Draft, June 30, 1996

Learning To Learn: Introduction

SEBASTIAN THRUN http://www.cs.cmu.edu/~thrun/
Computer Science Department and Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213

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., [31, 75]).

"Learning to learn" is an exciting new research direction within machine learning [14]. 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., [2, 24, 40]).

A deeper understanding of computer programs that improve their ability to learn has potentially an enormous practical impact. 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.

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 [35], 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 is expected to improve with experience. For example, supervised learning (see various references in [35]) 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 inverse proportional to the squared approximation error. Reinforcement learning [5, 63], 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 is expected to improve 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.

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., [21, 26, 72, 73]). Algorithms that learn to learn can side-step some of these bounds, by transferring knowledge across learning tasks (see e.g., [7, 66]).

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 [9].** *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,*
>
> $$x \ \geq\ \frac{1}{-\ln(1-\varepsilon)}\left(\ln(|H|)+\ln\left(\frac{1}{\delta}\right)\right) \tag{1}$$
>
> *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) 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 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 [66].

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), 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*

$$1 \;-\; \frac{l\ \ln\left(\frac{1}{p}\right)}{\ln\left(\frac{1}{\delta}\right) + \ln(|H|)} \tag{2}$$

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 [18, 26]. A similar analysis can be found in [7].

The key idea underlying this analysis is to make explicit two levels of learning: a *meta-level* and a *base-level* [48, 53, 71]. 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 [7] 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. [34, 76]), not outruling the fact that there might be provably best algorithms for special cases of the general problem.

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.

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 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 rigor, 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 [1, 7, 11, 45, 55, 60, 61]. 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., [23]). 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 [7]. Various practical findings confirm these results [1, 13, 60].

- **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 [5, 63]. 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 [16, 25, 32, 49, 50, 58, 69, 74], 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 [74], different coarse-grained resolution in a hierarchy of control [16], Voronoi tessellation for specific geometric "landmark" states [25], or pre-designed behavioral decompositions [32, 58]. In [69] 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* [29], *STABB* [70, 71], *inductive logic programming* [17, 41, 47], *theory revision* [37], and *RALPH* [51]. Procedural bias is learned in *genetic programming* [15, 27, 28, 64, 65], and in an approach by Schmidhuber and colleagues [52, 53, 54]. 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 [62]. The choice of the inductive learning algorithm, too, can be described by a control parameter. Approaches such as *VBMS* [48], and *MCS* [10] 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* [19, 38, 59]. The distance metric used when comparing instances in memory-based learning can be parameterized. In [4, 6, 12, 20, 22, 33, 39, 68], 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.

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. [44]). 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 [8] 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 [43]. 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 [57], who proposed to encode certain (known) invariances in character recognition by directional slope information. The EBNN [66, 67] 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 [66]).

- **Internal constraints.** Constraints can also be learned and superimposed for internal aspects of the function approximator. For example, Lando/Edelman [30] 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 [46]. 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 has been systematically evaluated in the context of multiple learning tasks is still considerably small.

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., [45, 55]), incremental approaches do not have to memorize training data, thus consume less memory. However, non-incremental approaches (cf. [1, 7, 11, 60, 61]) might discover commonalities between different learning tasks that are difficult to find if learning tasks are processed sequentially [12].

- **Unselective vs. selective transfer.** Most approaches weight learning tasks equally when transferring knowledge between them. As shown in a recent study [68], 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 [68], 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* [1, 60, 61] and Caruana's *multitask learning* [11, 12, 13] 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 [7] 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., [45]), whereas others incorporate this knowledge as a constraint during the search (see e.g., [36, 67]). In a recent study, O'Sullivan has compared both methodologies empirically and characterized the key advantages of each of them [42].

- **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 [7] has analyzed both cases for a particular architecture for which he pointed out commonalities and differences.

5. Perspective

Surprisingly many real-world learning scenarios naturally give raise to multiple learning problems, hence provide 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! A recent paper by Caruana [12] provides an excellent discussion of real-world problems that provide the opportunity for learning from multiple tasks.

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., [2, 3, 24, 40]).

6. Outline

`[[Outline of the book]]`

- **`text suggested by J. Schmidhuber, might be modified once I've read the chapter:`** Schmidhuber et al.'s approach to "learning to learn," e.g., [53, 52, 54]. The learner leads a single life during which actions are continually executed according to the systems internal state and current "policy"—an action is considered a learning algorithm if it can also modify the system's policy. There is no explicit difference between learning, meta-learning, meta-meta-learning, etc. Effects of learning processes on later learning processes are measured using reward/time ratios. Occasional backtracking enforces histories of policy modifications corresponding to lifelong reward accelerations,

Acknowledgements

The idea of putting together this book has been sparked through an exciting 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 [14]. Most of the contributors to this book participated in the workshop. Some of the papers in this volume will also appear in the international journal Machine Learning, in a special issue edited by Lorien Y. Pratt and myself.

