
Toward a Formal Framework for Continual Learning

Presented at the NIPS 2005 Workshop on Inductive Transfer, Whistler, British Columbia, Canada

Mark B. Ring
University of Alberta
Edmonton, Alberta, Canada T6G 2E8

Abstract

This paper revisits the continual-learning paradigm I described at the previous NIPS workshop on inductive transfer in 1995. It presents a framework that formally merges ideas from reinforcement learning and inductive transfer, potentially broadening the scope of each. Most research in RL assumes a stationary (non-changing) world, while research in transfer primarily focuses on supervised learning. Combining the two approaches yields a learning method for an agent that constantly improves its ability to achieve reward in complex, non-stationary environments.

To design a learning algorithm is to make an assumption. The assumption is that there is structure in the learning task. If there is no structure, then there is no relationship between training data and testing data, there is nothing to be learned, and all learning fails. So we assume that there are regularities common to the training and testing data and we develop algorithms to find and exploit these regularities.

In general, we assume there is a (usually stochastic) function, $f : \mathcal{X} \rightarrow \mathcal{Y}$ that generated both the training and testing output data, which is to say that the task of the learning agent is to discover this function (or a function with sufficiently similar behavior).

1 Inductive Transfer

Inductive transfer works when the functions learned in different tasks can be decomposed into functions over subfunctions. For example, a task a may be to learn some function $f^a : \mathcal{X} \rightarrow \mathcal{Y}$, based on (x, y) pairs sampled from a joint distribution over $\mathcal{X} \times \mathcal{Y}$. And it may be that f^a can be decomposed, say, into an ordered set of k functions $F^a = \{f_1^a, \dots, f_k^a : \mathcal{X} \rightarrow \mathbb{R}\}$ and a combining function $f_C^a : \mathbb{R}^k \rightarrow \mathcal{Y}$ such that

$$f^a(x) = f_C^a(f_1^a(x), f_2^a(x), \dots, f_k^a(x)),$$

or, more succinctly:

$$f^a(x) = f_C^a(F^a(x)).$$

The functions $f_i^a \in F^a$ form a *minimal basis* for the mapping $f^a : \mathcal{X} \rightarrow \mathcal{Y}$, when (1) $f^a(x)$ is sufficiently well approximated, and (2) no f_i^a is unnecessary. Condition (1) can be achieved in the standard way through the introduction of an appropriate loss function. Condition (2) is achieved when there is no proper subset $G \subset F^a$ nor combining function $g_C^a : \mathbb{R}^{|G|} \rightarrow \mathcal{Y}$ such that for all $x \in \mathcal{X}$,

$$f^a(x) = g_C^a(G(x)).$$

Given two tasks, a and b , generated respectively by functions, f^a and $f^b : \mathcal{X} \rightarrow \mathcal{Y}$, inductive transfer is possible when there is a minimal basis for each with members in common; i.e., for some sets of basis functions F^a and F^b , there are combining functions $f_C^a : \mathfrak{R}^{|F^a|} \rightarrow \mathcal{Y}$ and $f_C^b : \mathfrak{R}^{|F^b|} \rightarrow \mathcal{Y}$ such that

$$\begin{aligned} f^a(x) &= f_C^a(F^a(x)) \\ f^b(x) &= f_C^b(F^b(x)) \\ F^a \cap F^b &\neq \emptyset \end{aligned}$$

Such functions that can share subfunctions are referred to here as *transfer compatible* functions. Conditions (1) and (2) are fairly weak if the combining functions f_C^a can be of arbitrary complexity, though the conditions are fairly strong if the combining functions are, for example, linear. Intuitively it seems likely the functional and structural similarity [4] of functions f^a and f^b (and hence the potential benefit of transfer from one task to the other) increases as $|F^a \cap F^b|/|F^a \cup F^b|$ increases and the VC dimension of the combining functions decreases.

Inductive transfer is therefore the search for common bases, as was done, for example, quite explicitly by Baxter [1], and implicitly in the case of multi-task learning (as for example by Caruana [2]).

