Cumulative Learning: Towards Designing Cognitive Architectures for Artificial Agents that Have a Lifetime

Samarth Swarup, M. M. Hassan Mahmud, Kiran Lakkaraju, and Sylvian R. Ray

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL, USA

Abstract

Cognitive architectures should be designed with learning performance as a central goal. A critical feature of intelligence is the ability to apply the knowledge learned in one context to a new context. A cognitive agent is expected to have a lifetime, in which it has to learn to solve several different types of tasks in its environment. In such a situation, the agent should become increasingly better adapted to its environment. This means that its learning performance on each new task should improve as it is able to transfer knowledge learned in previous tasks to the solution of the new task. We call this ability *cumulative learning*. Cumulative learning thus refers to the accumulation of learned knowledge over a lifetime, and its application to the learning of new tasks. We believe that creating agents that exhibit sophisticated, long-term, adaptive behavior is going to require this kind of approach.

1 Introduction

A cognitive architecture is traditionally memory-centric. A representative statement of this view could be – a cognitive architecture consists of its representational assumptions, the characteristics of its memories, and the processes that operate on those memories [Langley and Laird, 2002]. This means that the focus of the design of the architecture is on what is stored in memory, how it is organized, how it is accessed and how the accessed information is used. While this is undoubtedly an important concern, from the point of view of *performance* this is not the central issue.

We think that one of the central goals of a cognitive architecture should be an ability to gain, and use, *experience* about its environment. The agent should adapt to its environment, and become increasingly better at solving new tasks. This puts the emphasis directly on performance. Thus the agent should be able to use information from previously encountered tasks to enhance the learning of new tasks (if the tasks are similar).

An impairment in the ability to transfer learning from one situation to another is in fact considered one of the hallmarks of mental retardation in humans and animals [Strupp *et al.*, 1994]. Artificial cognitive architectures need to be evaluated by the same metric if we wish to achieve comparable behavior. Cumulative learning ability can thus provide a much-needed means of comparing and evaluating cognitive architectures [Langley and Laird, 2002].

In the domain of machine learning, the idea of transferring knowledge between learning tasks has been variously referred to as lifelong learning [Thrun and Mitchell, 1995], meta-learning [Vilalta and Drissi, 2001], and learning to learn [Thrun and Pratt, 1998].

However, when lifelong learning is put into the context of a cognitive system, it must be done a little differently. This is because most machine learning approaches deal with a fixed domain, whereas cognitive agents need to transfer knowledge across domains. For example, when we say "That's like the pot calling the kettle black!", the situation generally involves neither pots, nor kettles, nor even black things. This suggests that the cognitive system should be able to accumulate a store of information about the environment, which is abstracted from all the learning tasks that are encountered. This "abstract knowledge" then represents the agent's general "understanding" of the world, akin to what is known as "semantic memory" in humans. We call this process of learning multiple tasks and building up a store of abstract knowledge, *cumulative learning*.

Cumulative learning refers to the accumulation of (learned) knowledge over a "lifetime". In meta-learning in general, the "abstract knowledge" is implicit in the set of known tasks. However, we believe it is important to have *explicit* abstract knowledge because we would like agents to be able to communicate with, and teach, each other. This ability would greatly increase the adaptability and the performance of the agents, as they would no longer have to solve each task independently.

In a realistic scenario, all the agents would not be identical. Some individual specialization should be provided so that a range of special abilities are available. They may have different sensors and effectors, and different physical dimensions. This means that their internal states would not be identical, and therefore they could not communicate by just copying or sharing their internal states. Therefore, a public "language" is required in order to transfer information from one agent to another. Development of such a language is one of the most important problems in multi-agent research today [Steels, 2003], [Klingspor *et al.*, 1997].

The goal of the present paper is to examine the issues involved in designing a cognitive architecture around the notion of cumulative learning. We are not going to present a particular design for a cognitive architecture here. Rather, we are going to discuss some general principles that must, we think, be incorporated in a cognitive architecture.

