LEARNING TO LEARN

edited by

Sebastian Thrun
Lorien Pratt

Springer Science+Business Media, LLC

LEARNING TO LEARN

edited by

Sebastian Thrun
Carnegie Mellon University

and

Lorien Pratt
Evolving Systems, Inc.

SPRINGER SCIENCE+BUSINESS MEDIA, LLC

Library of Congress Cataloging-in-Publication Data

A C.I.P. Catalogue record for this book is available
from the Library of Congress.
ISBN 978-1-4613-7527-2 ISBN 978-1-4615-5529-2 (eBook)
DOI 10.1007/978-1-4615-5529-2

Printed on acid-free paper.

Contents

Preface

"Learning to Learn" is a promising research direction within machine learning that has recently gained considerable attention. As with traditional inductive machine learning methods, algorithms that learn to learn induce general functions from examples. Learning to learn methods include an extra feature, however, which is that their learning bias is chosen based on learning experiences in other tasks. Humans often generalize correctly after a small number of training examples by transferring knowledge acquired in other tasks; systems that learn to learn mimic this ability. As various chapters in this book demonstrate, algorithms that learn to learn often produce superior results to those that are not given the extra information that comes from other tasks.

This book provides a comprehensive collection of research on algorithms that learn to learn. It is organized into four parts:

Part I: Overview articles (chapter 1-3)
in which basic taxonomies and the cognitive foundations for algorithms that "learn to learn" are introduced and discussed,

Part II: Prediction/Supervised Learning (chapter 4-8)
in which specific algorithms are presented that exploit information in multiple learning tasks in the context of supervised learning,

Part III: Relatedness (chapter 9-10)
in which the issue of "task relatedness" is investigated and algorithms are described that selectively transfer knowledge across learning tasks, and

Part IV: Control (chapter 11-13)
in which algorithms specifically designed for learning mappings from percepts to actions are presented.

These parts contain the following chapters:

Chapter 1 (Part I). Thrun and Pratt introduce the topic and, along with the basic definitions, provide a basic taxonomy of algorithms that learn to learn.

Chapter 2. Pratt and Jennings survey recent neural network approaches and introduces various helpful characteristics leading to an taxonomy of different methods.

Chapter 3. Robins discusses the phenomenon of "transfer" from a cognitive and a psychological point of view.

Chapter 4 (Part II). Baxter analyzes the learning to learn problem from a theoretical standpoint of view, using Bayesian statistics.

Chapter 5. Caruana describes a specific neural network architecture and provides, along with empirical results, an elaborative discussion as to where his and others' methods are applicable.

Chapter 6. Intrator and Edelman propose a similar algorithm, which generalizes across multiple learning tasks through learning internal, low-dimensional representations.

Chapter 7. Baxter provides an algorithm for learning distance metrics in memory-based learning, which he also theoretically analyzes.

Chapter 8. Thrun surveys several methods and compares them empirically in the context of an object recognition task.

Chapter 9 (Part III). Silver and Mercer, building on Caruana's approach, introduce a method for selectively transferring knowledge across learning tasks based on a measure of task-relatedness.

Chapter 10. Thrun and O'Sullivan describe an algorithm for discovering clusters in the space of learning tasks and exploiting them for selectively transferring knowledge between them.

Chapter 11 (Part IV). Ring describes a reinforcement learning algorithm which develops hierarchical representations that boost learning when applied to families of related reinforcement learning tasks.

Chapter 12. Schmidhuber, Zhao, and Schraudolph describe a reinforcement learning algorithm capable of modifying its own learning algorithm.

Chapter 13. Maclin and Shavlik, investigating the problem of transferring knowledge from humans to machines, describe an algorithm that utilizes human advice to guide reinforcement learning.

SEBASTIAN THRUN AND LORIEN PRATT

I Overview Articles

1 LEARNING TO LEARN: INTRODUCTION AND OVERVIEW

Sebastian Thrun and Lorien Pratt

1.1 INTRODUCTION

Over the past three decades or so, research on machine learning and data mining has led to a wide variety of algorithms that learn general functions from experience. As machine learning is maturing, it has begun to make the successful transition from academic research to various practical applications. Generic techniques such as decision trees and artificial neural networks, for example, are now being used in various commercial and industrial applications (see e.g., [Langley, 1992; Widrow et al., 1994]).

"Learning to learn" is an exciting new research direction within machine learning (see e.g., a recent workshop [Caruana et al., 1996]). Similar to traditional machine learning algorithms, the methods described in this book induce general functions from experience. However, the book investigates algorithms that can change the way they generalize, i.e., practice the task of learning itself, and improve on it.

To illustrate the utility of learning to learn, it is worthwhile to compare machine learning to human learning. Humans encounter a continual stream of learning tasks. They do not just learn concepts or motor skills, they also learn *bias*, i.e., they learn how to generalize. As a result, humans are often able to generalize correctly from extremely few examples—often just a single example suffices to teach us a new thing (see e.g., [Ahn and Brewer, 1993; Hume and Pazzani, 1996; Moses et al., 1993]).

A deeper understanding of computer programs that improve their ability to learn can have a large practical impact on the field of machine learning and beyond. In recent years, the field has made significant progress towards a theory of learning to learn along with practical new algorithms, some of which led to impressive results in real-world applications. This book provides a survey of some of the most exciting new research approaches, written by leading researchers in the field. Its objective is to investigate the utility and feasibility of computer programs than can learn how to learn, both from a practical and a theoretical point of view.

1.2 DEFINITION

What does it mean for an algorithm to be capable of learning to learn? Aware of the danger that naturally arises when providing a technical definition for a folk-psychological term—even the term "learning" lacks a satisfactory technical definition—this section proposes a simplified framework to facilitate the discussion of the issues involved.

Let us begin by defining the term *learning*. According to Mitchell [Mitchell, 1993], given

1. a task,
2. training experience, and
3. a performance measure,

a computer program is said to *learn* if its performance at the task improves with experience. For example, supervised learning (see various references in [Mitchell, 1993]) addresses the task of approximating an unknown function f where the experience is in the form of training examples that may be distorted with noise. Performance is usually measured by the ratio of correct to incorrect classifications, or measured by the inverse of the squared approximation error". Reinforcement learning [Barto et al., 1995; Sutton, 1992], to name a second example, addresses the task of selecting actions so as to maximize one's reward. Here performance is the average cumulative reward, and experience is obtained through interaction with the environment, observing state, actions, and reward.

