
<th>Parameterized Parts</th>
<th>Single Recursion</th>
<th>Multiple Recursion</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="prototype.png" alt="Diagram" /></td>
<td><img src="nested_prototype.png" alt="Diagram" /></td>
<td><img src="parts.png" alt="Diagram" /></td>
<td><img src="parameterized_parts.png" alt="Diagram" /></td>
<td><img src="single_recursion.png" alt="Diagram" /></td>
<td><img src="multiple_recursion.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Set GCM</th>
<th>Transition GCM</th>
<th>Tree GCM</th>
<th>Generative Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prototype</td>
<td>0.589</td>
<td>0.751</td>
<td>0.803</td>
<td>0.748</td>
</tr>
<tr>
<td>Nested Prototype</td>
<td>0.544</td>
<td>0.851</td>
<td>0.937</td>
<td>0.904</td>
</tr>
<tr>
<td>Parts*</td>
<td>0.320</td>
<td>0.617</td>
<td>0.705</td>
<td>0.835</td>
</tr>
<tr>
<td>Parameterized Parts</td>
<td>0.298</td>
<td>0.591</td>
<td>0.778</td>
<td>0.911</td>
</tr>
<tr>
<td>Single Recursion</td>
<td>0.284</td>
<td>0.499</td>
<td>0.637</td>
<td>0.773</td>
</tr>
<tr>
<td>Multiple Recursion</td>
<td>0.505</td>
<td>0.561</td>
<td>0.451</td>
<td>0.770</td>
</tr>
</tbody>
</table>
If concepts are probabilistic programs, then concept learning is *probabilistic program induction*.

- Boolean categories
- Quantified concepts
- Natural number concepts
- Generative kinds
- Program induction
Algorithms for induction

• What algorithms are capable of learning concepts in a language of thought?

• All results so far were computed using MCMC based on constituent regeneration (Goodman, et al, 2008).

• Is this cognitively plausible? Maybe...

• But this is probably not enough on its own.

(Ullman, Goodman, Tenenbaum, 2010)
Program induction is especially hard. How could it be done?

Idea: syntactic analogy + argument compression (+ search/MCMC).

- "inverse inlining":
  \[
  \frac{1 + (2 \times 3)}{1 + (5 \times 3)}
  \]

- "de-argument":
  \[
  \frac{\text{define } (F \ x) \ (1 + (x \times 3))}{\text{define } (F) \ (+ 1 \ (* \ (\text{gaussian} \ 3.5 \ 1.0) \ 3))}
  \]

- "de-argument":
  \[
  \text{define } (F \ x) \ (+ 1 \ (* \ x \ 3))
  \]

  \[
  \text{if} \ (\text{flip}) \ (+ 1 \ (* \ (F) \ 3)) \ 1
  \]
Program induction

- Program induction is especially hard. How could it be done?
- Idea: syntactic analogy + argument compression (+ search/MCMC).

```
(define (F x) (+ 1 (* x 3)))
(/ (F 2) (F 5))
```

"inverse inlining"

```
(define (F)
  (+ 1 (* (gaussian 3.5 1.0) 3)))
(/ (F) (F))
```

"de-argument"

```
(define (F x)
  (+ 1 (* x 3)))
(define (F (F 1))
```

"de-argument"
Program induction

- Program induction is especially hard. How could it be done?
- Idea: syntactic analogy + argument compression (+ search/MCMC).

```
(define (F x) (+ 1 (* x 3)))
(/ (F 2) (F 5))
```

"inverse inlining"

```
(define (F)
  (+ 1 (* (gaussian 3.5 1.0) 3)))
(/ (F) (F))
```