The rest of this paper is organized as follows. First, we discuss some of the main attempts at creating cognitive agents in the past. Then we discuss why cumulative learning is an attractive design principle for cognition. After that we present some of the work that we are currently doing to explore the idea of cumulative learning. Finally, we discuss the steps necessary to build a complete cognitive architecture around our suggested approach, which will also suggest important directions for future work.

2 Related Work

In this section we discuss two areas of research that are related to our method. First we briefly survey some cognitive architectures, and then we discuss Lifelong Learning techniques.

2.1 Cognitive Architectures

Many cognitive architectures have been developed by Artificial Intelligence researchers since the inception of the field. In the following, we discuss some representative architectures in terms of how each improves its performance from experience.

The first group of architectures we consider is typified by architectures such as Soar [Newell, 1990], Prodigy [Veloso *et al.*, 1995], Icarus [Laird, 1990] etc. All of these use some type of symbolic language (such as FOPL, or the STRIPS language) to encode knowledge of the agent. This knowledge describes the entities that exist in the world (e.g. `drill-bits` in a robot drill press application) and the known effect of the agents actions on these entities (e.g. `apply-drill` causes `hole-in-metal-plate`). In general, these architectures implement sophisticated extensions to classical planning, which learn to improve planning performance from experience. The actual methods employed vary from architecture to architecture, but they usually take the form of learning macro-actions [Russell and Norvig, 2003]. For example, Prodigy uses Analogical Learning to determine what sequence of actions will be useful in a particular task using knowledge about solutions/sequences of actions used in simi! lar tasks. These methods are also able to refine their domain knowledge, or acquire new knowledge from experience. For example, in the Prodigy architecture, if the effects of some particular action were not known a-priori, the agent can perform observations to learn about the effects of the action.

Another group of architectures is centered around the subsumption architecture (SA) [Brooks *et al.*, 1999], where the focus is on constructing a robotic agent that works in the real world in the presence of human beings and other active entities. The robot is implemented in terms of a set of interacting behaviors that operate in parallel, and are implemented in hardware by human designers. Examples of behaviors are "avoid walls", "find coke-can" etc. The behaviors are actually very flexible constructs and in general can be any control mechanism. Learning in these systems is almost exclusively concerned with improving the performance of the existing behaviors through experience. One of the most important design philosophies of SA is that there is no explicit monolithic behavior spanning representation of the world in the agent. Rather the world is used as its own model. We should, however, note that behaviors have states and thus a behavior can store knowledge that is pertinent to tha! t particular behavior.

There are also architectures that combine the ideas of these two broad groups mentioned above. We will consider the 3T architecture [Bonasso *et al.*, 1997] for concreteness. 3T stands for 3 Tiered architecture and, as its name implies, it consists of three tiers. The lowest tier implements skills/behaviors as in the subsumption architecture, and the next tier executes these skills in some sequence. The final tier implements a planner similar to a classical planner and uses the skill sequences in the second level as its actions or operators. The planner is also able to deal with situations when there are multiple agents in the domain. At this point, the learning in this system is limited to that in subsumption architectures - i.e. only at the lowest tier. But it is not that difficult to envision integrating learning mechanisms from the first group with this method.

2.2 Meta-Learning

Meta-learning focuses on how learning algorithms can change their bias when given several tasks. Base-learners, like traditional neural-networks, decision trees and SVMs, have a fixed bias that limits the hypothesis space through which they search. The more examples base-learners are given, the better they will be able to find the correct hypothesis. But when a base-learner must solve multiple related tasks, they cannot transfer the knowledge they gained from one task to another. Meta-Learners, on the other hand, can modify their hypothesis space from task to task, so that after learning one task, a second related task will be easier to learn.

Meta-learning has been approached in many ways. One approach is to dynamically change the bias of an algorithm. This would allow the hypothesis space to cover the hypotheses that are relevant to the tasks at hand, rather than being fixed. Another approach uses simple base-learners on a set of tasks, then chooses a more advanced learner based on the success of the simple base-learners. Lifelong learning is an approach to meta-learning that finds invariants among tasks. These invariants are used to bias the learning algorithm [Vilalta and Drissi, 2002].

