Learning Structured Generative Concepts

The MIT Faculty has made this article openly available. Please share how this access benefits you. Your story matters.

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>As Published</td>
<td><a href="http://toc.proceedings.com/09137webtoc.pdf">http://toc.proceedings.com/09137webtoc.pdf</a></td>
</tr>
<tr>
<td>Publisher</td>
<td>Cognitive Science Society</td>
</tr>
<tr>
<td>Version</td>
<td>Author's final manuscript</td>
</tr>
<tr>
<td>Accessed</td>
<td>Sat Mar 09 00:28:53 EST 2019</td>
</tr>
<tr>
<td>Citable Link</td>
<td><a href="http://hdl.handle.net/1721.1/112758">http://hdl.handle.net/1721.1/112758</a></td>
</tr>
<tr>
<td>Terms of Use</td>
<td>Creative Commons Attribution-Noncommercial-Share Alike</td>
</tr>
<tr>
<td>Detailed Terms</td>
<td><a href="http://creativecommons.org/licenses/by-nc-sa/4.0/">http://creativecommons.org/licenses/by-nc-sa/4.0/</a></td>
</tr>
</tbody>
</table>
Learning Structured Generative Concepts

Andreas Stuhlmüller, Joshua B. Tenenbaum, Noah D. Goodman
Brain and Cognitive Sciences, MIT
{ast, jbt, ndg}@mit.edu

Abstract

Many real world concepts, such as “car”, “house”, and “tree”, are more than simply a collection of features. These objects are richly structured, defined in terms of systems of relations, subparts, and recursive embeddings. We describe an approach to concept representation and learning that attempts to capture such structured objects. This approach builds on recent probabilistic approaches, viewing concepts as generative processes, and on recent rule-based approaches, constructing concepts inductively from a language of thought. Concepts are modeled as probabilistic programs that describe generative processes; these programs are described in a compositional language. In an exploratory concept learning experiment, we investigate human learning from sets of tree-like objects generated by processes that vary in their abstract structure, from simple prototypes to complex recursions. We compare human categorization judgments to predictions of the true generative process as well as a variety of exemplar-based heuristics.

Introduction

Concept learning has traditionally been studied in the context of relatively unstructured objects that can be described as collections of features. Learning and categorization can be understood formally as problems of statistical inference, and a number of successful accounts of concept learning can be viewed in terms of probabilistic models defined over different ways to represent structure in feature sets, such as prototypes, exemplars, or logical rules (Anderson, 1990; Shi, Feldman, & Griffiths, 2008; Goodman, Tenenbaum, Feldman, & Griffiths, 2008). Yet for many real world object concepts, such as “car”, “house”, “tree”, or “human body”, instances are more than simply a collection of features. These objects are richly structured, defined in terms of features connected in systems of relations, parts and subparts at multiple scales of abstraction, and even recursive embeddings (Markman, 1999). A tree has branches coming out of a trunk, with roots in the ground; branches give rise to smaller branches, and there are leaves at the end of the branches. A human body has a head on top of a torso; arms and legs come out of the torso, with arms ending in hands, made of fingers. A house is composed of walls, roofs, doors, and other parts arranged in characteristic functional and spatial relations that are harder to verbalize but still easy to recognize and reason about. Besides objects, examples of structured concepts can be found in language (e.g. the mutually recursive system of phrase types in a grammar), in the representation of events (e.g. a soccer match with its fixed subparts), and processes (e.g. the recipe for making a pancake with steps at different levels of abstraction).

Such concepts have not been the focus of research in the probabilistic modeling tradition. Here we describe an approach to representing structured concepts—more typical of the complexity of real world categories—using probabilistic generative processes. We test whether statistical inference with these generative processes can account for how people categorize novel instances of structured concepts and compare with more heuristic, exemplar-based approaches.

Because a structured concept like “house” has no single, simple perceptual prototype that is similar to all examples, learning such a concept might seem very difficult. However, each example of a structured concept itself has internal structure which makes it potentially very informative. Consider figure 1, where from only a few observations of a concept it is easy to see the underlying structural regularity that can be extended to new items. The regularities underlying structured concepts can often be expressed with instructions for generating the examples: “Draw a sequence of brown dots, choose a branch color, and for each brown dot draw two dots of this color branching from it.”

![Figure 1: Three examples of a structured concept described by a simple generative process.](image-url)
than deterministic rules, our concepts denote distributions. Finally, the probabilistic program approach can be seen as a generalization of previous approaches to generative representations of concepts (Kemp, Bernstein, & Tenenbaum, 2005; Rehder & Kim, 2006; Feldman, 1997).