Following Mitchell's definition, we will now define what it means for an algorithm to be capable of *learning to learn*. Given

1. a family of tasks
2. training experience for each of these tasks, and
3. a family of performance measures (e.g., one for each task),

an algorithm is said to *learn to learn* if its performance at each task improves with experience *and* with the number of tasks. Put differently, a learning algorithm whose performance does not depend on the number of learning tasks, which hence would not benefit from the presence of other learning tasks, is not said to learn to learn. For

an algorithm to fit this definition, some kind of *transfer* must occur between multiple tasks that must have a positive impact on expected task-performance.

For some learning scenarios, it is easy to specify an algorithm that learns to learn. In particular, if all learning tasks are equivalent, the training experience for each individual task could just be added together, and any regular learning algorithm would, by definition, fit our notion of learning to learn. Of particular interest, however, are scenarios in which the learning tasks differ. For example, consider the tasks of learning to recognize different faces. Unless every person's face looks alike, examples of one learning task cannot blindly be used to augment the set of examples in another. However, one might hypothesize that all face recognition tasks share certain invariances (e.g., the identity of a person is invariant to the facial expression, the viewing perspective, and the illumination). If these invariances are learned and transferred across different learning tasks, an algorithm can improve its performance with the number of tasks.

1.3 ANALYSIS

Recent theoretical research on the complexity of learning have shown fundamental bounds on the performance achievable when learning from experience (see e.g., [Geman et al., 1992; Kearns and Vazirani, 1994; Valiant, 1984; Vapnik, 1982]). Algorithms that learn to learn can side-step some of these bounds, by transferring knowledge across learning tasks (see e.g., [Baxter, 1995b; Thrun, 1996]).

To see, consider the problem of learning a function from noise-free examples. The following standard result by Blumer and colleagues relates the size of the hypothesis space and the number of (noise-free) training examples required for learning a function:

> **Theorem [Blumer et al., 1987].** *Given a function f in a space of functions H, the probability that any hypothesis $h \in H$ with error larger than ε is consistent with f on a (noise-free) dataset of size x is less than $(1-\varepsilon)^x |H|$. In other words,*
>
> $$x \geq \frac{1}{-\ln(1-\varepsilon)} \left(\ln(|H|) + \ln\left(\frac{1}{\delta}\right) \right) \tag{1.1}$$
>
> *training examples suffice to ensure, with probability $1-\delta$, that any hypothesis consistent with the data will not produce an error larger than ε on future data.*

This bound is independent of the learning algorithm—it is only required that it produces a hypothesis that is consistent with the data. It also holds independently of the choice of f and the sampling distribution, as long as this distribution is the same during training and testing. Notice that Equation (1.1) is logarithmic in the hypothesis set size $|H|$.

Now consider the problem of learning n functions from (noise-free) examples. To enable a learning algorithm to improve its performance with the number of learning tasks n, let us assume that all target functions share a common set of *properties*. In

the domain of face recognition, for example, every target function might be invariant with respect to translation and scaling and facial expression. Such invariances can be understood as "properties" which all target functions obey. Suppose these properties are initially unknown. Instead, the learning algorithm considers a pool of m candidate properties, denoted by $P_1, P_2, \ldots, P_m$. The key idea here is that, by identifying the "right" properties, the hypothesis H space can be diminished, yielding more accurate generalization from less data. To simplify the formal analysis, let us assume each property P_j (with $j = 1, \ldots, m$) holds true only for a subset of all functions in H. Let p denote the fraction of functions in H which have property P_j (for reasons of simplicity we assume p is the same for all P_j). Let us also assume that all properties are *independent*, i.e., knowing that certain properties are correct for all target functions does not tell us anything about the correctness of any other property. Finally, let us make the assumption that we have an algorithm that can check with constant error $q \geq 0$ the correctness of a property P_j from the training examples of each of the n learning tasks.

This simplistic model allows to make assertions about the reduction of the hypothesis space.

> **Lemma.** *Any set of l properties that is consistent with all n learning tasks reduces the size of the hypothesis space H by a factor of p^l. The probability that this reduction removes future target functions from the hypothesis space, which will be considered a failure, is bounded above by $l\ m^l\ (p+q)^n$.*

Hence, if all learning tasks have l common properties, the correct ones can be identified with probability p^l. The proof of the lemma is straightforward and can be found in [Thrun, 1996].

Smaller hypothesis spaces yield better generalization, at least when they contain the target function. By applying the Lemma to Blumer et al.'s Theorem (1.1), the advantage of smaller hypothesis spaces can be expressed formally through the reduction in the sampling complexity when learning a new function.

> **Corollary.** *Under the conditions of the Lemma, the upper bound on the number of training examples according to Blumer et al.'s Theorem is reduced by a factor of*
>
> $$1 \quad - \quad \frac{l \ \ln\left(\frac{1}{p}\right)}{\ln\left(\frac{1}{\delta}\right) + \ln(|H|)} \tag{1.2}$$
>
> *The probability that this reduction erroneously removes the target function f is bounded above by $l\ m^l\ (p+q)^{n-1}$.*

An analogous logarithmic lower bound can be obtained using results derived by Ehrenfeucht and colleagues [Ehrenfeucht et al., 1989; Kearns and Vazirani, 1994]. A similar analysis can be found in [Baxter, 1995b].

The key idea underlying this analysis is to make explicit two levels of learning: a *meta-level* and a *base-level* [Rendell et al., 1987; Schmidhuber, 1987; Utgoff, 1986b]. The base-level learning problem is the problem of learning functions, just like regular supervised learning. The meta-level learning problem is the problem of learning properties of functions, i.e., learning entire function spaces. Learning at the meta-level bears close resemblance to base-level learning and, as best exemplified by Baxter's work [Baxter, 1995b] as well as the analysis shown here, many of the standard bounds relating accuracy to sample complexity and hypothesis space size can be applied to the meta-level as well. An important consequence is that any algorithm that learns to learn must possess bias (at both levels), just like a regular learning algorithm. Bias at the meta-level constitutes a priori assumptions concerning the relatedness of learning tasks, just like regular (base-level) bias brings to bear assumptions concerning the relation of individual data points. There does not exist a uniquely best algorithm for the general problem of learning to learn, just like there is no best algorithm for learning per se (cf. [Mitchell, 1980; Wolpert, 1994]), not outruling the fact that there might be provably best algorithms for special cases of the general problem.

