Cross-Domain Knowledge Transfer Using Structured Representations

Samarth Swarup
Department of Computer Science
University of Illinois at Urbana
Urbana, IL 61801
swarup@uiuc.edu

Sylvian R. Ray
Department of Computer Science
University of Illinois at Urbana
Urbana, IL 61801
ray@cs.uiuc.edu

Abstract

Previous work in knowledge transfer in machine learning has been restricted to tasks in a single domain. However, evidence from psychology and neuroscience suggests that humans are capable of transferring knowledge across domains. We present here a novel learning method, based on neuroevolution, for transferring knowledge across domains. We use many-layered, sparsely-connected neural networks in order to learn a structural representation of tasks. Then we mine frequent sub-graphs in order to discover sub-networks that are useful for multiple tasks. These sub-networks are then used as primitives for speeding up the learning of subsequent related tasks, which may be in different domains.

1 Introduction

We are interested in the design of agents that work for an extended period of time [7]. During this "lifetime", an agent may encounter several, possibly related, learning tasks. It is desirable that the agent should be able to improve its learning performance with each new task encountered, i.e., it should gain some experience about "how to learn" each time it has to learn a new task. Techniques for accomplishing this are called, variously, cumulative learning, lifelong learning, meta-learning, learning to learn etc. [11, 12, 9].

Most methods for knowledge transfer in machine learning assume that all tasks come from the same domain. A domain is an input-output space. There are several reasons for this. Inputs can be thought of as coming from sensors, e.g. in robots, which then have to learn several tasks in a given domain. Thus the input space is fixed, and knowledge transfer is done by, e.g, learning with a neural network in which the first layer is shared by several tasks [1]. However, for more higher-level, "cognitive" tasks, it is unreasonable to try to learn directly from sensory inputs. Instead learning can be thought to occur in some state space, or feature space, that is extracted from the sensory inputs. It would still be very useful to be able to transfer knowledge across tasks. However, we may now have to deal with input-output spaces of different dimensions. It is relatively difficult, though, to imagine how to transfer information between two neural networks of different input and hidden dimensions.

Evidence from psychology, however, suggests that humans can, and often do, transfer in-

formation across domains. For example, Dunbar and Blanchette studied analogies used in political speeches, and analogies used by scientists in labs [3]. In a study conducted by videotaping and audiotaping molecular biologists and immunologists over the course of several lab meetings, they found both within-domain and cross-domain analogies being made. While the majority of analogies (75%) were within-domain and involved similarity of superficial features, cross-domain analogies dominated when the scientists were trying to formulate new hypotheses. 80% of the analogies used in this case were cross-domain and used structural features (i.e. relationships). In an analysis of analogies used by politicians and journalists during a referendum campaign in Quebec in 1995, they found that only 24% were from politics, i.e. over 75% of the analogies were based on structural comparisons with other domains. In contrast, experiments on analogy making conducted in laboratory settings often find that people mostly make analogies based on superficial features. This is not surprising perhaps, because these experiments are generally timed, and structure mapping is a more computationally intensive procedure. The point is that people transfer knowledge across domains quite naturally. It has even been argued that analogies form a central part of cognition [4].

Support for this idea also comes from the study of mirror neurons [6]. Mirror neurons are neurons that are active both when an action is performed and when it is observed. They have also been implicated in empathy, the evolution of language, and multi-modal learning.

These studies suggest that knowledge transfer in machine learning should be extended from dealing with multiple tasks in a single domain to tasks in multiple domains. We present here a learning method for achieving this goal. The key idea is to use a structured representation, which allows extraction of domain-independent knowledge. The representation and learning method are described in more detail in the next section.

The rest of this article is organized as follows. In the next section we describe the representation and learning algorithm. This is followed by a set of preliminary experiments that demonstrate the benefit of knowledge transfer across domains in a set of Boolean-function learning tasks. Finally, we discuss some application areas and directions for future work.

2 Representation, Learning, and Knowledge Transfer

Representation: To capture the notion of the structure of a task, we use many-layered, sparsely-connected neural networks [10]. These are constructed out of a set of *primitives*, which are also networks. To begin with, the set of primitives is small and contains just networks with a single node. In our implementation, each node of a network computes a sigmoid function of the weighted sum of its inputs, but this does not have to be the only case. A node (or a primitive) might compute any function of its inputs. A network is a composition of primitives into a directed acyclic graph. There is no reason recurrent connections should not be allowed, but they have been disallowed here for simplicity.

A network is also represented by a *genome*. A genome, unlike a network, is linear. Each *gene* in the genome is a structure that corresponds to a node in the network. It describes the type of the node (i.e. the function it computes), the position of the node in the network, and which nodes provide input to this node. Some nodes get external inputs; these are simply denoted by a -1 for the input number in the corresponding gene. One node is designated as the output node. Again, for simplicity, we are only considering tasks with a single output.