The introduction has already begun to benefit from discussions with various contributors to the book.

References

1. 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.
2. 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.
3. 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.
4. 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.
5. A. G. Barto, S. J. Bradtke, and S. P. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, to appear.
6. J. Baxter. The canonical metric for vector quantization. Submitted for publication, 1995.
7. J. Baxter. *Learning Internal Representations*. PhD thesis, Flinders University, Australia, 1995.

8. D. Beymer and T. Poggio. Face recognition from one model view. In *Proceedings of the International Conference on Computer Vision*, 1995.
9. A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Occam's razor. *Information Processing Letters*, 24:377–380, 1987.
10. 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.
11. 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.
12. 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.
13. R. Caruana and S. Baluja. Using the future to 'sort out' the present: Rankprop and multitask learning for medical risk evaluation. In D. Touretzky and M. Mozer, editors, *Advances in Neural Information Processing Systems 8*, Cambridge, MA, 1996. MIT Press. to appear.
14. 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.
15. 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.
16. 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.
17. L. DeRaedt, N. Lavrač, and S. Džeroski. Multiple predicate learning. In *Proceedings of IJCAI-93*, pages 1037–1042, Chamberry, France, July 1993. IJCAI, Inc.
18. 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.
19. R. Franke. Scattered data interpolation: Tests of some methods. *Mathematics of Computation*, 38(157):181–200, January 1982.
20. J. H. Friedman. Flexible metric nearest neighbor classification. November 1994.
21. S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–58, 1992.
22. T. Hastie and R. Tibshirani. Discriminant adaptive nearest neighbor classification. Submitted for publication, December 1994.
23. 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.
24. 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.
25. 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.
26. M. Kearns and U. Vazirani. *Introduction to Computational Learning Theory.* MIT Press, Cambridge, MA, 1994.
27. J. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, Cambridge, MA, 1992.
28. J. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs.* MIT Press, Cambridge, MA, 1994.
29. J. Laird, P. Rosenbloom, and A. Newell. Chunking in SOAR: The anatomy of a general learning mechanism. *Machine Learning*, 1(1):11–46, 1986.

30. 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.
31. P. Langley. Areas of application for machine learning. In *Proceedings of the Fifth International Symposium on Knowledge Engineering*, Sevilla, 1992.
32. L.-J. Lin. *Self-supervised Learning by Reinforcement and Artificial Neural Networks.* PhD thesis, Carnegie Mellon University, School of Computer Science, Pittsburgh, PA, 1992.
33. 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.
34. 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.
35. T. M. Mitchell. *Machine Learning.* McGraw-Hill, NY, in preparation.
36. 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.
37. 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.
38. A. W. Moore. *Efficient Memory-based Learning for Robot Control.* PhD thesis, Trinity Hall, University of Cambridge, England, 1990.
39. 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.
40. 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.
41. S. Muggelton. *Inductive Logic Programming.* Academic Press, New York, 1992.
42. J. O'Sullivan. Integrating initialization bias and search bias in artificial neural networks. Internal report, January 1996.
43. 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.
44. 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.
45. 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.
46. L.Y. Pratt and B. Jennings. A review of transfer between connectionist networks. *Connection Science*, 1996. Special issue on "transfer", to appear.
47. J. R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.
48. L. Rendell, R. Seshu, and D. Tcheng. Layered concept-learning and dynamically-variable bias management. In *Proceedings of IJCAI-87*, pages 308–314, 1987.
49. 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.
50. M. B. Ring. *Continual Learning in Reinforcement Environments.* R. Oldenbourg Verlag, München, Wien, 1995.
51. S.J. Russell. Prior knowledge and autonomous learning. *Robotics and Autonomous Systems*, 8:145–159, 1991.

52. J. H. Schmidhuber. On learning how to learn learning strategies. Technical Report FKI-198-94, Technische Universität München, January 1995. Revised version.
53. 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.
54. 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.
55. 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.
56. 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.
57. 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.
58. S. P. Singh. Transfer of learning by composing solutions for elemental sequential tasks. *Machine Learning*, 8, 1992.
59. C. Stanfill and D. Waltz. Towards memory-based reasoning. *Communications of the ACM*, 29(12):1213–1228, December 1986.
60. 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.
61. 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.
62. 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.
63. R. S. Sutton, editor. *Reinforcement Learning*. Kluwer Academic Publishers, Boston, MA, 1992.
64. 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.
65. 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.
66. S. Thrun. *Explanation-Based Neural Network Learning: A Lifelong Learning Approach*. Kluwer Academic Publishers, Boston, MA, 1996.
67. 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.
68. 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.
69. 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.
70. P. E. Utgoff. *Machine Learning of Inductive Bias*. Kluwer Academic Publishers, 1986.
71. 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.
72. L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, 1984.
73. V. Vapnik. *Estimations of dependences based on statistical data*. Springer Publisher, 1982.
74. 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.
75. 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.

76. 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.