1.4 REPRESENTATIONS

The key to learning to learn is representation. To improve the performance of a learning algorithm with increasing number of tasks, an algorithm must change the way it generalizes, thus, must be capable of representing knowledge that determines bias. To date, there appears to be two major families of approaches: (1) approaches that partition the parameter space of a conventional learning algorithm into task-specific parameters and general (i.e., cross-task) parameters, and (2) approaches that learn shape constraints, which are superimposed when learning a new function.

1.4.1 Partitioning the Parameter Space

Any conventional learning algorithm can be transformed into an algorithm that learns to learn by subdividing its parameters into a set of task-specific parameters, and a set of general parameters that are the same in all tasks. The first type of approach rests on this observation. All approaches outlined here search parameters in a space that combines task-specific parameters and general parameters. While in principle this distinction does not have to be rigorous, most existing algorithms in fact sharply distinguish task-specific and general parameters.

- **Recursive functional decomposition.** Functional decomposition approaches rest on the assumption that maximum performance in each task can be achieved by a function of the form $f = h_i \circ g$ (or, alternatively, $f = g \circ h_i$), where h_i is task-specific whereas g is the same for all f_is. When learning a specific f_i, training examples for f_i are used to learn both g and h_i. Since g is the

same for all tasks, knowledge about g can improve the results when learning a new function. Transfer in the functional decomposition approach is particularly effective if the complexity of g is much larger than that of the individual h_is.

Examples of functional decomposition with $f = h_i \circ g$ have become popular in recent neural network literature [Abu-Mostafa, 1993; Baxter, 1995b; Caruana, 1993; Pratt, 1993; Sharkey and Sharkey, 1992; Suddarth and Holden, 1991; Suddarth and Kergosien, 1990]. All these approaches assume that each f_i can be represented by two-layered multilayer perceptrons which share the same first hidden layer (input-to-hidden weights). Examples of the opposite functional decomposition, i.e., $f_i = g \circ h_i$, are often found in speaker-adaptive speech recognition and adaptive filtering (see e.g., [Hild and Waibel, 1993]). Here g might for example be a complex module that recognizes speech, and f_i is a low-complexity filter that increases the understandability of the signal (e.g., filter out the accent of a non-native speaker).

Both families of approaches make assumptions on the nature of the learning tasks. Baxter has shown analytically that if (1) the decomposition assumption is correct and (2) the complexity of h_i is small when compared to that of g, the reduction in sample complexity can be dramatic [Baxter, 1995b]. Various practical findings confirm these results [Abu-Mostafa, 1993; Caruana and Baluja, 1996; Suddarth and Holden, 1991].

- **Piecewise functional decomposition.** A related family of approaches rest on the assumption that each function f_i can be represented by a collection of functions $h_1, h_2, \ldots, h_m$, each of which is only partially defined (i.e., for a subspace of the input space). If the number of "building blocks" m is small compared to the number of learning tasks n and their complexity, such an approach can also reduce the sample complexity.

  Piecewise functional decomposition has been popular for learning families of sequential decision tasks [Barto et al., 1995; Sutton, 1992]. Here the assumption is that multiple policies for the selection of actions consist of the same building blocks which, once they are known, need "only" be combined to yield a new policy. In [Dayan and Hinton, 1993; Kaelbling, 1993; Lin, 1992; Ring, 1993; Ring, 1995; Singh, 1992; Thrun and Schwartz, 1995; Whitehead et al., 1993], various reinforcement learning algorithms are described that basically can learn to learn via piecewise functional decomposition (although most of them have not been proposed in this context). Most of these approaches rely on static ways to determine the appropriate pieces. For example, partial functions are defined via a hierarchy of (sub-)goals [Whitehead et al., 1993], different coarse-grained resolution in a hierarchy of control [Dayan and Hinton, 1993], Voronoi tessellation for specific geometric "landmark" states [Kaelbling, 1993], or predesigned behavioral decompositions [Lin, 1992; Singh, 1992]. In [Thrun and

Schwartz, 1995] an approach is proposed that identifies the decomposition on-the-fly, utilizing a minimum description argument.

- **Learning declarative/procedural bias.** Piecewise and recursive functional decomposition has been particularly well explored in symbolic and algorithmic approaches to machine learning that learn declarative and/or procedural knowledge. If bias is represented by rules (declarative bias) or straight program code (procedural bias), the same representation may be employed for both bias and learned functions. Consequently, knowledge acquired in one task may be used as bias in another. Examples of learning systems that modify declarative bias are *SOAR* [Laird et al., 1986], *STABB* [Utgoff, 1986a; Utgoff, 1986b], *inductive logic programming* [DeRaedt et al., 1993; Muggelton, 1992; Quinlan, 1990], *theory revision* [Mooney and Ourston, 1992], and *RALPH* [Russell, 1991]. Procedural bias is learned in *genetic programming* [Cramer, 1985; Koza, 1992; Koza, 1994; Teller, 1996; Teller and Veloso, 1996], and in an approach by Schmidhuber and colleagues [Schmidhuber, 1995; Schmidhuber, 1987; Schmidhuber, 1996]. Notice that rules and program code can be viewed as partially defined functions that are concatenated recursively, thus learning declarative bias and symbolic program code is a version that combines piecewise and recursive functional decomposition in an elegant way.

- **Learning control parameters.** Most function approximators possess control parameters. Control parameters provide a very natural way of partitioning the parameter space into two groups. Sutton proposed to learn and transfer control parameters for search in the hypothesis space [Sutton, 1992]. The choice of the inductive learning algorithm, too, can be described by a control parameter. Approaches such as *VBMS* [Rendell et al., 1987], and *MCS* [Brodley, 1994] select and combine entire learning algorithms out of a pool of algorithms. The net effect of transferring knowledge across multiple learning tasks is of course bounded above by the richness and the expressiveness of the control parameter space.

  A popular example of learning control parameters can be found in the context of *memory-based methods* such as *nearest neighbor* [Franke, 1982; Moore, 1990; Stanfill and Waltz, 1986]. The distance metric used when comparing instances in memory-based learning can be parameterized. In [Atkeson, 1991; Baxter, 1995a; Caruana, 1996; Friedman, 1994; Hastie and Tibshirani, 1994; Mel, 1996; Moore et al., 1992; Thrun and O'Sullivan, 1996], various algorithms are described for adjusting a parameterized *distance metric*. Assuming that the "optimal" distance metric, i.e., the metric which yields the best generalization performance given a fixed amount of data, is the same in *all* tasks, approaches that optimize the distance metric based on multiple tasks effectively fit our definition of learning to learn.