Meta-learning approaches suffer from three main problems though:

First of all, the domains of the tasks must be the same. It is apparent that humans can learn from different tasks, in different domains. Analogical reasoning is based on this. Meta-learning techniques, though, require the domains (i.e. the hypothesis space) to be the same.

Secondly, meta-learning techniques are not memory efficient, as they require storage of the training sets. In a realistic situation, agents will not have unbounded memory capabilities, and thus will not be able to store the training sets from

all the tasks they have witnessed.

Finally, in Meta-learning, task knowledge is stored implicitly, as a bias on the hypothesis space the learner is looking at. Since the knowledge is implicitly stored, there is no way to use this knowledge in different domains, or to analyze the knowledge.

3 The Argument for Cumulative Learning

The most persuasive argument for cumulative learning is that people use analogies to solve problems all the time. Analogy-making is widely considered to be one of the core properties of cognition [Hofstadter, 2001]. Further, people often make *cross-domain* analogies. For example, Blanchette and Dunbar analyzed all the analogies reported in newspapers in the final week of a referendum campaign in Canada. They found that over two-thirds of the analogical sources were non-political [Blanchette and Dunbar, 2001]. Analogical thinking is also a key component of all aspects of scientific reasoning, from formulating theories, to designing experiments, to explaining results [Dunbar, 2001]. Cross-domain analogies are structural, which means that the relationships between the entities correspond across domains, but not the entities themselves (by definition). This means that people deal with multiple *ontologies* concurrently, as they deal with multiple tasks.

A task is defined by three things. The first is the ontology of the task, which defines what entities exist. For instance, in the blocks world, we have blocks, and the table. In addition, the ontology defines the relationships between the entities, such as on(block, table).

The second component of a task is the situation, which is the particular placement of entities in the world. Continuing the blocks world example, a situation might be: on(block1, block2), on(block2, table), on(block3, table).

Finally, the third component of a task is the goal state or states. These specify the desired situation to be achieved. For instance, a goal situation might be: on(block3,block1).

The problem with current cognitive architectures is that, when learning over multiple tasks, the agent cannot acquire a new ontology without (possibly significant) designer intervention. So such an agent, for instance, will need designers to manually encode a new ontology in the agent if it is asked to build with silly putty instead of blocks. Cumulative Learning, on the other hand, considers the situations where the ontology may change across tasks, and thus focuses on extracting abstract knowledge that is valid, not only from task to task, but also across ontologies. This extraction of abstract knowledge, as we have pointed out earlier, is the main distinguishing feature of a cumulative learning approach.

In the next section we look at the ways we are using the notion of similarity to transfer knowledge across tasks, which might require different and new ontologies , and extract abstract knowledge.

4 Current Work

Our current work on cumulative learning, described below, focuses on ways of identifying similarity between tasks and situations, and on mechanisms of transferring knowledge between tasks and between agents. In the future work section, we discuss how we might go about building a cognitive architecture around our current work.

4.1 Cumulative learning using structural similarity

It is well known that if two tasks are similar, we can transfer knowledge from one to the other to speed up the learning of the new task. It is less well understood how to measure the similarity between two tasks. Generally, if two tasks are from the same domain - if they are both face recognition tasks, e.g. - then they are assumed to be related. Approaches to measuring similarity have mainly involved analyzing the weights of learned classifiers or distance metrics for applicability to the new task, or for measuring similarity [Silver and Mercer, 2001], [Thrun and O'Sullivan, 1998].

These approaches do not really capture the notion of the *structure* of the task. In order to capture this notion, we use many-layered, sparsely connected, neural networks [Utgoff and Stracuzzi, 2002]. Learning is done using a neuro-evolution method similar to NEAT [Stanley and Miikkulainen, 2002]. Task similarity, then, is a problem of finding the structural similarity between these networks, and graph matching techniques such as [Hagenbuchner *et al.*, 2003], [Meuss and Schulz, 1998], etc. can be used.