2 Reinforcement Learning

In the standard reinforcement-learning framework (cf. Sutton and Barto, 1998), a learning agent interacts with a Markov Decision Process (MDP) over a series of time steps $t \in \{0, 1, 2, \dots\}$. At each time step the agent takes an action $a_t \in \mathcal{A}$ in its current state $s_t \in \mathcal{S}$ and receives a reward $r_t \in \mathfrak{R}$. The dynamics underlying the environment are described as an MDP with state-to-state transition probabilities $\mathcal{P}_{s's'}^a = Pr\{s_{t+1} = s' | s_t = s, a_t = a\}$ and expected rewards $\mathcal{R}_s^a = E\{r_{t+1} | s_t = s, a_t = a\}$. The agent's decision-making process is described by a policy, $\pi(s, a) = Pr\{a_t = a | s_t = s\}$ which the agent refines through repeated interaction with the environment so as to increase $E_{\pi, s_0}[r_0, r_1, \dots, r_\infty]$, the reward it can expect to receive if it follows policy π from state s_0 . Alternatively, the agent may sample observations $o \in \mathcal{O}$ related to the current state (possibly stochastically), where s_t may or may not be uniquely identified by the current observation o_t , perhaps when taken in combination with previous observations and actions $(o_0, a_0, o_1, a_1, \dots, o_{t-1}, a_{t-1})$.

3 Continual Learning

In traditional reinforcement learning, the world is modeled as a stationary MDP: fixed dynamics and states that can recur infinitely often. The agent's learning "task" is to improve performance by improving its policy, which generally entails developing an estimate of the expected cumulative reward attainable from individual states (the state-value function) or from state-action pairs (the action-value function). Augmenting RL with concepts from inductive transfer quickly hits a snag: small changes to the structure of the state space (especially small changes to the placement of rewards) can introduce major changes in the value function. One solution is to model the relationships between the states separately from the value function and then when the rewards change, recalculate the value function from the model using dynamic-programming methods. Predictable changes in the relationships between the states, however, are more difficult to capture. The alternative explored here is to step away from the MDP foundation of RL and instead describe the environment in terms of regular relationships between history and future.

3.1 The Continual-Learning Problem, Formally

A continual-learning agent's inputs are observations, $o_t \in \mathcal{O}$. The agent may learn from all its past experiences, collectively known as its *history*. Each *moment* of history is a triple,

$m = (o, r, a) \in \mathcal{M}$ where $\mathcal{M} = \mathcal{O} \times \mathfrak{R} \times \mathcal{A}$ and $a \in \mathcal{A}$ is an agent action.¹ In the discrete-time case, a history is a discrete series of moments, $h(t) = (m_0, \dots, m_t)$, while in the continuous-time case, the history is a continuous function of time $h : [0, t] \rightarrow \mathcal{M}$.

The future is slightly different from the past in that it is not yet known. Instead, there are probability distributions over possible futures contingent on the policy. Each possible future is an infinite-length trajectory of moments. In the continuous case it can be represented as a function mapping time to possible future moments $\xi : (t, \infty) \rightarrow \mathcal{M}$. The set of all possible futures is \mathcal{Z} , and for each policy and history there is a probability distribution \mathcal{D} over \mathcal{Z} ; i.e., $\mathcal{D}(\xi|h \in \mathcal{H}, \pi \in \Pi)$, where \mathcal{H} is the space of all possible histories, and Π is the space of all possible policies.

The agent’s aim is to maximize its expected return by estimating the reward part of this distribution and finding the policy with greatest expected reward. If $\mathcal{R}(m)$ is the reward part of a moment, then the expected return for a particular future is

$$\mathcal{R}(\xi, t) = \int_{\tau=t}^{\infty} \gamma^{\tau-t} \mathcal{R}[\xi(\tau)] d\tau \quad (1)$$

where $0 < \gamma \leq 1$ is a discounting factor that, if less than 1, keeps the integral from diverging. The overall expected return for a given policy and history is:

$$\mathbf{R}(\pi, h, t) = \int_{\xi \in \mathcal{Z}} \mathcal{D}(\xi|h, \pi) \mathcal{R}(\xi, t) d\xi. \quad (2)$$

The agent’s aim is to choose a policy that maximizes its expected reward:

$$\pi_t^* = \operatorname{argmax}_{\pi} [\mathbf{R}(\pi, h, t)] \quad (3)$$

3.2 Examples

Space does not allow an extended illustration of non-stationary domains where the environment and/or reward structure changes in predictable ways. One example is a “shaping” situation where an agent can be taught to negotiate a certain simple environment or perform a simple maneuver that is changed bit by bit to increasingly more complicated situations. Non-stationary environments and changing value functions can be problematic for traditional RL approaches, even when the changes seem reasonably small. The continual-learning framework, however, is particularly appropriate for situations where the learning agent can find regularities in the environment, use them and build upon them to find new regularities, thus attaining reward increasingly well; i.e., where skills can be developed in a seemingly hierarchical fashion.