1.4.2 *Learning Constraints*

A second family of algorithms learns constraints. Constraints are usually represented by separate data structures, thus, constraint learning algorithms cannot be obtained by partitioning the parameter space of a conventional machine learning algorithm. The number of existing algorithms that falls into this category is considerably smaller.

- **Synthetic data.** Training data imposes constraints on the function to be learned. Thus, one way to impose constraints on a function approximator is to generate new synthetic data from real data (cf. [Pomerleau, 1993]). The assumptions underlying such an approach are (1) that the "rules" for transforming data are the same in all learning task, and (2) that they are considerably easy to learn, so that once learned, real data can be replaced or augmented by artificially synthesized data. Often, these rules correspond to the invariances in the domain. Beymer and Poggio [Beymer and Poggio, 1995] have exploited this idea to generate virtual views of faces from a single view of a face, based on a generic technique for learning pose parameters in face recognition [Poggio and Vetter, 1992]. Since the transformations are *learned*, the performance of the system increases with the number of learning tasks.

- **Slope constraints.** A second group of constraint learning approaches learns *slope constraints*. For each data point, previously acquired knowledge is used to constrain the slope of the function to be learned.

  The idea of incorporating slopes into neural network learning is due to Simard and colleagues [Simard et al., 1992], who proposed to encode certain (known) invariances in character recognition by directional slope information. The EBNN [Thrun, 1996; Thrun and Mitchell, 1993] algorithm extends their approach in that it *learns* the slope constraints. EBNN rests on the same assumption as the recursive functional decomposition approaches listed above: new target functions can be composed of (themselves and) previously learned functions. Instead of "constructing" new target function based on previously learned function, however, EBNN derives slope information, which is then used to constrain the target function. EBNN has been shown empirically to be robust to errors in various application domains (such as chess, object recognition, robot control—see [Thrun, 1996]).

- **Internal constraints.** Constraints can also be learned and superimposed for internal aspects of the function approximator. For example, Lando/Edelman [Lando and Edelman, 1995] proposed in the context of face recognition to learn the "directions" (sub-manifolds) along which face images are invariant. This is done by learning changes in activations when faces are rotated or translated, in a specific internal representational space. The invariances are assumed to be

equivalent for all faces—hence once learned, they can be used to project new faces back into a canonical (frontal) view, in which they are easier to recognize.

A more detailed overview specifically of connectionist approaches can be found in Chapter 2. The reader may notice that not all of the approaches listed here have been applied to the problem of learning to learn. The number of algorithms that have been systematically evaluated in the context of multiple learning tasks is still considerably small.

1.4.3 Other Issues

Apart from different representations, researchers have explored various facets of the general problem.

- **Incremental vs. non-incremental approaches.** Learning tasks can attacked incrementally one by one, or all in parallel. Both methodologies have potential advantages and disadvantages. If tasks arrive one after another (see e.g., [Pratt, 1993; Sharkey and Sharkey, 1992]), incremental approaches do not have to memorize training data, thus consume less memory. However, non-incremental approaches (cf. [Abu-Mostafa, 1993; Baxter, 1995b; Caruana, 1993; Suddarth and Holden, 1991; Suddarth and Kergosien, 1990]) might discover commonalities between different learning tasks that are difficult to find if learning tasks are processed sequentially [Caruana, 1996].

- **Unselective vs. selective transfer.** Most approaches weigh learning tasks equally when transferring knowledge between them. As shown in a recent study [Thrun and O'Sullivan, 1996], transferring knowledge *un-selectively* might hurt the overall performance in cases where the learning tasks do not meet the inductive assumptions (implicit or explicit) underlying the learning algorithm. Approaches that examine the relation of learning tasks and transfer knowledge *selectively*, such as the *TC* algorithm proposed in [Thrun and O'Sullivan, 1996], are more robust to unrelated learning tasks.

- **Data sharing.** In some scenarios, data is partially shared between different learning tasks. For example, Suddarth's and AbuMostafa's *learning from hints* [Abu-Mostafa, 1993; Suddarth and Holden, 1991; Suddarth and Kergosien, 1990] and Caruana's *multitask learning* [Caruana, 1993; Caruana, 1996; Caruana and Baluja, 1996] rests on the assumption that the input patterns are the same across all tasks; only the output labels differ. In contrast, Baxter's theoretical analysis of the same architecture [Baxter, 1995b] assumes that training examples are generated independently for each learning task.

- **Initial search bias vs. search constraints.** A merely technical matter is the way knowledge from other tasks is incorporated. Some approaches use previously

learned knowledge as an initial point for the parameter search (see e.g., [Pratt, 1993]), whereas others incorporate this knowledge as a constraint during the search (see e.g., [Mitchell and Thrun, 1993; Thrun and Mitchell, 1993]). In a recent study, O'Sullivan has compared both methodologies empirically and characterized the key advantages of each of them [O'Sullivan, 1996].

- **Performance tasks.** In some scenarios, the performance for all tasks is important, whereas others contain a designated *performance task* which is the only task whose performance matters. Baxter [Baxter, 1995b] has analyzed both cases for a particular architecture for which he pointed out commonalities and differences.

1.5 PERSPECTIVE

Surprisingly many real-world learning scenarios naturally give rise to multiple learning problems, hence providing the opportunity for synergy between them. For example, a mobile service-robot that is being trained to find-and-fetch objects might in fact learn and benefit from a variety of other tasks, such as perceptual tasks (recognize an object, or recognize a landmark), control tasks (avoid collisions with obstacles), prediction tasks (predict the location of obstacles or objects), to name just a few. There is a huge variety of other application domains that naturally contain families of learning tasks, e.g.,

- cursive handwriting recognition,
- speech recognition,
- computer vision, e.g., face recognition or object recognition,
- stock market analysis and prediction,
- language acquisition,
- personalized user interfaces,
- software agents, e.g., Internet agents,

and numerous others. It is up to the imagination and expertise of the reader to find more!