We investigate human learning for classes of generating processes that vary in their abstract structure, from simple prototypes to complex multiply recursive programs. We compare predictions for categorization judgments based on the true generative model to the predictions of exemplar models, which exploit the relational structure of the examples to varying degrees but cannot detect more abstract structure. We find two regimes: for concepts with simple prototype-like structures, human judgements are well described by a relational exemplar model, but humans can also easily learn more abstract regularities—such as sub-concepts and recursion—which are better captured by a model using more expressive generative descriptions based on probabilistic programs.

**Formal Framework**

In the following, we first explain the formal language we use to describe generative processes, then the different methods of categorization (or generalization) we compare to subjects’ judgements.

**Concept Representation**

We analyze concepts as generative models, i.e. as formal descriptions of processes that generate observations. We do so within a simple domain where we can fully know and manipulate the actual generating processes behind complex objects. We use tree-structured graphs with colored nodes as observations in our experiments—these are a simple proxy for many real-world concepts, where the dependencies among parts are hierarchical or tree-like. Human bodies, buildings, and events all consist of parts that themselves contain parts, with each part standing in interesting relation to the others.

We represent these trees as nested lists: each list denotes a tree, with the first element in the list specifying the color of the root node and the remaining elements describing the children of this node, each child itself being a list (tree). For example, the second tree shown in figure 1 can be represented as 

\[
\text{(((node \ (node \ (node \ \text{false})))) (node \ \text{false}) (node \ \text{false}))}.
\]

We formalize the processes that generate these observations using a subset of Church, a Lisp-like stochastic programming language\(^1\) (Goodman, Mansinghka, Roy, Bonawitz, & Tenenbaum, 2008). Programs in Church describe processes that produce values; running a program corresponds to generating a value from such a process. Because Church contains primitive functions that randomly choose from a distribution on values (e.g. the function \(\text{flip}\) that randomly chooses \(\text{true}\) or \(\text{false}\)), Church programs describe stochastic processes. The meaning of a Church program is a distribution on return values—which may be complex values such as nested lists—and any given execution results in a sample from this distribution. In what follows we describe Church programs which sample colored trees.

We group generative models into classes by the abstract constructions they use. Table 1 illustrates each of these types using a single concept program and observations drawn from this program. The simplest tree-generating processes in our language use only the stochastic function \(\text{node}\), which takes as its first argument a color symbol and as its remaining arguments subtrees. With high probability, \(\text{node}\) returns a tree that has the given color symbol at its root and the given subtrees as its children, but with some probability \(\varepsilon\), it switches to a noise process that can return any tree, that is, \(\text{node}\) introduces a random noise process into the tree construction. Under the noise process, the number of children for a node is sampled from a geometric distribution with parameter \(\varepsilon\) and the node color is sampled uniformly.

Programs like 

\[
\text{((node \ (node \ (node \ \text{false})) \ (node \ \text{false}))}
\]

...denote stochastic prototypes. They are most likely to generate the tree that corresponds to the given colors, in this case 

\[
\text{((\text{false} ) (\text{false} ) (\text{false} ))},
\]

... but they can return any tree with a certain probability. The more a tree deviates from the prototype, the less likely this process is to generate it. For example, the simple program described above could switch at the third \(\text{node}\) to the noise process and produce 

\[
\text{(((node \ (node \ \text{false})) (node \ \text{false}))}
\]

... instead of the prototype. By introducing the noise process, \(\text{node}\) turns a deterministic prototype into a stochastic process.

All of the more abstract ways of formalizing generative models in our tree domain compose these basic processes. Nested prototypes formalize the intuition that a concept or a part of a concept can be “either this or that”. Running the program \((\text{if (flip} .5) \ (\text{node} \ \text{false}) \ (\text{node} \ \text{false}))\) will flip a fair coin and return a sample from \((\text{node} \ \text{false})\) with probability .5, otherwise a sample from \((\text{node} \ \text{false})\).

One of the central reasons for analyzing concepts as represented in a language of thought is that they compose analogously to the components of natural and artificial languages—parts similarly allow composition through reuse in our domain. A part concept is defined first and can then be used in arbitrarily many places within other concepts. For example, the program \((\text{define (part)} \ (\text{node} \ (\text{node} \ \text{false})))\) names a simple part consisting of only two nodes. This part can now be reused in other concepts. For example, the most likely return value for \((\text{node} \ \text{false} \ (\text{part} \ (\text{part})))\) is 

\[
\text{((\text{false} ) (\text{false} ) (\text{false} ))}.
\]

When parts are defined, they are available to the noise process. This leads to some invariance to the position of parts and captures the idea that a generating process may give rise to observations that contain a part in a different place, although with lower probability compared to an observation with the part in the correct place.