Knowledge is transferred by extracting a set of frequent sub-graphs by using a graph mining algorithm such as gSpan [Yan and Han, 2002]. This set of frequent sub-graphs represents the "abstract knowledge" or the common knowledge. We can then use these sub-graphs instead of individual nodes to generate candidate neural networks, and thus speed up learning. Figure 1 shows this scenario.

This allows transfer of information across ontologies, because a sub-graph represents a partial solution to a task, and thus is not restricted to a single domain. The semantics of the domain are not inherent in the subgraph. It merely represents a useful function to compute for multiple tasks.

In a cognitive architecture, the process of mining frequent subgraphs would run during idle-time, i.e. when the agent is unoccupied with an immediate task, or is "sleeping".

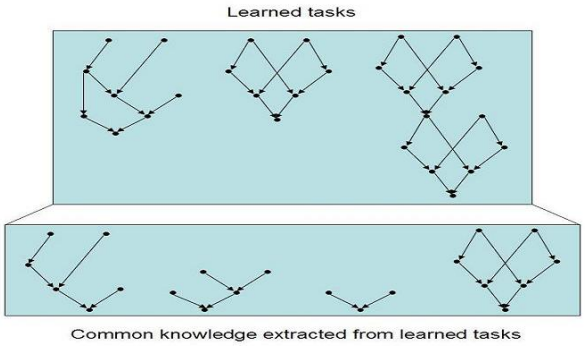

Figure 1: Tasks are represented by sparsely connected multi-layered feedforward networks. "Abstract knowledge" is extracted by mining frequent sub-graphs.

4.2 Cumulative learning using functional similarity

Qualitatively speaking, we say *two situations are functionally similar (FS)* **with respect to an action of the agent**, *if the action induces similar changes in both the situations*. FS situations can be seen everywhere in real life. For example, 'playing racquetball' and 'playing squash' are FS situations because squash balls and racquetball balls (along with the courts) 'behave' similarly with respect to racquet strokes.

In general, we exploit functional similarity by first training classifiers on situations that are FS to each other. This classifier can then be used to determine if a novel situation (situation not encountered often or ever) is FS to the previously encountered situation. If it is, then the behavior of older situations w.r.t. the corresponding action can be used to predict behavior of the novel situation w.r.t. the action. So in the example above, we identify that racquetball is FS to squash, and hence use knowledge learned in learning to play squash in learning to play racquetball. [Mahmud and Ray, 2005] formally describes one possible way to define and use FS in Markov Environments and goal directed MDPs (e.g. [Moore *et al.*, 1999]) and also suggests various extensions to basic FS, such as similar actions. In the following we speculate on how to use functional similarity to create a semantic network in a general setting (i.e. abstract domains along with the 3D real-world humans inhabit).

Now, as Minsky pointed out in [Minsky, 1988], when we say we *understand* something what we really mean is we know how it relates to *everything else we know*. For instance, we understand a pencil in terms of 'can make marks on *paper*', 'can poke *someone we do not like*' and so on. Since the 'basis' object we can relate anything we understand to is ourselves, and since functional similarity captures how the world relates to the agent, a model of the world in terms of functional similarity leads to, in a non-phenomenal sense, basic semantics.

Furthermore, in human interactions with the real world, situations are usually functionally similar if they contain the same or similar objects - e.g. the balls and the courts in the racquetball and squash examples. Thus we can identify the commonalities between FS situations as objects (with respect to the action). Also, we can describe the semantic relationship between different objects in terms of how, on application of an action, change in one object relates to change in another object. As an example, application of the FOREHAND-STROKE action on a RACQUET results in the SQUASH-BALL object moving in a certain way. Thus functional similarity can help us develop a non-phenomenal semantic network that is grounded in the domain. We can also model other active entities in the world (such as other agents) using the FS approach.

Thus, in the FS based approach, instead of commiting to a particular ontology, the agent is learning the ontology by learning the existence of objects and their relationships. This in turn helps address the problem pointed out in section 3. The next step, now, is to formalize the ideas presented here along the lines suggested in section 5 (for example, by extending the work in [Mahmud and Ray, 2005]).