Traditional machine learning approaches have often tackled learning problems separately; the thoughts and results in this book let us hope that by learning them simultaneously, better performance can be achieved from less data. Putting technical detail aside, the ideas and algorithms described here appear to applicable to any application that involves *cheap data* and *expensive data*. This includes systems that must be trained by a customer (where data is often expensive), and that can practice the learning task itself while still at the factory (where data is cheap). We also believe that a deeper understanding of learning to learn will, in the long run, provide new explanations for human learning abilities, as the change of bias and the transfer of knowledge appears to play an important role in human generalization (see e.g., [Ahn and Brewer,

1993; Ahn et al., 1987; Hume and Pazzani, 1996; Moses et al., 1993] and a chapter by Robins in the same volume).

Acknowledgments

The idea of putting together this book has been sparked through a workshop entitled "Learning to Learn: Knowledge Consolidation and Transfer in Inductive Systems," which was led by Richard Caruana, Daniel L. Silver and co-organized by Jonathan Baxter, Tom M. Mitchell, Lorien Y. Pratt, and myself as part of the NIPS workshops in Vail in December 1995 [Caruana et al., 1996]. Most of the contributing authors participated in the workshop.

References

Y. S. Abu-Mostafa. A method for learning from hints. In S. J. Hanson, J. Cowan, and C. L. Giles, editors, *Advances in Neural Information Processing Systems 5*, pages 73–80, San Mateo, CA, 1993. Morgan Kaufmann.

W.-K. Ahn and W. F. Brewer. Psychological studies of explanation-based learning. In G. DeJong, editor, *Investigating Explanation-Based Learning*. Kluwer Academic Publishers, Boston/ Dordrecht/London, 1993.

W.-K. Ahn, R. Mooney, W. F. Brewer, and G. F. DeJong. Schema acquisition from one example: Psychological evidence for explanation-based learning. In *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, Seattle, WA, July 1987.

C. A. Atkeson. Using locally weighted regression for robot learning. In *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, pages 958–962, Sacramento, CA, April 1991.

A. G. Barto, S. J. Bradtke, and S. P. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72:81–138, 1995.

J. Baxter. The Canonical Distortion Measure for Vector Quantization and Function Approximation. Chapter 7 in this book.

J. Baxter. *Learning Internal Representations*. PhD thesis, Flinders University, Australia, 1995.

D. Beymer and T. Poggio. Face recognition from one model view. In *Proceedings of the International Conference on Computer Vision*, 1995.

A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Occam's razor. *Information Processing Letters*, 24:377–380, 1987.

C.E. Brodley. *Recursive Automatic Algorithm Selection for Inductive Learning*. PhD thesis, University of Massachusetts, Amherst, MA 01003, August 1994. also available as COINS Technical Report 94-61.

R. Caruana. Multitask learning: A knowledge-based of source of inductive bias. In P. E. Utgoff, editor, *Proceedings of the Tenth International Conference on Machine Learning*, pages 41–48, San Mateo, CA, 1993. Morgan Kaufmann.

R. Caruana. Algorithms and applications for multitask learning. In L. Saitta, editor, *Proceedings of the Thirteenth International Conference on Machine Learning*, San Mateo, CA, July 1996. Morgan Kaufmann.

R. Caruana and S. Baluja. Using the future to 'sort out' the present: Rankprop and multitask learning for medical risk evaluation. In D. Touretzky, M. Mozer, and M.E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, Cambridge, MA, 1996. MIT Press. to appear.

R. Caruana, D.L. Silver, J. Baxter, T.M. Mitchell, L.Y. Pratt, and Thrun. S. Workshop on "Learning
to learn: Knowledge consolidation and transfer in inductive systems". Workshop, held at NIPS-95, Vail, CO, see World Wide Web at `http://www.cs.cmu.edu/afs/cs.cmu.edu/user/caruana/pub/transfer.html`, December 1995.

N.L. Cramer. A representation for the adaptive generation of simple sequential programs. In J.J. Grefenstette, editor, *Proceedings of First International Conference on Genetic Algorithms and their Applications*, pages 183–187, Pittsburgh, PA, 1985.

P. Dayan and G. E. Hinton. Feudal reinforcement learning. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems 5*, San Mateo, CA, 1993. Morgan Kaufmann.

L. DeRaedt, N. Lavrač, and S. Džeroski. Multiple predicate learning. In *Proceedings of IJCAI-93*, pages 1037–1042, Chamberry, France, July 1993. IJCAI, Inc.

A. Ehrenfeucht, D. Haussler, M. Kearns, and L. Valiant. A general lower bound on the number of examples needed for learning. *Information and Computation*, 82:247–261, 1989.

R. Franke. Scattered data interpolation: Tests of some methods. *Mathematics of Computation*, 38(157):181–200, January 1982.

J. H. Friedman. Flexible metric nearest neighbor classification. November 1994.

S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–58, 1992.

T. Hastie and R. Tibshirani. Discriminant adaptive nearest neighbor classification. Submitted for publication, December 1994.

H. Hild and A. Waibel. Multi-speaker/speaker-independent architectures for the multi-state time delay neural network. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pages II 255–258. IEEE, April 1993.

T. Hume and M.J. Pazzani. Learning sets of related concepts: A shared task model. In *Proceedings of the Eighteenth Annual Conference of the Cognitive Science Society*, 1996.

L. P. Kaelbling. Hierarchical learning in stochastic domains: Preliminary results. In P. E. Utgoff, editor, *Proceedings of the Tenth International Conference on Machine Learning*, pages 167–173, San Mateo, CA, 1993. Morgan Kaufmann.

M. Kearns and U. Vazirani. *Introduction to Computational Learning Theory*. MIT Press, Cambridge, MA, 1994.

J. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, 1992.

J. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs.* MIT Press, Cambridge, MA, 1994.

J. Laird, P. Rosenbloom, and A. Newell. Chunking in SOAR: The anatomy of a general learning mechanism. *Machine Learning*, 1(1):11–46, 1986.

M. Lando and S. Edelman. Generalizing from a single view in face recognition. Technical Report CS-TR 95-02, Department of Applied Mathematics and Computer Science, The Weizmann Institute of Science, Rehovot 76100, Israel, January 1995.

P. Langley. Areas of application for machine learning. In *Proceedings of the Fifth International Symposium on Knowledge Engineering*, Sevilla, 1992.

L.-J. Lin. *Self-supervised Learning by Reinforcement and Artificial Neural Networks.* PhD thesis, Carnegie Mellon University, School of Computer Science, Pittsburgh, PA, 1992.

B. Mel. Seemore: A view-based approach to 3-d object recognition using multiple visual cues. In M.C. Mozer D.S. Touretzky and M.E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8.* MIT Press, December 1996.