Parameterized parts can capture both deterministic structure and random choices and reuse them in multiple places. When a part like 

\[
(\text{define (part} x) \ (\text{node} \ x \ x))
\]

is used, for example in the program \((\text{part} \ (\text{node} \ \text{false}))\), it evaluates

---

\(^1\)Church uses prefix notation, i.e. function application is written with the operator first, the operands following. For example, \((\text{node} \ x y)\) means that the function \(\text{node}\) is called with the arguments \(x\) and \(y\).
the body of the part—here (node • x x)—with x assigned to its argument, here (node •). Evaluating the program (part [node •]) is therefore most likely to result in the observation '(* •)(* •).'

Allowing parts to call themselves introduces recursion, a means to capture a large amount of repetitive observed structure in a single short definition. For example, the part (define (p) (if (flip) (node •) (node • (p)))) can generate arbitrarily deep lists of single blue nodes, with shorter ones being more likely.

The power of these program constructs is that they may be used compositionally to build more complex concepts, such as those shown in table 1.

**Categorization**

In order to model generalization and categorization behavior of human subjects, we need not only a way to represent concepts, but also a way to compute the probability of any given observation belonging to a known concept. We analyze our experimental results using four models that differ in how much they make use of representational structure.

On the unstructured end of the scale, we use a model that computes generalization judgments solely by comparing the fraction of nodes that have a given color. On the other end of the scale, a generative Bayesian model uses the likelihood under the true generative process to judge category membership. In between, an exemplar model makes use of tree structure in the observations, but not of the more abstract generative process that led to the observations.

**Generative Model**

In modeling concept learning as Bayesian program induction, we follow the approach taken by Goodman, Tenenbaum, et al. (2008). Since we formalize concepts as probabilistic programs, the likelihood \( P(O|C) \) of an observation \( O \) under a given concept \( C \) corresponds to the probability of the program making its random choices such that it returns the observation as its value (see Goodman, Mansinghka, et al. (2008)). The posterior probability of a concept \( C \) given observations \( O \) is proportional to this likelihood multiplied by the prior:

\[
P(C|O) \propto P(O|C)P(C)
\]

In the last section, we described a language for programs which generate trees; a prior \( P(C) \) could be derived from this language, as in Goodman, Tenenbaum, et al. (2008). An ideal learner would then infer the posterior distribution \( P(C|O) \) over concepts \( C \) given the observation \( O \) and make predictions about whether a new observation \( t \) belongs to the category of the observed objects using each concept \( C \in C \) in proportion to its posterior probability:

\[
P(t|O) \propto \sum_C P(t|C)P(C|O)
\]

In order to make computational modeling tractable, we make the simplifying assumptions that (1) subjects’ reasoning is dominated by the maximum a posteriori (MAP) estimate of this distribution, i.e. by the single concept that has the highest posterior probability and that (2) the true generating concept \( C_{true} \) is a good approximation to the MAP estimate. Thus, for each of the concept types we investigate, we model subjects’ behavior using the program from which the training data was sampled. The likelihood of a new observation \( t \) belonging to this concept is simply \( P(t|C_{true}) \) which we compute using an adaptive importance sampling algorithm.

We do not claim that subjects necessarily identify the true generating concept from a few examples; this approximation is made for computational tractability. The full Bayesian model, which maintains uncertainty over generating concepts, can make different predictions in certain cases, but it is not clear whether this represents a bias for or against the approximation—to the extent that people remain uncertain of the concept after a few examples, the Bayesian model would capture human inferences better than our approximation.

**Tree Exemplar Model**

This and the next two models are versions of the exemplar-based generalized context model (GCM) (Nosofsky, 1986). For observations \( O_1, \ldots, O_n \) from category \( C \) and a new observation \( t \) for which we would like to estimate the likelihood under category \( C \), we use

\[
P(t \in C|O_1, \ldots, O_n) \propto \sum_{C \in C} e^{-d(O, t)}
\]

where \( d \) is a distance measure that is sensitive to the tree structure of the observations. Starting from the root node, this measure matches the trees as much as possible, incrementing by 1 for each node that differs in color between the two trees and for each node that must be generated because it exists in one tree but not in the other tree. This approach is similar to the structure mapping approach used by Tomlinson and Love (2006).