Learning and Knowledge Transfer: Learning is done by an evolutionary algorithm that constructs a population of networks from the primitives. Each network in the population is evaluated on the training set and assigned a fitness corresponding to its accuracy on the training set. A subset of the population is chosen to create the next generation via mutation and crossover. There can be three kinds of mutations: *adding*, *deleting*, and *replacing*

mutations. A deleting mutation simply removes a gene from the genome of a network. An adding mutation adds a *primitive* to the genome. Remember, a primitive can be an entire network. Similarly, a replacing mutation replaces a gene with a primitive. A new network is created from two parents of high fitness by a one-point crossover. This is followed by a mutation with some small probability. Some of the high fitness networks are carried over unchanged to the next generation, and the process is repeated until some accuracy or time criterion is met.

The set of primitives is augmented by running a graph mining algorithm on the set of tasks that have been solved, to discover frequent sub-networks. These represent partial solutions and are common to several tasks. This allows transfer of knowledge across tasks in different domains because these sub-networks are not bound to a particular input space.

3 Experiments

We did experiments with a set of Boolean functions where the output (0 or 1) depends on just the absence or presence of a certain number of *adjacent* 1's in the input. For example, in one set of tasks, the output is 1 if there are 2 adjacent 1's anywhere in the input vector. The initial set of primitives were nodes that compute the AND, OR, and NOT functions, as well as an "input" node that just copies its input unchanged to its output.

The tasks were defined over four domains, i.e. input spaces of four different dimensions - 4, 8, 12, and 16 inputs. Ten tasks were learned over the four domains, differing in the number of adjacent 1's necessary for a 1 output. Figure 1 shows all the learning curves. The 4-inputs, 2-adj-ones task was learned first, though the learning curve for that task is not shown. The tasks were learned in left to right, top to bottom order (note that fig. 1 is rotated, so the top of the figure is on the right of the page), and the CloseGraph graph mining algorithm ([13]) was used to extract a set of common sub-networks from all the previous networks after each new task network was learned. We see that learning with transfer far outperforms learning without transfer in each case. We call this approach *cumulative learning* because the set of sub-networks, which represents a store of abstract or common knowledge, grows in a cumulative way over the course of learning (i.e. over an agent's lifetime).

Another major advantage of cumulative learning is that when we learn by transferring knowledge from previous tasks, not only do we learn faster, we also discover solutions that are similar to the ones we already know (for other tasks). There are generally several ways of solving a particular task. The advantage of finding similar solutions is that we then discover and reinforce[1] the patterns that are useful for multiple tasks. This means that in the future we will use these patterns more in solving related tasks, and thus discover solutions that again reinforce these patterns. This is the *cumulative advantage* [2].

4 Future Work

The main direction for future work is to try this approach in a real-world situation. The idea of primitives carries over easily to robotics and motion planning where the primitives are a set of basic movements or abilities, and complex actions are carried out by composition of primitives [5, 8]. Our cumulative learning approach could be used to coordinate multiple limbs and learn complex actions quickly.

[1]The word *reinforce* is not being used in a technical sense here.