T. M. Mitchell. The need for biases in learning generalizations. Technical Report CBM-TR-117, Computer Science Department, Rutgers University, New Brunswick, NJ 08904, 1980. Also appeared in: *Readings in Machine Learning*, J. Shavlik and T.G. Dietterich (eds.), Morgan Kaufmann.

T. M. Mitchell. *Machine Learning.* McGraw-Hill, NY, in preparation.

T. M. Mitchell and S. Thrun. Explanation-based neural network learning for robot control. In S. J. Hanson, J. Cowan, and C. L. Giles, editors, *Advances in Neural Information Processing Systems 5*, pages 287–294, San Mateo, CA, 1993. Morgan Kaufmann.

R. J. Mooney and D. Ourston. A multistrategy approach to theory refinement. In R.S. Michalski and G. Teccuci, editors, *Proceedings of the International Workshop on Multistrategy Learning*, pages 207–214. Morgan Kaufmann, 1992.

A. W. Moore. *Efficient Memory-based Learning for Robot Control.* PhD thesis, Trinity Hall, University of Cambridge, England, 1990.

A. W. Moore, D. J. Hill, and M. P. Johnson. An Empirical Investigation of Brute Force to choose Features, Smoothers and Function Approximators. In S. Hanson, S. Judd, and T. Petsche, editors, *Computational Learning Theory and Natural Learning Systems, Volume 3.* MIT Press, 1992.

Y. Moses, S. Ullman, and S. Edelman. Generalization across changes in illumination and viewing position in upright and inverted faces. Technical Report CS-TR 93-14, Department of Applied Mathematics and Computer Science, The Weizmann Institute of Science, Rehovot 76100, Israel, 1993.

S. Muggelton. *Inductive Logic Programming.* Academic Press, New York, 1992.

J. O'Sullivan. Integrating initialization bias and search bias in artificial neural networks. Internal report, January 1996.

T. Poggio and T. Vetter. Recognition and structure from one 2d model view: Observations on prototypes, object classes and symmetries. A.I. Memo No. 1347, 1992.

D. A. Pomerleau. Knowledge-based training of artificial neural networks for autonomous robot driving. In J. H. Connell and S. Mahadevan, editors, *Robot Learning*, pages 19–43. Kluwer Academic Publishers, 1993.

L. Y. Pratt. *Transferring Previously Learned Back-Propagation Neural Networks to New Learning Tasks*. PhD thesis, Rutgers University, Department of Computer Science, New Brunswick, NJ 08904, May 1993. also appeared as Technical Report ML-TR-37.

L.Y. Pratt and B. Jennings. A review of transfer between connectionist networks. *Connection Science*, 8(2):163–184, 1996. Reprinted as Chapter 2 in this book.

J. R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

L. Rendell, R. Seshu, and D. Tcheng. Layered concept-learning and dynamically-variable bias management. In *Proceedings of IJCAI-87*, pages 308–314, 1987.

M. B. Ring. Two methods for hierarchy learning in reinforcement environments. In *From Animals to Animats 2: Proceedins of the Second International Conference on Simulation of Adaptive Behavior*, pages 148–155. MIT Press, 1993.

M. B. Ring. *Continual Learning in Reinforcement Environments*. R. Oldenbourg Verlag, München, Wien, 1995.

S.J. Russell. Prior knowledge and autonomous learning. *Robotics and Autonomous Systems*, 8:145–159, 1991.

J. H. Schmidhuber. On learning how to learn learning strategies. Technical Report FKI-198-94, Technische Universität München, January 1995. Revised version.

J.H. Schmidhuber. Evolutionary principles in self-referential learning, or on learning how to learn: The meta-meta-... hook. Master's thesis, Technische Universität München, München, Germany, 1987.

J.H. Schmidhuber. A general method for incremental self-improvement and multi-agent learning in unrestricted environments. In X. Yao, editor, *Evolutionary Computation: Theory and Applications*, Singapore, 1996. Scientific Publishing Co.

N. E. Sharkey and A. J. C. Sharkey. Adaptive generalization and the transfer of knowledge. In *Proceedings of the Second Irish Neural Networks Conference*, Belfast, 1992.

B. Silver. *Using Meta-level inference to Constrain Search and to Learn Strategies in Equation Solving*. PhD thesis, Department of Artificial Intelligence, University of Edinburgh, 1984.

P. Simard, B. Victorri, Y. LeCun, and J. Denker. Tangent prop – a formalism for specifying selected invariances in an adaptive network. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 895–903, San Mateo, CA, 1992. Morgan Kaufmann.

S. P. Singh. Transfer of learning by composing solutions for elemental sequential tasks. *Machine Learning*, 8, 1992.

C. Stanfill and D. Waltz. Towards memory-based reasoning. *Communications of the ACM*, 29(12):1213–1228, December 1986.

S. C. Suddarth and A. Holden. Symbolic neural systems and the use of hints for developing complex systems. *International Journal of Machine Studies*, 35, 1991.

S. C. Suddarth and Y. L. Kergosien. Rule-injection hints as a means of improving network performance and learning time. In *Proceedings of the EURASIP Workshop on Neural Networks*, Sesimbra, Portugal, Feb 1990. EURASIP.

R. S. Sutton. Adapting bias by gradient descent: An incremental version of delta-bar-delta. In *Proceeding of Tenth National Conference on Artificial Intelligence AAAI-92*, pages 171–176, Menlo Park, CA, July 1992. AAAI, AAAI Press/The MIT Press.

R. S. Sutton, editor. *Reinforcement Learning*. Kluwer Academic Publishers, Boston, MA, 1992.

A. Teller. Evolving programmers: The co-evolution of intelligent recombination operators. In P. Angeline and K. Kinnear, editors, *Advances in Genetic Programming II*, Cambridge, MA, 1996. MIT Press.

A. Teller and M. Veloso. PADO: A new learning architecture for object recognition. In K. Ikeuchi and M. Veloso, editors, *Symbolic Visual Learning*. Oxford University Press, 1996.

S. Thrun. *Explanation-Based Neural Network Learning: A Lifelong Learning Approach*. Kluwer Academic Publishers, Boston, MA, 1996.

S. Thrun and T. M. Mitchell. Integrating inductive neural network learning and explanation-based learning. In *Proceedings of IJCAI-93*, Chamberry, France, July 1993. IJCAI, Inc.