3.3 Solution Method

Since \mathcal{D} is not known in advance, it must be estimated from available evidence, namely, from the history seen so far. To help estimate \mathcal{D} , a functional model is built and maintained that explains the historical data. The agent runs the following processes continually:

1. Fit or refit a model to the historical data,
2. Use the model to estimate/predict future reward from a given policy and history,
3. Modify the policy to increase predicted reward.

¹Alternatively, the control parameters, $\theta \in \Theta$, which fully describe the agent’s interaction with the world (its policy), are recorded instead of actions, $m = (o, r, \theta) \in \mathcal{M}$, where $\mathcal{M} = \mathcal{O} \times \mathfrak{R} \times \Theta$. This alternative has the advantages that (a) θ may change more smoothly over time than the actions, and (b) it places no constraints on the way that actions unfold over time. In this case, the agent’s only explicit knowledge of its actions are through its control parameters θ and its observations, which may also — as a choice of implementation — extensively describe the actions taken.

The model is constantly being rebuilt, so short-term objectives may change at any moment; thus the notion of “learning task” is greatly challenged, but in a structured world the new model and the previous one should be transfer compatible. It may therefore be helpful to think of the continual-learning paradigm as a process of continual transfer, where the “transfer-to” task is potentially new at every time step and the source or “transfer-from” task consists of everything seen so far: i.e.,

$$\begin{aligned} f^t(x) &= f_C^t(F^t(x)) \\ f^{t+1}(x) &= f_C^{t+1}(F^{t+1}(x)) \\ F^t \cap F^{t+1} &\neq \emptyset, \end{aligned}$$

where f^t could be one of several different mappings. In principle, f^t is a model of the environment, but eventually we want to use f^t for choosing agent policies, or at the very least, agent actions; therefore f^t might (1) be a probabilistic function that mimics \mathcal{D} , i.e., $f^t : \Pi \times \mathcal{H} \times t \rightarrow \mathcal{O} \times \mathfrak{R}$ which can be used to draw estimated sample trajectories from the future for evaluation. (2) directly estimate the expected reward of a policy, $f^t : \mathcal{H} \times \Pi \rightarrow \mathfrak{R}$ (Equation 2), or of just the next action, $f^t : \mathcal{H} \times \mathcal{A} \rightarrow \mathfrak{R}$, (3) simply choose the next policy $f^t : \mathcal{H} \rightarrow \Pi$ (Equation 3) or next action $f^t : \mathcal{H} \rightarrow \mathcal{A}$. (Much more can be said about these possibilities than space allows.)

In simple cases $F^t \subseteq F^{t+1}$ and all regularities discovered to explain events so far are maintained for predicting future events. Since we assume a structured world, it makes sense to bias our learning algorithms in favor of uncovering this structure (i.e., towards finding functions with minimal VC dimension) and then increasing capacity over time to accommodate new data. One way to achieve this is by continually augmenting F with new basis functions. Yet it should be noted that each f^t is an approximation of f^{t+1} and we are continually seeking the maximally structured function f^∞ that fits all the data seen and yet to be seen.

Inductive transfer and reinforcement learning are natural extensions of each other, particularly when the reinforcement-learning problem is reformulated in terms of continual function fitting. It should be noted that the paradigm presented is a superset of classical reinforcement learning in the sense that the target function of the environment (f^∞) can be many things, including an MDP.

Acknowledgments

Thanks to Rich Sutton, Yaakov Engel, Eddie Rafols, Cosmin Paduraru, and David Silver for helpful discussions.

References

- [1] Jonathan Baxter. *Learning Internal Representations*. PhD thesis, The Flanders University of South Australia, 1994.
- [2] Rich Caruana. Multitask learning: A knowledge-based source of inductive bias. In *Machine Learning: Proceedings of the tenth International Conference*, pages 41–48. Morgan Kaufmann Publishers, June 1993.
- [3] Mark B. Ring. *Continual Learning in Reinforcement Environments*. PhD thesis, University of Texas at Austin, Austin, Texas 78712, August 1994.
- [4] Daniel L. Silver. *Selective Transfer of Neural Network Task Knowledge*. PhD thesis, University of Western Ontario, London, Ontario, 2000.
- [5] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.