4.3 Cumulative learning by transferring information between agents

Cumulative learning can be approached from the perspective of knowledge integration. Knowledge about the structure of tasks, or the structure of the solutions of tasks, from previously encountered tasks affects the learning of new, similar tasks.

It is clear though, that humans do not learn independently, rather they gain knowledge from other humans. This can be through seeing another human perform the task, or having the another human tell how to solve the task. Similarly, we can look at the situation of an agent integrating knowledge about a task that *another* agent, possibly with an entirely different internal structure, has learned.

Integration of knowledge from other agents poses some problems though. First of all, internal states cannot be communicated, since the agents might be totally different. As each agent has a different internal ontology to describe the world, a mapping between the different ontologies is needed. A solution to this is to create a common language through which the agents can communicate. The creation of this language and the issue of each agent learning the language is a big problem as well.

Once a language has been defined, the question becomes how an agent can use the knowledge passed to it from another agent. A language is a symbolic system, with a finite alphabet and a grammar that determines the structure of sentences. On the other hand, the agents cognitive architectures might be non-symbolic, like neural networks. The issue is to be able to integrate knowledge presented in a symbolic form, like propositional logic, into a non-symbolic framework, like a neural network [Coradeschi and Saffiotti, 2003].

Currently we are studying how an agent with a feed-forward fully connected neural network can integrate knowledge from other agents that is represented in a propositional logic format. Each agent must solve the task of differentiating between poisonous and non-poisonous mushrooms, given the feature vector of the mushrooms; this is a basic classification task.

In addition to being given a training set of mushrooms for the agent to learn to classify, via the standard backpropagation algorithm, each agent can receive propositional logic statements from other agents that identify key features that will be useful to learn the classification task.

In this framework, we are investigating several questions:

- How can an agent transform its internal knowledge into a common language (in this case propositional logic)?
- How can an agent incorporate knowledge from other agents into its neural network?
- How much can an agent trust the knowledge from another agent?

5 Building a Cumulative Learning Cognitive Architecture

As we mentioned earlier, a cumulative learning system is based on the idea of sharing information across similar tasks. In the previous section, we identified some ways of doing this.

We now outline how, given the specification of how to extract similarity knowledge, and hence abstract knowledge, we may construct a cumulative-learning based cognitive architecture. The reader may note that we include the capability to deal with high-dimensional, noisy, streaming sensor data and perform anticipation as part of the outline. This is because any practical architecture needs to be able to deal with the former, while the latter is necessary to act and react in real time. We also note that cumulative learning techniques can help accomplish this, e.g. by using similarity between situations to predict the outcome of actions in new situations.

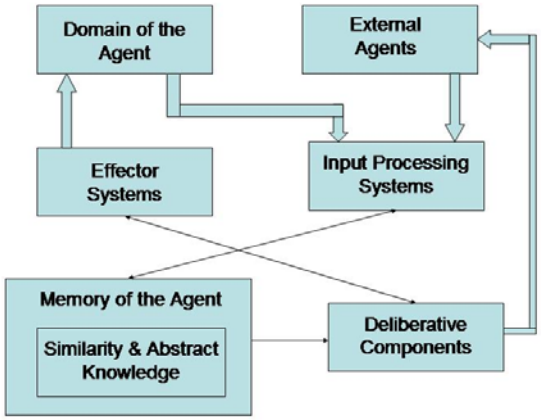

Figure 2: High level information flow in the CL agent. Flow within each element is not shown.

At the very highest level, a CL system is given a set of tasks to complete (which may be changed later on), some initial knowledge in its memory and then allowed to operate autonomously on its own to complete the tasks. We may give a more detailed description in terms of how information flows through the CL system (please see figure 2).

Information flows from the outside world and into the sensors of the agent, which are then processed by the input processing systems. The input processing systems of the CL agent are arranged in a hierarchy, with each system processing output from other layers, or the sensors, or information from external agents to generate its own output. The latter consists of identified similarity measures and abstract knowledge specified for that particular layer. The human vision system is an example of this kind of system [Lee and Mumford, 2003].