S. Thrun and J. O'Sullivan. Discovering structure in multiple learning tasks: The TC algorithm. In L. Saitta, editor, *Proceedings of the Thirteenth International Conference on Machine Learning*, San Mateo, CA, July 1996. Morgan Kaufmann.

S. Thrun and A. Schwartz. Finding structure in reinforcement learning. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems 7*, Cambridge, MA, 1995. MIT Press.

P. E. Utgoff. *Machine Learning of Inductive Bias*. Kluwer Academic Publishers, 1986.

P. E. Utgoff. Shift of bias for inductive concept learning. In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach, Volume II*. Morgan Kaufmann, 1986.

L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, 1984.

V. Vapnik. *Estimations of dependences based on statistical data*. Springer Publisher, 1982.

S. Whitehead, J. Karlsson, and J. Tenenberg. Learning multiple goal behavior via task decomposition and dynamic policy merging. In J. H. Connell and S. Mahadevan, editors, *Robot Learning*, pages 45–78. Kluwer Academic Publishers, 1993.

B. Widrow, D. E. Rumelhart, and M. A. Lehr. Neural networks: Applications in industry, business and science. *Communications of the ACM*, 37(3):93–105, March 1994.

D. H. Wolpert. Off-training set error and a priori distinctions between learning algorithms. Technical Report SFI TR 95-01-003, Santa Fe Institute, Santa Fe, NM 87501, 1994.

2 A SURVEY OF CONNECTIONIST NETWORK REUSE THROUGH TRANSFER

Lorien Pratt and Barbara Jennings

That article was originally published in *Connection Science*, Vol. 8, No. 2, 1996.

Abstract: Connectionist networks that have learned one task can be reused on related tasks in a process that is called "transfer". This paper surveys recent work on transfer. A number of distinctions between kinds of transfer are identified, and future directions for research are explored. The study of transfer has a long history in cognitive science. Discoveries about transfer in human cognition can inform applied efforts. Advances in applications can also inform cognitive studies.

2.1 INTRODUCTION

Connectionist learning most often depends only on the training data that is available for a given task. However, as connectionist networks become more widely used, there is an increasing need for methods that avoid "reinventing the wheel" by utilizing other sources of information. One source, which is the focus of this book, are networks that learn related tasks. Learning methods that facilitate such communication between

tasks are said to perform learning "transfer". Typically, the network from which information is extracted is called the "source"; it is transferred to the "target" network.

Connectionist networks that can exploit the relationships between tasks are a natural extension to today's single-task learning systems, and so represent an important step in the maturation of the field of connectionist software systems. Systems that include transfer components are sometimes said to do "lifelong learning" [Thrun and Mitchell, 1993]. Transfer can also be viewed as facilitating the reuse of connectionist software components.

This article surveys recent connectionist transfer efforts.

Transfer example. A common theme to transfer efforts is the presence of two or more tasks, between which information is shared. For example, consider an image recognition network that is trained under a particular set of lighting conditions. It may be possible to use this network in a different location, or when the lighting has changed, and to obtain reasonable accuracy. However, if the two sets of conditions are sufficiently different, then the accuracy of the old network may be inadequate for the new task. In such cases it is necessary to collect training data for the new condition. The question is then how best to use this new data. The traditional answer is to train networks from random initial weights when data changes. This approach ignores the possibly relevant network that was trained on the previous task, and which might be exploited to facilitate learning. Instead, by using the previous network as a starting point, learning can be expedited and/or generalization performance can be improved.

A related paradigm is where two networks for different lighting conditions share information *during* the learning process. This constraint can provide an important bias. The contrast between this and the previous example is the time scale during which the information is shared – here the two tasks are learned simultaneously. Following Silver and Mercer [Silver and Mercer, 1996], we will refer to this paradigm as *functional* transfer and the previous situation, where the tasks are learned at different times, as *representational* transfer [Baxter, 1996].

Why is transfer important, and to whom. The study of transfer is relevant to both cognitive modeling and to connectionist applications. On the modeling side, it makes sense to study methods that allow learners to build on their past experience – people do not begin with a tabula rasa every time they learn, but instead bring to bear their experience with similar problems. Transfer issues have a long tradition of study in cognitive modeling literature. In a companion article, Anthony Robins [Robins, 1996b] describes the history of transfer in cognition.

It is important to study transfer for applications because connectionist network learning often takes a very long time (on the order of days or weeks for reasonably complex problems). The amount of training time seems, in some settings, to be an exponential function of problem size [Judd, 1988]. As connectionist networks are being

used more frequently outside of a research setting, and more complex learning tasks are being addressed, this problem is becoming more pronounced (cf. [Hertz et al., 1991; Fahlman and Lebiere, 1990; Sony, 1992]).

Neurophysiological research indicates that systems of neurons begin learning tasks to some extent "pre-wired" for some purpose; learning is a process of updating existing connections for new tasks (cf. [Durbin et al., 1989], [Toates, 1980]). Transfer is also more generally important because the population of connectionist networks, whose experience can be exploited, is growing.

Finally, training data is often scarce. If it can be supplemented with the results of prior learning, then asymptotic classifier accuracy can potentially be improved.

2.1.1 Overview.

In this article, Section 2.2 presents a number of distinctions that provide a framework for understanding different formulations of the transfer problem. Section 2.3 then addresses methods that we are not considering as connectionist transfer, and so are not reviewed in detail. These methods are closely related, however, and so bear mention. Sections 2.4 and 2.5 then survey several recent articles in the areas of representational and functional transfer, respectively, and describe how they fit this framework. Papers are presented chronologically within each section. Section 2.6 revisits selected aspects of the transfer process, identifying areas for future study.

This article attempts to describe a representative group of recent work on the transfer problem. We apologize for any omissions.

2.2 IMPORTANT DISTINCTIONS BETWEEN APPROACHES TO TRANSFER

Approaches to transfer can be characterized by the following distinctions.

Functional vs. representational transfer: Recall from Section 2.1 that, in functional transfer, learning in the source and target happens simultaneously. In representational transfer, source and target learning are separate in time, and an explicit representation is transferred from one to the other. Functional transfer methods may share different parts of the network. In [Caruana, 1993] and [Silver and Mercer, 1996], networks share the same input and hidden representations. In [de Sa, 1994], however, the output representations are shared but the network inputs come from different modalities.

There is a larger body of literature on representational transfer, so most of the following distinctions apply there; some are illustrated in Figure 2.1.