"de-argument"

```
(define (F x)
  (+ 1 (* x 3)))
(F (F 1))
```

"de-argument"
Program induction

- Program induction is especially hard. How could it be done?
- Idea: syntactic analogy + argument compression (+ search/MCMC).

```
(define (F x)
  (+ 1 (* x 3)))
(define (F)
  (+ 1 (* (gaussian 3.5 1.0) 3)))
(define (F)
  (if (flip) (+ 1 (* (F) 3)) 1))
```

```
+/ (+ 1 (* 2 3))
(+ 1 (* 5 3))
```

```
`` inverse inlining
``
```
``
```
``
```
`` de-argument
``
```
```
``
```
`` de-argument
``
```
```
``
```
``
```
``
```
``
```
``
```
``
```
``
```
``
```
``
```
``
```
``
```
``
```
``
```
``
```
``
```
``
```
``
```
``
```
``
```
Program induction

• Program induction is especially hard. How could it be done?

• Idea: syntactic analogy + argument compression (+ search/MCMC).

\[
\text{Program} \quad \frac{+ 1 \times 2 \times 3}{+ 1 \times 5 \times 3} \\
\text{Inverse inlining} \quad (\text{define } (F \ x) \ (+ 1 \times (x \times 3))) \quad \frac{(F \ 2) \ (F \ 5)}{(F \ 2) \ (F \ 5)}
\]

\[
\text{de-argument} \quad (\text{define } (F \ x) \ (+ 1 \times (\text{gaussian } 3.5 \times 1.0) \times 3)) \quad \frac{(F) \ (F \ F)}{(F) \ (F \ F)}
\]

\[
\text{de-argument} \quad (\text{define } (F) \ (+ 1 \times ((F) \times 3))) \quad \frac{(F \ (F \ 1))}{(F \ (F \ 1))}
\]
**Program induction**

- **Bayesian program merging algorithm:**
  (Cf. Stolcke & Omohundro, 1994)
- Initial state is exemplar program (mixture of data).
- Transform programs via syntactic analogy and argument compression.
- Beam search* with respect to the posterior score.
  (*Or stochastic search, Monte Carlo, etc.)
- Likelihood: marginal probability of data given program (computed by lazy particle filter).
- Prior: syntactic complexity.

Hwang, Stuhlmueller, Goodman (in prep)
Example

"data incorporation"

(define (F x) (node r (node b) x))
(F (F (F (F b)))))

"inverse inline"

(define (F) (if (flip 0.8)
    (node r (node b) (F))
    b))
(F)

"de-argument"

sample
Example

\[
\text{(begin (define F4 (lambda (V7 V8) (node (F1 V7 0.1) V8)))}
\]
\[
\text{(define F3 (lambda (V6) (node (F1 V6 0.3)))))}
\]
\[
\text{(define F2}
\]
\[
\text{(lambda (V3 V4))}
\]
\[
\text{((lambda (V5) (F4 V3 (F4 V4 V5)))}
\]
\[
\text{(if (flip 9/11)}
\]
\[
\text{(F2 81.0 85.0)}
\]
\[
\text{(uniform-choice}
\]
\[
\text{(node (F1 204.0 0.3) (F3 199.0)}
\]
\[
\text{(F3 243.0) (F3 233.0) (F3 240.0)))}
\]
\[
\text{(F4 151.0)}
\]
\[
\text{(node (F1 -21.0 0.3) (F3 7.0)}
\]
\[
\text{(F3 49.0) (F3 3e1) (F3 44.0))))))))}
\]
\[
\text{(define F1}
\]
\[
\text{(lambda (V1 V2))}
\]
\[
\text{(data (color (gaussian V1 25)) (size V2))})
\]
\[
\text{(lambda ()}
\]
\[
\text{(uniform-choice}
\]
\[
\text{(node (F1 13.0 1) (F2 89.0 111.0)}
\]
\[
\text{(F2 85.0 121.0))))}}
\]
Analogy and number?

• In simple MCMC over a rich concept space some transitions are very unlikely.

• E.g. N-knower => CP-knower.
Analogy and number?

• In simple MCMC over a rich concept space some transitions are very unlikely.

• E.g. N-knower => CP-knower.

• Proposals via Syntactic analogy + recursion detection?
Analogy and number?

• In simple MCMC over a rich concept space some transitions are very unlikely.

