Chapter 7
Transfer between Different Reinforcement Learning Methods

The previous two chapters introduced three different methods that use inter-task mappings to transfer between tasks with different state variables and actions, but all methods required agents in the source task and target task to use the same type of underlying RL method.

1. Value Function Transfer (Section 5.2) used an action-value function learned in the source task to initialize an action-value function in the target task, with the requirement that both source and target task agents use value-function learning, such as Q-Learning or Sarsa.

2. Q-Value Reuse (Section 6.1) also required TD learners in the source and target task, but copied an entire Q-value function, rather than using it to initialize a target task’s action-value function. Thus the target task agent must use value function learning.

3. Policy transfer (Section 6.2) transfers between policy search methods which use neural network action selectors.

In this chapter, we introduce three additional transfer methods which do not require the source and target tasks to be learned by the same type of RL algorithm. Previous work (c.f., [Taylor et al. (2006)]) has shown that characteristics of a particular task may favor one type of RL algorithm over another. If one can determine what type of RL algorithm would be best for a given target task, a TL method would, ideally, be flexible enough to reuse knowledge from a source task, even were that source task learned by a different type of algorithm.

Section 7.1 introduces TIMBREL [Taylor et al. (2008c)], a method that transfers observed instances between tasks. (Recall that we use instance to refer to an experienced \(\langle s, a, r, s' \rangle\) tuple.) This method utilizes inter-task mappings to directly transfer experienced instances between tasks. The instances are used to construct an initial model of \(T\) and \(R\) (the transition and reward functions) in the target task, which can significantly reduce the amount of experience needed to learn in the target task. Although our experiments use an instance-based RL algorithm in both the source task and target task (Fitted R-MAX, as discussed in Section 2.3.3), any RL algorithm (such as Sarsa or NEAT) could gather instances in the source task to enable beneficial transfer.

M.E. Taylor: Transfer in Reinforcement Learning Domains, SCI 216, pp. 139-179, springerlink.com © Springer-Verlag Berlin Heidelberg 2009
Next, Rule Transfer \cite{Taylor2007a} is introduced in Section 7.2. The key difference between this method and others in this monograph is that a higher-level abstraction is transferred between tasks, rather than recorded instances, full policies, or action-value functions. After the source task is learned using some RL algorithm, the agent records instances in the source task. It then uses a rule-learning algorithm to extract rules which describe the policy (i.e., “If the state is \( s \), then take action \( a \)”). These rules are used in the target task to increase the speed of learning with Sarsa, relative to not using transfer. As in TIMBREL, instances are used, but now higher-level rules are the information transferred. Additionally, the transferred rules are used directly as an initial control policy, whereas when instances are directly transferred, they are used to help improve learning speed instead of to direct the actions of the target task agent.

Section 7.3 discusses two types of representation transfer \cite{Taylor2007b}. The goal of representation transfer is broader than other transfer algorithms in this monograph. In addition to transfer between tasks, the goals of representation transfer are also to enable transfer between different:

- Function approximator parameterizations (e.g., adding or removing state variables from a CMAC)
- Function approximators (e.g., change from an ANN to a RBF)
- Learning methods (e.g., change from Sarsa to policy search)

In all of these cases, the goal of representation transfer is to reuse knowledge between representations that the target representation can be learned faster, relative to not using transfer.

Lastly, Section 7.4 summarizes the six TL methods in this monograph, all of which can utilize the same inter-task mappings. In addition to providing guidelines about when each method would be most appropriate, we provide a chart summarizing the experiments, RL algorithms, and function approximators used.

### 7.1 TIMBREL: Instance-Based Transfer

Model-free algorithms such as Q-Learning and Sarsa learn to predict the utility of each action in different situations but they do not learn the effects of actions. In contrast, model-based (or model-learning) methods, such as Dyna-Q \cite{Sutton1998}, PEGASUS \cite{Ng2000}, R-MAX \cite{Brafman2002a}, and Fitted R-MAX \cite{Jong2007}, use their experience to learn an internal model of how the actions affect the agent and its environment, an approach empirically shown to often be more sample efficient. Such a model can be used in conjunction with dynamic programming \cite{Bellman1957} to perform off-line planning, often enabling performance superior to model-free methods because better performance can be achieved with fewer environmental samples. Building these models may be computationally intensive, but using CPU cycles to reduce data collection time is a highly favorable tradeoff in many domains, such as in physically embodied agents. In order to further reduce sample complexity, this