Purpose of transfer: applications vs. modeling: Some researchers build learning algorithms for cognitive modeling instead of focusing on success on applied tasks,

One particular kind of non-literal transfer happens when the transferred information is translated through a different representational formalism, such as rules, on the way to being used in the target network [Fu, 1991; Towell et al., 1990; Towell and Shavlik, 1992; Berenji and Khedkar, 1992]. By using a technique that converts weights into rules, followed by one that converts rules into weights, a kind of non-literal transfer could be possible.

In practice, many studies have inserted rules that are human generated into networks, rather than using rules that have in turn been extracted from other networks to solve different tasks. Although these methods are effective and closely related to transfer, we do not consider them transfer nor review them here. This is because we have limited our scope to methods that use information from different but related learning tasks, and not different representations of the same task.

Using general learners vs. specialized transfer algorithms: Some approaches to transfer have explored how specialized learning methods can be designed to work in a transfer setting [Naik et al., 1992; Agarwal et al., 1992; Sutton, 1992]. An alternative is to retain back-propagation [Rumelhart et al., 1987], conjugate gradient (cf. [Barnard, 1992]), or other algorithms that are not designed specifically for transfer.

Availability of source data: If source training data is available to the transfer method, it may be possible to determine how the problem changes from source to target by comparing source and target data sets. This may or may not be a reasonable assumption, but if it is, a more accurate estimation of the invariances present is possible. Note that the source data is typically part of the training process in functional transfer – this may be a source of the power of these methods.

2.3 METHODS NOT CONSIDERED HERE

We have drawn the admittedly fluid boundary around what we consider as connectionist transfer methods for two purposes: (1) to follow the definition of transfer in the current literature and (2) to limit the number of articles considered to a reasonable size. Towards this end, we focus only on situations where there is a clear division between two or more tasks between which information is shared. This means that adaptive methods where the task to be solved changes gradually are not included [Rumelhart and McClelland, 1986; Barron et al., 1984; Elman, 1989; Carpenter and Grossberg, 1991]. In these systems, use of the classifier alternates with further training. In contrast, the transfer formulation considered here involves only one movement of information from the source to the target task, and the target task is different from the source (it is not just the source with more training data added).

In this section we review a number of topics that, although their solutions may inform transfer efforts, do not fit this definition, so are not covered in the remainder of this paper.

Invariance. We do not include methods that learn invariances within a single task. For example, [Poggio et al., 1993] describe a system for face recognition that converts all faces to a canonical pose before recognition. Although the conversion in this setting is done only within the single task of face recognition, the same approach might be used for between-task transfer, for instance to convert a network that could recognize faces in one pose to a network that could recognize faces in another – clearly a related task. A biologically plausible approach to a similar problem is also addressed by [Lando and Edelman, 1995]. In general, systems that learn to ignore variations in their input may need to include transformations that convert inputs to a canonical representation. Such a transformation may be useful in a transfer setting.

These studies illustrate the fact that whether a study can be viewed as "transfer" or not depends on the boundaries that are drawn separating one "task" from another. If recognizing any face correctly is considered a single task, then no transfer occurs in the above systems, only good generalization over varying inputs. This might be viewed as analogous to, say, a network to solve "the" speaker independent speech recognition task. Another recent example is a patient-adaptive ECG monitoring system described by [Watrous and Towell, 1995].

If, at the other extreme, every different person's face is considered a "task", then this is transfer according to our definition, even though the algorithms used to support both situations are the same. This distinction is not so blurred in representational transfer, where the difference between tasks may be reflected by the difference between training sets available at different times. In functional transfer, however, especially where multiple output units represent a variety of tasks, the difference between learning many tasks and learning many output units for a single "task" is less clear.

Hybrid representations. Another representational conversion that is sometimes done within a single task is typified by [Kubat, 1996], who converts a decision tree representation into a connectionist network in an attempt to gain benefits of each. Several other efforts involve such hybrid representations (cf. [Wynne-Jones, 1992]); the mechanism that is used for this combination could also be used for a between-task transfer algorithm.

Analogy. Transfer as presented here has strong conceptual ties to the idea of analogy that is studied in symbolic machine learning (see [Hall, 1988] for a survey). There are a many methods in that field that use information from a source problem to aid in solving a related target task. This is exactly the intuition behind transfer. The difference is the role of analogy in the overall learning process. Transfer in connectionist

networks uses analogy to aid in inductive learning, meaning that it assumes the availability of a set of training examples that represent the target task. Here, the role of transfer is supplemental, providing an additional source of learning bias. In contrast, most machine learning analogy methods are not integrated with an inductive component; rather all learning is done through analogy. One consequence of this difference for connectionist systems is that the inductive component allows target learning to be robust against a faulty analogy source – the inductive component can "take over" when the source information is not adequate. A secondary point of contrast to older analogy work is that we focus here on connectionist representations, whereas symbolic machine learning analogy representations are usually logical.

The framework for understanding analogy introduced by [Hall, 1988] is nonetheless instructive when examining transfer methods. There, the solution of a new problem using analogy is a four step process:

1. **Recognition**: the relationship between the target task and an analogous source is identified.
2. **Elaboration**: the relationship between the source and target are explored. Central to the elaboration phase of many machine learning efforts is the process of *abstraction*, where aspects of the learner that are relevant to the source but not the target task are removed from the source.
3. **Evaluation**: the effectiveness of the analogical mapping is determined.
4. **Consolidation**: the new task is stored in such a way as to facilitate recognition and use in future tasks.

Overall, there is far less focus on recognition in connectionist efforts than in analogy. Consolidation usually occurs, but not as a separate step from the consolidation of the training data that happens anyway during source learning. Exceptions are [Robins, 1996a] and [Silver and Mercer, 1995], who show how a network can learn to consolidate training information from other networks. There is also not usually as clear a distinction between the elaboration and evaluation phases in connectionist systems as in analogy.

Non-literal transfer as reviewed in Section 2.2 approaches may be seen as involving an elaboration phase, where the source and target are compared. Literal transfer may be interpreted as skipping any elaboration and simply applying the source problem to the initial conditions of target task directly, though the further training that is done after that point might be considered a sort of elaboration. A deeper review of the relationship between neural networks and analogy can be found in [Holyoak and Barnden, 1994].

Subtasking. A number of studies have shown how learning can be improved by using networks constructed for subtasks. [Waibel, 1989] showed how learning could

Index