The output of a particular input system is used by one or more deliberative components to decide which actions to take to achieve their respective current goals. The actions of a component affect either effectors of the agent or behaviors of other components. The former may also result in communication being sent to other agents. A component affects other components by changing parameters that determine their behaviors - but we expect this will mostly be done by changing the goals of the other components. Initially, only some of the components will have any goals (given by the user in the beginning), but other components will be assigned goals by these initial components as part of the control process.

Finally, we expect that both deliberative components and input processors may perform anticipation to improve its performance. The former may use this to predict behavior of other components or the environment or the effectors of the agent to plan ahead [Munoz *et al.*, 2000]. And the latter may use it to speed up processing by anticipating what inputs lower layers will send it next [Lee and Mumford, 2003].

6 Future Work and Conclusions

In this paper we have outlined a new approach to designing cognitive architectures that we have termed cumulative learning. The most interesting feature of this approach is a focus on improving learning performance through the transfer of knowledge between tasks over the lifetime of the agent, which current cognitive architectures lack. The discovery of "abstract knowledge", which captures general concepts that would be useful across tasks, allows the transfer of knowledge between tasks. The key, then, to the transfer of knowledge between tasks is the creation of an "abstract knowledge" store.

We also briefly described existing cognitive architectures and pointed out that none of them implement a store of abstract knowledge of the type we described. Meta-learning techniques are also limited in their capabilities to transfer knowledge between tasks. We then described our current work in progress that focused on identifying similarity across tasks as the means to improve performance with experience.

The directions for future work are many. The main goal is to implement a complete cognitive architecture based on the principle of cumulative learning. In section 5, we suggested a high-level outline of such an architecture. However, in order to do this, we must first investigate a methodology to identify similarities across tasks in which the domains (ontologies) are different. Directions for this may come from psychological studies of analogy in humans.

Another important problem is the development of a common language that can be used to transfer information between agents. Transferring "abstract knowledge" between agents will enable them to teach each other to improve their learning performance, not just their performance on a particular task. The language needs to evolve over time, as the environment is expected to change over the lifetime of the agents.

Cumulative learning could also help in providing a means to compare and evaluate cognitive architectures.

References

[Blanchette and Dunbar, 2001] Isabelle Blanchette and Kevin Dunbar. Analogy use in naturalistic settings: The influence of audience, emotion, and goals. *Memory and Cognition*, 29(5):730–735, 2001.

[Bonasso *et al.*, 1997] R. Peter Bonasso, R. James Firby, Erann Gat, David Kortenkamp, David P. Miller, and Marc G. Slack. Experiences with an architecture for intelligent, reactive agents. *Journal of Experimental & Theoretical Artificial Intelligence*, 1997.

[Brooks *et al.*, 1999] Rodney Brooks, Cynthia Breazeal, Matthew Marjanovic, Brian Scassellati, and Matthew Williamson. The cog project: Building a humanoid robot. *Computation for Metaphors, Analogy, and Agents, Lecture Notes in Computer Science*, 1999.

[Coradeschi and Saffiotti, 2003] Silvia Coradeschi and Alessandro Saffiotti. An introduction to the anchoring problem. *Robotics and Autonomous Systems*, 43:85–96, 2003.

[Dunbar, 2001] Kevin Dunbar. *Analogy: Perspectives from Cognitive Science*, chapter The Analogical Paradox. The MIT Press, Cambridge, MA, 2001.

[Hagenbuchner *et al.*, 2003] Markus Hagenbuchner, Alessandro Sperduti, and Ah Chung Tsoi. A self-organizing map for adaptive processing of structured data. *IEEE Transactions on Neural Networks*, 14(3):491–505, May 2003.

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

[Klingspor *et al.*, 1997] Volker Klingspor, John Demiris, and Michael Kaiser. Human robot communication and machine learning. *Applied Artificial Intelligence Journal*, 11:719–746, 1997.

