Learning Graphical Concepts

KEVIN ELLIS, EYAL DECHTER, RYAN P. ADAMS, JOSHUA B. TENENBAUM

**INTRODUCTION**

Many of the artifacts humans produce exhibit a compositional structure. One example is the graphical model. Machine learning researchers make frequent use of graphical motifs and abstractions such as trees, chains, rings, grids, mixtures, and plates to constrain the space of graphical models they consider.

We show how one could learn these motifs and abstractions, or “graphical concepts,” by finding programs that generate common graphical models. In particular, we present the Compositional Exploration/Compression (CEC) algorithm, a general purpose multitask program induction algorithm.

**Graphs as Programs**

One may represent a family of graphical models as a function mapping visible nodes to graphs incorporating those visible nodes. Four graph combinators serve to create larger graphs out of two smaller graphs.

![Graphs as Programs](image)

Figure 1: Left: Graphical models can be represented as programs that produce graphs from lists of visible nodes. Right: Four combinators serve to combine smaller graphs into larger ones.

**REFERENCES**


**The CEC Algorithm**

We formulated multitask program induction as inference in a hierarchical Bayesian model similar to that of [2], but extended to incremental program synthesis via function composition.

\[ G = \text{a stochastic grammar over programs.} \]

\[ \rho_n \]

\[ \ell_n \]

\[ \text{A}_{\text{combinators}} \text{els as a function mapping visible nodes to graphs} \]

\[ \text{Inference is by an EM procedure, with update equations given by} \]

\[ q(\rho_n) \propto P(t_n | \rho_n) P(\rho_n)^{G_{\text{old}}} \]

\[ \sum_{n} \mathbb{E}_{q_n} [\ln P(\rho_n) | G] \]

Estimation of \( q(\cdot) \) is approximated by a search over programs, the exploration step, while the maximization is accomplished by the compression step.

**EXPLORATION STEP**

Our objective is either to sample from \( q(\rho_n) = P(\rho_n | G, t_n) \) or search for the \( \rho_n \)'s with high probability under \( q(\cdot) \). To sample/search for programs of type \( \tau \), we first sample programs of type \( \tau \) from \( G \). Then, we modify those programs by applying them to programs of type \( \tau \rightarrow \tau \), also sampled from \( G \), as diagrammed below:

\[ \text{This gives an MCMC algorithm that serves as a proposal distribution for } q(\cdot). \]

\[ T(e_n^{i+1} \circ \rho_n | e_n^{i} \circ \rho_n) \propto P(t_n | e_n^{i+1} \circ \rho_n) P(e_n^{i+1} | G) \]

and the proposal weights are given by

\[ w(e_n^{i} \circ \cdots \circ e_n^{i}) = \left[ \prod_{j < i} P(t_j | e_n^{i} \circ \cdots \circ e_n^{j}) \right]^{-1} \]

Alternatively, \( T(\cdot) \) can serve as a successorship operator in a beamed search, which we found to work better.

**COMPRESSION STEP**

The objective

\[ -\ln P(G) - \sum_{n} \mathbb{E}_{q_n} [\ln P(\rho_n) | G] \]

has a natural interpretation as a form of compression: pick the grammar, \( G \), minimizing the description length of \( G \) plus the expected description length of a program sampled from \( q_n(\cdot) \).

As in [1], \( G \) is assumed to consist of a weighted set of subtrees, forming a library of procedures that can be reused in subsequent program synthesis. We pick the prior

\[ P(G) \propto \exp (-\lambda(\# \text{ symbols in } G)) \]

and approximate the log likelihood of \( \rho_n \) as

\[ -\ln P(\rho_n | G) \approx \sum_{e_n^{i} \in \rho_n} (\# \text{ symbols in } e_n^{i} | G) \]

which gives a MAP solution of \( G \) that incorporates subtree \( t \) iff

\[ \sum_{e_n^{i} \in \rho_n} | e(t, \rho_n) | \geq \lambda \]

where \( e(t, \rho_n) \) is the number of times subtree \( t \) occurs in \( \rho_n \).

**RESULTS**

On a set of twelve synthetic test cases, including ising models, rings, Hidden Markov Models, etc., CEC completely solved eight tasks and partially solved the other four. Simpler graphical models bootstrapped the learning of complex graphical models via compression. For example, first learning a Markov model allowed subsequent learning of the Ising model. The final grammar learned contained the following library routines:

- Connect a new latent node to the first node.
- Connect a new latent node to each node in the graph.
- Connect \( i^{th} \) node to \( i^{n+1} \) node.
- Connect \( i^{th} \) node to \( i^{n+1} \) node, and the first node to the last node.
- Create a new latent node for each visible node, and then connect one of the latent nodes to one of the visible nodes.

Below a solution to the cylinder graphical model is diagrammed.

**FUTURE WORK: TOWERS**

[Diagram of a solution to the cylinder graphical model]