• E.g. N-knower => CP-knower.

• Proposals via Syntactic analogy + recursion detection?

A three-knower lexicon:

```
(define lexicon
 (λ (word)
   (case word
     ("one" (λ (set)
       (empty? (diff set (select set)))))
     ("two" (λ (set)
       ((lexicon "one")
         (diff set (select set))))
     ("three" (λ (set)
       ((lexicon "two")
         (diff set (select set))))
     (else null)))
```
Analogy and number?

- In simple MCMC over a rich concept space some transitions are very unlikely.
- E.g. N-knower => CP-knower.
- Proposals via Syntactic analogy + recursion detection?

A three-knower lexicon:

```
(define lexicon
  (λ (word)
    (case word
      ("one" (λ (set)
                (empty? (diff set (select set))))
      ("two"  (λ (set)
                 ((lexicon "one")
                  (diff set (select set)))
      ("three" (λ (set)
                 ((lexicon "two")
                  (diff set (select set)))
      (else null)))
```
Analogy and number?

- In simple MCMC over a rich concept space some transitions are very unlikely.
- E.g. N-knower => CP-knower.
- Proposals via Syntactic analogy + recursion detection?

A CP-knower lexicon:

```
(define lexicon
  (λ (word)
    (if (= word "one")
      (λ (set)
        (empty? (diff set (select set))))
      (λ (set)
        ((lexicon (prev word))
          (diff set (select set)))))))

("three" (λ (set)
            ((lexicon "two")
              (diff set (select set))))
      (else null)))
```
Analogy and number?

- In simple MCMC over a rich concept space some transitions are very unlikely.
- E.g. N-knower => CP-knower.
- Proposals via Syntactic analogy + recursion detection?

A CP-knower lexicon:

```
(define lexicon
  (λ (word)
    (if (= word "one")
      (λ (set)
        (empty? (diff set (select set))))
      (λ (set)
        ((lexicon (prev word))
          (diff set (select set)))))))

("three" (λ (set)
            ((lexicon "two")
             (diff set (select set))))
      (else null))
```
Outline

If concepts are probabilistic programs, then concept learning is *probabilistic program induction*.

- Boolean categories
- Quantified concepts
- Natural number concepts
- Generative kinds
- Program induction
An evolving conceptome

• Concepts (and theories) are represented in the PLoT, induced in response to evidence.

• Learned concepts become ‘effective primitives’ for new concepts.
Role of existing concepts

- 8 concepts all have same “logical” structure.
- In Number domain people have pre-existing concepts that are useful to varying degrees.
- Learning can be enhanced or retarded by existing concepts.

<table>
<thead>
<tr>
<th>Label</th>
<th>@($x$) when $x$ is:</th>
<th>@($x$) when $x$ is:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1   2   3   4   5   6   1   2   3   4   5   6</td>
<td></td>
</tr>
<tr>
<td>+7</td>
<td>8   9   10  11  12  13  ♡  ♠  ♦  ♦  ♣  ♣</td>
<td></td>
</tr>
<tr>
<td>14−</td>
<td>13  12  11  10  9   8   ♣  ♣  ♣  ♣  ♣  ♣</td>
<td></td>
</tr>
<tr>
<td>+8*</td>
<td>9   10  11  12  13  8   ♣  ♣  ♣  ♣  ♣  ♣</td>
<td></td>
</tr>
<tr>
<td>mem</td>
<td>12  9   13  10  11  8   ♣  ♣  ♣  ♣  ♣  ♣</td>
<td></td>
</tr>
</tbody>
</table>

Ouyang & Goodman (in prep)
An evolving conceptome

• Concepts (and theories) are represented in the PLoT, induced in response to evidence.

• Learned concepts become ‘effective primitives’ for new concepts.

• The evidence comes from perception, language, and social interaction.

• Rich trajectories of conceptual change emerge.
Conclusion

Probabilistic inference

Generative models

Compositional representations

Probabilistic language of thought

Conclusions