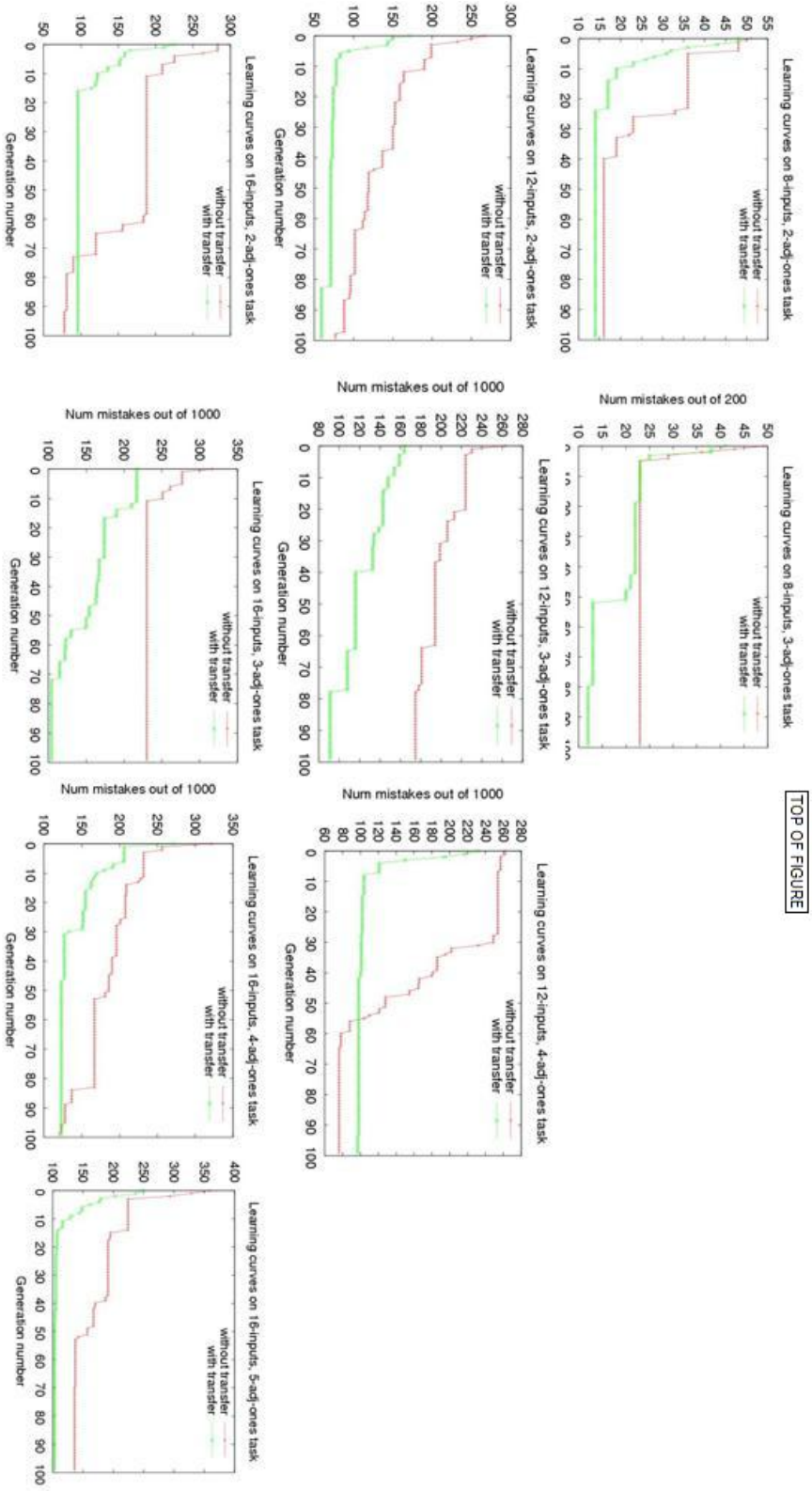

Figure 1: Learning curves with and without transfer of information, for each task. Tasks were learned in top to bottom, right to left order, i.e. 8-inputs-2-adj-ones, 8-inputs-3-adj-ones, 12-inputs-2-adj-ones, etc.

References

[1] Rich Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.

[2] Derek de Solla Price. A general theory of bibliometric and other cumulative advantage processes. *Journal of the American Society for Information Science*, 27:292–306, 1976.

[3] Kevin Dunbar and Isabelle Blanchette. The inVivo/inVitro approach to cognition: The case of analogy. *Trends in Cognitive Sciences*, 5:334–339, 2001.

[4] Douglas R. Hofstadter. *The Analogical Mind: Perspectives from Cognitive Science*, chapter Analogy as the Core of Cognition, pages 499–538. The MIT Press, 2001.

[5] Marcelo Kallmann, Robert Bargmann, and Maja Matarić. Planning the sequencing of motor primitives. In *Proceedings of the International Conference on Simulation of Adaptive Behavior (SAB 2004)*, Los Angeles, CA, July 13-17 2004.

[6] G. Rizzolatti, L. Fadiga, M. Matelli, V. Bettinardi, E. Paulescu, D. Perani, and G. Fazio. Localization of grasp representations in humans by positron emission tomography: Observation versus execution. *Experimental Brain Research*, 111:246–252, 1996.

[7] Samarth Swarup, M. M. Hassan Mahmud, Kiran Lakkaraju, and Sylvian R. Ray. Cumulative learning: Towards designing cognitive architectures for artificial agents that have a lifetime. Technical Report UIUCDCS-R-2005-2514, University of Illinois at Urbana-Champaign, Dept of Computer Science, 201 N. Goodwin Ave, Urbana, IL 61801, USA, 2005.

[8] Kurt A. Thoroughman and Reza Shadmehr. Learning of action through adaptive combination of motor primitives. *Nature*, 407:742–747, October 2000.

[9] Sebastian Thrun and Lorien Pratt. *Learning to Learn*, chapter Learning to Learn: Introduction and Overview. Kluwer Academic Publishers, 1998.

[10] Paul E. Utgoff and David J. Stracuzzi. Many-layered learning. *Neural Computation*, 14(10), Oct 2002.

[11] Ricardo Vilalta and Youssef Drissi. Research directions in meta-learning. In H. R. Arabnia, editor, *Proceedings of the International Conference on Artificial Intelligence*, Las Vegas, Nevada, USA, 2001.

[12] Ricardo Vilalta and Youssef Drissi. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 18(2):77–95, October 2002.

[13] Xifeng Yan and Jiawei Han. CloseGraph: Mining closed frequent graph patterns. In *Proceedings of the 9th ACM SIGKDD conference on Knowledge-Discovery and Data Mining (KDD 2003)*, 2003.