[Laird, 1990] A. Laird. Unified theories of cognition. 1990.

[Langley and Laird, 2002] Pat Langley and John E. Laird. Cognitive architectures: Research issues and challenges. Technical report, Institute for the Study of Learning and Expertise, Palo Alto, CA, 2002.

[Lee and Mumford, 2003] Tai Sing Lee and David Mumford. Hierarchical Bayesian inference in the visual cortex. *Journal of the Optical Society of America*, 20(7):1434–1448, 2003.

[Mahmud and Ray, 2005] M. M. Hassan Mahmud and Sylvian Ray. Cumulative learning using functionally similar states. Technical Report UIUCDCS-R-2005-2511, Department of Computer Science, University of Illinois at Urbana Chamapaign., 2005.

[Meuss and Schulz, 1998] Holger Meuss and Klaus Schulz. DAG matching techniques for information retrieval on structured documents. Technical Report 98-112, CIS, University of Munich, 1998.

[Minsky, 1988] Marvin L. Minsky. *The Society of Mind*. First Touchstone Edition, 1988.

[Moore *et al.*, 1999] A. W. Moore, L. C. Baird, and L. Kaelbling. Multi-value-functions: Efficient automatic action hierarchies for multiple goal MDPs. *Proceedings of the International Joint Conference on Artificial Intelligence, Stockholm*, 1999.

[Munoz *et al.*, 2000] D. P. Munoz, M. C. Dorris, M. Paré, and S. Everling. On your mark, get set: brainstem circuitry underlying saccadic initiation. *Canadian Journal of Physiology and Pharmacology*, 78:934–944, 2000.

[Newell, 1990] A. Newell. *Unified Theories of Cognition*. Harvard University Press, Cambridge, MA, 1990.

[Russell and Norvig, 2003] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Upper Saddle River, NJ, 2nd edition, 2003.

[Silver and Mercer, 2001] Daniel Silver and Robert Mercer. Selective functional transfer: Inductive bias from related tasks. In M. H. Hamza, editor, *Proceedings of the IASTED International Conference on Artificial Intelligence and Soft Computing (ASC2001)*, pages 182–189, Cancun, Mexico, May 2001. ACTA Press.

[Stanley and Miikkulainen, 2002] Kenneth O. Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002.

[Steels, 2003] Luc Steels. The evolution of communication systems by adaptive agents. In E. Alonso, D. Kudenko, and D. Kazakov, editors, *Adaptive Agents and Multi-Agent Systems: Adaptation and Multi-Agent Learning. LNAI 2636*, pages 125–140. Springer Verlag, Berlin, 2003.

[Strupp *et al.*, 1994] B. J. Strupp, M. Bunsey, D. A. Levitsky, and K. Hamberger. Deficient cumulative learning: An animal model of retarded cognitive development. *Neurotoxicology and Teratology*, 16(1):71–79, 1994.

[Thrun and Mitchell, 1995] Sebastian Thrun and Tom Mitchell. Lifelong robot learning. *Robotics and Autonomous Systems*, 15:25–46, 1995.

[Thrun and O'Sullivan, 1998] Sebastian Thrun and Joseph O'Sullivan. *Learning to Learn*, chapter Clustering Learning Tasks and the Selective Cross-Transfer of Knowledge. Kluwer Academic Publishers, 1998.

[Thrun and Pratt, 1998] S. Thrun and L. Y. Pratt, editors. *Learning To Learn*. Kluwer Academic Publishers, Boston, MA, 1998.

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

[Veloso *et al.*, 1995] M. Veloso, J. Carbonell, A. Perez, D. Borrajo, and E. Fink. Integrating planning and learning: The prodigy architecture. *Journal of Experimental and Theoretical Artificial Intelligence*, 7(1), 1995.

[Vilalta and Drissi, 2001] 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.

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

[Yan and Han, 2002] Xifeng Yan and Jiawei Han. gSpan: Graph-based substructure pattern mining. In *Proceedings of the 2002 International Conference on Data Mining (ICDM'02)*, Maebashi, Japan, December 2002.