**Frequency-based Exemplar Models**

As in the tree exemplar model, we use a distance measure \( d \) to estimate the likelihood of an observation belonging to a category for which we have only positive examples. In this version of the model, \( d(t_1, t_2) \) is the RMSE between the transition count vectors of \( t_1 \) and \( t_2 \). For each pair of node colors, the transition count vector contains the number of times this pair occurs adjacent (as parent-child) in the given tree. We call this model Transition GCM. We also investigate a simplified version that uses the distance between the color count vectors. The length of this vector corresponds to the number of possible node colors, with each entry in the vector denoting how often this node color appears in the tree of interest. We call this Set GCM.

**Experiment**

This experiment is an exploratory investigation into generalization from observations of structured objects. Since our main goal in this study is to investigate the representation of concepts and their use for categorization and generalization rather than the memory aspects of learning, we use a paradigm that minimizes memory demands. By doing so, we hope to focus on how people represent the commonalities between observed instances of a concept and how they use this knowledge to generalize to new instances. We chose a
domain that both contains observations with simple structure and allows for interesting generative processes—the domain of colored trees generated by probabilistic programs.

**Methods**

**Participants** 250 members of Amazon’s crowdsourcing service Mechanical Turk took part in the online experiment. Subjects were compensated for participation.

**Stimuli** Subjects were told that they are looking at newly discovered kinds of plants that grow in extreme environments. Each subject saw 18 pages, with each page consisting of 15 training examples, a control question, and a test example together with a classification question. Both training and test examples were images of simple trees with colored nodes drawn from tree-generating programs (see e.g. table 3). For each of the concept types shown in table 1, there were three tree-generating programs, and for each program there were 7 test examples. These test examples were chosen to cover a wide range of both intuitive and model judgements of category membership. Both training example order and stimuli colors were randomized.

**Procedure** In order to ensure that subjects process the training stimuli, a control question on each page asked how many of the training trees consist of more than 7 dots. 55 subjects answered less than 13 out of the 18 control questions correctly within an error margin of 2. We did not include these subjects in the analysis.

The categorization question asked: “How likely is it that the following plant is the same kind of plant as the plants above?” Subjects chose on a seven-step scale ranging from “certainly the same kind” to “certainly not the same kind”. For each subject, the responses were normalized to $[0, 1]$.

**Results**

Table 2 summarizes the correlation results for all models. Figure 2 shows for each concept type human results and model results for both the exemplar and generative model. For each concept type, three different concepts were part of the experiment, and for each concept, seven different test observations were shown. A single point in the scatterplot contains information on the mean subject response for a single test tree and on the model prediction for this tree.

Neither of the two exemplar models based on simple stas-
Six concept types, three examples were shown; the color of the dots indicates to which example any given datapoint belongs. Empty circles which maintain a distribution over generative processes, may that the learner infers a single generating concept from the ex-
ter recursions, and multiple recursions as "more structured", then
ected different concept types under the two models that make use of tree structure. Both results are ex-
ated part cases (see text).
Table 2: Human-model correlations for the experiment. Each row shows how well the different models predicted subjects’ perfor-
ance for a particular concept type. *Correlations excluding iso-
ated part cases (see text).

<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>
parameter arguments in the observations.

For the single recursion example in table 3, changing the color of a few nodes within the recursion results in a significantly lower generalization. At the same time, a very similar manipulation does not result in a significant change in the generalization rating for the multiple recursion example. Intuitively, we sometimes see a change as destroying a very obvious pattern structure whereas at other times, the change in structure is not assumed to be relevant. Future research needs to characterize when subjects infer that such a pattern exists, and when they instead assume coincidence.

The comparison between the frequency based exemplar models and the two models that rely on tree structure in the observations makes clear that subjects do make use of the fact that the observations are structured in their generalization judgements. Furthermore, comparing the tree exemplar model to the true generative model that makes use of more abstract structure hints at the possibility that subjects are relying on recursive structure in the observations. The individual response patterns in the results of our exploratory experiment highlight ways in which both the exemplar-based model and the generative model can be improved to more closely reflect human generalization patterns.

**Conclusion**

Most studies of concept learning have focused on relatively unstructured objects based on simple features. We have suggested viewing concepts as probabilistic programs that describe stochastic generative processes for more structured objects. In this view concepts denote distributions over objects, and these distributions are built compositionally. We explored this idea within a domain of tree-like objects, and carried out a study of human generalization using a broad variety of concepts in this domain. Our results suggest that humans are able to extract abstract regularities, such as recursive structure, from examples, but also that there are many subtle effects to be discovered and accounted for in such domains.

**Acknowledgements** We thank Frank Jäkel and Brenden Lake for useful comments. This work was funded in part by grants from the ONR (N00014-09-0124) and the AFOSR (FA9550-07-1-0075).

**References**


